Otto Ellonen

STACK-järjestelmän hyödyntäminen Vaasan yliopiston piirianalyysin opetuksessa

Tekniikan ja innovaatio johtamisen yksikkö
Sähkötekniikan Diplomityö
Energia- ja informaatiotekniikka, DI

Vaasa 2021

VAASAN YLIOPISTO
Tekniikan ja innovaatiojohtamisen yksikkö
Tekijä: Otto Ellonen
Tutkielman nimi: STACK-järjestelmän hyödyntäminen Vaasan yliopiston piirianalyysin opetuksessa
Tutkinto: Diplomi-insinööri
Oppiaine: Sähkötekniikka
Työn valvoja: Professori Timo Vekara
Työn ohjaaja: DI Maarit Vesapuisto
Valmistumisvuosi: 2021 **Sivumäärä:** 118

TIIVISTELMÄ:
Tämän tutkimuksen tavoitteena on kartoittaa STACK-järjestelmän (System for Teaching and Assessment using a Computer Algebra Kernel) ominaisuuksien hyödyntämistä ja sen kehitystä osana sähkötekniikan opetusta Vaasan yliopistossa. Tutkimus perustuu opetus- ja kulttuuriministeriön rahoittaman ÄlyOppi-kärkihankkeen aikana tuotetun materiaalin kehittämiseen ja testaamiseen Vaasan yliopiston piirianalyysin opetuksessa. Automaattisesti tarkastettavien STACK-tehtävien tarkoituksena on kaventaa luennoilla käydyn teorian ja laskuharjoitusten välistä kuilua tarjoamalla opiskelijalle laskuharjoituksista yksinkertaistettuja versioita, joita hän pystyy suorittamaan itsenäisesti ja interaktiivisesti selaimen välityksellä.

STACK-tehtävissä hyödynnettiin avoimesti saatavissa ollutta JSXGraph-kirjastoa piirikaavioiden luonnissa, ja sen avulla voitiin vaivattomasti kehittää myös useita satunnaisesti arvottuja versioita piirikaavioista. Jokaisen suorituskerran alussa opiskelijoille arvottiin uusi versio tehtävästä, joten he saivat siis eri suorituskerroilla eteensä uudenlaisen ongelman samankaltaisesta piiristä. JSXGraph-kirjaston käyttö mahdollisti perinteisistä tehtävistä poikkeavien ns. dynaamisten tehtävien luonnin, jossa opiskelijan antama vastaus on sidottu sähköisen komponentin sijaintiin piirikaaviossa tai vastaus annettiin puhtaasti kuvan avulla. Uudenlaisissa tehtävätyypeissä opiskelijan ei siis tarvinnut kirjoittaa yhtään mitään, vaan hän antoi vastauksensa komponentteja tai liukusäätimiä liikuttelemalla. Tehtävien ratkaiseminen poikkesikin merkittävästi perinteisistä tehtävistä ja se alkoi muistuttaa pelin pelaamista.

Työssä kehitettyjä STACK-tehtäviä ehdittiin jo testaamaankin piirianalyysin kursseilla ja niihin pohjautuen kyselyllä ja järjestelmään tallennettujen tietojen avulla pyrittiin kartoittamaan ja analysoimaan tehtävien hyödyllisyyttä ja toimivuutta.

Tutkimuksen tuloksena havaitaan, että opiskelijoiden suhtautuminen työssä kehitettyihin STACK-tehtäviin oli pääsääntöisesti myönteistä ja heidän toivomuksenaan oli saada enemmän STACK-tehtäviä. Palautteesta voidaan tulkita, että tehtävät ovat auttaneet vastanneita opiskelijoita sisäistämään luennoilla käydyt asiat entistä paremmin. Lisäksi niiden avulla opiskelija on voinut kehittää omaa, oppimisen kannalta tärkeää laskurutiiniaan suorittamalla uudestaan perustehtäviä. Kehitettävääkin kuitenkin vielä löytyy esimerkiksi tehtövien ohjeistuksen ja esillepanon osalta.

AVAINSANAT: STACK-järjestelmä, piirianalyysi, opetus, interaktiivisuus, JSXGraph, visuaalisuus, automaattinen tarkistus, satunnaisuus, arpominen

ALKULAUSE

Tämä diplomityö sisältää minun ja Välkky-projektitiimin tuotoksia viimeisen parin vuoden ajalta. Projektin alkaessa tietämyksemme STACK-järjestelmästä ja miten sitä pysytyisi hyödyntämään piirianalyysissä oli aika suppea. Tähän saatiin kuitenkin apua yliopistonlehtori Matti Laaksoselta, joka oli käyttänyt järjestelmää omilla kursseillaan. Häneltä saatiin korvaamatonta tietoa, joka mahdollisti alkuun pääsemisen.

Alun ihmettelemisen jälkeen alkoi syntyä tuloksia ja opimme hyödyntämään JSXGraph-kirjastoa piirikaavioiden kanssa. Saimme myös aikaiseksi järjestelmän, joka luo tehtävästä aina erilaisen version eri suorituskerralle. Ensimmäiseksi arvottiin vain jännitenuolien suunnat. Myöhemmin piirikaavioissa alettiin myös arpomaan erilaisia komponentteja, jotta tilanteita tuli vielä enemmän. Lopuksi kehitettiin vielä pari tehtävää, jossa opiskelija antaa vastauksensa liikuttelemalla komponentteja oikeille paikoille piirissä ja antaa vastauksensa sitä kautta. Lähtökohdat huomioon ottaen olemme edenneet näinkin pienellä ryhmällä todella pitkälle, mutta tekemistä ja kehittämistä liittyen STACK-järjestelmään löytyy vielä runsaasti.

Kiitän erityisesti Välkky-ryhmämme jäseniä professori Timo Vekaraa, Maarit Vesapuistoa ja Matti Laaksosta, joiden kanssa on ollut ilo työskennellä projektin aikana. Olen oppinut heiltä paljon ja heidän ansiosta työn tekeminen on ollut äärimmäisen mielenkiintoista ja mielekästä. Lisäksi haluan kiittää myös perhettäni ja lähipiiriäni, jotka ovat olleet tukenani opiskelujeni ja työn aikana. Lisäksi osoitan vielä kiitokset Valdu ry:lle (rekistöröimätön yhdistys), jota ilman en olisi tässä tilanteessa.

Vaasa 19.5.2021

Sisällys

Kuvat

Taulukot

Algoritmit

Symbolit

E_x	Jännitelähde
$\vec{E}_{\rho_s}$	Tasovarauksen aiheuttama sähkökentän voimakkuus
$\vec{E}_{\rho_{ly}}$	Viivavarauksen aiheuttaman sähkökentän voimakkuuden y-komponentti
P	Pätöteho
Q	Loisteho
S	Näennäisteho
U_x	Komponentin jännite, *x*-alaindeksin tilalle tarkennetaan, mistä komponentista kyse.
φ_x	Vaihekulma, *x*-alaindeksin tilalle tarkennetaan, mistä komponentista kyse.

Lyhenteet

CAS	Computer Algebra System, matemaattisten yhtälöiden kirjoittamiseen tarkoitettu ohjelmisto
HTML	Hypertext Markup Language, hypertekstin merkintäkieli
LaTeX	Ladontajärjestelmä
MIT	Massachusetts Institute of Technology, Massachusettsin teknillinen korkeakoulu
MOOC	Massive Open Online Course, massiivinen avoin verkkokurssi
Moodle	Modular Object-Oriented Dynamic Learning Environment, avoimen lähdekoodin virtuaalinen oppimisympäristö
STACK	System for Teaching and Assessment using a Computer Algebra Kernel, tietokonealgebra-ohjelmisto opetukseen ja arviointiin
TIM	The Interactive Media, interaktiivinen media
ÄlyOppi	Älykkäät Oppimisympäristöt, OKM:n kärkihanke 2018-2021

Keskeiset käsitteet

Dynaamisuus	Muuttuva ympäristö, diplomityössä tällä tarkoitetaan muuttuvaa tehtävää esim. satunnaisuuden kautta
Interaktiivisuus	Opiskelijan ja koneen välinen vuorovaikutus esim. verkossa sijaitsevaa opiskelumateriaalia selatessa
Piirianalyysi	Sähköisten virtapiirien toimintaa kartoittava prosessi
Satunnaisuus	Diplomityössä esitettyjen tehtävien riippumattomuus suorituskerrasta, eli opiskelijoille annetaan erilainen versio jokaisella suorituskerralla
Visuaalisuus	Jonkin aiheen tai ongelman selittäminen kuvien tai piirrosten avulla

1 Johdanto

Tekniikan alat pohjautuvat luonnontieteisiin ja matematiikkaan. Ilman tarpeeksi hyvää matemaattista pohjaa aloilla tarvittavien suunnittelutaitojen opiskelu on vaivalloista ja erittäin työlästä ja joissakin tapauksissa jopa mahdotonta. Tämä saattaa näkyä esimerkiksi korkeakouluopiskelijoiden kohdalla epäonnistuneiden suoritusten takia pidentyneenä opiskeluaikana. Heikon laskutaitopohjan omaavalla yksilöllä voi olla vaikeuksia pysyä mukana opetuksen tahdissa, mikä osaltaan luo stressiä ja epätoivoa. Tämän seurauksena motivaatio voi heikentyä ja opiskelija ”putoaa kyydistä” ja sen seurauksena pyrkiikin suorittamaan kurssin vasta seuraavana vuonna. Heikolla matematiikan osaamisella on myös valtava vaikutus joillakin teknillisillä aloilla työelämässä, jossa tarvitaan huippuosaajia.

Sähkötekniikka yhdistää ennen muuta matematiikan, fysiikan ja niiden soveltamiskohteet yhteen kokonaisuuteen, jotka tukevat toisiaan. Mikäli opiskelijalla jotkin perusteiden osaamiset eivät ole kunnossa, katoaa myös muiden osa-alueiden ymmärtäminen. On vaikeaa ymmärtää esimerkiksi sähkö- ja magneettikenttien toimintaa ja olemassaoloa ilman teoreettista pohjaa. Ilmiöiden selittäminen matematiikan avulla on hankalaa, mutta ei mahdotonta, kunhan perusteet ovat hallussa. Samalla lailla sovelluskohteiden, kuten sähköverkkojen ja niihin liittyvien laitteiden, toimintaa ei voi ymmärtää tai oppia, mikäli opiskelija ei ole sisäistänyt sähkötekniikan perusteita.

Perinteisten opetusmetodien, kuten fyysisten ja vuorovaikutteisten luentojen tilalle ja rinnalle on ilmestynyt joukko uusia tapoja vaikuttaa opiskelijoiden oppimiseen. Tekniikan kehittyminen on mahdollistanut esimerkiksi sen, että nykyään lähes kaikki oppimateriaali löytyy netistä Moodlen kaltaisista virtuaalisista oppimisympäristöistä. Näissä opettajalla on mahdollisuus jakaa luentokalvoja, animaatioita tai vaikkapa tallennettuja opetusvideoita, joita ei välttämättä aiemmin hyödynnetty. Materiaalin siirtyminen nettiin on myös mahdollistanut joissain oppiaineissa sen, että kurssi voidaan käytännössä suorittaa kotisohvalta käsin ja lähestulkoon itsenäisesti omaan tahtiin, jopa ilman kontaktia muihin opiskelijoihin tai opettajaan.

Vuoden 2020 keväällä Covid-19-pandemia puhkesi maailmanlaajuisesti. Sen vaikutukset näkyivät heti myös Suomessa, kun suuri osa korkeakouluista ja yliopistoista siirtyi välittömästi hybridi- tai etäopetukseen. Hybridimallissa järjestetään lähiopetuksena vain välttämättömät opetustapahtumat, kuten laboratorioharjoitukset. Muuten kaikki luennot ym. harjoitukset pidetään netin välityksellä. Etäopetukseen siirtymisellä oli monia vaikutuksia niin opettajiin kuin opiskelijoihinkin. Opiskelijoille tietokoneruudun tuijottaminen fyysisen esityksen sijaan on saattanut tuntua vieraalta ja jopa hankalammalta opiskelutavalta. Kotona oikean opiskelumotivaation löytäminen saattaa olla vaikeampaa kuin perinteisesti luokassa istuessa, sillä virikkeitä on tarjolla enemmän ja kynnys esimerkiksi tehdä jotain aivan muuta luennon aikana on paljon pienempi ilman valvontaa. Myös sosiaalisten kontaktien vähenemisellä on saattanut olla vaikutusta motivaation löytymiseen (Ylioppilaslehti, 2020).

Opettajan näkökulmasta siirtyminen on voinut olla raskasta varsinkin kursseilla, missä digitaalista materiaalia on ollut vähän käytössä. Nopean siirtymisen myötä uutta materiaaliakin täytyi luoda nopeasti. Myös uudet opetusmenetelmät esim. Zoom-videokeskustelupalvelun välityksellä täytyi omaksua nopeasti. Mikäli kurssi on vaatinut esimerkiksi piirtämistä taululle ja sen avulla käsitteiden selittämistä, on sekin muodostanut omat haasteensa eri ohjelmistojen ja apuvälineiden opettelussa, jotta opiskelijoille voidaan tarjota lähes samanlaatuista opetusta kuin perinteisestikkin.

Nyt näyttää siltä, että etäopetus on tullut ainakin jossain muodossa jäädäkseen. Puhutaan ns. hybridiopetuksesta, joka yhdistää lähi- ja etäopetusta. Mikäli tulevaisuudessa tulee uusi samanlainen tilanne kuin nyt valloillaan oleva pandemia, etäopetus nousee taas keskiöön. Siksi onkin äärimmäisen tärkeää, että sen kehittämiseen panostettaisiin kaikilla osa-alueilla, sillä opiskelijoille täytyy tarjota laadukasta opetusta tilanteesta riippumatta. Uusia metodeja ja apuvälineitä tulee kehittää kaaosmaisen tilanteen välttämiseksi. Myös kaikenlaisen verkko-opetuksen määrä on yleisestikin kasvamassa.

Tämä diplomityö käsittelee 2000-luvun alkupuolella kehitettyä STACK-järjestelmää (System for Teaching and Assessment using a Computer algebra Kernel) ja sen hyödyntämistä Vaasan yliopistossa sähkötekniikassa (STACK, 2021). Järjestelmän avulla voidaan luoda erilaisia tehtäviä, joita opiskelijat voivat itsenäisesti ratkaista vaikkapa juuri kotisohvalta käsin, sillä tehtäviin sisältyy automaattinen arviointi ja palautteen antaminen. Järjestelmän käyttöä on hyödynnetty jo vuodesta 2019 alkaen Vaasan yliopiston piirianalyysin kursseilla, mutta sen merkittävyys etäopetuksen aikana on kasvanut. Järjestelmää itsessään on hyödynnetty yliopistossa jo aikaisemminkin yliopistonlehtori Matti Laaksosen matematiikan kursseilla.

1.1 Tutkimuskysymykset

Työn päätarkoituksena on kartoittaa, miten STACK-järjestelmään voidaan kehittää nimenomaan piirianalyysin tehtäviä, jotka edellyttävät esimerkiksi tehtävänannon lisäksi kuvan piiristä. Tarkoituksena on myös selvittää, voidaanko luoda perinteisistä tehtävistä poikkeavia tehtävätyyppejä, joissa opiskelijalle annettaisiin interaktiivinen kuva, jossa voi liikutella erilaisia asioita ja näin ollen saataisiin vastaus. Tutkimuksen tavoitteena on löytää vastaukset seuraaviin tutkimuskysymyksiin:

- Miten STACK-järjestelmää voidaan hyödyntää piirianalyysin käsittelyyn?
- Miten opiskelijalle voidaan visualisoida tehtävän ratkaisun kannalta oleellisia asioita kuten piirikaavioita, kuvaajia yms.?
- Voidaanko luoda tehtävätyyppejä, joissa kirjoittamista ja laskemista ei olisi, vaan ratkaisu perustuu interaktiivisen kuvan käyttämiseen?
- Mitä mieltä opiskelijat ja opettajat ovat työssä kehitetyistä tehtävistä ja niiden toimivuudesta?

Työssä esitetään STACK-järjestelmän tehtäviä, joita voidaan hyödyntää piirianalyysin kursseilla. Tutkimuksen aikana kehitetään ja testataan erilaisia tehtävätyyppejä ja keinoja, joiden avulla voidaan parantaa opiskelijoiden oppimiskokemusta. Tutkimus keskittyy tarkastelemaan vuoden 2020 keväällä järjestettyä ”Piirianalyysi A” -kurssia ja siellä

käytettyjä STACK-tehtäviä. Joitain tehtäviä on testattu jo vuonna 2019, mutta vuonna 2020 tehtäviä oli tarpeeksi, jotta niillä voitiin kattaa kurssin keskeisin sisältö. Tutkimuksen toteutuksen materiaali koostuu STACK-tehtävistä ja opiskelijoilta kerättävästä palautteesta. Lisäksi tarkastellaan kirjallisuutta, joka liittyy STACK-järjestelmän käyttöön opetuksessa.

1.2 Rakenne

Diplomityö voidaan jakaa kolmeen osaan:

- Tutkimuksen tausta ja STACK-järjestelmän esittely (luvut 1 ja 2).
- STACK-tehtävien luonti ja soveltuvuus piirianalyysin opetuksessa (luvut 3–5).
- Kokemukset STACK-järjestelmän kehityksestä ja käytöstä sekä opiskelijoilta saatu palaute (luku 6).

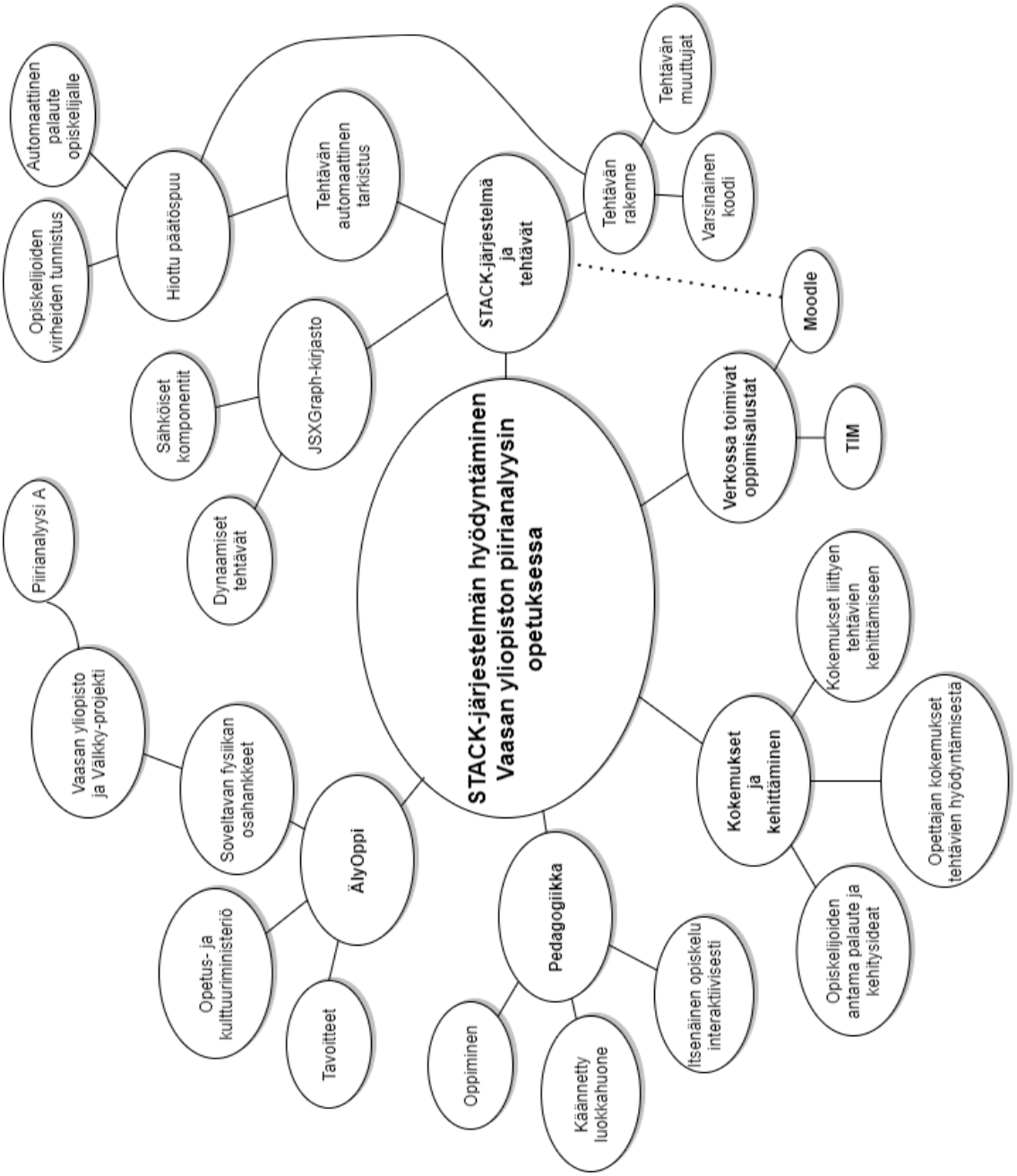
Luvuissa 1 ja 2 käydään läpi syitä, minkä takia juuri STACK-järjestelmä valikoitui kokeilukohteeksi ja mitkä muut tekijät vaikuttavat taustalla. Osiossa tutustutaan myös tarkemmin sähkötekniikan opintosuuntaan ja kursseihin, joissa järjestelmää on hyödynnetty. Teoria-osuudessa on myös pyritty selvittämään STACK-tehtävien käyttöä muualla ja niihin liittyviä keskeisiä käsitteitä, kuten *interaktiivisuus, dynaamisuus* ja *pelillistäminen*.

Luvuissa 3–5 käsittellään itse STACK-järjestelmää. Aluksi selvitetään keskeiset käsitteet ja ominaisuudet, jotka kuuluvat järjestelmään. Osiossa tarkastellaan, miten STACK-tehtäviä luodaan ja mitä kaikkea pakollista ja tarpeellista yksinkertainen tehtävä toimiakseen pitää minimissään sisällään. Kun järjestelmää on esitelty yleisesti, paneudutaan tutkimaan, miten sitä on lähdetty hyödyntämään sähkötekniikassa ja piirianalyysissä. Osiossa merkittäväksi apuvälineeksi nousee JSXGraph-kirjasto (JSXGraph, 2020), jonka avulla voidaan luoda graafisia merkkejä, esimerkiksi piirikaavioita, ja kuinka sen eri elementtejä voidaan hyödyntää tehtävän proseduaalisessa generoinnissa. Kirjaston avulla selvitetään myös, voidaanko luoda perinteisistä piirianalyysin laskuista

poikkeavia uudenlaisia tehtäviä, joissa opiskelijan ei välttämättä tarvitse kirjoittaa mitään vastausta antaessaan, pelkkä elementin liikuttaminen riittää. Tällöin tehtävä voi vaikuttaa mielenkiintoisemmalta ja helposti lähestyttävämmältä verrattuna pitkien yhtälöiden kirjoittamiseen.

Luvussa 6 käydään läpi opiskelijoilta kerätty palaute diplomityössä luoduista uusista STACK-tehtävistä. Tarkoituksena on analysoida tehtävien toimivuutta ja kartoittaa yleisiä mielipiteitä niihin ja niiden kehittämiseen liittyen. Palautteen pohjalta saadaan siis myös kehitysideoita, joilla tehtäviä voisi parannella. Osiossa käydään läpi tehtävät myös opettajien näkökulmasta ja pohditaan niiden toimivuutta. Viimeisessä luvussa kootaan johtopäätökset ja tehdään yhteenveto.

Kuvassa 1 on esitetty diplomityöhön liittyvät keskeiset aihealueet ja käsitteet ajatuskartan muodossa. Ajatuskartan viisi isoa aluetta *pedagogiikka, ÄlyOppi, kokemukset ja kehittäminen, verkossa toimivat oppimisalustat* sekä *STACK-järjestelmä ja tehtävät* muodostavat diplomityön pohjan. Jokaiseen isoon pääaiheeseen on kytketty myös sen alueen keskeisimpiä osa-alueita. Diplomityössä selvitetään, mitä kaikkea kartassa esitetyt käsitteet pitävät sisällään ja miten ne liittyvät toisiinsa.



Kuva 1. Diplomityön keskeiset aiheet ja käsitteet.

1.3 ÄlyOppi-hanke

Tämä diplomityö on toteutettu osana opetus- ja kulttuuriministeriön rahoittamaa kärkihanketta nimeltään ÄlyOppi (Pirttinen, 2019), jossa olen ollut mukana Vaasan yliopistossa vuoden 2019 syksystä lähtien ensin tutkimusavustajana ja sitten diplomityöntekijänä. Tehtäviini ovat kuuluneet erilaisten tehtävien suunnitteleminen, kehittäminen ja laatiminen STACK-järjestelmässä sekä STACK-järjestelmän uusien mahdollisuuksien kartoittaminen piirianalyysin osalta. STACK (System for Teaching and Assessment using a Computer algebra Kernel) on järjestelmä, jonka avulla voidaan luoda netissä ratkottavia pääsääntöisesti matemaattisia tehtäviä, joita voidaan tarkastaa automaattisesti.

ÄlyOppi-hanke on kollaboraatio yhdentoista suomalaisen yliopiston ja korkeakoulun kesken ja sitä koordinoi Aalto-yliopisto. Hanke aloitettiin vuonna 2018 ja se on jaettu useampaan osahankkeeseen: tietotekniikka, fysiikka, matematiikka, ja soveltava fysiikka. Oppilaitokset osallistuivat juuri näiden osahankkeiden kautta (Pirttinen, 2019).

Hankkeen tarkoitus on tuottaa digitaalista oppimismateriaalia ja kullekin osa-alueelle ja kartoittaa uusia tietoteknisiä mahdollisuuksia, joita voidaan käyttää etäopetuksessa. Esimerkiksi matematiikan osahanke keskittyy tuottamaan ja laajentamaan Abacus-materiaalipankkia, johon on kerätty eri tieteenalojen STACK-tehtäviä (Abacus, 2021). Pankin tehtäviä voidaan hyödyntää yhteistyön piiriin haluavissa korkeakouluissa.

Vaasan yliopisto osallistui ÄlyOppi-hankkeeseen matematiikan ja soveltavan fysiikan osalta, jossa keskityimme kehittämään automaattisesti tarkastettavia tehtäviä piirianalyysin näkökulmasta. Tehtävät pyrkivät pienentämään luentojen teorioiden ja laskuharjoitusten välistä kuilua ja parantamaan opiskelijan mahdollisuuksia kurssin suorittamiseksi. Tehtävissä kansainvälisestikkin aivan pioneerimaisesti aloitettiin visualisoimaan piirikaavioita käyttämällä JSXGraph-kirjastoa luomalla satunnaisesti generoituja piirejä. ÄlyOppi-hanke päättyy vuoden 2021 kesällä.

2 Pedagoginen näkökulma STACK-järjestelmän hyödyntämiseen teoreettisessa sähkötekniikassa

Tämä luku käsittelee STACK-järjestelmän ja sen hyödyntämisen kannalta keskeisiä aiheita ja käsitteitä. Luvussa käydään läpi opetuksen teoriaa ja pedagogista taustaa ja kartoitetaan sovelluskohteen eli piirianalyysin taustoja Vaasan yliopistossa. Luvussa käsitellään:

- opetuksen ja oppimisen teoriaa
- didaktiikka ja visuaalisuus oppimisen ytimessä
- modernit opetustyylit, jotka hyödyntävät verkkopohjaista materiaalia
- interaktiivisuus ja pelillistäminen
- piirianalyysi ja teoreettinen sähkötekniikka Vaasan yliopistossa.

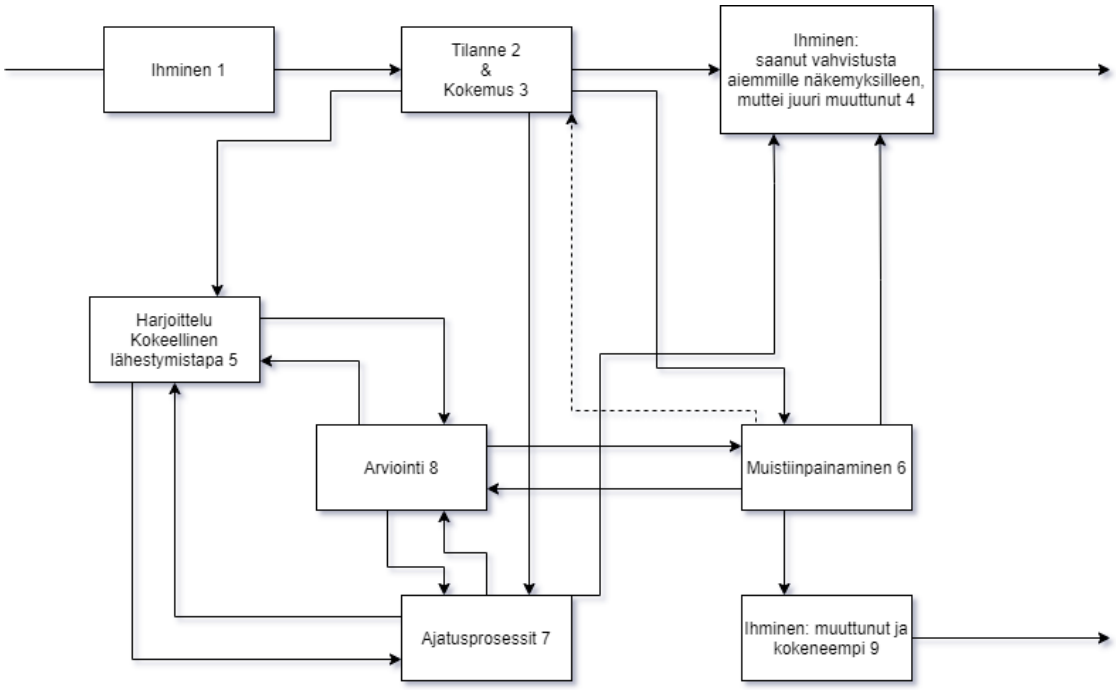
STACK-järjestelmää, sen rakennetta ja käyttöä käydään läpi luvuissa 3 ja 4.

2.1 Opetuksen ja oppimisen teoriaa

Oppiminen on ihmiselle tuttu ja jokapäiväinen rutiini. Modernissa maailmassa, jossa rajattomaan tiedon lähteeseen pääsee käsiksi parilla sormen pyyhkäisyllä älypuhelimen avulla, ei ole mikään ihme, että ihmiset törmäävät päivittäin uusiin asioihin. Jokaisella ihmisellä on siis käsitys siitä, mitä oppiminen on, sillä jokaisella on kokemusta oppimisesta. Yksinkertaisesti oppimista ei kuitenkaan voida selittää, vaan se on prosessi, joka tapahtuu erilaisten mallien avulla. Kaikki oppiminen juontaa juurensa kunkin yksilön omista elämänkokemuksista (Jarvis, 1993).

Peter Jarvisin mukaan kaikki oppiminen rakentuu aikaisempien kokemusten päälle ja jokainen yksilö suhtautuu kokemaansa omalla tavallaan. Sinikka Huhtalan väitöskirjassa *”Lähihoitajaopiskelijan oma matematiikka”* on esitetty Jarviksen

oppimisprosessin malli, jossa selvennetään ihmisen reagointi oppimistilanteessa. Malli on esitetty kuvassa 2.



Kuva 2. Peter Jarviksen oppimisprosessien malli. (Huhtala, 2000, s. 35).

Jarviksen mukaan kokemuksiin voidaan reagoida oppimalla, ei-oppimalla tai tiedostamattomalla oppimisella. Oppimisprosessin aikana kokemus muunnetaan erilaisiin osiin, kuten tiedoiksi, taidoiksi, tai käsityksiksi, joiden pohjalta ihminen poistuu oppimistilanteesta jommalla kummalla kahdesta eri vaihtoehdosta (kuvassa laatikot 4 ja 9). Jarviksen mallin mukaan ihminen (1) törmää aluksi tilanteeseen (2) ja sitä kautta kokemukseen (3). Tästä eteenpäin ihminen liikkuu mallin mukaisesti eri kohtien välillä riippuen yksilön reaktiosta tilanteeseen. Jarviksen mukaan tilanteeseen voidaan reagoida yhdeksällä eri tavalla, jotka on esitetty taulukossa 1 (Huhtala, 2000).

Taulukko 1. Peter Jarviksen mallin mukaiset reaktiot eri oppimistavoissa (Huhtala, 2000, s. 36).

Oppimistapa	**Reaktio**
Ei-oppiminen	Ennakko-oletus Huomiottajättäminen Torjunta
Ei-tiedostava oppiminen	Esitietoinen Taidot Ulkoaoppiminen
Tiedostava oppiminen	Pohdiskelu Ajattelun taidot Kokeellinen tieto

Ei-oppimisessa ihminen poistuu prosessista muuttamatta aiempaa näkemystään asiasta. Ennakko-oletustapauksessa ihmisen mielestä mitään muutosta ei ole tapahtunut ja uutta ei tarvitse oppia. Kuvan 2 prosessikaaviossa tämä kuvataan reittiniä 1–4. Huomiotta jättäminen ja torjunta taas tarkoittavat sitä, että vaikka ihminen tiedostaa, että tilanteessa on jotain uutta ja opittavaa, päättää hän silti sivuuttaa tilanteen. Näin voi käydä esimerkiksi tilanteessa, jossa ihminen kokee sen entuudestaan ja ennakkoluuloisesti liian haastelliseksi ja päättää ohittaa tilanteen. Kaikissa ei-oppimisen tilanteissa prosessista poistutaan laatikon 4 kautta (Huhtala, 2000).

Ei-tiedostavassa oppimisessa yksilö saatta kokea erilaisia asioita, mutta ei pohdi tai ajattele niitä sen enempää. Oppiminen tapahtuu siis alitajuisesti ja voi usein olla eräänlaista matkimista. Ihmiset saattavat omaksua muiden ihmisten tapoja, kuten puhetyylin, käyttäytymisen, naurutyylin yms. ajattelematta asiaa ollenkaan. Asioita voi myös opetella ulkoa, jolloin tiedetään ratkaisuja ja tapoja ongelmiin, mutta ei varsinaisesti ymmärretä, miksi näin tehdään, koska ihminen ei pohdiskele asiaa tarpeeksi. Ei-tiedostavassa oppimisessa ihminen voi poistua kuvan 2 prosessikaaviosta laatikon 4 tai 9 kautta. (Huhtala, 2000).

