NTNU
Det skapande universitetet

Implementering av relévern i ATPDraw

Torstein Stadheim

Master of Science in Electric Power Engineering
Innlevert: Juni 2012
Hovedveiledar: Hans Kristian Høidalen, ELKRAFT

Noregs teknisk-naturvitskaplege universitet
Institutt for elkraftteknikk

Oppgåvetekst

ATP-EMTP er eit program brukt for simulering av elektriske transientar i kraftsystem. ATP-EMTP krev inndata i form av kompliserte tekstfiler. Det er utvikla ein pre-prosessor til dette programmet her ved NTNU som heiter ATPDraw. ATPDraw lettar arbeidet med å setje opp tekstfila som ATP-EMTP nyttar til simulering, ved hjelp av eit grafisk grensesnitt og eit rikt bibliotek av ferdige komponentar. Dette biblioteket inneheld per i dag ikkje relévern. Etterspurnaden frå brukarmassen har vore der, og ei implementering av relévern vil gi ATPDraw auka allsidigheit og nytteverdi.

Oppgåver for implementering av relévern i ATPDraw:

- Litteraturstudie av relévern.
- Gjere seg kjent med MODELS, som er det integrerte programmeringsspråket for programmering av modellar i ATPDraw.
- Gjere seg kjent med signalbehandling, og korleis dette vert gjort i reelle relévern.
- Implementere signalbehandlingsmodellar, herunder RMS og FFT.
- Implementere overstraum-, differensial- og distansevernmodellar.
- Verifisere relévernmodellane ved hjelp av enkle ATP-simuleringar og samanlikne desse med respons frå reelle relévern via COMTRADE-filer.
- Implementere funksjonar for handtering av magnetiseringsstraumar og effektpendling for differensial- og impedansvernmodellar.

Oppstartsdato: 30. januar 2012

Hovedveileder: Professor Hans Kristian Høidalen, NTNU

Samandrag

Rapporten omhandlar utviklinga av relévernmodellar for bruk i ATPDraw. ATPDraw er eit simuleringsprogram for elektriske kretsar, og inneheld per i dag ikkje modellar av relévern.

Fyrste del av rapporten tek for seg grunnleggande teori og verkemåte til dei vanlegaste typane relévern, samt diskré signalbehandling. Relévern omtala er overstraumvern, differensialvern og impedansvern. Av signalbehandling er det sett på omgjering av diskré signal til effektivverdiar og frekvenskomponentar, med amplitude og fase.

I andre del av rapporten er resultata frå utviklingsarbeidet presentert. Kjeldekode for kvar modell er forklart, og flytskjema er vist for dei modellane med omfattande kode. Her er det også forklart kort om korleis modellane kan nyttast i ATPDraw.

Til slutt er nokre av modellane verifisert mot eit fysisk relévern. Relévernet som vart nytta til verifisering var eit impedansvern levert av Siemens, modellnummer 7SA610. På bakgrunn av relévernets funksjonalitet vart kun modellane av impedansvern utan effektpendlingsfunksjon og overstraumvern samanlikna med relévernet.

Modellane presentert i denne oppgåva har som funksjon å etterlikne reelle relévern. Verna vart bygd opp frå grunnleggande teori om verkemåte funnen i relevant litteratur og relevante artiklar. Denne verkemåten, som er presentert i fyrste del, er oppnådd på alle modellar. Av verifiserte modellar viser det seg at overstraumverna og signalbehandlingsmodellane har ein verkemåte tilsvarande reelle relévern, medan impedansvernmodellen detekterar feil litt raskare enn relévernet. Dette kjem truleg av at relévernet nyttar seg av ytterligare filtrering i tillegg til FFT.

Abstract

This thesis presents the development of a new protective relay class in ATPDraw. This is not standardly available today and it could give the simulation software a broader area of use.

The study has been divided in two main parts. The first part describes some of the basic theory and background, whereas the second part shows the resulting source code programmed in MODELS.

The first part of the paper covers discrete signal processing and the models developed are; two types of rms calculators, a discrete Fourier transform (DFT) model, an 8 sample fast Fourier transform (Radix-2) model. The relays implemented are; inverse and instantaneous overcurrent relays, differential relays and various types of distance protection with and without out of step (OOS) logic.

In the second part mode of operation and resulting outputs are shown, and different signal processing units compared. The paper also contains a short verification where some of the models functionality is compared to real world relays.

All developed models are organized into separate building blocks and a future work is to combine these into full-blood, 3-phase relay models.

Forord

Denne rapporten er eit resultat av masterprosjektet i MSc. Electric Power Engineering ved Norges Teknisk-Naturvitenskapelige Universitet. Rapporten vart utarbeidd vårsemesteret 2012, men byggjer delvis på informasjon innhenta under arbeid med spesialiseringsprosjektet hausten 2011. Det vert i denne rapporten ikkje vist til hausprosjektet. Aktuell innhenta informasjon vert her nytta i sin heilheit.

Bakgrunn for at eg valde denne oppgåva er mi interesse for relévern og programmering. Arbeidet med denne oppgåva starta allereie sommaren 2011, der eg, saman med veileder Prof. Hans Kristian Høidalen, fekk utforma ei problemstilling som traff mine interesseområde svært godt. Spesialiseringsprosjektet hausten 2011 vart nytta som forberedande studie til masterprosjektet. Der fekk eg muligheiten til å setje meg grundig inn i bakgrunn og teori om arbeidet eg skulle utføre våren 2012.

Arbeidet vart stort sett utført gjennom studering av relevant litteratur og programvare for så å gjengi dette med programmering, der det på haustsemesteret vart nytta mykje tid til studering av teori, medan vårsemesteret stort sett vart nytta til programmering. Mot slutten av vårsemesteret vart resultatet utprøvd i fornybarlaboratoriet. Takk til SINTEF/NTNU for at dette vart stilt til min disposisjon.

Eg vil også rette ei stor takk til fakultetet, og ikkje minst Hans Kristian, som har gitt meg rask og god veiledning i dei stundene det har vore naudsynt. Samstundes vil eg takke gode studiekameratar og kollegaer, som har gitt meg verdifull fagleg og sosial støtte dette siste året. Det har vore eit lærerikt og spennande år, og eg er svært takknemleg for den kompetansen og erfaringa eg no har tileigna meg.

Trondheim, 21. juni 2012

Torstein Stadheim

Innhold

Figurer

Kapittel 1

Introduksjon

Denne rapporten er eit resultat av masterprosjektet, utført ved Norges Teknisk-Naturvitenskaplige Universitet. Arbeidet vart i hovudsak utført vårsemesteret 2012, men noko av innhaldet er henta frå fordjupingsprosjektet utført hausten 2011. Rapporten frå haustprosjektet er ikkje publisert, så aktuelt innhald vert teke oppatt i denne rapporten. Masteroppgåva er avsluttande arbeid av den 2-årige mastergraden, MSc. Electrical Power Engineering.

Oppgåva har tittelen «Implementering av vern i ATPDraw». ATPDraw er ein forprosessor til ATP versjonen av ElectroMagnetic Transient Program(EMTP).

EMTP er ei samlenemning for program som simulerer elektriske kretsars transiente oppførsel. Utgangspunktet for simuleringsmotoren til dei fleste av EMTP programma er basert på arbeidet til Professor Herman Dommel med fleire. Den fyrste versjonen av denne typen simuleringsmotoren vart laga i samarbeid med Bonneville Power Administration (BPA) tidleg på 60-tallet. Utviklinga av ATP starta i 1984, då Dr. W. Scott Meyer og Dr. Tsu-huei Liu ikkje godtok at «the EMTP Development Coordination Group» og «the Electric Power Research Institute» ville kommersialisera BPA sin opne versjon av EMTP. Utviklinga av ATP baserte seg på ein kopi av den, til da, opne versjonen av EMTP.

Eksempel på andre versjonar av EMTP er PSCAD-EMTDC, RTDS, MT-EMTP og EMTP-RV.

For at ATP skal kunne simulere lyt det ha eit sett med parameterar. I desse parameterane ligg det mellom anna informasjon om simuleringstid og kretsparameterar. ATP krev at denne informasjonen vert gitt i form av ei eller fleire tekstfiler. Desse tekstfilene kan,

om nettverket er stort, verte uoversiktlege. Som ein konsekvens er oppsettet av ei slik fil omfattande, lite intuitiv og utsatt for menneskeleg feil.

ATPDraw er ein verktøy som kan brukast for å forenkle bruken av ATP. ATPDraw har eit grafisk grensesnitt og eit omfattande bibliotek med komponentar og modellar. Desse komponentane kan veljast og knytast saman i eit enkelt og oversiktleg grafisk grensesnitt. Når kretsen er ferdig konstruert og simuleringsparametrane satt skapar ATPDraw tekstfila som ATP brukar. ATPDraw vart utvikla av Hans Kristian Høidalen, som også er rettleiar på dette prosjektet.

Sjølv om biblioteket i ATPDraw er omfattande, inneheld det ikkje relévern. Ved transient analyse er det ikkje tradisjon for å inkludere vern i kalkuleringa, men etterspurnaden frå brukarmassen har vore der. Ei implementering av relévern vil gi programmet auka allsidigheit og nytteverdi. Bruksområdet vil vere innleiande studiar av relévernrespons og pedagogiske læresituasjonar. I praktisk relévernplanlegging må vern typisk testast fysisk, og ikkje kun gjennom simuleringar.

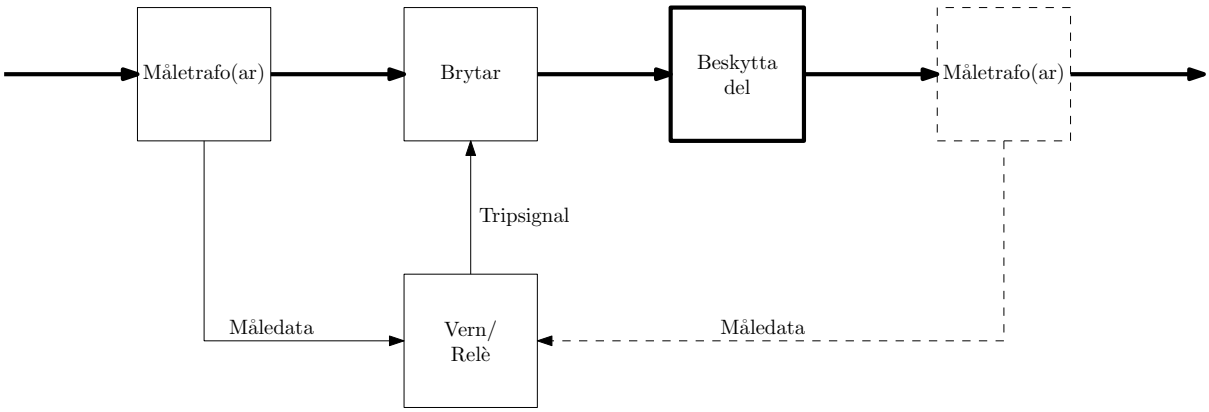
Kapittel 2

Teori

Frå tid til annan vert eit elektrisk kraftsystem utsett for feil. Verna si hovudoppgåve er todelt[4, s.1]:

- detektera og kople ut den feilande komponenten av anlegget.
- Gjere dette så fort som råd samtidig som ein sikrar at minst muleg av det friske anlegget vert råka(selektivitet).

Vern, med unntak av sikringar, er einingar som treng måledata og ein eller fleire brytarar for å fungere. Måledata vert forsynt vernet gjennom måletransformatorar. Dersom vernet detekterar feil vil det sende signal til brytar(ane) om utkopling.



Figur 2.1: Prinsippskisse vern

I byrjinga av 1900-tallet var det vanleg med elektromekaniske relévern. Desse nytta målte straumar og spenningar direkte, gjennom spolar og induksjonsdiskar, for å detektera feil. Dette førte til usikre utkoplingstider på opp mot fleire hundre millisekund. I dagens moderne relévern er det meir og meir vanleg at den elektromekaniske styringa er skifta

ut med indirekte styring ved hjelp av mikroprossesorar. Dette har gitt kortare og sikrare utkoplingstider, i visse tilfelle heilt ned mot ein periode av inngangssignalet. Relévern styrt av mikroprosessorar vert kalla numeriske relé. Ei utfordring med numeriske relé i forhold til elektromekaniske relé er at målte straumar og spenningar må omformast til digitale data som mikroprosessoren kan nyttegjere seg av. Denne omforminga vert i denne rapporten kalla signalhandtering.[3, s.11]

I dette kapittelet er det ei kort oversikt over dei mest nytta verntypane og deira verkemåte, samt ei kort innføring i dei mest brukte signalbehandlingstypane i numeriske relévern.

2.1 Signalhandtering

2.1.1 Diskré signal

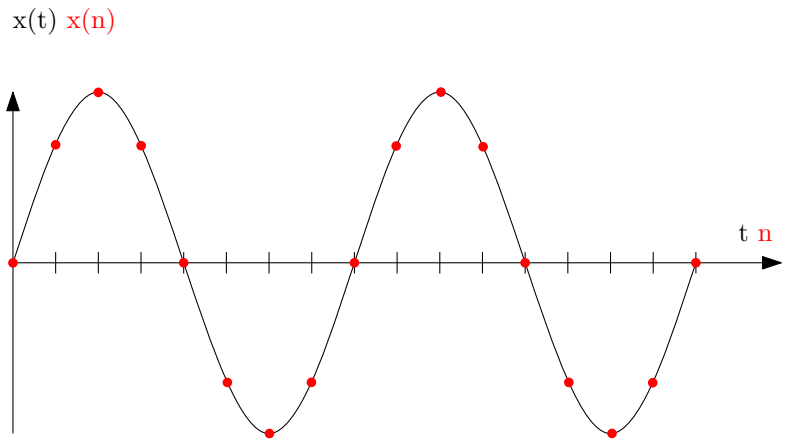
2.1.2 Effektivverdi

Root Mean Square(RMS) er vanleg å nytte dersom vernet skal koordinerast med andre typar vern som sikrar ved hjelp av varmeutvikling (t.d. sikringar). Vanlegvis brukar overstraumsvern denne metoden, gjerne kombinert med filter. Filteret har då som hensikt å fjerne støy og over-harmoniske komponentar der dette er ynskjeleg. Filter er ofte naudsynt i anlegg med kraftelektronikk/likerettarar[1, s.100].

$$\text{AnalogRMS} = \sqrt{\frac{1}{T2 - T1} \cdot \int_{T1}^{T2} x(t)^2 \cdot dt} \tag{2.1}$$

Over er definisjonen på RMS-verdi av eit kontinuerleg signal. Under er ei tilnærming for eit diskré signal med like stor avstand mellom kvart datapunkt i tid. Sjå figur 2.2. Figuren viser korleis datapunkta må vere jamt fordel over X-aksen om ein skal få ei korrekt utrekning av effektivverdien til funksjonen.

$$\text{DigitalRMS} = \sqrt{\frac{1}{N} \cdot \sum_{n=1}^{N} x(n)^2} \tag{2.2}$$



Figur 2.2: Svart: kontinuerleg signal, raud: diskré ekvivalent

2.1.3 Fouriertransformasjon

Dette delkapittelet tek for seg analyse av periodiske sinusforma signal.

I numeriske vern er fourier transformasjon ein vanleg måte for å kalkulere fase og effektivverdi av eit signal[3, s.358]. Det finst mange måtar og gjere ein fourier transformasjon på[2, kap.4]. Forskjellige utgåver av Fast Fourier transformasjon(FFT) er den mest nytta ved behandling av diskré signal. Bakgrunn for utvikling av FFT-algoritmar er at dei krev mykje mindre datakraft enn diskré fourier transformasjon (DFT). Dette gir fordelar med tanke på at ein ikkje treng like mykje reknekraft på verna, som igjen gjer dei rimelegare og mindre.

Diskré Fourier transformasjon

Diskré signal er, i motsetning til eit kontinuerleg signal, bygd opp av ei rekke datapunkt. For å prosessere signalet lyt ein samle inn desse datapunkta. Dess fleire datapunkt ein samlar inn, dess fleire kalkulasjonar trengst det for å behandle dataa. Samlar ein inn for lite datapunkt vil ein miste viktig informasjon om signalet, samlar ein inn for mykje vil prosessen verte treg. difor lyt ein i kvart tilfelle finna balanse mellom krav til hurtigheit og informasjon[2, kap.2].

Dersom ein samlar inn for lite data kan ein risikere at dei ulike frekvenskomponentane i signalet overskygga kvarandre og lagar støy. Dette vert kalla aliasing[2, s.7 kap.2]. For å unngå aliasing bør ein difor alltid samle inn data ved ein frekvens høgare enn det dobbelte av den høgaste frekvens til signalet:

$$f_s \geq 2 \cdot B \tag{2.3}$$

Denne likninga vert i teorien ofte kalla Nyquist-kriteriet. B er den høgaste frekvensen som signalet kan ha og f_s er innsamlingsfrekvensen.

Dersom signalet likevel har frekvenskomponentar som er større enn bandbreidda til fourierfilteret, vil desse forårsake støy på signalet. Dette kan til dømes vere støy ved operasjon av ein effektbrytar. I praksis vil det svært sjeldan vere signal som ikkje inneheld høgfrekvente komponentar i større eller mindre grad. Ein kunne då sett bandbreidda svært høg, men dette vil ført til ein omfattande kalkulasjonsprosess grunna behov for store mengder data. I praksis vert det difor brukt filter som filtrerar vekk høgfrekvente komponentar(lav-pass filter)[2, s.9 kap.2].

FFT er eigentleg berre ein effektiv måte å berekne DFT. Ein startar difor med å forklare framgangsmåten for ein DFT kalkulasjon.

$$X(f) = \int_{-\infty}^{\infty} x(t) \cdot e^{-j2\pi \cdot t \cdot f} dt \tag{2.4}$$

der $X(f)$ er eit komplekst tall i frekvensdomenet, mens $x(t)$ er signalet i tidsdomenet. Dette er definisjonen av fouriertransformasjon til eit kontinuerleg signal. Under er likning (2.4) tilpassa diskré signal.

$$X(m, N) = \sum_{n=0}^{N-1} x(n) \cdot e^{-j2 \cdot \pi \cdot n \cdot m/N} \tag{2.5}$$

Der N er antall prøvar i dataserien, m er nummeret på den harmoniske som skal bereknast og n er prøvenummeret.

Eulers formel:

$$e^{-j \cdot \phi} = \cos(\phi) - j \sin(\phi) \tag{2.6}$$

Likning (2.5) kombinert med likning (2.6)

$$X(m, N) = \sum_{n=0}^{N-1} x(n) \left[\cos(\frac{2 \cdot \pi \cdot n \cdot m}{N}) - j \sin(\frac{2 \cdot \pi \cdot n \cdot m}{N}) \right] \tag{2.7}$$

$X(m, N)$	Komplekst tal som angir den m'te frekvenskomponenten
m	Heiltal som angir nummer på harmoniske
$x(n)$	Dataserien
n	Dataprøvenummer
N	Mengd datapunkt i serien

Då m skal vere eit heiltal ved DFT-analyse, kan ein ikkje angi ein vilkårleg frekvens for analyse. Frekvensen $X(m)$ representerar vert gitt av avstand mellom kvart datapunkt i dataserien og heiltalet m. For at DFT analysen skal bli rett, må avstanden mellom datapunkta i serien vere like stor.

Dersom avstanden mellom kvart punkt er Δt (gitt i sekund), vil frekvensspekteret $X(m, N)$ representera vere gitt av[2, s.3 kap.3]

$$f(m, N) = \frac{m}{N \cdot \Delta t} = \frac{m \cdot f_s}{N} \tag{2.8}$$

Innsamlingsfrekvensen, f_s, må møtekoma likning (2.3)(Nyquist). Maksimal frekvens signalet som skal analyserast kan ha ved DFT er difor gitt av:

$$f_{max} = \frac{f_s}{2} \tag{2.9}$$

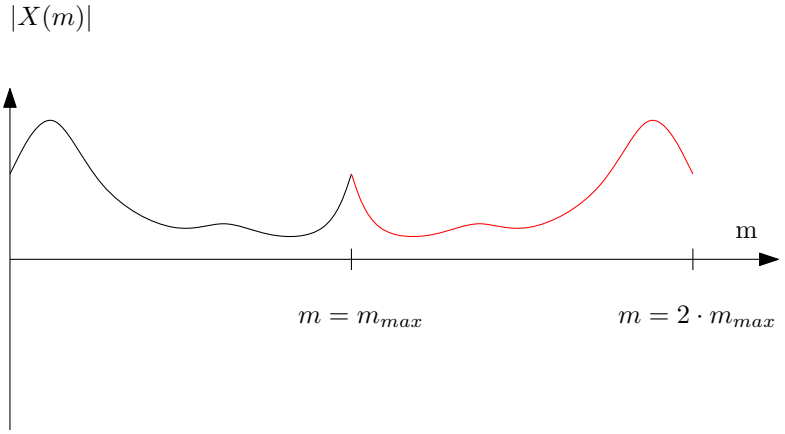
Ein kan såleis analysera frekvensspekteret mellom 0 og f_{max} med steg på

$$f_\Delta = \frac{f_s}{N} \tag{2.10}$$

der maksimal mengd steg er frå 0 er

$$m_{max} \leq \frac{N}{2} \tag{2.11}$$

Dersom det blir brukt m-verdiar større enn dette, vil spekteret frå $m = 0$ til $m = m_{max}$ bli duplisert og invertert langs frekvensaksen[2, s.81 kap.3](figur 2.3).

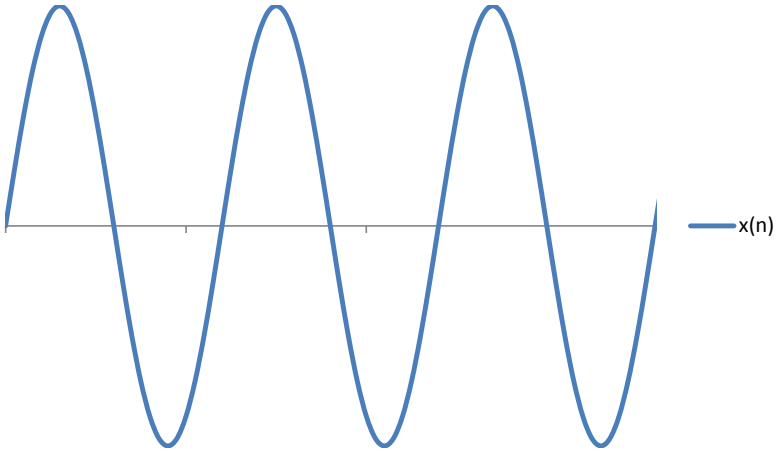


Figur 2.3: Raud: Frekvensrespons dersom $m \geq m_{max}$

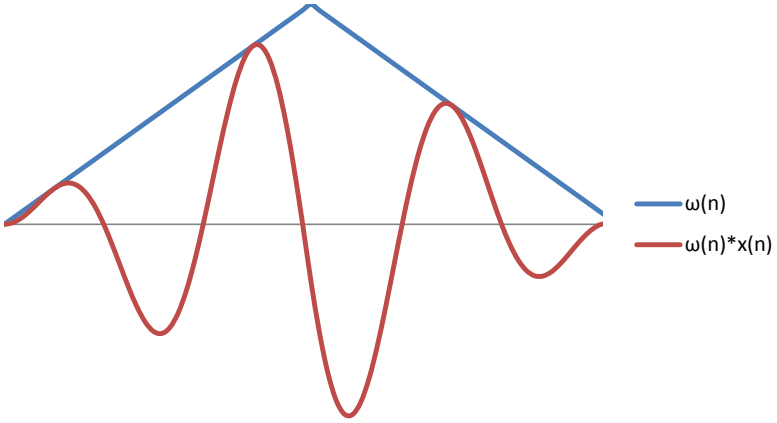
Om signalet inneheld frekvenskomponentar som ligg mellom to frekvenssteg, til dømes $1.5 \cdot f_\Delta$, vil det gi utslag på dei næraste frekvensstega. Dette vert kalla lekkasje[2, s.21 kap.3]. Ved bruk av DFT er det ikkje muleg å fjerne denne effekten, men den kan reduserast ved hjelp av ein teknikk kalla «windowing». Enkelt sagt vert signalet då multiplisert med ein funksjon. Denne funksjonen kan ha fleire former alt etter kva effekt ein ynskjer[2, s.28 kap.3].

$$X(m,N) = \sum_{n=0}^{N-1} x(n) \cdot \omega(n) \left[\cos(\frac{2 \cdot \pi \cdot n \cdot m}{N}) - j \sin(\frac{2 \cdot \pi \cdot n \cdot m}{N}) \right] \tag{2.12}$$

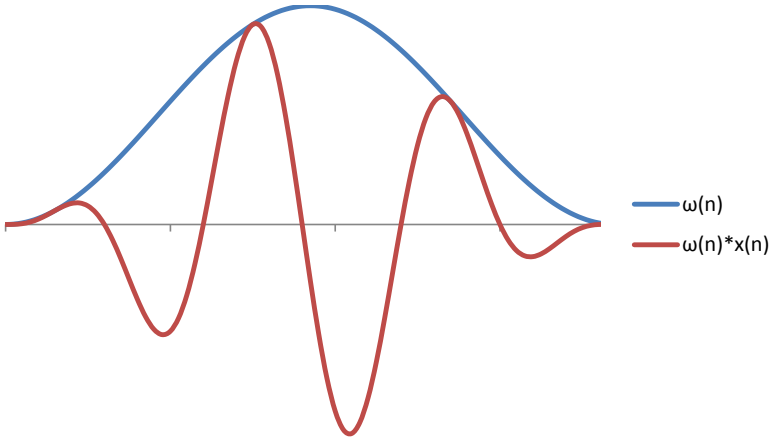
Der $\omega(n)$ er windowfunksjonen. Døme på window-funksjonar i figur 2.4.



(a) Ubehandla signal



(b) Rektangulær windowfunksjon



(c) Hanning windowfunksjon

Figur 2.4: Døme på window-funksjonar [2, s.30 kap.3]

Kvar funksjon har individuelle fordelar og ulemper, og må difor tilpassast analysesituasjonen.

Mengd kalkulasjonar ved DFT analyse er direkte proporsjonal med mengd prøvar. difor er det eit mål i seg sjølv å halde N så liten som muleg. Den minste prøvemengda ved ein gitt innsamlingsfrekvens og eit gitt frekvenssteg er

$$N_{min} = \frac{f_s}{f_\Delta} \tag{2.13}$$

Døme på minimering av kalkulasjonar ved DFT-analyse: Er interessert i effektivverdi og fase på eit signal som har frekvens på 50Hz. Signalet inneheld fleire komponentar med høgare frekvens, men det er filtrert slik at den største frekvenskomponenten er 400Hz. Innsamlingsfrekvens vert difor 800Hz(likning (2.3)). Då ein kun er interessert i 50Hz komponenten må f_Δ vere 50Hz. Mengd prøvar vert difor $800/50 = 16$.

Det kan vere verd å merke seg at den minste mengda prøvar spenner seg over eit tid tilsvarande ein periode av den fyrsteharmoniske komponenten:

Tid mellom kvar prøve:	$1/800Hz = 1.25ms$
Mengd prøvar:	16
Periodetid:	$16 \cdot 1.25 = 20ms$
Frekvens:	$1/20ms = 50Hz$

Effektivverdien til eit sinussignal analysert med DFT er gitt av [2, s.15 kap.3]

$$|X(m,N)|_{rms} = \sqrt{X_{re}^2 + X_{im}^2} \cdot \frac{\sqrt{2}}{N} \tag{2.14}$$

med vinkel

$$\angle(X(m,N)) = \tan(\frac{X_{im}}{X_{re}}) \tag{2.15}$$

«Radix-2» Fast Fourier transformasjon

Den mest populære FFT-transformasjonen er kalla «Radix-2». Denne utnyttar symmetri i likning (2.5).

Ein vilkårleg sum kan brytast ned til to delsummar:

$$X(m) = \sum_{n=0}^{N-1} x(n) \cdot \alpha_m = \sum_{n=0}^{N/2-1} x(2n) \cdot \alpha_{2n,m} + \sum_{n=0}^{N/2-1} x(2n+1) \cdot \alpha_{2n+1,m} \tag{2.16}$$

Ein slik delsum kan på same måte brytast ned i to nye delsummar. Denne prosessen kan repeterast til ein har N/2 delsummar. Difor er det eit krav ved Radix-2 FFT at signalet

må innehalde

$$N = 2^{heiltall} \tag{2.17}$$

dataprøvar[2, s.3 kap.4]. Dersom denne nedbrytinga vert gjort på likning (2.5) vil ein kunne redusere mengda med multiplikasjonar kraftig.

Døme: I dette tilfellet er det N=8 prøvar.

$$\begin{aligned}
&X(m,8) = \sum_{n=0}^{7} x(n)\cdot e^{-j2\cdot\pi\cdot n\cdot m/8} \\
&= \sum_{n=0}^{3} x(2n)\cdot e^{-j2\cdot\pi\cdot 2n\cdot m/8} + \sum_{n=0}^{3} x(2n+1)\cdot e^{-j2\cdot\pi\cdot(2n+1)\cdot m/8} \\
&= \sum_{n=0}^{3} x(2n)\cdot e^{-j2\cdot\pi\cdot 2n\cdot m/8} + e^{-j2\cdot\pi\cdot m/8}\sum_{n=0}^{3} x(2n+1)\cdot e^{-j2\cdot\pi\cdot 2n\cdot m/8} \\
&= \sum_{n=0}^{1} x(2\cdot 2n)\cdot e^{-j2\cdot\pi\cdot 2\cdot 2n\cdot m/8} + \sum_{n=0}^{1} x(2\cdot(2n+1))\cdot e^{-j2\cdot\pi\cdot 2\cdot(2n+1)\cdot m/8} \\
&+e^{-j2\cdot\pi\cdot m/8}\left[\sum_{n=0}^{1} x(2\cdot 2n+1)\cdot e^{-j2\cdot\pi\cdot 2\cdot 2n\cdot m/8} + \sum_{n=0}^{1} x(2\cdot(2n+1)+1)\cdot e^{-j2\cdot\pi\cdot 2\cdot(2n+1)\cdot m/8}\right] \\
&= \sum_{n=0}^{1} x(4n)\cdot e^{-j2\cdot\pi\cdot 2\cdot 2n\cdot m/8} + e^{-j2\cdot\pi\cdot 2\cdot m/8}\sum_{n=0}^{1} x(4n+2)\cdot e^{-j2\cdot\pi\cdot 2\cdot 2n\cdot m/8} \\
&+e^{-j2\cdot\pi\cdot m/8}\left[\sum_{n=0}^{1} x(4n+1)\cdot e^{-j2\cdot\pi\cdot 2\cdot 2n\cdot m/8} + e^{-j2\cdot 2\cdot\pi\cdot m/8}\sum_{n=0}^{1} x(4n+3)\cdot e^{-j2\cdot\pi\cdot 2\cdot 2n\cdot m/8}\right]
\end{aligned} \tag{2.18}$$

Skiftar vinkelkoeffisienten $e^{-j2\pi}$ ut med α:

$$\begin{aligned}
&X(m,8) = \sum_{n=0}^{1} x(4n)\cdot\alpha^{\frac{n\cdot m}{2}} + \alpha^{\frac{m}{4}}\cdot\sum_{n=0}^{1} x(4n+2)\cdot\alpha^{\frac{n\cdot m}{2}} \\
&+\alpha^{\frac{m}{8}}\cdot\left[\sum_{n=0}^{1} x(4n+1)\cdot\alpha^{\frac{n\cdot m}{2}} + \alpha^{\frac{m}{4}}\cdot\sum_{n=0}^{1} x(4n+3)\cdot\alpha^{\frac{n\cdot m}{2}}\right]
\end{aligned} \tag{2.19}$$

Det er eitt produkt som går igjen i kvart ledd. Dette kombinert med likning (2.6):

$$\alpha^{\frac{n\cdot m}{2}} = \cos(\frac{2\cdot pi\cdot n\cdot m}{2}) - j\sin(\frac{2\cdot pi\cdot n\cdot m}{2}) = \cos(pi\cdot n\cdot m) - j\sin(pi\cdot n\cdot m) \tag{2.20}$$

Då m og n skal vere heiltal vil sinusleddet alltid vere null og cosinusleddet alltid 1 eller -1. Det går då an å setje opp tabellar for når cosinusleddet er positivt/negativt slik at det ikkje vert naudsynt å multiplisere desse produkta. Denne tabellen må tilpassast ei gitt mengde prøvar (N).

Døme på berekning av fyrsteharmoniske ($X(1,8)$)

$$\begin{aligned} X(1,8) &= \sum_{n=0}^{1} x(4n)\cdot\alpha^{\frac{n}{2}} + \alpha^{\frac{1}{4}}\cdot\sum_{n=0}^{1} x(4n+2)\cdot\alpha^{\frac{n}{2}} \\ &+\alpha^{\frac{1}{8}}\cdot\left[\sum_{n=0}^{1} x(4n+1)\cdot\alpha^{\frac{n}{2}} + \alpha^{\frac{1}{4}}\cdot\sum_{n=0}^{1} x(4n+3)\cdot\alpha^{\frac{n}{2}}\right] \\ &= x(0) - x(4) + \alpha^{\frac{1}{4}}\cdot[x(2) - x(6)] + \alpha^{\frac{1}{8}}\cdot\Big[x(1) - x(5) + \alpha^{\frac{1}{4}}\cdot[x(3) - x(7)]\Big] \end{aligned} \tag{2.21}$$

I eksempelet er det behov for tre komplekse multiplikasjonar i motsetning til åtte i likning (2.5). Slik kan Radix-2 FFT tilpassast alle signal som møter kriteriet i likning (2.17). Til fleire prøvar det er i signalet, til raskare er FFT samanlikna med DFT.

2.2 Overstraumvern

Overstraumvern reagerer på straumar større enn ein førehandsinnstilt verdi. Dersom straumen overstig denne verdien vil vernet sende signal til ein brytar slik at feilen vert kopla ut[4, p.78].

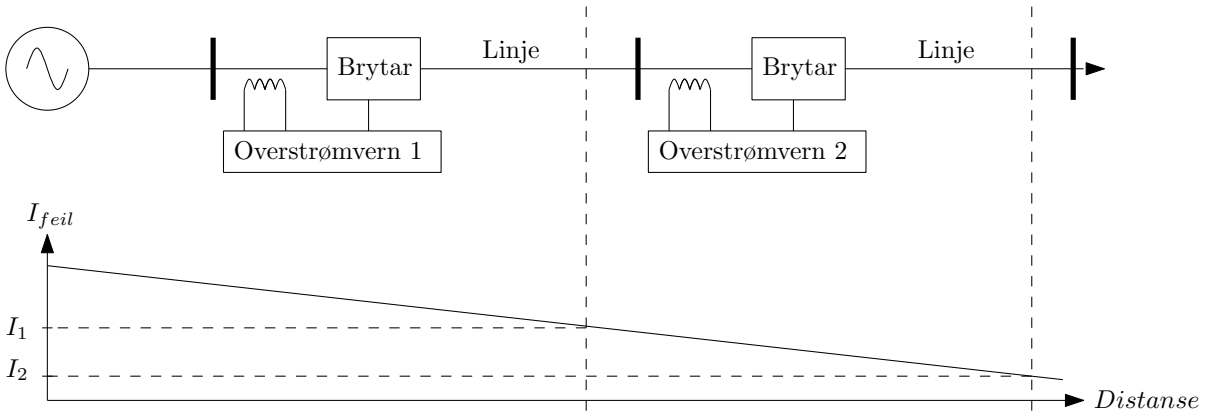
For å ha stabil og sikker drift skal overstraumvern ikkje detektera feil ved tunglast og normale straumvariasjonar i nettet:

$$\text{Maksimal last}\cdot 1.5 \leq \text{Utløysarstraum} \leq \frac{\text{Minimum feilstraum}}{2} \tag{2.22}$$

Likning (2.22)[5, s.167] viser anbefalte verdiar for utløysarstraum. I maksimal last må det, ved fase-fase vern, takast omsyn til den største straumen som kan oppstå, t.d. startstraumar på motorar og normal høg last. For jordfeilvern er den største straumen gitt av naturleg null-sekvens ubalanse og fase-fase-nøytral laster.

2.2.1 Momentan overstraumvern

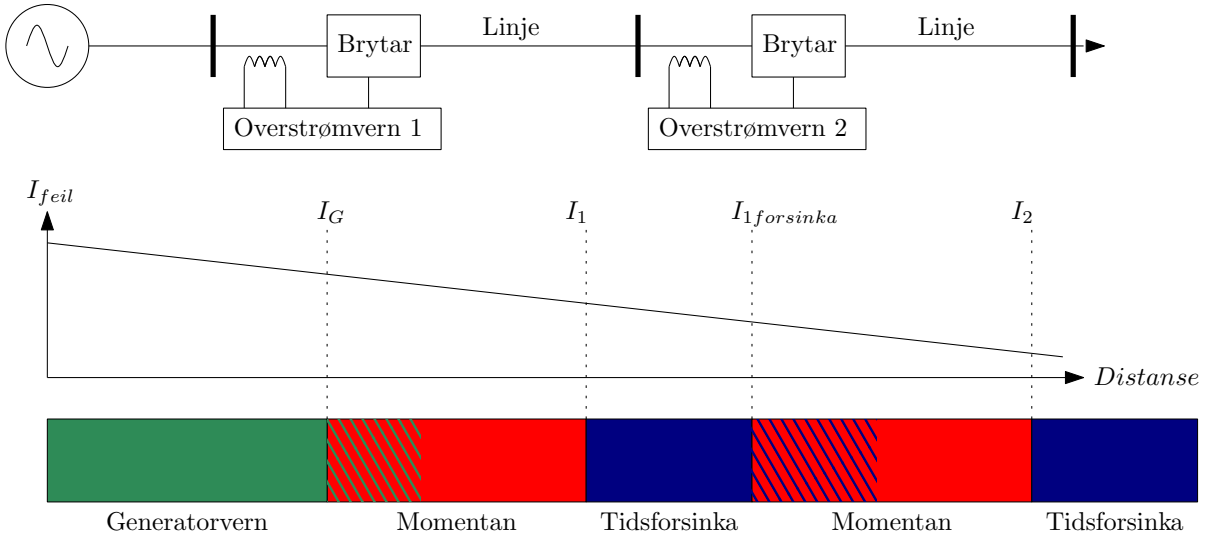
Momentan overstraumvern detekterar feil umiddelbart(16-20 ms[5, s.175]) når straumen når ein førehandsinnstilt verdi. Ved vern av linjer vil feilstraumen vere proporsjonal med avstand til feil, og innstilt verdi på vernet bør vere så stor at ein ikkje løyser ut ved feil i andre vern sitt verneområde. Det er difor vanleg å bruke momentanvern i kombinasjon med tidsforseinka vern[4, p.88].



Figur 2.5: Feilstraum som funksjon av avstand, forenkla

For å unngå at oppstraums momentanvern skal detektera ved feil i nedstrøms vern sitt område, bør deteksjonsverdi vere større enn feilstraumen som oppstår ved feil på 80% av linjeavstanden til neste momentanvern(opp mot 90% dersom ein nyttar numeriske vern)[4, s.89].

I figur 2.5 bør minste tillatne deteksjonsverdi på momentanvern 1 difor vere større enn I_1. Ved ei slik innstilling vert det eit område på 10-20% av linja som ikkje er dekka av vern. For å dekke denne delen kan ein, som nemnt, nytte seg av tidsforseinka overstraumvern i kombinasjon med momentanvernet.



Figur 2.6: Kombinasjon av momentant og tids-forseinka overstraumvern

Dersom ein nyttar seg av denne løysinga kan ein sikre alle delar av anlegget samstundes som selektivitet vert opprettholdt. Ulempa med denne løysinga er at nokre deler av

anlegget får lengre utkoplingstid(blått område figur 2.6). Dei tidsforseinka verna har også overlapping inn i nedstrøm verns momentanområde(som reservevern).

Døme figur 2.6:
Overstraumvern 1 har momentan deteksjon på straumar mellom I_G og I_1 og tidsforseinka deteksjon på straumar mindre enn I_1. Overstraumvern 2 har momentan deteksjon på straumar mellom $I_{1forseinka}$ og I_2 og tidsforseinka deteksjon på straumar mindre enn I_2.

2.2.2 Konstant tid

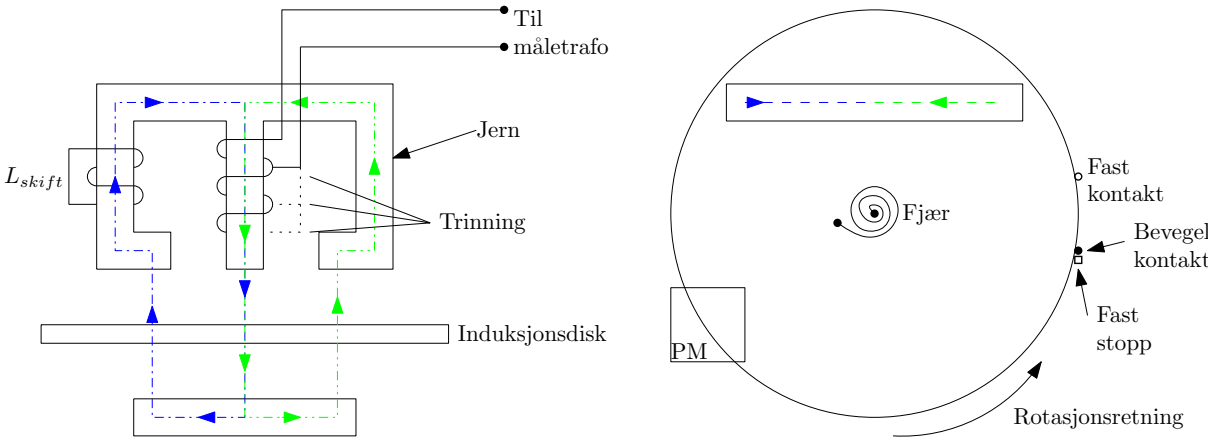
Desse verna har same verkemåte som momentan vern, men ein kan velje kor lang tid det skal gå frå feilkriteriet vert oppfylt, til signal om utkopling vert gitt.

2.2.3 Invers, tidsforseinka

Dei fyrste overstraumverna vart utvikla for 60-70 år sidan og bestod då av ein enkel kilowattime-målar med kontaktar som slo inn når induksjonsdisken nådde ein gitt fart. Utviklinga har ført til at ein i dag ikkje vil finne igjen denne typen vern. Dagens overstraumvern kan ein finne i fleire utføringar, både analoge og digitale utgåver[5, s.170].

Dei analoge verna bruker prinsippet om induksjonsdisk. Prinsippet om induksjonsdisk går ut på at ein bruker målt straum til å drive ein induksjonsdisk rundt. Dette vert gjort ved at straumen set opp eit magnetfelt som går gjennom disken. På eine sida vert fluksen faseforskyvd ved hjelp av ein spole. Dette gjere at ein får ei rotasjonskraft som virkar på disken. Dess større straumen er, dess større er denne krafta. I tillegg er det satt opp ein permanentmagnet og ei fjær som dempekraft. Fjæra har også som funksjon å tilbakestille kontakten når den ytre rotasjonskrafta opphøyrer. Dersom rotasjonskrafta satt opp av straumen er større enn motkrafta frå fjær vil den bevegelige kontakten rotere mot den faste kontakten. Dersom desse møtest vil vernet sende signal om feil[5, s.170].

Minimum straum som skal til for å starte rotasjon av disken kallast startstraum(pickup current).



Figur 2.7: Induksjonsprinsipp

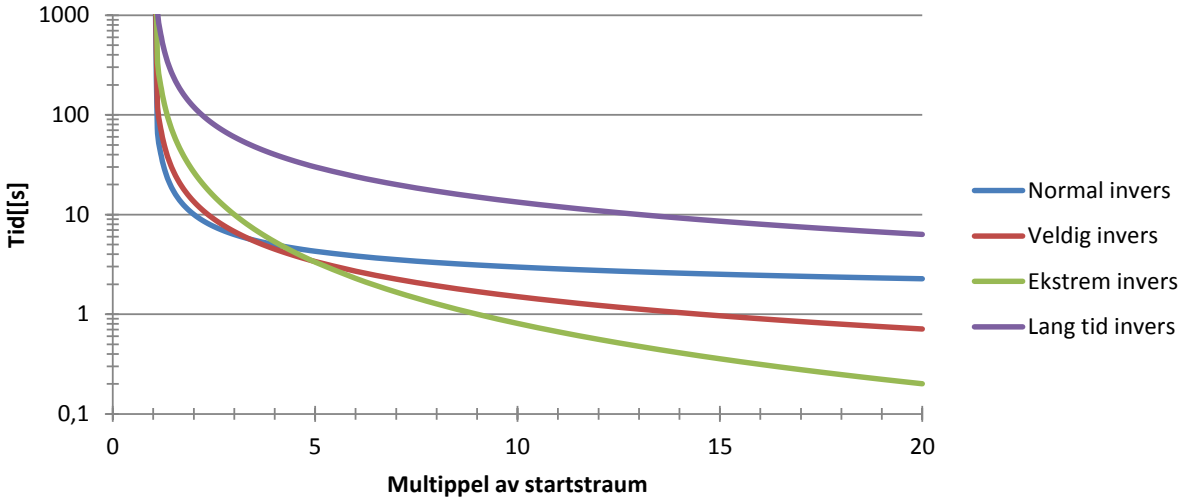
Tid-straum karakteristikken er i hovudsak gitt av ein kombinasjon av straum frå måletrafo og metning i jernet[6, s.373]. Straumen set opp ein fluks der momentet er gitt av

$$T = \Phi_1 \cdot \Phi_2 \cdot \sin\theta. \tag{2.23}$$

Φ_1 er fluksen ein venstre bein, Φ_2 i høgre bein og θ vinkelen mellom dei (figur 2.7). Då fluksen er proporsjonal med straumen kan ein skrive momentet som

$$T = k \cdot I^2. \tag{2.24}$$

der k er ein variabel konstant som endrar seg etter kvart som jernet går i metning. k vert redusert dess meir metta jernet er. Det gjer at ein til slutt når ei øvre grense på moment, og dermed utkoplingstid.



