ELE130 Anvendt fysikk og matematikk i robotprogrammering

Øving 2

Introduksjon til simulering i Simulink[1]

Levert av ....

[1] Tilhørende filer: `oving2_skallfiler.zip` og LaTeX-mal i `oving2.zip`

Om øvingen og innleveringen

- For å bli kjent med Simulink på en effektiv måte, er alle oppgavene i denne øvingen svært nyttige, og de vil fungere som et oppslagsverk for deg.
- For å tilrettelegge for at du skal gjøre flest mulig oppgaver, så skal du laste ned og pakke opp `oving2_skallfiler.zip`. Denne inneholder følgende to filer:
  - Simulink-skallfilen `oving2_skallfil.slx` hvor relaterte oppgaver kan implementeres i egne subsystem.
  - skallfil-skriptet `oving2_i_o_p_skallfil.m` som du må kjøre før du kjører `oving2_skallfil.slx` for oppgavene 2i), 2o) og 2p).

  Simulink-skallfilen er spesifisert med *Eulers forovermetode* som integrasjonsmetode og tidsskritt T_s lik 0.01 sekund.
- Resultater i scope kan du overføre til en Matlab-figur med menyvalget `File -> Print to Figure`, som du igjen kan bruke `SaveMyFigure` på.
- Dersom du ikke er en erfaren programmerer, anbefales det at du prøver deg på alle oppgavene slik at du får mengdetrening.
- For å få øvingen godkjent må du gjøre minst følgende oppgaver:
  a), e), f), g), i), k), o), p)
- Oppgavene b), c), d), h), j), l), m), n) er også lærerike og nyttige, men frivillige. Dette er markert i toppteksten.
- For å ta eksamen i emnet ELE130 så må denne øving være godkjent. Husk at øvingene må leveres individuelt.
- Basis for denne øvingen er kapittel 4 og 5 i kompendiet.
- På samme måte som i øving 1 finner du en LaTeX-mal i filen `oving2.zip`.

Så langt det er mulig skal du benytte blokker fra `Eget bibliotek` slik at nyttig informasjon om blokkene vises i blokkdiagrammet. Du skal benytte `Scope`-blokkene som heter `Scope, tynne linjer` eller `Scope, tykke linjer` fra `Eget bibliotek` til å vise resultatene i.

Øvingen og skallfilen er organisert i følgende deler/subsystem:

- "Om *Integrator*-blokken", oppgave 2a) - 2e)
- "Om *Sine Wave*-blokken", oppgave 2f) - 2h)
- "Om *Lookup Table*-blokken", oppgave 2i) - 2l)
- "Litt diverse", oppgave 2m) - 2p)

Forskjellig LaTeX-kode

- Resultatet fra scopet er vist i figur 1. Husk å oppdatere referansen til figuren for hver ny figur.

Figur 1: Resultat av blokkskjemaet i modellen.

- Svaret er gitt i ligning (1).
$$y = 4 \tag{1}$$

- Eksempel på bruk av `align` er vist i ligning (2). For å bruke kommandoen `\uuline` må du bruke pakken `ulem` som `\usepackage{ulem}`.

$$\begin{aligned}
A &= \int_0^1 2\cdot \sin(5\cdot t)dt \\
&= -\frac{2}{5}\cdot \cos(5\cdot t)\Big|_0^1 \\
&= -\frac{2}{5}\cdot \cos(5\cdot 1) - \left(-\frac{2}{5}\cos(0)\right) \\
&= \underline{\underline{0.287}} \qquad (2)
\end{aligned}$$

- Du finner også mye LaTeX-kode i hver av .tex-filene til delxoppgavene.

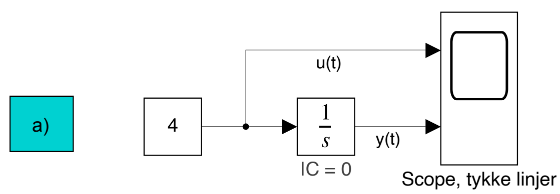
Om *Integrator*-blokken

Alle modellene i denne delen av øvingen skal implementeres i subsystemet `Om Integrator-blokken, oppgave 2a)-2e)` i skallfilen `oving2_skallfil.slx`, og alle modellene skal benytte integratorblokken som utfører numeriske beregninger som tilsvarer følgende operasjon

$$y(t) = \int_0^t u(\tau)d\tau + y(0) \tag{3}$$

hvor $y(0)$ er initialverdien.

a) I denne deloppgaven er $u(t)$ en konstant, og du skal relatere simuleringsresultatet til integralregning fra matematikken. Implementer modellen vist under.

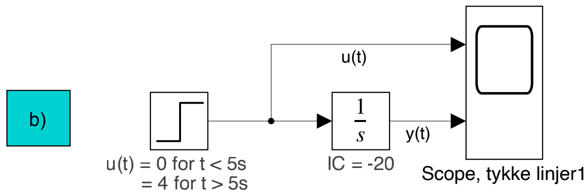


Utgangen fra `Constant`-blokken er $u(t)$ og utgangen fra `Integrator`-blokken er $y(t)$. La initialverdien til integratorblokken være $y(0)$=0. La simuleringstiden være 25 sekund.

