Universitetet
i Stavanger

Prosjekt i ING100
Ingeniørfaglig innføringsemne

Institutt for data- og elektroteknikk
Universitet i Stavanger

Innhold

Kapittel 1

Introduksjon

1.1 Praktisk informasjon

Les HELE dette første kapittelet nøye før dere begynner på prosjektet. Det inneholder mye informasjon, og som regel finner dere svar her på mye av det dere lurer på. Kapittel 2 omhandler installering av programvare og kapittel 3 gir en introduksjon til noen av de matematiske metodene dere skal bruke i prosjektet.

- I prosjektdelen av emnet ING100 skal du lære mer om programmering ved bruk av Matlab og LEGO NXT. Som en motivasjon er en del av prosjektet lagt opp som en uformell konkurranse hvor dere skal benytte en joystick til å kjøre en Lego-robot langs en bane, hvor målet er å gjøre dette med minst mulig avvik fra baneprofilen og på kortest mulig tid. Det er viktig å ikke bruke for mye tid på å optimalisere kjøringen. Husk, dette er fag hvor dere skal lære programmering, ikke vise hvor gode dere er til å kjøre Lego-robot.
- I tillegg til å bruke å Matlab sammen med LEGO NXT er det også mulig å lage programmer som kjøres i Matlab alene, som f.eks. forskjellige spill som 3 på rad ved bruk av tastatur, skytespill med bruk av joystick, og reaksjonstestspill. Det er bare fantasien som setter grenser.
- All informasjon vil bli gitt på itsLearning eller sendt til deg på e-postadressen: fornavn.etternavn@stud.uis.no.

  Pass på å lese denne e-posten daglig, eventuelt legg inn en viderekopling til den e-postadressen du bruker til vanlig.

- Ved spørsmål som du ikke finner svar på i dette dokumentet, send en e-post til: tormod.drengstig@uis.no
- Arbeidet skal gjennomføres **i grupper på 3 eller 4 studenter per gruppe** (avhengig av totalt antall studenter som tar faget). Navnet på gruppene er **GruppeYYXX**, hvor **YY** er de to siste tallene i årstallet og **XX** starter på 01 og øker fortløpende for hver gruppe.

1.1.1 Viktige datoer og tidspunkt

- **Tirsdag 16. september kl. 12.30–14.00 i rom E-101** blir det holdt et informasjons-møte om prosjektet.

  Siden veldig mye viktig informasjon blir gitt i dette møtet, vil jeg, basert på navneopprop, vite hvor mange som skal ta prosjektet. Dersom du er forhindret i å møte opp, send meg en epost om dette.

- I perioden **onsdag 17. september** til **onsdag 24. september** skal gruppene gjøre følgende:
  - Ta ut et LEGO-NXT-sett med batterieliminator og en joystick hos lab-ingeniør Romuald Bernacki på E-455c. Siden hver gruppe er økonomisk ansvarlig for dette utstyret, skal dere selv gå gjennom og kontrollere at alle delene er i boksen, og bekrefte at alt utstyret er mottatt ved å signere på en deleliste som Romuald har. Deretter skal dere låse utstyret inn i skapene som står på E-453. Hengelås (kodelås) henger allerede på skapene. Husk å endre koden.
  - Installere programvare på UiS sine PC'er og teste at alt virker som det skal. Dette skal godkjennes av lab-ingeniør innen **24. september**. Installasjonsveiledningen finner du i kapittel 2.2.
- Formell oppstart av prosjektet er **mandag 6. oktober** i uke 41.
- Innleveringsfrist er **fredag 21. november kl. 23.59** i uke 47, noe som totalt gir 7 ukers prosjektperiode (se detaljert informasjon om selve innleveringen i kap. 1.5).

1.1.2 Eksamen

- Muntlig presentasjon av prosjektet vil skje **mandag 24. november** (prosjektveileder og ekstern sensor vil være tilstede). For at alle skal bli eksaminert i løpet av en dag, har hver gruppe 10 min til rådighet som skal fordeles slik:

  - 5 min presentasjon og visning av filmer av prosjektene
  - 3 min spørsmål og svar
  - 2 min til organisering mellom gruppene

  Hele gruppen må være tilstede på denne presentasjonen (kun legeerklæring gir gyldig fravær), og det enkelte gruppemedlem må kunne svare på spørsmål om programmene som de selv har bidratt i. Ugyldig fravær gir karakter F for vedkommende (ikke for hele gruppen). I utgangspunktet får hele gruppen samme karakter, men dersom samarbeidet i gruppen har vist seg å fungere dårlig, kan gruppemedlemmer kreve individuell bedømming ved sensur.

- På den muntlige presentasjonen/eksamen tar dere med en bærbar PC/Mac hvor dere har *en* sammenhengende redigert film av alle delprosjektene klar til avspilling. Filmen skal ikke være mer enn 3 minutt. Ved lange sekvenser kan dere redigere filmen slik at disse sekvensene vises i dobbel hastighet. Denne filmen skal også leveres inn på itsLearning, men da må filmen være mindre enn 30Mb.

- Dersom prosjektet i sin helhet (eller individuelle kandidater) bedømmes til karakteren F, har dere/du 14 dager frem til **8. desember** på å jobbe med prosjektet for om mulig å bestå. Dersom dette ikke lykkes, må dere vente til neste høst for å ta prosjektet på ny.

- Eksamen i Matlab-delen av faget er **18. desember.**

- For å få bestått emnet ING100, må både skriftlig eksamen og prosjektet være bestått. En vektet karakter av prosjekt og eksamen beregnes (se kap. 1.6 for mer detaljer). De som ikke består den skriftlige eksamen, har en ny mulighet til å ta den i februar neste år.

1.1.3 Veiledning med prosjektveileder

- I løpet av prosjektet skal dere ha 3 veiledningsmøter á 20 minutt med prosjektveileder som foregår på kontor **E-453c** som er kontoret i gangen mellom E-453 og E-455. Tidene står i tabellen nedenfor.

Oddetallsuker uke 41, 43, 45. Gruppe:	Partallsuker uke 42, 44, 46. Gruppe:	Møtet begynner fredager fra uke 41	Møtet begynner mandager fra uke 42
XX01	XX22	08:40	
XX02	XX23	09:00	
XX03	XX24	09:20	
XX04	XX25	09:40	
XX05	XX26	10:00	
XX06	XX27	10:20	
XX07	XX28	10:40	
XX08	XX29	11:00	
XX09	XX30	11:20	
XX10	XX31	11:40	
—PAUSE—	—PAUSE—		
XX11	XX32	12:30	
XX12	XX33	13:10	
XX13	XX34	13:30	
XX14	XX35	13:50	
XX36	XX15		12:20
XX37	XX16		12:40
XX38	XX17		13:00
XX39	XX18		13:20
XX40	XX19		13:40
XX41	XX20		14:00
XX42	XX21		14:20
XX43			14:40

Disse møtene er obligatoriske for alle gruppemedlemmene. Det er likevel tillatt med ett fravær uten legeattest. Det skal ellers tungtveiende grunner til å forandre på møtetidspunkt, som f.eks. offshore-arbeid.

- Før veiledningsmøtene er minimumskravet for hva som skal være ferdig beskrevet i tabellen nedenfor. **Mange av dere vil være ferdig med alle tre punktene på første møte, så fremdriften under kvalifiserer for karakteren E.**

	Hva skal være ferdig? **(Dette er absolutte minimumskrav)**
1. møte	LEGO-NXT robot ferdig bygget og roboten kan kjøres manuelt langs banen.
2. møte	Skisser til løsning av integrering, derivering og filtrering.
3. møte	Implementasjon av integrering, derivering og filtrering som virker. Presentasjon av et noenlunde ferdig program av roboten som kjøres manuelt langs baneprofilen.

- Til hvert veiledningsmøte tar dere med utskrift av all ferdig/uferdig kode dere har laget, samt en liste over de prosjektene dere er ferdige med og hvilke dere har tenkt å begynne på. Indiker for hvert program hvem av gruppemedlemmene det er som har jobbet med det. Jeg kommer til å gå gjennom koden for å gi tips og forslag til forbedringer. Kodeutskriftene samler jeg sammen i mappen til hver gruppe.

1.1.4 Veiledning med studentassistenter

Veiledning med studentassistenter skjer til følgende tider på rom **E-453** og **E-455** for de som jobber på UiS sine PC'er, og **E-458** for de som jobber på egne PC'er:

- **Mandager kl. 1215-1600**
  - Gruppe 22-42 har timene mellom 1215-1400.
  - Gruppe 01-21 har timene mellom 1415-1600 (i denne tiden er ikke **E-458** tilgjengelig, men **D-351** er tilgjengelig).
- **Fredager kl. 0815-1600**
  - Gruppe 01-21 har timene mellom 0815-1200.
  - Gruppe 22-42 har timene mellom 1215-1600.

Så lenge det er ledig plass på **E-458** er det mulig for alle gruppene å jobbe der.

1.1.5 Annen praktisk informasjon

- Dere kan få nøkkelkort (koster 20 kroner) av lab-ingeniør Romuald Bernacki (E-455c) for å få tilgang på E-453 etter kl. 1600 og i helgene.
- PC'ene på E-453 kan brukes alle 7 ukedager mellom kl. 0700 og kl. 2200 (bruk eget nøkkelkort).
- PC'ene på E-455 kan brukes alle arbeidsdager mellom kl. 0800 og 1600, bortsett fra når andre fag har lab der (disse tidspunktene som står markert i timeplanen som henger på døren inn til E-455).
- Dere har mulighet til å ta med Lego-settene hjem, og erfaringsmessig er dette veldig lurt å gjøre siden du da får anledning til å sitte i ro og mak og jobbe med koden. Rotér innad i gruppen hvem som tar roboten med hjem.
- Når prosjektet er ferdig, skal dere demontere alt og sortere brikkene i boksen, samt fylle ut antall av hver komponent i delelisten dere signerte på ved uttak. Dere må ikke betale noe dersom et begrenset antall smådeler mangler, men det er uansett viktig at vi får beskjed om dette slik at vi kan komplementere settene til neste år. Dersom veldig mange Legodeler mangler, eller store enheter som NXT-enheten, motorer og sensorer, er dere erstatningspliktige.

Karakter blir ikke gitt før alt Lego-utstyr og joystick er tilbakelevert.

- Det skal være unødvendig å minne om dette, men plagiering og forsøk på juks kan få store konsekvenser for din videre utdanning (se mer detaljer i kap. 1.7)
- I prosjektet kan dere benytte dere av all litteratur i bokform og på nett. Husk å notere ned hvor informasjonen er hentet, slik at det blir med i referanselista i prosjekt-rapporten som skal leveres inn (se eget skriv "Mal for prosjektrapport.docx"). Manglende referering er det samme som juks.
- Ikke ta ut nettverkskabelen fra UiS sine stasjonære PC'er. Punktet låses automatisk ved tilkopling av en annen PC.
- Ikke kople skjermene på lab'en til egen bærbar PC.
- Et alternativ til Word som tekstbehandlingsprogram er LaTeX (uttales latek). Labingeniør Per Jotun har laget en blogg hvor du finner informasjon om hvordan installere og komme i gang, samt at du finner eksempelprogram som er klar til kompilering (Per kan også hjelpe til med å få det til å fungere):

  `http://ulikt.blogspot.no/search/label/LaTeX`

1.2 Detaljert prosjektbeskrivelse, obligatorisk del

Den overordnede tittelen på prosjektoppgaven er "Matlab og LEGO NXT i anvendt matematikk og fysikk", og mer spesifikt er prosjektet delt opp i følgende deler (for å få bestått med karakter E må punktene 1, 2, 3 og 4 vist under være gjort):

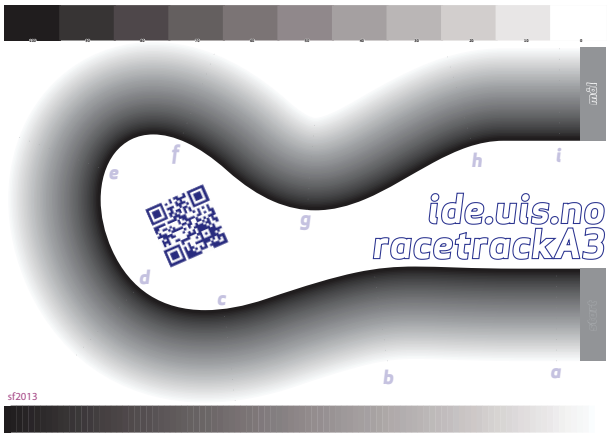
1. Bygg en LEGO-robot fra byggebeskrivelsen som følger med LEGO-utstyret, se under

   a. Bygg først frem til og med side 22 og benytt høyre motor på utgang B og venstre motor på utgang C som vist.
   b. **Ikke** programmer NXT'en direkte som vist på side 23, 27, 31, osv. fordi dette opptar plass i minnet.
   c. Instrumentér opp NXT'en og behold alle sensorene på (ikke fjern de slik det er vist i veiledningen):
      i. s. 24-26 (lydsensor, inngang 2)
      ii. s. 28-30 (ultralydsensor, inngang 4)
      iii. s. 32-34 (lyssensor, inngang 3).
   d. Monter deretter akselerometeret under selve NXT'en på en egnet måte (står ikke i byggebeskrivelsen). Kople denne til inngang 1.

e. Lag en konstruksjon av Legodeler som holder USB-kabelen opp i luften.

f. Roboten er nå ferdig.

2. Før dere går videre, må dere ha lest gjennom og utført det som står i kapittel 2.2 Ta utgangspunkt i koden i filen `EksempelMotorA.m` og lag et nytt program hvor joysticken benyttes til å kjøre LEGO-roboten **manuelt** både fremover og bakover, samt svinge til høyre og venstre (motor A skal nå være frakoplet). Målet er å kjøre roboten langs banen vist under, og denne baneprofilen ligger på pulten ved hver PC på E-453 og E-455.



