Rodger Pušin

OTSESE JA PÖÖRDKINEMAATIKA KALKULAATOR KUUE VABADUSASTMEGA FANUC ROBOTITE JAOKS

LÕPUTÖÖ

Tehnikainstituut

Robotitehnika õppekava

Juhendaja: Elena Safiulina

Tallinn 2023

Autori deklaratsioon ja lihtlitsents

Mina,
Rodger Pušin,
tõendan, et lõputöö on minu kirjutatud. Töö koostamisel kasutatud teiste autorite, sh juhendaja teostele on viidatud õiguspäraselt.
Kõik isiklikud ja varalised autoriõigused käesoleva lõputöö osas kuuluvad autorile ainuisikuliselt ning need on kaitstud autoriõiguse seadusega.
Juhendaja Elena Safiulina, allkirjastatud digitaalselt.

Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks

Mina, Rodger Pušin
sünnikuupäev: 07.11.2000
annan Tallinna Tehnikakõrgkoolile (edaspidi kõrgkool) tasuta loa (lihtlitsentsi) enda loodud teose

OTSESE JA PÖÖRDKINEMAATIKA KALKULAATOR KUUE VABADUSASTMEGA FANUC ROBOTITE JAOKS

1. elektroonseks avaldamiseks kõrgkooli repositooriumi kaudu;
2. kui lõputöö avaldamisele on instituudi direktori korraldusega kehtestatud tähtajaline piirang, lõputöö avaldada pärast piirangu lõppemist.

Olen teadlik, et nimetatud õigused jäävad alles ka autorile ja kinnitan, et:

1. lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest tulenevaid ega muid õigusi;
2. PDF-failina esitatud töö vastab täielikult kirjalikult esitatud tööle.

Tallinnas, allkirjastatud digitaalselt.

SISUKORD

SISSEJUHATUS

Rohkem kui 100 aastat tagasi Tšehhi kirjanik Karel Čapek niiöelda leiutas roboti, kes oli inimese koopia. See oli inimese kunstlik koopia, mis ei omanud inimese tundeid ja parameetreid. Sellest sai kirjandusteoste peategelane. Pärast seda sai robotite temaatika väga populaarseks, robotid hakkasid paljunema suure kiirusega. Iga endast lugupidav ulmekirjanik hakkas kirjutama robotitest. Igaühel olid omad robotid. Selles, et neid oli palju, ei ole midagi erilist. Sest iga inimene saab omamoodi aru erinevatest mõistetest ja terminitest, mida omakorda teised inimesed tõlgendavad sageli erinevalt.

Mõne aja pärast hakkasid robotid ilmuma mitte ainult kirjanduses, aga ka projektides, tööjoonistel, teadusartiklites, nendega hakkasid tõsiselt tegelema insenerid. Ja kohe läksid asjad robotitehnikaga sassi.

Sel hetkel, millal robot oli ainult ulmekirjanduse tegelane, polnud kellelgi vaja teada termini „robot“ täpset definitsiooni või selle täpseid üksikasju. Ja kes vastutab selle eest, kellelt küsida ja kas on üldse vaja? Las igaüks mõtleb sellest omamoodi. Aga sellest hetkest, millal midagi saab reaalseks, hakkab tekkima tehnika-, majandus- ja teadushuvi, tahetakse kohe teada, kui palju reaalne asi erineb virtuaalsest. Kuidas näeb välja? Mida suudab teha? Kas kell on robot? Kas elektriline hakklihamasin on robot? On vaja proovida seda täpselt defineerida, sest tuleb välja, et inimesed tegelevad sellega, millest ise midagi ei tea.

20. sajandi keskel, umbes 20 aastat pärast Karel Čapeki teose ilmumist, on 156 teadlast välja mõelnud oma selgituse – mis on robot. Siin võitis Rooma meetod, mis väidab, et kui 7 inimest on üht asja öelnud, siis see on tõde. Lõpuks võitis selline definitsioon: „Robot on mobiilne süsteem, mis on võimeline „õppima“ ja leiab lühima tee juhuslikult paiknevate takistuste vahel, ilma kokkupõrgeteta, määratud sihtmärgini“. Sellise definitsioonini jõudsid 120 teadlast 156-st. [1]

Tööstusautomaatikaga seotud ettevõtted lahendavad palju tähtsaid ja olulisi ülesandeid tehes erinevaid projekte. Siia kuuluvad ülesandega tutvumine, alusmaterjalidega tutvumine, suhtlemine tellijaga, kontseptsiooni väljamõtlemine, projekteerimine, valmistamine/ehitamine, paigaldus, hooldamine, käit ja kasutaja koolitamine. Kõikides nendes projekti tegevustes on tähtis meetod ja süsteemsus, mis aitavad kiiremini ja kvaliteetsemalt lahendada ülesandeid. Selleks, et oleks paremini arusaadav, toon näite. Näiteks teil on projekt, kus on kasutuses mitu andurit, mitu ajamit ja kontrollerit. Selleks, et

täpselt teada ja planeerida kontrolleri versiooni, sisendite ja väljundite arvu, on vaja hakata projekteerima kasutades mingit meetodit. On võimalik võtta vihik ja panna kõiki need punktid kirja ja pärast anda see programmeerijale. Aga võib kulutada ka oma tööaega ja valmistada oma Exceli tabel, mis võimaldaks lihtsasti arvutada punkte, lisada kommentaare, lisada algus- ja lõpp-punkte kaabeldusele, jne.

Tänapäeval on üheks tähtsamaks suunaks robotitehnikas mitme vabadusastmega robotite analüüsimis- ja kontrollsüsteemide arendamine. See on tingitud sellest, et tööstus tunneb nälga robotite analüüsimis- ja kontrollsüsteemidest. Lahendus, mida tihti kasutavad minu kolleegid on mitte meetodi väljatöötamine, vaid uue tarkvara hange, mis tihti ei võimalda tekitada parameetrilist lahendust. Tööstusrobot on väga kallis investeering, aga tänu oma vabadusastmetele võib sellest saada väga parameetriline ja kasulik agregaat. Tänu sellele saab robotit piisavalt kiiresti ümber seadistada ja anda uue käsu, kui rääkida mingitest lihtsatest töödest nagu paletiseerimine või pakkimine.

Käesoleva bakalaurusetöö eesmärgiks on luua tarkvaraline meetod, mis aitaks ettevõttel E-431 OÜ piisavalt kiiresti projekteerida roboti integreerimine mingisse protsessi. Tarkvaralise meetme nimi on „Otsese ja pöördkinemaatika kalkulaator kuue vabadusastmega Fanuc robotite jaoks“. See võimaldab projekteerida Fanuc roboti üldpositsioone, kasutades koos sellega mingit CAD programmi, mis on juba ettevõttel olemas. Selles töös kasutatakse Solid Edge programmi. Minu rakendus on kirjutatud Visual Studio baasil C# keeles. See koosneb tulpadest, kuhu on vaja sisestada roboti parameetrid ja välja valida lõpp-punkt, kuhu roboti tööriist peab jõudma. Probleem, mida lahendab minu rakendus on aja ja raha sääst. Sest suurem osa protsesside programmeerimisest ei nõua midagi rohkemat kui robotipulti. Aga selleks, et valmistada roboti tööruumi ehk inglis keeles *Robot Cell*, on vaja liigutada virtuaalset robotit ja katsetada selle erinevaid positsioone.

Selleks, et valmistada sellist tarkvara, on vaja tegelda pöördinseneeriaga ja alustuseks saada aru rakendusmatemaatikast, mis on kasutuses selleks, et saada arvutada positsioone ja pöördenurki. Pärast seda on vaja valida platvorm, mis võimaldab valmistada mugava tarkvara. Ja lõpuks tuleb leida võimalus ettevõttes kasutatavas CAD programmis simuleerida robotit.

Selline meetod võib olla kasutuses selleks, et säästa aega ja raha. Aja säästmine toimub tänu sellele, et igakordsel roboti integreerimisel kasutab ettevõte samu väljatöötatud samme. Raha säästmine toimub tänu sellele, et programm on valmistatud lõpputöö raames koos lahtise koodiga ja ei maksa midagi.

Fanuc on üks liidritest robotite turul, neid kasutakse väga laialt üle maailma. Kindlasti on Fanucil oma tarkvara, aga lihtsamate protsesside programmeerimiseks ei ole seda lihtsalt mõtet osta, näiteks paletiseerimine. Aga keerulisemate protsesside jaoks on seda vaja eraldi õppida ja alati ei saa teha seda protsessi parameetriliseks ja ikkagi tuleb kasutada oma meetmeid. Hea näide keerulistest protsessidest on näiteks freesimine ja keevitamine. Roboti asukoha ehk paigalduspunkti kontrollimiseks saab kasutada lõputöös väljatöötatud tarkvara. See tähendab seda, et ei ole vaja üles laadida ja eksportida kõiki integreerimisega seotud agregaate teise tarkvara sisse. Allpool on toodud näide tarkvara kasutamisest Solid Edge programmiga.

Tarkvara on mitmeosaline. Esimene pool vastutab otsese kinemaatika arvutamise eest. Otsese kinemaatika ülesande tähistame FGM (*Forward Geometric Model*). Sel juhul on ülesandeks leida lõpp-positsioon millimeetrites ja orientatsioon nurgakraadides. Et see õnnestuks, on vaja teada manipulaatori roboti andmeid – kui kõrge on mingi osa või kui kaugel on üks osa teisest; kuidas on seotud oma vahel roboti liigeste koordinaadid ja kus need asuvad. Pärast seda võib koostada maatrikseid DH-Parameetrite (Denavit-Hartenbergi parameetrid) abil, korrutada iga liigese maatriks järgnevaga ja saada lõppmaatriks. [2]

Lõppmaatriksist peame saama lõpporientatsiooni Euleri nurkade abil. [3]

Teine pool tarkvarast on pöördkinemaatika osa ehk IGM (*Inverse Geometric Model*). Matemaatiliselt on see lahendatud geomeetria meetodiga. [4] See vajab nii roboti parameetreid kui ka lõpp-positsiooni koos nurkadega. Sisestades need, saame kõik vajalikud liigendite nurgad. Need nurgad, mida saame, ongi vaja sisestada CAD tarkvarasse selleks, et visuaalselt näha kuhu ja kuidas robot jõuab.

Tähtis on mitte unustada, et sisenditesse paneme mõõtmed millimeetrites ja nurgad kraadides. Programm ise konverteerib need radiaanidesse ja pärast tagasi.

1 ROBOTITE TÜÜBID JA BRÄNDID

20. sajandi keskel, umbes 20 aastat pärast Karel Čapeki teose ilmumist, on 156 teadlast välja mõelnud oma selgituse – mis on robot. Siin võitis Rooma meetod, mis väidab, et kui 7 inimest on üht asja öelnud, siis see on tõde. Lõpuks võitis selline definitsioon: „Robot on mobiilne süsteem, mis on võimeline „õppima“ ja leiab lühima tee juhuslikult paiknevate takistuste vahel, ilma kokkupõrgeteta, määratud sihtmärgini“. Sellise definitsioonini jõudsid 120 teadlast 156-st.

Nagu ulmekirjaduses nii ka täna tööstuses, kasutakse mitut robotitüüpi, täpsemalt kuut ja nimetakse neid erinevalt. Käesolevas lõputöös kasutatakse järgmisi nimetusi:

- Esimene robotitüüp – „Polaarsete koordinaatide robot“. Sellel robotil on tsentraalne pöörav võll ja pikendus- ja pöörlemisvõimalusega vars.
- Teine tüüp – „Silinderkoordinaatide robot“, see sarnaneb eelmisega, kuid erinevus seisneb varre liikumises. Selle tüübi vars liigub vertikaalselt libisedes mitte pöörates. Palju varasemad robotid kulusid selle tüüpi alla.
- Kolmas tüüp – „Descartes'i koordinaatidega robot“, see on väga täpne, libiseb kolme telje suunas lineaarselt ja ei pöörle. Meenutab arkaadkonsooli.
- Neljas tüüp, mis on kõige levinub tänu oma mobiilsusele on „Liigendrobot“. Struktuur sarnaneb inimese käega. Robot on väga parameetriline, aga seda on raskem programmeerida.
- Viies tüüp – „SCARA robot“ spetsialiseerub külgsuunalistele liikumistele. Pöördvõllid on vertikaalsed nii, et roboti haarats saab liikuda ainult horisontaalselt. Ainult viimane telg saab muuta oma vertikaalse positsiooni selleks, et suuta võtta haaratsiga detaili. Nende eelis on kergete detailide kiir liikumine.
- Kuues tüüp – „Paralleelrobotid“ ehk „Delta robotid“, mille haarats on kinnitatud paralleelse liigendühendusega. Tööpiirkond on piiratud, aga konstruktsioon võimaldab teha kiireid liigutusi.[1]

Ajakirja „*Technology“* andmetel kuuluvad kümne parema roboti brändi hulka järgmiste firmade looming: Mitsubishi Electric, Kawasaki, Epson, Universal Robots, Omron, Yaskawa Electric, Fanuc, KUKA, Denso, ABB. [5]

[1] https://robotics.kawasaki.com/ja1/xyz/en/1803-01/

Need brändid lähtuvad oma programmide valmistamisel erinevatest seisukohtadest. See tekitab suure mure uue projekti tegemisel. Näiteks teie ettevõte spetsialiseerub Kawasaki robotite paigaldamisele. Teie oskate hästi neid programmeerida, seadistada ja tööle panna tänu Kawasaki K-ROSET-le, mis võimaldab neid *offline* režiimis programmeerida ja testida virtuaalset robotit. Kõik teie elus on hästi, teie ei higista. Aga tekib projekt, mis on huvitav nii majanduslikult kui ka tehniliselt, projekt, mille kriteeriumiks on Fanuc roboti kasutamine. Ülesanne ei pea olema väga keeruline, aga ikkagi on nüüd teil vaja töötada uue roboti ja uue programmiga. Neid programme võib kasutada keskmise keerukusega projektides. Sest kui projekt on väga lihtne, siis tekib küsimus, miks ei saa lihtsalt käsitsi puldist programmeerida. Kui see on väga keeruline ja parameetriline, siis on vaja nii kui nii olla spetsialist kinemaatikas ja saata robotile punkte oma kontrollerist. Programmides võib näiteks genereerida punkte ebastandardse paleteerimise jaoks ja auto kere värvimiseks, seda juhul, kui ei ole plaani perspektiivis lisada uusi punkte või teha neid väga parameetrilisteks.

Selle töös tegeleme liigendrobotiga. Selleks, et paremini aru saada, kuidas sellest sai see mis sai, on vaja teada natukene selle ajaloost. Esimene agregaat, mis vähemalt imiteeris liigendrobotit, valmistas Raymond Goertz. Tema valmistas mehaanilise käe, mis töötas „juhi ja orja“ põhimõttel. Ühelt poolt oli võimalik käsitsi liigutada üht kätt ja teine käsi kordas sünkroonselt kõiki liigutusi [6]. Hakkasid tekkima uued ettevõtted, uued tehased ja lõpuks aastal 1983, Yaskawa, kes oli ja on Jaapani ettevõte, valmistas maailmas esimese kuue teljega roboti. [7] Selleks, et seda saaks teostada, valmistas Yaskawa uue kontrollerite generatsiooni nimega RX, mis esimesena võimaldas mällu salvestada 249 roboti tööd ehk eraldi programmi, 1200 eraldi instruktsiooni ehk loogikat ja 5000 positsiooni ehk robotpunkti. [8]

Töös kasutatakse Fanuc robotit, mis on ka Jaapanis toodetud. Nagu oli listis välja toodud, on Fanuc üks maailma liidreid robotitehnika valdkonnas. Ettevõte on valmistanud aastast 1956 erinevaid masinaid, mootoreid ja seadmeid. Aastast 1974 on Fanuc hakanud ise kasutama roboteid oma tehastes. Siin võib näha õiget strateegiat – Fanuc esialgu valmistas ja hakkas kasutama roboteid oma tehastes selleks, et neid paremini testida. Ja kolme aasta pärast, 1977, algas Fanuc ROBOT-MODEL 1 tootmine ja müük. [2]

[2] https://www.fanuc.co.jp/en/profile/history/index.html

Fanuc on välja töötanud programmi nimega ROBOGUIDE. Selles programmis on võimalik modelleerida nii roboti liikumist kui ka kogu tööprotsessi. Programmi võib kasutada nii *online* kui ka *offline* režiimis. On olemas ka programmi erinevad laiendusvõimalused.[3]

Online programmeerimise all on mõeldud programmeerimist otse robotiga töötades. Robot on vaja tööst vabastada ja programmeerida. Programmeeritakse läbi TP – *Teach Pendant*, eesti keeles – õpetamispult. *Offline* programmeerimine on kas teksti programmi kirjutamine ja robotisse laadimine või on see virtuaalse roboti programmeerimine ja pärast tema tööpunktide eksportimine. [9]

[3] https://www.fanuc.eu/uk/en?srb=1

2 KINEMAATIKA ANALÜÜS

2.1 Baasteadmised

Selleks, et kirjeldada tööstusroboti geomeetriat, kasutakse niinimetatud kinemaatilist skeemi, mis kujutab endast graafilisi jooniseid järjestikku olevatest manipulaatori osadest, mis on omavahel ühendatud.

Eristatakse kas baas või elementaarset liigenditüüpi ühe vabadusastmega: pöörlev ja lineaarne. Juhul, kui on esimest tüüpi, siis leiame osade positsiooni nurga muutuja abil, teise variandi puhul, nagu ka nimi ütleb, lineaarse nihke abil. Mõlemal juhul nimetatakse neid muutujaid üldistatud koordinaatideks. Lepime kokku, et üldistatud koordinaate tähistame tähega q. Selle alla kuuluvad siis pöörlev vabadusaste, mida tähistame θ-ga (Theta) ja lineaarne vabadusaste, mida tähistame d-ga.

Robotitehnikas on palju keerulisi masinaid mitme vabadusastmega ehk rohkem kui ühega. Sellistes olukordades reeglina vaadeldakse neid kombinatsioonis, mis omakorda võib koosneda nii θ kui ka d osadest. Kõikide üldistatud koordinaatide arvu, mis täpselt defineerib roboti, nimetatakse konfiguratsiooniks.

Roboti kinemaatilise analüüsi all mõeldakse kahe põhiülesande – otsese ja pöördkinemaatika ülesande – lahendamist.

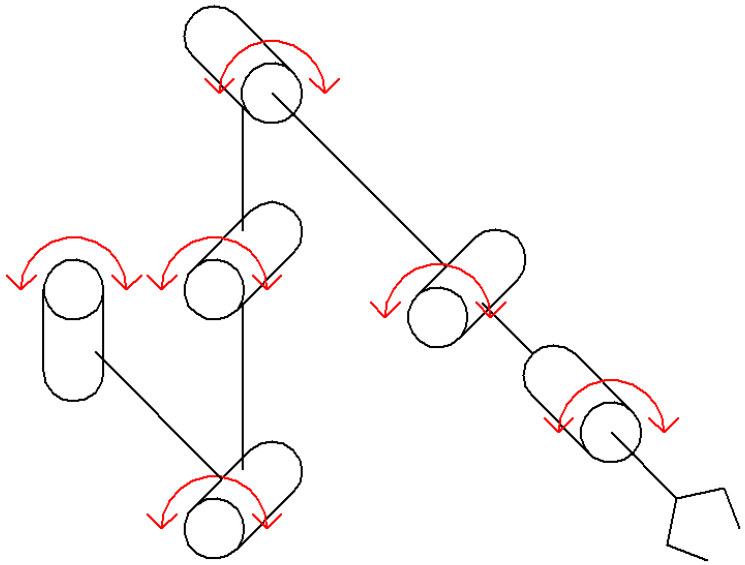
FGM ehk otsese kinemaatika ülesanne seisneb lõpp-punkti ehk TCP (*Tool Center Point*) koordinaatteljestiku positsiooni ja orientatsiooni leidmises, teades manipulaatori üldistatud koordinaatide arvu.

IGM ehk pöördkinemaatika ülesanne seisneb üldistatud koordinaatide muutujate leidmises, teades lõpp-punkti ehk TCP koordinaatteljestiku positsiooni ja orientatsiooni.

2.2 Kinemaatika otsene ülesanne

On teada, et keha positsioon maailmas on määratud kuue koordinaadiga. Kolm neist on lineaarsed (Descartes'i kordinaadid) ja kolm on nurgakoordinaadid (Euleri nurgad). [10]

Nagu oli juba mainitud, selles töös kasutatakse DH-Parameetreid. Need parameetrid lihtsustavad positsiooni leidmist ja kasutavad nelja parameetrit, kuue asemel. Selline lihtsustus tekib tänu süsteemile, mis seob koordinaattelje kokku roboti lülidega. Meid huvitab kuue vabadusastmega robot sfäärilise randmega. Sfääriline ranne tähendab seda, et roboti viimased kolm telge ristuvad ühes punktis. Kasutame null-konfiguratsiooni, kus kõik üldistatud koordinaadid võrduvad nulliga. Fanuc 710iC/50 kinemaatika skeemi null-konfiguratsioonis võib näha joonisel (Joonis 1). Punased kaared näitavad, et liigend saab pöörata nii paremale kui ka vasakule.



Joonis 1. Kinemaatika null-konfiguratsioon

DH-Parameetrite järgi koosneb lahendus järgmistest sammudest:

1) Koordinaatsüsteemide sidumine liigenditega;
2) DH-Parameetrite määramine;
3) Homogeensete ehk pöördmaatriksite konstrueerimine (Homogeneous Coordinates Jules Bloomenthal and Jon Rokne Department of Computer Science The University of Calgary);
4) Euleri nurkade arvutamine lõplikust pööramise maatriksist.

2.2.1 Koordinaatsüsteemide määramise algoritm

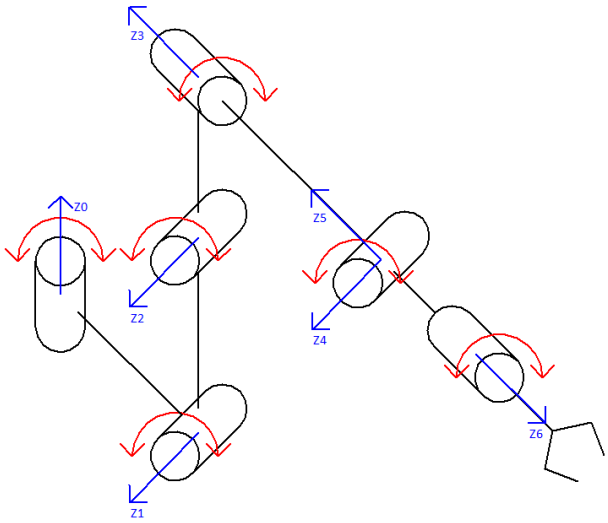
Toon välja ühe algoritmi variatsiooni, neid võib olla mitu, aga lõpuks peaksid need andma ühe kindla vastuse, tänu FGM ülesande selgesõnalisele lahendusele. Iga roboti telje keskpunktis on ühendatud kaks liigest, nimetame neid i_{-1} ja i. See tähendab seda, et kuue vabadusastmega robotil on seitse liigest, mis on nummerdatud nullist kuni kuueni, kus null-liigend sümboliseerib „maad“ ehk nullpunkti.

Selleks, et paremini aru saada, on vaja lisada, et iga i-liiges on jäigalt seotud oma koordinaatteljestikuga. Siis, kui i-liiges hakkab liikuma, tänu i-liigenduse tõttu, i-koordinaatteljestiku süsteem muudab oma positsiooni eelmise süsteemi suhtes.

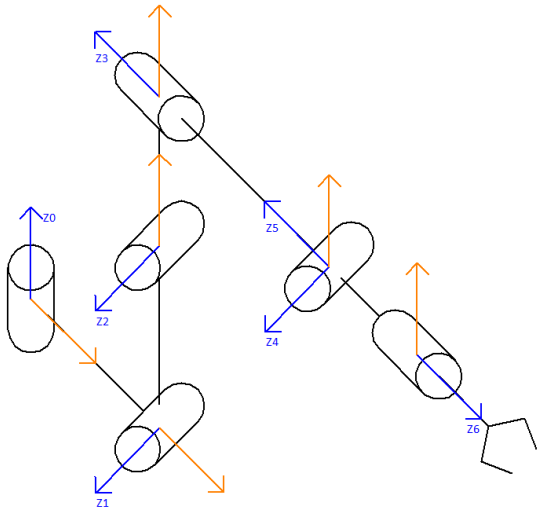
Vaatame läbi näidisalgoritmi, kuidas siduda i-liiges oma i-koordinaatteljestikuga. On vaja tähele panna, et see töötab kõikide liigendite jaoks, välja arvatud viimane. Lõplik koordinaatsüsteem valitakse eraldi allpool esitatud meetodiga. Üldistades algoritm koosneb kolmest sammust.

Valime telje Z_i, nii et see langeks kokku pöörlemisteljega või lineaarse liigendi liikumissuunaga. See tähendab, et iga liigendi suhtelise positsiooni saame defineerida kas Z-telje pööramisel või liikumisel lineaarselt. Selles töös kasutame robotit, mis on tehtud kuuest liigendist. Kasutades juba omandatud teadmisi, valime Z_i-teljed, mis langevad kokku pöörlemistelgedega nagu näidatud joonisel (Joonis 2).

X-telje valimisel on mitu huvitavat lähenemist. Võib leida arvamusi, et on vaja paigutada seda nii, et selle suund ristuks eelmise telje Z_i-ga. Aga see töötab hästi, siis kui meil on väljamõeldud, mitte reaalse robotiga ülesanne. Kui ülesandes on kasutusel reaalne robot, siis on vaja lihtsalt otsida õige paigutus, proovides erinevaid variatsioone. Kindlasti on üks põhikriteeriumitest see, et X_i-telg peab olema oma Z_i-teljega perpendikulaarne. Teisiti lihtsalt ei tohi olla. Teine kriteerium, mis ei tööta alati, aga võib töötada, on X_i-telje suunamine kas peale või eemale eelmisest koordinaatteljestiku keskpunktist. Arvutus algab igast koordinaatteljestiku keskpunktist, seal kus ristuvad Z_i- ja X_i-teljed. Järgmisel pildil on näha, kuidas standardsel Fanuc robotil on paigutatud X_i-teljed. Miks see on nii, näeme edaspidi, kui hakkame täitma DH-Parameetreid. X_i-telgede paigutus on näidatud joonisel (Joonis 3). Kõik X_i-teljed on joonestatud oranži värviga. Neljanda ja viienda koordinaatsüsteemi Y-teljed asuvad samas kohas nagu null-konfiguratsioonil.

