

Joni Näsinkoski

**TAAJUUSMUUTTAJAN PÄÄPIIRIN
SIMULAATIOMALLIN RAKENTAMINEN
MATLAB SIMULINK-OHJELMISTOON
CHATGPT:LLÄ**

Kandidaatintutkielma
Informaatioteknologian ja viestinnän tiedekunta
Tarkastaja: Associate Professor Paavo Rasilo
Helmikuu 2024

TIIVISTELMÄ

Joni Näsinkoski: Taajuusmuuttajan pääpiirin simulaatiomallin rakentaminen MATLAB Simulink-ohjelmistoon ChatGPT:llä
Kandidaatintutkielma
Tampereen yliopisto
Tieto- ja sähkötekniikan kandidaatintutkinto
Helmikuu 2024

Tekoäly on yksi teknologian nopeimmin kehittyvistä aihealueista tällä hetkellä ja sille etsitään uusia sovelluskohteita jatkuvasti. Tässä työssä tutkitaan suurelle yleisölle saatavilla olevan tekoälyn tämänhetkistä kyvykkyyttä rakentaa simulaatiomalleja. Työssä selvitetään, kuinka hyvin tekoäly suoriutuu simulaatiomallin rakentamisesta, ja dokumentoidaan prosessi, jolla simulaatiomallin voi rakentaa tekoälyä käyttäen.

Työssä käytettäväksi tekoälyksi valittiin ChatGPT:n versio 3.5. Systemiksi, jolle simulaatiomalli rakennetaan, valittiin yksinkertainen taajuusmuuttajan pääpiiri. Simulaatioalustana käytetään MATLAB Simulink-ohjelmistoa. Simulink-ohjelmiston käyttäminen mahdollistaa myös MatGPT-rajapintasovelluksen hyödyntämisen.

Työssä rakennetaan simulaatiomalli taajuusmuuttajan pääpiirille. Pääpiirin moduuleista tasasuuntaaja ja DC-välipiiri rakennettiin komponenttitasolla. Vaihtosuuntaajalle sekä ohjauslohkolle käytettiin valmiita ohjelmiston kirjastosta löytyviä lohkoja. Simulaatiomallin rakentaminen tehtiin käyttämällä MatGPT:tä sekä ChatGPT:n selainversiota.

Työn tuloksena saatiin rakennettua yksinkertainen simulaatiomalli taajuusmuuttajan pääpiirille käyttäen tekoälyä. Mallin toiminta pystyttiin verifioimaan, ja rakentamisprosessi saatiin dokumentoitua ja eriteltyä selkeisiin kokonaisuuksiin.

Työstä saatujen tulosten pohjalta pystytään muodostamaan johtopäätös, että simulaatiomallien rakentaminen tekoälyllä ei ole vielä käytännöllistä. Tekoälyä koskevat rajoitteet ja simulaatiomallien ohjelmalliseen rakentamiseen liittyvän tiedon harvuus vaikeuttaa ja hidastaa rakentamisprosessia. Myös tekoälylle opetetusta tiedosta muodostuvan tietokannan rajoittuminen haittaa sovellettavuutta. Manuaalinen simulaatiomallien rakentaminen on vielä nopeampaa ja johtaa parempaan lopputulokseen. Voidaan kuitenkin olettaa tekoälyn kehittyvän myös simulaatiomallien rakentamisessa. On myös mahdollista, että ChatGPT:n uusin versio, GPT-4, onnistuu tehtävässä jo edeltäjänsä paremmin.

Avainsanat: ChatGPT, MATLAB, Simulink, taajuusmuuttaja, tekoäly

Tämän julkaisun alkuperäisyys on tarkastettu Turnitin Originality Check -ohjelmalla.

ALKUSANAT

Tämä tutkielma toimii päätöksenä kandidaatinvaiheen opinnoilleni. Työn aihe oli mielenkiintoinen, koska aloittaessani työstämään tutkielmaa, en ennalta tiennyt, miten mallin rakentaminen tekoälyllä onnistuisi. Esiintyneistä haasteista huolimatta työlle asetetut tavoitteet mielestäni saavutettiin, ja olen tyytyväinen saatuun lopputulokseen. Työtä tehdessä opin myös paljon tekoälyn toiminnasta.

Haluan esittää kiitokset Danfossille mielenkiintoisesta ja ajankohtaisesta aiheesta. Haluan myös kiittää ohjaajiani Oskaria Danfossilta sekä Paavoa yliopistolta, saamastani ohjeistuksesta ja parannusehdotuksista.

Joni Näsinkoski
Tampereella, 05.02.2024

SISÄLLYSLUETTELO

1. JOHDANTO	1
1.1 Tutkimuksen tavoite	1
1.2 Tutkimusmenetelmät ja aiheen rajaaminen	2
2. TAUSTATIETOA.....	3
2.1 Tekoäly ja ChatGPT	3
2.1.1 Historia	3
2.1.2 Toimintaperiaate	4
2.1.3 ChatGPT	5
2.2 Taajuusmuuttaja	7
2.2.1 Taajuusmuuttajan pääpiiri	7
2.2.2 Taajuusmuuttajan toiminta	8
2.2.3 Ohjaus ja modulointi	9
2.3 MATLAB Simulink ja MatGPT	10
3. MALLIN RAKENTAMINEN.....	11
3.1 Tasasuuntaaja	11
3.2 Laitteen muut osat	16
4. TULOSTEN ESITTELY	21
4.1 Tekoälyn tuottama koodi ja simulaatiomalli	21
4.2 Mallin rakentamisprosessi	26
4.2.1 Rakentamisprosessin vaiheet	27
4.2.2 Työskentelytavat	29
4.3 Vaikeudet ja kehityskohteet.....	30
5. JOHTOPÄÄTÖKSET	33
5.1 Tavoitteisiin pääsy	33
5.2 Tilanearvio	34
LÄHTEET	35
LIITE A: CHATGPT:N TUOTTAMA KOODI TAAJUUSMUUTTAJASIMULOINTIIN	37

LYHENTEET

AC	Alternating current Vaihtovirta
AI	Artificial intelligence Tekoäly
DC	Direct current Tasavirta
DITV	Data, informaatio, tietämys, viisaus
GPT	Generative pre-trained transformer Generatiivinen esikoulutettu muunnin
IGBT	Insulated gate bipolar transistor Eristyshilainen bipolaaritransistori
MOSFET	Metal-oxide-semiconductor field effect transistor Metallioksidipuolijohdekanavatransistori
PAM	Pulse-amplitude modulation Pulssinamplitudimodulaatio
PWM	Pulse-width modulation Pulssinleveysmodulaatio
RLHF	Reinforced learning with human feedback Vahvistettu ihmisen palautteeseen perustuva oppiminen

1. JOHDANTO

Tekoäly (AI) on käsite, joka on saanut paljon julkisuutta viime vuosien aikana saavuttamiensa edistysaskelien ansiosta. Tekoälyn pitkään jatkunut kehitys on tuottanut useita vakuuttavia tuloksia, ja sen potentiaali vaikuttaa rajattomalta. Tekoälyn kantama potentiaali onkin nostanut sen yhdeksi eniten kiinnostusta herättävistä konsepteista tulevaisuuden kehityksen kannalta.

Tekoälylle etsitään uusia sovelluskohteita tauotta, ja niitä myös löydetään lupaavaan tahtiin. Tekoälyä sovelletaan jo laajalti esimerkiksi lääketieteessä. Joissakin tapauksissa tekoäly pystyy tunnistamaan vakavia sairauksia aikaisemmassa vaiheessa kuin ihminen ja pystyy auttamaan lääkäreitä tekemään tarkempia diagnooseja [1, 2]. Lisäksi tekoälyn hyödyntämistä on tutkittu sairaanhoitajien koulutuksessa positiivisin tuloksin [3]. Tekoälyllä uskotaan myös olevan suuri rooli taistelussa ilmastonmuutosta vastaan ja liikennejärjestelmien kehittämisessä [1].

Tekoäly suoriutuu jo monesta aiemmin ihmisen vaatineesta tehtävästä, ja sen kehitykseen panostetaan jatkuvasti. Voidaan siis sanoa, että odotukset ovat korkealla tekoälyn suhteen. Nämä merkittävät tekoälyä koskevat saavutukset yhdessä sen enteilevän tulevaisuuden potentiaalinn kanssa toimivat motivaationa tämän tutkimuksen tekemiselle.

1.1 Tutkimuksen tavoite

Tämän tutkimuksen tavoitteena on kartoittaa tämänhetkisen suurelle yleisölle saatavilla olevan tekoälyn kyvykkyyttä rakentaa simulaatiomalleja sähkötekniikan sovelluskohteisiin. Sähkötekniikassa hyödynnetään erilaisia simulointiohjelmia laajalti työelämässä ja koulutuksessa. Erilaisia simulaatiomalleja tarvitaan työelämässä esimerkiksi tuotesuunnitteluun ja verifiointiin. Koulutuksessa simulaatioita hyödynnetään oppimisen tukena, ja ne usein auttavat opiskelijoita ymmärtämään teoriapohjaisia konsepteja syvällisemmin.

Erilaisia tilanteita, joissa simulaatioita voidaan hyödyntää, on käytännössä rajattomasti. Tästä johtuen voidaan olettaa, että kysyntä ja tarve simulaatiomalleille tulevat kasvamaan tulevaisuudessa. Simulaatiomallien rakentaminen manuaalisesti voi kuitenkin olla työlästä ja vaatia paljon resursseja. Tästä johtuen olisikin kätevää, jos

tekoäly voitaisiin valjastaa mallien luomiseen. Tätä kautta säästettäisiin resursseja ja mahdollisesti minimoitaisiin inhimilliset virheet mallin rakentamisessa.

1.2 Tutkimusmenetelmät ja aiheen rajaaminen

Tässä tutkimuksessa rakennetaan yksinkertaisen taajuusmuuttajasovelluksen pääpiirin simulaatiomalli MATLAB:iin pohjautuvalle Simulink-ohjelmistolle. MATLAB:in käytetty versio on R2023a. Simulaatiomallin luomiseen käytettäväksi tekoälyksi valittiin ChatGPT, joka on tullut tunnetuksi viime vuosien aikana. ChatGPT:n ja Simulink:in väliseen viestintään on myös olemassa MatGPT-niminen rajapintasovellus, jota voi käyttää alustojen väliseen viestintään.

Tämä tutkimus on suoritettu yhteistyössä Danfoss Drives:in kanssa. Systemiksi, jolle simulaatiomalli rakennetaan, valittiin yksinkertainen versio tyypillisestä taajuusmuuttajan pääpiiristä. Käytettäväksi tekoälyksi valittiin ChatGPT sen saavuttaman julkisuuden ja sen tarjoamien käyttömahdollisuuksien vuoksi. Simulaatio-ohjelmistoksi, jolle malli rakennetaan, valittiin MATLAB Simulink, koska se on laajalti alan yrityksillä ja sähkötekniikan koulutuksessa käytössä. Se tarjoaa myös mahdollisuuden MatGPT:n käyttöön.

2. TAUSTATIIETOA

Tässä luvussa on esitetty työn kannalta tärkeiden konseptien, kuten tekoälyn, taajuusmuuttajan ja MATLAB Simulink -ohjelmiston toimintaa. Työssä keskitytään ensisijaisesti mallin rakentamiseen, joten esitetty teoria on yleistasoista. Sen tuntemisesta on kuitenkin apua työn ja sen tavoitteiden ymmärtämisessä.

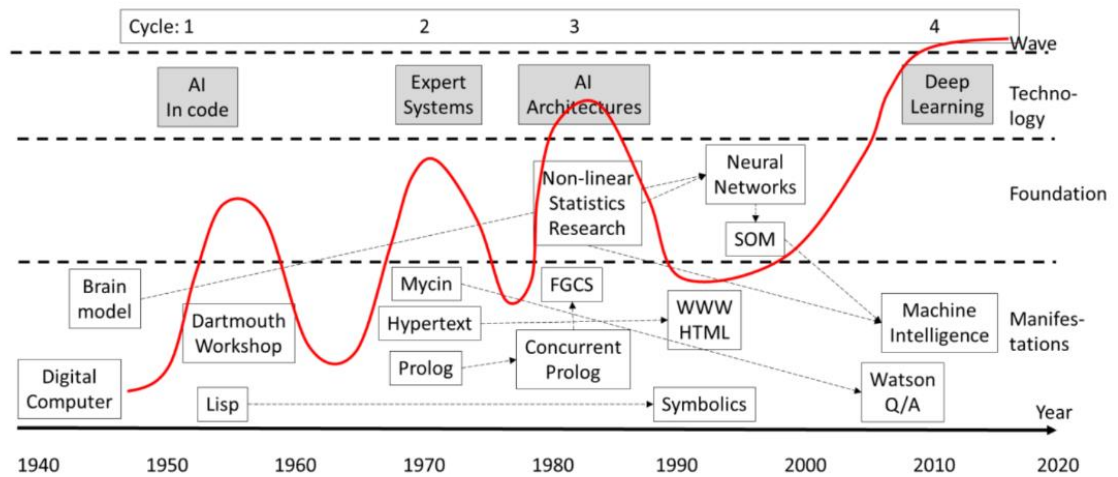
2.1 Tekoäly ja ChatGPT

Tekoälylle ei ole yhtä tarkkaa määritelmää, vaan määritelmät muuttuvat teknologian kehittyessä. Kaur et al. [4] kirjoittamassa artikkelissa tekoäly määritellään tavaksi kouluttaa koneita suorittamaan älykkyyttä vaativia tehtäviä. Jaakkolan et al. [5] kirjoittamassa konferenssijulkaisussa tekoäly määritellään tietokoneen kyvyksi suorittaa tehtäviä, jotka normaalisti vaativat älykkään olennon. Tällaisia tehtäviä ovat esimerkiksi puheen ja kuvien tunnistaminen, kielten kääntäminen ja päätöksenteko. Tekoälyn eroavaisista määritelmistä huolimatta niiden pääpiirre on sama: tavoitteena on oppiva kone, joka kykenee suoriutumaan älykkyyttä vaativista tehtävistä.

2.1.1 Historia

Käsitys tekoälystä ja sen tarjoamista mahdollisuuksista on muuttunut paljon sen kehityksen myötä. Tekoäly sen nykyisellä nimellä (AI) tuotiin ensimmäistä kertaa esille vuonna 1956 Dartmouthin konferenssissa McCarthyn et al. toimesta. Tällöin ehdotettu tutkimus lähti ajatuksesta, että oppimisen tai muun älyllisen prosessin toimintaperiaatteen voisi kuvata niin tarkasti, että kone kykenee simuloimaan sen. [6] Tutkimuksen tavoite oli jo silloin suorittaa koneella tehtäviä, jotka perinteisesti vaativat ihmisen.

Tekoälyn kehitys hidastui ja myös kiinnostus sitä kohtaan laantui vuoden 1969 jälkeen, kun esille nousi koneoppimista koskevia ongelmia. Vuosituhannen lopulla kuitenkin tapahtui uusia läpimurtoja, jotka herättivät uudelleen mielenkiinnon tutkimusalaan. Etenkin 2000-luvun alkupuolella saataville tullut massadata (eng. big data) antoi kehitykselle uutta eloa. [7] Tekoälyn kehityskulku on esitetty kuvassa 1.