- Bruk ligning (3) og la $u(t)$=4. Hva blir det matematiske uttrykket for $y(t)$ når $y(0)$=0?
- Beregn verdien av $y(t)$ ved tidspunkt t=25 sekund.
- Simuler modellen og ta med resultatet i besvarelsen din. Viser simuleringsresultatet det samme som svaret du fant i forrige spørsmål?
  Som hjelpemiddel kan du gjerne bruke avlesningsverktøyet som du finner under `Tools -> Measurements -> Cursor Measurements`.
  Ved bruk av menyvalget `File -> Print to Figure` fra scopet, så blir ikke avlesningsdelen med i Matlab-figuren.
- Endre initialverdien til $y(0)= -20$ og simuler. Hvordan endres det matematiske uttrykket $y(t)$? Vis at simuleringsresultatet overstemmer med det matematiske uttrykket.

Svar

b) Lag følgende modellen hvor `Constant`-blokken i oppgave a) er byttet ut med en `Step`-blokk som går fra 0 til 4 ved t=5 som vist under



Uttrykket for utgangen $u(t)$ er

$$u(t) = \begin{cases} 0 & \text{for } t < 5 \\ 4 & \text{for } t \geq 5 \end{cases} \tag{4}$$

La simuleringstiden fortsatt være 25 sekund.

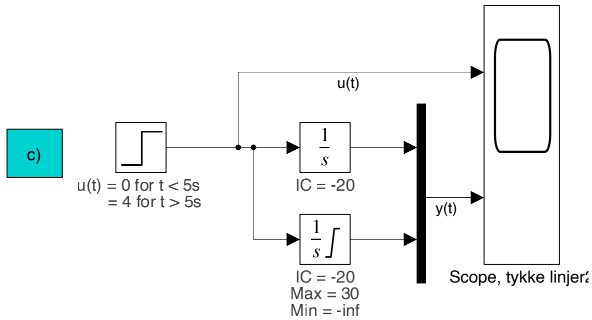
- Benytt initialverdi $y(0) = -20$, og simuler modellen. Skriv ned det matematiske uttrykket for responsen $y(t)$ formulert som

$$y(t) = \begin{cases} \ldots & \text{for } t < 5 \\ \ldots & \text{for } t \geq 5 \end{cases} \tag{5}$$

- Er simuleringsresultatet ved t=25 sekund identisk med tilsvarende verdi funnet fra $y(t{=}25)$ fra ligning (5).

Svar

c) Lag følgende modell som er en utvidelse av modellen fra oppgave b).

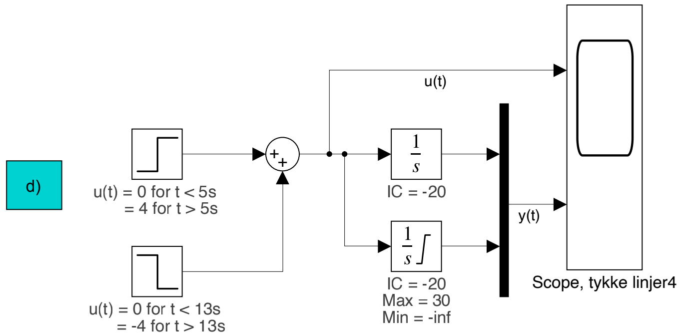


hvor det benyttes en `Integrator med begrensing`-blokk i parallell med den første integrator-blokken hvor den nye blokken også har initialverdi $y(0) = -20$, men hvor utgangen er begrenset oppad til y_{max}=30. La simuleringstiden fortsatt være 25 sekund.

- Simuler modell og vis at utgangen fra den nye integratoren stoppes ved y_{max}.
- Hvordan ville du prinsipielt implementert integratorbegrensing i kode?

Svar

d) Lag følgende modell, som er en utvidelser av modellen i oppgave c)

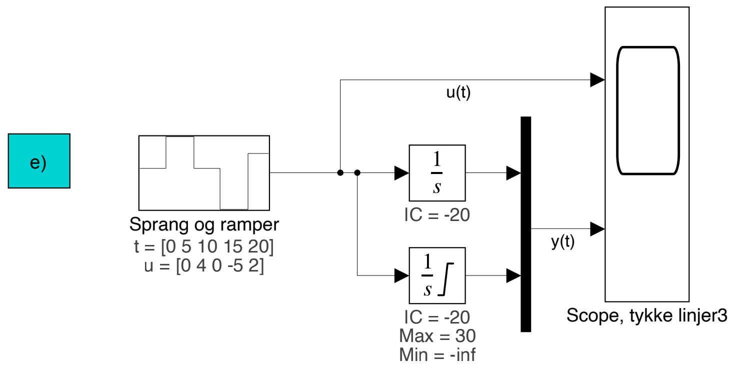


hvor signalet $u(t)$ fortsatt går fra 0 til 4 ved t=5 sekund, men at det går tilbake til 0 ved t=13 sekund. La simuleringstiden fortsatt være 25 sekund. Simuler modellen.