Tiedostavassa oppimisessa on kyse varsinaisesta *syvällisestä oppimisesta*, jossa ihminen muuttaa ajatuksiansa tapahtuman pohjalta ja näin ollen tiedostaa muutoksen.

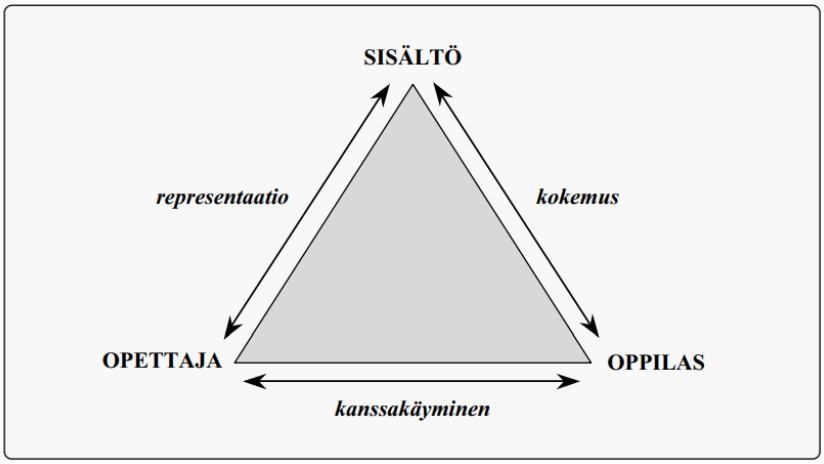
Oppiminen voi tapahtua analyyttisesti pelkästään ajattelemalla, kokeilemalla tai yhdistämällä molempia. Tiedostavan oppimisen prosessi päättyy laatikon 9 kautta (Huhtala, 2000).

Ihanteellisessa oppimistilanteessa opiskelija poistuu oppimisprosessista laatikon 9 kautta. Tällöin opiskelija on tiedostanut uudet asiat, hyväksynyt ne ja pystyy muuttamaan omaa tarkastelunäkökulmaansa uuden tiedon valossa. Uuden tiedon avulla opiskelija voi keskittyä ratkaisemaan haastavampia aiheeseen liittyviä ongelmia. Opiskelija kykenee tämän uuden tiedon valossa soveltamaan aikaisempia kokemuksia ja opittuja taitoja yhä monimutkaisemmissa tapauksissa ja jos opiskelija suhtautuu myös näihin tilanteisiin tiedostavan oppimisen vaativalla tavalla, on tuloksena onnistumisen tunne.

Kuvan 2 Prosessimallin mukaan vaarallisin tilanne on, jos opiskelija suhtautuu uuteen tilanteeseen jo lähtökohtaisesti negatiivisesti. Esimerkiksi jokin matematiikan ongelma voi tuoda mieleen aikaisemmat ikävät oppimiskokemukset matematiikan osalta, jolloin opiskelija saattaa torjua tilanteen kokonaan (Huhtala, 2000). Tällöin on riskinä, että opiskelija saattaa jättää kurssilla käydyn kriittisen osion käymättä ja putoaa pois suunnitellulta opiskelureitiltä. Tämän seurauksena syntyy tilanteita, joissa kurssilla aikaisemmin huomiotta jätetty kohta aiheuttaa sen, että myös myöhemmät siihen perustuvat asiat syrjäytetään negatiivisten tunteiden johdosta. Se johtaa nopeaan epäonnistumisten syöksykierteeseen, josta on hyvin vaikea päästä pois. Tämä ruokkii opiskelijan epätoivoa entuudestaan ja huonolla tuurilla se saattaa vaikuttaa myös muiden kurssien aiheiden omaksumiseen ja jopa kurssien läpäisyyn. On siis äärimmäisen tärkeää, että opiskelija itse pyrkii suhtautumaan jokaiseen oppimistilanteeseen avoimin mielin niin hyvin kuin mahdollista ja että hänelle olisi tarjolla myös tarvittavat apuvälineet. Opiskelijan tarpeiden täyttäminen ja niiden huomioiminen onkin yksi opettajan monista haasteista.

Didaktinen näkökulma

Didaktiikan tarkoituksena on tutkia opetusta ja etsiä määritelmää hyvälle opetukselle. Yksinkertaisuudessaan opetustapahtuma on kolmen eri tekijän relaatio toistensa välillä. Tätä kutsutaan myös *didaktiseksi kolmioksi* (*engl.* pedagogical triangle) ja se on esitetty kuvassa 3. Didaktisen kolmion määritelmä kehitettiin jo 1800-luvulla Johann Friedrich Herbartin pedagogiikkaan perustuen (Friesen & Osguthorpe, 2017).



Kuva 3. Didaktinen kolmio (Patrikainen, 2012).

Kolmiossa on siis sisälletty opetustapahtuman keskeiset asiat: opettaja, opetettava ja opetettava asia (Patrikainen, 2012). Didaktisen kolmion yksi kulmista on oppilas, joka on vuorovaikutuksessa opettajan ja opetustapahtuman aiheen kanssa. Kolmiosta voidaan huomata, että sisällön oppiminen ei välttämättä tarvitse opettajaa, vaan oppilas voi oppia myös itsenäisesti. Opettajan tehtävänä on taas toimia eräänlaisena yhdistävänä tekijänä sisällön ja oppilaan välillä ja aiheen esittämisen kautta auttaa oppilasta ymmärtämään paremmin. Kolmioon saatetaan joskus sisällyttää myös katkonainen viiva ”OPETTAJA – kokemus” -välillä, jonka tarkoituksena on kuvata opettajan aikomusta muuttaa oppilaan oppimiskokemusta (Friesen & Osguthorpe, 2017). Yleisenä trendinä on, että opettajan rooli opiskelijan ohjaajana korostuu esim. käänteisessä opetuksessa.

2.1.1 Visualisointi opetuksessa

Visualisoinnilla on tärkeä näkökulma melkein missä tahansa tilanteessa, jossa joku selittää toiselle jotain. Visualisointi tarkoittaa jonkin abstraktin käsitteen tai tiedon esittämistä erilaisten fyysisten apuvälineiden kanssa. Näitä apuvälineitä voivat olla esimerkiksi luonnokset, piirrokset, kuvat, kuvaajat, animaatiot, fyysinen malli ja melkein mikä tahansa kuvaa vastaava apukeino. Visualisoinnin tarkoituksena on siis muuntaa dataa tiedoksi (Gee, 2005). Mikäli asia on selitettävälle henkilölle täysin tuntematon, on visualisoinnilla todella iso merkitys selitettävän asian sisäistämisessä.

Opetuksessa ja uuden oppimisessa visuaalisuudella on myös suuri merkitys. Tarkoituksena on saada opiskelija, joka ei välttämättä aluksi tiedä aiheesta juuri mitään, sisäistämään haluttu asia tai taito. Joidenkin asioiden tai tehtävien ratkaiseminen voi osoittautua melko hankalaksi ilman toimivaa visualisointia. Esimerkiksi opetustilanteessa matematiikan geometrinen tehtävä voi olla hankala selittää pelkän tekstin avulla, saati sen ymmärtäminen tai hahmottaminen (Guzman, 2002). Tästä syystä tehtävien esittämisessä käytetään tilannekuvaa, josta opiskelija voi melkein heti huomata oikeanlaisen lähestymistavan ongelman ratkaisemiseksi. Tehtävän ratkaiseminen kannattaa myös aloittaa kuvan piirtämisellä annettujen lähtötietojen perusteella, jotta tehtävän hahmottaminen ja ratkaiseminen helpottuvat.

Fysiikan puolella visualisoinnin tärkeys korostuu, kun erilaisia fysikaalisia suureita aletaan kuvaamaan vektoreilla. Vektoreiden ja niihin liittyvien laskutoimitusten merkitystä ja konseptia on vaikea ymmärtää ilman auttavaa visualisointia. Tämä on huomattu myös sähkömagneettisen kenttäteorian puolella, missä sähkö- ja magneettikenttien käyttäytyminen on vaikea asia sisäistää. Tätä varten on kehitetty esimerkiksi animaatioita, joiden avulla voidaan paremmin kuvata sähkömagneettisia ilmiöitä ajan suhteen. Myös simulaatio-ohjelmistojen avulla pyritään visualisoimaan ilmiöitä, jotta opiskelija ymmärtää syvällisemmin miten esim. kenttäviivat kulkevat eri tilanteissa (Vesapuisto, Vekara & Korpinen, 2013). Visualisoinnin merkitys korostuu erityisesti tehtävissä, jotka joudutaan ratkaisemaan kolmiulotteisessa avaruudessa.

Myös piirianalyysin puolella visualisointi on äärimmäisen tärkeää jo tehtävän asettelun kannalta. Tehtävien laatiminen ja ratkaisu edellyttävät lähes poikkeuksetta piirikaaviota, johon kaikki tehtävässä käytetyt muuttujat, tehtävänanto ja ratkaisukin perustuvat. Piirikaavioon voidaan merkitä tarvittavia referenssinuolia ja kysymyksenasettelussa voidaan määrittää haluttu suure piirtämällä se piiriin, kuten jonkin komponentin yli oleva jännite. Piirikaavion merkitys painottuu myös siinä tapauksessa, kun suoritetaan esimerkiksi lähdemuunnoksia tai muita piirimuunnoksia analyysin helpottamiseksi. Ilman kuvaa on melko vaikea havainnollistaa, miten kukin komponentti kytkeytyy toisiin komponentteihin piirimuunnosten jälkeen (Duffy, Sorby & Bowe, 2016).

Visualisointi on myös tärkeä mahdollisuus opiskelijan mielenkiinnon herättämisen kannalta. Abstraktien käsitteiden esittäminen pelkän tekstin avulla voi olla uuvuttavaa ja hahmottaminen hankalaa. Tämä voi johtaa aikaisemmin mainittuun negatiiviseen oppimiskokemukseen, jonka seurauksena opiskelija putoaa tavoitellulta opintopolulta. Opiskelijan oppimismotivaatiota voidaan lisätä selventämällä vaikeasti ymmärrettäviä käsitteitä visualisoinnin kautta. Eräässä tutkimuksessa (Irawan, Mukhlash, Adzkiya & Sanusi, 2019) todettiin, että trigonometrian opetuksessa visualisoinnilla oli myönteinen vaikutus opiskelijoiden motivaatioon, vaikka tulosten perusteella se ei suoraan vaikuttanut konseptin ymmärtämiseen. Pelkkä visualisointi ei siis riitä asian ymmärtämisen kannalta, vaan kuvia pitää osata myös tulkita kontekstissaan.

2.1.2 Verkkomateriaali ja etäopiskelu

Käsitteen verkkomateriaali alle voidaan sijoittaa kaikki se materiaali mitä löytyy verkosta. Materiaali voi olla käytännössä mitä vain Power Point -esityksistä animaatioihin ja videoihin. Verkon ja ohjelmistojen kehitys on mahdollistanut ja tehnyt tiedon jakamisen erittäin helpoksi. Sen tuottaminen on myös helpottunut erilaisten työkalujen kehityksen seurauksena.

Nykyään korkeakoulukursseista valtaosa hyödyntää jo verkkomateriaalia. Moodle-pohjaiset ja sen kaltaiset oppimisalustat, johon korkeakoulut voivat lisätä omat

kurssialueensa, ovat yleistyneet 2000-luvulla, jonka alkupuolella Moodlea alettiin kehittämään (Moodle, 2020). Erilaisia oppimisalustoja voidaan pitää kurssien ”kotisivuna”, josta löytyy kurssin suorittamiseen tarvittavat tiedot ja materiaalit, kuten luentokalvot ja tehtävät.

Verkossa käytettävien oppimisalustapohjaisten kurssien etuna on niiden skaalautuvuus suurelle opiskelijamäärälle. MOOC-alustat (engl. Massive Open Online Course) mahdollistavat suurienkin osallistujamäärien hallinnoimisen. Sitä kautta on helppo jakaa tietoa monelle eikä se välttämättä vaadi sitä, että kaikki opiskelijat kokoontuisivat yhteen luentosaliin, vaan opiskelijat voivat osallistua kursseille vaikka ympäri maailmaa. Esimerkiksi MIT (Massachusetts Institue of Technology) tarjoaa ilmaiseksi verkossa suoritettavia MOOC-kursseja kenelle vain riippumatta sijainnista tai ajasta (Massachusetts Institute of Technology, 2021).

Nopeaa tiedon jakamista massoille voidaan siis hyödyntää MOOC-kursseilla. Useat oppimisalustat, kuten Moodle, mahdollistavat myös automaattisesti tarkastettavien tenttien ja testien luomisen. Se taas mahdollistaa opiskelijalle kokonaisten kurssien käymisen täysin itsenäisesti kotoa käsin. Järjestäjän osalta se on myös helppo toteuttaa isollekin joukolle, sillä automaattinen tarkastus eliminoi kätevästi siihen tarvittavien resurssien määrää. Yleensä tähän soveltuvat tehtävät ovat ainakin aluksi monivalinta-kysymyksiä tai muita vastaavanlaisia, joissa arvosteluasteikko on kutakuinkin hyväksytty tai hylätty. Monivalintakoe ei kuitenkaan riitä tai sovellu kaikkialle ja esimerkiksi laskentapainoitteisilla aihealueilla sen järjestäminen reilulla ja oppimista testaavalla tavalla on hankalaa. Tätä varten on kehitetty omanlaisiaan ohjelmia, kuten STACK-järjestelmä, jotka toimivat jo enenevästi matemaattisilla aloilla.

Toisaalta automaattinen ja ei-valvottu ns. kotitentti mahdollistaa myös sen, että opiskelija pystyy helpommin lunttaamaan tai käyttämään ei-sallittuja apuvälineitä. Connecticutin yliopiston taloustieteiden laitoksella tehdyssä tutkimuksessa (Harmon & Lambrinos, 2008) tarkasteltiin kahta taloustieteen online-kurssia, joissa toisessa

viimeinen tentti oli valvottu ja toisessa ei. Tutkimuksen ja siinä kehitetyn mallin mukaan voidaan olettaa, että vilppiä oli tapahtunut ainakin enemmän juuri ei-valvotussa tentissä. Samaan lopputulokseen päätyi myös toinen tutkimus (King, Guyette & Piotrowski, 2009), jossa kartoitettiin opiskelijoiden mielipiteitä vilpin suhteen verkossa järjestettävien ja perinteisten kurssien välillä. Kyselyn mukaan opiskelijat olivat suurimmaksi osaksi sitä mieltä, että verkossa järjestettävässä kokeessa on helpompaa harrastaa vilppiä erilaisilla keinoilla. Myös Norjassa tehdyssä tutkimuksessa (Chirumamilla, Sinde & Nguyen-Duc, 2020) päädyttiin samanlaisiin tuloksiin.

Yksi tärkeä näkökulma on myös tentin suorittavan opiskelijan henkilöllisyyden varmentaminen. Fyysisessä tentissä opiskelijalta varmennetaan henkilöllisyys palautuksen yhteydessä opsikelijakortin, ajokortin tms. avulla. Verkossa järjestettävässä tentissä riittää, että opiskelija on kirjautunut alustalle opiskelijatunnuksilla. Varmennusta siitä, että kyseessä on tietty opiskelija, ei ole.

Nämä seikat täytyy ottaa huomioon, kun suunnitellaan verkossa järjestettävää suoritusta. Ihanteellisessa tilanteessa lunttaamisesta ei varsinaisesti olisi mitään hyötyä eli tentti olisi materiaaliin pohjautuva tentti, jossa kaikki apuvälineet ovat sallittuja. Verkossa järjestettävän tentin tilanteessa ei voida kuitenkaan olla varmoja esim. siitä, että opiskelija suorittaa tenttiä yksin. Opiskelija voi olla yhteydessä muihin opiskelijoihin, jolloin he voivat ratkaista tehtävät yhdessä. Tällöin on hyvin vaikeaa arvioida, omaako tenttiä suorittava yksilö itse tarvittavat tiedot ja taidot. Tällaisten tilanteiden välttämiseksi on kehitetty sähköinen tenttipalvelu Exam (EXAM, 2021). Exam-tentissä opiskelija varaa itsellensä suoritusajan ja käy suorittamassa tentin koulun omassa erityisessä Exam-luokassa. Luokkahuone on videovalvottu ja siellä voi olla vain tietty määrä opsikelijoita paikalla samanaikaisesti. Exam-tenttiminen mahdollistaa myös esim. toisten korkeakoulujen järjestämien Exam-tenttien suorittamisen oman oppilaitoksen Exam-huoneessa.

2.1.3 Interaktiivisuus ja pelillistäminen

Interaktiivisuudelle löytyy monta määritelmää, mutta yleisesti sillä tarkoitetaan ihmisen ja tietokoneen välistä vuorovaikutusta. Uudet sukupolvet 1980-luvulla syntyneistä eteenpäin ovat kasvaneet valtavan digitalisaation ja multimedian kehityksen aikana ja ero käyttäytymisessä vanhempiin sukupolviin nähden on merkittävä. Niin kutsuttu *diginatiivi*-sukupolvi on tottunut löytämään informaationsa ennemmin netistä kuin perinteisistä lähteistä, kuten kirjoista. Iso osa myös toisten ihmisten kanssakäymisestä tapahtuu älypuhelimen tai jonkin muun digitaalisen käyttöympäristön välityksellä (Pradono, Astriani & Moniaga, 2013). Myös erilaiset videopelit ovat nykyaikana valtavirtaviihdettä ja ne ovat juurruttaneet paikkansa perinteisten medioiden, kuten television ja elokuvien, rinnalla. Se onkin yksi nopeimmin kasvavista teollisuuden aloista.

Digitaalisesta kehityksestä johtuen, ovat myös sen aikana varttuneet nuoretkin oppimiskäytökseltään erilaisia verrattuna aikaisempiin sukupolviin. Diginatiivit ovat kasvamisensa aikana tottuneet tekemään päivittäisiä asioita kynän ja paperin lisäksi tai sijasta myös tietokoneella. Luonnollisesti myös oppiminenkin tapahtuu tietokoneen avustuksella. Tästä johtuen opiskelijoille kannattaakin tarjota interaktiivista materiaalia, jonka käsittelyyn he ovat jo tottuneet normaaleissa elämäntilanteissaan, opiskelumotivaation ylläpitämiseksi tai synnyttämiseksi (Bennett, Maton & Kervin, 2008).

Interaktiivinen materiaali voi olla videoituja luentoja, animaatioita, yksinkertaiseen monivalintakysymykseen vastaamista luentomateriaalin keskellä tai muuta netin kautta jaettavaa materiaalia. Tarkoituksena on tarjota opiskelijalle mahdollisimman mielenkiintoista ja mukaansatempaavaa aineistoa, joka herättää mielenkiinnon opiskeltavaa ainetta kohtaan. Yksi tälläinen metodi on viime vuosina yleistynyt *pelillistäminen*. Pelillistämisessä on kyse oppimateriaalin ja oppimiskokemuksen muuntamisesta muotoon, joka muistuttaa enemmän peliä kuin perinteisiä menetelmiä (Soman & Huang, 2013). Videopelit tarjoavat yksilöille onnistumisen tunteita pienistäkin asioista ja koukuttavat pelaajaa suorittamaan lisää. Pelillistämisen tarkoituksena on

napata videopeleistä tutut motivaatiokeinot ja suorittamisen hauskuus ja sisällyttää ne oppimateriaaleihin ja opetusmetodeihin.

Yksi mahdollinen keino voi olla pelityyppisen oppimiskokemuksen järjestäminen eräänlaisena kilpailuna. Voidaan järjestää erilaisia oppimistasoja, joiden saavuttamisesta on palkintona jotain, kuten esimerkiksi videopeleistä tuttu muille näytettävä todiste tai *saavutus* (engl. achievement), jonka voi laittaa näytille vaikka omaan Moodle-profiiliinsa. Voidaan myös järjestää erilaisia kilpailuja vaikkapa tehtävän ratkaisemisesta, jossa on viikoittain päivittyvä tulostaulu ja parhaimmille on luvassa palkinto. Tällöin toisten kanssa kilpaileminen voisi toimia motivaationa ainakin joillekin yksilöille.

2.2 Teoreettinen sähkötekniikka Vaasan yliopistossa

Sähkö- ja energiatekniikka on yksi tekniikan kandidaatin opintosuunnista Vaasan yliopistossa. Suunnan kandidaatin tutkinto noudattaa rakenteeltaan yleistä teknillisten aineiden tutkintorakennetta, joka koostuu perusopinnoista, suunnan opinnoista ja moduulista. Perusopinnot ovat kaikille tekniikan alan opiskelijoille samat ja käsittelevät kaikkia opintosuuntia pohjustavia aiheita, kuten matematiikkaa, fysiikkaa ja ohjelmointia. Suunnan opinnot eroavat toisistaan riippuen opintosuunnasta ja ne keskittyvät syventämään oman alan osaamista.

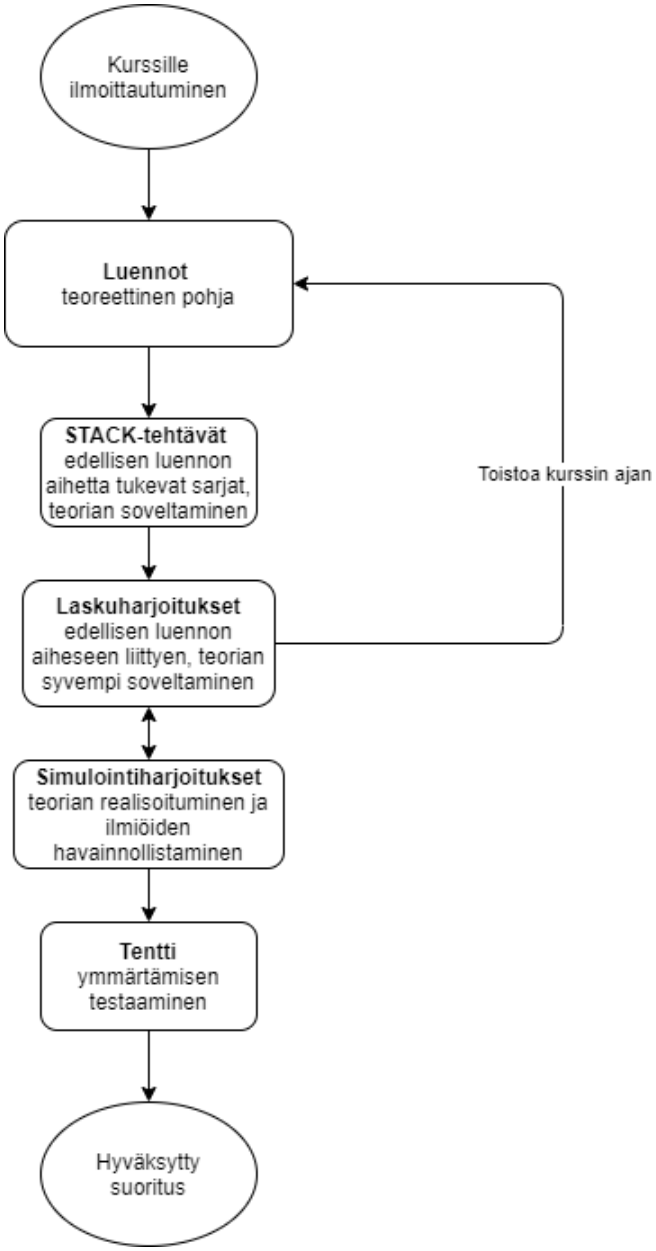
Tkk-tutkinnon sähkö- ja energiatekniikan suunnan opintoihin kuuluu teoreettisen sähkötekniikan peruskursseja kuten piirianalyysin liittyvät ”Piirianalyysi A” ja ”Piirianalyysi B”, jotka käsittelevät tasa- ja vaihtovirtatekniikan perusteita, muutosilmiöitä ja siirtojohtoja. ÄlyOppi-hankkeen tuotoksia on pääsääntöisesti testattu ”Piirianalyysi A” -kurssilla, joissa ne ovat toimineet perinteistä materiaalia tukevana työkaluna opiskelijoille. Kurssin ”Piirianalyysi A”:n rakenne koostuu vuorovaikutteisista luennoista ja laskuharjoituksista, joissa opiskelijoita rohkaistaan keskustelemaan omista ratkaisumenetelmistään opettajan ja muiden opiskelijoiden kanssa, sekä simulointiharjoituksista. Vuodesta 2019 alkaen uutena lisänä kursseilla on

käytetty myös STACK-tehtäviä, joissa on jatkuva reaaliaikainen vuorovaikutus automaattitarkistuksen kautta (Vaasan yliopisto, 2019).

STACK-tehtäviä on tarjottu opiskelijoille osana ”Piirianalyysi A” -kurssia yliopiston erillisellä STACK-alustalla eripituisina tehtäväsarjoina. Tehtäväsarjassa voi olla useampi versio samasta piiristä siten, että jotain on kuitenkin muutettu erilaiseksi tai se on voinut koostua myös erilaisista tehtävätyypeistä. Esim. kompleksilukuja käsiteltäessä tehtäväsarja voi sisältää tehtäviä, joissa koordinaatistoon täytyy merkata kompleksiluvun osoitin tai seuraavassa tehtävässä täytyy laskea yhteenlaskuja kompleksiluvuilla. Tehtävien piireissä voivat muuttua niin referenssisuunnat, kuin komponentitkin siten, että opiskelija saa eteensä aina uudenlaisen tehtävän.

Opiskelijoilta vaadittiin kurssin aikana täydet suoritukset kaikista annetuista STACK-tehtävistä, siten että jokainen tehtäväsarja täytyi olla täysin oikien. Kaikkia sarjoja opiskelijat saivat kuitenkin yrittää ilman rajoituksia niin monta kertaa kuin halusivat.

Piirianalyysi A -kurssi koostuu luennoista, STACK-tehtävistä, laskuharjoituksista, simulointiharjoituksista ja tentistä. Kurssin suoritusjärjestys ja siihen kuuluvat osasuoritukset on kuvattu vuokaavion avulla kuvassa 4. Esitetyssä kaaviossa opiskelija osallistuu aluksi viikottaiselle luennolle, jossa luodaan teoreettinen pohja kyseisen luentoviikon aiheeseen liittyen. Heti tämän jälkeen opiskelijalle annetaan juuri käytyyn teoriaan liittyen STACK-tehtäviä, joiden avulla voidaan vahvistaa perusteiden havainnollistamista. Tämä diplomityö keskittyy seuraavissa luvuissaan käymään läpi STACK-järjestelmää ja piirianalyysissä käytettyjä tehtäviä. Kun teoria ja sen perusteet on sisäistetty, opiskelija siirtyy ratkaisemaan varsinaisia laskuharjoitustehtäviä.



Kuva 4. Piirianalyysi A -kurssin etenemisjärjestys.

Piirianalyysin kursseilla STACK-tehtävät keskittyvät laskuharjoituksista eroavasti ennen muuta lausekkeiden muodostamiseen, esim. yhtälöiden muodostamiseen siis ilman lukuarvoja. Laskuharjoitustehtävät käsittelevät samaa aihetta kuin luennot ja STACK-tehtävät, mutta ne paneutuvat vielä vähän syvällisemmin siihen. Käytännössä ne ovat monimutkaisempia tehtäviä, joissa tarkoituksena on laskea haluttuja luku- tai numeerisia arvoja annettuihin arvoihin perustuen. Kun luennon aiheen laskuharjoitukset on laskettu ja tarkastettu, aloitetaan toistosilmukka seuraavan viikon luennolla ja aiheella.

Tietyssä vaiheessa kurssia siirrytään suorittamaan simulointiharjoituksia, joiden tarkoituksena on havainnollistaa teoriassa ja laskuharjoituksissa käytyjä asioita erilaisten simulaatiokokeiden avulla. Ne visualisoivat opiskelijalle piirin eri komponenttien toimintoja ja käyttäytymistä eri tilanteissa. Simulointiharjoitukset tukevat myös laskuharjoituksista suoriutumista konkreettisilla esimerkeillä, joiden avulla opiskelija voi saada oivaltavia kokemuksia ja näin syvällisesti oppia aiheen uudesta näkökulmasta.

Kun teoria on omaksuttu ja opiskelija osaa soveltaa oppimaansa myös konkreettisissa ja soveltavissa tilanteissa, on aika suorittaa kurssin viimeinen osasuoritus eli tentti. Tentissä testataan opiskelijan kykyä sisäistää kurssilla käytyjä asioita siten, että niitä voidaan soveltaa uudenlaisessa ongelmatilanteessa. Tenttitehtävät ovat samantapaisia kuin laskuharjoituksissa käydyt tehtävät. Yleensä niissä kysytään jotain eri asiaa tutusta piiristä tai jännitteiden ja virtojen referenssinuolien suuntaa on muutettu, joten tenttitehtävä ei ole identtinen laskuharjoitusten kanssa.

Suoritettuaan kaikki osasuoritukset hyväksytysti, opiskelija on läpäissyt kurssin ja voidaan todeta, että opiskelijalla on riittävät tiedot ja taidot kyseisestä aiheesta.

Piirianalyysi ja matematiikka

Piirianalyysi on sähkömagneettisen kenttäteorian erikoistapaus. Kenttäteoria käsittelee varausten liikkumista ja niihin liittyviä käsitteitä, kuten sähkö- ja magneettikenttiä. Tästä näkökulmasta kenttäteorian ymmärtäminen ennen piirianalyysiin siirtymistä olisi tarkoituksenmukaisempaa, jotta opiskelijalla olisi taustalla ymmärrys, siitä, mihin piirianalyysi perustuu ja miksi sitä voidaan hyödyntää ja miten erilaiset komponentit toimivat. Kenttäteoria on kuitenkin matemaattisesti vaativampaa vektorilaskennan, integraalien ja derivointien takia, joten piirianalyysiä käydään läpi opintojen aikaisemmassa vaiheessa (Nilsson & Riedel, 2015).

Näin tehdään myös Vaasan yliopistossa. Piirianalyysin täydellinen ymmärtäminen ei vaadi kenttäteorian syvällistä osaamista, vaan lukion fysiikan oppimäärä tarjoaa riittävät edellytykset komponenttien toimintojen ymmärtämiseen. Piirianalyysi keskittyy soveltamaan kenttäteoriaa yksinkertaistaen ilmiöitä, jotta virtapiirien analysoiminen helpottuu. Piirianalyysi sisältää:

- Tasa -ja vaihtovirtapiirien analysointia Kirchhoffin ja Ohmin lakien avulla.
- Virtapiirien esittämistä keskitetyn mallin avulla käyttäen vastuksia, keloja, kondensaattoreita, sekä jännite- ja virtalähteitä.
- Komponenttien tehojen laskemista.
- Tehokkaiden analysointimenetelmien, kuten silmukka- ja solmumenetelmä, soveltamista monimutkaisissa piireissä.
- Vaihtovirtapiirien analysoimista osoitinlaskennan avulla.
- Muutosilmiöitä ja taajuusanalyysiä.
- Siirtojohtojen toimintaa ja laskemista.

Matematiikan osalta tasa- ja vaihtovirtapiireissä piirianalyysin oppiminen edellyttää lukion matematiikan, matriisien ja kompleksilukujen ymmärtämisen. Opintosuunnitelman mukaisesti kaikki tarvittavat matematiikan ja fysiikan kurssit on suoritettu piirianalyysin alkaessa. Kurssin läpäisemisen sekä aiheen sisäistämisen ja oppimisen kannalta onkin tärkeää, että opiskelija on suorittanut edeltävät matematiikan kurssit ja näin ollen saavuttanut osaamisen matematiikan saralta. Mikäli opiskelijan matemaattiset taidot ovat puutteelliset, niin hänelle on hyvin vaikeaa sisäistää tarvittavat piirianalyysin käsitteet ryhmänsä mukana, jos samanaikaisesti täytyisi keskittyä oppimaan myös matemaattiset ratkaisukeinot. Jarviksen mallissa tämä saattaa johtaa helposti uuden oppimisen sijasta torjuntaan ja johtaa lopuksi epäonnistumiseen.

Sinikka Huhtalan väitöskirjassa on pohdiskeltu lähihoitajaopiskelijoiden matemaattista osaamista. Se on yksi tärkeä osa lähihoitajakoulutusta, koska sitä sovelletaan työssä lääkelaskujen muodossa. Lähihoitajan on kyettävä antamaan potilaalle juuri oikea määrä

tarvittavaa lääkettä, muuten seuraukset voivat olla katastrofaaliset. Matemaattisten ja yleisesti tehtäviin liittyvien virheiden osalta lähihoitajat ja tekniikan opiskelijat eivät välttämättä eroa toisistaan kovinkaan paljon. Taulukkoon 2 on kirjattu toisaalta Huhtalan havaitsemia lähihoitajien tekemiä virheitä ja toisaalta verrattu niitä piirianalyysin kursseilla esiintyviin virheisiin.

Taulukko 2. Opiskelijan tekemien virheiden analogia tekniikan ja lähihoitaja-opiskelijoiden välillä. Molemmissa käsitellään käytännönläheisiä matemaattisia tehtävissä (Huhtala, 2000 & Vesapuisto 2021).

Lähihoitajien lääkelaskuissa esiintyviä virheitä (Huhtala, Helsingin yliopisto)	**Piirianalyysi (Vaasan yliopisto)**
Tuloksen analysoiminen esim. onko vastaus järkevä ja suhteellisuuden taju.	Suhteellisuuden taju tuloksessa? Esim. voiko piirissä esiintyvä teho olla enemmän kuin sinne syötetty?
Kirjallisen kysymyksen hahmottaminen.	Kysymyksen hahmottaminen ja tehtävänannon ymmärtäminen.
Oikean ratkaisun yhtälön muodostaminen	Oikean ratkaisun yhtälön muodostaminen
Vastauksen muodostaminen siten, että siitä on mahdoton päätellä, mitä opiskelija on ajatellut.	Vastauksen muodostaminen siten, että siitä on mahdoton päätellä, mitä opiskelija on ajatellut.