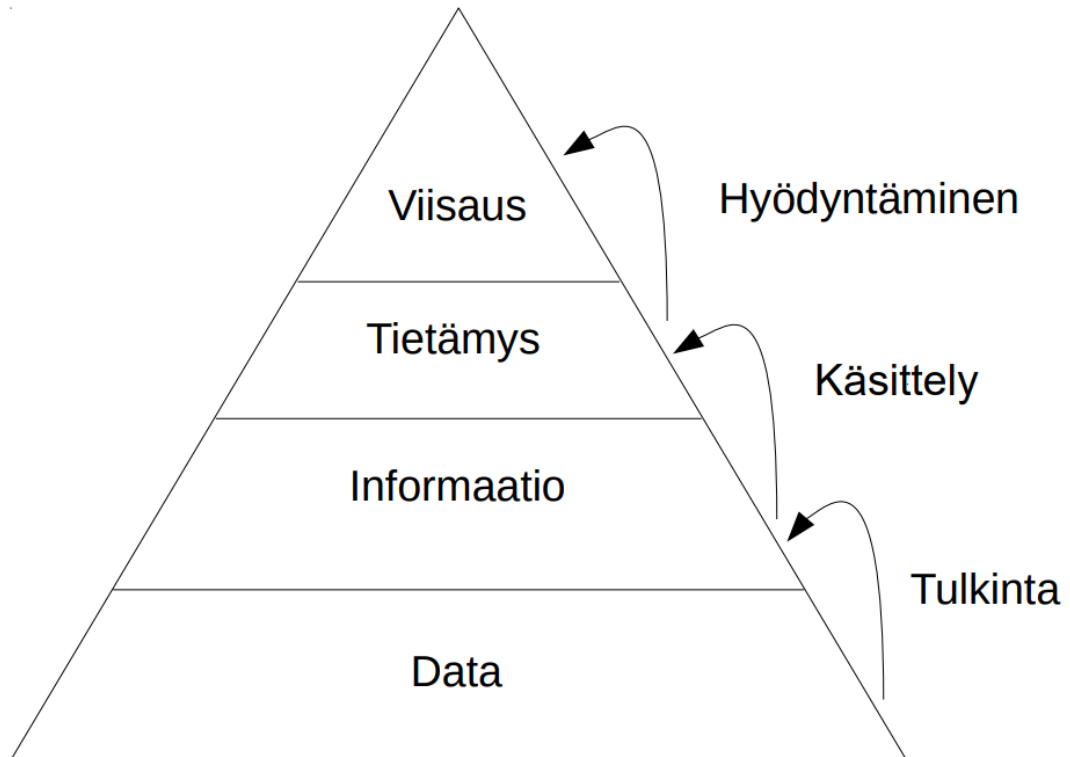


Kuva 1. Tekoälyn historia [5]

Nykypäivänä tekoäly on levinnyt jo pitkälle pelkästä teoriasta käytännön maailmaan, ja sille haetaan jatkuvasti uusia sovelluskohteita. Tekoäly jakaa paljon mielipiteitä sen tulevaisuutta koskien: osa ihmisistä pelkää tekoälyn kehittyvän niin radikaalisti, että se syrjäyttää ihmiset kokonaan. Toiset taas vähättelevät sen potentiaalia ja korostavat sitä koskevia rajoitteita. Samaa mieltä ollaan kuitenkin siitä, että tekoäly tulee olemaan läsnä tulevaisuuden yhteiskunnassa.

2.1.2 Toimintaperiaate

Tekoälyn toiminta perustuu pitkälti suurien datamäärien käsittelyyn ja niiden pohjalta toistuvien kaavojen löytämiseen. Toimintaa voidaan kuvastaa yleisellä tasolla DITV-pyramidin (data, informaatio, tietämys, viisaus) avulla. Pyramidi on suomennettu ja esitetty kuvassa 2.



Kuva 2. DITV-pyramidi, suomennettu [5]

Ensimmäisessä pyramidin osassa on kaikki saatavilla oleva data, jota ei ole vielä tulkittu informaatioksi. Siirryttäessä seuraavaan vaiheeseen kone liittää dataan tulkinnan, jolloin siitä muodostuu informaatiota. Käsittelmällä informaatiota erilaisien algoritmien avulla voidaan päästä tietämykseen. Tietämyksen voi ajatella olevan informaatiota, jota voidaan hyödyntää kontekstissa. Viisaudella tässä yhteydessä tarkoitetaan saadun tietämyksen soveltamista asianmukaisesti. [5]

Edellä esitetty malli tekoälyn toiminnalle on hyvin yleistasoisen ja yksinkertaistettu verrattuna modernin tekoälyn todelliseen toimintaan. Malli ei ota kantaa monimutkaisiin datanprosessointialgoritmeihin, syväoppimiseen tai muihin nykyaikaisiin mekanismeihin, mutta se antaa alkeellisen käsityksen tekoälyn toiminnasta. Samankaltaista mallia voitaisiin soveltaa pitkälti myös ihmisajatteluun.

2.1.3 ChatGPT

ChatGPT on OpenAI-nimisen yhdysvaltalaisen vuonna 2015 perustetun tutkimusorganisaation kehittämä keskustelubotti. ChatGPT julkistettiin vuoden 2022 lokakuussa, ja se saavutti nopeasti suuren suosion asiantuntijoiden, yritysten sekä yksityisten henkilöiden joukossa. [8]

Organisaation perustamisesta lähtien OpenAI on tutkinut ja kehittänyt generatiivisia muuntavia kielimalleja, ja eräs näistä on GPT-3.5-turbo, joka tunnetaan ChatGPT:n käyttämänä mallina [8]. Tämä OpenAI:n merkitsevimmäksi noussut generoiva malli edustaa monille tekoälyn kehitystä.

Muuntavat kielimallit kehitettiin toistuvien neuroverkkojen rajoitteiden ylipääsemiseksi. Neuroverkot ovat lyhyesti selitettynä toisiinsa yhdistyviä solmupisteitä, joilla on sisääntulo, painotuskerroin ja raja-arvo. Jos solmupisteen raja-arvo saavutetaan, niin data lähetetään verkoston seuraavalle kerrokselle. Jos rajaa ei saavuteta, niin dataa ei lähetetä. Neuroverkot ovat koneoppimisen alakategoria ja ne ovat syväoppimisen ytimessä. Toistuvia neuroverkkoja kuitenkin rajoittaa niiden rinnastamisen vaikeus ja rajallinen konteksti. Transformointi ylittää rajoitteet käyttämällä toistuvuuden sijaan itsehuomiointimekanismia, joka mahdollistaa rinnakkaisuuden ja pitkien aikavälien riippuvuuksien huomioinnin. [8, 9]

ChatGPT:n käyttämä arkkitehtuuri GPT-3 on abstraktissa mielessä todella suuri. Data, jonka pohjalta malli on koulutettu, pohjautuu neljään tiedonlähteeseen. Nämä lähteet sisältävät massiivisia kokoelmia internetistä kerääntynyttä tietoa, kirjakokoelmia sekä Wikipedian sisällön. Yhteenlaskettuna dataa on 489 miljardin tokenin verran. Yksi tokeni vastaa noin neljää englannin kielen merkkiä, joten tietopohjan kooksi voidaan arvioida noin 374 miljardia sanaa. Tätä tietoa käytetään 175 miljardin parametrin kouluttamiseen 96 kerroksen kesken. [8]

Malli, johon ChatGPT perustuu, on hienosäädetty erityisesti keskustelua varten. Osa tätä hienosäätöä on ihmispalautteeseen pohjautuva oppiminen. ChatGPT käsittelee käyttäjän antaman syötteen ja generoi nopeasti sopivan ihmismäisen vastauksen syötteeseen. Algoritmi arvioi vastauksen onnistumista käyttäjän antaman palautteen pohjalta ja pyrkii maksimoimaan käyttäjätyytyväisyyden tämän perusteella. [8, 10] ChatGPT onnistuu generoimaan parempia vastauksia kysymysten pohjautuessa dataan, jolla malli on koulutettu [11].

Generoivat kielimallit, kuten ChatGPT voivat luoda harhaanjohtavan vaikutelman niiden älykkyydestä. Ihmismäisistä vastauksista huolimatta mallit eivät ole virheettömiä. Mallit eivät ymmärrä sanojen merkityksiä, ja niiden päättelykyvyssä sekä tiedoissa esiintyy usein virheitä. [12] Tämä on huomioitava generoivia kielimalleja hyödyntäessä. Oikein sovellettuina generoivista kielimalleista voi hyötyä suuresti nyt ja tulevaisuudessa, satunnaisista virheistä huolimatta [13].

Työssä käytetään paljon sanaa ”tekoäly” mallin rakentamisen yhteydessä. Tällä viitataan tässä työssä käytettävään tekoälyyn, eli ChatGPT:n versioon GPT-3.5-turbo. On

mainitsemisen arvoista, että tekoälyn sovelluksia on muitakin kuin ChatGPT. Näitä ei käsitellä työssä.

2.2 Taajuusmuuttaja

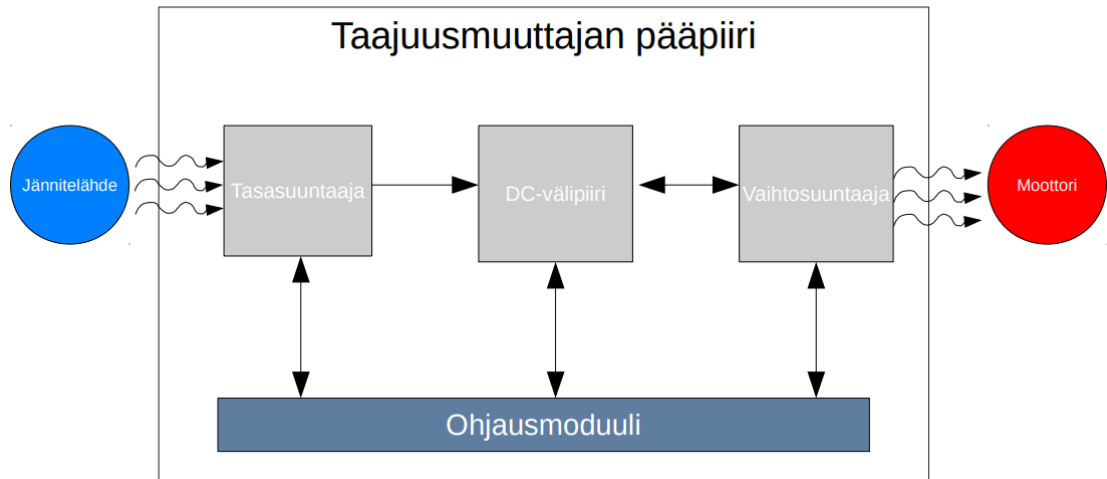
Nykymaailmassa yksi tärkeimmistä toimintaa ohjaavista periaatteista on kestävä kehitys. Etenkin teollisuudessa yksi kestävä kehityksen vaatimuksista on, että moottorikäyttö on energiatehokasta. Tämä on yksi motivaattori tehoelektroniikan laitteiden, kuten taajuusmuuttajan kehittämiseksi.

Taajuusmuuttaja on sähkölaite, jonka yleisin käyttökohde on ohjata sähkömoottorin toimintaa muuttamalla sen tehonsyötön taajuutta sekä jännitettä, vaikuttaen moottorin pyörimisnopeuteen. Taajuusmuuttajat mahdollistavat sähkömoottorin ohjaamisen prosessin kannalta optimaalisella tavalla säätämällä moottorin pyörimisnopeutta ja vääntömomenttia. Tällä tavoin pystytään takaamaan energiatehokkuus, laskemaan melutasoa sekä vähentämään mekaanista rasitusta, mikä pidentää osien käyttöikää. Taajuusmuuttajia hyödynnetään myös moottorin käynnistyksen ja sammutuksen yhteydessä. [14]

Taajuusmuuttajia on kehitetty jo monen vuosikymmenen ajan, ja niissä käytettävä teknologia on kehittynyt huomattavasti. Modernit taajuusmuuttajat sisältävät pitkälle kehitetyn ohjelmiston ja useita oheislohkoja. Tässä tutkielmassa on tarkoitus keskittyä taajuusmuuttajan pääpiiriin, joten työssä ei huomioida oheislohkoja tai esimerkiksi suojaustoimintoja tarkemmin. Taajuusmuuttajan pääpiirin tyypillinen rakenne on esitetty kuvassa 3.

2.2.1 Taajuusmuuttajan pääpiiri

Taajuusmuuttajan pääpiirillä tarkoitetaan virtapiiriä, jossa käsiteltävä kuormavirta kulkee. Tämä virtapiiri koostuu ensisijaisesti tasasuuntaajasta, DC-välipiiristä sekä vaihtosuuntaajasta eli invertteristä. Mainittujen lohkojen lisäksi tässä työssä oleellisia mallin osia ovat kolmivaiheinen jännitelähde, moottorikuorma ja ohjausmoduuli.



Kuva 3. Taajuusmuuttajan pääpiiri, suomennettu [15, s. 48]

Sovelluskohteesta riippuen pääpiirin eri lohkoille on saatavilla erilaisia konfiguraatioita. Tasasuuntaaja voi koostua diodeista, jolloin se on ohjaamaton, tai siinä voidaan hyödyntää tyristöireita tai IGBT-transistöireita, jolloin sen toimintaa voidaan ohjata. DC-välipiiri yksinkertaisimmillaan koostuu suuresta kondensaattorista, mutta se voi myös sisältää kuristimen tai tehokertoimen muokkauspiirin. Vaihtosuuntaaja toteutetaan nykyään transistöireilla. Taajuusmuuttajia on saatavilla myös ilman välipiiriä, jolloin niitä kutsutaan suoriksi konverttereiksi (engl. Direct converter). Mahdollisia transistorivaihtoehtoja vaihtosuuntaajalle on bipolaari-, IGBT- ja MOSFET-transistorit, joista IGBT on yleisimmin käytössä. Vaihtosuuntaajan transistöireiden kanssa on rinnankytketty diodi, jonka pääasiallinen tehtävä on johtaa induktiivista virtaa transistöirin ollessa poiskytkettynä. Edellä esiteltyjä lohkoja yhdistää se, että ne perustuvat lähes aina samoihin komponentteihin ja toimintaperiaatteisiin. Näiden kolmen lohkon lisäksi tarvitaan vielä ohjausmoduuli. Moduulin tehtävänä on ohjata puolijohdekomponentteja, ohjata dataliikennettä, mitata toimintaa, havaita viat toiminnassa ja suojata laitetta ja moottoria. [15, s. 48]

2.2.2 Taajuusmuuttajan toiminta

Taajuusmuuttaja ottaa muokattavan jännitteen yleensä kolmivaiheisesta vaihtojännitelähteestä, kuten sähköverkosta. Taajuusmuuttajan sisääntulo kytketään jännitelähteen kolmeen vaiheeseen, josta vaihtosuuntainen sähkövirta kulkee tasasuuntaajaan. Tasasuuntaajan konfiguraatio sallii sähkövirran kulkemisen vain yhteen suuntaan, joten sen ulostulo on pulssimuotoista tasavirtaa. Ohjatulla tasasuuntaajalla voidaan kontrolloida komponenttien johtavuutta. Tällä tavoin voidaan rajoittaa suuria käynnistysvirtoja. [15, s. 50]

Tasasuunnattu sähkö kulkee tasasuuntaajalta DC-välipiirille. Välipiirin pääasiallinen tehtävä on toimia energiavarastona, josta voi ottaa energiaa, tai johon voi palauttaa energiaa invertterin kautta moottorin kuormituksen muuttuessa äkillisesti. Välipiiri voi toimia myös irtikytkentänä tasa- ja vaihtosuuntaajan välillä, ja sitä voidaan hyödyntää verkon häiriöiden poistamisessa. [15, s. 53]

Vaihtosuuntaaja eli invertteri on viimeinen pääpiirin lohko ennen ajettavaa kuormaa. Invertteri ottaa tasasuunnatun sähköön välipiiriltä ja muuttaa sen takaisin vaihtosuuntaiseksi sähköksi. Tämä tapahtuu kontrolloimalla vaihtosuuntaajan transistoreiden johtavuutta, millä saadaan tuotettua moottorille haluttu jännite ja taajuus. [15, s. 57]

2.2.3 Ohjaus ja modulointi

Taajuusmuuttajan ohjattavien puolijohdekomponenttien ohjaussignaalit tulevat ohjausmoduulista. Moduulissa sijaitsee ohjauslogiikka, jolla määritetään, missä järjestyksessä eri komponenttien tulee olla johtavassa tilassa, jotta laitteen ulostuloon saadaan haluttu jännite ja taajuus.