- Estimer arealet under $u(t)$ og vis at det tilsvarer endringen i $y(t)$.
- Hvorfor beholder $y(t)$ sin verdi når $u(t)$ går tilbake til 0?

Svar

e) Lag følgende modellen



Spesifiser `Sprang/ramper`-blokken med følgende verdier

- `Tidspunkter` som `[0 5 10 15 20]`
- `Utgangsverdier` som `[0 4 0 -5 2]`

som betyr at

$$u(t) = \begin{cases} 0 & \text{for } 0 \le t < 5 \\ 4 & \text{for } 5 \le t < 10 \\ 0 & \text{for } 10 \le t < 15 \\ -5 & \text{for } 15 \le t < 20 \\ 2 & \text{for } t \ge 20 \end{cases} \tag{6}$$

La simuleringstiden fortsatt være 25 sekund.

- Simuler modellen og ta med resultatet i besvarelsen din. Estimer arealet under $u(t)$ (både positiv og negativt) og vis at summen tilsvarer endringen i $y(t)$.
- Endre signalet $u(t)$ fra sprang til ramper ved å bytte interpolasjonsmetode fra `Trappetrinn` til `Linear` i rullegardinmenyen inne i `Sprang/ramper`-blokken. Simuler modellen på ny. Gjør en grovt estimat av arealet $u(t)$ og vis at dette tilsvarer endringen i $y(t)$.

Vær klar over at du kan skrive både matematiske uttrykk, kommentarer, funksjoner og kode i alle feltene i alle Simulink-blokkene. Under vises noen eksempler på hva du kan skrive i feltet `Utgangsverdier`:

- `[0 4 0 -5 2]*0.1`
- `[0 -3 2 0 1]    %[0 4 0 -5 2]`
- `randn(1,5)`
- `1:0.1:1.4`

Svar

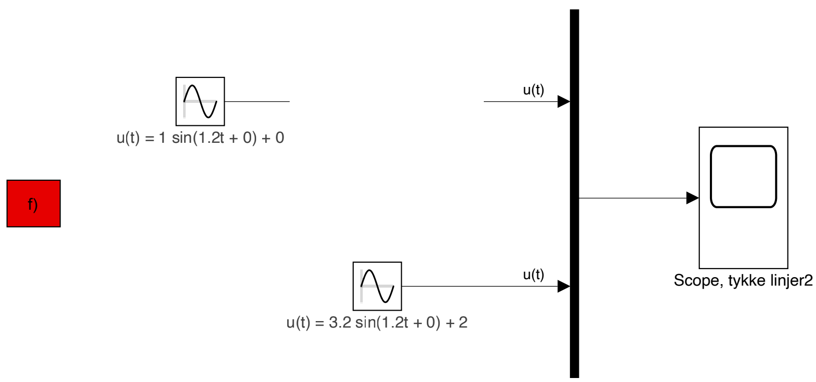
Om *Sine Wave*-blokken

Alle modellene i denne delen av øvingen skal implementeres i subsystemet `Om Sine Wave-blokken, oppgave 2f)-2h)` i skallfilen `oving2_skallfil.slx`.

f) I denne oppgaven skal du implementere følgende signal på 2 måter.

$$u(t) = 3.2\sin(1.2t) + 2 \tag{7}$$

Den ene varianten tar utgangspunkt i en `Sine Wave`-blokk som kun gir ut $\boxed{\sin(1.2t)}$. Det betyr at du må bygge opp signalet ved å bruke et utvalg av andre blokker, som f.eks. `Gain`, `Constant`, `Product` og `Sum`. Det er flere måter å gjøre dette på, og detaljene om dette er skjult i modellen under.



Den andre varianten er en `Sine Wave`-blokk direkte som vist nederst i modellen. La simuleringstiden fortsatt være 25 sekund.

- Bygg opp og simuler modellen og vis at kurvene ligger over hverandre. Endre gjerne den røde kurven til å være stiplet slik at det er lettere å se at de ligger over hverandre.
- Uttrykket for $u(t)$ i ligning (7) kan generelt skrives som

$$u(t) = U \cdot \sin(\omega \cdot t) + u_A \tag{8}$$

Vis hvordan du ved kun hjelp av avlesninger fra kurven scopet kan beregne estimater av følgende størrelser:

- amplituden U
- frekvensen ω
- likevektsverdien u_A

Svar

g) I denne oppgaven skal du implementere følgende 3 signal ved bruk av en `Ramp`-blokk og to `Sine Wave`-blokker.