3. Lyssensoren (inngang 3) måler refleksjon av lys fra baneprofilen (hvitt=mye refleksjon, svart=lite refleksjon). Dette skal dere bruke til følgende:

   a. Når dere kjører roboten manuelt med joysticken, er målet å kjøre midt på baneprofilen (ca. 50% grått). Målet å følge denne gråtoneverdien best mulig (med minst mulig avvik), og til dette trenger vi et kvantitativt (tallbasert) mål for sammenligning. Konkret betyr dette at mens roboten kjører skal absoluttverdien av avviket (avvik i begge retninger gir positive verdier) summeres, det vil si numerisk integreres som funksjon av tiden, se utfyllende forklaring i kap. 3.1. Numerisk betyr å gjøre en tilnærming til den analytiske (den eksakte) løsningen gjennom en program-algoritme. Denne verdien skal kontinuerlig vises og oppdateres på skjermen mens du kjører (enten som et tall eller en kurve).

   b. Videre skal dere lage en rutine som finner ut om roboten kjører ut mot lys side eller mot mørk side. Til dette skal dere bruke numerisk derivasjon av lyssignalet, se utfyllende forklaring i kap. 3.2. Denne informasjonen skal

presenteres på skjermen på en eller annen måte (enten tekstbasert, piler, farger eller andre måter).

c. Siden lyssensoren gir støyfylte målinger, skal dere lage rutiner for å filtrere (glatte) lyssignalet, se utfyllende forklaring i kap. 3.3. Her må dere eksperimentere litt for å finne ut hvilken type glatting/filtrerer som er best. Dersom styresignalet fra joysticken gir veldig rykkete kjøring kan dere også velge å glatte/filtrere dette.

4. Dokumentér slik som beskrevet i kap. 3.1, 3.2 og 3.3 at rutinene i pkt. 3 fungerer, samt ved å kjøre LEGO-roboten langs banen og lag film av dette (bruk mobilen). Den gruppen som gjør dette best (minst mulig avvik på kortest mulig tid) vinner masse heder og ære. Sammenligningskriteriet er:

```
verdi = tid(i sekund)*100 + integrert avvik
```

Lyssensoren skal under kjøring ikke komme ut på den hvite delen av arket.

Som nevnt må deloppgaven frem til hit være gjort og tilstrekkelig dokumentert for å få karakteren E (det er alikevel mulig å gjøre mer ut av disse 4 punktene og dermed forbedre karakteren ved f.eks. å teste ut ulike integrasjonsrutiner eller teste ut mange forskjellige filtervarianter). På de neste sidene er det gitt mange forslag til prosjekt med anvendt matematikk/fysikk som dere programmere i den kreative delen og som vil ytterligere forbedre karakteren.

1.3 Forslag til prosjekter i kreativ del

De kreative prosjektene er delt i 3 kategorier, avhengig av vanskelighetsgrad. For de av dere som ikke har programmert noe før, kan det være lurt å starte med de enkleste prosjektene og heller avansere etterhvert. De enkle prosjektene er ofte slik at dere kan ta tak i eksisterende grunnkode for f.eks. legoroboten og deretter legge inn 2-10 ekstra kodellinjer så har dere en "nytt" prosjekt.

Husk at disse prosjektene også skal filmes og inkluderes i sluttfilmen.

1.3.1 Enkle småprosjekter

1. Programmér roboten til å kjøre automatisk langs baneprofilen (forhåpentligvis raskere og med mindre avvik enn ved manuell kjøring). En avansert variant er gitt i pkt. 11 hvor du må lese deg opp på reguleringsteknikk i kapittel 3.4.

2. Bruk lysmåleren til å estimer hastigheten på roboten ved å la roboten kjør over et ark med svarte/hvite streker (som et fortau). Sammenlign eventuelt med virkelig hastighet beregnet fra rotasjonen av hjulene.

3. Bruk lysmåleren til å lese gråtoner på et eller flere A4-ark, hvor hver gråtone representerer en tone, og den fysiske lengden av gråtonene på arket representerer tidslengde som hver tone skal spilles. La roboten kjøre over arket og spille melodien.

4. Test ut hva rekkefølgen på derivering og integrering av et målesignal har å si.

5. La roboten følge etter deg (håndflaten din) i en fast avstand, uavhengig av hastighet på håndbevegelsen. Denne kan utvides, se pkt. 9.

6. Lag et program som bruker tastaturet til å teste reaksjonsevnen din (presenter resultatene med gjennomsnitt, varians, max/min). Dette kan utvides i mange former.

7. Spill musikk med joystick.

8. Få roboten til å reagere på f.eks. klapping eller plystring. La den utføre forskjellige ting dersom du klapper en gang, to ganger, tre ganger osv.

1.3.2 Mellomstore prosjekter

9. Utvid oppgaven med å la roboten følge etter deg (håndflaten din) med å montere ultralydsensoren på den siste ledige motoren slik at kan den kan søke som en radar helt til den finner et objekt som den skal følge. Dersom objektet ikke er rett foran roboten, må roboten rotere seg mot objektet før den kan følge etter.

10. Lag funksjoner av integrering, derivering og filtrering slik at dere kan kalle disse funksjonene i andre prosjekter.

11. Implementer en PID regulator for automatisk kjøring langs banen. Pass på nivået av integratoren i regulatoren mens du

    kjører. Les deg opp på begrepet integratorbegrensing.

12. Mens du kjører roboten på banen, tegn et estimat av baneprofilen i en figur i Matlab basert på motorpådrag og lysmåling.

13. Bruk både akselerometer, ultralydmåler og motorposisjon til å estimere hastigheten på roboten ved å kjøre mot en vegg (sammenlign disse estimatene i samme figur).

14. Lag et piano som benytter tastatur som tangenter.

15. I 2013 var den noen studentgrupper som prøvde å bygge en Segway. Dette fungerte ikke så bra siden hastigheten på kommunikasjonen over USB-ledningen er for treg til at den klarte å balansere. Men dersom du hjelper roboten litt med "å holde den" i en ledning, er det mulig å få det til.

16. Bruk akselerometeret til å animere en kunstig horisont som i et flysimulatorspill.

17. Beregn pappeskeareal basert på en ultralydsensor som roterer rundt vha. en motor inne i en pappeske. Tegn opp pakkeskens utforming (kan være for eksempel 4 kantet, 6 kantet, 7 kantet, rund, osv.). Sammenlign med det arealet du selv har beregnet. Denne kan dere utvide til å måle volum av esken. Dersom dere har et skjevt lokk på toppen, må dere beregne volumet av hvert kakestykke.

18. Lag et skytespill hvor et Matlab-program presenterer svarte sirkler i en Matlab-figur i vilkårlige posisjoner. Bruk joysticken til å bevege kursoren over sirkelen og skyt. Ta tiden fra sirkelen vises til du treffer den med joysticken. Utvid med skudd og og telling av skudd og mulighet for påfyll av mer skudd.

19. Bruk joysticken til å prøve å følge en sinusfunksjon generert i Matlab (kun en begrenset del av kurven vises på skjermen). Hva skjer med etterslepet (kalles fase) når frekvensen øker? Plott etterslep som funksjon av frekvens. Dette kalles frekvensrespons.

20. Bygg et sykkelhjul (se egen byggebeskrivelse på its learning) med en lyssensor som fungerer som en magnetsensor for å registrere at hjulet har rotert en runde. Anta at NXT'en er et sykkelcomputer som viser hastighet, gjennomsnittshastighet og tilbakelagt strekning. Presenter resultatene både som nåverdier og grafer (her må dere benytte en filter-funksjon for å få en glattet presentasjon). Sammenlign med verdier som kan hentes fra motoren (se kapittel 2.6). For de som er kjent med GPS, kan en ekstra utfordring være å legge inn en lengde som skal tilbakelegges, for deretter å beregne ETA (estimated time of arrival). Du må da anta en fartsgrense på ruten du skal følge.

21. Plotting av to variable mot hverandre. Benytt for eksempel to ultralyd-sensorer som du holder i hver sin hånd (med lang ledning). Beveg ultralydsensorene individuelt mot en flate samtidig som du plotter dem mot hverandre i Matlab. På den måten kan du lage for eksempel en sirkel, en ellipse, en lineær kurve, osv. Dette krever en del koordinering.

22. Finn ut hvordan `handles` i Matlab fungerer, og benytt disse ved plotting til figurer.

23. Lag et Angry Bird lignende spill vha. joystick eller tastatur. Et godt utgangspunkt er eksempel 5.2 side 180 i Matlabboka deres. Husk å referere til boka. En mulig utvidelse kan være å inkludere eksempel 12.2.

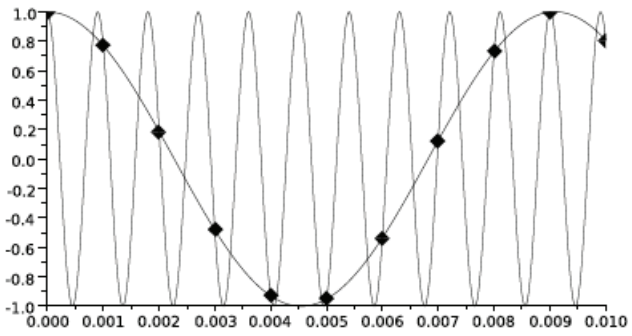
24. Bruk akseleromteret til å kjøre roboten.

25. Plot kurver i 3D ved å bruke joystick.

1.3.3 Avanserte prosjekter

26. Bygg en scanner som scanner objekter eller figurer og gjengir disse i Matlab.

27. Tving while-løkken i Matlab til å gå med en fast *lav* frekvens og benytt en eller annen sensor (roter f.eks. lyssensoren raskt rundt i en sirkel på robotbanen slik at du genererer en fin sinuskurve som inngangssignal) til å vise effekten av aliasing (http://en.wikipedia.org/wiki/Aliasing) og hvordan dette henger sammen med Nyquistfrekvensen og Shannons samplingsteorem, se figur 1.1.



Figur 1.1: Aliasing av to sinuskurver. Ved å sample en rask sinus med lav samplingsfrekvens, vil det komme frem en annen kurve med lavere frekvens enn det hurtigsvingende originalsignalet.

Dette temaet får elektrostudentene i senere fag.

28. En NXT sender morsesignal for en tekst, mens en annen NXT mottar morsesignalet og tolker det.

29. Bruk akselerometeret til å lage en hivkompensator.

30. Skriv tekst med roboten (utstyr den med en tusj og skriv bokstaver).

31. Lag en GUI (graphical user interface) hvor du lager en knapp for hvert program dere har.

32. Det finnes mange former for matematikkoppgaver som dere kan lage, f.eks. en kalkulator. Denne kan være basert på museklikk slik som på `grubble.net`. En annen kan være å lage et polynom av valgfri orden som skal tilpasses forskjellige punkter trykket inn med joystick eller mus, se kap 13.3 i Matlabboka. Her må du bruke polyfit- og polyvalfunksjonene.

33. Lag forskjellige spill. Space invaders, snake, manøvrere cursor gjennom trange kanaler vha. av joystick.

34. Bygg en katapult som dere benytter til å finne sammenhengen mellom hastighet/utgangsvinkel (avlest med endring av vinkelposisjon og bruk av tacholimit) og kastelengden målt med metermål. Sett resultatet opp som en tabell. Stemmer denne med teorien fra fysikken?

35. Bygg et system for estimering av tyngdekraften g.

1.4 Gjennomføring av prosjekt

Det blir her gitt noen konkrete tips/forslag til hvordan dere kan gjennomføre prosjektet.

- Gruppene består som regel av studenter med forskjellig bakgrunn, noen har programmert siden ungdomsskolen mens andre har aldri skrevet en for-løkke. For at alle gruppemedlemmene skal ha læringsutbytte av faget er det viktig å rotere på hvem som sitter med tastaturet. En konkret løsning er å programmere 30 minutt hver, og deretter gi tastaturet til nestemann. Det er alle gruppemedlemmenes ansvar å følge opp dette.

- ALLE gruppemedlemmmene skal lære seg numerisk integrasjon, numerisk derivasjon og filtrering, men det er det ofte upraktisk å jobbe 3 eller 4 personer mot bare én robot.

  Ved å ta vare på sensorsignalene i en datavektor sammen med en tidsvektor som tar vare på den absolutte klokketiden ved hver gjennomkjøring i while-løkken, er dere ikke avhengig av å ha tilgang til Lego-NXT'en i program-utviklingen. Lagre disse vektorene i en egen .mat-fil (sjekk syntax med save-kommandoen) som deretter kan lastes inn med load-kommandoen uten tilgang til NXT.

På denne måten kan ALLE gruppemedlemmene jobbe med å implementere integrering, derivering og filtrering med utgangspunkt i et lagret sensorsignal, uavhengig av sanntidsbruk av NXT'en.

Denne teknikken kan dere bruke på mange av prosjektene som er listet opp i forrige kapittel.

- Etter at dere har jobbet med den innledende delen av prosjektet hvor roboten kjøres manuelt og dere utfører integrering, derivering og filtrering, kan dere dele gruppen i 2 delgrupper (2+1 eller 2+2). Den ene delgruppen kan jobbe med prosjekter som involverer joystick og Lego-NXT mens den andre delgruppen kan jobbe med prosjekter som benytter f.eks. tastatur som input (bruk f.eks. piltaster) eller X-box konsoll for de som har det liggende hjemme. Etter en uke kan dere bytte på hvem som jobber med Lego-NXT'en.

- I begynnelsen er det ofte bedre å programmere flere småprosjekter enn å satse alt på ett stort prosjekt.

- Etter hvert gjennomførte prosjekt, bør dere dokumentere resultatet i prosjektrapporten (se "Mal for prosjektrapport.doc") med beskrivende tekst og kodeutdrag som forklares i detalj i rapporten. Suppler med matlabfigurer som viser at koden fungerer slik dere ha beskrevet i teksten. Det er også viktig å filme (med mobilen) alle prosjektene. Husk å få med eventuelle figurer på PC-skjermen i filmen.