Ohjauksen toteuttamiseen on monia eri tapoja, joista lähes kaikki on toteutettu ohjelmistolla. Yksinkertaisimmat taajuusmuuttajasovellukset eivät tarvitse monimutkaista ohjelmaa, mutta kriittiset sovelluskohteet, jotka tarvitsevat tarkkaa ja nopeaa säätöä edellyttävät monimutkaisempien ohjelmistojen hyödyntämistä. [15, s.67]

Taajuusmuuttajissa moduloinnilla tarkoitetaan tapaa, jolla vaihtosuuntaajan transistoreita ohjataan, jotta saadaan muokattua DC-välipiirin jännitteestä haluttu jännite ja taajuus ajettavalle kuormalle. Nämä voidaan generoida kahden modulaatioperiaatteen mukaisesti: pulssinamplitudimodulaatiolla (PAM) tai pulssinleveysmodulaatiolla (PWM). Modulaatiomenetelmistä PWM on enemmän käytetty, koska sen vaatimukset laitteistolta ovat pienemmät ja pienien taajuuksien ohjaus on parempi. [15, s. 60]

Pulssinleveysmodulaatio perustuu vaihtosuuntaajassa olevien transistoreiden nopeaan kytkemiseen johtavan ja ei-johtavan tilan välillä. Tällä tavoin voidaan luoda halutun pituisia jännitepulsseja laitteen ulostuloon. Laitteen ulostulon jännitettä voidaan ohjata muuttamalla pulssien kestoa ja niiden välistä taukoa. Ulostulon taajuuteen voidaan vaikuttaa muuttamalla nopeutta, jolla jännitepulssien polaarisuus vaihtelee. [15, s. 61]

Ohjausmoduuli määrittää transistoreille vaihtojärjestyksen, jolla saadaan tuotettua jännitekäyrämuoto, jonka taajuus ja amplitudi vastaavat laitteelle määritetyt arvoja. Ohjauksen tehtävänä on myös varmistaa, että vaihtosuuntaajan saman haaran transistorit eivät ole johtavassa tilassa samanaikaisesti oikosulun välttämiseksi.

2.3 MATLAB Simulink ja MatGPT

Tässä työssä rakennettu simulaatiomalli on tehty MATLAB:iin pohjautuvalle simulointialustalle Simulink:ille. MATLAB on ohjelmointiin ja numeeriseen laskentaan tehty alusta, jota käytetään paljon data-analyysiin, algoritmien kehittämiseen sekä erilaisiin simulointeihin [16]. Työssä käytetty MATLAB-versio on R2023a.

Simulink on MATLAB:iin pohjautuva graafinen ohjelmointiympäristö, joka soveltuu dynaamisten järjestelmien mallintamiseen ja mallipohjaiseen suunnitteluun. Sillä pystyy tekemään systeemitason suunnittelua ja simulointeja sekä automaattisesti generoimaan koodia. Simulink on integroitu MATLAB:iin, mikä takaa alustojen välisen yhteistoiminnan. [17]

Työssä rakennettu Simulink-malli käyttää "Simscape electrical" -kirjaston komponentteja. Komponenttien hyödyntäminen edellyttää, että "Simscape electrical" -laajennus on lisätty Simulink:iin.

Työssä on käytetty mallin rakentamisessa MatGPT-nimistä MATLAB-pohjaista rajapintasovellusta. MatGPT mahdollistaa ChatGPT:n ohjelmointirajapinnan käytön MATLAB:in kautta. [18] MatGPT:n käytöstä tässä työssä on kerrottu mallin rakentamisen yhteydessä.

3. MALLIN RAKENTAMINEN

Tässä kappaleessa esitellään simulaatiomallin rakentamisen kulku tekoälyä käyttäen. Mallin rakentamisen yhteydessä ei ole dokumentoitu kaikkia aiempia yrityksiä saada tekoäly rakentamaan malli, vaan tarkoituksena on antaa käsitys rakentamisprosessista ja siihen liittyvistä haasteista. Tekoälylle annetut käskyt noudattavat samaa esitystapaa, jota on käytetty ChatGPT:n syötteen kehittämistä käsittelevässä kirjoituksessa [19].

Mallin rakentaminen aloitetaan käyttäen MatGPT:tä. Rakentaminen MatGPT:ssä aloitetaan sillä, että tekoälylle annetaan rooli. Tämä tarkoittaa käytännössä sitä, että tekoälylle kerrotaan, millaisia vastauksia siltä odotetaan. MatGPT:ssä olevassa "English to Simulink-model"-roolissa alustamiseen kuuluu ohjeistus, että tekoäly generoi MATLAB-koodin, joka lisää tarvittavat lohkot, yhdistää lohkot toisiinsa ja järjestää mallin. Tämän lisäksi sovellus ajaa generoidun koodin ja valinnaisesti tarkistaa onko siinä virheitä. Mallin rakentaminen onnistuu myös ilman MatGPT-lisäosaa ChatGPT:n selainversiossa, mutta tällöin on tarkennettava, että halutaan tekoälyn generoivan MATLAB-koodia.

Alustava käsky voi olla hyvin yleistasoinen. Käskynä voisi suoraan sanoa tekoälylle "rakenna taajuusmuuttajan simulaatiomalli". Tällöin generoitu koodi hyödyntäisi valmiita "DC to AC" ja "AC to DC"- lohkoja. Työn lähtökohtana on kuitenkin mallin rakentaminen moduulikohtaisesti. Moduulikohtainen suunnittelu antaa paremman käsityksen tekoälyn kyvyistä rakentaa malleja myös muihin sovelluskohteisiin ja antaa mahdollisuuden mukauttaa malleja.

3.1 Tasasuuntaaja

Aloitetaan mallin rakentaminen tasasuuntaajalle käyttäen MatGPT:tä. Alustavan käskyn tulee olla suuntaa antava, mutta sen ei tarvitse olla täydellinen. Tasasuuntaajan rakentaminen voidaan aloittaa esimerkiksi käskyllä

- "Can you build me a three-phase rectifier model using ideal diodes"

Käskyn tuottama koodi ei vastaa haluttua tulosta. Kuvassa 4 on esitetty osa tekoälyn generoimasta koodista.


```

model = 'three_phase_rectifier';
open_system(new_system(model));

% Add three-phase AC voltage source
add_block('simulink/Sources/Sine Wave', [model '/AC Voltage']);
set_param([model '/AC Voltage'], 'Amplitude', '100', 'Frequency', '50');

% Add three ideal diodes
add_block('simulink/Discrete/Unit Delay', [model '/Unit Delay1']);
add_block('simulink/Discrete/Unit Delay', [model '/Unit Delay2']);
add_block('simulink/Discrete/Unit Delay', [model '/Unit Delay3']);

% Add three-phase rectifier
add_block('simulink/Discrete/Relational Operator', [model '/Relational Operator1']);
add_block('simulink/Discrete/Relational Operator', [model '/Relational Operator2']);
add_block('simulink/Discrete/Relational Operator', [model '/Relational Operator3']);

% Add output blocks
add_block('simulink/Sinks/Scope', [model '/Scope']);
add_block('simulink/Sinks/To Workspace', [model '/To Workspace']);

% Connect blocks
add_line(model, 'AC Voltage/1', 'Relational Operator1/1');
add_line(model, 'AC Voltage/1', 'Relational Operator2/1');
add_line(model, 'AC Voltage/1', 'Relational Operator3/1');
add_line(model, 'Relational Operator1/1', 'Unit Delay1/1');
add_line(model, 'Relational Operator2/1', 'Unit Delay2/1');
add_line(model, 'Relational Operator3/1', 'Unit Delay3/1');

```

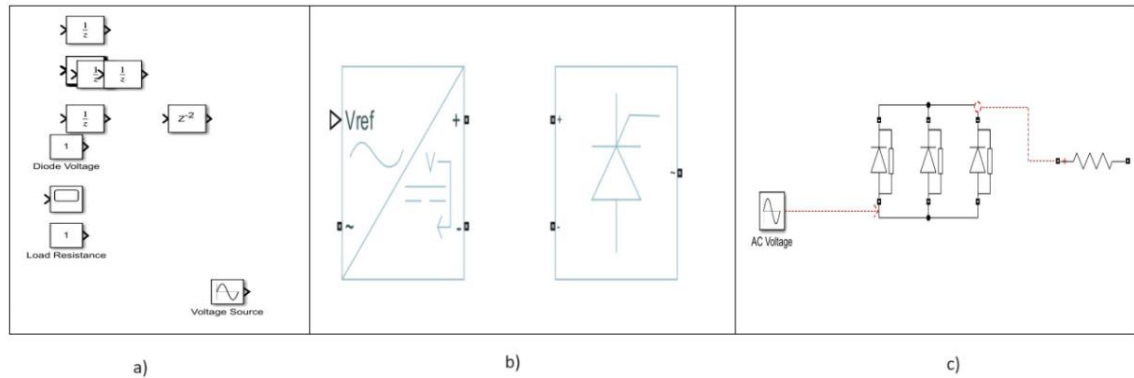
Kuva 4. Esimerkki tekoälyn generoimasta koodista

Tekoälyn generoima koodi käyttää väärinä komponentteja mallin rakentamisessa, eikä osaa viitata komponentteihin koodissa oikealla tavalla. Esimerkiksi ohjelmassa käytetään diodien sijaan viivekomponentteja.

Koska tekoälyn generoima koodi on kaukana halutusta, on helpompaa muotoilla alustava käsky uudelleen. Alustava käsky voidaan esittää samassa keskustelussa, jos halutaan tekoälyn käyttävän aiemmin keskustelussa esitettyä tietoa. Vaihtoehtoisesti käsky voidaan esittää uudessa keskustelussa, jos halutaan aloittaa kokonaan alusta. Aloitetaan tässä tapauksessa alusta.

- "Create a three-phase rectifier using diodes"

Vaikka käsky ei eroa paljoa alkuperäisestä, niin tekoälyn generoima koodi on täysin erilainen. Kuvassa 5 on esitetty kolme esimerkkiä siitä, millaiselta alustavana tuloksena saatavat mallit näyttäisivät. Vaikka tekoälyn generoima koodi ei rakenna mallia koodin sisältämien virheiden takia, voidaan kuvista hahmottaa, kannattaako käsky muotoilla uudelleen.



Kuva 5. Eri alustavia tuloksia visualisoituna. Mallissa a) on käytetty väärää komponentteja. Mallissa b) on käytetty valmiita lohkoja moduuleille. Malli c) on lähimpänä haluttua tulosta.

Ensimmäisen alustavan käskyn tuottama koodi rakentaisi kuvan a) mukaisen mallin. Tekoälyä käskettiin rakentamaan tasasuuntaajan malli käyttäen 'ideaalisia diodeita'. Saatu malli sisältää viive- ja vertailulohkoja, mutta se ei sisällä diodeita. Toisena alustavana käskynä käskettiin tekoälyä rakentamaan tasasuuntaajan malli käyttäen 'diodeita'. Toisen alustavan käskyn tuottama koodi rakentaisi kuvan c) mukaisen mallin. Malli sisältää jännitelähteen, kolme diodia ja kuormavastuksen. Tilanne b) kuvastaa mallia, jossa tekoäly hyödyntää haluttua korkeamman tason lohkoja. Kuvassa 5 esitetyistä tuloksista a) ja b) voidaan luokitella huonoiksi työn tavoitteita ajatellen. Tulos c) on myös virheellinen, mutta mallia voidaan lähteä korjaamaan.

Alustavan käskyn jälkeen opetetaan tekoälyä korjaamaan alustavan tuloksen sisältämät virheet. Tämä on iteratiivinen prosessi, joka joudutaan toistamaan useaan kertaan. Jos komponenttien lukumäärä on virheellinen, ohjeistetaan komponenteille oikeat lukumäärät. Jos koodissa käytetään väärää komponenttia, täytyy tekoälylle kertoa mistä kirjastosta löydetään oikea komponentti. Jos tekoäly ei osaa viitata komponentteihin oikein, täytyy sille ohjeistaa oikea tapa. Tämä tarkoittaa polun täsmenmistä, josta komponentti löytyy. Saatua tulosta täsmennetään jatkokäskyllä:

- “The model needs to be three-phase, so it should contain at least 6 diodes. The load is not needed yet”

Saatu koodi lisää jännitelähteen ja kuusi diodia, mutta viittaukset diodeihin ja niiden väliset yhteydet ovat väärin. Tekoälylle voidaan kertoa, että sen viittaukset komponentteihin ovat virheellisiä, jolloin tekoäly muuttaa viittauksia. Tekoäly ei kuitenkaan onnistu omin avuin korjaamaan viittaustekniikkaa. Kerrotaan tekoälylle oikea viittaustekniikka, kuten esitetty alla.

- “The correct path for the diodes is 'sps_lib/Power Electronics/Diode'”

Nyt tekoäly onnistuu generoimaan koodin, joka lisää jännitelähteen ja kuusi diodia malliin. Komponentteja ei kuitenkaan ole aseteltu mallin sisällä, eikä niiden välillä ole yhteyksiä. Tämä johtuu siitä, että tekoälyn käyttämä jännitelähde ei ole yhteensopiva diodien kanssa, eivätkä viittaukset komponenttien portteihin ole toimivia. Kerrotaan tekoälylle ensin polku jännitelähteeseen, jota halutaan käyttää.