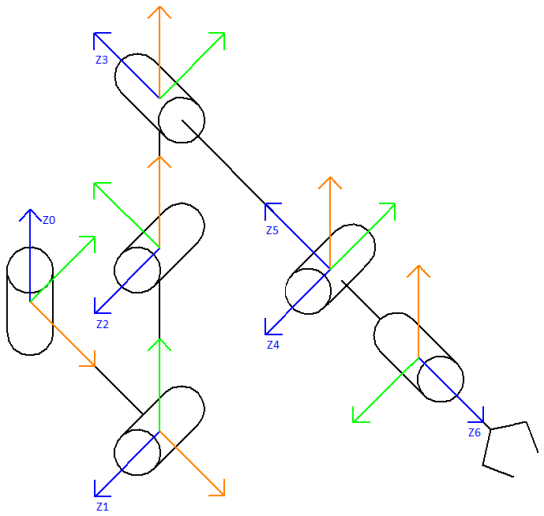


Joonis 2. Pöörded ümber liigendite Z_i-telgede



Joonis 3. Roboti liigendite X_i- ja Z_i-teljed

Valime Y_i-telje nii, et selle paiknemine vastaks vektorkorrutisele $Y = Z \times X$. Joonestame Z-telje kuue vabadusastmega roboti jaoks (Joonis 4). Y_i-teljed on joonestatud rohelise värviga. Neljandat telge X_4 ei ole joonestatud, sest see asub samas kohas, kus Z_5.



Joonis 4. Roboti liigendite X-, Y- ja Z-teljed

2.2.2 Denavit-Hartenberg Parameetrite määramine

Eelpool oli mainitud, et DH-Parameetrid võimaldavad vähendada koordinaatide arvu kuuest neljani. On olemas neli parameetrit:

- a_i – kaugus mööda Z_i-telge, Z_{i-1}-teljest Z_i-teljeni;
- α_i – nurk ümber X_i-telje, Z_{i-1}-teljest Z_i-teljeni;
- s_i – kaugus mööda Z_{i-1}-telge, X_{i-1}-teljest X_i-teljeni;
- θ_i – nurk ümber Z_{i-1}-telje, X_{i-1}-teljest X_i-teljeni.

Pöörame tähelepanu sellele, et parameetrid a_i ja α_i määratakse X_i-telgede ümber ja s_i ja θ_i määratakse Z_{i-1}-telgede ümber, see tähendab Z_i-teljest eelmise telje ümber. Selles meetodis on parameetrid a_i ja α_i konstantsed ja sõltuvad robotmanipulaatori mudelist. Ülejäänud s_i ja θ_i, sõltuvad ka manipulaatori mudelist. Roboti mingites osades võib üks neist olla konstantne ja teine mitte. Näiteks, kui teil on lineaarne liigend, siis s_i on parameetriline ehk muutuv, aga θ_i on konstantne. Meie Fanuc roboti korral, on kõik θ_i-d parameetrilised ehk muutuvad aga s_i-d on konstantsed, sest kõik liigendid on pöörlevad.

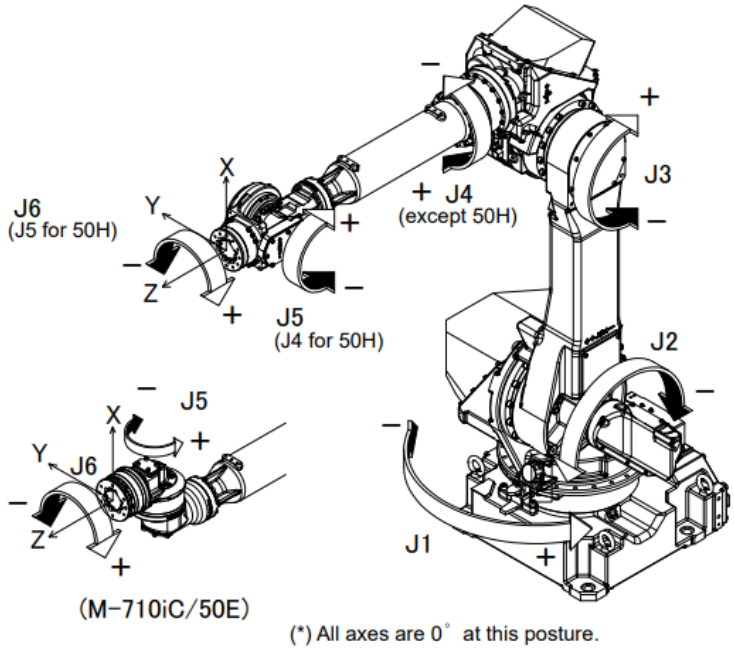
Robot Fanuc mudel M-710iC/50 DH-Parameetreid on esitatud tabelis (Tabel 1). Seal on eraldi ka toodud seitsmes liigend, mis hakkab tähistama TCP nihet kuuenda liigese keskpunktist.

Tabel 1. Fanuc M-710iC/50 DH-Parameetrite tabel

Liigend, i	$\boldsymbol{a_i}$ **(mm)**	$\boldsymbol{\alpha_i}$ **(deg)**	$\boldsymbol{s_i}$ **(mm)**	$\boldsymbol{\theta_i}$ **(deg)**
1	150	90	0	θ_1
2	870	0	0	$90 - \theta_2$
3	170	−90	0	$\theta_2 + \theta_3$
4	0	90	−1016	θ_4
5	0	−90	0	θ_5
6	0	180	−175	θ_6
7	0	−	0	−

Nagu on näha tabelist on roboti liigestel kolm nullilist a_i ja neli nullilist s_i parameetrit.

Ja lõpuks, nagu võib näha, saadakse θ_2 üldistatud nurk, kui 90°-st lahutatakse nurk θ_2. See on tavapärane Fanuci liigendi koordinaatide seos. Tänu sellele, kui liigutada teist *jointi*, liigub ka kolmas. Selleks, et lihtsamini aru saada, vaadake Fanuc originaaljoonist (Joonis 5). Näiteks, tahame liigutada $J2$ 15° võrra. Paneme 15° valemisse, siis $90^\circ - 15^\circ = 75^\circ$, pöörab 75° ette nagu on ka Fanucil joonega (Joonis 5) näidatud positiivne suund. On ka näha, et X_i-telgedel on kaks suunavariandi, kahel esimesel on need suunatud ette ja viiel ülejäänul on suunatud üles. Näeme ka, et teise ja kolmanda X_i-telgede vahel on nurk 90°, mida oligi vaja lisada eraldi tabelisse (Tabel 1).



Joonis 5. Fanuc M-710iC/50E liigendid [11]

2.2.3 Maatriksi homogeensed teisendused

Meenutame, et otsese kinemaatika ülesanne seisneb lõppliigendi või TCP koordinaatide ja nurkade leidmises. Selle lahendamisel kasutame baaskoordinaatsüsteemi, mis on seotud „maaga“ $X_0Y_0Z_0$ ja lõpusüsteemiga $X_iY_iZ_i$. Nende süsteemide seos määratakse kolme lineaarse ja kolme nurga koordinaadiga. Seda võib vaadata kui kahte koordinaatide kogumit, mis näitavad nihet ühest punktis teise.

Vaatame siis kaht koordinaatide kogumit. Nendeks las olla ühe ja sama maailma punkti koordinaadid k^0 ja k^i väljendatud süsteemide $X_0Y_0Z_0$ ja $X_iY_iZ_i$ kaudu. Saame valemi (1).

$$k^0 = T_i^0 \, k^i, \tag{1}$$

kus T_i^0 – lineaarse nihke informatsiooni kandev teisendus ja ühe süsteemi ruumiline orientatsioon teise suhtes.

$$T_i^0 = \begin{bmatrix} n_x & s_x & o_x & p_x \\ n_y & s_y & o_y & p_y \\ n_z & s_z & o_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} n_i^0 & s_i^0 & o_i^0 & p_i^0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} R_i^0 & p_i^0 \\ 0 & 1 \end{bmatrix}, \quad (2)$$

kus T_i^0 – lineaarse nihke informatsiooni kandev teisendus ja ühe süsteemi ruumiline orientatsioon teise suhtes.

n_i^0, s_i^0 ja o_i^0 – on vektorid, mis väljendavad telje suundi vastavalt $X_0Y_0Z_0$ koordinaatide suhtes;

R_i^0 – pööramise maatriks $X_iY_iZ_i$ koordinaatsüsteemi $X_0Y_0Z_0$ suhtes;

p_i^0 – lineaarne nihkevektor $X_iY_iZ_i$ koordinaatsüsteemi $X_0Y_0Z_0$ suhtes.

Selgitus: Maatriksit T_i^0 valem (2) nimetakse homogeenseks teisendusmaatriksiks, selle sees on koordinaatsüsteemide seos $X_0Y_0Z_0$ ja $X_iY_iZ_i$ vahel.

Nagu on näha maatriks on T_i^0 mõõtmetega $(4 \times 4ah)$, viimane neljas rida on lisatud selleks, et korrutada maatrikseid omavahel. Kui see oleks maatriks (3×4), siis seda ei saaks korrutada sama (3×4) maatriksiga. Seda võib lühidalt kirjutada nii nagu valemis (3).

$$\begin{bmatrix} x^0 \\ y^0 \\ z^0 \\ 1 \end{bmatrix} = T_n^0 \begin{bmatrix} x^n \\ y^n \\ z^n \\ 1 \end{bmatrix} \quad (3)$$

Kuidas leida baaspööramise maatriksit, on kirjeldatud allikas – (Robot modeling and control. Wiley New York, 2006.).

Oluline on suunata tähelepanu pööretele. Alustame nullpöördega, nullpöördele vastab identiteedimaatriks, valem (4).

$$R_{\beta=0} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = I \quad (4)$$

Pööre negatiivses suunas arvutatakse valemiga (5).

$$R_{-\beta} = R_\beta^{-1} = R_\beta^T \quad (5)$$

On olemas ka baaspööramise maatriksite valemid (6), (7), (8).

$$R_{x,\beta} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\beta & -\sin\beta \\ 0 & \sin\beta & \cos\beta \end{bmatrix}, \quad (6)$$

$$R_{y,\beta} = \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix}, \tag{7}$$

$$R_{z,\beta} = \begin{bmatrix} \cos\beta & -\sin\beta & 0 \\ \sin\beta & \cos\beta & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{8}$$

kus $\beta -$ mingi nurk.

Järjekordne pööramine ümber telgede määratakse paremalt poolt korrutamisega, ühe pööre maatriksi teiste pöörete maatriksite peale. Erinevatel robotitel on erinevad pööramise järjekorrad ja neid on päris mitu. Aga domineerivad on XYZ ja ZYX. Fanuc robot kasutab järjekorda ZYX, mida võib näha valemist (9).

$$R_{zyx} = R_{z,\theta_1} R_{y,\theta_2} R_{x,\theta_3} =$$

$$= \begin{bmatrix} \cos\theta_1 & -\sin\theta_1 & 0 \\ \sin\theta_1 & \cos\theta_1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta_2 & 0 & \sin\theta_2 \\ 0 & 1 & 0 \\ -\sin\theta_2 & 0 & \cos\theta_2 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta_3 & -\sin\theta_3 \\ 0 & \sin\theta_3 & \cos\theta_3 \end{bmatrix} =$$

$$= \begin{bmatrix} \cos\theta_1\cos\theta_2 & \cos\theta_1\sin\theta_2\sin\theta_3 & \cos\theta_1\sin\theta_2\cos\theta_3 \\ & -\sin\theta_1\cos\theta_3 & +\sin\theta_1\sin\theta_3 \\ \sin\theta_1\cos\theta_2 & \sin\theta_1\sin\theta_2\sin\theta_3 & \sin\theta_1\sin\theta_2\cos\theta_3 \\ & +\cos\theta_1\cos\theta_3 & -\cos\theta_1\cos\theta_3 \\ -\sin\theta_2 & \cos\theta_2\sin\theta_3 & \cos\theta_2\cos\theta_3 \end{bmatrix}, \tag{9}$$

kus θ_1 −nurk ümber Z-telje;

θ_2 −nurk ümber Y-telje;

θ_3 −nurk ümber X-telje.

Kui minna tagasi DH-Parameetrite juurde, tuletame meelde, et üle-eelmisel sammul oli saadud neli meile vajalikku parameetrit iga robot liigendi jaoks. Nüüd on nendest parameetritest vaja koostada eraldi maatriksid, valem (10).

$$T_i = T_{z,\theta_i} T_{z,d_i} T_{x,a_i} T_{x,\alpha_i} =$$

$$= \begin{bmatrix} R_{z,\theta_i} & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} I & p_{d_i} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} I & p_{a_i} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} R_{x,\alpha_i} & 0 \\ 0 & 1 \end{bmatrix} =$$

$$\begin{bmatrix} \cos\theta_i & -\sin\theta_i \cos\alpha_i & \sin\theta_i \sin\alpha_i & a_i \cos\theta_i \\ \sin\theta_i & \cos\theta_i \cos\alpha_i & \cos\theta_i \sin\alpha_i & a_i \sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{10}$$

kus $i -$ liigendi number;

$R_{z,\theta_i}, R_{x,\alpha_i} -$ baaspöörmise maatriksid, vaata valemeid (6) ja (8).

Osa maatriksitest koosnevad nullvektor komponentidest, valem (11).

$$p_{d_i} = \begin{bmatrix} 0 \\ 0 \\ d_i \end{bmatrix}, p_{a_i} = \begin{bmatrix} a_i \\ 0 \\ 0 \end{bmatrix}. \tag{11}$$

Nüüd saame koostada esimesed kuus maatriksit, kuue liigendi jaoks. Neid on vaja korrutada omavahel selleks, et leida kuuenda liigendi asukoha. Juhul, kui meie tahame kasutada mingit TCP-d koos selle parameetritega, siis on vaja korrutada kuus maatriksit, veel ühega, mis vastab TCP nihkele. See koosneb identiteedimaatriksis 3×3 ja kahe nullvektori komponentidest nihetega mööda X- ja Y-telge, valem (12).

$$T_t = \begin{bmatrix} 1 & 0 & 0 & a_t \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_t \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{12}$$

kus $a_t -$ TCP nihe X-teljel;

$d_t -$ TCP nihe Z-teljel.

Lõppmaatriksi, milles ühinevad kõik koordinaatteljestikud, saame korrutamise tagajärjel, valem (13).

$$T_n^0(q) = T_1(q)T_2(q) \dots T_n(q)T_t(q) = \begin{bmatrix} R_n^0(q) & p_n^0(q) \\ 0 & 1 \end{bmatrix}, \tag{13}$$

kus $R_n^0(q) -$ pööramise maatriks.

2.2.4 Euleri nurkade arvutamine

Otsene kinemaatika ülesanne – on leida lineaar- ja nurgakoordinaadid $o_n x_n y_n z_n$, mis kirjeldavad haaratsi või mõne muu tööorgani asendit baaskoordinaatsüsteemi $o_0 x_0 y_0 z_0$ suhtes. Eelmisel sammul saime maatriksi $T_n^0(q)$, mille sees on pööramise maatriks $R_n^0(q)$ ja vektor $p_n^0(q)$. Kui rääkida

lineaarsetes ehk Descartes'i süsteemi koordinaatides, siis vektori $p_n^0(q)$ sisu on esitatud valemiga (14). Need komponendid on FGM ülesande konkreetne lahendus positsiooni suhtes.

$$p_n^0(q) = \begin{bmatrix} x_n^0(q) \\ y_n^0(q) \\ z_n^0(q) \end{bmatrix}. \tag{14}$$

Olemas olev pöörete maatriks $R_n^0(q)$ üheksa elemendiga ei ole mugav, sest see ei anna alati võimalust määrata, millise orientatsiooni tema seab. Tekib vajadus leida neid nurki korrektsemalt, nii, et meie saaksime kolm arvu ehk kolm pöördenurka ümber kolme telje.

Nagu oli mainitud eelnevalt, on erinevatel robotitel erinevad pööramise järjekorrad. DH-Parameetrid töötavad nelja parameetriga, aga lõpuks saab lõpp-punkt kõik kuus parameetrit. Kolm nendest vastutavad positsiooni ja kolm orientatsiooni eest. See tähendab, et edasi meie võrdsustame korrutamisel saadud pöörete maatriksi pööramise järjekorra maatriksiga ZYX, valem (15).

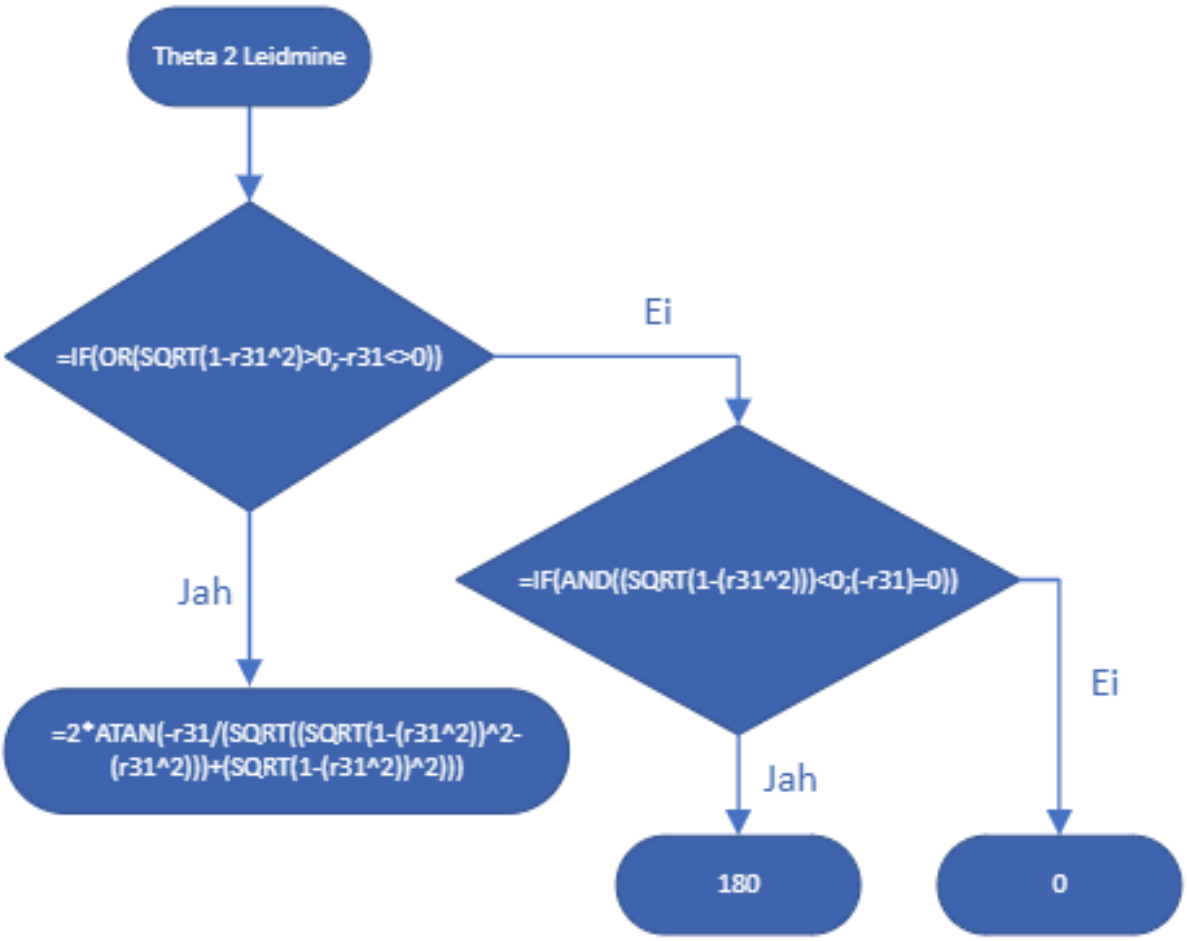
$$R_n^0(q) = R_{zyx} =$$

$$= \begin{bmatrix} r_{11}(q) & r_{12}(q) & r_{13}(q) \\ r_{21}(q) & r_{22}(q) & r_{23}(q) \\ r_{31}(q) & r_{32}(q) & r_{33}(q) \end{bmatrix} =$$

$$= \begin{bmatrix} \cos\theta_1 \cos\theta_2 & \cos\theta_1 \sin\theta_2 \sin\theta_3 - \sin\theta_1 \cos\theta_3 & \cos\theta_1 \sin\theta_2 \cos\theta_3 + \sin\theta_1 \sin\theta_3 \\ \sin\theta_1 \cos\theta_2 & \sin\theta_1 \sin\theta_2 \sin\theta_3 + \cos\theta_1 \cos\theta_3 & \sin\theta_1 \sin\theta_2 \cos\theta_3 - \cos\theta_1 \cos\theta_3 \\ -\sin\theta_2 & \cos\theta_2 \sin\theta_3 & \cos\theta_2 \cos\theta_3 \end{bmatrix}. \tag{15}$$

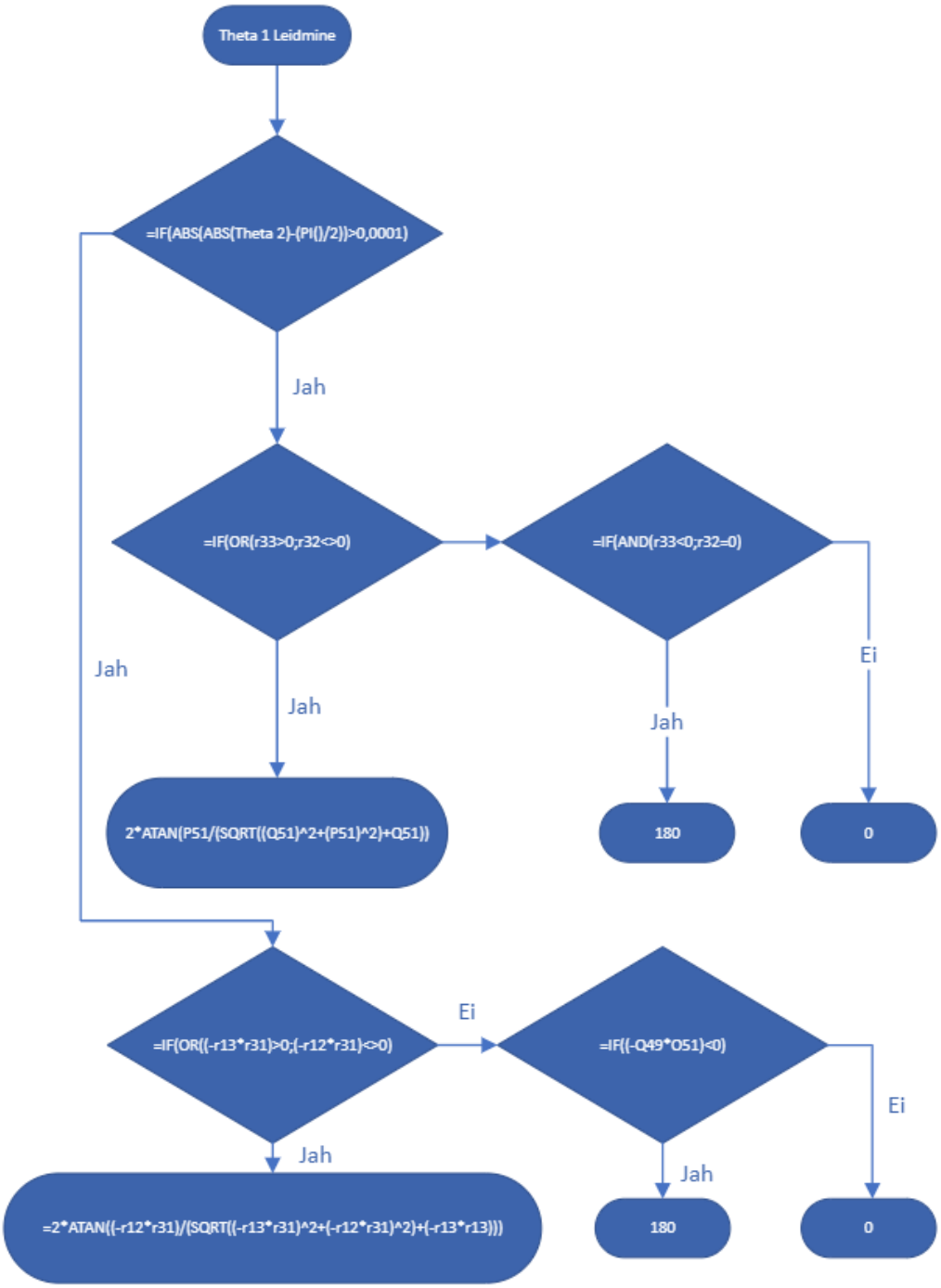
Ülesanne seisneb selles, et määrata nurgad $\theta_1(q), \theta_2(q), \theta_3(q)$ antud maatriksi $R_n^0(q)$ abil. UML skeemil on kasutatud Excel-loogika süntaksit. Matemaatika poolel on kasutatud ATAN2 funktsiooni, mida ei peaks lahti kirjutama, vaid kasutada Exceli standardfunktsiooni (=ATAN2). Siin on seda tehtud selleks juhuks, kui programmis, milles arvutate, ei ole olemas seda funktsiooni, siis on vaja kasutada, nagu on näidatud joonistel, kompleksargumenti. [12]

Esialgu on vaja arvutada teine pöördenurk $\theta_2(q)$ ehk Theta 2. Vastuse saame radiaanides, mille teisendame kraadidesse. Skeemil (Joonis 6) on näidatud kaks konkreetset lõpptulemust kraadides: 180 ja 0.



