LAPPEENRANNAN TEKNILLINEN YLIOPISTO
LUT School of Energy Systems
LUT Kone

*Karli Kund*

**VIRTUAALIPROTOTYYPIN HYÖDYNTÄMINEN VIILUSORVIN TUOTEKEHITYKSESSÄ**

Työn tarkastaja(t): Professori Aki Mikkola.
TkT Kimmo Kerkkänen.

**TIIVISTELMÄ**

Lappeenrannan teknillinen yliopisto
LUT Energiajärjestelmät
LUT Kone

Karli Kund

**Virtuaaliprototyypin hyödyntäminen viilusorvin tuotekehityksessä**

Diplomityö

2017

76 sivua, 40 kuvaa, 2 taulukkoa ja 2 liitettä

Tarkastajat: Professori Aki Mikkola
TkT Kimmo Kerkkänen

Hakusanat: Viilusorvi, virtuaaliprototyyppi, monikappaledynamiikka, simulointi, tuotekehitysprosessi

Nykyään yhä useammassa tuotekehitysprosessissa on otettu dynaaminen virtuaaliprototyyppi käyttöön. Dynaaminen virtuaaliprototyyppi mahdollistaa eri osastojen suunnittelijoiden nähdä ja tutkia oman työn vaikutusta koko järjestelmään.

Tämän työn tavoite oli mallintaa Rauten viilusorvista virtuaaliprototyyppi käyttäen monikappaledynamiikka (Multi-body system, MBS) menetelmää. Malli luodaan Mevean Modeller -ohjelman avulla ja mallia voidaan ajaa Mevean Solver -ohjelmassa reaaliaikaisena simulaationa. Mallin luomiseen tarvittavat tiedot, Ohjelmisto mahdollistaa myös anturi-, voimansiirto-, hydrauliikka- ja sähkökomponenttien lisäämisen MBS-malliin.

Vaneri tuotantolinjaston tärkein vaihe on vaneriviilun sorvaaminen. Tulevassa uuden sukupolven viilusorvissa suurin osa hydrauliikkasylintereistä on korvattu lineaarisilla karamoottoreilla. Karamoottoreita ohjataan moottorin sulautetulla ohjausjärjestelmällä, jonka avulla moottoria voidaan ohjata aseman, nopeuden ja väännön mukaan. Tämän takia, työn toinen tavoite oli muodostaa moottoriohjausjärjestelmä, joka jäljittelee moottoreiden sulautetun järjestelmän toimintaa.

Jotta saataisiin virtuaalisesta prototyypistä mahdollisimman realistinen, luotiin viilusorvauksesta erillinen prosessimalli. Prosessimallin tehtävä on simuloida viilun sorvaamista puusta ja kuinka sorvausprosessi vaikuttaa sorvausmekanismiin. Sorvausprosessimalli luodaan toisena projektina tässä hankkeessa, jolle luodaan rajapinta malliin.

Lopputuloksena valmistui viilusorvista MBS-malli virtuaaliprototyyppiä varten. Malli sisältää suuren määrän kappaleita (34), mikä vaati paljon laskentatehoa, jotta malli pysyisi reaaliajassa. Jotta mallia pystyi ohjamaan Rauten ohjausjärjestelmällä, luotiin ohjausjärjestelmän ja mallin välille moottoriohjausjärjestelmä. Järjestelmä mahdollistaa mallin moottoreiden ohjaamisen samalla tavalla kuin fyysisessä koneessa. Virtuaaliprototyypin avulla tuotekehityksessä pystytään vertailemaan erilaisia moottorivaihtoehtoja ja testaamaan uusia ohjausjärjestelmiä.

**ABSTRACT**

Lappeenranta University of Technology
LUT School of Energy Systems
LUT Mechanical Engineering

Karli Kund

**The utilization of virtual prototype in the product development of peeling lathe**

Master's thesis

2017

76 pages, 40 figures, 2 tables ja 2 appendices

Examiners: Professor Aki Mikkola
D.Sc. (Tech.) Kimmo Kerkkänen

Keywords: Multi-Body-System, MBS , veneer peeling lathe, dynamic virtual prototype, product development process

These days dynamic virtual prototypes becoming a useful tool in the product development process. When a dynamic virtual prototype is put in practice, it enables the possibility for designers from different development sections to test and monitor the result of their own work as well as track the impact on the entire system.

The main aim of this thesis is to build a dynamic virtual prototype based on a Raute peeling machine using a multibody system (MBS) approach. To build the MBS model Mevea Modeller software is utilized and it can be simulated in real-time in Mevea Solver software. This software already contains all the components required to build the model, but it also allows to add sensors, power transmission, hydraulics systems and electric engines as a part of the MBS model.

The core part of the veneer production line is a veneer peeling lathe. It is the new generation veneer peeling lathe, where most of the hydraulic cylinders are replaced with electrical linear actuators. Actuators have embedded control system, which is used to regulate their position, speed or torque. Therefore, another goal of this work is to simulate the actuators embedded control system to interact with the Raute control system.

Additionally, to create an elaborate dynamic virtual prototype an MBS model was complimented with an interface for the wood peeling process model. The purpose of it is to simulate the actual process of the veneer peeling and its impact on the machine mechanism. The peeling process model is developed as another part of this project.

As a result, an MBS model was built for a dynamic virtual prototype of a peeling machine. This model includes a large amount of bodies (34) and that results in a higher demand of the processing power to perform a real-time simulation. To connect Raute control system with the MBS model, a motor control system was developed to replicate the operation of the actuators embedded control system. By utilizing this dynamic virtual prototype, it was possible to test compatibility of different electric linear actuators as well as the test new control system.

**ALKUSANAT**

Haluan kiittää Meveaa ja Rautea antamasta mielenkiintoisesta diplomityö aiheesta. Kiitän Mevean työväkeä, kaikesta avusta jonka annoitte, kun muodostin ensimmäisen tämän kokoluokan monikappaledynamiikka mallin. Ilman teitä, olisin vieläkin rakentamassa mallia. Kiitän myös Rautea työväkeä siitä, että sain tutusta puunsorvaamisen maailmaan ja kuinka sorvaus koneita suunnitellaan. Kiitän myös työn tarkastajia Kimmo Kerkkästä ja Aki Mikkolaa ohjaavasta palautteesta, jonka avulla tämäkin työ saatiin loppuun.

Ennen kaikkea haluan kiittää opiskelija tovereita, joita ilman en olisi tässä pisteessä. Ilman heitä, ajatuksen juoksu olisi aika yksipuolista ja tämäkin diplomityö sisältäisi enemmän kirjoitusvirheitä. Kiitos myös tyttöystävälle kärsivällisyydestä ja kannustuksesta. Kiitos myös ihmisille jotka jaksoivat kysyä ”milloin meinasit valmistua?”, joka piti työn loppuun saattamisen aina mielessä. Kiitos myös lukijalle.

Karli Kund

Lappeenrannassa 21.05.2017

# SISÄLLYSLUETTELO

**TIIVISTELMÄ**
**ABSTRACT**
**ALKUSANAT**
**SISÄLLYSLUETTELO**
**SYMBOLILUETTELO**
**1 JOHDANTO . . . . . 9**
1.1 Työn tavoitteet . . . . . 10
1.2 Työn rajaus . . . . . 11
**2 ESITIEDOT . . . . . 12**
2.1 Virtuaalimallin hyödyntäminen tuotesuunnitteluprosessissa . . . . . 12
2.2 Monikappaledynamiikka . . . . . 16
2.2.1 Kinematiikka . . . . . 17
2.2.2 Kinetiikka . . . . . 20
2.3 Virtuaalisen mallin luominen . . . . . 24
2.3.1 Mevea ohjelmisto . . . . . 24
2.3.2 Ratkaisu menetelmä . . . . . 25
2.3.3 Mevea ohjelmiston I/O -rajapinnan käyttö . . . . . 25
2.3.4 Voimanlähteet . . . . . 26
2.3.5 Sähkömoottorit ja kuularuuvit . . . . . 28
2.3.6 Hydrauliikka ja pneumatiikka . . . . . 28
2.4 Reaaliaikasimulointi . . . . . 29
2.5 Julia-ohjelmointikieli . . . . . 31
**3 VIRTUAALIMALLIN RAKENTAMINEN . . . . . 33**
3.1 Vaneriviilun sorvaus . . . . . 33
3.1.1 Vanerin koostumus ja vaatimukset . . . . . 35
3.1.2 Pöllin keskittäminen . . . . . 35
3.1.3 Pöllin sorvaus . . . . . 36
3.2 Mallin muodostaminen . . . . . 38
3.3 Kappaleet . . . . . 40
3.4 Topologia ja liitokset . . . . . 41

3.5 Voimat . . . . . 44
3.6 Rajapintojen luominen malliin . . . . . 47
3.6.1 Sorvausprosessimallin liittäminen simulaattoriin . . . . . 48
3.6.2 Ohjausyksikön liittäminen simulaattoriin . . . . . 50
3.7 Kontaktivoimat . . . . . 54
3.8 Puupartikkeli . . . . . 56
**4 TULOKSET . . . . . 58**
4.1 Virtuaaliporotyypin suorituskyky . . . . . 58
4.2 Rajapintatoiminta . . . . . 60
4.2.1 Moottoriohjausjärjestelmän toiminta . . . . . 61
4.2.2 Moottoriohjausjärjestelmän testaaminen . . . . . 62
4.3 Puupartikkelit . . . . . 64
4.4 Virtuaaliprototyyppi osana tuotekehitystä . . . . . 66
4.4.1 Massan muutosesimerkki . . . . . 68
4.4.2 Moottorin asemamuutosesimerkki . . . . . 69
**5 JOHTOPÄÄTÖKSET . . . . . 72**
5.1 Kehityskohteet . . . . . 73
**LÄHTEET . . . . . 75**
**LIITTEET**
LIITE I Julia rajapinta Mevea ohjelmistoon
LIITE II Julia rajapinnan määrittely

## SYMBOLI- JA LYHENNELUETTELO

| | |
|---|---|
| $\boldsymbol{\theta}$ | Kiertymäkoordinaatiston vektori. |
| $\rho$ | Kappaleen tiheys. |
| $\mathbf{A}$ | Kiertomatriisi. |
| $\mathbf{C}$ | Rajoiteyhtälöiden vektori. |
| $\mathbf{C}_q$ | Rajoiteyhtälöiden Jacobin matriisi. |
| $\boldsymbol{F}_a$ | Ulkoisten voimien vektori. |
| $\boldsymbol{F}_i$ | Inertiavoimien vektori karteesisessa koordinaatistossa. |
| $h$ | Kappaleen korkeus. |
| $\boldsymbol{I}$ | Kappaleen inertiamatriisi. |
| $\mathbf{M}$ | Massamatriisi. |
| $M$ | Kappaleen massa. |
| $n_b$ | Kappaleiden vapausasteiden määrä. |
| $n_c$ | Kappaleiden välisten rajoitteiden määrä. |
| $P_{screw}$ | Kularuuvin nousu. |
| $\boldsymbol{q}$ | Yleistettyjen koordinaattien vektori. |
| $\ddot{\boldsymbol{q}}$ | Yleistettyjen koordinaattien kiihtyvyys vektori. |
| $\boldsymbol{R}$ | Lokaalin koordinaatiston asemavektori globaalissa koordinaatistossa. |
| $R_x$, $R_Y$ & $R_Z$ | Asemavektorin $\boldsymbol{R}$ komponentit. |
| $\mathbf{r}$ | Pisteen asemavektori kiinteässä koordinaatistossa. |
| $\delta\boldsymbol{r}$ | Virtuaalinen siirtymä. |
| $\ddot{\mathbf{r}}$ | Pisteen kiihtyvyysvektori kiinteässä koordinaatistossa |
| $r$ | Kappaleen säde. |
| $r_{screw}$ | Hammaspyörän säde. |
| $T$ | Matriisin transformaatio. |
| $t$ | Aika. |
| $\bar{\mathbf{u}}$ | Pisteen paikka vektori paikallisessa koordinaatistossa. |
| $v_n^{real}$ | Ohjausruuvin nopeus. |
| $v_{n-1}^{real}$ | Edellisen aika-askeleen ohjausruuvin nopeus. |
| $v_n^{ref}$ | Ohjausruuvin ohjausnopeus. |

| | |
|---|---|
| $\delta W$ | Virtuaalityö. |
| $\delta W_c$ | Rajoitevoimien tekemä virtuaalinen työ. |
| $\delta W_i$ | Inertiavoimien tekemä virtuaalinen työ. |
| $\delta W_e$ | Ulkoisten voimien tekemä virtuaalinen työ. |
| $x_n^{real}$ | Kuuljetusruuvin asema. |
| $xd_n^{real}$ | Kuuljetusruuvin todellinen nopeus. |

| | |
|---|---|
| CAD | Computer Aided Design, tietokoneavusteinen suunnittelu. |
| CAN | Controller Area Network. |
| DOF | Degrees of freedom, Vapausasteiden määrä. |
| FEM | Finite Element Method, elementtimenetelmä. |
| IP | Internetin protokolla osoite. |
| MBS | Multi-Body System, monikappalesysteemi. |
| PID | Proportional-integral-derivative-säädin. |
| Purilas | Pöllin sorvauksessa jäljelle jäänyt poikkileikkaukseltaan pyöreä kappale. |
| Pölli | Määrättyyn mittaan tukista katkaistu kappale. |
| Soket | Kommunikointi tapa, joka käyttää tietokoneen IP-osoitetta ja porttia. |
| TCP | Transmission Control Protocol. |
| Viilu | Pöllistä sorvaamalla tai leikkaamalla erottu ohut levy. |

# 1 JOHDANTO

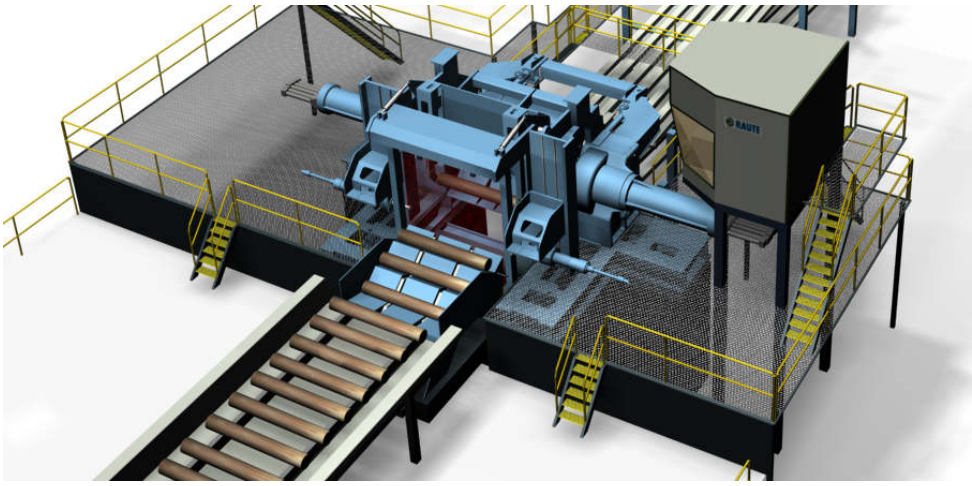
Tässä diplomityössä luodaan Raute Oyj:n viilusorvista virtuaaliprototyyppi käyttäen matemaattista monikappaledynamiikka (multi-body system, MBS) menetelmää. Virtuaaliprototyyppi tullaan luomaan Mevea-ohjelmiston avulla, josta löytyy valmiit komponentit MBS-mallin luomista varten. Ohjelmiston avulla voidaan malliin lisätä virtuaaliantureita, voimansiirtokomponentteja, hydrauliikkajärjestelmä ja sähkömoottoreita. Ohjelmisto on suunniteltu reaaliaikaiseen simulointiin, mikä mahdollistaa mallin ajon ja ohjaamisen reaaliajassa.

Monikappalemalli tuo merkittävän parannuksen koneiden ja mekaanisten järjestelmien suorituskyvyn arviointiin, kun verrataan nykyisiin menetelmiin, jotka perustuvat joko analyyttisiin laskelmiin tai kokeellisiin menetelmiin. Nykyisiin menetelmiin verrattuna, monikappalemallin avulla saadaan tietää nopeammin ja tehokkaammin eri tekijöiden vaikutus mekaniikan dynamiikkaan. (Kortelainen 2011, s. 5.)

Työssä käydään läpi mallin muodostamien ja kuinka mallia voidaan hyödyntää tuotekehityksessä. MBS-mallin avulla voidaan tarkastella, miten eri mekaniikkaratkaisut toimivat kokoonpanossa ja kuinka eri kokoonpanot sopivat yhteen. Se tuo eri osa-alueiden kehittäjiä yhteen tuotekehitysprosessin aikaisemmassa vaiheessa, mikä tehostaa tuotekehityksen nopeutta ja tuottavuutta. Samalla suunnitteluvirheet ilmenevät aikaisemmassa vaiheessa, jotka muuten ilmentyisivät vasta tuotekehityksen lopussa. Mevea-ohjelmisto mahdollistaa myös mekanismimalliin liitettyjen voimalähteiden, kuten sähkömoottoreiden ja hydrauliikkajärjestelmien, toiminnan tutkimisen. Voimalähteitä voidaan ohjata yrityksen omalla ohjausjärjestelmällä, jolloin ohjausjärjestelmän kehittäjät pystyvät myös käyttämään mallia tuotekehityksessä.

Mevea Oy on Lappeenrannan teknillisestä yliopistosta lähtöisin oleva reaaliaikaiseen simulointiin erikoistunut yritys, jonka ohjelmistoa käytetään simulointimallin luomiseen ja sen testaukseen. Ohjelmiston avulla pystytään helposti luomaan monikappalejärjestelmiä ja siihen pystytään liittämään eri ohjaustoimintoja. Ohjelmistolla luotua mallia voidaan käyttää tuotekehityksessä, henkilöstökoulutuksessa ja markkinoinnissa.

Raute Oyj on Nastolassa sijaitseva puunjalostuskoneita valmistava yritys. Rauten kehittämä ja tuottama viilusorvi sopii niin kovalle kuin pehmeälle puulle, jolloin heidän koneita voidaan käyttää viilun valmistukseen ympäri maailmaa. Viilusorvi pystyy käsittelemään minuutissa 15 tukkia, joiden halkaisija on 250 mm. Sorvi pystyy työstämään jopa halkaisijaltaan 1000 mm:ä olevia puita, ja tuottamaan alle 0,5 mm paksuista viilua. Kuvassa 1 on esitetty Rauten viilusorvimallin markkinointikuva.



**Kuva 1.** Rauten viilunsorvaus linjasto.

Mevea Oy ja Raute Oyj käynnistivät yhteisen projektin nimeltä Romulus, jonka tavoitteena on luoda Rautelle virtuaaliprototyyppi viilusorvista. Tässä diplomityössä luodaan viilusorvin mekaniikasta MBS-malli, johon liitetään lineaariset karamoottorit ohjamaan mekaniikkaa. Virtuaaliprototyypin ja Rauten ohjausjärjestelmän välille luodaan moottoriohjausjärjestelmä, jonka simuloi moottoreiden sulautettua ohjausjärjestelmää. Tämä mahdollistaa mallin ohjaamisen samalla tavalla kuin oikean viilusorvin. Rajapintaa voidaan myös hyödyntää ohjausjärjestelmän kehitystyössä. Malliin luodaan toinen rajapinta sorvausprosessimallille, joka simuloi sorvausprosessi ja siinä syntyviä voimia, jotka lähetetään rajapinnan kautta MBS-mallille.

## 1.1 Työn tavoitteet

Diplomityön päätavoitteena on luoda MBS-malli Rauten viilusorvista. Mallin voimalähteinä toimivat lineaariset sähkömoottorit ja hydrauliikkasylinterit, joita tullaan ohjaamaan Rauten ohjausjärjestelmällä. Ohjausjärjestelmän ja MBS-mallin väliin tullaan luomaan moottoriohjausjärjestelmä, mikä tulkitsee ohjausjärjestelmän lähettämät käskyt ja ohjaa näiden perusteella sähkömoottoreiden ja hydraulisylintereiden liikkeitä.

Mevea-ohjelmisto mahdollistaa mallin reaaliaikaisen ajamisen, jolloin voidaan testata toimintoja reaaliaikaisella ohjauksella. Viilusorvissa on paljon liikkuvia kappaleita, joten mallin suunnitteluun tulee kiinnittää huomiota. Mallin rakenne ei saa ylikuormittaa laskentaa, jotta malli pysyisi reaaliajassa. MBS-mallia tullaan käyttämään virtuaaliprototyyppinä Rauten tuotesuunnitteluprosessissa. Virtuaaliprototyypin käytöllä voidaan vähentää tarvittavien fyysisten prototyyppien määrää tuotesuunnitteluprosessissa. Kun virtuaaliprototyyppi on valmis, voidaan mallia hyödyntää viilusorvin oikeiden ajoasetusten määrittämiseen ja asiakkaan henkilöstön kouluttamiseen. Sorvausprosessista on luotu oma prosessimalli, joka simuloi puun vuolentaa. MBS-malliin luodaan rajapinta sorvausprosessimallille, jonka kautta MBS-malli lähettää kappaleiden asematietoja, ja prosessimalli lähettää leikkauksesta syntyneet voimat MBS-mallille. Näiden mallien yhdistämisen tavoitteena on saada mahdollisimman realistinen virtuaaliprototyyppi tulevasta viilusorvista.

## 1.2 Työn rajaus

Diplomityö käsittelee sorvisimulaattorin MBS-mallin luomisen Mevea-ohjelmistolla. Ohjelmiston komponenteilla luodaan malliin rajapinta sorvausprosessimallille ja sorvin ohjausjärjestelmälle. Lisäksi lineaarisille sähkömoottoreille ja hydrauliikkajärjestelmälle luodaan oma moottoriohjausjärjestelmä, joka simuloi lineaaristen karamoottoreiden ja hydrauliikkajärjestelmän sulautettua ohjausjärjestelmää. Moottoriohjausjärjestelmä lukee Rauten ohjausjärjestelmän lähettämät käskyt ja ohjaa niiden perusteella liikkeitä. Puun käyttäytyminen sorvausprosessissa simuloidaan erillisenä moduulina, joka tullaan liittämään määritellyn rajapinnan avulla. Rajapinnan läpi mekaniikkamalli lähettää kappaleiden asema-ja nopeustiedot, ja sorvausprosessimalli lähettää sorvauksesta syntyneet vastavoimat mekaniikalle. Työssä ei käsitellä tarkemmin viilusorvin mekaanisia ratkaisuja eikä laitteen toimintaa. Diplomityön tavoite on luoda virtuaaliprototyyppi, jota voidaan jatkojalostaa ja säätää paremmaksi Rauten tuotekehitysprosessi.

# 2 ESITIEDOT

Virtuaaliprototyyppi luodaan matemaattisella monikappaledynamiikka menetelmällä. Mallin rakentaminen vaatii tietoa MBS-menetelmästä, kuten esimerkiksi vapausasteista ja kääntömenetelmistä. Matemaattisen MBS-mallin luominen käsin saattaa tuotesuunnitteluprosessissa olla liian hidasta ja haastavaa, jonka takia Mevean tarjoamat valmiit komponentit nopeuttavat mallin muodostamista. Seuraavissa kappaleissa käydään läpi pintapuolisesti MBS-menetelmän teoriaa ja kuinka MBS-mallia voidaan hyödyntää tuotekehityksessä.

## 2.1 Virtuaalimallin hyödyntäminen tuotesuunnitteluprosessissa

Suunnittelijoilla on suurin vaikutus tuotanto- ja käyttökustannuksissa, laadussa ja tuotteen valmistusajassa. Koska suunnittelulla on suuri painoarvo kustannuksissa, tulee suunnittelijoiden jatkuvasti arvioida saavutettavia päämääriä. Kilpailun kiristyttyä vaaditaan suunnittelijoilta yhä tarkempaa tietoa ja virheetöntä suunnittelua. (Pahl et al. 2007, s. 7.)

Rauten tapauksessa tuotetta aletaan valmistaa vasta kun tilaus on tullut, ja tuotteitten pienet sarjat vaativat korkeaa onnistumisprosenttia tuotannossa. Hankintakustannukset ovat asiakkaalle suuret ja kustannuskilpailu on kovaa, kun puhutaan tuotantolinjahankinnoista. Kilpailuedun vahvistamiseksi voidaan parantaa laitteen toimintaa ja laatua, mikä yleensä tarkoittaa alijärjestelmien lisäämistä laitteeseen. Näin saadaan laitteeseen lisää ominaisuuksia, jolla voidaan erottua kilpailijoista. Laitteen myyntiä voidaan myös lisätä tekemällä laitteesta energiatehokas. Yhä tiukkenevassa energiakilpailussa on nykyään tärkeätä tietää, paljonko laitteen käyttö vaatii energiaa. Laitteen energiatehokkuudella voidaan saavuttaa merkittäviä etuja kilpailutilanteessa. Nämä kaikkien ominaisuuksien lisääminen vaatii laajaa ja hyvin toteutettua suunnitteluprosessia, jolla pystytään välttämään suunnitteluvirheitä tuotesuunnittelussa.

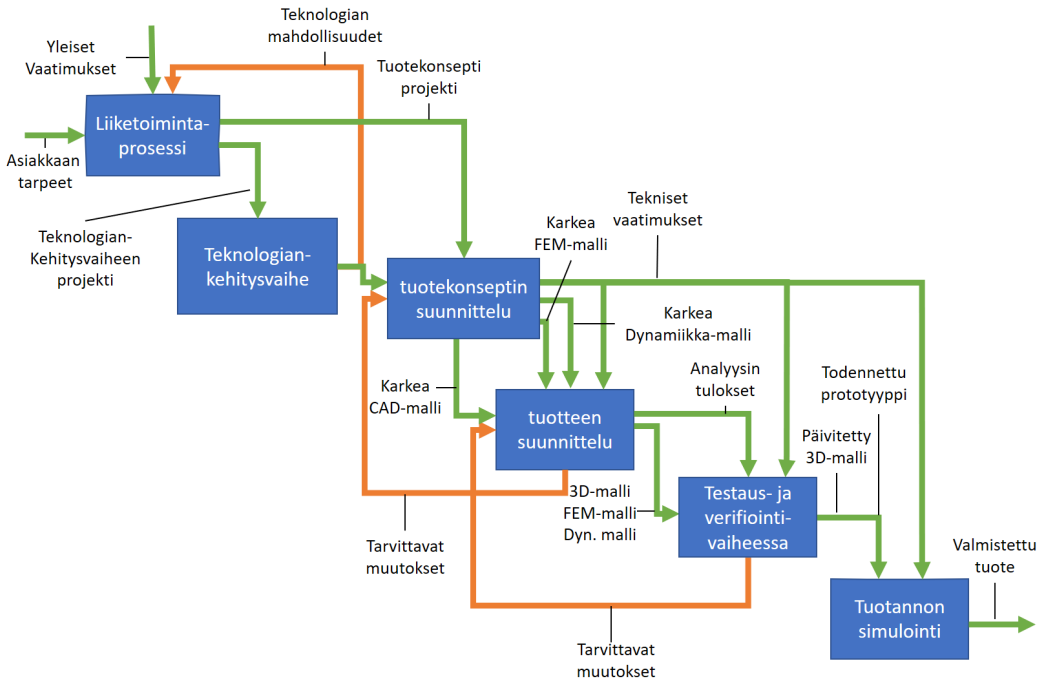
Ohjelmistosuunnittelussa on 1960-luvulta lähtien sovellettu ja työstetty erilaisia uusia suunnitteluprosessimalleja, mutta koneensuunnittelijat ovat vasta 2000-luvun alussa alkaneet ottaa uusia prosessimalleja käyttöön. Ohjelmistosuunnittelun päätavoite on ollut joustavampi ja asiakaslähtökohtaisempi suunnitteluprosessi, ja samoja tavoitteita on nykyään konesuunnittelussa. Tietokoneavusteisen suunnittelun ja simuloinnin kehitys ovat tuoneet uusia

työkaluja konesuunnitteluun ja vieneet suunnittelua kohti joustavampaa simulointipohjaista suunnitteluprosessimallia. (Ahola et al. 2011, s. 23.)

Myös samoilla markkinoilla kilpalevien yritysten aiheuttama paine vaatii yhtiöitä nopeuttamaan tuotteiden valmistusta, pitäen samalla tuotteen valmistuskustannukset alhaalla. Tuotteen kehitys saattaa muuttaa tuotteen omaa järjestelmää yhä monimutkaisemmaksi kasvattaen samalla alijärjestelmien virheiden todennäköisyyttä. Näiden virheiden sivuvaikutuksia voivat olla esim. ylimääräinen melu, värähtelyongelmat tai järjestelmän toimimattomuus. Näiden ongelmien välttämiseksi voidaan lisätä virtuaaliprototyyppien määrää tuotannossa, jolloin suunnittelijat saavat heti tuotannon alkuvaiheessa palautetta heidän työstään. Tuotannon alkuvaiheessa löydetyt virheet ovat helpompia ja halvempia korjata, kuin tuotannon lopussa löydetyt virheet. (Kortelainen 2011, s. 23.)

Tuotteen simulointi tuotesuunnittelussa helpottaa eri osastojen välistä kommunikointia ja auttaa eri osastoja kohtamaan yhteiset ongelmat heti kehityksen alkuvaiheessa.

Tekes käynnisti projektin nimeltä KONEMASINA, jossa selvitettiin yritysten nykyisiä tuotekehitysprosesseja. Näiden tietojen avulla kehitettiin kuvan 2 mukainen kuusivaiheinen tuotekehitysprosessi, jossa suunnitteluprosessissa on mukana MBS-malleja.