Taulukosta voidaan nähdä, että samanlaisia tilanteita virheiden osalta voidaan löytää molemmilla koulutusaloilla. Yksi tärkeä seikka tehtävän ratkaisussa on siitä saadun vastauksen realistisuuden tarkastelu. Sen avulla voidaan heti huomata, onko tehtävän ratkaisu mennyt jossain määrin pieleen. Huhtala nostaa esimerkkinä lääkelaskun, johon opiskelija oli vastannut, että potilaalle tulee tarjota kerralla 8 000 000 tablettia (Huhtala, 2000). Syitä tuon vastauksen antamiseen on varmasti monia, mutta todennäköisesti se on kaikille selvää, että vastaus ei voi pitää paikkaansa. Samanlaisena voidaan pitää tilannetta piirianalyysin puolella, jossa opiskelijan antama vastaus on täysin ristiriidassa esimerkiksi energian säilymislain kanssa.

3 STACK-järjestelmän ominaisuudet ja käyttö

Tässä luvussa perehdytään STACK-järjestelmään (System for Teaching and Assessment using a Computer algebra Kernel) ja sen erilaisiin toimintoihin ja mahdollisuuksiin perusperiaatteiltaan. Luvun keskeisimmät aiheet voidaan tiivistää seuraavan luettelon avulla:

- tarvittavat ohjelmointikielet
- JSXGraph-JavaScritp -kirjasto
- STACK-tehtävän koostumus ja toiminta.

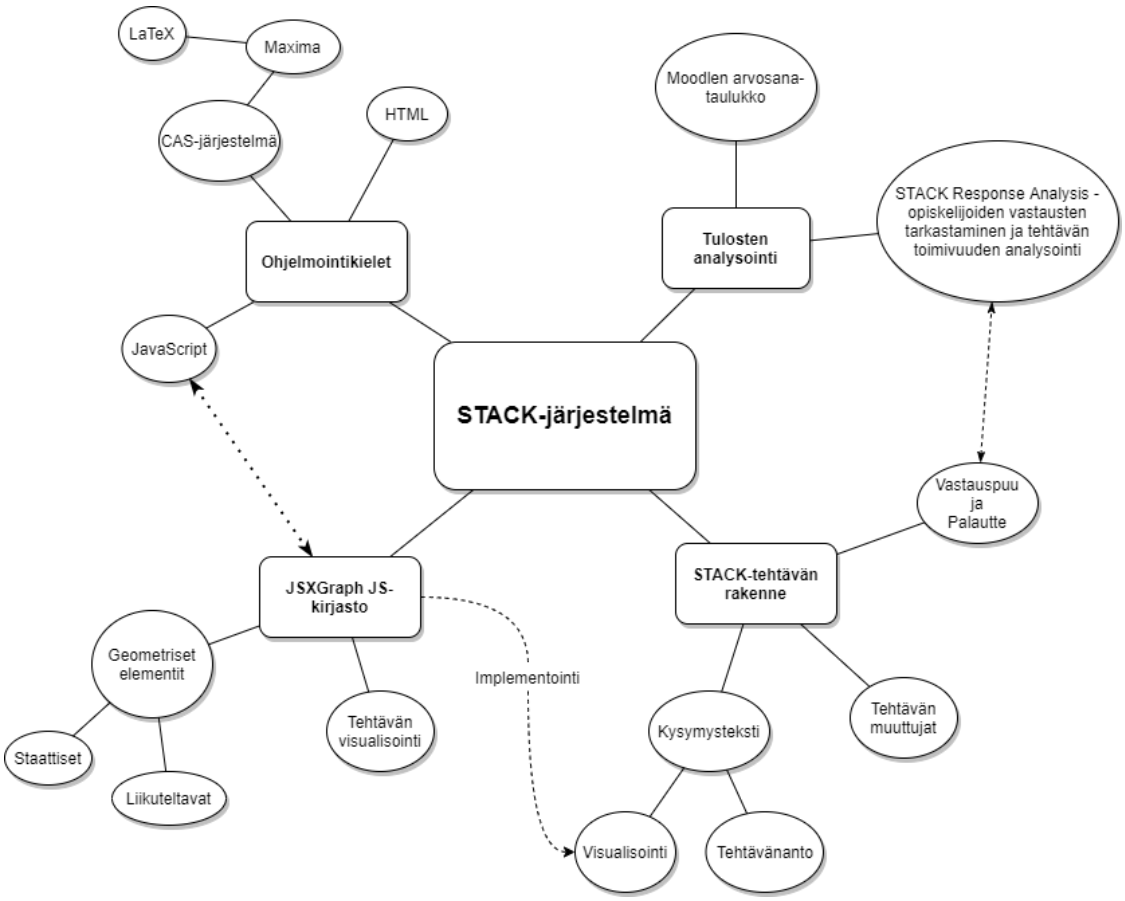
STACK syntyi 2000-luvun alkupuolella Chris Sangwinin tuotoksena ja sen pääsääntöinen tehtävä on tuottaa Moodle-pohjaisille alustoille matemaattisia tehtäviä, joita opiskelijat voivat ratkaista omalla tietokoneellansa (Sangwin, 2013).

Tehtäviin voidaan lisätä vaihtelevutta satunnaisgeneroinnin avulla, jonka seurauksena samanlaisesta tehtävästä luodaan aina eri versio eri suorituskerroille. Tällä tavoin voidaan estää oppimisen kannalta haitallista ulkoaopettelua ja sen sijaan kannustaa opiskelijoita ymmärtävään oppimiseen.

STACK-järjestelmään voidaan luoda tai tuoda myös tehtäviin liitettäviä kuvia, jotta opiskelijoille voidaan visualisoida kysyttyä ongelmaa paremmin. Kuvia voidaan tuoda perinteisellä keinolla, jossa ne luodaan etukäteen jossain muussa ohjelmistossa ja tuodaan tehtävään sen jälkeen tai voidaan hyödyntää tuettua JSXGraph-JavaScript -kirjastoa, jonka avulla voidaan piirtää erilaisia geometrisiä kuvioita. Erilaisten kuvien tuominen ja luominen mahdollistavat muidenkin kuin pelkästään matematiikkaan liittyvien tehtävien luomisen.

Vaasan yliopistossa STACK-tehtäviä on käytetty pitkään mm. matematiikan lineaarialgebrassa, mutta vuodesta 2019 eteenpäin myös sähkötekniikan puolella

piirianalyysissä. Tehtäviin voidaan rakentaa erilaisia tarkastelupolkuja, joissa verrataan opiskelijan antamaa vastausta etukäteen määritettyihin parametreihin, jonka seurauksena voidaan tutkia oppilaan tekemiä mahdollisia virheitä ja antaa mukautuvaa palautetta riippuen annetusta vastauksesta automaattisesti. Kuvassa 5 on esitetty STACK-järjestelmään liittyvät keskeiset osat ja käsitteet.



Kuva 5. STACK-järjestelmään liittyvät keskeiset käsitteet.

STACK-järjestelmään tehtäviä luodaan käyttämällä järjestelmän sisäistä editoria, jolloin kaikkia kysymyksen toimintaan vaikuttavia osia ei tarvitse itse koodata. Editoria voidaan käyttää selaimen välityksellä, jolloin tehtävä voidaan kehittää vaikka opettajan omalla tietokoneella.

3.1 Ohjelmointikielet

STACK hyödyntää tehtävissään montaa erilaista ohjelmistoa, jotta lopputuloksena saadaan toimivia ja näyttäviä tehtäviä. Kysymykset itsessään rakentuvat HTML-pohjalle, jossa hyödynnetään Maxima-ohjelmisoa, JavaScriptiä ja JSXGraph-ohjelmistoa. STACK hyödyntää myös LaTeX-muotoilua, jonka avulla voidaan kirjoittaa tyylikkään näköisiä matemaattisia yhtälöitä. (STACK, 2021).

3.1.1 Maxima-ohjelmisto

Maxima on tietokonealgebrajärjestelmä (Computer Assessment System, CAS), jota hyödynnetään STACK-järjestelmässä. Sen avulla määritellään tehtävässä esiintyvät muuttujat. Maximan tai yleisesti CAS-järjestelmän avulla suoritetaan laskutoimitukset ja muut matemaattiset operaatiot muuttujien välillä sekä tehtävässä että myöhemmin tehtävän tarkastuksessa ja arvioimisessa. CAS-järjestelmää voidaan pitää yhtenä peruspilarina, jonka päälle STACK rakentuu. Maximan avulla voidaan luoda monimutkaisia muuttujia, jotka muuntautuvat muiden tehtävässä käytettyjen ja arvottujen muuttujien mukaan hyödyntämällä esimerkiksi sen tukemaa *if else* -rakennetta niinkuin monessa muussakin ohjelmointikielessä.

Maxima on 1960-luvulla kehitetyn CAS-järjestelmä Macsyman jälkeläinen ja se kehitettiin Yhdysvalloissa Massachusetin teknillisessä korkeakoulussa (MIT). Maxima itsessään syntyi siis Macsyman pohjalle ja 1990-luvun lopulla siitä tuli avoimen lähdekoodin ohjelmisto (Maxima, 2020).

3.1.2 JavaScript- ja HTML -ohjelmointikielet

STACK-tehtävä on STACK-järjestelmään luotu tehtävätyyppi, joka generoidaan HTML-koodina (enlg. *Hypertext Markup Language*) ja johon voidaan lisätä LaTeX-ladontajärjestelmän avulla matemaattisia merkintöjä. Jotta tehtävät olisivat myös dynaamisia ja interaktiivisia, voidaan tehtävän kehittämisessä hyödyntää JavaScript-

ohjelmointikieltä. Sen avulla voidaan luoda erilaisia objekteja, kuten painikkeita ja kuvioita, joita voidaan hyödyntää tehtävissä. JavaScriptin avulla on mahdollista luoda tehtäviä, joissa opiskelijan ei tarvitse kirjoittaa vastausta lainkaan, vaan vastaus annetaan manipuloimalla tehtävässä tehtävän luojan laatimaa kuvaa. Esimerkiksi tehtävässä vastaus voidaan antaa siirtämällä piste oikealle paikalle koordinaatistossa (STACK, 2021).

3.2 JSXGraph-JavaScript -kirjasto

STACK tukee avoimen lähdekoodin JSXGraph-JavaScript -kirjastoa, jonka avulla voidaan luoda tehtävään erilaisia geometrisiä kuvioita, kuten ympyröitä, kolmioita, suoria, funktioita yms. Itse tehtävässä kyseisiä kuvioita voidaan siirrellä ja muokata, jolloin kuviin saadaan mukaan uutena piirteenä myös dynaamisuutta (JSXGraph, 2020). JSXGraph-kirjastoa on hyödynnetty pääasiassa matemaattisissa tehtävissä, joissa tarvitaan graafista esitystä, kuten esimerkiksi funktion kuvaajaa (Nakamura, Higuchi, Ichikawa, Miyazaki, Yoshitomi, & Nakahara, 2019) tai vektoreita ja niiden laskemista (Tanskanen, 2017). Viime aikoina JSXGraphia on sovellettu myös fysiikan osa-alueilla, kuten mekaniikassa ja sähkötekniikassa, yhdessä STACK-järjestelmän kanssa. Kirjastoa on hyödynnetty sähkötekniikan osalta osoitinlaskennassa ja komponenttien sarjaan- ja rinnankytkentöjä määrittäessä (Klischat, Becker, Vasko, 2019).

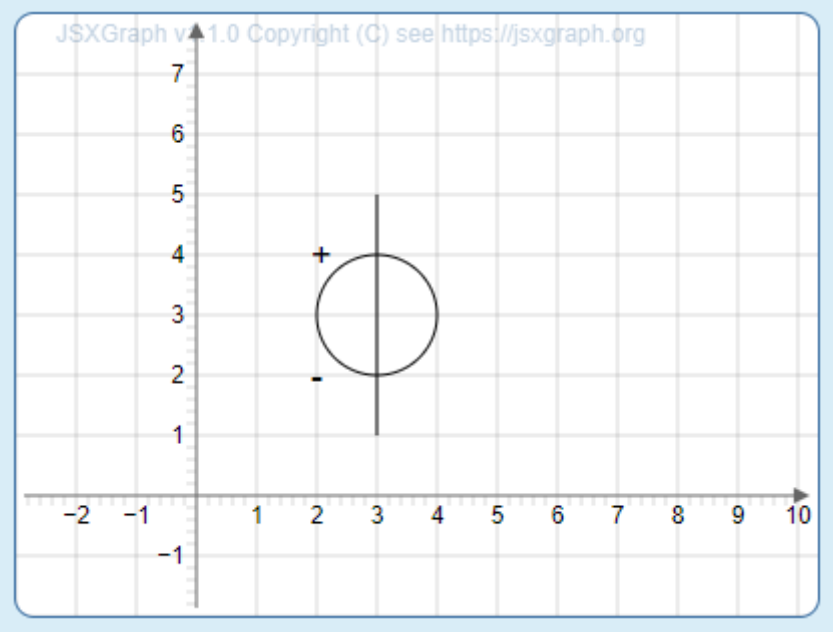
STACK-tehtävässä JSXGraphin avulla opiskelijaa voidaan esimerkiksi pyytää siirtämään voimavektorit oikeille paikoilleen tai täydentämään piirikaavioon oikeat komponentit. Tämän seurauksena pystytään kätevästi luomaan myös interaktiivisia tehtävätyyppejä, joissa opiskelija väistämättä *joutuu* työskentelemään enemmän itse kuvan kanssa. Algoritmissa 1 on esitetty yksinkertainen esimerkki JSXGraphin implementoinnista STACK-tehtävään.

```
<script type="text/javascript" src="https://jsxgraph.uni-
bayreuth.de/distrib/jsxgraphcore.js"></script>
<div id="mybox" class="jxgbox" style="text-align: left; width:
400px; height: 300px;"><br></div>
<script type="text/javascript">

        var brd = JXG.JSXGraph.initBoard('mybox',
{boundingbox:[-3,8,10,-2], keepaspectratio: true,axis:true,
showNavigation:false});
        var c1 = brd.create('circle',[[3,3],[3,4]],
{strokeColor:'#000000',strokeWidth:1, fixed:true});
        brd.create('line',[[3,1],[3,5]],
{strokeColor:'#000000', straightFirst:false,
straightLast:false, strokeWidth:1, dash:0});
        brd.create('text',[1.9,4, '+'], {fontSize:18,
fixed:true});
        brd.create('text',[1.9,2, '-'], {fontSize:18,
fixed:true});
</script>
```

Algoritmi 1. Esimerkki JSXGraph-elementin käyttöönottamisesta.

Koodi alkaa JSXGraph-kirjaston lataamisella nettisivulta. Seuraavaksi -elementin sisällä määritetään laatikko, jonka sisälle JSXGraphista saatavat kuviot sijoitetaan. Laatikon leveys ja korkeus määritellään pikseleinä. Tämän jälkeen määritellään ensimmäinen JSXGraph elementti muuttujalle *brd*, joka on tässä tapauksessa ”Bounding Box” eli kuvan koordinaatisto, jonka avulla määritellään muut elementit. Useimmat kuvioelementit edellyttävät jonkinlaisia koordinaatteja, joiden perusteella ne piirretään. Muuttujalle *c1* määritetään ympyräelementti (circle). Ympyrää määrittäessä annetaan kaksi koordinaattia. Ensimmäinen kertoo keskipisteen paikan ja toinen säteen suuruuden mittaamalla matkan keskipisteestä. Tässä esimerkissä säteeksi on määritelty yksi pituusyksikkö. Seuraavana elementtinä on määritelty viiva (line), joka kulkee ympyrän keskipisteen läpi. Viimeiset kaksi elementtiä ovat molemmat tekstejä (text), joissa ensimmäinen saa arvoksi ”+” ja toinen ”-”. Elementteihin voi lisätä erilaisia attribuutteja kuten värejä, viivojen muotoiluja, fontin kokoa yms.

Edellä mainittu algoritmi 1 muodostaa kuvassa 6 esitetyn jännitelähteen. JSXGraph soveltuu pääosin matematiikan tehtävien tehtailuun, mutta sitä voidaan hyödyntää

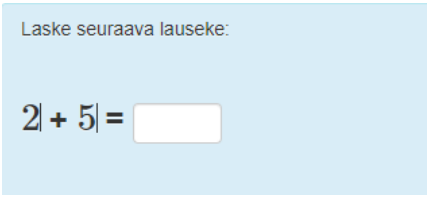


Kuva 6. Yksinkertainen JSXGraphin avulla luotu jännitelähde, joka koostuu neljästä elementistä.

myös muilla tieteenaloilla. Esimerkiksi sähkötekniikassa elementtien avulla voidaan mallintaa erilaisia piirikaavioita ja sen lisäksi niihin voidaan liittää dynaamisuutta ja satunnaisuutta. Elementtien koordinaattien tilalle voidaan nimittäin sijoittaa tehtävässä määriteltyjä muuttujia, joiden avulla voidaan räätälöidä lukuisia eri versioita samasta tehtävätyypistä.

3.3 Esimerkki STACK-tehtävästä

Seuraavaksi käydään läpi lyhyt esimerkki STACK:llä luodusta tehtävästä. Kuvassa 7 on esitetty yksinkertaisemmasta päästä oleva tehtävä. Kyseisessä esimerkissä ei ole hyödynnetty minkäänlaista grafiikkaa vaan sen tehtävänä on ainoastaan kysyä kahden eri luvun summaa. Tehtävässä esiintyvät luvut arvotaan etukäteen määritetystä listasta ja sen lisäksi ensimmäisen luvun kerroin arvotaan myös negatiivisen ja positiviisen väliltä. Tässä kyseisessä esimerkissä luvuiksi on siis arvottu ”2” ja ”5”, sekä ensimmäiselle luvulle kerroin ”1”. Mikäli opiskelija vastaa oikein, tehtävä tulostaa opiskelijalle palautteen, josta käy ilmi, että tehtävä on osattu ratkaista oikein. Mikäli taas opiskelija vastaa väärin, palaute kertoo hänelle, että tehtävän ratkaisussa on virhe.

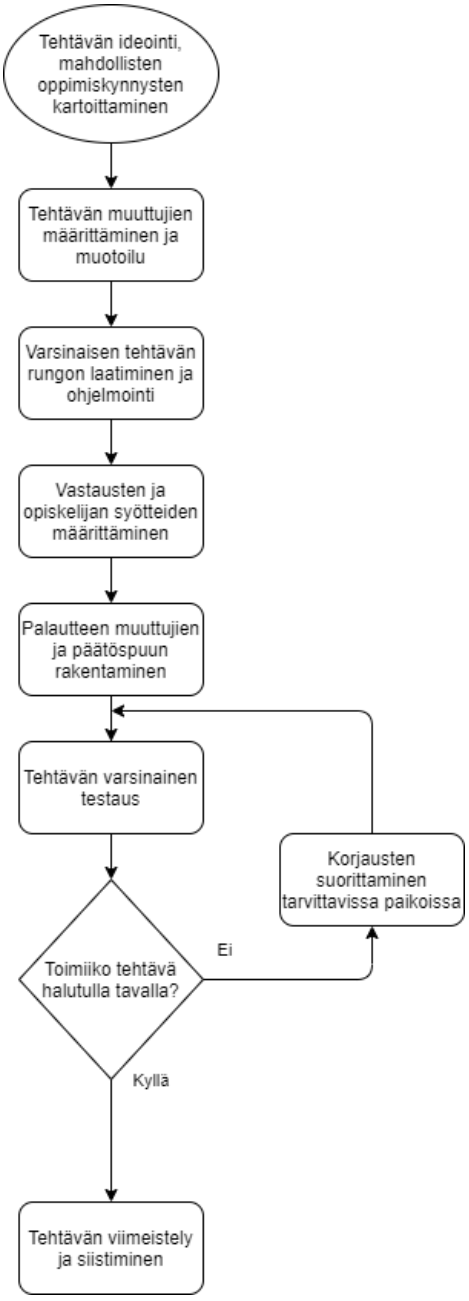


Kuva 7. Yksinkertainen STACK-tehtävä, jossa opiskelijan tehtävänä on laskea triviaalinen yhteenlasku annetuilla arvoilla.

Tämänkaltaisissa äärimmäisen yksinkertaisissa tehtävätyypeissä palautteen ei välttämättä tarvitse olla tuon tarkempaa ja monimutkaisempaa. STACK kuitenkin mahdollistaa monimutkaisemman palautteen antamisen, jossa pyritään etsimään sitä kohtaa tehtävästä, jossa opiskelija teki virheen. Esimerkiksi, jos opiskelija summaamisen sijaan vähentäisikin toisen luvun ensimmäisestä luvusta, voitaisiin tälle rakentaa tarkastus, joka taas antaisi paljon yksityiskohtaisemman palautteen.

3.4 Tehtävän luominen

STACK-tehtävän muodostuminen valmiiksi harjoitukseksi, jonka opiskelija voi sitten suorittaa, on jokseenkin monimutkainen ja työläskin prosessi. Työvaiheita on monta erilaista. Ne kaikki on käytävä läpi ja useimmiten useampaan otteeseen riippuen tehtävän vaativuudesta toteuttamisen suhteen, sillä korjattavaa ja hiomista löytyy pitkään. Kuvassa 8 on esitetty tehtävän luonnin eri vaiheet vuokaaviona. Tehtävän laatiminen alkaa suunnitteluvaiheesta, jossa esim. huomattuun puutteeseen oppimisessa yritetään keksiä apukeinoja. Kun tehtävän tavoite on selvillä, aloitetaan tehtävän rakentaminen eli ohjelmointi. STACK:ssä tehtävän ohjelmointi on jaettu useampaan osa-alueeseen: muuttujien määrittely, tehtävän runko ja päätöspuu. Kun edelliset vaiheet on suoritettu, seuraa testausvaihe, jossa pyritään löytämään vielä viimeiset virheet ja palataan mahdollisesti korjaamaan niitä.



Kuva 8. STACK-tehtävän laadinnan vaiheet.

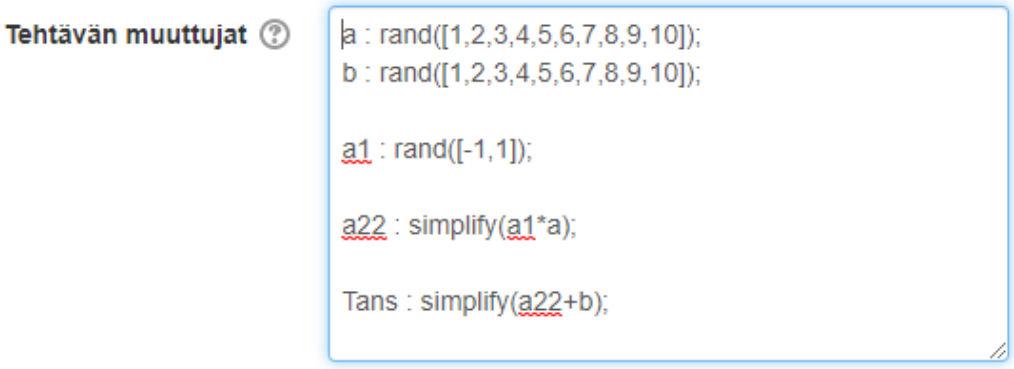
Testausta pitää suorittaa myös muiden ohjelmointivaiheiden aikana mikäli siihen ilmenee tarvetta. Lopuksi suoritetaan vielä viimeistelyvaihe, jossa poistetaan esimerkiksi testauksessa käytettyjä merkintöjä. Tämän jälkeen tehtävä voidaan antaa opiskelijan ratkaistavaksi. Ensimmäinen opetuskierros on luonteeltaan protokierros, josta saadaan kerättyä tärkeä opiskelijapalaute. Seuraavissa alaluvuissa on käyty yksityiskohtaisemmin läpi STACK-tehtävän laadinnan vaiheet.

3.4.1 Tehtävän suunnittelu

Tehtävän rakentaminen alkaa hahmottelemalla ja tekemällä suunnitelma, jonka pohjalta ohjelmoidaan varsinainen tehtävä. Suunnitteluvaiheessa pyritään ideoimaan tehtävää jonkin tietyn tarpeen mukaan. Esimerkiksi on huomattu, että kurssille olisi hyvä saada kertausta Kirchhoffin jännitelaista tai tietynkaltainen tehtävä puuttuu kokonaan. Tämän jälkeen alkaa mietintä varsinaisesta toteutuksesta. Antaako opiskelija mahdollisen ratkaisun esimerkiksi yhtälön muodossa vai sisältääkö tehtävä laskettavan osuuden? Tehtävä voi olla myös hieman pelin kaltainen, jossa ratkaisija joutuu esimerkiksi siirtämään komponentteja oikealle paikalleen. Kun on päätetty, mitä tehtävä sisältää ja millä tavalla se toteutetaan, siirrytään seuraavaan vaiheeseen.

3.4.2 Muuttujien määrittäminen ja muotoilu

STACK:ssä ensimmäiseksi määritellään tehtävän nimi ja mahdollinen kategoria, jonne se sijoitetaan. Tämä helpottaa tehtävän löytämistä myöhemmin. Tämän jälkeen ympäristöstä löytyy ”Tehtävän muuttujat” -kenttä johon voidaan määrittää tehtävässä käytettävät muuttujat. Kenttä tukee CASText-muotoilua, mikä mahdollistaa muuttujien käytön myöhemmin varsinaisen tehtävän luonnissa. Käytettävät muuttujat voivat saada arvoiksi esimerkiksi lausekkeita, yksittäisiä numeroita, luetteloita jne. Tässä kohdassa yleensä määritellään myös oikean ratkaisun yhtälö, jota käytetään myöhemmin vastauspuussa oikean palautteen antamiseksi. Kuvassa 9 on esitetty alaluvun 3.3 esimerkkitehtävän muuttujat. Muuttujat ”a” ja ”b” ovat tehtävässä esiintyvät luvut, jotka saavat arvon yhdestä kymmeneen satunnaisena aina kun tehtävä käynnistetään. Yhdeksi muuttujaksi on määritetty myös ”a1”, joka taas saa arvon -1 tai 1. Tätä käytetään ensimmäisen luvun kertoimena muuttujana ”a22”, joten oikea ratkaisu voi myös sisältää negatiivisia arvoja. Muuttuja ”Tans” on tehtävän ratkaisuyhtälö, joka antaa oikean vastauksen.



Kuva 9. Kuvassa 7 esitetyn STACK-tehtävän muuttujien määrittely.

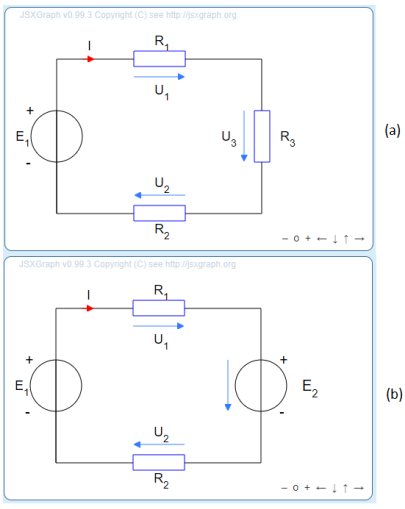
3.4.3 Varsinainen tehtävä ja kysymysteksti

Tehtävän rakenne ja ulkoasu määritellään osiossa ”kysymysteksti”. Tähän osioon lisätään kaikki se, mitä tehtävässä halutaan näyttää opiskelijalle. Se voi sisältää tehtävänannon, erilaisia kuvia tai alaluvussa 3.4.2 määriteltyjä muuttujia. Osion täytyy myös sisältää pakolliset elementit: yksi tai useampi syötekenttä ja syötteentarkistus. Syötekenttiin opiskelijat antavat vastauksensa. Tehtävätekstiä luotaessa voidaan hyödyntää myös jo aikaisemmin määritettyjä muuttujia. Kuvan 5 esimerkissä luvut 2 ja 5 on arvottu tietyille muuttujille, jotka tuodaan opiskelijoille näkyviin tehtävätekstissä. Koodissa referoidaan siis näihin muuttujiin. Tehtäväteksti kirjoitetaan pääasiassa CASTextinä. CASText on HTML-koodia, johon on implementoitu LaTeX ja CAS -toimintoja. Sen seurauksena teksti tukee erilaisia HTML-elementtejä, kuten taulukoita tai JavaScriptiä.

Joillekin tehtävätyypeille on välttämätöntä, että opiskelijalle annetaan myös visuaalista apua tehtävän hahmottamiseksi, jotta tehtävän ratkaiseminen olisi helpompaa. Esimerkiksi fysiikkaan keskittyvät tehtävät vektoreineen ja tilannekuvauksineen tarvitsevat jonkinlaisen kuvan selittämään tarkasti tilannetta. Kuva voidaan tuoda tehtävään esimerkiksi hakemalla se joltain nettisivulta. STACKiin voi myös tallentaa sarja omia kuvia, jotka voidaan satunnaisesti arpoa. Tämän seurauksena tehtävän kuva ja tilanne ovatkin aina erilaisia, kun opiskelija suorittaa sitä. Tällöin oikean ratkaisun saamiseksi ulkoamuistaminen ei välttämättä riitäkään, vaan tämä toimintatapa pakottaa

opiskelijan ajattelemaan erilaisia vaihtoehtoja. STACK tukee myös JSXGraph-JavaScript -kirjastoa. Kirjaston avulla voidaan luoda geometrisiä elementtejä, kuten palloja, neliöitä tai nuolia ja niiden avulla voidaan rakentaa tilanteesta kuva. Etuna edellä mainittuihin staattisiin kuviin on se, että yksittäisiä JSXGraph-elementtejä voidaan muuttaa helposti koodin sisällä. Lisäksi opiskelijalla on mahdollisuus liikutella kyseisiä elementtejä, mikä taas mahdollistaa uusien tehtävätyyppien luomisen.

Esimerkiksi sähkötekniikassa piirikaavio voidaan esittää JSXGraph-elementtien avulla. Sähköpiirissä esiintyvät elementit voidaan arpoa aina uudelleen, kun tehtävä käynnistetään, jolloin tehtävän oikea vastaus yleensä muuttuu eikä ulkoaopetteleminen onnistu. Myös sähköisten komponenttien paikat voidaan muuttaa arpomalla ja sen lisäksi voidaan muuttaa virtojen ja jännitteiden referenssisuuntanuolet. Tämän seurauksena samaiseen piiriin voi löytyä useita eri tilanteita. Kuvassa 10 on esitetty saman tehtävän eri versioita. Ensimmäisellä suorituskerralla kohdassa 10a komponentiksi on arvottu ja luotu vastus R_3. Toisella kerralla taas vastuksen tilalle onkin luotu jännitelähde E_2 (10b). Näillä keinoilla voidaan luoda useita eri versioita ja tilanteita suhteelisen vaivattomasti.



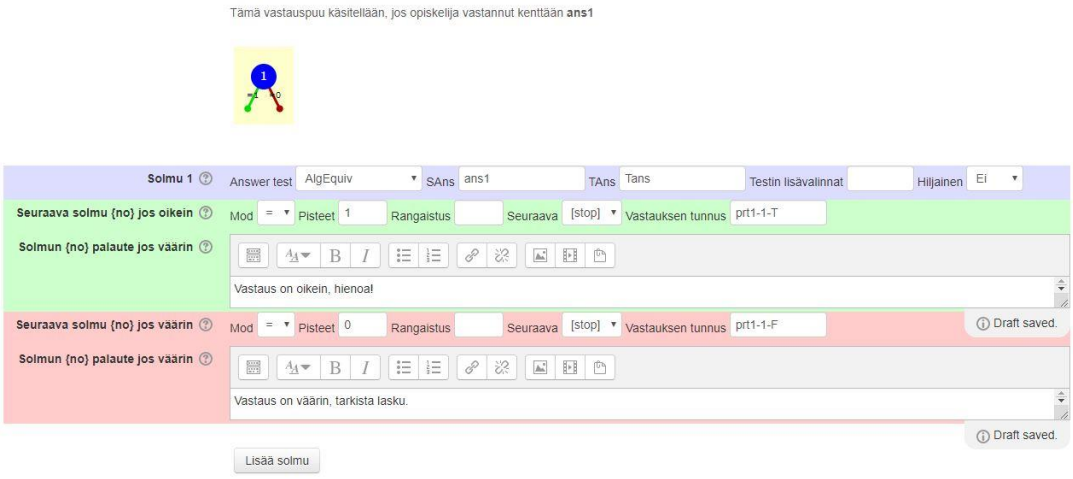
Kuva 10. JSXGraphilla luotu piirikaavio samasta tehtävästä, kun komponentiksi on arvottu vastus R_3 (a) ja jännitelähde E_2 (b).

Jos taas jännitteen referenssinuolien suuntia muutettaisiin ja tehtävänä olisi kirjoittaa piirin jänniteyhtälö, olisi kuvan 8 triviaalisessa piirissä jo useampi erilainen tilanne. Näin siis elementtien paikkojen tai suuntanuolien arpominen mahdollistaa useiden samantasoisten tehtävien laatimisen vaivattomasti.

3.4.4 Vastauspuu

Jotta opiskelijan antama vastaus tulkitaan ja arvioidaan oikein, tarvitaan vastauspuu, jossa vastausta käsitellään. Päällimmäinen idea puussa on annetun vastauksen vertaileminen erilaisiin opettajan tunnistamiin tilanteisiin. Esimerkiksi mikäli vastaus todetaan vertailussa oikeaksi, annetaan siitä täydet pisteet ja palautteen avulla kerrotaan se myös opiskelijalle. Huomattavasti tärkeämpi ominaisuus on kuitenkin virheellisen vastauksen analysoiminen ja siihen sopivan palautteen antaminen.

Päätöspuu koostuu ns. solmuista, jotka toimivat vertailupisteinä. Solmuissa tarkastellaan annettua vastausta johonkin aikaisemmin määriteltyyn tilanteeseen. Mikäli vastaus on oikein, annetaan siitä pisteet ja joko lopetetaan tehtävän tarkastus tai käsitellään mahdolliset muut vastauskentät. Kuvassa 11 on esitetty yksinkertaisen tehtävän kaikista yksinkertaisin päätöspuu, joka koostuu vain yhdestä solmusta.



Kuva 11. Yksinkertaisen vastauspuun näkymä.