- I prosjektrapporten skal dere i overskriften til hvert delprosjekt indikere hvilke gruppe-medlemmer som har jobbet med de enkelte delprosjekt.

- Dersom et gruppemedlem er fraværende en kortere periode, kan vedkommende få i oppgave å dokumentere/lage kommentarer til koden som de andre gruppemedlemmene har laget. På denne måten vil hele gruppen forstå koden.

1.5 Innlevering av prosjekt

For å kunne levere prosjektet, er det opprettet en *Oppgave* i itsLearning som heter **Prosjektrapport**. Her leverer dere følgende (husk på å inkludere alle gruppens medlemmer ved opplasting av filer):

1. Prosjektrapporten hvor alle matlabfilene er inkludert som vedlegg. Benytt liten font på selve Matlabkoden, typisk font 6, og behold fargekodene (grønn for kommentar, svart for kode og rosa for argumenter) fra Matlab siden dette

øker lesbarheten. Ikke levér matlabfiler enkeltvis. Microsoft Word vil automatisk slenge på en veldig irriterende grønn krøllete understrek i koden på grunn av stavekontrollen. For å fjerne denne, gå inn i innstillinger ta bort stavekontroll.

Navnet på prosjektrapporten skal være:

GruppeXXXX.pdf

der XXXX er gruppens nummer. Bruk "Mal for prosjektrapport.doc" som er lagt ut på emnets hjemmeside på ItsLearning. **Husk å konvertere prosjektrapporten til pdf til slutt og kontroller at alt ser riktig ut i pdf-utgaven.**

2. Den redigerte filmen av alle prosjektene deres. Vær klar over at itsLearning har maksgrense på ca 30Mb for en innlevering. Derfor må dere klippe/redigerer filmen (evenuelt lag den slik at den spiller av i dobbel hastighet) og lagre den i et format som gir god kompresjon. Filmen skal ikke være lenger enn 3 minutt.

   For at filmen totalt sett skal bli mindre enn 30Mb, kan dere gjøre følgende:

   - reduser kvaliteten/oppløsningen på opptaket i mobiltelefonen
   - last ned programmet Free Video Converter fra freemake.com, og konverter filmen til et valgfritt format og en størrelse mindre enn 30Mb (denne størrelsen kan dere sette i programmet).

   Som tidligere nevnt, må dere på prosjekteksamen ha med en bærbar PC/Mac med filmen klar til avspilling.

1.6 Vurdering av prosjektet

Prosjektet blir gitt bokstavkarakteren A, B, C, D, E eller F. Antall program og kompleksiteten av arbeidet teller mye, men ikke alt. Det hjelper ikke å komme med et briljant program, dersom man ikke kan forklare dybden i det eller hvordan man har laget det. Selve arbeidsprosessen teller også en del. Det vil si, evnen til å lære seg ting etter hvert og ta i bruk det man har lært. Rapportens kvalitet og matlabfilenes lesbarhet og kvalitet teller selvsagt også med i karakterfastsettelsen.

Dette representerer den enkelte bokstavkarakter:

A	Svært godt	Et svært godt arbeid med veldig bra selvstendig innsats. Innslag av originalitet. Klar og stringent presentasjon.
B	Meget godt	Et meget godt arbeid. Bra selvstendig innsats. God forståelse av stoffet. Meget god presentasjon.
C	Godt	Et godt arbeid. Noko selvstendig innsats. Grei forståelse av stoffet. God presentasjon.
D	Brukbart	Et brukbart arbeid. Liten selvstendig innsats. Rutinemessig bruk av standard metodikk eller oversiktsbetont arbeid av relativt lett tilgjengeleg stoff. Akseptabel presentasjon.
E	Tilstrekkelig	Dokumentert arbeidsinnsats av tilstrekkelig omfang. Oversiktsarbeid eller rutinemessig arbeid med svært liten selvstendig innsats. Varierande grad av vellykket presentasjon.
F	Ikke tilstrekkelig	Ikke bestått.

Prosjektet teller 60% av sluttkarakteren, og skriftlig eksamen 40%. Det er ikke mulig å klage på eksamensresultatet, verken prosjektdelen eller den skriftlige eksamen.

Sluttkarakter basert på alle mulige permutasjoner av karakterene fra den skriftlige eksamen og prosjektet. (P=prosjekt 60%, S=Skriftlig eksamen 40%, T=Total karakter.)

P	S	T	P	S	T	P	S	T	P	S	T	P	S	T	P	S	T
A	A	A	B	A	B	C	A	B	D	A	C	E	A	C	F	A	F
A	B	A	B	B	B	C	B	C	D	B	C	E	B	D	F	B	F
A	C	B	B	C	B	C	C	C	D	C	D	E	C	D	F	C	F
A	D	B	B	D	C	C	D	C	D	D	D	E	D	E	F	D	F
A	E	C	B	E	C	C	E	D	D	E	D	E	E	E	F	E	F
A	F	F	B	F	F	C	F	F	D	F	F	E	F	F	F	F	F

1.7 Plagiering

Plagiering betyr å utgi noe som andre har laget, som sitt eget. Plagiering eller forsøk på plagiering er juks. Dersom man blir tatt for juks, kan det medføre utestengelse fra alle eksamener ved alle universiteter og høgskoler i hele Norge i inntil ett år.

Dette er plagiering, og dermed ikke lov:

- Man har fått eller funnet kildekode eller annet materiale tilhørende en annen prosjekt-gruppe.

- Man har funnet kildekode eller lignende på Internett og bruker den som om det skulle være egenskrevet.

Dette er lovlig:

- Samarbeid og kommunikasjon med andre grupper, der man drøfter ulike problemstillinger og om hvordan de kan løses. Referer til disse andre gruppene og beskriv hvordan samarbeidet var.
- Man har funnet Matlab kildekode på Internett eller i Matlabboka som man bruker, men det kommer tydelig fram i rapporten og i kildekoden om hva som er egenskrevet kode og hva som er hentet kode. (For mye av denne type kode er ikke heldig med tanke på karakteren på prosjektet.)

Kapittel 2

Matlab, Lego NXT og Mindstorm Toolbox

2.1 Introduksjon

I dette kapittelet presenteres en Matlab-verktøykasse (*toolbox*) som er utviklet ved RWTH-universitet i Aachen i Tyskland (`www.mindstorms.rwth-aachen.de`). Verktøykassen heter RWTH-Mindstorm og denne gjør at vi kan bruke Matlab til å styre LEGO NXT-roboten (`mindstorms.lego.com`).

Vi skal se på hva du må gjøre for å etablerer kommunikasjon mellom Matlab og LEGO NXT på UiS sine PC'er på rom E-453 og E-455, samt hvordan du kan benytte din egen bærbare PC med win7 eller win8. De som har Mac må bruke Dual Boot for å få det til å virke. I tillegg blir det vist hvordan joysticken (vist i figuren under) kan styre en LEGO NXT-motor.

Videre blir de mest vanlige Matlab-kommandoene for å konfigurer NXT-sensorene vist, samt hvordan du kan konfigurere og styre motorene.

Viktig tips: Ikke bruk kreative symboler som mellomrom (space), æ, ø, å, :, ;, -, ?, + eller andre symbol i verken katalognavn eller filnavn. Dette gjelder HELE katalogstien. HUSK Å SJEKKE DETTE!!!!

2.2 Forberedelse på UiS-PC

Det som står i dette kapittelet må du utføre på en av UiS sine PC'er på rom E-453 eller E-455 i perioden **17. september** til **24. september**. Lab.ingeniør Romuald Bernacki på E-455c skal godkjenne at alt virker.

Dersom du også har egen PC du ønsker å bruke må du også gjøre forberedelsene vist i kapittel 2.7. Dette trenger du ikke å få godkjent.

2.2.1 Installasjon av USB-driver for NXT

Utgår høsten 2014.

2.2.2 Installasjon av RWTH Mindstorms toolbox og Joystick-programvare

På itsLearning ligger en fil som heter Lego.zip.

- Last denne ned og lagre den rett på F:-disken din.

Gå til F-disken din i Windows Explorer og finn filen. Originale innstillinger i Windows Explorer viser ikke fil-type automatisk, dvs. .zip vises ikke i filnavnoversikten. Siden det er veldig nyttig å vite hvilken type filer vi ser på, kan vi endre dette i Windows Explorer ved å gå inn i Organiser (øverst til venstre), deretter Mappe- og søkelternativ. Velg fanen Vis og ta bort Skjul filetternavn for kjente filtyper. Trykk deretter på knappen Bruk på mapper slik at det gjelder for hele katalogstrukturen, og deretter OK.

Høyreklikk på Lego.zip og velg 7-Zip og deretter Extract Here. Da har du fått en ny katalog på F-disken din som heter Lego. Under denne ligger driveren til joysticken i en egen katalog som heter Joystick og RWTH Mindstorms toolboxen i katalogen som heter RWTHMindstormsNXT. I tillegg ligger det en katalog som heter Prosjektfiler hvor du skal lagre filene dine.

Ikke lagre dine egne Matlab-filer på D-disken fordi det er ikke sikkert du jobber på samme lab-PC hver gang.

2.2.3 Setting av søkesti (path) i Matlab

Ved første gangs oppstart av Matlab, eller når du starter Matlab på en av UiS sine PC'er som du tidligere ikke har brukt, må du legge følgende 3 kataloger inn i søkestien (*path*) til Matlab:

- Skriv `pathtool` i Command Window (alternativt kan du trykke på knappen Set Path under HOME-fanen. Trykk deretter på **Add Folder...**, finn katalogen `F:\Lego\RWTHMindstormsNXT\` og trykk OK.
- Benytt samme prosedyre og legg til `F:\Lego\RWTHMindstormsNXT\tools\`
- Benytt samme prosedyre og legg til `F:\Lego\Joystick\`
- Trykk deretter **Save** og så **Close**.

Dersom du bruker en PC du har gjort dette på tidligere, vil disse dataene være lagret på din bruker. Dette kan du enkelt sjekke i Matlab ved å skrive path i Command Window. Ved å klikke en gang i Command Window og deretter bla deg oppover med Page Up-knappen vil du i toppen av listen se pathen du tidligere har satt.

Dersom du hver gang jobber på ny UiS-PC, kan du lage deg en Matlab-fil som legger disse katalogene automatisk til path'en ved å bruke `addpath`-funksjonen. Skriv `help addpath` så ser du hvordan den virker. Inkluder 3 slike addpath linjer i en fil og lagre den som f.eks. `LegoPath.m`, som du kjører ved oppstart på ny PC.

2.2.4 Første oppkopling mot joystick

For å benytte joysticken, trengs en C-kompilator i Matlab. Skriv `mex -setup` i Matlab, trykk på bokstaven N for *No* på første spørsmål, og velg deretter `Microsoft Visual C++ 2010` ved å skrive inn hvilket nummer denne har. Trykk på bokstaven Y for *Yes* på resten av spørsmålene.

- Fra Matlab-miljøet (Command window), gå inn inn katalogen `F:\Lego\Joystick\` ved å dobbleklikke deg frem i katalogstrukturen eller rett og slett ved å skrive `cd F:\Lego\Joystick\`. Husk at Matlab har autofullfør ved å trykke Tab.
- Åpne filen `build.m` og kjør denne (trykk f.eks. F5).

  Dersom du får feilmeldinger her, pass på at du står i riktig katalog. Dette sjekker du ved å skrive `pwd` (present working directory) i Command window. Dersom du ikke står i `F:\Lego\Joystick\` har du ikke gjort forrige punkt riktig. Feilmeldingen går ut på at build.m ikke finner .c filen siden katalogen src ikke er på den katalogen du nå står i.
- Sett USB fra joystick inn i PCen.
- Åpne deretter `joytest.m` fra `F:\Lego\Joystick\` og kjør denne. Da skal du få opp et vindu som leser av joystick-posisjon og trykknapper. Dersom du får en feilmelding i Command Window, restart Matlab og prøv igjen.
- Steng vinduet for å avslutte når du er ferdig

2.2.5 Testing med EksempelMotorA.m

For å teste at alt fungerer skal du gjøre følgende:

- Kople USB-ledningen mellom NXT og PC og skru på NXT'en, kople en motor i port A på NXT'en, lydsensoren i inngang 2, og kople USB-ledningen mellom joystick og PC.

- Åpne programmet `EksempelMotorA.m` som ligger under `F:\Lego\Prosjektfiler\` og kjør denne filen (trykk f.eks. F5).
- Benytt joysticken til å gi gass (bøy forover). Motor A skal nå gå samtidig som det øverst i figur 1 vises en søylerepresentasjon av gasspådraget og nederst en representasjon av pådraget som funksjon av tid. I figur 2 vises lydverdien øverst, og nederst er lydverdi (blå) og pådrag til motor A (rød) vist i samme subplot.
- For å stoppe programmet, trykk på bryteren du har ved pekefingeren (i front av joysticken)

Dersom dette ikke virker som forventet, ta kontakt med lab.ingeniør Romuald på E-455c.

2.3 Om kommunikasjonen mellom PC og NXT

I dette kapittel beskrives i mer detalj kommandoene som er brukt i EksempelMotorA.m. For å teste kommandoene som presenteres nedenfor kan du lage deg en ny .m fil og lagre den under F:\Lego\Prosjektfiler\, og deretter teste ved å legge inn linje for linje det som presenteres.

2.3.1 Initielle kommandoer

Kommunikasjonen mellom NXT og Matlab benytter såkalte håndtak (*handles*) som er returvariabelen fra kommandoen `COM_OpenNXT` [1]. Disse håndtakene ser ut som vanlig variable i Matlab (men er egentlig en *struct* med mye data tilknyttet).