- “The correct path for the voltage source is ‘sps_lib/Sources/Three-Phase Source’”

Tämän jälkeen korjataan viittaustyyli komponenttien portteihin. Oikeaa viittaustyyliä voidaan yrittää muuttaa kertomalla tekoälylle viittaustyylin olevan väärä. Kuten komponenttien kanssa, tekoäly ei onnistu korjaamaan virhettä, ja myös viittaustyyli joudutaan opettamaan manuaalisesti.

- “The correct way to use `add_line` with Simscape components would be for example `add_line('model','AC Voltage/RConn1','Diode1/LConn1')`. RConn1 is first right-side port and LConn1 is first left side port”

Jotta simulaatiomalli toimisi, siihen täytyy lisätä vielä `powergui`-lohko. Lohko voidaan lisätä ohjeistamalla sen sijainnin tekoälylle. Tämän jälkeen kaikki tarvittavat komponentit löytyvät mallista. Komponenttien väliltä puuttuu vielä yhteyksiä. Käsketään tekoälyä lisäämään ne.

- “Can you add the necessary connections so that the diodes form a rectifier”

Tässä vaiheessa malli sisältää kaikki tasasuuntaajaan tarvittavat komponentit. Nopea katsaus simulaatiomalliin kuitenkin näyttää, että diodien väliset yhteydet eivät ole oikein. Tekoälyä voidaan käskä korjaamaan komponenttien väliset yhteydet oikean kaltaiseksi, mutta tekoäly ei onnistu siinä ilman apua.

MatGPT tulostaa vikakoodin keskustelujen kasvaessa pitkiksi, joten sillä on epäkäytännöllistä tehdä monia pieniä muutoksia koodiin. Ongelma voidaan kiertää sillä, että kopioidaan generoitu koodi uuteen keskusteluun, mutta tällöin aiemmin opetetut käytännöt joudutaan opettamaan uudestaan. Kyseisessä tilanteessa ChatGPT:n selainversio on käytännöllisempi. Myös selainversiolle joudutaan opettamaan samat käytännöt, mutta siinä keskustelun pituus ei aiheuta ongelmia.

Jatketaan mallin rakentamista ChatGPT:n selainversiossa kopioimalla MatGPT:n tuottama koodin ja antamalla sen ChatGPT:lle. Tämän yhteydessä ohjeistetaan tekoälyä analysoimaan saamansa koodi ja korjaamaan yhteydet komponenttien välillä.

- “Here is a code. Can you analyze the code and change the connections so that the rectifier works?”

Selainversioon siirtyminen tarkoittaa, että tekoälyllä ei ole tietämystä sille aiemmin ohjeistetuista korjauksista, ja tämän takia sen tekemät muutokset koodiin eivät ole heti halutun kaltaisia. Tekoäly yrittää esimerkiksi tehdä yhteyksiä porttien välillä, joita ei ole olemassa. Ongelmasta selvittää kuvaillemalla tekoälylle sen tekemät virheet, jolloin se pyrkii korjaamaan ne. Tässä tapauksessa porttien nimeämiseen liittyvä ongelma saadaan korjattua ohjeistuksella:

- “There is no LConn2 because diodes only have one port each side, the voltage source has 3 ports so there should be RConn2 and 3 for the source”

Annetun ohjeistuksen jälkeen tekoälyn generoima koodi sisältää oikeat yhteydet jännitelähteen ja diodien välillä, mutta siitä puuttuu vielä osa diodien välisistä yhteyksistä. Ohjeistetaan tekoälyä lisäämään diodien väliset puuttuvat yhteydet.

- “There are still some connections missing between the diodes that complete the rectifier. Can you add them?”

Lisätyt yhteydet eivät vielä ole tasasuuntaajan mukaiset. Tekoäly muuttaa yhteyksiä, kun sille kerrotaan yhteyksien olevan väärin, mutta se ei onnistu muodostamaan puuttuvia yhteyksiä. Tästä johtuen kerrotaan puuttuvat yhteydet tekoälylle manuaalisesti. Puuttuvat yhteydet nähdään simulaatiomallista ajamalla koodin MATLAB:issa.

Ohjeistetaan tekoälyä ensin palaamaan koodin versioon, joka sisältää yhteydet jännitelähteen ja diodien välillä. Tämän jälkeen ohjeistetaan yksi kerrallaan puuttuvat yhteydet diodien välille.

- “Go back to the initial code”
- “Add a connection between right ports of diodes 1 and 2”
- “Add a connection between right ports of diodes 2 and 3”
- “Add a connection between left ports of diodes 4 and 5”
- “Add a connection between left ports of diodes 5 and 6”

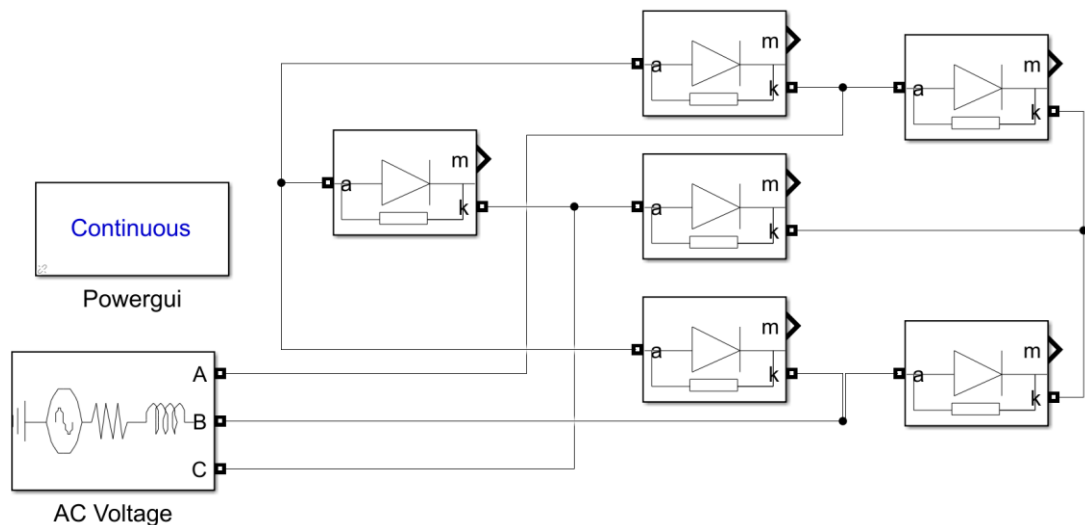
Manuaalisen ohjeistuksen jälkeen koodi sisältää kaikki tarpeelliset komponentit ja niiden väliset yhteydet. Tämän jälkeen asetetaan vielä moduulin lohkojen parametrit halutuiksi. Diodien ja powergui-lohkon oletusparametrit soveltuvat tähän malliin. Asetetaan jännitelähteen taajuudeksi 50 Hz ja vaiheiden väliseksi jännitteeksi 400 V.

- “Can you set the phase-to-phase voltage to 400 V and frequency to 50 Hz”

Koodi ei suoraan toimi, koska tekoäly käyttää virheellistä parametrin nimeä jännitteelle. Tämä korjautuu kertomalla, että jännitteen parametrin nimi on "Voltage", eikä "Amplitude", jolloin ongelma korjaantuu.

- "Parameter name for voltage is 'Voltage' instead of 'Amplitude'"

Tämän jälkeen saatu koodi voidaan ajaa MATLAB:issa. Tuloksena on Simulink-malli, joka sisältää jännitelähteen ja kuusi diodia sekä simulointiin tarpeellisen powergui-lohkon. Jännitelähde ja diodit ovat kytketty toisiinsa muodostaen tasasuuntaajan. Malli on esitetty kuvassa 6.



Kuva 6. Tasasuuntaajan malli

Tasasuuntaajan mallista nähdään, että ohjelma ei asettele diodeja perinteisen tasasuuntaajan mallin mukaisesti, mutta tämä on mallin toiminnan kannalta merkityksetöntä. Diodeissa näkyviä m-portteja voidaan käyttää niiden läpi kulkevien signaalien tarkasteluun, mutta niiden kytkeminen ei ole tarpeellista.

3.2 Laitteen muut osat

Laitteen muut moduulit voidaan rakentaa seuraamalla samaa käytäntöä, kun tasasuuntaajan tapauksessa. Mallin rakentamisessa toistuvat pitkälti samat ongelmat. Kun ongelmiin tiedetään ratkaisut, on mallin rakentaminen suoraviivaisempaa. Jatketaan rakentamista lisäämällä malliin DC-välipiiri.

- "Can you add a series RLC branch to the circuit?"

DC-välipiirin kondensaattorina käytettävän komponentin nimi on "series RLC branch". Tekoäly tulkitsee tämän käskynä kytkeä komponentti sarjaan diodin kanssa, joten sille joudutaan ohjeistamaan oikea kytkentä manuaalisesti. Myös komponentin sijainti on eri, kuin diodeilla, joten myös tämä joudutaan korjaamaan.

- "Do not connect the RLC branch yet"
- "The correct path for RLC branch is 'sps_lib/Passives/Series RLC Branch'"

Tekoäly muuttaa sille aiemmin ohjeistettuja diodien välisiä yhteyksiä, kun sitä käsketään lisäämään komponentti. Tämän korjaamiseksi ohjeistetaan tekoälyä olemaan tekemättä muutoksia komponenttien välisiin yhteyksiin.

- "Do not change the connections that form the rectifier bridge when adding the RLC branch"

Tekoäly ei onnistu palauttamaan oikeita yhteyksiä, joten palataan koodin versioon, jossa komponenttien väliset yhteydet ovat oikein. Tämän jälkeen muotoillaan käsky uudestaan. Aiempaan versioon voidaan palata antamalla koodi tekoälylle syötteenä. Jatketaan antamalla tekoälylle syötteenä ohjelma, joka rakentaa tasasuuntaajan mallin.

- "Go back to this version"
- – Tasasuuntaajan koodi –
- "Now add the series RLC branch without adding or changing any connections"
- "Do not change the connections"
- "Those are not the connections I provided"

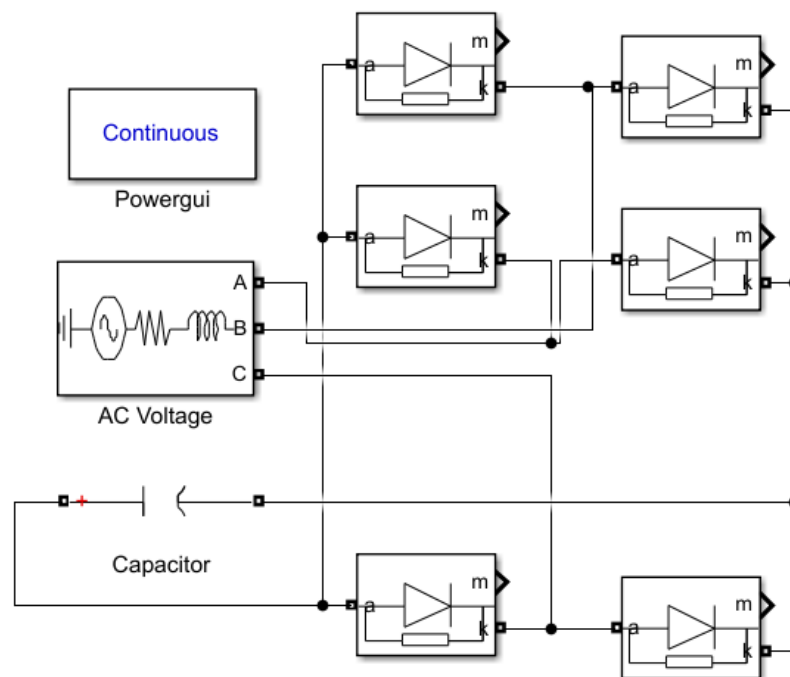
Tekoälyllä on vaikeuksia säilyttää aiemmin oikeaksi vahvistetut komponenttien väliset yhteydet lisätessä uutta komponenttia. Muutaman yrityksen jälkeen tekoälyltä onnistuu lisätä RLC-komponentti malliin säilyttäen aiemmat yhteydet. Tämän jälkeen ohjeistetaan lisättävät yhteydet tekoälylle. Tekoälylle täytyy tarkentaa, että uuteen komponenttiin viittaaminen noudattaa samoja käytäntöjä, kuin muutkin komponentit.

- "Add a connection between right ports of diode 3 and RLC branch"
- "You need to refer to RLC ports the same way as with diodes"
- "The connection is supposed to be between the right ports of diode 3 and RLC branch"
- "Add a connection between the left ports of diode 6 and RLC branch"

Nyt välipiiri on yhdistetty halutulla tavalla. Ohjeistetaan tekoälylle, että komponentin halutaan toimivan kondensaattorina. Asetetaan komponentin tyyppi kondensaattori ja kondensaattorin kapasitanssin arvoksi 5000 μF .

- "Can you change the RLC branch parameters so that it operates as a capacitor?"
- "Can you set the capacitance to be five thousand microfarads?"

Kondensaattorin lisäämisen jälkeen simulaatiomallista löytyy tasasuuntaaja ja DC-välipiiri. Simulaatiomalli on esitetty kuvassa 7.



Kuva 7. Tasasuuntaajan ja DC-välipiirin malli

Myös loppumalli voitaisiin rakentaa samantyyllisesti komponenttitasolla korjaten vaiheittain esiintyvät virheet. Tämän seurauksena on kuitenkin, että ohjelmakoodi venyy pitkäksi, eikä ChatGPT generoi koko ohjelmaa, vaan pelkästään uusimmat lisäykset. Tämä aiheuttaisi myös paljon toistoa dokumentaatioon, koska prosessi vastaa pitkälti tasasuuntaajalle ja välipiirille käytettyä. Jotta koodi ei kasvaisi turhan pitkäksi, käytetään valmiita lohkoja vaihtosuuntaajalle, sekä PWM-signaalin generoinnille.

Valmiiden kirjastosta löytyvien lohkojen hyödyntäminen toimii samalla tavalla kuin komponenttienkin. Aloitetaan ohjeistamalla tekoälyä lisäämään vaihtosuuntaaja malliin ja kerrotaan lohkon sijainti. Mallin selventämiseksi ohjeistetaan vielä tekoälyä kääntämään komponentti.