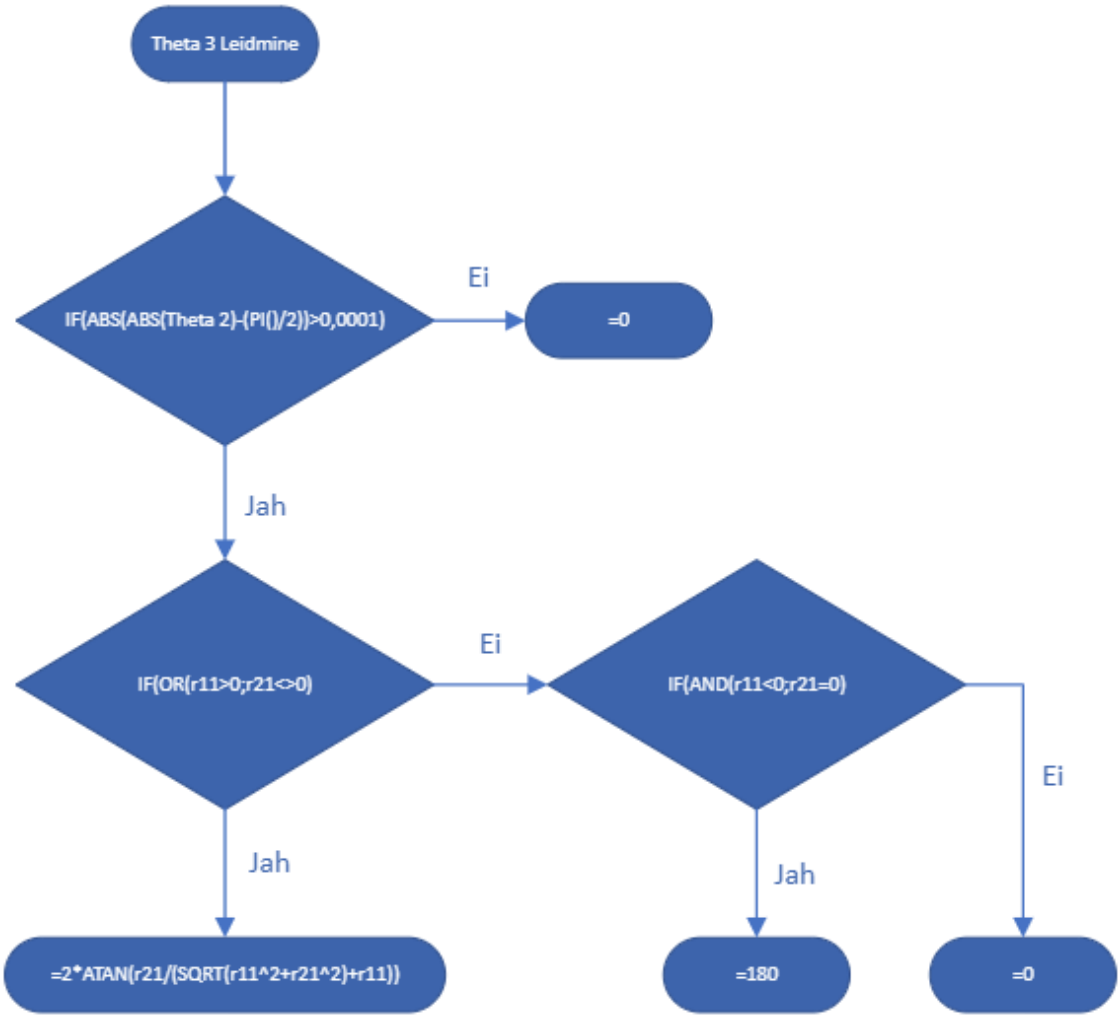
Joonis 6. Pöördenurga Theta 2 arvutamine

Järgmisena hakkame arvutama $\theta_1(q)$ ehk pöördenurka Theta 1. See on otseses sõltuvuses pöördenurgast Theta 2 ja algsuuruseks võtame Theta 2 absoluutväärtuse (Joonis 7).



Joonis 7. Pöördenurga Theta 1 arvutamine

Ja viimasena arvutame pöördenurga Theta 3, selle leidmise algoritmi on esitatud joonisel (Joonis 8). See lahendus on võetud NASA materjalidest. [13]



Joonis 8. Pöördenurga Theta 3 arvutamine

2.3 Kinemaatika pöördülesanne

Kinemaatika pöördülesanne ehk IGM on keerulisem kui FGM, sest sel võib olla mitu lahendust. See tähendab, et erinevad roboti konfiguratsioonid võivad anda sama lõpptulemuse. Robot võib olla pööratud kas vasakule või paremale, aga mõnel juhul jõuda samasse lõpp-punkti. Lisaks sellele on IGM vägagi sõltuvuses roboti konfiguratsioonist ehk selle geomeetrilistest parameetritest. See tähendab, et ei saa kasutada universaalset koodi kõikide robotite jaoks. Praegu iga roboti mudel on unikaalne ja sellega on vaja arvestada. Nagu mainitud, ülesande mõte on leida üldistatud koordinaadid, teades lõpp-punkti lineaar- ja nurgakoordinaate. Algandmetesse kuuluvad:

- p_n^0, kolm lineaarkoordinaati;
- kolm nurgakoordinaati, näiteks kolm Euleri nurka $\theta_1, \theta_2, \theta_3$. Tänu neile saab tekitada R_n^0;

- fikseeritud DH-Parameetrid, mis sõltuvad robotmanipulaatori konstruktsioonist.

Geomeetriline ehk analüütiline meetod IGM-i lahendamiseks seisneb analüütiliste võrrandite koostamises, kasutades trigonomeetria funktsioone, mis sõltuvad roboti manipulaatori konstruktsioonist. Kuue vabadusastmega sfäärilisega randmega robotid on laialt kasutuses tänu selle headele funktsionaalsetele võimalustele.

2.3.1 Pöördenurga Theta 1 arvutamine

Alustuseks peame teadma, milline on meie lõpp-punkt, kuhu meie tahame jõuda. Selle lõpp-punktiga on seotud eelmise koordinaatteljestiku keskpunkt, viimane koordinaatteljestik on sellest eemal s_6 võrra. Konkreetselt on selle töö raames see nihe s_6 võrdne 175 millimeetriga. Tekitame esialgu uue DH-Parameetrite maatriksi nimega Mtx_1, mis kajastab positsiooni ja orientatsiooni. See osa, mis vastutab orientatsiooni eest, peaks oma konfiguratsioonis kattuma R_{zyx}, pööramise järjekorraga, valem (9).

Samas tekitame ka maatriksi nimega Mtx_2, mis hakkab vastutama TCP nihke eest, valem (16).

$$Mtx_2 = \begin{bmatrix} 1 & 0 & 0 & -(a_7) \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -(s_7) \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{16}$$

Seejärel on vaja need maatriksid korrutada. Tähtis on ka korrutada õiges järjekorras, valem (17).

$$Mtx_1 \times Mtx_2 = Mtx_3. \tag{17}$$

Järgmisena tekitame maatriksi Mtx_4 $[4 \times 1]$, mis kajastab s_6 väärtuse positsiooni, valem (18).

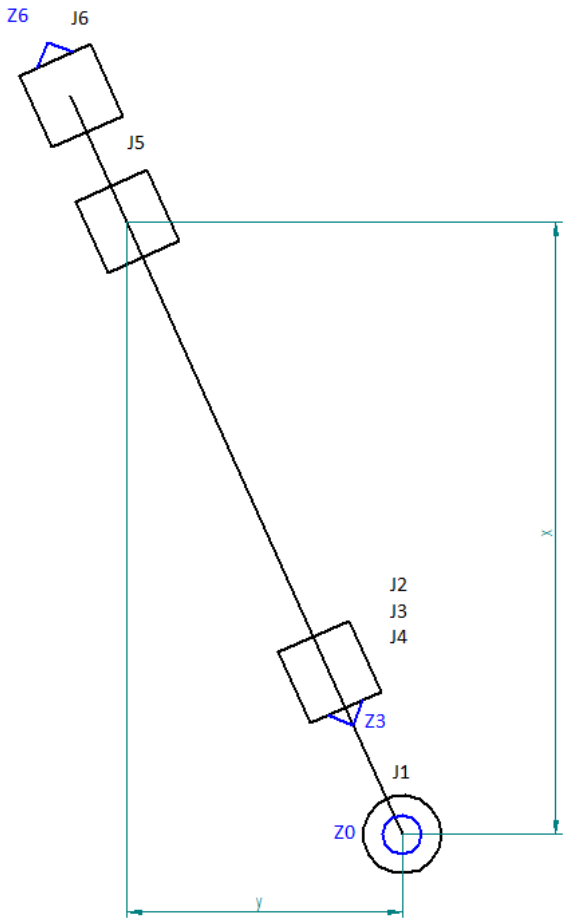
$$Mtx_4 = \begin{bmatrix} 0 \\ 0 \\ s_6 \\ 1 \end{bmatrix}. \tag{18}$$

Viimane samm viienda koordinaatteljestiku keskpunkti leidmiseks on veel üks korrutamine. On vaja korrutada Mtx_3 maatriksiga Mtx_4, mille tulemusena tekib uus maatriks Mtx_5, mõõtmetega $[4 \times 1]$ valem (19). Sellest saame konkreetsed viienda koordinaatteljestiku positsiooni koordinaadid.

$$Mtx_5 = \begin{bmatrix} x_5 \\ y_5 \\ z_5 \\ 1 \end{bmatrix}. \tag{19}$$

Pöördenurga *Theta* 1 arvutamine on tänu kuue vabadusastmega manipulaatori konstruktsioonile lihtne ja piisavalt täpne. Sellised manipulaatorid, kui vaadata nende peale ülevalt, on alati pööratud sinna, kus asub viienda telje koordinaatteljestiku keskpunkt, X,Y-tasapinnal (Joonis 9). Pöördenurga leiame ATAN2 funktsiooni abil, Excel-süntaksiga valemiga (20). [10]

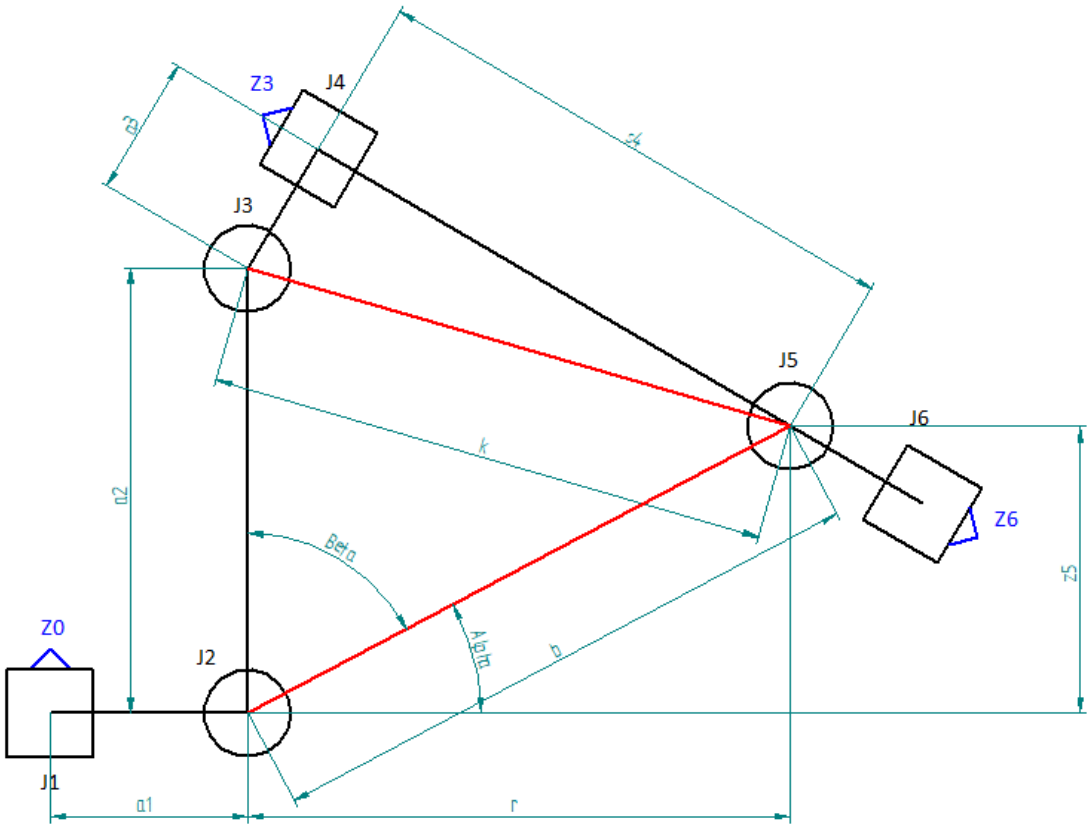
$$Theta\ 1 = ATAN2(x_5; y_5). \tag{20}$$



Joonis 9. Viienda liigendi keskpunkti asukoht X, Y- tasapinnal

2.3.2 Pöördenurga Theta 2 arvutamine

Alustame joonisest, millel on esitatud kõik meile vajalikud distantsid (Joonis 10).



Joonis 10. Roboti kinemaatikaskeemi külgvaade 1

Esiteks leiame J2 ja J5 vahelise kauguse *r*. Ülevalt vaadates *r* moodustab täisnurkse kolmnurga kaateti osa. Osa, sest meid ei huvita a_1 pikkus, see on meil olemas. Leiame hüpotenuusi ja lahutame sellest a_1, valem (21).

$$r = \sqrt{{x_5}^2 + {y_5}^2} - a_1. \tag{21}$$

Teiseks leiame *b* pikkuse. Selle leidmiseks sobib vektori mooduli pikkuse valem (22). Selleks kasutame *J2* leitud positsiooni. Lihtsaim võimalus seda leida, on panna leitud Theta 1 ja DH-Parameetrid maatriksi õigetesse kohtadesse valemis (10).

$$|\bar{b}| = \sqrt{(x_5 - x_2)^2 + (y_5 - y_2)^2 + (z_5 - z_2)^2}. \quad (22)$$

Kolmandaks sammuks on *k* leidmine. Selle leidmiseks kasutame Pythagorase teoreemi, kaatetiteks saavad on a_3 ja s_4.

Viimaseks sammuks on muutujate *Alpha* ja *Beta* leidmine. Alustame *Alpha*-st, selle leidmiseks kasutame Exceli-süntaksis ATAN funktsiooni valemit (23).

$$Alpha = ATAN\left(\frac{z_5}{r}\right). \quad (23)$$

Beta leidmine on natuke keerukam, selle leidmiseks kasutan koosinusteoreemi valemit (24).

$$Beta = \cos^{-1}\left(\frac{{a_2}^2 + b^2 - k^2}{2 \cdot a_2 \cdot b}\right). \quad (24)$$

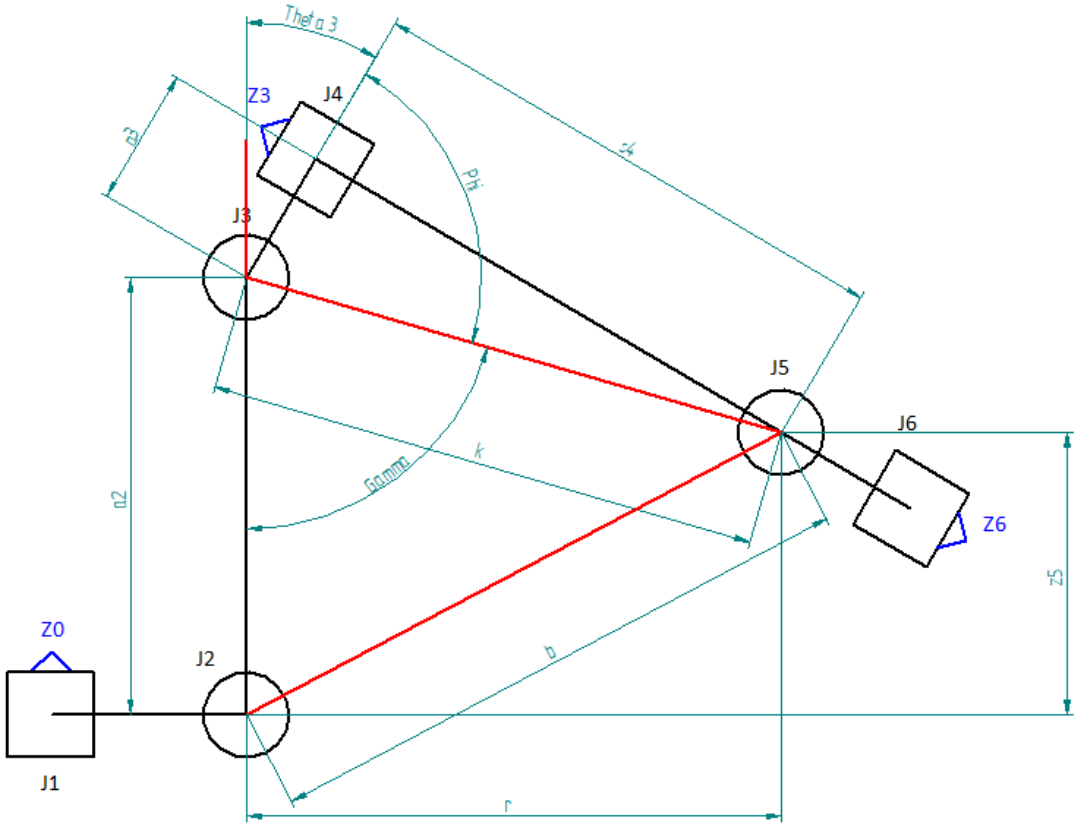
On jäänud leida Theta 2. Selle leidmisel on vaja arvestada otsese kinemaatika Theta 2 konfiguratsiooniga, sest ka siin on vaja 90°-st lahutada *Beta* ja *Alpha*, valem (25).

$$Theta\ 2 = \frac{PI()}{2} - Beta - Alpha. \quad (25)$$

2.3.3 Pöördenurga Theta 3 arvutamine

Pöördenurga Theta 3 arvutamist alustame koosinusteoreemiga nurga *Gamma* leidmisest, valem (26) ja joonis (Joonis 11).

$$Gamma = \cos^{-1}\left(\frac{{a_2}^2 + k^2 - b^2}{2 * a_2 * k}\right). \quad (26)$$



Joonis 11. Roboti kinemaatikaskeemi külgvaade 2

Järgmisena on meil vaja analüüsida robotit, millega töötame. Juhul kui $a_3 \neq 0$, siis meil on vaja arvutada *Phi* väärtus, valem (27).

$$Phi = \tan^{-1}\left(\frac{-s_4}{a_3}\right). \tag{27}$$

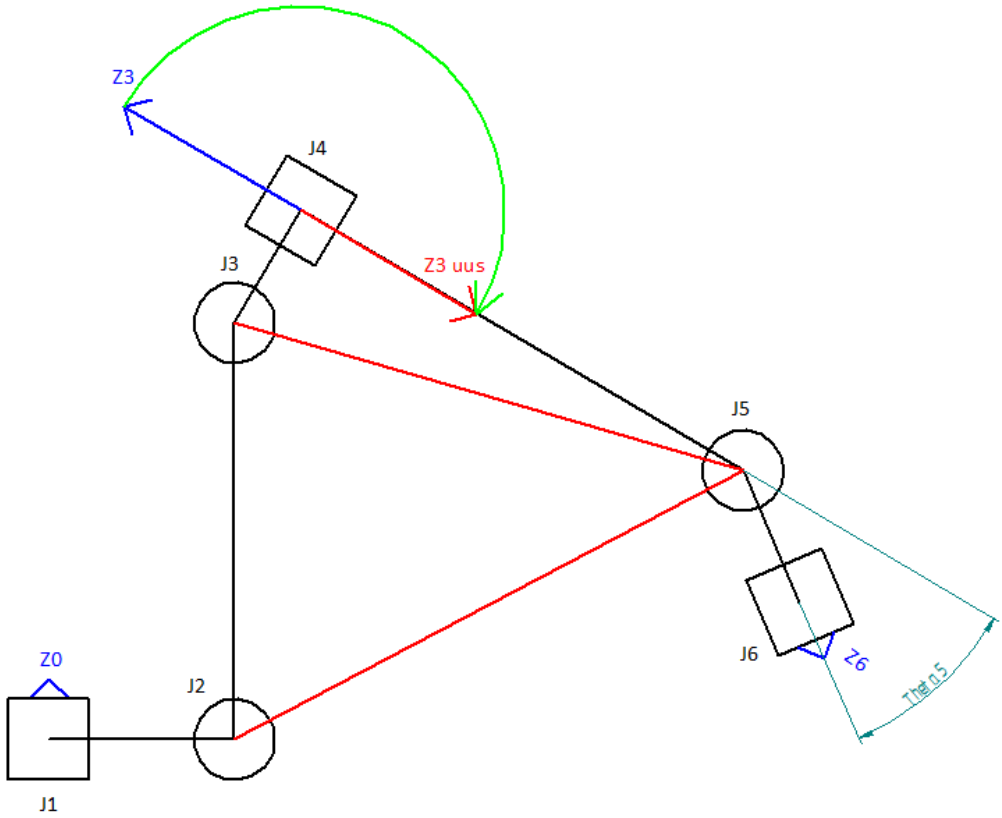
Arvutame Theta 3, selleks on vaja arvestada otsese kinemaatika Theta 3 konfiguratsiooniga, sest ka siin on vaja summeerida nurki Theta 2-ga, valem (28).

$$Theta\ 3 = -(\pi - Gamma - Phi + \theta_2) \tag{28}$$

2.3.4 Pöördenurga Theta 5 arvutamine

Nüüd meil on arvutatud kolm esimest pöördenurka. Meie järgmine samm on tekitada kolm DH-Parameetritega maatriksit vastavalt valemile (10), täites need saadud pöördenurkadega. Pärast seda

võtame saadud maatriksi osa 3×3, mis vastutab pööramise eest ja korrutame selle pöörete maatriksiga, mis koosneb $Z\ ja\ X$ pööretest. Selle maatriksi võtame DH-Parameetrite maatriksist, ka pööramisosa 3×3 ja α pöördenurgaks paneme π. See liigutus pöörab Z_3 vektorit 180°võrra (Joonis 12).



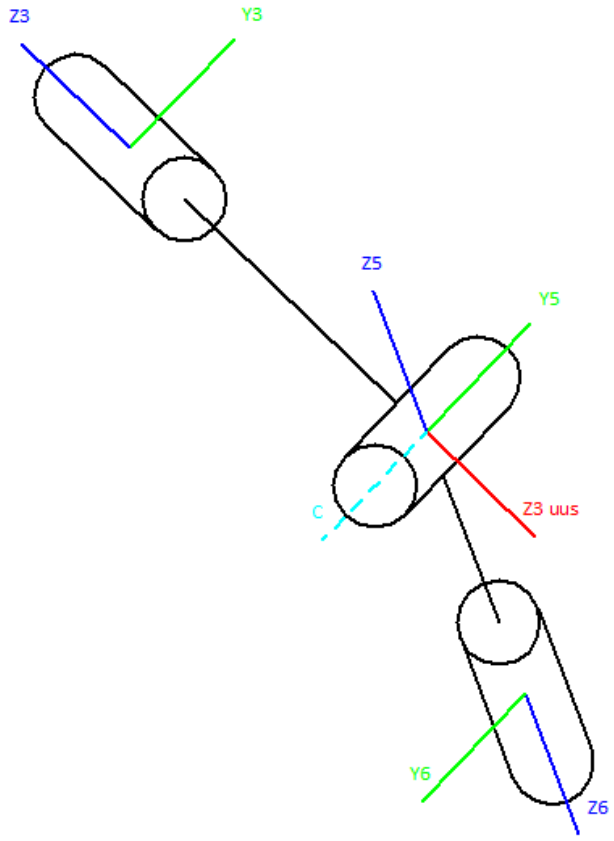
Joonis 12. Z_3 pööramissuuna muutmine

Kasutame ka maatriksit Mtx_1, sellest leiame Z orientatsiooni. Nüüd selleks, et leida pöördenurka Theta 5, kasutame kahe vektori skalaarkorrutist, valem (29), selles kasutame pööratud Z ja etteantud Z_6. Kui leiame Theta 5, siis dubleerime seda teise vastusega, aga miinusmärgiga. Edasi toimub analüütiline valik, mis nendest kahest sobib ja mis mitte.

$$Theta\ 5 = \left(\frac{z_3 \cdot z_6}{|z_3| \cdot |z_6|}\right). \tag{29}$$

2.3.5 Pöördenurga Theta 4 arvutamine

Pöördenurga Theta 4 arvutamiseks on vaja vektorkorrutise abil tekitada uus vektor C, (Joonis 12). Korrutame Z_6 orientatsiooni, pööratud ehk uue Z_3 orientatsiooniga. Edasi leiame pöördenurga m, kasutades skalaarkorrutise valemit (29). Selleks korrutame Y_3 vektori saadud C vektoriga. Theta 4 saame, lahutades π-st saadud pöördenurga m.