Figur 2.8: Døme på karakteristikk, IEC [7, s.594].

Dess større straumen er i forhold til startstraumen, dess meir flatar kurvene for utkopling ut fordi jernet går i metning. For små straumar mellom 1 og 1.5-2 gonger startstraumen er det ikkje vanleg for produsentar av vern å vise karakteristikk fordi små avvik i straumen kan gi store avvik i tid. Overstraumvern er dermed ikkje praktiske i bruk dersom ein ynskjer å beskytte anlegg ned mot startstraum på grunn av usikkerheit ved utkoplingstid, selektivitet og koordinering.

For at ein skal nytte analoge vern i fleire høve er det gjort tilpassingar i konstruksjonen. Ein kan endre storleiken på rotasjonskrafta ved hjelp av å endre kor mykje av spolen ein køyrer straum gjennom(trinning), og dermed endre startstraumen. Slik kan ein forskyve karakteristikken langs x-aksen. For å unngå og teikne ein graf for kvart trinn er x-aksen som regel gitt som multippel av startstraumen[5, s.170].

$$Multippel = M = \frac{I_{maalt}}{I_{start}}. \tag{2.25}$$

Den digitale versjonen av overstraumvern simulerer verkemåten til sin analoge motpart ved hjelp av matematiske formlar. Fordelen med digitale vern er at ein får større bruksområde, justerbar karakteristikk og mindre byrde[5, s.170].

For å implementere kor langt induksjonsdisken har rotert i eit numerisk vern, kan ein integrere ein funksjon av straumsignalet som angir momentet, eller rotasjonskrafta, påført disken. Denne funksjonen må delast i to. Ein funksjon som omhandlar positiv rotasjonskraft og ein som omhandlar negativ rotasjonskraft. Funksjonen er definert som([12], [7, s.596]):

$$f_t = \frac{1}{t_{kraft}} \tag{2.26}$$

der $M \geq 1$ (positiv rotasjonskraft) gir

$$t_{kraft} = t_{start} = T_D \cdot \left(\frac{A}{M^P - 1} + B\right) + K \tag{2.27}$$

og $M < 1$(negativ rotasjonskraft) gir

$$t_{kraft} = t_{reset} = T_D \cdot \left(\frac{t_r}{M^Q - 1}\right). \tag{2.28}$$

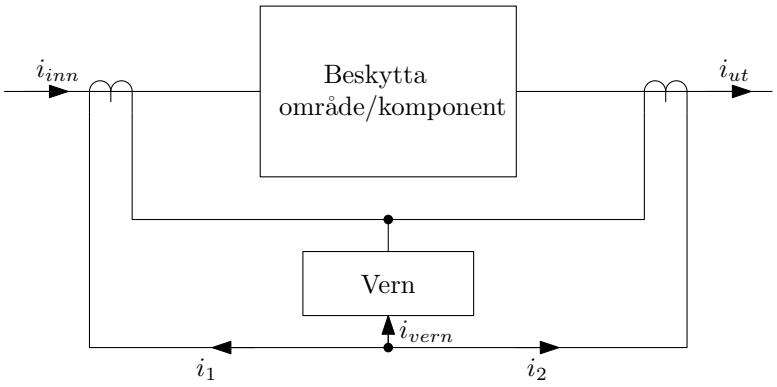
Vernet måler såleis rotasjon ved

$$\text{Rotasjon} = \int_{t_0}^{t} f_t dt \tag{2.29}$$

Der feilkriterie er ein full rotasjon. Ein full rotasjon er ekvivalent med at rotasjonssummen når verdien 1.

2.3 Differensialvern

'The best protection technique now and for more than 50 years is that known as differential protection'[5, p.163]. Differensialvernprinsippet går ut på at ein måler straum inn, og ut av, eit beskytta område eller komponent.



Figur 2.9: Differensialvern prinsipp

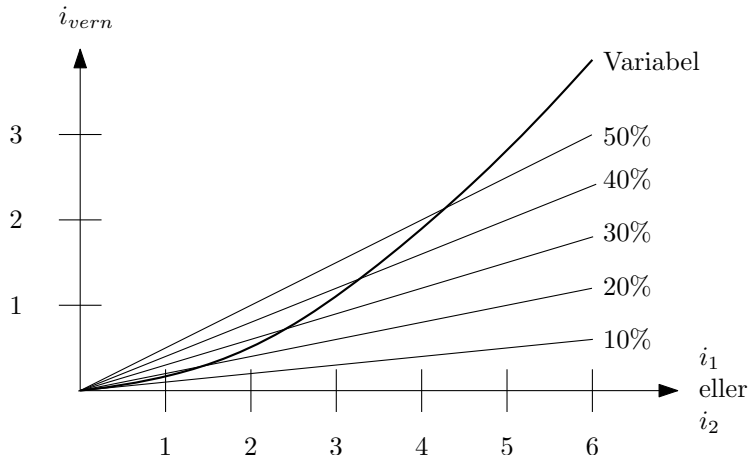
Basert på Kirchoffs straumlov vil $i_{inn} - i_{ut} = 0$ dersom det verna området ikkje inneheld feil. i_1 vil vere proporsjonal med i_{inn} og i_2 vil vere proporsjonal med i_{ut}. $i_{vern} = -i_1 - i_2$ som også vil vere null om det ikkje er feil på komponenten. Feil vert såleis detektert om $i_{vern} \neq 0$.

I eit praktisk tilfelle vil det også i eit feilfritt tilfelle gå straum gjennom vernet. Dette kjem mellom anna av at ein har tap i det beskytta området, ulike omsetningsforhold på straumtrafoane og transiente eigenskapar ved dei. I tillegg vil to straumtrafoar med same merkedata ha små forskjellar.

Differnsialvernet bør difor tolerere differensialstraum opp til eit visst nivå, avhengig av kvart enkelt tilfelle. Ein bør også bruke tidsforseinka logikk for feildeteksjon for å hindre at eksterne feil vert detektert av differensialvernet.[5, s.164].

Ved å nytte prosentkarakteristikk(figur 2.10) kan ein unngå uønskt deteksjon i dei tilfella der differensialstraumen eksisterar, men ikkje som eit teikn på feil. Dersom straumen i komponenten aukar, vil nøyaktigheiten på straumtrafoane verte redusert og differensialstraumen vil kunne auke. Om ein då samtidig måler ein eller begge inngangsstraumane kan

ein lage ein utkoplingskarakteristikk som tek omsyn til faktisk belastning på straumtrafoane. Kven av inngangsstraumen som vert nytta, eller om begge vert nytta, er avhengig av konstruksjonen til vernet[5, s.165].



Figur 2.10: Differnsialvern karakteristikk (aksenummerering kun for illustrasjon)

Døme: Dersom ein vel 10% vil vernet detektera feil dersom i_{vern} er større enn $\frac{1}{10}$ av $i_{inngang}$, der $i_{inngang}$ er i_1 eller i_2(figur 2.9).

Når ein straumtransformator vert nytta i anlegg der han opererer med straumar som er langt under merkestraumen kan ein bruke lav prosent på karakteristikken(liten toleranse), fordi straumtrafoar i desse områda er forholdsvis nøyaktige. Om målt straum nærmar seg merkestraum, kan ein få problem med metning. I desse tilfella er det difor betre å bruke høgare toleranse for å unngå uønska feildeteksjon. Dersom straumtrafoen skal operere i eit anlegg der han vert utsett for både høge og lave straumar, kan det vere nyttig å bruke variabel karakteristikk for oppnå god sensitivitet og driftssikkerheit[5, p.166].

2.3.1 Transformatorvern

Dersom differensialvernet skal verne ein transformator er det nokre spesielle omsyn ein må ta ved tanke på innkopling av transformatoren. Ved innkopling av transformatorar vil det vere forskjell på straum i primær- og sekundærvikling. Dette kjem av at transformatoren vert magnetisert. På grunn av dette ville vernet detektert feil, sjølv om det ikkje er noko gale med transformatoren. Fleire metodar for å unngå uønskt deteksjon av feil under magnetiseringsforløp er presentert i [16].

Den vanlegaste metoden for å skille eit magnetiseringsforløp frå ein feilsituasjon er ved hjelp av restriksjon basert på forholdet mellom den fyrsteharmoniske og den andrehar-

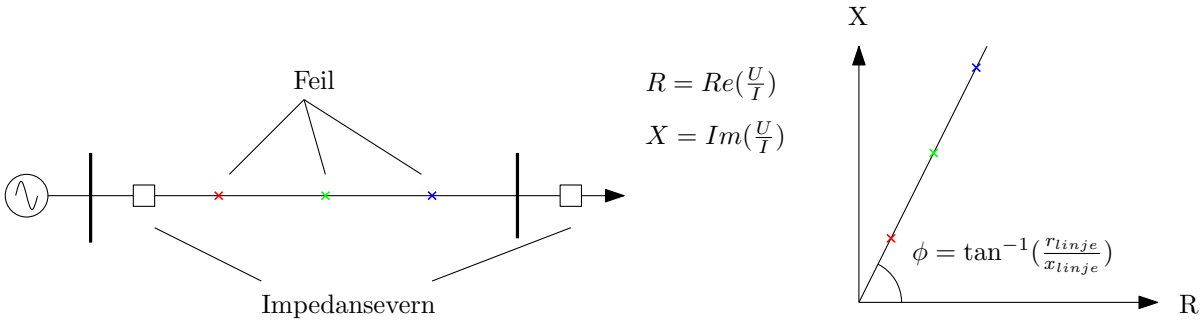
moniske komponenten til differensialstraumen[15]. Forholdet vert berekna som fylgjer:

$$\text{ratio} = \frac{I_{diff2}}{I_{diff1}} \tag{2.30}$$

Der I_{diff1} er den fyrsteharmoniske komponenten og I_{diff2} den andreharmoniske komponenten til differensialstraumen. Dersom dette forholdet overstig ein førehandsinnstilt verdi, vil vernet ikkje detektera feil. Normalt sett vil denne verdien ha ei standardinnstilling på omlag 20%, men med innstillingsmuligheitar frå 1% til 40%. Dersom verdien vert sett for høg, vil vernet kunne detektera feil under eit feilfritt magnetiseringsforløpet. Dersom verdien vert sett for lav kan ein risikere at vernet ikkje detekterar reelle feil i transformatoren. Erfaring tilseier at ein verdi på 20% i dei fleste tilfelle vil gi tilstrekkeleg beskyttelse og driftsikkerheit[15].

2.4 Impedansvern

Impedansvern er normalt nytta til vern av overføringslinje mellom to samleskinner. Ved bygging av slike linjer vert fasane med jamne mellomrom revolvert. Det gir symmetrisk induktans i fasane. Saman med resistansen dannar det impedansen i linja, som difor er lik i alle tre fasane og proporsjonal med lengden. Impedansevernet måler spenning og straum, og kan såleis rekne ut impedansen i systemet. Då impedansen er proporsjonal med lengden vert impedansevern også kalla distansevern[4, s.100].



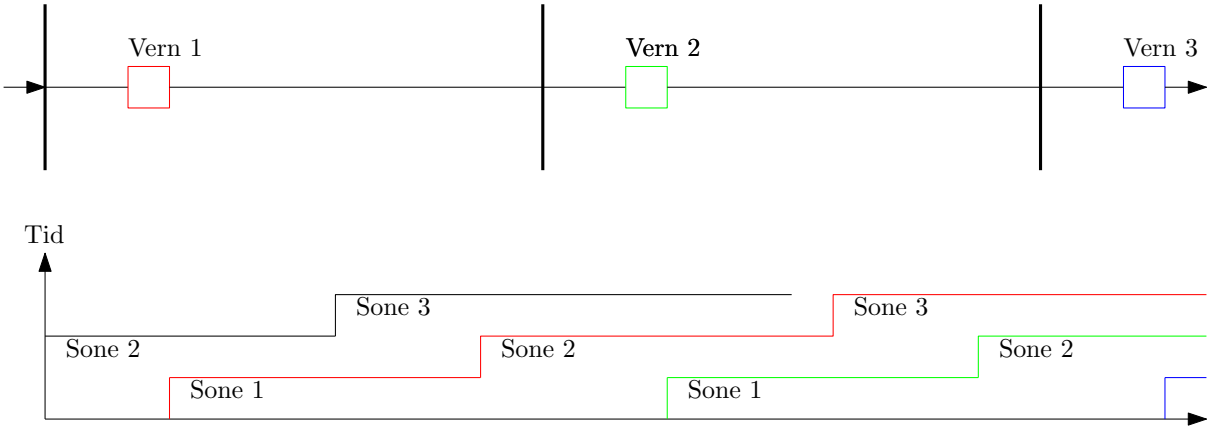
Figur 2.11: Impedans ved ulike feilsituasjonar(kryss)

R-X diagram Då vernet måler straum og spenning, kunne det vert naturleg med eit U-I diagram for innstilling av vern. Dette er ikkje hensiktsmessig, då ein også må ta omsyn til vinkel mellom spenning og straum for å definere i kva retning feilen er. Ein kombinerer difor U og I til ein variabel, Z = U/I. Slik kan ein kombinere tre variablar inn i eit diagram der ein har R og X, eller Z og vinkel(figur 2.11). Det er fleire måtar og lese dette diagrammet på, men om ein set vinkelen til straumen som referansevinkel og

samtidig nytte p.u. verdiar på straum og spenning, vil ein kunne tolke impedansen direkte som spenningsvektoren i målepunktet, med vinkel referert til straumvinkel[4, s.103].

Overføringslinjer er som regel bygd i fleire seksjonar med avgreiningar/samleskinner undervegs. Ein har då som regel eit impedansevern per seksjon. Vernet skal kun detektera feil i sin seksjon, då ein har eit anna vern som tek seg av neste seksjon. Ein kunne stilt vernet inn til å detektera feil fram til neste seksjon, men linjedata er vanskeleg å setje med stor nøyaktigheit. Dersom vernet har for stort arbeidsområde kan ein risikere at det agerer på feil i anna verns arbeidsområde, og dermed koplar ut ein større del av systemet enn naudsynt. Normalt stiller ein difor vern sitt arbeidsområde til 85-90%[4, s.100] av seksjonen si lengd.

I overnemnte tilfelle vil 10-15% av seksjonen stå utan beskyttelse. For å bøte på dette vert det nytta fleire soner med stigande tidsforseinking, der lavt sonetall gir kort utkoblingstid. Slik kan vernet beskytte resterande del av seksjonen, ved at det gir underliggande vern tid til å agere ved feil i eige område. Normalt er sone 2 satt fra slutten av sone 1 til 120-150%[4, s.101] av seksjonen. Dette gir reservebeskyttelse om underliggande vern skulle feile. Typisk tid for utkobling i sone 2 er 0,3 sekund lenger enn i sone 1.



Figur 2.12: Soneinndeling distansevern

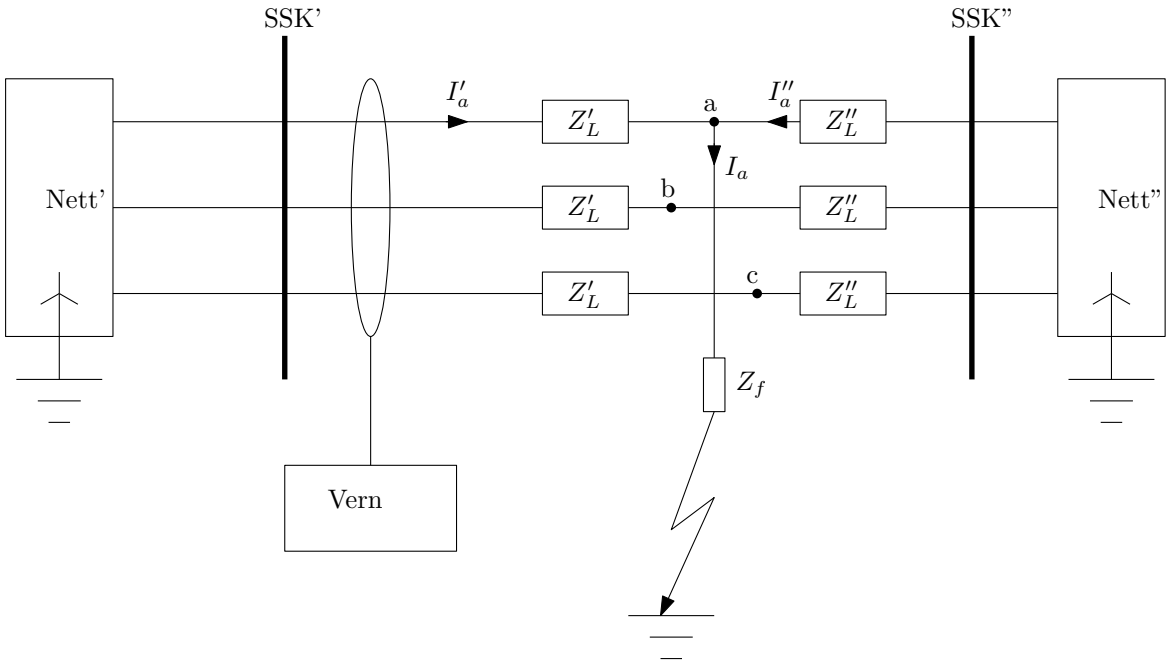
Då sone 2 som regel ikkje er tilstrekkeleg for å gi reservebeskyttelse for heile seksjonen til underliggande vern(figur 2.12), er også ei tredje sone vanleg. Denne har enda lenger utkoblingstid, ofte rundt 1 sekund. Arbeidsområde sone 3 er typisk til 150%[4, s.101] av underliggande verns seksjon.

I alle tilfeller er det viktig at soner med same utkoplingstid ikkje overlappar kvarandre for å oppretthalde høgast muleg selektivitet.

Ei utfordring ved innstilling av reservesoner, er om ein har parallelle linjer i nedstrøms seksjonar. Dersom dette er tilfelle kan ein endre impedansen alt etter kor mange linjer som er i bruk.

Det kan i alt oppstå 10 typar feil på ei trefase overføringslinje, som igjen kan sorterast i 4 kategoriar. Ved feilsituasjonar ser ein vekk frå belastningstraumar i nettet før feilen inntreff, samt at alle spenningskilder i nettet er i fase og av same storleik[9, s. 475].

2.4.1 Fase - jord (3stk)



Figur 2.13: Fase A - jord feil

Denne feilen oppstår dersom ein av fasane får kontakt med jord. Det er muligheit for ein feil per fase, totalt tre feil. Om feil oppstår som i figur 2.13, mellom fase a og jord, vil straumen i fase a vere lik feilstraumen, mens straumane i fase b og c vere lik null. Dette gir usymmetrisk belastning.

$$I_b = I_c = 0 \tag{2.31}$$

$$V_a = I_a \cdot Z_f = (I'_a + I''_a) \cdot Z_f \tag{2.32}$$

For å studere denne typen feil lyt ein difor bryte systemet opp i symmetriske komponentar. Forholdet mellom symmetriske og usymmetriske komponentar er gitt av:

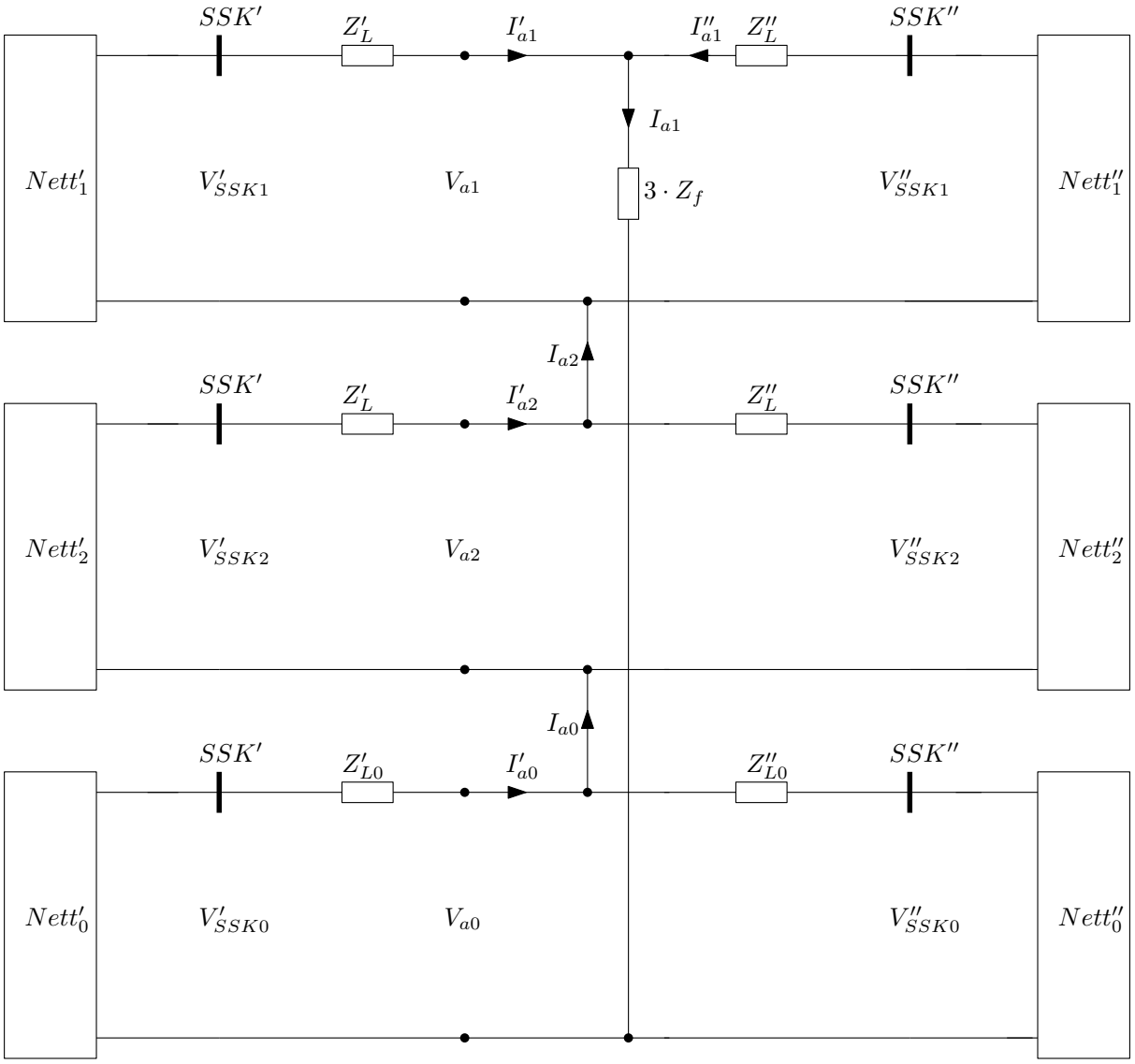
$$\begin{bmatrix} I_0 \\ I_1 \\ I_2 \end{bmatrix} = \frac{1}{3} \begin{bmatrix} 1 & 1 & 1 \\ 1 & \alpha & \alpha^2 \\ 1 & \alpha^2 & \alpha \end{bmatrix} \cdot \begin{bmatrix} I_a \\ I_b \\ I_c \end{bmatrix} \tag{2.33}$$

$\alpha = 1 \angle 120°$. Forholda gjeld også for spenningar. 0 indikerar null sekvens, 1 indikerar positiv sekvens og 2 indikerar negativ sekvens.

Om ein kombinerar (2.31) og (2.33) vil feilstraumen vere gitt av

$$I_0 = 1_1 = 1_2 = \frac{1}{3} \cdot I_a = \frac{1}{3} \cdot I_f \tag{2.34}$$

Då det er same straumane i alle komponentane kan sekvensnettverket koplast saman i serie, og vert sjåande slik ut:



Figur 2.14: Samankopling av sekvensnettverk, fase-jord feil

For positive og negative sekvensar er linjeimpedansen lik. Feilimpedansenz Z_f vert multiplisert med tre fordi sekvensstraumen er ein tredjedel av straumen gjennom feilmotstanden((2.34) og (2.32)). Ved kalkulering av linjeimpedansen vert det ikkje teke omsyn til thevenin-impedansar og spenningar frå oppstrøms nett. Dette fordi vernet måler spenning og straum frå samleskinna, samt at vernet skal 'sjå' i nedstrøms retning. V'_{SSKa} og

I'_a er altså kjent. Spenninga på samleskinna i fase a:

$$\begin{aligned}
V'_{SSKa} &= V'_{SSK1} + V'_{SSK2} + V'_{SSK3} \\
V'_{SSKa} &= I'_{a1} \cdot Z'_L + I'_{a0} \cdot Z'_{L0} + I'_{a2} \cdot Z'_L + I_{a1} \cdot 3 \cdot Z_f \\
&= Z'_L \cdot (I'_{a1} + I'_{a2}) + Z'_{L0} \cdot I'_{a0} + I_{a1} \cdot 3 \cdot Z_f \\
I'_{a1} + I'_{a2} &= I'_a - I'_{a0} \\
I'_a &= 3 \cdot I'_{a1} \\
V'_{SSKa} &= Z'_L \cdot I'_a - I'_{a0} \cdot Z'_L + I'_{a0} \cdot Z'_{L0} + I_a \cdot Z_f \qquad (2.35) \\
Z'_L &= (V'_{SSKa} - Z'_{L0} \cdot I'_{a0} - I_a \cdot Z_f)/(I'_a - I'_{a0}) \qquad (2.36)
\end{aligned}$$

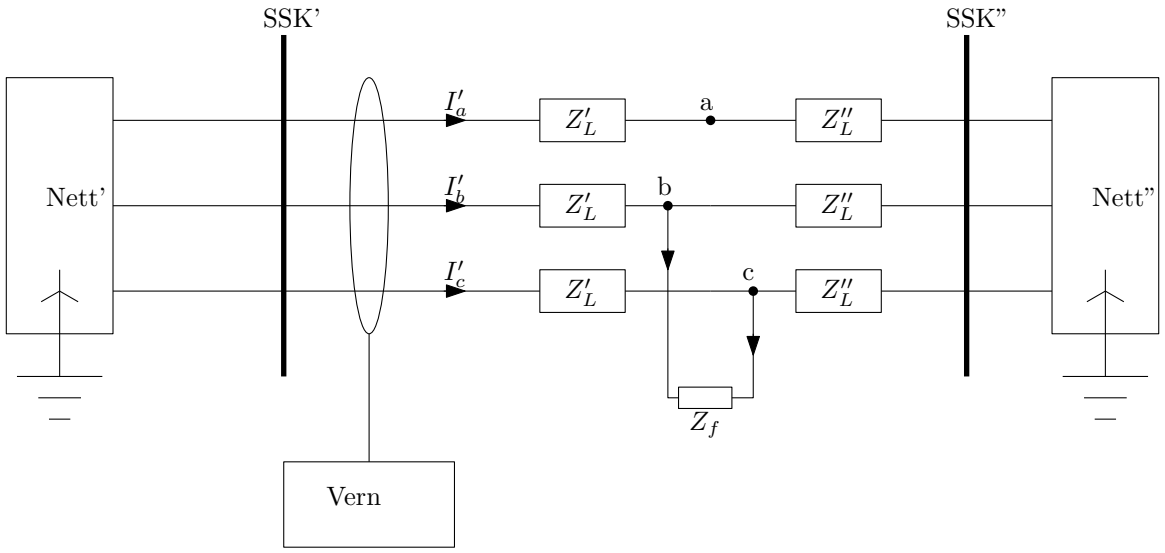
(2.36) viser korleis ein kan rekne ut positiv sekvens impedans, og dermed distanse, frå samleskinne til feil. Ulempa med denne metoden er at null sekvens impedans også er avhengig av feildistansen. Tek utgangspunkt i (2.35).

$$\begin{aligned}
I'_{a0} \cdot (Z'_{L0} - Z'_L) &= I'_{a0} \cdot (Z'_{L0} - Z'_L) \cdot \frac{Z'_L}{Z'_L} = k \cdot I'_{a0} \cdot Z'_L \\
k &= \frac{Z'_{L0} - Z'_L}{Z'_L} = \frac{Z_{L0} - Z_L}{Z_L} \qquad (2.37) \\
V'_{SSKa} &= Z'_L \cdot I'_a + k \cdot I'_{a0} \cdot Z'_L + I_a \cdot Z_f \\
Z'_L &= \frac{V'_{SSKa}}{I'_a + k \cdot I'_{a0}} - \frac{I_a \cdot Z_f}{I'_a + k \cdot I'_{a0}} \qquad (2.38)
\end{aligned}$$

Forholdet mellom null og positiv sekvens impedans er at begge er proporsjonale med lengda til linja. (2.37) erstattar den distanseavhengige impedansen i (2.36) med forholdet mellom null og positiv sekvens impedans, som er konstant og kjent. Andre ledd i (2.38) tek for seg endring i målt impedans som kjem av impedans mellom fase og jord, Z_f. Feilmotstanden og feilstraumen er ukjent for vernet, da dette kun måler eitt av to bidrag($I_a = I'_a + I''_a$). Dette leddet kan løysast ved effektmåling i vernet[11].

2.4.2 Fase - fase (3stk)

Totalt er det 3 mulige kombinasjonar av fase-fase feil; a-b, b-c og a-c.



Figur 2.15: Fase b - fase c feil

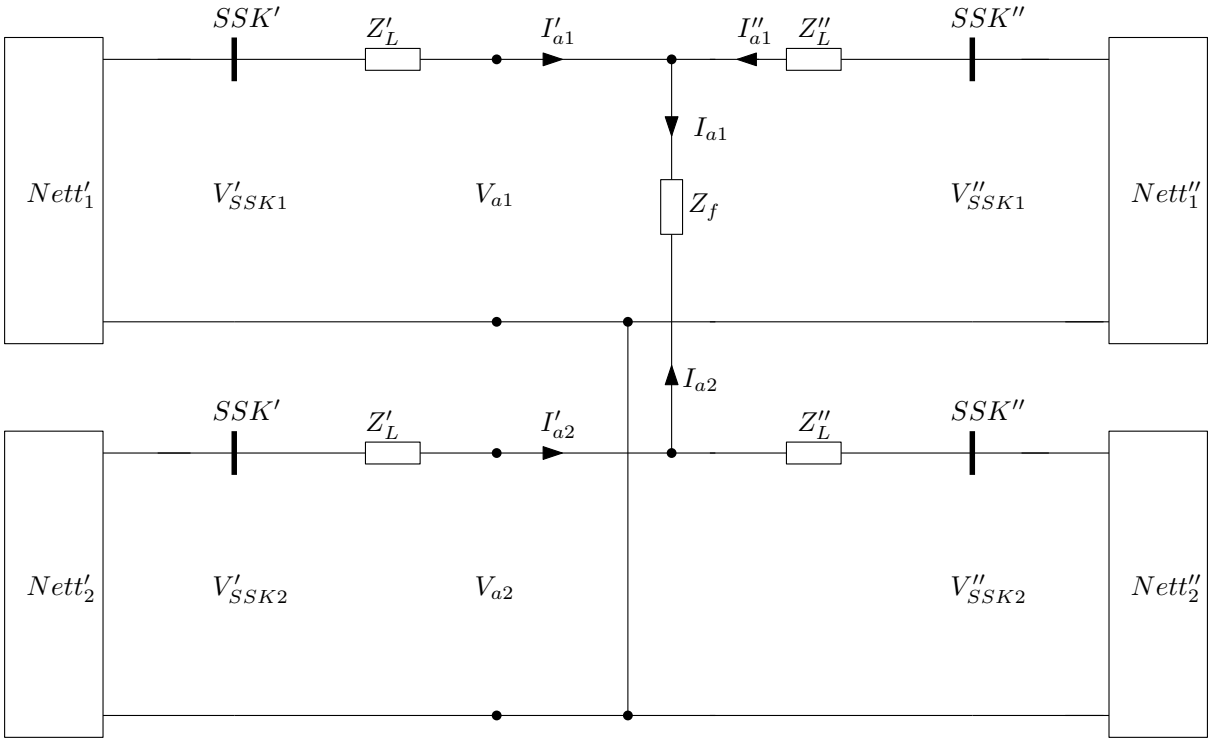
Ved feil mellom to fasar vil straumen i den friske fase vere lik null. Dei sjuke fasane er kopla i serie gjennom feilen og vil såleis ha same straum, med motsatt retning.

$$\begin{aligned} I_a &= 0 \\ I_b &= -I_c = I_f \\ V_b - V_c &= I_f \cdot Z_f \end{aligned} \tag{2.39}$$

Ved å kombinere (2.33) og (2.39) vil $I_{a0} = 0$ og $I_{a1} = -I_{a2}$. Difor vert sekvensnettverket kobla saman i parallell som i figur 2.16. Kretsanalyse av figuren gir

$$\begin{aligned} V'_{SSK1} &= V_{a1} + I'_{a1} \cdot Z'_L \\ V_{a1} &= I_{a1} \cdot Z_f - I'_{a2} \cdot Z'_L - V'_{SSK2} \\ V'_{SSK1} &= I'_{a1} \cdot Z'_L - I'_{a2} \cdot Z'_L + I_{a1} \cdot Z_f - V'_{SSK2} \\ Z'_L \cdot (I'_{a1} - I'_{a2}) &= V'_{SSK1} - V'_{SSK2} - I_{a1} \cdot Z_f \\ Z'_L &= \frac{V'_{SSK1} - V'_{SSK2}}{I'_{a1} - I'_{a2}} - \frac{I_{a1} \cdot Z_f}{I'_{a1} - I'_{a2}} \end{aligned} \tag{2.40}$$

Likning (2.40) viser utrekning av feilimpedans ved hjelp av sekvenskomponentar. Andre ledd kjem av impedansebidrag frå feilmotstanden, som må subtraherast frå linjeimpedan-



Figur 2.16: Samankopling av sekvensnettverk, fase-fase feil

sen. Ved hjelp av (2.33) kan symmetriske verdiar omgjerast til fasverdiar.

$$Z_f = 0 \tag{2.41}$$

$$\begin{aligned} V'_{SSKb} &= V'_{SSK0} + \alpha^2 \cdot V'_{SSK1} + \alpha \cdot V'_{SSK2} \\ V'_{SSKc} &= V'_{SSK0} + \alpha \cdot V'_{SSK1} + \alpha^2 \cdot V'_{SSK2} \end{aligned}$$

$$V'_{SSKb} - V_{SSKc} = (V'_{SSK1} - V'_{SSK2}) \cdot (\alpha^2 - \alpha) \tag{2.42}$$

Tilsvarande framgangsmåte for straumkomponentane gir

$$I'_{SSKb} - I_{SSKc} = (I'_{a1} - I'_{a2}) \cdot (\alpha^2 - \alpha) \tag{2.43}$$

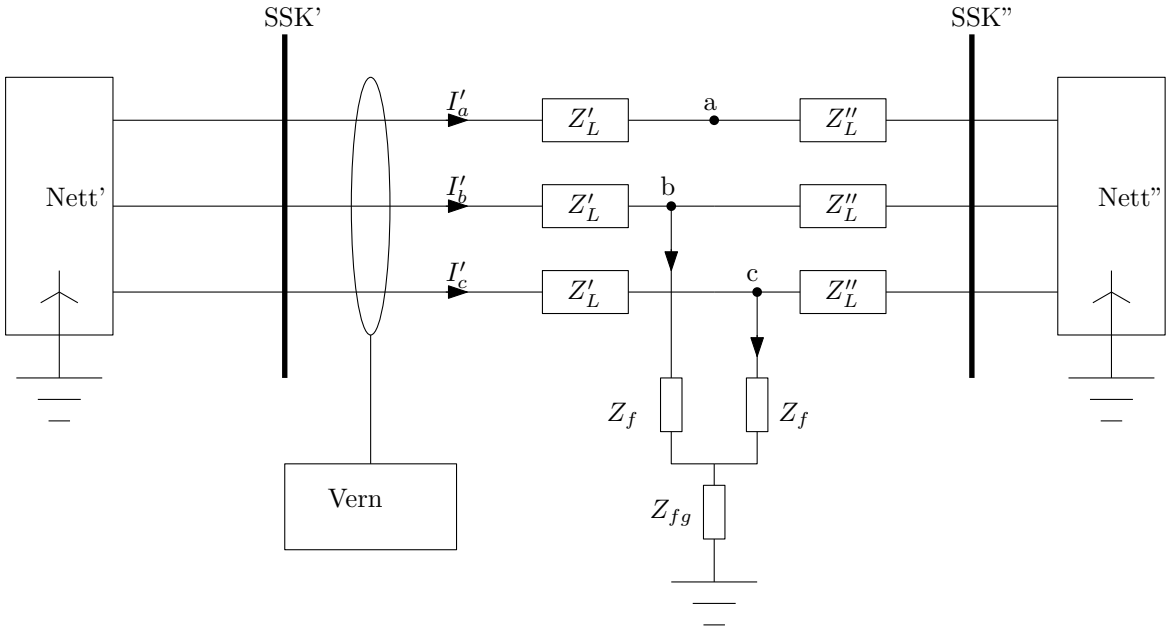
Kombinasjon av (2.40) - (2.43) gir

$$Z'_L = \frac{V'_{SSKb} - V'_{SSKc}}{I'_{SSKb} - I_{SSKc}} \tag{2.44}$$

Som er linjeimpedans ved feil mellom fase b og c utan overgangsmotstand. Tilsvarande likning er gjeldane for alle typar fase-fase feil.

2.4.3 Fase - fase - jord (3stk)

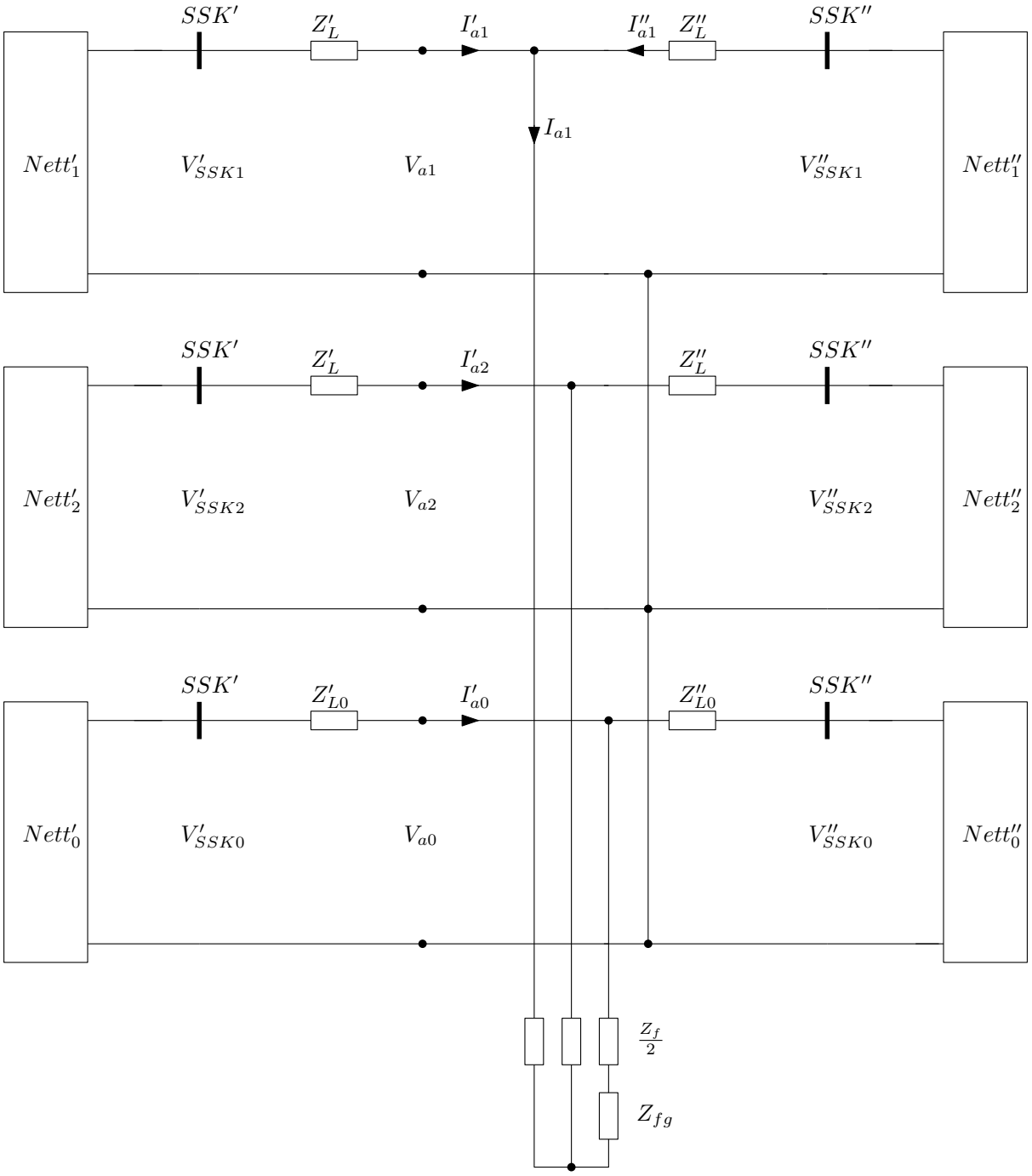
Fase-fase-jord er som fase-fase feil, men grunna kopling mot jord vil det også vere bidrag frå null-sekvens systemet.



Figur 2.17: Fase b - fase c - jord feil

Kretsanalyse av figur 2.18 viser at metoden for kalkulasjon av linjeimpedans ikkje er forskjellig frå den brukt ved fase-fase feil(Likning (2.40)). Forskjellen ved introduksjon av jordingsimpedans, Z_{fg}, er at nullstraumen gir eit bidrag til I_{a1}. Framgangsmåten for å finne linjeimpedansen, Z'_L, når jording og feilimpedans er lik null, er som ved fase-fase feil.

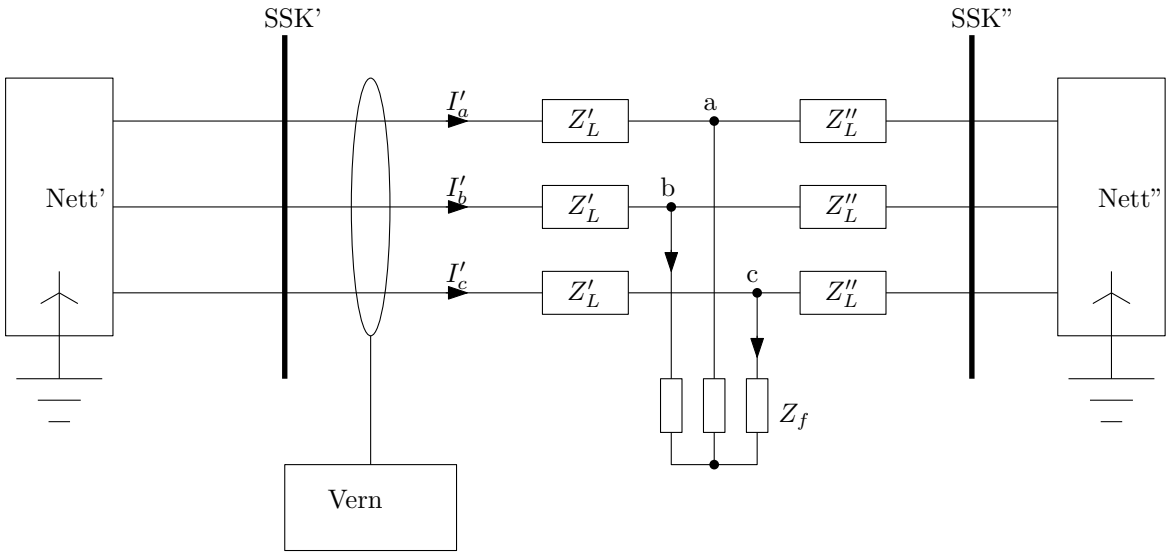
$$Z'_L = \frac{V'_{SSKb} - V'_{SSKc}}{I'_{SSKb} - I'_{SSKc}} \tag{2.45}$$



Figur 2.18: Sekvensnettverk, fase b - fase c - jord feil.

2.4.4 Fase - fase - fase (1stk)

Feil i alle tre fasane gir symmetrisk kortslutning. Symmetriske komponentar er difor ikkje naudsynt.



Figur 2.19: Trefase symmetrisk feil

Linjeimpedans kalkulert ved fase a vert då

$$\begin{aligned} I'_a &= \frac{V'_{SSKa}}{Z'_L + Z_f} \\ Z'_L &= \frac{V'_{SSKa}}{I'_a} - Z_f \end{aligned} \tag{2.46}$$

som også kan skrivast som

$$\begin{aligned} V'_{SSKa} - V'_{SSKb} &= Z'_L \cdot (I'_a - I'_b) + Z_f(I'_a - I'_b) \\ Z'_L &= \frac{V'_{SSKa} - V'_{SSKb}}{I'_{SSKa} - I'_{SSKb}} - Z_f \end{aligned} \tag{2.47}$$

Om ein ser vekk frå feilimpedansen vert linjeimpedansen ved trefase feil kalkulert på same måte som ved fase-fase og fase-fase-jord feil.