Dersom et program som du har laget avbrytes på grunn av feil kode eller lignende, vil disse kommunikasjonshåndtakene fortsatt være åpne neste gang du prøver å kjører programmet. Av den grunn er det viktig å alltid lukke alle NXT-håndtak, slette alle variable og lukke alle figurer i begynnelsen av et program, før kommunikasjonen på ny opprettes mot en NXT som vist under:

```
COM_CloseNXT all              % lukker alle NXT-håndtak
close all                     % lukker alle figurer
clear all                     % sletter alle variable
handle_NXT = COM_OpenNXT();   % etablerer nytt håndtak
```

hvor `handle_NXT` er selve håndtaket.

[1] Skriv `help COM_OpenNXT` i Command Window så ser du hvordan funksjonen er definert.

2.3.2 Spill en lyd

Når du ønsker å utføre noe på den tilkoblede NXT'en, kan du gjøre det ved å sende over håndtaket til funksjonen du bruker, for eksempel:

```
NXT_PlayTone(440,500,handle_NXT);
```

I dette eksempelet spilles en tone. Dersom du ikke sender med `handle_NXT` som siste argument til `NXT_PlayTone()` [2], vet ikke funksjonen hvilken NXT som skal brukes.

Dersom du bare benytter én NXT (og ikke flere NXT'er samtidig mot samme PC), er det mulig å definere et standard (*default*) håndtak, slik at du ikke trenger å benytte håndtak i funksjonskallene. Dette gjøres på følgende måte :

```
COM_CloseNXT all                % lukker alle NXT-håndtak
close all                       % lukker alle figurer
clear all                       % sletter alle variable
handle_NXT = COM_OpenNXT();     % etablerer nytt håndtak
COM_SetDefaultNXT(handle_NXT);  % setter globalt standard-håndtak
NXT_PlayTone(440,500);          % spill tone
COM_CloseNXT(handle_NXT);       % Ikke glem å lukke håndtaket!
```

Som du ser, velger vi å lukke håndtaket i slutten av programmet (selv om vi lukker alle i begynnelsen av programmet også).

2.4 Sensorer

De 5 sensorene som følger med byggesettet er (engelsk navn som brukes i programmeringen er gitt i parantes):

- bryter (Switch)
- lyd (Sound)
- lys (Light)
- ultralyd (Ultrasonic)
- akselerometer (Accelerator)

[2] Skriv `help NXT_PlayTone` i Command Window så ser du hvordan funksjonen er definert.

2.4.1 Initialisere/åpne sensorene

Disse kan du kople til NXT'en i inngangsportene 1 til 4, hvor syntaxen for portnummeret er `SENSOR_1` til `SENSOR_4`. For å benytte sensorene må du først *åpne* de via kommandoen `Open...` som benytter porten som argument, for eksempel:

```
OpenSwitch(SENSOR_1)
OpenSound(SENSOR_2,'DB')      % spesifiseres som DB eller DBA
OpenLight(SENSOR_3,'ACTIVE') % spesifiseres som ACTIVE eller DEACTIVE
                              % (sender/sender ikke ut rødt lys)
OpenUltrasonic(SENSOR_4)
OpenAccelerator(SENSOR_4)
```

Alle disse kommandoene kan du lese mer om ved å skrive `help Open...`.

2.4.2 Avlese sensorene

For å avlese verdien til en sensor benyttes kommandoen `Get...`, som også benytter porten som argument:

```
GetSwitch(SENSOR_1)       % gir verdien 0 eller 1 (inntrykket)
GetSound(SENSOR_2)        % gir verdi mellom 0 og 1023 (bare lydvolum, ikke frekvens
GetLight(SENSOR_3)        % gir verdi mellom 0 og 1023, tilsvarer mørkt og lyst
GetUltrasonic(SENSOR_4)   % gir verdi mellom 0 og 30 cm (gir verdi 255
                          % dersom ingen refleksjon)
GetAccelerator(SENSOR_4) % gir verdi mellom -400 og 400 (tilsvarer +/- 2g)
```

For å finne ut mer om hver sensor, skriv `help Get...`.

2.4.3 Lukke sensorene

På slutten av programmet lukkes alle åpnede sensorer slik:

```
CloseSensor(SENSOR_1)
CloseSensor(SENSOR_2)
CloseSensor(SENSOR_3)
CloseSensor(SENSOR_4)
```

Vit at funksjonen `GetUltraSonic` er halvparten så rask som de andre sensorene siden 4 pakker må sendes over USB-kommunikasjonen i forhold til 2 pakker for de andre. Vi har også et par gyro-instrumenter for de som vil bruke det mot slutten av prosjektet.

2.5 Styring av én NXT-motor

I dette kapittelet skal vi gå gjennom styring av NXT-motorene.

2.5.1 Beskrivelse av motoregenskaper

For å betjene motorer må vi opprette motorobjekter, dvs. instanser/objekter av klassen `NXTMotor`. Disse objektene er i stand til å sende ulike kommandoer til en eller flere motorer, eller de kan brukes til å hente data fra motorene. For å studere denne klassen, skriv i Command Window (du trenger ikke ha noen motor tilkoplet):

```
NXTMotor
```

og du vil se følgende egenskaper som kan benyttes for å kontrollere NXT-motorene:

- `Port`
  Angir hvilken port motoren er tilknyttet.
- `Power`
  Setter retning og kraft (dvs. energimengde) til motoren. Integerverdi: Fra -100 (full fart bakover) til 100 (full fart fremover). Defalt 0.
- `SpeedRegulation`
  Bestemmer hvorvidt turtallet skal reguleres. Verdi 0 eller 1 (default). Når turtallsregulering er aktivert, vil verdien du har satt i `Power` representerer *ønsket turtall* istedenfor *bare kraft* (tilsvarer default-situasjonen).
  - Dersom en motor gjør tungt arbeid, vil regulatoren internt i firmware øke effekten til motoren (hvis mulig) for å holde konstant turtall. Utvis derfor forsiktighet ved bruk av denne egenskapen siden tannhjulene kan ødelegges dersom motoren møter en hindring.
  - Ved lave hastigheter vil turtallsreguleringen ofte resultere i det motsatte, altså at turtallet blir ustabilt og varierer mye mer enn ønskelig.
  - Når det er ønskelig med konstant *moment* (ikke turtall), må `SpeedRegulation` deaktiveres.
- `SmoothStart`
  Angir om motoren skal akselerere rolig opp til maks fart (selv om det skjer ganske fort). Dette er for eksempel nyttig ved kjøring av roboter slik at hjulene ikke spinner.
  - Denne egenskapen fungerer *kun* når `TachoLimit > 0` er satt (se neste egenskap).

  - Den fungerer ikke med `ActionAtTachoLimit = 'Coast'` (se siste egenskap lenger ned).

- `TachoLimit`
  Setter en vinkelverdi mellom 0 og 999999 som motoren skal gå til, hvor en runde er 360 grader. Nøyaktighet ± 1 grad. Default 0 (=ingen grense). Se eksempel i kap 2.5.3.
- `ActionAtTachoLimit`
  Bestemmer hva som skal skje når vinkelverdien er nådd. Valgene er:
  - `Coast`
    Når `TachoLimit` er nådd slås motoren av og motoren vil fortsette å rotere (på grunn av rotasjonsmomentet) helt til den rolig stopper.
  - `Brake`
    Motoren bremser rolig opp før `TachoLimit` er nådd slik at den stopper på `TachoLimit` (default). Motoren slås deretter av, og kan rotere fritt.
  - `Holdbrake`
    Samme som `Brake`, men de elektriske bremsene aktiveres når `TachoLimit` er nådd slik at motoren ikke roterer fritt.

2.5.2 Initialisere motor. Start og stopp av motor

- Benytt .m-filen du har laget med følgende (kjente) initielle kommandoer:

```
COM_CloseNXT all              % lukker alle NXT-håndtak
close all                     % lukker alle figurer
clear all                     % sletter alle variable
handle_NXT = COM_OpenNXT();   % etablerer nytt håndtak
COM_SetDefaultNXT(handle_NXT);   % setter globalt standard-håndtak
```

- Kople en motor i port A på NXT'en
- For å teste om motoren virker skal vi kjøre den forover med 50% kraft med myk start og ingen turtallsregulering. Du kan da benytte en av følgende 2 måter å konstruere motorobjektet på:
  - **Alternativ 1** som går veien via et tomt motorobjekt (første linje kalles konstruktør), for deretter å spesifisere ønskede egenskaper (fra listen i kap. 2.5.1) fortløpende:

```
motorA = NXTMotor();     % oppretter først et tomt motorobjekt
motorA.Port = 'A';       % alternativt motorA.Port = MOTOR_A;
motorA.Power = 50;       % halv kraft forover
motorA.SpeedRegulation = false;  % ingen hastighetsregulering
motorA.SmoothStart = 1; % myk start (kunne også brukt true)
```

Denne måten er mest egnet dersom du allerede har opprettet et objekt hvor du senere i programmet ønsker å endre bare en ting, f.eks. `Power`.

– **Alternativ 2** spesifiserer alt samtidig ved å benytte Matlabs måte å spesifisere egenskaper på `...,'Property',value,'Property',value`, dvs:

```
motorA = NXTMotor('A','Power',50,...
                  'SpeedRegulation',false,'SmoothStart',1);
```

- For å få utført disse motoregenskapene på dette motorobjektet, må du skrive følgende:

```
motorA.SendToNXT();
```

Legg merke til at `SendToNXT()` ikke er en egenskap til `motorA` selv om det kan se slik ut siden funksjonen kalles med et punktum på samme måte som `motorA.Power`. Poenget er at `SendToNXT()` er en egen funksjon som trenger et objekt i forkant av funksjonskallet, skriv `help SendToNXT` i Command Window.

- For å stoppe, skriv:

```
motorA.Stop;
```

Dette slår kun av spenningen til motoren slik at den fortsetter å rotere på grunn av momentet (du kunne også skrevet `motorA.Stop('off')`). Motoren vil på denne måten stoppe på en myk måte (tilsvarende *Coast* beskrevet over). Dersom du ønsker å stoppe umiddelbart må du skrive

```
motorA.Stop('brake');
```

I dette tilfellet vil NXT'en aktivere bremsen for å stoppe motoren. Bremsen holder motoren i posisjon etter at den er stoppet (fungerer som et håndbrekk). Skriv `help Stop` for utfyllende informasjon.

Et siste alternativ (kriseløsningen) til å stoppe motoren er å trykke på den oransje knappen på NXT (av/på-knappen) og deretter trykke Ctrl-C i Command Window.

2.5.3 Mer om `TachoLimit`

Her vises et eksempel på kode for å få motoren til å rotere nøyaktig en runde:

```
motorA.TachoLimit = 360;  % en hel runde
motorA.SendToNXT();
```

Kommandoen `motorA.SendToNXT()` setter bevegelsen i gang og NXT'en passer selv på å stoppe etter en runde (motoren har altså intern posisjonsmåling). Dette betyr at du kan sende andre instruksjoner, dvs. instruksjoner som ikke påvirker `motorA` siden denne skal gå en hel runde først, til NXT'en som den da utfører parallelt, f.eks.:

```
motorA = NXTMotor('A','Power',50,'TachoLimit',1000);
motorA.SendToNXT();
NXT_PlayTone(440,500)
```

Dersom du ønsker å lage lyd *etter* at motoren har gått 1000 grader, må du benytte `motorA.WaitFor()` funksjonen slik:

```
motorA = NXTMotor('A','Power',50,'TachoLimit',1000);
motorA.SendToNXT();
motorA.WaitFor();
NXT_PlayTone(440,500)
```

Dette innebærer at `NXT_PlayTone(440,500)`-kommandoen blir satt på vent til motoren har rotert 1000 grader.

Som nevnt over kan du ikke sende nye instruksjoner til en motor mens den roterer seg frem til `TachoLimit`. Det betyr at følgende kode *ikke* fungerer:

```
motorA = NXTMotor('A','Power',50,'TachoLimit',360); % opprett objektet
motorA.SendToNXT();
motorA.Power = -80;  % setter ny hastighet, motsatt vei
motorA.SendToNXT();  % DEN NYE HASTIGHETEN BLIR IKKE SATT SIDEN
                     % MOTOREN ER OPPTATT MED Å ROTERE EN HEL RUNDE
```

mens følgende kode uten `TachoLimit` fungerer:

```
motorA = NXTMotor('A','Power',50); % opprett objektet
motorA.SendToNXT();
motorA.Power = -80;  % setter ny hastighet, motsatt vei
motorA.SendToNXT();  % DEN NYE HASTIGHETEN SETTES UMIDDELBART
```

2.6 Avlesing og nullstilling av motorposisjon

I tillegg til å sende kommandoer *til* motorer, kan vi også hente informasjon *fra* dem, som for eksempel å lese av den nåværende posisjon (avlest i grader) eller å nullstille posisjonen. Til å gjøre dette benyttes kommandoen `ReadFromNXT` og `ResetPosition`.

Når du kaller funksjonen `ReadFromNXT()` på et motorobjekt, returneres en MATLAB struct som inneholder nyttige egenskaper, bl.a. `Position`. Dette er en vinkelteller som oppdateres ved enhver motorbevegelse, enten du snur motoren for hånd eller du kjører motoren automatisk. Kjører du motoren med positiv verdi på `Power`, øker verdien på `Position`. Kjører du motoren med negativ verdi på `Power`, minker verdien på `Position`.