$$v_1(t) = 3.2\sin(1.2t) + 2 \tag{9}$$

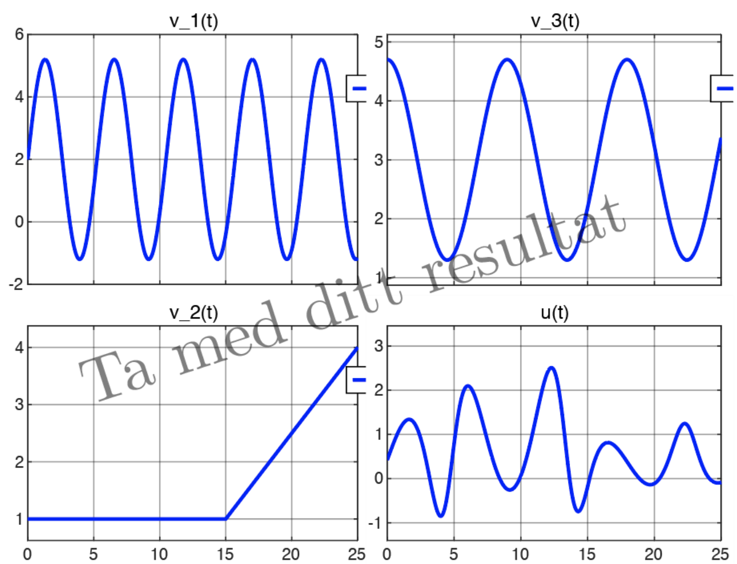
$$v_2(t) = \begin{cases} 1 & \text{for } 0 \leq t < 15 \text{ sekund} \\ 1 + 0.3{\cdot}(t-15) & \text{for } t \geq 15 \text{ sekund} \end{cases} \tag{10}$$

$$v_3(t) = 1.7\cos(0.7t) + 3 \tag{11}$$

Lag deretter et signal $u(t)$ som baserer seg på de 3 signalene på følgende måte.

$$u(t) = \frac{v_1(t)}{v_2(t){\cdot}v_3(t)} \tag{12}$$

Bruk kun én `Produkt`-blokk til å lage $u(t)$. Send alle 4 signalene til et scope, og vis at du får følgende resultat. La simuleringstiden fortsatt være 25 sekund.



Figur 2: Simuleringsresultatet

Velg et vilkårlig tidspunkt etter t=15 sekund og les av en verdi fra $u(t)$. Sett inn samme tidspunkt i ligning (12) innsatt $v_1(t)$, $v_2(t)$ og $v_3(t)$, og vis at resultatet stemmer med simuleringen.

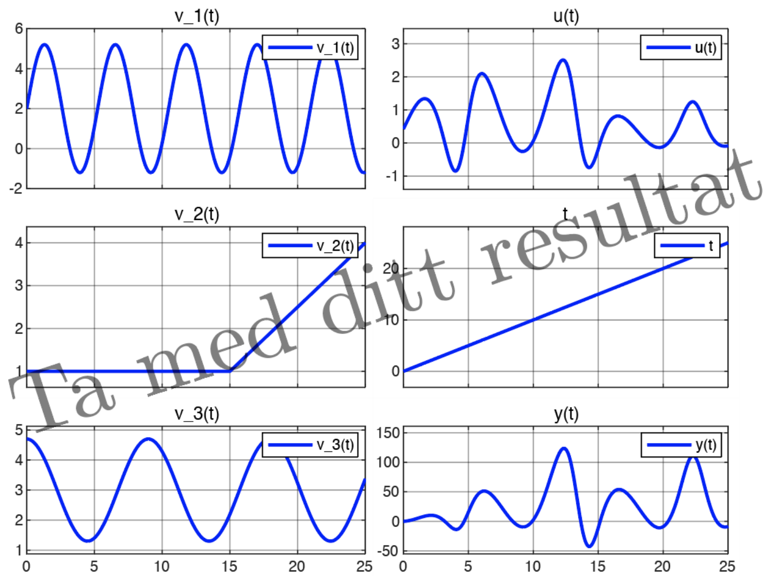
Svar

h) Kopier modellen fra g) og lag et nytt signal som

$$y(t) = 4{\cdot}u(t){\cdot}t \tag{13}$$

ved å bruke en `Gain`-blokk, en `Clock`-blokk og en `Product`-blokk.

Utvid `Scopet` til 6 innganger med 6 delfigurer for å vise $v_1(t)$, $v_2(t)$, $v_3(t)$, $u(t)$, t og $y(t)$, og vis at du får følgende resultat. La simuleringstiden fortsatt være 25 sekund.