Kuvan 11 yllimmällä rivillä suoritetaan halutun muuttujan vertaaminen toiseen muuttujaan. Tässä esimerkissä käytetään AlgEquiv-testiä (algebraic equivalent), joka tarkastaa ovatko muuttujat algebrallisesti samoja ja se on määritelty kohdassa ”Answer test”. Kohtaan ”SAns” (Student answer) sijoitetaan tehtävätekstissä määritelty opiskelijan antama syöte, joka on tässä tapauksessa muuttuja ”ans1”. Kohtaan TAns (Teacher Answer) taas sijoitetaan haluttu vertailukohde; muuttuja ”Tans”. Tässä yksinkertaisessa esimerkissä vertaillaan siis opiskelijan antamaa vastausta vain haluttuun oikeaan vastaukseen. Mikäli testi antaa tulokseksi ”tosi”, tulostetaan opiskelijalle vihreällä alueella sijaitsevan tekstikentän palaute ja pisteytetään tehtävä kohdassa ”Pisteet” määritetyllä määrällä eli yksi piste. Jos taas testi antaa tulokseksi ”epätosi”, tulkitaan opiskelijan vastaus vääräksi. Opiskelijalle tulostetaan punaisella alueella sijaitsevan tekstikentän sisältö ja annetaan tehtävästä esimerkin mukaisesti nolla pistettä. Edellä mainittu yksinkertainen päätöspuu voi olla jo riittävä, mikäli tehtävä on suhteellisen triviaali, kuten edellä esitetty kahden muuttujan summa (kuva 7).

Vastauspuusta voidaan kuitenkin rakentaa tarkempi ja monimutkaisempi lisäämällä enemmän solmuja. Kuvassa 12 on esitetty kuvan 7 vastauspuu, mutta siihen on lisätty yksi solmu lisää. Solmut on yhdistetty siten, että mikäli opiskelijan vastaus tulkitaan ensimmäisessä solmussa vääräksi, siirrytään toiseen solmuun. Uudessa solmupisteessä tarkastellaan, onko opiskelijan antama vastaus sama kuin muuttuja ”Ero1”. Muuttuja on määritelty vastauspuun ”Palautteen muuttujat” -kentässä siten, että se saa arvoksi ensimmäisen ja toisen luvun erotuksen. Kentässä voidaan määrittää uusia muuttujia, joiden avulla voidaan laskea uusia asioita käyttäen opiskelijan antamaa vastausta ja muita muuttujia (STACK, 2020).

Lopuksi annetaan vielä opiskelijalle palaute riippuen vertailun tuloksesta. Mikäli huomataan, että onkin laskettu muuttujien erotus summan sijasta, kerrotaan se opiskelijalle ja näytetään oikeaa vastausta. Mikäli taas solmussa kaksi ei vieläkään tunnisteta virheen laatua, annetaan palautteena vain tieto väärästä vastauksesta tai

Kuva 12. Kahdesta solmusta koostuva vastauspuu.

sitten puuta mahdollisesti jatkettaisiin vieläkin pidemmäksi ja yksityiskohtia enemmän sisältäväksi.

3.4.5 Testaus ja viimeistely

Jotta tehtävän oikea toiminta voidaan tarkastaa, täytyy tehtävää testata. Yleensä erillistä testausvaihetta ei ole, vaan toimivuutta pyritään tarkistamaan useiden pienien muokkausten avulla jo tehtävää luotaessa. Mahdollisia virheitä pyritään selvittämään tarkastelemalla muuttujien toimintoja säätämällä niitä näkyviksi. STACKin mukana tuleva editori ei itsessään anna palautetta, mikäli esimerkiksi JSXGraph-kuvaa luotaessa esiintyy virhe. Tämän takia tehtävää kannattaakin luoda pieni pala kerrallaan varsinkin siinä tapauksessa, jos tarkoituksena on kokeilla jotain uutta.

Kun tehtävä todetaan toimivaksi erilaisten testauksien jälkeen, suoritetaan vielä viimeistely. Viimeistelyssä lähinnä muokataan ja viimeistellään tehtävän ulkoasu halutuksi ja esimerkiksi standardinmukaisiin merkintöihin kiinnitetään huomiota.

Koodissa esiintyvät turhat merkinnät ja testerit siivotaan myös pois. Näiden vaiheiden jälkeen tehtävä on valmis kokeiltavaksi ensi kertaa kurssilla ja mahdollisten korjausten jälkeen, se voidaan lisätä osaksi materiaalia.

3.4.6 Opettajan osaamisvaatimukset tehtävän luomisen kannalta

STACK-tehtävien luominen voi olla hyvin yksinkertaista tai todella monimutkaista. Tähän vaikuttaa luotavan tehtävän luonne ja kehittäjän aikaisempi ohjelmointiosaaminen. Editori auttaa kehittäjää jonkin verran piilottamalla ja automaattisesti lisäämällä tarvittavat pakolliset koodin palaset tai visualisoimalla päätöspuun rakennetta, mutta yleisesti sanottuna lopulta ilman ohjelmointia tehtävänluomisesta ei selviä.

Yksinkertaisten tehtävien, kuten alaluvussa 3.3. esitetyn summa-tehtävän ohjelmointi ei vaadi juuri mitään muuta kuin oikean Maxima-syntaksin tuntemista. STACK-järjestelmän dokumentaatio tarjoaa tarvittavat ohjeet alkuun pääsemiseen, joten varsinaista aikaisempaa ohjelmointikokemusta sen osalta opettaja ei tarvitse.

Tehtävää luodessa haastavimmaksi osuudeksi muodostuu poikkeuksetta hyvän vastauspuun luominen ja se harvemmin on riippuvainen ohjelmointikokemuksesta, joskin siitä on apua. Opiskelijan vastauksen tarkastaminen ja mahdollisten virheiden tunnistaminen ennakolta edellyttää sitä, että kehittäjä tiedostaa itse minkälaisia virheitä kyseisessä tehtävässä on mahdollista tehdä. Tämän tiedon avulla on mahdollista luoda puuhun tarkastuslogiikka, joka käy läpi opiskelijoiden oppimisen kannalta kaikista kriittisimmät virheet. Mitä enemmän erilaisia tilanteita tarkastellaan, sitä monimutkaisemmaksi logiikka muodostuu.

Mikäli tehtävään haluaa sisällyttää jotain visualisointia, kuten kuvia, on niitäkin helppo upottaa tehtävään sisälle. Editorissa voidaan upottaa kuva tekstin sekaan samalla tavalla kuin esim. Moodlessa. Haastavammaksi ohjelmointi muuttuu, jos aletaan luomaan tehtäviä, joissa halutaan kuvan muuttuvan satunnaisesti jokaisella ratkaisukerralla. Tällöin kuvan muuttaminen täytyy ottaa huomioon esimerkiksi oikean vastausyhtälön

ohjelmoinnissa. Toisaalta, jos vastausyhtälö muuttuu, täytyy se huomioida myös vastauspuussa opiskelijan vastausta tarkasteltaessa.

JSXGraph-kirjaston sisällyttäminen tehtävään on suhteellisen vaivatonta ja siihenkin löytyy hyvin tietoa sekä STACK-järjestelmän omasta dokumentaatiosta ja myös kirjaston kehittäjien kotisivulta (JSXGraph, 2020). STACK-versiosta riippuen sen käyttöönotto eroaa hieman, mutta ei vaikuta monimutkaisuuteen. Mikäli taas kirjastoa halutaan hyödyntää siten, että opiskelija antaa vastauksensa esim. liikuttelemalla jotain elementtiä, muuttuu tehtävän ohjelmointi huomattavasti haastavammaksi ja ilman aikaisempaa kokemusta JavaScriptistä tai HTML-ohjelmoinnista, voi tehtävän luominen olla todella vaikeata.

Tiivistettynä voidaan ajatella, että mitä yksinkertaisempi tehtävä, niin sitä yksinkertaisempi se on myös ohjelmoida. Triviaalisten tehtävien tekemiseen ei tarvitse osata juuri ollenkaan ohjelmointia ja sellaisten tekemiseen löytyy hyvin tietoa ja ohjeistusta jo pelkästään STACK-dokumentaatiosta. Monimutkaisten tehtävien kohdalla aletaan jo kuitenkin vaatimaan ohjelmointikielten tuntemusta huomattavasti enemmän.

4 STACK-järjestelmän hyödyntäminen Vaasan yliopiston sähkötekniikan opetuksessa

Tässä luvussa tarkastellaan, miten STACK-järjestelmää voidaan hyödyntää teoreettisen sähkötekniikan opetuksessa ja tarkemmin ilmaistuna piirianalyysin parissa. Luvun keskeisiä aiheita ovat:

- Piirianalyysi Vaasan yliopistossa.
- STACK-järjestelmän ja JSXGraphin hyödyntäminen piirianalyysissä.
- Piiritehtävien visualisointi järjestelmässä.
- Uusien interaktiivisten tehtävien hyödyntäminen.

STACK-järjestelmää on hyödynnetty aikaisemmin Vaasan yliopistossa matematiikan kursseilla, kuten piirianalyysiinkin liittyvässä lineaarialgebrassa. Hyvien kokemusten perusteella päädyttiin kokeilemaan, kuinka hyvin järjestelmää voidaan soveltaa myös sähkötekniikassa. Piirianalyysi on yleisesti opiskelijoiden keskuudessa mielletty haasteelliseksi kokonaisuudeksi. Se yhdistää matematiikkaa, fysiikkaa ja loogista päättelyä, ja tehtävät saattavat olla monimutkaisia. Pienikin virhe voi huomaamatta muuttaa lopputulosta dramaattisesti, joten kaiken päälle ratkaisun laadinnassa pitää olla jatkuvasti tarkkana. Tätä helpottamaan on monien vuosien jälkeen Vaasan yliopistossa päädytty STACK-järjestelmän implementointiin muiden kurssimateriaalien jatkeena.

Toistaiseksi erilaisia STACK-tehtäviä on luotu Piirianalyysi A -kurssille, joka käsittää piirianalyysin perusteet AC- ja DC-piireissä. Ensimmäisiä tehtäviä ehdittiinkin jo kokeilla vuoden 2019 keväällä järjestetyllä kurssilla. Tehtäviä alettiin kehittää 2018 syksyllä, joten erilaisia tehtäviä oli tuolloin niukasti ja vain suurimmat opiskelijoille ongelmia aiheuttaneet osa-alueet olivat katettuna. Vuoden 2020 kurssille tehtäviä kehitettiin lisää kattamaan loputkin tarpeelliset alueet. Sen lisäksi kurssille kehiteltiin myös puhtaasti matematiikan tehtäviä, joissa käsitellään kompleksilukuja ja niiden hyödyntämistä etenkin vaihtosähkön puolella. Tehtävien pääasiallinen tarkoitus onkin siis olla

opiskelijalle apuväline muun kurssimateriaalin lisäksi. Opiskelijat eivät ole sidottuja aikaan tai paikkaan, sillä tehtäviä voidaan tehdä omalta laitteelta missä ja milloin tahansa, kunhan laitteella on selain ja yhteys internettiin. Tehtävien tarkastus ja oikean palautteen antaminen opiskelijalle toteutetaan automaattisesti, joten tehtävän palautettuaan opiskelija saa samalla välittömästi tiedon siitä, menikö tehtävä oikein ja mahdollisesti myös tarkennuksen, mikä meni väärin. Tässä luvussa käydään läpi tarkemmin Piirianalyysi A -kurssille luotuja tehtäviä, niiden pääkohtia ja ideoita tehtävien takana.

4.1 Piirianalyysi

Piirianalyysi luo perusteet ja edellytykset sähkötekniikan piirien ratkaisemiselle eri sovelluksissa. Monimutkaisten laitteiden, kuten sähkömoottoreiden, kehitystoiminta pohjautuu vankasti perusteiden ymmärtämiseen ja niiden soveltamiseen. Samalla tavalla esimerkiksi sähkönsiirto ja siihen liittyvät tekniikat ja laitteet pohjautuvat perusasioihin, jotka eivät muutu. Mikäli opiskelijan perustiedot ovat heikot, on myös monimutkaisten sovelluksien toiminnan ymmärtäminen vaikeampaa jollei jopa mahdotonta. Tästä syystä onkin äärimmäisen tärkeää, että opiskelijoille luodaan järkkymätön ymmärrys aivan sähkötekniikan perusteista, jota voidaan soveltaa myöhemmillä kursseilla ja valmistumisen jälkeisessä työelämässä (Valtonen & Lehtovuori , 2011, 2017).

Piirianalyysi perustuu erilaisten mallien kytkentöjen analysoimiseen. Sähkötekniset laitteet voidaan esittää virtapiirinä sähköisten perusmallien, kuten resistanssien, kapasitanssien ja induktanssien avulla. Virtapiiriä analysoitaessa hyödynnetään erilaisia laskennallisia menetelmiä riippuen piirin rakenteesta. Kurssin Piirianalyysi A tavoitteena on saada paitsi opiskelijat ymmärtämään erilaisten komponenttien toiminta tasa- ja vaihtovirtapiireissä niin myös erilaisten ratkaisumenetelmien hyödyntäminen. Jotta näiden asioiden oppiminen olisi sujuvaa, vaaditaan opiskelijoilta myös jonkin verran matemaattista osaamista. Esimerkiksi vaihtovirtapiirien ratkaiseminen ilman kompleksilukujen osaamista on käytännössä mahdotonta, sillä se on tärkeä osa osoitinlaskentaa. Myös erilaisten yhtälöiden kirjoittaminen virtapiirille perustuen

Kirchhoffin ja Ohmin lakeihin ja niiden yhdisteleminen tuottaa vaikeuksia. Kaikki ratkaisumenetelmät, joita kurssilla käydään, perustuvat juuri näihin lakeihin. Mikäli ymmärrys kyseisistä laeista on puutteellista, on muiden ratkaisumenetelmien, kuten silmukka- ja solmumenetelmien ymmärtäminen ja opetteleminen haasteellista. Tämä voi johtaa nopeasti eräänlaiseen ”tuhon kierteeseen”, jonka seurauksena opiskelija turhautuu ja motivaatio romahtaa. Tämä taas voi johtaa ulkoaopetteluun, joka ei tuota haluttua lopputulosta, kuten luvussa 2 mainittiin. Motivaation ylläpito onkin yksi tärkeimmistä edellytyksistä oppimisen onnistumiselle.

Piirianalyysin tehtävät sisältävät poikkeuksetta piirikaavion johon tehtävässä esitetyt kysymykset ja alkuarvot liittyvät. Piirikaavio toimii eksaktin viestinnän apuvälineenä niin opiskelijalle kuin opettajallekin. Jotta ei-triviaali piirien ratkaiseminen ja tarvittavien yhtälöiden muodostaminen olisi ylipäätänsä mahdollista, täytyy piirikaavioon merkitä referenssisuuntia kuvaavat merkinnät. Virtojen ja jännitteiden suunnat piirissä voidaan poikkeuksetta aluksi valita vapaasti ja niiden pohjalta rakennetaan tarvittavat yhtälöt. Referenssisuuntien merkinnät toimivat myös apuvälineenä opettajalle tehtävän tarkastuksessa. Mikäli merkinnät puuttuvat tai ovat puutteellisia, on tehtävän tarkastaminen mahdotonta (Vesapuisto, 2004). Tämä voi olla myös syynä heikkoihin tuloksiin joidenkin laskusuoritusten kohdalla.

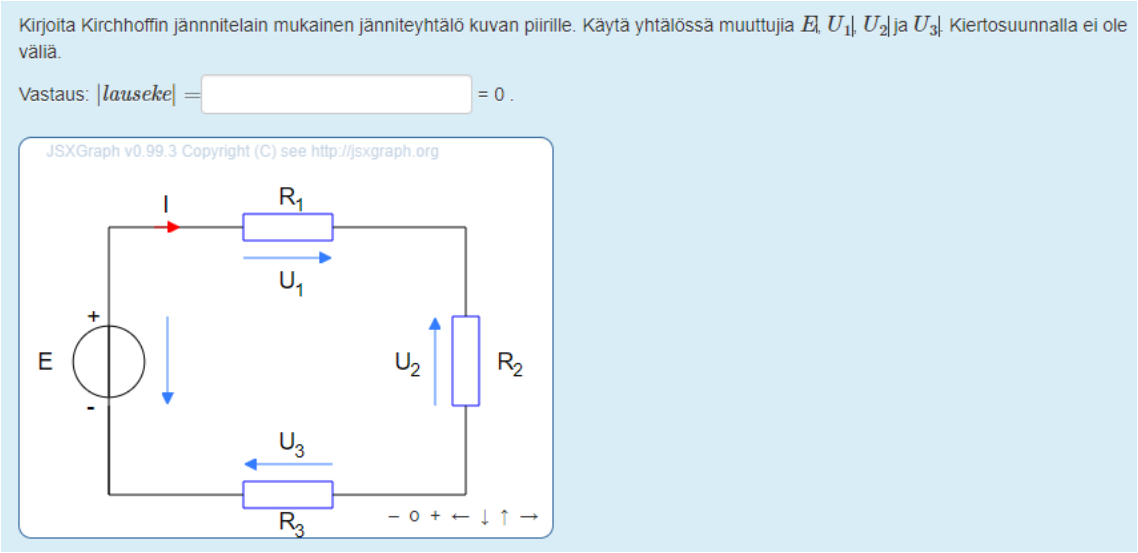
Toinen tärkeä huomio on tarvittavien yhtälöiden kirjoittaminen Kirchhoffin ja Ohmin lakien mukaisesti, jotka perustuvat virtapiiriin merkittyihin referenssisuuntiin. Kirchhoffin ja Ohmin lait yhdessä luovat yhden tärkeimmistä oppimistavotteista, sillä lopuksi kaikki perustuvat näihin. Tämän seurauksena onkin luontevaa luoda opiskelijoille tehtäviä, joissa toistojen kautta pyritään opettelemaan yhtälöiden kirjoittamista piireihin, joissa on jo referenssisuunnat merkittynä valmiiksi. Tämänkaltaiset tehtävät ovat olleet kurssin ensimmäisiä laskuharjoituksia ja niissä on käyty läpi Kirchhoffin ja Ohmin lait. STACK-järjestelmän avulla voidaan kuitenkin luoda vastavaanlaisia lukuisia erilaisia automaattisia harjoituksia, joissa suuntanuolet arvotaan aina, kun opiskelija suorittaa tehtävän uudestaan. Tällä tavoin ratkaistavaksi tulee joka yrityksellä uusi tilanne ja

ulkoaopetteleminen ei toimikkaan, mikä taas pakottaa opiskelijan ajattelemaan ja oppimaan tarvittavat säännöt.

4.2 Staattisten sähkötekniikan tehtävien luominen

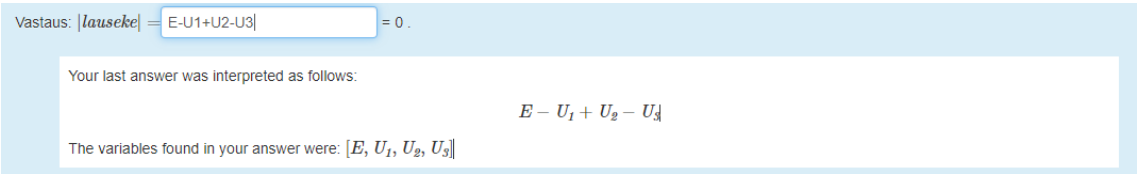
Piirianalyysi A -kurssille tuotetut STACK-tehtävät keskittyvät enimmäkseen yhtälöiden muodostamiseen, sillä valmiiseen yhtälöön arvojen sijoittaminen ja sen naputteleminen laskimeen ei pitäisi tuottaa opiskelijalle vaikeuksia. Yhtälön muodostaminen on tärkein tavoite, sillä sen avulla voidaan ratkaista piiristä mitä vain. Luodut tehtävät etenevät aihepiireittäin kurssin muun materiaalin kanssa, joten nekin alkavat Kirchhoffin ja Ohmin lakien painottamisella ja yhtälön muodostamisesta niiden avulla.

Tarkastellaan seuraavaksi kuvassa 13 esitettyä tehtävää. Tämä tehtävä esiteltiin ensimmäisenä kurssin opiskelijoille. Tarkoituksena on siis kirjoittaa Kirchhoffin jännitelain mukainen yhtälö yhden silmukan piirille, johon on valmiiksi merkitty referenssisuunnat. Opiskelija saa ratkaisussaan kiertää piirin kumpaan suuntaan tahansa ja tämä tulee huomioida myös tehtävän tarkastuksessa.



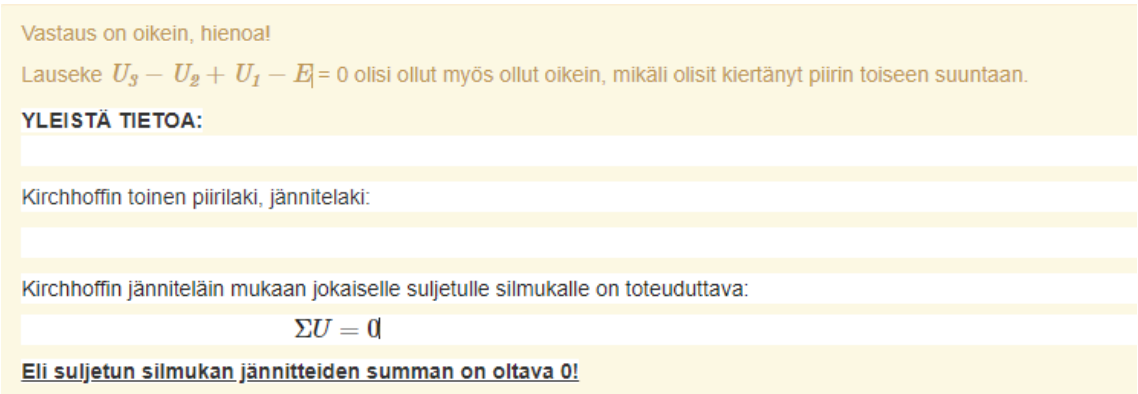
Kuva 13. Piirianalyysi A -kurssilla oppilaille esitetty STACK-tehtävä yhden silmukan piiristä.

Käyttäen muuttujia E, U_1, U_2 ja U_3, oikea vastaus kirjoitetaan kuvan 14 mukaisesti tehtävän syötekenttään. Tehtävä sisältää pakollisen validointielementin, joka näyttää, miten opiskelijan antama vastaus tulkitaan. Tässä tapauksessa STACK tunnistaa, että kirjaimen jälkeen tuleva numero on alaindeksi.



Kuva 14. Yksi oikea vastaus kuvan 11 tehtävään kirjoitettuna tehtävän syötekenttään.

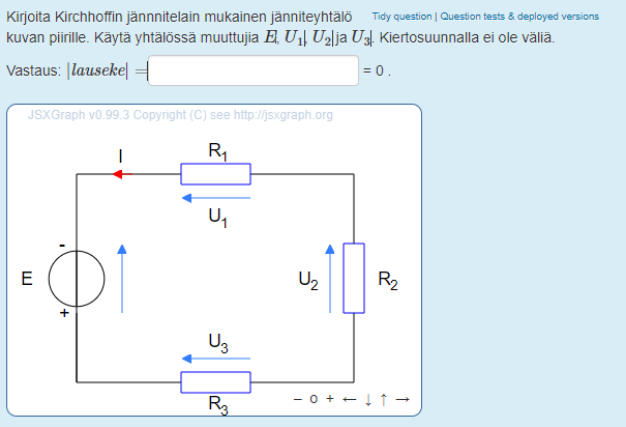
Mikäli opiskelija vastasi oikein kyseiseen tehtävään, tulostetaan hänelle kuvassa 15 esitetty palaute. Palautteesta käy ilmi, että annettu vastaus on oikein ja sen lisäksi opiskelijalle näytetään vaihtoehtoinen ratkaisu, jossa piiri on kierretty eri suuntaan. Kyseinen palaute sisältää myös ”Yleistä tietoa”-osion, joka tulostetaan riippumatta siitä, oliko vastaus oikein vai väärin.



Kuva 15. Oikean vastauksen luoma palaute, jossa on vaihtoehtoinen ratkaisu.

Tämän kaltaisia suhteellisen nopeasti ratkaistavia tehtäviä ei opiskelijoille esitetty erikseen vaan niistä rakennettiin useamman tehtävän sarja, jotta aiheen oppiminen vahvistuisi muuttuvien piirikaavioiden avulla. Sen lisäksi opiskelijoiden täytyi saada täysin oikeat vastaukset, sarjan jokaisesta tehtävästä, jotta suoritusta voitiin pitää hyväksyttynä. Kuvan 13 esimerkkiä voidaan pitää sarjan ensimmäisenä tehtävänä.

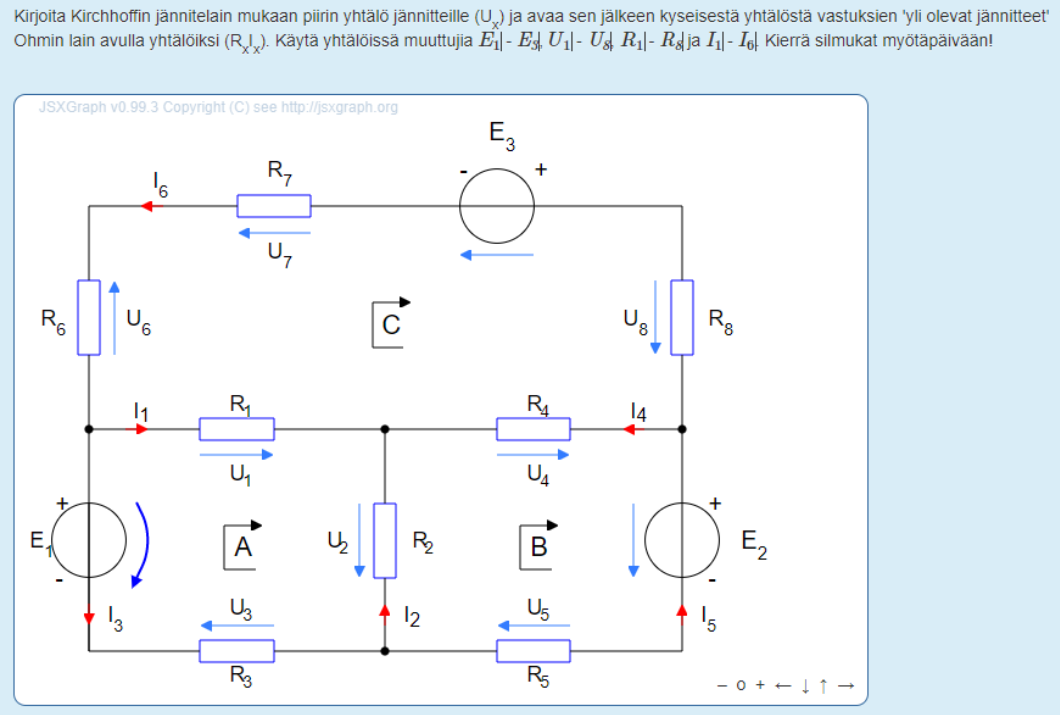
Kuvassa 16 esitetty tehtävä taas voisi olla sarjan toinen, joka näytetään opiskelijalle kun hän on antanut ensin vastauksen edelliseen tehtävään.



Kuva 16. Tehtäväsarjan toinen tehtävä, joka on periaatteiltaan samanlainen kuin ensimmäinenkin, mutta referenssisuuntia on muutettu.

Jos verrataan sarjan ensimmäistä ja toista tehtävää (kuvat 13 ja 16) keskenään voidaan huomata, että tehtävät eroavat toisistaan referenssisuuntien muutoksen takia. Tämän seurauksena myös vastaus tehtävään on eri. Myös sarjan kolmannessakin tehtävässä referenssisuunnat olisivat taas eriävät jne. Kuvien kaltaisessa yhden silmukan trivialisessa piirissä erilaisia vastauksia ei kuitenkaan kovin montaa ole, mutta siinä esitetyt säännöt pätevät myös monimutkaisempiinkin piireihin. Esimerkiksi kolmen silmukan piirissä, jossa jokaisessa haarassa on vähintään yksi komponentti, erilaisia vastauksia on huomattavasti enemmän jolloin ulkoa muistaminen vaikeutuu ja opiskelijan on pakko soveltaa yhden silmukan piirissä oppimaansa taktiikkaa.

Kuvassa 17 on esitetty kurssilla käytetty tehtävä, jossa piirikaaviosta löytyy kolme silmukkaa ja lukuisia muita komponentteja, joilla kaikilla on oma satunnainen referenssisuuntanuolensa. Tämän lisäksi uutena elementtinä on myös pakotetut kiertosuunnat ja lisäksi tehtävässä pitää myös kirjoittaa silmukoille jänniteyhtälöt Ohmin lain mukaisesti. Jänniteyhtälöiden muodostaminen toimii kuitenkin samalla tavalla kuin yksinkertaisimmassa piirissä (Ellonen, Vesapuisto & Vekara, 2020).



Kuva 17. STACK-tehtävän piirikaavio kolmella silmukalla (Piirianalyysi A 2020).

Tämän tehtävän osattuaan voidaan siirtyä toisiin ratkaisumetodeihin, kuten silmukka- ja solmupistemenetelmiin.

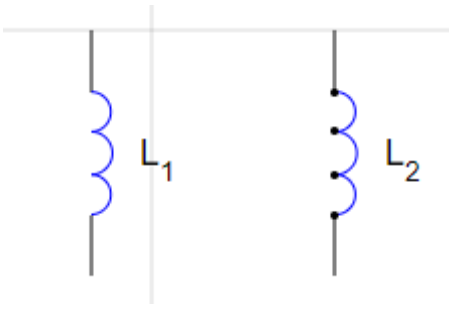
Kolmisilmukkainen tehtävä on siinä mielessä tärkeä, sillä sen ratkaistuaan opiskelija osaa muodostaa tarvittavat yhtälöt monimutkaisemmillekin piireille, joiden avulla voidaan sitten ratkaista kaikkia haluttuja suureita.

4.2.1 JSXGraph piiritehtävien visualisoinnin apuna

Tehtävien tärkeänä visuaalisena apuvälineenä toimii lähes poikkeuksetta piirikaavio. Tämä voidaan luoda hyödyntämällä JSXGraph-kirjastoa ja sen geometrisiä elementtejä. Jokainen sähköisen komponentin piirrosmerkki voidaan luoda yhdistelemällä sopivia kuvioita, jotta saadaan haluttu tulos. Taulukossa 2 on esitetty piirikaaviossa tarvittavat piirrosmerkit ja niiden muodostamiseen tarvittavat JSXGraph-elementit. Jokaiseen komponenttiin liitetään myös erikseen *Text*-elementti, jolla nimetään kyseinen komponentti esim. R_1, R_2, ...jne. Sen lisäksi eri komponenteissa käytetyt samat elementit

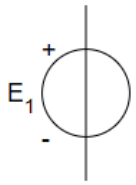
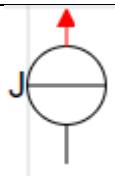
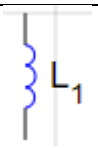
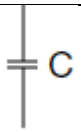
voivat sisältää eri attribuutteja. Esimerkiksi referenssisuunnassa *Line*-elementti saa *lastArrow*-attribuutin, jonka seurauksena viivan päätepisteeseen piirretään nuolenpää suunnan osoittamista varten. Tätä ei kuitenkaan käytetä esim. jännitelähteen *Line*-elementissä. Lisäksi attribuuteilla voidaan määrittää piirtämisessä käytettäviä värejä, joilla voidaan havainnollistaa haluttuja asioita ja myös vahvistaa oppimista (Vesapuisto, 2021). Attribuuttien avulla voidaan myös muokata fontteja ja niiden kokoa, mikäli on tarpeen.

Yksi tärkeimmistä attribuuteista on ”*fixed*”. Normaalisti, kun elementti piirretään, sitä voidaan liikutella vapaasti koordinaatistossa tehtävän aikana. Tämä ei kuitenkaan välttämättä ole haluttua tehtävien piirikaavioissa, jotka saattavat mennä sotkuisiksi, jos komponentteja siirreltäisiin kesken ratkaisun. Siksi voidaankin määrittää ”*fixed:true*” elementeille, jotka halutaan tehdä staattisiksi. Näin opiskelija ei vahingossa voi muuttaa piiriä kesken ratkaisun. Toinen tärkeä attribuutti on ”*visible*”. Sen avulla voidaan piilottaa tiettyjä elementtejä, joita kuitenkin tarvitaan kaaviota piirrettäessä. Esimerkiksi kuvassa 18 kelojen puoliympyrät määritellään pisteiden avulla, mutta toisessa kelassa pisteet on piilotettu asettamalla ”*visible:false*”, jotta lopputuloksesta tulee halutusti siisti.



Kuva 18. *Visible*-attribuutin hyödyntäminen kuvioita piirrettäessä. Kelassa L_1 on pisteiden yhdeksi attribuutiksi asetettu *visible:false* (a) ja kelassa L_2 *visible:true* (b).

Taulukko 3. Piirrosmerkkien luomiseen tarvittavat JSXGraph-elementit.

Piirikaavion komponentti	Tarvittavat JSXGraph-elementit	Komponentti piirrettynä (JSXGraph)
Jännitelähde	• Line • Circle • Text (x2)	
Virtalähde	• Line (x3) • Circle	
Vastus tai Impedanssi	• Line (x2) • Point (x4) • Polygon	
Kela	• Line (x2) • Point (x4) • semicircle (x3)	
Kondensaattori	• Line (x4)	
Johdin (komponenttien välillä)	• Line	
Referenssisuunnat	• Line	
Solmupiste	• circle	

Taulukossa 3 esiintyviä komponenttien piirrosmerkkejä yhdistelemällä voidaan siis rakentaa mielivaltaisia piirikaavioita. Elementistä riippuen komponenteille asetetaan koordinaatit, jonka mukaan kyseinen kuvio piirtyy määriteltyyn koordinaatistoon.