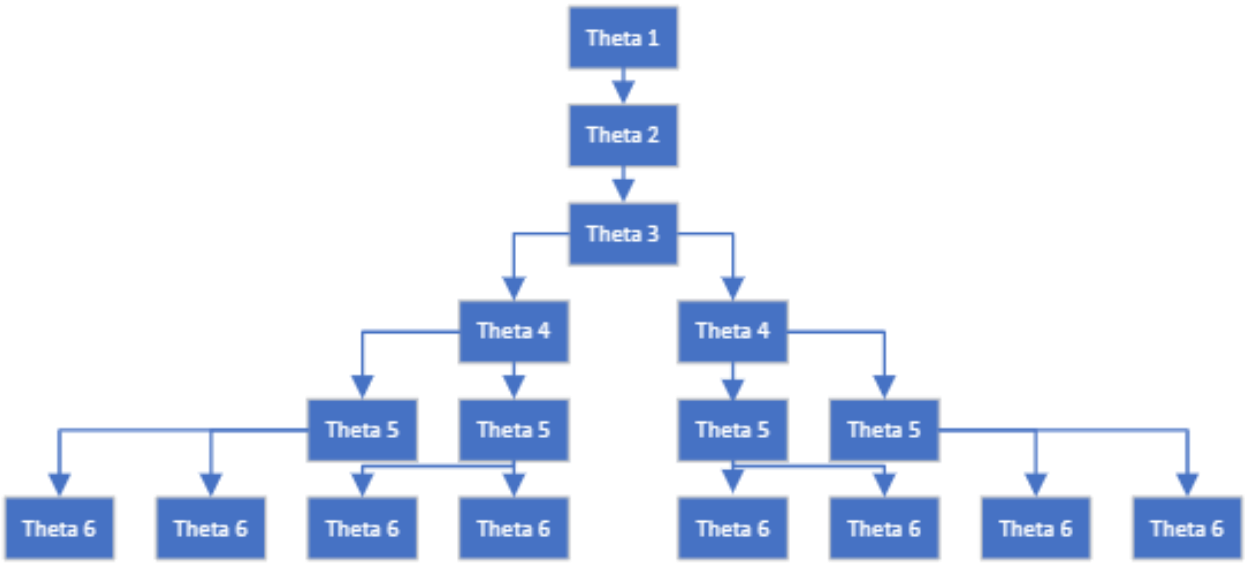


Joonis 13. Lisa C vektori arvutamiseks

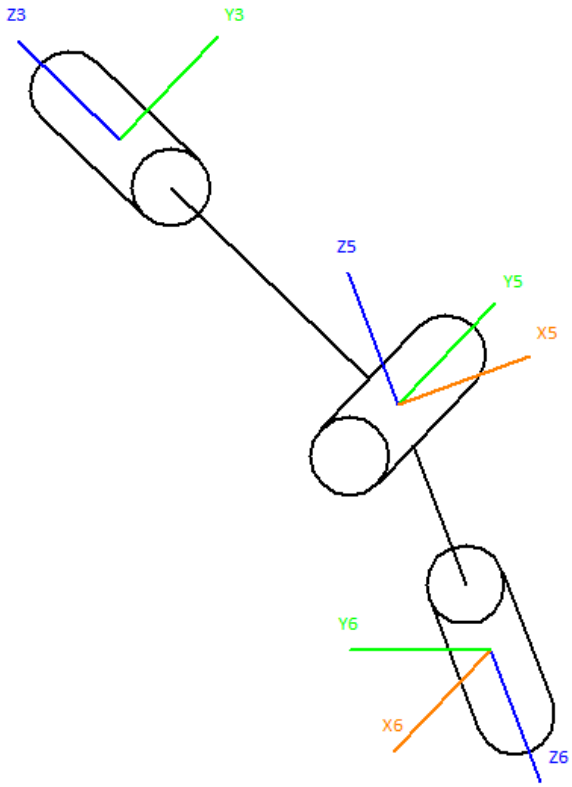
2.3.6 Pöördenurga Theta 6 arvutamine

Kui pöördenurkadel Theta 4 ja Theta 5 võis olla kaks vastust ja robotitehnikul oli vaja valida, millist kahest tema kasutab, siis Theta 6 puhul on vaja valida 8 vastuse vahel. Miks tekib selline vastuste pluralism? Visuaalne vastus on esitatud joonisel (Joonis 14) ja see on tingitud sellest, et meie peame

jälle formeerima viis maatriksit ja korrutama neid üksteisega. Aga erinevus on selles, et tänu IGM-i mittekonkreetsetele vastustele ja valikute vajadusele, on vaja korrutada esimene, teine ja kolmas maatriks kõikide võimalike ülejäänud maatriksite nurkadega. Lõpuks tekib neli võimalikku X_5-telje orientatsiooni, mida peame, kasutades skalaarkorrutise valemit, võrdlema X_6-telje orientatsiooniga ja leidma nendevahelise nurga (Joonis 15). Seejärel lisandub igale neljast leitud nurgale veel üks miinusmärgiga nurk. Nii saamegi kaheksa võimaliku pöördenurka Theta 6-le.

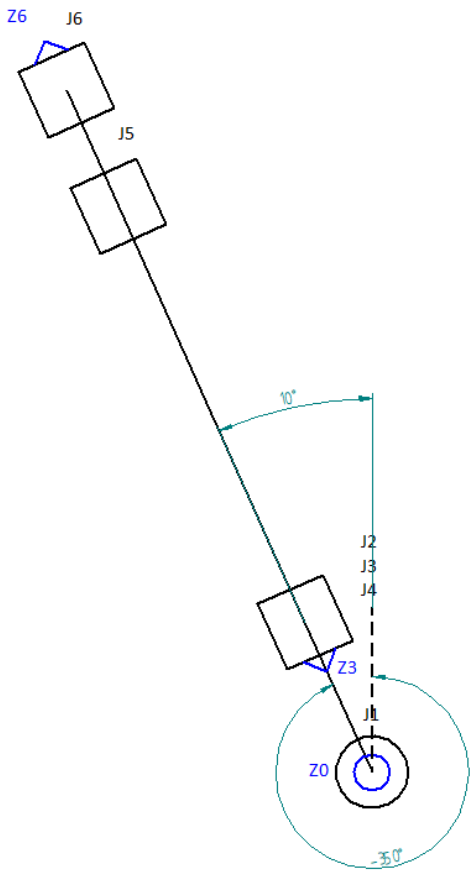


Joonis 14. Pöördenurga Theta 6 mitmed võimalikud vastused



Joonis 15. Pöördenurga Theta 6 arvutamine X_5- ja X_6-telgede vahel

Pärast kõikide nurkade leidmist, võib robotitehnik kasutada FGM kalkulaatori rakendusosa selleks, et valida õige nurkade konfiguratsioon. Seejuures kasutab robotitehnik kalkulaatorit abitööriistana, mitte dogmana. Mõnikord FGM ja IGM annavad erineva tulemuse. Milles võib olla põhjus? Vaatame tihti tekkivat probleemi keeruliste punktide puhul. Üks kalkulaatori osa näitab pöördenurga Theta 1 suuruseks 10° ja teine näitab nurka -350°. See ei tähenda, et üks eksib, see tähendab, et on vaja analüüsida vastuseid ja leia sobiv variant. Sest lõpuks robot jõuab TCP-ga sama lõpp-punkti, vahe on ainult selles, kas robotil on vaja pöörata kümme kraadi või on mõtet pöörata 350 kraadi ja lisada mingi liigutus, mida teha sinna jõudmise aja vältel (Joonis 16).



Joonis 16. Võimalikud pöörded

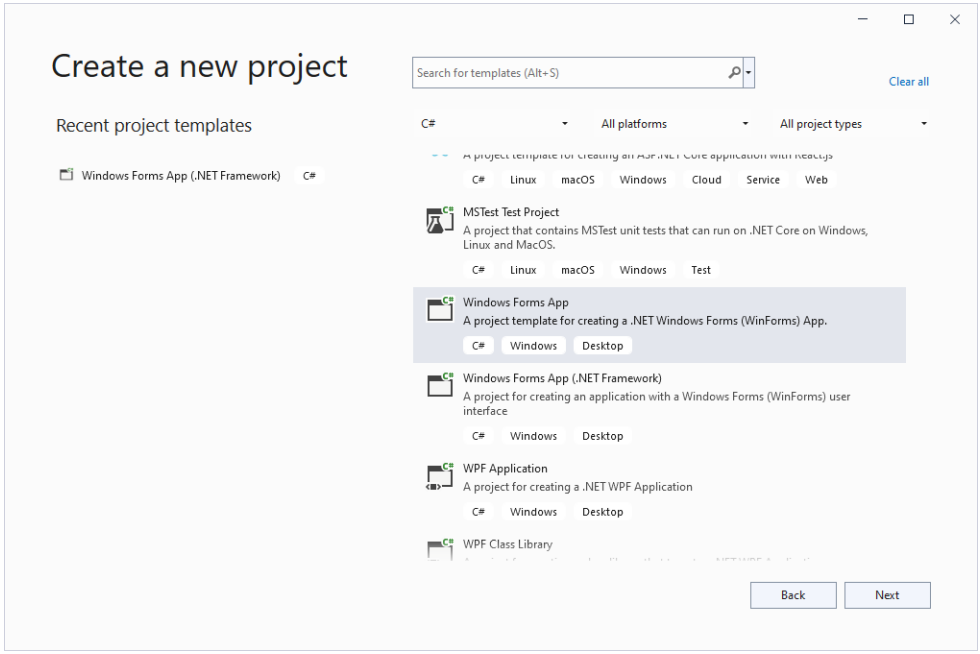
3 KALKULAATORI PROGRAMMEERIMINE

Kalkulaatori loomiseks on valitud arenduskeskkond Visual Studio. Seda võib kasutada koodi kirjutamiseks, redigeerimiseks, kokkupanemiseks ja parandamiseks. Lõpuks, kui rakendus on valmis, võib selle postitada. Võib valmistada rakendusi erinevatele robotitüüpide. Keskkond omab abivahendeid probleemide kiiremaks lahendamiseks ja abiks.[4]

Kõik eeltoodud keskkonna plussid olid põhjuseks, miks sai valitud kalkulaatori valmistamiseks Visua Studio. Raudne põhjus on ka see, et programmi saab kasutada tasuta. Rakendus on kirjutatud C# keeles. C# on kaasaegne ja objektorienteeritud programmeerimiskeel.[5]

3.1 Kujundus

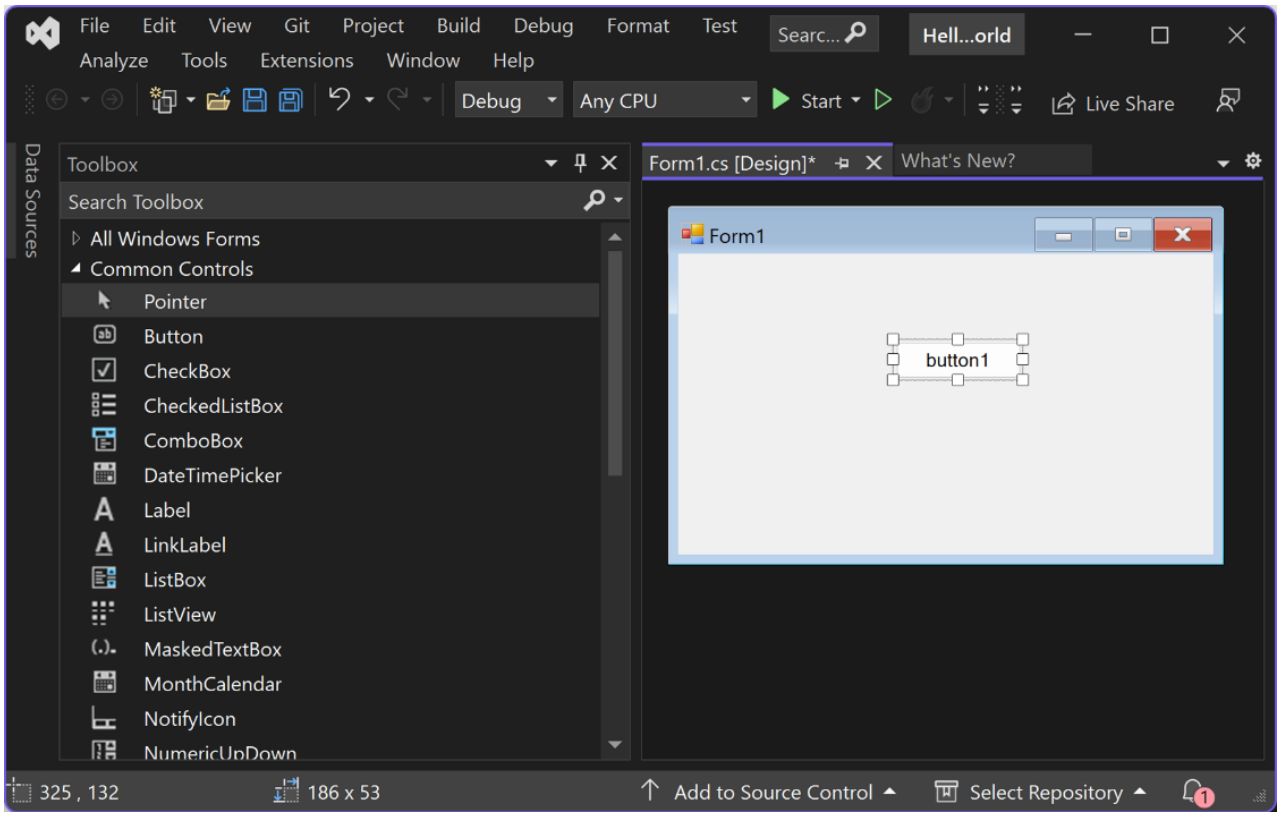
Kalkulaatori valmistamiseks on valitud „Windows Forms app" rakenduse tüüp (Joonis 17). Meie arvates on see kõige sobilikum rakenduse tüüp, mis tööle võiks sobida. Seda tüüpi rakendust on piisavalt lihtne luua ja on olemas mitu kasutusjuhendit.



Joonis 17. Uue rakenduse loomine

[4] https://visualstudio.microsoft.com/ru/
[5] https://learn.microsoft.com

Alustasime sellest, et ekraan peaks olema parameetriline ja tekkima koodi abil. Mida see tähendab? See tähendab, et kasutasime mitte kõige lihtsamat meetodit ja ohverdasime palju aega, selleks, et formeerida oma programmi. Joonisel (Joonis 18) on näha vorm ehk tuleviku programmi algset vaadet. Klassikaliseks ja lihtsamaks teeks, oleks lihtsalt võtta mingi *Label*-i ehk sisestusaken ja panna see õigesse kohta. Samal joonisel on vasakul näha komponentide puu, millest saab valida endale vajaliku komponendi ja tõmmata see vormi peale. Pärast seda anda sellele nimi, lisada funktsioone ja kasutada. Meie läksime teist teed. Meetod on selles, et me ei tõmba mingeid komponente käsitsi ehk manuaalselt vormi peale, vaid kirjutame koodiga, milliseid komponente milliste omadustega tahame ja tekitame need rakenduse peale. Seda oli alguses keerulisem teha, aga hiljem tundus see meetod väga mugavana, sest ei ole vaja liikuda mööda erinevaid arenduskeskkonna lehti, et muuta mõnda parameetrit, vaid teed kõik vajaliku oma koodi sees.



Joonis 18. Disaini aken

Peaaegu kõik *Label*-id ja *TextBox*-id on tekitatud töös tsükliga *For* (Joonis 19), [14]. Koodi meetodi kasutus kujutab endast samm-sammulist parameetrite deklareerimist. Alguses on vaja kirjutada ehk deklareerida, millist komponendi tahad tekitada. Samal joonisel deklareerin *Label*-i ja TextBox-e.

Edasi võib anda komponendile väärtuse ehk teksti, mida see hakkab näitama, või anda sellele nime ehk niiöelda aadressi, mille kaudu saab hiljem programmis selle komponendiga manipuleerida. Igale komponendile on vaja anda mõõdud. Hea tava mõõtude andmisel on võimalusel pidada meeles kuldselõigu seost või selle ligilähedast suhet [15]. Igal komponendil peab olema tema asukoht. Selleks kirjutame esimesele funktsioonis tekkivale komponendile staatilise *X*- ja *Y*-telgedel asuva positsiooni. Teistele komponendile määrame parameetrilise positsiooni (Joonis 20) nii, et iga järgmine komponent liigub kas *X*- või *Y*-teljel eemale eelmisest komponendist. Jääb ainult määrata stiil, *Label*-ite jaoks valisime 3D stiili, ja lisada vajalikud komponendid.

```
for (int i = 0; i < 7; i++) // FGM Label-id ja TextBox-id
{
    System.Windows.Forms.Label Lbl = new System.Windows.Forms.Label(); // Deklareerin millist Vormi rakenduse komponendi tahan saada.
    System.Windows.Forms.TextBox xA = new System.Windows.Forms.TextBox();
    System.Windows.Forms.TextBox Al = new System.Windows.Forms.TextBox();
    System.Windows.Forms.TextBox Th = new System.Windows.Forms.TextBox();
    System.Windows.Forms.TextBox zS = new System.Windows.Forms.TextBox();

    if (countLbl == 7)
    {
        Lbl.Text = "Tool"; // Viimase rida või tööriista osa nimetus.
    }
    else
    {
        Lbl.Text = "J" + countLbl.ToString(); // Esimeste kuue osade nimetused.
    }


    Lbl.Name = "Lbl" + this.countLbl.ToString(); // Label-i nimi, koosneb kirjutatud "Lbl" osast ja numbrist. Näiteks Lbl1.
    Lbl.Size = new System.Drawing.Size(50, 35); // Label-i suurus.

    xA.Name = "xA" + this.countLbl.ToString(); // Tekst kasti nimi, koosneb kirjutatud "Lbl" osast ja numbrist. Näiteks Lbl1.
    xA.Text = "0"; // Label-i esimene kirjutatud väärtus.
    xA.Size = new System.Drawing.Size(50, 35);


    Al.Name = "Al" + this.countLbl.ToString();
    Al.Text = "0";
    Al.Size = new System.Drawing.Size(50, 35);


    Th.Name = "Th" + this.countLbl.ToString();
    Th.Text = "0";
    Th.Size = new System.Drawing.Size(50, 35);


    zS.Name = "zS" + this.countLbl.ToString();
    zS.Text = "0";
    zS.Size = new System.Drawing.Size(50, 35);

    if (countLbl == 1) // Label-ite algus punkt.
    {
        Lbl.Location = new System.Drawing.Point(40, 100);
        xA.Location = new System.Drawing.Point(95, 100);
        Al.Location = new System.Drawing.Point(150, 100);
        Th.Location = new System.Drawing.Point(205, 100);
        zS.Location = new System.Drawing.Point(260, 100);
    }
```

Joonis 19. *For* tsükliga tekivad *Label*-id ja *TextBox*-id

```
88          else // Label-ite üle jäänud punktid.
89          {
90              Lbl.Location = new System.Drawing.Point(40, 100 + 40 * YLbl);
91              xA.Location = new System.Drawing.Point(95, 100 + 40 * YLbl);
92
93              zS.Location = new System.Drawing.Point(260, 100 + 40 * YLbl);
94
95
96              Al.Location = new System.Drawing.Point(150, 100 + 40 * YLbl);
97              Th.Location = new System.Drawing.Point(205, 100 + 40 * YLbl);
98          }
99
100         Lbl.BorderStyle = System.Windows.Forms.BorderStyle.Fixed3D; // Label-ite stiilid 3D.
101         xA.BorderStyle = System.Windows.Forms.BorderStyle.Fixed3D;
102         Al.BorderStyle = System.Windows.Forms.BorderStyle.Fixed3D;
103         Th.BorderStyle = System.Windows.Forms.BorderStyle.Fixed3D;
104         zS.BorderStyle = System.Windows.Forms.BorderStyle.Fixed3D;
105         YLbl++;
106         countLbl++;
107         this.Controls.Add(Lbl); // Label-ite lisamine
108         this.Controls.Add(xA);
109         this.Controls.Add(zS);
110
111         if (countLbl != 8) // Piiran Label-ite tekkimist, tööriistal neid parameetreid ei pea olema.
112         {
113             this.Controls.Add(Al);
114             this.Controls.Add(Th);
115         }
116     }
117
118     int countNms = 1;
119     int XNms = 0;
```

Joonis 20. Komponentide omadused

3.2 Programmi FGM-i osa

Muudame kasutajana sisestatud andmeid, *string* väärtuse *double* väärtuseks. Formeerime maatriksid. Selleks kasutame massiive. Maatriksid on suurusega 4 × 4, see tähendab, et massiivides peab olema 16 muutujat (Joonis 21). Edasi nimetame kõiki tekitatud massiive, mis kujutavad maatrikseid – maatriksiteks. (Lisa)

```
567         string ZS1str = (this.Controls.Find("zS1", true)[0]).Text; // s väärtus (mm)
568         string ZS2str = (this.Controls.Find("zS2", true)[0]).Text;
569         string ZS3str = (this.Controls.Find("zS3", true)[0]).Text;
570         string ZS4str = (this.Controls.Find("zS4", true)[0]).Text;
571         string ZS5str = (this.Controls.Find("zS5", true)[0]).Text;
572         string ZS6str = (this.Controls.Find("zS6", true)[0]).Text;
573         string ZS7str = (this.Controls.Find("zS7", true)[0]).Text;
574         double zs1 = double.Parse(ZS1str);
575         double zs2 = double.Parse(ZS2str);
576         double zs3 = double.Parse(ZS3str);
577         double zs4 = double.Parse(ZS4str);
578         double zs5 = double.Parse(ZS5str);
579         double zs6 = double.Parse(ZS6str);
580         double zs7 = double.Parse(ZS7str);
581
582         var dhMtx1 = Array.CreateInstance(typeof(double), 16); // Tekkitame maatriksi.
583         dhMtx1.SetValue(Math.Cos(th1), 0); dhMtx1.SetValue(-(Math.Sin(th1)) * Math.Cos(al1), 4); dhMtx1.SetValue(Math.Sin(th1) * Math.Sin(al1), 8); dhMtx1.SetValue(a1 * Math.Cos(th1), 12);
584         dhMtx1.SetValue(Math.Sin(th1), 1); dhMtx1.SetValue(Math.Cos(th1) * Math.Cos(al1), 5); dhMtx1.SetValue(-(Math.Cos(th1)) * Math.Sin(al1), 9); dhMtx1.SetValue(a1 * Math.Sin(th1), 13);
585         dhMtx1.SetValue(0, 2); dhMtx1.SetValue(Math.Sin(al1), 6); dhMtx1.SetValue(Math.Cos(al1), 10); dhMtx1.SetValue(zs1, 14);
586         dhMtx1.SetValue(0, 3); dhMtx1.SetValue(0, 7); dhMtx1.SetValue(0, 11); dhMtx1.SetValue(1, 15);
```

Joonis 21. DH-Parameetrite muutujad ja esimene maatriks

Kui maatriksid on formeeritud, tekitame mitu tühja maatriksit. Need on vajalikud selleks, et kirjutada nende väärtuseks teiste maatriksite korrutamise tulemusi. Eraldi on vaja tekitada funktsioon, mis

hakkab maatrikseid omavahel korrutama. Joonisel on ühe sellise funktsiooni nimi „mulMtxByMtx“ (Joonis 22).

```
618             var dhMtx7 = Array.CreateInstance(typeof(double), 16);
619             dhMtx7.SetValue(Math.Cos(th7), 0); dhMtx7.SetValue(-(Math.Sin(th7)) * Math.Cos(al7), 4); dhMtx7.SetValue(Math.Sin(th7) * Math.Sin(al7), 8); dhMtx7.SetValue(a7 * Math.Cos(th7), 12);
620             dhMtx7.SetValue(Math.Sin(th7), 1); dhMtx7.SetValue(Math.Cos(th7) * Math.Cos(al7), 5); dhMtx7.SetValue(-(Math.Cos(th7)) * Math.Sin(al7), 9); dhMtx7.SetValue(a7 * Math.Sin(th7), 13);
621             dhMtx7.SetValue(0, 2); dhMtx7.SetValue(Math.Sin(al7), 6); dhMtx7.SetValue(Math.Cos(al7), 10); dhMtx7.SetValue(zs7, 14);
622             dhMtx7.SetValue(0, 3); dhMtx7.SetValue(0, 7); dhMtx7.SetValue(0, 11); dhMtx7.SetValue(1, 15);
623
624             var Matrix1 = Array.CreateInstance(typeof(double), 16); // Tekkitame uue tühja maatriksi.
625             var Matrix2 = Array.CreateInstance(typeof(double), 16);
626             var Matrix3 = Array.CreateInstance(typeof(double), 16);
627             var Matrix4 = Array.CreateInstance(typeof(double), 16);
628             var Matrix5 = Array.CreateInstance(typeof(double), 16);
629             var Matrix6 = Array.CreateInstance(typeof(double), 16);
630
631             mulMtxByMtx(ref dhMtx1, ref dhMtx2, ref Matrix1); // Korrutame tekkitatud maatrikseid õiges järjekorras.
632             mulMtxByMtx(ref Matrix1, ref dhMtx3, ref Matrix2);
633             mulMtxByMtx(ref Matrix2, ref dhMtx4, ref Matrix3);
634             mulMtxByMtx(ref Matrix3, ref dhMtx5, ref Matrix4);
635             mulMtxByMtx(ref Matrix4, ref dhMtx6, ref Matrix5);
636             mulMtxByMtx(ref Matrix5, ref dhMtx7, ref Matrix6);
637             double tW = 0;
638             double tP = 0;
639             double tR = 0;
640             mtx2xyzwpr(ref Matrix6, w: ref tW, ref tP, ref tR);
641
642             void mulMtxByMtx(ref Array mtxIn1, ref Array mtxIn2, ref Array mtxOut) // Maatriksite 4x4 korrutamise tehe.
643             {
644                 for (int i = 0; i < 4; i++)
645                 {
646                     for (int j = 0; j < 4; j++)
647                     {
648                         mtxOut.SetValue((double)mtxIn1.GetValue(i) * (double)mtxIn2.GetValue(j * 4)
649                                       + (double)mtxIn1.GetValue(i + 4) * (double)mtxIn2.GetValue(j * 4 + 1)
650                                       + (double)mtxIn1.GetValue(i + 8) * (double)mtxIn2.GetValue(j * 4 + 2)
651                                       + (double)mtxIn1.GetValue(i + 12) * (double)mtxIn2.GetValue(j * 4 + 3),
652                                       i + j * 4);
653                     }
654                 }
655             }
```