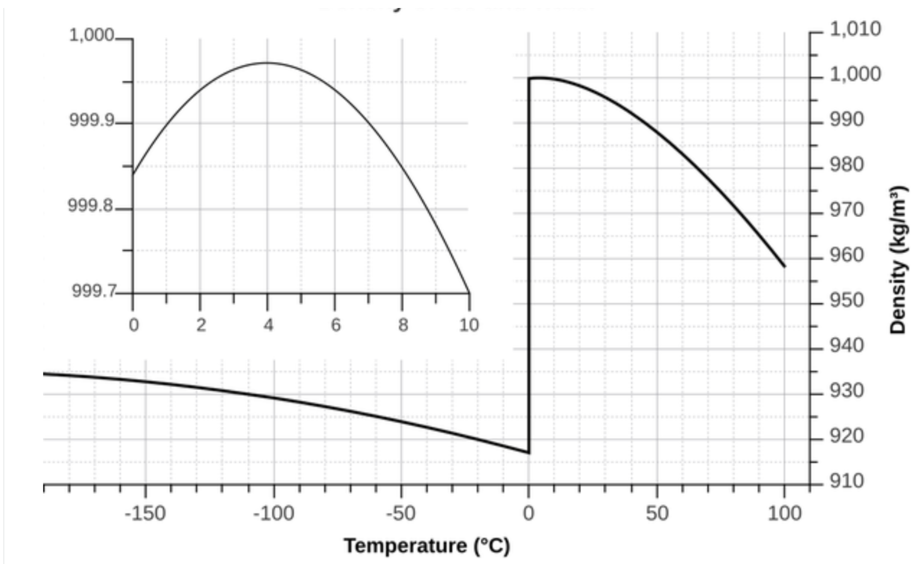
Figur 3: Simuleringsresultatet

Svar

Om *Lookup Table*-blokken

Alle modellene i denne delen av øvingen skal implementeres i subsystemet `Om Lookup Table-blokken, oppgave 2i)-2l)` i skallfilen `oving2_skallfil.slx`.

i) I denne oppgaven skal du begynne på en ny modell. Tettheten til vann som funksjon av temperatur er vist i figuren under



Figur 4: Tettheten vann som funksjon av temperatur. Detaljene mellom 0 og 10 grader er vist i egen delfigur.

Målet med oppgaven er å implementere sammenhengen i figur 4 i en `1-D Lookup table`-blokk som tar tempertur som et inngangssignal og gir ut tetthet som utgangssignal. Denne type blokk brukes for å implementere sammenhenger som ikke kan uttrykkes matematisk med en ligning. Blokken må spesifiseres med to vektorer med tallverdier som representerer kombinasjoner av verdier fra henholdsvis x-aksen og y-aksen. Ved å interpolere mellom punktene får du en kontinuerlig sammenhengen mellom inngangssignalet og utgangssignalet.

For å få med endringene i tetthet rundt 0°C samt detaljene mellom 0 og 10°C vist i delfiguren i figur 4, så tar vi utgangspunkt i følgende vektor med temperaturer fra −150°C til 100°C. Om du vil kan du legge inn flere punkter selv.

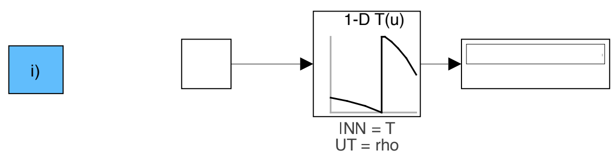
```
T = [-150, -100, -50, -0.001, 0, 2, 4, 6, 8, 10, 30, 50, 70, 100]
```

- Åpne skallfilen `oving2_i_o_p.m` og fullfør den tilhørende vektoren `rho` med tettheter avlest fra figur 4. Dette innebærer at for hver verdi i vektoren `T` ovenfor, så leser du av tilhørende tetthet fra figuren.

I .m-filen lages en figur som viser sammenhengen slik at du får bekreftet at du har avlest riktige verdier, og riktig antall verdier. Bakgrunnen for å bruke en .m-fil først er at det er lettere å oppdage feil der enn i selve `Lookup Table`-blokken.

- I Simulinkmodellen henter du inn en `1-D Lookup table`-blokk hvor du benytter disse to vektorene til å spesifisere blokken. Du kan nå velge å enten
  - kopiere inn vektorene med tallverdier i feltene
  - eller skrive `rho` og `T` i feltene. Dette betinger at du kjører .m-filen først slik av disse variablene er tilgjengelig i `Workspace`.

  Fronten på blokken vil vise sammenhengen mellom temperatur og tetthet som vist under



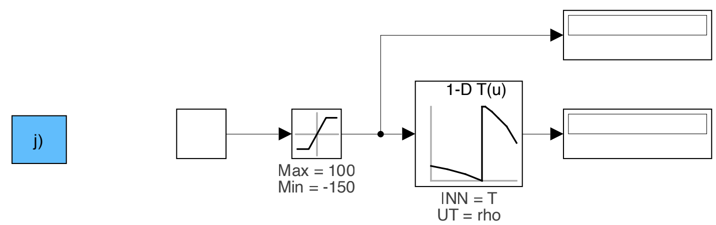
  Inngangen kommer fra en `Constant`-blokk som representerer temperaturen, og utgangen går til en `Display`-blokk som viser tilhørende tetthet. Du kan dobbelklikke på `Display`-blokken og velge `Numeric display format` som `Long`.