2.4.5 Oppsummering feiltypar

Om ein ser vekk frå feilimpedans vil målt linjeimpedans vere uavhengig av type feil, med unntak av fase-jord feil. Dette gjer at ein treng seks vern for total beskyttelse av linja. Tre av verna skal verne mot jordfeil(2.38), og treng difor ei anna innstilling på utløysarimpedans enn verna som skal verne mot feil der det er meir enn ein fase involvert((2.44),(2.45) og (2.47)). I moderne impedansvern kan ein ta alle måledata inn på eitt vern. Det blir då opp til vernet å skilje mellom to typar karakterstikk; ein for einfasa jordfeil og ein for andre feil.

2.4.6 Effektpendling

Ved normal drift opererer kraftsystemet med ein frekvens og ei spenning innanfor visse grenseverdiar. Ved ein forstyrrelse kan dette endre seg, og ein kan få effektpendling i nettet. Forstyrrelsar som kan forårsake dette kan til dømes vere fra-/tilkopling av ein generator og/eller ei last[19].

Den elektriske effekten levert frå generator til last er gitt av

$$P_e = \frac{E_g \cdot E_l}{X} \cdot \sin \delta \tag{2.48}$$

der E_g er indusert spenning på generator, E_l lastspenning, X reaktans mellom generator og last og δ vinkeldifferansen mellom E_g og E_l.

Den mekaniske effekten til generatoren, P_m, er gitt av turbinen. Ved normal drift vil denne vere lik den elektriske effekten P_e. Ved ein forstyrrelse i nettet vil lasta i nettet endre seg momentant, og dermed også behovet for elektrisk effekt, P_e. Den mekaniske effekten, P_m, kan ikkje endre seg momentant. Slik vil det verte ein differanse mellom mekanisk effekt inn i, og elektrisk effekt ut av, generatoren.

$$P_a = P_m - P_e \tag{2.49}$$

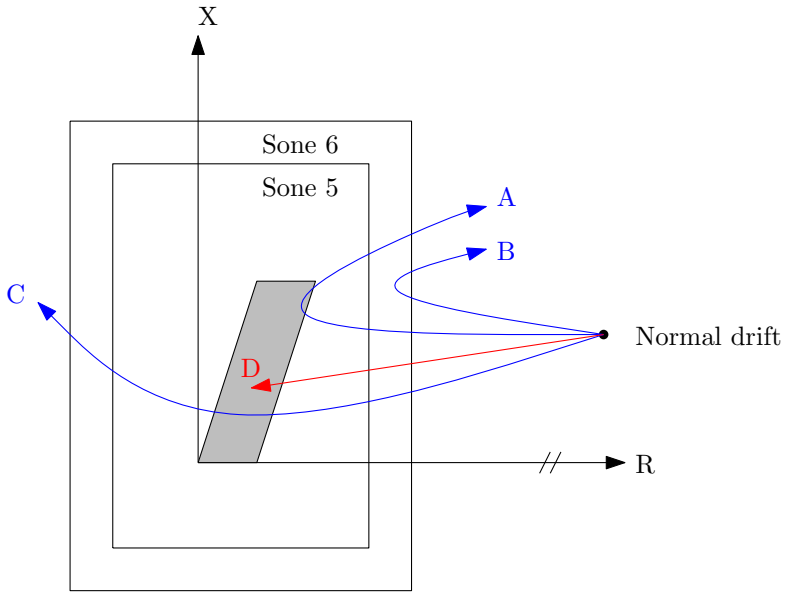
der P_a er differansen mellom tilført mekanisk effekt og avgitt elektrisk effekt. Dersom denne differansen er positiv, vil generatoren akselerere. Er den negativ vil den deakselerere. Frå ei slik fartsendring kan det vere to utfall, avhengig av det tilknytta elektriske nettet og maskinas kontrollsystem. Eit utfall er at den mekaniske effekten vert tilpassa behovet for elektrisk effekt og at maskina då stabiliserar seg på eit nytt likevektspunkt. Det andre utfallet er at maskina akselerar så mykje at den ikkje klarar å stabilisera seg, og til slutt svingar ut av fase med resten av nettet[20].

I begge overnemnte tilfeller kan den målte impedansen på overføringslinja mellom maskin og nett svinge innafor utkoplingssonene til relévernet. Ved svinging(effektpendling) er det ikkje alltid ein ynskjer ei utkopling av alle vern som detekterar denne svinginga. Det kan vere stabilitetsgrunnar til å selektivt kople ut deler av nettet, eller ikkje kople ut i det heile. Utfordringa vert då å skilje mellom stabil svinging, ustabil svinging og feil.

Då effektpendling alltid er ein symmetrisk hendelse, bør blokkering av vern alltid opphøyre så fort det er usymmetri/jordfeil i målepunktet[4, s.250]. Tradisjonelt sett

vert effektpendlingsblokkering oppheva dersom det oppstår negativ sekvens straum i målepunktet[23].

Den tradisjonelle metoden for å skilje mellom feil og effektpendling, er å måle kor raskt impedansen endrar seg innanfor visse grenser/soner.



Figur 2.20: Effektpendling

Figur 2.20 viser eit døme på impedansen ved stabil effektpendling(A og B), ustabil effektpendling(C) og feil(D). Ved feil vil impedansen endre seg tilnærma momentant, medan impedansen ved effektpendling endrar seg gradvis over tid. Tida vert i dette tilfellet målt mellom fra impedansen kjem inn i sone 6 til han går ut av sone 6 eller inn i sone 5. Dersom impedansen kjem inn i sone 5 og tida er større enn ein innstillt verdi, vil vernet detektera effektpendling. Grå sone er utkoplingssone for impedansvernet. For å skilje mellom stabil og ustabil effektpendling kan vernet måle om det er forteiknsendring på resistansen når impedansen forlet sone 6 i forhold til når han kom inn i sone 6. Om impedansen har skifta forteikn, er det ei ustabil effektpendling(C)[19][7, s.141].

Med muligheitane moderne numeriske vern gir, er det i dag mange måtar ein kan detektera effektpendling. Det eksisterar alt frå metodar som krev grunding analyse av nettet på førehand for innstilling av vernet[20], til metodar der brukaren ikkje treng stille inn vernet i det heile[21][22].

Kapittel 3

Resultat - modellar

3.1 MODELS - bruk i ATPDraw

I dette delkapittelet vert dei vanlegaste utfordringane opplevd under utarbeiding av denne oppgåva, ved bruk av ATPDraw i kombinasjon med MODELS, forklart. For generell bruk av ATPDraw, sjå tilhøyrande manual [24]. For grunnleggande bruk av programmeringsspråket MODELS, sjå manual [25].

3.1.1 Lage ein ny modell

For å lage ein ny modell må ein høgreklikke i vindauga der ein byggjer simuleringskretsen. Deretter vel ein MODELS → default model. For å endre kjeldekoden til modellen lyt ein fyrst dobbelklikke på ikonet til modellen, så velge knappen *Edit*. Før modellen kan takast i bruk er det også viktig å gi han eit navn. Dette kan gjerast i tekstboksen *Use As*.

3.1.2 Sette rett inngang på modellen

Modellens inngang må setjast alt etter om modellen skal ha straum, spenning eller andre typar signal som inngangsdata. Dette gjerast ved å dobbeltklikke direkte på inngangen til modellen(svart runding på venstre side). Sjølv om data til dømes er henta frå ein straumprobe, vil modellen fortsatt måle spenning dersom inngangen ikkje er satt til *INPUT current*. Ved oppsett av ein ny modell er standardvalget *INPUT voltage*.

3.1.3 Bruke fleire modellar i serie

Dersom ein modell skal nytte seg av inngangsdata frå ein anna, må inngangen på modellen setjast til *INPUT MODELS*. Samtidig må ein sikre at den modellen som skal forsyne den neste med data, vert berekna fyrst. Dette kan gjerast ved hjelp av *ORDER* parameteren

i modell-dialogen. Dersom ein modell har order 0 vil den verte berekna fyrst, order 1 som nummer to osb. Denne funksjonen må også aktiverast i ATP-settings menyen (hurtigtast F3) under Format-fana og valget *Sorting by order*.

3.1.4 Eksport av simuleringsdata

Dersom det skal eksporterast simuleringsdata i klarttekst til for eksempel excel eller COMTRADE, kan dette gjerast på fleire måtar. Under arbeid med denne rapporten vart det nytta eit program som heiter ATPDesigner. For at dette programmet skal kunne lese data etter simuleringa, må det gjerast ei endring i fila startup *startup*, som ligg i

```
\<installDIR>\EEUGXX\GNUATP\startup
```

Det som då må endrast er attributten NEWPL4, som må setjast til 1. Etter at det er gjort kan ein køyra simuleringa i ATPDraw og importera den resulterande *.pl4 fila til ATPDesigner. Når fila er importert i ATPDesigner kan den eksporterast til tekst eller COMTRADE format.

3.2 RMS

Viser til kapittel 2.1.2.

Likning (2.1) viser at effektivverdiar kan bereknast ved hjelp av integralet til signalet over ein gitt periode. Ved utarbeiding av RMS-komponenten i ATPDraw er det skissert to løysingar.

3.2.1 Analog

Den fyrste komponenten er kalla $aRMS$, som står for analog RMS. $aRMS$ bereknar effektivverdien i eit tidsrom, T, angitt av brukar på fylgjande måte:

$$RMS_a(t) = \sqrt{\frac{1}{T}\int_{t-T}^{t} x(t)^2 dt} \tag{3.1}$$

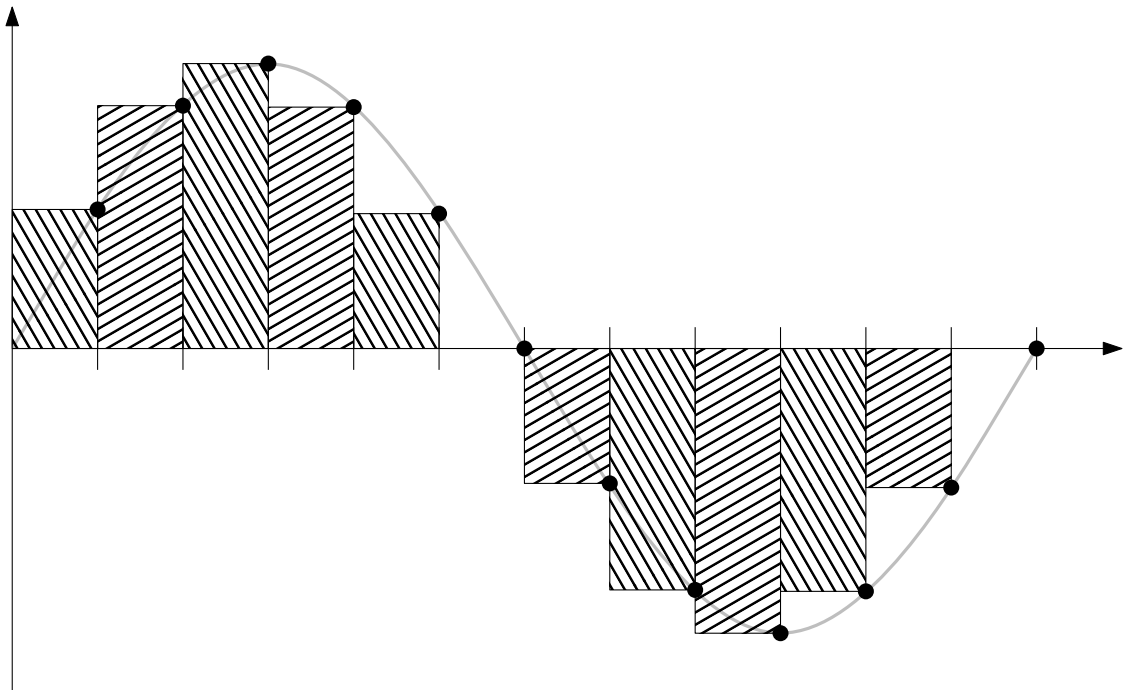
Tilpassa diskré signal:

$$aRMS(n) = \sqrt{\frac{1}{N_a}\sum_{n=0}^{N_a-1} x(n)^2} \tag{3.2}$$

Der $x(0)$ er den siste prøven. Mengd prøvar, N, vert berekna utifrå storleiken på tidssteget i simuleringa og tidsrommet T:

$$N_a = \frac{T}{timestep} \tag{3.3}$$

Denne modellen brukar såleis verdien som er simulert ved kvart tidssteg. Figur 3.1 viser korleis integralet vert kalkulert ved hjelp av verdiar henta ved kvart simuleringssteg. Dei svarte prikkane er simuleringssteg. Koden er vist i vedlegg A.1. T vert heretter kalla *tidskonstant* når den vert omttala i samanheng med aRMS-modellen.



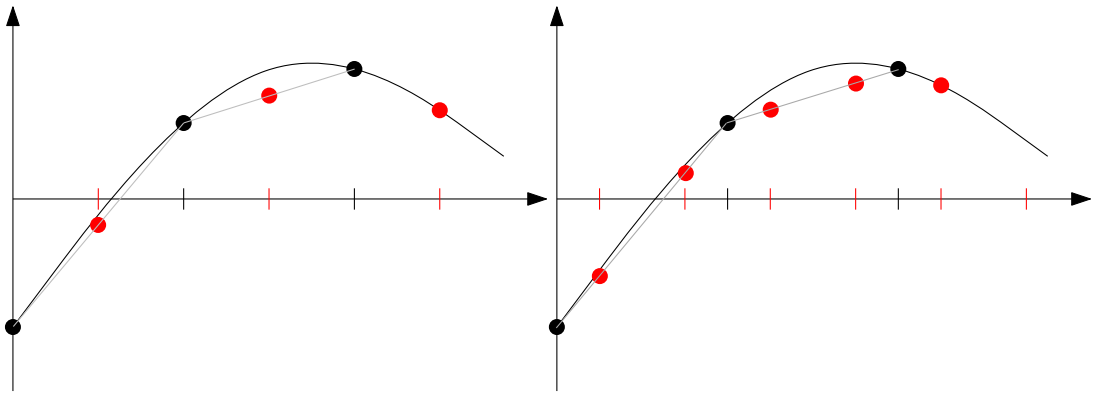
Figur 3.1: aRMS, tidsrom(T) = 1 periode

3.2.2 Digital

Den andre løysinga bereknar RMS-verdien av siste periode av fundamentalfrekvensen. Vert kalla $dRMS$(RMS digital). Fundamentalfrekvensen vert angitt av brukar. I staden for å endre tidsrommet til integralet, kan brukaren endre oppløysinga på integralet over ein periode. Dette kan gjerast ved hjelp å justere mengda prøvar. For at angitt mengd prøvar skal gå opp med ein periode, må avstanden mellom kvar dataprøve justerast. Dette vert gjort ved å multiplisere med ein avstandsfaktor, Δ_d.

$$dRMS(t) = \sqrt{\frac{1}{N_d} \sum_{n=0}^{N_d-1} x(n \cdot \Delta_d)^2} \tag{3.4}$$

Ved bruk av sjølvvald oppløysing er det alltid ein muligheit for at brukaren vel ei så stor oppløysing at det vert nytta avhengige prøvar. Ein prøve er avhengig dersom det er to eller fleire prøvar mellom kvart simuleringssteg. Dataprøvar som hamnar utanfor simuleringssteget vert kalkulert ved hjelp av interpolasjon. Då vil denne prøven verte tilnærma ved hjelp av det simuleringssteget som er før, og det som er etter prøven. Dei avhengige prøvane som er innafor same simuleringssteg vert såleis kalkulert ut frå same utgangspunkt(figur 3.2).



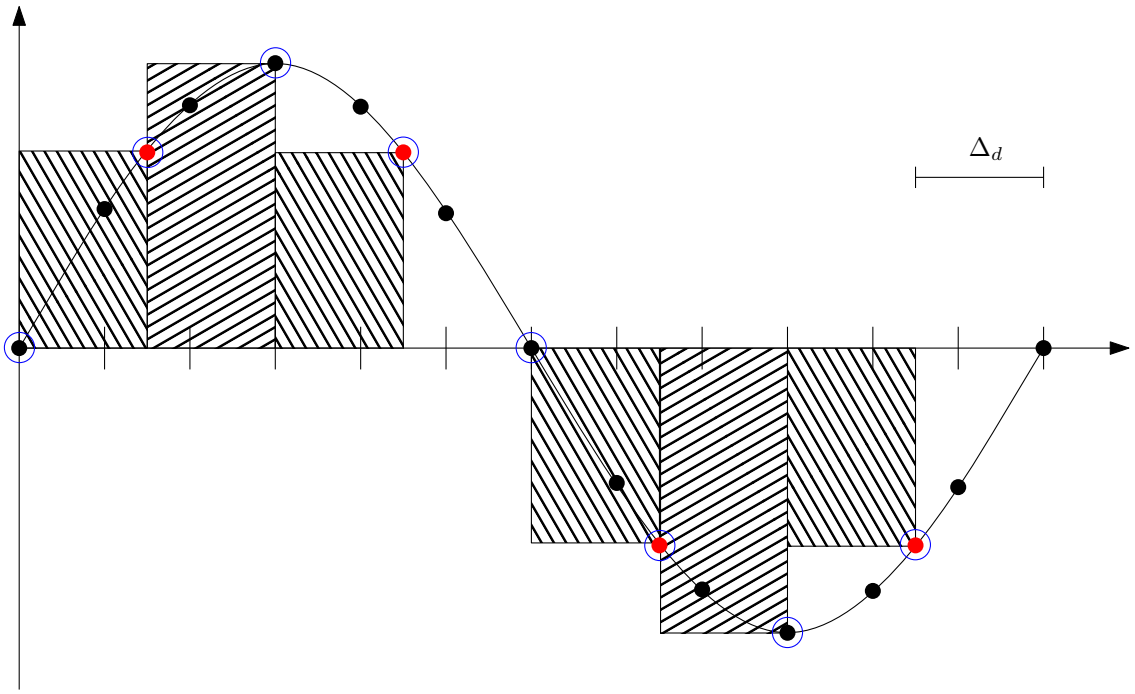
(a) Uavhengige prøvar (b) Avhengige prøvar

Figur 3.2: Uavhengige og avhengige prøvar

I figuren er simuleringsstega merka med svarte prikkar, medan dataprøvane nytta av modellen er merka med raudt. Modellen er satt opp slik at dersom brukaren vel for mange prøvar, vil mengda prøvar automatisk verte satt til maksimal mengd uavhengige prøvar. Dette gir også fordelar med tanke på simuleringstid, då færre prøvar gir færre kalkulasjonar. Maksimal mengd uavhengige prøvar per periode av fundamentalfrekvensen er gitt av:

$$N_{maks} = \frac{1}{timestep \cdot f_{fundamental}} \tag{3.5}$$

Figur 3.3 er eit eksempel der oppløysinga redusert til 8 prøvar per periode, sjølv om det er 12 simuleringssteg per periode. Dei svarte prikkane er simuleringssteg, medan dei raude er verdiar kalkulert ved hjelp av interpolasjon. Prøvane som vert nytta til berekning av rms-verdi er blå ring rundt.



Figur 3.3: dRMS, $N_d = 8$

3.2.3 Hastigheitmodifisering ved simulering(h.mod.)

For å samanlikne hastigheit er det satt opp ei simulering der simuleringssteget er $10\mu s$, og total simuleringstid $0.15s$. Resultatet i tabell 3.1.

Tabell 3.1: Hastigheit $aRMS$ og $dRMS$

Type	Tidskonstant[s]	Frekvens[Hz]	Prøvar	Simuleringstid[s]
aRMS	0.02			26.161
	0.01			13.088
dRMS		50	10000	26.072
		50	128	1.903
		50	8	0.187

Simuleringstida er avhengig av maskinvaren på den datamaskina som køyrer simuleringa. Tidene gitt i tabell 3.1 vil difor ikkje vere den same frå maskin til maskin. Likevel viser tabellen forholdet i hastigheit mellom kvar komponent og konfigurasjon.

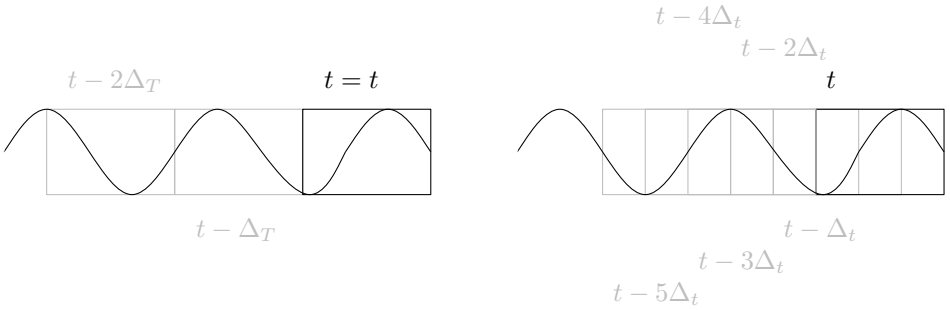
Når mengd prøver på $dRMS$ er 10000 er simuleringstida omtrent det same som ved aRMS med ein tidskonstant på $20ms$. 10000 prøvar er langt over N_{maks} = 2000. Difor vert mengda prøvar automatisk sett til 2000, noko som er den same mengda prøvar som vert nytta av $aRMS$ når tidskonstanten er ein periode ($20ms$). For å auke hastigheiten kan

også tidskonstanten på $aRMS$ reduserast ytterligare, men då vil det oppstå svingningar på rms-signalet.

Berekningar vil i begge dei ovanståande tilfella verte køyrd ved kvart tidssteg, uavhengig av mengd prøvar som skal nyttast. Dette sikrar at rms-verdien til ei kvar tid er oppdatert. For å ytterligare auka hastigheit er det lagt ved forslag til kode som aukar intervallet mellom kvar berekning. Koden er konstruert slik at ein ikkje mistar data, då berekningsintervalla overlappar kvarandre.

```
VAR
...
delta_T
...
INIT
...
-- dRMS --
delta_T := 1 / (f_base * n_max)
----------
-- aRMS --
delta_T := smoothing
----------
ENDINIT

EXEC

IF (t mod delta_T) < timestep THEN

    -- eksisterande kode kjem her --

ENDIF

ENDEXEC
...
```

Figur 3.4 viser dette grafisk. Ulempa med denne metoden er at rms-verdien «frys» mellom kvar berekning, ei tid tilsvarande storleiken på berekningsvindauga.



(a) Auka berekningsinterval (b) Berekning ved kvart simuleringssteg

Figur 3.4: Forskjell på auka og normalt berekningsinterval

Tabell 3.2: Simuleringsastigheit med h.mod. i forhold til utan h.mod.

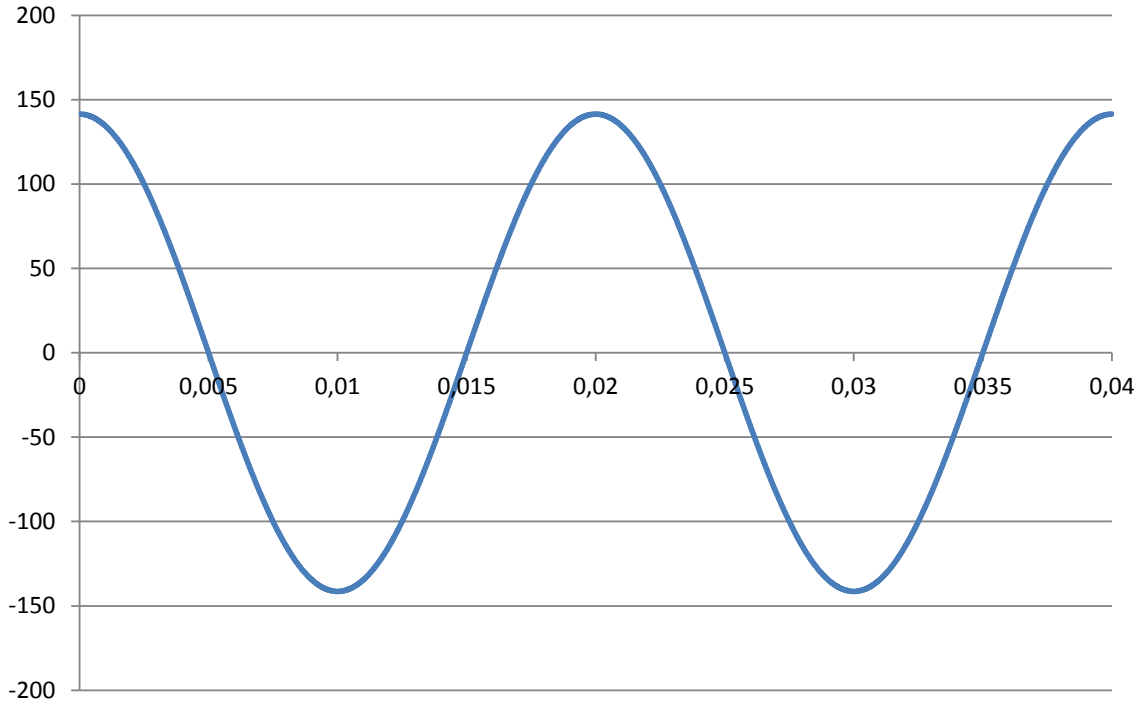
Type	Tidskonstant[s]	Frekvens[Hz]	Prøvar	$\frac{\text{tid utan h.mod.}}{\text{tid med h.mod}}$
aRMS	0.02			456
	0.01			256
dRMS		50	10000	1
		50	128	12
		50	8	4

Tabell 3.2 viser effekten av h.mod. Den har størst innverknad på aRMS-modellen. Hastigheitsauken er proporsjonal med storleiken på simuleringssteget og tidskonstanten. I tilfellet med tidskonstant på $20ms$ og simuleringssteg på $10\mu s$ var simuleringa 456 gonger raskare med hastigheitsmodifisering. Ved dRMS kan det også vere potensiale for hastigheitsauke, men denne minkar etter kvart som mengd prøvar nærmar seg N_{maks}. Ved dRMS og 8 prøvar er simuleringstida 4 gonger raskare med h.mod. i motsetning til 12 gonger raskare ved 128 prøvar. Grunnen til at det ikkje er så stor forskjell ved 8 prøvar er at då er simuleringstida allereie så kort at modellen ikkje lenger er den tyngste delen av simuleringa. Til samanlikning brukte simulering med h.mod. aktivert og 8 prøvar $45.7ms$. Når modellen vart fjerna tok same simuleringa $31ms$. Simuleringsprogrammet brukte altså 15ms ekstra tid med modellen, noko det også hadde gjort om modellen kun hadde innholdt enkle kalkulasjonar.

3.2.4 Resultat frå simulering

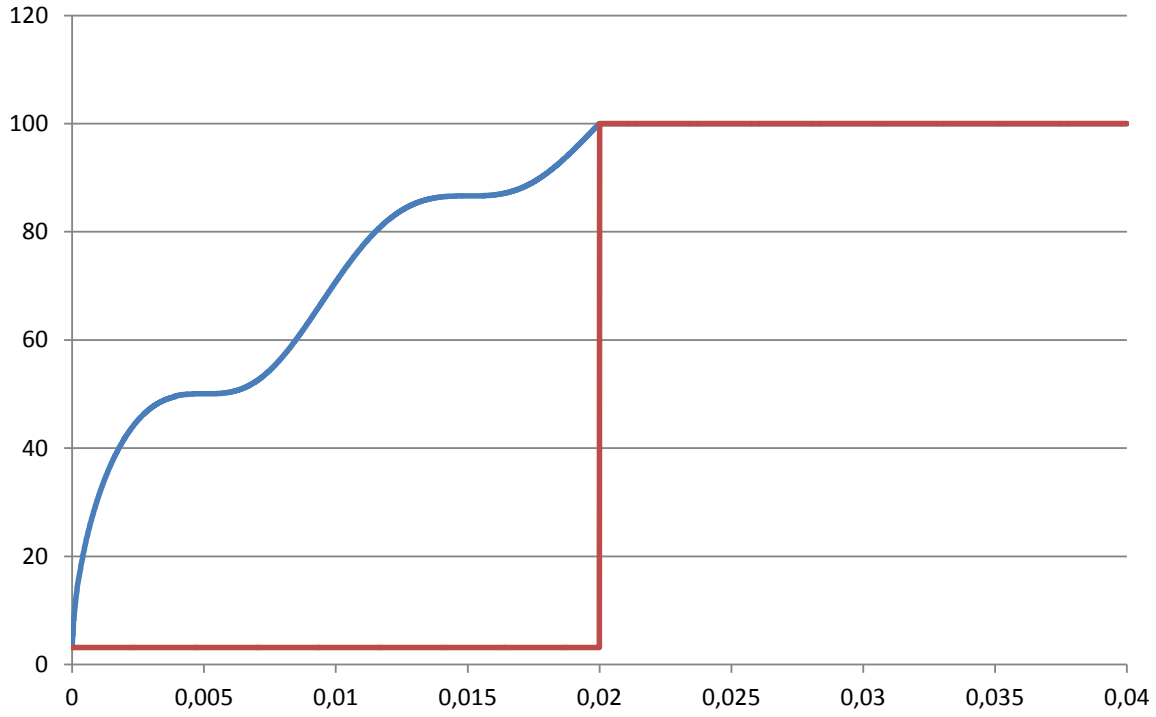
Vedlegg A.3 viser korleis dei ulike modellane bereknar utgangssignalet. Kvar modell er simulert i tre forskjellige situasjonar, med tre forskjellige parametersett. Totalt er det gjennomført 2 modellar x 3 situasjonar x 3 parametersett x 2(med/utan hastigheitsmodifikasjon)= 36 simuleringar. Kvar figur viser ein modell og eitt parametersett, der den raude grafen er med hastigheitsmodifikasjon(h.mod) og den blå utan.

Innsving Den fyrste situasjonen er oppstart av simulering. Frekvensen på inngangssignalet er 50Hz, simuleringssteget 10 μs og lengda på simuleringa 0.04s. Inngangssignalet er vist i figur 3.5.



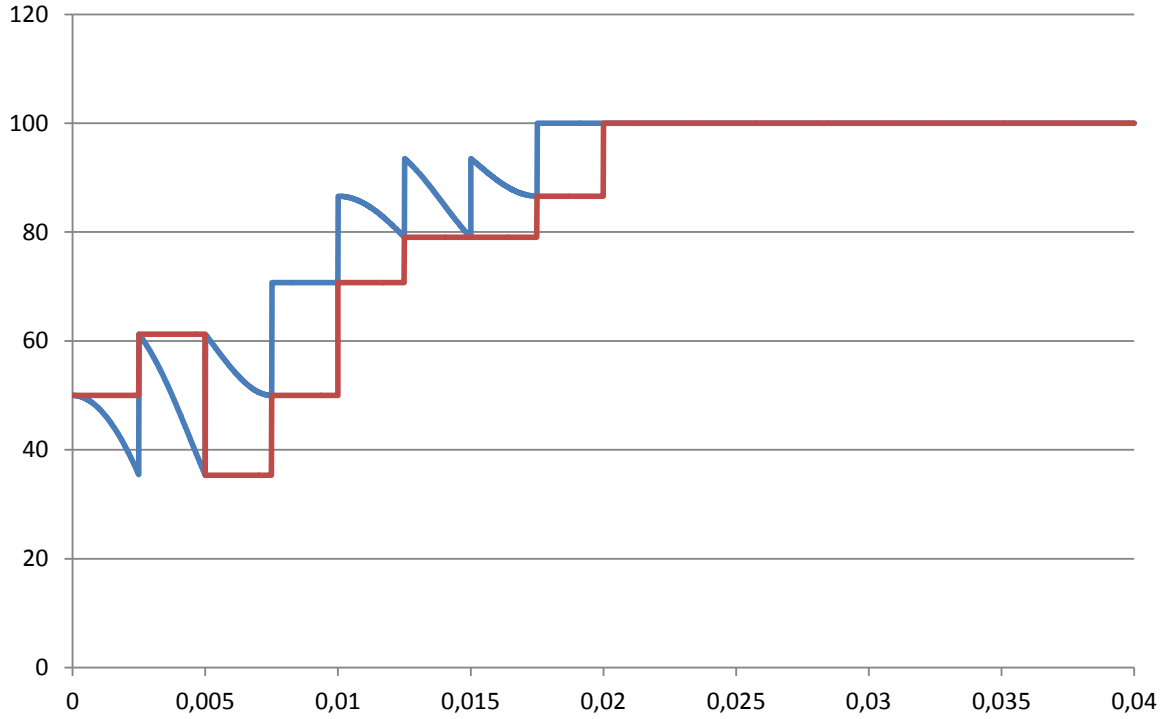
Figur 3.5: Inngangssignal - innsving

Figur A.1, 3.6 og A.2 viser resultatet frå aRMS, med aukande tidskonstant frå 10ms til 40ms. Figurane viser korleis resultatet utan h.mod. kontinuerleg aukar mot stabil verdi. Resultatet brukar ei tid lik tidskonstanten frå start til korrekt RMS verdi er nådd. Korrekt resultat vert også oppnådd samtidig om h.mod. vert nytta, men det vert ikkje registrert noko i tida mellom start og tidskonstanten. Dersom det er endringar i signalet mellom to berekningstrinn ved h.mod, vil dette ikkje verte registrert før det neste berekningstrinnet, og såleis skapa eit forseinka resultat samanlikna med modellen utan h.mod. Meir om dette kjem under steg- og impulssimuleringa.



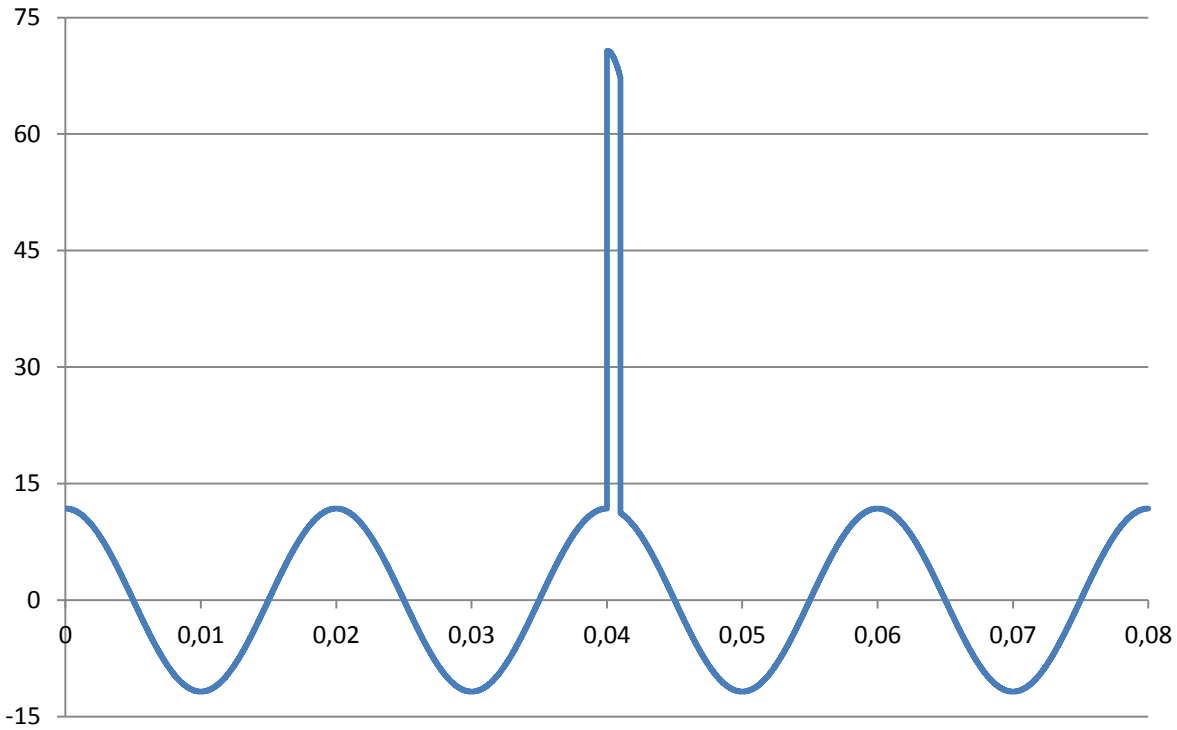
Figur 3.6: aRMS - $t = 0.02$ - raud: med h.mod - blå: utan h.mod

Resultata frå dRMS er vist i figur 3.7, A.3 og A.4. Frekvensen er satt til 50Hz, og mengd prøvar er 8, 128 og 10000. Ved 8 og 128 prøvar utgjer h.mod. liten forskjell på resultatet. Ved 10000 prøvar er det ikkje nokon forskjell på om h.mod. vert brukt eller ikkje. Dette kjem av at koden er satt opp på ein slik måte at det vert berekne ved kvart simuleringssteg uansett. Det er ved 10000 prøvar heller ikkje noko forskjell på resultatet frå aRMS med ein tidskonstant lik ein periode av $50Hz$ ($0.02ms$). H.mod. kan ut i frå fylgjande resultat med fordel nyttast i samband med dRMS for å spare simuleringstid.



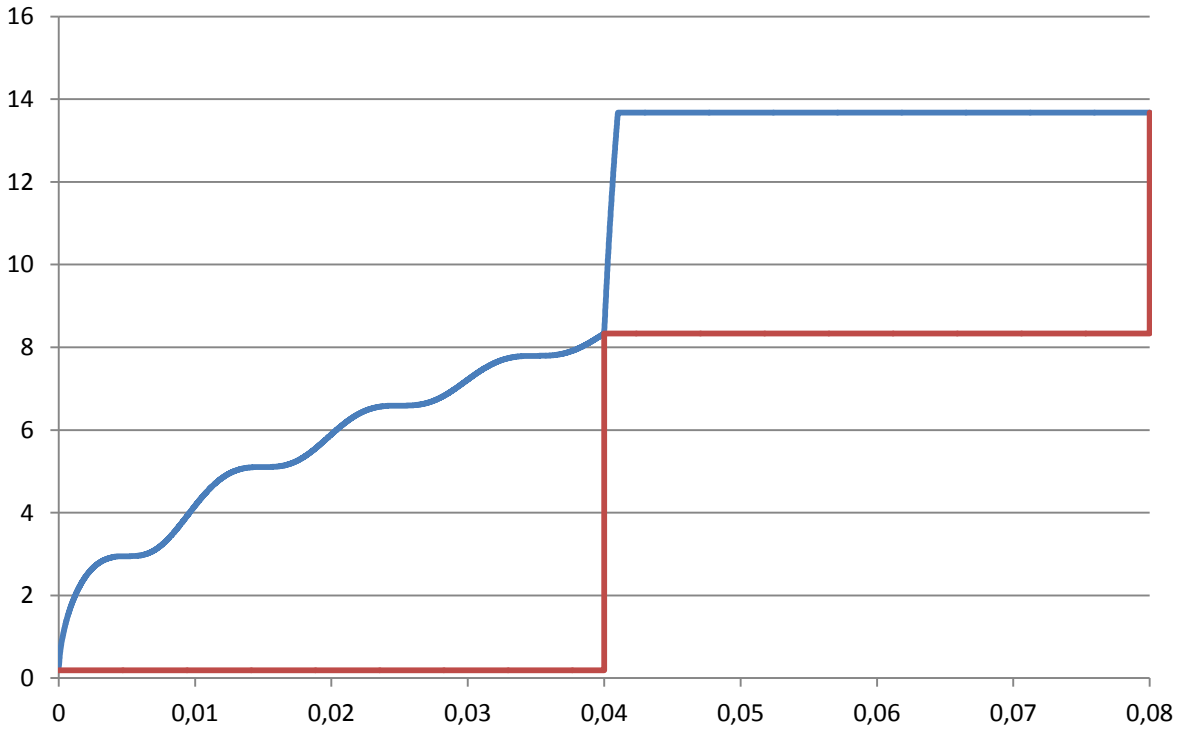
Figur 3.7: dRMS - $N = 8$ - $f_{base} = 50Hz$ - raud: med h.mod - blå: utan h.mod

Impuls I situasjon to er det sett på korleis resultata vert i forhold til eit impulssignal. Frekvensen på grunnsignalet er også her satt til $50Hz$, medan det ved $40ms$ kjem ein impuls som varar i $1ms$. Total simuleringstid er $80ms$ og tidssteget er $10\mu s$. Inngangssignalet er vist i figur 3.8.



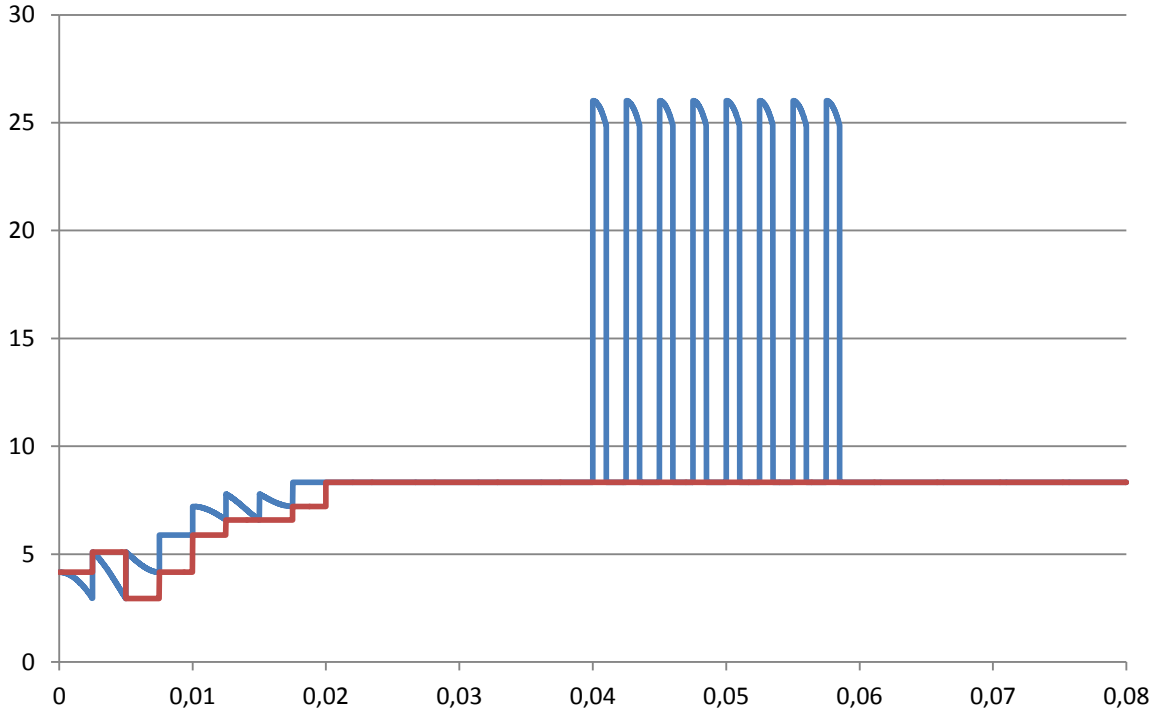
Figur 3.8: Inngangssignal - impuls

Forseinking ved bruk av h.mod. på aRMS modellen kjem tydeleg fram i figur A.5, A.6 og 3.9. Der det ikkje vert nytta h.mod. vert impulsen registrert med ein gang. I motsett tilfelle vert impulsen registrert med ei tidsforseinking lik tida mellom impuls og berekningsintervall. Ved desse simuleringane kjem impulsen på simuleringssteget etter berekningsintervallet. Difor er forsinkelsen lik tidskonstanten, t, til modellen. I figur 3.9 er forseinkinga så stor at det ikkje visar når signalet kjem tilbake til utgangspunktet. Då signalet er proporsjonalt med dei figur A.5 og A.6, vil det nå ordinær effektivverdi ved $120ms$. Hadde impulsen vore rett før eit berekningsintervall, ville forseinkinga vore mindre. Effektivverdien forårsaka av impulsen vert derimot mindre om tidskonstanten er stor, då denne vert fordelt ut over ei større mengd prøvar.