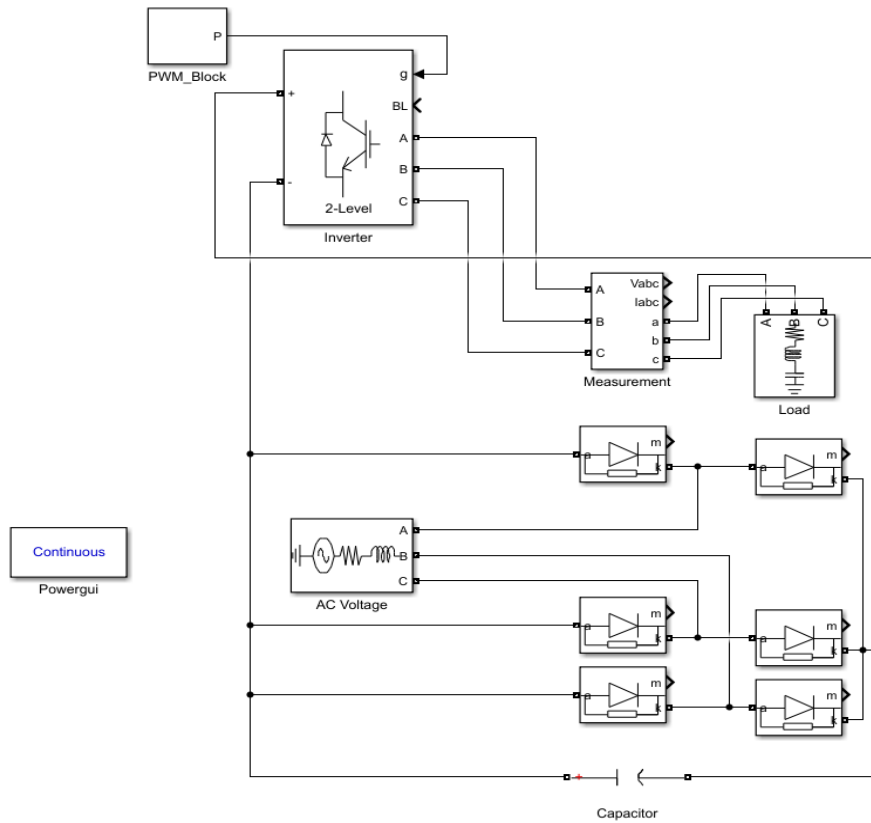
- “Add a 2-level inverter to the model without connecting it. The path for the model is 'sps_lib/Power Electronics/Two-Level Converter”
- “Can you add the inverter orientation value to 'left'?”

Mallin muut lohkot, eli PWM-lohko, mittauslohko ja 3-vaiheinen kuorma voidaan lisätä malliin samalla tavalla, kuin edellä esitetyt lohkot. Tässä työssä käytetty kuorma kuluttaa vain aktiivista tehoa, ja sen parametreinä käytetään oletusarvoja. Kuorman parametrejä, kuten induktiivista- ja kapasitiivista tehoa sekä kytkentäkonfiguraatiota voidaan haluttaessa muuttaa vastaamaan erilaisia kuormia.

Jotkin lohkot, kuten vaihtosuuntaaja, sisältävät erilaisia portteja. Tällöin niihin viitattaessa täytyy kiinnittää huomiota porttien nimiin. Myös lohkojen väliset nimet parametreille voi vaihdella. Esimerkiksi PWM-lohkon parametrissoinnissa joudutaan parametrien nimet kertomaan tekoälylle. 'ModulatingSignals' -parametrillä määritetään, generoidaanko PWM:n ohjaussignaali sisäisesti. Parametri 'Fc' tarkoittaa kantoaallon taajuutta, 'm' modulaatioindeksiä, 'Freq' perusaallon taajuutta ja 'Ts' näytteenottoaikaa.

- “The name of the parameter is 'ModulatingSignals', and the value is on”
- “Set the PWM parameters as follows: Fc to 2000, m to 0.5, Freq to 50 and Ts to 4e-6”
- “Use the parameter names 'Fc', 'm', 'Freq' and 'Ts”

Mallin loppujen lohkojen lisäämistä, toisiinsa yhdistämistä ja parametrintia ei ole dokumentoitu tarkemmin, koska niissä toistuu samat asiat, kuin edeltävissä mallin osissa. Näiden lohkojen lisäämisen jälkeen lopputuloksena on yksinkertainen taajuusmuuttajan pääpiirin simulaatiomalli. Malli on esitetty kuvassa 8.



Kuva 8. Taajuusmuuttajan pääpiirin malli

Mallissa on kaikki tarvittava taajuusmuuttajan pääpiirin simuloimista varten. Tekoälyn tuottama koodi simulaatiomallille on esitetty liitteessä A. Mallin rakentamisen jälkeen voidaan vielä tarkistaa mallin toiminta. Simulaatiomallin verifiointi on esitetty luvussa 4.1.

4. TULOSTEN ESITTELY

Tässä luvussa on esitetty mallin rakentamisen yhteydessä saadut tulokset. Tekoälyn simulaatiomallille tuottaman koodin lisäksi näihin tuloksiin kuuluu simulaatiomallin rakentamisprosessi. Myös erilaiset huomiot liittyen tekoälyn parissa työskentelyyn ovat tärkeitä arvioidessa, miten tekoälyä voidaan hyödyntää.

4.1 Tekoälyn tuottama koodi ja simulaatiomalli

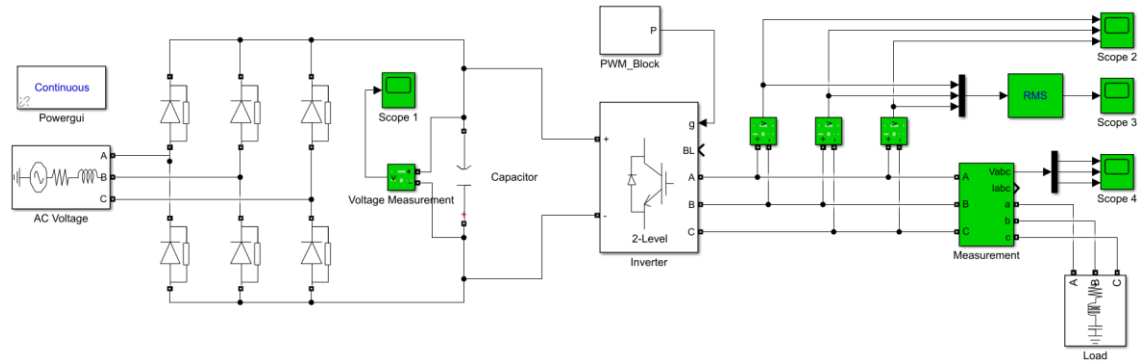
Tekoälyn tuottama koodi on hyvin loogista ja selkolukuista. Koodin rakenne on johdonmukainen, ja tekoäly lisää myös kommentteja, jotka auttavat ymmärtämään koodia. Tekoäly nimeää komponentit ja mallin järkevästi ja usein myös alustaa komponenttien parametrit sopiviin arvoihin. Tekoälyn käyttämä syntaksi on suurilta osin oikein. Lukuun ottamatta muutamaa tulosta, kuten täynnä viivelohkoja olevaa mallia, jo alustavana tuloksena saatavat ohjelmat voivat näyttää hyvinkin vakuuttavilta.

Tämä käsitys kuitenkin muuttuu katsoessa koodia tarkemmin. Alustavan koodin voi nopeasti huomata lisäävän malliin komponentteja, joita ei pitäisi olla mallissa. On myös mahdollista, että tekoäly käyttää valmiita lohkoja, vaikka tavoitteena on komponenttitaso suunnittelu. Usein komponenttien lukumäärät eivät vastaa odotettua, kuten työssä rakennetun tasasuuntaajan tapauksessa. Alustava ohjelma lisäsi tasasuuntaajan malliin vain kolme diodia. Alustavia tuloksia on visualisoitu kuvassa 5.

Ohjelman suorittaminen nostaa myös pintaan ongelmia, joita ei nopealla silmäyksellä välttämättä huomata. Ongelmia ovat esimerkiksi komponenttien sijainnit ja portteihin viittaaminen. Käyttäjän täytyy korjata tekoälyn tekemät antamalla tekoälylle palautetta. Palautteen antaminen ja tekoälyn opettaminen onkin rakennusprosessin tärkein osuus.

Tarvittavien komponenttien ja lohkojen lisäämisen, yhdistämisen, parametrisoinnin jälkeen tekoälyn tuottama koodi rakentaa kuvan 8 mukaisen simulaatiomallin. Mallin toiminta voidaan verifioida Simulink:istä löytyvillä komponenteilla. Näiden komponenttien avulla voidaan muun muassa tarkastella signaalien aaltomuotoja.

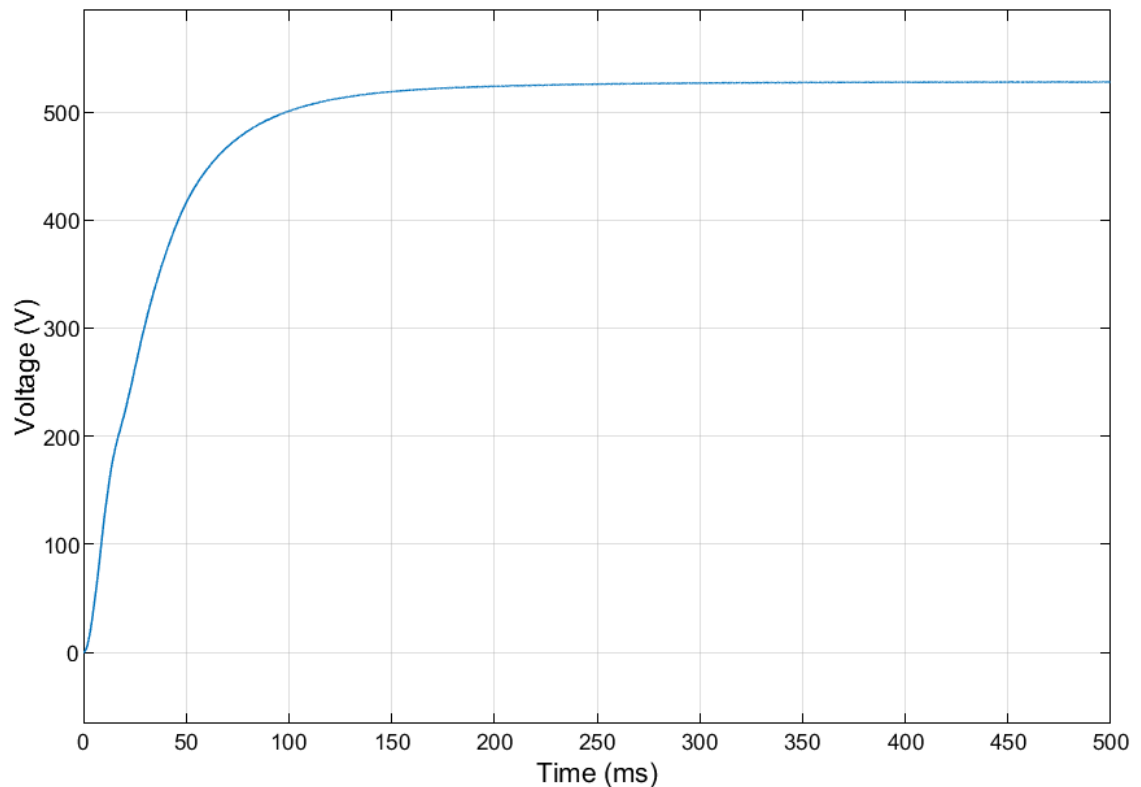
Koska automaattinen komponenttien sijoittelu ei ole ihanteellinen, mallista saa selkeämmän sijoittelemalla komponentit uudestaan. Kuvassa 9 on esitelty malli, jonka komponentit ovat sijoitettu uudelleen, ja johon on lisätty verifiointissa käytettäviä komponentteja. Mallin diodeista on piilotettu mittausportit, koska niitä ei hyödynnetä verifiointissa. Mallin verifiointissa käytettävät komponentit on merkitty vihreällä.



Kuva 9. Uudelleenjärjestelty simulaatiomalli ja mittauslohkot

Vaihtosuuntaajan ja kuorman välillä oleva mittauslohko lisättiin jo mallin rakentamisen yhteydessä. Myös muut verifiointissa käytetyt lohkot olisi voinut lisätä mallia rakentaessa, mutta tämä olisi tuonut turhaa toistoa ohjelmaan.

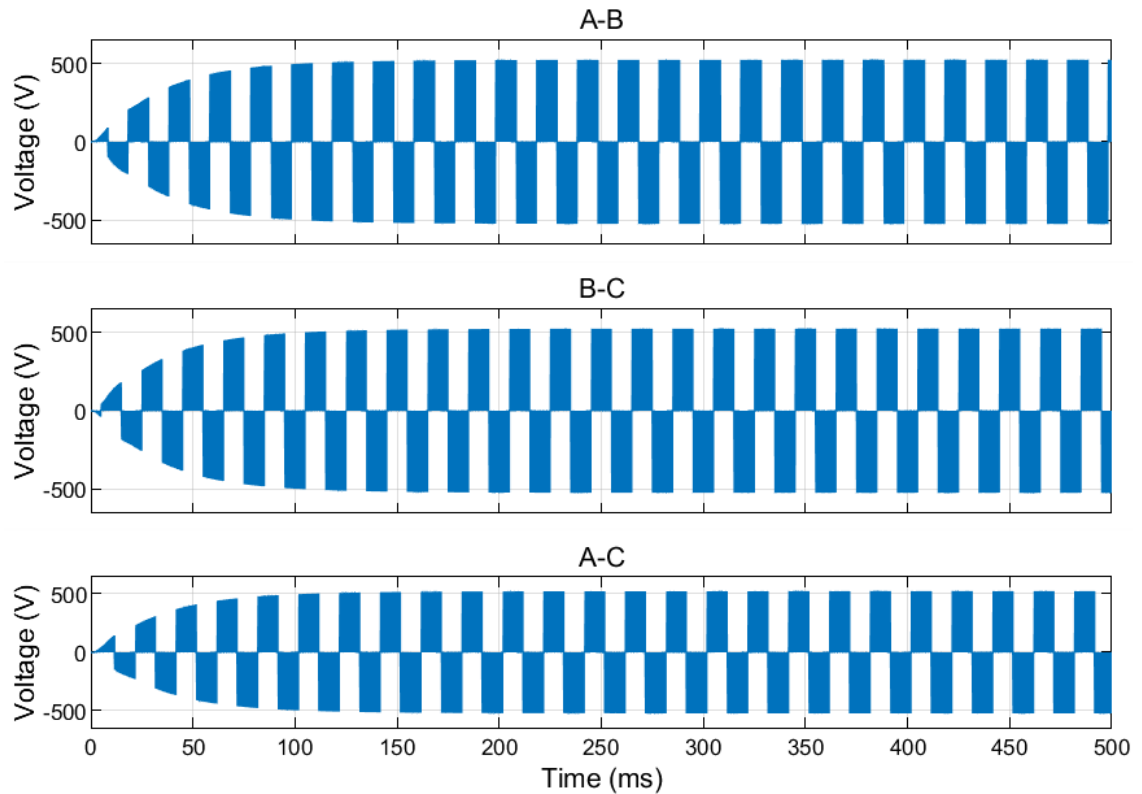
Jännitemittaria ja 'Scope 1'-komponenttia käytetään tasasuuntaajan toiminnan verifiointiin. Jännitemittari mittaa DC-välipiirin yli olevan jännitteen ja syöttää sen 'Scope 1'-komponentille, joka näyttää jännitteen aaltomuodon. Saatu kuvaaja on esitetty kuvassa 10.