Et eksempel på avlesing av motorposisjon, rett etter stoppsignal er gitt og etter at den til slutt er stanset å rotere, er vist under:

```
motorA = NXTMotor('A','Power',50);
motorA.ResetPosition();       % nullstill vinkelteller
motorA.SendToNXT();           % starter motor A
pause(1);                     % kjør i 1 sekund
motorA.Stop;                  % motor stoppes, fortsetter å roterer
data = motorA.ReadFromNXT();  % hent motorinformasjon og presenter
disp(sprintf('Motor A er for tiden i posisjon %d',data.Position));
pause (2);                    % Venter til motoren står i ro
data = motorA.ReadFromNXT (); % avles posisjonen igjen og presenter
disp(sprintf('Motor A stoppet til slutt i posisjon %d',data.Position));
```

2.7 Installasjon på egen PC med win7

Last ned filen `InstallasjonEgenPC.zip` fra itsLearning og pakk den ut på roten av f.eks. C:\ disken. Husk å endre utpakkingssted til `C:\` dersom utpakkingsprogrammet foreslår et annet katalognavn. Under C:\Lego\ finner du nå alle kataloger og filer du trenger for å installere på egen PC.

2.7.1 Installasjon på win8

Installasjonen i dette kapittelet er tilpasset win7. Dersom du har win8 på egen PC, kan du følge oppskriften gitt her:

`www.craftedge.com/tutorials/driver_install_windows8/driver_install_win8.html`

2.7.2 Installasjon av Matlab

Dersom du ikke allerede har installert Matlab, gå til `http://matlab.ux.uis.no` og last ned.

2.7.3 Installasjon av Visual Studio 2010

For at joysticken skal virke, må du installere Visual Studio 2010. Dette gjør du ved å følge installasjonsveiledningen i dokumentet som ligger på itsLearning:
Veiledning for installasjon av Visual Studio 2010.pdf

2.7.4 Installasjon av USB-driver for NXT

For å kommunisere med NXT'en via USB-porten, må du gjøre følgende (ligner delvis på det du har gjort på UiS sine PC'er i kap 2):

- Sett USB-kabelen i PC og NXT, og skru deretter på NXT'en.
- Gå inn i `C:\Lego\InstallasjonEgenPC\libusb\bin\` og kjør programmet `inf-wizard.exe`. Velg Unknown Device #1 under Device Selection.
- Lagre filen `Unknown_Device_#1.inf` der hvor det blir foreslått. Installer driverfil ved å trykke Install Now.. Dersom Windows mener du må gi tillatelse til å installere, trykk Ja. Dersom nok en melding dukker opp hvor Windows ikke kan kontrollere utgiver av denne driverprogramvare, trykk Installer denne driverprogramvaren alikevel. Trykk OK når installasjonen er ferdig.
- Gå deretter inn i `C:\Lego\InstallasjonEgenPC\fantom\`, kjør filen `setup.exe` og trykk Next og til slutt Finish.
- Gå deretter på ny inn i `C:\Lego\InstallasonEgenPC\libusb\bin\` og kjør programmet `inf-wizard.exe` en gang til. Velg LEGO MINDSTORMS NXT under Device Selection og trykk Next.
- Lagre filen `LEGO_MINDSTORMS_NXT.inf` der hvor det blir foreslått. Installer driverfil på samme måte som beskrevet over.

2.7.5 Installasjon av RWTH Mindstorms toolbox og Joystick-programvare

På `C:\Lego\` ligger allerede de tilsvarende katalogene som du installerte på UiS sine PC'er (joystick-driver og RWTH-toolbox), nemlig katalogen Joystick og RWTH Mindstorms toolbox katalogen RWTHMindstormsNXT. Også her ligger det i tillegg en katalog som heter Prosjektfiler hvor du skal lagre filene dine. Her trenger du med andre ord ikke installere noe som helst.

2.7.6 Setting av søkesti (path) i Matlab

På samme måte som på UiS sine PC'er, må du sette søkestien (path):

- Legg inn katalogen `C:\Lego\RWTHMindstormsNXT\` i søkestien (*path*) til Matlab ved å benytte samme prosedyre som på UiS sine PC'er, kapittel 2.2.3.
- Benytt samme prosedyre og legg katalogen `C:\Lego\RWTHMindstormsNXT\tools\` i path'en.
- Benytt samme prosedyre og legg katalogen `C:\Lego\Joystick\` i path'en.
- Trykk deretter **Save** og så **Close**. Dersom Windows nok en gang mener du må gi tillatelse, trykk Ja, før du trykker **Close**.

2.7.7 Første oppkopling mot joystick

For å benytte joysticken, trengs en C-kompilator i Matlab (på UiS sine PC'er var denne allerede installert). Skriv `mex -setup` i Matlab, trykk på bokstaven N for *No* på første spørsmål, og velg deretter Microsoft Visual C++ 2010 (denne må du altså ha installert først som beskrevet i kap. 2.7.3) ved å skrive inn hvilket nummer denne har. Trykk på bokstaven Y for *Yes* på resten av spørsmålene.

- Fra Matlab-miljøet (Command window), gå inn inn katalogen `C:\Lego\Joystick\` ved å navigere deg frem i katalogstrukturen eller rett og slett ved å skrive `cd C:\Lego\Joystick\`. Husk at Matlab har autofullfør ved å trykke Tab.

- Åpne filen `build.m` og kjør denne (trykk f.eks. F5).

  Dersom du får feilmeldinger her, pass på at du står i riktig katalog. Dette sjekker du ved å skrive `pwd` (present working directory) i Command window. Dersom du ikke står i `C:\Lego\Joystick\` har du ikke gjort forrige punkt riktig. Feilmeldingen går ut på at build.m ikke finner .c filen siden katalogen src ikke er på den katalogen du nå står i.

- Sett USB fra joystick inn i PCen.
- Åpne deretter `joytest.m` fra `C:\Lego\Joystick\` og kjør denne. Da skal du få opp et vindu som leser av joystick-posisjon og trykknapper.
- Steng vinduet for å avslutte når du er ferdig

2.7.8 Testing

På samme måte som på UiS sine PC'er, skal du teste at alt fungerer ved å gjøre følgende:

- Kople USB-ledningen mellom NXT og PC og skru på NXT'en, kople en motor i port A på NXT'en, og kople USB-ledningen mellom joystick og PC.
- Åpne programmet EksempelMotorA.m som ligger under `C:\Lego\Prosjektfiler\` og kjør denne filen (trykk f.eks. F5).
- Benytt joysticken til å gi gass (bøy forover). Motor A skal nå gå samtidig som en figur vises på skjermen som viser pådraget.
- For å stoppe programmet, trykk på bryteren du har ved pekefingeren (i front av joysticken).

2.8 Tips om Matlab og NXT

Her er det en liste med Matlabtips som kan være nyttige:

- Ikke gi kataloger og filer samme navn.
- Matlab har autofullfør i ved bruk av tabulator.

- Hjelp til de forskjellige funksjonene i RWTHMindstorms toolboxen finner du under `..\Lego\RWTHMindstormsNXT\doc\`. Her er det `.html` sider som du kan åpne.

- Hvordan finne nyttige funksjoner i Matlab? Dersom du ikke finner kommandoen direkte med `help` kommandoen, kan du bruke kommandoen `lookfor` som søker gjennom hele hjelpeteksten til hver funksjon.

  Som et eksempel skal dere bruke noen ferdige funksjoner i Matlab for avlesing av tid når dere skal numerisk integrere. Siden `time` ikke er en funksjon, vil `help time` ikke gi noe resultat. Derimot vil `lookfor time` gi egentlig litt for mange forslag, men ved å lese hver setning som beskriver hver funksjon i den lange listen vil dere finne mange alternative funksjoner for å få tak i tid som kan brukes i integrasjonen.

  Ved å søke på Google vil du også finne mye informasjon.

- Hvordan finne ut hvordan joysticken virker og hvilke verdier den returnerer?

  I koden, ta bort semikolon bak linjen som leser joystickverdier. Disse vil da presenteres i Command Window under kjøring.

- Hvordan finne ut hvor det er feil i koden?

  En måte er å debug-funksjonen i Matlab og steppe dere gjennom koden. En annen måte er å ta bort semikolonet bak de linjene dere mistenker er feil. I tillegg kan dere skrive variabelnavn uten semikolon i selve koden slik at verdien av variabelen skrives ut til Command Window under kjøring.

- Viktig tips:

Ikke bruk kreative symboler som mellomrom, æ, ø, å, :, ;, -, ?, + eller andre symbol i verken katalognavn eller filnavn. Dette gjelder HELE søkestien til den delen av katalogstrukturen hvor matlabfilene dine er.

- Samplingstid T_s varierer med hvorvidt du har figurer som plotter i hver loop i while-løkka. Dette kan påvirke hvor godt roboten kjører langs banen. Test evnt dette med å plotte alt til slutt etter while-løkken

- Sirkelbevegelse av lyssensor på banen generer en sinus.

- Husk å benytte `WaitFor()` for å unngå at NXT piper

- Dersom du bruker integrasjon, derivasjon og filtrering i mange småprogrammer, kan det være smart å lage disse om til funksjoner

- Plotting av figurer krever mye ressurser i Matlab, noe som gjør at kommunikasjonshastigheten over USB ledning reduserer dramatisk. Dette kan innvirke på mange ting, bl.a. automatisk kjøring. En måte å unngå dette på er å bruke handles til figurer, eller å plotte til slutt.
- Joysticken står nødvendigvis ikke i senter når den er ubelastet, og motoren(e) vil da gå selv om du ikke gir gass. Lag da en initialiseringsrutine som helt først i programmet leser posisjonen til joysticken ubelastet og trekker fra disse posisjonsverdiene i alle beregninger slik at du lager deg et nytt 0-pkt.
- Dersom du lager veldig mange if-løkker etter hverandre eller benytter mye repeterende kode, prøv å komprimer koden ved å bruke vektorer/indekser eller funksjoner.
- Husk å benytte innrykk i koden slik at koden er lettere å lese. Trykk da Ctrl-A (merk alt) og deretter Crtl-I (for indentering).
- Vit at laderne av og til svikter. Denne får du ny hos Romek.
- Vit også at NXT-utstyret kan svikte. For å finne ut om så er tilfelle, feilsøk ved å sammenligne motorer i forhold til hvor fort de sviver rundt (avles posisjon på begge ved samme pådrag), eventuelt lån sensorutstyr av nabogruppa for å sammenligne med egne sensorer. Test også med å bytte ledninger (gjelder både USB og flatkabelen).
- Dersom NXTen ikke virker (når den burde virket), sjekk at filen MotorControl22.rxe ligger inne (bla i menyen på NXT). Dersom den er borte, kan Romek legge den inn på ny.

Kapittel 3

Numerisk integrasjon, derivasjon og filtrering

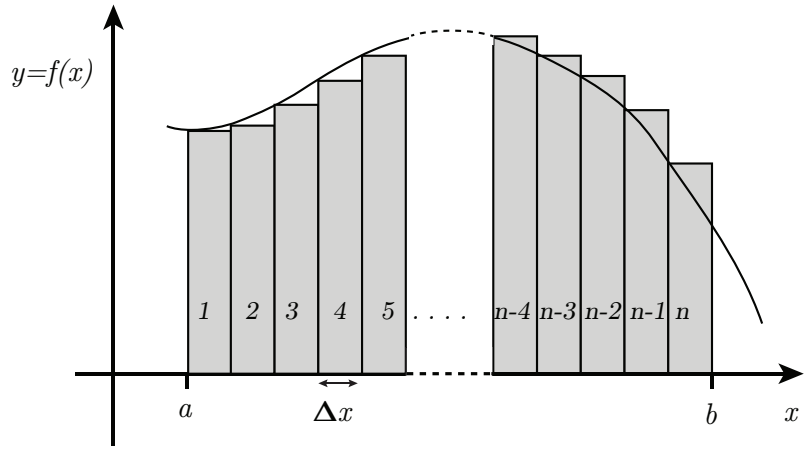
3.1 Numerisk integrasjon

3.1.1 Fra matematikkteori: Integrasjon av en *kontinuerlig* funksjon

Når vi i matematikken løser likninger, beregner integral, eller løser differensialligninger, vil vi ofte ha eksakte (analytiske) løsninger. I praksis (dvs. anvendt matematikk) er dette ofte vanskelig å få til, og da må vi nøye oss med tilnærmede (numeriske) løsninger. Her skal vi se på en forholdsvis enkel teknikk for å finne en slik numerisk løsning der det bestemte integralet av en vilkårlig funksjon $f(x)$ fra $x = a$ til $x = b$ ikke lar seg løse eksakt. Denne teknikk kalles altså *numerisk integrasjon*. Matematisk kan problemet formuleres som i ligning (3.1)

$$\int_a^b f(x)dx \tag{3.1}$$

og fra matematikken husker vi at integrasjonsteorien tar utgangspunkt i at dette integralet kan betraktes som arealet under kurven $f(x)$ fra $x = a$ til $x = b$. Ved å dele arealet opp i n rektangler med lik bredde og summere arealene av disse, kan vi altså finne en tilnærmet verdi for integralet, se figur 3.1.



Figur 3.1: Prinsippet for integrasjon.

Hvert rektangel har altså bredde $\Delta x = \frac{b-a}{n}$, og arealet A kan beregnes som summen av arealet av alle rektanglene:

$$A = \sum_{i=0}^{n} f(a + i \cdot \Delta x) \cdot \Delta x \tag{3.2}$$

Den numeriske metoden vi skal se på, bygger på samme prinsipp. Ved å la Δx gå mot 0, og dermed antall rektangler n gå mot uendelig, finner vi det nøyaktige arealet under kurven.