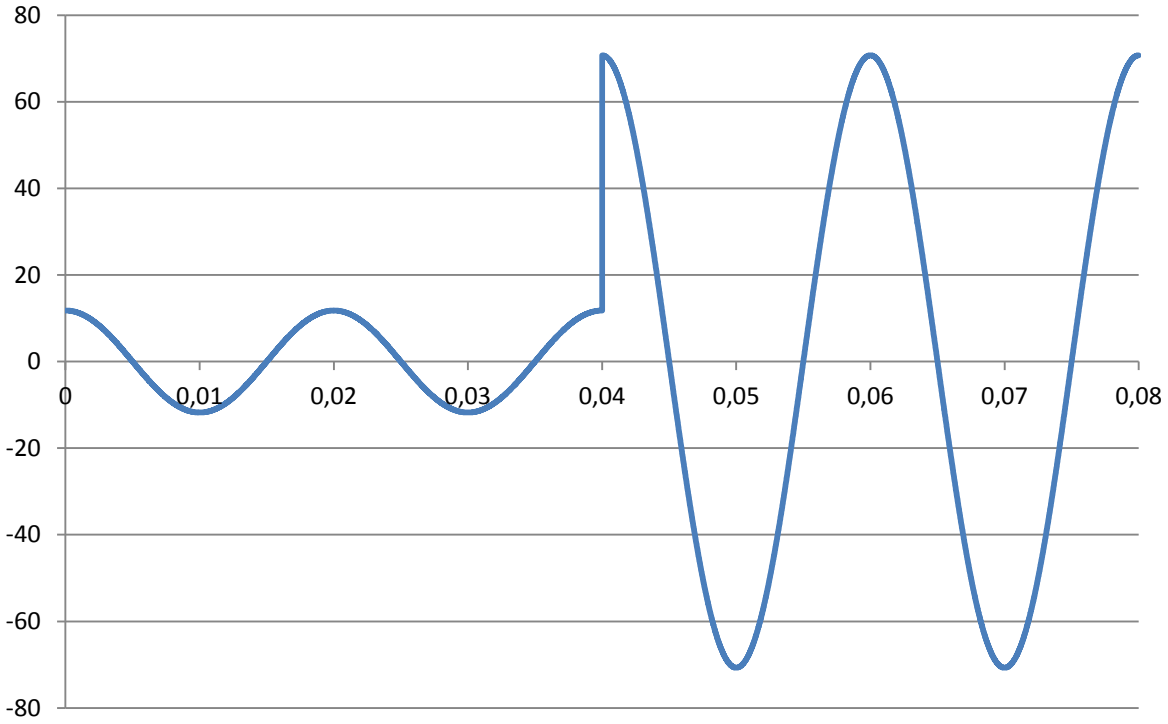
Figur 3.9: aRMS - $t = 0.04$ - raud: med h.mod - blå: utan h.mod

Figur 3.10, A.7 og A.8 viser dRMS modellens resultat på same signal. Ved 8 prøvar (3.10), og utan h.mod. vert det ei forholdsvis kraftig svinging på resultatet. Dette fordi impulsen er så kort at den ikkje vert teken med i berekninga ved kvart simuleringssteg. Med h.mod. aktivert vart det i dette tilfellet ikkje registrert noko impuls. Dette fordi impulsen var så kort at berekninga ikkje «traff» data som innehaldt impulsen. Dersom impulsen hadde vore lagt til eit anna simuleringssteg kunne det ved h.mod. aktivert vert muligheit for ein resultatimpuls av same typen som det er åtte av når h.mod. ikkje er aktivert. Ved 128 prøvar vert forskjellen med og utan h.mod. liten, samstundes som svinginga vert redusert. Ved 10000 prøvar er det ikkje noko forskjell med og utan h.mod, og resultatet er det same som ved aRMS med ein tidskonstant på 20ms.



Figur 3.10: dRMS - $N = 8$ - $f_{base} = 50Hz$ - raud: med h.mod - blå: utan h.mod

Steg Den siste situasjonen simulert er eit steg i inngangssignalet. Frekvens er $50Hz$, simuleringssteget $10\mu s$ og simuleringstid $80\mu s$. Signalet er vist i figur 3.11. Resultata(A.9 - A.14) her er proporsjonale med resultata under innsving.



Figur 3.11: Inngangssignal - steg

3.2.5 Oppsummering av rms-alternativa

Tabell 3.3: Hastigheit $aRMS$ og $dRMS$ oppsummert

Type	Tidskonstant[s]	Frekvens[Hz]	Prøvar	Simuleringstid[s]	med h.mod[s]
aRMS	0.02			26.161	0.0573
	0.01			13.088	0.0511
dRMS		50	10000	26.072	26.134
		50	128	1.903	0.159
		50	8	0.187	0.0468

aRMS Utan h.mod. er aRMS ein treg og omstendeleg måte å berekne rms-verdiar på. Likevel er det den sikraste og mest stabile, der alle simuleringssteg vert teke omsyn til. Ved bruk av h.mod. er denne varianten fullt på høgde med dRMS i hastigheit, men signalet «frys» mellom kvart berekningstrinn, som er lik tidskonstanten. I verste fall kan difor signalet verte forseinka med ei tid lik tidskonstanten. Fordelen med denne metoden i forhold til dRMS er at modellen ikkje under noken omstendigheitar går glipp av transiente hendelesar.

dRMS Utan h.mod. er denne løysinga raskare, og gir tilsvarande resultat som aRMS utan h.mod. Ulempa er at dersom oppløysinga er for lav, kan det oppstå svingingar på utgangssignalet ved impulsar. Høgare oppløysing vil gi jamnare signal, med lavare amplitude på svinginga. Med h.mod. aktivert er dRMS den modellen som har minst forsinkelse på resultatet i forhold til berekningstid. Ulempa er at modellen kan «hoppe over» transiente data om oppløysinga er for lav. Dersom brukaren veit tida på den kortaste transiente hendelsen, kan han sikre seg at denne vert med ved å setje oppløysinga på dataprøvane mindre enn tida på den transiente hendelsen. Med tanke på simuleringstid anbefalast dRMS med h.mod. aktivert i dei fleste tilfelle, der oppløysinga er tilpassa signalet.

3.3 Fouriertransformasjon

Det er skissert to forskjellige løysingar basert på fourier transformasjon. Den eine løysinga nyttar «Radix-2» algoritmen. Denne algoritmen lyt setjast opp spesifikt etter mengd harmoniske signalet skal filtrerast i. Det vart difor utvikla ein diskré fourier transformasjon algoritme for dei tilfella der brukaren ynskjer ei anna mengd harmoniske enn den gitt av «Radix-2» løysinga. Då filter ikkje er implementert i modellane, lyt brukaren ta omsyn til at inngangssignalets frekvenskomponentar ikkje overstig $f_{base} \cdot n_{harmoniske}$ for å unngå skygging.

3.3.1 Diskré Fourier transformasjon (DFT)

DFT-modellen er basert på likning (2.7). Inngangsparameterar er frekvensen til den fyrsteharmoniske og mengd med harmoniske ynskt analysert. Dersom den fyrsteharmoniske er satt til $50Hz$ vil den andreharmoniske vere $100Hz$ osb.

Mengd harmoniske som skal analyserast er valfritt mellom 0 og 26. Talet 26 kjem av ei avgrensing i tabellstorleiken på utgangssignalet. Dersom brukaren vel fleire harmoniske enn dette, vil modellen automatisk setje mengd harmoniske til 26 for å unngå feil under simulering.

Simuleringshastigheita vert lengre etter kvart som ynskt mengd harmoniske aukar. Fordelen med å auke mengd harmoniske er at bandbredda vert større. Bandbredda til DFT-modellen er gitt av likning (2.3), som omsett i mengd harmoniske og grunnfrekvens gir:

$$B = n_{harmoniske} \cdot f_{base} \tag{3.6}$$

I grunnmodellen av DFT-blokka(vedlegg B.1.1), er utgangssignalet gitt i skalert rektangulærform etter likning (2.14). Dette er gjort fordi fleire av modellane kan nytte seg av dette formatet direkte. Såleis kan ein spare ein del tid på å unngå omrekning frå rektangulærform til polarform og vice versa.

Dersom brukaren også ynskjer å måle effektivverdi og fase direkte frå modellen, er det skissert ei løysing der utgangsdata er effektivverdi og vinkel(vedlegg B.1.2). *Ved slutten av studiet vart det oppdaga ein funksjon, atan2(), i MODELS som handterar kvadrantar* Vinkelreferansen er eit cosinus-signal med frekvens lik den harmoniske og vinkel 0 når t = 0. Vinkelreferanse til eit m harmonisk signal vist under.

$$angle_{ref}(m) = \cos(m \cdot 2 \cdot \pi \cdot f_{base} \cdot t) \tag{3.7}$$

I tillegg kan det, som ved rms-modellane, nyttast hastigheitsmodifisering. Prinsippet er også her at ein aukar tida mellom kvar berekning jamfør figur 3.4. Forskjellen i forhold til eksisterande modellar vert då at ein legg all kode i EXEC blokka inn i ei if løkke som vist under:

```
EXEC

IF t mod delta_T <= timestep

    -- Eksisterande kode

ENDIF

ENDEXEC
```

delta_T er allereie definert i kjeldekoden, så det er ikkje naudsynt å setje tidssteget på nytt. Under følgjer ein tabell av berekningstid med og utan hastigheitsmodifisering. Simuleringstid er satt til $50ms$ og simuleringssteget er på $10\mu s$.

Tabell 3.4: Hastigheit DFT grunnmodell

Harmoniske	Simuleringstid[s]	S.tid med h.mod[s]
26	50.791	1.563
17	21.891	0.531
8	5.063	0.172

3.3.2 «Radix 2» fast Fourier transform (FFT)

Dei fleste reelle relévern nyttar seg av fyrste- og i nokre tilfelle andre- og tredjeharmoniske. Det er difor laga ein modell av ein «Radix 2» FFT algoritme som bereknar desse. Denne modellen gir same resultat som ein tilsvarande DFT, men er mykje raskare. Kjeldekoden er basert på informasjonen i kapittel 2.1.3. Bandbredda er $150Hz$, så filter bør nyttast om det er mistanke om større frekvenskomponentar i signalet. Kjeldekoden er vist i vedlegg B.2.1.

Utgangssignala er i rektangulærform og skalert til rms-verdi. Dersom ein reknar ut vinkelen til kvar komponent, vil denne vere i forhold til eit cosinus-signal med tilsvarande frekvens og vinkel 0 grader. Modellen nyttar ein data-parameter. Denne er frekvensen til den fyrsteharmoniske, f_{base}. Utgangane er organisert på same måte som ved DFT modellen, der det er ein utgang for dc-komponenten og to utgangar for dei harmoniske. Begge dei harmoniske utgangane er tabellar, som kvar inneheld tre data. Den eine tabellen er for reelldelen, medan den andre er for imaginærdelen. Det fyrste innsteget i tabellen er den fyrsteharmoniske, osb.

Modellen er hardkoda etter figur 4-5 i [2]. Dette er gjort for å unngå bruk av løkker og modulusfunksjonar, slik at mengda operasjonar vert lavast muleg. Alle multiplikasjonar som ikkje er avhengige av data som endrar seg under simuleringa, vert gjort på førehand. Til slutt vert signala skalert til effektivverdiar etter likning (2.14).

I vedlegg B.2.2 er det skissert forslag til ein FFT modell som konverterar utgangssignala frå rektangulær til polarform. Dette kan vere ein fordel dersom modellen skal brukast direkte, og ikkje som ein del av eit vern. Her kan brukaren velje om fasen til signalet skal vere i grader eller radianar ved hjelp av parameteren *degrees*.

Ved simulering er det også nytta hastigheitsmodifisering. Dette reduserar bereknigstida kraftig. Dette er implementert på fylgjande måte:

```
EXEC

IF t mod delta_T <= timestep

    -- Eksisterande kode

ENDIF
```

```
ENDEXEC
```

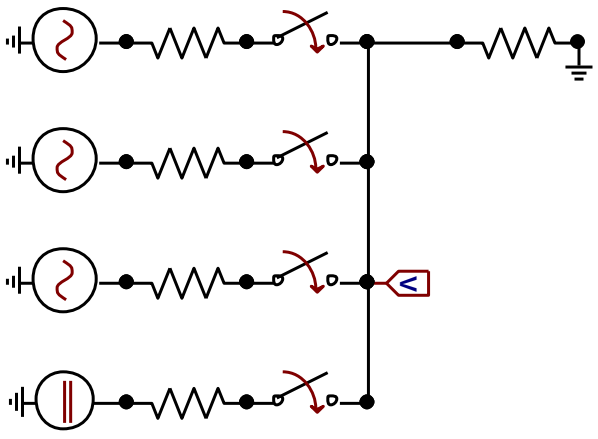
3.3.3 Resultat frå simulering

Tabell 3.5: Simuleringstid FFT samanlikna med DFT

Type	Mengd harmoniske	Simuleringstid[s]	Simuleringstid h.mod. [s]
DFT	4	97.875	8.845
FFT	4	29.079	3.962

Tabell 3.5 viser forskjell i hastigheit mellom FFT og DFT. Tidene i tabellen er avhengig av simuleringslengd, simuleringssteg og maskinvaren til den maskina som simulerer. Såleis kan tidene kun nyttast til innbyrdes samanlikning av modellane. Differansen mellom DFT og tilsvarande FFT vil auke om mengda harmoniske aukar. Det er i dette tilfellet valgt 4 harmoniske, inkludert DC-komponenten. Dette fordi FFT-modellen, som fast har 4 harmoniske, skal kunne samanliknast med DFT-modellen.

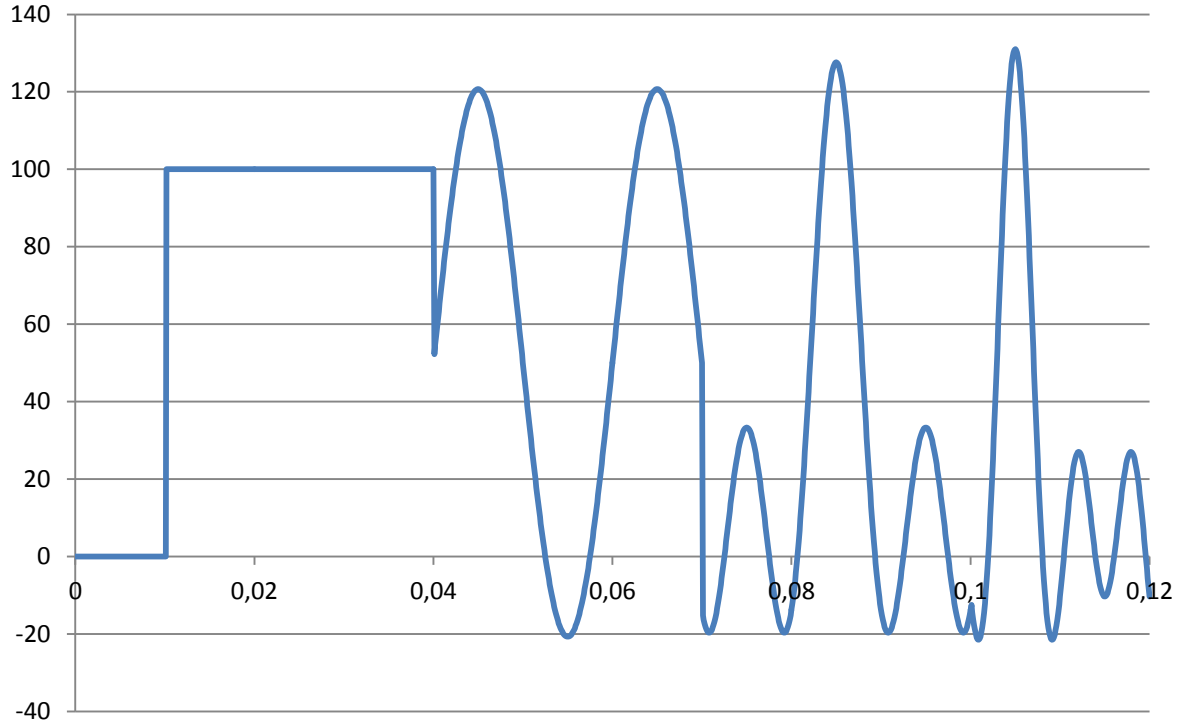
For utprøving av modellane vart det satt opp ein krets med fire kjelder(figur 3.12). Kjeldenes frekvens og innkoplingstidspunkt er gitt i tabell 3.6. Signalet vart målt som spenning på samleskinna til høgre for brytarane. Kilderesistansane vart sett til $1\mu\Omega$ medan jordingsresistanse var 1000Ω. Jordingsresistansen kan difor betraktast som eit brot når meir enn ei kjelde er innkopla. Då spenninga og motstanden er lik på alle kjeldene, vert det difor ei spenningsdeling lik antall kjelder innkopla. Resulterande signal er vist i figur 3.13.



Figur 3.12: Målekrets

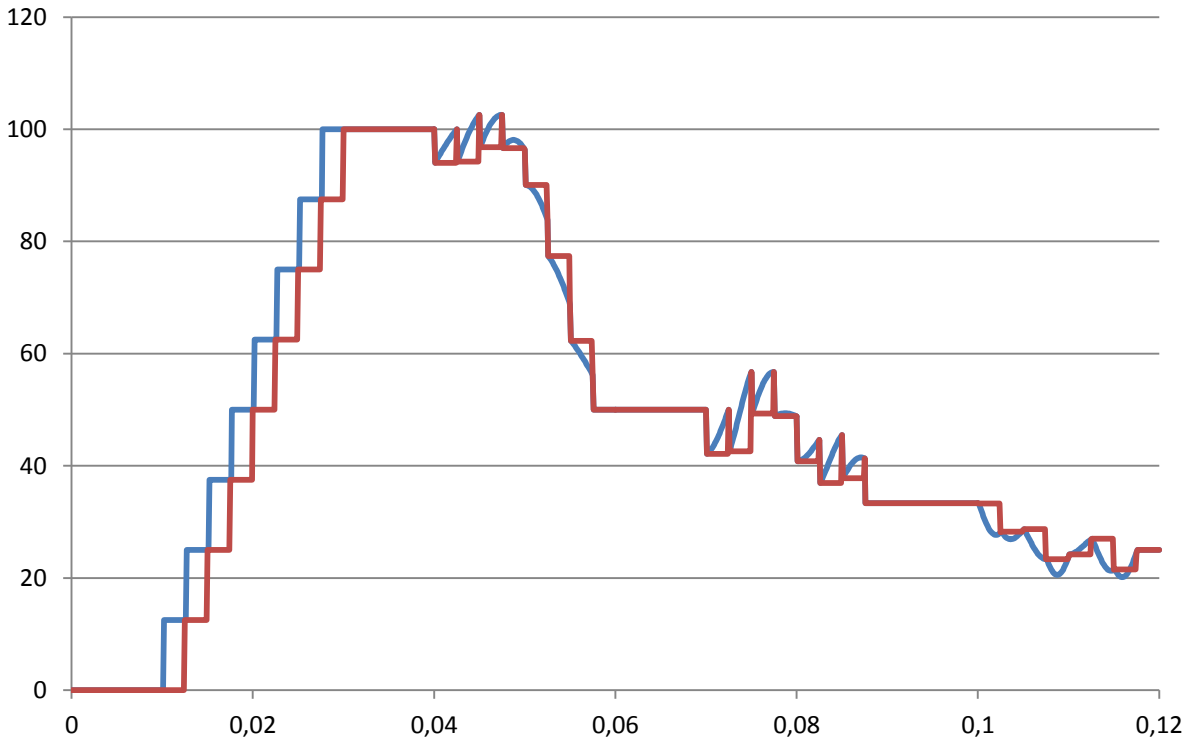
Tabell 3.6: Signalkjelder med innkoplingstider

Kjelde	Innkoplingstid[ms]	Frekvens[Hz]	Fase [$grader$]
100V AC	100	150	270
100V AC	70	100	180
100V AC	40	50	90
100V DC	10	0	0



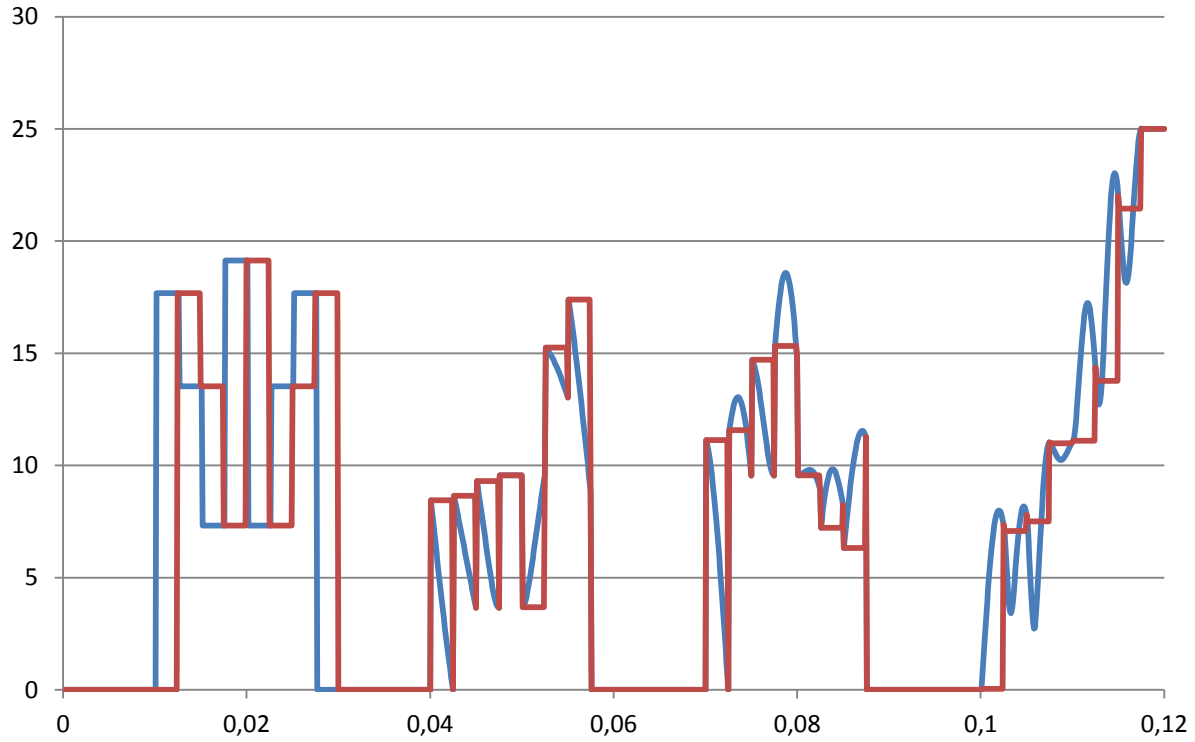
Figur 3.13: Inngangssignal - grunnleggande fourier analyse

FFT-modellen har fast analyseintervall opp til og med tredjeharmoniske. DFT modellen er innstilt tilsvarande. Resultata frå simuleringane er vist i figur 3.14 og B.1 til B.5. Som forventa vart det ikkje noko forskjell på signalet frå DFT-modellen samanlikna med FFT modellen. Difor er det i figurane kun synt resultat med(raud graf) og utan(blå graf) hastigheitsmodifisering. Vinklane på resultata er gitt i grader.



Figur 3.14: DC-komponent effektivverdi - tilhøyrande signal 3.13

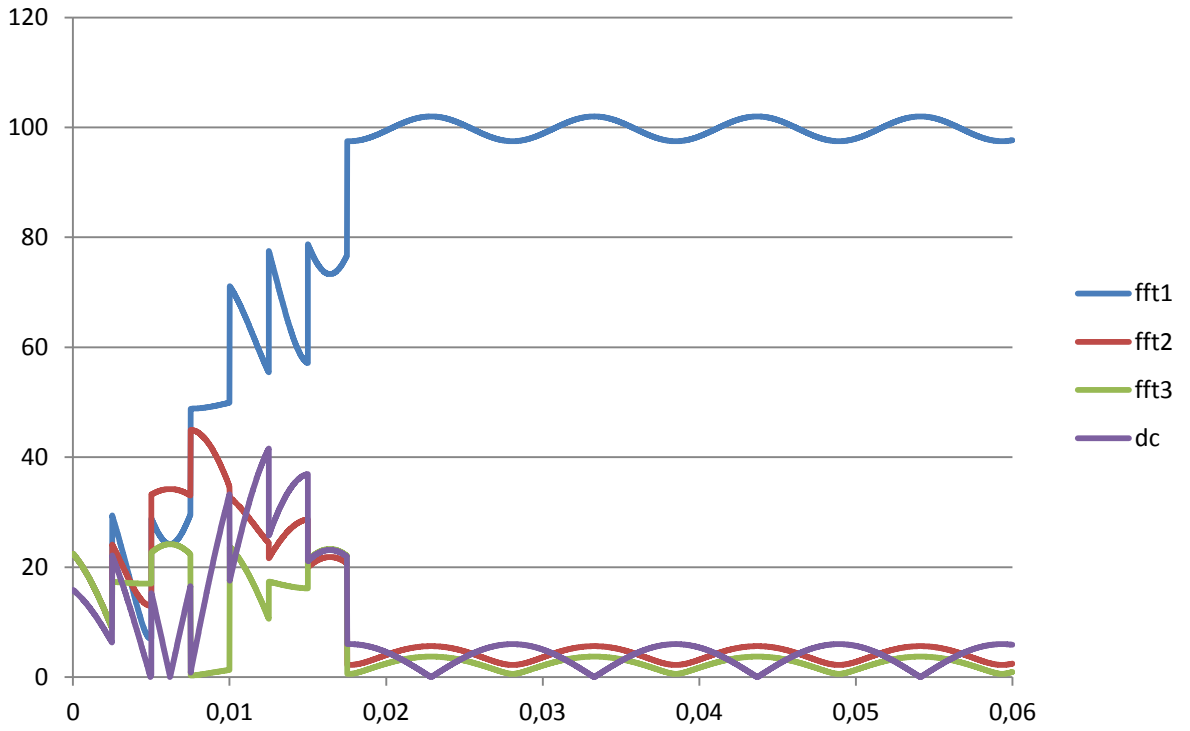
Resultata viser at modellane etter ei endring i ein harmonisk komponent i snitt omlag treng ein periode med data før korrekt resultat er oppnådd(i dette tilfellet $18 - 20ms$). Både effektivverdi og fase stig til den aktuelle komponenten i denne tida opp mot korrekt resultat utan nemneverdige svingingar eller overskyt. Om det derimot skjer ei endring i ein anna harmonisk komponent, vil denne skape svingar i resten av dei harmoniske komponentane. Dette gjer seg spesielt om den harmoniske komponenten som vert utsatt for forstyrrelsar på førehand har ein effektivverdi tilnærma null. Dette vises spesielt godt i figur 3.15, som ikkje skal ha nokon effektivverdi før etter $100ms$. Svingingar i figuren mellom 0 og $100ms$ kjem av endring i andre harmoniske komponentar. I motsatt fall gjer svingingane seg mindre gjeldane om ein komponent allereie har ein effektivverdi, og vert utsett for ei endring(figur 3.14).



Figur 3.15: 3 harmoniske komponent - effektivverdi - tilhøyrande signal 3.13

Hastigheitsmodifisering utgjer liten forskjell i resultatet, men stor forskjell i berekningstid. I verste fall kan signalet vere forseinka ei tid lik tida mellom kvar prøva nytta i berekninga dersom hastigheitsmodifisering vert nytta. Dersom ein aukar antall harmoniske, vil tida mellom kvar prøve verte mindre. Difor vil det ved auka antall harmoniske vere kortare tid mellom kvar oppdatering av utgangssignalet.

Dersom frekvensen på inngangssignalet ikkje er lik ein av frekvenskomponentane til modellen, vil dette skape svingingar i større eller mindre grad på alle utgangar. Dette er vist i figur 3.16 og B.6 til B.8. Her er modellen satt opp med f_{base} = $50Hz$, medan inngangssignalet har ein frekvens på 48, 49 og $49.5Hz$. Effektivverdien til signalet er i alle tilfella 100, medan fasen er 26 grader. Dei største svinginane i utganssignalet har ein ved det største avviket i frekvens(figur 3.16). Fasen til signalet(figur B.8) er det ikkje mulig å lese av korrekt, om ikkje frekvensen til inngangssignalet er lik frekvensen til modellen. Dette fordi signal som ikkje har same frekvens kontinuerleg vil forskyve seg i forhold til kvarandre.



Figur 3.16: Effektivverdi når inngangssignal har ein frekvens på $48Hz$ og effektivverdi på 100. $f_{base} = 50Hz$

3.3.4 Oppsummering av Fourier alternativa

Dersom tredjeharmoniske er tilstrekkeleg bandbredde, er det ingen ingen ulemper med å nytte Radix-2 algoritma framfor DFT. Den gir same resultat, men krever mindre berekningskraft. I begge tilfella gir hastigheitsmodifisering tilstrekkeleg gode resultat og bør difor også nyttast. Dersom det krevst større bandbredde eller fleksibilitet, kan DFT algoritmen nyttast. DFT gir valgfritt antall harmoniske frå 1 til 26.

Om modellen skal brukast direkte som måleverktøy for effektivverdi og fase, bør brukaren vere obs på at dersom signalet ikkje har nøyaktig same frekvens som ein av dei harmoniske komponentane, vil det oppstå svingingar i utgangssignalet. Denne svingina vil vere større jo større avviket på frekvensen til inngangssignalet er i forhold til innstilt verdi på modellen(f_{base}).

3.4 Overstraumvern

3.4.1 Konstant tid

Kjeldekoden for konstant tid overstraumsvern er vist i vedlegg C.2.1. Modellen er bygd opp slik at den detekterar om signalet er større enn ein satt grenseverdi, *limit*. Dersom signaler er større, vil ein timer starte. Om timeren sin verdi vert større enn den innstilte forsinkelsen, vil vernet detektera feil med å setje utgangen, *trip* lik 1. Modellen nullstiller seg sjølv så fort signalet er mindre enn grenseverdien. For momentan utkopling kan forsinkelsen setjast til null.

Dersom modellen skal detektera effektivverdiar kan den nyttast i kombinasjon med ein av rms-modellane. Om dette er tilfelle må inngangen på modellen endrast slik at den aksepterar signal frå andre modellar(INPUT = MODELS).

3.4.2 Invers tid

Denne modellen byggjer på teori i kapittel 2.2.3. Kjeldekoden er vist i vedlegg C.2.2 og flytskjema er vist i vedlegg C.1-C.2.

Rotasjonslengda er gitt i variabelen *integral*. Dersom denne når verdien 1, vil modellen setje utgangen, *trip*, lik 1. Modellen er satt opp slik at integralet ikkje kan overstige 1, eller bli mindre enn 0. Dette er gjort fordi invers tid vern skal simulere induksjonsdiskprinsippet. Induksjonsdisken kan ikkje rotere lenger enn til trip i positiv retning, og heller ikkje lenger enn til startposisjon i negativ retning[12].

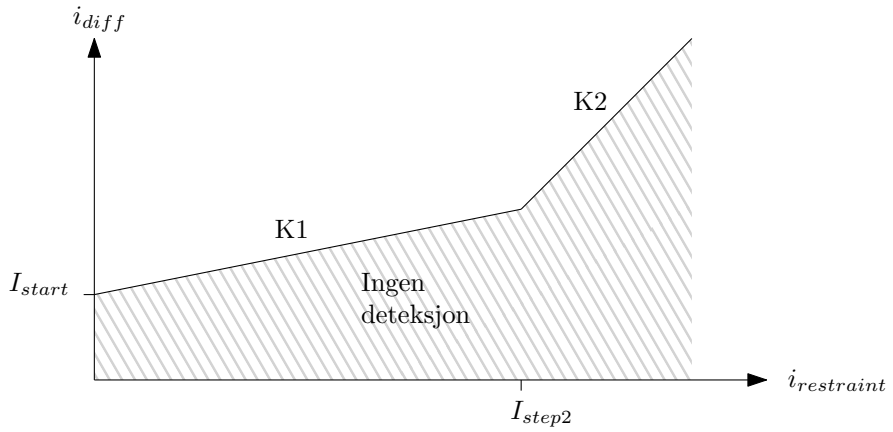
Invers tid karakteristikken vert satt av brukaren ut i frå dataparameterane i modellen.

Kjeldekoden er todelt. Ein del for positivt- og ein del for negativt moment. Både likning for positivt og negativt moment nyttar seg av ein subtraksjon i nemnaren ved utrekning av momentfunksjonen.

3.5 Differensialvern

Kjeldekoden er vist i vedlegg D.2. Flytskjema tilhøyrande kjeldekoden er vist i figur D.1-D.2.

Deteksjonskarakteristikk nytta i modellen er vist i figur 3.17. Dette er ein to-stegs karakteristikk der formålet er god sensitivitet ved interne feil samt god sikkerheit mot utkopling ved eksterne feil[14](jamfør variabel karakteristikk i kapittel 2.3).



Figur 3.17: Deteksjonskarakteristikk

Karakteristikken vert satt ved hjelp av dataparameterane i modellen. $K1$ og $K2$ er konstantar som sett stiginga på karakterstikken, medan I_{start} er minste differensialstraum vernet detekterar feil ved. I_{step2} beskriv kor stor $i_{restraint}$ må vere for at karakterstikken skal endre stigning til $K2$.

Det er mange måtar å berekne restraint, men i_{diff} og $i_{restraint}$ vert berekna ut frå likning (3) og (4) i [13].

$$i_{diff} = |\overrightarrow{I_1} + \overrightarrow{I_2}| \tag{3.8}$$

$$i_{restraint} = (|\overrightarrow{I_1}| + |\overrightarrow{I_2}|) \cdot k \tag{3.9}$$

I tillegg vert $i_{restraint}$ skalert med dataparameteren k, som er satt av brukaren. $i_{restraint}$ skal angi ein snittverdi av straumen på begge sider av det verna objektet. Ved bruk av straumtransformatorar med to viklingar er det difor vanleg å setje k til 0.5 [17].

Det er også implementert ein funksjon for transformatorbeskyttelse basert på forholdet mellom fyrste- og andreharmoniske komponenten til differensialstraumen(delkapittel 2.3.1). Kjeldekode og flytskjema for dette er synt i vedlegg D.3.2 og figur D.3.

Begge modellane krev inngangsdata på rektangulærform.

3.6 Impedansvern

Modellane av impedansverna i denne oppgåva nyttar seg av resistans og reaktans som inngangsdata. Det er difor konsturert modellar som bereknar impedans mellom fasar og mellom fase og jord.

3.6.1 R-X fase-fase kalkulator

Kjeldekode er vist i vedlegg E.1. Koden er basert på likning (2.45). Som inngangsdata er straum og spenning på rektangulærform nytta for å unngå unødig omgjering mellom rektangulær og polarform.

R og X vert kalkulert ved hjelp av fylgjande forhold for divisjon av komplekse tal:

$$\frac{a + j \cdot b}{c + j \cdot d} = \frac{a \cdot c + b \cdot d + j(b \cdot c - a \cdot d)}{c^2 + d^2} \tag{3.10}$$

der nemnaren vert kalkulert fyrst.

3.6.2 R-X fase-jord kalkulator

Kjeldekode er vist i vedlegg E.2. Koden er basert på ein modifisert versjon av likning (2.38), der det ikkje er teke omsyn til bidrag frå feilimpedansen. Likninga for målt fase-impedans brukt i denne modellen er såleis

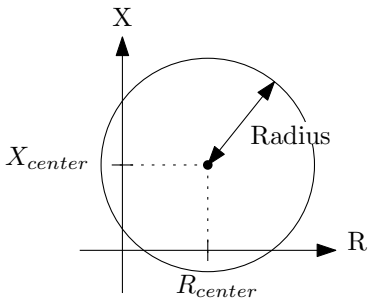
$$Z_L = \frac{V_{fase}}{I_{fase} + I_0 \cdot k} \tag{3.11}$$

der k er forholdet mellom null-sekvens og positiv sekvens impedans sett frå målepunkt til enden av den beskytta linja.

$$k = \frac{Z_0 - Z_1}{Z_1} \tag{3.12}$$

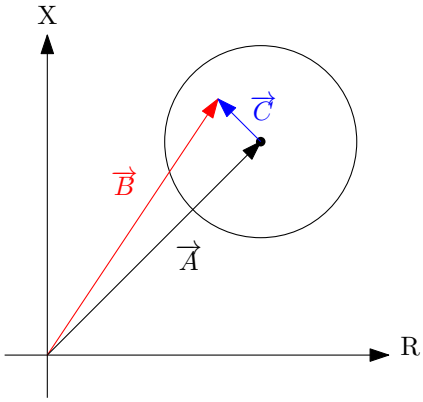
3.6.3 Sirkelkarakteristikk

Kjeldekoden er vist i vedlegg E.3.1. Då koden er relativt oversiktleg er det ikkje laga flytskjema til modellen. Modellens karakteristikk(figur 3.18) vert satt opp av brukaren ved hjelp av dataparameterane.



Figur 3.18: Sirkelkarakterstikk - impedansevern

For å detektera om målt impedans er innanfor karakteristikken, vert differansen mellom målt impedansvektor, $\vec{B}$, og impedansvektoren til sentrum på sirkelen, $\vec{A}$, kalkulert. Dette er vist i figur 3.19, der $\vec{C} = \vec{B} - \vec{A}$. Dersom differansen er mindre enn radiusen på sirkelen, vil målt impedans vere innanfor.

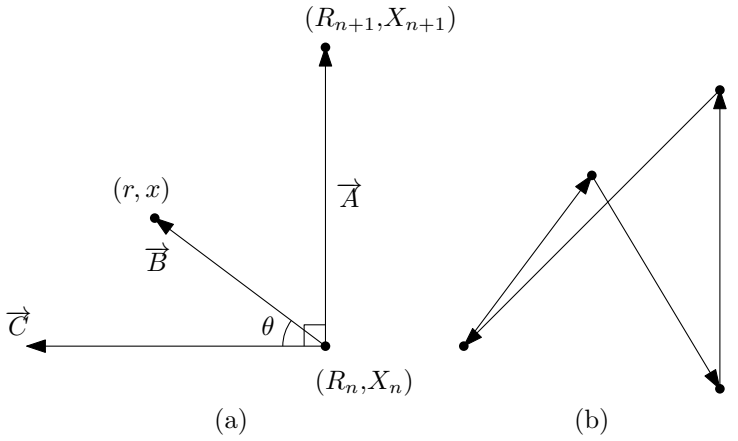


Figur 3.19: Sirkelkarakterstikk - impedansevern - differanse av Z-vektor

3.6.4 Polygonkarakterstikk

Kjeldekoden for impedansvern med polygonkarakteristikk er vist i vedlegg E.4.2. Flytskjema er vist i figur E.1-E.3.

Denne modellen er laga med tre firekanta polygon, altså tre utkoplingssoner. Desse vert definert ut frå dataparameterane. Modellen kontrollerer om eit punkt ligg innanfor sona ved å sjå på kvar kant i polygonet, og kva side av denne kanten punktet ligg på. Dette er i fylgje Eric Haines[18] den raskaste måten for å finne ut om eit punkt ligg innanfor eit konvekst polygon, så sant polygonet har mindre enn 5-10 kantar.Med konvekst meinast det at polygonet ikkje har indre hjørner med vinklar > 180 grader. Av dette fylgjer det at ei side av polygonet ikkje kan krysse ei anna(figur 3.20b).



Figur 3.20: (a)Vektoroppsett for kontroll av impedanspunkt - (b)Feil oppsett av polygon

Figur 3.20a viser korleis vektorane vert satt opp for kontroll av kva side punktet ligg på. $\overrightarrow{A}$ er den kanten av polygonet som skal kontrollerast mot punktet (r, x). Fyrst vert $\overrightarrow{A}$ transformert 90 grader mot klokka, til $\overrightarrow{C}$. Dette vert gjort ved

$$\begin{bmatrix} C_r \\ C_x \end{bmatrix} = \begin{bmatrix} \cos 90° & -\sin 90° \\ \sin 90° & \cos 90° \end{bmatrix} \begin{bmatrix} A_r \\ A_x \end{bmatrix}. \tag{3.13}$$

Av dette følgjer det at

$$\overrightarrow{C} = [-A_x, A_r]. \tag{3.14}$$

For å avgjere kva side av $\overrightarrow{A}$ punktet (r, x) ligg på, vert skalarproduktet mellom $\overrightarrow{C}$ og $\overrightarrow{B}$ nytta. Skalarproduktet er definert som

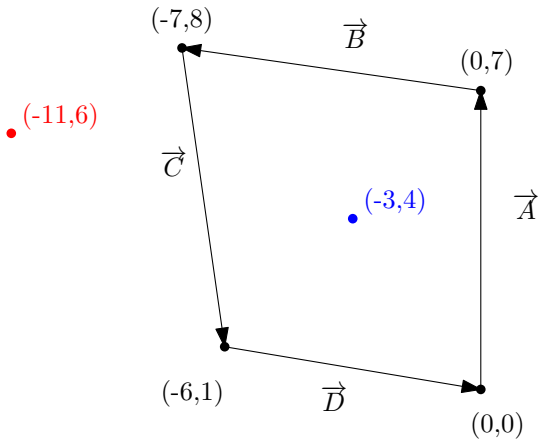
$$\overrightarrow{B} \cdot \overrightarrow{C} = |\overrightarrow{B}| \cdot |\overrightarrow{C}| \cdot \cos\theta \tag{3.15}$$

der vinkelen, θ, er referert til figur 3.20a. Dersom denne vinkelen er mellom -90 og 90 grader, vil punktet (r, x) vere plassert til venstre for $\overrightarrow{A}$. Dersom dette er tilfelle, vil skalarproduktet i likning (3.15) alltid vere positivt. Dersom punktet ligg til høgre for vektoren, vil produktet vere negativt. Om skalarproduktet er null, ligg punktet på linje med vektoren.

$$check(r, x) = ((r - R_n) \cdot (X_n - X_{n+1})) + ((x - X_n) \cdot (R_{n+1} - R_n)) \tag{3.16}$$

Likning (3.16), referert til figur 3.20a, viser funksjonen nytta i modellen for å avgjere kva side av ei side punktet ligg på. Dersom funksjonen er positiv, ligg punktet til venstre. Denne funksjonen er ekvivalent med (3.15).

Ved bruk av denne metoden er det viktig at alle vektorane er definert i same retning i forhold til kvarandre. Ein vektor skal alltid slutte der den neste startar.



Figur 3.21: Døme på retning ved impedansepunkt-kontroll

Dersom ein tek utangspunkt i figur 3.21 vil punktet måtte ligge til venstre for alle fire vektorane for at det skal vere innanfor(blått punkt). Kva rekkefølgje vektorane vert kontrollert i, vil ikkje påverke resultatet. Det raude punktet ligg derimot ikkje til venstre for $\overrightarrow{C}$, og er såleis utafor polygonet.

Modellen er satt opp slik at polygonet må ha retning som i figur 3.21. Dei tre sonene vert definert ut frå dataparametertabellar. For at modellen skal fungere, må sone 3 vere ytterst, sone 2 innafor der, og sone 1 innerst. Sone 2 må ikkje skjere nokon linjer i sone 3, og sone 1 må ikkje skjere sone 2. Dette fordi modellen, for å spare tid, ikkje sjekkar sone 2 og sone 1 om punktet er utafor sone 3.

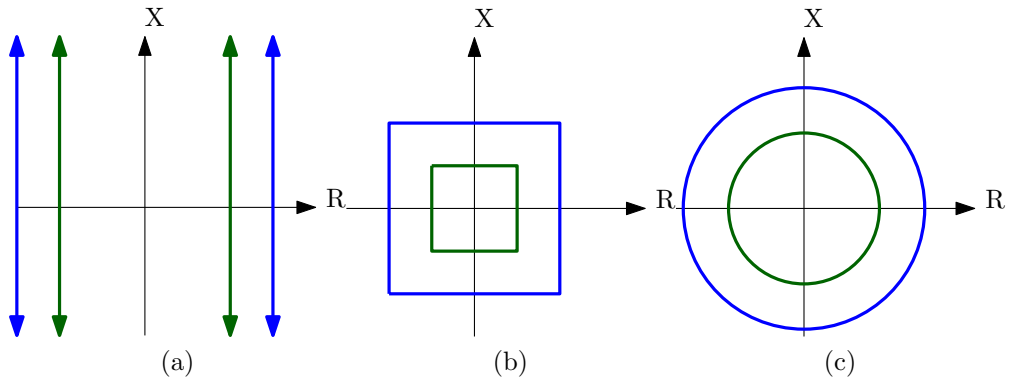
Utanom å definere sonene, må brukaren definere ein tidsforsinkelse for kvar sone. Målt impedanse må då vere innanfor tilhøyrande sone i angitt tid for at vernet skal gi signal om feil. Dersom brukaren set tidsforsinkelsen til null, vil vernet gi signal om feil momentant når verdien på inngangen er innafor sona. Eventuell forsinkelsar frå feilen inntreff vil då kun vere avgrensa av treghet i filter på inngangssignal.

I tillegg til utgang for feildeteksjon er det implementert tre start-utgangar, ein for kvar sone. Dersom impedansen er innafor ei sone, vil den respektive utgangen gi signal om det, uavhengig av eventuell innstillt forsinkelse på utgang for feildeteksjon.

3.6.5 Effektpendlingsblokkering

Denne modellen byggjer på teori i delkapittel 2.4.6.

Modellen byggjer på den tradisjonelle metoden for å detektera effektpendling, der den måler tida når impedansen flyttar seg mellom to soner. Det er implementert tre forskjellige karakteristikkar basert på [20]: blender(figur 3.22a), sirkel(figur 3.22c) og polygon(figur 3.22b).



Figur 3.22: Karakteristikk effektpendlingsblokkering

I figur 3.22 er blått den ytre sona og grønt den indre. Tida vert målt frå impedansen kjem inn i blå sone til den forlet blå sone, eller går inn i grøn sone. Ved bruk av blenderkarakteristikk er det, i tillegg til å spesifisere kvar linjene skjer R-aksen, lagt inn data for stigningstal i grader. 90 grader vil gi ein vektor parallellt med reaktans-aksen. Ser ein på hastigheit vil sirkel og blender vere raskare ved simulering enn polygon.

Det er implementert to utgangar. Ein utgang for stabil effektpendling, og ein utgang for ustabil effektpendling. Utgangen for stabil effektpendling vert aktivert dersom impedansen har brukt meir enn ei førhandsinnstillt tid til å flytte seg frå blå til grøn sone, og nullstilt om impedansen forlet blå sone. Utgangen for ustabil effektpendling vert aktivert om resistansen har endra forteikn når impedansen forlet den blå sona[19][7, s.141]. Denne utgangen må nullstillast manuelt gjennom reset-inngangen på modellen, eller ved å starte simuleringa på nytt. Då det i denne oppgåva er fokusert på per-fase komponentar har ikkje modellen i seg sjølv muligheit for å nullstille seg sjølv om det oppstår negativ-sekvens straumar. Det må då opprettast ein eigen modell som eventuelt aktiverar reset-inngangen til modellen.