Joonis 22. Maatriksite korrutamine

Kui lõppmaatriks on korrutamise teel leidud, siis saab sellest määrata konkreetse positsiooni ja orientatsiooni. Orientatsiooni leiame eraldi funktsiooniga nimega „mtx2xyzwpr“ (Joonis 23). Saadud vastused sisestame FGM-i vastuste *Label*-itesse.

```
657             void mtx2xyzwpr(ref Array mtx, ref double w, ref double p, ref double r) // Euleri nurkade arvutamine.
658             {
659
660                 r = Math.Atan2((double)mtx.GetValue(1), (double)mtx.GetValue(0)) * 180 / Math.PI; // Z nurga arvutamine.
661                 p = Math.Atan2(-(double)mtx.GetValue(2), Math.Sqrt(1 - (double)mtx.GetValue(2) * (double)mtx.GetValue(2))) * 180 / Math.PI; // Y nurga arvutamine.
662                 w = Math.Atan2((double)mtx.GetValue(6), (double)mtx.GetValue(10)) * 180 / Math.PI; // X nurga arvutamine.
663             }
664
665             this.Controls["dV1"].Text = String.Format("{0:N3}", Convert.ToDouble(Matrix6.GetValue(12))); // X positsioon.
666             this.Controls["dV2"].Text = String.Format("{0:N3}", Convert.ToDouble(Matrix6.GetValue(13))); // Y positsioon.
667             this.Controls["dV3"].Text = String.Format("{0:N3}", Convert.ToDouble(Matrix6.GetValue(14))); // Z positsioon.
668
669             this.Controls["dVa1"].Text = String.Format("{0:N3}", Convert.ToDouble(tW)); // X nurga arvutamine.
670             this.Controls["dVa2"].Text = String.Format("{0:N3}", Convert.ToDouble(tP)); // Y nurga arvutamine.
671             this.Controls["dVa3"].Text = String.Format("{0:N3}", Convert.ToDouble(tR)); // Z nurga arvutamine.
```

Joonis 23. Euleri nurkade arvutamine

Nii FGM-is kui ka IGM-is on maatriksites teatud kindlad valemid. Näiteks tabelis (Tabel 1) on näha, et kolmanda koordinaatteljestiku nurk on seotud teisega ja teisele omakorda liidetakse 90°, sama seost on vaja kasutada ka programmis (Joonis 24). Seda on vaja meeles pidada ja juhul, kui parameetrid muutuvad, tuleb muuta ka nende omavahelisi seoseid programmis. Fanuc kuulub ümberkonfigureeritavate robotite hulka. See tähendab, et kui roboti algne konfiguratsioon või koordinaatteljestiku omavahelised seosed ei vasta teie vajadustele, võite neid muuta.

```
553         string TH1str = (this.Controls.Find("Th1", true)[0]).Text; // Theta väärtus (deg).
554         string TH2str = (this.Controls.Find("Th2", true)[0]).Text;
555         string TH3str = (this.Controls.Find("Th3", true)[0]).Text;
556         string TH4str = (this.Controls.Find("Th4", true)[0]).Text;
557         string TH5str = (this.Controls.Find("Th5", true)[0]).Text;
558         string TH6str = (this.Controls.Find("Th6", true)[0]).Text;
559         double th1 = double.Parse(TH1str) * Math.PI / 180.0;
560         double th2 = (Math.PI / 2) - (double.Parse(TH2str) * Math.PI / 180.0);
561         double th3 = (double.Parse(TH2str) + double.Parse(TH3str)) * Math.PI / 180.0;
562         double th4 = double.Parse(TH4str) * Math.PI / 180.0;
563         double th5 = double.Parse(TH5str) * Math.PI / 180.0;
564         double th6 = double.Parse(TH6str) * Math.PI / 180.0;
565         double th7 = 0.0;
```

Joonis 24. Theta-de seosed

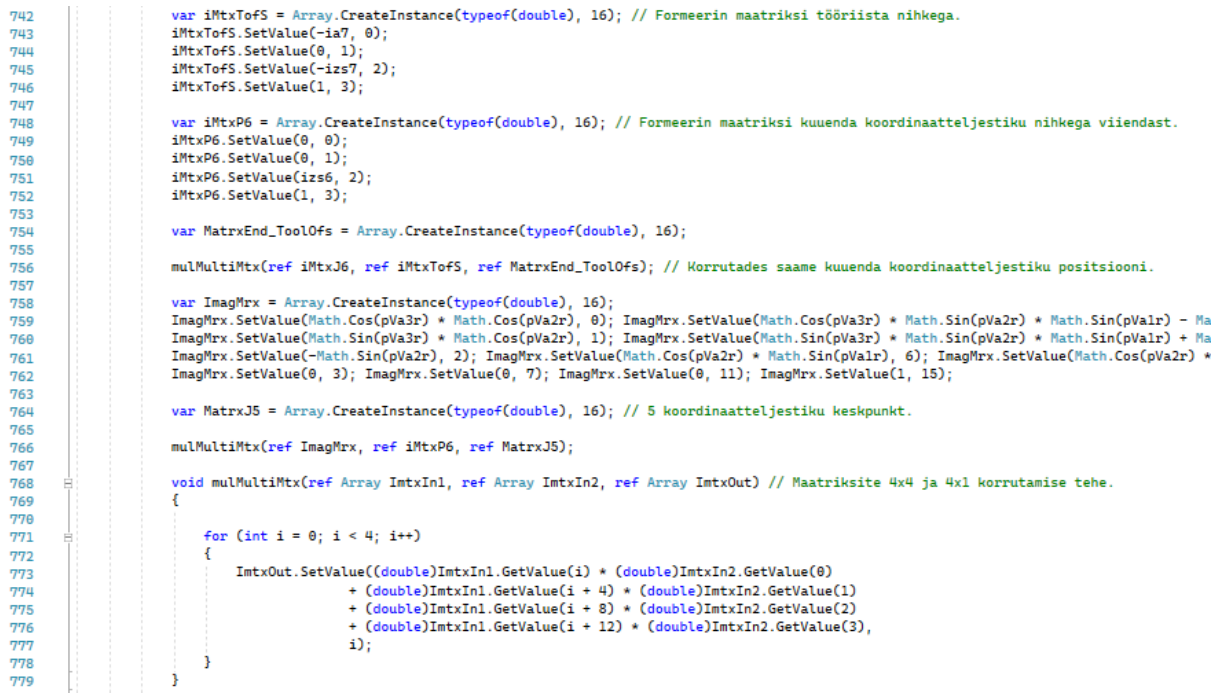
IGM-i puhul on õiged seosed vaja sisestada mitte tekitatud muutujasse, vaid otse maatriksi massiividesse (Joonis 25).

```
805         var idhMtx1 = Array.CreateInstance(typeof(double), 16); // Maatriks 1.
806         idhMtx1.SetValue(Math.Cos(rJ1), 0); idhMtx1.SetValue(-(Math.Sin(rJ1)) * Math.Cos(ial1), 4); idhMtx1.SetValue(Math.Sin(rJ1) * Math.Sin(ial1), 8);
807         idhMtx1.SetValue(Math.Sin(rJ1), 1); idhMtx1.SetValue(Math.Cos(rJ1) * Math.Cos(ial1), 5); idhMtx1.SetValue(-(Math.Cos(rJ1)) * Math.Sin(ial1), 9);
808         idhMtx1.SetValue(0, 2); idhMtx1.SetValue(Math.Sin(ial1), 6); idhMtx1.SetValue(Math.Cos(ial1), 10); idhMtx1.SetValue(izs1, 14);
809         idhMtx1.SetValue(0, 3); idhMtx1.SetValue(0, 7); idhMtx1.SetValue(0, 11); idhMtx1.SetValue(1, 15);
810
811         var idhMtx2 = Array.CreateInstance(typeof(double), 16); // Maatriks 2.
812         idhMtx2.SetValue(Math.Cos((Math.PI / 2) - rJ2), 0); idhMtx2.SetValue(-(Math.Sin((Math.PI / 2) - rJ2)) * Math.Cos(ial2), 4); idhMtx2.SetValue(Math
813         idhMtx2.SetValue(Math.Sin((Math.PI / 2) - rJ2), 1); idhMtx2.SetValue(Math.Cos((Math.PI / 2) - rJ2) * Math.Cos(ial2), 5); idhMtx2.SetValue(-(Math.
814         idhMtx2.SetValue(0, 2); idhMtx2.SetValue(Math.Sin(ial2), 6); idhMtx2.SetValue(Math.Cos(ial2), 10); idhMtx2.SetValue(izs2, 14);
815         idhMtx2.SetValue(0, 3); idhMtx2.SetValue(0, 7); idhMtx2.SetValue(0, 11); idhMtx2.SetValue(1, 15);
```

Joonis 25. IGM-i maatriksid

3.3 Programmi IGM-i osa

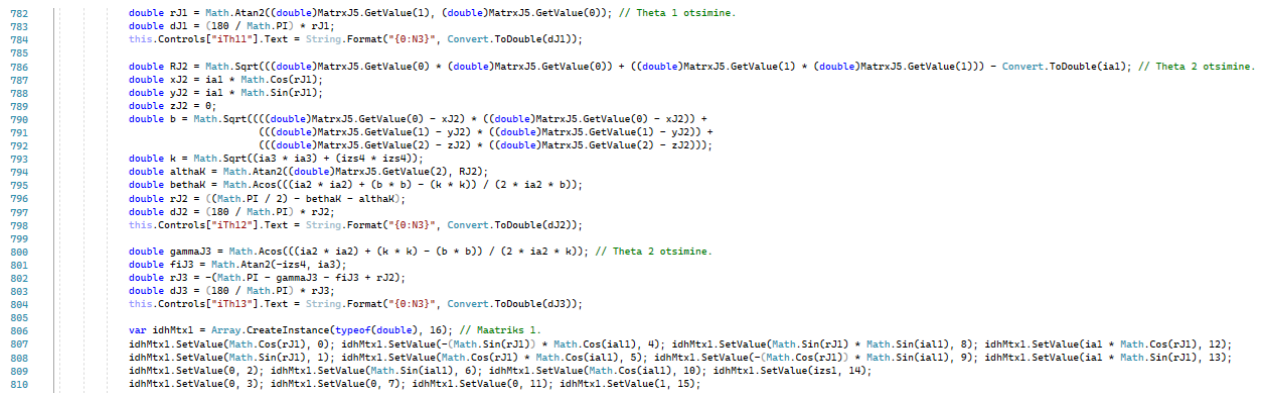
IGM-i puhul tekitame rohkem *Label*-e, aga kasutame sama programmikoodi osasid, nagu ka FGM-i osas. Lahendi alguses peame aga formeerima uute mõõtmetega maatrikseid 4×1, koos uue korrutamisfunktsiooniga, mille nimeks on „mulMultiMtx“ (Joonis 26).



```
742             var iMtxTofS = Array.CreateInstance(typeof(double), 16); // Formeerin maatriksi tööriista nihkega.
743             iMtxTofS.SetValue(-ia7, 0);
744             iMtxTofS.SetValue(0, 1);
745             iMtxTofS.SetValue(-izs7, 2);
746             iMtxTofS.SetValue(1, 3);
747
748             var iMtxP6 = Array.CreateInstance(typeof(double), 16); // Formeerin maatriksi kuuenda koordinaatteljestiku nihkega viiendast.
749             iMtxP6.SetValue(0, 0);
750             iMtxP6.SetValue(0, 1);
751             iMtxP6.SetValue(izs6, 2);
752             iMtxP6.SetValue(1, 3);
753
754             var MatrxEnd_ToolOfs = Array.CreateInstance(typeof(double), 16);
755
756             mulMultiMtx(ref iMtxJ6, ref iMtxTofS, ref MatrxEnd_ToolOfs); // Korrutades saame kuuenda koordinaatteljestiku positsiooni.
757
758             var ImagMrx = Array.CreateInstance(typeof(double), 16);
759             ImagMrx.SetValue(Math.Cos(pVa3r) * Math.Cos(pVa2r), 0); ImagMrx.SetValue(Math.Cos(pVa3r) * Math.Sin(pVa2r) * Math.Sin(pVa1r) - Ma
760             ImagMrx.SetValue(Math.Sin(pVa3r) * Math.Cos(pVa2r), 1); ImagMrx.SetValue(Math.Sin(pVa3r) * Math.Sin(pVa2r) * Math.Sin(pVa1r) + Ma
761             ImagMrx.SetValue(-Math.Sin(pVa2r), 2); ImagMrx.SetValue(Math.Cos(pVa2r) * Math.Sin(pVa1r), 6); ImagMrx.SetValue(Math.Cos(pVa2r) *
762             ImagMrx.SetValue(0, 3); ImagMrx.SetValue(0, 7); ImagMrx.SetValue(0, 11); ImagMrx.SetValue(1, 15);
763
764             var MatrxJ5 = Array.CreateInstance(typeof(double), 16); // 5 koordinaatteljestiku keskpunkt.
765
766             mulMultiMtx(ref ImagMrx, ref iMtxP6, ref MatrxJ5);
767
768             void mulMultiMtx(ref Array ImtxIn1, ref Array ImtxIn2, ref Array ImtxOut) // Maatriksite 4x4 ja 4x1 korrutamise tehe.
769             {
770
771                 for (int i = 0; i < 4; i++)
772                 {
773                     ImtxOut.SetValue((double)ImtxIn1.GetValue(i) * (double)ImtxIn2.GetValue(0)
774                                 + (double)ImtxIn1.GetValue(i + 4) * (double)ImtxIn2.GetValue(1)
775                                 + (double)ImtxIn1.GetValue(i + 8) * (double)ImtxIn2.GetValue(2)
776                                 + (double)ImtxIn1.GetValue(i + 12) * (double)ImtxIn2.GetValue(3),
777                                 i);
778                 }
779             }
```

Joonis 26. Erineva mõõtmetega maatriksite korrutamine

Eriti tuleks pöörata tähelepanu sellele, kuidas on kirjutatud valemid, kus kasutatud muutuja on teises astmes ehk ruudus. Juhul, kui seda on vaja kasutada, siis korrutame seda muutujat tema endaga, näide on esitatud joonisel (Joonis 27).

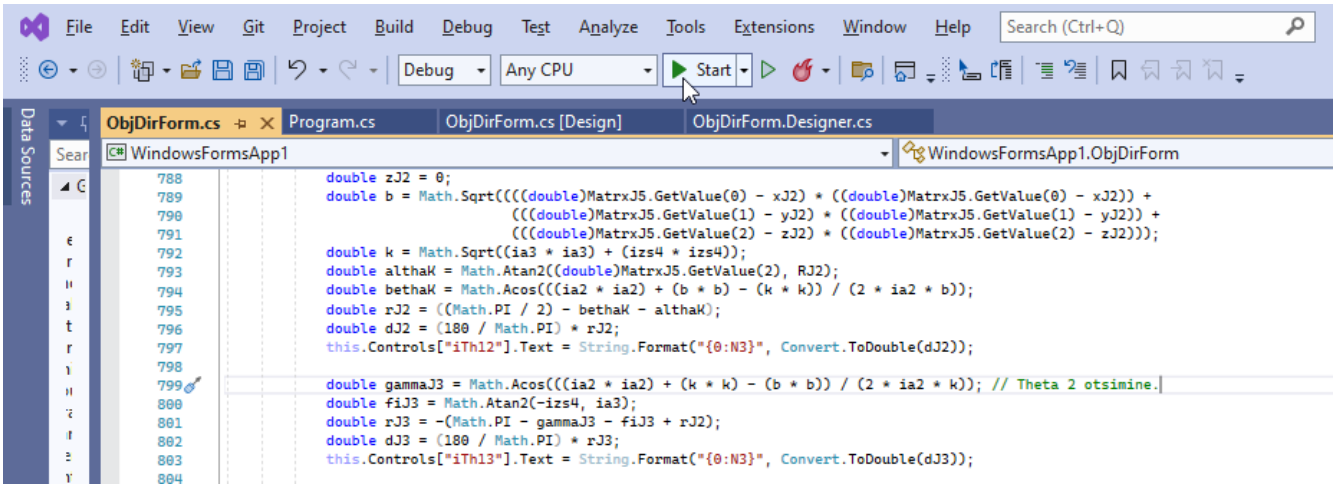


```
782             double rJ1 = Math.Atan2((double)MatrxJ5.GetValue(1), (double)MatrxJ5.GetValue(0)); // Theta 1 otsimine.
783             double dJ1 = (180 / Math.PI) * rJ1;
784             this.Controls["iTh11"].Text = String.Format("{0:N3}", Convert.ToDouble(dJ1));
785
786             double RJ2 = Math.Sqrt(((double)MatrxJ5.GetValue(0) * (double)MatrxJ5.GetValue(0)) + ((double)MatrxJ5.GetValue(1) * (double)MatrxJ5.GetValue(1))) - Convert.ToDouble(ia1); // Theta 2 otsimine.
787             double xJ2 = ia1 * Math.Cos(rJ1);
788             double yJ2 = ia1 * Math.Sin(rJ1);
789             double zJ2 = 0;
790             double b = Math.Sqrt((((double)MatrxJ5.GetValue(0) - xJ2) * ((double)MatrxJ5.GetValue(0) - xJ2)) +
791                                  (((double)MatrxJ5.GetValue(1) - yJ2) * ((double)MatrxJ5.GetValue(1) - yJ2)) +
792                                  (((double)MatrxJ5.GetValue(2) - zJ2) * ((double)MatrxJ5.GetValue(2) - zJ2)));
793             double k = Math.Sqrt((ia3 * ia3) + (izs4 * izs4));
794             double althaK = Math.Atan2((double)MatrxJ5.GetValue(2), RJ2);
795             double bethaK = Math.Acos(((ia2 * ia2) + (b * b) - (k * k)) / (2 * ia2 * b));
796             double rJ2 = ((Math.PI / 2) - bethaK - althaK);
797             double dJ2 = (180 / Math.PI) * rJ2;
798             this.Controls["iTh12"].Text = String.Format("{0:N3}", Convert.ToDouble(dJ2));
799
800             double gammaJ3 = Math.Acos(((ia2 * ia2) + (k * k) - (b * b)) / (2 * ia2 * k)); // Theta 2 otsimine.
801             double fiJ3 = Math.Atan2(-izs4, ia3);
802             double rJ3 = -(Math.PI - gammaJ3 - fiJ3 + rJ2);
803             double dJ3 = (180 / Math.PI) * rJ3;
804             this.Controls["iTh13"].Text = String.Format("{0:N3}", Convert.ToDouble(dJ3));
805
806             var idhMtx1 = Array.CreateInstance(typeof(double), 16); // Maatriks 1.
807             idhMtx1.SetValue(Math.Cos(rJ1), 0); idhMtx1.SetValue(-(Math.Sin(rJ1)) * Math.Cos(ial1), 4); idhMtx1.SetValue(Math.Sin(rJ1) * Math.Sin(ial1), 8); idhMtx1.SetValue(ial * Math.Cos(rJ1), 12);
808             idhMtx1.SetValue(Math.Sin(rJ1), 1); idhMtx1.SetValue(Math.Cos(rJ1) * Math.Cos(ial1), 5); idhMtx1.SetValue(-(Math.Cos(rJ1)) * Math.Sin(ial1), 9); idhMtx1.SetValue(ial * Math.Sin(rJ1), 13);
809             idhMtx1.SetValue(0, 2); idhMtx1.SetValue(Math.Sin(ial1), 6); idhMtx1.SetValue(Math.Cos(ial1), 10); idhMtx1.SetValue(izs1, 14);
810             idhMtx1.SetValue(0, 3); idhMtx1.SetValue(0, 7); idhMtx1.SetValue(0, 11); idhMtx1.SetValue(1, 15);
```

Joonis 27. Erinevad matemaatikatehted

3.4 Kalkulaatori kasutamine

Kui kood on valmis, käivitamine kalkulaatori vajutades nupule Start (Joonis 28). Siis ilmub teie ekraanile kalkulaator, mida on vaja täita.



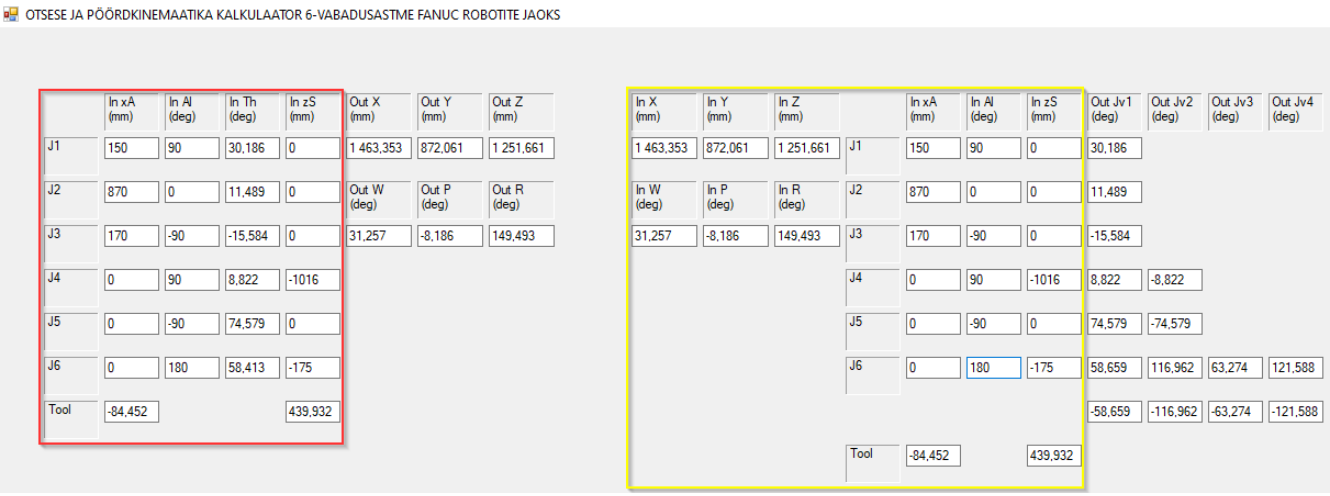
Joonis 28. Start nupu asukoht

Programmi kujundamisel on hea tava kujundada see tavalise lugemisloogika alusel, vasakult paremale. Ka see programm on tehtud nii, et kasutaja täidab vasaku osa ja vastuse näeb paremal.

Erinevates programmides on DH-Parameetrid antud erinevas järjekorras. Selles töös kasutatakse FGM-i osas samasugust järjekorda nagu kasutasid meetodi väljatöötanud insenerid oma esimeses artiklis. [2]

Selleks aga, et saada vastust FGM-i osas, on vaja täita osa, mis on märgistatud joonisel (Joonis 29) punasega, IGM-i osa vastuse saamiseks on vaja täita osa, mis on märgistatud samal joonisel kollasega. Mittenaturaalarvude sisestamisel tuleb kasutada kindlasti koma. Nendesse *Label*-itesse, kus ei ole väärtust, peab olema sisestatud „0“.

Kõik algab sündmusest, ka rakendusel on ettenähtud sündmus, aga mitte nii konkreetne nagu nupp. Selleks, et pärast andmete sisestamist kalkulaator arvutaks vastuse, on vaja teha topeltklõps programmi ehk rakenduse hallil taustal, kus ei ole *Label*-eid või *TextBox*-e.
Alguses oli programmis vastuste osas ainult *TextBox*-id. See tundus mõistlik selle poolest, et võimaldas mitte ajada sassi FGM-i ja IGM-i osasi. Aga testimise käigus tuli välja, et *TextBox*-st ei saa kopeerida andmeid ja panna *Label*-isse. Programmi lõpp versioonis on enamik *TextBox*-e välja vahetatud *Label*-itega. See võimaldab mugavalt kopeerida IGM-i osast FGM-i osasse nurki ja kontrollida tulemusi analüütilise meetodiga.



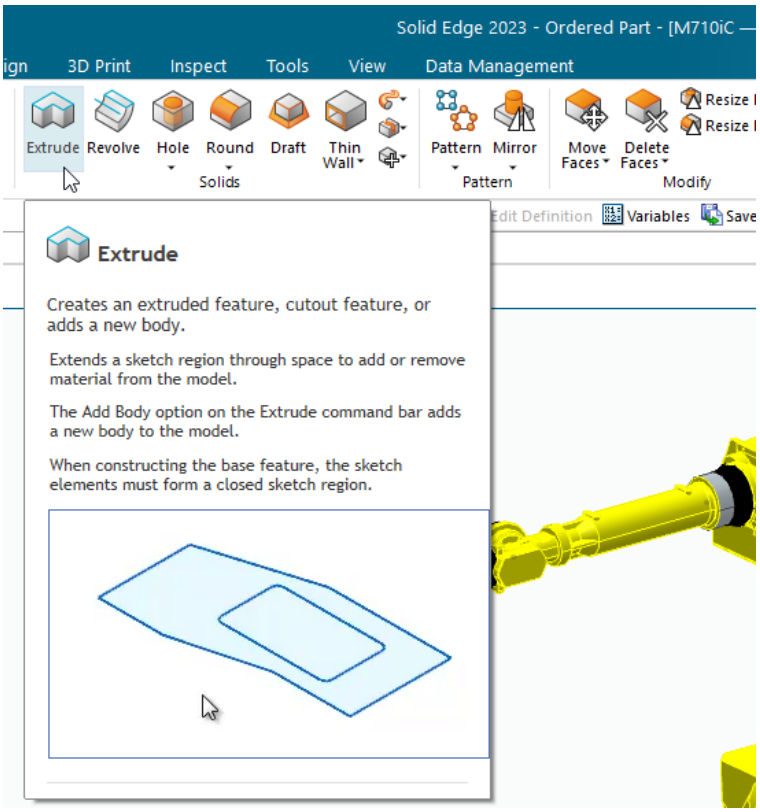
Joonis 29. Programmi täitmine

4 ROBOTI TÖÖRUUM CAD PROGRAMMIS

Nagu oli juba mainitud, kalkulaator on abitööriist, mida võib kasutada ükskõik mis CAD programmiga. Näiteks täna on mugav kasutada üht programmi, sest suurem osa kliente kasutab sama programmi. Võimalik, et homme hakkab litsents rohkem maksma ja see saab põhjuseks uue programmi valikuks. See ei ole suur mure, sest kalkulaator ei ole seotud konkreetse CAD programmiga.