4.2.2 Satunnainen generointi

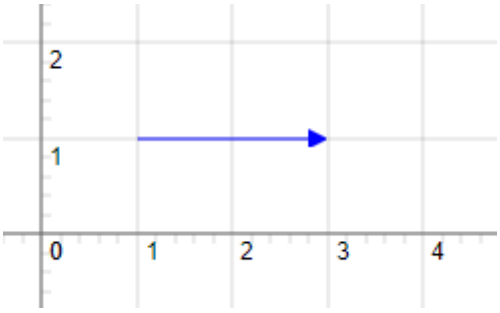
Alaluvussa 4.2.1 käytiin lyhyesti läpi, että ”Tehtävän muuttujat”-kentässä voidaan muuttujille luoda arvoja poimimalla satunnaisesti jokin numero. Tätä voidaan hyödyntää myös JSXGraph-elementtien koordinaattien määrittelyssä. Tehtävän muuttujia voidaan hyödyntää kysymystekstissä ja niitä voidaan sijoittaa myös elementtien sisälle. Tällä tavoin voidaan satunnaisesti määrittää esimerkiksi referenssisuunnat tehtävään, jotta opiskelija saisi aina erilaisen version piirikaaviosta. Tarkastellaan seuraavaa koodia:

```
s1 : rand([-1,1]);
x1 : 4-s1;
x2 : 4+s1;.
```

Kolme muuttujaa *s1*, *x1* ja *x2* on määritelty ”Tehtävän muuttujat”-kentässä. Ensimmäinen muuttuja *s1* saa satunnaisesti arvon -1 tai 1. Sitten *x1* ja *x2* saavat arvoiksi joko 5 tai 3 niiden lausekkeisiin perustuen. Mikäli *s1* = -1 , niin *x1* = 5 ja *x2* = 3. Jos taas *s1* = 1 saa *x1* arvoksi 3 ja *x2* taas 5. Tällä menettelyllä *x1* ja *x2* vaihtelevat kahden arvon välillä suorituskerrasta riippuen, mutta saavat aina keskenään eri arvon. Referenssinuolet kaaviossa piirretään *line*-elementillä, joka tarvitsee aloitus- ja lopetuspisteen koordinaatit. Määrittäminen tapahtuu seuraavasti:

```
var vec2 = brd.create('line',[[1,1],[3,1]],
{strokeColor:'#0000FF', straightFirst:false,
straightLast:false, strokeWidth:1, lastArrow:true,
fixed:true});.
```

Koodilla saadaan aikaiseksi kuvassa 19 esitelty kuvio, jossa referenssinuolen alkupistellä on koordinaatit [1,1] ja loppupisteellä [3,1].

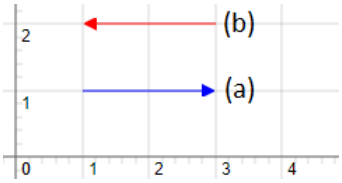


Kuva 19. Staattisilla koordinaateilla määritelty referenssinuoli.

Implementoidaan seuraavaksi muuttujat *x1* ja *x2* osaksi koordinaattien määrittelyä lisäämällä uusi *line*-elementti ja korvaamalla alku- ja loppupisteen x-koordinaatit niillä:

```
var vec3 = brd.create('line',[[x1,2],[x2,2]],
{strokeColor:'#ff0000', straightFirst:false,
straightLast:false, strokeWidth:1, lastArrow:true,
fixed:true});.
```

Lopputuloksena saadaan kuvan 20 kaltaiset tilanteet, joissa nuolen suunta saattaa vaihtua aina, kun tehtävä aloitetaan uudelleen. Kuvassa 20a on alkuperäinen referenssinuoli ja kuvassa 20b taas uusi, joka voi satunnaisesti muuttaa suuntaansa riippuen muuttujan *s1* arvosta.



Kuva 20. Referenssisuuntanuolia. Sininen nuoli (a) on muodostettu staattisilla koordinaateilla ja (b) satunnaisilla *x*-koordinaateilla.

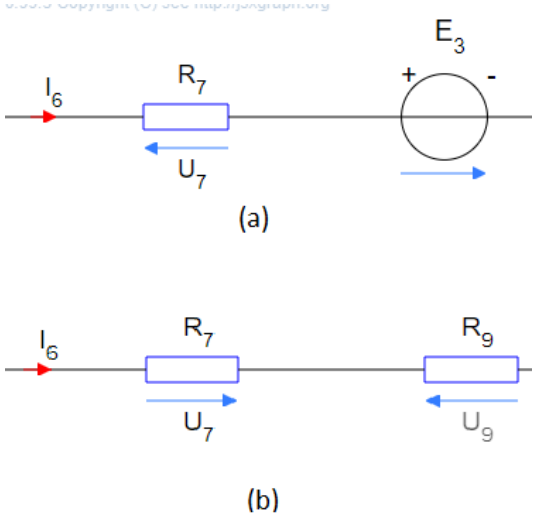
Tämänlaisella yksinkertaisella menetelmällä voidaan arpoa piirikaavion kaikkien virtojen ja jännitteiden referenssinuolien suunnat, mikä mahdollistaa useiden eri versioiden luomisen piirikaaviosta ilman tämän suurempaa vaivaa. Satunnaisgenerointia voidaan hyödyntää myös, kun halutaan arpoa erilaisia komponentteja eri paikkoihin. Tehtävään voidaan lisätä *if-else* -rakenne, minkä seurauksena voidaan satunnaisesti generoida

tiettyyn paikkaan haluttuja komponentteja. Voidaan esimerkiksi arpoa muuttujalle ”Tehtävän muuttujat” -kentässä kahdesta arvosta jompikumpi ja sen seurauksena piirtää toinen halutuista. Tarkastellaan algoritmia 2.

```
/* Objektin arpominen*/
(function(){
var a1 = '{#a1#}'
if(a1 == -1) {
        var p21 = brd.create('point',[10,12.3], {name:'',
face:'[]', size:0, fixed:true});
        var p22 = brd.create('point',[12,12.3], {name:'',
face:'[]', size:0, fixed:true});
        var p23 = brd.create('point',[12,11.7], {name:'',
face:'[]', size:0, fixed:true});
        var p24 = brd.create('point',[10,11.7], {name:'',
face:'[]', size:0, fixed:true});
        var R6 =
brd.createElement('polygon',[p21,p22,p23,p24],
{strokeColor:'#000000', name:'R6', strokeWidth:3,
fillColor:'#ffffff', fixed:true});
        brd.create('text',[10.8,13, 'R_9'], {fontSize:18,
fixed:true});

var vec25 = brd.create('line',[[z13,11.3],[z14,11.3]],
{strokeColor:'#337CFF', lastArrow:true, straightFirst:false,
straightLast:false, strokeWidth:1, fixed:true});
brd.create('text',[10.8, 10.8, 'U_9'], {fontSize:18,
fixed:true});
}
else {
 var p349 = brd.create('circle',[[11,12],[12,12]],
{strokeColor:'#000000',strokeWidth:1, fixed:true});
 brd.create('line',[[10,12],[12,12]], {strokeColor:'#000000',
straightFirst:false, straightLast:false, strokeWidth:1,
dash:0, fixed:true});
 brd.create('text',[z13, 13, '+'], {fontSize:20});
 brd.create('text',[z14, 13, '-'], {fontSize:20});
 brd.create('text',[10.8, 14, 'E_3'], {fontSize:20});
var vec24 = brd.create('line',[[z13,10.7],[z14, 10.7]],
{strokeColor:'#337CFF', lastArrow:true, straightFirst:false,
straightLast:false, strokeWidth:1, fixed:true});
}

})();
```

Algoritmi 2. Esimerkki *if-else* -rakenteen hyödyntämisestä komponentin arpomisessa.

Algoritmissä 2 muuttuja *a1* toimii arpana, joka voi saada arvon -1 tai 1, ja se on määritelty ”Tehtävän muuttujat” -kentässä. Mikäli *a1* saa arvon -1, piirretään kuvaan vastus, joka määritellään *Polygon*-elementillä. Vastuksen rinnalle piirretään myös referenssisuuntanuoli kuvaamaan jännitettä U_9. Tämän nuolen suunta riippuu muuttujista *z13* ja *z14*, jotka toimivat samalla tavalla kuin kuvassa 20 on esitettynä. Lisäksi komponenttiin lisätään vielä teksti, josta käy ilmi, että kyseessä on vastus R_9. Mikäli arpa *a1* saa jonkin muun arvon tai tässä tapauksessa arvon 1, suoritetaan koodin *else*-osio. Tämän osion seurauksena tilalle piirretäänkin jännitelähde, jossa muuttujat *z13* ja *z14* määrittelevät jännitenuolen lisäksi myös lähteen napaisuuden *Text*-elementtien avulla. Kuvassa 21 on esitetty molemmat tilanteet. Tilanteessa 21a on komponentiksi arvottu jännitelähde ja tilanteessa 21b vastus algoritmin 2 avulla. Myöskin molemmissa tilanteissa jännitenuolien suunnat on arvottu osoittamaan eri suuntia. Lisäämällä tehtäviin komponenttien arpomista voidaan luoda vielä useampia erilaisia versioita alkuperäisestä piiristä.



Kuva 21. If-else -rakenteella piirretyt komponentit riippuen tilanteesta. Kohdassa (a) komponentiksi on arvottu jännitelähde ja kohdassa (b) vastus.

4.2.3 Oikean vastauskaavan generointi satunnaismuuttujien kanssa

Komponenttien ja referenssisuuntien arpomisella on tietenkin vaikutus myös tehtävän oikeaan vastaukseen ja sen muotoilemiseen. Jotta tehtävä osaa muodostaa automaattisesti oikean vastauksen perustuen nuolien suuntiin ja eri komponenttien käyttämiseen, täytyy ne ottaa myös huomioon.

Referenssisuuntien kanssa oikean ratkaisulausekkeen muodostaminen on melko yksinkertaista. Ajatellaan vaikka kuvan 13 kaltaista yksinkertaista yhden silmukan piiriä. Kyseisen piirin jänniteyhtälön oikea vastaus on:

$$E - U_3 + U_2 - U_1 = 0 \quad (1)$$

tai

$$-E + U_3 - U_2 + U_1 = 0, \quad (2)$$

missä *E* on jännitelähde ja U_1–U_3 ovat vastuksien yli olevia jännitteitä. Tällöin myös tehtävän oikean ratkaisun yhtälökin olisi samanlainen. Ongelma ilmenee, kun jokin referenssisuunta vaihtaa suuntaa. Tämä voidaan kuitenkin ennakoida helposti. Oikean vastauksen lausekkeesta voidaan huomata, että jokainen muuttuja *E* ja $U_{1\text{-}3}$ voivat olla joko positiivisia tai negatiivisia. Referenssinuolien suunta taas riippuu yhdestä muuttujasta, joka saa arvon -1 tai 1. Käyttämällä tätä muuttujaa ratkaisulausekkeessa kertoimena jokaisen tehtävän muuttujan edessä, saadaan oikea ratkaisu tulostettua aina automaattisesti riippumatta siitä, mihin suuntaan nuolet kuvassa lopuksi osoittavat. Edellisen yhtälön ja kuvan 8 piirin oikeaksi vastaukseksi voidaan antaa seuraava lauseke:

```
TAns : simplify(s3*E+s1*U1+s2*U2+s4*U3);.
```

TAns-muuttuja saa arvoksi edellä mainitun lausekkeen. Muuttujat *s1*, *s2*, *s3* ja *s4* ovat yhteyksissä kaavion referenssisuuntiin ja määräävät niiden suunnan saamalla arvon -1 tai 1. Ne asetetaan lopulliseen yhtälöön kertoimiksi jännitteiden eteen. *Simplify*-

komento pitää huolen siitä, että kertoimien numerot 1 tai -1 häipyvät ja jäljelle jää vain kaavan 1 tai 2 mukainen yhtälö, jossa on etumerkit oikein.

Edellä esitetyllä menetelmällä voidaan siis arpoa piirissä esiintyvät referenssisuunnat virroille ja jännitteille ja tuottaa oikean vastauksen yhtälö, joka on linkitetty nuolien suuntiin. Jännitelähteen napaisuuden tuottaminen satunnaisesti tehdään myös samalla tavalla, vaikka se koostuukin kahdesta *Text*-elementistä: ”+” ja ”-”. Lähteen orientaatiosta riippuen, molempien elementtien *x*- tai *y*-arvot saavat samalla tavalla arvotut muuttujat, minkä seurauksena ne vaihtelevat paikkaa.

Monimutkaisuutta tehtävän vastauksen muodostamiseen tulee silloin, kun tehtävään lisätään esimerkiksi pakotettu ja arvottu kiertosuunta. Kiertosuunnan arpominen toimii samalla tavalla kuin referenssisuuntienkin. Oikean lausekkeen muodostamisessa tämä pitää kuitenkin ottaa myös huomioon jokaisen komponentin kohdalla. Toisin sanottuna jokaisen komponentin kerroin kerrotaan myös kiertosuunnan arpa-muuttujalla. Tämäkään ei ole ihan niin yksinkertaista. Ajatellaan vaikka kuvan 13 tapausta ja komponentteja R_1 ja R_3. Jos piiri kierretään myötäpäivään, ovat molemmat referenssisuunnat samansuuntaisia kiertosuunnan kanssa ja molemmat esiintyvät vastauksessa positiivisena. Referenssinuolet osoittavat kuitenkin koordinaatistossa eri suuntiin, minkä seurauksena oikean vastauksen yhtälö olisi väärä. Tästä johtuen, jokainen komponentti täytyy kertoa joko positiivisella tai negatiivisella kiertosuunnan arpa-kertoimella riippuen niiden sijainnista piirissä. Oikean vastauksen yhtälö muttuu siis seuraavanlaiseksi:

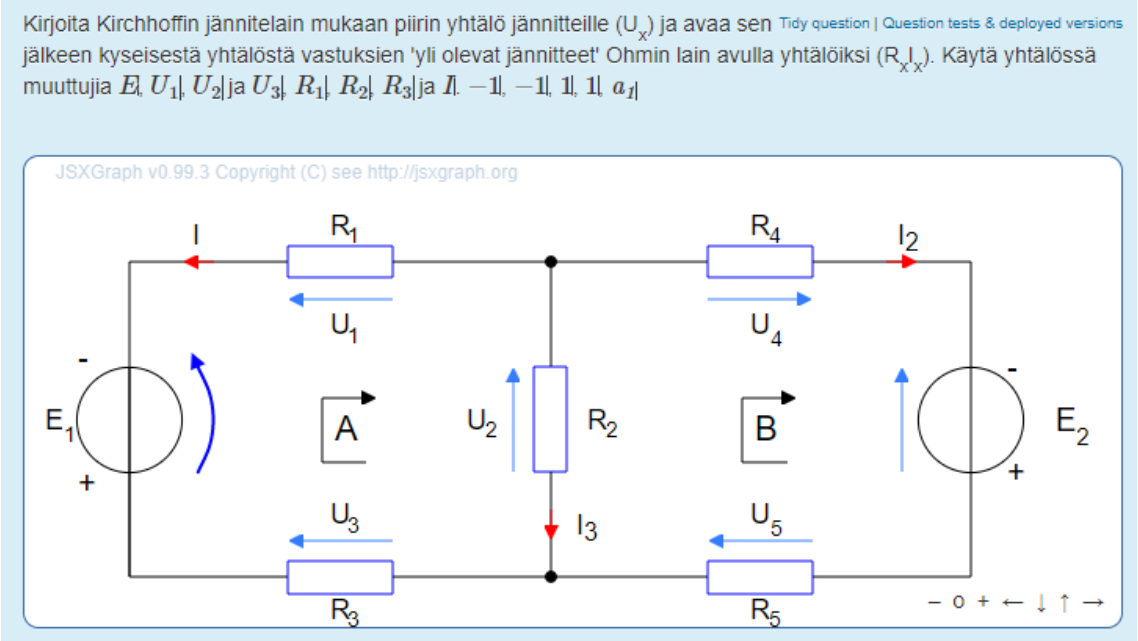
```
TAns : simplify(sa*s3*E+sa*s1*U1+sa*s2*U2+sa*s4*U3);.
```

Muuttuja *sa* on kiertosuunnan arpa, joka saa arvon -1 tai 1. Tässä yhtälössä muuttuja on myös aina positiivinen vaikka vastakkaisilla reunoilla sijaitsevat komponentit pitäisi ottaa huomioon. Tämä johtuu siitä, että korjaus on tehty jo aikaisemmin eli kun määritetään referenssisuuntia komponenteille. Tarkastellaan esimerkiksi U_1 ja U_3 suuntien määrittelyä:

```
/* U1*/
s1 : rand([-1,1]);
x1 : 4-s1;
x2 : 4+s1;

/*U3*/
s4 : rand([-1,1]);
m1 : 4+s4;
m2 : 4-s4;.
```

Tällöin molempien referenssisuuntien alku- ja loppupiste ovat ns. vastakkain mikäli *s1* ja *s4* saavat arvoiksi saman. Sen seurauksena niiden suunnat ovat jo valmiiksi määritelty oikean vastauksen lausekkeeseen ja niitä ei tarvitse korjata.

Uusi haaste yhtälön generoinnissa ilmenee, mikäli piirikaavioon arvotaan eri komponentteja tai siinä muutetaan kytkentää muulla tavalla. Koska komponenttien paikat voidaan arpoa yksinkertaisella *if-else* -rakenteella, voidaan myös vastaukset kirjoittaa saman tyyppisesti. Eli oikea lauseke saa aina tietyn yhtälön riippuen arvotuista komponenteista. Kuvassa 22 on esitetty kahden silmukan piiri, jossa referenssisuunnat on arvottu ja kiertosuunnat on määrätty aina myötäpäivään. Lisäksi B-silmukassa jännitelähteen E_2 tilalle voidaan arpoa vastus R_6. Tehtävänä on kirjoittaa silmukoiden jänniteyhtälöt ja sen jälkeen kirjoittaa ne Ohmin lain avulla haluttuun muotoon. Arvottu komponentti vaikuttaa B-silmukan vastauksiin, joten se täytyy ottaa kaikissa tilanteissa huomioon.



Kuva 22. Kahden silmukan STACK-tehtävä, jossa jännitelähteen E_2 tilalle voidaan arpoa myös vastus R_6.

Mikäli piirissä arvottaisiin muitakin komponentteja täytyisi nekin ottaa huomioon jokaisessa eri tilanteessa. Sen seurauksena oikean vastauksen yhtälön luominen voi nopeasti muuttua todella monimutkaiseksi ja virheiden ilmenemisen riski kasvaa.

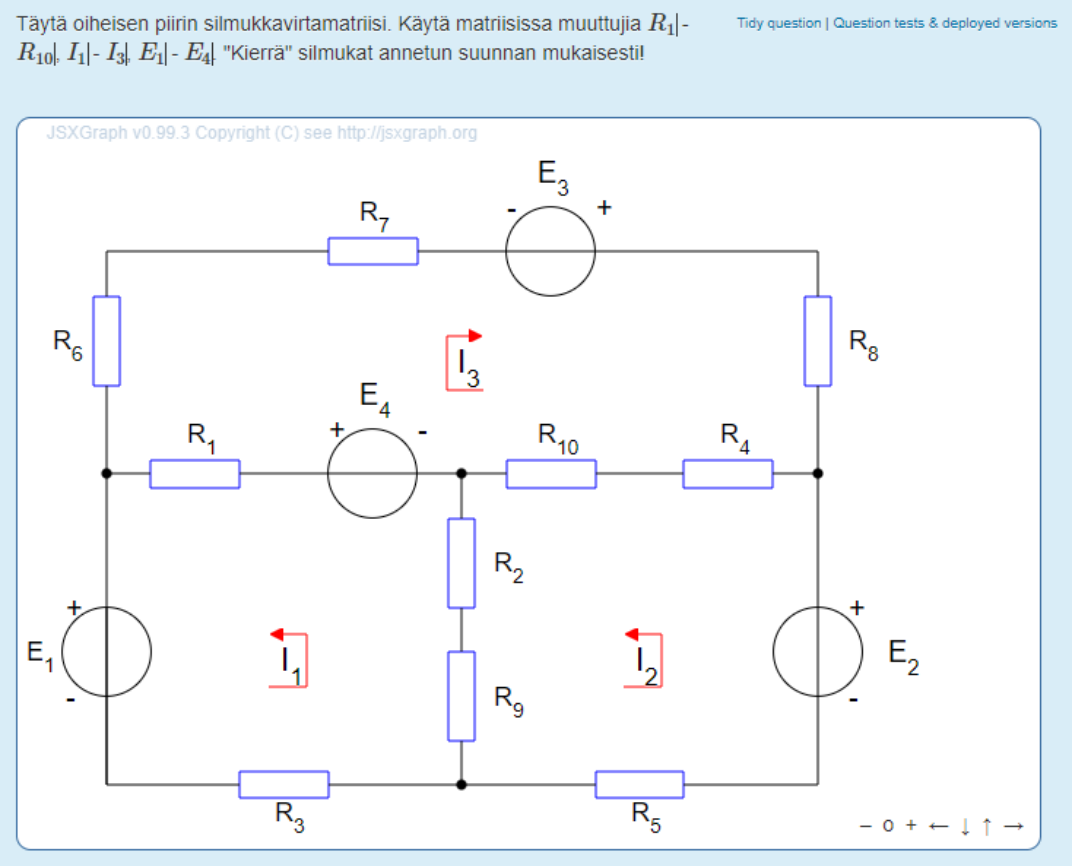
Algoritmi 3 esittää oikean vastauksen yhtälöt kaikkiin neljään eri kysymykseen eli A- ja B-silmukan jänniteyhtälöt ja Ohmin lain avulla avatut yhtälöt.

```
/* Oikeat vastaukset riippuen objektista */
/* Jänniteyhtälöt */
TansA : simplify(+s3*E1+s1*U1+s2*U2+s4*U3);
TansB : if(b305 = 1) then (simplify(-s6*E2-s8*U4+s7*U5-s2*U2))
else (simplify(-s6*U6-s8*U4+s7*U5-s2*U2));

TansC : simplify(s3*E1-s3*R1*I1-s3*R3*I1+s10*R2*I2);

TansD : if(b305 = 1) then (simplify(-s6*E2-s9*R5*I2-s9*R4*I2-
s10*R2*I3))
else (simplify(-s9*R6*I2-s9*R5*I2-s9*R4*I2-s10*R2*I3));
```

Algoritmi 3. Oikean vastauksen yhtälöt kuvan 20 kahden silmukan tehtävään.

Muuttujat *TansA* ja *TansC* ovat vastaukset silmukka A:lle ja *TansB* ja *TansD* taas silmukka B:lle. Koska silmukassa A mikään ei muutu, noudattavat ne edellä mainittuja esimerkkejä oikean vastauksen yhtälöistä. Sen sijaan silmukassa B ja siihen liittyvissä vastauksissa täytyy ottaa huomioon muuttuva komponentti. Tässä tapauksessa voidaan hyödyntää yksinkertaista *if-else* -rakennetta. Muuttuja *b305* toimii arpana komponentille ja voi saada arvon -1 tai 1, sillä komponenttejakin on vain kaksi. Mikäli se saa arvoksi yksi, luodaan piiriin jännitelähde E_2 ja muussa tapauksessa vastus R_6. Tästä syystä vastauksissa noudatetaan samaa yhtälöä, jossa *TansB* ja *TansD* saavat jomman kumman yhtälön riippuen muuttujasta *b305*.

Yksittäisen komponentin muutos voidaan siis suorittaa suhteellisen helposti, mutta jos muutoksia on useampia eri komponenttien paikoilla, monimutkaistuu myös vastauksen luominen *if else* -rakenteella, sillä jokainen muutos luo uuden tilanteen, joka pitää ottaa huomioon myös muissakin muutoksissa. Esimerkiksi kuvassa 23 on esitetty kolmen silmukan piirikaavio, jossa esiintyvät komponentit E_4, R_9 ja R_{10} voivat muuttua riippuen arpa-muuttujan saamasta arvosta. Muutos tapahtuu siten, että jännitelähde E_4 voi esiintyä jossakin kolmesta haarasta eli arpa saa yhden kolmesta eri arvosta. Vastaus rakennetaan *if else*:llä, siten, että jokaiselle tilanteelle luodaan uusi yhtälö. Tehtävänä on täydentää piirin silmukkavirtamatriisi, joten vastaukset itsessään ovat myös hyvin pitkiä ja sisältävät paljon kertoimia, jotka ovat yhteydessä arvottuihin kiertosuuntiin.



Kuva 23. Esimerkki silmukkavirtatehtävästä, jossa jännitelähde E_4 voi sijaita eri haaroissa.

Seuraavaksi algoritmissä 4 on esitetty resistanssimatriisin ratkaisukaavassa käytetty koodi *if else* -rakenteella.

```
Tans1 : if (s95 = 1) then matrix([R1+R2+R3+R9,-s93*s4*R2,-
s93*s96*R1+(-s93)*s96*R9],[-s93*s4*R2,R2+R4+R5+R10,-
s96*s4*R4+(-s96)*s4*R10],[-s93*s96*R1+(-s93)*s96*R9,-
s4*s96*R4+(-s4)*s96*R10,R1+R4+R6+R7+R8+R9+R10])

else if (s95 = 2) then matrix([R1+R2+R3+R9+R10,-s93*s4*R2+(-
s93)*s4*R10,-s93*s96*R1+(-s93)*s96*R9],[-s93*s4*R2+(-
s93)*s4*R10,R2+R4+R5+R10,-s96*s4*R4],[-s93*s96*R1+(-
s93)*s96*R9,-s4*s96*R4,R1+R4+R6+R7+R8+R9])

else matrix([R1+R2+R3+R9,-s93*s4*R2+(-s93)*s4*R9,-
s93*s96*R1],[-s93*s4*R2+(-s93)*s4*R9,R2+R4+R5+R9+R10,-
s96*s4*R4+(-s96)*s4*R10],[-s93*s96*R1,-s4*s96*R4+(-
s4)*s96*R10,R1+R4+R6+R7+R8+R10]);
```

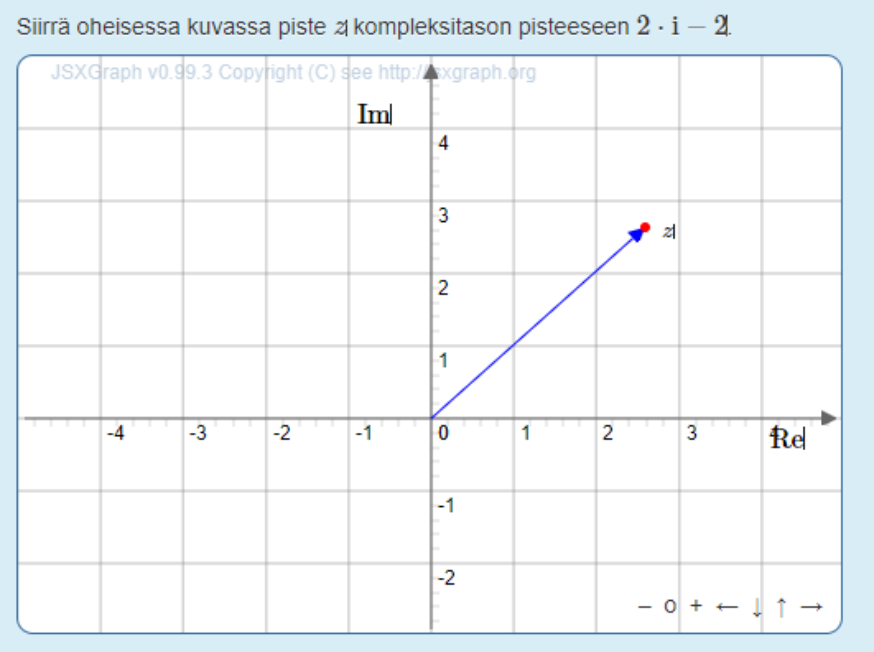
Algoritmi 4. Matriisitehtävän oikean vastauksen rakenne.

Kiertosuuntaa merkitsevinä muuttujina esiintyvät muuttujat *s4*, *s93* ja *s96*. Muuttujan *s95* tehtävänä on arpoa jännitelähteen paikka. Kuten ratkaisun koodista voidaan huomata, niin erilaisia kertoimia kuin lausekkeitakin on paljon, joten virheitä voi helposti syntyä. Sen lisäksi, mikäli virhe ilmenee, voi sen löytämiseen kulua paljon aikaa.

4.3 Tehtävien pelillistäminen ja interaktiivisten piiritehtävien kehittäminen

Edellä esitetyt esimerkit ovat siinä mielessä staattisia piirikaavioita tai tehtäviä, että opiskelija ei koske piirrettyyn kuvaan millään tavalla vaan tehtävä välitetään tehtävätekstin ja kuvan avulla, minkä seurauksena vastaus kirjoitetaan erilliseen kenttään. Tätä voidaan pitää perinteisenä tehtävätyyppinä. Lisäksi STACK-järjestelmä ja JSXGraph kuitenkin mahdollistavat myös toisenlaisten tehtävien luomisen, joissa ratkaisua ei tarvitse kirjoittaa lainkaan. Tehtävä ratkeaa liikuttelemalla JSXGraph-elementtejä kuvassa oikealle paikalleen. Tällöin vastauskenttä on piilotettu opiskelijalta ja se täytetään automaattisesti, kun kuvaa muokataan siirtämällä komponentteja. Tämänlainen lähestymistapa voi vaikuttaa tekijälle mielenkiintoisemmalta kuin pitkien lausekkeiden ja yhtälöiden kirjoittaminen.

Yksi hyvä sovelluskohde on matemaattiset tehtävät, jotka sisältävät vektoreita. Tällöin opiskelijaa voidaan pyytää siirtämään vektori koordinaatistossa oikeaan kohtaan annettujen tietojen avulla. Sähkötekniikassa ja etenkin vaihtosähkön puolella kompleksiluvut korostuvat osoitinlaskennan käytön seurauksena. Niinpä kompleksiluvuistakin voidaan luoda tämäntyyppisiä tehtäviä, joissa siis siirretään asioita oikeille paikoilleen annettujen tietojen perusteella. Kuvassa 24 on esitetty yksinkertainen tehtävä, jossa tarkoituksena on siirtää kuvassa piste ”*z*” annettuun paikkaan kompleksitasossa. Opiskelija voi tarttua pisteestä kiinni hiireä painamalla ja raahaamalla pisteen oikeaan paikkaan. Palaute annetaan samalla tavalla kuin staattisissa tehtävissä riippuen opiskelijan antamasta vastauksesta.



Kuva 24. Dynaaminen tehtävä, jossa kuvan pistettä ”z” täytyy liikuttaa oikeaan paikkaan oikean vastauksen saamiseksi.

Opiskelijan vastauksen tallentaminen on vähän mutkikkaampaa kuin perinteisissä tehtävissä, sillä sitä ei varsinaisesti syötetä kenttään opiskelijan toimesta. Vastaus- ja validointikentät kuitenkin ”löytyvät” tehtävästä, koska toimiakseen järjestelmä edellyttää niiden olemassaolon. Koska oppilaan on tarkoitus antaa vastaus siirtämällä pistettä ”z” eikä kirjoittamalla vastausta suoraan, ovat vastauskentät piilotettuja opiskelijalta. Vastaus syötekenttään haetaankin opiskelijan siirtämästä elementistä, eli tässä tapauksessa pisteestä ”z”. Tarkastellaan tehtävän koodia osissa. Huomioitavia asioita ovat, minkä tyyppisenä vastaus halutaan hakea, jotta sen vertailu on mahdollisimman helppo toteuttaa. Toisekseen pitää myös luoda mekanismi, jolla opiskelijan antama vastaus tallennetaan oikeaan muotoon. Tarkastellaan algoritmiä 5.

```
<p>Siirrä oheisessa kuvassa piste \(z\) kompleksitason
pisteeseen \({@Tquestion@}\).</p>

<div hidden="">\([x,y]=\)[[input:wans1]]</div>
<div>[[validation:wans1]]</div>

<p>
<script>
var XYansfield;

var XYinputs = document.getElementsByTagName('input');
// haetaan vastauskenttä
for (var i=0; i < XYinputs.length; i++) {
   if (XYinputs[i].id.indexOf('wans1') >= 0)
      XYansfield = XYinputs[i];
}

var xA = {#xA#};
var yA = {#yA#};
var ansnow = [xA,yA];
if (XYansfield.value.length>0) {
   ansnow = eval(XYansfield.value);
}
JXG.Options.text.useMathJax = true;
</script>.
```

Algoritmi 5. Syötekentän muuttaminen manipuloitavaan muotoon (Tanskanen, 2017).

Ylimmässä osiossa tehtävän koodissa sijaitsee tehtäväteksti. Muuttuja *Tquestion* on ”Tehtävän muuttujat” -kentässä määritelty lauseke, joka arvotaan aina eri suorituskerroilla. Kuvan 24 esimerkissä se saa arvokseen ”2·i–2.” Se on määritelty siten, että muuttuja *x* on kompleksiluvun reaaliosa ja *y* imaginääriosa. Seuraavassa osiossa syötekenttä ”[[input:wans1]]” piilotetaan sijoittamalla se <div hidden> -tagin sisälle. Myös validointielementti lisätään. Jotta syötekenttään voitaisiin syöttää JavaScriptillä erilaisia arvoja, luodaan uusi muuttuja *XYinputs* johon tallennetaan sivun ”input”-elementit listaksi. *For*-silmukan avulla etsitään listasta elementti, jonka *id* on ”wans1” ja se tallennetaan *XYansfield* -muuttujan arvoksi. Eli nyt meillä on tallennettuna tehtävän vastauskenttä JavaScript-muuttujaksi, jota voidaan muokata tehtävän seuraavissa vaiheissa (Tanskanen, 2017). Muuttujat *xA* ja *yA* on määritelty ”Tehtävän muuttujat”-kentässä ja ne toimivat lähtöarvoina myöhemmin luotavalle pisteelle. Muuttujalle *ansnow* annetaan arvoksi ”[xA,yA]”, jotta sitä voidaan käyttää suoraan *point*-elementin

koordinaattien määrittämisessä. Seuraava *If*-lauseke tarkistaa, onko opiskelija antanut vastausta ja se tallennetaan *ansnow* muttujaan. Viimeinen rivi mahdollistaa MathJax-symboleiden ja -elementtien käyttämisen JSXGraphissa.

Vastauskentän manipuloitavaan muotoon muuttamisen jälkeen kirjoitetaan kysymyksen JSXGraph-osio samalla tavalla kuin staattisissakin esimerkeissä. Luodaan aluksi laatikko grafiikalle, jonka jälkeen laatikkoon sijoitetaan koordinaatisto. Tämän jälkeen sinne lisätään piste, josta voidaan tarttua kiinni ja sijoittaa se haluttuun paikkaan, sekä nuoli, joka osoittaa origosta pisteeseen. Tarkastellaan algoritmia 6.

```
var pa = brd.create('point', ansnow, {
   name:'\\[z\\]',
   size:1,
   infoboxDigits: false
});

var vec1 = brd.create('line',[[0,0],pa],
{strokeColor:'#0000ff', lastArrow:true, straightFirst:false,
straightLast:false, strokeWidth:1, fixed:true});

pa.on('up', function() {
XYansfield.value= '[' + Math.round(pa.X()) + ',' +
Math.round(pa.Y()) + ']';
   ansnow = eval(XYansfield);
});
```

Algoritmi 6. Pisteen sijainnin yhdistäminen syötekenttään (Tanskanen, 2017).