Det er ikkje lagt inn impedansvern-element i modellen. Om funksjonen med effektpendlingsblokkering skal nyttast, må den nyttast i kombinasjon med eksisterande impedansvern som eventuelt skal blokkerast. For å få til dette må impedansvernmodellen endrast slik at den kan ta i mot eit blokkeringssignal. Koden for dette er vist under. Det blir

her lagt til ein ekstra inngang på impedansvernmodellen, *block*. Dersom denne inngangen vert satt til 1, vil impedansvernet vere blokkert frå å gi signal om utkopling.

```
...
INPUT    ...
         ...
         ...
         block


...
EXEC
     IF block != 1 THEN
         ...
         ...
         All eksisterande kode i exec-blokka til ↙
            ↳ impedansvernmodellen
         ...
         ...
     ELSE
         trip := 0
     ENDIF
ENDEXEC
...
```

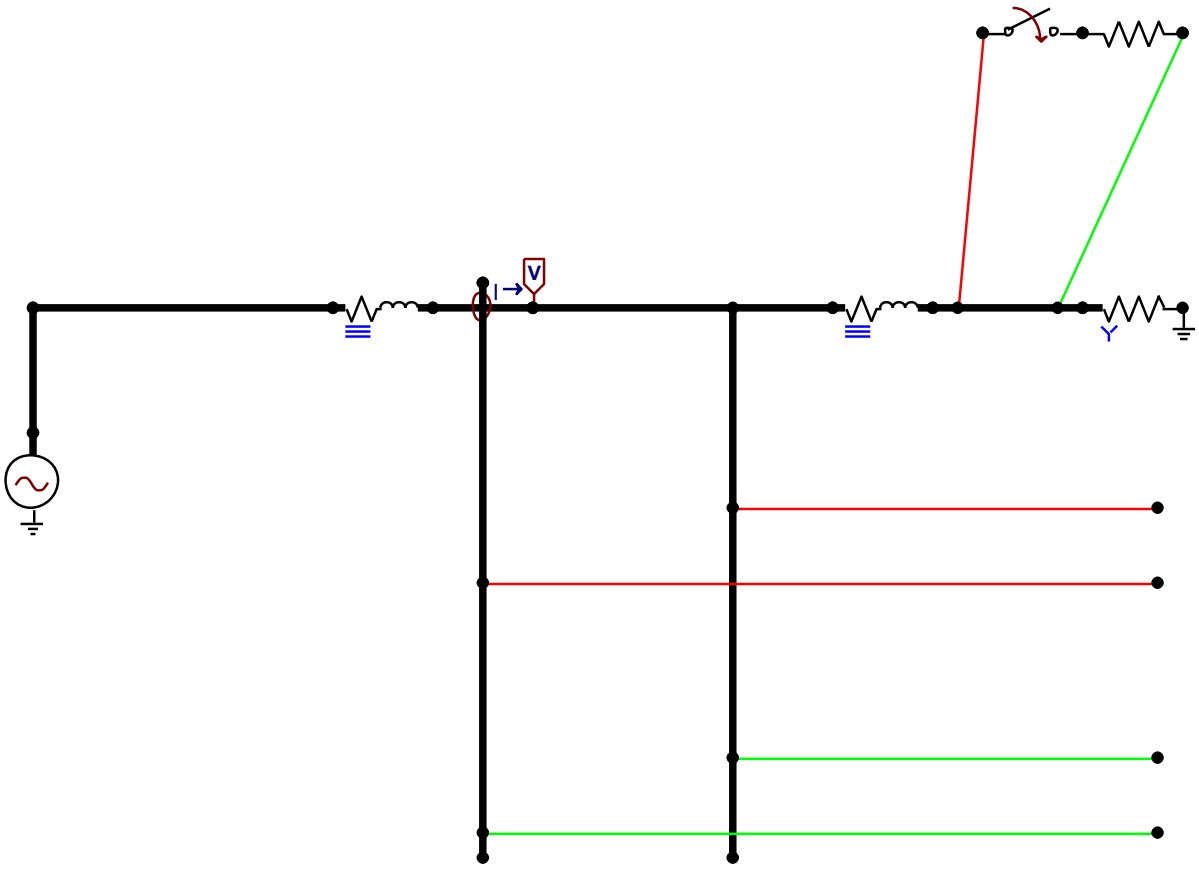
Denne koden vil setje impedansvernet i pause, så lenge det vert blokkert av effektpendlingsvernet. Om blokkeringa skulle verte oppheva dersom det til dømes oppstår ein usymmetrisk kortslutning på linja under effektpendlingsforløpet, vil vernet fortsette der det slapp før blokkering(verdiar på tidtaking/forsinkelsar vil verte hugsa).

Flytskjema for modellen er satt opp uavhengig av karakteristikk. Modellen med sirkelkarakteristikk nyttar same metode som impedansmodellen med sirkelkarakteristikk, medan modellane med blender og polygonkarakteristikk nyttar same metode som impedansmodell med polygonkarakterstikk for å avgjere om impedansen er utanfor eller innanfor sona..

Kapittel 4

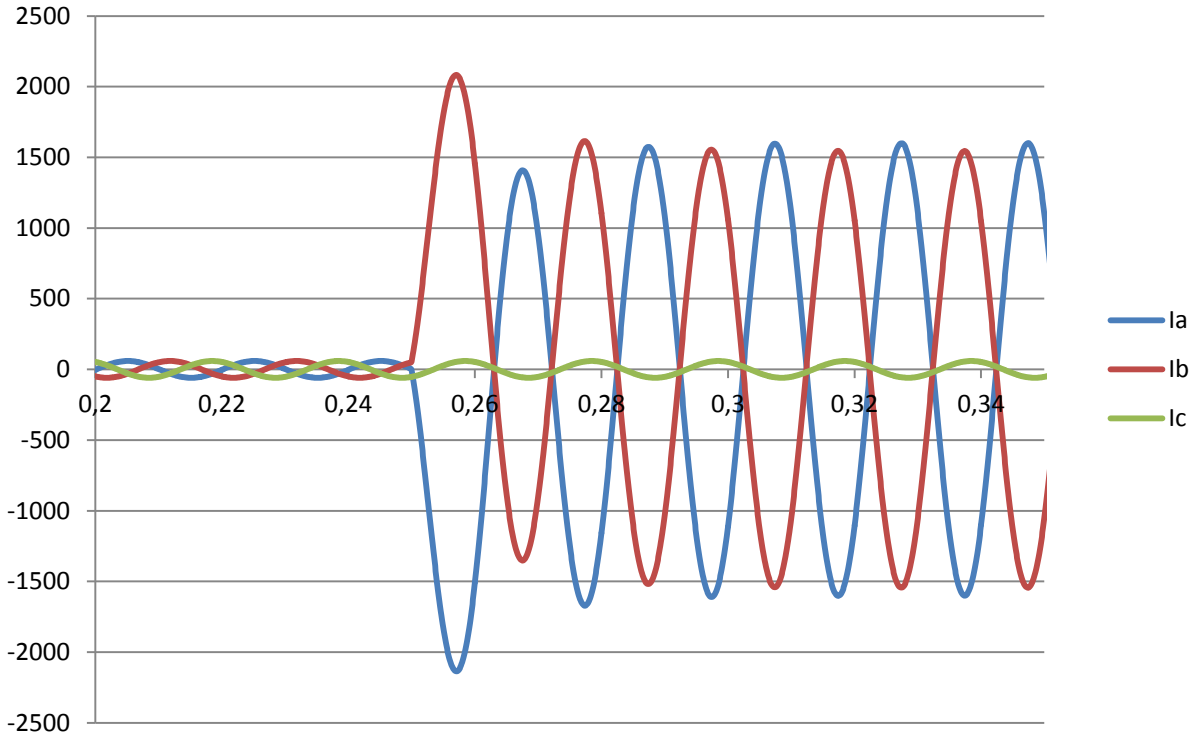
Verifisering

For verifisering er resultata i nokre av vern-modellane samanlikna med resultata til eit Siemens 7SA610 vern. Dette er i utgangspunktet eit trefasa numerisk impedansvern, men har også fleire tilleggsfunksjonar. Ved verifisering vart impedansfunksjonen, i tillegg til overstraumsfunksjonen, nytta til verifisering.

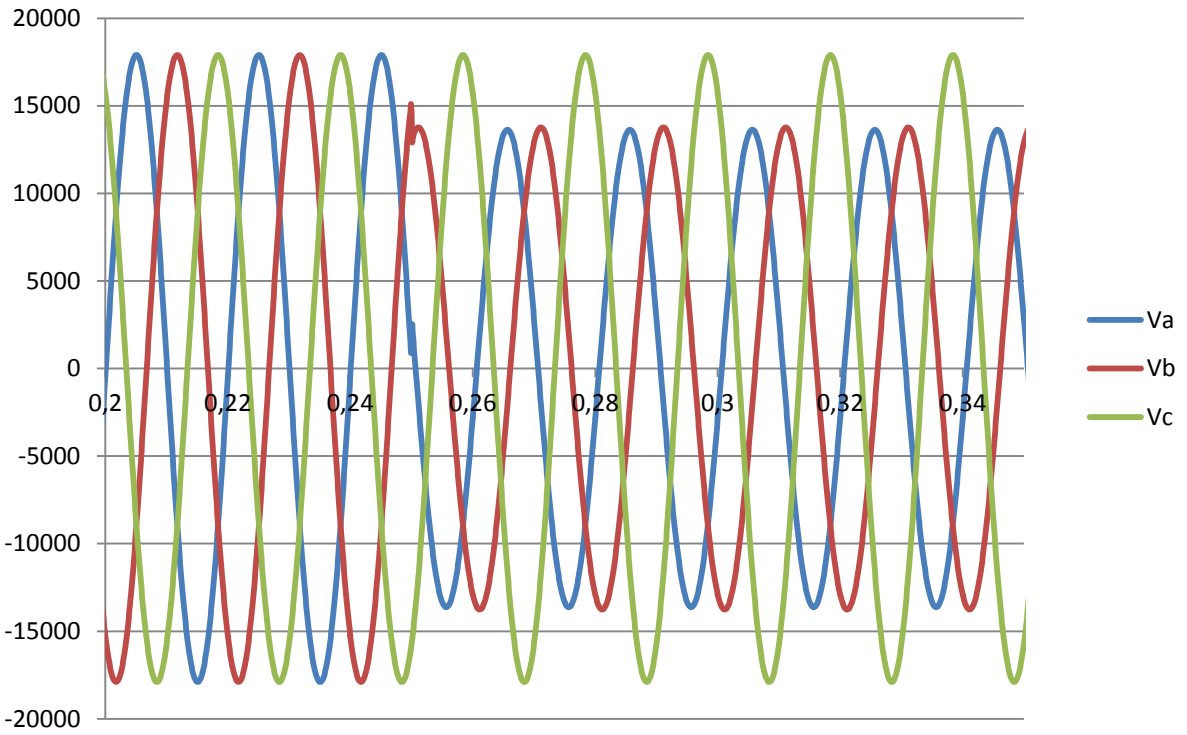


Figur 4.1: Verifiseringskrets

Det vart satt opp ein trefasa krets(figur 4.1), som vart brukt i alle testtilfella. Kilden har ein effektivverdi på $22kV$ fase-fase og ein frekvens på $50Hz$. Fasen på kilden er satt til 270 grader. Den fyrste linjeimpedansen er $1 + j3.14\Omega$, den andre $2 + j6.28\Omega$, lasta 300Ω og feilresistansen $1\mu\Omega$, per fase. Feilen er mellom fase A og fase B, og vert lagt inn ved $250ms$. Total simuleringstid er $0.5s$, og simuleringssteget er $0.1ms$. Eit snitt av straum og spenningar er henholdsvis vist i figur 4.2 og 4.3. Dette er faseverdiar målt mellom den fyrste og andre linjeimpedansen.



Figur 4.2: Verifiseringskrets - straummåling



Figur 4.3: Verifiseringskrets - spenningsmåling

Signalet frå simuleringa vart lagt inn i ei COMTRADE-fil, og spelt av for Siemens-vernet ved hjelp av ein Omicron(modell CMC-56A). Tripsignalet frå vernet vart registrert ved avspeling av simuleringssignala. For avspeling vart vernets analoge inngangar kopla frå dei eksisterande straum- og spenningstransformatorane. Vidare vart dei analoge utgangane på Omicron direkte tilkopla dei respektive inngange på vernet.

Signala frå simuleringa er primærverdiar med toppverdiar på $17,9kV$ og $1600A$ per fase. Maksimalverdiar Omicron kan levere er $125V$ og $10.3A$. Vernet har nominelle verdiar på $125V$ og $5A$, men tåler større verdiar enn det Omicron kan levere over korte tidsrom[7, s.569]. Det er Omicron som gir avgrensinga, så denne vart dimensjonerande for transformatorane. Signala frå simuleringa må difor skalerast, og dette vart gjort ved å leggje inn kunstige straum- og spenningstransformatorar i programvaren til Omicron og vern. Spenningstransformator nytta var $1/200V$ medan straumtransformatoren vart sett til $1/400A$. Dette gav ei maksimal spenning på $\approx 90V$ og ein maksimal straum på $\approx 4A$, som er lågare enn maksimalverdiane til Omicron.

Modellen for effektpendling og differensialvern kunne ikkje verifiserast fordi vernet ikkje har ein differensialfunksjon og brukar ein anna metode for deteksjon av effektpendling[7, s.138].

4.1 Konstant tid overstraumvern

Ved verifisering av konstant tid vart både vern og modell stilt inn med ein grenseverdi på $500A$ og ei forseinka utkopling på $50ms$. Modell C.2.1 (overstraumvern) vart nytta i kombinasjon med A.2 (dRMS). På rms-modellen var frekvensen satt til $50Hz$, antall prøvar 64 og ein startverdi på $0V$. Modellen vart kopla til straummåling på fase A.

Tabell 4.1: Resultat verifisering overstraumvern

Type	Triptid[ms]
Simulering	55.4
7SA610	73.8

Tabell 4.1 viser at modellen detekterar feil raskare enn vernet. I [12, s. 606] står det at vernet ved overtstraum har ei minimum start-tid på $20ms$. Når det i tillegg er innstillt med $50ms$ forsinkelse vil det gi ei minste deteksjonstid på $20 + 50 = 70ms$. Modellens tidtaking, derimot, startar så snart rms-verdien når innstilt grenseverdi, som gir ei deteksjonstid lik $50ms$ + forsinkelse i rms-kalkulator. I dette tilfellet vart resultatverdien frå rms-modellen $500A$ etter $5.4ms$.

4.2 Invers tid overstraumvern

Vernet vart stilt med ein startstraum på 200A og karakteristikken vart satt til ANSI extremely inverse. Ved hjelp av [7, s.596] vart modellen stilt inn etter tilsvarande karakterstikk. Modellen vart kopla til straummåling på fase A. Konstantane på modellen vart satt opp som fylgjer:

Konstant	Verdi
A	5.64
B	0.02434
K	0
P	2
Q	2
t_reset_data	29.1

Konstantane Q og t_reset_data vart ikkje funne i [7, s.596], så dei vart henta frå [12] under kategorien extremely inverse. Tid for deteksjon er vist i tabell 4.2.

Tabell 4.2: Deteksjonstid invers tid

Type	Prosessering	Tid[ms]
Vern	Uvisst	203.6
Modell	Ingen	214.8
Modell	aRMS - 0.01s	205.4
Modell	aRMS - 0.02s	209.9

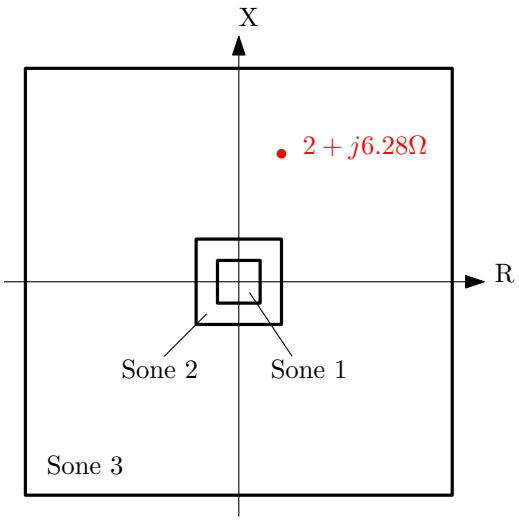
Modellen, utan bruk av prosessering, har i dette tilfellet omlag $10ms$ tregare deteksjon enn vernet. Relativt sett er ikkje dette stor forskjell(deteksjonstid $203.6ms$ for vern og $214.8ms$ for modell).

Fyrst ved bruk av aRMS-modellen(tidskonstant lik ein halvperiode) i kombinasjon med inverstid-modellen, var det muleg å oppnå omtrent same deteksjonshastigheita. Dette kan tyde på at vernet, ved bruk av inverstid-funksjonen, nyttar seg av signalprosessering med kortare innsamlingsvindauge under transiente forløp[3, s.364].

4.3 Impedansvern

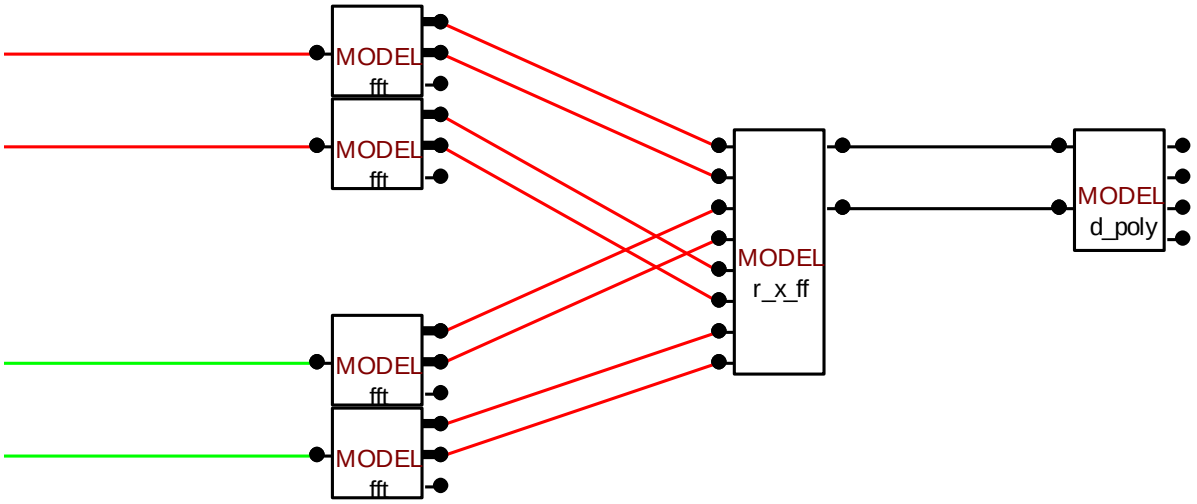
Ved verifisering av impedansmodellen vart både polygon-modellen og Siemens-vernet stillt inn med fylgjande soner:

Sone nr.	R nedre [Ω]	R øvre [Ω]	X nedre [Ω]	X øvre [Ω]	Forsinkelse [s]
1	-1	1	-1	1	0.001
2	-2	2	-2	2	0.05
3	-10	10	-10	10	0.1



Figur 4.4: Soneinnstilling impedansvern

Raudt punkt i figur 4.4 viser feilimpedans. Ved simuleringa vart modellen nytta saman med fire FFT-modellar utan hastigheitsmodifisering, der to berekna fasestraumar og to berekna fasespenningar. Signala deira vart kopla på fase-fase R-X kalkulator-modellen, og derfra levert til vernet(figur 4.5). Figuren viser korleis simuleringa vart satt opp i ATPDraw. Dei to øverste fft-modellane bereknar fyrsteharmoniske komponentar av fase A spenning og straum. Dei to nederste fft-modellane gjer det same for fase B. FFT-modellane er satt opp med ein grunnfrekvens på $50Hz$.



Figur 4.5: Verifiseringskrets - impedansmodell

Type	Triptid[ms]
Simulering	108.4
7SA610	134.2

Resultat frå impedansmålinga er synt i figur F.1. Utkoplingstid til modellen vart $108.4ms$ medan vernet detekterte feilen ved $134.2ms$. Dette tyder på at vernet nyttar filtrering utanom FFT, som skapar forsinkelsar ytterlegare forsinkelsar på signalet[3, s.364].

Kapittel 5

Vidare arbeid

I dette studiet er det skissert løysingar på relévern i ATPDraw ved hjelp av kjeldekode og flytskjema. For å auke brukarvenlegheiten og nytteverdien bør denne koden implementerast i det eksisterande biblioteket i ATPDraw som ferdige modellar med tilhøyrande ikon.

Ved simulering vil som regel to eller fleire modellar nyttast i serie, der ein modell brukar data frå ein anna. For betre oversikt og enklare oppsett av ein simuleringskrets, kan modellar som skal brukast saman setjast opp som ein enkelt modell. Vidare er modellane bygd opp som per-fase komponentar, medan dei fleste reelle relévern kjem i trefasa utgåver. Difor bør også dette vere eit tilgjengeleg val i ATPDraw.

Relle relévern som impedansvern og effektpendlingsvern nyttar filter på inngangssignalet i tillegg til FFT-rutinar. Ein slik funksjon er ikkje implementert i denne oppgåva, men bør vektleggast. Dette kan til dømes vere filter som dempar støy med frekvens over bandbredda til FFT-rutinen.

Dersom det, under simulering, skal hentast ut overharmoniske signal med frekvens større enn tre gonger grunnharmoniske, må DFT-modellen nyttast. Denne er treg i forhold til FFT-rutinen. Om behovet er stort for frekvenskomponentar større enn dette kan det vurderast om ein skal utvikle FFT-rutinar tilpassa dette.

For å knytte relévernmodellane opp mot reelle relévern bør det også implementarast ein funksjon for opptak av data i feilsituasjonar, gjerne med muligheit for eksport av data til COMTRADE format.

Ikkje alle modellane kunne verifiserast mot reelle relévern. For å sikre at modellane fungerer slik dei skal, bør kvar modell verifiserast før den implementerast i ATPDraw.

Kapittel 6

Konklusjon

Modellane skissert i denne rapporten har oppnådd ein verkemåte tilsvarande den funnen i relevante artiklar og lærebøker.

Signalbehandlingsmodellane, som består av ein «Radix-2» FFT rutine, ein DFT rutine, to RMS rutinar og to impedans-kalkulatorar, gir korrekte resultat dersom dataparameterane er justert korrekt i forhold til signalkjelda. Dersom signalet er periodisk vil resultatet frå FFT, DFT og RMS rutinane gi resultat som er forseinka med opp til ein periode i forhold til inngangssignalet. Ved bruk av FFT og DFT rutinane er det viktig å ta omsyn til bandbredda for korrekt resultat. Om signalet er større enn bandbredda lyt det implementerast filter med knekkfrekvens lågare enn bandbredda. Filter er ikkje omtala i denne oppgåva.

Konstant og invers tid overstraumsmodellane har ved verifisering oppnådd eit resultat tilsvarande eit reelt overstraumvern.

Impedansmodellen har ved simulering vist ei deteksjonstid som er kortare enn eit reelt impedansvern. Dette kjem truleg av at det reelle vernet nyttar ei meir innfløkt signalprosessering med fleire filter, noko som gir ytterligare forsinkelse. Verkemåten til impedansmodellen i seg sjølv er såleis i tråd med teori.

Modellane for deteksjon av effektpendling og differensialvern med og utan andreharmonisk blokkering(transformatorvern), vart av avgrensingar i funksjonar på det fysiske verifiseringsvernet, ikkje verifiserte. Ved simulering viser også desse ein verkemåte i tråd med teorien.

Tillegg A

RMS - MODELS

A.1 aRMS - kjeldekode

```
MODEL aRMS

COMMENT
Model that calulates the analog RMS value of a signal,
by Torstein Stadheim, 20.05.2012

Source: Protective Relaying Theory and Applications
        Page 100,
        Second Edition, ABB, 2004

If error #644 occurs during simulation, increase the ↙
   ↳ simulation timestep.
<KILL = 644.  The real storage area is overflowing into the↙
   ↳  real stack.  Increase the storage allocation for ↙
   ↳ MODELS.>

If a periodic signal is used as input, one period of the ↙
   ↳ signal should be used for correct results.

ENDCOMMENT

DATA    smoothing{dflt:0.02} -- Integral span time [s]
        initRms{dflt:0} -- Default initial rms value

INPUT   signal

OUTPUT  rms

VAR     rms
        samples
        sumR
```

```
HISTORY signal{dflt:initRms} -- Fills the history term with↙
   ↳  the initial rms value

DELAY CELLS (signal) : smoothing/timestep -- Ensures one ↙
   ↳ integral span of data

INIT
    rms:=0
    samples:=smoothing/timestep -- Total number of samples ↙
       ↳ in integral span
ENDINIT

EXEC

  SumR:=0   -- Resets the integral

  FOR i:=0 TO (samples-1) DO
    SumR:=SumR+delay(signal,timestep*i,1)**2     -- ↙
       ↳ Calculates the new integral
  ENDFOR

  rms:=sqrt((1/samples)*SumR)   -- Calculates the new rms ↙
     ↳ value

ENDEXEC

ENDMODEL
```

A.2 dRMS - kjeldekode

```
MODEL dRMS

COMMENT
Model that calulates the digital RMS value of a signal,
by Torstein Stadheim, 20.05.2012

Source: Protective Relaying Theory and Applications
        Page 100,
        Second Edition, ABB, 2004
ENDCOMMENT

DATA    f_base { dflt:50 }      -- Fundamental frequency [↙
   ↳ Hz]
        n_sample { dflt:64 }    -- Number of samples per ↙
           ↳ period of fundamental frequency
```

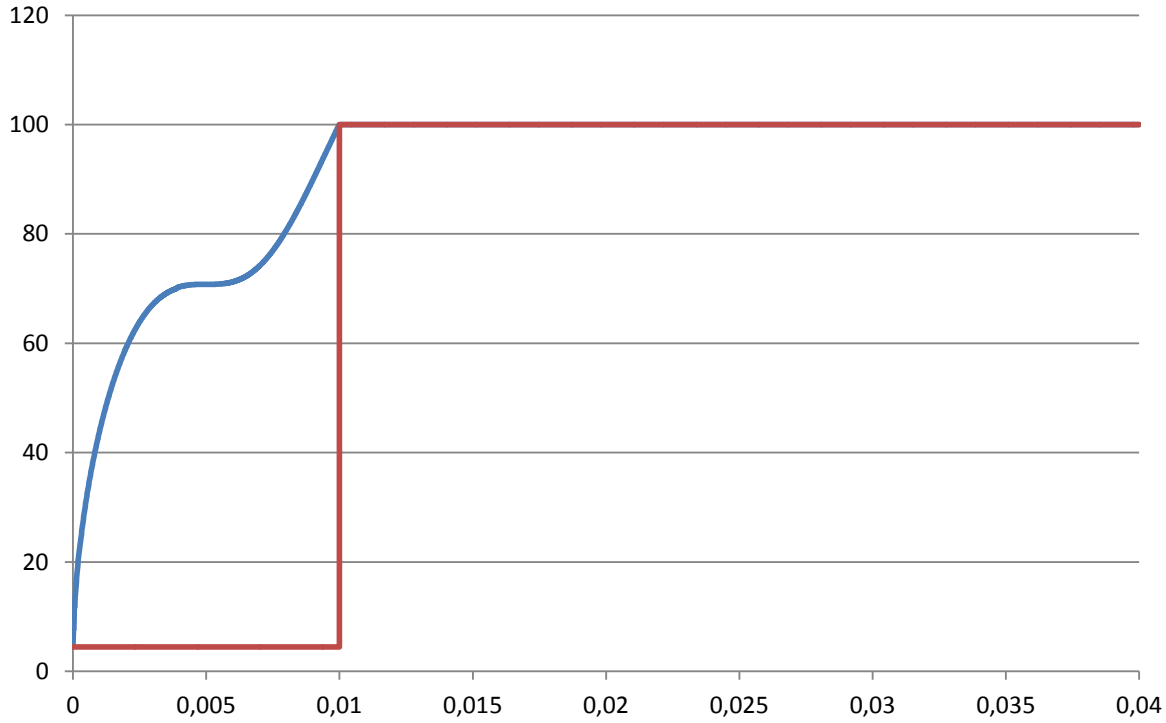
```
        initRMS { dflt:0 }      -- Inital RMS value

INPUT   signal

OUTPUT  rms

VAR     n_max
        ratio
        sumRMS
        rms

HISTORY signal { dflt:initRMS } -- Fills the history term ↙
   ↳ with the initial rms value

DELAY CELLS (signal) : 1 / ( timestep * f_base ) + 1  -- ↙
   ↳ Ensures one period of data

INIT
    rms := 0
    sumRMS := 0

    n_max := 1 / ( timestep * f_base ) -- Maximum number of↙
       ↳  independent samples

    IF n_sample >= n_max THEN -- If wanted number of ↙
       ↳ samples is higher than maximum number of ↙
       ↳ independent samples
        ratio := 1
    ELSE
        ratio := n_max / n_sample
        n_max := n_sample
    ENDIF

    ratio := ratio*timestep

ENDINIT


EXEC
    sumRMS := 0
    -- Calculates the integral of one period
    FOR i := 0 TO ( n_max - 1 ) DO
        sumRMS := sumRMS + delay(signal,i*ratio,1)**2
    ENDFOR

    -- Calculates the rms value of the integral
```

```
    rms := sqrt(( 1 / n_max ) * sumRMS )

ENDEXEC
ENDMODEL
```

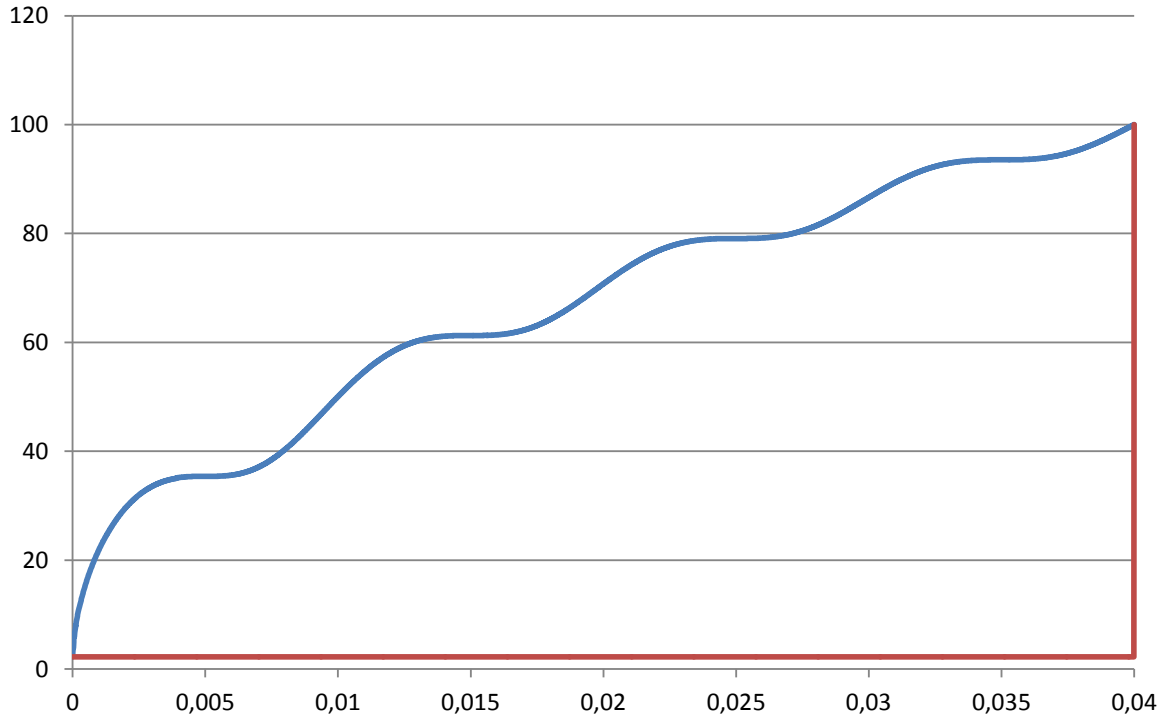
A.3 Resultat frå simuleringar

I dette delkapittelet har alle figurar den same fargemarkeringa. Blå graf er modellen utan modifikasjon for hastigheitsauke jf. kap 3.2.3. Raud graf er med modifikasjon for auka hastigheit.

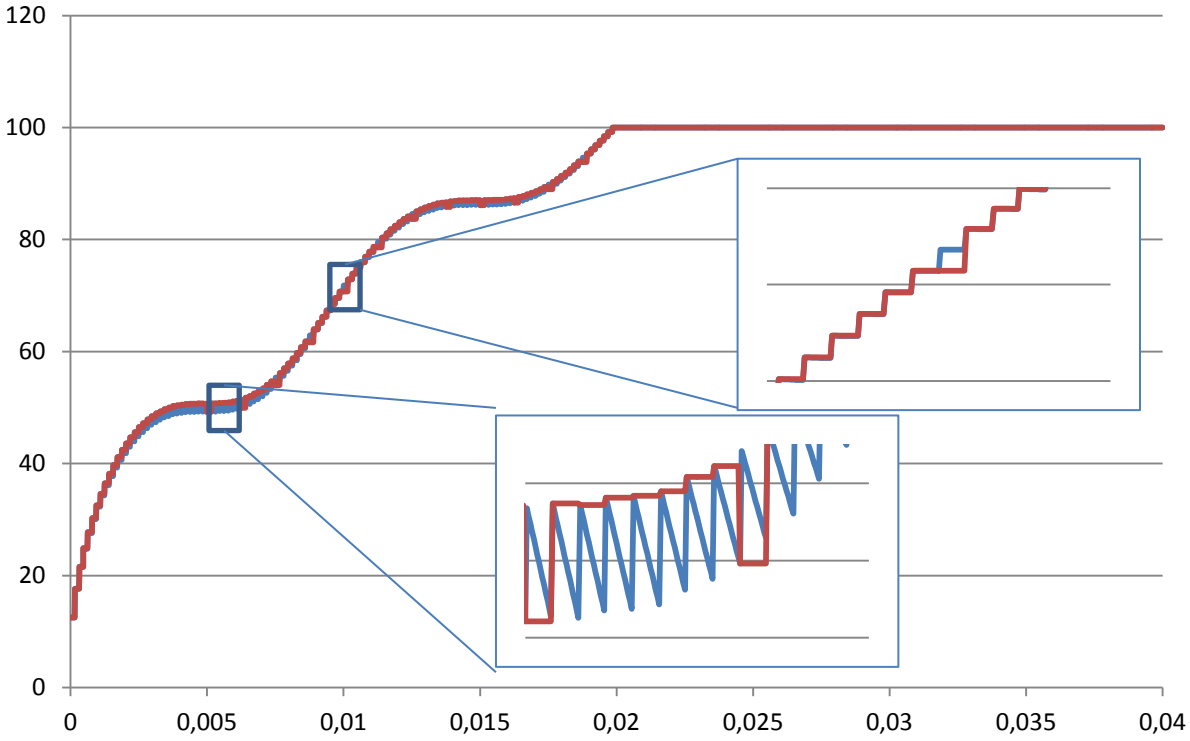
A.3.1 Innsving



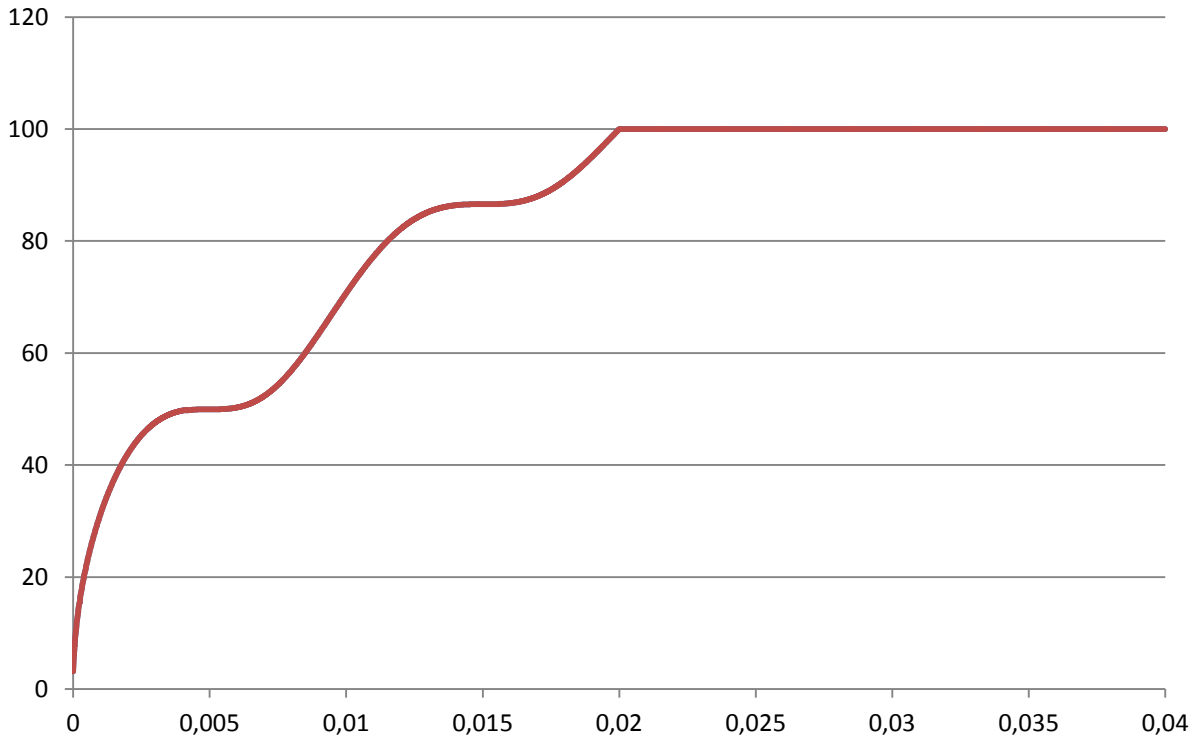
Figur A.1: aRMS - $t = 0.01$ - raud: med h.mod - blå: utan h.mod



Figur A.2: aRMS - $t = 0.04$ - raud: med h.mod - blå: utan h.mod

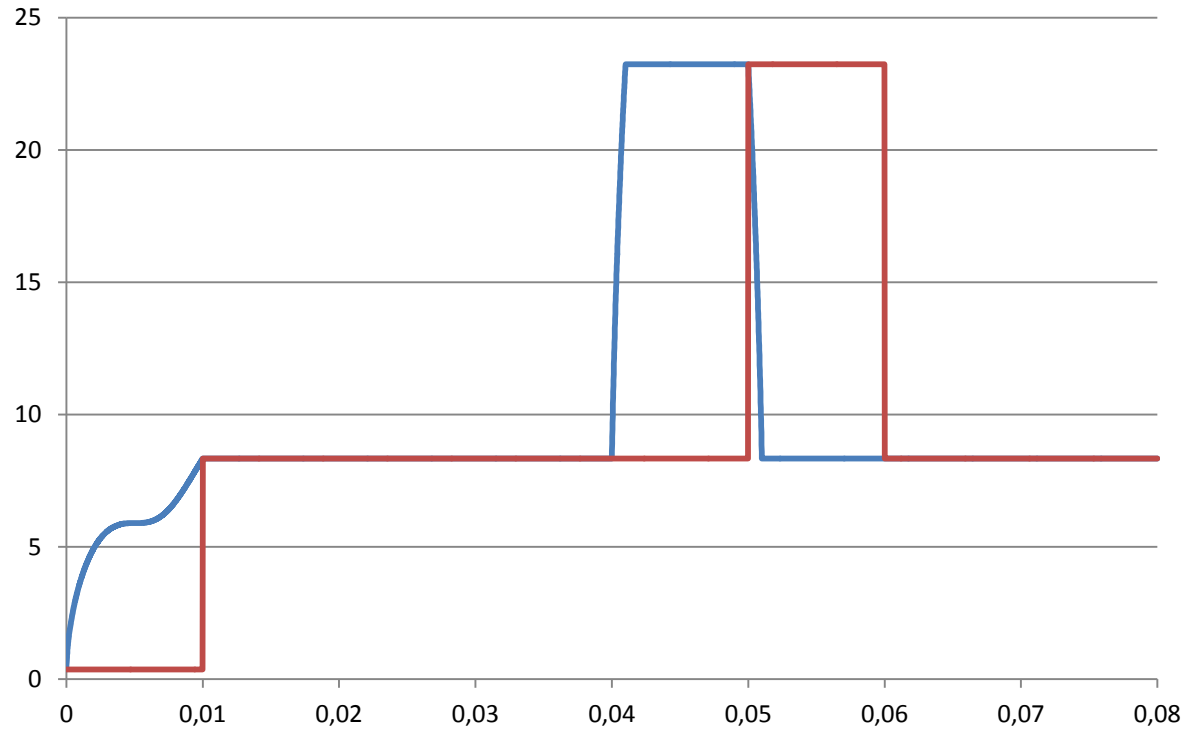


Figur A.3: dRMS - $N = 128$ - $f_{base} = 50Hz$ - raud: med h.mod - blå: utan h.mod

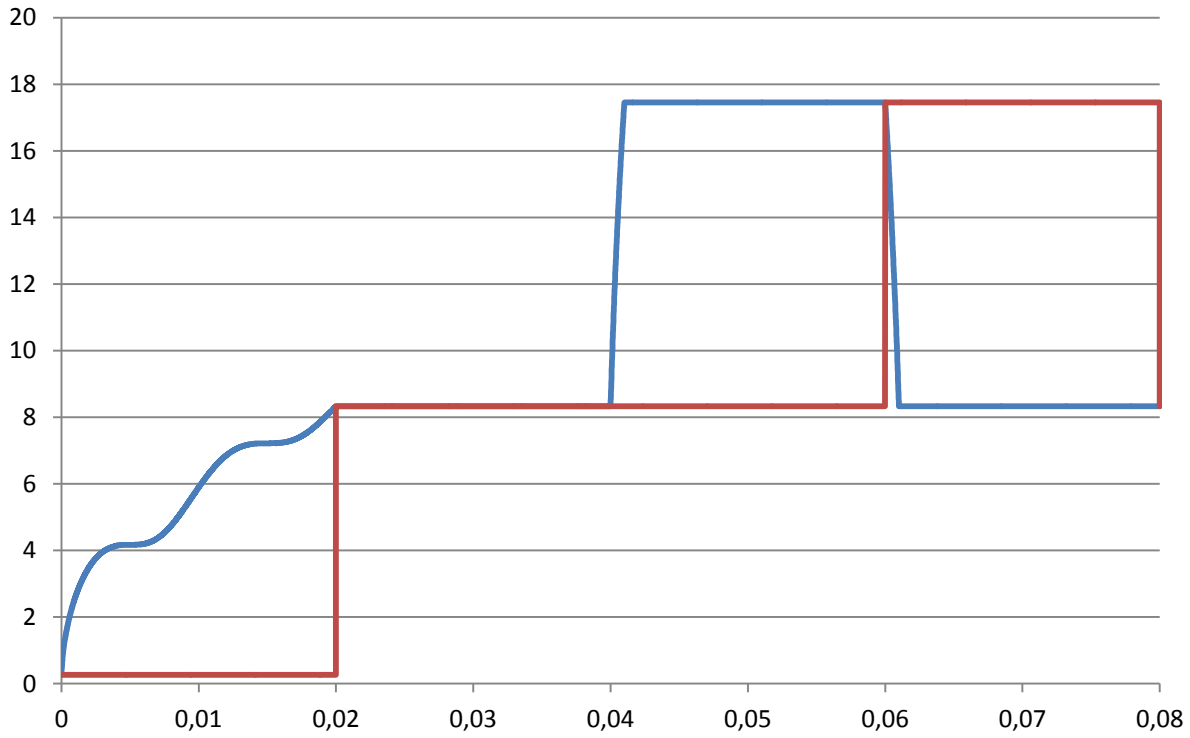


Figur A.4: dRMS - $N = 10000$ - $f_{base} = 50Hz$ - raud: med h.mod - blå: utan h.mod