Selles töös kasutatakse Solid Edge programmi. Töö programmis algab roboti mudeli otsimisega. Fanuc-i mudeli „M-710iC_50“ saab alla laadida „MyFanuc“ originaalveebilehelt tasuta.[6]

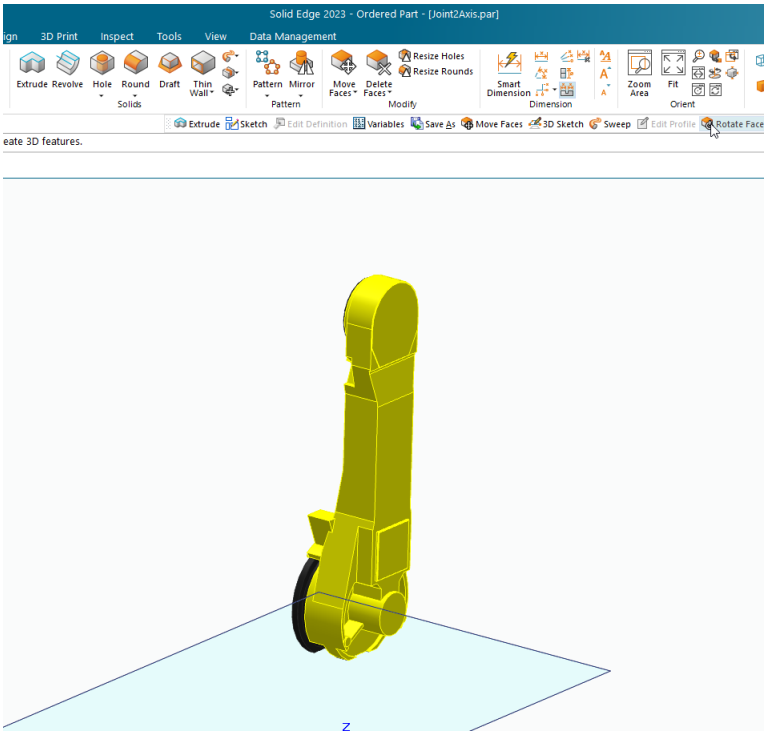
Pärast selle allalaadimist on vaja tekitada mitu eraldi liigendit. Solid Edge programmis saab seda teha käsuga *Extrude* (Joonis 30).



Joonis 30. *Extrude* käsk

[6] https://my.fanuc.eu/

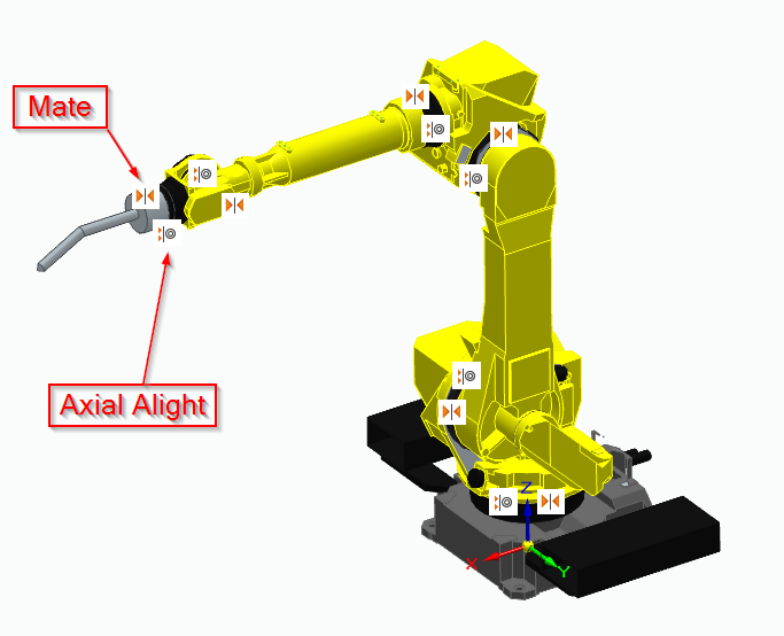
Ülesanne seisneb selles, et teha mitu roboti koopiat ja pärast ükshaaval lõigata nendest valmis 7 roboti liigendit. Kui ei õnnestu leida vajaliku tööriista mudelit, on vaja see eraldi joonistada *Part* failina (Joonis 31). [16]



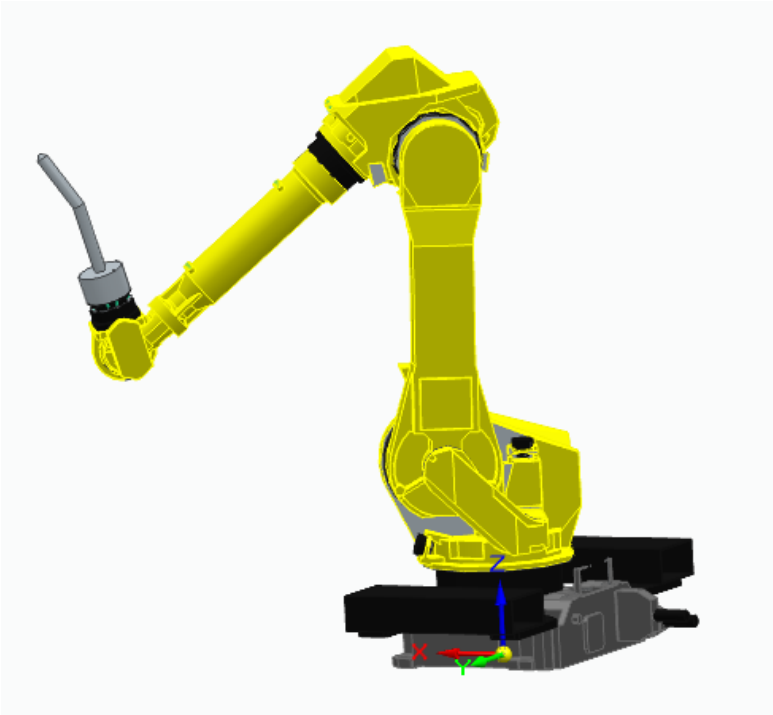
Joonis 31. Roboti Fanuc M-710iC_50 liigend

Kui on tekitatud seitse eraldi osa ehk liigendit, on need vaja omavahel siduda. Pärast seda tekitame uue koostu ehk *Assembly* faili. Faile on vaja lisada ükshaaval ja neile on vaja kirjutada vastavasse aknasse positsioonid $X, Y, Z = 0{,}0{,}0$. Tänu sellele saab iga liigend kohe oma õige algse asukoha. Seostamiskäsud on *Assemble* all. Esimesena soovitan kasutada tasapinnalist joondust ehk *Mate* seost. Järgmisena seome liigendeid aktsiaalse seosega *Axial Alight* (Joonis 32). [17] Kui seosed on tekitatud, võib robotiga manipuleerida.

Kuidas toimub roboti kujundamine Solid Edge programmis. Klient annab ülesande, ettevõte mõtleb välja kontseptsiooni ja joonistab mingi variandi. Selles joonises peaksid olema kõik vajalikud agregaadid, tööriistad, pingid, turvaväravad, jne. Juhul, kui roboti manipuleerimisel tekkis viga ja kontseptsioon või lihtsalt mingi osa positsioon ei sobinud, on vaja oma osasid ümber joonestada või positsioneerida paremini. (Joonis 33)



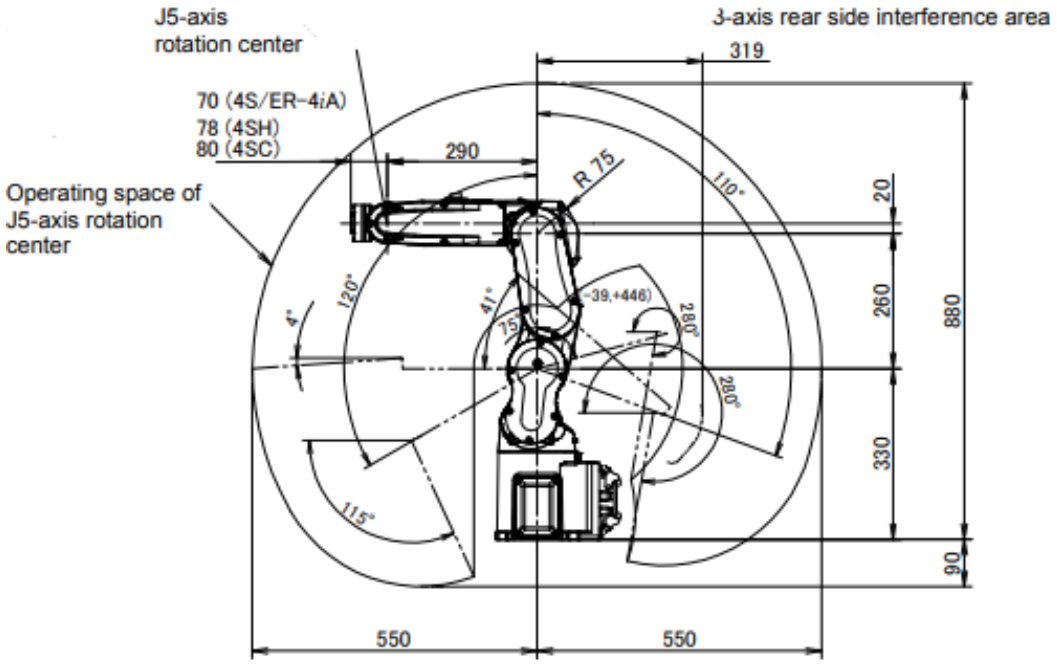
Joonis 32. Roboti *Assemble* liigendite seoste funktsioonid



Joonis 33. Robot tööasendis

5 KUUE VABADUSASTMEGA FANUC ROBOTID

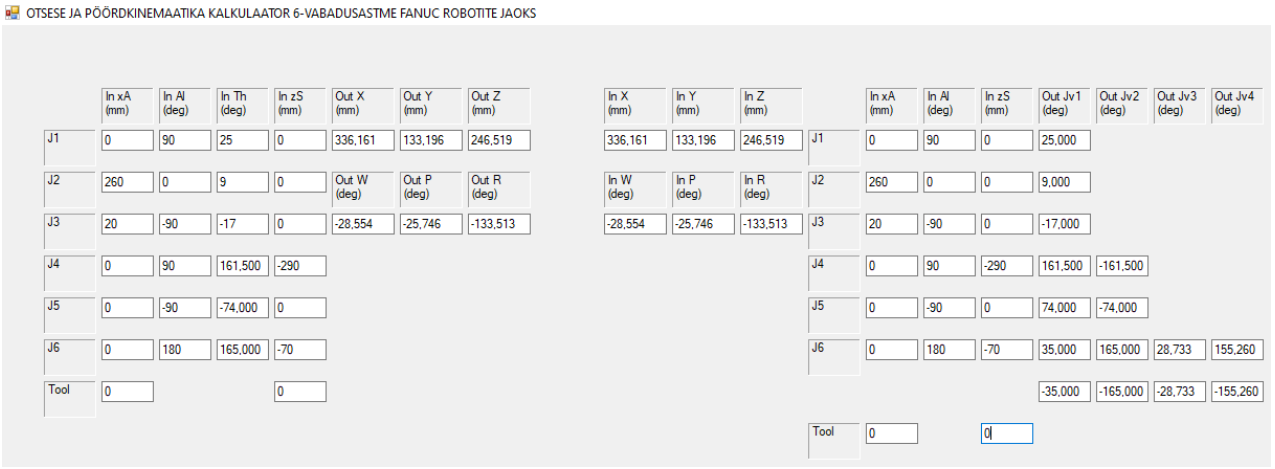
Selle lõputöö raames on loodud kalkulaator, mis töötab abitööriistana selleks, et visualiseerida roboti Fanuc M-710iC/50 positsioone. Seda kalkulaatorit saab kasutada erinevate robotite jaoks ja mitte ainult Fanuc brändiga. Selles lõpputöös on tehtud kaks kinemaatika roboti mudelit, Fanuc M-710iC/50 ja ER-4iA, DH-Parameetritega, mis töötavad korralikult. Selles kajastub erinevate koordinaatteljestike seostamise õnnestumine. Tänu sellele töötavad nii FGM kui ka IGM. FANUC roboti ER-4iA mõõtmed on esitatud joonisel (Joonis 34). DH-Parameetritest paljud muutusid väiksemaks, sest see robot on valmistatud õppimise jaoks ja saab tõsta ainult 4 kg raskust, tema DH-Parameetreid on esitatud tabelis (Tabel 2). Need on ühesuguste koordinaatteljestike seostega robotid, nende positsiooni ja nurkade arvutused on esitatud joonisel (Joonis 35). [18]



Joonis 34. FANUC ER-4iA roboti mõõtmed

Tabel 2. ER-4iA roboti DH-Parameetrid.

Liigend, i	a_i **(mm)**	α_i **(deg)**	s_i **(mm)**	θ_i **(deg)**
1	0	90	0	θ_1
2	260	0	0	$90 - \theta_2$
3	20	−90	0	$\theta_2 + \theta_3$
4	0	90	−290	θ_4
5	0	−90	0	θ_5
6	0	180	−70	θ_6
7	0	–	0	–



Joonis 35. ER-4iA roboti positsiooni arvutamine

Kui tekib vajadus, saab programmi kas kopeerida või muuta. Muuta võib nii, et lisate blokeerivaid kommentaare koodi või teete olemasolevatele arvutustele kommentaare ja lisate uusi arvutusi. Kuidas muuta koordinaatteljestike seoseid, on selgitatud programmeerimise osas. Võin igaks juhuks lisada, et kui teil on teise ettevõtte robot, siis on mõttekas uurida, millist Euleri nurkade järjekorda robotis kasutatakse ja muuta vajadusel Fanuc ZYX oma, näiteks teise levinud variandi peale XYZ.

KOKKUVÕTE

Lõputöö eesmärgiks oli luua kinemaatika kalkulaator, mis aitaks ettevõttel projekteerida roboti integreerimine mingisse protsessi. See võimaldab projekteerida Fanuc roboti üldpositsioone, kasutades kalkulaatorit koos mingi CAD programmiga, mis on juba ettevõttel olemas. Tarkvara loomine oli keeruline protsess. Kalkulaator koosneb kahest osast, matemaatilisest ja programmilisest. Failis on eraldi välja toodud FGM-i mudeliga seotud matemaatilised arvutused. FGM-i arvutamiseks on vaja teada, kuidas tekitada DH-Parameetritega maatrikseid ja leida Euleri nurkade abil mittekonkreetseid nurki. IGM-i ülesanne on kirjeldatud rohkemate etappidena, mis on seotud selle ülesanne eripäraga, selles leiame iga nurga eraldi ja oma meetodiga.

Järgmises peatükis on kirjeldatud, kuidas kujundati kalkulaator. Selles peatükis kirjeldatakse, kuidas alustada rakenduse loomist, kuidas valida sellele kujundus ja tuuakse näiteid, kuidas kirjutada koodi kõikidele vajalikele valemitele programmis.

CAD osa on vajalik selleks, et näidata, kuidas võib kasutada saadud IGM-i tulemusi rakenduses. Eraldi on välja toodud, kuidas tekitada valitud robotile mudel liigendite kaupa ja simuleerida selle erinevaid positsioone.

Viimases peatükis on esitatud tulemused veel ühe Fanuc roboti näitel. Kui eelnevas tööosas oli näitena kasutusel mudel Fanuc 710iC/50, siis viimases peatükis on selleks mudel Fanuc ER-4iA. See on tehtud selleks, et näidata erinevate mudelite simuleerimise võimalust töös väljatöötatud kalkulaatoriga, mille kood on lisatud lisasse.

Tulemuseks sai FGM-i ja IGM-i kalkulaator. Seda kalkulaatorit on mugav kasutada, sest üheks sihtmärgiks oli vamistada intuitiivselt arusaadav programmi kujundus. Kasutuselevõetud IGM-i lahendusmeetod annab tihti väga täpse vastuse. Tänu selle saab nii sisestada kui ka väljundist lugeda arve kolme numbriga pärast koma. Oleks õige natukene ka kritiseerida oma tööd. Ainuke mure, täpsustan mitte probleem on – pimedad vead. Juhul, kui kasutaja sisestab ebaõigeid andmeid või kalkulaator annab mitte kõige sobivama vastuse, peab kasutaja ise kas parandada oma vea või analüütiliselt valima sobivaima nurga.

VIIDATUD ALLIKAD

[1] К. А. Е. Артоболевский И. И., Знакомьтесь — роботы!, Молодая гвардия, 1979, p. 239.

[2] J. D. a. R. Hartenberg, „A Kinematic Notation for Lower-Pair Mechanisms Based on Matrices,“ *JOURNAL OF APPLIED MECHANICS,* 1955.

[3] S. K. S. J. K. D. S. V. Shah, „Denavit-Hartenberg,“ *Journal of Computational and Nonlinear Dynamics,* 2012.

[4] S. K. Krzysztof Tokarz, „Geometric approach to inverse kinematics for arm manipulator,“ Silesian University of Technology, p. 6.

[5] M. Law, „Top 10 industrial robotics companies in the world in 2023,“ *Technology,* 2023.

[6] R. C.Goertz.U.S.A. Patent 2,632,574, 1949.

[7] B. McMorris, „https://futura-automation.com/,“ 2022. [Võrgumaterjal]. Available: https://futura-automation.com/2019/05/15/a-history-timeline-of-industrial-robotics/.

[8] Yaskawa, „https://www.yumpu.com/en/document/read/3621168/motoman-robot-history,“ 2012. [Võrgumaterjal]. Available: https://www.yumpu.com/en/document/read/3621168/motoman-robot-history.

[9] A. Owen-Hill, „https://robodk.com/,“ 9 10 2018. [Võrgumaterjal]. Available: https://robodk.com/blog/program-robot-tips/.

[10] В. Г. А. П. О.И. Борисов, МЕТОДЫ УПРАВЛЕНИЯ РОБОТОТЕХНИЧЕСКИМИ ПРИЛОЖЕНИЯМИ, Санкт-Петербург: УНИВЕРСИТЕТ ИТМО, 2016, p. 105.

[11] FANUC, „FANUC Robot M-710iC-50 Mechanical Unit Operators Manual,“ 2017.

[12] L. V. Ahifors, Complex Analysis, 1979.

[13] D.M.Henderson, „Euler Angles, Quaternions, and Transformation Matrices,“ 1977.

[14] J. A. a. B. Albahari, C# 7.0 in a nutshell, Москва, Санкт-Петербург: Диалектика, 2019.

[15] K. Elam, Geometry of design, Москва: КоЛибри, 2021.

[16] Siemens. [Võrgumaterjal]. Available: https://solidedge.siemens.com/en/resource/tutorial/fidget-spinner/#before.

[17] Siemens. [Võrgumaterjal]. Available: https://solidedge.siemens.com/en/resource/tutorial/3d-cad-tutorial-robot-claw/#part-2.

[18] FANUC, „ER-4iA product information“.

[19] „https://robotics.kawasaki.com/,“ 2018. [Võrgumaterjal]. Available: https://robotics.kawasaki.com/ja1/xyz/en/1803-01/.

[20] Fanuc, „https://www.fanuc.co.jp/,“ [Võrgumaterjal]. Available: https://www.fanuc.co.jp/en/profile/history/index.html.

[21] Fanuc, „https://www.fanuc.eu/,“ [Võrgumaterjal]. Available: https://www.fanuc.eu/dk/en/robots/accessories/roboguide.

[22] S. K. S. J. K. D. S. V. Shah, „Denavit-Hartenberg Parameterization of Euler Angles,“ *Journal of Computational and Nonlinear Dynamics,* 2012.

[23] „https://visualstudio.microsoft.com,“ [Võrgumaterjal]. Available: https://visualstudio.microsoft.com/#vs-section.

[24] „https://learn.microsoft.com,“ 13 02 2023. [Võrgumaterjal]. Available: https://learn.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/.

[25] „https://learn.microsoft.com,“ 26 01 2023. [Võrgumaterjal]. Available: https://learn.microsoft.com/et-ee/visualstudio/ide/create-csharp-winform-visual-studio?view=vs-2022.

[26] [Võrgumaterjal]. Available: https://my.fanuc.eu/.

LISA. FGM-I JA IGM-I KOOD.