Aluksi siis luodaan *point*-elementti *pa*. Pisteen alku-koordinaateiksi annetaan muuttuja *ansnow*, jolle asetettiin edellisessä kohdassa lähtöarvot muuttujien *xA* ja *yA* avulla. Seuraavaksi luodaan *line*-elementti *vec1*, jonka aloituspiste on origossa eli sen koordinaatit ovat [0,0] ja loppupiste on sidottu pisteen *pa*-koordinaattehin. Tällä tavoin nuoli osoittaa pisteeseen vaikka sitä siirreltäisiin. Viimeisessä osiossa pisteeseen *pa* lisätään *on()*-metodi ja tapahtumakäsittelijä *'up*'. Tämän seurauksena, kun pisteestä päästetään irti nostaessa hiiren painiketta, tapahtuu metodin sisällä olevat funktiot ja komennot. Tässä tapauksessa vastauskenttä *XYansfield* saa arvoksi pisteen *x*- ja *y*-koordinaatit pyöristettynä. Lisäksi ne on sijoitettuna hakasulkujen sisään ja erotettu

pilkulla. Tämä sattuu olemaan myös se muoto, miten ”Tehtävän muuttujat” -kentässä oikea vastaus on määritelty. Lopuksi myös *ansnow*-muuttuja saa arvoksi saman kuin *XYansfield*. Tätä hyödynnetään kun tehtävä palautetaan ja sivu ladataan uudelleen palautetta varten. Sen sijaan, että tehtävän piste sijaitsisi nyt alkupisteessä, löytyykin se sieltä mihin opiskelija oli sen siirtänyt vastausta annettaessaan, koska *ansnow* määrittää pisteen koordinaatit. Eli lyhyesti summattuna suoritetaan seuraavat toimenpiteet:

- Vastauskentän piilottaminen.
- Muutetaan syötekenttä JavaScriptillä muokattavaan muotoon.
- Luodaan pisteen koordinaateille muuttuja, joka pitää sisällään tämän tiedon oikeassa muodossa.
- Pisteeseen lisätään tapahtumakäsittelijä, joka toteutuu aina kun pisteestä päästetään irti.
- Tapahtumakäsittelijän funktio määritetään tallentamaan sen hetkisen pisteen sijainti *input*-kenttään vastaukseksi.
- Muutetaan myös koordinaattien muuttuja saamaan samat arvot, jotta vastaus säilyisi myös palautetta annettaessa.

Kuvan uudelleen piirtäminen opiskelijan oppimisen kannalta on äärimmäisen tärkeää, sillä se mahdollistaa helpon vertailun palautteen ja oman vastauksen välillä. Vertailemalla kuvaa eli vastausta ja palautetta opiskelijan on helpompi huomata tekemänsä virheet tai onnistumiset. Tämän lisäksi myös itse palautteeseen voidaan piirtää JSXGraph-kuva visualisoinnin parantamiseksi.

Interaktiiviset piiritehtävät

Edellisessä kohdassa mainitun menetelmän pohjalta voidaan luoda mielivaltaisia tehtäviä matemaattisille aihepiireille, joissa lähinnä mielikuvitus on rajana. Tätä samaa menetelmää voidaan soveltaa myös piirianalyysin tehtävissä. Staattisen puolen tehtävissä luotiin komponentteja, jotka koostuivat JSXGraph-elementeistä. Elementeille määriteltiin tällöin omat koordinaatit siten, että ne muodostivat lopulta kokonaisen

komponentin. Nyt, jos jotain elementtiä liikutellaan, niin loogisesti muut eivät liiku mukana. Mikäli kokonainen komponentti halutaan liikuteltavaksi, täytyy elementit linkittää toisiinsa jollain tavalla. Yksi mahdollinen tapa on luoda tartuntapiste, josta komponenttia voidaan liikutella ja sitten sitoa muiden elementtien koordinaatit suhteessa tähän pisteeseen. Eli elementtien koordinaatit muuttuvat aina kun pisteenkin koordinaatit muuttuvat. Tarkastellaan algoritmiä 7, jossa luodaan kaksi pistettä, joista toinen piste saa ensimmäisen pisteen *x*-koordinaatit, mutta *y*-koordinaatti on riippumaton.

```
<p><jsxgraph width="400" height="300" box="mybox">
   (function() {

var brd = JXG.JSXGraph.initBoard('mybox', {boundingbox:[-
2,8,10,-1],  keepaspectratio: true,axis:false});

var pa = brd.create('point', [0,4], {name:'A', size: 3});

var pb = brd.create('point', [function(){return pa.X();},5],
{name:'B', size: 3});


                  })();
</jsxgraph></p>
```

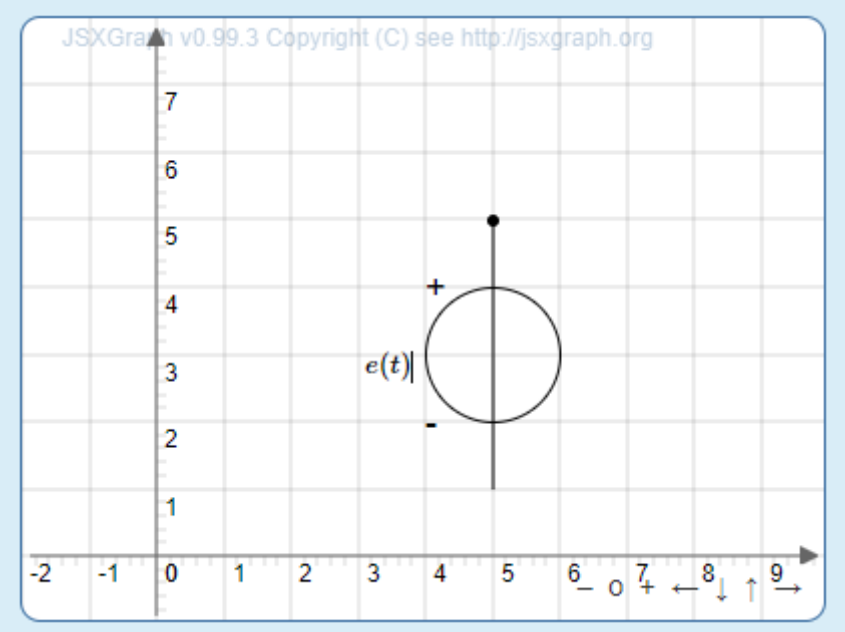
Algoritmi 7. Kahden pisteen luominen, jossa toisen pisteen x-koordinaatti riippuu ensimmäisen pisteen sijainnista (JSXGraph wiki, 2020).

Koordinaatistoon luodaan ensimmäinen piste *pa*, joka sijoitetaan pisteeseen [0,4] ja nimetään se pisteeksi A. Luodaan myös toinen piste *pb*, jonka y-koordinaatiksi asetetaan 5, mutta *x*-koordinaatti haetaan muuttujalta *pa* anonyymillä funktiolla *pa.X()*. Tämän seurauksena piste *pb* seuraa pistettä *pa* *x*-akselilla, kun pistettä *pa* siirretään (JSXGRaph wiki). Kuvassa 25 on esitetty kyseisellä koodilla tuotettu tilanne.

Kuva 25. Dynaamisten pisteiden luonti. Alkutilanne (a) ja lopputilanne (b), kun pistettä A on siirretty (JSXGraph wiki, 2020).

Kuvassa 25a eli alkutilanteessa pisteet ovat niille luoduilla paikoillaan ja voidaan heti huomata, että pisteen B *x*-koordinaatti on sama kuin pisteellä A. Jälkimmäisessä tilanteessa (kuva 25b), jossa pistettä A on siirretty, seuraa piste B *x*-akselia pitkin ja saa aina samat *x*-koordinaatit kuin piste A. Voidaan kuitenkin huomata, että *y*-koordinaatti ei ole riippuvainen toisen pisteen sijainnista vaan se pysyy aina vakiona.

Kun haetaan jonkin pisteen koordinaatteja edellisen esimerkin tapaan, voidaan suorittaa vielä matemaattisia operaatioita kyseiselle arvolle, jotta lopputulos on halutun mukainen. Tämä on pakko suorittaa, jotta sähköiset komponentit piirtyvät oikein. JSXGraph-elementit sidotaan siis yhteen pisteeseen, joka liikkuu koordinaatistossa *x*- ja *y*-suuntiin. Tämä tarkoittaa sitä, että pisteen molemmat koordinaatit pitää hakea jokaiselle liitettävälle elementille ja suorittaa tämän jälkeen tarvittavat laskutoimitukset, jotta elementit sijoittuvat oikeille paikoilleen. Tarkastellaan kuvaa 26, joka esittää dynaamiseksi luotua jännitelähdettä. Lähteen yläpuolella oleva piste-elementti ”*j* [5,5] ” toimii tarttumispisteenä, josta komponenttia voidaan liikutella.



Kuva 26. Jännitelähde, jonka elementtien koordinaatit perustuvat ylimmän pisteen [5,5] koordinaatteihin.

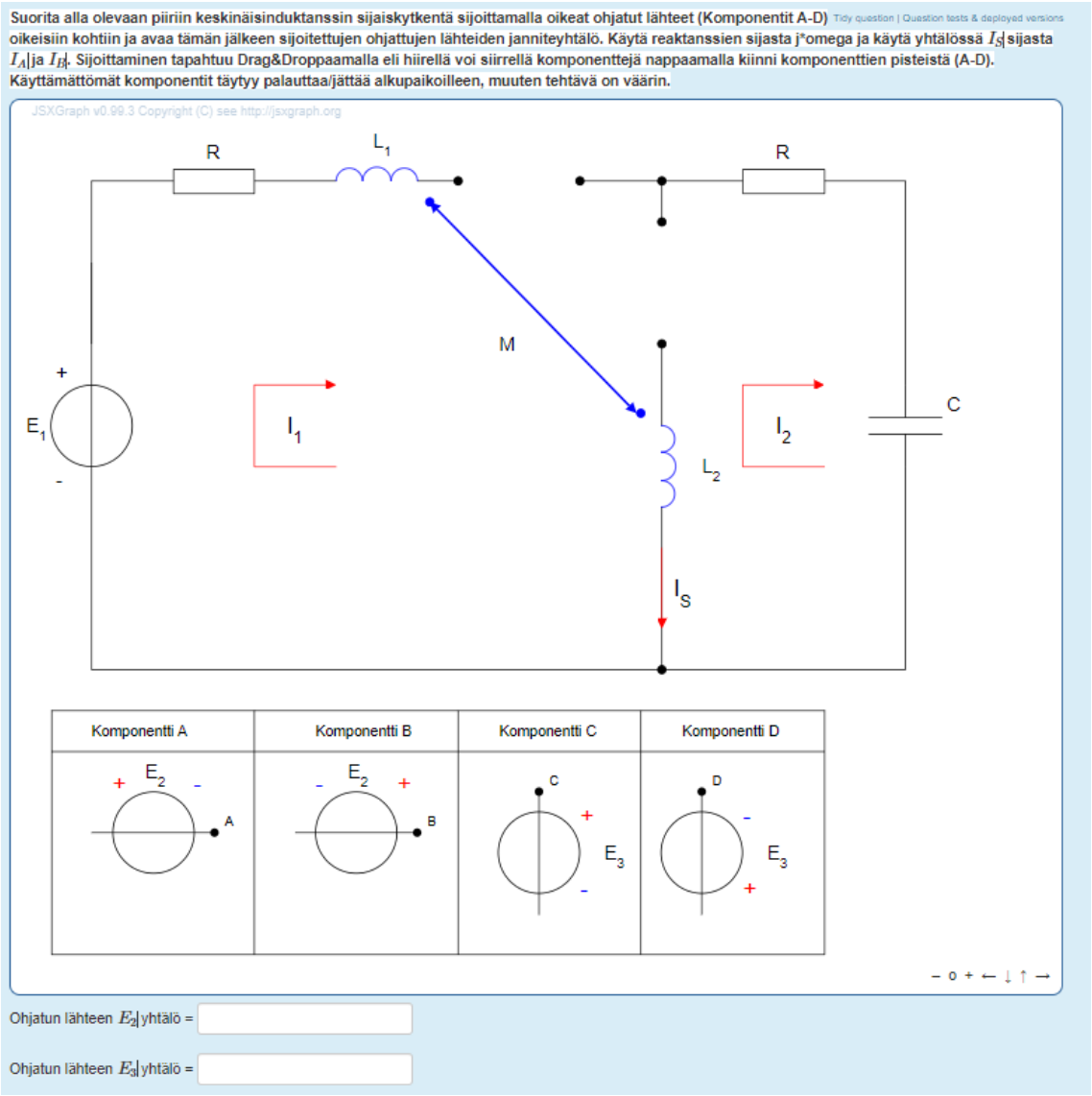
Kaikkien muiden elementtien koordinaatit ovat määritelty ja suhteessa tähän pisteeseen.

Tarkastellaan *line*-elementin määrittelyä seuraavalla koodilla:

```
brd.create('line',
[[function(){ return j.X();},
function(){ return j.Y();}],

[function(){ return j.X();},
function(){ return j.Y()-4;}]],

{strokeColor:'#000000', straightFirst:false,
straightLast:false, strokeWidth:1, dash:0, fixed:true});.
```

Kuten aikaisemminkin, elementin sijainti määritellään tässä alku- ja loppupisteen koordinaateilla. Ensimmäiset kaksi funktiota hakasulkujen sisällä määrittävät viivan alkupisteen koordinaatit ja tässä tapauksessa ne saavat arvoiksi liikuteltavan pisteen *j* koordinaatit. Loppupiste saa saman *x*-koordinaatin. *Y*-koordinaatti otetaan myös pisteestä *j*, mutta siitä vähennetään arvo neljä, joten loppupisteen paikka on aina samalla kohtaa *x*-akselilla, mutta neljä yksikköä alempana *y*-akselilla. Samalla tavalla voidaan määrittää myös komponentin muut elementit.

Aikaisemmin alaluvussa 4.3. esitettiin mahdollisuus muuttaa tehtävän vastaustapaa siten, että vastaus annetaan siirtämällä jotain pistettä koordinaatistossa. Tätä voidaan hyödyntää myös komponenttien kohdalla käyttämällä tarttumispisteen koordinaatteja. Esimerkiksi piirianalyysissa yleisen piirimuunnostehtävän muuttaminen STACK-tehtäväksi voisi olla melko vaikeata ilman liikuteltavia komponentteja tai muuta interaktiivista materiaalia. Keskinäisinduktanssien tapauksessa voidaan annettu piiri esittää sijaiskytkennän avulla. Tarkastellaan kurssilla Piirianalyysi A (2020) käytössä ollutta kuvan 27 tehtävää. Opiskelijan täytyy täydentää puuttellinen piirikaavio oikealla komponentilla, jotta se toteuttaa oikeanlaisen keskinäisinduktanssien sijaiskytkennän. Tämän jälkeen täytyy vielä kirjoittaa liikuteltujen ohjattujen lähteiden yhtälöt. Tehtävä siis yhdistelee liikuttelua ja perinteistä yhtälöiden kirjoittamista. Toimiakseen tarkoitetulla tavalla, opiskelijan täytyy siirtää komponentit oikeille paikoilleen, sillä tehtävä tarkistaa lopuksi komponenttien paikat. Mikäli komponentit sijaitsevat väärässä kohdassa, tulkitaan vastaus vääräksi. Tämä taas tarkoittaa sitä, että käyttämättömät komponentit täytyy löytyä lähtöpaikoiltaan ja oikeat komponentit taas oikeilta paikoiltaan. Komponenttien liikuteltavuutta on helpotettu asettamalla pisteelle *snapToGrid*-attribuutti, minkä seurauksena piste liikkuu vain kokonaisluvullisten koordinaattien välillä. Tämä helpottaa myös vastausten tarkistuksessa, jolloin liukulukua ei tarvitse pyöristää lähimpään kokonaislukuun. Tehtävässä hyödynnetään myös satunnaisuutta arpomalla käämimissuuntaa indikoivan pisteen sijainti molemmissa käämeissä ja virtojen I_1 ja I_2 suunnat.



Kuva 27. Keskinäisinduktanssin sijaiskytkentään liittyvä tehtävä, jossa yhdistyy komponenttien liikutteleminen ja perinteinen kaavojen täydentäminen.

Tehtävän monimutkaisuuden, dynaamisten vastausten ja satunnaisuuden seurauksena myös oikeiden vastausten kirjoittaminen on siis monimutkaista ja virheitä syntyy helposti, joten huolellisuutta ja tarkkuuttaa vaaditaan myös tehtävän laatijalta.

Interaktiivisten tehtävien ja komponenttien liikutteleminen voi myös luoda uusia tarkistussolmuja vastauspuuhun. Joissakin tehtävätyypeissä voi olla tarpeen tarkistaa, onko kaksi komponenttia sijoitettu samaan kohtaan, mikäli se on tehtävässä mahdollista. Esimerkiksi, jos tehtävän kaavioon voidaan lisätä kaksi komponenttia sarjankytkentänä

eikä ole väliä kumpaan kohtaan kumpikin komponentti menee. Tämänlaisessa tilanteessa voi olla hyvä idea huomauttaa opiskelijaa siitä, että piiriä ei voida rakentaa tällä tavalla.

Interaktiivisten kuvaajien hyödyntäminen tehokolmion hahmoittamisessa

JSXGraph mahdollistaa myös graaffien piirtämisen koordinaatistoon *functiongraph*-elementillä. Jos ajatellaan vaihtosähköä ja sen visualisointia, on tärkeää esittää opiskelijalle myös oikean vastauksen löytäminen kuvan avulla. Loistehon kompensoinnissa voidaan todeta, että piirin loisteho on kompensoitu kun se on nolla. Loisteho voidaan visualisoida esimerkiksi tehokolmion avulla, jossa se on toinen suorakulmaisen kolmion lyhyemmästä sivusta toisen ollessa pätöteho ja hypotenuusa esittää näennäistehoa *S* sillä

$$S = \sqrt{P^2 + Q^2}, \tag{3}$$

missä *P* on pätöteho ja *Q* loisteho. Loistehoa ja sen kompensointia voidaan myös visualisoida vertaamalla piirin jännitteen ja virran funktioita. Tässä tapauksessa loistehon seurauksena funktioiden välille syntyy vaihekulma φ . Mikäli vaihekulma on nolla, loisteho on myös kompensoitu, koska

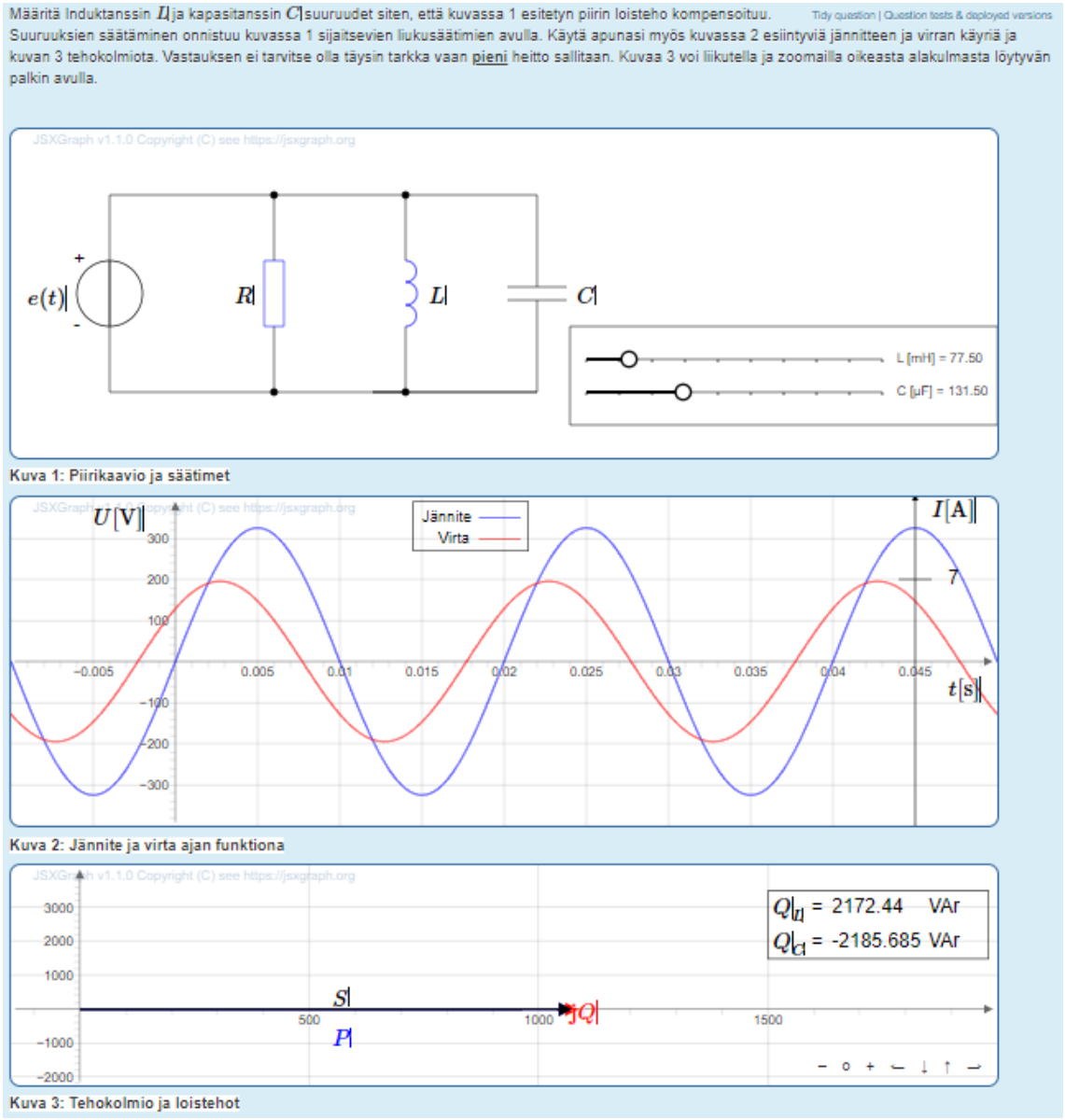
$$Q = |U||I| \sin(\varphi), \tag{4}$$

missä *U* on vaihtojänniteen tehollisarvo ja *I* on vaihtovirran tehollisarvo. Mikäli vaihekulma φ = 0, niin loistehon arvoksi saadaan myös 0 Var, sillä

$$\sin(0) = 0. \tag{5}$$

Tarkastellaan tehtävää, jossa opiskelijan tehtävänä on suorittaa loistehon kompensointi ilman laskemista, pelkästään säätämällä rinnankytkettyjen kelan ja kondensaattorin

arvoja. Kuvassa 28 on esitetty kompensointitehtävä, joka sisältää kolme kuvaa. Ensimmäisestä kuvasta löytyy itse piirikaavio ja säätimet induktanssille ja kapasitanssille.



Kuva 28. Loistehon kompensointiin liittyvä STACK-tehtävä. Oppilas suorittaa kompensoinnin ilman laskemista visuaalisten apuvälineiden avulla.

Visuaalisena apuna on toisessa kuvassa esitetty piirin kokonaisjännitteen ja kokonaisvirran funktiot, jotka reagoivat dynaamisesti ensimmäisen kuvan arvojen muutoksiin. Johtuen *slider*-elementin epätarkkuudesta kyseisessä tehtävässä, sallitaan

opiskelijan vastauksessa pieni heitto. Alimmaisena esitetystä tehokolmiosta voi kuitenkin nähdä aika selvästi, milloin loisteho on suurin piirtein kompensoitu.

Tehtävän vastaus saadaan ottamalla *slider*-elementeistä induktanssin ja kapasitanssin arvot, joista lasketaan ’esiripun takana’ tarvittavat arvot käyttäen piirianalyysin yhtälöitä. Mikäli kokonaisloisteho saa arvokseen suurin piirtein nolla, eli kun kondensaattorin ja kelan loistehon vaikutukset kumoavat toisensa, on vastaus oikein. *Slider*-elementtien arvot voivat siis olla useammassa eri kohdassa ja vastaus voi silti olla oikein, kunhan arvojen seurauksena kumoamisehto täyttyy. Vastaamisen helpottamiseksi on viimeisessä kuvassa vielä annettu loistehon arvot lukuina, jotta pääpainopiste siirtyy havainnollistamiseen ja konseptin ymmärtämiseen, siis laskemisen sijasta.

Interaktiivisia tehtäviä voidaan todennäköisesti hyödyntää entistä enemmän myös tulevaisuudessa. Uusia tehtävätyyppejä voidaan kehittää ja esimerkiksi kytkinten ottaminen osaksi tehtäviä loisi taas uusia mahdollisuuksia piirianalyysin osalta esimerkiksi muutosilmiöitä tarkasteltaessa. Toistaiseksi Vaasan yliopiston STACK-tehtävissä ei ole ilmennyt teknisiä ongelmia, ja vaikka osa tehtävistä onkin varsin raskaita koodin osalta, ovat ne kuitenkin toimineet moitteettomasti, joten grafiikkaakin voidaan vielä lisätä tarpeen niin vaatiessa.

STACK-järjestelmän käyttö sähkötekniikassa ja lähinnä piirianalyysissä voidaan tiivistää seuraaviin havaintoihin:

- JSXGraphin avulla voidaan luoda erilaisia piirikaavioissa käytettäviä sähköisten komponenttien piirrosmerkkejä, joiden avulla voidaan luoda tehtävään standardinmukainen piirikaavio.
- Referenssisuuntien suunnat voidaan arpoa jokaiselle suorituskerralle, jotta vastaus tehtävään muuttuisi ja näin ulkoaopetteleminen vaikeutuisi.
- Arpomisen avulla voidaan luoda useita eri versioita helposti, joten ’erilaisia’ tehtäviä voi luoda nopeasti ja helposti.

- Staattisten tehtävien lisäksi voidaan luoda dynaamisia tehtäviä, joissa oppilas antaa vastauksensa kuvassa liikuteltavilla elementeillä kirjoittamisen sijaan.
- Sähköisten komponenttien piirrosmerkit voidaan myös rakentaa liikuteltaviksi, jolloin tehtäväksi voi muodostua vaikkapa puutteelisen piirin täydentäminen.
- Opiskelijalle voidaan näyttää informaatiota piirin käyttäytymisestä *functiongraph*-elementin avulla ja tämän pohjalle voidaan taas rakentaa uusia tehtävä tyyppejä.

5 STACK-järjestelmän käyttömahdollisuuksia tulevaisuudessa

Tässä luvussa tarkastellaan STACK-järjestelmän kannalta tulevaisuuden näkymiä. Luvussa esitetään erilaisia mahdollisia sovelluskohteita ja uudempien versioiden tarjoamia mahdollisuuksia. Keskeisimmät kohdat voidaan tiivistää seuraavanlaisesti:

- Piirianalyysi A -kurssilla saatujen tuloksien syvempi analyysi järjestelmän tarjoamilla työkaluilla.
- Mahdolliset muut oppiaineet, jotka voivat hyötyä järjestelmästä.
- STACK-järjestelmän uudempi versio eli versio 4 ja sen tuomat mahdollisuudet.
- Uudet oppimisalustat, joihin STACK-tehtäviä voidaan integroida.
- Järjestelmän hyödyntäminen käänteisessä opetuksessa.

Vaikka pahimmat kompastuskiviaiheet piirianalyysissä on nyt jo katettu STACK-tehtävillä, voidaan niihin lisätä vielä uusia ja erilaisia tehtävätyyppejä. Myös loppuihin aiheisiin voisi vielä rakennella kertaavia ja uutta näkökulmaa tuovia tehtäviä.

Tämänhetkisessä tutkintorakenteessa kurssia Piirianalyysi A seuraa kurssi Piirianalyysi B, joka käsittelee monimutkaisempia aiheita, kuten muutosilmiöitä. Tehtäviä näihin aiheisiin ei ole vielä lainkaan kehitetty, mutta tarvetta todennäköisesti on. Luvussa 4 mainittiinkin jo, että muutosilmiöiden kohdalla tehtäviä voitaisiin havainnollistaa esimerkiksi kytkimien ja kuvaajien avulla. Sovelluskohteita löytyisi varmaankin myös siirtojohdoista. Interaktiivisen tehtävätyypin sovelluskohteita voisi olla vaikkapa lähteen sovittaminen Smithin kartan avulla, johon voitaisiin sijoittaa pisteitä ja tarvittavia kehiä.

5.1 STACK Response Analysis -työkalu

Vaasan yliopiston käyttämässä STACK-versiossa on asennettuna ”STACK Response Analysis” -liitännäinen. Se tarjoaa käyttäjälle mahdollisuuden tarkastella opiskelijoiden antamia vastauksia yksityiskohtaisemmin luomalla niistä yhteenvedot.

Opiskelijan vastaus tarkastetaan siis tehtävän vastauspuu-logiikan avulla, jossa siirrytään seuraavaan solmuun perustuen vastaustestien tuloksesta. Liitännäisen toteuttama analyysi perustuukin juuri tehtävän vastauspuuhun. Se käy kaikki kyseisen tehtävän palautukset läpi ja kerää niistä vastauspuun datat. Tulosteena saadaan kaksi erilaista taulukkoa ja palautuksien Maxima-koodit. Ensimmäisessä taulukossa on esitetty erilaiset reitit vastauspuun läpi, joita oppilaan vastaukset ovat kulkeneet. Toisessa taulukossa on lueteltu yksittäiset solmut ja niistä saatujen True/False-testien tulosten määrä. Kuvassa 29 ja 30 on esitetty esimerkit molemmista taulukoista.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
prt1-1-T | prt1-2-T | prt1-3-T | prt1-4-T	1				1	2								1			1			1			
prt1-1-T | prt1-2-T | prt1-3-F | prt1-4-F		1	1		1		1	2				1			1		2	1					
prt1-1-F | prt1-2-F | prt1-3-F | prt1-4-F				1			1			1						1							

Kuva 29. STACK Analysis Response -työkalun tulostama taulukko, jossa on esitetty vastausten eri reitit päätöspuusssa ja niiden määrät

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
prt1-1-T	1	1	1		2	2	2	2	2			1		1	2		3	1
prt1-2-T	1	1	1		2	2	2	3	2			1	2	1	1		3	1
prt1-3-T	1				1	2								1	1		1	
prt1-4-T	1				1	2	1	1	2					1			1	
prt1-3-F		1	1	1	1		3	3	2	1		1	2		1	1	2	1
prt1-4-F		1	1	1	1		2	2		1		1	2		2	1	2	1
prt1-1-F				1			1	1		1			2			1		
prt1-2-F				1			1			1					1	1		

Kuva 30. Työkalun tulostama toinen taulukko, josta käy ilmi yksittäiset solmut ja niistä saatujen ulostulojen määrät.

Molemmissa taulukoissa esiintyy ylärivillä numeroita kasvavassa järjestyksessä. Kyseiset numerot kertovat, mikä variaatio tehtävästä on kysessä. Tehtävien eri variaatiot on myös esitetty työkalun avulla ja ne perustuvat tehtävän tekijän määrittämiin parametreihin ”Tehtävän erotteluteksti” -osiossa tehtävän luomisen yhteydessä. Osiossa voidaan määrittää muuttujia, joiden avulla voidaan erotella variaatiot toisistaan esim. kertomalla, mihin suuntaan jotkin jännitenuolet osoittavat. Tämä helpottaa tehtävien analysointia. Taulukon vasemmanpuoleisimmassa sarakkeessa on mainittu vastauspuun vastausten tunnukset. Tunnus toimii viittauksena solmun eri ulostuloihin ja ne voidaan myös nimetä uudelleen vastauspuuta tehtäessä analysoinnin helpoittamiseksi. Molemmissa taulukoissa tunnukseksi on määritetty esim. *prt1-1-T*, jossa *prt1* on vastauspuun nimi, numero kertoo solmun numeron ja *T* tai *F* ilmaisee kyseisen solmun True- tai False-tulosteen. Ensimmäisessä taulukossa reitti on merkitty tunnuksilla, jotka on erotettu toisistaan pystyviivalla.

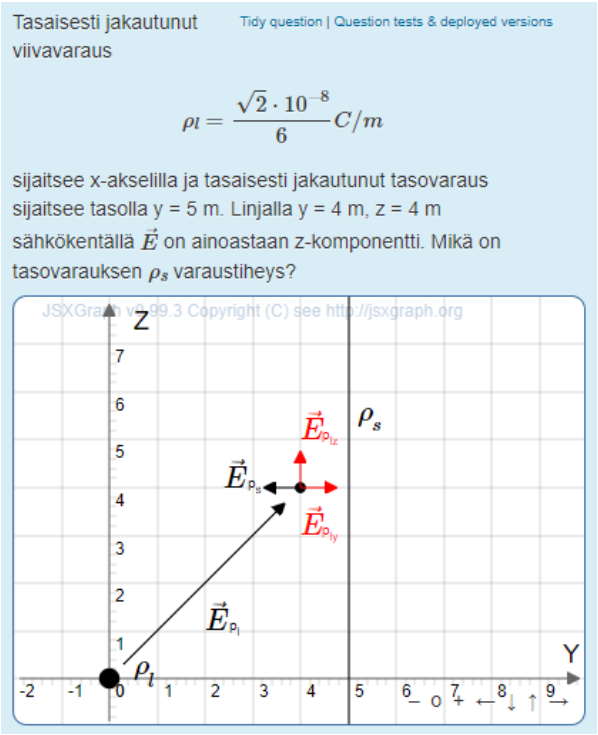
Tarkastelemalla työkalun tulostamia taulukoita voidaan tutkia tehtävän toimivuutta ja opiskelijoiden oppimisen ongelmakohtia tarkasti. Sen avulla voidaan siis löytää solmuja tai kohtia tehtävässä ja itse substanssin ymmärtämisessä, joissa kyseisellä osalla opiskelijoista ilmenee ongelmia. Analysoinnissa tulee ottaa myös huomioon, että kyseessä voi olla tehtävän asetelmasta johtuva ongelma, kuten esimerkiksi uudentyyppinen tehtävä, jonka seurauksena opiskelija ei välttämättä riittävästi tiedä,

miten siihen kuuluisi vastata. Tätä voidaan kuitenkin ehkäistä antamalla opiskelijoille tarpeeksi selvät ja yksinkertaiset ohjeet tehtävänannossa sekä pyytämällä opiskelijoilta palautetta. Analysoinnin perusteella voidaan tehdä tarvittavia muutoksia tehtävään tai voidaan luoda uusia tehtäviä ongelmien paikkaamiseksi.