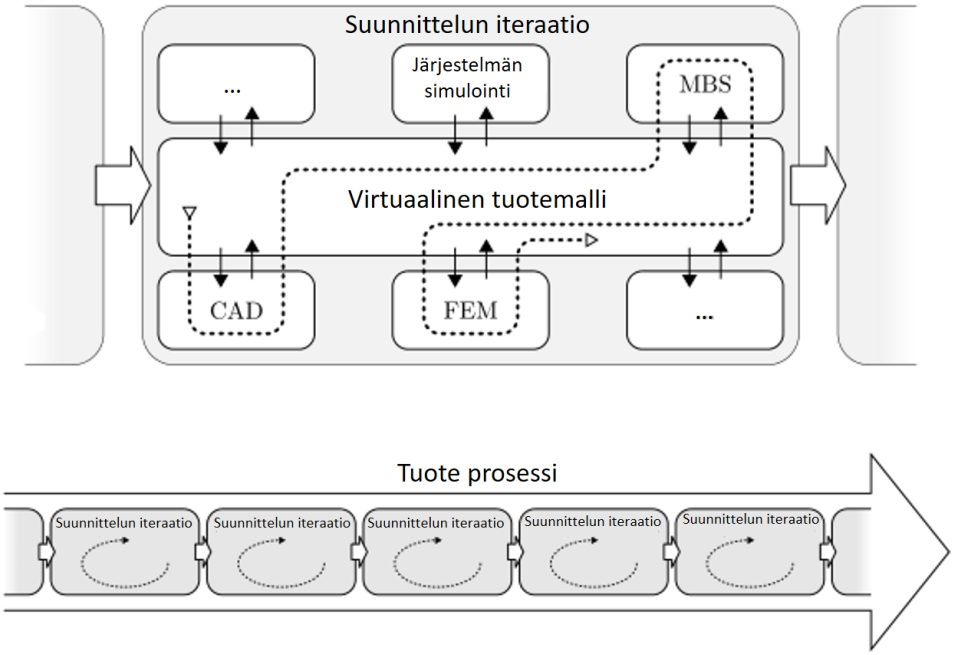

**Kuva 2.** Kuusiportainen simulointipohjainen tuotekehitysprosessi (muokattu Lehtonen 2006, s. 24).

Tuotekehitysprosessin ensimmäinen vaihe on liiketoimintaprosessi (Business process), jossa tutkitaan tulevaa tuotetta liiketoiminnan kannalta. Tuotetta tarkastellaan asiakastarpeen ja tuotteen yleisten vaatimusten perusteella. Seuraavassa vaiheessa on teknologian kehitysvaihe (Technology development process), jossa tarkastellaan mahdollisia uusia komponentteja ja uutta teknologiaa. Nämä kerätyt tiedot siirretään tuotekonseptin suunnittelu vaiheeseen (Design concept), joiden perusteella suunnitellaan ensimmäiset karkeat CAD-mallit (Computer-aided design, CAD) ja samalla tarkennetaan vaatimusmäärittelyä. Karkeiden tuotemallien avulla voidaan luoda ensimmäisen simulointimallit ja tehdä lujuustarkastelua, joiden perusteella CAD-malleja muokataan. (Lehtonen 2006, s. 24.) Karkeiden simulointimallien avulla verrataan eri suunnitteluratkaisuja, selvitetään tuntemattomien parametrien arvoja ja luodaan pohja tarkemmalle mallinnukselle. (Ahola et al. 2011, s. 29.)

Seuraava vaihe on tuotteen suunnittelu (Design product), joka muistuttaa konseptisuunnittelua, mutta on paljon yksityiskohtaisempi. Mallien massat, inertiat ja liitostiedot alkavat tässä vaiheessa olla lähempänä todellisuutta, jolloin voidaan luoda kevyt reaaliaikainen simulointimalli. Mallin avulla voidaan tutkia ihminen-kone-vuorovaikutuksia. Testaus- ja verifiointivaiheessa (Test and verify) virtuaalista prototyyppiä testataan ja todennetaan, ja samalla rakennetaan ensimmäinen fyysinen prototyyppi tuotteesta. Fyysinen prototyyppi rakennetaan virtuaalisen prototyypin geometrialla ja materiaaleilla, mutta ei välttämättä valmisteta samalla tavalla kuin tuotantolinjassa valmistettuna. Testien avulla selvitetään, onko tuote suunnitelmien ja asiakkaan tarpeiden mukainen. Tulosten perusteella virtuaalista mallia päivitetään vastamaan fyysistä prototyyppiä ja tuotteeseen tehdään vielä tarvittavat muutokset. Tuotteen valmistus tuotantokoneilla simuloidaan, tehdään tehdassuunnittelu ja suunnitellaan materiaalivirtojen hallinta. Tuotannon simuloinnin (Ramp-up production) avulla poistetaan mahdolliset tulevat ongelmat ja varmistetaan tuotannon joustava aloittaminen. (Lehtonen 2006, s. 24.)

Mallintamisen ja simuloinnin tuomat edut tulevat nopeasti esille tuotekehityksen eri vaiheissa. Järjestelmän mallintaminen helpottaa suunnittelijaa hahmottamaan järjestelmän rakennetta ja ymmärtämään kappaleiden vaikutukset toisiinsa. Lisäksi järjestelmän mallintaminen auttaa luomaan tiiviin, mutta yksityiskohtaisen dokumentoinnin järjestelmän rakenteesta, sen kappaleista ja toiminnollisuudesta. Järjestelmän mallintaminen edesauttaa muita ymmärtämään järjestelmää ja sen dynamiikkaa, ja suunnittelijan tekemiä ratkaisuja. (Kortelainen 2011, s. 23.)

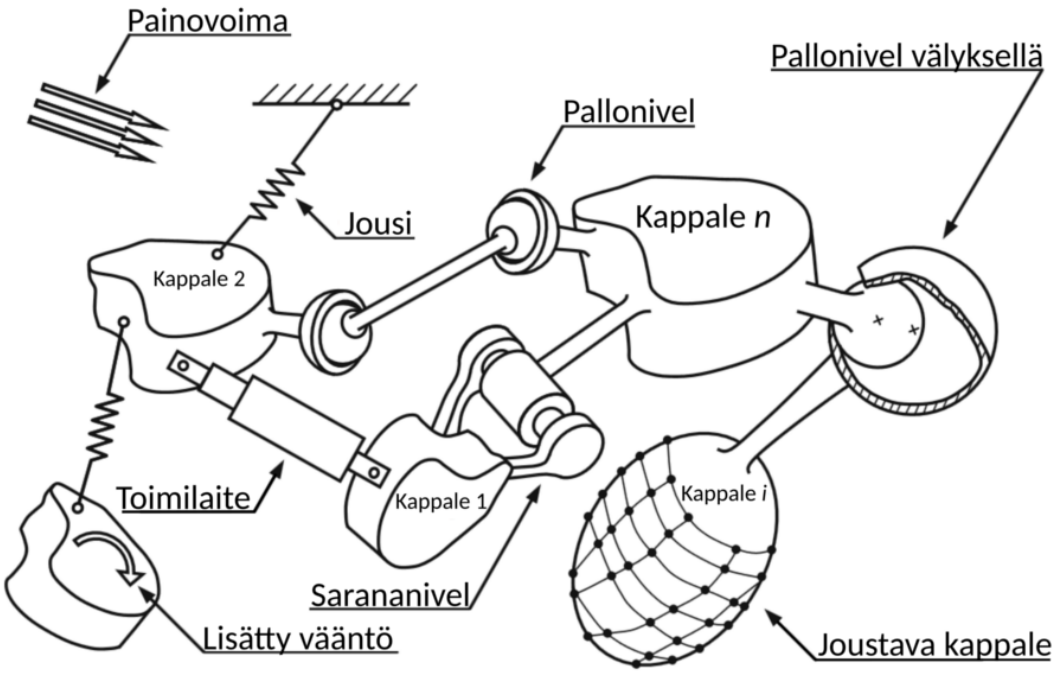
Kun lähdetään tarkastelemaan tuotesuunnittelua lähemmin, on laskentamenetelmien kehitys muokannut tuotekehityksen prosessia ja suunnitteluprosessin mallia tehokkaammaksi. Matemaattisten prototyyppien käyttö on lisännyt eri osa-alojen yhteistä tutkimusta, ja on lyhentänyt suunnittelun iteroinnin kiertonopeutta. Matemaattiset mallit ovat myös helpottaneet eri osastojen, kuten mekaniikan ja elektroniikan, välistä kommunikointia ja ymmärrystä eri toteutuksille, mikä nopeuttaa yhteistoimintaa heti tuotekehityksen alussa. Kuvassa 3 on esitetty yhden iterointikierron vaiheita, joissa havainnoidaan simuloinnin ja analysoinnin ideaalista käyttöä mekaniikan suunnittelussa. Kuvan 3 ylälaidassa kappaleen muotoja muutettu, jonka seurauksena MBS-mallia tulee päivittää. Tämän jälkeen analysoidaan uudelleen koko järjestelmän mekaniikka, josta saadaan rakenteiden kuormituksille uudet suureet. Kuvassa järjestelmän päivitysprosessi on kuvattu katkoviivanuolella. Kuvan 3 alalaidassa on demonstroitu tuotekehitysprosessia, jossa käytetään virtuaalista prototyyppiä, jonka avulla hallitaan mallinnusta ja simulointia läpi koko prosessin. Suunnittelun iteraatiossa luodaan eri simulointimalleja, joiden toiminnan perusteella kehitetään tuotetta. (Kortelainen 2011, s. 23.)




**Kuva 3.** Esimerkki tietokoneavusteisesta tuotekehitysprosessista (muokattu Kortelainen 2011, s. 24).

## 2.2 Monikappaledynamiikka

Modernit mekaaniset koneet ovat usein erittäin monimutkaisia ja ne koostuvat monista eri kappaleista, jotka on liitetty toisiinsa erilaisilla menetelmillä. Näitä koneita on voitu jakaa osajärjestelmiin, joita on voitu tutkia tietokonesimuloinnin avulla. Järjestelmien vuorovaikutuksien tutkimista toisiinsa on ollut nykyisin erittäin epätarkkaa, kun esimerkiksi voimakomponentit ovat kuvattu vakioina tai aikafunktioina. Mekanismin kappaleeseen saattaa vaikuttaa voimia, kuten nopeita kiihtyvyyksiä tai painottomuutta, joiden vaikutusta koko järjestelmään on haastava laskea perinteisillä laskentamenetelmillä. Jotta voidaan helpottaa koko mekaanisen järjestelmän dynamiikan laskentaa, on luotu numeerisia mallinnusmenetelmiä kappaleiden välisistä vuorovaikutuksista, mitä kutsutaan monikappaledynamiikaksi. Mallissa mekanismi on jaettu eri jäykkiin tai joustaviin kappaleisiin, jotka ovat vuorovaikutuksessa toisiinsa nivelillä ja erilaisilla voimakomponenteilla kuten jousilla, vaimentimilla ja toimilaitteilla. Kuvassa 4 on esitetty abstrahoitu esimerkki monikappaledynamiikka järjestelmästä. Kuvassa on karkeasti esitetty matemaattisen mallin rakennetta, jossa on esitetty erilaiset kappaleet ja niiden väliset vuorovaikutukset. Useimmissa mekaanisissa laitteissa laitteiden ei haluta kappaleiden joustavan tai kappaleet käyttäytyvät reaalimaailmassa jäykkinä, jolloin kappaleet voidaan mallintaa jäykkinä tai muuttumattomina kappaleina. (Shabana 1998, s. 3; Rouvinen 2003, s. 2.) Tässä työssä kaikki kappaleet mallinnetaan jäykkinä, koska kappaleiden joustavuus oletetaan mitättömäksi. MBS-mallin kuormitustietojen avulla voidaan kappaleen jäykkyyttä optimoida ja todentaa kappaleen joustavuutta FEM-menetelmällä (Finite Element Method, FEM), kun voimasuuret ja niiden suunnat ovat tiedossa.




**Kuva 4.** Monikappale järjestelmä (muokattu Flores 2015, s. 2).

Monikappaledynamiikka voidaan erotella kahteen eri ryhmään: kinetiikka ja kinematiikka. Kinematiikassa (liikeoppi) tutkitaan kappaleen asemaa, nopeutta ja kiihtyvyyttä ajan suhteen ilman voimia ja momentteja. Kinetiikassa tutkitaan voimien, massojen ja liikkeiden suhteita. Seuraavissa kappaleissa käydään läpi näitä kahta ryhmää. (Airila 1996, s. 6.)

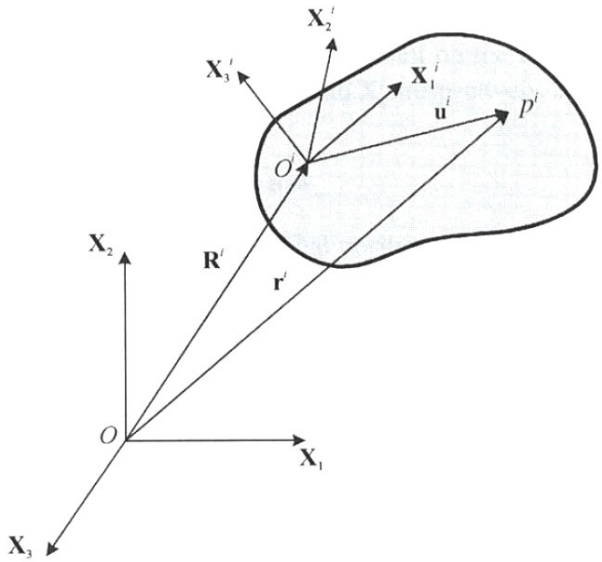
### 2.2.1 Kinematiikka

Kolmiulotteisessa avaruudessa jokaiselle kappaleella on kuusi vapausastetta: syvyys, korkeus, leveys ja kierto jokaisen ulottuvuuden akselin ympäri. Kappaleen sijaintia ja kiertymää kuvataan yleistetyillä koordinaateilla, joiden avulla kuvataan täydellistä kappaleen asemaa globaalissa avaruudessa. (Shabana 1998, s. 91.)

Kappaleen asema ilmaistaan globaalin koordinaatiston suhteen, mutta kappaleella on myös oma paikalliskoordinaatisto, jonka suhteen ilmaistaan kappaleen mikä tahansa kohta. Jäykän kappaleen $i$:n mikä tahansa pisteen $P$:n sijainti voidaan esittää globaalissa koordinaatistossa kappaleen paikalliskoordinaatiston suhteen, mikä esitetään globaalin koordinaatiston suhteen. Kuvassa 5 on esitetty jäykän kappaleen pisteen $P$ asema globaalissa koordinaatistossa. (Kortelainen 2011, s. 40–46.) Matemaattisesti pisteen asema voidaan esittää seuraavalla tavalla:

$$\boldsymbol{r} = \boldsymbol{R} + \mathbf{A}\bar{\boldsymbol{u}} \tag{1}$$

jossa $\boldsymbol{r}$ ilmaisee pisteen $P$:n aseman globaalilla asemavektorilla, $\boldsymbol{R}$ ilmaisee jäykän kappaleen $i$:n paikalliskoordinaatiston aseman globaalilla asemavektorilla, **A** on kiertomatriisi eli transformaatiomatriisi, mikä ilmaisee miten kappaleen paikalliskoordinaatisto on asettunut globaaliin koordinaatistoon nähden ja $\bar{\boldsymbol{u}}$ ilmaisee pisteen $P$ aseman paikallisella asemavektorilla. Kappaleen ollessa jäykkä, voidaan olettaa kappaleen minkä tahansa pisteen etäisyyden kappaleen paikalliskoordinaatistosta pysyvän vakiona, jolloin mille tahansa kappaleen pisteelle pätee yhtälö 1. (Kortelainen 2011, s. 40–46.)




**Kuva 5.** Jäykän kappaleen pisteen P:n määrittely (Shabana 1998, s. 11).

Jotta pystytään ilmaisemaan pisteen $P$ sijainti globaalissa koordinaatiossa, tarvitaan transformaatiomatriisi **A** muuttamaan paikallisen koordinaatiston vektori globaalin koordinaatistoon. Kappaleen kierto voidaan esittää eri tavoilla, joista konetekniikassa yleisimmin käytössä ovat Eulerin kulmat ja Eulerin parametrit. Tässä työssä käytetään Eulerin kulmia. Eulerin kulmien menetelmässä toteutetaan yksittäinen kierto kiertoakselin $X$, $Y$ tai $Z$ ympäri määritetyssä järjestyksessä. Kiertojen toteutus järjestyksellä on merkitystä, jonka takia mahdollisia kiertojärjestyksiä on 12 erilaista, kuten $X - Y - Z$ tai $Z - X - Z$. Asema- ja kiertokoordinaateista voidaan muodostaa yleistettyjen koordinaattien vektori:

$$\boldsymbol{q} = [R_x \; R_y \; R_z \; \boldsymbol{\theta}^T]^T \tag{2}$$

jossa $R_x$, $R_y$ ja $R_z$ ovat asemakoordinaatteja ja $\boldsymbol{\theta}$ on kiertymäkoordinaatiston vektori.

(Kortelainen 2011, s. 40–46.)

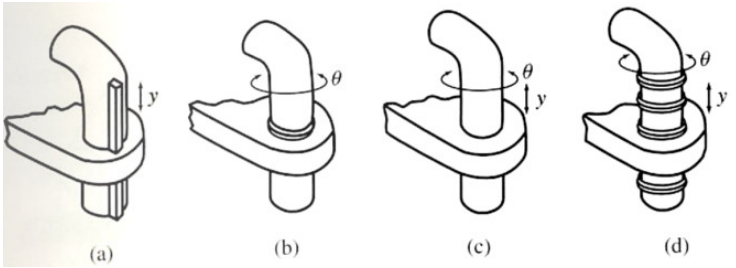
Monikappalejärjestelmässä kappaleet ovat kiinnitetty toisiinsa, mikä rajoittaa kappaleen liikettä kiinnitystavan mukaisella rajoitteella. Kappaleiden välisiä rajoitteita eli niveliä voidaan muodostaa lukitsemalla kappaleiden vapausasteita. Näistä rajoitteista saadaan rajoiteyhtälöt, jotka sallivat kappaleiden liikkuvan vain rajatulla tavalla. Esimerkiksi jos kappaleelta poistetaan kaikki vapausasteet, ei pysty kappale liikkumaan alkutilanteesta. Jos kappale on kiinnitetty sarananivelellä toiseen kappaleeseen, kappaleelle jää yksi kiertymävapausaste vapaaksi suhteessa toiseen kappaleeseen. Laskemalla kappaleiden vapausasteiden määrän ($n_b$) ja kappaleiden välisten rajoitteiden määrän ($n_c$), saadaan seuraavalla yhtälöllä laskettua mekanismin vapaaksi jäävien liikeratojen eli vapausasteiden lukumäärän (Degrees of freedom, DOF):

$$DOF = 6 \times n_b - n_c \tag{3}$$

Tätä yhtälöä kutsutaan Kutzbachin yhtälöksi. (Shabana 1998, s. 19.)

Mekanismin vapausasteiden määrällä voidaan havainnoida kappaleketjun liikemahdollisuudet. Mekanismi pystyy liikkumaan, jos vapausasteita on vähintään yksi. Jos taas vapausasteiden määrä on 0, mekanismi on staattisesti määrätty tai määräämätön liikkumaton rakenne. Kaava ei kuitenkaan pidä paikkaansa kaikissa tapauksissa, joten monikappaledynamiikan teoriaa on myös hyvä ymmärtää tarkistaakseen yhtälön paikkansapitävyyttä. (Airila 1996, s. 4.)

Kuvassa 6 on esitetty joitakin mekaanisia niveltyyppejä ja mitkä vapausasteet jäävät vapaiksi kyseisessä nivelessä. (Shabana 1998, s. 19.) Teoriassa liitoksia eli rajoitteita voi olla niin monta erilaista kuin on mahdollista muodostaa yhdistelmiä vapausasteista. Käytännössä kuitenkin liitos vaihtoehtoja on rajallinen määrä.




**Kuva 6.** Esimerkki eri mekaanisista nivelistä. (a) translaationivel (b) kiertonivel (c) sylinterinivel (d) ruuvinivel (Shabana 1998, s. 3).

### 2.2.2 Kinetiikka

Kinetiikassa sovelletaan kinematiikan yhtälöitä ja lasketaan monikappalejärjestelmän virtuaalinen työ. Monikappalejärjestelmän liikeyhtälöt perustuvat Newtonin toiseen liikelakiin, joka tunnetaan myös dynamiikan peruslakina, ja d'Alembertin virtuaalisen työn periaatteeseen. Newtonin toinen liikelaki voidaan kirjoittaa seuraavalla tavalla:

$$\boldsymbol{F}_a = \int_V \rho \ddot{\boldsymbol{r}} dV \tag{4}$$

jossa $\boldsymbol{F}_a$ on kappaleeseen kohdistuvan ulkoisten voimien vektori, $\rho$ on kappaleen tiheys, $V$ on kappaleen tilavuus ja $\ddot{\boldsymbol{r}}$ on kappaleen määritetyn pisteen kiihtyvyys. Tämä tarkoittaa, että kappaleen inertiavoimat tai vaikuttava momentti on yhtä suuri kuin kappaleeseen kohdistuvien ulkoisten voimien aiheuttama momentti. D'Alembertin virtuaalisen työn periaate on esitetty seuraavassa yhtälössä:

$$\delta W = \sum \left( \boldsymbol{F}_a^T - \int_V \rho \ddot{\boldsymbol{r}}^T dV \right) \delta \boldsymbol{r} = 0 \tag{5}$$

jossa $\delta W$ ilmaisee virtuaalista työtä ja $\delta \boldsymbol{r}$ ilmaisee virtuaalista siirtymä. Newtonin klassisen mekaniikan yhtälö 4 kuvaa vapaiden kappaleiden dynamiikkaa ja d'Alember periaate ottaa huomioon myös rajoitteiden reaktiovoimat. Soveltaen d'Alembertin periaatetta ja dynamiikan peruslakia, voidaan inertiavoimien tekemä virtuaalinen työ kirjoittaa seuraavaan muotoon:

$$\delta W_i = \boldsymbol{F}_i^T \delta \boldsymbol{r} = \int_V \rho \ddot{\boldsymbol{r}}^T dV \delta \boldsymbol{r} \tag{6}$$

missä $\boldsymbol{F}_i$ on inertiavoimien vektori karteesisessa koordinaatistossa. (Kortelainen 2011, s. 40–46.)

Inertiatiedot eli kappaleen massahitausominaisuudet ilmoitetaan inertiamatriisina yhtälön 7 mukaan. Inertia on kappaleen massan aiheuttama vastustus ulkopuolisten voimien ja momenttien aiheuttamalle kiihtyvyyksille. Kappaleen inertiaominaisuudet voidaan kirjoittaa symmetriseen 3x3 matriisi muotoon joka sisältää hitausmomentit ja hitaustulot. Massa vastustaa translaationalista kiihtyvyyttä, kun taas hitausmomentti ja hitaustulot vastustavat hitaustulon tekijöitä.

$$I = \begin{bmatrix} I_{xx} & -I_{xy} & -I_{xz} \\ -I_{yx} & I_{yy} & -I_{yz} \\ -I_{zx} & -I_{zy} & I_{zz} \end{bmatrix} \tag{7}$$

Matriisissa vastakkailla hitausinertioilla on yhtäläinen vaikutus, joten voidaan todeta matriisin olevan symmetrinen. Hitaus arvot voidaan laskea minkä tahansa kappaleen pisteen suhteen. Yleisin tapa on laskea massakeskipisteen suhteen, mutta tässä työssä hitausarvot lasketaan kappaleen paikalliskoordinaatiston suhteen.

Jäykän kappaleen monikappalejärjestelmän liikeyhtälöt voidaan määrittää soveltamalla d'Alembertin periaatetta ja asettamalla inertiavoimien tekemä virtuaalinen työ yhtä suureksi, kuin ulkoisten voimien tekemä virtuaalinen työ, jolloin saadaan seuraavat ehdot:

$$\delta W_i = \delta W_e \tag{8}$$

jossa $\delta W_e$ on ulkoisten voimien tekemä virtuaalinen työ. Jotta yhtälö voidaan soveltaa monikappalejärjestelmän liikeyhtälönä, tulee siihen lisätä kappaleiden välisten nivelvoimien synnyttämä virtuaalinen työ. Rajoitevoimat voidaan ottaa huomioon joko lisäysmenetelmällä tai sijoitusmenetelmällä. (Kortelainen 2011, s. 46.)

Tämän työn ohjelmisto käyttää oletuksena lisäysmenettelyä. Lisäysmenettelyssä nivelvoimat on lisätty liikeyhtälöihin Lagrange kertoimen avulla. Virtuaalityön yhtälö voidaan nyt kirjoittaa seuraavaan muotoon:

$$\delta W = \delta W_i - \delta W_e - \delta W_c = 0 \tag{9}$$

jossa $\delta W_c$ on rajoitevoimien tekemä virtuaalinen työ. Tällöin liikeyhtälöt voidaan kirjoittaa uuteen muotoon:

$$\begin{aligned} (\ddot{\boldsymbol{q}}^T \mathbf{M} - \boldsymbol{Q}_v^T - \boldsymbol{Q}_e^T - \boldsymbol{Q}_c^T)\delta \boldsymbol{q} &= 0 \\ \Rightarrow \mathbf{M}\ddot{\boldsymbol{q}} - \boldsymbol{Q}_v - \boldsymbol{Q}_e - \boldsymbol{Q}_c &= 0 \end{aligned} \tag{10}$$

jossa $\ddot{\boldsymbol{q}}^T \mathbf{M}$ kuvaa massahitauden tekemää työtä ja $\boldsymbol{Q}_c$ on yleistettyjen rajoitevoimien vektori. Yhtälössä $\mathbf{M}$ on kappaleen massamatriisi ja $\ddot{\boldsymbol{q}}$ on yleistettyjen koordinaatistojen kiihtyvyys vektori. Rajoitevoimien vektori $\boldsymbol{Q}_c$ voidaan kirjoittaa muotoon $\boldsymbol{C}_{\boldsymbol{q}}^T \boldsymbol{\lambda}$, missä $\boldsymbol{C}_{\boldsymbol{q}}$ on yleistettyjen koordinaattien $\boldsymbol{q}$ suhteen derivoitu rajoitevektori $\boldsymbol{C}$ ja $\boldsymbol{\lambda}$ on Lagrangen kertoimien vektori.

(Kortelainen 2011, s. 44–48.)

Näillä lisäyksillä virtuaalisen työn lauseke soveltuu monikappalejärjestelmän laskemiseen, mutta samalla siihen syntyy uusia muuttujia Lagrange kertoimen muodossa. Jotta yhtälöryhmän pystytään ratkaisemaan, lisätään siihen algebrallinen rajoiteyhtälöjoukko, kuten on lisätty seuraavaan liikeyhtälö joukkoon:

$$\left.\begin{aligned} \mathbf{M}\ddot{\boldsymbol{q}} - \boldsymbol{Q}_v - \boldsymbol{Q}_e + \mathbf{C}_q^T\boldsymbol{\lambda} &= 0 \\ \mathbf{C} &= 0 \end{aligned}\right\} \tag{11}$$

Monikappalejärjestelmässä nivelet ja rajoitteiden liikkeet rajoittavat toisiinsa liitettyjen kappaleiden liikkeitä. Näitä rajoitteita voidaan kuvata seuraavasti:

$$\boldsymbol{C} = \boldsymbol{C}(\boldsymbol{q}, t) = 0 \tag{12}$$

jossa $\boldsymbol{C}$ on lineaarinen itsenäisten rajoiteyhtälöiden vektori, $t$ on aika ja $\boldsymbol{q}$ on monikappalejärjestelmän yleistettyjen koordinaattien vektori. Rajoiteyhtälöt ovat lisäyhtälöitä jotka määrittävät esimerkiksi kahden kappaleen paikallisessa koordinaatistossa olevan nivelpisteen $P$ olevan globaalissa koordinaatistossa samassa asemassa. (Kortelainen 2011, s. 47.)

Lisäysmenettelyllä saadaan monikappalejärjestelmän virtuaalisen työn lauseke seuraavan muotoon:

$$\begin{bmatrix} \mathbf{M} & \mathbf{C}_q^T \\ \mathbf{C}_q & 0 \end{bmatrix} \begin{bmatrix} \ddot{\boldsymbol{q}} \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \boldsymbol{Q}_e + \boldsymbol{Q}_v \\ \boldsymbol{Q}_c \end{bmatrix} \tag{13}$$

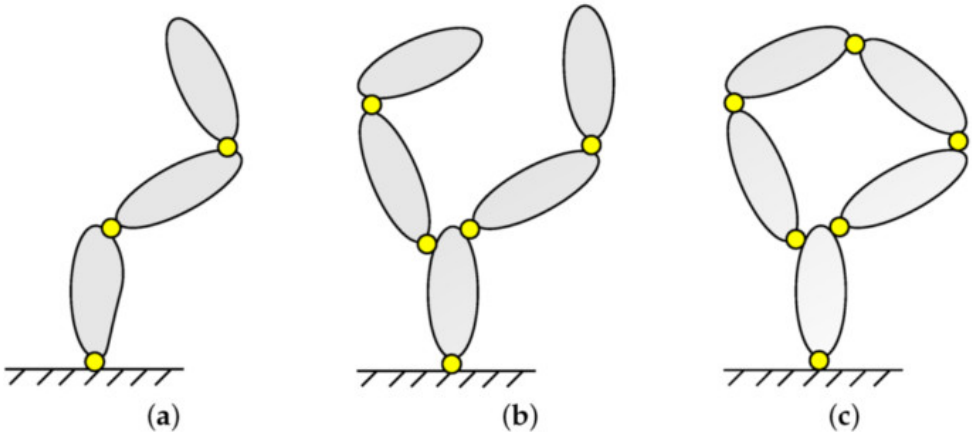
jossa $\mathbf{C}_q$ on rajoiteyhtälöiden Jacobin matriisi ja T tarkoittaa matriisin transformaatiota. Tässä yhtälössä tuntemattomat muuttujat ovat kiihtyvyydet $\ddot{\boldsymbol{q}}$ ja Lagrangen kertoimet $\boldsymbol{\lambda}$. Lagrangen kertoimien avulla voidaan määrittää rajoitevoimat yhtälössä $\boldsymbol{C}_{\boldsymbol{q}}^T\boldsymbol{\lambda}$, ja integroimalla kiihtyvyydet voidaan määrittää kappaleiden nopeudet ja yleistetyt koordinaatit kappaleille eli kappaleiden asemat. (Shabana 2001, s. 429–430.)