Kalkulaatori kood, kirjutatud Visual Studio programmiga.

```
1   using System;
2   using System.Collections.Generic;
3   using System.ComponentModel;
4   using System.Data;
5   using System.Drawing;
6   using System.Linq;
7   using System.Text;
8   using System.Threading.Tasks;
9   using System.Windows.Forms;
10  using Microsoft.Office.Interop.Excel;
11  using MathNet.Numerics.LinearAlgebra;
12  using MathNet.Numerics.LinearAlgebra.Double;
13
14                  namespace WindowsFormsApp1
15                  {
16                  public partial class ObjDirForm : Form
17                  {
18                  public ObjDirForm()
19                  {
20                  InitializeComponent();
21                  this.DoubleClick += doublbCli;
22                  }
23
24              int countLbl = 1;
25              int YLbl = 0;
26
27              int invCount = 1;
28              int invY = 0;
29
30              public void Form1_Load(object sender, EventArgs e)
31              {
32
33                      this.Text = "OTSESE JA PÖÖRDKINEMAATIKA KALKULAATOR 6-
                        VABADUSASTME FANUC ROBOTITE JAOKS";
34
35                          this.WindowState = FormWindowState.Maximized;
36
37
38
39                          for (int i = 0; i < 7; i++) // FGM Label-id ja TextBox-id
```

```
40          {
41          System.Windows.Forms.Label Lbl = new
                System.Windows.Forms.Label(); // Deklareerin millist Vormi rakenduse
                komponendi tahan saada.
42          System.Windows.Forms.TextBox xA = new
                        System.Windows.Forms.TextBox();
43          System.Windows.Forms.TextBox Al = new
                    System.Windows.Forms.TextBox();
44          System.Windows.Forms.TextBox Th = new System.Windows.Forms.TextBox();
45          System.Windows.Forms.TextBox zS = new
                        System.Windows.Forms.TextBox();
46
47              if (countLbl == 7)
48              {
49              Lbl.Text = "Tool"; // Viimase rida või tööriista osa
                nimetus.
50              }
51              else
52              {
53              Lbl.Text = "J" + countLbl.ToString(); // Esimeste kuue
                osade nimetused.
54              }
55
56
57          Lbl.Name = "Lbl" + this.countLbl.ToString(); // Label-i nimi, koosneb kirjutatud "Lbl" osast ja
            numbrist. Näiteks Lbl1.
58          Lbl.Size = new System.Drawing.Size(50, 35); // Label-i suurus.
59
60          xA.Name = "xA" + this.countLbl.ToString(); // Tekst kasti nimi, koosneb kirjutatud "Lbl"
            osast ja numbrist. Näiteks Lbl1.
61          xA.Text = "0"; // Label-i esimene kirjutatud väärtus.
62          xA.Size = new System.Drawing.Size(50, 35);
63
64
65          Al.Name = "Al" + this.countLbl.ToString();
66          Al.Text = "0";
67          Al.Size = new System.Drawing.Size(50, 35);
68
69
70          Th.Name = "Th" + this.countLbl.ToString();
71          Th.Text = "0";
72          Th.Size = new System.Drawing.Size(50, 35);
73
74
75          zS.Name = "zS" + this.countLbl.ToString();
```

```
76                  zS.Text = "0";
77                  zS.Size = new System.Drawing.Size(50, 35);
78
79                  if (countLbl == 1) // Label-ite algus punkt.
80                  {
81                  Lbl.Location = new System.Drawing.Point(40, 100); 82      xA.Location = new
                    System.Drawing.Point(95, 100);
83                      Al.Location = new System.Drawing.Point(150, 100);
84                      Th.Location = new System.Drawing.Point(205, 100);
85                      zS.Location = new System.Drawing.Point(260, 100);
86                      }
87
88                      else // Label-ite üle jäänud punktid.
89                      {
90                      Lbl.Location = new System.Drawing.Point(40, 100 + 40 *
                        YLbl);
91                      xA.Location = new System.Drawing.Point(95, 100 + 40 *
                      YLbl);
92
93                                  zS.Location = new System.Drawing.Point(260, 100 + 40 *
                        YLbl);
94
95
96                      Al.Location = new System.Drawing.Point(150, 100 + 40 *
                        YLbl);
97                      Th.Location = new System.Drawing.Point(205, 100 + 40 *
                        YLbl);
98                      }
99
100                 Lbl.BorderStyle =
                      System.Windows.Forms.BorderStyle.Fixed3D; // Label-ite stiilid 3D.
101                 xA.BorderStyle = System.Windows.Forms.BorderStyle.Fixed3D;
102                 Al.BorderStyle = System.Windows.Forms.BorderStyle.Fixed3D;
103                 Th.BorderStyle = System.Windows.Forms.BorderStyle.Fixed3D;
104                 zS.BorderStyle = System.Windows.Forms.BorderStyle.Fixed3D;
105                 YLbl++;
106                 countLbl++;
107                 this.Controls.Add(Lbl); // Label-ite lisamine
108                 this.Controls.Add(xA);
109                 this.Controls.Add(zS);
110
111                     if (countLbl != 8) // Piiran Label-ite tekkimist,
                    tööriistal neid parameetreid ei pea olema.
112                     {
113                     this.Controls.Add(Al);
114                     this.Controls.Add(Th);
115                     }
```

```
116                     }
117
118             int countNms = 1;
119             int XNms = 0;
120
121                 for (int i = 0; i < 4; i++) // DH-Parameetrite nimetuste Label-id.
122                 {
123                 System.Windows.Forms.Label Nms = new
                                System.Windows.Forms.Label();
124
125                     Nms.Name = "Nms" + countNms.ToString();
126
127                 Nms.Size = new System.Drawing.Size(50, 35);
128                 Nms.BorderStyle =
                       System.Windows.Forms.BorderStyle.Fixed3D;
129
130                     if (countNms == 1)
131                     {
132                     Nms.Location = new System.Drawing.Point(95, 60);
133                     }
134                     else
135                     {
136                     Nms.Location = new System.Drawing.Point(95 + 55 *
                      XNms, 60);
137                     }
138
139                         switch (countNms)
140                         {
141                         case 1:
142                         Nms.Text = "In xA \n(mm)";
143                         break; 144    case 2:
145                         Nms.Text = "In Al \n(deg)";
146                         break; 147    case 3:
148                         Nms.Text = "In Th \n(deg)";
149                         break; 150    case 4:
151                         Nms.Text = "In zS \n(mm)";
152                         break;
153                         }
154
155                 countNms++;
156                 XNms++;
157                 this.Controls.Add(Nms);
158
159             }
160
```

```
161                 int countAnser = 1;
162                 int XAnser = 0;
163
164                     for (int i = 0; i < 3; i++) // Positsiooni ja Orientetsiooni vastuste Label-id ja
                    TextBox-id.
165                     {
166                     System.Windows.Forms.Label cXYZ = new
                                System.Windows.Forms.Label();
167                     System.Windows.Forms.TextBox dV = new
                                System.Windows.Forms.TextBox();
168                     System.Windows.Forms.Label aXYZ = new
                                System.Windows.Forms.Label();
169                     System.Windows.Forms.TextBox dVa = new
                                System.Windows.Forms.TextBox();
170
171                     cXYZ.Name = "cXYZ" + countAnser.ToString();
172                     dV.Name = "dV" + countAnser.ToString();
173                     aXYZ.Name = "aXYZ" + countAnser.ToString();
174                     dVa.Name = "dVa" + countAnser.ToString();
175
176 cXYZ.Size = new System.Drawing.Size(60, 35); 177 cXYZ.BorderStyle =
                        System.Windows.Forms.BorderStyle.Fixed3D;
178
179                     dV.Size = new System.Drawing.Size(60, 35);
180                     dV.BorderStyle = System.Windows.Forms.BorderStyle.Fixed3D;
181                     dV.Text = "0";
182
183 aXYZ.Size = new System.Drawing.Size(60, 35); 184 aXYZ.BorderStyle =
                        System.Windows.Forms.BorderStyle.Fixed3D;
185
186                     dVa.Size = new System.Drawing.Size(60, 35);
187                     dVa.BorderStyle =
                        System.Windows.Forms.BorderStyle.Fixed3D;
188                     dVa.Text = "0";
189
190                         if (countAnser == 1)
191                         {
192                         cXYZ.Location = new System.Drawing.Point(315, 60);
193                         dV.Location = new System.Drawing.Point(315, 100);
194                         aXYZ.Location = new System.Drawing.Point(315, 140);
195                         dVa.Location = new System.Drawing.Point(315, 180);
196
197                         }
198                         else
```

```
199                         {
200                         cXYZ.Location = new System.Drawing.Point(315 + 65 *
                        XAnser, 60);
201                         dV.Location = new System.Drawing.Point(315 + 65 *
                        XAnser, 100);
202                         aXYZ.Location = new System.Drawing.Point(315 + 65 *
                        XAnser, 140);
203                         dVa.Location = new System.Drawing.Point(315 + 65 *
                        XAnser, 180);
204                         }
205
206                             switch (countAnser)
207                             {
208                             case 1:
209                             cXYZ.Text = "Out X (mm)";
210                             aXYZ.Text = "Out W (deg)";
211                             break; 212    case 2:
213                             cXYZ.Text = "Out Y (mm)";
214                             aXYZ.Text = "Out P (deg)";
215                             break; 216    case 3:
217                             cXYZ.Text = "Out Z (mm)";
218                             aXYZ.Text = "Out R (deg)";
219                             break;
220                             }
221
222                     countAnser++;
223                     XAnser++;
224                     this.Controls.Add(cXYZ);
225                     this.Controls.Add(dV);
226                     this.Controls.Add(aXYZ);
227                     this.Controls.Add(dVa);
228
229                 }
230
231 // IGM.
232 //-------------------------------------------------------------------------------------------------------------
                    ------------------------------------
233
234                 int countPoint = 1;
235                 int Xpoint = 0;
236
237                     for (int i = 0; i < 3; i++) // Positsiooni ja Orientetsiooni sisendi Label-id ja TextBox-id.
238                     {
239                     System.Windows.Forms.Label poXYZ = new
                                System.Windows.Forms.Label();
240                     System.Windows.Forms.TextBox pV = new
                                System.Windows.Forms.TextBox();
```

```
241                     System.Windows.Forms.Label paXYZ = new
                                System.Windows.Forms.Label();
242                     System.Windows.Forms.TextBox pVa = new
                                System.Windows.Forms.TextBox();
243
244                     poXYZ.Name = "poXYZ" + countPoint.ToString();
245                     pV.Name = "pV" + countPoint.ToString();
246                     pV.Text = "0";
247
248                     paXYZ.Name = "paXYZ" + countPoint.ToString();
249                     pVa.Name = "pVa" + countPoint.ToString();
250                     pVa.Text = "0";
251
252
253 poXYZ.Size = new System.Drawing.Size(60, 35); 254          poXYZ.BorderStyle =
                        System.Windows.Forms.BorderStyle.Fixed3D;
255                      poXYZ.TabStop = false;
256
257                     pV.Size = new System.Drawing.Size(60, 35);
258                     pV.BorderStyle = System.Windows.Forms.BorderStyle.Fixed3D;
259
260 paXYZ.Size = new System.Drawing.Size(60, 35); 261          paXYZ.BorderStyle =
                        System.Windows.Forms.BorderStyle.Fixed3D;
262
263                     pVa.Size = new System.Drawing.Size(60, 35);
264                     pVa.BorderStyle =
                        System.Windows.Forms.BorderStyle.Fixed3D;
265
266                       if (countAnser == 1)
267                       {
268                       poXYZ.Location = new System.Drawing.Point(575, 60);
269                       pV.Location = new System.Drawing.Point(575, 100);
270                       paXYZ.Location = new System.Drawing.Point(575, 140);
271                       pVa.Location = new System.Drawing.Point(575, 180);
272
273                       }
274                       else
275                       {
276                       poXYZ.Location = new System.Drawing.Point(575 + 65 *
                      Xpoint, 60);
277                       pV.Location = new System.Drawing.Point(575 + 65 *
                      Xpoint, 100);
278                       paXYZ.Location = new System.Drawing.Point(575 + 65 *
                      Xpoint, 140);
```

```
279                     pVa.Location = new System.Drawing.Point(575 + 65 *
                    Xpoint, 180);
280
281
282             }
283
284                     switch (countPoint)
285                     {
286                     case 1:
287                     poXYZ.Text = "In X \n(mm)";
288                     paXYZ.Text = "In W \n(deg)";
289                     break; 290    case 2:
291                     poXYZ.Text = "In Y \n(mm)";
292                     paXYZ.Text = "In P \n(deg)";
293                     break; 294    case 3:
295                     poXYZ.Text = "In Z \n(mm)";
296                     paXYZ.Text = "In R \n(deg)";
297                     break;
298
299             }
300                                                                                 ↵
301             countPoint++;                                                       ↵
302             Xpoint++;
303             this.Controls.Add(poXYZ);                                           ↵
304             this.Controls.Add(pV);
305             this.Controls.Add(paXYZ);                                           ↵
306             this.Controls.Add(pVa);
307                                                                                 ↵
308
309         }
310
311             for (int i = 0; i < 7; i++) // IGM Label-id ja TextBox-id.
312             {
313             System.Windows.Forms.Label iLbl = new
                  System.Windows.Forms.Label();   // Label J1 till Tool
314             System.Windows.Forms.TextBox ixA = new
                            System.Windows.Forms.TextBox();
315             System.Windows.Forms.TextBox iAl = new
                            System.Windows.Forms.TextBox();
316             System.Windows.Forms.TextBox izS = new
                            System.Windows.Forms.TextBox();
317             System.Windows.Forms.TextBox iTh1 = new
                            System.Windows.Forms.TextBox();
318
```

```
319                 if (invCount == 7)
320                 {
321                 iLbl.Text = "Tool";
322                 }
323                 else
324                 {
325                 iLbl.Text = "J" + invCount.ToString();
326                 }
327
328             iLbl.Name = "iLbl" + this.invCount.ToString();
329             iLbl.Size = new System.Drawing.Size(50, 35);
330
331             ixA.Name = "ixA" + this.invCount.ToString();
332             ixA.Text = "0";
333             ixA.Size = new System.Drawing.Size(50, 35);
334
335             iAl.Name = "iAl" + this.invCount.ToString();
```

```
336                 iAl.Text = "0";
337                 iAl.Size = new System.Drawing.Size(50, 35);
338
339                 izS.Name = "izS" + this.invCount.ToString();
340                 izS.Text = "0";
341                 izS.Size = new System.Drawing.Size(50, 35);
342
343                 iTh1.Name = "iTh1" + this.invCount.ToString();
344                 iTh1.Text = "0";
345                 iTh1.Size = new System.Drawing.Size(50, 35);
346
347                     if (invCount == 1)
348                     {
349                     iLbl.Location = new System.Drawing.Point(770, 100);
350                     ixA.Location = new System.Drawing.Point(825, 100);
351                     iAl.Location = new System.Drawing.Point(880, 100);
352                     izS.Location = new System.Drawing.Point(935, 100);
353                     iTh1.Location = new System.Drawing.Point(990, 100);
354                     }
355                     else
356                     {
357                     if (invCount != 7)                                    ↵
358                     {
359                     iLbl.Location = new System.Drawing.Point(770, 100     ↵
                    + 40 * invY);
360                     ixA.Location = new System.Drawing.Point(825, 100 +    ↵
                      40   * invY);
361                     iAl.Location = new System.Drawing.Point(880, 100 +    ↵
                      40   * invY);
362                     izS.Location = new System.Drawing.Point(935, 100 +    ↵
                      40   * invY);
363                     iTh1.Location = new System.Drawing.Point(990, 100
                    + 40 * invY);
364                     }
365                     else
366                     {                                                     ↵
367                     invY++;
368                     iLbl.Location = new System.Drawing.Point(770, 100     ↵
                    + 40 * invY);
369                     ixA.Location = new System.Drawing.Point(825, 100 +    ↵
                      40   * invY);
370                     izS.Location = new System.Drawing.Point(935, 100 +
                      40   * invY);
371                     }                                                     ↵

                                                                            ↵
```

```
372                     }
373
374             iLbl.BorderStyle =
                    System.Windows.Forms.BorderStyle.Fixed3D;
375             ixA.BorderStyle = System.Windows.Forms.BorderStyle.Fixed3D;
376             iAl.BorderStyle =
                    System.Windows.Forms.BorderStyle.Fixed3D;
377             izS.BorderStyle =
                    System.Windows.Forms.BorderStyle.Fixed3D;
378             iTh1.BorderStyle =
                    System.Windows.Forms.BorderStyle.Fixed3D;
379
380             this.Controls.Add(iLbl);
381             this.Controls.Add(ixA);
382
383                 this.Controls.Add(izS);
384                 if (invY != 7)
385                 {
386                 this.Controls.Add(iAl);
387                 this.Controls.Add(iTh1);
388                 }
389
390
391
392                     if (invCount != 1 && invCount != 2 && invCount != 3 && invCount != 7)
                        // Thetade vastuste TextBox-id.
393                     {
394                     System.Windows.Forms.TextBox iTh2 = new
                            System.Windows.Forms.TextBox();
395
396                     iTh2.Name = "iTh2" + invCount.ToString();
397                     iTh2.Text = "0";
398                     iTh2.Size = new System.Drawing.Size(50, 35);
399                     iTh2.Location = new System.Drawing.Point(1045, 100
                    + 40 * invY);
400                     iTh2.BorderStyle =
                    System.Windows.Forms.BorderStyle.Fixed3D;
401                     this.Controls.Add(iTh2);
402                     }
403
```

```
404             if (invCount == 6)
405             {
406             System.Windows.Forms.TextBox iTh3 = new
                        System.Windows.Forms.TextBox();
407             System.Windows.Forms.TextBox iTh4 = new
                    System.Windows.Forms.TextBox();
408
409             iTh3.Name = "iTh3" + invCount.ToString();
410             iTh3.Text = "0";
411             iTh3.Size = new System.Drawing.Size(50, 35);
412             iTh3.Location = new System.Drawing.Point(1100, 100 +
            40 * invY);
413             iTh3.BorderStyle = System.Windows.Forms.BorderStyle.Fixed3D;
414
415             iTh4.Name = "iTh4" + invCount.ToString();
416             iTh4.Text = "0";
417             iTh4.Size = new System.Drawing.Size(50, 35);
418             iTh4.Location = new System.Drawing.Point(1155, 100 +
            40 * invY);
419             iTh4.BorderStyle =
            System.Windows.Forms.BorderStyle.Fixed3D;
420
421             System.Windows.Forms.TextBox i2Th16 = new
                        System.Windows.Forms.TextBox();
422             System.Windows.Forms.TextBox i2Th26 = new
                        System.Windows.Forms.TextBox();
423             System.Windows.Forms.TextBox i2Th36 = new
                    System.Windows.Forms.TextBox();
424             System.Windows.Forms.TextBox i2Th46 = new
                        System.Windows.Forms.TextBox();
425
426             i2Th16.Name = "i2Th16" + invCount.ToString();
427             i2Th16.Text = "0";
428             i2Th16.Size = new System.Drawing.Size(50, 35);
429             i2Th16.Location = new System.Drawing.Point(990, 100 +
            40 * 6);
430             i2Th16.BorderStyle =
            System.Windows.Forms.BorderStyle.Fixed3D;
431
432             i2Th26.Name = "i2Th26" + invCount.ToString();
```

```
433                 i2Th26.Text = "0";
434                 i2Th26.Size = new System.Drawing.Size(50, 35);
435                 i2Th26.Location = new System.Drawing.Point(1045, 100 +
                    40 * 6);
436                 i2Th26.BorderStyle =
                System.Windows.Forms.BorderStyle.Fixed3D;
437
438                 i2Th36.Name = "i2Th36" + invCount.ToString();
439                 i2Th36.Text = "0";
440                 i2Th36.Size = new System.Drawing.Size(50, 35);
441                 i2Th36.Location = new System.Drawing.Point(1100, 100 +
                    40 * 6);
442                 i2Th36.BorderStyle =
                System.Windows.Forms.BorderStyle.Fixed3D;
443
444                 i2Th46.Name = "i2Th46" + invCount.ToString();
445                 i2Th46.Text = "0";
446                 i2Th46.Size = new System.Drawing.Size(50, 35);
447                 i2Th46.Location = new System.Drawing.Point(1155, 100 +
                    40 * 6);
448                 i2Th46.BorderStyle =
```

```
                    System.Windows.Forms.BorderStyle.Fixed3D;
449
450                     this.Controls.Add(iTh3);
451                     this.Controls.Add(iTh4);
452                     this.Controls.Add(i2Th16);
453                     this.Controls.Add(i2Th26);
454                     this.Controls.Add(i2Th36);
455                     this.Controls.Add(i2Th46);
456                     }
457
458                 invY++;
459                 invCount++;
460
461             }
462
463             int icountLbl = 1;
464             int iXlbl = 0;
465
466                 for (int i = 0; i < 7; i++) // Positsiooni ja Orientetsiooni nimede Label-id ja
                    TextBox-id.
467                 {
468                 System.Windows.Forms.Label iNms = new
                                System.Windows.Forms.Label();
469                 ;
470
471                     iNms.Name = "iNms" + icountLbl.ToString();
472
473
474 iNms.Size = new System.Drawing.Size(50, 35); 475  iNms.BorderStyle =
                        System.Windows.Forms.BorderStyle.Fixed3D;
476
477                     if (icountLbl == 1)
478                     {
479                     iNms.Location = new System.Drawing.Point(825, 60);
480                     }
481                     else
482                     {
483                     iNms.Location = new System.Drawing.Point(825 + 55 *
                    iXlbl, 60);
484                     }
485
486                         switch (icountLbl) // Selles Switch-is tänu sellele, et
```

```
                    Theta vastusi saab olla rohkem kui üks. Theta nimetus on muudetud Jv. peale.
487                 {
488                 case 1:
489                 iNms.Text = "In xA \n(mm)";
490                 break;
491                 case 2:
492                 iNms.Text = "In Al \n(deg)";
493                 break; 494    case 3:
495                 iNms.Text = "In zS \n(mm)";
496                 break; 497    case 4:
498                 iNms.Text = "Out Jv1 \n(deg)";
499                 break; 500    case 5:
501                 iNms.Text = "Out Jv2 \n(deg)";
502                 break; 503    case 6:
504                 iNms.Text = "Out Jv3 \n(deg)";
505                 break; 506    case 7:
507                 iNms.Text = "Out Jv4 \n(deg)";
508                 break;
509                 }
510                 icountLbl++;
511                 iXlbl++;
512                 this.Controls.Add(iNms);
513                 }
514                 }
515
516 //Sündmus mille tagajärjel kalkulaator arvutab vastust.
517 //------------------------------------------------------------------------------------------------------------------------------
        ------------------------------

518
519         private void doublbCli(object sender, System.EventArgs e) // Topeltklõps algatab arvutusi.
520         {
521
522             this.Focus();
523
524             string A1str = (this.Controls.Find("xA1", true)[0]).Text; // Tekkitame String või rida
                formaadiga muutujad ja täidame teda Label-i väärtusega.
525             string A2str = (this.Controls.Find("xA2", true)[0]).Text; // a väärtus (mm).
526             string A3str = (this.Controls.Find("xA3", true)[0]).Text;
527             string A4str = (this.Controls.Find("xA4", true)[0]).Text;
528             string A5str = (this.Controls.Find("xA5", true)[0]).Text;
529             string A6str = (this.Controls.Find("xA6", true)[0]).Text;
530             string A7str = (this.Controls.Find("xA7", true)[0]).Text;
531             double a1 = double.Parse(A1str); // Konverteerime String või rida - double formaati.
532             double a2 = double.Parse(A2str);
533             double a3 = double.Parse(A3str);
534             double a4 = double.Parse(A4str);
```

```
535             double a5 = double.Parse(A5str);
536             double a6 = double.Parse(A6str);
537             double a7 = double.Parse(A7str);
538
539             string AL1str = (this.Controls.Find("Al1", true)[0]).Text; // Alpha väärtus (deg).
540             string AL2str = (this.Controls.Find("Al2", true)[0]).Text;
541             string AL3str = (this.Controls.Find("Al3", true)[0]).Text;
542             string AL4str = (this.Controls.Find("Al4", true)[0]).Text;
543             string AL5str = (this.Controls.Find("Al5", true)[0]).Text;
544             string AL6str = (this.Controls.Find("Al6", true)[0]).Text;
545             double al1 = double.Parse(AL1str) * Math.PI / 180.0;
546             double al2 = double.Parse(AL2str) * Math.PI / 180.0;
547             double al3 = double.Parse(AL3str) * Math.PI / 180.0;
548             double al4 = double.Parse(AL4str) * Math.PI / 180.0;
549             double al5 = double.Parse(AL5str) * Math.PI / 180.0;
550             double al6 = double.Parse(AL6str) * Math.PI / 180.0;
551             double al7 = 0.0; // Tööriista parameeter mis võrdub nulliga sest seda ei kasuta.
552
553             string TH1str = (this.Controls.Find("Th1", true)[0]).Text; // Theta väärtus (deg).
554             string TH2str = (this.Controls.Find("Th2", true)[0]).Text;
555             string TH3str = (this.Controls.Find("Th3", true)[0]).Text;
556             string TH4str = (this.Controls.Find("Th4", true)[0]).Text;
557             string TH5str = (this.Controls.Find("Th5", true)[0]).Text;
558             string TH6str = (this.Controls.Find("Th6", true)[0]).Text;
559             double th1 = double.Parse(TH1str) * Math.PI / 180.0;
560             double th2 = (Math.PI / 2) - (double.Parse(TH2str) * Math.PI / 180.0);
561             double th3 = (double.Parse(TH2str) + double.Parse(TH3str)) * Math.PI / 180.0;
562             double th4 = double.Parse(TH4str) * Math.PI / 180.0;
563             double th5 = double.Parse(TH5str) * Math.PI / 180.0;
564             double th6 = double.Parse(TH6str) * Math.PI / 180.0;
565             double th7 = 0.0;
566
567             string ZS1str = (this.Controls.Find("zS1", true)[0]).Text; // s väärtus (mm)
568             string ZS2str = (this.Controls.Find("zS2", true)[0]).Text;
569             string ZS3str = (this.Controls.Find("zS3", true)[0]).Text;
570             string ZS4str = (this.Controls.Find("zS4", true)[0]).Text;
571             string ZS5str = (this.Controls.Find("zS5", true)[0]).Text;
572             string ZS6str = (this.Controls.Find("zS6", true)[0]).Text;
573             string ZS7str = (this.Controls.Find("zS7", true)[0]).Text;
574             double zs1 = double.Parse(ZS1str);
575             double zs2 = double.Parse(ZS2str);
576             double zs3 = double.Parse(ZS3str);
577             double zs4 = double.Parse(ZS4str);
578             double zs5 = double.Parse(ZS5str);
579             double zs6 = double.Parse(ZS6str);
580             double zs7 = double.Parse(ZS7str);
```

```
581
582             var dhMtx1 = Array.CreateInstance(typeof(double), 16); // Tekkitame maatriksi.
583             dhMtx1.SetValue(Math.Cos(th1), 0); dhMtx1.SetValue(-(Math.Sin
                    (th1)) * Math.Cos(al1), 4); dhMtx1.SetValue(Math.Sin(th1) *
                    Math.Sin(al1), 8); dhMtx1.SetValue(a1 * Math.Cos(th1), 12);
584             dhMtx1.SetValue(Math.Sin(th1), 1); dhMtx1.SetValue(Math.Cos
                    (th1) * Math.Cos(al1), 5); dhMtx1.SetValue(-(Math.Cos(th1))
                    * Math.Sin(al1), 9); dhMtx1.SetValue(a1 * Math.Sin(th1), 13);
585             dhMtx1.SetValue(0, 2); dhMtx1.SetValue(Math.Sin(al1), 6);
                dhMtx1.SetValue(Math.Cos(al1), 10); dhMtx1.SetValue(zs1, 14);
586             dhMtx1.SetValue(0, 3); dhMtx1.SetValue(0, 7); dhMtx1.SetValue
                    (0, 11); dhMtx1.SetValue(1, 15);
587
588             var dhMtx2 = Array.CreateInstance(typeof(double), 16);
589             dhMtx2.SetValue(Math.Cos(th2), 0); dhMtx2.SetValue(-(Math.Sin
                    (th2)) * Math.Cos(al2), 4); dhMtx2.SetValue(Math.Sin(th2) *
                    Math.Sin(al2), 8); dhMtx2.SetValue(a2 * Math.Cos(th2), 12);
590             dhMtx2.SetValue(Math.Sin(th2), 1); dhMtx2.SetValue(Math.Cos
                    (th2) * Math.Cos(al2), 5); dhMtx2.SetValue(-(Math.Cos(th2))
                    * Math.Sin(al2), 9); dhMtx2.SetValue(a2 * Math.Sin(th2), 13);
591             dhMtx2.SetValue(0, 2); dhMtx2.SetValue(Math.Sin(al2), 6);
                dhMtx2.SetValue(Math.Cos(al2), 10); dhMtx2.SetValue(zs2, 14);
592             dhMtx2.SetValue(0, 3); dhMtx2.SetValue(0, 7); dhMtx2.SetValue
                    (0, 11); dhMtx2.SetValue(1, 15);
593
594             var dhMtx3 = Array.CreateInstance(typeof(double), 16);
595             dhMtx3.SetValue(Math.Cos(th3), 0); dhMtx3.SetValue(-(Math.Sin
                    (th3)) * Math.Cos(al3), 4); dhMtx3.SetValue(Math.Sin(th3) *
                    Math.Sin(al3), 8); dhMtx3.SetValue(a3 * Math.Cos(th3), 12);
596             dhMtx3.SetValue(Math.Sin(th3), 1); dhMtx3.SetValue(Math.Cos
                    (th3) * Math.Cos(al3), 5); dhMtx3.SetValue(-(Math.Cos(th3))
                    * Math.Sin(al3), 9); dhMtx3.SetValue(a3 * Math.Sin(th3), 13);
597             dhMtx3.SetValue(0, 2); dhMtx3.SetValue(Math.Sin(al3), 6); dhMtx3.SetValue(Math.Cos(al3), 10);
                dhMtx3.SetValue(zs3, 14);
598             dhMtx3.SetValue(0, 3); dhMtx3.SetValue(0, 7); dhMtx3.SetValue
                    (0, 11); dhMtx3.SetValue(1, 15); 599
600             var dhMtx4 = Array.CreateInstance(typeof(double), 16);
601             dhMtx4.SetValue(Math.Cos(th4), 0); dhMtx4.SetValue(-(Math.Sin
                    (th4)) * Math.Cos(al4), 4); dhMtx4.SetValue(Math.Sin(th4) *
                    Math.Sin(al4), 8); dhMtx4.SetValue(a4 * Math.Cos(th4), 12);
602             dhMtx4.SetValue(Math.Sin(th4), 1); dhMtx4.SetValue(Math.Cos
                    (th4) * Math.Cos(al4), 5); dhMtx4.SetValue(-(Math.Cos(th4))
                    * Math.Sin(al4), 9); dhMtx4.SetValue(a4 * Math.Sin(th4), 13);
603             dhMtx4.SetValue(0, 2); dhMtx4.SetValue(Math.Sin(al4), 6);
                dhMtx4.SetValue(Math.Cos(al4), 10); dhMtx4.SetValue(zs4, 14);
604             dhMtx4.SetValue(0, 3); dhMtx4.SetValue(0, 7); dhMtx4.SetValue
```

```
(0, 11); dhMtx4.SetValue(1, 15);
605
606     var dhMtx5 = Array.CreateInstance(typeof(double), 16);
607     dhMtx5.SetValue(Math.Cos(th5), 0); dhMtx5.SetValue(-(Math.Sin
        (th5)) * Math.Cos(al5), 4); dhMtx5.SetValue(Math.Sin(th5) *
        Math.Sin(al5), 8); dhMtx5.SetValue(a5 * Math.Cos(th5), 12);
608     dhMtx5.SetValue(Math.Sin(th5), 1); dhMtx5.SetValue(Math.Cos
        (th5) * Math.Cos(al5), 5); dhMtx5.SetValue(-(Math.Cos(th5))
        * Math.Sin(al5), 9); dhMtx5.SetValue(a5 * Math.Sin(th5), 13);
609     dhMtx5.SetValue(0, 2); dhMtx5.SetValue(Math.Sin(al5), 6);
        dhMtx5.SetValue(Math.Cos(al5), 10); dhMtx5.SetValue(zs5, 14);
610     dhMtx5.SetValue(0, 3); dhMtx5.SetValue(0, 7); dhMtx5.SetValue
        (0, 11); dhMtx5.SetValue(1, 15);
611
612     var dhMtx6 = Array.CreateInstance(typeof(double), 16);
613     dhMtx6.SetValue(Math.Cos(th6), 0); dhMtx6.SetValue(-(Math.Sin
        (th6)) * Math.Cos(al6), 4); dhMtx6.SetValue(Math.Sin(th6) *
        Math.Sin(al6), 8); dhMtx6.SetValue(a6 * Math.Cos(th6), 12);
614     dhMtx6.SetValue(Math.Sin(th6), 1); dhMtx6.SetValue(Math.Cos
        (th6) * Math.Cos(al6), 5); dhMtx6.SetValue(-(Math.Cos(th6))
        * Math.Sin(al6), 9); dhMtx6.SetValue(a6 * Math.Sin(th6), 13);
615     dhMtx6.SetValue(0, 2); dhMtx6.SetValue(Math.Sin(al6), 6);
        dhMtx6.SetValue(Math.Cos(al6), 10); dhMtx6.SetValue(zs6, 14);
616     dhMtx6.SetValue(0, 3); dhMtx6.SetValue(0, 7); dhMtx6.SetValue
        (0, 11); dhMtx6.SetValue(1, 15);
617
618     var dhMtx7 = Array.CreateInstance(typeof(double), 16);
619     dhMtx7.SetValue(Math.Cos(th7), 0); dhMtx7.SetValue(-(Math.Sin
        (th7)) * Math.Cos(al7), 4); dhMtx7.SetValue(Math.Sin(th7) *
        Math.Sin(al7), 8); dhMtx7.SetValue(a7 * Math.Cos(th7), 12);
620     dhMtx7.SetValue(Math.Sin(th7), 1); dhMtx7.SetValue(Math.Cos
        (th7) * Math.Cos(al7), 5); dhMtx7.SetValue(-(Math.Cos(th7))
        * Math.Sin(al7), 9); dhMtx7.SetValue(a7 * Math.Sin(th7),
```

```
    13);
621 dhMtx7.SetValue(0, 2); dhMtx7.SetValue(Math.Sin(al7), 6);
    dhMtx7.SetValue(Math.Cos(al7), 10); dhMtx7.SetValue(zs7, 14);
622 dhMtx7.SetValue(0, 3); dhMtx7.SetValue(0, 7); dhMtx7.SetValue
    (0, 11); dhMtx7.SetValue(1, 15);
623
624 var Matrix1 = Array.CreateInstance(typeof(double), 16); // Tekkitame uue tühja
    maatriksi.
625 var Matrix2 = Array.CreateInstance(typeof(double), 16);
626 var Matrix3 = Array.CreateInstance(typeof(double), 16);
627 var Matrix4 = Array.CreateInstance(typeof(double), 16);
628 var Matrix5 = Array.CreateInstance(typeof(double), 16);
629 var Matrix6 = Array.CreateInstance(typeof(double), 16); 630
631 mulMtxByMtx(ref dhMtx1, ref dhMtx2, ref Matrix1); // Korrutame tekkitatud maatrikseid õiges
    järjekorras.
632 mulMtxByMtx(ref Matrix1, ref dhMtx3, ref Matrix2);
633 mulMtxByMtx(ref Matrix2, ref dhMtx4, ref Matrix3);
634 mulMtxByMtx(ref Matrix3, ref dhMtx5, ref Matrix4);
635 mulMtxByMtx(ref Matrix4, ref dhMtx6, ref Matrix5);
636 mulMtxByMtx(ref Matrix5, ref dhMtx7, ref Matrix6);
637 double tW = 0;
638 double tP = 0;
639 double tR = 0;
640 mtx2xyzwpr(ref Matrix6, w: ref tW, ref tP, ref tR);
641
642                                 void mulMtxByMtx(ref Array mtxIn1, ref Array
                                    mtxIn2, ref Array mtxOut) // Maatriksite 4x4 korrutamise tehe.
643                                 {
644                                 for (int i = 0; i < 4; i++)
645                                 {
646                                 for (int j = 0; j < 4; j++)
647                                 {
648                                 mtxOut.SetValue((double)mtxIn1.GetValue(i) *
                                (double)mtxIn2.GetValue(j * 4)
649                                 + (double)mtxIn1.GetValue(i + 4) *
                                (double)mtxIn2.GetValue(j * 4 + 1)
650                                 + (double)mtxIn1.GetValue(i + 8) *
                                (double)mtxIn2.GetValue(j * 4 + 2)
651                                 + (double)mtxIn1.GetValue(i + 12) *
                                (double)mtxIn2.GetValue(j * 4 + 3),
652                                 i + j * 4);
653                                 }
654                                 }
655                                 }
656
```

```
657     void mtx2xyzwpr(ref Array mtx, ref double w, ref double p, ref double r) // Euleri nurkade arvutamine.
658     {
659
660                                     r = Math.Atan2((double)mtx.GetValue(1), (double)
                                    mtx.GetValue(0)) * 180 / Math.PI; // Z nurga arvutamine. 661       p             =
Math.Atan2(-(double)mtx.GetValue(2), Math.Sqrt(1 -
                                    (double)mtx.GetValue(2) * (double)mtx.GetValue(2))) * 180 / Math.PI; // Y
                                    nurga arvutamine.
662                                 w = Math.Atan2((double)mtx.GetValue(6), (double)
                                    mtx.GetValue(10)) * 180 / Math.PI; // X nurga arvutamine.
663                                 }
664
665                         this.Controls["dV1"].Text = String.Format("{0:N3}",
                            Convert.ToDouble(Matrix6.GetValue(12))); // X positsioon.
666                         this.Controls["dV2"].Text = String.Format("{0:N3}",
                            Convert.ToDouble(Matrix6.GetValue(13))); // Y positsioon.
667                         this.Controls["dV3"].Text = String.Format("{0:N3}",
                            Convert.ToDouble(Matrix6.GetValue(14))); // Z positsioon.
668
669                         this.Controls["dVa1"].Text = String.Format("{0:N3}",
                            Convert.ToDouble(tW)); // X nurga arvutamine.
670                         this.Controls["dVa2"].Text = String.Format("{0:N3}",
                            Convert.ToDouble(tP)); // Y nurga arvutamine.
671                         this.Controls["dVa3"].Text = String.Format("{0:N3}",
                            Convert.ToDouble(tR)); // Z nurga arvutamine. 672
673     // IGM arvutamine.
674     //-------------------------------------------------------------------------------------------------------------
                            ------------------------------------
675
676                         string pV1str = (this.Controls.Find("pV1", true)[0]).Text; // IGM positsioon.
677                         string pV2str = (this.Controls.Find("pV2", true)[0]).Text;
678                         string pV3str = (this.Controls.Find("pV3", true)[0]).Text;
679                         double pV1p = double.Parse(pV1str);
680                         double pV2p = double.Parse(pV2str);
681                         double pV3p = double.Parse(pV3str);
682
683                         string pVa1str = (this.Controls.Find("pVa1", true)
                            [0]).Text; // IGM pöörad.
684                         string pVa2str = (this.Controls.Find("pVa2", true)[0]).Text;
685                         string pVa3str = (this.Controls.Find("pVa3", true)[0]).Text;
686                         double pVa1d = double.Parse(pVa1str);
687                         double pVa2d = double.Parse(pVa2str);
688                         double pVa3d = double.Parse(pVa3str);
689                         double pVa1r = pVa1d * (Math.PI / 180.0);
690                         double pVa2r = pVa2d * (Math.PI / 180.0);
691                         double pVa3r = pVa3d * (Math.PI / 180.0);
```

```
692
693             string iA1str = (this.Controls.Find("ixA1", true)[0]).Text; // a väärtus (mm).
694             string iA2str = (this.Controls.Find("ixA2", true)[0]).Text;
695             string iA3str = (this.Controls.Find("ixA3", true)[0]).Text;
696             string iA4str = (this.Controls.Find("ixA4", true)[0]).Text;
697             string iA5str = (this.Controls.Find("ixA5", true)[0]).Text;
698             string iA6str = (this.Controls.Find("ixA6", true)[0]).Text;
699             string iA7str = (this.Controls.Find("ixA7", true)[0]).Text;
700             double ia1 = double.Parse(iA1str);
701             double ia2 = double.Parse(iA2str);
702             double ia3 = double.Parse(iA3str);
703             double ia4 = double.Parse(iA4str);
704             double ia5 = double.Parse(iA5str);
705             double ia6 = double.Parse(iA6str);
706             double ia7 = double.Parse(iA7str);
707
708             string iAL1str = (this.Controls.Find("iAl1", true)
                  [0]).Text; // Alpha väärtus (deg).
709             string iAL2str = (this.Controls.Find("iAl2", true)[0]).Text;
710             string iAL3str = (this.Controls.Find("iAl3", true)[0]).Text;
711             string iAL4str = (this.Controls.Find("iAl4", true)[0]).Text;
712             string iAL5str = (this.Controls.Find("iAl5", true)[0]).Text;
713             string iAL6str = (this.Controls.Find("iAl6", true)[0]).Text;
714             double ial1 = double.Parse(iAL1str) * Math.PI / 180.0;
715             double ial2 = double.Parse(iAL2str) * Math.PI / 180.0;
716             double ial3 = double.Parse(iAL3str) * Math.PI / 180.0;
717             double ial4 = double.Parse(iAL4str) * Math.PI / 180.0;
718             double ial5 = double.Parse(iAL5str) * Math.PI / 180.0;
719             double ial6 = double.Parse(iAL6str) * Math.PI / 180.0; 720
721             string iZS1str = (this.Controls.Find("izS1", true)
                  [0]).Text; // s väärtus (mm).
722             string iZS2str = (this.Controls.Find("izS2", true)[0]).Text;
723             string iZS3str = (this.Controls.Find("izS3", true)[0]).Text;
724             string iZS4str = (this.Controls.Find("izS4", true)[0]).Text;
725             string iZS5str = (this.Controls.Find("izS5", true)[0]).Text;
726             string iZS6str = (this.Controls.Find("izS6", true)[0]).Text;
727             string iZS7str = (this.Controls.Find("izS7", true)[0]).Text;
728             double izs1 = double.Parse(iZS1str);
729             double izs2 = double.Parse(iZS2str);
730             double izs3 = double.Parse(iZS3str);
731             double izs4 = double.Parse(iZS4str);
732             double izs5 = double.Parse(iZS5str);
733             double izs6 = double.Parse(iZS6str);
734             double izs7 = double.Parse(iZS7str);
735
736             var iMtxJ6 = Array.CreateInstance(typeof(double), 16); // Formeerin maatriksi mis
```