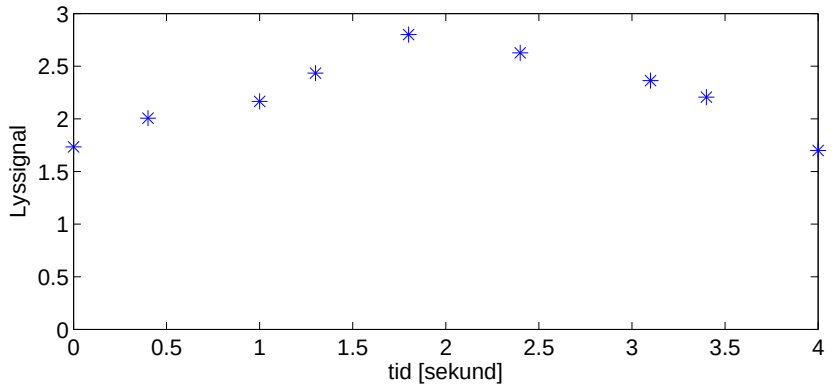
3.1.2 Fra anvendelse: Integrasjon av en *tidsdiskret* funksjon

I en praktisk anvendelse som f.eks. numerisk integrasjon av målesignalet fra lyssensoren til LEGO NXT, vil ikke lyssignalet se ut som den kontinuerlige kurven i figur 3.1. Blant annet er x-aksen blitt tiden t istedenfor variabelen x, og den kontinuerlige kurven er blitt erstattet av *tidsdiskret* målepunkter. Dette skyldes at når Matlab sender ut en forespørsel til NXT'en og ber om lyssignalet, vil NXT'en returnere lysverdien i akkurat det tidspunktet (og ingenting annet). Dersom du repeterer forespørselen rekursivt vil du få en rekke fortløpende verdier som er logget ved økende tidspunkter. Dette kalles ofte for sampling eller punktprøving, og hurtigheten dette skjer med kalles samplingsfrekvens (antall sample pr sek).

I tillegg vil tiden fra forespørsel i Matlab til retur av verdi fra NXT variere litt for hver forespørsel. Dette fordi NXT'en kan være opptatt med andre ting, kommunikasjonsraten over USB-ledningen varierer, eller at PC'en har varierende arbeidsoppgaver

(for eksempel plotting av figurer mens du kjører). Konkret betyr dette at Δx eller samplingsraten varierer med tiden, og vi kan ikke numerisk integrere (beregne arealet) som i ligning (3.2) hvor Δx var fast. Den aktuelle tidsforskjellen mellom to tidsverdier kalles ofte tidsskritt T_s eller steglengde.

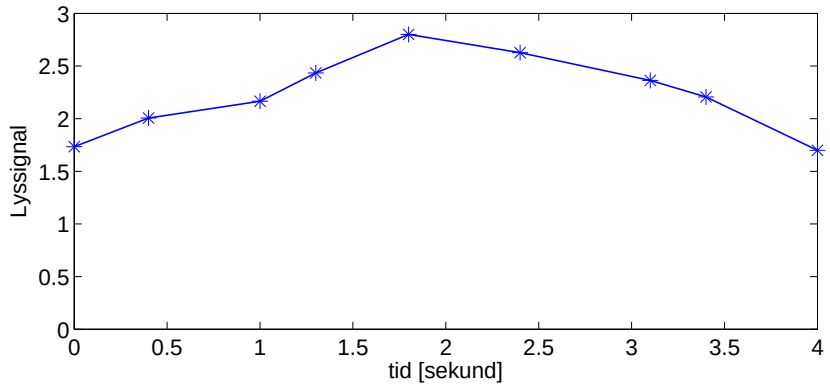
Ved å ta vare på lysverdiene og de tilhørende tidspunktene i to forskjellige vektorer og plotte disse mot hverandre, vil vi få et *tidsdiskret* lyssignal, se figur 3.2. Her ser vi at samplingstiden varierer siden vi har målinger til følgende tidspunkt:
$t \in \{0, 0.4, 1.0, 1.3, 1.8, 2.4, 3.1, 3.4, 4.0\}$.



Figur 3.2: Logget lyssignal. Samplefrekvensen er ca. 2Hz (dvs. 2 sample pr sekund)

For å vise at målesignalet er tidsdiskret, er kurven i figur 3.2 er plottet med kommandoen `plot(Tid,Lys,'*')` (`Tid` og `Lys` er vektorer med data). Vi vet ingenting om lysverdien mellom målepunktene, men dersom vi antar at samplingen foregår hurtig nok i forhold til endringene i målesignalet, er en rett linje mellom punktene det beste estimatet for å lage en kontinuerlig kurve[1], se figur 3.3.

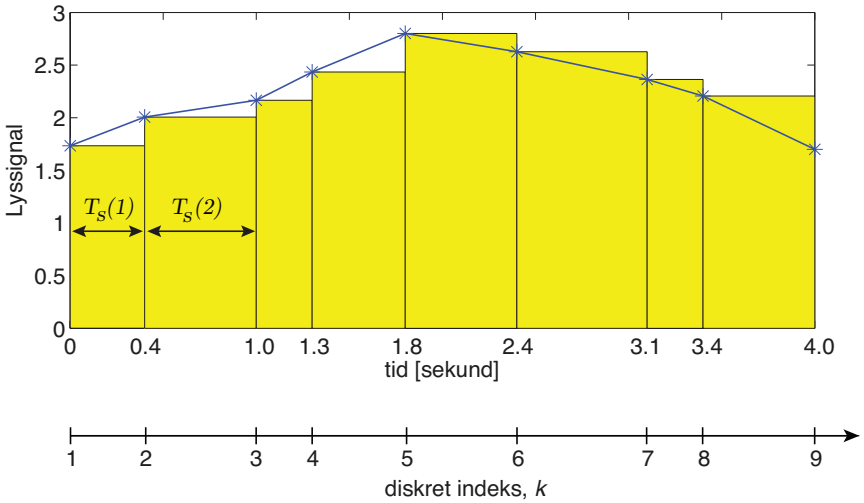
[1] Figur 3.3 er laget slik: `plot(Tid,Lys,'*'); hold on; plot(Tid,Lys)`



Figur 3.3: Lineær interpolasjon mellom målepunktene.

Dersom målingene foregår sjelden, kan det være mye informasjon vi går glipp av mellom målingene. Dette kalles aliasing, og viktig å være klar over i signalbehandling (se avansert prosjektforslag på side 12). For å lese mer om dette, gå f.eks. inn på Store norske leksikon (snl.no) og søk etter aliasing.

Siden vi tar vare på måleverdiene og tidsverdiene i vektorer, og disse indekserer med heltallsverdier, innfører vi diskrete heltallsverdier for tidspunktene, altså at $k{=}1$ når $t{=}0$, $k{=}2$ når $t{=}0.4$ osv. Dette er indikert i figur 3.4.



Figur 3.4: Arealberegning av et tidsdiskret signal.

For å beregne arealet under kurven skal dere summere rektangel-arealene, men siden dette skal utføres i sann tid mens dere kjører Legoroboten, skal dere altså beregne

arealet A under kurven fortløpende frem til nåværende diskret tid k, dvs. $A(k)$. Ved å studere figur 3.4 ser du at:

- i tidspunkt k=1 er arealet 0, dvs. $A(1)$=0
- i tidspunkt k=2 består arealet av det første rektangelet, dvs $A(2)$=$Lysverdi(1)\cdot T_s(1)$
- i tidspunkt k=3 består arealet av summen av de to første rektanglene som: $A(3)$=$Lysverdi(1)\cdot T_s(1) + Lysverdi(2)\cdot T_s(2)$

Ved å bruke at $A(1)$=0, kan vi uttrykke den fortløpende arealberegningen som:

$$A(k) = \sum_{n=2}^{k} Lysverdi(n-1)\cdot T_s(n-1) \tag{3.3}$$

hvor $T_s(n-1) = tid(n) - tid(n-1)$, se figur 3.4.

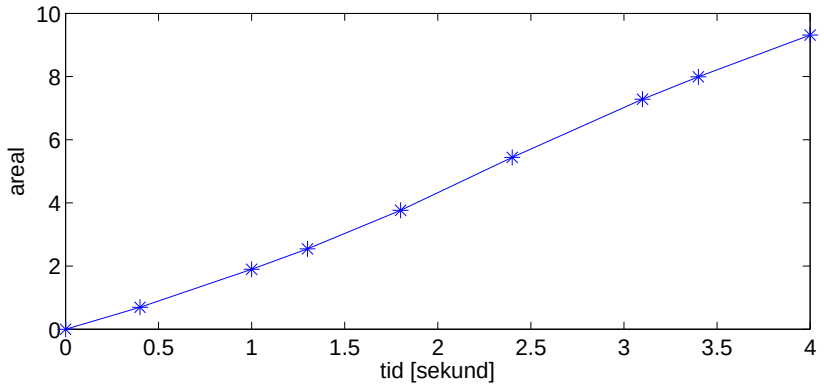
Ved å vise hvordan ligning (3.3) forløper i hvert tidsskritt, ser vi hvordan vi kan implementere denne ligningen i en algoritme for å beregne arealet rekursivt (i en while eller for-løkke).

$$\begin{aligned}
k=1: \quad & A(1) = 0 \\
k=2: \quad & A(2) = Lysverdi(1)\cdot T_s(1) \\
k=3: \quad & A(3) = \underbrace{Lysverdi(1)\cdot T_s(1)}_{A(2)} + Lysverdi(2)\cdot T_s(2) \\
& \quad = A(2) + Lysverdi(2)\cdot T_s(2) \\
k=4: \quad & A(4) = \underbrace{Lysverdi(1)\cdot T_s(1) + Lysverdi(2)\cdot T_s(2)}_{A(3)} + Lysverdi(3)\cdot T_s(3) \\
& \quad = A(3) + Lysverdi(3)\cdot T_s(3) \\
k=5: \quad & A(5) = A(4) + Lysverdi(4)\cdot T_s(4) \\
k=6: \quad & A(6) = A(5) + Lysverdi(5)\cdot T_s(5)
\end{aligned}$$

eller generelt

$$A(k) = A(k-1) + Lysverdi(k-1)\cdot T_s(k-1) \tag{3.4}$$

Som du ser må vi ta vare på arealets forrige verdi $A(k-1)$ for å beregne nåverdien av arealet. Den numeriske integrasjonen av dataene i figur 3.4 (eller arealets fortløpende utvikling) blir da som vist i figur 3.5.



Figur 3.5: Numerisk integrasjon av dataene i figur 3.4.

3.1.3 Pseudokode for anvendelse mot NXT

Siden alle lys-, lyd- og ultralydverdier er positive, kan vi definere en fiktiv nullverdi f.eks. midtveis på måleverdiskalaen (tilsvarende 50% grått for lyssensoren), som gjør at vi kan få både positive og negative verdier på signalene. Følgende pseudokode illustrerer numerisk integrasjon som dere skal benytte i poengberegningen når dere kjører roboten langs banen (kopier denne og endre den til den fungerer):

```
nullpunkt = GetLight;          % 50% grått (senterstrek)
k=1;                           % diskret tellevariabel
Lys(k)    = GetLight;          % første måling av lys
Tid(k)    = nå;                % starttidspunkt
A(k)      = 0                  % initialverdi areal, A(1)=0
k=k+1;  % øker diskret indeks og går inn i while-løkke
while true
   Lys(k)    = GetLight;              % ny måling av lys
   Tid(k)    = nå;                    % ny tidsverdi
   Ts(k-1)   = Tid(k)-Tid(k-1);       % beregn tidsskritt i sekund
   avvik(k-1) = Lys(k-1)-nullpunkt;  % beregn avvik fra nullpunkt
   A(k)      = A(k-1) + avvik(k-1)*Ts(k-1); % numerisk integrerer
   k=k+1;                           % oppdaterer tellevariabel
end
figure
plot(Tid,A)
xlabel('tid [sekund]')
ylabel('areal A')
```

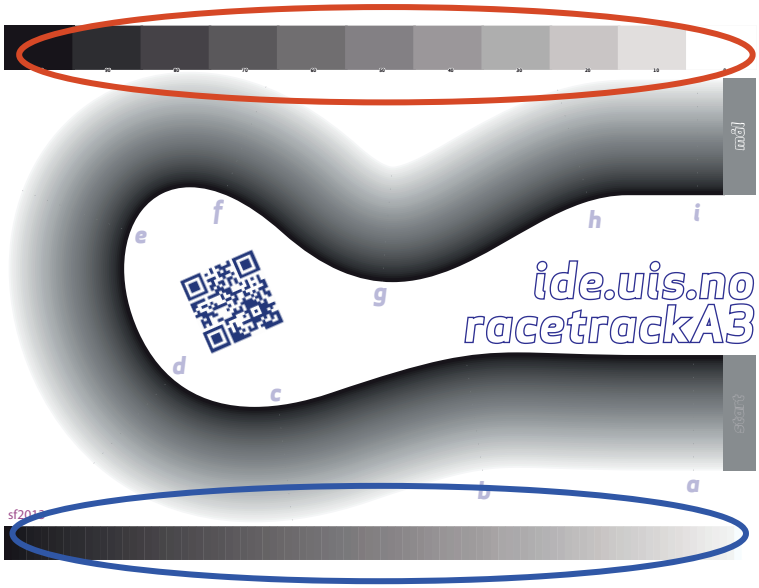
Metoden over kalles Eulers forovermetode. Det finnes andre mer nøyaktige metoder, f.eks. trapes-metoden, som du kan undersøke selv og benytte deg av i f.eks. den kreative delen av prosjektet (sjekk wikipedia).

3.1.4 Verifisering av koden for numerisk integrasjon

Før du benytter integrasjonsrutinen i applikasjoner, er det viktig å få verifisert at koden er riktig implementert. Denne verifiseringen skal inkluderes og dokumenteres i rapporten. For å verifisere skal du utføre den simpleste integrasjon som finnes, nemlig integralet fra 0 til t av en konstant a som:

$$A(t) = \int_0^t a \, dt = a \cdot t \tag{3.5}$$

Siden x-aksen er tidsaksen, blir dette integralet en rett linje med stigningstall a. For å få testet dette i praksis med Legoutstyret skal du lage en konstant verdi a med lyssensoren (signalet vil ikke være helt konstant p.g.a. målestøy) ved å bruke skalaen på banearket som tydelig viser de forskjellige gråtonene (indikert med rød ring øverst i figur 3.6).