Useammista CAD-ohjelmistoista löytyy inertian ja massan laskentatyökalut, joiden avulla saadaan tarvittavat tiedot kappaleista simulointia varten. Kappaleen inertia-arvoja laskiessa tulee huomioida minkä suhteen hitautta lasketaan: massakeskipisteen suhteen tai jonkun

muun pisteen suhteen, kuten kappaleen paikalliskoordinaatiston suhteen. Eri ohjelmistojen yhteiskäyttö saattaa olla hankalaa tuotekehityksen näkökulmasta, minkä takia monet ovat vältelleet MBS-mallien mallinnusta. Kuitenkin modernin suunnittelutyön siirtyminen enemmän virtuaaliseksi on tuonut samalla suunnittelijoita lähemmäksi monikappaledynamiikkaa ja mekaniikan simulointia. (Kortelainen 2011, s. 5.)

Kappaleen 2.2 perusteella jokaisesta järjestelmän kappaleesta tulee tietää inertia, massa ja nivelpaikka arvot. Lisäksi tarvitaan määrittelyt ulkoisista voimista ja toimilaitteista kuten sähkömoottoreista, hydrauliikasta, pneumatiikasta ja antureista. Näiden tietojen perusteella pystytään muodostamaan virtuaaliprototyypistä mahdollisimman realistinen. Suurin osa näistä arvoista saadaan tuotekehitysprosessin alkuvaiheesta, jonka jälkeen määrittelyt tarkentuvat. Ensimmäisessä simulaatioversiossa käytetään karkeita laskelmia ja arvioita kappaleiden arvoista. Tämän jälkeen simulaatiosta saatujen tulosten perusteella pysytään tarkentamaan arvoja. Tuotekehitysprosessin iteroinnissa on tärkeätä, että kaikki tietävät mitä arvoja tarvitaan ja missä muodossa.

Rakentaessa MBS-mallia tulee ottaa huomioon liitosten tyypit ja niiden määrä, jotta yhtälön 3 mukaan laskettu vapausasteiden määrä on enemmän kuin nolla. Jos vapausasteiden määrä on erittäin alhainen verrattuna kappaleiden määrään, saattaa mekanismi olla ylirajoittunut. Tämän takia kannattaa isoissa kokonaisuuksissa suunnitella laitteen topologia tätä ajatellen, jolloin kiinnittää helpommin huomiota, jos järjestelmä on ylirajoittunut. Myös kinemaattisen ketjun muodostamiseen tulee kiinnittää huomiota. Kuvassa 7 on esitetty tyypillisiä monikappalejärjestelmän kinemaattisen ketjun rakenteita. Tässä työssä käytetään Mevea ohjelmiston rekursiivista ratkaisijaa, joka on suunniteltu pääosin kuvan a ja b tapauksiin. Työssä muodostuu yksi kuvan c kohdan mukainen suljettuketju. Suljettuketju saadaan muutettua puumaiseksi rakenteeksi käyttämällä katkaisuniveltä. Katkaisunivel sijoitetaan yhteen suljetun ketjun niveleen, johon muodostuu molempien liitosten kohdalle virtuaalinen nivel. Näin malli pysyy realistisena eikä suljettuketju kaada matemaattista laskentaa.




**Kuva 7.** Tyypillisiä monikappalejärjestelmän topologioita. A) avoin silmukka topologia, b) puutopologia ja c) suljettu ketju.

## 2.3 Virtuaalisen mallin luominen

Edellisen kappaleen teorian pohjalta voidaan käydä läpi mallin komponentteja yleistetymmin. Seuraavassa kappaleessa käydään läpi käytettyä ohjelmistoa ja sen mahdollisuuksista.

### 2.3.1 Mevea ohjelmisto

Mevean simulointiohjelmisto perustuu Mevean reaaliaikaiseen simulointimoottoriin. Mallinnus työkalujen avulla voidaan tarjota mahdollisuus mallintaa ja testata uusia ratkaisuja jo konsepti alku vaiheessa. Ohjelmiston avulla voidaan luoda mekaniikasta monikappalemalli, johon voidaan liittää hydrauliikka, voimansiirto komponentteja, kitka, virtuaalisia antureita, erilaisia ympäristöjä ja ulkoisia voimia. Mevea ratkaisijaan (Solver) voidaan lähettää signaaleja tai lukea mallista erilaisia signaaleja simulointiajon aikana. Mallin ohjaaminen ulkoisella ohjelmalla tai hallintalaitteella onnistuu siihen luodun rajapinnan kautta. Simulointia mallia voidaan esittää tietokoneen ruudulta tai simulaatiokopista, jossa simulointi malli esitetään usealla eri näytöllä. (Mevea 2016.)

Ohjelmiston avulla voidaan tutkia monikappalejärjestelmää reaaliajassa. Mallin avulla voidaan tutkia mekaniikassa syntyneitä voimia ja miten eri ratkaisut vaikuttavat näihin voimiin. Malli mahdollistaa myös eri mekaniikkaratkaisujen testaamista koko koneessa, mikä lisää koneen yleisymmärrystä. Näillä tiedoilla pystytään nopeuttamaan tuotekehityksen iterointia. Ohjelmiston avulla voidaan myös testata ihmisen tai ohjausjärjestelmän vaikutusta mekaniikkaan, mitä on voitu aikaisemmin testata vasta fyysisen prototyypin valmistuttua. Tämä vähentää myös tarvetta päivittää mekaniikkaa myöhemmin. Samaa simulaatiomallia voidaan

myöhemmin käyttää henkilöstön koulutukseen tai tuote-esittelyyn markkinoilla. Tällöin ei tarvitse tuoda koko laitetta esittelyyn, vaan virtuaaliprototyypin avulla asiakas saa samanlaisen tuntuman kuin oikeasta koneesta.

### 2.3.2 Ratkaisu menetelmä

Erilaisia numeerisia mallinnusmenetelmiä on kehitetty tietokoneavusteiselle mekaniikan mallinnukselle. Suurin osa menetelmistä perustuvat Lagrangen menetelmään. Mevea ohjelmisto tarjoaa kaksi eri ratkaisumenetelmää monikappalejärjestelmälle: Lagrange menetelmä ja rekursiivinen ratkaisumenetelmä. Lagrange menetelmä on raskaampi, koska siinä lasketaan kaikki nivelvoimat. Mallin pitäminen kasassa on matemaattisesti haastavampi, mutta tällä menetelmällä pystytään tarkastella, minkälaisia voimia kohdistuu niveliin. Rekursiivisessa ratkaisumenetelmässä nivelten voimia ei lasketa. Tämä johtaa parempaan suorituskykyyn, mutta vähentää saatua tietoa. Kun mallia käytetään ohjausjärjestelmän testaamiseen, henkilöstön koulutukseen tai tuotteen esittelyyn, käytetään silloin rekursiivista menetelmää. Jos halutaan tarkastella niveliin kohdistuneita voimia, on ratkaisutavaksi valittava Lagrange menetelmä. Lagrangen menetelmä on myös erittäin tarkka nivelten oikein asettelusta. Jos halutaan tarkistaa, onko laitteen topologia määritelty oikein, voi Lagrangen ratkaisumenetelmällä sen tarkistaa. Menetelmä ei anna mallin käynnistyä, koska siinä syntyy matemaattisia virheitä.

Lagrangen menetelmä on yleispätevämpi, mutta samalla vaatii paljon laskentatehoa. Rekursiivinen laskenta on paljon tehokkaampi, mutta vaatii erityistä huomiota liitosnivelten kanssa. Rekursiivisessa menetelmässä nivelten pitää olla kinemaattisen ketjun mukaisessa järjestyksessä, jolloin suljetut kinemaattiset ketjut on avattava ja vain tietyt niveltyypit ovat käytettävissä. (Mevea solver reference Manual 2015, s. 12.)

### 2.3.3 Mevea ohjelmiston I/O -rajapinnan käyttö

Mevea ohjelmistoon on luotu erilaisia rajapintoja. Ohjelmiston avulla voi ottaa yhteyttä työkoneiden CAN-väylään (Controller Area Network, CAN) tai tietokoneiden väliseen kommunikointiin tarkoitettuun sokettiyhteyteen. Tässä työssä tullaan käyttämään Mevea ohjelmiston sokettirajapintaa ja Mevean I/O-poolia. Luodun rajapinnan kautta ulkopuolinen ohjelmisto pystyy kommunikoimaan ratkaisijan kanssa. Kahdella eri rajapinnalla on eri soveltavuus. Sokettiyhteys käyttää TCP/IP-protokollaa (Transmission Control Protocol, TCP, Internet protokol, IP) yhteyden luomiseen. Protokolla sisältää kommunikoinnin varmistuksen, mikä tarkoittaa, että molemmat osapuolet varmistavat jatkuvasti, onko perille tullut viesti oikea.

Tällöin toinen osapuoli odottaa aina vastauksen ennen, kuin jatkaa. TCP/IP-protokolla soveltuu yhteystavaksi kun synkronoitu yhteys on vaadittu.

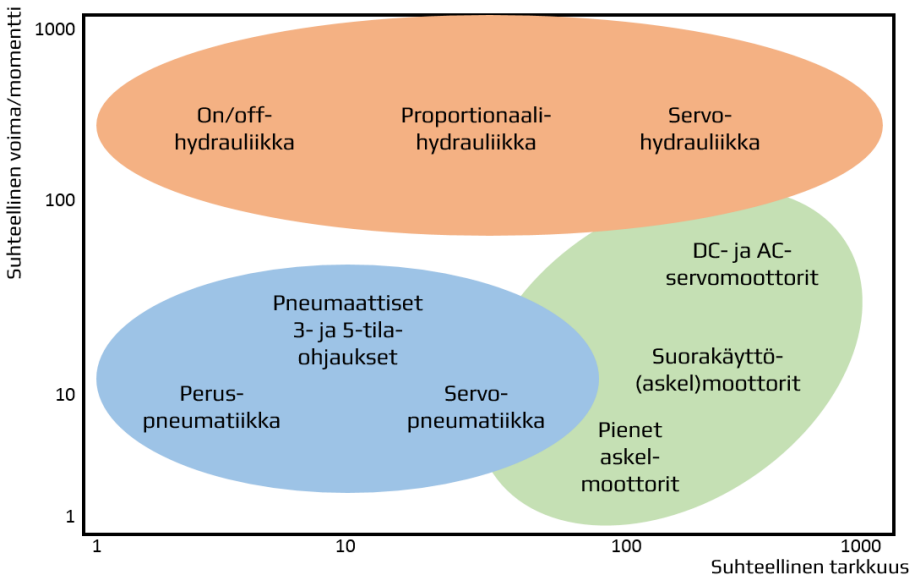
Mevean I/O-pooli on monipuolisempi kuin sokettiyhteys. Siinä varataan tietokoneen muistista tietty paikka kullekin yhteydelle, johon kirjoitetaan analogi- tai digitalisignaalia. Tietyt muistipaikat on jaettu eri lohkoihin, joiden sisällä on tietty määrä kanavia. Käytännössä I/O-poolissa on neljä eri rajapintaa: analoginen sisään ja ulos, ja digitaalinen sisään ja ulos. Jokaisella rajapinnalla on 32 lohkoa, joissa jokaisessa lohkossa on 32 kanavaa eli yhteensä 1024 kanavaa per rajapinta.

Tämä menetelmä ei pakota luomaan synkronointia mallin ja ohjauksen välille. Ohjelmistolle kerrotaan mihin kanavaan ja lohkoon kirjoitetaan mitäkin tietoja ja mistä lohkosta ja kanavasta luetaan ohjausarvoja. Tätä samaa menetelmää käyttävät myös useammat tietokonepeliohjaimet. Ominaisuus mahdollistaa erilaisten peliohjainten käytön virtuaaliprototyypin testaamiseen. Mevealla tehtyä mallia voidaan ohjata I/O-pooliin kautta, mutta I/O-poolin voi myös itse kirjoittaa signaaleja Mevean I/O C-kirjaston avulla. Kirjasto sisältää eri funktioita, joilla voidaan lukea halutusta kanavasta ja kirjoittaa haluttuun kanavaan.

### 2.3.4 Voimanlähteet

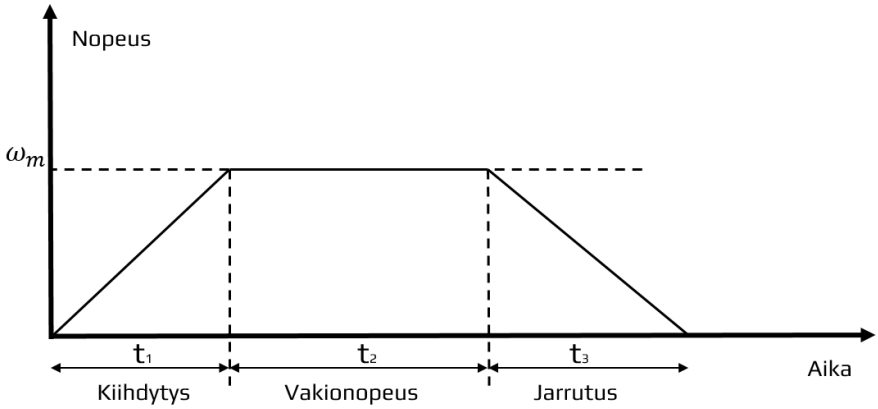
Uudensukupolven viilusorvissa suurin osa hydrauliikasta on korvattu sähkömoottoreilla. Näin parannetaan mekaniikan luotettavuutta, tarkkuutta ja kustannustehokkuutta. Järjestelmä ei myöskään ole niin herkkä lämpötilavaihteluille, eikä turhia vuotoja synny. Kuitenkaan sähkömoottoreilla ei pystytä korvamaan kaikkia hydrauliikka komponentteja, joten viilusorvissa on myös joitakin hydrauliikka- ja pneumatiikka komponentteja.

Kuvassa 8 on esitetty voimakomponenttien suhteellinen taulukko, mikä antaa suuntaa-antavaa tietoa minkälaisia voimakomponentteja tulisi valita kuhunkin toimintaan. Taulukosta huomataan, että suurin osa pneumatiikka ja hydrauliikkakomponenteista voidaan korvata sähkömoottoreilla, jolloin saavutetaan laitteen tarkempi toiminta. Kuitenkin suurimpia hydrauliikkakomponentteja ei pystytä korvamaan sähkömoottoreilla niiden voimakapasiteetin takia.




**Kuva 8.** Suuntaa-antava hydrauliikan, pneumatiikan ja sähkömoottorin käytöllä saavutettavasta suhteellisesta voima/momentista ja tarkkuudesta (muokattu Airila 1996, s. 2).

Suunniteltaessa uuden toimilaitteen asentamista tulee ottaa huomioon eri mitoitussääntöjä. Tärkeintä toimilaitteen valinnassa on tietää, minkälaiseen kuormitukseen toimilaite joutuu. Toimilaitteet liikkuvat yleensä työsykleissä, mikä koostuu kiihdytyksestä, tasaisesta liikkeestä ja jarrutuksesta. Kuvassa 9 on havainnoitu idealisoitu malli työsyklistä. Tätä kuvaajaa kutsutaan myös moottorin nopeusprofiiliksi. Toimilaitteen mitoitus aloitetaan analysoimalla keskimääräisen työkierron tilanteita, jossa otetaan huomioon ulkoinen kuormitus, hitausvoimat ja -momentit, kitka- ja impulssivoimat. (Airila 1996, s. 3.)



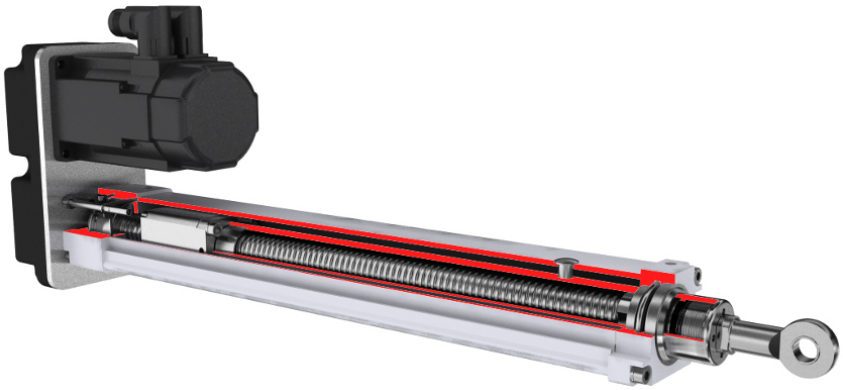

**Kuva 9.** Idealisoidun työkierron nopeus-aikakuvaaja (Airila 1996, s. 3).

Virtuaaliprototyyppi soveltuu työkierron analysointiin. Prototyypin avulla voidaan laitetta testata kuormittamalla sitä erilaisilla voimakomponenteilla, ja tutkia miten ne soveltuvat

annettuun työkiertoon. Virtuaaliprototyypin avulla saadaan tarkempaa arvoja, minkälaiseen rasitukseen toimilaite joutuu ja minkälainen toimilaitteen pitää olla.

### 2.3.5 Sähkömoottorit ja kuularuuvit

Uudessa viilusorvissa hydrauliikkasylinterit tullaan korvamaan sähköisillä lineaarisilla karamoottoreilla. Sähkömoottorien pyörimisliike muutetaan lineaariliikkeeksi kuula- tai rullaruuvilla. Valmistajat myyvät näitä yhdistelmiä pakettina, jolloin moottoreita ei joudu itse säätämään ja ohjelmoimaan. Kuvassa 10 on esitetty tyypillinen lineaarinen karamoottori. Siinä moottori on yhdistetty joko vaihteistolla tai hihnavälityksellä kierretankoon, joka liikuttaa kuula- tai rullaruuvia, johon sylinterin toinen pää on kiinnitetty. Karamoottoreiden sulautettu ohjausjärjestelmä on lineaariliikesäätöinen, joka tarkoittaa, että moottoria voidaan ohjata ruuvin asema-, nopeus- tai vääntösäädöllä. Moottorille lähetetään haluttu ruuvin asema, jolloin moottorin sulautettujärjestelmä tarkkailee kuularuuvin asemaa ja ohjaa sen mukaan sähkömoottoria. Näin jaetaan myös ohjausjärjestelmää pienempiin yksiköihin.



**Kuva 10.** Sähkökäyttöinen lineaarinen karamoottori (Heason Technology Ltd 2010).

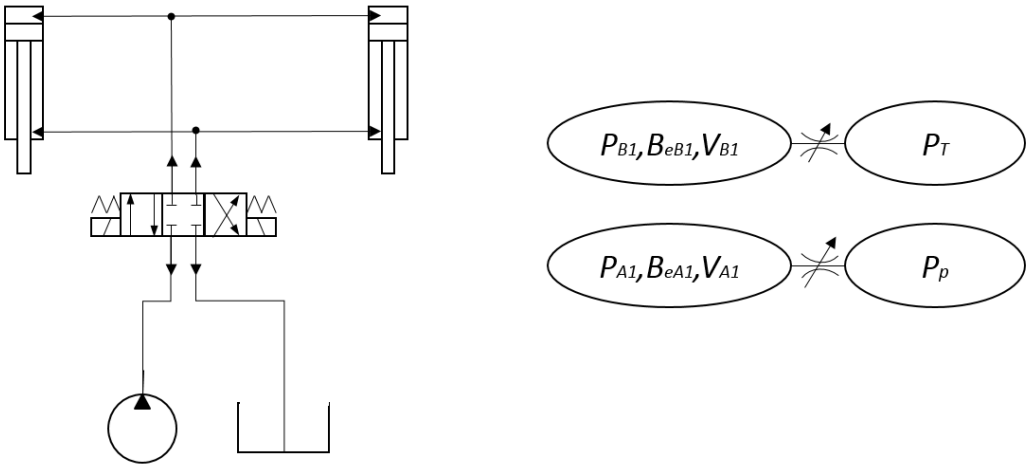
### 2.3.6 Hydrauliikka ja pneumatiikka

Vaikka suurin osa laitteen hydrauliikkakomponenteista on korvattu lineaarisilla karamoottoreilla, laitteessa on vielä yksi hydrauliikkajärjestelmä ja kolme pneumatiikkajärjestelmää. Hydrauliikkajärjestelmän tehtävä on pitää viiluterä oikeassa kulmassa sorvauksen aikana. Pneumatiikkajärjestelmällä ei ole tarkkuutta vaativia tehtäviä, vaan sen tulee keskittää pölli keskelle ja siirtää pyöristettyjä pöllejä karojen väliin. Jälkimmäinen tapaus toteutuu vain, jos koneeseen syötetään pyöristettyjä pöllejä.

Diplomityön aikana hydrauliikka- ja pneumatiikkapiirien suunnittelu oli keskeneräinen, joten järjestelmistä oli vähän tietoa. Toteutettujen järjestelmien komponenteille on haettu sellaiset

parametrit, että liikkeet onnistuvat normaalisti.

Hydrauliikkajärjestelmä, jossa ei esiinnyt akustisia paineaaltoja, mallinnetaan yleensä keskittyneiden paineiden teorialla. Teoriassa hydrauliikkapiiri jaetaan eri tilavuuksiin, joiden välillä paineet jakaantuvat tasaisesti. Näiden tilavuuksien välillä on kuristimia, jotka säätävät tilavuusvirtaa tilavuuksien välillä. (Galal Rabie 2009, s. 72.) Tämä yksinkertaistettu malli on esitetty kuvan 11 oikeassa laidassa.




**Kuva 11.** Idealisoitu hydrauliikkapiiri.

Kuvan 11 vasemmassa laidassa on esitetty idealisoitu hydrauliikkapiiri, jossa on kaksi kaksikammioista hydrauliikkasylinteriä, 4/3-suuntaventtiili, pumppu ja tankki. Mevea-ohjelmistolla pystytään mallintamaan hydrauliikkaa, jolloin vältytään erilliseltä hydrauliikka mallin rakentamiselta.

Pneumatiikkapiiri mallinnetaan matalapaineisena hydrauliikkapiirinä. Laitteessa pneumatiikkajärjestelmällä ei ole tarkkuutta vaativaa työtehtävää, joten yksinkertainenkin toimintatapa soveltuu simuloinnin tarkoituksiin.

## 2.4 Reaaliaikasimulointi

Virtuaaliprototyyppejä simuloitiin 2000-luvun alkuvaiheessa offline-tilassa, jolloin lyhyen testiajon analyysiin saattoi kulua huomattavasti aikaa. Tällöin kaikkien testien tuli olla tarkkaan suunniteltuja, eikä virheiden ja ihmisten reaaliaikaista vaikutusta voitu huomioida helposti. Ratkaisumenetelmien kehitettyä nopeammiksi ja mahdollisuus liittää malliin laitteen ohjausjärjestelmä, ovat mahdollistaneet käyttäjän vaikutuksen testaamisen reaaliajassa.

Reaaliaikainen simulaatio tarkoittaa sitä, että yhden sekunnin simulointiin menee maksimissaan yksi reaaliaikainen sekunti. Reaaliaikaisuus asettaa rajoitteita simulointimallin koolle ja ratkaisumenetelmille. Suurin osa monikappaledynamiikan mallinnusohjelmistoista eivät sovellu reaaliaikaiseen simulointiin, koska ne ovat suunniteltu offline-simulointiin. (Rouvinen 2003, s. 8.)

Reaaliaikasimuloinnissa on kiinnitettävä huomiota simuloinnin aika-askeleen suuruuteen. Aika-askeleen suuruus kertoo, kuinka tiheästi mallia päivitetään eli esimerkiksi, kuinka tiheästi kappaleen asemaa lasketaan. Tietokoneen laskentateho määrää, kuinka suurta aika-askelta voidaan mallinnuksessa käyttää. Jos aika-askel on pienempi kuin tietokoneen laskentateholla pystytään laskea, simulointi muuttuu liian raskaaksi ja malli alkaa poiketa reaaliajasta. Tällöin mallia ei voida enää ohjata reaaliajassa. Jos aika-askel on liian suuri, saattaa se aiheuttaa liian nopeita asemamuutoksia, ennen kun kappaleet reagoivat toisiinsa. Liian suuri aika-askel tarkoittaa, että kappaleiden asemia ei tarkkailla tarpeeksi tiheästi suhteessa kappaleiden liikkeisiin. Suuri aika-askel mahdollistaa kappaleiden myöhempää reagointia toisiinsa, mikä aiheuttaa suurempaa poikkeamaa kontaktivoimissa ja nivelvoimissa. Kappaleet kerkeävät yhden aika-askeleen aikana menemään enemmän sisäkkäin, ennen kun kontaktivoimat alkavat vaikuttaa. Lisäksi hydrauliikka järjestelmään syntyy miinus tilavuuksia, kun esimerkiksi männän sylinteri ei pysähdy ennen päätyä, mikä viimeistään pysäyttää koko mallin.

Tietokoneen laskentakapasiteetti asettaa kuitenkin siis rajat, kuinka täsmällinen simulaatiomalli voi olla. Koska laskentakapasiteetti ei ole rajaton, laskennasta joudutaan jättämään joitakin vaikuttajia pois, tai yksinkertaistamaan laskentamenetelmiä. Nämä yksinkertaistukset ja poistot on pyritty optimoimaan niin, että mallin täsmällisyys pysyy hyväksyttävissä rajoissa. Mevea ohjelmistossa ja tässä työssä on tehty seuraavia oletuksia ja yksinkertaistuksia:

- Liitoksissa ei ole välystä.
- Liitoksissa ei ole kitkaa.
- Kappaleet eivät jousta, vaan ovat mallinnettu jäykkinä.
- Kappaleiden massat ja inertiat eivät muutu.
- Hydraulinen paine on jakautunut tasaisesti koko tilavuudelle.
- Hydrauliset virrat ovat turbulenssisia.
- Hydrauliikkaöljyn viskositeetti ja lämpötila muutokset eivät ole otettu hydrauliikkalaskentaan mukaan.

- Hydrauliikkasylinterit ovat mallinnettuna dummy kappaleina.
- Hydrauliikkapiirissä ei oteta huomioon paineaaltoja.

(Mevea solver reference Manual 2015, s. 12.)

Mevea ohjelmistossa kappaleiden välinen kontakti on mallinnettu jousivaimennusmenetelmällä. Menetelmässä lasketaan kahden kappaleen geometrioiden uppoaman avulla kontaktivoiman suuruus. Tämä mallintamismenetelmä saattaa aiheuttaa joissakin tapauksissa seuraavanlaisia ilmiöitä:

- Jäykät kappaleet asettuvat sisäkkäin visuaalisesti.
- Jotkut kontaktitapaukset aiheuttavat pientä värähtelyä.

Näissä tapauksissa kyse ei ole epäjohdonmukaisista fysiikan kaavoista tai epätarkkuuden kumuloitumisesta. Ongelma saattaa johtua pyöristysvirheestä, epästabiilista tilasta, liian suuresta aika-askeleesta tai numeerisista approksimaatiovirheistä. Niiden takia jopa pienikin muutos järjestelmän parametreihin saattaa aiheuttaa odottamattomia muutoksia järjestelmän käyttäytymiseen. Suurin osa tämän kaltaisista tapauksista voidaan minimoida tai poistaa määrittelemällä ja hienosäätämällä materiaaliominaisuudet oikein.

Simuloinnissa käytetään CAD-työkaluilla tehtyjä 3D-kappaleita grafiikkana. Useimmissa tapauksissa CAD-mallit ovat liian yksityiskohtaisia. Yksityiskohtaiset grafiikat saattavat viedä liikkaa laskentatehoa, jolloin malli ei enää pysy reaaliajassa. Kappaleiden visuaalisen grafiikan yksinkertaistaminen keventää laskennan kuormitusta ja helpottaa mallin pysymistä reaaliajassa. Koska massa- ja inertiatiedot ovat laskettu CAD-ohjelmistossa, kappaleiden yksinkertaistus ei vaikuta itse mallin täsmällisyyteen.

## 2.5 Julia-ohjelmointikieli

MBS-mallin tarkkuutta lisätään mallintamalla puusorvausprosessi matemaattisena leikkausmallina. Prosessimalli luodaan Julia-ohjelmointikielellä. Lisäksi moottoreiden ohjaamista varten tullaan luomaan moottoriohjausjärjestelmä Rauten ohjausjärjestelmän ja simulaatiomallin välille, joka toteutetaan myös Julia-ohjelmointikielellä.