Kuva 10. DC-välipiirin jännitteen kuvaaja

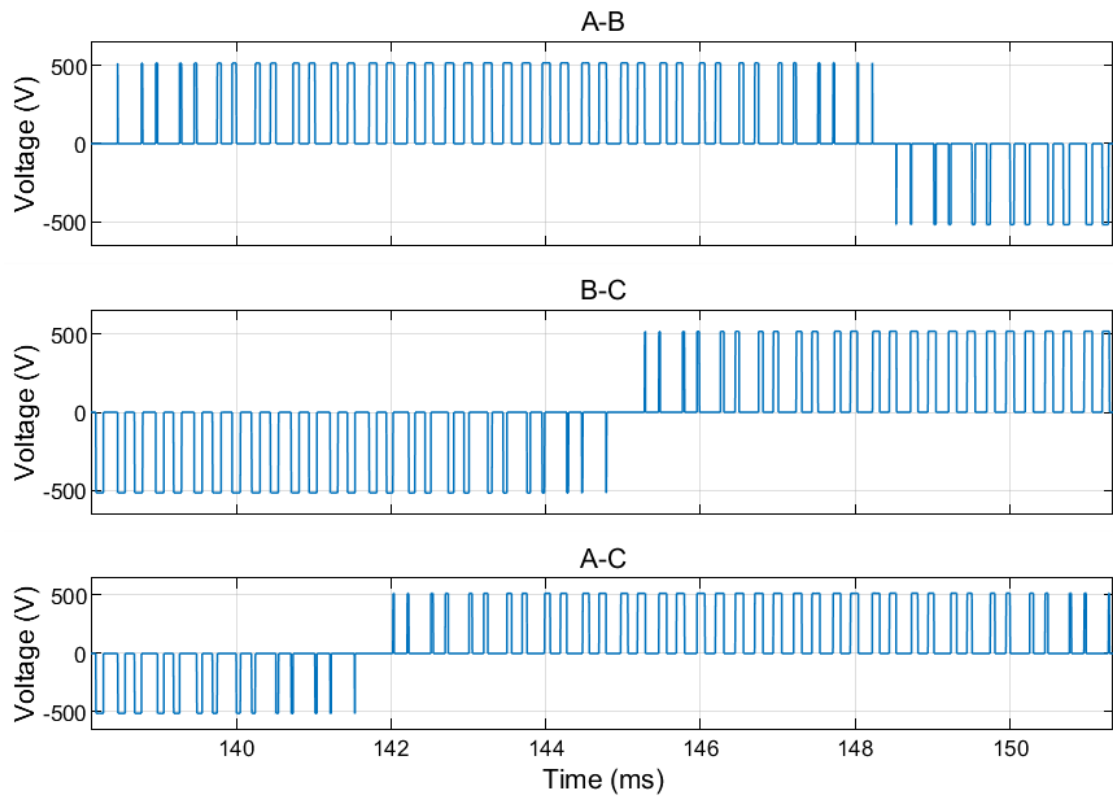
Vaiheiden väliset jännitteet eli pääjännitteet on mitattu käyttäen jännitemittarikomponentteja. Pääjännitteitä sekä niiden tehollisarvojen kuvaajia voidaan

tarkastella 'Scope 2' ja 'Scope 3' komponenttien avulla. Kuormaa edeltävän mittauslohkon avulla voidaan halutessa tarkastella vaihevirtoja ja -jännitteitä maata vasten. Vaihejännitteiden aaltomuodot nähdään 'Scope 4'-komponentin avulla. Pääjännitteiden aaltomuodot ovat esitetty kuvissa 11 ja 12. Pääjännitteiden tehollisarvojen kuvaajat ovat esitetty kuvassa 13. Vaihejännitteiden aaltomuodot on esitetty kuvassa 14.



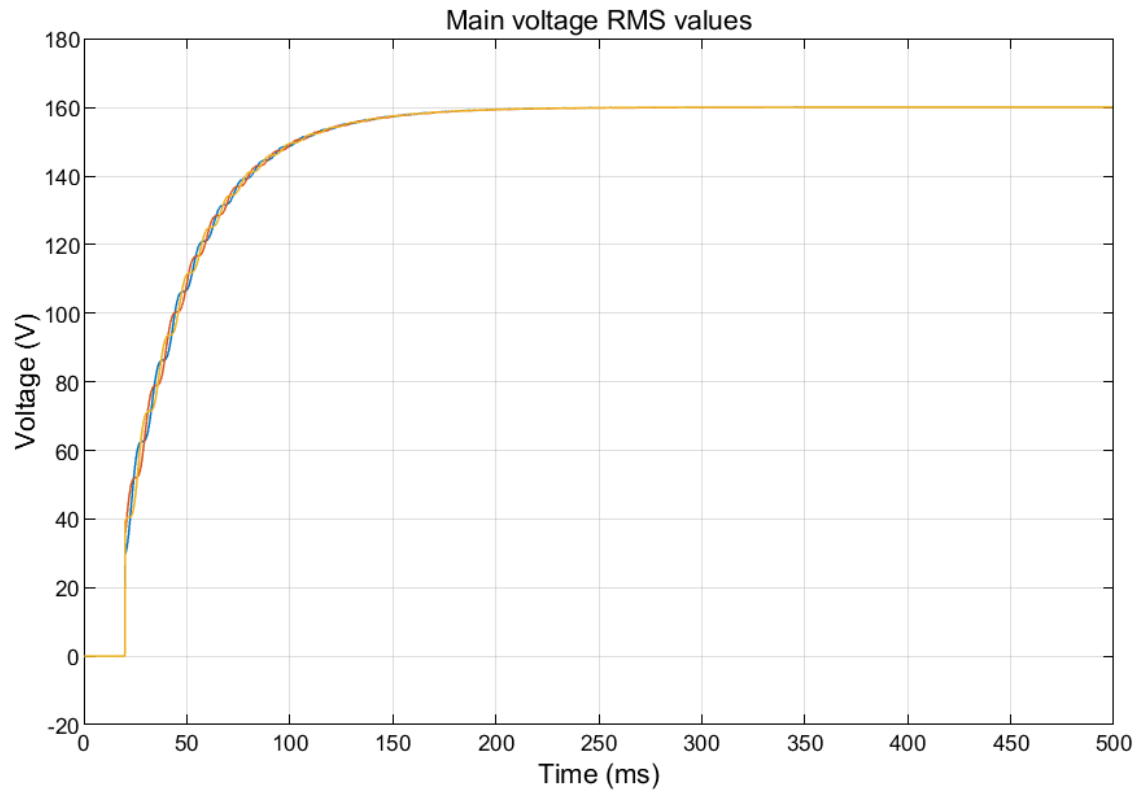
Kuva 11. Vaihtosuuntaajan tuottamien pääjännitteiden aaltomuodot

Kuvassa 11 näkyvät pääjännitteiden aaltomuodot vastaavat tyypillisiä PWM-modulaatiota käyttävän vaihtosuuntaajan tuottamia aaltomuotoja. Tarkentaessa kuvan aaltomuotoon voidaan nähdä paremmin pulssinleveyden vaihtelu. Tarkennus on esitetty kuvassa 12.



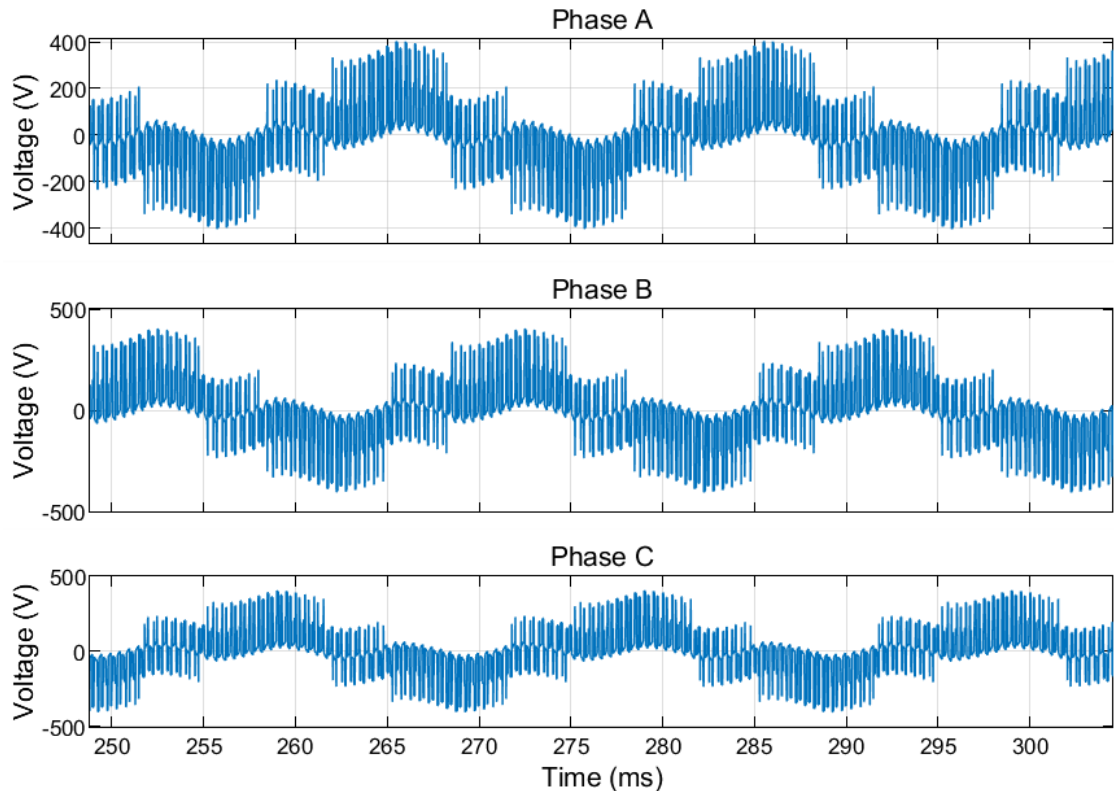
Kuva 12. Tarkennus pääjännitteiden aaltomuotoihin

Pulssien leveys ja polarisuus vaihtelee jaksoittain riippuen PWM-lohkon tuottamasta ohjaussignaalista. Käyttämällä RMS-lohkoa voidaan myös tarkastella pääjännitteiden tehollisarvoja. RMS-lohkosta voidaan valita, että tarkastellaan vain jännitteiden perusaaltoa. Pääjännitteiden perusaaltojen tehollisarvot ovat esitetty kuvassa 13.



Kuva 13. Pääjännitteiden tehollisarvot

Kuvasta 13 nähdään pääjännitteiden tehollisarvojen lähestyvän vakioarvoa. Kuormaa edeltävää mittauslohkoa käytetään mittaamaan vaiheiden ja maan välistä potentiaaliero, eli vaihejännitteitä. Vaihejännitteiden aaltomuodot ovat nopeaan kytkemiseen perustuvalle vaihtosuuntaajalle tyypilliset. Ominaisia piirteitä ovat signaalin piikkimäisyys, sekä selvästi erotettavat jännitetasot. Vaihejännitteiden aaltomuodot ovat esitetty kuvassa 14.



Kuva 14. Vaihejännitteiden aaltomuodot

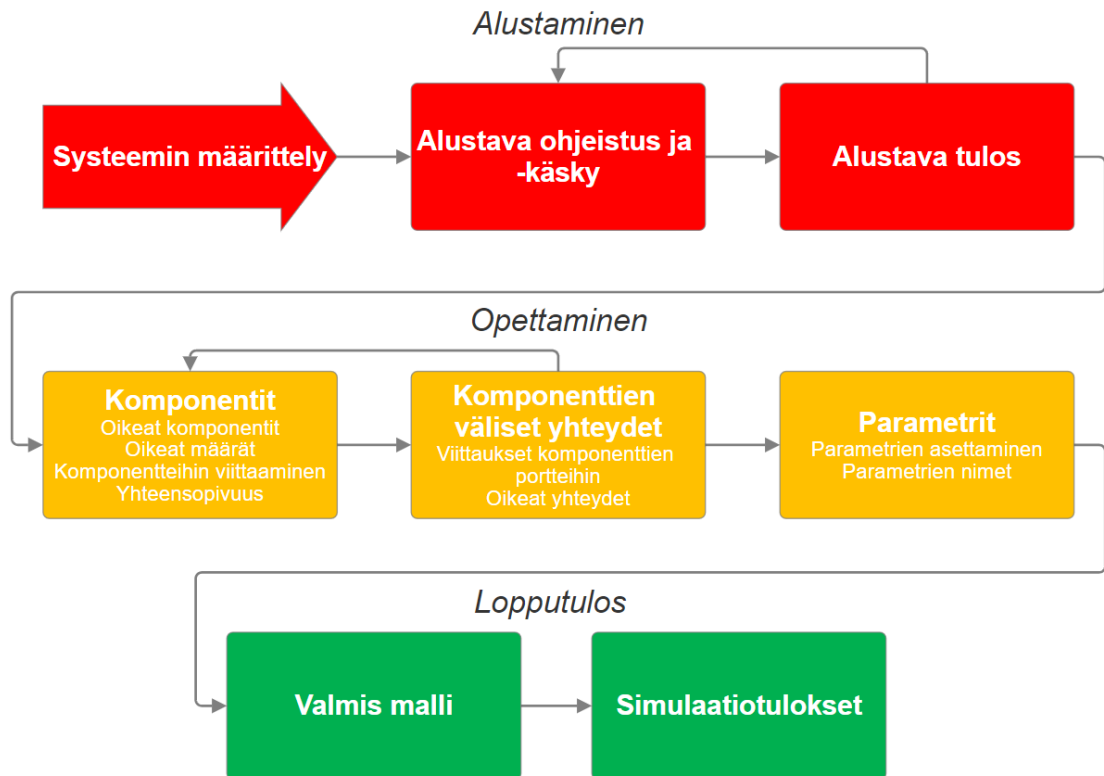
Kuvaajista nähdään signaalien aaltomuotojen olevan halutun kaltaisia ja simulaatiomallin toiminnan voidaan todeta vastaavan työn alussa asetettuja kriteereitä. Vaihtosuuntaajan tuottamiin jännitteiden taajuuksiin sekä amplitudeihin voidaan vaikuttaa muuttamalla mallin parametrejä, kuten PWM-lohkon modulaatioindeksiä. Työn päätavoitteena on kuitenkin esitellä mallin rakennusprosessia, eikä ohjausta käsitellä työssä tarkemmin.

4.2 Mallin rakentamisprosessi

Simulaatiomallin rakentamisprosessin tekoälyllä on todella iteratiivinen prosessi. Tekoälyn rakentamaan malliin joudutaan tekemään paljon korjauksia, ennen kuin saadaan haluttu tulos. Välillä on jopa helpompaa palata aiempaan ohjelman versioon, jos tekoälyn lisäykset ovat huonoja. Mallin rakentaminen tekoälyllä on kaikista huolimatta mahdollista.

Mallin rakentamisen voi ajatella prosessiksi, joka lähtee liikkeelle systeemin määrittelystä, jolle halutaan rakentaa simulaatiomalli. Kun tiedetään systeemi, jolle halutaan rakentaa malli, kerrotaan tekoälylle, millaisia tuloksia halutaan. Tämän jälkeen esitetään alustava käsky ja opetetaan tekoälyä korjaamaan sen tekemät virheet

yksitellen, kunnes saadaan haluttu tulos. Prosessi voidaan kuvata 8-vaiheisella lohkokaaaviolla. Lohkokaavio on esitetty kuvassa 15.