5.2 STACK-järjestelmästän hyödyntäminen piirianalyysin ulkopuolella

STACK-tehtäviä voitaisiin soveltaa myös muilla sähkötekniikan kursseilla samalla tavalla kuin piirianalyysissä, sillä järjestelmä taipuu hyvin matemaattisten aiheiden käsittelemiseen. Esimerkiksi tutkintoon kuuluvilla fysiikan kursseilla voisi mahdollisesti hyödyntää STACK-järjestelmää opiskelijan oppimista tukevana materiaalina. JSXGraphin avulla voitaisiin piirtää tehtävistä interaktiivisia kuvia, joiden avulla opiskelijan mahdollisuudet oppia ja siten läpäistä kurssi paranevat. Välkky-projektissa olemme havainneet, että yleisestikin kuvien piirtäminen annetusta ongelmasta on digiaikana vähentynyt ja se taas vaikeuttaa tehtävän havainnollistamista, oppimista ja syvän ymmärtämisen saavuttamista. Myös tätä ongelmaa voitaisiin mahdollisesti korjata järjestelmän avulla.

Kenttäteorian puolella kuvan piirtäminen tehtävästä on yksi onnistumisen edellytys. Kolmiulotteisessa tilassa vektoreiden ja vektorikenttien suuntien hahmottaminen on todella vaikeata ilman minkäänlaista referenssipiirrosta. Tehtävän alkutietojen avulla piirretty kuva voi antaa yllättävän nopeastikin vastauksia siitä, minkälainen tehtävän lopullinen ratkaisu voisi olla. Omaa vastausta voi verrata piirrettyyn kuvaan ja pohtia, voisiko se olla oikein. Lisäksi kuvan piirtäminen mahdollistaa nopeamman prosessoinnin sille, miten tehtävää mahdollisesti kannattaisi lähteä ratkaisemaan. Esimerkiksi kuvassa 31 on esitetty kenttäteorian kursseilla esiintynyt tehtävä muunnettuna STACK-järjestelmään. Tehtävässä on tällä hetkellä piirrettynä tarvittavat vektorit, mutta se voitaisiin esimerkiksi helposti muokata siten, että opiskelijan pitäisi itse sijoittaa ne oikein.



Kuva 31. Kenttäteorian perustehtävä muunnettuna STACK-järjestelmään, jossa etsitään tasovarauksen ρ_s varaustiheyttä.

Joka tapauksessa, kun sähkökentän voimakkuuksia kuvaavat vektorit ovat liitetty kuvaan, voidaan heti huomata, miten tehtävää lähdetään ratkaisemaan ja mikä on teoriassa oikea vastaus. Tasovaraus aiheuttaa pisteeseen *y*-suuntaisen komponentin, kun taas viivavaraus aiheuttaa +*z*- ja +*y*-komponentit. Mikäli *y*-komponentit kumoavat toisensa, jää jäljelle vain *z*-komponentti, joten oikea vastaus tullaan ratkaisemaan yhtälöstä

$$-\vec{E}_{\rho_s} = \vec{E}_{\rho_{ly}}. \tag{6}$$

Ilman kuvaa tehtävän hahmottaminen ja ratkaisun aloittaminen voi siis olla hankalaa ja ratkaisun oikeellisuuden todentaminen vielä vaikeampaa. Kuvan piirtäminen mahdollistaa myös oikeanlaisen tarkastuksen esimerkiksi tenttivastauksessa. Mikäli referenssikuvaa ei ole piirretty, voi tarkastajan olla vaikea käsittää, miten tehtävää on yritetty ratkaista ja pisteytys vaikeutuu tai muuttuu jopa mahdottomaksi.

Käytännössä STACK-järjestelmässä on paljon erilaisia ulottuvuuksia ja sitä voidaan soveltaa hyvin monimutkaisiinkin aiheisiin. Parhaimpiin sovelluskohteisiin voisi sisällyttää ne aihepiirit, joissa korostuvat matematiikka ja visuaalisuuden tarve, mutta mahdolliset sovelluskohteet eivät rajoitu yksinomaan näihin.

5.3 STACK-Järjestelmän uudistuksia

STACK-järjestelmää kehitetään koko ajan ja uusia ominaisuuksia ja mahdollisuuksia tulee tehtävien tekijöiden tarpeita varten. Tällä hetkellä STACK mahdollistaa tehtävissä useamman syötekentän käyttämisen, mutta tehtävät ovat aina ns. yksivaiheisia, joissa opiskelija kirjoittaa syötekenttiin vastaukset ja sitten tarkastetaan, menikö ratkaisu oikein vai väärin, ja tuotetaan palaute. Tämän jälkeen väärän vastauksen annettuaan, opiskelija suorittaa tehtävän uudelleen. Yksi merkittävä kehitteillä oleva mahdollisuus onkin ns. tilamuuttujien käyttö. Tilamuuttuja-järjestelmä mahdollistaa sen, että opiskelijan antama vastaus otetaan tarkempaan syyniin, mikäli se oli väärin. Järjestelmässä liikutaan eri ’tilojen’ välillä, jotka toimivat omalla tavallaan vastauspuun solmujen tapaan, mutta ne eivät ole varsinaisesti palautteita vaan uusia tehtäviä opiskelijan yksilöllisestä oppimishistoriasta riippuen. Näiden tilojen avulla ja annetun väärän vastauksen jälkeen, voidaan opiskelija siirtää ratkaisemaan tehtävää paloittain painottaen perusasioita, joista oikea vastaus sitten lopuksi rakentuu. Järjestelmän avulla voidaan siis puuttua tehtävän tekohetkellä esiintyviin virheisiin paremmin kuin vasta pitkän tehtävän lopussa. Monimutkaisemmat tehtävät voidaan alun alkaenkin pilkkoa tämän takia pienempiin paloihin, joihin opiskelijan aiemmin tehdyt vastaukset vaikuttavat (Harjula, Malinen, Rasila, 2017).

Sähköisten komponenttien piirrosmerkkien piirtäminen JSXGraph-elementtien avulla voi olla jokseenkin työläs prosessi, kun uutta tehtävää aloitetaan luomaan. Tämän seurauksena onkin toivottu, että kunpa komponentit saataisiin implementoitua JSXGraph-kirjastoon omina elementteinään, joita voitaisiin käyttää samalla tavalla kuin tämänhetkisiäkin. Tehtävien koodi siistiytyisi myös tämän johdosta.

5.4 Moodle ja TIM-järjestelmä

Vuosien varrella Moodle on vakiinnuttanut paikkansa osana korkeakoulujen opetusta verkosta löytyvänä oppimisalustana. Myös Vaasan yliopistossa opettajat jakavat kurssin materiaalin pääsääntöisesti Moodlen kautta. Materiaalin jakaminen Moodlen välityksellä on suhteellisen vaivatonta, mutta sen moduulimainen rakenne rajoittaa sinne lisättävän materiaalin esimerkiksi tiettyhin tiedostotyyppeihin. Tämän seurauksena kurssialue näyttää yleensä vain kokoelmalta erilaisia pdf-tiedostoja ja linkkejä luentojen nauhotteisiin. Mikäli kurssi sivun haluaisi näyttävän luontevalta ja yhtenäisemmältä, on syytä tarkastella muita vaihtoehtoja (Moodle, 2020).

TIM (The Interactive Material) on Jyväskylän yliopistossa Informaatioteknologian tiedekunnassa kehitetty avoimen lähdekoodin pilvipalvelu, jonka tarkoituksena on tuottaa interaktiivista materiaalia verkkoon. TIM:ssä luotua sivustoa voi ajatella eräänlaisena sähköisenä kirjana, johon sisällytetään luentomateriaalia perinteisen kirjan tapaan lukuina, jotka on sijoitettu loogiseen järjestykseen. Materiaali voi olla perinteistä staattista tekstiä, kuvia, videoita, tiedostoja yms., mutta TIM:iin luodut e-kirjat voivat sisältää myös interaktiivista materiaalia, kuten upotettuja tehtäviä, joihin opiskelijat voivat vastata. Alustan vahvuutena voidaan pitää sen kehittynyttä muokattavuutta, esimerkiksi Moodleen verrattuna. TIM:iin voidaan melko vapaasti koodata useilla eri kielillä erilaista materiaalia, kuten esim. tehtäviä ja ne voidaan upottaa kätevästi muun materiaalin sekaan tukemaan muussa materiaalissa esitettyjä asioita (Jyväskylän yliopisto, 2021).

TIM-alustalle voidaan myös upottaa STACK-tehtäviä. Tämä mahdollistaisi tehtävien ja oppimateriaalin yhteensitomisen paljon tiukemmin ja paremmin kuin Vaasan yliopistossa tällä hetkellä oleva järjestely, jossa luentomateriaali löytyy Moodlesta ja STACK-tehtävät erikseen omalta palvelimeltaan. Materiaalin keskittäminen yhteen paikkaan muodostaisi opiskelijalle paljon tiiviimmän ja yhtenäisemmän opiskeluaineiston, jossa ei tarvitsisi itse hyppiä eri sivustojen ja palvelinten välillä.

Esimerkiksi piirianalyysin kurssilla TIM:ssä voisi olla normaaliin tapaan aluksi käyty läpi Kirchhoffin lakien teoria ja heti perään olisi STACK-tehtävä tai tehtäviä liittyen siihen. Näin edellä mainittu teoreettinen pohjustus ja sen tarkoitus saataisiin heti realisoitumaan opiskelijan silmissä, mikä voisi parantaa substanssin ymmärtämistä.

STACK-tehtävää upotettaessa TIM:iin sitä täytyy kuitenkin muuttaa hieman, jotta se toimisi alustalla. Piirianalyysissä käytettyjä tehtäviä on upotettu TIM:iin onnistuneesti pienillä muokkauksilla XML-tiedoston koodiin. Toistaiseksi on kokeiltu vain staattisia tehtäviä, joissa opiskelijan vastausta ei lueta JSXGraph-elementtiä liikuttamalla, vaan opiskelija syöttää vastauksensa syötekenttään. Tehtävien upottamisen avulla aikaisemmin kuvattu tilanne materiaalin yhdistämisestä olisi täysin mahdollista jo nyt. Dynaamisten tehtävien osalta tilanne vaatii kuitenkin lisää tutkimista, sillä vastauskenttiä ei välttämättä saa yhdistettyä elementteihin nykyisellä menetelmällä.

5.5 Käänteinen opetus

Verkkomateriaali ja ohjelmistojen kehitys on mahdollistanut myös uusien opetusmetodien käyttöönoton. Yksi viime aikoina yleistyneistä kurssin järjestämistavoista on ns. *käänteinen opetus* (engl. flipped classroom). Käänteisen opetuksen keskiössä on nimen mukaisesti päinvastainen tilanne kuin normaalissa opetuksessa, jossa luennoilla käydään teoria, jota sitten syvennetään kotona tekemällä erilaisia soveltavia harjoituksia. Käänteisessä opetuksessa taas teoriat pyritään käymään kotona ja mahdolliset soveltavat harjoitukset suoritetaan yhdessä ohjaavan opettajan opastuksella. Tämänlainen toimintamalli mahdollistaa opettajan ja opiskelijoiden välisen vuorovaikutuksen keston kasvamisen.

Maailmanlaajuisesti käänteisen opetuksen kehittäjää ja sen modernin mallin luojana voidaan pitää Salman Khania, joka perusti vuonna 2006 verkossa toimivan Khan Academy -nimisen oppimisympäristön (Ash, 2012). Oppimisympäristöön on koottu erilaisia lyhyitä opetusvideoita, jotka ovat ilmaiseksi kaikkien saatavilla (Khan Academy,

2020). Tätä pidetään yhtenä käänteisen oppimisen kulmakivenä, vaikka nykyiset ohjelmistot mahdollistavat paljon erilaisia ja syvempiä menetelmiä.

Suomessa käänteistä luokkahuonetta, *”flippausta”,* on kokeiltu ja toteutettu useammassa yliopistossa, mutta Itä-Suomen yliopistossa siihen ja muihin joustaviin oppimismetodeihin on panostettu paljon (Itä-Suomen yliopisto, 2020 & Valtonen, Leppänen, Hyypiä, Kokko, Manninen, Vartiainen, Sointu & Hirsto, 2020). YLE:n julkaisemassa artikkelissa *”Opettaja luopui luennoista – yhtäkkiä lähes kaikki opiskelijat läpäisivät vaikean yliopistokurssin”* (YLE, 2017), Itä-Suomen yliopiston lehtori ja oppimisympäristöjen kehittämispäällikkö Markku Saarelainen käy läpi sen tuomia etuja. Esiin nousee esimerkiksi se, kuinka aikaisemmin vaikeasta kenttäteorian kurssista harva pääsi läpi ja miten flippauksen jälkeen valtaosa suoriutuu samoista vaadittavista asioista, tosin pieniä osa-alueita kerrallaan vain testaten, huomattavasti paremmin. Samalla myös opettamiseen tarvittavat tunnit putosivat neljännekseen. Saarelaisen mukaan opiskelijoille jäi kääntämisen seurauksena enemmän aikaa asioiden varsinaiseen sisäistämiseen. Opiskelijoiden näkökulmasta kurssit vaativat kuitenkin edelleen työtä, mutta riittävän hyvillä taustatiedoilla niiden suorittaminen kyllä onnistuu. Saatavillani ei ollut tästä tutkimusta, josta olisi selvinnyt laadullinen vertailu oppimistuloksissa, jossa esim. koko kurssin asiat testattaisiin yhdellä kertaa.

STACK-järjestelmää voidaan valjastaa tukemaan *käänteistä opetusta*, sillä tehtäviä voidaan ratkoa kotona omalla kotikoneella. Tehtäviä voitaisiin sisällyttää muun materiaalin kanssa käteviksi oppimiskokonaisuuksiksi, joiden avulla opiskelijat voivat sisäistää tarvittavat tiedot ja taidot paremmin.

6 Opiskelijoiden ja opettajien kokemukset STACK-järjestelmän käytöstä

Tässä luvussa kerrotaan STACK-tehtäviin ja niiden käyttöönottoon liityvistä kokemuksista opiskelijoiden ja opettajan näkökulmasta. Tehtävien toimivuuden varmistamiseksi niille suoritetaan testausta ennen kuin ne ovat täysin valmiita ja niitä annetaan eteenpäin. Testauksessa voi kuitenkin jäädä joitain teknisiä ongelmia huomiotta varsinkin, jos tehtävä ja siihen liittyvä koodi on monimutkainen ja yleensä nämä nousevatkin esille vasta varsinaisen käytön yhteydessä. Keväällä 2020 järjestetyllä Piirianalyysi A -kurssilla joitain tehtäviä testattiin ensimmäistä kertaa opiskelijoiden avulla. Mikäli opiskelija huomasi tai epäili tehtävän olevan virheellinen esimerkiksi vastauksen osalta, oli hänellä mahdollisuus olla yhteydessä heti opettajaan. Opiskelija ilmoitti huomion yhteydessä myös ajankohdan jolloin oli suorittanut kyseisen tehtävän. STACK-palvelimelta voitiin käydä tarkastamassa juuri kyseisen opiskelijan tekemä tehtävä ja sen variaatio, jolloin virheen paikantaminen onnistui helposti. Näin tehtävissä esiintyneet ongelmat saatiin eliminoitua nopeasti. Yleisesti tekniset ja aivan alussa ilmenneet ongelmat liittyivät lähinnä muuttujien määrittelyyn tehtävissä, jolloin korjaaminenkin oli suhteellisen nopeata ja se yleensä ratkaistiin jo parin tunnin sisällä ilmoituksesta.

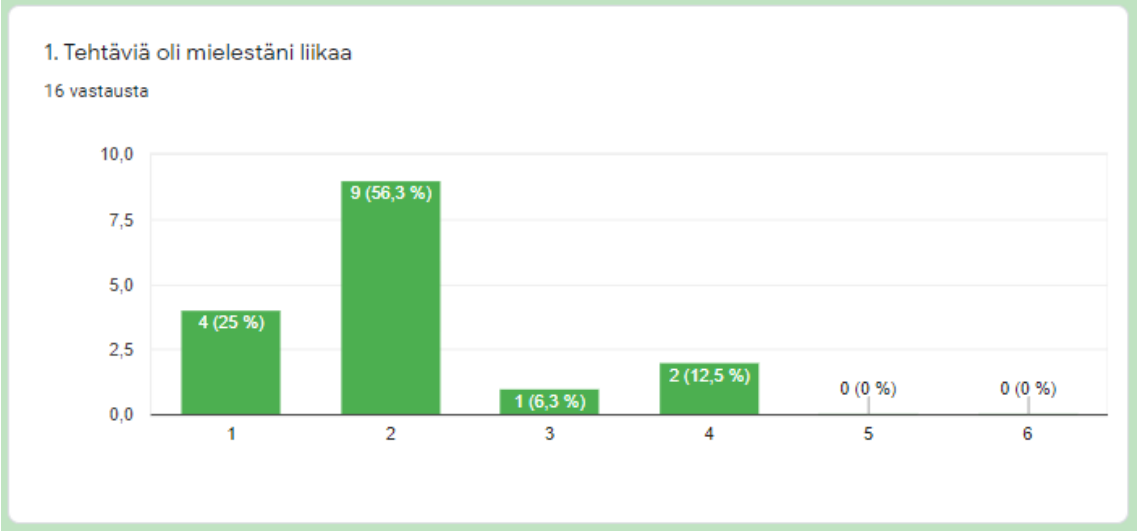
6.1 Vuoden 2020 palautekyselyn tulokset

Toimivuuden tarkastamiseksi opiskelijoilta kerättiin myös normaalin kurssipalautteen lisäksi palautetta liittyen STACK-järjestelmän käyttöön. Palautteen tehtävänä oli kartoittaa opiskelijoiden mietteitä ja huomioita tehtävistä ja yleisesti STACKin käyttöön liittyviä kokemuksia. Palaute kerättiin Google Forms -lomakkeen (liite 1) avulla ja vastaaminen oli täysin vapaaehtoista.

Palautekyselyyn vastasi 25:stä kurssin loppuun suorittaneesta opiskelijasta 16 (N=16). Kysely koostui kymmenestä eri kysymyksestä, joiden tarkoituksena oli kerätä tietoa tulevaa kehitystä varten. Kahdeksaan kysymykseen vastattiin yksinkertaisesti

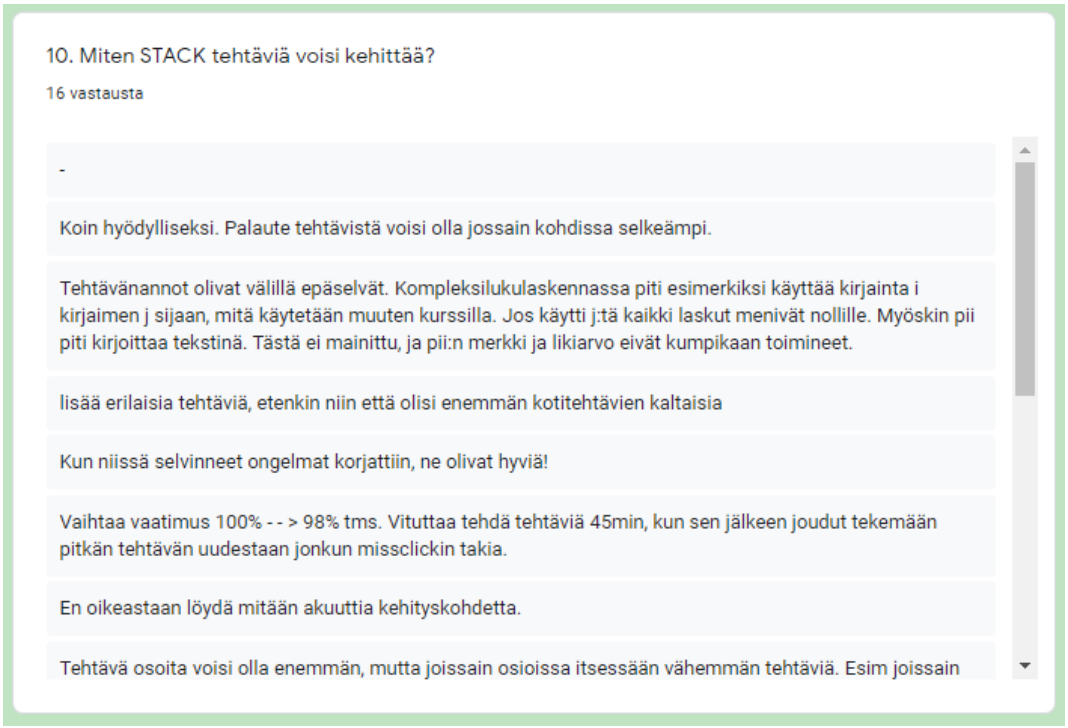
valitsemalla sopiva numero asteikolla 1–6, jossa yksi tarkoitti ”täysin eri mieltä” ja kuusi ”täysin samaa mieltä”. Kahdessa viimeisessä kysymyksessä opiskelijoille annettiin vapaa sana positiivisista kokemuksista ja kehittämistarpeista.

Palautekyselyn tuloksista (liite 2) voidaan tulkita, että pääosin STACK-tehtävät olivat opiskelijoiden mielestä mielekkäitä eikä niitä ollut liian paljon tarjolla. Kuvassa 32 on esitetty opiskelijoiden antamat vastaukset palautteen kysymykseen 1, ”Tehtäviä oli mielestäni liikaa”.

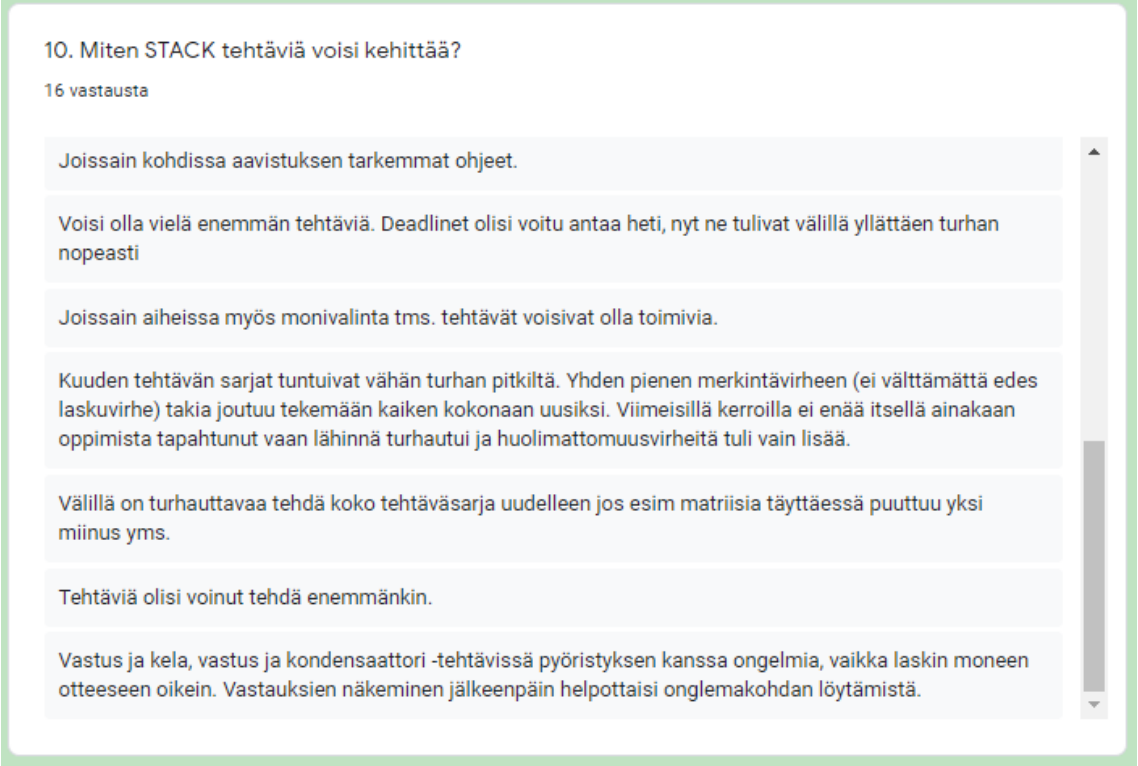


Kuva 32. Palautekyselyn kysymyksen 1 vastaukset (Palautekysely 2020, Liite 2).

Kuvasta nähdään, että reilusti suurin osa vastanneista oli sitä mieltä, että tehtäviä ei ollut liikaa. Joskin sanallisessa palautteessa kehitysideoista (kysymys 10) mainitaan, että joissain kohdissa tehtäviä olisi voinut olla hieman vähemmän per sarja turhautumisen välttämiseksi. Toisaalta yleisesti ottaen toivottiin myös, että tehtäviä olisi ollut paljon enemmän. Erilaisia tehtäväsarjoja oli kurssin aikana 25 ja ne koostuivat 1–6 tehtävästä riippuen tehtävän pituudesta. Kuvissa 33 ja 34 on esitetty kehitysideat sanallisesta palautteesta.



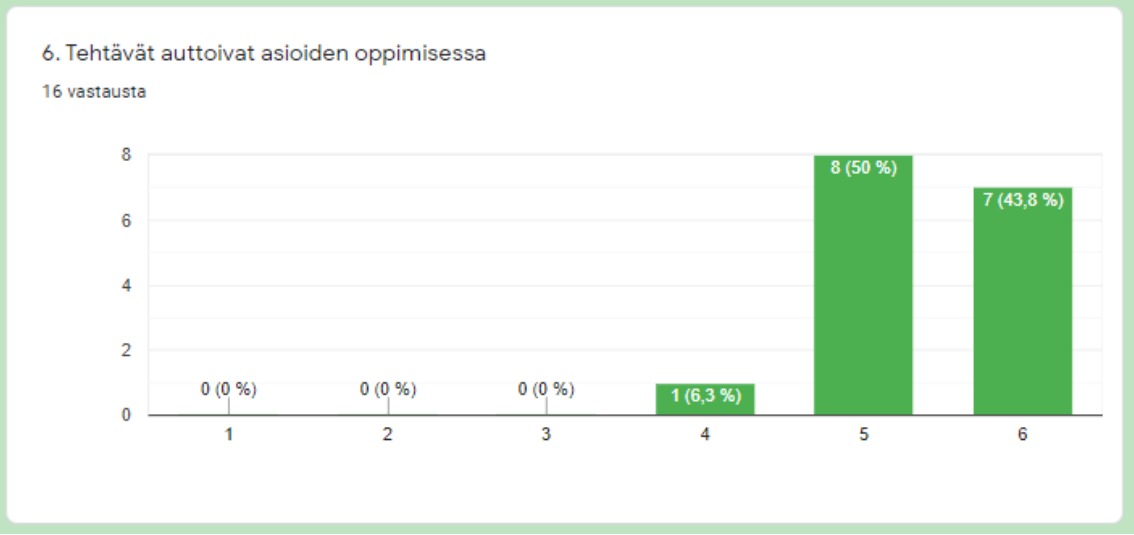
Kuva 33. Sanallinen kysymys 10, alkuosa (Palautekysely 2020, liite 2).



Kuva 34. Sanallinen kysymys 10, loppuosa (Palautekysely 2020, liite 2).

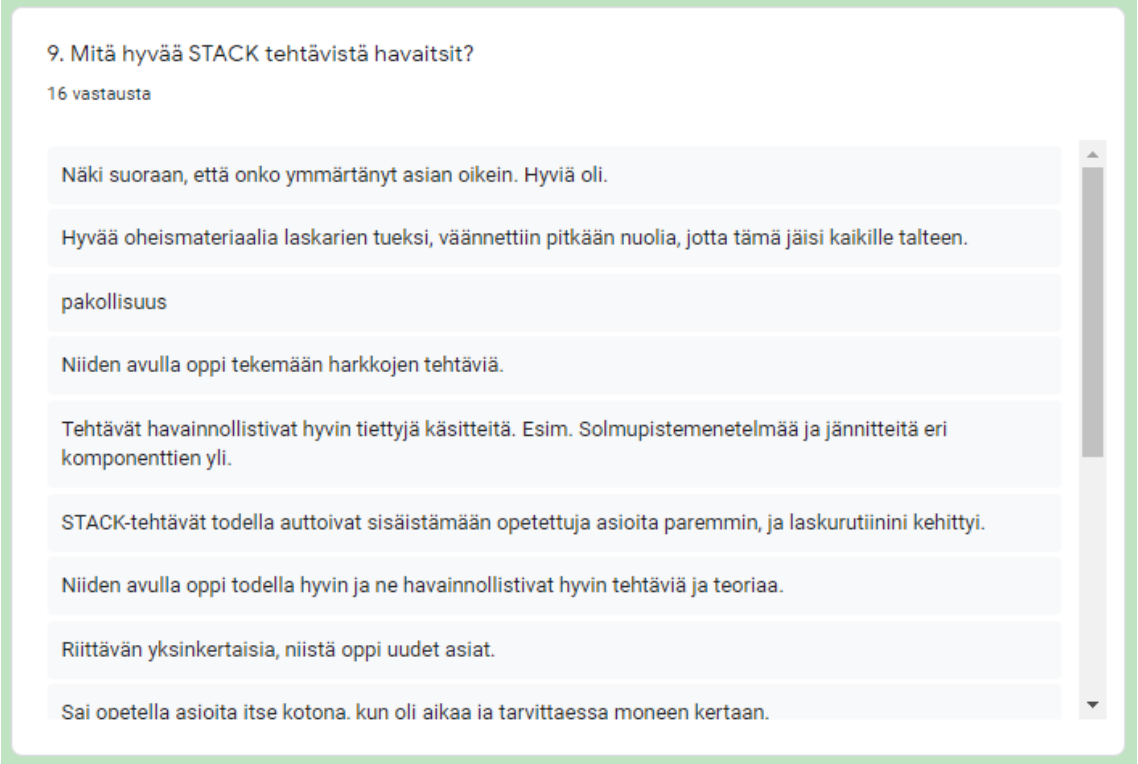
Muina ehdotettuina kehityskohteina esitettiin kuvien 33 ja 34 mukaisesti, että lisää tehtäviä pitäisi saada ja että tehtävien antamat palautteet ja ohjeet tehtäviin voisivat olla parempia. Joissain tehtävissä virheen tekeminen tarkoitti myös sitä, että aikaa joutui käyttämään paljon, sillä tehtävä täytyi tehdä uudestaan.

Yksi tärkeä kysymys palautelomakkeessa oli kysymys 6 ”Tehtävät auttoivat asioiden oppimisessa” ja sen tulokset ovat esitetty kuvassa 35.

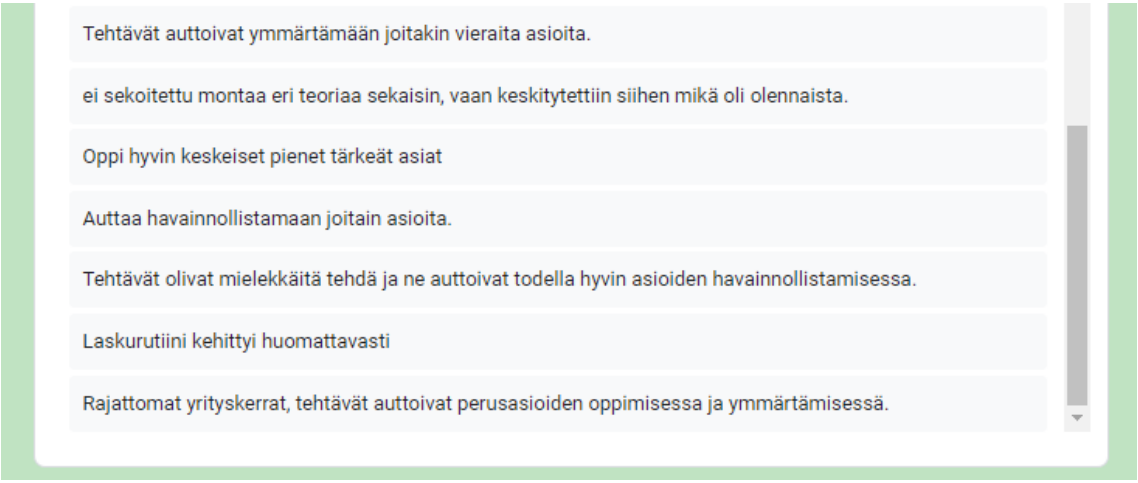


Kuva 35. Lomakkeen kysymys 6 (Palautekysely 2020, liite 2).

Jokainen palautekyselyyn vastanneista oli ainakin jossain määrin sitä mieltä, että tehtävistä oli apua niiden käsittelemien aiheiden sisäistämisessä. Vastanneista seitsemän olivat täysin samaa mieltä ja kahdeksan lähes samaa mieltä väittämän kanssa. Opiskelijat siis kokivat tehtävien olevan hyödyllisiä muun materiaalin lisäksi. Tämä kävi ilmi myös sanallisen kysymyksen 9 palautteesta. Kuvissa 36 ja 37 on esitetty positiivinen sanallinen palaute.



Kuva 36. Sanallinen kysymys 9, alkuosa (Palautekysely 2020, liite 2).



Kuva 37. Sanallinen kysymys 9, loppuosa (Palautekysely 2020, liite 2).

Palautteesta näkee, että uusien asioiden oppiminen nousee useasti esille. Myös tavoiteltu laskurutiinin kehittyminen ja sen harjoitteleminen onnistui STACK-tehtäviä tekemällä. Ne auttoivat lisäksi myös laskuharjoitusten ratkomisessa.

6.2 Tiivistelmä mielipiteistä STACK-järjestelmään liittyen

Edellä mainittuja kyselyn tuloksia voidaan pitää yleisesti erittäin hyvinä ja sanallisesta palautteesta nähdään, että STACK-tehtävät toimivat juuri niin kuin olikin alunperin ideana. Tehtävien tarkoituksena oli toimia eräänlaisena siltana luentojen ja laskuharjoitusten välillä, jotta luennolla käydyt teoriat ja niiden sisäistäminen vahvistuisi entisestään. Tehtävien yksinkertaisuus ja niiden pakollinen suorittaminen loivat opiskelijoille paljon loivemman reitin asioiden sisäistämisen ja niiden soveltamisen osalta. Toistot toivat myös rutiinia tehtävien ratkaisemisessa. Palautteen avulla saatiin kerättyä myös arvokkaita kehittämisideoita liittyen tehtäviin ja niiden esitystapaan.