- Skrive inn en HELT vilkårlig temperaturverdier (gjerne med masse desimaler) mellom $-150°\text{C}$ til $100°\text{C}$ i `Constant`-blokka. La simuleringstiden fortsatt være 25 sekund. `Display`-blokken viser da en interpolert verdi. Ta med skjermdump av dette i innleveringen.

  Måten blokken interpolerer på er bestemt i fanen `Algorithm` under `Interpolation method`. Defaultverdien er som du ser `Linear point shape`.

Svar

j) Kopier modellen fra i), og skriv inn et tall utenfor område $-150°C$ til $100°C$. Som du ser vil blokka gi en verdi som er lineært interpolert utover det område blokka er spesifisert for. Siden dette ikke er korrekt i forhold til figur 4, skal du nå begrense verdien på det som sendes inn ved å benytte en `Saturation`-blokk som vist under



Spesifiser nedre grense lik -150 og øvre grense lik 100, og plasser blokken som vist over.

Legg gjerne til en ny `Display`-blokk som viser hvilken verdi som sendes inn til `1-D Lookup table`-blokken.

Test ut med å skrive inn f.eks. -200, 200 eller 1000 i `Constant`-blokken.

La simuleringstiden fortsatt være 25 sekund.

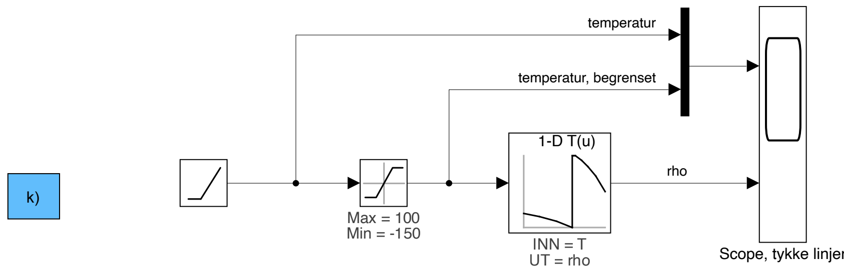
Ta med skjermdump av resultatet.

Svar

k) Kopier modellen fra j), og erstatt $\boxed{\texttt{Constant}}$-blokken med en $\boxed{\texttt{Ramp}}$-blokk. Spesifiser rampen på en slik måte at tempaturen stiger fra −200°C til +200°C i løpet av simuleringstiden på 25 sekund ut fra følgende sammenheng

$$T(t) = -200 + a{\cdot}t \tag{14}$$

Endre videre modellen som vist under, og simuler modellen.

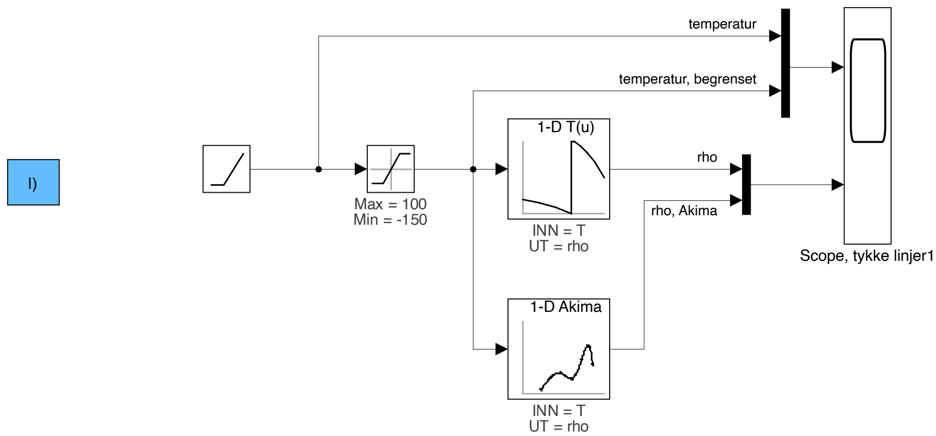


La simuleringstiden fortsatt være 25 sekund.

Forstørr utgangssignalet mellom 12<t<17 sekund og ta med i innleveringen. Som du ser består kurven av lineære elementer som viser den lineære interpolasjonen som blokken utfører.

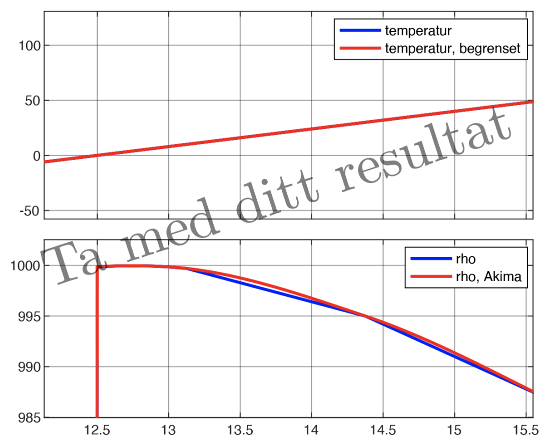
Svar

l) Kopier modellen fra deloppgave k). For å se forskjellen mellom to aktuelle interpolasjonsmetoder, skal du kopiere den allerede eksisterende `1-D Lookup Table`-blokka og legge den i parallell med den første som vist under.