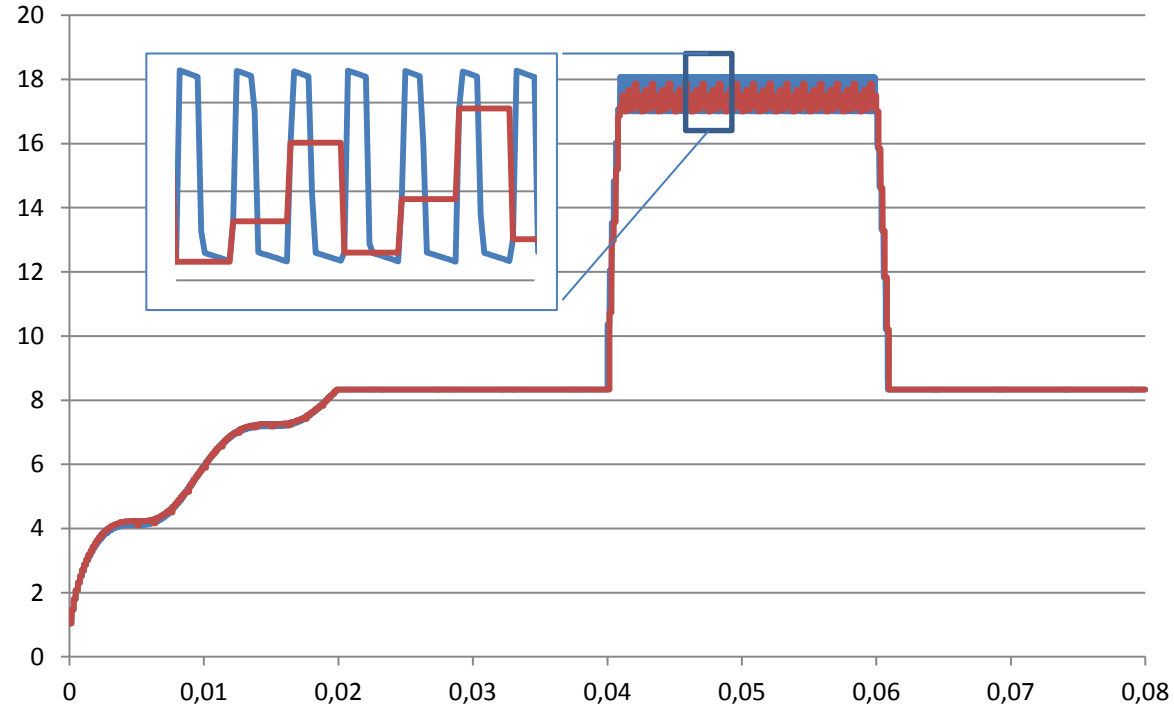
A.3.2 Impuls



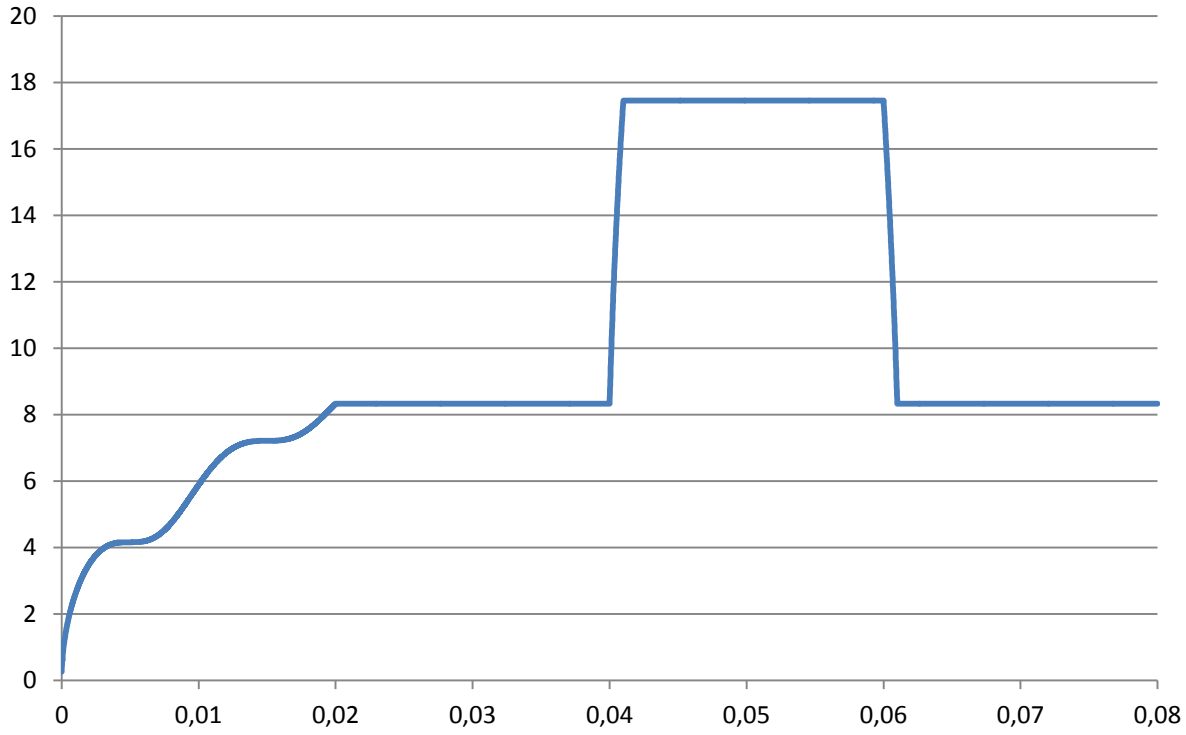
Figur A.5: aRMS - $t = 0.01$ - raud: med h.mod - blå: utan h.mod



Figur A.6: aRMS - $t = 0.02$ - raud: med h.mod - blå: utan h.mod

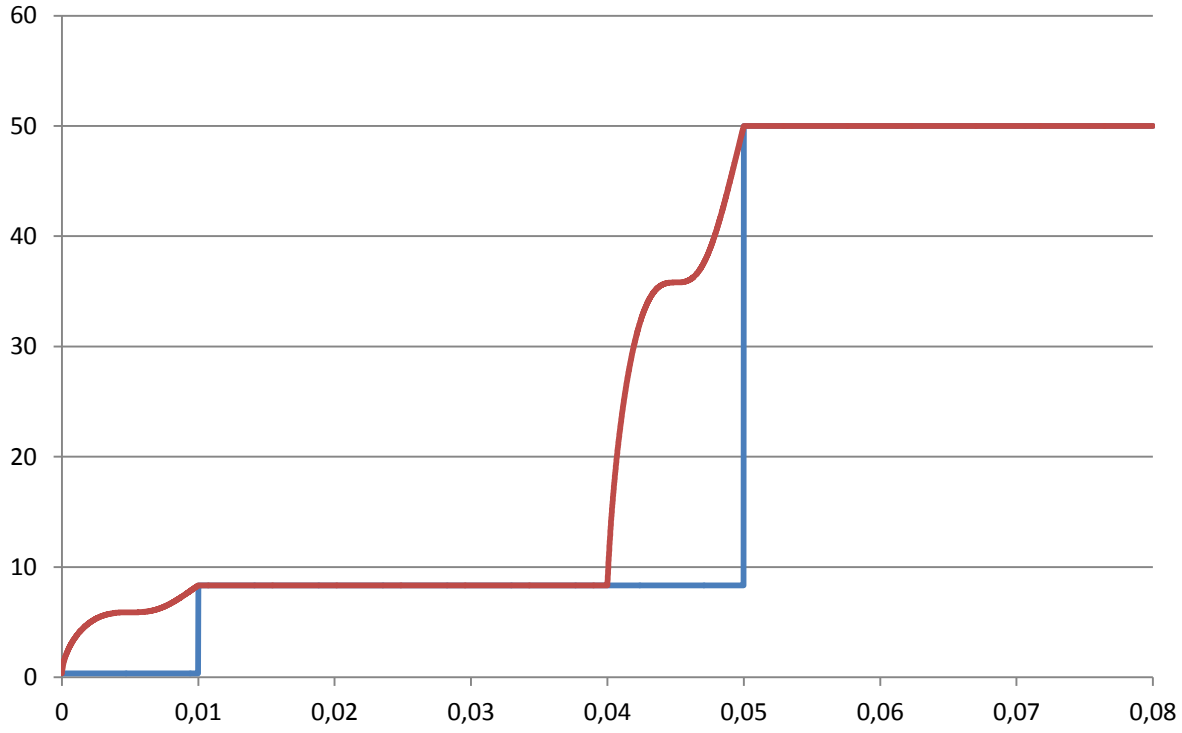


Figur A.7: dRMS - $N = 128$ - $f_{base} = 50Hz$ - raud: med h.mod - blå: utan h.mod

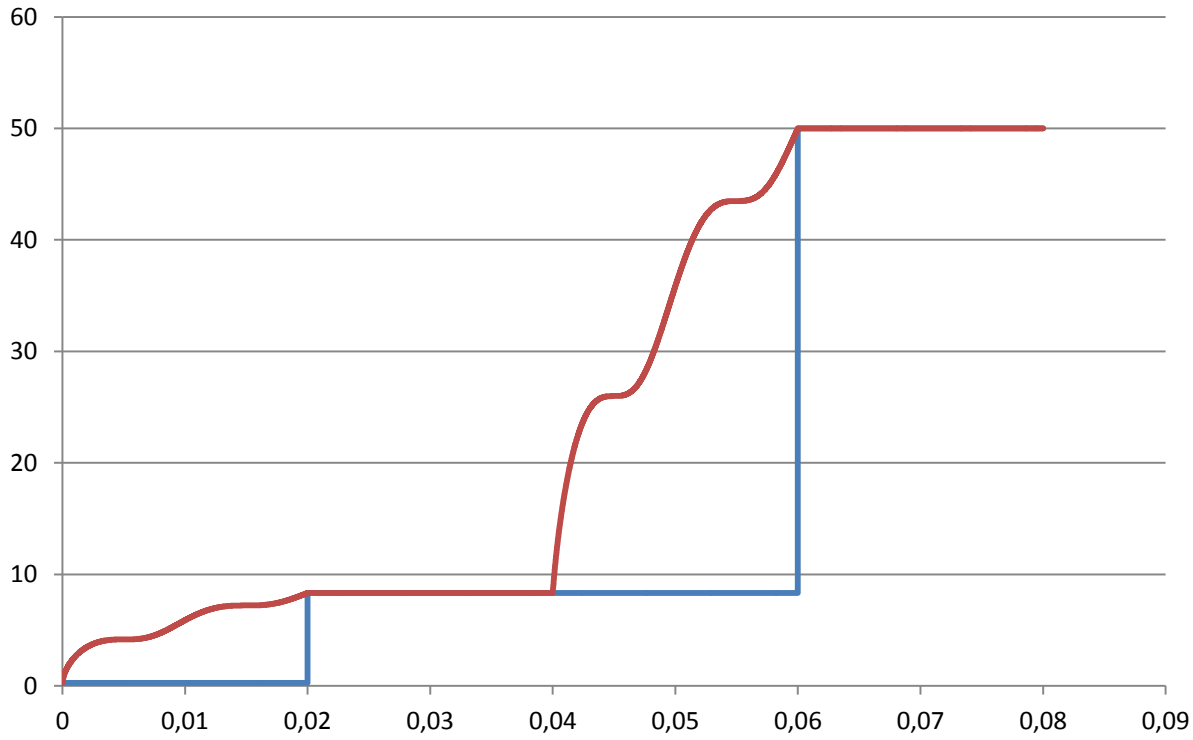


Figur A.8: dRMS - $N = 10000$ - $f_{base} = 50Hz$ - raud: med h.mod - blå: utan h.mod

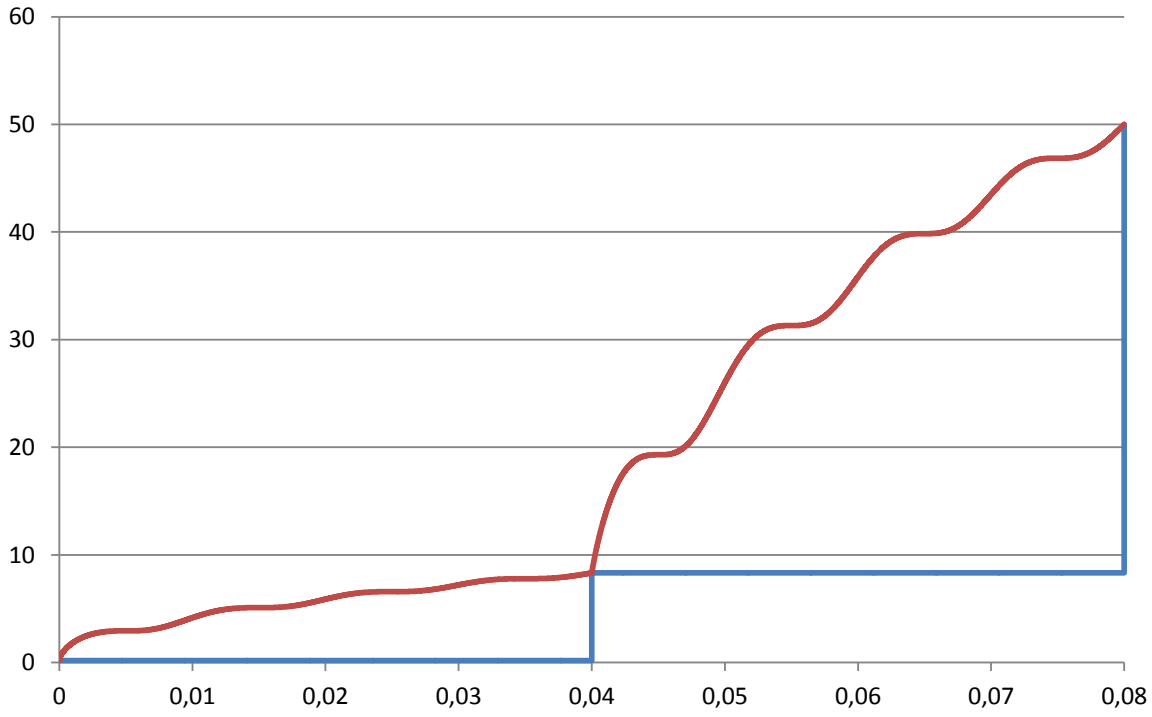
A.3.3 Steg



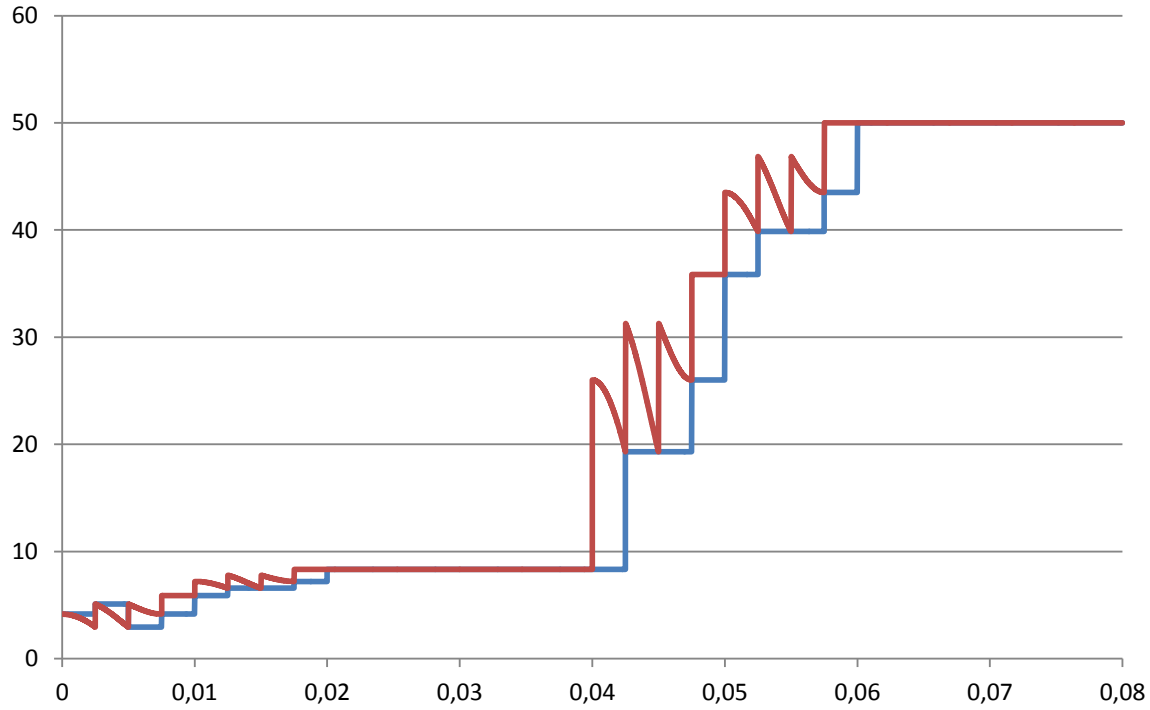
Figur A.9: aRMS - $t = 0.01$ - raud: med h.mod - blå: utan h.mod



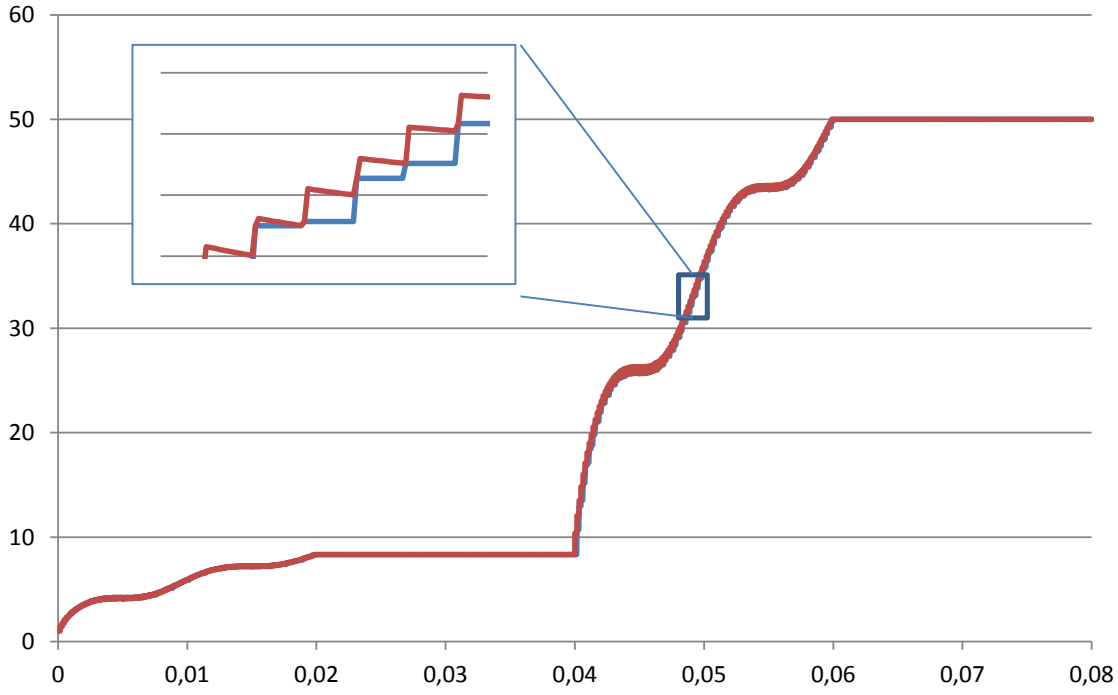
Figur A.10: aRMS - $t = 0.02$ - raud: med h.mod - blå: utan h.mod



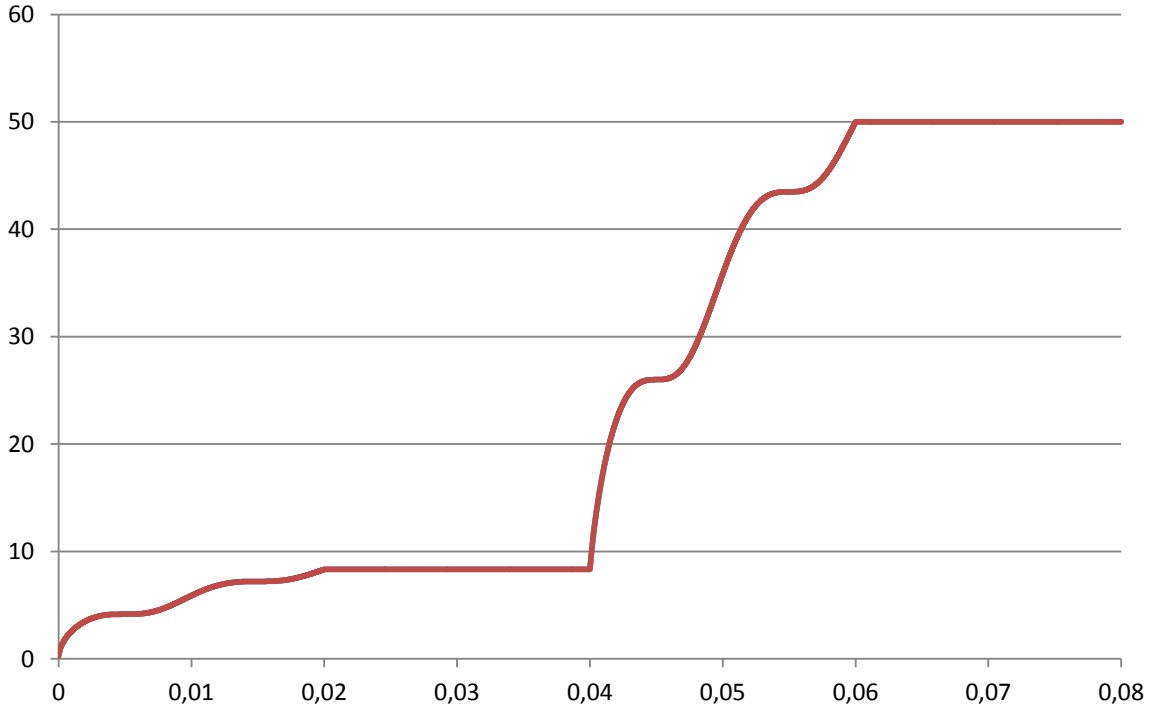
Figur A.11: aRMS - $t = 0.04$ - raud: med h.mod - blå: utan h.mod



Figur A.12: dRMS - $N = 8$ - $f_{base} = 50Hz$ - raud: med h.mod - blå: utan h.mod



Figur A.13: dRMS - $N = 128$ - $f_{base} = 50Hz$ - raud: med h.mod - blå: utan h.mod



Figur A.14: dRMS - N = 10000 - f_{base} = $50Hz$ - raud: med h.mod - blå: utan h.mod

Tillegg B

Fourieranalyse - MODELS

B.1 Diskré Fourier transformasjon (DFT) - kjeldekode

B.1.1 DFT - grunnmodell - kjeldekode

```
MODEL DFT
COMMENT

Discrete fourier transform, 23.05.2012,
by Torstein Stadheim

Base model with output data in scaled rectangular form. The↙
   ↳  magnitude is scaled to rms value and the angle ↙
   ↳ reference is cos(m*2*pi*f_base*t).

Source: Understandig digital signal processing,
        Prentice Hall, Second Edition, 2004
        Chapter 3

ENDCOMMENT

DATA    nHarmData{dflt:8}    -- Number of harmonics to ↙
   ↳ calculate, excluding the DC component. Maximum number ↙
   ↳ of harmonics to calculate is 25, because of limits in ↙
   ↳ the output array.
        f_base{dflt:50}      -- Base frequency (frequency of↙
           ↳  the first harmonic)


INPUT X
```

```
OUTPUT  re[1..26]    -- Real part of the harmonics
        im[1..26]    -- Imaginary part of the harmonics
        dc           -- The DC component of the input

VAR Re[1..26]
    Im[1..26]
    dc
    nSampl
    delta_T
    phi
    nHarmonic
    scale


HISTORY X {DFLT:0}  -- Sets the history of the input to ↙
   ↳ zero

DELAY CELLS dflt: 1/(f_base*timestep)   -- Records one ↙
   ↳ period

INIT
    re[1..26]:=0
    im[1..26]:=0

    IF nHarmData < 27 THEN
        nSampl := 2*(nHarmData+1)   -- Number of samples ↙
           ↳ needed.
        nHarmonic := nHarmData
    ELSE    -- If nHarmonic is larger than the allowed 26
        nHarmonic := 27
        nSampl := 2*nHarmonic
    ENDIF
    scale := sqrt(2)/nSampl
    delta_T := 1 / (f_base*nSampl)
    phi := 2*PI/nSampl
    dc := 0
ENDINIT
EXEC
        -- Calculating the dc-component
        dc := 0
        for n := 0 to nSampl-1 do
            dc := dc + delay(X,delta_T*n,1)
        endfor
        -- Scaling the dc-component
        dc := dc/nSampl
```

```
        -- Calculating DFT
        for m:=1 to nHarmonic do
            Re[m]:=0
            Im[m]:=0
            for n:=0 to nSampl-1 do
                Re[m]:=Re[m]+delay(X,delta_T*n,1)*cos(phi*(↙
                   ↳ nSampl-n)*m)
                Im[m]:=Im[m]-delay(X,delta_T*n,1)*sin(phi*(↙
                   ↳ nSampl-n)*m)
            endfor
            Re[m]:=Re[m]*scale
            Im[m]:=Im[m]*scale
        endfor

ENDEXEC
ENDMODEL
```

B.1.2 DFT med effektivverdi og fase som utgangsdata - kjeldekode

```
MODEL DFT
COMMENT

Discrete fourier transform, 23.05.2012,
by Torstein Stadheim

Modified version of the basic DFT-model. Output is ↙
   ↳ magnitude and angle.
The angle can be in either radians or degrees, decided from↙
   ↳  the data parameter degrees.
If degrees is 1, then the angle is given in degrees. ↙
   ↳ Otherwise it is given in radians.
The angle reference for an m harmonic is cos(m*2*pi*f_base*↙
   ↳ t).

Source: Understandig digital signal processing,
        Prentice Hall, Second Edition, 2004
        Chapter 3

ENDCOMMENT

DATA    nHarmData{dflt:8}   -- Number of harmonics to ↙
   ↳ calculate, excluding the DC component. Maximum number ↙
   ↳ of harmonics to calculate is 25, because of limits in ↙
   ↳ the output array.
```

```
        f_base{dflt:50}     -- Base frequency (frequency of↙
           ↳  the first harmonic)
        degrees{dflt:1}     -- Angle of the harmonic ↙
           ↳ referenced to a cosinus signal with the same ↙
           ↳ frequency.


INPUT X

OUTPUT  Mag[1..26]      -- Rms-magnitude of the harmonics
        Phase[1..26]    -- Angle of the harmonics
        dc              -- The DC component of the input

VAR Re[1..26]
    Im[1..26]
    Mag[1..26]
    Phase[1..26]
    dc
    nSampl
    delta_T
    phi
    nHarmonic
    scale
    aRef


HISTORY X {DFLT:0}  -- Sets the history of the input to ↙
   ↳ zero

DELAY CELLS dflt: 1/(f_base*timestep)   -- Records one ↙
   ↳ period

INIT
    re[1..26]:=0
    im[1..26]:=0

    IF nHarmData < 27 THEN
        nSampl := 2*nHarmData   -- Number of samples needed↙
           ↳ .
        nHarmonic := nHarmData
    ELSE    -- If nHarmonic is larger than the allowed 26
        nHarmonic := 26
        nSampl := 2*26
    ENDIF
    scale := sqrt(2)/nSampl
    delta_T := 1 / (f_base*nSampl)
```

```
    phi := 2*PI/nSampl
    dc := 0
ENDINIT
EXEC
        dc := 0
        for n := 0 to nSampl-1 do
            dc := dc + delay(X,delta_T*n,1)
        endfor

        -- Calculating DFT
        for m:=1 to nHarmonic do
            Re[m]:=0
            Im[m]:=0
            for n:=0 to nSampl-1 do
                Re[m]:=Re[m]+delay(X,delta_T*n,1)*cos(phi*(↙
                   ↪ nSampl-n)*m)
                Im[m]:=Im[m]-delay(X,delta_T*n,1)*sin(phi*(↙
                   ↪ nSampl-n)*m)
            endfor
        endfor

        -- Calculating the phase and angle
        for m:=1 to nHarmonic do
            Mag[m]:=sqrt(Re[m]*Re[m]+Im[m]*Im[m])*scale
            if Mag[m] >= 1E-10 then -- inhibit oscillation ↙
               ↪ on the angle if magnitude is low
                if Re[m] > 1E-10 then -- 1 and 4 quadrant
                    Phase[m]:=atan(Im[m]/Re[m])
                elsif Re[m] < -1E-10 then -- 2 and 3 ↙
                   ↪ quadrant
                    Phase[m]:=PI+atan(Im[m]/Re[m])
                else -- avoid dividing with zero
                    if Im[m] > 0 then
                        Phase[m]:=PI/2
                    else
                        Phase[m]:=3*PI/2
                    endif
                endif
                aRef:=(t mod (1/(f_base*m)))*2*PI*f_base*m ↙
                   ↪ -- calculating reference angle
                Phase[m]:=Phase[m]-aRef
                while Phase[m] < 0 do
                    Phase[m]:=Phase[m]+2*PI
                endwhile
            else
                Phase[m]:=0
```

```
            endif

            if degrees = 1 then -- converts from radians to↙
               ↪  degrees
                Phase[m]:=Phase[m]*180/PI
            endif
        endfor

ENDEXEC
ENDMODEL
```

B.2 Fast Fourier transform(FFT) - kjeldekode

B.2.1 Radix-2 grunnmodell - kjeldekode

```
MODEL FFT
COMMENT

RADIX-2 FFT calculator for calculating 8 point FFT,
by Torstein Stadheim, 16.05.2012

Source: Understanding Digital Signal Processing, Second ↙
   ↪ Edition
        Richard G. Lyons, Prentice Hall, 2004
        Chapter 4

        Base model with output data in scaled rectangular ↙
           ↪ form. The magnitude is scaled to rms value and↙
           ↪  the angle reference is cos(m*2*pi*f_base*t).
ENDCOMMENT

DATA    f_base  {dflt:50}   -- Frequency of the first ↙
   ↪ harmonic

INPUT   x   -- The input signal which is to be analyzed

OUTPUT  re[1..3]        -- Array containing RMS values of ↙
   ↪ the frequency components
                        -- whereas index 1 is the DC, 2 is ↙
                           ↪ the first harmonic...
        im[1..3]        -- Array containing angular values ↙
           ↪ of the frequency components
                        -- whereas index 1 is the DC, 2 is ↙
                           ↪ the first harmonic...
        dc
```

```
VAR re[1..3], im[1..3], dc
    alpha8Re[0..7]  -- Angular constants
    alpha8Im[0..7]  -- Angular constants
    step1[0..7]
    step2Re[0..7]
    step2Im[0..7]
    step3Re[0..3]
    step3Im[0..3]
    delta_T[0..7]
    scale

DELAY CELLS dflt: 1/(f_base*timestep) -- Need at least one ↙
   ↳ first harmonic periode of sampling

HISTORY X {DFLT:0}  -- Fills the signal with zeroes when ↙
   ↳ time <= 0

INIT
    -- Sets the starting values to zero
    re[1..3]:=0
    im[1..3]:=0

    -- Fill the tables with angular and time constants
    FOR i:= 0 TO 7 DO
        alpha8Re[i] := cos(2*PI*i/8)
        alpha8Im[i] := sin(2*PI*i/8)
        delta_T[i] := i/(f_base*8)
    ENDFOR
    scale := sqrt(2)/8

ENDINIT
EXEC

-- Hardcoding step 1 of figure 4-5 page 13 chapter 4
    step1[0] := delay(x,delta_T[0],1) + delay(x,delta_T↙
       ↳ [4],1)
    step1[1] := delay(x,delta_T[0],1) - delay(x,delta_T↙
       ↳ [4],1)
    step1[2] := delay(x,delta_T[2],1) + delay(x,delta_T↙
       ↳ [6],1)
    step1[3] := delay(x,delta_T[2],1) - delay(x,delta_T↙
       ↳ [6],1)
    step1[4] := delay(x,delta_T[1],1) + delay(x,delta_T↙
       ↳ [5],1)
```

```
    step1[5] := delay(x,delta_T[1],1) - delay(x,delta_T↙
       ↳ [5],1)
    step1[6] := delay(x,delta_T[3],1) + delay(x,delta_T↙
       ↳ [7],1)
    step1[7] := delay(x,delta_T[3],1) - delay(x,delta_T↙
       ↳ [7],1)

-- Hardcoding step 2 of figure 4-5 page 13 chapter 4, to ↙
   ↳ avoid the use of the modolus function (save ↙
   ↳ calculating time)
    step2Re[0] := step1[0] + step1[2]*alpha8Re[0]
    step2Re[1] := step1[1] + step1[3]*alpha8Re[2]
    step2Re[2] := step1[0] + step1[2]*alpha8Re[4]
    step2Re[3] := step1[1] + step1[3]*alpha8Re[6]
    step2Re[4] := step1[4] + step1[6]*alpha8Re[0]
    step2Re[5] := step1[5] + step1[7]*alpha8Re[2]
    step2Re[6] := step1[4] + step1[6]*alpha8Re[4]
    step2Re[7] := step1[5] + step1[7]*alpha8Re[6]
    step2Im[0] := step1[2]*alpha8Im[0]
    step2Im[1] := step1[3]*alpha8Im[2]
    step2Im[2] := step1[2]*alpha8Im[4]
    step2Im[3] := step1[3]*alpha8Im[6]
    step2Im[4] := step1[6]*alpha8Im[0]
    step2Im[5] := step1[7]*alpha8Im[2]
    step2Im[6] := step1[6]*alpha8Im[4]
    step2Im[7] := step1[7]*alpha8Im[6]

-- Hardcoding step 3 of figure 4-5 page 13 chapter 4, to ↙
   ↳ avoid use of the modules funtion
    step3Re[0] := step2Re[0] + step2Re[4]*alpha8Re[0] - ↙
       ↳ step2Im[4]*alpha8Im[0]
    step3Im[0] := step2Im[0] + step2Re[4]*alpha8Im[0] + ↙
       ↳ step2Im[4]*alpha8Re[0]
    step3Re[1] := step2Re[1] + step2Re[5]*alpha8Re[1] - ↙
       ↳ step2Im[5]*alpha8Im[1]
    step3Im[1] := step2Im[1] + step2Re[5]*alpha8Im[1] + ↙
       ↳ step2Im[5]*alpha8Re[1]
    step3Re[2] := step2Re[2] + step2Re[6]*alpha8Re[2] - ↙
       ↳ step2Im[6]*alpha8Im[2]
    step3Im[2] := step2Im[2] + step2Re[6]*alpha8Im[2] + ↙
       ↳ step2Im[6]*alpha8Re[2]
    step3Re[3] := step2Re[3] + step2Re[7]*alpha8Re[3] - ↙
       ↳ step2Im[7]*alpha8Im[3]
    step3Im[3] := step2Im[3] + step2Re[7]*alpha8Im[3] + ↙
       ↳ step2Im[7]*alpha8Re[3]
```

```
    -- Scaling AC the output
    FOR i:=1 TO 3 DO
        Re[i] := step3Re[i]*scale
        Im[i] := step3Im[i]*scale
    ENDFOR

    -- Scaling the DC output
    dc := sqrt(step3Re[0]**2 + step3Im[0]**2)/8

ENDEXEC
ENDMODEL
```

B.2.2 Radix-2 med effektivverdi og fase som utgangsdata - kjeldekode

```
MODEL FFT
COMMENT

RADIX-2 FFT calculator for calculating 8 point FFT,
by Torstein Stadheim, 16.05.2012

Source: Understanding Digital Signal Processing, Second ↙
   ↳ Edition
        Richard G. Lyons, Prentice Hall, 2004
        Chapter 4

        Output is in scaled rectangular form

        Angle

ENDCOMMENT

DATA    f_base  {dflt:50}   -- Frequency of the first ↙
   ↳ harmonic
        degrees {dflt:0}    -- 1 for angles in degrees. ↙
           ↳ Radians otherwise.

INPUT   x   -- The input signal which is to be analyzed

OUTPUT  mag[1..3]       -- Array containing RMS values of ↙
   ↳ the frequency components
                        -- whereas index 1 is the DC, 2 is ↙
                           ↳ the first harmonic...
        phase[1..3]     -- Array containing angular values ↙
           ↳ of the frequency components
```

```
                        -- whereas index 1 is the DC, 2 is ↙
                           ↳ the first harmonic...
        dc


VAR re[1..3], im[1..3], dc
    alpha8Re[0..7]  -- Angular constants
    alpha8Im[0..7]  -- Angular constants
    step1[0..7]
    step2Re[0..7]
    step2Im[0..7]
    step3Re[0..3]
    step3Im[0..3]
    delta_T[0..7]
    scale
    mag[1..3], phase[1..3], aref

DELAY CELLS dflt: 1/(f_base*timestep) -- Need at least one ↙
   ↳ first harmonic periode of sampling

HISTORY X {DFLT:0}  -- Fills the signal with zeroes when ↙
   ↳ time <= 0

INIT
    -- Sets the starting values to zero
    re[1..3]:=0
    im[1..3]:=0

    -- Fill the tables with angular and time constants
    FOR i:= 0 TO 7 DO
        alpha8Re[i] := cos(2*PI*i/8)
        alpha8Im[i] := sin(2*PI*i/8)
        delta_T[i] := i/(f_base*8)
    ENDFOR
    scale := sqrt(2)/8

ENDINIT
EXEC

-- Hardcoding step 1 of figure 4-5 page 13 chapter 4
    step1[0] := delay(x,delta_T[0],1) + delay(x,delta_T↙
       ↳ [4],1)
    step1[1] := delay(x,delta_T[0],1) - delay(x,delta_T↙
       ↳ [4],1)
    step1[2] := delay(x,delta_T[2],1) + delay(x,delta_T↙
       ↳ [6],1)
```

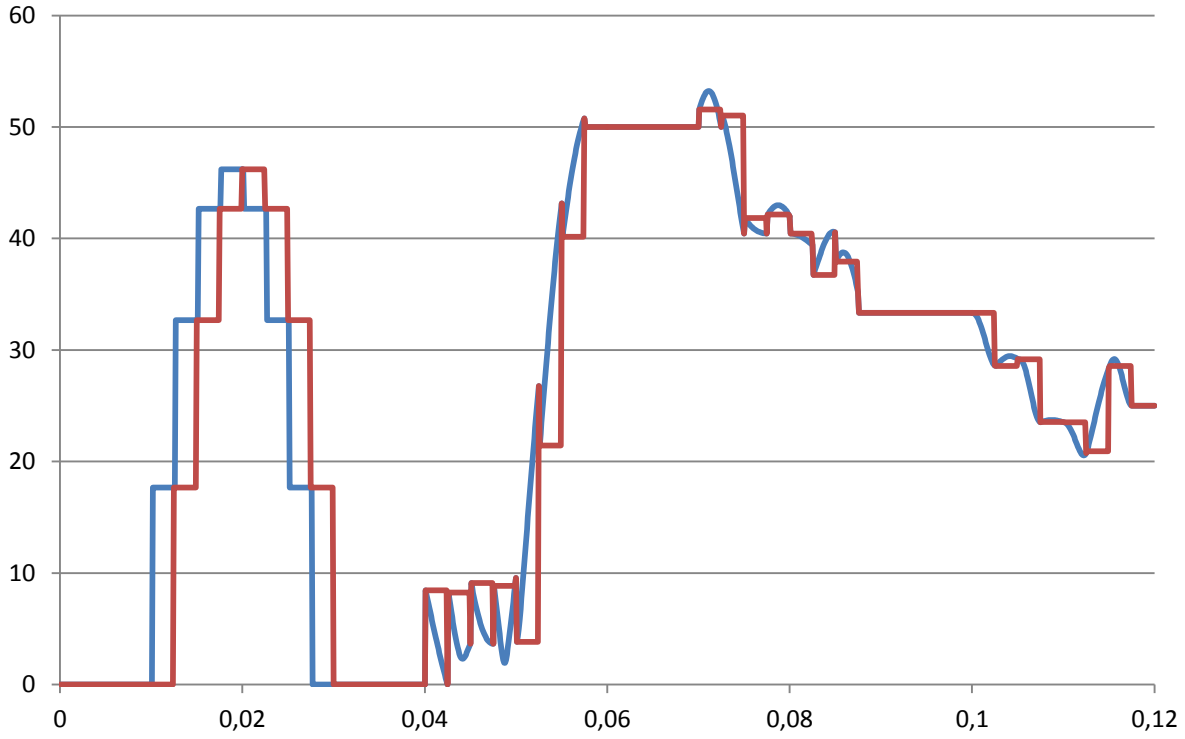
```
    step1[3] := delay(x,delta_T[2],1) - delay(x,delta_T↙
       ↳ [6],1)
    step1[4] := delay(x,delta_T[1],1) + delay(x,delta_T↙
       ↳ [5],1)
    step1[5] := delay(x,delta_T[1],1) - delay(x,delta_T↙
       ↳ [5],1)
    step1[6] := delay(x,delta_T[3],1) + delay(x,delta_T↙
       ↳ [7],1)
    step1[7] := delay(x,delta_T[3],1) - delay(x,delta_T↙
       ↳ [7],1)

-- Hardcoding step 2 of figure 4-5 page 13 chapter 4, to ↙
   ↳ avoid the use of the modolus function (save ↙
   ↳ calculating time)
    step2Re[0] := step1[0] + step1[2]*alpha8Re[0]
    step2Re[1] := step1[1] + step1[3]*alpha8Re[2]
    step2Re[2] := step1[0] + step1[2]*alpha8Re[4]
    step2Re[3] := step1[1] + step1[3]*alpha8Re[6]
    step2Re[4] := step1[4] + step1[6]*alpha8Re[0]
    step2Re[5] := step1[5] + step1[7]*alpha8Re[2]
    step2Re[6] := step1[4] + step1[6]*alpha8Re[4]
    step2Re[7] := step1[5] + step1[7]*alpha8Re[6]
    step2Im[0] := step1[2]*alpha8Im[0]
    step2Im[1] := step1[3]*alpha8Im[2]
    step2Im[2] := step1[2]*alpha8Im[4]
    step2Im[3] := step1[3]*alpha8Im[6]
    step2Im[4] := step1[6]*alpha8Im[0]
    step2Im[5] := step1[7]*alpha8Im[2]
    step2Im[6] := step1[6]*alpha8Im[4]
    step2Im[7] := step1[7]*alpha8Im[6]

-- Hardcoding step 3 of figure 4-5 page 13 chapter 4, to ↙
   ↳ avoid use of the modules funtion
    step3Re[0] := step2Re[0] + step2Re[4]*alpha8Re[0] - ↙
       ↳ step2Im[4]*alpha8Im[0]
    step3Im[0] := step2Im[0] + step2Re[4]*alpha8Im[0] + ↙
       ↳ step2Im[4]*alpha8Re[0]
    step3Re[1] := step2Re[1] + step2Re[5]*alpha8Re[1] - ↙
       ↳ step2Im[5]*alpha8Im[1]
    step3Im[1] := step2Im[1] + step2Re[5]*alpha8Im[1] + ↙
       ↳ step2Im[5]*alpha8Re[1]
    step3Re[2] := step2Re[2] + step2Re[6]*alpha8Re[2] - ↙
       ↳ step2Im[6]*alpha8Im[2]
    step3Im[2] := step2Im[2] + step2Re[6]*alpha8Im[2] + ↙
       ↳ step2Im[6]*alpha8Re[2]
```

```
step3Re[3] := step2Re[3] + step2Re[7]*alpha8Re[3] - ↙
   ↳ step2Im[7]*alpha8Im[3]
step3Im[3] := step2Im[3] + step2Re[7]*alpha8Im[3] + ↙
   ↳ step2Im[7]*alpha8Re[3]

-- Scaling AC the output
for i:=1 to 3 do
    Re[i] := step3Re[i]*scale
    Im[i] := step3Im[i]*scale
endfor

-- Scaling the DC output
dc := sqrt(step3Re[0]**2 + step3Im[0]**2)/8

-- Calculating the phase and magnitude
for m:=1 to 3 do
    Mag[m]:=sqrt(Re[m]*Re[m]+Im[m]*Im[m])
    if Mag[m] >= 1E-10 then -- inhibit oscillation on ↙
       ↳ the angle if magnitude is low
        if Re[m] > 1E-10 then -- 1 and 4 quadrant
            Phase[m]:=atan(Im[m]/Re[m])
        elsif Re[m] < -1E-10 then -- 2 and 3 quadrant
            Phase[m]:=PI+atan(Im[m]/Re[m])
        else -- avoid dividing with zero
            if Im[m] > 0 then
                Phase[m]:=PI/2
            else
                Phase[m]:=3*PI/2
            endif
        endif
        aRef:=(t mod (1/(f_base*m)))*2*PI*f_base*m -- ↙
           ↳ calculating reference angle
        Phase[m]:=Phase[m]-aRef
        while Phase[m] < 0 do
            Phase[m]:=Phase[m]+2*PI
        endwhile
    else
        Phase[m]:=0
    endif

    if degrees = 1 then -- converts from radians to ↙
       ↳ degrees
        Phase[m]:=Phase[m]*180/PI
    endif
endfor
```

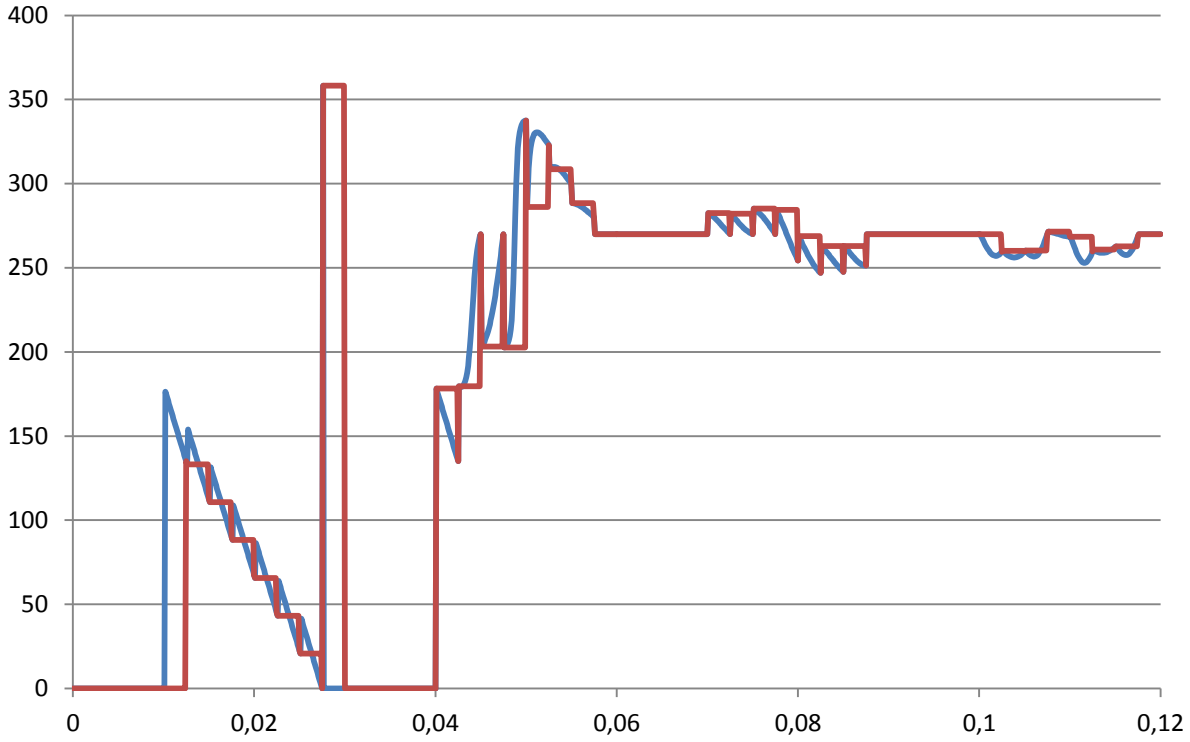
```
ENDEXEC
ENDMODEL
```

B.3 Resultat frå simuleringar

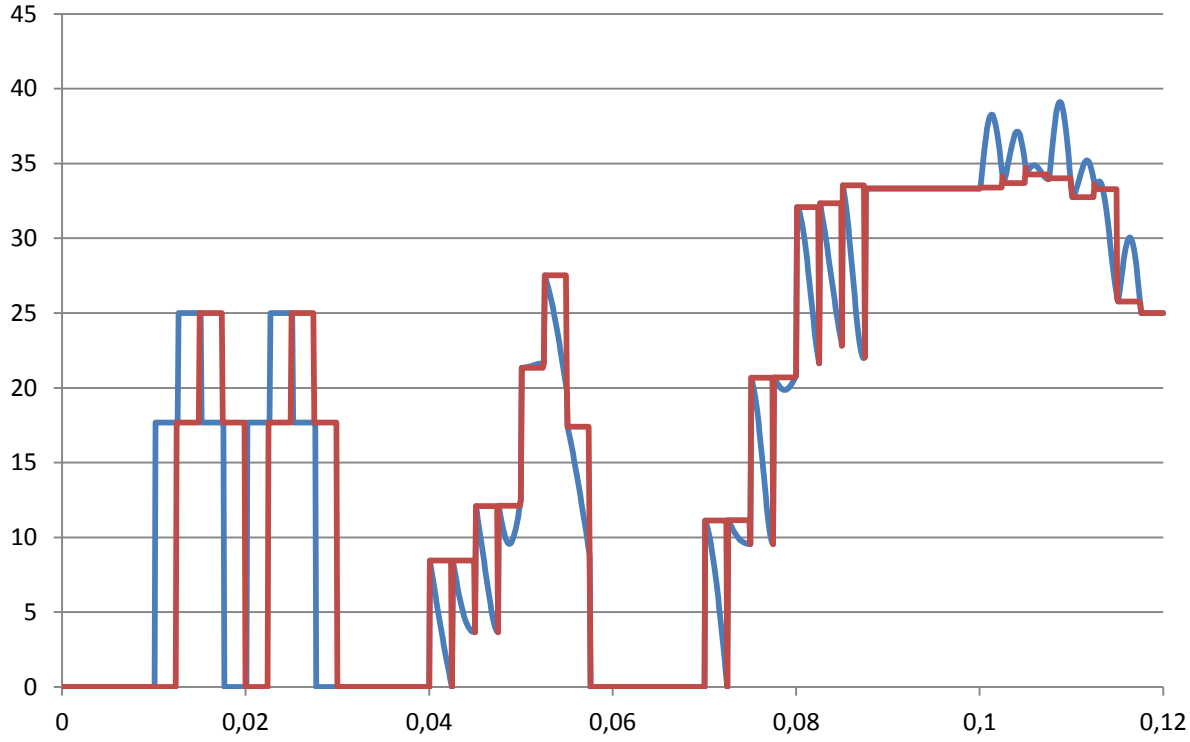
I figur 3.14 til B.5 er resulatata frå fourieranalysen vist. Blå graf er utan hastigheits-modifisering og raud graf er med.