Julia on korkean tason ja korkean suorituskyvyn dynaaminen ohjelmointikieli, joka on suunniteltu tekniseen- ja tutkimuslaskentaan. Julian just-in-time(JIT) kääntäjä tekee kielestä lähes yhtä suorituskykyisen kuin C -ohjelmointikieli. Kuvassa 12 on vertailtu eri ohjelmointikielien suorituskykyä suhteessa C-kieleen. Testissä on ajettu samanlaisia

ohjelmointitehtäviä eri kielillä ja verrattu näiden suoritusaikaa. Jos kerroin on yks, kieli on suoriutunut tehtävästä yhtä nopeasti kuin C-kieli. Taulukosta voidaan todeta, että Julia on kokonaisuudessaan nopein korkeantason kieli. Julia perustuu avoimeen lähdekoodiin, jonka kirjastoista suurin osa on kirjoitettu omalla kielellä. Niihin kuuluu suurin osa C- ja Fortan-kirjastoista. (Julia programming language 2016.)

| | **Fortran** | **Julia** | **Python** | **R** | **Matlab** | **Octave** | **Mathe-matica** | **JavaScript** | **Go** | **LuaJIT** | **Java** |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | gcc 5.1.1 | 0.4.0 | 3.4.3 | 3.2.2 | R2015b | 4.0.0 | 10.2.0 | V8 3.28.71.19 | go1.5 | gsl-shell 2.3.1 | 1.8.0_45 |
| fib | 0.70 | 2.11 | 77.76 | 533.52 | 26.89 | 9324.35 | 118.53 | 3.36 | 1.86 | 1.71 | 1.21 |
| parse_int | 5.05 | 1.45 | 17.02 | 45.73 | 802.52 | 9581.44 | 15.02 | 6.06 | 1.20 | 5.77 | 3.35 |
| quicksort | 1.31 | 1.15 | 32.89 | 264.54 | 4.92 | 1866.01 | 43.23 | 2.70 | 1.29 | 2.03 | 2.60 |
| mandel | 0.81 | 0.79 | 15.32 | 53.16 | 7.58 | 451.81 | 5.13 | 0.66 | 1.11 | 0.67 | 1.35 |
| pi_sum | 1.00 | 1.00 | 21.99 | 9.56 | 1.00 | 299.31 | 1.69 | 1.01 | 1.00 | 1.00 | 1.00 |
| rand_mat_stat | 1.45 | 1.66 | 17.93 | 14.56 | 14.52 | 30.93 | 5.95 | 2.30 | 2.96 | 3.27 | 3.92 |
| rand_mat_mul | 3.48 | 1.02 | 1.14 | 1.57 | 1.12 | 1.12 | 1.30 | 15.07 | 1.42 | 1.16 | 2.36 |

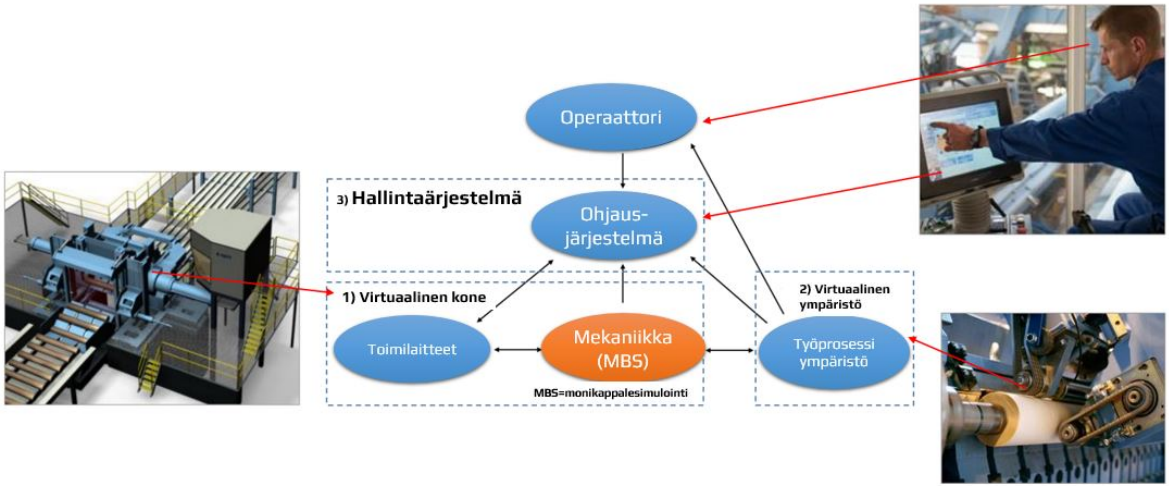
**Kuva 12.** Ohjelmointikielien vertailu suhteessa C-ohjelmointikieleen (Julia programming language 2016).

Julia-ohjelmointikieli on valittu tähän projektiin sen suorituskyvyn takia. Lisäksi kyseessä on korkean tason ohjelmointikieli, mikä helpottaa ohjelman rakentamista kiinnittämättä huomiota ohjelman rakenteeseen tai muistin varaukseen. Lisäksi laajan kirjaston takia ohjelmointikieltä pystytään hyödyntämään monipuolisesti.

# 3 VIRTUAALIMALLIN RAKENTAMINEN

Mekanismin toimintaan on hyvä tutustua ennen mallin muodostamista. Tiedon perusteella pystytään hahmottamaan mekanismin dynamiikka ja miten koneen eri osat on liitetty toisiinsa. Mitä paremmin tunnetaan koneen vaatimukset, sitä paremmin voidaan varmistua, että simulaattori toimii oikein. Näiden tietojen perusteella voidaan aloittaa muodostamaan virtuaaliprototyyppiä.

Kuvassa 13 on esitetty, mistä kokonaisuuksista viilusorvin virtuaaliprototyppi koostuu. Tässä työssä keskitytään kohtaan 1. virtuaalinen kone, johon kuuluu toimilaitteiden ja mekaniikan mallinnus. Lisäksi mallia varten luodaan moottoriohjausjärjestelmä, joka tulkitsee ohjausjärjestelmän käskyt ja ohjaa niiden perusteella liikkeitä. Lisäksi malliin luodaan rajapinta sorvausprosessimallille, mikä simuloi puun käyttäytymistä sorvissa.



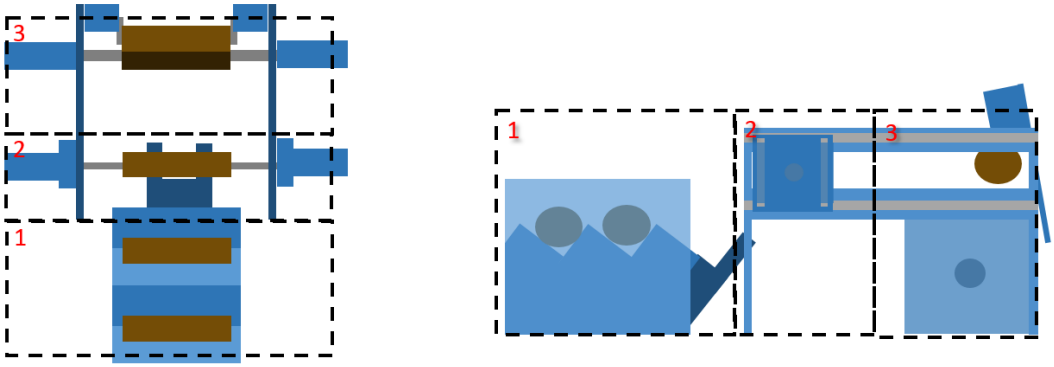

**Kuva 13.** Viilusorvauskoneen simulaattorin elementit.

## 3.1 Vaneriviilun sorvaus

Vanerin tuotantoprosessissa merkittävin kone on viilusorvi, jonka avulla vuollaan viilumattoa pöllistä. Viilusorvilla on suurin ja tärkein osuus vanerin valmistuksessa, jolloin sen virhetilaprosentti pyritään pitämään matalana. Viilusorvin asetuksilla voidaan määrittää viilun paksuutta, tasaisuutta, sileyttä ja lujuutta. Sorvatulle viilulle on monia ja haastavia vaatimuksia, kuten pintaviilun ulkonäkö, halutun vanerin mitat, lujuusominaisuudet ja

vanerin valmistusvaatimukset. Puun käytön hyötysuhde on Suomessa pienempi kuin monissa kilpailijamaissa, koska tukit ovat Suomessa pienihalkaisijaisia. Tämä korostaa kustannustehokkuutta entisestään. Hyötysuhdetta pystytään parantamaan keskittämällä pölli optimaalisella tavalla sorviin. Näin saadaan esimerkiksi koivutukista hyödynnettyä 58 % ja havupuusta 65 % vanerin valmistukseen. (Koponen 2002, s. 44.)

Diplomityön viilusorvi voidaan jakaa kuvan 14 mukaiseen kolmeen osakokonaisuuteen: pöllin syöttö (1), pöllin keskittäminen (2) ja pöllin sorvaus (3). Tässä kappaleessa käydään läpi näiden osakokonaisuuksien toimintamallia. Viilusorvin eri vaiheiden toimintaa voidaan kuvata seuraavasti: Ensiksi pölli siirretään porrasannostelijalle (1), josta porrasannostelija siirtää pöllin keskitykseen (2). Keskityksessä karat pyörittävät tukin ympäri samalla kun puun muoto mitataan (3). Tietokone laskee mittaustietojen perusteella sorvikarojen paikat, jonka jälkeen pölli siirretään sorvin karojen väliin (4). Tämän jälkeen pöllistä sorvataan viilumattoa (5), joka siirretään leikkaukseen tai kuivaukseen (6). Jäljelle jäänyt pyöristysjäte siirretään sellun raaka-aineeksi (7). Jos viilun tuotossa ilmenee epätasaisuuksia, sorvin terä täytyy huoltaa (8). (Koponen 2002, s. 38–39.)




**Kuva 14.** Viilusorvi jaettu 3:een osaan. Vasen kuva on laitteen yläpuolelta ja oikea kuva on laitteen sivusta. 1) Pöllin syöttö, 2) pöllin keskittäminen ja 3) pöllin sorvaus.

Kyseinen viilusorvi pystyy käsittelemään pöllejä, joiden halkaisija on pienimmillään 150 mm ja suurimmillaan 600 mm, ja pituudet voivat olla pienimillään 950 mm ja suurimmillaan 1650 mm. Laitetta testataan keskimääräisellä puulla, jonka halkaisija on 240 mm ja pituus on 1350 mm.

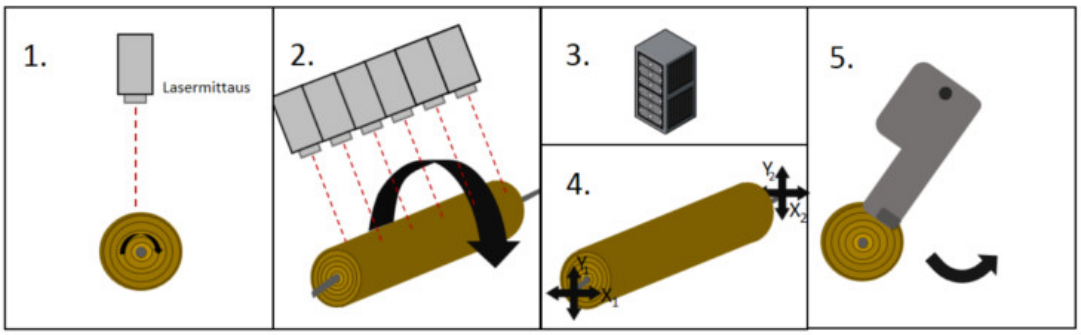
### 3.1.1 Vanerin koostumus ja vaatimukset

Vaneri koostuu levyn suuruisista ja tason suuntaisista viiluista, jotka on liimattu yhteen. Vanerituotteet voidaan jakaa rakenteen, pintaviilun laadun, liimauksen kosteuden- ja säänkestävyyden ja lujuusominaisuuksien perusteella eri standardiluokkiin. Pelkästään Euroopassa on käytössä lähes sata eri vaneri EN-standardia. Kuivauskoneen ja leikkauskoneen toimintaa helpottaakseen viilumaton on oltava ehjä ja mahdollisimman pitkä. (Koponen 2002, s. 38.)

Viilun laatu määrää vanerilevyn hinnan. Sorvinkäyttäjä tarkkailee työstöjälkeä ja tekee tarvittavia muutoksia koneen säätöihin, jos viilu ei vastaa laatuvaatimuksia. Kokenut sorvinkäyttäjä osaa jo silmämääräisesti arvioida, missä vaneriluokassa viilua voidaan käyttää. Tätä kokemusperäistä tietoa voidaan nostaa simulaattoriharjoituksilla.

### 3.1.2 Pöllin keskittäminen

Jotta pöllistä saadun viilun määrä ja laatu olisi optimaalinen, täytyy pölli asettaa sorvauskarojen väliin optimaalisessa asennossa. Pöllin pintakerroksista saa arvokkaampaa viilua kuin sydänosata, jolloin oikein toteutettu keskittäminen voi nostaa puun taloudellista hyötyä. Tätä varten pöllin geometria mitataan laserskannauksella ennen sorvausta, minkä tiedon perusteella löydetään parhaat karojen paikat. Ennen sorvausta pölli siirtyy keskittimelle, jossa pölli pyöräytetään kerran ympäri samalla kun pöllin muotoja mitataan. Laserit mittaavat määrätyin välein pölliä, jolloin mittaus datasta muodostuu kiekkoja, joiden perusteella tietokone laskee keskitysasennon. Menetelmän vaiheet ovat kuvattu kuvassa 15. Keskityksen jälkeen pölli asetetaan tietokoneen laskemassa asennossa sorvin karojen väliin. (Koponen 2002, s. 39.)

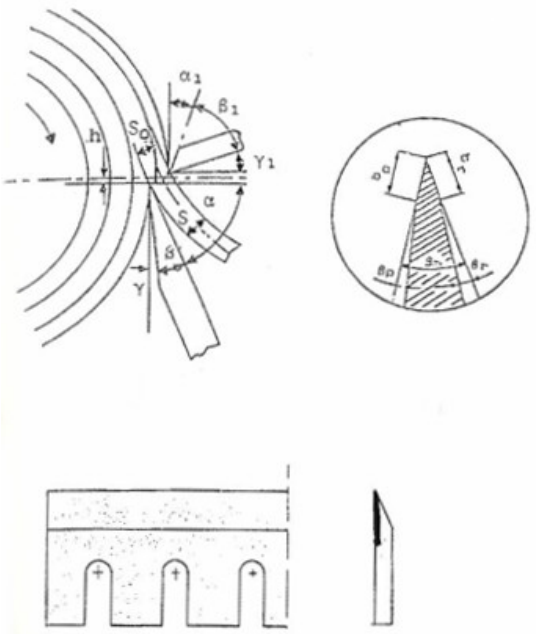



**Kuva 15.** Tietokoneavusteisen pöllin keskitysmenetelmän vaiheet: 1) ja 2) muodonmittaus lasersäteillä, 3) tietokone laskee optimaalisen aseman, 4) pölli keskitetään ja 5) pöllin kuljetus karojen väliin (muokattu Koponen 2002, s. 40).

Riippuen mittaussektorien tiheydestä voi liian harva tiheys jättää oksat huomiomatta, jolloin pölli käyttäytyy eri tavalla mitä tietokone on laskenut. Tämän seurauksena terä saattaa vaurioitua. Tämänhetkisessä keskittimessä keskimääräisesti mittauspisteiden välimatka on 3 mm, jolloin kaikki oksat tulevat huomioon. Lasermittausten avulla luodaan pöllistä pistepilvi, jonka perusteella asetetaan pölli sorviin.

### 3.1.3 Pöllin sorvaus

Viilun valmistustapoja on erilaisia, joista yleisin on sorvaaminen. Sorvaamisessa viilu syntyy, kun pölliä pyöritetään terää vasten ja samanaikaisesti siirtäen terää lineaarisesti kohti pöllin keskijanaa. Terä vuolee pöllistä viilua, joka puristuu terän ja vastaterän välisestä raosta. Kuvan 16 vasemmalla ylälaidassa on esitetty, miten terän ja vastaterän välistä puristuu viilua. (Koponen 2002, s. 37.)



| Leikkuuterä | | Vastaterä | |
|---|---|---|---|
| rintakulma | $\alpha$ | puristuskulma | $\alpha_1$ |
| teroituskulma | $\beta$ | teroituskulma | $\beta_1$ |
| päästökulma | $\gamma$ | päästökulma | $\gamma_1$ |
| mikrokulma | $\beta_m$ | | |
| – päästöpinta | $\beta_p$ | | |
| – rintapinta | $\beta_r$ | | |
| mikroviiste | | | |
| – päästöpinta | $b_p$ | | |
| – rintapinta | $b_r$ | | |

| Asete | | |
|---|---|---|
| Viilun nimellispaksuus | (mm) | Sn |
| Vastaterän suun etäisyys leikkuuterästä | (mm) | So |
| Leikkuuterän suun etäisyys tasosta A–A | (mm) | ho |
| Puristusaste p (%) | | 100(Sn–So)/Sn |

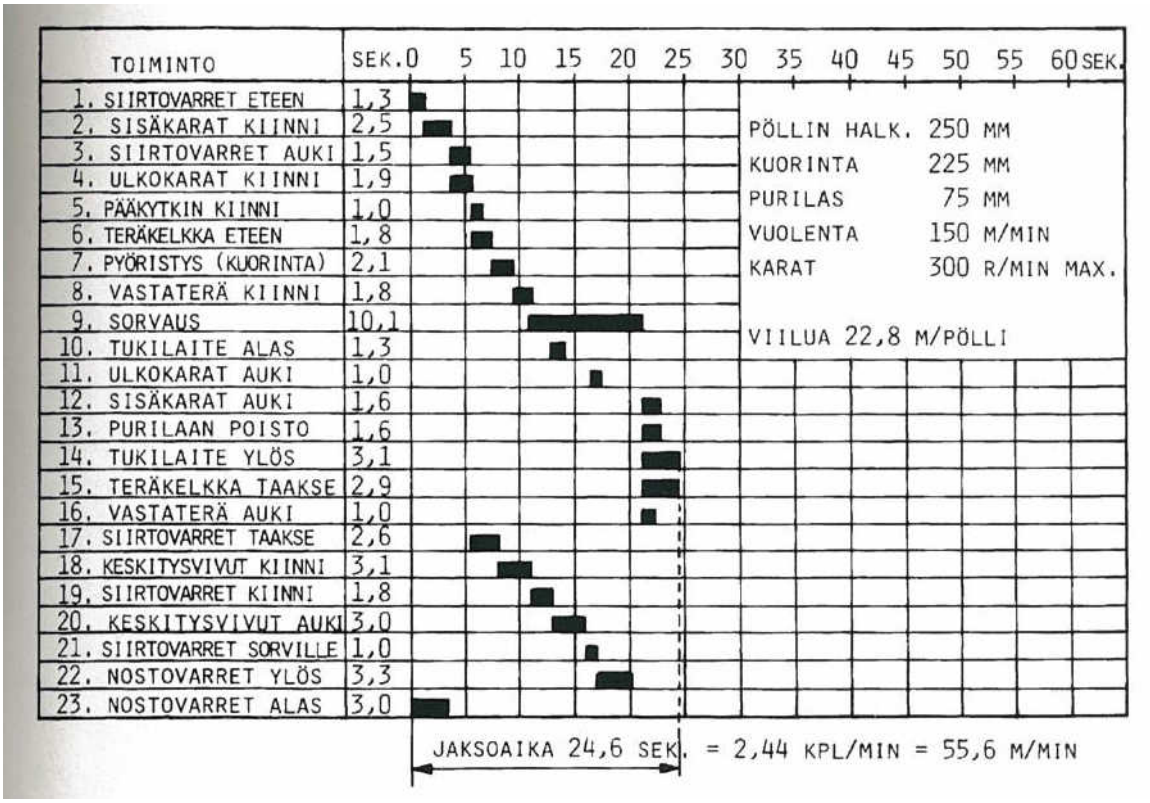
**Kuva 16.** Viilun sorvaus (Koponen 2002, s. 43).

Viilun valmistukselle on asetettu korkeat vaatimukset, jotka ovat tavoitteena jo vanerituotannon alussa. Tuotannon alkupään päätökset vaikuttavat loppupään tuotantoon. Esimerkiksi asettamalla pölli oikein sorviin voidaan estää viilun epätasaisen kuivumisen. Tällä estetään syntymästä ns. pitkää- tai lyhytpäistä viilua. Pöllin asettelu oikein, vaatii kokemuspohjaista tietoa. Myös viilujen liimaus asettaa suuria vaatimuksia viilulle. Jotta liima levittyisi tasaisesti ja kulutus olisi pientä, tulee viilun olla tasapaksuinen ja pinnan sileä. Jos sorvaus vaihe jättää

epätasaista jälkeä, aiheuttaa se taloudellisia menetyksiä valmistusprosessin liimausvaiheessa. Vanerin luokitus siirtyy huonompaan, jolloin vanerin arvo laskee. Viilun paksuutta ja poikittaisvetolujuutta pystytään mittamaan, mutta viilun tasomaisuutta, jännityksettömyyttä, suoruutta ja eheyttä arvioidaan silmämääräisesti kokemuspohjalta. (Koponen 2002, s. 37.)

Eri puulajeille asetetaan eri sorvausarvot, kuten esimerkiksi vastaterän puristusparametrit. Jotta viilulaatu olisi hyvä, tulisi seuraavat asetukset laitteistossa olla puulajin mukaiset: puristusasetukset, leikkuuterän korkeus ja leikkuukulma, vastaterän asetus sekä karojen ja leikkuuterän yhdensuuntaisuus. Onnistunut sorvaus on lopulta kiinni sorvaajasta, vaikka eri prosessit on pyritty automatisoimaan. Sorvaajan keskeisiä tehtäviä ovat viilun laadun arvioiminen, vikojen korjaaminen ja koneiden sekä laitteiden kunnon tarkkailu. (Koponen 2002, s. 44.) Sorvi voidaan pääpiirteittäin jakaa seuraaviin osiin: Koneen runko, tartuntaelimet, teräkelkka, tukirullasto ja sorvin käyttölaitteet. Sorvauksessa ensin tarttuvat läpimitaltaan suuret karat pölliin ja alkavat pyörittää pölliä. Tämän jälkeen teräpenkki ja tukirullat lähestyvät pölli niin pitkään, että sorvausprosessi alkaa. Pöllin saavutettua tietty läpimitta, irtoavat suuret karat ja sisäiset pienemmät karat jäävät paikoilleen. Näin saadaan mahdollisimman paljon hyötykäytettyä puuainesta.

Leikkaavaterä on teräpalkissa kiinni. Vastaterä puristuu hieman puuta vasten, jolloin viilu joutuu puristumaan terän ja vastaterän välisestä raosta, jolloin syntyy tasaisempi viilupinta. Vastaterän vähentää viilun halkeilua ja viilun puristusasteella voidaan vaikuttaa viilun laatuun. Vastaterää ei käytetä pöllin pyöristyksen aikana. Vastaterä ja leikkaavaterä aiheuttavat pölliin voimia, jotka aiheuttavat puun etääntymisen terästä. Pöllin taipumista estetään tukirullastolla vastakkaiselta puolelta. (Koponen 2002, s. 40.) Yhden pöllin sorvaaminen kestää keskimäärin 24,6 sekuntia. Sorvausjakson eri vaiheet ovat esitetty kuvassa 17. Tämä on vain yhden koneen keskimääräinen ajoaika.




**Kuva 17.** Sorvauksen työvaiheet (Koponen 2002, s. 42).

Tämän työn sorvilaitteistossa karojen toiminta on muutettu. Alkuperäisen sorvauskoneen ongelmana oli karojen jarruttaminen sorvauksen jälkeen. Karat omaavat suuren massan, jolloin niiden kiihdytys ja jarrutus vaativat paljon energiaa. Tulevassa koneessa pyritään ratkaisemaan karojen kiihdytys ja jarrutusongelmat. Samalla pyritään vähentämään ylijäämä puun määrää.

## 3.2 Mallin muodostaminen

Koneen rakenteen ja toiminnan selvityksen jälkeen voidaan alkaa rakentaa mallia suunnittelemalla mallin rakenne ja keräämällä kappaleista tietoja. Mallia lähdetään luomaan hahmottamalla ensiksi tulevan mallin topologia. Topologian määrittelyssä koneen osat jaetaan liikkuviin kappaleisiin, ja määritellään näille kappaleille liitostavat. Mekaniikan topologia voidaan muodostaa kappaleiden niveltiedoilla. Kun tulevan mallin rakenne ja kappaleiden määrät ovat selvillä, voidaan kerätä tietoa kappaleiden tai kappalekokonaisuuksien massoista ja inertioista. Mevea-ohjelmistossa on myös kattava valikoima erilaisia voimalähteitä, joten erillistä voimayhtälöä moottoreista ja hydrauliikasta ei tarvitse muodostaa. Malliin lisätään voimakomponentti, joka toimii joko jousivaimenninsysteeminä tai komponentille asetetaan ulkopuolinen voimanlähde, kuten hydrauliikkasylinteri tai lineaarinen karamoottori.

MBS-mallin tuominen tuotekehitysryhmälle vaatii ryhmän perehtymistä monikappaledynamiikkaan. Useimmissa tapauksissa perehdyttäminen aloitetaan oman koneen malleilla. Tämä malli pitää olla rakennettu tuotekehitysryhmän kanssa, jotta ryhmä näkee mitä tietoja mallin rakentaminen vaatii.

Kun virtuaaliprototyypin käyttö on vakiintunut tuotekehityksessä, voidaan mallin rakentamisen eri vaiheet jakaa eri osa-alueille, kuten oikeassakin koneessa. Esimerkiksi mekaniikkamallintaja luo ensiksi pelkän monikappalemallin, mistä selviää mekaniikan toiminta. Osasto, jonka tehtävä on suunnitella koneen liikkeet, voi tämän jälkeen liittää malliin voimakomponentteja. Lopulta ohjausjärjestelmää kehittävä osasto liittää malliin oman ohjausjärjestelmänsä. Tämä kaikki vaatii ryhmältä paljon yhteistä kommunikointia.

Mevea-ohjelmistossa käytetään SI-perusyksiköitä, paitsi joitakin hydrauliikka komponentteja lukuun ottamatta. Koko ryhmän on hyvä tietää millä yksiköillä kommunikoidaan. Ryhmälle tulee olla heti alusta asti selvä, miten globaalikoordinaatisto asettuu laitteen suhteen. Tässä työssä globaalikoordinaatiston nollakohta on asetettu sorvikarojen keskelle. Koordinaatiston positiivinen X-akseli osoittaa vastakkaiseen puun kulkusuuntaan ja koordinaatiston positiivinen Z-akseli osoittaa vasemmalle, kun katsotaan porrasannostelijalta sorvauspenkkiin päin. Koordinaatiston positiivinen Y-akseli osoittaa ylöspäin. Nämä tiedot tulee olla kaikkien tiedossa heti alussa.

Tämän työn alussa tuotekehityksessä oli luotu kappaleista ensimmäiset CAD-mallit, joista muodostettiin kappalekokonaisuus. Myös alustavia voimakomponenttivalintoja oli tehty, jolloin pystyttiin testaamaan näiden komponenttien soveltuvuutta koneeseen. Laitteen rakenne ja kappaleiden muodot muuttuivat vielä työn tekovaiheessa, mutta malli haluttiin saada mahdollisimman pian valmiiksi, jolloin suunnittelijat pääsevät aikaisemmin testaamaan eri mekaanisia ratkaisuja.

Mallin topologista rakennetta pyrittiin yksinkertaistamaan ja optimoimaan, jotta simulaatiota olisi helpompi pitää reilaajassa. Topologian muodostuksessa mallin kappaleiden ja rajoitteiden pyritään pitämään vähälukuisina, jotta niiden laskenta ei veisi paljon laskentatehoa. Mevea-ohjelmassa osa kappaleista voidaan liittää malliin *dummy* kappaleina. Tällöin kappale ei kuulu kinemaattiseen ketjuun, vaan dummy kappaleen massa- ja inertiatiedot liitetään kappaleeseen, johon dummy on liitetty. Näitä kappaleita voivat olla esimerkiksi

hydrauliikkasylinteri ja -mäntä.

## 3.3 Kappaleet

Kappaleen 2.2 perusteella tarvittiin jokaisesta jäykästä kappaleesta seuraavat tiedot: massa, massakeskipisteen asema, inertia ja nivelasemat. Kuvassa 18 on esimerkki esitietotaulukosta, jonka taulukon tietojen perusteella voidaan muodostaa MBS-malli koneesta. Vaadittavat tiedot saadaan helposti nykyään CAD-ohjelmistoista. Tärkeintä on tietää minkä suhteen lasketaan inertiat. Tässä työssä inertiat laskettiin kappaleen paikalliskoordinaatiston suhteen. Jos inertia- ja massa-arvot eivät ole kohdallaan, malli saattaa hajota helposti liian suurista voimista. Tämä on syytä tarkistaa, jos malli käyttäytyy epätavallisesti. Myös kappaleen muut pisteet lasketaan kappaleen paikalliskoordinaatiston suhteen.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Kappale: | Linearfeeder_frame | | | | | | |
| ID: | 4 | Tiedosto: | | 7_4_linearfeeder | | | |
| Massa[kg] | Inertiat | [kgmm^2] | | | Massa keskipiste | | [mm] |
| 559,02 | Ixx, Ixy,Ixz | 1,31E+08 | -4,1E+07 | -25829,6 | x | y | z |
| | Iyy,Iyz | -4,1E+07 | 1,96E+08 | -66843,8 | 409,6328 | 285,2119 | 0,193979 |
| | Izz | -25829,6 | -66843,8 | 2,32E+08 | Massa keskipiste | | [m] |
| | Inertiat | [kgm^2] | | | x | y | z |
| | Ixx, Ixy,Ixz | 130,9866 | -41,0399 | -0,02583 | 0,409633 | 0,285212 | 0,000194 |
| | Iyy,Iyz | -41,0399 | 196,1437 | -0,06684 | | | |
| | Izz | -0,02583 | -0,06684 | 231,6442 | | | |

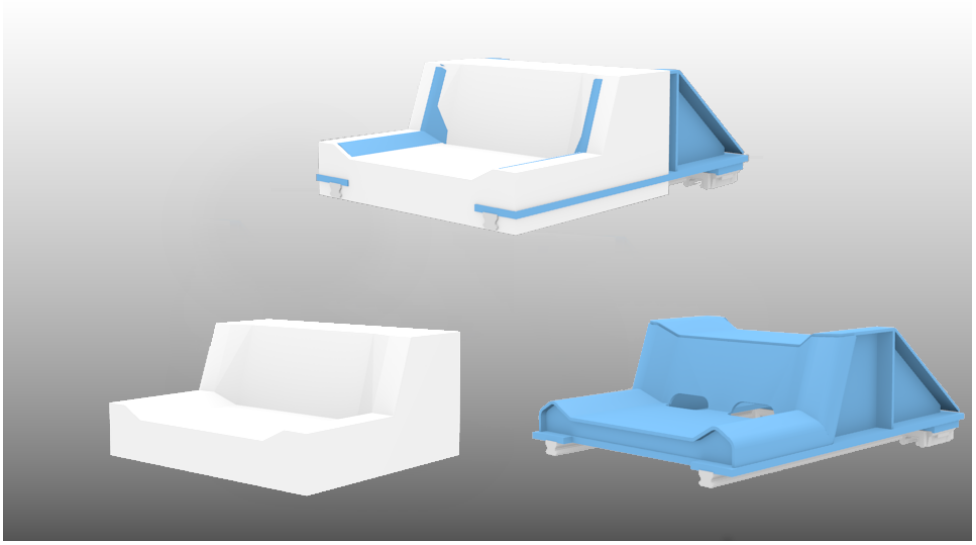
| Coordinate system | | [m] | | Orientaatio [rad] | | |
|---|---|---|---|---|---|---|
| | x | y | z | Phi | Theta | Psi |
| Carriage_sylinter | 0.7 | 0.316585 | 0 | 1.5707963 | 0.9599310 | 0 |

| LOC in Global coordinate | | | [m] | Orientaatio [rad] | | |
|---|---|---|---|---|---|---|
| | Global X | Global Y | Global Z | Phi | Theta | Psi |
| LOC | 1,31 | -0,415 | 0 | 0 | 0 | 0 |

**Kuva 18.** Jäykästä kappaleesta vaadittavat tiedot.

Järjestelmän laskentaa voidaan keventää mallintamalla osa kappaleista *Dummyina*. Dummyt ovat jäykkiä kappaleita, jotka toimivat staattisesti liitettyinä kappaleina. Nämä kappaleet voidaan lisätä muihin komponentteihin kuten jäykkiin kappaleisiin, renkaisiin ja *body-to-body* voimiin. Dummy-kappaleet kasvattavat isäntäkappaleen massaa, inertia-arvoja ja muuttavat isäntäkappaleen massakeskipistettä. Dummy-kappaleita käytetään esimerkiksi hydrauliikka sylinterinä ja kulkuvälineen renkaina. Menetelmä vähentää kappaleiden ja rajoiteyhtälöiden määrää merkittävästi ja mahdollistaa pienemmän aika-askeleen käytön. (Mevea solver reference Manual 2015.)