Endre deretter interpolasjonsmetoden i den kopierte blokka til `Akima Spline`, sjekk https://en.wikipedia.org/wiki/Akima_spline. La simuleringstiden fortsatt være 25 sekund.

Simuler modellen på ny, forstørr et område rundt der hvor tettheten er ρ=1000, og vis at du får resultatet vist under.



Hva viser resultatet om forskjellen på interpolasjonsmetodene?

Svar

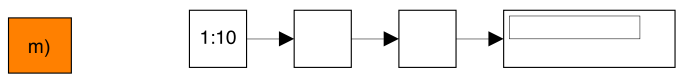
Litt diverse

Alle modellene i denne delen av øvingen skal implementeres i subsystemet `Litt diverse, oppgave 2m)-2p)` i skallfilen `oving2_skallfil.slx`.

m) Implementer en Simulinkmodell for ligning (15)

$$a = \sum_{n=1}^{10} n^2 \tag{15}$$

Ta utgangspunkt i følgende struktur



hvor du ser at i `Constant`-blokken er tallene for n som `1:10` angitt. Bruk ellers en `Math Function`-blokk (velg operasjon fra menyen), en `Sum of elements`-blokk og en `Display`-blokk.

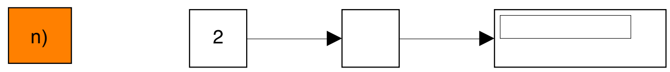
La simuleringstiden fortsatt være 25 sekund.

Svar

n) Implementer en modell som bruker eksponentialfunksjonen

$$e^x \tag{16}$$

Ta utgangspunkt i følgende struktur hvor konstantblokken med verdi 2 innebærer at $x = 2$.

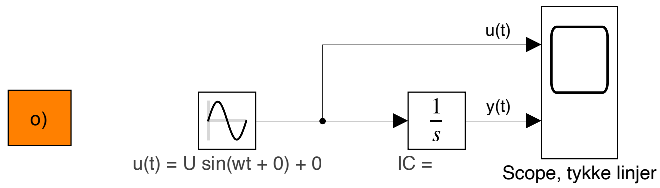


Bruk igjen en `Math Function`-blokk (velg riktig funksjon i menyen) og en `Display`-blokk.

La simuleringstiden fortsatt være 25 sekund.

Svar

o) Kjør først filen `oving2_i_o_p.m`, og lag deretter modellen vist under



Som du ser har vi spesifiser $u(t)$ som

$$u(t) = U \sin(\omega \cdot t) \tag{17}$$

hvor U=0.5 og ω=0.3 rad/s gitt i .m-filen. Ved å integrere $u(t)$ med initialverdi $y(0)$=0 finner vi uttrykket for $y(t)$ som

$$y(t) = -\frac{U}{\omega} \cdot \cos(\omega \cdot t) + \frac{U}{\omega} \tag{18}$$

La simuleringstiden nå være 100 sekund, og ikke 25 sekund som tidligere.

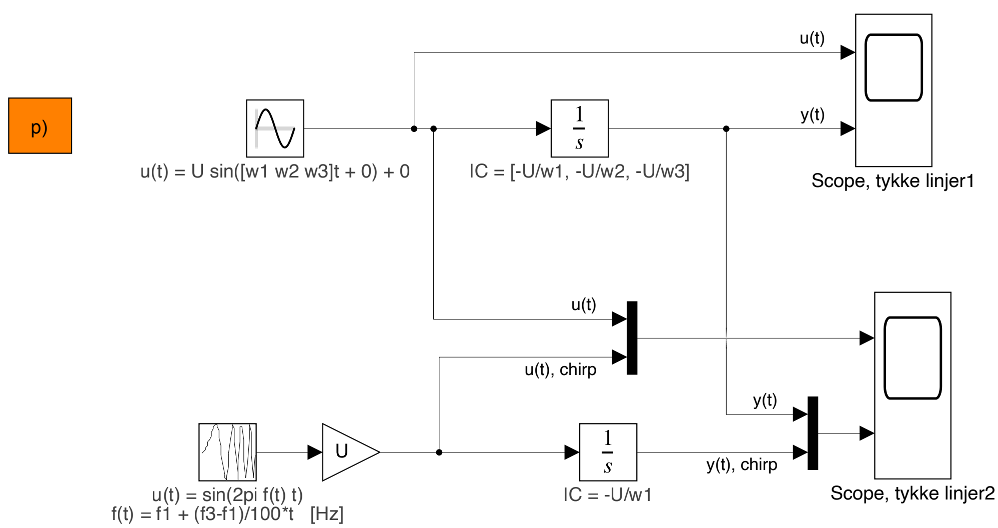
- Sett initialverdien til $\boxed{0}$ i integratoren.
  Simuler modellen og vis at responsen til $y(t)$ alltid er større enn 0.
- Vi ønsker å få $y(t)$-kurven til å svinge omkring y=0, og den eneste måten å få dette til på er å redusere initialverdien $y(0)$ i integratoren. Bestem ny initialverdi som gjør at $y(t)$ svinger omkring y=0, og simuler på ny.