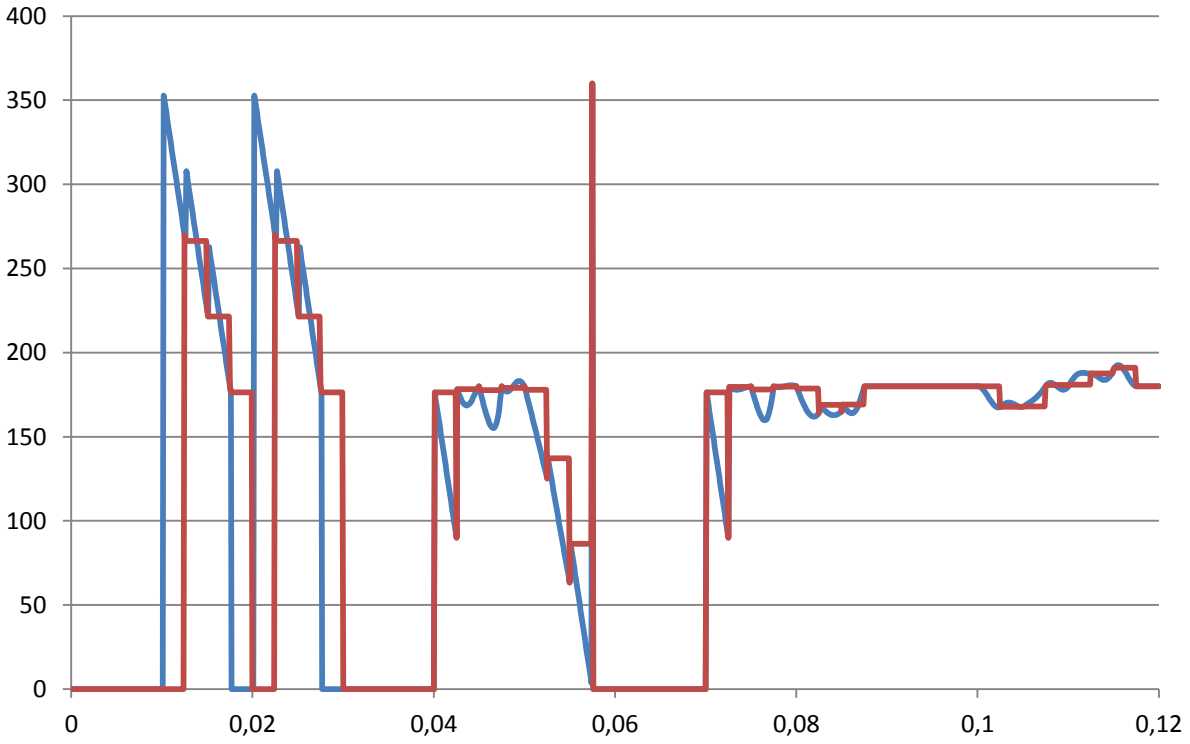
Figur B.1: 1 harmoniske komponent - effektivverdi - tilhøyrande signal 3.13



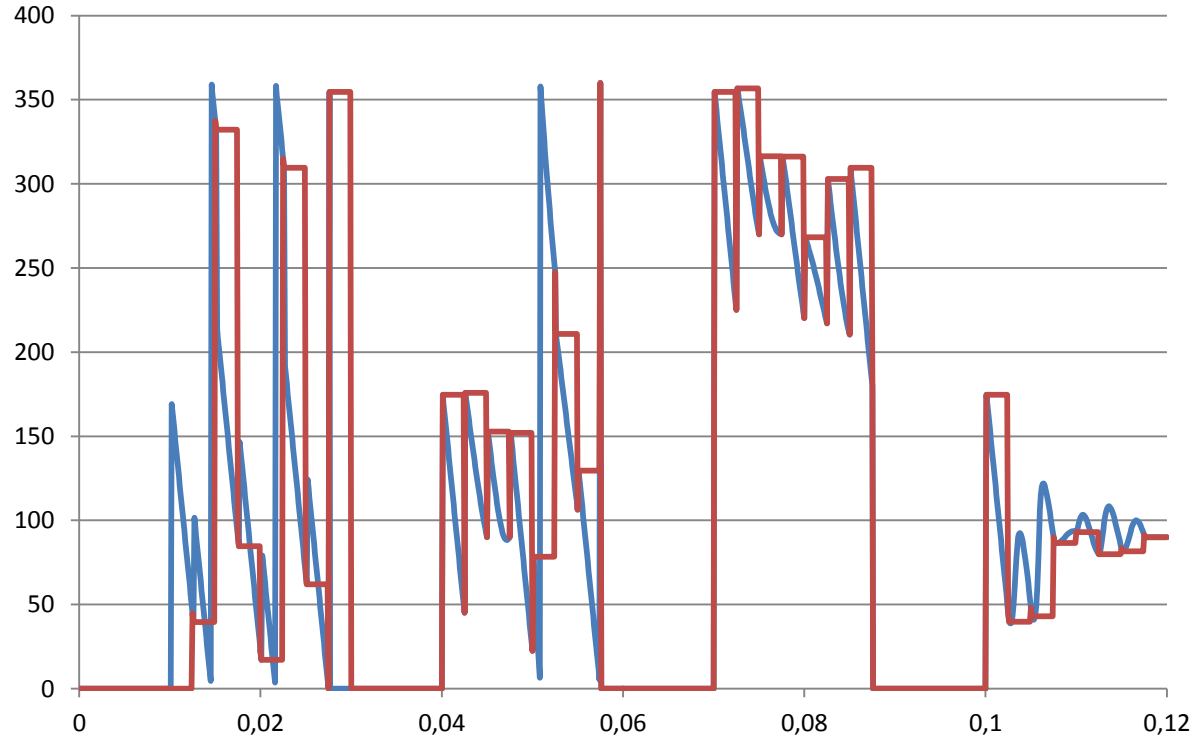
Figur B.2: 1 harmoniske komponent - fase - tilhøyrande signal 3.13



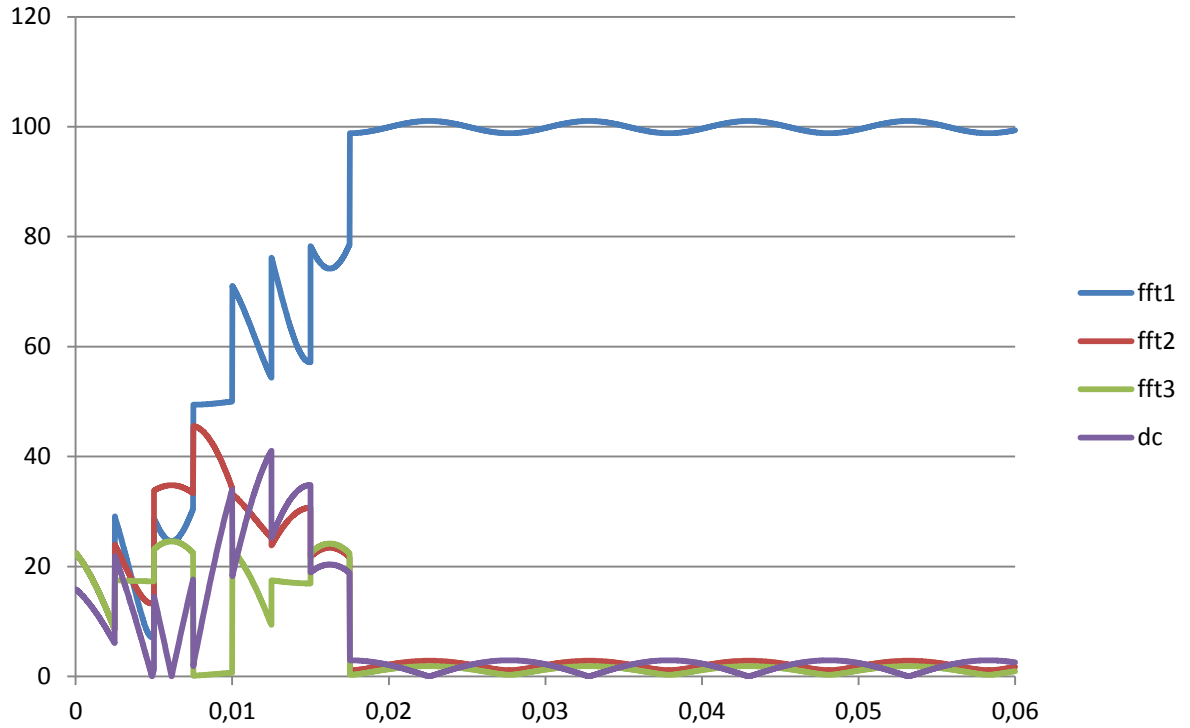
Figur B.3: 2 harmoniske komponent - effektivverdi - tilhøyrande signal 3.13



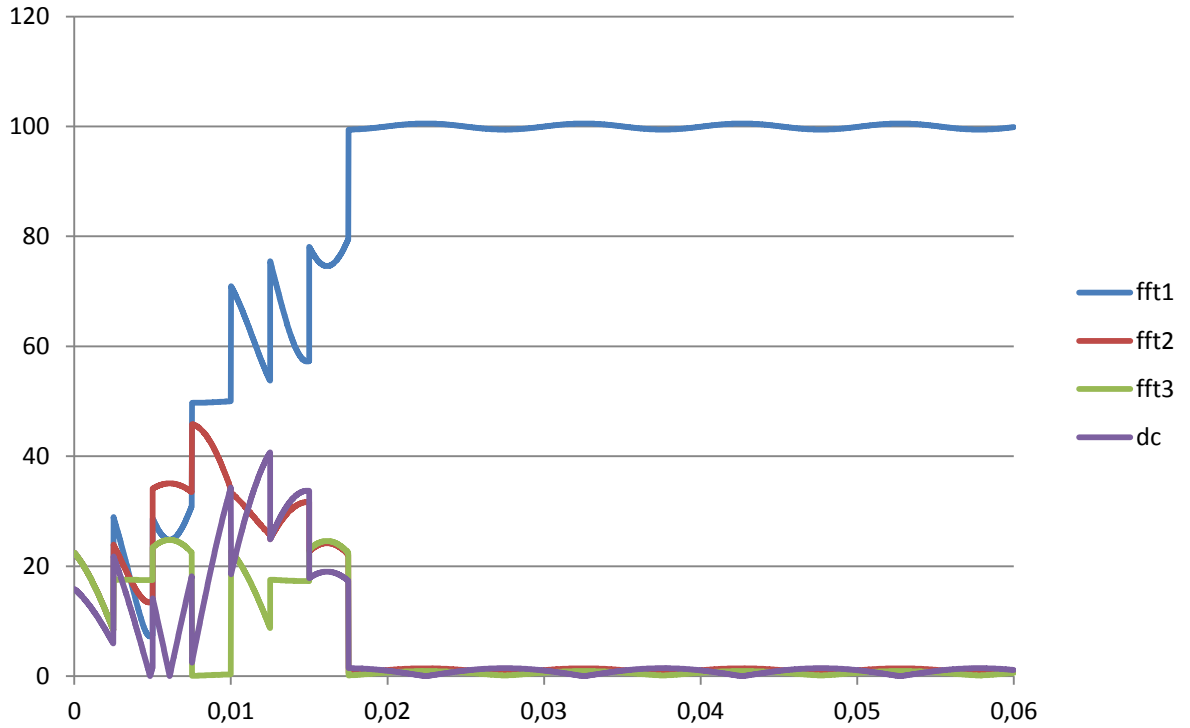
Figur B.4: 2 harmoniske komponent - fase - tilhøyrande signal 3.13



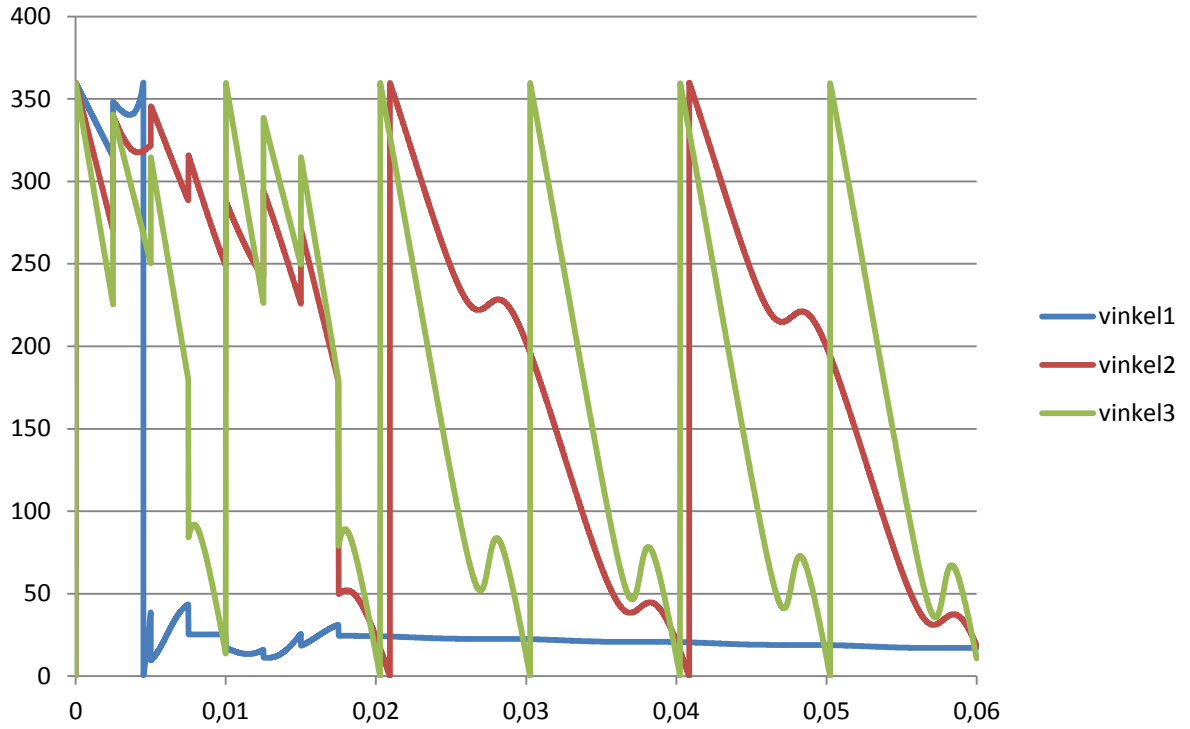
Figur B.5: 3 harmoniske komponent - fase - tilhøyrande signal 3.13



Figur B.6: Effektivverdi når inngangssignal har ein frekvens på $49Hz$ og effektivverdi på 100. $f_{base} = 50Hz$



Figur B.7: Effektivverdi når inngangssignal har ein frekvens på $49.5Hz$ og effektivverdi på 100. $f_{base} = 50Hz$



Figur B.8: Fase når inngangssignal har ein frekvens på $49.5Hz$ og fase på 26 grader. $f_{base} = 50Hz$

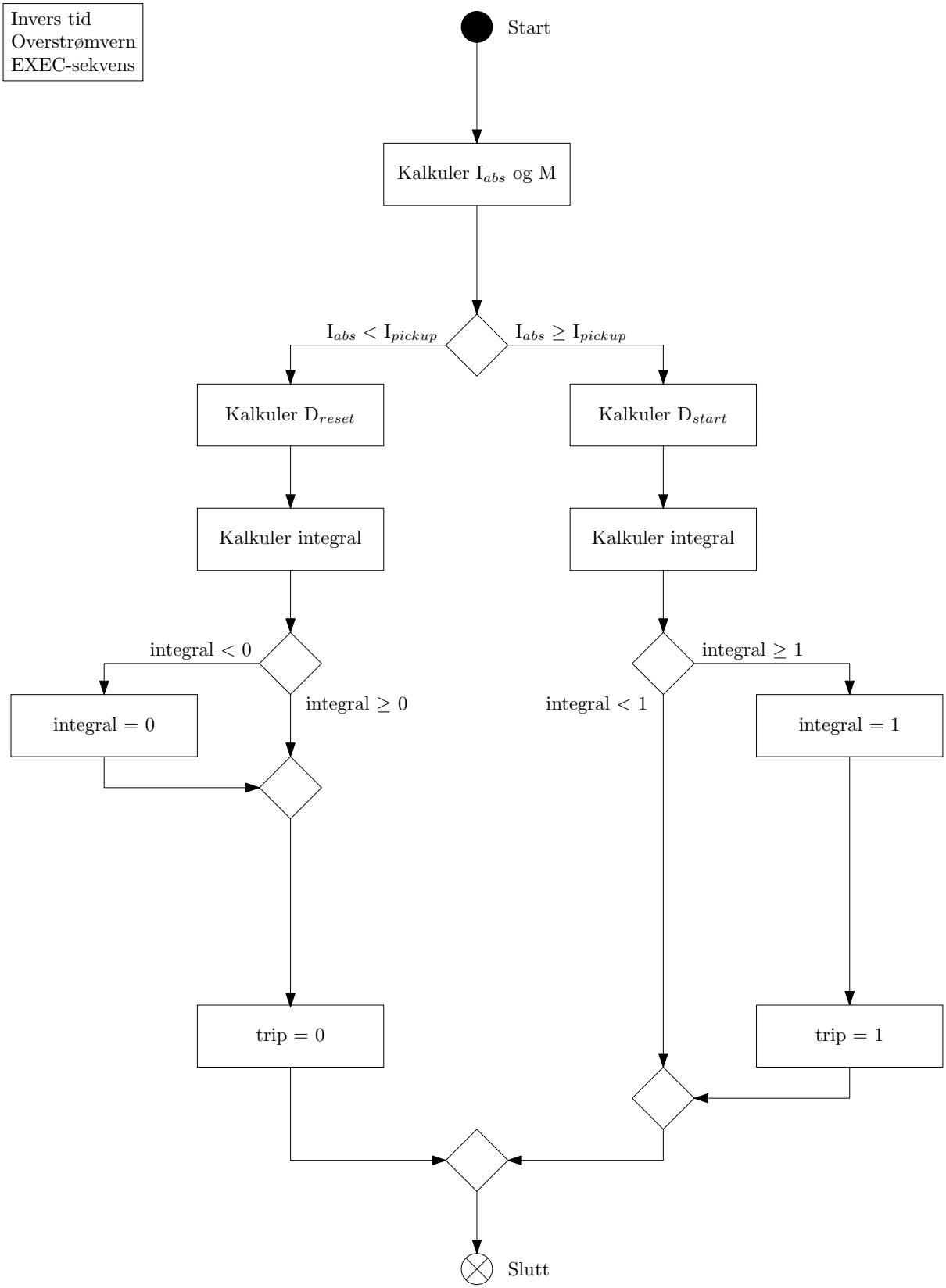
Tillegg C

Overstraumvern - MODELS

C.1 Flytskjema

C.1.1 Invers tid

Figur C.1: Flytskjema - inverstid - INIT



Figur C.2: Flytskjema - inverstid - EXEC

C.2 Kjeldekode

C.2.1 Konstant tid

```
MODEL oc_c

COMMENT

Constant time overcurrent relay, 01.06.2012,
by Torstein Stadheim

No processing of the input signal. Timer starts when signal
   ↪ higher than the limit and resets when its lower.

Trip when timer times out.

Recommend used together with rms-model.

ENDCOMMENT

DATA    limit { dflt:500 } -- Tripvalue [A]
        t_s { dflt:0.0 } -- Delay of trip [s]

INPUT   current          -- Input signal [A]

OUTPUT  trip             -- 1 if trip, 0 otherwise

VAR     time    -- Timer
        trip
INIT
    time := 0
    trip := 0
ENDINIT

EXEC
    IF current >= limit THEN -- Start
        time := time + timestep
        IF time >= t_s THEN -- Trip
            trip := 1
        ENDIF
    ELSE    -- Reset
        trip := 0
        time := 0
    ENDIF
ENDEXEC
ENDMODEL
```

C.2.2 Invers tid

```
MODEL oc_i

COMMENT
Inverse time overcurrent relay,
by Torstein Stadheim, 04.06.2012

The mathematecial formula used is:

    M => 1

    t_start = TimeDial*(A/(M^P - 1) + B) + K

    M < 1

    t_reset = TimeDial*(t_reset_data/(M^Q - 1))

This formula is able to simulate the IEC and ANSI inverse ↙
   ↳ time standards.

Source:
        IEEE Std. C37.112-1996
        Siemens SIPROTEC 7SA6 Distance protection manual p.↙
           ↳  594-596

The default values given in the data parameteres is ↙
   ↳ constants for a moderately inverse curve, given in ↙
   ↳ IEEE Std. C37.112-1996.
ENDCOMMENT

DATA    I_pickup        { dflt:5 }      -- Pickup current
        TimeDial        { dflt:1 }      -- Inverse ↙
           ↳ characteristic constant
        A               { dflt:0.0515 } -- Inverse ↙
           ↳ characteristic constant
        B               { dflt:0.1140 } -- Inverse ↙
           ↳ characteristic constant
        K               { dflt:0 }      -- Inverse ↙
           ↳ characteristic constant
        P               { dflt:0.02 }   -- Inverse ↙
           ↳ characteristic constant
        Q               { dflt:2 }      -- Inverse ↙
           ↳ characteristic constant
        t_reset_data    { dflt:4.85 }   -- Inverse ↙
           ↳ characteristic constant
```

```
INPUT current

OUTPUT  trip

VAR trip
    integral
    t_start      -- time constant if M => 1
    t_reset      -- time constant if M < 1
    M            -- Multiple (input current / pickup current↙
       ↳ )
    I_abs        -- Absolute value of the current
    f_t          -- Tourqe function

INIT
    trip := 0
    f_t := 0
    integral := 0
    M := 0
ENDINIT

EXEC
    -- Calculates the ratio of pickup current to input ↙
       ↳ current
    I_abs := abs( current )
    M := I_abs / I_pickup

    -- I_abs >= I_pickup
    IF M >= 1 THEN

        -- Calculates the time constant
        t_start := TimeDial * ( B + A * recip(M**P - 1) ) +↙
           ↳  K

        -- Updates the integral sum
        f_t := 1 * recip(t_start)
        integral := integral + f_t * timestep

        -- Stops the integral from exceeding the sum of 1, ↙
           ↳ and trips.
        IF integral >= 1 THEN
            integral := 1
            trip := 1
        ENDIF

    -- I_abs < I_pickup AND reset = 1
```

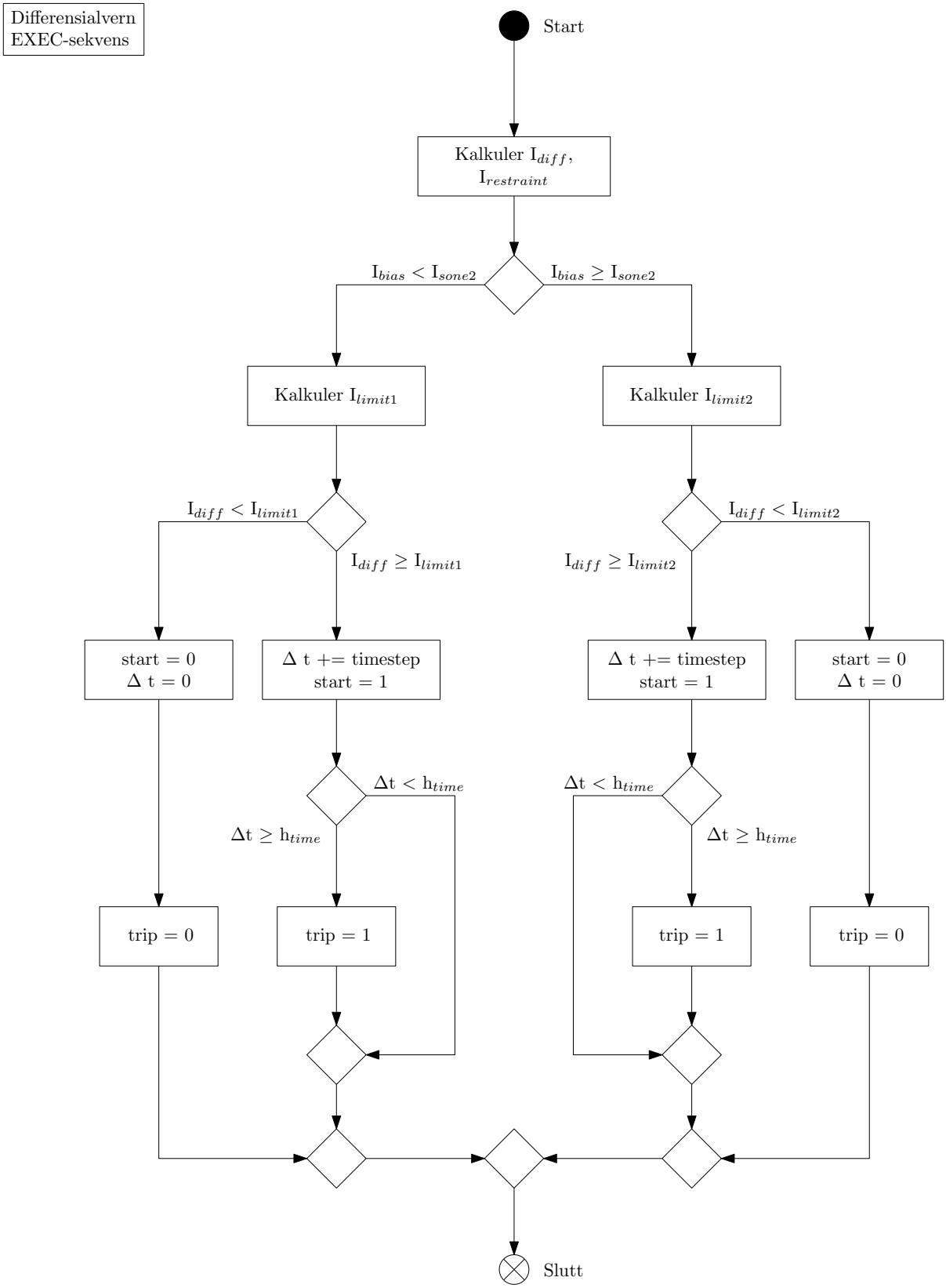
```
    ELSE

        -- Calculates the time constant
        t_reset := TimeDial * ( t_reset_data * recip(M**Q -↙
           ↳  1) )

        -- Updates the integral sum
        f_t := 1 * recip(t_reset)
        integral := integral + f_t * timestep

        -- Stops the integral from exceeding the lower ↙
           ↳ limit of 0.
        IF integral < 0 THEN
            integral := 0
        ENDIF

        trip := 0

    ENDIF

ENDEXEC
ENDMODEL
```

Tillegg D

Differensialvern - MODELS

D.1 Flytskjema

Figur D.1: Flytskjema - Differensialvern - INIT



Figur D.2: Flytskjema - Differensialvern - EXEC

D.2 Kjeldekode

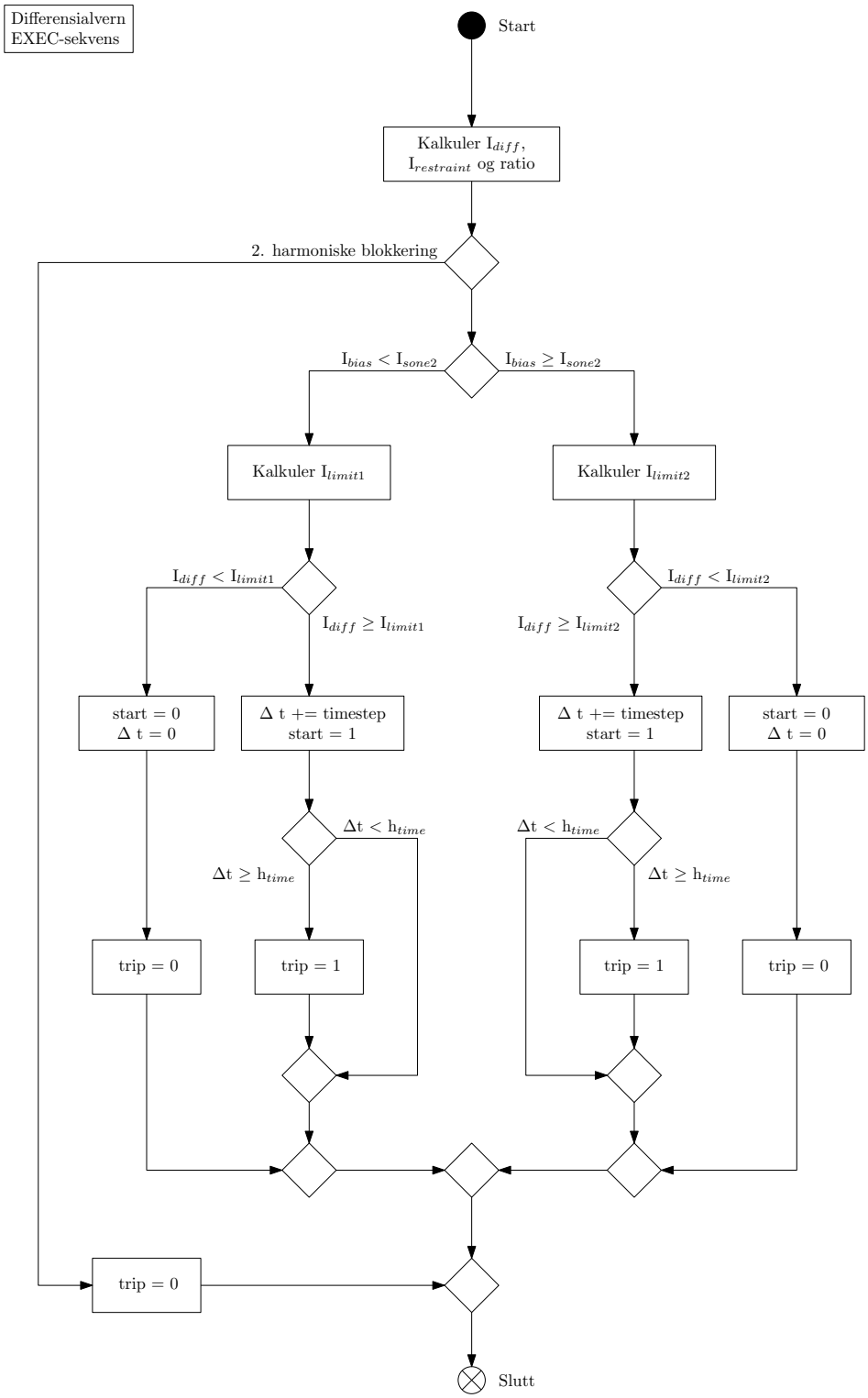
```
MODEL diff
COMMENT
Differential current relay,
by Torstein Stadheim, 04.06.2012

Source:
[1]Current Differential Line Protection Setting ↙
   ↳ Considerations,
by S. Ward and T. Erwin, RFL Electronics Inc, 2005.

The current I1 and I2 is defined as current entering  the ↙
   ↳ protected area or componenent.
The calculations is based on equation 3,4 in[1].

The source data(current) should be in rectangular form.

The restraint area is divided in two parts to aim at a ↙
   ↳ combination of a percentage restraint and variabel ↙
   ↳ restraint.
ENDCOMMENT

DATA    K1          -- Restraint area constant, slope of ↙
   ↳ first restraint area
        K2          -- Restraint area constant, slope of ↙
           ↳ second restraint area
        i_start     -- Restraint area constant, ↙
           ↳ differential current start of first area
        i_step2     -- Restraint area constant, restraint ↙
           ↳ current start of second area
        t_delay     -- Trip delay[s]
        k           -- Restraint current scalar

INPUT   i1re_1      -- First harmonic real part
        i1im_1      -- First harmonic imaginary part
        i2re_1      -- First harmonic real part
        i2im_1      -- First harmonic imaginary part

OUTPUT  trip

VAR     start
        trip
        i_diff_1        -- First harmonic differential ↙
           ↳ current
```

```
        i_restraint     -- Restraint current(first harmonic↙
           ↳ )
        deltaT          -- Elapsed time from fault ↙
           ↳ detection
        i_limit1        -- For restraint calculation
        i_limit2        -- For restraint calculation

INIT
    trip:=0
    start:=0
    deltaT:=0
ENDINIT
EXEC
    -- Calculates differential and restraint currents
    i_diff_1:=sqrt((i1re_1+i2re_1)**2+(i1im_1+i2im_1)**2)
    i_restraint:=k*(sqrt(i1re_1**2+i1im_1**2) + sqrt(i2re_1↙
       ↳ **2+i2im_1**2))

    -- REGION 1
    IF i_restraint < i_step2 THEN
        i_limit1 := i_start + K1 * i_restraint
        IF i_diff_1 > i_limit1 THEN
            deltaT := deltaT + timestep
            start := 1
            IF deltaT > t_delay THEN
                trip := 1
            ENDIF
        -- RESET
        ELSE
            start := 0
            deltaT := 0
            trip := 0
        ENDIF
    -- REGION 2
    ELSE
        i_limit2 := i_start + K2 * i_restraint - ( K2 - K1 ↙
           ↳ ) * i_step2
        IF i_diff_1 > i_limit2 THEN
            deltaT := deltaT + timestep
            start := 1
            IF deltaT > t_delay THEN
                trip := 1
            ENDIF
        -- RESET
        ELSE
            start := 0
```

```
            deltaT := 0
            trip := 0
        ENDIF
    ENDIF



ENDEXEC
ENDMODEL
```

D.3 Transformatorvern

D.3.1 Flytskjema



Figur D.3: Flytskjema - Differensialvern med 2. harmoniske restriksjon - EXEC

D.3.2 Kjeldekode

```
MODEL diff_T
COMMENT
Differential current relay, with second harmonic restraint,
by Torstein Stadheim, 04.06.2012

Source:
[1]Current Differential Line Protection Setting ↙
   ↳ Considerations,
by S. Ward and T. Erwin, RFL Electronics Inc, 2005.
[2]Practical Experience in Setting Transformer Differential↙
   ↳  Inrush Restraint,
by R. Hunt, J. Scaefer and B. Bentert, 2007
[3]A New Approach to Current Differential Protection for ↙
   ↳ Transmission Lines,
by M.G. Adamiak, G.E. Alexander, Dr. W. Premerlani, GE ↙
   ↳ digital energy

The current I1 and I2 is defined as current entering  the ↙
   ↳ protected area or componenent.
The calculations is based on equation 3,4 in[1]. The second↙
   ↳  harmonic option was found in [2].

The source data(current) should be in rectangular form.

The restraint area is divided in two parts to aim at a ↙
   ↳ combination of a percentage restraint and variabel ↙
   ↳ restraint.
ENDCOMMENT

DATA    K1          -- Restraint area constant, slope of ↙
   ↳ first restraint area
        K2          -- Restraint area constant, slope of ↙
           ↳ second restraint area
        i_start     -- Restraint area constant, ↙
           ↳ differential current start of first area
        i_step2     -- Restraint area constant, restraint ↙
           ↳ current start of second area
        t_delay     -- Trip delay[s]
        k           -- Restraint current scalar
        h2_res {dflt:20}    -- Second harmonic restraint, ↙
           ↳ in percent of first harmonic magnitude [%]

INPUT   i1re_1      -- First harmonic real part
        i1im_1      -- First harmonic imaginary part
```

```
        i2re_1      -- First harmonic real part
        i2im_1      -- First harmonic imaginary part
        i1re_2      -- Second harmonic real part
        i1im_2      -- Second harmonic imaginary part
        i2re_2      -- Second harmonic real part
        i2im_2      -- Second harmonic imaginary part


OUTPUT  trip

VAR     start
        trip
        i_diff_1        -- First harmonic differential ↙
           ↳ current
        i_diff_2        -- Second harmonic differential ↙
           ↳ current
        i_restraint     -- Restraint current(first harmonic↙
           ↳ )
        deltaT          -- Elapsed time from fault ↙
           ↳ detection
        i_limit1        -- For restraint calculation
        i_limit2        -- For restraint calculation
        ratio           -- Ratio of second harmonic to ↙
           ↳ first harmonic magnitude

INIT
    trip:=0
    start:=0
    deltaT:=0
ENDINIT
EXEC
    -- Calculates differential and restraint currents
    i_diff_1:=sqrt((i1re_1+i2re_1)**2+(i1im_1+i2im_1)**2)
    i_diff_2:=sqrt((i1re_2+i2re_2)**2+(i1im_2+i2im_2)**2)
    i_restraint:=k*(sqrt(i1re_1**2+i1im_1**2) + sqrt(i2re_1↙
       ↳ **2+i2im_1**2))

    ratio := 100 * i_diff_2 * recip(i_diff_1)

IF ratio < h2_res THEN  -- second harmonic blocking
    -- REGION 1
    IF i_restraint < i_step2 THEN
        i_limit1 := i_start + K1 * i_restraint
        IF i_diff_1 > i_limit1 THEN
            deltaT := deltaT + timestep
            start := 1
```

```
            IF deltaT > t_delay THEN
                trip := 1
            ENDIF
        -- RESET
        ELSE
            start := 0
            deltaT := 0
            trip := 0
        ENDIF
    -- REGION 2
    ELSE
        i_limit2 := i_start + K2 * i_restraint - ( K2 - K1 ↙
           ↳ ) * i_step2
        IF i_diff_1 > i_limit2 THEN
            deltaT := deltaT + timestep
            start := 1
            IF deltaT > t_delay THEN
                trip := 1
            ENDIF
        -- RESET
        ELSE
            start := 0
            deltaT := 0
            trip := 0
        ENDIF
    ENDIF
ELSE
    trip := 0
ENDIF



ENDEXEC
ENDMODEL
```

Tillegg E

Impedansvern - MODELS

E.1 R-X fase-fase kalkulator - kjeldekode

```
MODEL R_X_fj
COMMENT
Phase-earth R-X calculator,
by Torstein Stadheim, 06.06.2012

Calculates measured impedance between phase and earth.
ENDCOMMENT


DATA    k{dflt:1.2} -- Ratio (Z0-Z1)/Z1 of protected line. ↙
   ↳ Z0 is zero sequence impedance, and Z1 is positive ↙
   ↳ sequence impedance seen from the relay to the end of ↙
   ↳ the protection zone.

INPUT   v1re    -- Real part of phase voltage (rms - values↙
   ↳ )
        v1im    -- Imaginary part of phase voltage (rms - ↙
           ↳ values)
        i1re    -- Real part of phase current (rms - values↙
           ↳ )
        i1im    -- Imaginary part of phase current (rms - ↙
           ↳ values)
        i0re    -- Real part of earth(zero sequence) ↙
           ↳ current (rms - values)
        i0im    -- Imaginary part of earth(zero sequence) ↙
           ↳ current (rms - values)

OUTPUT  R   -- Measured resistance between phase A and B [↙
   ↳ Ohm]
        X   -- Measured reactance between phase A and B [↙
           ↳ Ohm]
```

```
VAR RecVsum[1..2]
    RecIsum[1..2]
    denominator
    numeratorRe
    numeratorIm
    R
    X

INIT
ENDINIT
EXEC
------------------Subtracts voltage and current↙
   ↳ ---------------------
RecVsum[1..2] := [v1re , v1im]
RecIsum[1..2] := [i1re + (i0re * k) , i1im + (i0im * k)]

------------------Divides↙
   ↳ ----------------------------------------
denominator := RecIsum[1]**2 + RecIsum[2]**2
numeratorRe := RecVsum[1] * RecIsum[1] + RecVsum[2] * ↙
   ↳ RecIsum[2]
numeratorIm := RecVsum[2] * RecIsum[1] - RecVsum[1] * ↙
   ↳ RecIsum[2]

-- Calculates the ouput
R := numeratorRe * recip(denominator)
X := numeratorIm * recip(denominator)


ENDEXEC
ENDMODEL
```

E.2 R-X fase-jord kalkulator - kjeldekode

```
MODEL R_X_ff
COMMENT
Phase-earth R-X calculator,
by Torstein Stadheim, 06.06.2012

Calculates measured impedance between phase and earth.
ENDCOMMENT


DATA    k{dflt:1.2} -- Ratio (Z0-Z1)/Z1 of protected line. ↙
   ↳ Z0 is zero sequence impedance, and Z1 is positive ↙
```

```
   ↳ sequence impedance seen from the relay to the end of ↙
   ↳ the protection zone.

INPUT    v1re     -- Real part of phase voltage (rms - values↙
   ↳ )
         v1im     -- Imaginary part of phase voltage (rms - ↙
            ↳ values)
         i1re     -- Real part of phase current (rms - values↙
            ↳ )
         i1im     -- Imaginary part of phase current (rms - ↙
            ↳ values)
         i0re     -- Real part of earth(zero sequence) ↙
            ↳ current (rms - values)
         i0im     -- Imaginary part of earth(zero sequence) ↙
            ↳ current (rms - values)

OUTPUT   R    -- Measured resistance between phase A and B [↙
   ↳ Ohm]
         X    -- Measured reactance between phase A and B [↙
            ↳ Ohm]

VAR RecVsum[1..2]
    RecIsum[1..2]
    denominator
    numeratorRe
    numeratorIm
    R
    X

INIT
ENDINIT
EXEC
------------------Subtracts voltage and current↙
   ↳ ---------------------
RecVsum[1..2] := [v1re , v1im]
RecIsum[1..2] := [i1re + (i0re * k) , i1im + (i0im * k)]

------------------Divides↙
   ↳ ----------------------------------------
denominator := RecIsum[1]**2 + RecIsum[2]**2
-- Avoids dividing with zero
IF denominator = 0 THEN
    denominator := 1E-15
ENDIF
numeratorRe := RecVsum[1] * RecIsum[1] + RecVsum[2] * ↙
   ↳ RecIsum[2]
```

```
numeratorIm := RecVsum[2] * RecIsum[1] - RecVsum[1] * ↙
   ↳ RecIsum[2]

-- Calculates the ouput
R := numeratorRe / denominator
X := numeratorIm / denominator


ENDEXEC
ENDMODEL
```