Kun kappalekomponentit oli luotu, kappaleille luotiin oma visuaali- ja kontaktigrafiikka. Kuvassa 19 vasemmalla alhaalla on havainnoitu kontaktigrafiikka ja oikealla alhaalla kappaleen visuaalinen grafiikka. Monikappadynamiikka-menetelmä ei huomio laskennassa kappaleen muotoja, muuten kuin inertia-tiedoilla, joten grafiikka ei vaikuta itse MBS-mallin laskentaan. Kontaktilaskennassa käytetään kontaktigrafiikka, mutta käyttäjälle näkyy vain visuaalinen grafiikka. Kontaktimallinnuksessa tulee määritellä mitkä kappaleet voivat olla kontaktissa toistensa kanssa.



**Kuva 19.** Kontaktigrafiikka ja visuaalinen grafiikka.

Ratkaisijan laskentaa kuormittaa myös kontaktigrafiikoiden ja visuaalisten grafiikoiden tarkkuus. Visuaalinen grafiikka sisältää kaikki yksityiskohdat kappaleesta, jotka voidaan havainnoida simulaatioissa. Kontaktigrafiikka on erittäin raaka ja yksinkertaistettu versio todellisesta kappaleesta. Mevea-ohjelmistossa kontaktivoimat lasketaan kappaleiden kontaktigrafiikoiden törmäyksestä. Kun tiedetään paljon grafiikat menevät sisäkkäin kontaktissa, voidaan sen perusteella laskea vastavoima jousivaimennus-menetelmällä. Kontakti voi sisältää myös kitkaa kahden kontaktigrafiikan välillä. Kontaktigrafiikan tulee olla erittäin yksinkertaistettu malli oikeasta kappaleesta, jotta laskenta-aika pysyy reaaliajassa. Itse kontaktigrafiikka ei näy itse simulaation aikana. (Mevea solver reference Manual 2015.)

Rauten mallissa oli yhteensä 34 jäykkää kappaletta. Koska kaikki kappaleet eivät ole kosketuksessa puun kanssa, tarvittiin vain 13 kappaleelle oma kontaktigrafiikka. Kontakti materaaliparametreihin asetettiin rakenneteräksen ominaisuudet. Näitä kuitenkin jouduttiin muokkaamaan, jotta kontaktit toimisivat realistisesti. Moottoreita ja kuularuuvien pyörimistä ei ole mallinnettu, vaan nämä ovat osana jäykkää kappaletta.

## 3.4 Topologia ja liitokset

Mallinnuksessa käytetään vain olennaisimpia mekanismin kappaleita, jotta MBS-mallista ei tule liian raskas ja samalla malli pysyy reaaliajassa. Kuten aikaisemmin todettiin, ovat kaikki kappaleet mallinnettu jäykkinä, joten erillinen joustavien kappaleiden mallinnus ei ole tarpeen.

Viilusorvin MBS-malli koostuu 34 jäykästä kappaleesta ja 34 rajoitteesta. Taulukossa 1 on

eritelty mallinnuksessa käytetyt kappaleet, niiden väliset rajoitteet ja kunkin rajoitteen määrä. Mallissa on käytetty kahta liikkuvaa rajoitetyyppiä: kierto- ja translaationivel. Molemmat nivelet jättävät yhden vapausasteen vapaaksi, vapausasteiden määrää tulee kiinnittää huomiota, jotta kappaleesta ei tule ylirajoittunut.

*Taulukko 1. Viilusorin kappaleet, nivlet ja rajoitteiden määrä mallissa.*

| Nro. | Nivel | Niveltyyppi | Kappale I | Kappale J | RA |
|---|---|---|---|---|---|
| 1 | G - Stepfeeder_frame | Jäykkä | Maa | Porrasannostelijan runko | 6 |
| 2 & 3 | Block_even_ender | translaatio | Porrasannostelijan runko | Pöllin tasain | 5 |
| 4 | Stepfeeder_carriage | translaatio | Porrasannostelijan runko | Vaunu | 5 |
| 5 | G-Linearfeeder_frame | jäykkä | Maa | Lineaarisyötin | 6 |
| 6 | Linear_carriage | translaatio | Lineaarisyötin | Vaunu | 5 |
| 7 | G-Charger_frame | jäykkä | Maa | Keskittäjän runko | 6 |
| 8 & 9 | Charger_x-carriage | translaatio | Keskittäjän runko | X-kelkka | 5 |
| 10 & 11 | Charger_y-carriage | translaatio | X-kelkka | Y-kelkka | 5 |
| 12 & 13 | Charger_shaft | kierto | X-kelkka | Kara-akseli | 5 |
| 14 & 15 | Charger_spindle | translaatio | Kara-akseli | Kara | 5 |
| 16 & 17 | ArmSlider | translaatio | Keskittäjän runko | Liukuvarsi | 5 |
| 18 & 19 | Clamp | translaatio | Liukuvarsi | Varsi | 5 |
| 20 | G-Lathe_frame | jäykkä | Maa | Sorvin runko | 6 |
| 21 & 22 | Lathe_shaft | kierto | Sorvin runko | Kara-akseli | 5 |
| 23 & 24 | Spindle | translaatio | Kara-akseli | Kara | 5 |
| 25 & 26 | Trunnion_bearing | translaation | Sorvin runko | Kaarilaakerit | 5 |
| 27 | Kinfe_bar | kierto | Kaarilaakerit | Teräpalkki | 5 |
| 28 | Round_bar_carriage | translaatio | Teräpalkki | Vastateräpalkki | 5 |
| 29 | Round_bar | kierto | Vastateräpalkki | Pyörivä vastaterä | 5 |
| 30 | Back_up_frame | translaatio | Sorvin runko | Tukilaitteen runko | 5 |
| 31 & 32 | Back_up_roll | kierto | Tukilaitteen runko | Tela | 5 |

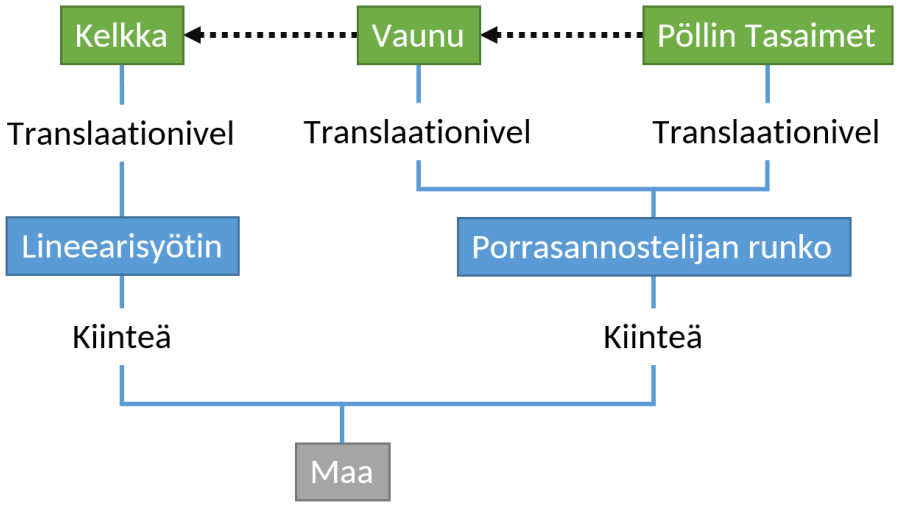
*Taulukko 1 jatkuu. Viilusorin kappaleet, nivlet ja rajoitteiden määrä mallissa.*

| Nro. | Nivel | Niveltyyppi | Kappale I | Kappale J | RA |
|---|---|---|---|---|---|
| 33 | Back_up_doser | kierto | Tukilaitteen runko | Pyöritetyn pöllin annosteilja | 5 |
| 34 | Back_up_catcher | translaatio | Tukilaitteen runko | Pyöritetyn pöllin kaappari | 5 |
| Rajoitteiden määrä on 174 | | | | | |

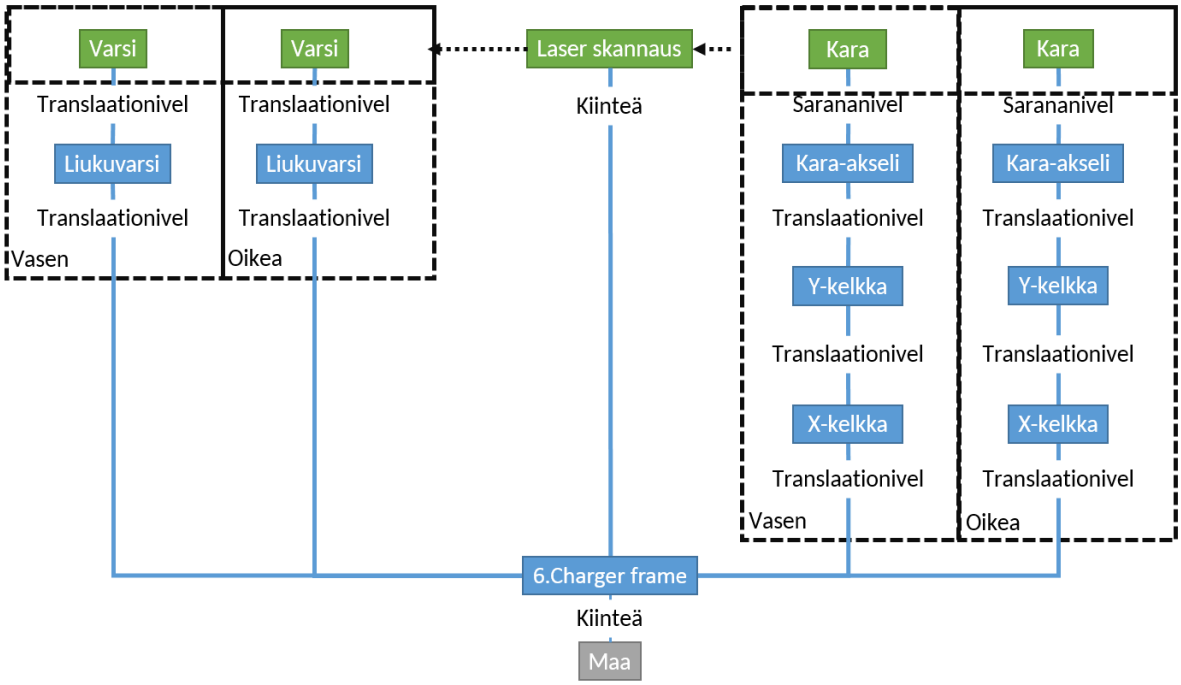
Taulukosta 1 nähdään, että monikappalejärjestelmässä on 174 rajoiteyhtälöä. Kutzbachin kaavan 3 avulla voidaan laskea mekanismin vapausasteiden määrän:

$$DOF = 6 \times 34 - 6 \times 4 - 5 \times 30 = 30 \tag{14}$$

Malli on tasapainossa, koska mallissa on 30 liikkuvaa kappaletta ja rajoitteita on 30. Mallin kappalekokonaisuutta ja toimivuutta pystytään havainnoimaan mekanismin topologia kuvilla. Sorvi on jaettu eri osakokonaisuuksiin, joiden topologia on esitetty kuvissa 20 ja 21. Kuvassa neliöt ovat kappaleita, ja siniset viivat esittävät liitoksia kappaleiden välissä. Vihreät neliöt ovat kappaleita, jotka ovat kontaktissa puun kanssa. Puun kulku on esitetty katkoviiva nuolilla. Topologian kuvilla pystytään hahmottamaan MBS-mallin rakenne ja samalla nähdään mitkä kappaleet ovat kontaktissa puun kanssa, jolloin tiedetään mille kappaleille tulee luoda kontaktigrafiikka. Topologia kuvista havaittiin, että sorvin rungon ja teräpalkin väliin syntyy suljettu ketju. Jotta kinemaattinen ketjua pystytään laskemaan, tulee suljettuun ketjuun asettaa katkaisunivel. Tässä tapauksessa katkaisunivel on asetettu oikean kaarilaakerin ja teräpalkin väliin.




**Kuva 20.** Lineaarisyöttimen ja porrasannostelijan topologia.

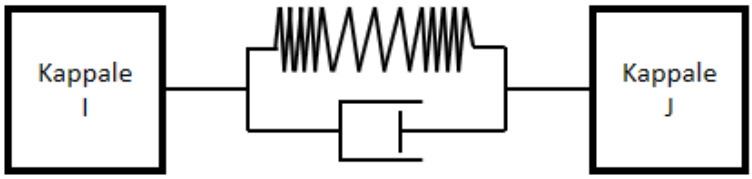



**Kuva 21.** Keskittäjän topologia.

## 3.5 Voimat

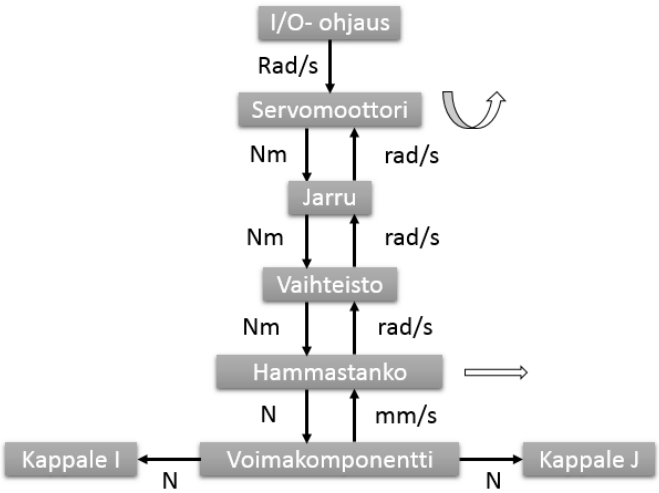
Jokaisen sähkömoottorin, hydrauliikka- ja pneumatiikkasylinterin paikalle luotiin malliin ulkopuolinen voima- tai vääntökomponentti, johon yhdistettiin tarvittavat voimalähteet. Voimakomponentti on jousivaimennuskomponentti, joka sellaisenaan toimisi järjestelmässä jousen ja vaimentimen yhdistelmänä kuten kuvassa 22 on esitetty. Komponentti sisältää myös jousen päätyvaimennukset, jonka tehtävänä on estää jousen siirtymästä läpi päädyn.

Voimakomponentista voidaan poistaa sisäiset ominaisuudet, ja korvata ne eri voimalähteillä. Voimakomponentilla määritellään liikkeen liikerata ja kiinnityspisteet. Voimakomponentille asetetaan minimi- ja maksimipituus, minkä jälkeen päätyvaimennus vaimentaa kappaleen liikkeen. Voimakomponentin minimipituus ei saa olla liian lyhyt tai nolla, koska kappale saattaa suuressa nopeudessa vaatia matkaa pysähtymiseen. Jos jousen minimipituus on 0, saattaa kappale mennä päätypisteestä läpi, minkä jälkeen matemaattiseen malliin syntyy negatiivinen voima. Useammissa tapauksissa jousen minipituudeksi asetettiin 10 mm, jolloin vältyttiin jousen kääntymästä ympäri.




**Kuva 22.** Jousivaimenninkomponentin kuvas.

Suurin osa viilusorvin voimakomponenteista on lineaarisia karamoottoreita. Virtuaaliprototyypin ja fyysisen prototyypin lineaariset karamoottorit ovat rakennettu eri tavalla, mutta komponenttien toiminta ja ohjaustapa on samanlainen. Kuvassa 23 on esitetty karamoottorin rakenne viilusimulaattorissa.




**Kuva 23.** Meveassa käytetty mallinustapa kuularuuvimoottoreille.

Kuvan mustat nuolet esittävät komponenttien välisiä rajapintoja ja valkoiset nuolet esittävät voiman liikemuotoa. Moottorin ohjausta varten luodaan uusi analoginen ohjauskanava, jolla ohjataan moottorin pyörimisnopeutta ($rad/s$). Moottorin pyörimisnopeus saadaan lineaariseksi liittämällä *Rack and pinion* komponentti moottoriin. Rack and pinion -komponentti koostuu hammaspyörästä ja hammastangosta, jossa yhden kierroksen aikana hammastankoa siirretään hammaspyörän kehäpituuden verran. Fyysisessä moottorissa moottorin pyörimisnopeus muutetaan lineaariseksi kuula- tai rullaruuvilla. Virtuaalisessa mallissa kuularuuvin kierteen nousu muutetaan hammaspyörän kehäpituudeksi. Näin nopeudet vastaavat kuularuuvin tuottamaa nopeutta. Hammaspyörän tarvittava halkaisija saadaan laskettua seuraavalla kaavalla:

$$r_{screw} = \frac{P_{screw} \times 10^{-3}}{2\pi} \tag{15}$$

jossa $P_{screw}$ on kuularuuvin nousu millimetreissä ja $r_{screw}$ on hammaspyörän säde metreissä. Hammastangon liikevoima ohjataan translaatiovoimakomponentille, mikä on kiinnitettynä kahteen eri kappaleeseen. Kaikissa yhdistelmissä ei ole vaihteistoa välissä, vaan itse kuularuuvi toimii vaihteena. Pyöriviin liikkeisiin sähkömoottori on liitetty samalla tavalla kuin kuvassa 23 on esitetty, mutta ilman hammastankoa. Sähkömoottorin nopeus muutetaan vaihteen avulla sopivaksi, jolloin saadaan esimerkiksi kara-akseli pyörimään halutulla nopeudella. Mevea-ohjelmistossa voimansiirtokomponentit muodostavat ketjun, jonka mihin tahansa väliin voi asettaa eri voimasiirtokomponentteja. Ketjun alkupäänä toimivat voimalähteet, kuten moottori, ja ketjun päädyssä on voimakomponentti, joka välittää voiman määritettyihin pisteisiin. Tämän työn mallissa luotiin 25 eri voimaketjua.

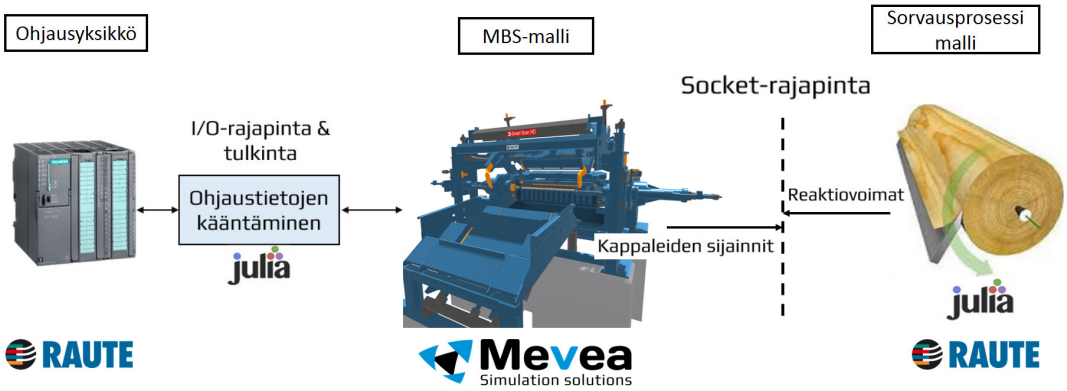
Hydrauliikkajärjestelmän suunnittelu oli vielä keskeneräinen diplomityön aikana, joten hydrauliikkapiirit mallinnettiin ideaalisoiduin mallin mukaan. Tässä työssä käytetty idealisoitu hydrauliikkapiiri on esitetty kuvassa 11. Piirissä pumppu on mallinnettu vakiopainelähteenä, jonka painetta ohjataan 4/3-suuntaventtiilillä. Suuntaventtiiliä ohjataan ±10 V jännitteellä. Sylinterit ovat mallinnettu kaksikammioisina, mikä tasapainoittaa sylintereiden hallintaa.

Pneumatiikkapiiri on mallinnettu pienpaineisena hydrauliikkajärjestelmänä, jossa on 6 bar:n paine. Tämä ratkaisu on riittävä tähän mallin, koska painelaitteilla ei ole suuria toiminnollisia vaikutuksia tai niitä ei välttämättä edes ohjata.

## 3.6 Rajapintojen luominen malliin

Mevea-ohjelmisto mahdollistaa erilaisten rajapintojen liittämisen virtuaaliprototyyppiin. Malliin voidaan liittää ulkopuolisia signaalilähteitä ja mallista pystytään lukemaan määrättyjä signaaleja. Mevea modeller -ohjelmasta löytyy valmiiksi luotuja rajapintoja Simulinkille, Matlabille, Scilabille ja sokettiyhteydelle. Näissä tapauksissa Mevean ohjelmisto on suoraan yhteydessä ulkopuoliseen ohjelmistoon yhteisen rajapinnan kautta. Toinen tapa on käyttää Mevean I/O toolbox -työkalua, jonka avulla voidaan muodostaa yhteyksiä erilaisiin väylätyyppeihin kuten CAN-väylään tai soketti porttiin. Mevea I/O toolbox on erillinen ohjelma, josta voidaan lukea signaaleja ja kirjoittaa signaaleja. Ohjelman avulla ulkopuoliset signaalit ohjataan mallinohjaukseen. Signaaleja voidaan lukea ja kirjoittaa myös muilla ohjelmointikielillä käyttäen Mevean C++ -kirjastoa.

Tämän työn virtuaaliprototyyppiin luotiin kaksi eri rajapintaa: toinen sorvausprosessimallille ja toinen ohjausjärjestelmälle. Rajapintojen käyttö on esitetty kuvassa 24.




**Kuva 24.** Luodut rajapinnat simulaatiomallissa.

Tämän työn tapauksessa simulointimalliin halutaan liittää ulkopuolinen puuprosessimalli, joka mallintaa puun käyttäytymistä viilun sorvauksessa. Prosessimallille lähetetään mallista terän, karojen ja puun asema- ja orientaatiotiedot. Prosessimalli lähettää vastaavasti terään kohdistuvat vastareaktiovoimat ja karojen pyörimistä vastustava vääntö. Molempien mallien tulee edetä samassa ajassa, jolloin mallien välille ei synny vääristymää. Prosessimallin eristäminen omaksi ohjelmaksi mahdollistaa molempien mallien erillään kehittämisen. Kun rajapinnat ovat tarkoin määritelty, voidaan jatkaa molempien mallien kehitystyötä vaikuttamatta toisen mallin sisältöön.

Sorvausprosessimallin täytyi olla synkronoitu sorvisimulaattorin kanssa, jolloin siihen soveltui parhaiten tietokoneiden väliseen kommunikointiin tarkoitettu TCP/IP-yhteys. Ohjausjärjestelmä

lähettää 10 ms:in välein ohjausarvoja ja ohjausjärjestelmä tarkkailee kappaleiden sijaintia, jonka perusteella ohjausjärjestelmä ohjaa kappaleita. Tämän takia synkronoitua yhteyttä ei tarvita, vaan yhteyden muodostamiseen käytettiin Mevea I/O toolbox -työkalua. Sillä muodostettiin yhteys heidän ohjausjärjestelmään sokettiyhteyden kautta, jonka käskyt kirjoitettiin Mevean I/O-pooliin. Siitä moottoriohjausjärjestelmä luki arvot ja kirjoitti uudet arvot eri kanaville, josta virtuaaliprototyyppi luki ne.

### 3.6.1 Sorvausprosessimallin liittäminen simulaattoriin

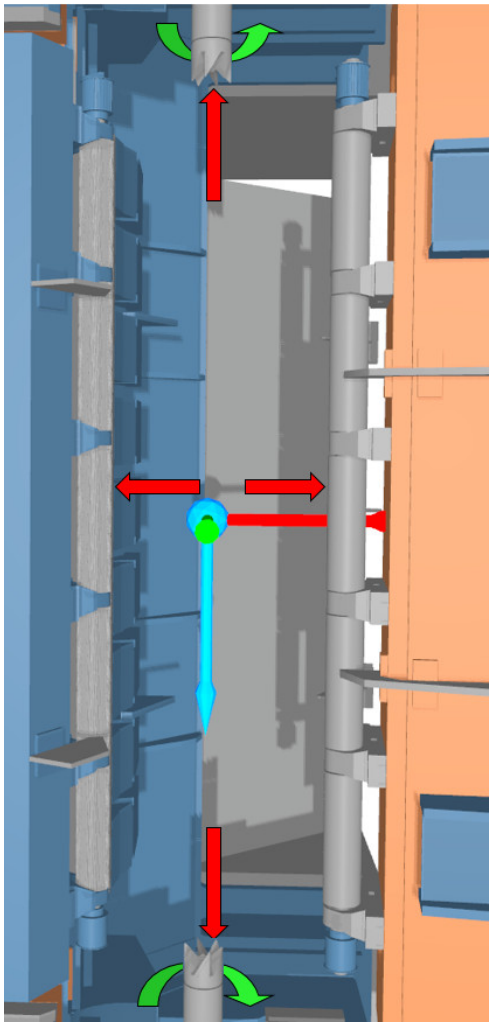
Sorvausprosessimalli haluttiin alusta lähtien pitää erillisenä ohjelmana simulaattorista, jolloin prosessimallia voidaan kehittää omana projektina, ja se voidaan aina yhdistää malliin rajapinnan kautta. Parhaassa tapauksessa samaa prosessimallia voidaan soveltaa myös tulevien sorvauskoneiden simuloinnissa, käyttämällä samoja rajapintamääritelmiä.

Sorvausprosessimallille luotiin oma sokettiyhteys, mikä käyttää TCP/IP-protokollaa. Yhteydessä määritellään molemmille osapuolille omat IP-osoitteet (Internet Protocol, IP) ja portti, jota kautta kommunikointi tapahtuu. Tässä tapauksessa prosessimalli ja mekaniikkamalli on samalla tietokoneella, jolloin käytetään tietokoneen kotiosoitetta *127.0.0.1* ja valitaan vapaana oleva portti, kuten *5111*. TCP/IP-yhteys mahdollistaa myös mallien ajamisen eri tietokoneilla, käyttäen samaa porttia ja tietokoneiden omia IP-osoitteita. Virtuaaliprototyyppiin luodaan sokettiyhteys lisäämällä Mevea modeller -ohjelmassa sokettiyhteys-komponentti, jossa määritellään yhteyden kautta tulevat ja ulos menevät signaalit.

Julia-ohjelmointikielestä löytyy sisään rakennetut funktiot sokettiyhteyden muodostamista varten, mutta yhteyden luomista varten tuli tietää miten Mevea solver-ohjelma avaa yhteyden. Solver-ohjelma tarkistaa ensin onko molemmilla ohjelmilla sama määrä signaaleja määriteltyinä, jonka jälkeen se avaa kommunikoinnin. Liitteessä I on Julia-ohjelmointikielellä toteutettu yhteys Mevean solver:iin. Liiteen koodissa solveri lähettää 23 eri signaalia, eikä Julia lähetä mitään signaaleja takaisin. Täällä tarkastusmenetelmällä pystytään luomaan varmistettu tcp/ip-yhteys ohjelmien välillä. Protokollan ansioista ohjelmat tarkastavat jatkuvasti tuliko oikea määrä signaaleja perille.