Svar

p) I denne oppgaven skal du integrerer følgende 3 signal.

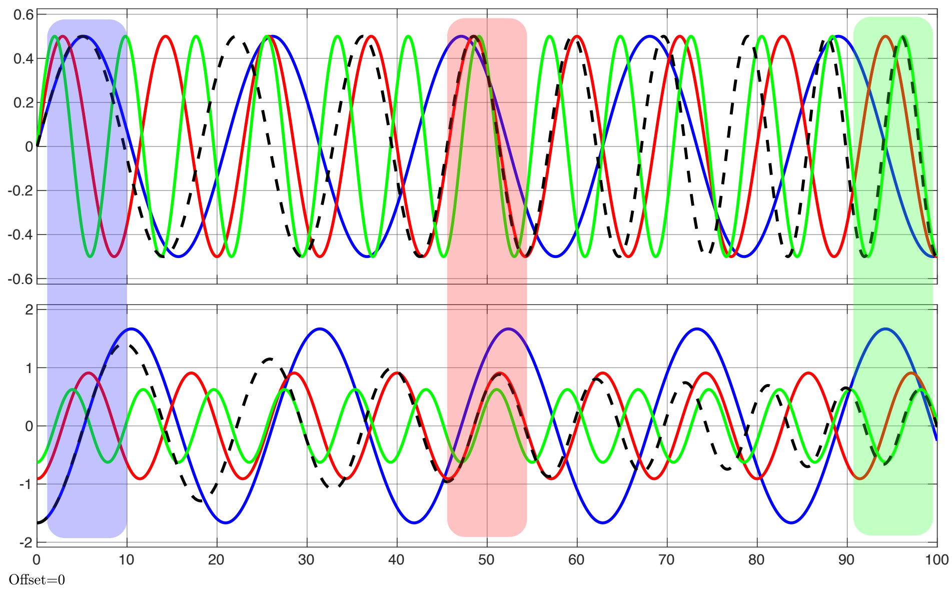
$$u_1(t) = 0.5\sin(0.3t) \tag{19}$$
$$u_2(t) = 0.5\sin(0.55t) \tag{20}$$
$$u_3(t) = 0.5\sin(0.8t) \tag{21}$$

Hensikten med oppgaven er å vise hvordan disse 3 sinussignalene henger sammen med `Chirp` -blokken. Kjør først filen `oving2_i_o_p.m`, og lag deretter modellen vist under



La simuleringstiden nå være 100 sekund.

- I `Sine Wave` -blokken spesifiserer du `Frequency`-feltet som `[w1, w2, w3]`.
- I `Chirp` -blokken spesifiserer du
  - `f1` som `Initial frequency`,
  - `f3` som `Target frequency` og
  - `100` som `Target time` (tilsvarer simuleringstiden).
- Spesifiser initialverdiene som vist i figuren slik at cosinuskurvene svinger omkring y=0.
- Simuler modellen og vis at du får et resultat som ligner på responsen i figur 5, hvor vi har endret den svarte kurven til stiplet slik at det er lettere å identifisere *Chirp*-signalet.



Figur 5: Simuleringsresultat. De fargelagte områdene er gjort i tegneprogram etterpå.

- Som du ser "overlapper" det svart-stiplede chirp-signalet
  - med den blå sinuskurven i det blå feltet i begynnelsen,
  - med den røde sinuskurven i det røde feltet i midten,
  - med den grønne sinuslurven i det grønne feltet kurven i slutten

  og dette gjelder både i den øverste og nederste delfiguren.
- Som du ser så reduseres amplituden Y i integralet $y(t)$ når frekvensen ω øker. Forklar, ut fra et matematisk ståsted, hvorfor dette skjer? I forklaringen din kan du ta utgangspunkt i at uttrykket for $y(t)$ er gitt av følgende sammenheng.

$$
\begin{aligned}
y(t) &= \int_0^t u(\tau) d\tau + y(0) \\
&= \int_0^t U \cdot \sin(\omega \cdot \tau) d\tau + y(0) \\
&= \dots \\
&= \dots
\end{aligned}
$$

Fortsett på denne utledningen og finn det analytiske uttrykket for $y(t)$, og bruk dette til å forklare hvorfor amplituden til $y(t)$ synker med økende frekvens ω.

Svar