E.3 Sirkelkarakteristikk

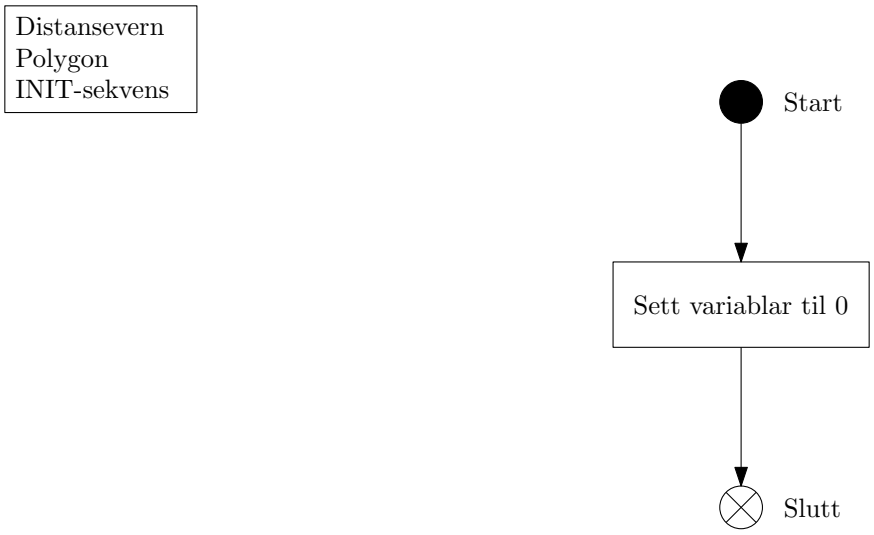
E.3.1 Kjeldekode

```
MODEL d_circle
COMMENT
Impedance element relay, 11.06.2012,
by Torstein Stadheim

Impedance(distance) relay with circle characteristic.

To be used together with R-X calculator model.

ENDCOMMENT

FUNCTION    Mag( ReX , ImX )    := sqrt( ReX * ReX + ImX * ↙
   ↳ ImX )

DATA    radius { dflt:1 }   -- radius of trip circle
        r_centre { dflt:0 } -- center of circle, r-axis
        x_centre { dflt:0 } -- center of circle, x-axis
        t_delay { dflt:0.02 }   -- delay of trip

INPUT   R   -- [Ohm]
        X   -- [Ohm]

OUTPUT  trip    -- 1 if trip
        start   -- 1 if pickup

VAR     trip
        start
        r_tran  -- transformed z-vector
        x_tran  -- transformed z-vector
        Z_length    -- length of z-vector
        delta_T -- timer-variable
```

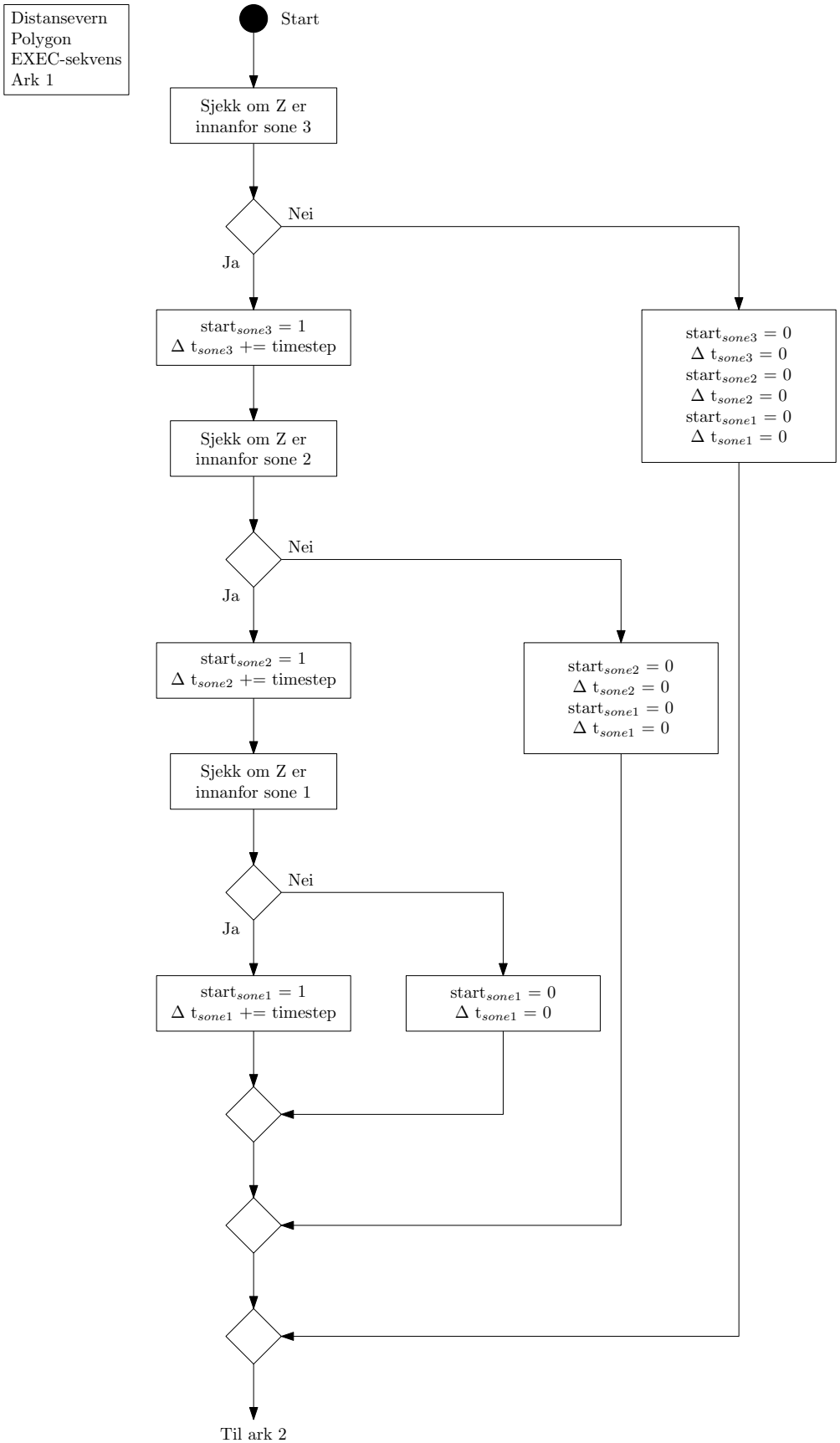
```
INIT
    trip:=0
    delta_T:=0
    start:=0
ENDINIT

EXEC

    -- Calculates difference between measured impedance and↙
       ↳  center of circle.
    r_tran := R - r_centre
    x_tran := X - x_centre

    -- Calculates the length of the difference
    Z_length := Mag(r_tran,x_tran)

    IF Z_length <= radius THEN
        start:=1
        delta_T := delta_T + timestep
        IF delta_T > t_delay THEN
            trip:=1
        ENDIF
    ELSE
        delta_T := 0
        start:=0
    ENDIF

ENDEXEC
ENDMODEL
```

E.4 Polygonkarakteristikk

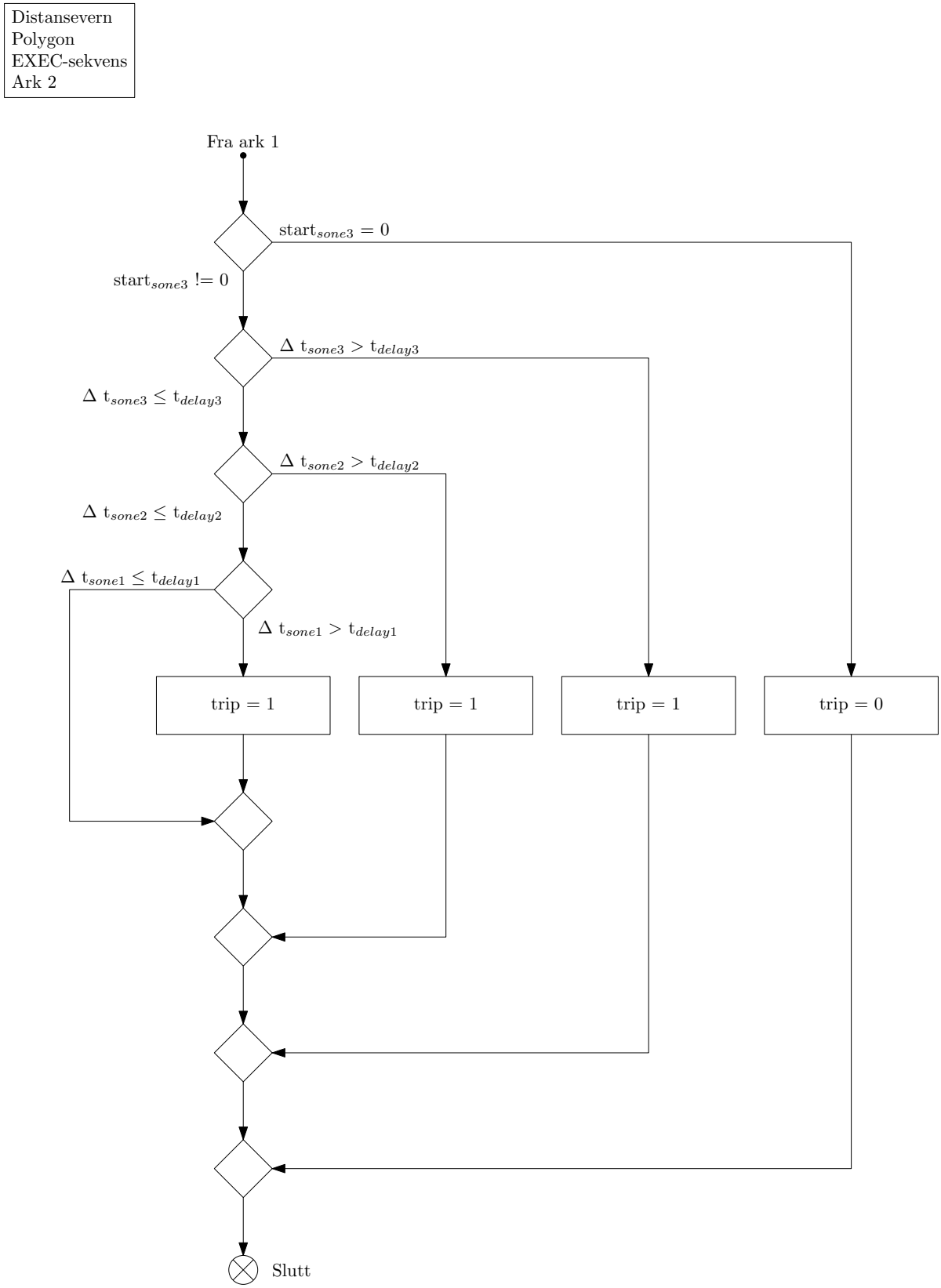
E.4.1 Flytskjema



Figur E.1: Flytskjema - impedansvern - polygonkarakteristikk - INIT



Figur E.2: Flytskjema - impedansvern - polygonkarakteristikk - EXEC - ark 1



Figur E.3: Flytskjema - impedansvern - polygonkarakteristikk - EXEC - ark 2

E.4.2 Kjeldekode

```
MODEL d_poly

COMMENT
Impedance element relay, 11.06.2012,
by Torstein Stadheim

Impedance(distance) relay with polygon characteristic. To ↙
   ↳ be used together with R-X calculator model.
The model are using the principle of convex polygons to ↙
   ↳ check IF a point is inside a polygon. For this model ↙
   ↳ to work, the points should be defined counter-↙
   ↳ clockwise, and no vector from point n to point n+1 ↙
   ↳ should cross another point(m)-to-point(m+1) vector.

The polygons should be defined in the following manner:
                          |X
                    3-----|------2
                    | 3---|---2  | etc...
                    | |   |   |  |
                    | |   |   |  |
----------------------------------------------------> R
                    | |   |   |  |
                    | 4---|---1  | ZONE 1 INNERMOST
                    |     |      |
                    4-----|------1
                          |
Sources:    http://erich.realtimerendering.com/ptinpoly/

ENDCOMMENT

-- positive IF a point(x,y) is on the left side of vector [↙
   ↳ x2-x1,y2-y1], starting at (x1,y1)
FUNCTION check(x,y,x1,y1,x2,y2) := ((x2-x1)*(y-y1))-((x-x1)↙
   ↳ *(y2-y1))

DATA    t_zone1{dflt:0.2}   -- tripdelay zone 1 [s]
        t_zone2{dflt:0.5}   -- tripdelay zone 2 [s]
        t_zone3{dflt:1}     -- tripdelay zone 3 [s]
        -- zone 1 characterstic parameters
        R1_s1{dflt:1}   -- R coordinate point.1 zone 1
        X1_s1{dflt:-1}  -- X coordinate point.1 zone 1
        R2_s1{dflt:1}   -- R coordinate point.2 zone 1
        X2_s1{dflt:1}   -- X coordinate point.2 zone 1
        R3_s1{dflt:-1}  -- R coordinate point.3 zone 1
```

```
        X3_s1{dflt:1}   -- X coordinate point.3 zone 1
        R4_s1{dflt:-1}  -- R coordinate point.4 zone 1
        X4_s1{dflt:-1}  -- X coordinate point.4 zone 1
        --zone 2 characterstic parameters
        R1_s2{dflt:2}   -- R coordinate point.1 zone 2
        X1_s2{dflt:-2}  -- X coordinate point.1 zone 2
        R2_s2{dflt:2}   -- R coordinate point.2 zone 2
        X2_s2{dflt:2}   -- X coordinate point.2 zone 2
        R3_s2{dflt:-2}  -- R coordinate point.3 zone 2
        X3_s2{dflt:2}   -- X coordinate point.3 zone 2
        R4_s2{dflt:-2}  -- R coordinate point.4 zone 2
        X4_s2{dflt:-2}  -- X coordinate point.4 zone 2
        --zone 3 characterstic parameters
        R1_s3{dflt:3}   -- R coordinate point.1 zone 3
        X1_s3{dflt:-3}  -- X coordinate point.1 zone 3
        R2_s3{dflt:3}   -- R coordinate point.2 zone 3
        X2_s3{dflt:3}   -- X coordinate point.2 zone 3
        R3_s3{dflt:-3}  -- R coordinate point.3 zone 3
        X3_s3{dflt:3}   -- X coordinate point.3 zone 3
        R4_s3{dflt:-3}  -- R coordinate point.4 zone 3
        X4_s3{dflt:-3}  -- X coordinate point.4 zone 3

INPUT   R   -- [Ohm]
        X   -- [Ohm]


OUTPUT  trip        -- 1 IF trip
        startS1     -- 1 IF impedance is inside zone 1
        startS2     -- 1 IF impedance is inside zone 2
        startS3     -- 1 IF impedance is inside zone 3

VAR trip
    startS1
    startS2
    startS3
    delta_T1    -- timer zone 1
    delta_T2    -- timer zone 2
    delta_T3    -- timer zone 3

INIT
    trip:=0
    delta_T1:=0
    delta_T2:=0
    delta_T3:=0
    startS1:=0
    startS2:=0
```

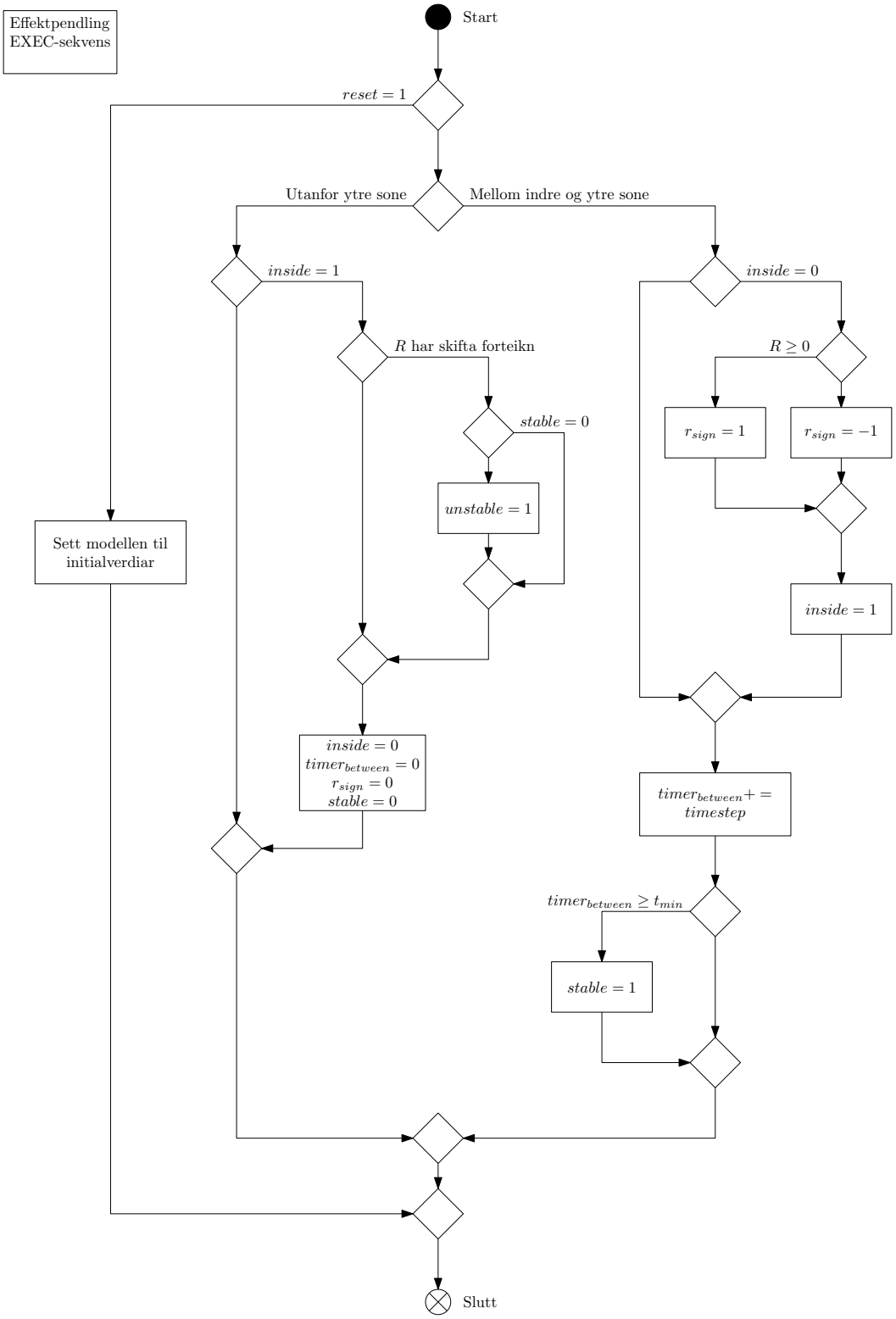
```
    startS3:=0
ENDINIT
EXEC

-------------------------check IF inside zone 3-----------
IF check(R,X,R1_s3,X1_s3,R2_s3,X2_s3) > 0 THEN
    IF check(R,X,R2_s3,X2_s3,R3_s3,X3_s3) > 0 THEN
        IF check(R,X,R3_s3,X3_s3,R4_s3,X4_s3) > 0 THEN
            IF check(R,X,R4_s3,X4_s3,R1_s3,X1_s3) > 0 THEN
                startS3 := 1 -- inside zone 3
                delta_T3 := delta_T3 + timestep
            ELSE -- outside zone 3
                startS3 := 0
                delta_T3 := 0
                startS2 := 0
                delta_T2 := 0
                startS1 := 0
                delta_T1 := 0
            ENDIF
        ELSE
            startS3 := 0
            delta_T3 := 0
            startS2 := 0
            delta_T2 := 0
            startS1 := 0
            delta_T1 := 0
        ENDIF
    ELSE
        startS3 := 0
        delta_T3 := 0
        startS2 := 0
        delta_T2 := 0
        startS1 := 0
        delta_T1 := 0
    ENDIF
ELSE
    startS3 := 0
    delta_T3 := 0
    startS2 := 0
    delta_T2 := 0
    startS1 := 0
    delta_T1 := 0
ENDIF
-------------------------check IF inside zone ↙
   ↳ 2-----------------
-- does not check zone 2 IF point is outside zone 3
```

```
IF check(R,X,R1_s2,X1_s2,R2_s2,X2_s2) > 0 and (startS3 = 1)↙
   ↳  THEN
    IF check(R,X,R2_s2,X2_s2,R3_s2,X3_s2) > 0 THEN
        IF check(R,X,R3_s2,X3_s2,R4_s2,X4_s2) > 0 THEN
            IF check(R,X,R4_s2,X4_s2,R1_s2,X1_s2) > 0 THEN
                startS2 := 1 -- inside zone 2
                delta_T2 := delta_T2 + timestep
            ELSE -- outside zone 2
                startS2 := 0
                delta_T2 := 0
                startS1 := 0
                delta_T1 := 0
            ENDIF
        ELSE
            startS2 := 0
            delta_T2 := 0
            startS1 := 0
            delta_T1 := 0
        ENDIF
    ELSE
        startS2 := 0
        delta_T2 := 0
        startS1 := 0
        delta_T1 := 0
    ENDIF
ELSE
    startS2 := 0
    delta_T2 := 0
    startS1 := 0
    delta_T1 := 0
ENDIF
-------------------------check IF inside zone ↙
   ↳ 1-----------------
-- does not check zone 1 IF point is outside zone 2
IF check(R,X,R1_s1,X1_s1,R2_s1,X2_s1) > 0 and (startS2 = 1)↙
   ↳  THEN
    IF check(R,X,R2_s1,X2_s1,R3_s1,X3_s1) > 0 THEN
        IF check(R,X,R3_s1,X3_s1,R4_s1,X4_s1) > 0 THEN
            IF check(R,X,R4_s1,X4_s1,R1_s1,X1_s1) > 0 THEN
                startS1 := 1 -- inside zone 1
                delta_T1 := delta_T1 + timestep
            ELSE -- outside zone 1
                startS1 := 0
                delta_T1 := 0
            ENDIF
        ELSE
```

```
            startS1 := 0
            delta_T1 := 0
        ENDIF
    ELSE
        startS1 := 0
        delta_T1 := 0
    ENDIF
ELSE
    startS1 := 0
    delta_T1 := 0
ENDIF

------------Check delay, and trip of timer runs out↙
   ↳ -----------------------------
IF startS3 = 0 THEN -- reset function
    trip := 0
ELSIF   delta_T3 > t_zone3 THEN -- trip zone 3
    trip := 1
ELSIF delta_T2 > t_zone2 THEN -- trip zone 2
    trip := 1
ELSIF delta_T1 > t_zone1 THEN -- trip zone 1
    trip := 1
ENDIF

ENDEXEC
ENDMODEL
```

E.5 Effektpendling

E.5.1 Flytskjema



Figur E.4: Flytskjema - effektpendling - EXEC

E.5.2 Kjeldekode - sirkelkarakteristikk

```
MODEL oos_c

COMMENT
Out of Step relay with circle characteristic
by Torstein Stadheim, 14.06.2012
ENDCOMMENT

FUNCTION    Mag( ReX , ImX )    := sqrt( ReX * ReX + ImX * ↙
   ↳ ImX )

DATA    r_centre    -- R coordinat of the circles center
        x_centre    -- X coordinat of the circles center
        r_inner     -- Radius of the inner circle
        r_outer     -- Radius of the outer circle
        t_min       -- Minimum time impedance must be ↙
           ↳ between the inner and outer circle for a ↙
           ↳ stable powerswing



INPUT   R
        X
        reset   -- Manually resets the relay to initial ↙
           ↳ values. 1 activates, 0 deactivates

OUTPUT  stable      -- 1 if a stable powerswing is detected
        unstable    -- 1 if an unstable powerswing is ↙
           ↳ detected (Out of step)

VAR
    timer_between   -- Timer that measures the time the ↙
       ↳ impedance is between the inner and outer zone
    inside  -- Variable that controls if the impedance ↙
       ↳ enters, or exits the zone. 0 if the impedance is ↙
       ↳ outside the outer zone, 1 if it is between, and 2 ↙
       ↳ if it is inside the inner zone
    r_sign -- variable that gives the sign of the r-↙
       ↳ variable when entering the outer zone. 1 if ↙
       ↳ positive, and -1 if negative. 0 is initial value.
    stable
    unstable
    r_tran  -- Used in vector subtraction, z-vector
    x_tran  -- Used in vector subtraction, z-vector
    z_length -- Length of z-vector
```

```
INIT
    stable := 0
    unstable := 0
    inside := 0
    r_sign := 0
    timer_between := 0
ENDINIT

EXEC

IF reset = 0 THEN
    -- Calculates difference between measured impedance and↙
       ↳  center of circle.
    r_tran := R - r_centre
    x_tran := X - x_centre

    -- Calculates the length of the difference
    z_length := Mag(r_tran,x_tran)

    -- Outside outer circle
    IF z_length > r_outer THEN
        -- If the impedance exits from the outer circle
        IF inside = 1 THEN
            -- Check if unstable powerswing is happening
            IF r_sign = 1 AND R < 0 AND stable = 1 THEN -- ↙
               ↳ positive upon entering, negative upon ↙
               ↳ exiting
                unstable := 1
            ELSIF r_sign = -1 AND R >= 0 AND stable = 1 ↙
               ↳ THEN  -- negative upon entering, positive ↙
               ↳ upon exiting
                unstable := 1
            ENDIF
            inside := 0 -- impedance vector is now outside ↙
               ↳ the outer circle
            timer_between := 0 -- Resets the timer
            r_sign := 0
            stable := 0
        ENDIF

    -- Between inner and outer circle
    ELSIF z_length <= r_outer AND z_length > r_inner THEN
        -- If the impedance enters the powerswing zone from↙
           ↳  outside the outer circle:
        IF inside = 0 THEN
```

```
            IF R >= 0 THEN
                r_sign := 1 -- Positive R upon entering
            ELSE
                r_sign := -1    -- Negative R upon entering
            ENDIF
            inside := 1 -- The impedance has now entered ↙
               ↳ the powerswing zone
        ENDIF
        -- Updates the timer
        timer_between := timer_between + timestep

        -- Checks the timer
        if timer_between >= t_min THEN
            stable := 1
        ENDIF

    ENDIF

-- External reset
ELSE
    stable := 0
    unstable := 0
    timer_between := 0
    r_sign := 0
ENDIF

ENDEXEC
ENDMODEL
```

E.5.3 Kjeldekode - blenderkarakteristikk

```
MODEL oos_b

COMMENT
Out of Step relay with blinder characteristic
by Torstein Stadheim, 14.06.2012

The function "check" desides if a poing(x,y) is left/right ↙
   ↳ of a line [x2-x1,y2-y1] going through the point (x1,y1↙
   ↳ ). If the function is positive, it is left of the line↙
   ↳ . If it is zero, it is on the line.

Blinder characteristic:
                    ^X
    |   |           |       |   |
    |   |           |       |   |
    |   |           |       |   |
```

```
     |   |            |        |   |
     |   |            |        |   |
---r_1-r_2-----------------r_3-r_4--------->R
     |   |            |        |   |
     |   |            |        |   |
     |   |            |        |   |
     |   |            |        |   |
     |   |            |        |   |
     |   |            |        |   |


Angle definition:
     ^X
     |            x    ->    (r2_n , x_angle)
     |           /
     |          /) angle
------------r_n------------>R
     |        /
     |       /
     |      /
ENDCOMMENT

-- positive if a point(x,y) is on the left side of vector [↙
   ↳ x2-x1,y2-y1]
FUNCTION check(x,y,x1,y1,x2,y2) := ((x2-x1)*(y-y1))-((x-x1)↙
   ↳ *(y2-y1))

DATA     r_1 {dflt:-10}  -- R coordinat of left outer ↙
   ↳ blinder
         r_2 {dflt:-5}   -- R coordinat of left inner ↙
            ↳ blinder
         r_3 {dflt:5}    -- R coordinat of right inner ↙
            ↳ blinder
         r_4 {dflt:10}   -- R coordinat of right outer ↙
            ↳ blinder
         angle {dflt:90} -- Angle in degrees of all blinders↙
            ↳ . 90 degrees is parallell with X axis
         t_min {dflt:0.5}    -- Minimum time impedance must ↙
            ↳ be between the inner and outer circle for a ↙
            ↳ stable powerswing



INPUT    R
         X
```

```
        reset   -- Manually resets the relay to initial ↙
          ↳ values. 1 activates, 0 deactivates

OUTPUT  stable      -- 1 if a stable powerswing is detected
        unstable    -- 1 if an unstable powerswing is ↙
          ↳ detected (Out of step)

VAR
    timer_between   -- Timer that measures the time the ↙
      ↳ impedance is between the inner and outer zone
    inside  -- Variable that controls if the impedance ↙
      ↳ enters, or exits the zone. 0 if the impedance is ↙
      ↳ outside the outer zone, 1 if it is between, and 2 ↙
      ↳ if it is inside the inner zone
    r_sign -- variable that gives the sign of the r-↙
      ↳ variable when entering the outer zone. 1 if ↙
      ↳ positive, and -1 if negative. 0 is initial value.
    stable
    unstable
    outside -- Used for controlling the main IF-structure. ↙
      ↳ 1 if the imepdance is outside both outer blinders,↙
      ↳  0 if the impedance is in the powerswingzone.
    r_angle
    x_angle
    r2_1    -- r-coordinat of the second point in the ↙
      ↳ associated line
    r2_2    -- = --
    r2_3    -- = --
    r2_4    -- = --


INIT

    stable := 0
    unstable := 0
    inside := 0
    r_sign := 0
    timer_between := 0
    outside := 1
    r_angle := cos(angle*PI/180)
    x_angle := sin(angle*PI/180)
    r2_1 := r_1 + r_angle
    r2_2 := r_2 + r_angle
    r2_3 := r_3 + r_angle
    r2_4 := r_4 + r_angle
```

```
ENDINIT

EXEC

IF reset = 0 THEN

    -- Check if impedance is left of outer right blinder ↙
       ↳ AND right of outer left blinder:
    IF check(R , X , r_4 , 0 , r2_4 , x_angle) >= 0 AND ↙
       ↳ check(R , X , r_1 , 0 , r2_1 , x_angle) < 0 THEN
        -- Check if impedance is right of inner right ↙
           ↳ blinder OR left of inner left blinder
        IF check(R , X , r_3 , 0 , r2_3 , x_angle) < 0 OR ↙
           ↳ check(R , X , r_2 , 0 , r2_2 , x_angle) >= 0 ↙
           ↳ THEN
            outside := 0
        ELSE
            -- Inside inner blinders
            outside := -1
        ENDIF
    ELSE
        -- Outside outer blinders
        outside := 1
    ENDIF

    -- Outside outer blinders
    IF outside = 1 THEN
        -- If the impedance exits from the outer circle
        IF inside = 1 THEN
            -- Check if unstable powerswing is happening
            IF r_sign = 1 AND R < 0 AND stable = 1 THEN -- ↙
               ↳ positive upon entering, negative upon ↙
               ↳ exiting
                unstable := 1
            ELSIF r_sign = -1 AND R >= 0 AND stable = 1 ↙
               ↳ THEN  -- negative upon entering, positive ↙
               ↳ upon exiting
                unstable := 1
            ENDIF
            inside := 0 -- impedance vector is now outside ↙
               ↳ the outer circle
            timer_between := 0 -- Resets the timer
            r_sign := 0
            stable := 0
        ENDIF
```

```
    -- Between inner and outer blinders
    ELSIF outside = 0 THEN
        -- If the impedance enters the powerswing zone from↙
           ↳  outside the outer circle:
        IF inside = 0 THEN
            IF R >= 0 THEN
                r_sign := 1 -- Positive R upon entering
            ELSE
                r_sign := -1    -- Negative R upon entering
            ENDIF
            inside := 1 -- The impedance has now entered ↙
               ↳ the powerswing zone
        ENDIF
        -- Updates the timer
        timer_between := timer_between + timestep

        -- Checks the timer
        IF timer_between >= t_min THEN
            stable := 1
        ENDIF

    ENDIF

-- External reset
ELSE
    stable := 0
    unstable := 0
    timer_between := 0
    r_sign := 0
    outside := 1
ENDIF

ENDEXEC
ENDMODEL
```

E.5.4 Kjeldekode - polygonkarakteristikk

```
MODEL oos_p

COMMENT
Out of Step relay with polygon characteristic
by Torstein Stadheim, 17.06.2012

The function "check" desides if a poing(x,y) is left/right ↙
   ↳ of a line [x2-x1,y2-y1] going through the point (x1,y1↙
   ↳ ). If the function is positive, it is left of the line↙
   ↳ . If it is zero, it is on the line.
```

```
The model are using the principle of convex polygons to ↙
   ↳ check if a point is inside a polygon. For this model ↙
   ↳ to work, the points should be defined counter-↙
   ↳ clockwise, and no vector from point n to point n+1 ↙
   ↳ should cross another point(m)-to-point(m+1) vector.

The polygons should be defined in the following manner:
                          |X
                   3------|------2
                   | 3----|----2  | etc...
                   | |    |    |  |
                   | |    |    |  |
-------------------------------------------------------> R
                   | |    |    |  |
                   | 4----|----1  | ZONE 5 INNERMOST
                   |      |       |
                   4------|------1
                          |
Sources:    http://erich.realtimerendering.com/ptinpoly/



ENDCOMMENT

-- positive if a point(x,y) is on the left side of vector [↙
   ↳ x2-x1,y2-y1]
FUNCTION check(x,y,x1,y1,x2,y2) := ((x2-x1)*(y-y1))-((x-x1)↙
   ↳ *(y2-y1))

DATA    t_min{dflt:0.2} -- Minimum time impedance must be ↙
   ↳ between the inner and outer circle for a stable ↙
   ↳ powerswing
        -- Inner zone characterstic parameters
        R1_s5{dflt:1}   -- R coordinate point.1 zone 5
        X1_s5{dflt:-1}  -- X coordinate point.1 zone 5
        R2_s5{dflt:1}   -- R coordinate point.2 zone 5
        X2_s5{dflt:1}   -- X coordinate point.2 zone 5
        R3_s5{dflt:-1}  -- R coordinate point.3 zone 5
        X3_s5{dflt:1}   -- X coordinate point.3 zone 5
        R4_s5{dflt:-1}  -- R coordinate point.4 zone 5
        X4_s5{dflt:-1}  -- X coordinate point.4 zone 5
        -- Outer zone characterstic parameters
        R1_s6{dflt:2}   -- R coordinate point.1 zone 6
        X1_s6{dflt:-2}  -- X coordinate point.1 zone 6
        R2_s6{dflt:2}   -- R coordinate point.2 zone 6
```

```
        X2_s6{dflt:2}    -- X coordinate point.2 zone 6
        R3_s6{dflt:-2}   -- R coordinate point.3 zone 6
        X3_s6{dflt:2}    -- X coordinate point.3 zone 6
        R4_s6{dflt:-2}   -- R coordinate point.4 zone 6
        X4_s6{dflt:-2}   -- X coordinate point.4 zone 6



INPUT   R
        X
        reset    -- Manually resets the relay to initial ↙
           ↳ values. 1 activates, 0 deactivates

OUTPUT  stable       -- 1 if a stable powerswing is detected
        unstable     -- 1 if an unstable powerswing is ↙
           ↳ detected (Out of step)

VAR
    timer_between    -- Timer that measures the time the ↙
       ↳ impedance is between the inner and outer zone
    inside  -- Variable that controls if the impedance ↙
       ↳ enters, or exits the zone. 0 if the impedance is ↙
       ↳ outside the outer zone, 1 if it is between, and 2 ↙
       ↳ if it is inside the inner zone
    outside -- Used for controlling the main IF-structure. ↙
       ↳ 1 if the imepdance is outside both outer blinders,↙
       ↳  0 if the impedance is in the powerswingzone.
    r_sign -- variable that gives the sign of the r-↙
       ↳ variable when entering the outer zone. 1 if ↙
       ↳ positive, and -1 if negative. 0 is initial value.
    stable
    unstable

INIT
    stable := 0
    unstable := 0
    inside := 0
    r_sign := 0
    timer_between := 0
ENDINIT

EXEC

IF reset = 0 THEN

    -- Check if impedance is inside outer polygon
```

```
IF check(R, X, R1_s6, X1_s6, R2_s6, X2_s6) >= 0 THEN
    IF check(R, X, R2_s6, X2_s6, R3_s6, X3_s6) >= 0 ↙
       ↳ THEN
        IF check(R, X, R3_s6, X3_s6, R4_s6, X4_s6) >= 0↙
           ↳  THEN
            IF check(R, X, R4_s6, X4_s6, R1_s6, X1_s6) ↙
               ↳ >= 0 THEN
                -- Inside outer polygon
                -- Check if impedance is outside inner ↙
                   ↳ polygon
                IF check(R, X, R1_s5, X1_s5, R2_s5, ↙
                   ↳ X2_s5) <= 0 THEN
                    -- Outside inner polygon
                    outside := 0
                ELSIF check(R, X, R2_s5, X2_s5, R3_s5, ↙
                   ↳ X3_s5) <= 0 THEN
                    -- Outside inner polygen
                    outside := 0
                ELSIF check(R, X, R3_s5, X3_s5, R4_s5, ↙
                   ↳ X4_s5) <= 0 THEN
                    -- Outside inner polygon
                    outside := 0
                ELSIF check(R, X, R4_s5, X4_s5, R1_s5, ↙
                   ↳ X1_s5) <= 0 THEN
                    -- Outside inner polygon
                    outside := 0
                ELSE
                    -- Inside inner polygon
                    outside := -1
                ENDIF
            ELSE
                outside := 1
            ENDIF
        ELSE
            outside := 1
        ENDIF
    ELSE
        outside := 1
    ENDIF
ELSE
    outside := 1
ENDIF

-- Outside outer polygon
IF outside = 1 THEN
    -- If the impedance exits from the outer circle
```

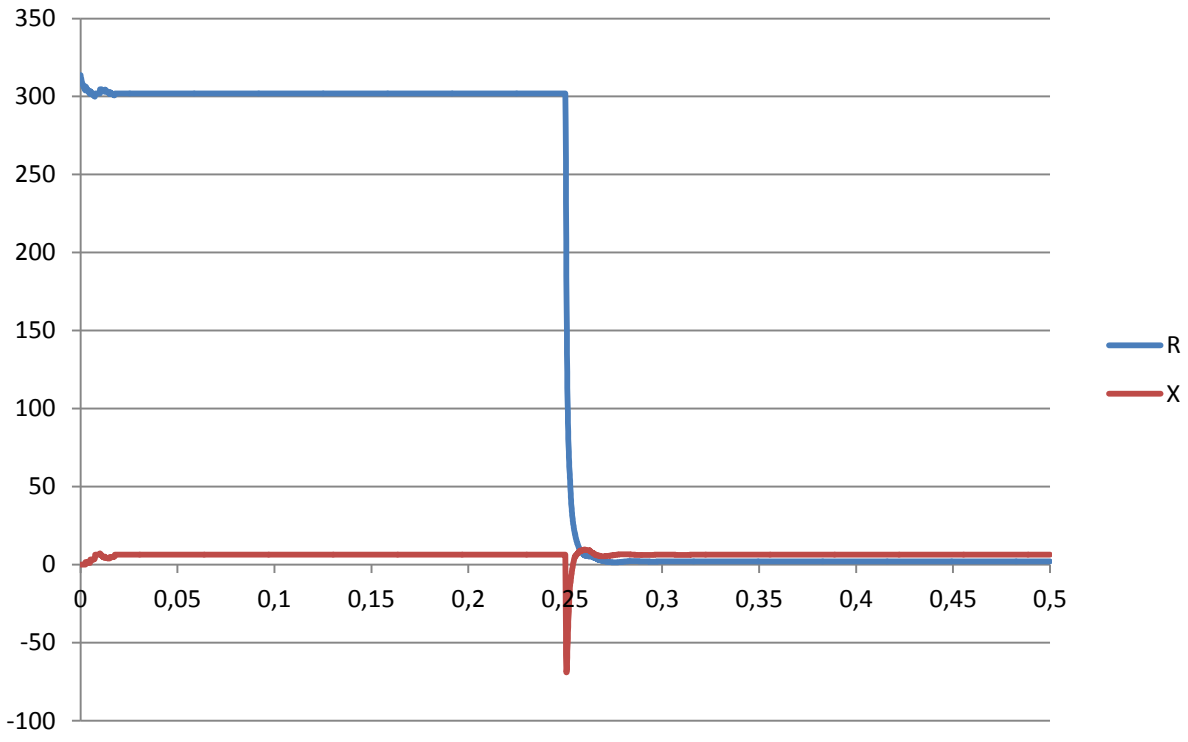
```
        IF inside = 1 THEN
            -- Check if unstable powerswing is happening
            IF r_sign = 1 AND R < 0 AND stable = 1 THEN -- ↙
               ↳ positive upon entering, negative upon ↙
               ↳ exiting
                unstable := 1
            ELSIF r_sign = -1 AND R >= 0 AND stable = 1 ↙
               ↳ THEN  -- negative upon entering, positive ↙
               ↳ upon exiting
                unstable := 1
            ENDIF
            inside := 0 -- impedance vector is now outside ↙
               ↳ the outer circle
            timer_between := 0 -- Resets the timer
            r_sign := 0
            stable := 0
        ENDIF

    -- Between inner and outer polygon
    ELSIF outside = 0 THEN
        -- If the impedance enters the powerswing zone from↙
           ↳  outside the outer circle:
        IF inside = 0 THEN
            IF R >= 0 THEN
                r_sign := 1 -- Positive R upon entering
            ELSE
                r_sign := -1    -- Negative R upon entering
            ENDIF
            inside := 1 -- The impedance has now entered ↙
               ↳ the powerswing zone
        ENDIF
        -- Updates the timer
        timer_between := timer_between + timestep

        -- Checks the timer
        IF timer_between >= t_min THEN
            stable := 1
        ENDIF

    ENDIF

-- External reset
ELSE
    stable := 0
    unstable := 0
    timer_between := 0
```

```
    r_sign := 0
    outside := 1
ENDIF

ENDEXEC
ENDMODEL
```

Tillegg F

Verifisering

F.1 Impedansmåling - simulering



Figur F.1: Verifiseringskrets - impedansmåling

Bibliografi

[1] *Protective Relaying Theory and Applications*. ABB, Second Edition, 2004.

[2] Richard G. Lyons *Understanding Digital Signal Processing*. Prentice Hall PTR., Second Edition, 2004.

[3] Gerhard Ziegler, *Numerical distance protection*. Siemens, Second Edition 2006.

[4] Horowitz, Phadke, *Power System Relaying*. Research Studies Press Ltd., 1992.

[5] Blackburn, Domin, *Protective Relaying, Principles and Applications*. Research Studies Press Ltd., Third Edition, 2007.

[6] The Electricity Council, *Power System Protection, Volume 1*. Macdonald & Co Ltd., 1969.

[7] Siemens manual *SIPROTEC 7SA6 V.461 and higher Distance Protection*. Siemens, 2005.

[8] Sognekraft AS, telefonkontakt *Harald Stadheim*. 10.11.2011.

[9] John J. Grainger, William D. Stevensjon, Jr., *Power System Analysis*. McGraw-Hill, Inc. 1994.

[10] Bill Fleming, *Negative-sequence impedance directional element*. Schweitzer Engineering Laboratories, Inc. 1998.

[11] M. M. Eissa *Ground Distance Relay Compensation Based on Fault Resistance Calculation*. IEEE, 4 October 2006.

[12] *IEEE Standard Inverse-Time Characteristic Equations for Overcurrent Relays*. IEEE, 19 September 1996.

[13] S. Ward, T. Erwin *Current Differential Line Protection Setting Considerations*. RFL Electronics Inc, 2005.

[14] M.G. Adamiak, G.E. Alexander, Dr. W. Premerlani *A New Approach to Current Differential Protection for Transmission Lines*. GE Digital Energy, 1998.

[15] R. Hunt, J. Scaefer, B. Bentert *Practical Experience in Setting Transformer Differential Inrush Restraint*. 2007.

[16] R.W. Patterson, W.P.McCannon, G.L.Kobet *A Consideration of Inrush Restraint Methods in Transformer Differential Relays*. 2000.

[17] B. Kasztenny, A.Kulidjian, B. Campbell, M. Pozzuoli *Operate and Restraint Signals of a Transformer Differential Relay*. 2000.

[18] Eric Haines *Point in polygon Strategies* article in *Graphics Gems IV*. ed. Paul Heckbert, Academic Press, 1994.

[19] U.N. Khan, L. Yan *Power Swing Phenomena and its Detection and Prevention*.

[20] J. Mooney, P.E, N. Fischer *Application Guidelines for Power Swing Detection on Transmission Systems*. IEEE, 2006.

[21] G. Benmouyal, D. Tziouvaras, and D. Hou *Zero-Setting Power-Swing Blocking Proctection*. Proceedings of the 31st Annual Western Protective Relay Conference, 2004.

[22] J. Mooney, P.E, N. Fischer *New Out-of-Step Blocking Algorithm for Detecting Fast Power Swing Frequencies*. Proceedings of the 30th Annual Western Protective Relay Conference 2003.

[23] D. A. Tziouvaras, D. Hou *Out-of-Step Protection Fundamentals and Advancements*. Schweitzer Engineering Laboratories, Inc. 2003.

[24] L. Prikler, H.Kr. Høidalen *ATPDRAW Users' Manual for version 5.6*. November 2009.

[25] Laurent Dubé *Users Guide to Models in ATP*. April 1996.