Rajapinnan määrittelyä varten käytiin keskustelua sorvausprosessimallin tekijän kanssa, mitkä olivat ne arvot, jotka mekaniikkamallista tarvitaan ja minkälaisia arvoja sorvausprosessimalli tulee lähettämään. Rajapinnasta pyrittiin luomaan mahdollisimman yksinkertainen ja selkeä,

jolloin saman rajapinnan luominen onnistuu myös muille mekaniikkamalleille. Kuvassa 25 on havainnoitu mitä voimalähteitä lähetetään simulaattorille. Kuvassa punaiset nuolet ovat translaatiovoimia ja vihreät nuolet ovat vääntövoimia. Translaatiovoimat lähetetään *newton* yksikkönä ja vääntömomentti lähetetään *newtonmetri* yksikkönä. Rajapinta on määritelty tarkemmin liitteessä II. Koska kaikki mekaniikan kappaleet mallinnettiin jäykkinä, voidaan sorvauksen kontaktivoima asettaa pistevoimina. Voimakomponenttien määrää voidaan muuttaa tai asettaa uudelleen, kun malli muuttuu. Tämä vaatii samalla rajapinnan uudelleen määrittelyn.



**Kuva 25.** Prosessimallin lähettämät voimalähteet.

Tämän työn tapauksessa simulointimalliin halutaan liittää ulkopuolinen puuprosessimalli, joka kuvaa puun käyttäytymistä viilun sorvauksessa. Prosessimallille lähetetään mallista terän, karojen ja puun asema- ja orientaatiotiedot. Prosessimalli lähettää vastaavasti terään kohdistuvat vastareaktiovoimat ja karojen pyörimistä vastustava vääntö. Molempien mallien tulee edetä samassa ajassa, jolloin ei synny ylimääräisiä voimia vääristyneestä aika-askeleesta.

Prosessimallin eristäminen omaksi ohjelmaksi mahdollistaa molempien mallien erillään kehittämisen. Kun rajapinnat ovat tarkoin määritelty, voidaan jatkaa molempien mallien kehitystyötä vaikuttamatta toisen mallin sisältöön. Tämän yhteystyypin mahdollisuus ja haastavan prosessin nopea ohjelmointi puoltavat Julia-ohjelmointikielen valintaa.

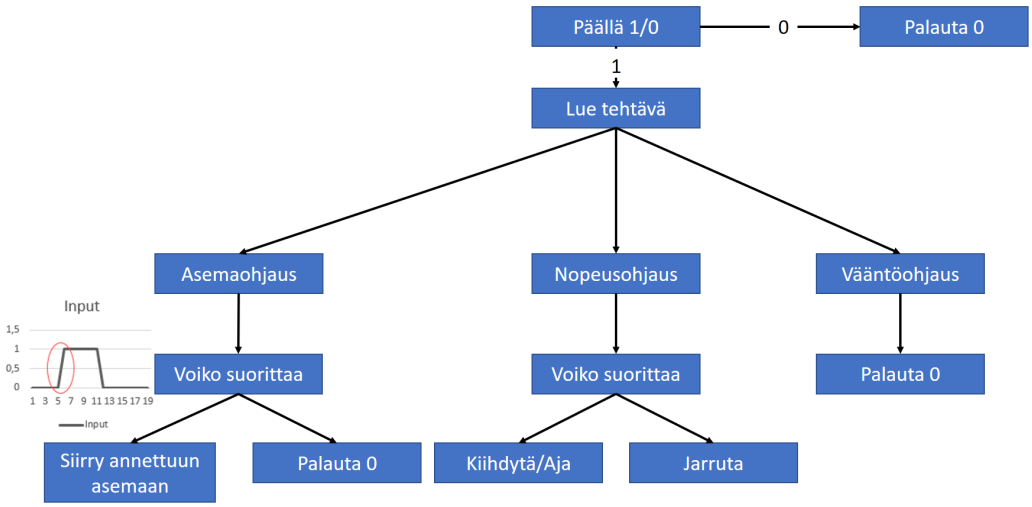
### 3.6.2 Ohjausyksikön liittäminen simulaattoriin

Virtuaaliprototyypin haluttiin ohjata Rauten logiikkaohjelmalla, jolloin pystytään testaamaan ja kehittämään viilusorvin ohjausjärjestelmää. Rauten tulevassa viilusorvissa suurin osa liikkeistä on toteutettu sähkömoottoreilla, jotka ovat asema-, nopeus- ja vääntöohjattavia. Tämä tarkoittaa sitä, että moottorille lähetetään vain tieto halutusta asemasta, nopeudesta tai väännöstä. Moottorin sulautettu ohjausjärjestelmä ohjaa moottoria siten, että annettu asema-, nopeus- tai vääntöarvo saavutetaan.

Koska kyseessä on moottoreiden sulautettu järjestelmä, jonka toiminta perustuu valmistajan tekemiin säätöihin ja ohjelmointiin, ei voitu simuloida täysin samanlaista käyttäytymistä moottorille kuin oikeassa moottorissa. Logiikkaohjelman tulkintaan etsittiin avoimen lähdekoodin ohjelmistoja, mutta monet avoimen lähdekoodin ohjelmista olivat tehty askelmoottoreille, eikä avoimen lähdekoodin ohjelmia oltu suunniteltu näillä syötteillä ja ohjauksilla. Päädyttiin siihen tulokseen, että oman ohjelman luominen vie vähemmän aikaa kuin valmiin ohjelma muuttaminen sopivaksi. Moottoriohjauksen tehtävänä oli tulkita ohjausyksikön lähettämät käskyt ja ohjata tämän perusteella moottoria. Ohjausjärjestelmän ohjelmointi vei projektissa eniten aikaa, mutta ilman sitä, mallin hyödyt olisivat jääneet paljon vähemmälle.

Ohjelman rakentaminen aloitettiin tiedustelemalla Rautelta, minkälaisia moottoreita heillä oli suunnitteilla käyttää ja miten niitä tullaan ohjamaan. Näiden tietojen avulla pystyttiin moottoriohjausjärjestelmästä tekemään yhteensopiva Rauten ohjausjärjestelmän kanssa. Heidän tulevassa laitteessa moottoreita voidaan ohjata joko asema-, nopeus- tai vääntöohjauksella. Moottoreita ei varsinaisesti ohjata, vaan niille lähetetään tavoitearvoja. Lineaarinen karamoottorin sulatettu järjestelmä mittaa jatkuvasti missä asemassa kuljetusruuvi on ja ohjaa sähkömoottorin toimintaa sen perusteella. Kuitenkin Mevea-ohjelmistossa moottoreita ohjataan joko kierrosnopeuden tai väännön mukaan analogisella signaalilla. Moottoreiden ohjaustapoja ei voitu vaihtaa kesken simuloinnin, jolloin jompikumpi ohjaustapa tuli valita. Päädyttiin ohjamaan moottoreita pyörimisnopeudella, koska sillä saadaan moottoreista asema- ja nopeusohjeisia.

Moottorin ohjauksesta saatujen tietojen perusteella voidaan muodostaa kuvan 26 mukainen lohkokaavio. Moottorin sulautetun ohjausjärjestelmän logiikka tiedustelee ensin, ollaanko moottoria ohjaamassa nopeuden vai aseman mukaan. Asemaohjauksessa, ohjausyksikkö lähettää moottoriohjaukselle tiedon halutusta kuljetusruuvin asemasta ja minkälaisen nopeusprofiilin mukaan kuljetusruuvin kulkee. Moottorin ohjausjärjestelmä tarkkailee kuularuuvin asemaa ja nopeutta, minkä perusteella ohjelma ohjaa moottorin pyörimisnopeutta. Nopeusohjauksessa ohjausyksikkö lähettää moottoriohjaukselle tavoitenopeuden ja sallitut nopeudenmuutokset. Moottori pyrkii saavuttamaan tämän nopeuden ja pitämään sen, kunnes ohjausjärjestelmästä tulee käsky pysähtyä. Nopeusohjauksessa moottori pysäytetään manuaalisesti. Asemaohjauksessa ohjausjärjestelmä laskee jatkuvasti tarvittavan jarrutusmatkan ja päättää sen mukaan, milloin aloittaa hidastamisen. Näin tavoite nopeus saavutetaan pehmeästi ja nopeusprofiilia noudattaen.

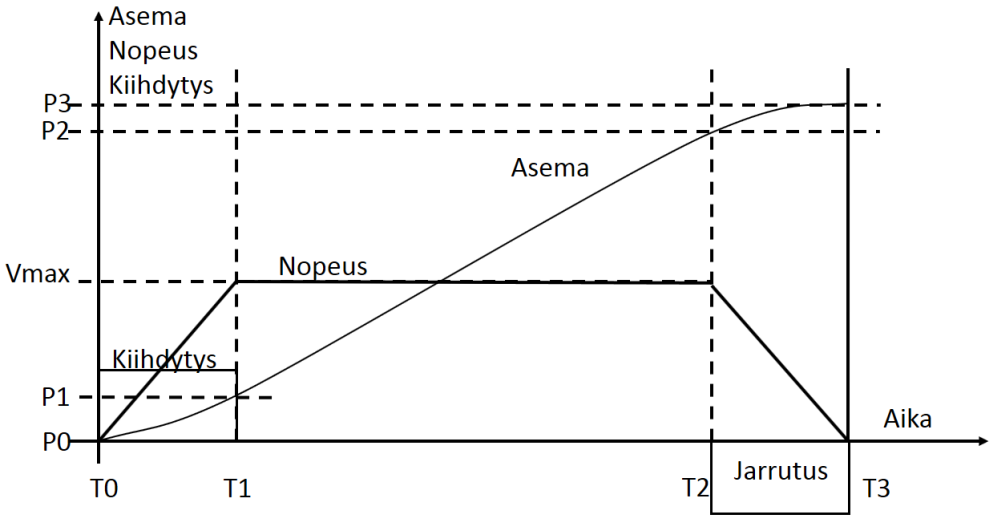



**Kuva 26.** Moottorin sulautetunohjausjärjestelmän lohkokaavio

Kuvasta voidaan havainnoida, että ohjausjärjestelmä luki uuden asema-arvon vain, kun execute signaalin oli nollattu. Lisäksi asemaohjauksen ja nopeusohjauksen tilaa piti saada vaihtaa kesken ajon, mikä loi lisää haasteita ohjelman luomisessa.

Asemaohjauksessa moottorin kiihtyvyys, nopeus ja asemanmuutos voidaan havainnoida paremmin kuvasta 27. Kuvasta voidaan havainnoida nopeusprofiili ja sen eri vaiheet. Nopeusprofiilia ei lasketa valmiiksi jokaiselle matkalle, vaan se toimii ohjeena matkan eri vaiheissa. Ohjausjärjestelmä tarkkailee jatkuvasti nykyisen nopeuden ja annetun

jarrutusnopeuden perusteella vaadittavaa jarrutusmatkaa. Jos jäljellä oleva matka on yhtä suuri kuin vaadittava jarrutusmatka, ohjausjärjestelmä ohjaa hidastamaan moottoria. Jos matkaa on enemmän, pyrkii ohjajärjestelmä joko kiihdyttämään tavoitenopeuteen $V_{max}$ tai pitämään moottori tavoitenopeudessa. Kuvassa kiihdytys tapahtuu P0 ja P1 välillä, ajo tapahtuu P1 ja P2 välillä ja jarrutus tapahtuu P2 ja P3 välillä. Tästä saadaan matkan ja nopeuden suhteinen nopeusprofiili, johon lisätään PID-säätimen (Proportional-integral-derivativ, PID) ohjauslisä. Ilman PID-säädintä moottori saattoi poiketa tavoitenopeudesta, jolloin moottorin tavoiteasemaa ei saavutettu. Lisäksi moottoriohjauksessa tuli ottaa huomioon minkälainen toleranssi oli sallittu esimerkiksi tavoiteaseman tai -nopeuden saavuttamisessa. Liian korkea toleranssi saattoi jättää moottorin ohjauskierteeseen. Jos esimerkiksi tavoitekiihtyvyys oli asetettu suureksi ja toleranssi korkealle, saattoi laite kiihdyttää ohi tavoiteaseman, jolloin ohjausjärjestelmä jäi kierteeseen.




**Kuva 27.** Karan asema-, nopeus- ja nopeusmuutosprofiili.

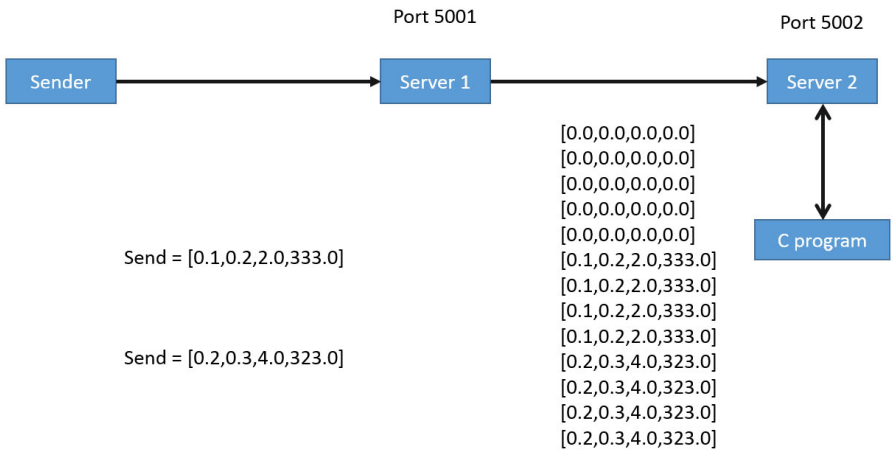
Moottorin tuli pyrkiä noudattamaan nopeusprofiilia myös kuormituksessa. Tätä varten moottoriohjakseen lisättiin PID-säädin. PID-säätimestä ei voida tehdä suoraan nopeussäätäjää, koska kuularuuvin tuli noudattaa annettua kiihdytystä ja jarrutusta. PID-säädin ei suoraan voi verrata nopeuden poikkeamaa, koska kiihdytyksessä ja jarrutuksessa tavoitenopeus on jatkuvassa muutoksessa, jolloin PID-säätimen korjaus ei toimi halutulla tavalla. Yksi vaihtoehto on purkaa nopeusprofiili osiin, jolloin kiihdyksissä ja jarrutuksissa PID-säädin pyrkii pitämään halutun nopeudenmuutoksen, ja tasaisessa nopeudessa PID-säädin pyrkii pitämään halutun nopeuden. Tämä vaatisi kaksi erillistä PID-säädintä, ja lisäksi tämän toteuttaminen nopeusohjaukselle olisi turhan monimutkainen.

Ratkaisu johon lopulta päädyttiin, oli muodostaa matkasuhteinen nopeusprofiili, johon lisättiin PID-säätimen laskema tavoitelisä. Matemaattisesti ratkaisu on ilmaistu yhtälössä 16.

$$v_n^{real} = v_n^{ref}(x_n^{real}) + PID(v_{n-1}^{real}, xd_n^{real}) \tag{16}$$

Yhtälössä $x_n^{real}$ on kuularuuvin asema edellisessä aika-askeleessa, jonka mukaan nopeusprofiilista $v_n^{ref}$ lasketaan ohjausnopeus. PID-säädin $PID$ vertaa edellistä tavoitenopeutta $v_{n-1}^{real}$ nykyiseen todelliseen nopeuteen $xd_n^{real}$ ja se lisätään profiilinopeuteen. Yhtälön tuloksena syntyy haluttu ohjausruuvin nopeus $v_n^{real}$.

Jotta moottoreiden ohjausjärjestelmän toiminta pystyttiin testaamaan, tehtiin Julia-ohjelmointikielellä Rauten ohjausyksiköstä yksinkertaistettu versio annettujen tietojen perusteella. Ensimmäinen yhteys tapa moottoriohjauksen ja ohjausyksikön välillä oli sokettiyhteys. Koska sokettiyhteys on synkronisessa tilassa virtuaaliprototypin aika-askeleen kanssa, täytyi ohjausyksikön lähettää samalla aika-askeleella arvoja virtuaaliprototyypille. Joten toteutettiin malli, jossa ohjelma A (ohjausjärjestelmä) lähettää manuaalisesti lähetetyt käskyt ohjelma B:lle. Ohjelma B lähettää jatkuvasti samalla aika-askellelle moottoriohjaukselle arvoja, joita se muuttaa kun ohjelma A:lta tulee uusi käsky. Tämä on esitetty kuvassa 28.

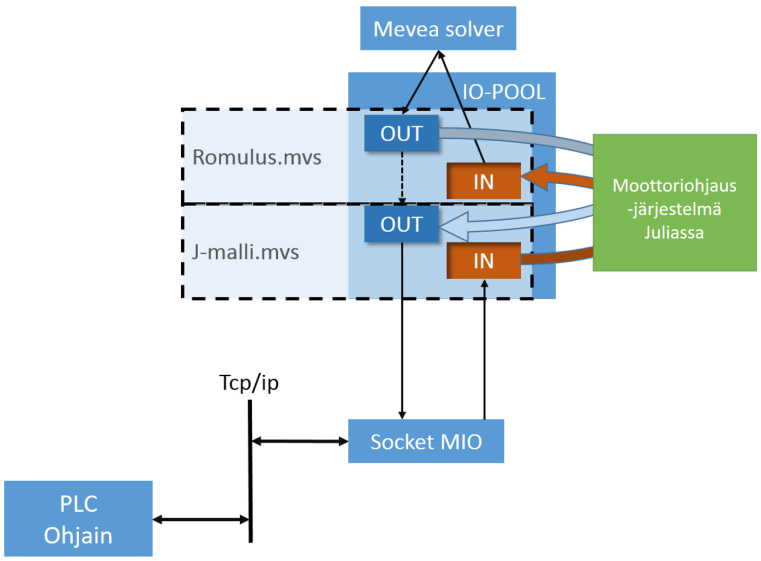



**Kuva 28.** Ohjausyksikön simulointi kolmella eri ohjelmalla.

Kuvassa ohjelma A (Sender), lähettää käskyjä ohjelma B:lle (Server 1). Tämä on jatkuvassa yhteydessä moottoriohjaukseen (Server 2), joka lähettää ohjaus-signaaleja mallille (C-program). Tämä ratkaisu osoittautui turhan raskaaksi ohjelmakokonaisuudeksi, eikä oikea ohjausyksikkö

lähettänyt dataa samalla aika-askeleella, kun sorvisimulaattori.

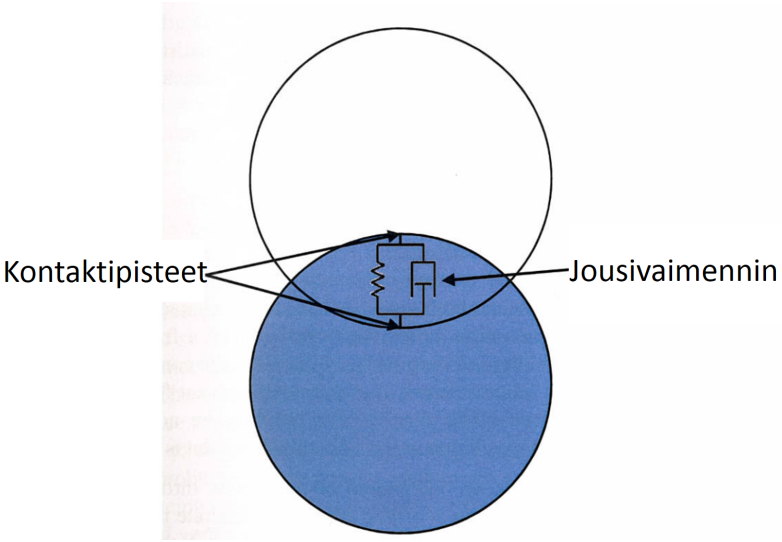
Toinen vaihtoehto oli käyttää Mevean I/O-poolia johon signaaleja voitiin kirjoittaa ja lukea milloin vain. Menetelmässä ohjausyksikkö lähettää sokettiyhteyden kautta käskyn tietokoneelle, jossa se ohjataan Mevea toolbox -työkalun avulla oikeaan I/O-poolin kanavaan. Moottoriohjaus lukee tiedot määritetystä kanavasta ja laskee tarvittavat kierrosnopeudet ja kirjoittaa ne eri kanavalle I/O-pooliin, josta virtuaaliprototyyppi lukee ne. Tämä prosessi on havainnoitu kuvassa 29. Kuvasta nähdään kuinka I/O-pooli on jaettu eri lohkoihin kullekin ohjelmalle. Tämä tapa osoittautui paremmaksi ja joustavammaksi kuin sokettiyhteys.




**Kuva 29.** Lopullinen moottoriohjauksen toiminta

## 3.7 Kontaktivoimat

Mevean kontaktimallissa ensiksi määritellään kappaleet, jotka ovat keskenään kontaktissa. Näille kappaleille luodaan oma kontaktigrafiikka, joiden geometrioiden avulla lasketaan, ovatko kappaleet kontaktissa ja kuinka paljon. Näiden tietojen perusteella ja Newtonin kolmanteen lakiin vedoten lasketaan molempiin kappaleisiin vaikuttava voima ja vastavoima. Meveassa kontaktivoima on toteutettu jousivaimennus menetelmällä kuten kuvassa 30 on esitetty.




**Kuva 30.** Kontaktimallinnus (muokattu Moisio 2013, s. 33).

Tämän toiminta perustuu siihen, kuinka paljon kappale uppoaa toiseen kappaleseen. Mitä enemmän kappaleet ovat sisäkkäin, sitä pidemmäksi jousi venyy ja sitä enemmän vastavoimaa syntyy. Simulaation suuri aika-askel voi aiheuttaa suuria kontaktivoimia, koska laskenta perustuu aika-askel menetelmään. Jokaisen aika-askeleen aikana tarkistetaan kappaleiden ja pintojen välinen etäisyys. Suuria kontaktivoimia voi aiheutua, jos kappale on yhden aika-askeleen aikana kerennyt edetä liian paljon toisen kappaleen sisälle kontaktipisteestä. Kontaktivoimien kannalta olisi aina parasta saada mahdollisimman alhainen aika-askel. Täysin realistista kontaktia ei voida mallintaa, koska laskennalla on rajattu laskentakapasiteetti ja jos aika-askel on liian pieni, simulaatio ei pysy enää reaaliajassa. Aika-askeleen määrittäminen asettaa myös haasteita kontaktikappaleiden materiaaliarvojen määrittämiselle.

Materiaalien fyysisiä ominaisuusarvoja ei voida suoraan käyttää mallinnusarvoina, vaan niistä voidaan saada suuntaa antavaa tietoa. Materiaalien jousiarvoja säätämällä pyritään saamaan materiaalin käyttäytyminen mahdollisimman todelliseksi. Materiaaleja pyritään säätämään niin, että kontaktissa syntyvät voimat ovat lähellä fyysisen materiaalin tuloksia. Tämä osuus on hankala, erityisesti jos kappaleet ovat puristuksessa, jolloin kappale saattaa alkaa värähdellä ja pahimmassa tapauksessa värähtely voimistuu ja simulaatioon syntyy ylimääräisiä voimia. Pitkä-aikainen puristus tai värähtely voi myös päästää kappaleet enemmän sisäkkäin pelkästään tietokoneen numeerisen käsittelyn takia.

Ensiksi mallinnuksessa haettiin puulle ja porrasannostelijan välille sopivat materiaaliarvot, jonka jälkeen asetettiin muille saman tyyppisille kontaktikappaleille samat arvot. Kappaleille, jotka

puristavat tai pitävät pölliä paikoillaan kuten karat, annettiin erilaiset materiaaliarvot kuin muille kappaleille.

### 3.8 Puupartikkeli

Laitteiston testausta varten luotiin kolme erilaista pöllipartikkelia. Pöllien koko määriteltiin laitteiston äärirajojen mukaan. Taulukossa 2 on esitetty suurin, keskikokoinen ja pienin mahdollinen pölli, joita pystytään ajamaan sorvissa. Pöllit ovat asetettu globaalin Z-akselin suuntaisesti, jolloin pöllin inertiamatriisi voidaan laskea kaavan 17 mukaan.

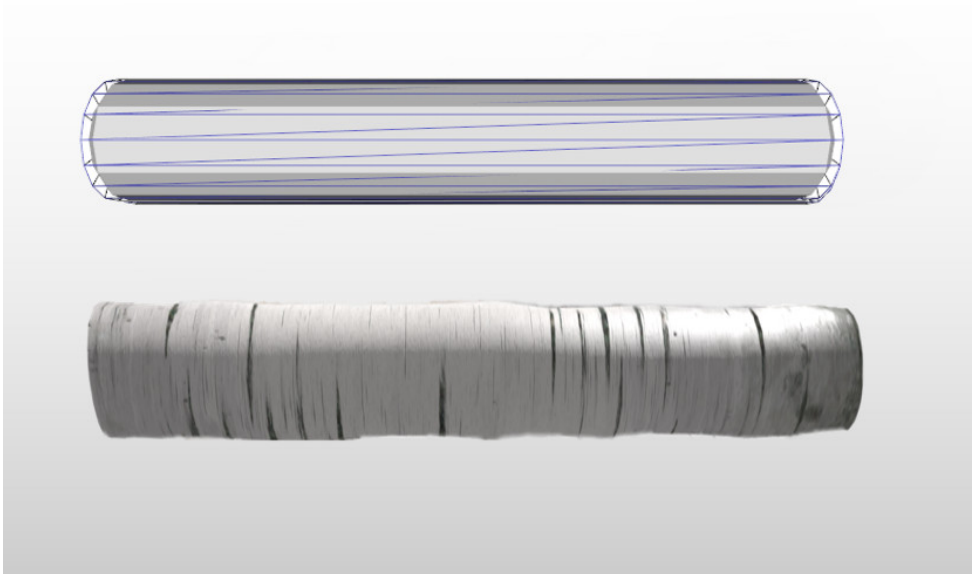
*Taulukko 2. Eri puiden parametrit*

| ID | Tiheys [kg/m3]. | Pituus [m] | Säde[m] | Tilavuus [m3] | Paino [kg] |
|---|---|---|---|---|---|
| 1 | 525 | 1,65 | 0,3 | 0,466527 | 244,9264 |
| 2 | | 1,35 | 0,12 | 0,061073 | 32,06309 |
| 3 | | 0,95 | 0,075 | 0,016788 | 8,81364 |

$$I = \begin{bmatrix} \frac{M}{12}(h^2 + 3r^2) & 0 & 0 \\ 0 & \frac{M}{12}(h^2 + 3r^2) & 0 \\ 0 & 0 & \frac{1}{2}Mr^2 \end{bmatrix} \tag{17}$$

Kaavassa $M$ on kappaleen kokonaismassa, $h$ on kappaleen korkeus eli tässä tapauksessa pituus ja $r$ on kappaleen säde. Sekä pöllille että pöllin kanssa kosketuksessa oleville kappaleille tuli luoda oma kontaktigrafiikka. Topologia kuvista 20 ja 21 nähdään katkoviivanuolella pöllin kulkusuunta ja vihreiden laatikoiden avulla nähdään, mitkä kappaleet ovat kontaktissa pöllin kanssa. Näille kappaleille luotiin omat kontaktigrafiikat. Pöllille asetettiin kuvan 31 mukainen yksinkertainen kontaktigrafiikka. Kuvan alalaidassa näkyy visuaalinen grafiikka ja ylälaidassa kontaktigrafiikka. Kontaktigrafiikkaa käytetään kontaktin laskentaan, eikä se ole näkyvissä simuloinnin aikana, kun taas visuaalisella grafiikalla havainnoidaan pölli simuloinnissa. Kontaktigrafiikasta pyritään tekemään yksinkertaistettu jäljennys alkuperäisestä kappaleesta, jotta kontaktilaskenta ei ylikuormittaisi laskentaa. Kontaktilaskennassa jaetaan kontaktigrafiikka kolmioihin, joiden asemaa tarkastellaan verrattuna toiseen kontaktigrafiikkaan. Mitä yksinkertaisempi kontaktigrafiikka on, sitä vähemmän joudutaan tarkastamaan eri pintojen kontaktia, jolloin laskentatehoa jää enemmän muuhun. Kontaktimalli saattaa reagoida poikkeavalla tavalla, jos kappaleessa on paljon teräviä kulmia. Terävät kulmat saattavat

upota liian helposti toiseen kappaleeseen, jolloin syntyy ylisuuria voimia. Itse kontaktivoimat lasketaan molempien kappaleiden materiaaliarvojen mukaan.



**Kuva 31.** Puupartikkelin visuaali- ja kontaktigrafiikka

# 4 TULOKSET

Virtuaaliprototyyppi on muodostettu tuotekehityksen ensimmäisten iterointien perusteella, joten lopulliseen koneeseen tulee vielä muutoksia. Tuloksissa ei tarkastella hydrauliikan ja pneumatiikan toimintaa, koska niiden toiminta ei ollut suunnitelmien mukainen. Hydrauliikka- ja pneumatiikkajärjestelmät ovat luotu yksinkertaisina malleina, ja niiden komponenttien tiedot ovat arvioitu, joten niiden tulokset eivät ole vertailukelpoisia.

Virtuaaliprototyyppimallin tulokset ovat erittäin riippuvaisia siitä, minkälaisessa tietokoneessa prototyyppiä ajetaan ja minkälaisen aika-askeleen sitä vastaava alusta mahdollistaa. Jos mallin aika-askel on pienempi kuin mitä tietokone pystyy laskemaan, prototyypin tulokset eivät ole luotettavia. Jos taas aika-askel on liian suuri, mallin tulokset alkavat poiketa fyysisestä laitteesta.