Kehittäjän näkökulmasta STACK-järjestelmä on suhteellisen yksinkertainen toiminnoiltaan ja pienehköllä ohjelmointitaustalla voi pystyä luomaan paljonkin tarvittavia asioita. Yksinkertaisimpia tehtäviä voi luoda käytännössä ilman aiempaa osaamista ohjelmoinnin osalta, kunhan tarvittavat avainelementit eli tieto siitä, mitä kaikkea tehtävä vaatii toimiakseen, on selvä. Tämän diplomityön liitteenä (liite 3) on ohjeistus hyvin yksinkertaisen tehtävän luomiseen STACK-järjestelmän versiossa 3 sekä yksinkertaisen JSXGraph-kuvion liitäminen. Tehtävien suuri etu on niiden vastauspuu ja sen tekeminen vaatii opetuskokemusta ja tietynlaista ajattelutapaa, jotta voi itse miettiä erilaisia kohtia tehtävässä, missä opiskelija voi tehdä erilaisia virheitä. Täydellisen vastauspuun tekeminen vaatiikin paljon keskittymistä ja ennen kaikkea aikaa.

Ajallisesti tehtävien luominen riippuu pitkälti siitä, minkälainen tehtävä on kyseessä. JSXGraphin implementoiminen ja varsinkin dynaamisten tehtävien kehittäminen vie monimutkaisuuden seurauksena paljon enemmän aikaa, kuin pelkästään sanallisten matemaattisten tehtävien koodin tekeminen. Tämä ei kuitenkaan tarkoita, että näin olisi aina. Tehtävänkehityksessä suuri osa ajasta kuluu vastauspuun kehittämiseen siten, että tehtävä osaa antaa yksityiskohtaista palautetta, joka riippuu opiskelijan tekemästä virheestä. Automaattisessa tarkastuksessa on tärkeää ottaa huomioon, että opiskelijalle tarjotaan vähintäänkin oikea vastaus virheen seurauksena, jotta hän saa jonkinlaisen

tarttumapinnan. Monimutkaisten vastauspuiden luominen on erittäin työlästä ja sen hiomisessa kannattaakin yrittää miettiä tehokasta ajankäyttöä.

7 Johtopäätökset ja yhteenveto

Tämän diplomityön aiheena on selventää STACK-järjestelmän hyödyntämistä Vaasan yliopiston sähkötekniikan opetuksessa. Työn päämääränä oli vastata seuraaviin tutkimuskysymyksiin:

- Miten STACK-järjestelmää voidaan hyödyntää piirianalyysin käsittelyssä?
- Miten opiskelijalle voidaan visualisoida tehtävän ratkaisun kannalta oleellisia asioita kuten piirikaavioita, kuvaajia yms.?
- Voidaanko luoda tehtävätyyppejä, jossa kirjoittamista ja laskemista ei olisi, vaan ratkaisu perustuu interaktiivisen kuvan käyttämiseen?
- Mitä mieltä opiskelijat ja opettajat ovat työssä kehitetyistä tehtävistä ja niiden toimivuudesta?

STACK-Järjestelmää on käytetty aikaisemminkin yliopistolla matematiikan opetuksessa, mutta vuodesta 2019 eteenpäin sitä on hyödynnetty myös sähkötekniikan puolella piirianalyysin kursseilla. Diplomityön saavutukset ja tulokset voidaan tiivistää seuraavasti:

- STACK-järjestelmään tutustuminen ja sen toimintojen kartoittaminen sähkötekniikan osalta.
- JSXGraph-kirjaston hyödyntäminen ja soveltuvuuden mahdollinen testaaminen piirianalyysissä.
- Interaktiivisten piirianalyysin tehtävien onnistunut luominen JSXGraph-kirjaston avulla.
- Dynaamisten piiritehtävien, joissa opiskelijan ei tarvitse kirjoittaa vastausta, luominen onnistuneesti.
- Kurssilla Piirianalyysi A käytettyjen STACK-tehtävien kirjoittaminen ja testaaminen.

Lisäksi opiskelijoilta saatu palaute oli suurimmaksi osaksi positiivista ja voidaan todeta, että tehtävistä oli hyötyä.

Tämän diplomityön aluksi käydään teoriaa ja tarkastellaan erilaisia STACK-järjestelmään liittyviä käsitteitä kuten etäopetusta ja interaktiivisuutta. Seuraavaksi selvitetään, mikä STACK-järjestelmä yleisesti on ja minkälaisia erilaisia työkaluja se tarjoaa kehittäjälle, opettajalle ja opiskelijoille. Varsinaisen tehtävän luominen sisältää kolme isoa aluetta, joiden määrittely täytyy löytyä tehtävästä:

- kysymyksen muuttujat
- varsinainen tehtäväteksti ja koodi
- vastauspuu.

STACK-tehtäviä lähdettiin kehittämään aluksi miettimällä, miten opiskelijoille saataisiin välitettyä tarvittavat visuaaliset seikat, kuten piirikaavio. Projektiryhmässä (Maarit Vesapuisto, Timo Vekara, Matti Laaksonen ja Otto Ellonen) päädyttiin kokeilemaan JSXGraph-JavaScript -kirjastoa, joka mahdollistaa geometristen elementtien piirtämisen koordinaatistoon. Kirjaston avulla voitiin näin ollen luoda erilaisia sähköisiä komponentteja, joiden avulla saatiin aikaiseksi haluttu piiri. Elementtien koordinaateiksi voidaan sijoittaa muuttujia, jotka voidaan määritellä halutulla tavalla. Tämä mahdollistaa sen, että elementtejä voi satunnaistaa ja sitä kautta piiri arvotaan jokaisella suorituskerralla. Sen seurauksena on mahdollista luoda nopeasti eri versioita samasta piiristä, jossa oikea vastaus todennäköisesti myös muuttuu. Tehtävän oikean vastauksen logiikka voidaan myös määritellä komponenttien koordinaattien avulla, joten jokaiselle eri versiolle ei tarvitse luoda omaa vastaustaan, mikä nopeuttaa tehtävien luomista.

JSXGraph ja sen elementtien koordinaattien sitominen muuttujiin mahdollistaa myös sen, että tehtävistä voidaan luoda interaktiivisia versioita, joissa opiskelijan ei tarvitse itse kirjoittaa vastaustaan vaan vastaus voidaan antaa klikkaamalla kuvaa ja sen elementtejä, joita opiskelija voi siirtää oikeille paikoilleen. Interaktiiviset tehtävät tarjoavat

uudenlaisen lähestymistavan piirianalyysiin perinteisiin kirjallisuudesta löytyviin tehtäviin verrattuna.

Työssä käytiin lyhyesti myös läpi eri näkökulmia STACK-järjestelmän tulevaisuuden kannalta ja miten ja missä sitä voisi hyödyntää myös muilla sähkötekniikan osa-alueilla. JSXGraphia voitaisiin hyödyntää vektorimatematiikkaa soveltavissa aiheissa, kuten esimerkiksi kenttäteoriassa, jossa visuaalisuus on avainasemassa substanssin ymmärtämisessä.

Opiskelijoiden antaman palautteen perusteella STACK-järjestelmästä on ollut arvokasta hyötyä kevään 2020 piirianalyysin kurssilla ja se on auttanut heitä paremmin sisäistämään kurssilla käytyjä perusteita ja teoriaa. Tulosten perusteella olisi hyvä tuottaa vielä lisää tehtäviä ja niitä voitaisiin kehittää myös muille kursseille, jotta myös siellä esiintyvät uudet konseptit pystyttäisiin sisäistämään paremmin. Vanhoja tehtäviä on myös hyvä paikkailla tulevankin palautteen osalta ja niiden kehittämistä esimerkiksi vastauspuun osalta voi olla hyvä tarkastella.

STACK-järjestelmä ja muut interaktiiviset oppimisympäristöt kehittyvät koko ajan ja onkin mielenkiintoista seurata, mitä uusia mahdollisuuksia ilmaantuu. Etäopetuksen ja digitalisaation seurauksena tarvetta on ja uusia järjestelmiä tullaan kehittämään vanhojen rinnalle. Opetus voi tapahtua tulevaisuudessa enenevästi virtuaalimaailmassa.

STACK-järjestelmän hyödyt voidaan tiivistää opiskelijan ja opettajan näkökulmasta seuraavien luetteloiden mukaisesti. Opiskelijoiden näkökulmasta STACK-järjestelmän hyödyt ovat:

- Laskurutiinin kehittäminen triviaalitehtävien avulla.
- Tehtäviä voi tehdä ajasta ja paikasta riippumatta omaan tahtiin.
- Tehtävistä saa heti suoran palautteen omasta suorituksesta ja uudelleen voi yrittää heti.

- Tehtävien avulla piirianalyysin aihealueiden käsittäminen helpottuu ja teoriasta varsinaisiin laskuharjoituksiin siirtyminen helpottuu.

Opettajan näkökulmasta järjestelmän hyötyjä ovat:

- Automaattinen tarkastaminen huolehtii opiskelijoiden vastauksista, joten järjestelmää voidaan käyttää isoilla osallistujamäärillä.
- Opiskelijat saavuttavat tarvittavan osaamistason ennen laskuharjoituksia, jolloin niiden läpikäyminen ja selittäminen koko ryhmälle on helpompaa.
- Yksinkertaisten, mutta tärkeiden perusteiden painottaminen on helppoa ja ajan käytön kannalta järkevää tehtävien avulla.

Lähteet

Abacus (2021). *ABACUS – Materiaalipankki*. Aalto-yliopisto. [Verkkosivu] Noudettu 15.8.2020 osoitteesta: https://abacus.aalto.fi/?lang=fi

Ash, K. (2012). *Educators Evaluate 'Flipped Classrooms'*. EducationWeek 2012. [Verkkojulkaisu] Noudettu 20.3.2021 osoitteesta: https://www.edweek.org/teaching-learning/educators-evaluate-flipped-classrooms/2012/08

Bennett, S., Maton, K., Kervin, L. (2008). *The 'Digital Natives' Debate: A Critical Review of the Evidence*. British Journal of Educational Technology, Vol. 39, Issue 5. DOI: https://doi.org/10.1111/j.1467-8535.2007.00793.x

Brazina, D., Fojtik, R., Rombova, Z. (2014). *3D Visualization in Teaching Anatomy.* Procedia - Social and Behavioral Sciences, Volume 143, August 2014, sivut 367–371. DOI: https://doi.org/10.1016/j.sbspro.2014.07.496.

Chirumamilla, A., Sindre, G., Nguyen-Duc, A. (2020). *Cheating in e-exams and paper exams: the perceptions of engineering students and teachers in Norway*. Assessment & Evaluation in Higher Education, Vol. 45, issue 7, 940–957, 2020. DOI: 10.1080/02602938.2020.1719975.

Duffy, G., Sorby, S., Bowe, B. (2016). *Visualizing Electric CIrcuits: The Role of SPatial Visualization Skills in Electrical Engineering*. 70th Midyear Technical Conference: Graphical Expressions of Engineering Design. Noudettu 20.3.2021 osoitteesta: https://commons.erau.edu/asee-edgd/conference70/papers-2016/16/.

Ellonen, O., Vesapuisto, M., Vekara, T. (2020). *Experiences on Development and Design of STACK Problems for Circuit Analysis*. Athens Journal of Technology & Engineering, Vol 7, Issue 3, p. 185–204, 2020. DOI: https://doi.org/10.30958/ajte.7-3-2

EXAM (2021). *Tulevaisuuden tenttiminen*. [Verkkosivu]. Noudettu 22.3.2021 osoitteesta: https://e-exam.fi/

Friesen, N., Osguthorpe, R. (2018). *Tact and the pedagogical tirangle: The authenticity of teachers in relation.* Teacher and Teacher Education, Volume 70, February 2018, sivut 255–264. DOI: https://doi.org/10.1016/j.tate.2017.11.023.

Gee, A. G., Li, H., Grinstein, G. (2005). *Dynamic and Interactive dimensional anchors for spring-based visualization.* [Verkkodokumentti] Noudettu 21.3.2021 osoitteesta: https://www.researchgate.net/publication/228618579_Dynamic_and_interactive_dimensional_anchors_for_spring-based_visualizations¨.

Guzmán, M. (2002). *The Role of Visualization in the Teaching and Learning of Mathematical Analysis.* International Conference on the Teaching of Mathematics (at the Undergraduate Level), Crete, Greece 2002. Noudettu 22.3.2021 osoitteesta: https://eric.ed.gov/?id=ED472047

Harjula, M., Malinen, J., Rasila, A. (2017). *STACK with state.* MSOR Connections, Vol. 15, No. 2, 2017. DOI: https://doi.org/10.21100/msor.v15i2.408

Harmon, O., Lambrinos, J. (2008). *Are Online Exams an Invitation to Cheat?* The Journal of Economic Education, 39(2):116-125, 2008. Noudettu 22.3.2021 osoitteesta: https://www.researchgate.net/publication/23645255_Are_Online_Exams_an_Invitation_to_Cheat.

Huhtala, S. (2000). *Lähihoitajaopiskelijan oma matematiikka*. Väitöskirja, Helsingin yliopiston opettajankoulutslaitos, sivut: 1–4, 34–37, 2000. ISBN: 951-45-9353-7

Irawan, M., Mukhlash, I., Adzkiya, D., Sanusi, D. (2019). *Develpoment of trigonometric visualization concepts to increase the study motivations of SMK students.* Journal of Physics: Conference Series 1218 (2019) DOI: 10.1088/1742-6596/1218/1/012049

Itä-Suomen yliopisto (2020). *Tervetuloa flippauksen pariin*. [Verkkosivu] Noudettu 21.3.2020 osoitteesta: https://sites.uef.fi/flippaus/

Jarvis, P. (1993) *Oppimisen paradokseja myöhäismodernissa*. Aikuiskasvatus, Vol 13 nro. 3 1993, sivut: 181–187. DOI: https://doi.org/10.33336/aik.96892.

JSXGraph (2020). *JSXGraph – Dynamic Mathematics with JavaScript*. [Verkkosivu]. Noudettu 18.7.2020 osoitteesta: https://jsxgraph.uni-bayreuth.de/wp/index.html

JSXGraph Wiki (2020). *JSXGraph Wiki – Point*. [Verkkosivu]. Noudettu 15.7.2021 osoitteesta: https://jsxgraph.uni-bayreuth.de/wiki/index.php/Point

Jyväskylän yliopisto (2021). *TIM - oppimisympäristö.* Jyväskylän yliopisto. Noudettu 9.8.2021 osoitteesta: https://tim.jyu.fi/view/tim/TIM-esittely/fi#tim-projekti

Khan Academy (2020). *Khan Academy - homepage*. [Verkkosivu] Noudettu 20.3.2021 osoitteesta https://www.khanacademy.org/about

King, C. G., Guyette, R. W. Jr., Piotrowski, C. (2009) *Online Exams and Cheating; An Empirical Analysis of Business Students' Views.* Journal of Educators Online, Vol. 6, nro. 1, 2009. Noudettu 21.3.2021 osoitteesta: https://eric.ed.gov/?id=EJ904058

Klischat, C., Becker, P., Vasko, M. (2019). *STACK is more than Maths – Development of Online-Problems for Mechanics and Electrotechnics.* 1st International STACK conference 2018, Fürth, Germany, 15–16 November 2018. DOI: 10.5281/zenodo.2577116

Massachusetts Institute of Technology (2021). *Open Courseware - Homepage*. [Verkkosivu] Noudettu 20.3.2021 osoitteesta: https://ocw.mit.edu/index.htm?utm_source=openlearning&utm_medium=ocwpage.

Maxima (2020). *Maxima, a Computer Algebra System*. [Verkkosivu]. Noudettu 7.8.2020 osoitteesta: https://maxima.sourceforge.io/

Moodle (2020). *History of Moodle* [Verkkosivu] Noudettu 20.3.2021 osoitteesta: https://docs.moodle.org/310/en/History.

Nakamura, Y., Higuchi, S., Ichikawa, Y., Miyazaki, Y., Yoshitomi, K., and Nakahara, T. (2019). *Effective Usage of Various Answer Types of Mathematics e-Learning System.* 2019 IEEE International Conference on Engineering, Technology and Education (TALE), Yogyakarta, Indonesia, 2019, pp. 1–5, doi: 10.1109/TALE48000.2019.9225907.

Nilsson, J., Riedel, S., (2015) *Electric Circuits* (10. painos). Pearson. Sivut 6–7. ISBN: 978-0-13-376003-3.

Patrikainen S. (2012). *Luokanopettajan pedagoginen ajattelu ja toiminta matematiikan opetuksessa.* Väitöskirja, University of Helsinki, Faculty of Behavioural Sciences, Department of Teacher Education 2012. ISBN: 978-952-10-7868-2.

Pirttinen, N. (2019). *Älykkäät oppimisympäristöt ja niiden sisällöntuotanto (ÄlyOppi).* eduuni. Noudettu 12.11.2019 osoitteesta https://wiki.eduuni.fi/pages/viewpage.action?pageId=75759210 .

Pradono, S., Astriani, M. S., Moniaga, J. (2013). *A Method for Interactive Learning*. International Journal of Communication & Information Technology, Vol. 7, No. 2, lokakuu 2013. DOI: 10.21512/commit.v7i2.583

Sangwin, C. (2010). *Who uses STACK? A report on the use of the STACK CAA system*. Maths Stats and OR Network, School of Mathematics, University of Birmingham, B15 2TT (2010). [Verkkodokumentti] Noudettu 25.7.2020 osoitteesta: https://www.researchgate.net/publication/228948202_Who_uses_STACK_A_report_on_the_use_of_the_STACK_CAA_system.

Sangwin, C. (2013). *Computer Aided Assessment of Mathematics* (1. painos). New York: Oxford University Press, sivut: 102–104 ISBN: 978-0-19-966035-3

Şirin T., Şimşek M. (2015) *Teaching physics with a computer algebra system*. 2015 Twelve International Conference on Electronics Computer and Computation (ICECCO), Almaty, Kazakhstan, 2015, pp. 1–5, doi: 10.1109/ICECCO.2015.7416903.

Soman, D., Huang, W. (2013). *A Practitioner's Guide To Gamification Of Education*. Rotman School of Management, University of Toronto, 2013. Noudettu 23.3.2021 osoitteesta https://www.academia.edu/33219783/A_Practitioners_Guide_To_Gamification_Of_Education

STACK (2021). *STACK - dokumentaatio versiolle 3.* [Verkkosivu]. Noudetteu 20.8.2020 osoitteesta https://stack.uwasa.fi/question/type/stack/doc/doc.php/

Tanskanen, H. (2017). *Dynaamista geometriaa Moodle-ympäristöön STACK- ja JSXGraph-järjestelmien testaamista monimuotoisten kysymysten laatimiseksi*. Pro Gradu -tutkielma, Itä-Suomen yliopisto 2017. Noudettu 20.3.2021 osoitteesta: https://erepo.uef.fi/handle/123456789/18620

Vaasan yliopisto (2019). *Opintojaksokuvaus 2019–2020 - sähkötekniikka*. [Verkkodokumentti] Noudettu 25.3.2021 osoitteesta: https://www.univaasa.fi/fi/opiskelijat/opinto-oppaat/tuotantotalouden-ja-tietojarjestelmatieteen-seka-tekniikan-opinto-oppaat

Valtonen M., Lehtovuori A. (2011). *Piirianalyysi, osa 1 - Tasa- ja vaihtovirtapiirien analyysi.* Unigrafia Oy, Helsinki 2011. ISBN: 978-952-92-8720-8.

Valtonen M., Lehtovuori A. (2017). *Piirianalyysi, osa 2 - Muutosilmiöt, systeemifunktiot ja siirtojohdot.* Unigrafia Oy, Helsinki 2017. ISBN: 978-952-93-8240-8.

Valtonen, T., Leppänen, U., Hyypiä, M., Kokko, A., Manninen, J., Vartiainen, H., Sointu, E., Hirsto, L. (2020) *Learning environments preferred by university students: a shift toward informal and flexible learning environments*. Learning Environments Research (2020) DOI: https://doi.org/10.1007/s10984-020-09339-6.

Vesapuisto, M. (2004). *Virtapiirien johdonmukaisuus opetuksen kannalta.* Diplomityö, Vaasan yliopisto 2004. Noudettu 25.3.2021 osoitteesta: http://lipas.uwasa.fi/~mave/DIPLOMITYO.pdf

Vesapuisto, M., Vekara, T., Korpinen, L. (2013). *The Use of Animation and Simulation to Aid Learning of Electromagnetics: Electrical Engineering at the University of Vaasa*. 2013 24th EAEEIE Annual Conference (EAEEIE 2013), Chania, Greece 2013. DOI: 10.1109/EAEEIE.2013.6576503.

YLE (2017). *Opettaja luopui luennoista – yhtäkkiä lähes kaikki opiskelijat läpäisivät vaikean yliopistokurssin*. [Verkkojulkaisu]. Noudettu 22.3.2021 osoitteesta: https://yle.fi/uutiset/3-9529446

Ylioppilaslehti (2020). *Moni opiskelija kokee jääneensä yksin korona-aikana, ja osa toivoo yliopistolta ja YTHS:ltä enemmän tukea tilanteeseen – YTHS: ”On tärkeää ottaa aina yhteyttä, kun tarvitsee apua”* [Verkkojulkaisu] Noudettu 25.3.2021 osoitteesta: https://ylioppilaslehti.fi/2020/12/moni-opiskelija-kokee-jaaneensa-yksin-korona-aikana-ja-osa-toivoo-yliopistolta-ja-ythslta-enemman-tukea-tilanteeseen-yths-on-tarkeaa-ottaa-aina-yhteytta-kun-tarvitsee-apua/

Zeynivandnezhad, F., Ismail, Z., Yusof, Y. M. (2015) *Teaching mathematical structures in differential equations using a computer algebra system to engineering students*. 2015 IEEE 7th International Conference on Engineering Education (ICEED), Kanazawa, Japan, 2015, pp. 10-15, doi: 10.1109/ICEED.2015.7451483.

Liitteet

Liite 1. Palautelomake STACK-tehtävistä.

Piirianalyysi A, STACK - Palaute, Kevät 2020

Kurssilla esiintyvien STACK-tehtävien tarkoituksena on pyrkiä antamaan opiskelijoille "rajattomasti" tehtäviä/testausmahdollisuuksia eri aihealueista, joita voi suorittaa etänä itsenäisesti. Tehtävien ideana oli myös auttaa oppilaita pääsemään sille tasolle, että laskuharjoitusten tekeminen onnistuisi.

Kysymyksissä käytetään asteikkoa 1-6, jossa 1 on "täysin eri

mieltä" ja 6 "täysin samaa mieltä" Kysymykset keskittyvät vain

STACK-tehtäviin!

Tämän kyselyn tuloksia tullaan analysoimaan anonyymisti tutkimuksessa, diplomityössä sekä STACK-tehtävien kehittämisessä.

*Pakollinen

Täysin eri mieltä ( ) ( ) ( ) ( ) ( ) ( ) Täysin samaa mieltä

Palautetta saa käyttää tutkimuksessa: *

Valitse kaikki sopivat vaihtoehdot.

- [ ] Kyllä
- [ ] Ei

1. Tehtäviä oli mielestäni liikaa *

Merkitse vain yksi soikio.

1 2 3 4 5 6

	2						
Täysin eri mieltä	◯	◯	◯	◯	◯	◯	Täysin samaa mieltä

2. Tehtävien tekemiseen jäi riittävästi aikaa * *Merkitse vain yksi soikio.*

	3	2	3	4	5	6	
Täysin eri mieltä	◯	◯	◯	◯	◯	◯	Täysin samaa mieltä

3. Törmäsitkö teknisiin ongelmiin? Jos kyllä niin minkälaisiin? * *Valitse kaikki sopivat vaihtoehdot.*

☐ En

Muu: ☐ _____

4. Tehtävänannot ja ohjeet olivat selkeitä * *Merkitse vain yksi soikio.*

	4	2	3	4	5	6	
Täysin eri mieltä	◯	◯	◯	◯	◯	◯	Täysin samaa mieltä

5. Tehtävien graafinen toteutus oli ymmärrettävää (piirikaaviot, komponentit jne.) *

Merkitse vain yksi soikio.

	5	2	3	4	5	6	
Täysin eri mieltä	◯	◯	◯	◯	◯	◯	Täysin samaa mieltä

6. Tehtävät auttoivat asioiden oppimisessa *

Merkitse vain yksi soikio.

	6	2	3	4	5	6	
Täysin eri mieltä	◯	◯	◯	◯	◯	◯	Täysin samaa mieltä

7. Tehtävät olivat liian vaikeita *

Merkitse vain yksi soikio.

	7	2	3	4	5	6	
Täysin eri mieltä	◯	◯	◯	◯	◯	◯	Täysin samaa mieltä

8. Olisi parempi, jos kaikki tehtävät olisivat julkaistu yhdellä kerralla ja olisin voinut edetä omaan tahtiin *

Merkitse vain yksi soikio.

	8	2	3	4	5	6	
Täysin eri mieltä	◯	◯	◯	◯	◯	◯	Täysin samaa mieltä

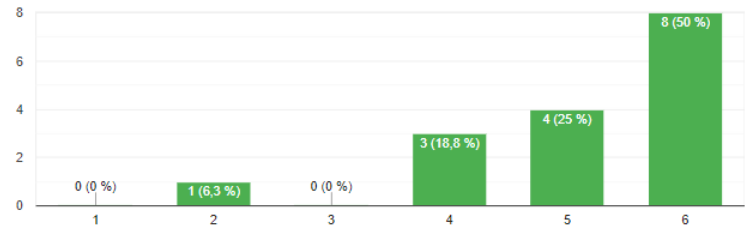
9. Mitä hyvää STACK tehtävistä havaitsit? *

10. Miten STACK tehtäviä voisi kehittää? *

Liite 2. Palautekyselyn vastaukset (2020).

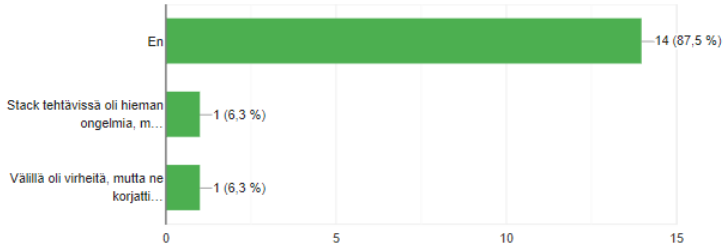
2. Tehtävien tekemiseen jäi riittävästi aikaa

16 vastausta



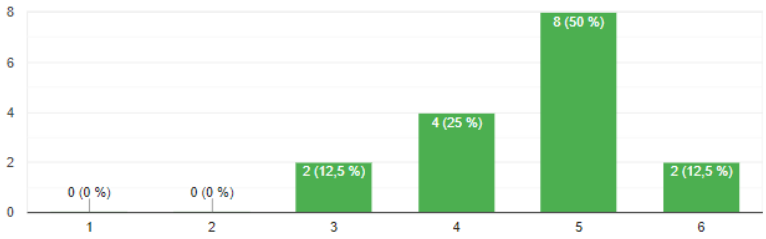
3. Törmäsitkö teknisiin ongelmiin? Jos kyllä niin minkälaisiin?

16 vastausta



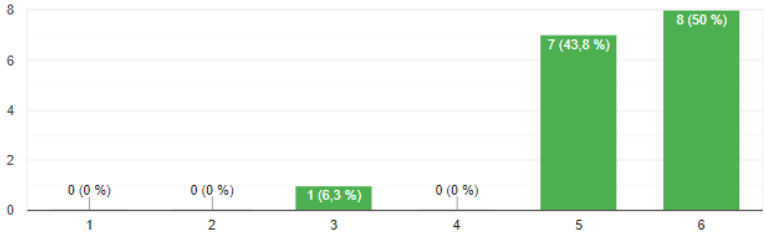
4. Tehtävänannot ja ohjeet olivat selkeitä

16 vastausta



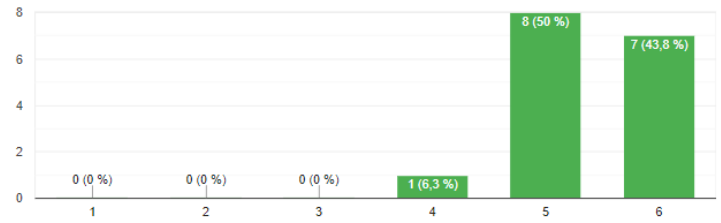
5. Tehtävien graafinen toteutus oli ymmärrettävää (piirikaaviot, komponentit jne.)

16 vastausta



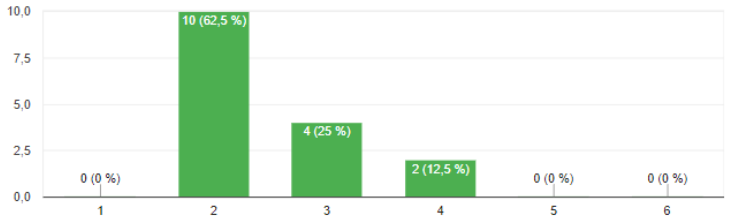
6. Tehtävät auttoivat asioiden oppimisessa

16 vastausta



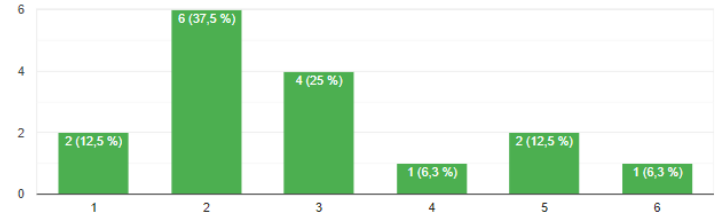
7. Tehtävät olivat liian vaikeita

16 vastausta



8. Olisi parempi, jos kaikki tehtävät olisivat julkaistu yhdellä kerralla ja olisin voinut edetä omaan tahtiin

16 vastausta



9. Mitä hyvää STACK tehtävistä havaitsit?

16 vastausta

- Näki suoraan, että onko ymmärtänyt asian oikein. Hyviä oli.
- Hyvää oheismateriaalia laskarien tueksi, väännettiin pitkään nuolia, jotta tämä jäisi kaikille talteen.
- pakollisuus
- Niiden avulla oppi tekemään harkkojen tehtäviä.
- Tehtävät havainnollistivat hyvin tiettyjä käsitteitä. Esim. Solmupistemenetelmää ja jännitteitä eri komponenttien yli.
- STACK-tehtävät todella auttoivat sisäistämään opetettuja asioita paremmin, ja laskurutiinini kehittyi.
- Niiden avulla oppi todella hyvin ja ne havainnollistivat hyvin tehtäviä ja teoriaa.
- Riittävän yksinkertaisia, niistä oppi uudet asiat.
- Sai opetella asioita itse kotona, kun oli aikaa ja tarvittaessa moneen kertaan.

9. Mitä hyvää STACK tehtävistä havaitsit?

16 vastausta

Riittävän yksinkertaisia, niistä oppi uudet asiat.

Sai opetella asioita itse kotona, kun oli aikaa ja tarvittaessa moneen kertaan.

Tehtävät auttoivat ymmärtämään joitakin vieraita asioita.

ei sekoitettu montaa eri teoriaa sekaisin, vaan keskitytettiin siihen mikä oli olennaista.

Oppi hyvin keskeiset pienet tärkeät asiat

Auttaa havainnollistamaan joitain asioita.

Tehtävät olivat mielekkäitä tehdä ja ne auttoivat todella hyvin asioiden havainnollistamisessa.

Laskurutiini kehittyi huomattavasti

Rajattomat yrityskerrat, tehtävät auttoivat perusasioiden oppimisessa ja ymmärtämisessä.

10. Miten STACK tehtäviä voisi kehittää?

16 vastausta

-

Koin hyödylliseksi. Palaute tehtävistä voisi olla jossain kohdissa selkeämpi.

Tehtävänannot olivat välillä epäselvät. Kompleksilukulaskennassa piti esimerkiksi käyttää kirjainta i kirjaimen j sijaan, mitä käytetään muuten kurssilla. Jos käytti j:tä kaikki laskut menivät nollille. Myöskin pii piti kirjoittaa tekstinä. Tästä ei mainittu, ja pii:n merkki ja likiarvo eivät kumpikaan toimineet.

lisää erilaisia tehtäviä, etenkin niin että olisi enemmän kotitehtävien kaltaisia

Kun niissä selvinneet ongelmat korjattiin, ne olivat hyviä!

Vaihtaa vaatimus 100% - - > 98% tms. Vituttaa tehdä tehtäviä 45min, kun sen jälkeen joudut tekemään pitkän tehtävän uudestaan jonkun missclickin takia.

En oikeastaan löydä mitään akuuttia kehityskohdetta.

Tehtävä osoita voisi olla enemmän, mutta joissain osioissa itsessään vähemmän tehtäviä. Esim joissain

10. Miten STACK tehtäviä voisi kehittää?

16 vastausta

Joissain kohdissa aavistuksen tarkemmat ohjeet.

Voisi olla vielä enemmän tehtäviä. Deadlinet olisi voitu antaa heti, nyt ne tulivat välillä yllättäen turhan nopeasti

Joissain aiheissa myös monivalinta tms. tehtävät voisivat olla toimivia.

Kuuden tehtävän sarjat tuntuivat vähän turhan pitkiltä. Yhden pienen merkintävirheen (ei välttämättä edes laskuvirhe) takia joutuu tekemään kaiken kokonaan uusiksi. Viimeisillä kerroilla ei enää itsellä ainakaan oppimista tapahtunut vaan lähinnä turhautui ja huolimattomuusvirheitä tuli vain lisää.

Välillä on turhauttavaa tehdä koko tehtäväsarja uudelleen jos esim matriisia täyttäessä puuttuu yksi miinus yms.

Tehtäviä olisi voinut tehdä enemmänkin.

Vastus ja kela, vastus ja kondensaattori -tehtävissä pyöristyksen kanssa ongelmia, vaikka laskin moneen otteeseen oikein. Vastauksien näkeminen jälkeenpäin helpottaisi onglemakohdan löytämistä.