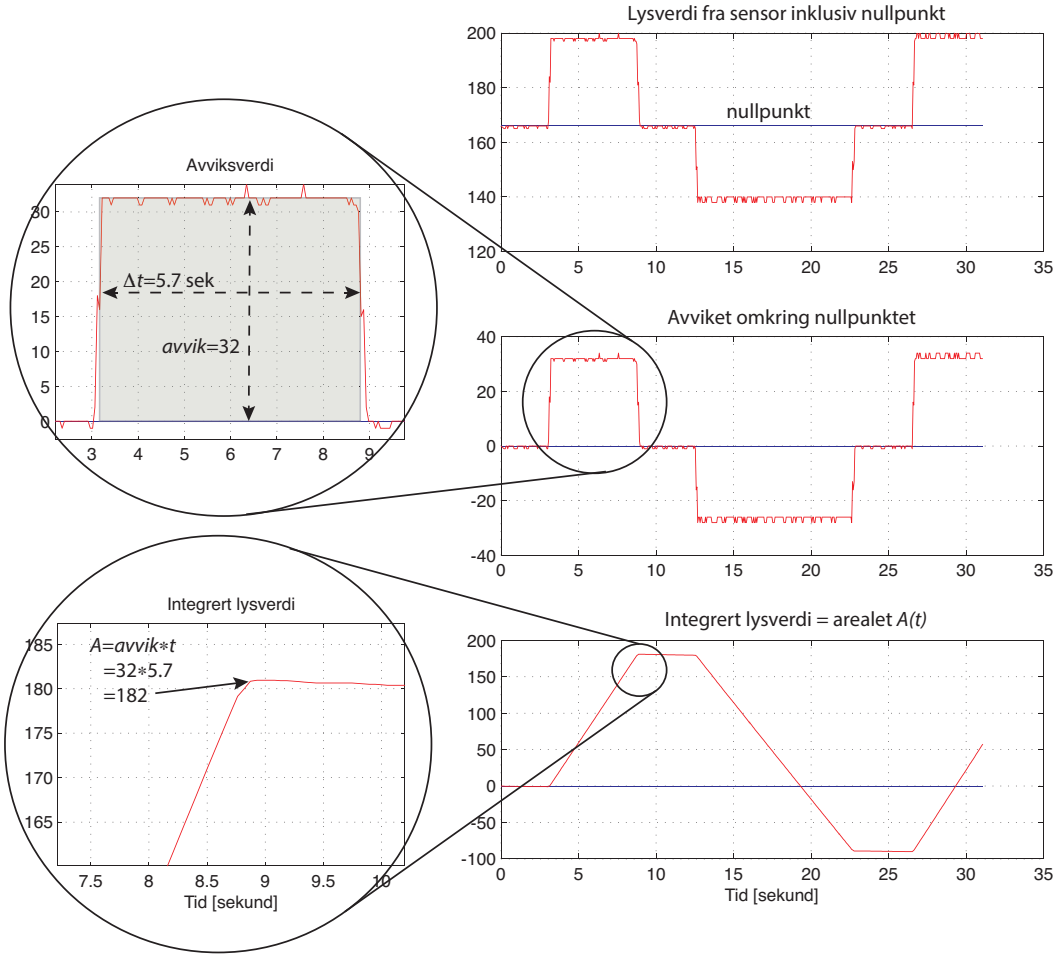
Figur 3.6: Bane for kjøring av robot inklusiv gråtoner for bruk til verifisering av integrasjon og derivasjon. Området markert med rød ring kan brukes ved integrasjon, og området markert med blå ring til derivasjon.

Husk at du i denne testen ikke skal integrerer absoluttverdien (slik som i kjøring langs banen), men de faktiske positive/negative verdiene av a.

Gjør følgende:

- Ta løs lyssensoren og benytt en av de lange ledningene mellom lyssensor og NXT og plassér lyssensoren helt ned på arket på ett av feltene i midten (det røde lyset på lyssensoren skal være aktivert)
- Start programmet
- Flytt lyssensoren to gråtoner til høyre, vent i noen sekunder.
- Flytt lyssensoren tilbake, vent noen sekunder.
- Flytt lyssensoren to gråtoner til venstre, vent i noen sekunder.
- Stopp programmet.
- Basert på de resultatene dere får, skal dere lage en lignende figur som figur 3.7 som dokumenterer at integrasjonen stemmer. Denne figuren skal inn i rapporten. Indiker også hvordan den integrerte verdien er arealet under avviket a under kurven multiplisert med tiden, $A(t) = a \cdot t$

Legg merke til at verdien på arealet A holder seg på et konstant nivå selv om avviket a som du integrerer settes tilbake til null. Basert på denne observasjonen kan vi betrakte integralet som et minne-element. *Nåverdien* av integralet (i dette tilfelle arealet A ved tidspunkt t) er avhengig av verdiene av *alle historiske* avvik og ikke bare nåverdien av avviket. Den egenskapen at integralet (arealet $A(t)$) kan ha en verdi forskjellig fra null selv om det som integreres (avviket a) er null (som ved $t = 10$ sekund), benyttes i industrielle regulatorer. Dette står det litt om i kapittel 3.4 i dette dokumentet, og elektrostudentene får lære mer om dette i faget Reguleringsteknikk i 5 semester.



Figur 3.7: Øverste figur viser nullpunkt og avleste lysverdier fra lyssensor. Midterste figur viser avviket omkring nullpunktet. Nederste figur viser at integrasjonsrutinen virker etter hensikten.

Under er det gitt noen tilleggsspørsmål/oppgaver som dere kan gjøre og inkludere i rapporten.

- Beveg lyssensoren på en slik måte at du får en kvadratisk kurve for arealet, f.eks. $A(t) = \frac{1}{2}a \cdot t^2$?
- Hva får du dersom du beveger sensoren i en sirkel på selve banen slik at profilen ligner en sinuskurve? Hint: Hva er uttrykket for $\int sin(t)$?

- Når du har verifisert integrasjonen, kan du lage en funksjon av rutinen i Matlab som du kan kalle på i andre program.

Rutinen for numerisk integrasjon skal du benytte for å beregne en tallverdi som sier noe om hvor bra Lego-roboten klarer å følge baneprofilen når du kjører den manuelt med joysticken. I tillegg kan du velge å benytte dette i automatisk kjøring av roboten med PID-regulator, se siste kapittel.

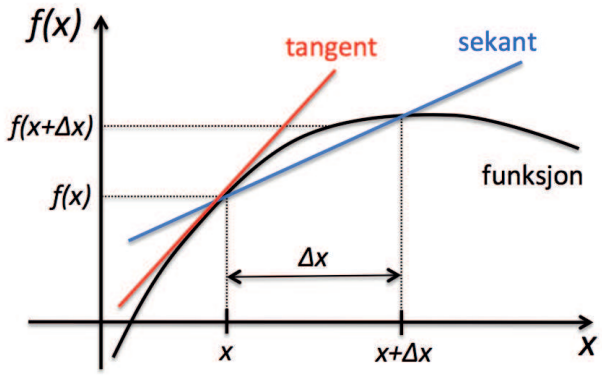
3.2 Numerisk derivasjon

3.2.1 Fra matematikkteori: Derivasjon av en kontinuerlig funksjon

Fra matematikken vet vi at den deriverte er definert som

$$f'(x) = \lim_{\Delta x \to 0} \frac{f(x + \Delta x) - f(x)}{\Delta x} \tag{3.6}$$

hvor den deriverte $f'(x)$ av og til skrives som $\dot{f}(x)$ eller $\frac{df(x)}{dx}$. Ut fra figur 3.8 tilsvarer dette den røde tangenten.

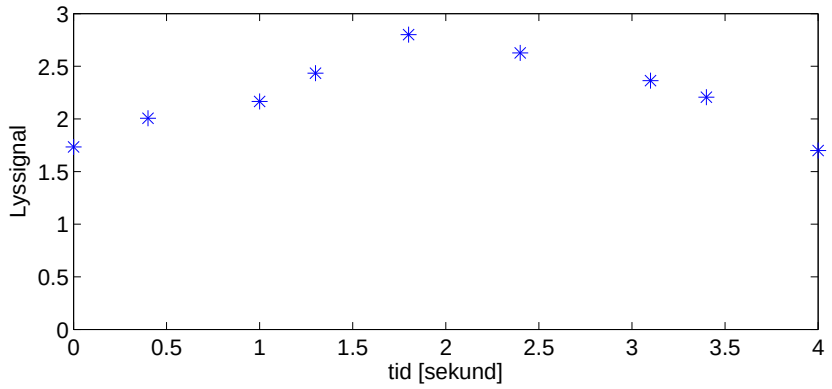


Figur 3.8: En funksjon $f(x)$ med tangenten. Sekanten er den tilnærmede deriverte.

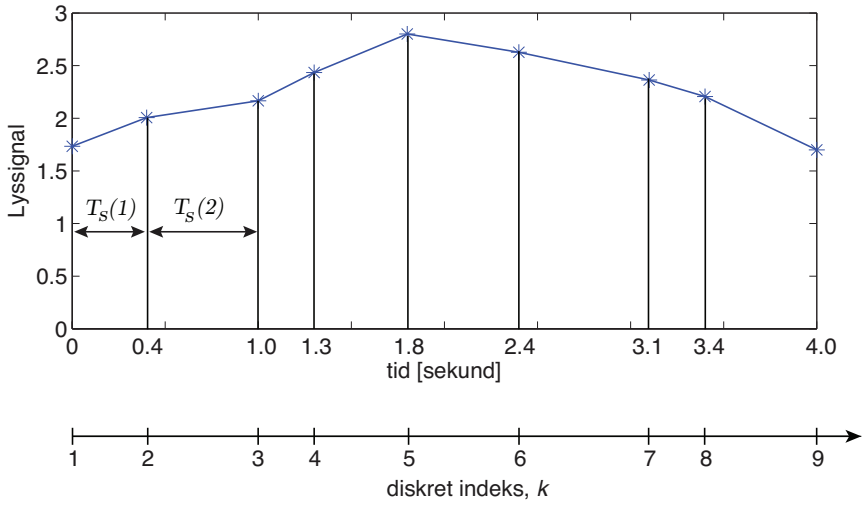
Tallverdien for den deriverte sier med andre noe om hvor fort ting endrer seg, og dette tilsvarer stigningen på tangenten. Dersom vi ikke lar $\Delta x \to 0$, men heller lar Δx ha en verdi, kan vi beregne en tilnærmet verdi for den deriverte, nemlig sekanten. Dess større verdi Δx har, dess dårligere tilnærming får vi for den deriverte.

3.2.2 Fra anvendelse: Derivasjon av en tidsdiskret funksjon

For å illustrere konseptet med numerisk derivasjon, benytter vi samme datasett som for numerisk integrasjon, dvs. figur 3.2, og denne er gjengitt under



Ved å la Δx tilsvare samplingstiden T_s, kan vi finne sekanten mellom to etterfølgende diskret måleverdier ved å tegne en strek mellom punktene, se figur 3.9 (dette er egentlig samme figur som figur 3.3).



Figur 3.9: Logget lyssignal med sekantene inntegnet. Dersom sekanten er stigende eller synkende, tilsvarer dette henholdsvis positiv og negativ verdi for den deriverte.

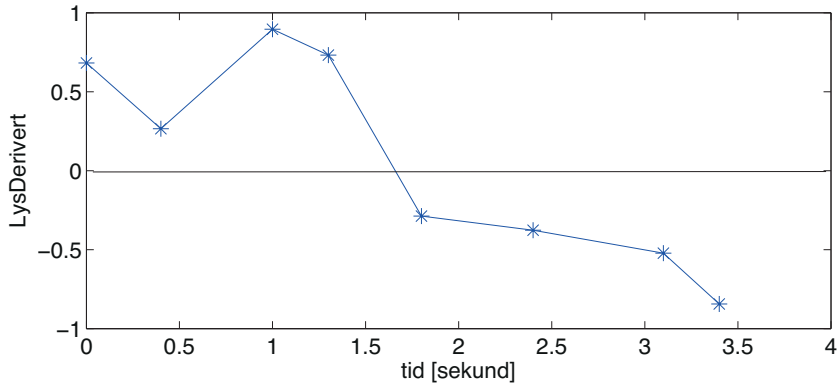
Sekanten, som vi kan kalle $LysDerivert$, i det diskret tidspunktet k blir da:

$$LysDerivert(k) = \frac{Lysverdi(k+1) - Lysverdi(k)}{T_s(k)} \tag{3.7}$$

hvor $T_s(k) = tid(k+1) - t(k)$. Som du ser, er vi i tidspunkt k avhengig av fremtidig og ukjent informasjon, dvs. $Lysverdi(k+1)$ og $tid(k+1)$, for å beregne den deriverte. Vi skifter derfor hele beregningen ett diskret tidspunkt:

$$LysDerivert(k-1) = \frac{Lysverdi(k) - Lysverdi(k-1)}{T_s(k-1)} \tag{3.8}$$

For dataene i figur 3.9 blir *stigningen/synkingen* til hver sekant et estimat av den deriverte, og verdien av dette estimatet vist i figur 3.10. Husk at positve verdier tilstaver stigning og at negative verdier tilsvarer reduksjon (synking).



Figur 3.10: Numerisk derivasjon beregnet ut fra om sekanten enten stiger (positiv derivert) eller synker (negativ derivert).