Kuva 15. Rakentamisprosessin lohkokaavio

Lohkokaaviossa on kolme eri tasoa. Ylin taso sisältää punaisella merkityt vaiheet. Nämä vaiheet kattavat tekoälyn- ja rakentamisprosessin alustamisen. Keskimäinen tason voidaan ajatella osuudeksi, jossa tekoälyä opetetaan korjaamaan sen tekemät virheet. Tämän tason vaiheet ovat merkitty keltaisella. Alin taso vastaa haluttua lopputulosta, eli tässä tapauksessa valmista simulaatiomallia ja siitä saatavia simulaatiotuloksia. Prosessi voidaan haluttaessa kiteyttää kolmeen tasoon: alustamiseen, opettamiseen ja lopputuloksiin.

4.2.1 Rakentamisprosessin vaiheet

Prosessin ensimmäinen vaihe on systeemin määrittely. Tässä vaiheessa määritellään systeemi, mille simulaatiomalli halutaan rakentaa. Määrittely asettaa kriteerit laitteessa käytettäville komponenteille ja lohkoille, laitteen parametreille ja sen toiminnalle. Vaihe vastaa pitkälti minkä tahansa suunnitteluprosessin alkuvaihetta.

Toisessa vaiheessa ohjeistetaan tekoälylle, millaisia tuloksia siltä odotetaan. MatGPT:ssä valmis ohjeistus voidaan valita sovelluksen tarjoamista ennakkovalinnoista. ChatGPT:n selainversiossa täytyy tekoälylle täsmentää, että halutaan tuloksena MATLAB-koodia.

Tekoälyn ohjeistamisen jälkeen systeemin määrittelystä muodostetaan alustava käsky tekoäylle. Alustavan käskyn on hyvä olla yleistasoinen. Turhan yksityiskohtaisen käskyn tuottamasta tuloksesta on vaikeaa nähdä, edetäänkö mallin rakentamisessa haluttuun suuntaan. Alustavaan käskyyn on myös hyödyllistä sisällyttää tieto siitä, minkä tason malli halutaan rakentaa. Tämä voi tarkoittaa esimerkiksi sitä, että kerrotaan mitä komponentteja tai lohkoja halutaan mallissa käyttää.

Kun alustava käsky esitetään tekoäylle, saadaan alustava tulos. Kuvan 15 prosessikaaviossa on nuoli molempiin suuntiin alustavan käskyn ja -tuloksen vaiheiden välillä. Tämä mallintaa sitä, että alustavan käskyn tuottama tulos voi olla huono. Tällöin on helpompaa muotoilla käsky uudestaan, kuin yrittää korjata kaikkia virheitä. Pienetkin erot alustavan käskyn muotoilussa voivat vaikuttaa suuresti saatuun tulokseen, kuten mallin rakentamisen yhteydessä huomattiin.

Kun alustava tulos on tarpeeksi hyvä, voidaan aloittaa tekoälyn opettaminen. Tekoälyn opettaminen on ylivoimaisesti rakennusprosessin vaativin osuus. Annettaessa palautetta tekoäylle sen tekemistä virheistä tekoäly muuttaa vastausta palautteen mukaan. On kuitenkin todennäköistä, että tekoäly ei onnistu korjaamaan kaikkia virheitä omin avuin. Tekoäly kykenee oppimaan, mutta ei etsimään uutta tietoa. Tästä johtuen käyttäjä joutuu usein itse etsimään ratkaisun ongelmaan ennen kuin sen voi opettaa tekoäylle. Ratkaisun löytäminen voi olla työlästä, mikä tekee tekoälyn opettamisesta vaativimman osuuden.

Opettaminen voidaan ryhmitellä kolmeen vaiheeseen: komponentteihin-, komponenttien välisiin yhteyksiin- ja parametreihin liittyvään opetukseen. Ensimmäisessä vaiheessa opetetaan tekoälyä käyttämään haluttuja komponentteja. Tekoälyltä onnistuu Simulink:in omien kirjastojen komponenttien lisääminen, mutta jos halutaan käyttää muita kirjastoja, joudutaan tekoäylle opettamaan oikea viittaustyyli kyseisen kirjaston komponentteihin. Tämän lisäksi on tarkastettava, että komponentteja on oikea määrä. Käyttäessä komponentteja eri kirjastoista on myös mahdollista, että komponentit eivät ole yhteensopivia. Yhteensopivuuteen liittyvä ongelma voidaan huomata, jos komponenttien välille ei voida lisätä yhteyksiä. Tällöin on vaihdettava käytettävää komponenttia. Tilannetta kuvastaa opetusvaiheiden välillä oleva vastakkaisuuntainen nuoli kuvassa 15.

Seuraavassa vaiheessa opetetaan tekoäly luomaan halutut yhteydet komponenttien välille. Tämä edellyttää, että tekoäly osaa viitata komponenttien portteihin oikealla tavalla. "Simscape electrical"-kirjaston komponenttien porteissa käytetään erilaista nimeämistapaa, kuin Simulink:in oman kirjaston komponenteissa. Tästä johtuen

tekoälylle pitää opettaa oikea tapa viitata komponenttien portteihin. Kun oikea viittaustyyli on opetettu, voidaan yhteydet lisätä komponenttien välille.

Tekoäly luo yhteyksiä komponenttien välille automaattisesti, mutta kuten mallin rakentamisen yhteydessä huomattiin, usein yhteydet ovat väärä tai puutteellisia. Tekoälylle voidaan antaa palautetta virheellisistä yhteyksistä, jolloin se pyrkii korjaamaan ne. Tekoäly ei kuitenkaan osaa korjata yhteyksiä ilman apua. Helpointa on kertoa manuaalisesti puuttuvat yhteydet.

Kun oikeat yhteydet saadaan lisättyä malliin, voidaan siirtyä tekoälyn opettamisen viimeiseen vaiheeseen, eli parametrien asettamiseen. Mallin sisältäessä kaikki tarvittavat komponentit ja yhteydet täytyy vielä komponenttien parametrit asettaa vastaamaan määriteltyä systeemiä. Ajoittain tekoäly asettaa komponenteille nimen lisäksi muitakin parametrejä automaattisesti, kuten jännitelähteen tapauksessa. Oletusparametrit tai tekoälyn parametrit ovat usein järkeviä, mutta monesti niitä halutaan muuttaa.

Parametrien muuttaminen onnistuu tekoälyltä suhteellisen hyvin. Parametrejä voidaan muuttaa kertomalla tekoälylle parametrin nimi ja parametrille asetettava arvo. Parametrin arvo voidaan myös asettaa komponentin lisäysvaiheessa. Ainoa parametreihin liittyvä ongelma esiintyy, jos tekoäly ei osaa viitata parametriin oikealla nimellä. Näin kävi esimerkiksi jännitelähteen ja PWM-lohkon parametrejä asettaessa. Ongelma korjaantuu, kun täsmennetään tekoälylle, millä nimillä parametreihin täytyy ohjelmassa viitata.

Parametrien asettaminen on tekoälyä opettavan osuuden viimeinen vaihe. Tämän jälkeen lopputuloksena on simulaatiomalli, joka vastaa alussa tehtyä spesifikaatiota. Mallin toiminta kannattaa vielä verifioida, kuten luvussa 4.1 on tehty. Mallin verifiointin jälkeen voidaan mallilla tuottaa halutut simulaatiotulokset.

4.2.2 Työskentelytavat

Mallin rakentaminen eteni pitkälti kokeellisella lähestymistavalla. Erilaiset ohjeistukset ja lähestymistavat toimivat paremmin kuin toiset. Jos vaikuttaa siltä, että mallin rakentaminen etenee väärään suuntaan, voidaan aina palata aiempaan versioon. Toimiva lähestymistapa löytyi pitkälti yrityksen ja erehdyksen kautta.

Samaa lähestymistapaa sovellettiin myös tekoälyn opettamisen parissa. Koska mallin ohjelmalliseen rakentamiseen liittyvää tietoa ei aina ole helppoa löytää, voi olla nopeampaa yrittää manuaalisesti korjata ohjelman sisältämät virheet. Jos ohjelman manuaalisella muokkauksella löydetään ratkaisun ongelmaan, voidaan se opettaa

tekoälylle. Manuaaliseen muokkaukseen kätevä työkalu on esimerkiksi MATLAB:in tarjoama "Live Script"-alusta.

Virheiden selvittämiseen tarvittava tieto joudutaan usein etsimään itse. Tarvittava tieto voidaan löytää esimerkiksi Simulink:in dokumentaatiosta tai itse ohjelmistosta. Simulink-komponentit sisältävät usein tarvittavan tiedon niiden ohjelmalliseen hyödyntämiseen.

4.3 Vaikeudet ja kehityskohteet

Mallin rakentaminen tekoälyllä ei ole suoraviivaista. Luvussa 3 esitetystä mallin rakentamisesta on näytetty, miten tekoälylle voidaan opettaa uusia viittaamistyyplejä ja syntaksia. Edellytyksenä tekoälyn opettamiselle on, että osataan korjata sen tekemät virheet.

Koska simulaatiomallin rakentaminen ohjelmallisesti ei ole yhtä yleistä, kuin graafinen rakentaminen, osaan ongelmista on haastavaa löytää tietoa. Tämä koskee etenkin muiden kuin Simulink:in omien kirjastojen komponentteihin liittyviä käytäntöjä, kuten komponentteihin ja niiden portteihin viittaamista ohjelmallisesti. Viittaustyyliin vaihtelevat kirjastojen välillä. Tekoälylle voidaan opettaa viittaustyyliä, mutta se ei omatoimisesti osaa soveltaa eri kirjastojen komponentteihin liittyviä viittaamistyyplejä.

Kuten luvun alussa mainittiin, tekoälyn tuottama koodi on loogista rakenteeltaan ja sen käyttämä MATLAB-syntaksi on lähes virheetöntä. Tekoälyltä onnistuu Simulink:in omien komponenttien lisääminen malliin ilman opastusta. On siis mahdollista, että tekoäly osaa rakentaa simulaatiomallin systeemille, joka ei hyödynnä ulkoisia kirjastoja.

Tekoäly myös tarjosi vaihtoehtoja sijaiskytkentää diodin mallintamiseen, kun diodin mallin paikantaminen epäonnistui. Jos tarvittavat komponentit voidaan mallintaa sijaiskytkentöjen avulla, tekoäly mahdollisesti osaa rakentaa malleja käyttämättä ulkoisia kirjastoja. Tämä tietenkin lisää mallien monimutkaisuutta ja vaatisi systeemitason suunnittelua. Systeemitason suunnittelu on myös yksi aiheista, jolle tekoälyn soveltamista voidaan tutkia.

ChatGPT tulkitsee sille annetun palautteen ja muuttaa vastausta sen mukaisesti. Tämä ei kuitenkaan tarkoita, että tekoäly ymmärtää täysin sille tarkoitettua kritiikkiä. Esimerkiksi tasasuuntaajan diodien välisten yhteyksien rakentamisessa tekoäly sai osan yhteyksistä tehtyä ilman apua ja osaa ei. Ohjeistaessa tekoälyä lisäämään puuttuvat yhteydet se usein muutti valmiiksi oikein olevia yhteyksiä. Sama ongelma toistui lisättäessä välipiirin kondensaattoria, jolloin tekoäly muutti täysin siihen liittymättömiä yhteyksiä. Tästä johtuen voi olla ajoittain helpompaa palata takaisin ohjelman aiempaan versioon.

Tekoäly tuntui vaiheittain jopa ”jääräpäiseltä”. Esimerkiksi korjatessa ulkoisen kirjaston komponenttien porttien nimeämiseen liittyvän ongelman tekoäly vastasi ymmärtävänsä kritiikin ja ottavansa ohjeistetun tavan käyttöön. Tästä huolimatta tekoäly kuitenkin ei muuttanut mitään monesta huomautuksesta huolimatta. On mahdollista, että data, jonka pohjalta tekoäly on koulutettu, on ristiriidassa sen saaman palautteen kanssa, ja tästä seurauksena tekoäly ei onnistu heti omaksumaan käytäntöä. Neuroverkkojen yhteydessä mainittiin painotuskertoimet, ja edellä mainitun tilanteen voisi myös kuvitella aiheutuvan siitä, että painotuskertoimet painottavat ennalta opetettua dataa käyttäjän palautetta enemmän.

Tekoäly osaa kyllä piirtää tasasuuntaajan tai taajuusmuuttajan lohkokaaavion kysyttäessä, mutta näiden yhteyksien lisääminen ja täydentäminen malliin tuottaa ongelmia. Ongelman voi tulkita siten, että toisin kuin ihminen, tekoäly ei osaa ”visualisoida” mallia, jota se on rakentamassa. Tästä johtuen tekoäly ei osaa tehdä malliin tarvittavia muutoksia koulutetun datan pohjalta.

Mallin rakentaminen sujuvoitui huomattavasti, kun tekoälylle opetettiin tarvittavat käytännöt. Kaikki tekoälylle opetettu tieto kuitenkin rajoittuu käydyn keskustelun sisälle. Käytännössä tämä tarkoittaa, että opetetut asiat tallentuvat kyseiseen keskusteluun, mutta keskustelun ulkopuolella tekoäly ei voi hyödyntää sille kyseisessä keskustelussa opetettua tietoa. Jos mallin rakentamista jatketaan eri keskustelussa tai eri rajapinnan kautta, on käyttäjän opetettava tekoälylle käytännöt uudestaan. Koodi voidaan syöttää tekoälylle analysoitavaksi, jolloin se pyrkii adaptoimaan käytetyn tyylin, mutta tämä ei poista tarvetta opettamiselle.

Opetettujen asioiden rajoittuminen käytyyn keskusteluun tarkoittaa myös, että opetetuista asioista muodostuva tietokanta on yksityinen. Opetettujen asioiden yksityisyys estää muita hyödyntämästä tekoälylle opetettuja asioita. Sovellettavuuden kannalta olisi tärkeää, että opetuksista muodostuva tietokanta voitaisiin tarvittaessa tuoda myös muiden saataville, esimerkiksi organisaation sisällä. Käyttöoikeuksien määrittäminen poistaisi tarpeen toistuvalla opetuksella ja mahdollistaisi tietokannan kehittämisen pidemmälle.

Opetetun tiedon rajoittumisesta aiheutuvan ongelman huomasi etenkin MatGPT:tä käyttäessä. Ideana MatGPT on erittäin kätevä työkalu simulaatiomallien rakentamiseen tekoälyn avulla. MatGPT:ssä on ongelmana kuitenkin se, että keskustelun kasvaessa pitkäksi MatGPT tulostaa virhekoodin. Ongelma voidaan ohittaa luomalla uusi keskustelu, mutta aiemmat opetukset eivät siirry uuteen keskusteluun. Tästä johtuen mallin rakentamisessa siirryttiin käyttämään ChatGPT:n selainversiota komponenttien

välisen yhteyksien rakentamisessa. Yhteyksiä korjatessa keskustelun pituus kasvoi nopeasti ja MatGPT:llä monien yksittäisten virheiden korjaaminen ei ollut käytännöllistä. MatGPT:llä kestää myös kauemmin generoida vastaus kuin ChatGPT:n selainversiolla.