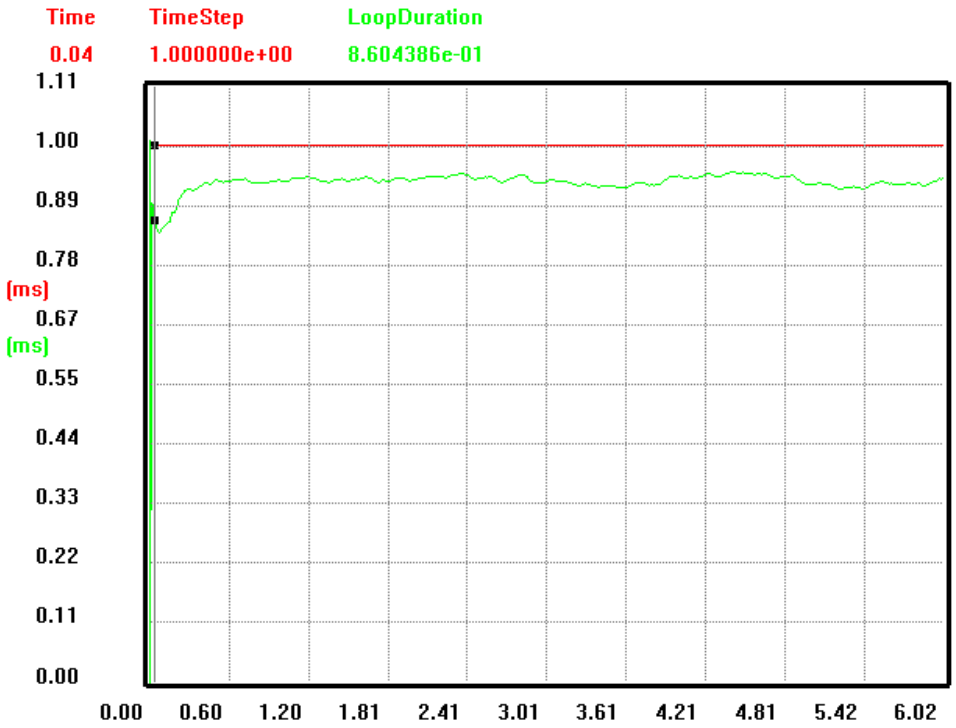
## 4.1 Virtuaaliporotyypin suorituskyky

Reaaliaikainen simulaatio vaatii sen, että simulaattorin yhden aika-askeleen laskenta ei vie enempää aikaa kuin määritetyn aika-askeleen verran. Jos laskenta vie enemmän aikaa, malli ei käyttäydy enää reaaliajassa. Tämä oli tärkein kriteeri mallin hyväksymiselle. Virtuaaliprototyypille asetettiin aika-askeleeksi 1 ms. Tällä asetusarvolla mallin kontaktivoimat pysyvät normaaleina, ja nykyaikaisilla tehokkaimmilla tietokoneilla laskenta-aika pysyy alle 1 ms:ssa. Jos mallin aika-askeleen nosti 2 ms:iin, aiheutti se mallissa epävakautta. Kun aika-askel oli asetettu 2 ms:iin ja pölli asetettiin karojen väliin puristukseen, liian suuret kontaktivoimien värähtelyt hajottivat MBS-mallin. Malli hajosi, koska karojen kontaktigrafiikka oli liian monimutkainen ja 2 ms:n aikana kara oli uponnut puuhun liian paljon, ennen kuin syntyi vastavoimaa. Asiaan vaikutti myös se, että kontaktimateriaalit haettiin 1 ms:n aika-askeleen mukaan. Jos aika-askeleen haluaisi vaihtaa 2 ms:ksi, vaatii se kontaktimateriaalien uudelleen määrittelyn.

Virtuaaliprototyyppiä testattiin tietokoneella, jossa oli 16 Gt:a keskusmuistia, Intel Core i7 3,4GHz -prosessori, Windows 7 ja NVIDIA GeForce GTX 780 -näytönohjain. Testejä kokeiltiin ajaa pienitehoisemmilla kannettavilla tietokoneilla, mutta niissä laskentateho ei riittänyt saavuttamaan tarvittavaa laskentanopeutta. Kannettaville tietokoneille saatiin parhaimmaksi keskimääräiseksi laskenta-ajaksi noin 4 ms:a.

Suurimman osan laskenta-ajasta vei kappaleiden ja rajoitteiden monikappaledynamiikan laskenta. Mallin kappalemäärää karsittiin ja toimintaa yksinkertaistettiin, jonka jälkeen mallissa oli 34 liitosta ja 34 kappaletta. Mallissa oli yhteensä 30 vapausastetta. Kappaleiden ja rajoitteiden suuri määrä vaatii paljon laskentatehoa. Tämän takia laitteen MBS-mallin suunnittelu pyritään tekemään huolellisesti, jotta mallissa ei olisi ylimääräisiä kappaleita tai rajoitteita, jolloin mekanismista tulisi ylirajoittunut. Mallin kappaleiden ja liitosten MBS-laskenta vei keskimäärin 0,75 ms:a. Tämä on suhteellisen suuri aika, kun vertaa keskimäärästä työkoneen simulaattoria, jossa on noin 17 kappaletta ja 16 rajoitetta, ja jonka kappaleiden MBS-laskenta vie noin 0,37 ms.

Testitietokoneella virtuaaliprototyypin laskenta-aika oli keskimääräisesti 0,95 ms kontaktigrafiikan ja Julia-moottoriohjauksen kanssa. Mallin aika-askeleen ja laskenta-ajan vertailu on esitetty kuvassa 32. Kuvassa punainen viiva esittää asetettua aika-askelta ja vihreä käyrä esittää mallin laskenta-aikaa. Testien aikana huomattiin laskenta-ajan laskevan, kun moottoriohjausjärjestelmä käynnistettiin. Tämä voi johtua joko kontaktigrafiikan asennon muutoksesta tai tietokoneen tavasta jakaa laskentatehoa eri ytimiin. Tietokoneen käyttöjärjestelmän toiminnalla ja taustalla käyvillä ohjelmilla oli myös suuri vaikutus laskenta-aikaan. Parhaat laskentatehot saatiin tietokoneesta, josta kaikki ylimääräiset sovellukset ja toiminnot olivat suljettu. Ilman Julia-moottoriohjausta laskenta-aika oli noin 0,9 ms:a ja ilman puupartikkeleita laskenta-aika oli 0,8 ms:a.




**Kuva 32.** Mallin laskenta-ajan ja aika-askeleen vertailu.

## 4.2 Rajapintatoiminta

Rauten ohjausjärjestelmälle luotiin ensiksi sokettiyhteys, mutta sokettiyhteys hidasti MBS-mallin toimintaa, koska ohjausjärjestelmä päivittää signaaleja 10 ms:n välein. Sokettiyhteydessä virtuaaliprototyyppi odottaa aina 10 ms, ennen kuin se laskee seuraavan aika-askeleen. Sokettiyhteydestä ei voitu luopua, koska se oli ainut tapa, jolla ohjausjärjestelmään saatiin yhteys. Mevea toolbox -työkalun avulla kuitenkin pystyttiin luomaan yhteys mallin ja ohjausjärjestelmän välille, ilman että jompikumpi joutui odottamaan. Toolbox -työkalun avulla muodostettiin ohjausjärjestelmään yhteys, josta signaalit kirjoitettiin Mevean I/O-pooliin. I/O-poolissa signaalien ei tarvinnut olla synkornisia ja jokaista kanavaa voitiin päivittää eri tahdissa. Mevealla oli tehty C-kielen kirjasto funktioista, joilla pystyttiin lukemaan ja kirjoittamaan analogista ja digitaalista signaalia. Kirjaston avulla pystyttiin Julia-ohjelmointikielellä lukemaan I/O-poolista signaaleja ja niiden perusteella ohjaamaan mallia.

I/O-poolin käyttöönotto helpotti huomattavasti signaalien järjestelyissä. Sokettiyhteydessä ei pystynyt tarkastamaan, mitä tietoja tiettynä hetkenä lähetetään ilman erillistä työkalua. Koko viestin pystyi purkamaan, mutta aina rajapintaa muuttaessa joutui myös tarkastelua muuttamaan. Lisäksi Mevea-ohjelmisto on rakennettu I/O-pool -ympäristöön, jolloin signaalien säätäminen

helpottui huomattavasti.

Sorvausprosessimalli täytyi olla synkronissa simulaattorin kanssa, minkä takia prosessimallin yhteydessä käytettiin sokettiyhteyttä. Sorvausprosessimalli oli myös luotu Julia-kielellä, josta löytyi sisäiset funktiot sokettiyhteyden muodostamiseen. Sokettiyhteyden muodostamista varten tuli selvittää, miten Mevean solveri -ohjelmisto kommunikoi sokettiyhteyden yli. Solveri tarkistaa ensin, kuinka monta signaalia toinen osapuoli on lähettämässä ja vastaanottamassa, minkä jälkeen se vertaa saatuja arvoja malliin asetettujen signaalien määrään. Jos arvot täsmäävät, yhteys voidaan luoda. Tämä tarkastus tehdään, jotta vältetään viestien väärää purkamista. Tästä yhteystavasta on esimerkki liitteessä II.

Sorvausprosessia ei päästy kunnolla testaamaan, koska sorvausprosessin kehitys oli vielä kesken tämän työn aikana. Molemmat osapuolet testasivat yhteyden toimivuuden, eikä siinä ollut ongelmia.

### 4.2.1 Moottoriohjausjärjestelmän toiminta

Moottoriohjaus oli rakennettu yhteen ohjelmaan, joka käy jokaisen moottorin läpi jokaisen aika-askeleen aikana. Ohjelmassa moottorit ovat määritetyssä järjestyksessä listassa, jonka perusteella ohjelmisto tietää mitä moottoria ollaan ohjaamassa. Lineaarimoottoreille asematiedot lähetettiin millimetreissä ja nopeudet $mm/s$. Pyöriville moottoreille asema lähetettiin 360 asteen vaiheena, mutta nopeus oli silti $mm/s$. Pyörivän moottorin asemaohjauksessa muodostui ongelmaksi asteiden jatkuvuus 0 ja 360 välillä. Jos moottori oli asetettu asemaan 360 astetta, saattoi pienikin liikahdus siirtää asema arvoksi 0. Tämän jälkeen moottorit teki kokonaisen kierroksen saavuttaakseen uudestaan määritetyn aseman. Tämä ratkaistiin mittaamalla kuljettujen asteiden jatkuvaa matkaa. Tällä menetelmällä pyörivien moottoreiden asemointi ei tuottanut samoja ongelmia.

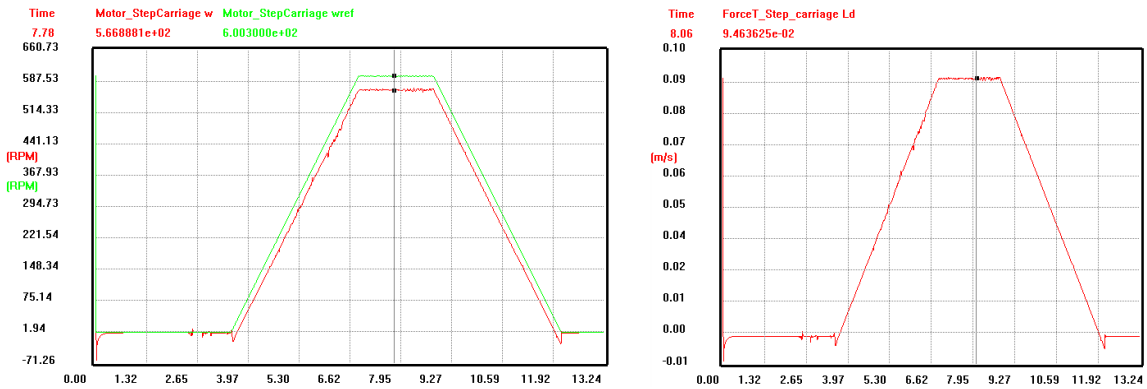
Muita haasteita ohjausjärjestelmän muodostamisessa oli myös Rauten ohjausjärjestelmän rajapinta ja ohjausjärjestelmältä vaadittu toleranssi. Ohjausjärjestelmän piti saada muutettua ohjaustoimintaa, asema- tai nopeussäätöiseksi kesken ajon. Tämä toi haasteita erityisesti PID-säätimelle, joka piti aina nollata, kun ohjaustapa muuttui.

Moottoreita ohjattiin lähettämällä mallille ohjausruuvin nopeusarvo, joka muutettiin moottorikohtaisella kertoimella moottorin pyörimisnopeudeksi. Koska välityssuhteet olivat vakioita, kertoimen pystyi laskemaan kuularuuvin noususta ja vaihteen välityssuhteesta.

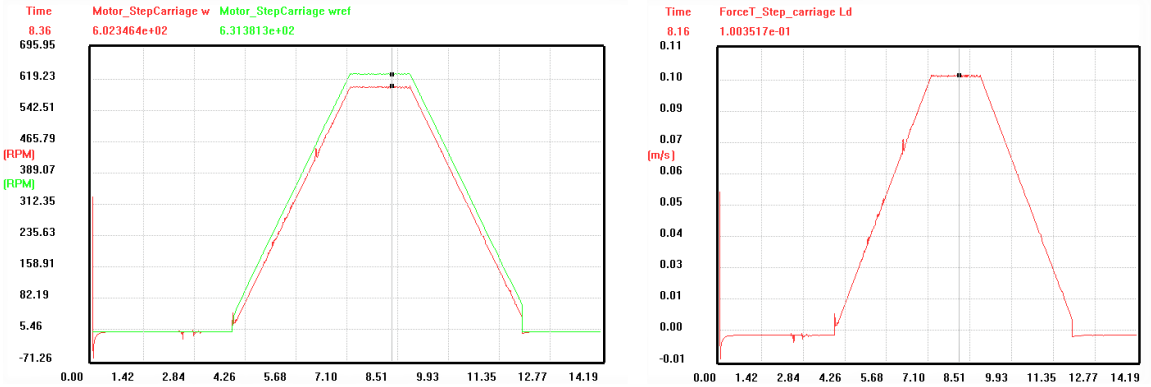
Eniten haasteita tuotti moottorin pysäyttäminen oikealle kohdalle. Vaikka nopeusprofiili oli laskettu valmiiksi, saattoi mallin moottori hidastua nopeammin tai hitaammin johtuen jarrun kitkasta, jarrun nopeudesta ja kuorman suuruudesta. Ohjausjärjestelmä pysäyttää moottorin, jos tavoiteasema on saavutettu, vaikka laitteella olisi nopeutta. Ohjausjärjestelmän pysäytysmekanismi ja nopeusprofiilin muodostaminen vaativat lisää tarkastelua, jotta tavoitellut toleranssit voidaan saavuttaa. Tämän työn pää tehtävänä ei kuitenkaan ollut muodostaa ohjausjärjestelmää, joten sen muodostamiseen ei hukattu liikaa aikaa.

### 4.2.2 Moottoriohjausjärjestelmän testaaminen

moottoriohjauksen toimintaa testattiin porrasannostelijan moottorilla. Moottori kuljettaa porrasannostelijaa ylös ja alas, jonka tuloksena pölli siirtyy portaikossa eteenpäin. Porrasannostelijan kyydissä oleva pöllin massa oli 32 kg. Testissä ensin moottoria ohjattiin ilman PID-säädintä ja toisen kerran PID-säätimen kanssa. Ohjausjärjestelmälle lähetetään tavoiteasemaski 500 mm, mikä on porrasannostelijan maksimiasema. Moottoriohjausjärjestelmälle lähetetään tavoite nopeusprofiilin, jonka kiihtyvyys on asetettu $30mm/s^2$:een, nopeus on asetettu 100 $mm/s$:iin ja jarrutus on asetettu -$30mm/s^2$:een. Testin tulokset ilman PID-säädintä näkyvät kuvassa 33 ja PID-säätimen kanssa kuvassa 34. Vasemmassa kuvassa on esitetty moottorin ohjaussignaali (Lwref, vihreä) ja moottorin todellinen pyörimisnopeus (Lw, punainen). Oikeanpuoleisessa kuvassa nähdään voimakomponentin liikenopeus $m/s$, mikä vastaa kuljetusruuvin nopeutta.




**Kuva 33.** Porrasannostelijan moottorin testaus ilman PID-säädintä.




**Kuva 34.** Porrasannostelijan moottorin testaus PID-säätimen kanssa.

Ensimmäisestä kuvasta havaitaan, että moottorin kuljetusruuvi ei pääse tavoitenopeuteensa, vaan kulkee 6 $mm/s$ hitaammin kuin tavoitenopeus. Tämä johtaa siihen, että porrasannostelija jää 4 mm:ä liian kauas asetusasemasta. Kuvasta voidaan myös nähdä, kuinka pyörimisnopeus laskee ensiksi miinuspuolelle. Tämä johtuu jarrujen vapautumisesta ja hetkellisestä tehonpuutteesta. Kuvassa 34 sama tilanne toistetaan, mutta tällä kertaa moottorin PID-säädin on päällä. Oikeanpuolisesta kuvasta voidaan havaita, että tässä tapauksessa moottori pysyy tavoitenopeudessa. Lopputuloksena kuularuuvin asema poikkeaa halutusta asemasta 0.02 mm, mikä on parempi toleranssi kuin mitä on vaadittu. Kuvasta voidaan myös nähdä, että PID-säädin ei anna moottorin tippua aluksi väärään suuntaan, jolloin pystytään paremmin noudattamaan nopeusprofiilia. Nopeusprofiilin lopussa nähdään äkillinen pysähtyminen, mikä johtuu moottoriohjausjärjestelmän toiminta tavasta. Moottoriohjaus nollaa lähetyssignaalin ja PID-säätimen, kun haluttu asema saavutetaan, vaikka moottorilla olisi vielä nopeutta. Tästä johtuen moottoriin kytketty jarru ottaa heti kiinni ja kuularuuvi pysähtyy siihen asemaan. Tämän tyyppinen pysäytys saattaa toimia mallissa, mutta voi jossakin tilanteissa aiheuttaa rajuja hidastuksia. Tämä ohjaustapa ei esimerkiksi soveltuisi hydrauliikkasylinterin ohjaukseen.

Moottoriohjausjärjestelmässä ilmeni myös tapauskohtaisia ongelmia. Ongelmat johtuivat joko järjestelmän toiminta tavasta tai monen tekijän yhteisvaikutuksesta. Esimerkiksi moottoriohjausjärjestelmään tarkkailee jatkuvasti, onko laite tavoiteaseman toleranssin sisäpuolella. Jos kuularuuvi menee yli tai jää liian kauas asetusasemasta, lähtee moottoriohjausjärjestelmä tavoittelemaan uudestaan haluttua asemaa. Jos nopeusprofiilin kiihtyvyysarvot olivat liian nopeita ja jarrun kitkaominaisuudet liian pieniä, kiihdyttää moottoriohjausjärjestelmä moottorin aina liian suureen nopeuteen, jota jarru ei pysty jarruttamaan tarpeeksi ajoissa, jolloin moottori liukuu yli tavoiteaseman. Tämä ominaisuus

saattoi ajaa moottorin asemahakukierteeseen.

Jossain tapauksissa moottorille oli asetettu tavoittamaton tavoiteasema. Tällöin moottorin liike pysähtyi päätyyn, mutta moottoriohjausjärjestelmä yritti edelleen ohjata moottoria tavoiteasemaan. Tällöin PID-säädin yrittää korjata tilannetta lisäämällä ohjausarvoa rajusti, jonka takia moottoriohjausjärjestelmään on lisätty PID-säätimen kontrolli, joka valvoo, että säädin ei anna liian suurta korjausta. Jos PID-säätimen korjaus ylittää tietyn rajan, pysäytetään moottori siihen paikkaan. Tämä taas johtaa kahteen uuteen ongelmaan. Jos ohjausjärjestelmän lähettämä nopeusprofiili on nopeampi, kuin mihin moottori kykenee, yrittää PID-korjata tilannetta liikaa, jonka tuloksena moottori pysähtyy. Toinen ongelmatilanne liittyy mittaustiheyteen. Tämä tuli esille, kun prototyyppiä ajettiin tehottomassa tietokoneessa ja laskenta-aika oli lähempänä 4,0 ms, vaikka asetettu aika-askel oli 1,0 ms. Tässäkin tapauksessa moottoriohjausjärjestelmä tulkitsee, että moottori ei pysy annetussa nopeudessa, jolloin PID-säädin yrittää korjata virhettä. Tämä johtaa liian suureen korjaukseen, jolloin moottori pysähtyy.

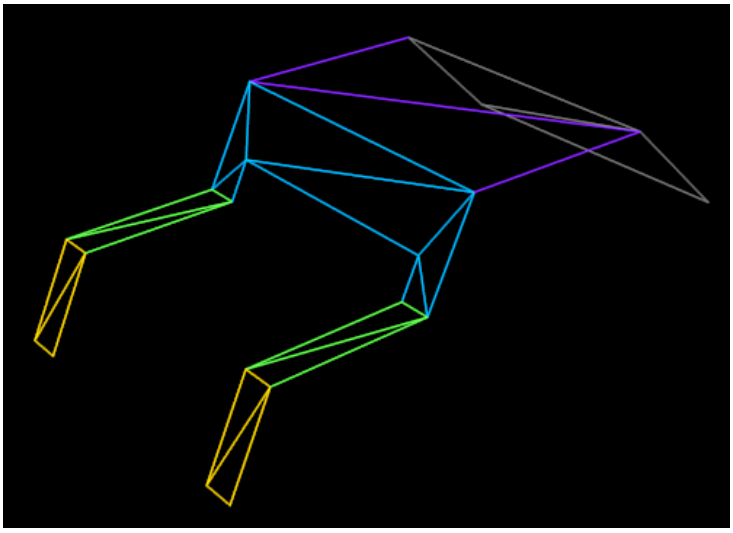
## 4.3 Puupartikkelit

Puupartikkeleiden ja muiden kappaleiden välisten kontaktivoimien säätäminen kohdilleen loi paljon haasteita. Puupartikkelit aiheuttivat paljon värähtelyjä erityisesti karoissa, jossa puu joutuu karojen puristukseen. Kun simulaation aika-askelta nostettiin 2,0 ms:iin, aiheutti pöllin ja karojen välinen kontaktivoima ylisuuria voimia, mikä teki mallista epävakaan.

Puupartikkelien kontaktivoimien laskenta vei toiseksi eniten laskentatehoa, heti monikappaledynamiikan laskennan jälkeen. Puupartikkeleiden kanssa simulaatiomalli tarvitsi tehotietokoneen laskentaa varten, koska heikkotehoisemmat tietokoneet eivät pysyneet reaaliajassa. Testitietokoneella puupartikkeli lisäsi noin 0,2 ms laskenta-aikaa. Puupartikkelit lisäävät laskentakuormitusta, koska jokaisen kappaleen geometriaa pitää vertailla ja tarkistaa, ovatko kappaleet kontaktissa keskenään. Kontaktigrafiikassa jokainen pinta-ala on ajettu kolmioihin. Eri kappaleiden geometrioita eli kolmioita tarkistellaan kontaktilaskennassa. Jos kontaktigrafiikka on erittäin yksityiskohtainen eli sisältää paljon kolmioita, rasittaa se tietokoneen laskentaa huomattavasti.

Kontaktigrafiikan laskentaa pystytään keventämään kontaktigrafiikan suunnittelulla. Grafiikasta tulee poistaa kaikki ylimääräiset pinnat ja ylimääräiset muodot, kuten ruuvien reiät. Laskentaa

pystyttiin keventämään huomattavasti jakamalla kontaktigrafiikan eri pinnat omiksi osiksi. Kuvassa 35 on esitetty yksinkertaistuksen tulos ja pintojen jakamisen idea.

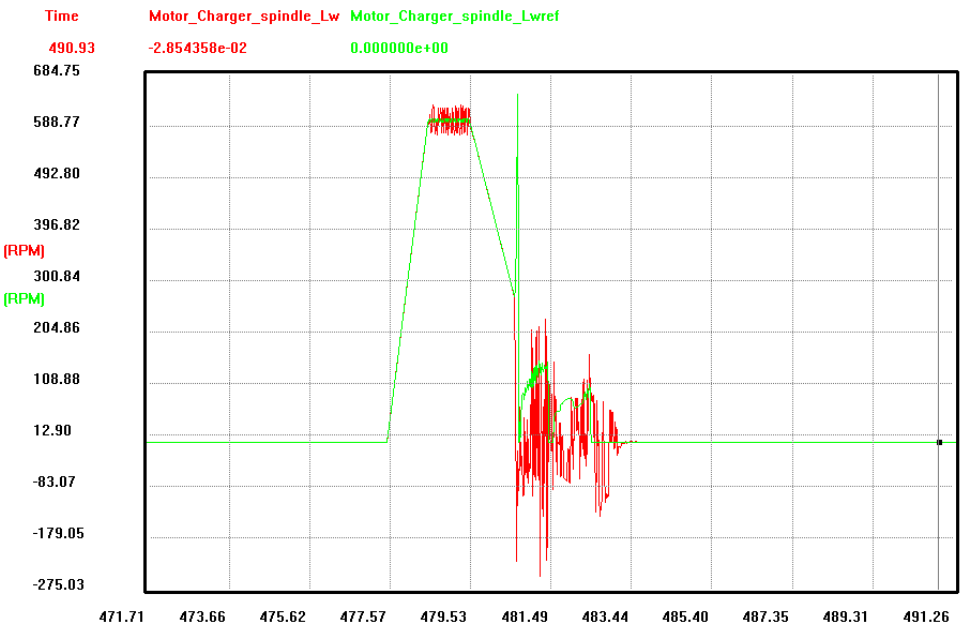


**Kuva 35.** Jaettu kontaktigrafiikka.

Ensimmäinen versio kontaktigrafiikasta on esitetty kuvassa 19, jossa kappaleen kontaktigrafiikassa oli monia turhia pintoja samassa kappaleessa. Tällaisen kontaktigrafiikan tarkastelu vei turhaan laskentatehoa. Kuvassa 35 eri värit kertovat miten kappale on jaettu eri pintoihin. Näin kontaktilaskenta käyttää vain lähintä pinta-alaa tarkasteluun, eikä tarkastele koko kappaleen kaikkia pintoja. Tämä nopeuttaa tarkastelua huomattavasi, koska tarkastelun ei tarvitse tarkistaa turhien pintojen suhdetta toisiinsa. Kuvasta voidaan myös havainnoida, miten pintojen grafiikka on jaettu kolmiorakenteisiin. Tällä menetelmällä saatiin kontaktilaskentaa kevennettyä noin 0,15 ms.

Kontaktimallissa haastavinta on kontaktimateriaalinen määrittäminen. Kun kahden kappaleen välille määrittelee tietyn materiaalin, se toimii parhaiten juuri siinä aika-askeleessa, jossa määrittely on tehty. Materiaalin todellisilla arvoilla voidaan saada osviittaa materiaaliominaisuuksiin, mutta loppujen lopuksi kontaktivoimat perustuvat aika-askel-laskentamenetelmää, jolloin materiaalien arvot tulee hakea aika-askelen mukaan.

Kontaktigrafiikoiden ja kontaktivoimien välisen tasapainon löytäminen loi paljon haasteita. Tämä ilmeni kappaleiden värähtelyllä tai pomppimisella. Kuvassa 36 on esitetty kuinka karan moottori käyttäytyi kun puupartikkeli oli karojen välissä puristuksessa. Kuvassa nähdään punaisena käyränä moottorin todellinen kierrosnopeus ja vihreänä viivana ohjausarvo.




**Kuva 36.** Vasemman karan moottorin toiminta.

Nopeusprofiilin jarrutuksessa huomataan äkillinen ohjaussignaalin nousu. Tämä syntyy PID-korjauksesta, jossa säädin yrittää saada moottorin jarruttamaan profiilin mukaan, vaikka karat ovat jo ottaneet puusta kiinni. Lopussa moottori pyrkii saavuttamaan uudelleen saman annetun aseman. Tämä aiheuttaa suurta värähtelyä kappaleissa. Ohjausjärjestelmä täytyi muuttaa sellaiseen muotoon, että se ei yritä ohjata karaa eteenpäin, kun puu on kiinni. Kuitenkin puu pitää olla pienessä puristuksessa, jotta se pysyy karojen välissä pyörityksen ajan.

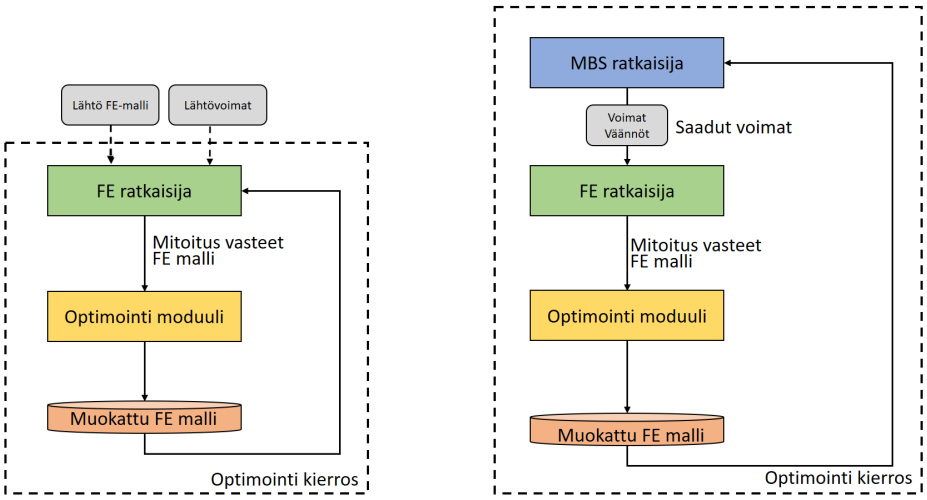
## 4.4 Virtuaaliprototyyppi osana tuotekehitystä

Sorvisimulaattoria hyödynnetään tuotekehityksessä, kuten kappaleessa 2.1 on esitetty. Tulevaisuudessa yhä useamman tuotesuunnittelijan työkaluna on myös monikappaledynamiikkaprototyyppi (MBS). MBS-mallit ovat seuraava askel sitten CAD-työkalujen käyttöönoton. Kun CAD-työkalujen avulla pystytään tarkastelemaan kappaleen geometriaa ja sen massan ominaisuuksia, MBS-mallin avulla voidaan arvioida laitteen kokonaisuutta ja laitteen mekaniikan toimivuutta. (Baumgartner & Pfeiffer 2014, s. 914.)