3.2.3 Pseudokode for anvendelse mot NXT

Pseudokoden for å beregne den numeriske deriverte iterativt i sann tid er vist under:

```
k=1;                              % diskret tellevariabel
Lys(k) = GetLight;                % få tak i ny måling
Tid(k) = nå;                      % ny tidsverdi
k=k+1;
while true
   Lys(k) = GetLight;             % få tak i ny måling
   Tid(k) = nå;                   % ny tidsverdi
   Ts(k-1) = Tid(k)-Tid(k-1);     % beregn tidsskritt
   LysDerivert(k-1) = (Lys(k)-Lys(k-1))/Ts(k-1); % numerisk derivering
   k=k+1;                         % oppdaterer tellevariabel
end
figure
plot(Tid(1:end-1),LysDerivert)  % tidsvektor er 1 lenger enn LysDerivert
xlabel('tid [sekund]')
```

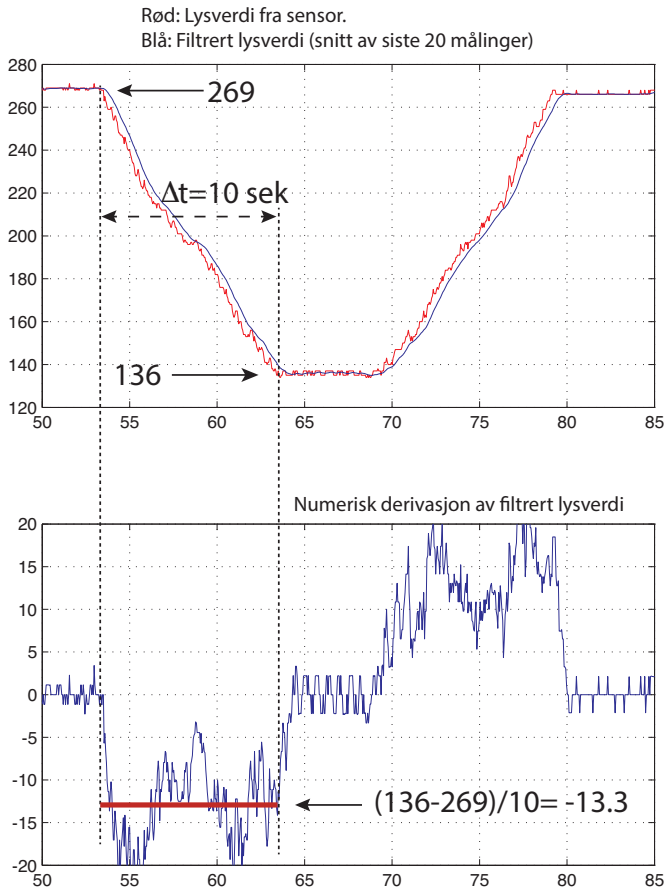
3.2.4 Verifisering av koden for numerisk derivasjon

Før du benytter derivasjonsrutinen i applikasjoner, er det viktig å få verifisert at koden er riktig implementert. Denne verifiseringen skal inkluderes og dokumenteres i rapporten. For å verifisere koden, skal dere utføre den simpleste derivasjonen som finnes, nemlig å derivere en lineært stigende eller fallende kurve $a{\cdot}t$

$$\frac{d}{dt}(a{\cdot}t) = a \tag{3.9}$$

For å lage en slik stigende/fallende linje som funksjon av tiden kan dere gjøre følgende:

- Plasser lyssensoren helt ned til feltet indikert med en blå ring i figur 3.6 og beveg den med konstant hastighet bortover feltet og tilbake igjen. Lyssignalet vil på denne måten bli en lineært stigende og synkende kurve som funksjon av tiden. Det er mulig du vil oppleve at for å få til en lineær kurve må du bevege sensoren i økende eller synkende hastighet. Dette fordi avlesingen av gråskalaene ikke er lineær.
- Som derivasjonsresultat får dere (en noenlunde) konstant verdi som funksjon av tiden t som representerer hvor mange enheter lysmålingen endres per tid.



Figur 3.11: Øverste figur viser lysverdier og filtrert lysverdi fra lyssensor. Nederste figur viser at derivasjonsrutinen virker etter hensikten.

- For at deriveringen skal gi noenlunde fornuftige verdier, er dere nødt å filtrere lyssignalet før dere deriverer det, se kapittel 3.3. I figur 3.11 ble det brukt et snitt av de siste 20 måligene hvor hver måling er vektet med verdien 0.05.

Under er det gitt noen tilleggsspørsmål/oppgaver som dere kan gjøre og inkludere i rapporten.

- Sammenlign det opprinnelige målesignalet med det signalet du ender opp med dersom du først integrerer og deretter deriverer det opprinnelige målesignalet.
- Beveg lyssensoren på en slik måte at du får en lineær stigende/fallende kurve etter derivering.

- Hva får du dersom du beveger sensoren i en sirkel på selve banen slik at profilen ligner en sinuskurve? Hint: Hva er uttrykket for $\frac{d}{dt}sin(t)$?
- Når du har verifisert derivasjonen, kan du lage en funksjon av rutinen som du kan kalle på i andre program.

Rutinen for numerisk derivasjon skal du implementere i programmet for å beregne hvilken vei roboten svinger når du kjører den langs banen (ut mot høyre eller ut mot venstre eller rett frem).

3.3 Filtrering

Siden signalene fra noen av sensorene på NXT'en er støybefengte, kan det være lurt å glatte (filtrere) signalet. Det finnes veldig mange måter å filtrere på, men her skal vi kort vise noen veldig enkle metoder.

Også her kan dere med fordel lage generelle funksjoner av filterne.

I rapporten skal dere vise at dere har jobbet med filtrering og sammenligne resultatene fra forskjellige filtre.

3.3.1 Glatting av måleverdier (FIR-filter)

Den mest intuitive måten å filtrere på er å bruke en vektet sum av et visst antall av tidligere måleverdier. Slike glattefiltre kalles ofte FIR-filter som står for Finite Impulse Response filter. Dette lærer elektrostudentene mer om i faget Signaler og systemer i 4 semester.

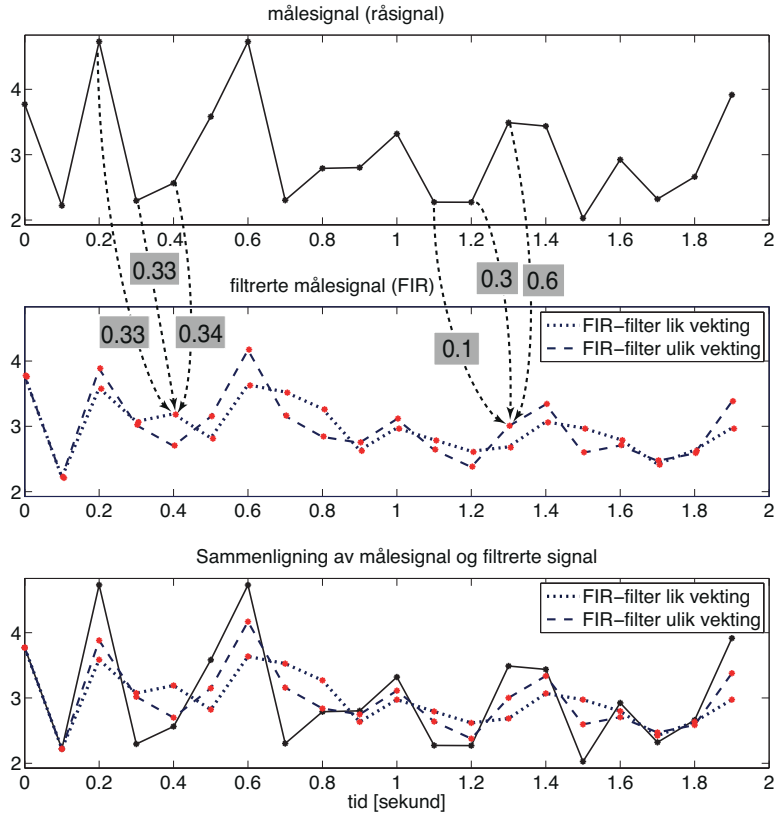
Som et eksempel kan det filtrerte signalet i tidspunkt k, dvs. $x_{filter}(k)$, beregnes ut fra de tre siste verdiene av målesignalet $x(k)$, dvs. $x(k)$, $x(k-1)$ og $x(k-2)$, som vist i ligning (3.10)

$$x_{filter}(k) = 0.34 \cdot x(k) + 0.33 \cdot x(k-1) + 0.33 \cdot x(k-2) \tag{3.10}$$

Måten disse tre verdiene vektes på kan variere. I ligning (3.10) vektes alle verdiene likt, mens i ligning (3.11) vektes siste måling mer enn forrige målinger (noe som kanskje er mer naturlig å gjøre)

$$x_{filter}(k) = 0.6 \cdot x(k) + 0.3 \cdot x(k-1) + 0.1 \cdot x(k-2) \tag{3.11}$$

En sammenligning av disse er vist i figur 3.12, hvor det opprinnelige målesignalet er vist øverst, mens de filtrerte signalene fra ligningene (3.10) og (3.11) er vist i midten. Nederst er alle kurvene samlet slik at det er lett å sammenligne resultatet.



Figur 3.12: Sammenligning av FIR-filter med like vekt på 3 siste målinger og FIR-filter med mest vekt på siste målinger.

3.3.2 Rekursiv filtrering (IIR-filter)

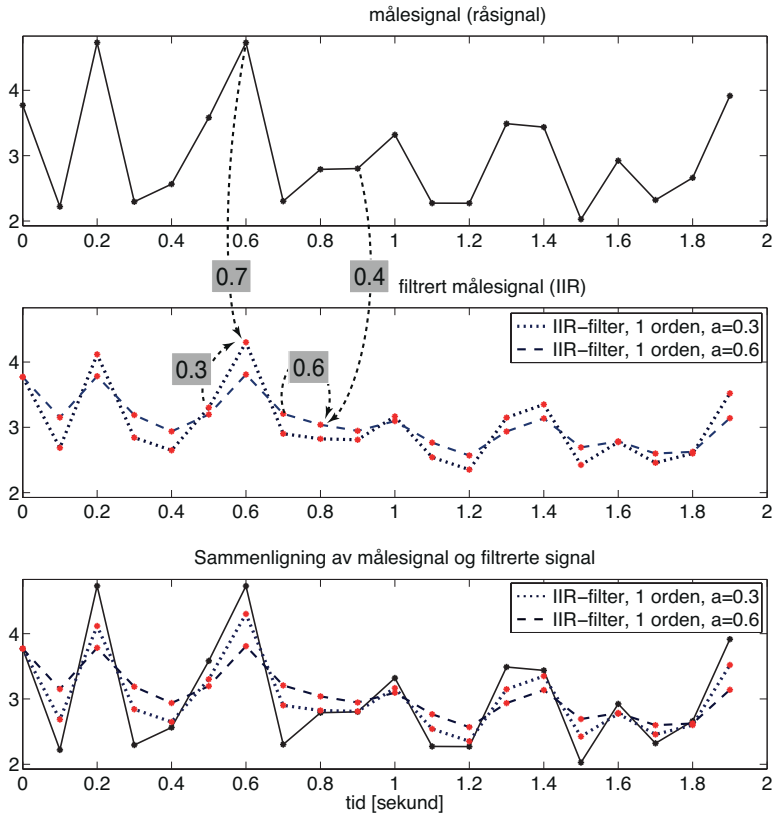
En alternativ måte å filtrere på er å bruke tidligere *filtrerte* verdier sammen med (tidligere) verdier av målesignalet til å beregne den nye filtrerte verdien. Et eksempel er vist i ligning (3.12).

$$x_{filter}(k) = 0.7 \cdot x(k) + 0.3 \cdot x_{filter}(k-1) \tag{3.12}$$

For å få mer glatting, kan vi redusere vekten på siste måling og øke vekten på den forrige filtrerte målingen som i ligning (3.13).

$$x_{filter}(k) = 0.4 \cdot x(k) + 0.6 \cdot x_{filter}(k-1) \tag{3.13}$$

Denne metoden kalles ofte rekursive (repeterende) filtre, eller IIR-filter, som står for Infinite Impulse Response filter. Fordelen med denne filtreringsmåten er generelt at antall filterparametre blir redusert for å oppnå (tilnærmet) samme filtrering som ved FIR-filter, se figur 3.13.



Figur 3.13: Øverst vises målesignalet, og nederst vises to eksempler på IIR-filtrert signal.

Det som er viktig å huske på dersom du velger å bruke flere eller færre målinger, eller andre tallverdier enn de som f.eks. er gitt i ligningene (3.10) og (3.12), er at summen av alle vektene må være 1[2].

[2]Hvorfor må det være slik? Hva skjer dersom summen ikke er 1? Prøv det ut.

3.4 Introduksjon til reguleringsteknikk

3.4.1 PID-regulator

For de av dere som ønsker å teste ut automatisk kjøring, kan PID-regulering være et alternativ, hvor P står for proposjonal, I står for integral og D står for derivat. Det betyr at dere kan benytte rutinene for integrasjon og derivasjon om igjen i automatisk kjøring av roboten langs banen. Selve ligningen for en PID-regulator er

$$u(t) = \underbrace{K_p \cdot e(t)}_{\text{P}} + \underbrace{\frac{K_p}{T_i} \int_0^t e(\tau) d\tau}_{\text{I}} + \underbrace{K_p \cdot T_d \cdot \frac{d}{dt} e(t)}_{\text{D}} \tag{3.14}$$

hvor $e(t)$ er avviket mellom ønsket verdi og målt verdi, K_p er forsterkning, T_i kalles integraltid og T_d kalles derivattid. I noen tekstbøker og på nettet vil dere kunne finne følgende alternative ligning for regulatoren

$$u(t) = K_P \cdot e(t) + K_I \int_0^t e(\tau) d\tau + K_D \cdot \frac{d}{dt} e(t) \tag{3.15}$$

3.4.2 Integratorbegrensing

Ved kjøring langs banen vil dere oppleve at etter den lange første svingen kjører roboten ut av banen. Dette skyldes ofte at I-leddet har integrert seg opp til en stor verdi (plott I-leddets verdi mens du kjører). Ved å begrense hvor stort dette leddet kan bli, vil dere få roboten til å kjøre mer stabilt. Integratorbegrensing kalles på engelsk for anti windup.

For å finne ut hvilken av P, I og D delene i ligning 3.14 som bidrar mest til fremdriften av roboten, kan det være lurt å plotte disse som funksjon av tid mens du kjører.