5. JOHTOPÄÄTÖKSET

Simulaatiomallin rakentaminen tekoälyllä on mahdollista, mutta ei käytännöllistä tällä hetkellä. Simulaatiomallin rakentaminen graafisesti on paljon helpompaa ja nopeampaa kuin sen rakentamisen ohjeistaminen tekoälylle. Tekoälyn tekemien virheiden korjaamiseen kuuluu huomattavasti aikaa, etenkin jos tietoa virheeseen liittyen on vaikea löytää. On kuitenkin tärkeää huomioida tekoälyn tärkein ominaisuus, eli kyky oppia.

Tekoäly analysoi saamaansa palautetta ja oppii korjauksista, jotka se saa käyttäjältä. Tekoälyn oppimista asioista muodostuu tietokanta, jota tekoäly muovaa jatkuvasti vastaamaan paremmin käyttäjän tarpeita. Jos saman keskustelun sisällä lähtee rakentamaan toista samankaltaista mallia, niin rakentaminen on jo helpompaa. Mitä enemmän malleja rakennetaan, sitä pidemmälle tietokanta kehittyy.

Ilmiö pystytään havaitsemaan jo ensimmäistä mallia rakentaessa. Tasasuuntaajan valmistuttua tekoälylle on jo opetettu suurin osa tarvittavista käytännöistä, joten loppumallin suunnittelu sujuu nopeammin. Jos tietokannan käyttöoikeuden ja -ympäristön pystyisi määrittelemään laajemmaksi, tekoälystä voisi nopeasti kehittyä tehokas apuväline simulaatiomallien rakentamiseen. Kuitenkin niin kauan kuin määrittäminen ei ole mahdollista, tekoälyn käytännöllisyys mallien rakentamisessa pysyy rajallisena.

5.1 Tavoitteisiin pääsy

Työn tavoitteena oli rakentaa toimiva Simulink-malli yksinkertaiselle taajuusmuuttajan pääpiirille käyttäen tekoälyä. Tavoitteen taustalla oli motivaationa kartoittaa millainen simulaatiomallin rakentamisprosessi voisi olla, ja myös muodostaa käsitys tekoälyn tämänhetkisistä sovellusmahdollisuuksista mallien rakentamisessa.

Lähtökohtaisena ideana oli rakentaa taajuusmuuttajan simulaatiomalli moduulikohtaisesti, mutta tasasuuntaajan komponenttitasoisen mallin rakentamisen jälkeen muiden moduulien samantyylinen rakentaminen olisi tuonut paljon tarpeetonta toistoa prosessin dokumentointiin. Tästä syystä vaihtosuuntaajalle ja ohjausmoduulille käytetään simulaatiomallissa valmiita lohkoja.

Tekoälyllä simulaatiomallin rakentaminen onnistuu vaikeuksista huolimatta. Simulaatiomallin asettelu ei ole ideaalinen, mutta tämä on seuraus MATLAB:in

tarjoamasta automaattisesta sijoittelusta, eikä käytetystä tekoälystä. Tekoälyn rakentama simulaatiomalli toimii, kuten oletettu. Mallin verifiointi on esitetty luvussa 4.1.

Simulaatiomallin rakentamisesta saadaan hyvin hahmotettua eri vaiheet mahdolliselle rakennusprosessille tekoälyä käyttäen. Prosessin vaiheet voidaan ryhmitellä kolmeen osuuteen: alustukseen, opettamiseen ja lopputulokseen. Rakentamisprosessi on esitetty luvussa 4.2.

Simulaatiomallin rakentava koodi on täysin tekoälyn generoimaa. Myös prosessi saatiin kuvailtua ja eroteltua selkeisiin kokonaisuuksiin. Kokonaisuudessa voidaan siis sanoa, että tutkielman ensisijaiset tavoitteet saavutettiin.

5.2 Tilannearvio

Simulaatiomallin rakentaminen käyttäen tekoälyä ei ole tällä hetkellä käytännöllistä. Syynä tähän johtopäätökseen on se, että manuaalisesti toteutettu graafinen simulaatiomallin rakentaminen on paljon nopeampaa ja helpompaa ja johtaa parempaan lopputulokseen. Tekoälyä koskevat rajoitteet, kuten kyvyttömyys tiedonhakuun vaikuttavat negatiivisesti tekoälyn mahdollisuuksiin rakentaa malleja.

Työn tulosten perusteella voidaan sanoa tekoälyn opettamisen simulaatiomallien rakentamiseen olevan työlästä ja onnistuvan harvoin ongelmitta. Koska simulaatiomallien ohjelmallinen rakentaminen ei ole kovin yleistä, on todennäköistä, että käyttäjä joutuu ensin itse opettelemaan tarvittavat käytännöt, ennen kuin ne voidaan opettaa tekoälylle. Mallin ohjelmallisen rakentamisen harvinaisuudesta johtuen, siihen liittyvää tietoa on ajoittain vaikeaa löytää.

Tämänhetkisestä tilanteesta huolimatta tekoäly kehittyy jatkuvasti. Jos tekoälyn kehityksen tahti pysyy viime vuosien kaltaisena, voidaan olettaa, että kehitys näkyy myös simulaatiomallien rakentamisessa. Mallin rakentaminen tekoälyllä onnistuu jo nyt, ja työn kirjoittamishetkellä ChatGPT:stä on julkaistu uudempi versio, GPT-4. ChatGPT:n uudempi versio luultavasti selviytyisi tehtävästä vielä edeltäjäänsä paremmin.

LÄHTEET

- [1] J. Kolari, *Tekoäly 123: matkaopas tulevaisuuteen*. Jyväskylä: Docendo, 2023.
- [2] M. Moor *ym.*, "Foundation models for generalist medical artificial intelligence", *Nature (London)*, vsk. 616, nro 7956, ss. 259–265, 2023, doi: [10.1038/s41586-023-05881-4](https://doi.org/10.1038/s41586-023-05881-4).
- [3] S. Y. Liaw *ym.*, "Artificial Intelligence Versus Human-Controlled Doctor in Virtual Reality Simulation for Sepsis Team Training: Randomized Controlled Study", *Journal of medical Internet research*, vsk. 25, ss. e47748–e47748, 2023, doi: [10.2196/47748](https://doi.org/10.2196/47748).
- [4] D. Kaur, S. Uslu, K. J. Rittichier, ja A. Durresi, "Trustworthy Artificial Intelligence: A Review", *ACM computing surveys*, vsk. 55, nro 2, ss. 1–38, 2023, doi: 10.1145/3491209.
- [5] H. Jaakkola, B. Thalheim, ja J. Henno, "About the Essence of Intelligence – Will Artificial Intelligence (Ever) Cover Human Intelligence?", teoksessa *Frontiers in Artificial Intelligence and Applications*, M. Tropmann-Frick, H. Jaakkola, B. Thalheim, Y. Kiyoki, ja N. Yoshida, Toim., IOS Press, 2022. doi: 10.3233/FAIA210475.
- [6] J. McCarthy, M. L. Minsky, N. Rochester, ja C. E. Shannon, "A Proposal for the Dartmouth Summer Research Project on Artificial Intelligence, August 31, 1955", *AI Magazine*, vsk. 27, nro 4, Art. nro 4, joulu 2006, doi: [10.1609/aimag.v27i4.1904](https://doi.org/10.1609/aimag.v27i4.1904).
- [7] F. Emmert-Streib, O. Yli-Harja, ja M. Dehmer, "Artificial Intelligence: A Clarification of Misconceptions, Myths and Desired Status", *Frontiers in artificial intelligence*, vsk. 3, ss. 524339–524339, 2020, doi: 10.3389/frai.2020.524339.
- [8] V. Alto, *Modern Generative AI with ChatGPT and OpenAI Models: Leverage the Capabilities of OpenAI's LLM for Productivity and Innovation with GPT3 and GPT4.*, 1st ed. Birmingham: Packt Publishing, Limited, 2023.
- [9] "What are Neural Networks? | IBM". Viitattu: 5. lokakuuta 2023. [Verkossa]. Saatavissa: <https://www.ibm.com/topics/neural-networks>
- [10] M. Salvagno, F. S. Taccone, ja A. G. Gerli, "Can artificial intelligence help for scientific writing?", *Critical care (London, England)*, vsk. 27, nro 1, ss. 75–75, 2023, doi: [10.1186/s13054-023-04380-2](https://doi.org/10.1186/s13054-023-04380-2).
- [11] J. White *ym.*, "A Prompt Pattern Catalog to Enhance Prompt Engineering with ChatGPT". arXiv, 21. helmikuuta 2023. doi: 10.48550/arXiv.2302.11382.
- [12] S. Monteith, T. Glenn, J. R. Geddes, P. C. Whybrow, E. Achtyes, ja M. Bauer, "Artificial intelligence and increasing misinformation", *British journal of psychiatry*, ss. 1–3, 2023, doi: 10.1192/bjp.2023.136.
- [13] E. A. M. van Dis, J. Bollen, W. Zuidema, R. van Rooij, ja C. L. Bockting, "ChatGPT: five priorities for research", *Nature (London)*, vsk. 614, nro 7947, ss. 224–226, 2023, doi: 10.1038/d41586-023-00288-7.

- [14] "Mikä on taajuusmuuttaja? | Danfoss". Viitattu: 5. lokakuuta 2023. [Verkossa]. Saatavissa: <https://www.danfoss.com/fi-fi/about-danfoss/our-businesses/drives/what-is-a-variable-frequency-drive/>
- [15] "Facts Worth Knowing about AC Drives | Danfoss". Viitattu: 5. lokakuuta 2023. [Verkossa]. Saatavissa: <https://assets.danfoss.com/documents/latest/242341/AV446558536912en-000101.pdf#/>
- [16] "MATLAB". Viitattu: 6. lokakuuta 2023. [Verkossa]. Saatavissa: <https://se.mathworks.com/products/matlab.html>
- [17] "Simulink Getting Started Guide". Viitattu: 6. lokakuuta 2023. [Verkossa]. Saatavissa: https://se.mathworks.com/help/pdf_doc/simulink/simulink_gs.pdf
- [18] "MatGPT". Viitattu: 10. lokakuuta 2023. [Verkossa]. Saatavissa: <https://se.mathworks.com/matlabcentral/fileexchange/126665-matgpt>
- [19] L. Giray, "Prompt Engineering with ChatGPT: A Guide for Academic Writers", *Annals of biomedical engineering*, vsk. 51, nro 12, ss. 2629–2633, 2023, doi: 10.1007/s10439-023-03272-4.

LIITE A: CHATGPT:N TUOTTAMA KOODI TAAJUUSMUUTTAJASIMULOINTIIN

```

model = 'three_phase_rectifier_with_Inverter';
open_system(new_system(model));

% Add three-phase AC voltage source with specified parameters
add_block('sps_lib/Sources/Three-Phase Source', [model '/AC Voltage'],
'Voltage', '400', 'Frequency', '50');

% Add six diodes for rectification
for i = 1:6
    add_block('sps_lib/Power Electronics/Diode', [model '/Diode'
num2str(i)]);
end

% Connect diodes to form a three-phase rectifier bridge
for phase = 1:3
    add_line(model, ['AC Voltage/RConn' num2str(phase)], ['Diode'
num2str(phase) '/LConn1']);
    add_line(model, ['Diode' num2str(phase + 3) '/RConn1'], ['AC Volt-
age/RConn' num2str(phase)]);
end

% Add connections between the right ports of Diode 1 and Diode 2
add_line(model, 'Diode1/RConn1', 'Diode2/RConn1');
% Add a connection between the right ports of Diode 2 and Diode 3
add_line(model, 'Diode2/RConn1', 'Diode3/RConn1');
% Add a connection between the left ports of Diode 4 and Diode 5
add_line(model, 'Diode4/LConn1', 'Diode5/LConn1');
% Add a connection between the left ports of Diode 5 and Diode 6
add_line(model, 'Diode5/LConn1', 'Diode6/LConn1');

% Add series RLC branch configured as a capacitor without connecting
add_block('sps_lib/Passives/Series RLC Branch', [model '/Capacitor']);
set_param([model '/Capacitor'], 'Resistance', '0', 'Inductance', '0',
'Capacitance', '5000e-6', 'BranchType', 'C');

% Add connection between right port of Diode 3 and RLC branch
add_line(model, 'Diode3/RConn1', 'Capacitor/RConn1');
% Add connection between left port of Diode 6 and RLC branch
add_line(model, 'Diode6/LConn1', 'Capacitor/LConn1');

% Add 2-level inverter without connecting
add_block('sps_lib/Power Electronics/Two-Level Converter', [model '/In-
verter'], 'Orientation', 'left');

```

```

% Add connection between right port of Capacitor and right port 1 of In-
verter
add_line(model, 'Capacitor/RConn1', 'Inverter/RConn1');
% Add connection between left port of Capacitor and right port 2 of In-
verter
add_line(model, 'Capacitor/LConn1', 'Inverter/RConn2');

% Add three-phase load
add_block('sps_lib/Passives/Three-Phase Series RLC Load', [model
'/Load']);

% Add measurement block
add_block('sps_lib/Sensors and Measurements/Three-Phase V-I Measurement',
[model '/Measurement']);

% Connect left port 1 of Inverter to left port 1 of Measurement block
add_line(model, 'Inverter/LConn1', 'Measurement/LConn1');
% Connect left port 2 of Inverter to left port 2 of Measurement block
add_line(model, 'Inverter/LConn2', 'Measurement/LConn2');
% Connect left port 3 of Inverter to left port 3 of Measurement block
add_line(model, 'Inverter/LConn3', 'Measurement/LConn3');
% Connect right port 1 of Measurement block to left port 1 of Load
add_line(model, 'Measurement/RConn1', 'Load/LConn1');
% Connect right port 2 of Measurement block to left port 2 of Load
add_line(model, 'Measurement/RConn2', 'Load/LConn2');
% Connect right port 3 of Measurement block to left port 3 of Load
add_line(model, 'Measurement/RConn3', 'Load/LConn3');

% Add PWM block
add_block('sps_lib/Power Electronics/Power Electronics Control/PWM Gener-
ator (2-Level)', [model '/PWM_Block']);

% Connect right port of PWM block to port 1 of Inverter
add_line(model, 'PWM_Block/1', 'Inverter/1');

% Set load parameters
set_param([model '/Load'], 'NominalVoltage', '400', 'NominalFrequency',
'50');

% Set PWM block to generate reference signal internally
set_param([model '/PWM_Block'], 'ModulatingSignals', 'on');

% Set PWM block parameters
set_param([model '/PWM_Block'], 'Fc', '2000', 'm', '0.5', 'Freq', '50',
'Ts', '4e-6');

```

```
% Add Powergui block
add_block('sps_lib/powergui', [model '/Powergui']);

% Arrange blocks
Simulink.BlockDiagram.arrangeSystem(model);

% Save the model
save_system(model);
```