MBS-mallin avulla voidaan esimerkiksi tarkkailla kuinka paljon energiaa yksi työsykli vie. Tämän avulla voidaan laskea, kuinka paljon tuleva koneisto tulee kuluttamaan energiaa, millä voidaan saada koneesta entistä energiatehokkaampi. Mallin avulla voidaan myös havainnoida, missä vaiheessa laite kuluttaa eniten energiaa ja minkälaiset muutokset vähentävät laitteen

energiakulutusta. Energiatehokkaalla laitteella voidaan saavuttaa merkittävä etu kilpailijoihin.

Mallin avulla nähdään myös minkälaisille voimille kappaleet altistuvat. Näillä tiedoilla voidaan kappale suunnitella uudestaan käyttäen FE-menetelmää. Perinteinen kappaleiden optimointijärjestelmä on havainnoitu kuvassa 37 vasemmalla. Siinä kappale tuodaan kerran sisälle ja sille asetetaan kerran lasketut voimat. Tämän jälkeen FE-laskentakierto optimoi kappaleen muodot sopivaksi. Tässä tapauksessa ei huomioida, miten kappaleen muodon vaikuttaa muihin kappaleisiin. Laajennetussa optimointijärjestelmässä, kuten kuvassa 37 oikealla, tämä on otettu huomioon MBS-mallin avulla. Järjestelmä ottaa huomioon kappaleen muodonmuutoksesta aiheutuva dynaamiset muutokset. Näin pystytään tarkemmin ottamaan huomioon yhden kappaleen muutos koko järjestelmään. (Albers et al. 2007, s. 4.)



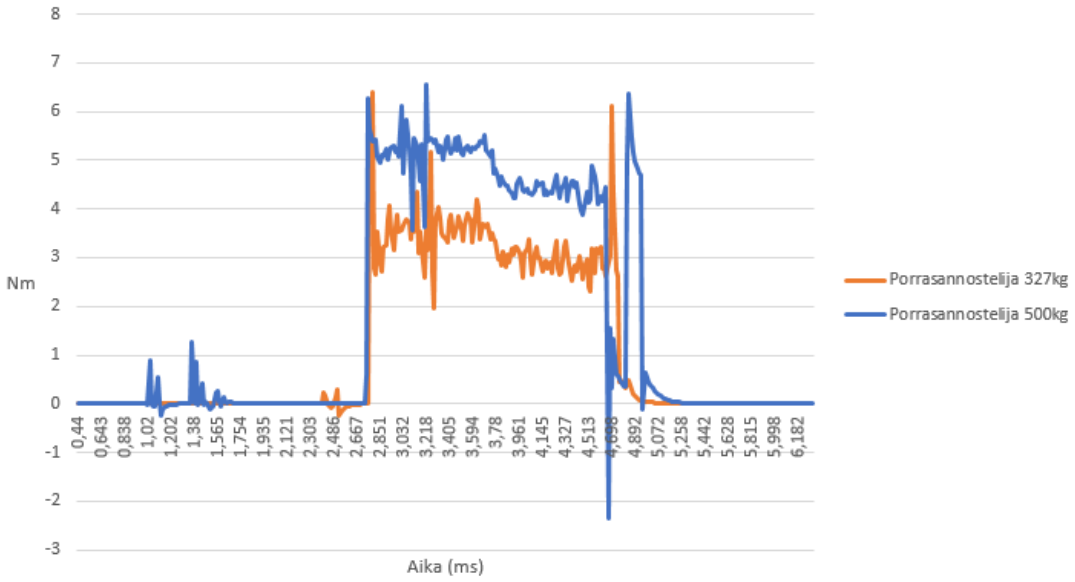

**Kuva 37.** Topologia optimointi MBS-mallin avulla (muokattu Albers et al. 2007, s. 4).

Moottorien ohjaamista voidaan jo kehittää virtuaaliprototyypin avulla, testaten mekaniikan toimivuutta samalla. Mallin avulla voidaan optimoida ajoajat jo ennen varsinaista fyysistä prototyyppiä, mikä lyhentää tuotekehityksen aikaa merkittävästi. Lisäksi jos tuotekehityksen alkuvaiheessa löytyy puutteita ohjelmoinnissa, ovat ne vähäkustanteisimpia, kun tuotekehityksen loppuvaiheessa.

Seuraavassa esimerkkitapauksissa esitellään miten MBS-mallia voidaan hyödyntää tuotekehityksessä. Esimerkeissä lisätään kappaleen massa ja katsotaan miten se vaikuttaa moottorin toimintaan. Toisessa esimerkissä siirretään moottorin asemaa ja tarkastellaan miten se vaikuttaa liitoksen voimiin.

### 4.4.1 Massan muutosesimerkki

Esimerkissä tarkastellaan porrasannostelijan lineaarista karamoottoria. Moottori pidetään samanlaisena ja sitä ohjataan samoilla arvoilla, mutta porrasannostelijan massaa muutetaan. Ensimmäisessä koeajossa porrasannostelijan massa on alkuperäinen eli 327 kg. Toisessa koeajossa porrasannostelijan massaa kasvatetaan 500 kg:aan. Suurella muutoksella kyetään selkeämmin esittämään massamuutoksen vaikutus moottorin toimintaan. Kuvassa 38 on esitetty moottorin vääntökäyrä, kun se kuljettaa kuljetusruuvin ääriasentoon. Kuvassa oranssi käyrä kuvaa moottorin vääntöä, kun porrasannostelijan massa on alkuperäinen, kun taas sininen käyrä esittää tilannetta jossa porrasannostelijan massa on 500 kg. Molemmissa tapaukissa porrasannostelijan kyydissä on ollut myös puupartikkeli, jonka massa on 32 kg.




**Kuva 38.** Porrasannostelijan moottorin vääntökäyrä eri massaisilla porrasannostelijoilla.

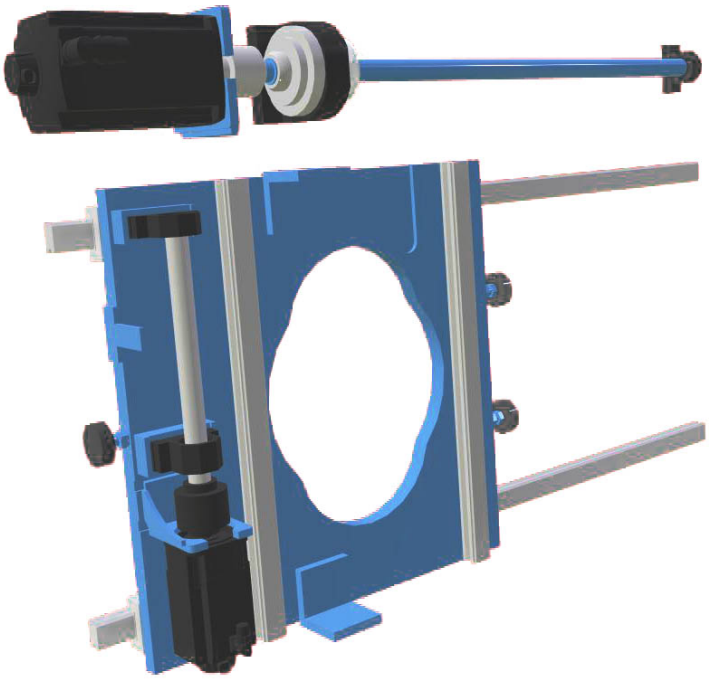
Kuvasta voidaan havaita, että kevyemmällä porrasannostelijalla moottorin ei tarvitse tuottaa yhtä paljon vääntömomenttia, kuin painavammalla porrasannostelijalla. Kuvaajasta nähdään, miten moottorin jarrutus alkaa puolessa välissä ajoa moottorin vääntömomentti vähentyessä. Lopussa huomataan jarrun äkillinen vaikutus, jonka jälkeen säädin korjaa vielä uudelleen asemaa. Näillä tilastoilla voidaan mitoittaa sopivan kokoinen sähkömoottori kyseiselle laitteelle, jolloin voidaan optimoida esimerkiksi moottorin energian käyttö. Esimerkin avulla voidaan myös huomata, onko moottorin ohjausjärjestelmä toimiva.

Esimerkin tapaan, voidaan koneen kaikkien moottoreiden tarvitsemaa vääntömomenttia tarkastella ja päättää sen mukaan, minkälaisia moottoreita koneeseen hankitaan. Mallin avulla

voidaan myös tutkia eri massaisten pöllien vaikutusta koneen kiertoajoon ja optimoida sen mukaan ajoasetuksia.

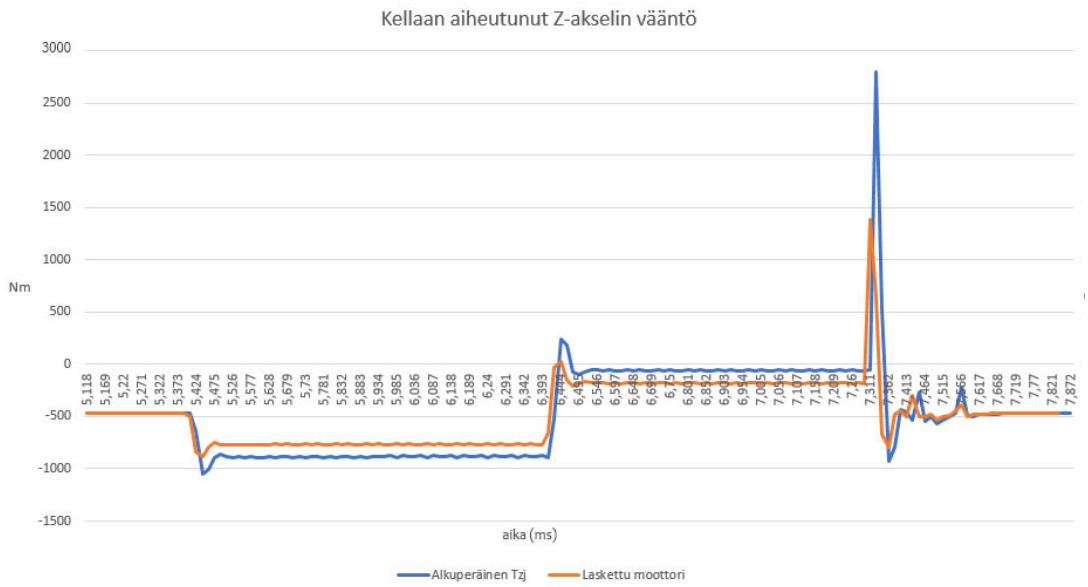
### 4.4.2 Moottorin asemamuutosesimerkki

Toisessa esimerkissä uudelleen paikoitettiin XY-keskittäjän X-kelkan kuljetusruuvi. Kuvassa 39 on esitettynä X-kelkka keskittäjän rungon kiskoilla ja sen yläpuolella olevaa kuljetusruuvia. Kuvassa X-ulottuvuuden suunta on kelkan kiskojen kanssa yhdensuuntainen ja Z-ulottuvuus on kohtisuorassa X-ulottuvuutta vastaan vaakasuorassa. Esimerkissä kuljetusruuvi uudelleen sijoitettiin 20 cm alemmaksi alkuperäisestä asemasta ja tämän jälkeen tutkittiin muutoksen vaikutusta X-kelkan translaationiveleen. MBS-mallin ratkaisumenetelmä tuli vaihtaa rekursiivisesta Lagrange-menetelmään, koska rekursiivinen ratkaisumenetelmä ei laske nivelvoimia.



**Kuva 39.** XY-keskittäjän X-kelkka kiskoilla. Ylhäällä olevaa moottoria uudelleen sijoitettiin 20 cm alemmaksi.

Kuvassa 40 on esitetty X-kelkan translaationivelen Z-akseliin kohdistuva vääntömomentti, kun kelkkaa ajetaan uuteen asemaan.




**Kuva 40.** X-kelkan translaatioliitoksen vääntömuutos moottorin uudelleenasettelun jälkeen.

Kuvassa sininen käyrä esittää liitokseen kohdistunutta vääntöä, kun kuularuuvi on alkuperäisessä asemassa ja oranssi käyrä esittää vääntökäyrää, kun kuularuuvi on uudelleen sijoitettu 20 cm alemassa. Kummassakin testitapauksessa ohjausjärjestelmälle on lähetetty sama käsky, jossa moottoriohjausjärjestelmä kuljettaa kelkan määrättyyn asemaan. Kuvaajassa väännön suunta riippuu moottorin sijainnista, kulkusuunnasta ja nopeusprofiilin vaiheesta. Kun moottori hidastaa, muuttuu väännön suunta. Kuvaajasta voidaan havainnoida liikkeen eri vaiheet. Ajon alussa nähdään vääntöpiikki, joka johtuu moottorin kiihdytyksestä. Tämän jälkeen laite saavuttaa tavoitenopeuden, jolloin kohdistunut vääntö tasaantuu. Merkittävin huomio on lopussa tapahtuva piikki väännössä. Tämä johtunee jarrun lukkiutumisesta ennen, kun kelkka on pysähtynyt. Ohjausjärjestelmä pysäyttää moottorin heti, jos asema on saavutettu. Tämä tarkoittaa, että moottoriohjauksen nopeusprofiili ei täsmää kuljetun matkan kanssa. Sama ilmiö voidaan havaita kuvassa 34, jossa moottori pysähtyy lopussa äkillisesti. Tästä huolimatta kelkka menee hieman liian pitkälle ja moottoriohjaus korjaa asemaa hieman takaisinpäin, joka näkyy kuvaajan lopussa kahtena piikkinä.

Moottorin uudelleenpaikoitus vähentää noin 100 Nm liitokseen aiheutunutta kuormitusta. Tietoa voidaan käyttää mekaniikka suunnittelun iterointivaiheissa, mikä lopulta vähentää mekanismin rasitusta.

Esimerkin avulla pystyttiin havaitsemaan suunnitteluvirhe moottorin ohjausjärjestelmässä jo ennen, kuin ohjausjärjestelmää testataan fyysisessä laitteessa. Mallin avulla voidaan myös

nähdä, minkälaiset vaikutukset tällaisella virheellä on mekanismiin, ilman minkäänlaisia vahinkoja. Tällaisen löydön tekeminen vasta fyysisellä prototyypillä olisi viivästyttänyt tuotekehitystä merkittävästi, mikä olisi tullut kalliiksi. Pahimmassa tilanteessa ohjelmointivirhe olisi voinut hajottaa koko nivelliitoksen fyysisestä prototyypistä, jolloin korjaustoiminnot olisivat lisänneet kustannuksia ja viivästyttäneet tuotteen valmistumista.

Virtuaaliprototyypin avulla voidaan testata ohjausjärjestelmää ja optimoida ajoajat jo ennen varsinaista fyysistä prototyyppiä, mikä lyhentää ohjausjärjestelmän kehitysaikaa merkittävästi. Lisäksi jos tuotekehityksen alkuvaiheessa löytyy puutteita ohjelmoinnissa, on puitteiden korjaaminen paljon vähäkustanteisimpia, kuin tuotekehityksen loppuvaiheessa.

# 5 JOHTOPÄÄTÖKSET

Työn tavoitteena oli muodostaa virtuaaliprototyyppi Rauten viilusorvista. Virtuaaliprototyyppi halutiin tuoda tuotekehityksen käyttöön, jonka avulla pystytään tehostamaan tuotekehitystyötä. MBS-mallia haluttiin ohjata Rauten omalla ohjausjärjestelmällä ja mallissa haluttiin testata koneen ajoa erilaisilla puilla. Ohjausjärjestelmää varten tuli luoda erillinen moottoriohjausjärjestelmä, joka kommunikoi Rauten ohjausjärjestelmän kanssa ja ohjaa sen perusteella mottoreiden liikkeitä. Koneen tärkein prosessi eli sorvausprosessi, erotettiin omaksi simulaatioksi. Sorvausprosessimallista luotiin oma projekti, jolle luotiin rajapinta MBS-malliin. Rajapinnan kautta lähetetään MBS-mallin kappaleiden asematietoja ja sorvausprosessimalli lähettää sorvauksesta syntyneitä voimia MBS-mallille.

Työn tuloksena syntyi MBS-malli, jossa on 34 kappaletta ja 34 liitosta. Mallissa on 30 liikkuvaa kappaletta ja 30 vapausastetta. Kappaleiden suuri määrä vaatii paljon laskentatehoa, mikä tekee reaaliaikaisesta simulaatiosta haastavaa. Jotta mallin laskenta-aika pysyy alle asetetun rajan, vaatii se mallin huolellista suunnittelua ja yksityiskohtien karsimista.

Rauten viilusorvissa liikkeet on toteutettu hydraulijärjestelmällä, mutta tässä versiossa haluttiin korvata hydraulisylinterit lineaarisilla sähkökaramoottoreilla. Lineaarisia karamoottoreita ei suoraan löytynyt mallinnus ohjelmistosta, mutta Mevean voimansiirtokomponenteilla pystyttiin toteuttamaan sähkömoottorin ja kuularuuvin yhdistelmä. Mallissa käytettiin kuularuuvin sijaan hammaspyörää ja hammastankoa, mikä muutti pyörimisliikkeen lineaariseksi. Rauten ohjausjärjestelmä ohjaa karamoottoreita joko aseman, nopeuden tai väännön mukaan. Jos moottoria ohjataan asemasäädöllä, moottorille annetaan vain haluttu kuularuuvin asema ja moottorin sulautettu ohjausjärjestelmä ohjaa moottoria kuularuuvin aseman mukaan. Mevea ohjelmistossa sähkömoottoreita pystyi ohjaamaan joko kierrosnopeuden tai väännön mukaan, eikä kuularuuvi pystynyt ohjaamaan aseman, nopeuden tai väännön mukaan. Tämän takia moottoreille tuli kehittää oma ohjausjärjestelmä, joka säätää moottoria annettujen asetusten mukaan. Ohjausjärjestelmä toteutettiin avoimen lähdekoodin ohjelmointikielellä Julia, jolla myös sorvausprosessimalli oli toteutettu.

Virtuaaliprototyyppiä voidaan käyttää tuotekehityksen tukena. Yksi työn tavoitteista oli myös tutustuttaa Rautea MBS-mallin mahdollisuuksiin ja sen muodostamisen haasteisiin. Työn

aikana Rautelle selvisi, mitä mallin rakentaminen vaatii ja kuinka mallia voidaan hyödyntää tuotekehityksessä. Kun viilusorvin mekaniikkaa käytiin ensimmäisen kerran läpi, huomasivat eri alueiden tuotekehittäjät mekaniikassa puutteita, joita ei oltu aikaisemmin huomattu. Jo tässä vaiheessa pystyttiin todistamaan, kuinka mallin käyttöönotto lisää eri osaajien vuorovaikutusta ja siten lisää koneen toiminnan yleistä ymmärrystä.

Kun lopullista mallia käytiin esittelemässä Rautelle, löydettiin mallille erilaisia käyttökohteita ja miten sitä voisi kehittää. Lisäksi eri osaajilla oli erilaisia ideoita, miten he pystyisivät hyödyntämään mallia omassa työssä. Työssä käytiin läpi joitakin esimerkkejä, miten mallia voidaan hyödyntää tuotekehityksessä.

## 5.1 Kehityskohteet

Mallin laskenta-aikaa pystytään alentamaan suunnittelemalla mallin topoliga eri tavalla. Esimerkiksi kaikki maassa kiinni olevat kappaleet voidaan yhdistää yhdeksi kappaleeksi. Lisäksi karojen laakereita ei välttämättä tarvitse tehdä erillisenä kappaleena, jos karat liittää y-kelkkaan sylinteriliitoksella. Näin saataisiin neljä kappaletta pois. Myös sorvinterää kuljettavat ohjauslaakeripesät voidaan mallintaa yhtenä kappaleena, koska ne liikkuvat aina synkronisesti samaan suuntaan. Mallin rakentamisen alussa oli ideana, että laakeripesien liikettä voidaan testata, kun ne ovat virhetilassa. Näillä muutoksilla saataisiin mallista pois 7 kappaletta. Rajoitteiden määrä vähenisi silti vain yhdellä, koska matemaattisesti malli pysyisi lähes samanlaisena.

Työn aikana Mevea-ohjelmistoon sisään rakennettiin python-ohjelmointikielen käyttömahdollisuus. Tämän avulla voitaisiin rakentaa uusi moottoriohjausjärjestelmä, jolloin ei tarvitsisi käynnistää erillistä ohjelmaa ohjaamaan moottoreita. Python-ohjelmointikieli toimii joissakin tilanteissa hitaammin kuin Julia-ohjelmointikieli, mutta moottoriohjauksessa tämä ero ei olisi nähtävissä. Moottoriohjauksen toimintaa voisi ohjelmoida lähemmäksi vastaamaan oikean moottoriohjauksen toimintatapaa. Lisäksi uuden ominaisuuden avulla voidaan jokaiselle moottorille luoda oma ohjausjärjestelmä ja sen avulla voidaan tarkistella ohjelmistoon tehtyjä muutoksia ilman mallin uudelleen käynnistystä, mikä nopeuttaa esimerkiksi säätimien arvojen löytämistä. Tässä työssä moottoriohjauksen teko ei kuitenkaan ollut pääaihe, jonka takia ohjausjärjestelmän tekoon ei laitettu liikaa aikaa.

Mevea-ohjelmistoon on työn aikana tullut myös sähkökomponentteja, joilla voisi korvata

nykyiset moottorimallit. Komponentit ovat lähempänä todellista sähkömoottoria, jolloin saadaan vielä parempi kuva moottorin energiantarpeesta. Mallin hydrauliikka- ja pneumatiikkapiirit voisi päivittää ajanmukaiseksi, kun työn mallissane ovat yksinkertaistettuja malleja, jossa on arvioiduilla säädöillä olevia komponentteja.

**LÄHTEET**

Albers, A., Ottnad, J., Minx, J & Häßler, P. 2007. Topology optimization in mechatronic systems. France, Paris: International conference on engineering design, Iced'07. 9 s.

Ahola, J., M., Hovila, J., Karhunen, E., Nevala, K., Schäfer, T & Nevala, T. 2011. Moniteknisen piensarjatuotteen digitaalinen tuoteprosessi. Espoo: VTT Technical Research Centre of Finland. 121 s.

Airila, M. 1996. [Luku 2:] Perusmekanismit. Teoksessa: Airila, M. Mekatroniikka. Neljäs korjattu painos. Espoo: Otatieto Oy. 6 s.

Baumgartner, D & Pfeiffer, A. 2014. Proceedings of the 10th International ModelicaConference: Automated Modelica Package Generation of Parameterized Multibody Systems in CATIA. [Viitattu 21.3.2017]. Saatavissa: http://elib.dlr.de/90918/1/ECP14096913_BaumgartnerPfeiffer.pdf

Emerson Industrial Automation. 2008. Raute Find Modular Drives A-Peeling. [Viitattu 21.3.2017]. Saatavissa: http://www.emersonindustrial.com/en-US /controltechniques/newsandevents/presscenter/pressreleases/Pages/raute_peeling_lathe.aspx

Flores, P. 2015. Concepts and Formulations for Spatial Multibody Dynamics. University of Minho.Springer. 83 s.

Galal Rabie, M. 2009. Fluid power engineering.Yhdysvallat: The McGraw-Hill Companies. 420 s.

Heason Technology Ltd : Press Releases. www-sivusto 2016. [Viitattu 21.3.2017]. Saatavissa: http://www.tacticalmarcomms.com/news/120/

Julia programming language. www-sivusto 2016. [Viitattu 21.3.2017]. Saatavissa: http://julialang.org/

Koponen, H. 2002. Puutuoteteollisuus 4: Puulevytuotanto. Helsinki. 201 s.

Juha Kortelainen. Semantic Data Model for Multibody System Modelling [Semanttinen tietomalli monikappaledynamiikan mallitiedon hallinnassa]. Espoo 2011. VTT Publications 766. 120 s.

Lehtonen, M. 2006. Simulation-based design process of smart machines. [Viitattu 30.5.2016]. Saatavissa: http://www.vtt.fi/inf/pdf/tiedotteet/2006/T2349.pdf

Mevea Simulation solutions. www-sivusto 2016. [Viitattu 8.6.2016]. Saatavissa: http://www.mevea.com

Mevea solver reference Manual. 2015. Versio 7.7.

Moisio, S. 2013. A soft contact collision method for real-time simulation of triangularized geometires in multibody dynamics. Väitöskirja, Lappeenrannan teknillinen yliopisto. 109 s.

Pahl, G., Beitz, W., Feldhusen, J. & Grote, K.-H. 2007. Engineering Design A Systematic Approach. Kolmas versio. Lontoo: Springer-Verlag. 608 s.

Rouvinen, A. 2003. Mekanismien dynamiikan simuloinnissa sovellettavia numeerisia ja mallinnusmenetelmiä. Tutkimusraportti 44.

Shabana, A. 2001. Computational Dynamics. Yhdysvallat: John Wiley & Sons. 522 s.

Shabana, A. 1998. Dynamics of multibody system. Toinen versio. Yhdysvallat: Cambridge universitypress. 108-120 s.

Julia rajapinta Mevea ohjelmistoon

```
println("Waiting for connection")
server = listen(5111) # Avataan Modelerissa asetettu portti

conn = accept(server)

ninputs = 23 # Mevea solverista tulevat outputit
noutputs = 0  # Mevea solveriin menevät inputit

params=Array{Int32}(3)
params2=Array{Int32}(3)
params[1]=1
params[2]=noutputs
params[3]=ninputs
params2 = read(conn,Int32,3) #Tarksitetaan että Ratkaisiassa ja
#Juliassa on sama määrä out-ja inputteja.

write(conn,params)

ins=Array{Float64}(ninputs)
outs=Array{Float64}(noutputs)
All=Array{Float64}(ninputs)

println("Connection established")
while isopen(conn) #Kommunikointi

  try
    ins = read(conn,Float64,ninputs)
    append!(All,ins)
    #ins[1] Simulation time [s]
    #ins[2] Knife bar blade angle [deg(90,-90)]
    #ins[3] Knife bar blade center Global location x [m]
```

```
    #ins[4] Knife bar blade center Global location y [m]
    #ins[5] Log global location x [m]
    #ins[6] Log global location y [m]
    #ins[7] Log global location z [m]
    #ins[8] Log euler parameter e0
    #ins[9] Log euler parameter e1
    #ins[10] Log euler parameter e2
    #ins[11] Log euler parameter e3
    #ins[12] Lathe spindle R Global location z [m]
    #ins[13] Lathe spindle L Global location z [m]
    #ins[14] Lathe spindle R orientation z [deg[90,-90]]
    #ins[15] Lathe spindle L orientation z [deg[90,-90]]
    #ins[16] Lathe spindle R angular velocity z [rad/s]
    #ins[17] Lathe spindle L angular velocity z [rad/s]
    #ins[18] Round bar center Global location x [m]
    #ins[19] Round bar center Global location y [m]
    #ins[20] Back up roll1 Global location x [m]
    #ins[21] Back up roll1 Global location y [m]
    #ins[22] Back up roll2 Global location x [m]
    #ins[23] Back up roll2 Global location y [m]

    write(conn,outs)
  catch
    break
  end
end

println("Connection closed")
close(conn)
close(server)
```

Julia rajapinnan määrittely

*Taulukko II.1. Julian ja Mevea ohjelmiston välisen rajapinnan komponentit.*

| Signaalin nimi | Kuvaus | Suunta | Raja-arvot | Yksikkö |
|---|---|---|---|---|
| Time | Simulaation aika | Ulos | 0 – ∞ | s |
| Knife blade angle | Sorvausterän kulma globaalissa koordinaatistossa | Ulos | $-90^o$ – $90^o$ | deg |
| Knife blade center global location x & y | Sorvausterän keskikohdan globaali sijainti | Ulos | -∞ – ∞ | m |
| Knife blade left global location x, y & z | Sorvausterän vasemman kulman globaali sijainti | Ulos | -∞ – ∞ | m |
| Knife blade right global location x, y & z | Sorvausterän oikean kulman globaali sijainti | Ulos | -∞ – ∞ | m |
| Log global location x, y & z | Pöllin globaali asema | Ulos | -∞ – ∞ | m |
| Log euler parameter e0, e1, e2 ja e3 | Pöllin eulerin parametrit | Ulos | -1 – 1 | - |
| Lathe spindle R & L global location z | Karojen Z-akselin suuntainen asemas globaalissa koordinaatistossa | Ulos | -∞ – ∞ | m |
| Lathe spindle R & L orientation z | Karojen orientaatio | Ulos | $-90^o$ – $90^o$ | deg |
| Lathe spindle R & L angular velocity z | Karojen pyörimisnopeus | Ulos | -∞ – ∞ | deg/s |
| Round bar center global location x & y | Vastaterän asema globaalissa koordinaatistossa | Ulos | -∞ – ∞ | m |
| Back up roll 1 global location x & y | Ylemmän tukirullan asema globaalissa koordinaatistossa | Ulos | -∞ – ∞ | m |

*Taulukko II.1 jatkuu. Julian ja Mevea ohjelmiston välisen rajapinnan komponentit.*

| Signaalin nimi | Kuvaus | Suunta | Raja-arvot | Yksikkö |
|---|---|---|---|---|
| Back up roll 2 global location x & y | Alemman tukirullan asema globaalissa koordinaatistossa | Ulos | -∞ – ∞ | m |
| Lathe shaft L & R reaction moment | Puun inertiasta ja viilunleikkauksesta aiheutuva vastustava vääntömomentti | Sisään | -∞ – ∞ | Nm |
| Back up roll 1 reaction force x, y & z | Tukirullaan 1 kohdistuva kontaktivoima | Sisään | -∞ – ∞ | N |
| Back up roll 2 reaction force x, y & z | Tukirullaan 2 kohdistuva kontaktivoima | Sisään | -∞ – ∞ | N |
| Knife round bar reaction force x, y & z | Vastaterään kohdiastuva kontaktivoima | Sisään | -∞ – ∞ | N |
| Knife blade reaction force x, y & z | Terään kohdistuva kontaktivoima | Sisään | -∞ – ∞ | N |