koosneb algandmetest.

```
737         iMtxJ6.SetValue(Math.Cos(pVa3r) * Math.Cos(pVa2r), 0);
            iMtxJ6.SetValue(Math.Cos(pVa3r) * Math.Sin(pVa2r) * Math.Sin
               (pVa1r) - Math.Sin(pVa3r) * Math.Cos(pVa1r), 4);
               iMtxJ6.SetValue(Math.Cos(pVa3r) * Math.Sin(pVa2r) * Math.Cos
               (pVa1r) + Math.Sin(pVa3r) * Math.Sin(pVa1r), 8);
               iMtxJ6.SetValue(pV1p, 12);
738         iMtxJ6.SetValue(Math.Sin(pVa3r) * Math.Cos(pVa2r), 1);
            iMtxJ6.SetValue(Math.Sin(pVa3r) * Math.Sin(pVa2r) * Math.Sin
               (pVa1r) + Math.Cos(pVa3r) * Math.Cos(pVa1r), 5);
               iMtxJ6.SetValue(Math.Sin(pVa3r) * Math.Sin(pVa2r) * Math.Cos
               (pVa1r) - Math.Cos(pVa3r) * Math.Sin(pVa1r), 9); iMtxJ6.SetValue(pV2p, 13);
739         iMtxJ6.SetValue(-Math.Sin(pVa2r), 2); iMtxJ6.SetValue(Math.Cos (pVa2r) *
            Math.Sin(pVa1r), 6); iMtxJ6.SetValue(Math.Cos
               (pVa2r) * Math.Cos(pVa1r), 10); iMtxJ6.SetValue(pV3p, 14);
740         iMtxJ6.SetValue(0, 3); iMtxJ6.SetValue(0, 7); iMtxJ6.SetValue
               (0, 11); iMtxJ6.SetValue(1, 15);
741
742         var iMtxTofS = Array.CreateInstance(typeof(double), 16); // Formeerin maatriksi
            tööriista nihkega.
743         iMtxTofS.SetValue(-ia7, 0);
744         iMtxTofS.SetValue(0, 1);
745         iMtxTofS.SetValue(-izs7, 2);
746         iMtxTofS.SetValue(1, 3);
747
748         var iMtxP6 = Array.CreateInstance(typeof(double), 16); // Formeerin maatriksi kuuenda
            koordinaatteljestiku nihkega viiendast.
749         iMtxP6.SetValue(0, 0);
750         iMtxP6.SetValue(0, 1);
751         iMtxP6.SetValue(izs6, 2);
752         iMtxP6.SetValue(1, 3);
753
754                 var MatrxEnd_ToolOfs = Array.CreateInstance(typeof(double),
               16);
755
756             mulMultiMtx(ref iMtxJ6, ref iMtxTofS, ref
               MatrxEnd_ToolOfs); // Korrutades saame kuuenda koordinaatteljestiku positsiooni.
757
758         var ImagMrx = Array.CreateInstance(typeof(double), 16);
759         ImagMrx.SetValue(Math.Cos(pVa3r) * Math.Cos(pVa2r), 0);
            ImagMrx.SetValue(Math.Cos(pVa3r) * Math.Sin(pVa2r) *
               Math.Sin(pVa1r) - Math.Sin(pVa3r) * Math.Cos(pVa1r), 4);
               ImagMrx.SetValue(Math.Cos(pVa3r) * Math.Sin(pVa2r) *
               Math.Cos(pVa1r) + Math.Sin(pVa3r) * Math.Sin(pVa1r), 8);
               ImagMrx.SetValue((double)MatrxEnd_ToolOfs.GetValue(0), 12);
760         ImagMrx.SetValue(Math.Sin(pVa3r) * Math.Cos(pVa2r), 1);
               ImagMrx.SetValue(Math.Sin(pVa3r) * Math.Sin(pVa2r) *
```

```
    Math.Sin(pVa1r) + Math.Cos(pVa3r) * Math.Cos(pVa1r), 5);
    ImagMrx.SetValue(Math.Sin(pVa3r) * Math.Sin(pVa2r) *
    Math.Cos(pVa1r) - Math.Cos(pVa3r) * Math.Sin(pVa1r), 9);
    ImagMrx.SetValue((double)MatrxEnd_ToolOfs.GetValue(1), 13);
761 ImagMrx.SetValue(-Math.Sin(pVa2r), 2); ImagMrx.SetValue
    (Math.Cos(pVa2r) * Math.Sin(pVa1r), 6); ImagMrx.SetValue
    (Math.Cos(pVa2r) * Math.Cos(pVa1r), 10); ImagMrx.SetValue
    ((double)MatrxEnd_ToolOfs.GetValue(2), 14);
762 ImagMrx.SetValue(0, 3); ImagMrx.SetValue(0, 7);
    ImagMrx.SetValue(0, 11); ImagMrx.SetValue(1, 15);
763
764 var MatrxJ5 = Array.CreateInstance(typeof(double), 16); // 5 koordinaatteljestiku keskpunkt.
765
766         mulMultiMtx(ref ImagMrx, ref iMtxP6, ref MatrxJ5);
767
768 void mulMultiMtx(ref Array ImtxIn1, ref Array ImtxIn2, ref Array ImtxOut) // Maatriksite 4x4 ja
    4x1 korrutamise tehe.
769 {
770
771                 for (int i = 0; i < 4; i++)
772                 {
773                 ImtxOut.SetValue((double)ImtxIn1.GetValue(i) *
                (double)ImtxIn2.GetValue(0)
774                 + (double)ImtxIn1.GetValue(i + 4) *
                (double)ImtxIn2.GetValue(1)
775                 + (double)ImtxIn1.GetValue(i + 8) *
                (double)ImtxIn2.GetValue(2)
776                 + (double)ImtxIn1.GetValue(i + 12) *
                (double)ImtxIn2.GetValue(3),
777                 i);
778                 }
779                 }
780
781 double rJ1 = Math.Atan2((double)MatrxJ5.GetValue(1), (double)
    MatrxJ5.GetValue(0)); // Theta 1 otsimine.
782 double dJ1 = (180 / Math.PI) * rJ1;
783 this.Controls["iTh11"].Text = String.Format("{0:N3}", Convert.ToDouble(dJ1));
784
785                 double RJ2 = Math.Sqrt(((double)MatrxJ5.GetValue(0) * (double)
                MatrxJ5.GetValue(0)) + ((double)MatrxJ5.GetValue(1) *
                (double)MatrxJ5.GetValue(1))) - Convert.ToDouble(ia1); // Theta 2
                otsimine.
786                 double xJ2 = ia1 * Math.Cos(rJ1);
787                 double yJ2 = ia1 * Math.Sin(rJ1);
```

```
788                                 double zJ2 = 0;
789                                 double b = Math.Sqrt((((double)MatrxJ5.GetValue(0) - xJ2) *
                        ((double)MatrxJ5.GetValue(0) - xJ2)) +
790                                 (((double)MatrxJ5.GetValue(1) - yJ2) *
                                ((double)MatrxJ5.GetValue(1) - yJ2)) +
791                                 (((double)MatrxJ5.GetValue(2) - zJ2) *
                                ((double)MatrxJ5.GetValue(2) - zJ2)));
792                                 double k = Math.Sqrt((ia3 * ia3) + (izs4 * izs4));
793                                 double althaK =
                                    Math.Atan2((double)MatrxJ5.GetValue(2), RJ2);
794                                 double bethaK = Math.Acos(((ia2 * ia2) + (b * b) - (k * k)) /
                                    (2 * ia2 * b));
795                                 double rJ2 = ((Math.PI / 2) - bethaK - althaK);
796                                 double dJ2 = (180 / Math.PI) * rJ2;
797                                 this.Controls["iTh12"].Text = String.Format("{0:N3}",
                                    Convert.ToDouble(dJ2));
798
799                 double gammaJ3 = Math.Acos(((ia2 * ia2) + (k * k) - (b * b)) / (2 * ia2 * k)); // Theta 2 otsimine.
800                 double fiJ3 = Math.Atan2(-izs4, ia3);
801                 double rJ3 = -(Math.PI - gammaJ3 - fiJ3 + rJ2);
802                 double dJ3 = (180 / Math.PI) * rJ3;
803                 this.Controls["iTh13"].Text = String.Format("{0:N3}", Convert.ToDouble(dJ3));
804
805                 var idhMtx1 = Array.CreateInstance(typeof(double), 16); // Maatriks 1.
806                 idhMtx1.SetValue(Math.Cos(rJ1), 0); idhMtx1.SetValue(-
                    (Math.Sin(rJ1)) * Math.Cos(ial1), 4); idhMtx1.SetValue
                    (Math.Sin(rJ1) * Math.Sin(ial1), 8); idhMtx1.SetValue(ia1 * Math.Cos(rJ1), 12);
807                 idhMtx1.SetValue(Math.Sin(rJ1), 1); idhMtx1.SetValue(Math.Cos (rJ1) *
                    Math.Cos(ial1), 5); idhMtx1.SetValue(-(Math.Cos
                    (rJ1)) * Math.Sin(ial1), 9); idhMtx1.SetValue(ia1 * Math.Sin (rJ1), 13);
808                 idhMtx1.SetValue(0, 2); idhMtx1.SetValue(Math.Sin(ial1), 6);
                    idhMtx1.SetValue(Math.Cos(ial1), 10); idhMtx1.SetValue(izs1, 14);
809                 idhMtx1.SetValue(0, 3); idhMtx1.SetValue(0, 7); idhMtx1.SetValue(0, 11); idhMtx1.SetValue(1,
                    15);
810
811                 var idhMtx2 = Array.CreateInstance(typeof(double), 16); // Maatriks 2.
812                 idhMtx2.SetValue(Math.Cos((Math.PI / 2) - rJ2), 0); idhMtx2.SetValue(-
                    (Math.Sin((Math.PI / 2) - rJ2)) * Math.Cos (ial2), 4);
                    idhMtx2.SetValue(Math.Sin((Math.PI / 2) - rJ2) * Math.Sin(ial2), 8);
                    idhMtx2.SetValue(ia2 * Math.Cos
                    ((Math.PI / 2) - rJ2), 12);
813                 idhMtx2.SetValue(Math.Sin((Math.PI / 2) - rJ2), 1);
                    idhMtx2.SetValue(Math.Cos((Math.PI / 2) - rJ2) * Math.Cos (ial2), 5);
                    idhMtx2.SetValue(-(Math.Cos((Math.PI / 2) rJ2)) * Math.Sin(ial2), 9);
                    idhMtx2.SetValue(ia2 * Math.Sin
```

((Math.PI / 2) - rJ2), 13);

```
814             idhMtx2.SetValue(0, 2); idhMtx2.SetValue(Math.Sin(ial2), 6);
                idhMtx2.SetValue(Math.Cos(ial2), 10); idhMtx2.SetValue(izs2, 14);
815             idhMtx2.SetValue(0, 3); idhMtx2.SetValue(0, 7); idhMtx2.SetValue(0, 11);
                idhMtx2.SetValue(1, 15);
816
817             var idhMtx3 = Array.CreateInstance(typeof(double), 16); // Maatriks 3.
818             idhMtx3.SetValue(Math.Cos(rJ2 + rJ3), 0); idhMtx3.SetValue(-
                   (Math.Sin(rJ2 + rJ3)) * Math.Cos(ial3), 4); idhMtx3.SetValue
                   (Math.Sin(rJ2 + rJ3) * Math.Sin(ial3), 8); idhMtx3.SetValue
                   (ia3 * Math.Cos(rJ2 + rJ3), 12);
819             idhMtx3.SetValue(Math.Sin(rJ2 + rJ3), 1); idhMtx3.SetValue
                   (Math.Cos(rJ2 + rJ3) * Math.Cos(ial3), 5); idhMtx3.SetValue
                   (-(Math.Cos(rJ2 + rJ3)) * Math.Sin(ial3), 9);
                   idhMtx3.SetValue(ia3 * Math.Sin(rJ2 + rJ3), 13);
820             idhMtx3.SetValue(0, 2); idhMtx3.SetValue(Math.Sin(ial3), 6);
                idhMtx3.SetValue(Math.Cos(ial3), 10); idhMtx3.SetValue(izs3, 14);
821             idhMtx3.SetValue(0, 3); idhMtx3.SetValue(0, 7); idhMtx3.SetValue(0, 11);
                idhMtx3.SetValue(1, 15);
822
823             var zMtx = Array.CreateInstance(typeof(double), 16); // Pööran Z3 telje 180 graadi võrra.
824             zMtx.SetValue(Math.Cos(0), 0); zMtx.SetValue(-(Math.Sin(0)) *
                   Math.Cos(Math.PI), 4); zMtx.SetValue(Math.Sin(0) * Math.Sin
                   (Math.PI), 8); zMtx.SetValue(0, 12);
825             zMtx.SetValue(Math.Sin(0), 1); zMtx.SetValue(Math.Cos(0) *
                   Math.Cos(Math.PI), 5); zMtx.SetValue(-(Math.Cos(0)) *
                   Math.Sin(Math.PI), 9); zMtx.SetValue(0, 13);
826             zMtx.SetValue(0, 2); zMtx.SetValue(Math.Sin(Math.PI), 6);
                   zMtx.SetValue(Math.Cos(Math.PI), 10); zMtx.SetValue(0, 14);
827             zMtx.SetValue(0, 3); zMtx.SetValue(0, 7); zMtx.SetValue(0,
                   11); zMtx.SetValue(1, 15);
828
829             var iMatrix1 = Array.CreateInstance(typeof(double), 16);
830             var iMatrix2 = Array.CreateInstance(typeof(double), 16);
831             var iMatrix3 = Array.CreateInstance(typeof(double), 16); 832
833             mulMtxByMtx(ref idhMtx1, ref idhMtx2, ref iMatrix1);
834             mulMtxByMtx(ref iMatrix1, ref idhMtx3, ref iMatrix2);
835             mulMtxByMtx(ref iMatrix2, ref zMtx, ref iMatrix3);
836
837             // Theta 5 otsimine.
838             double z3n = ((double)iMatrix3.GetValue(8) * (double) iMtxJ6.GetValue(8)) +
                ((double)iMatrix3.GetValue(9) *
                   (double)iMtxJ6.GetValue(9)) + ((double)iMatrix3.GetValue(10)
                     * (double)iMtxJ6.GetValue(10));
839             double modZ3 = Math.Sqrt(((double)iMatrix3.GetValue(8) *
                (double)iMatrix3.GetValue(8)) + ((double)iMatrix3.GetValue
                   (9)  *  (double)iMatrix3.GetValue(9))  +  ((double)  iMatrix3.GetValue(10)  *
```

```
(double)iMatrix3.GetValue(10))); 840          double modN6 = Math.Sqrt(((double)iMtxJ6.GetValue(8) *
                    (double)iMtxJ6.GetValue(8)) + ((double)iMtxJ6.GetValue(9) *
                    (double)iMtxJ6.GetValue(9)) + ((double)iMtxJ6.GetValue(10) *
                      (double)iMtxJ6.GetValue(10)));
841                 double rJ5 = Math.Acos(z3n / (modZ3 * modN6));
842                 double r2J5 = -rJ5;
843                 double dJ5 = 180 / Math.PI * rJ5;
844                 double d2J5 = -dJ5;
845                 this.Controls["iTh15"].Text = String.Format("{0:N3}", Convert.ToDouble(dJ5));
846                 this.Controls["iTh25"].Text = String.Format("{0:N3}", Convert.ToDouble(d2J5));
847
848                 // Theta 4 otsimine.
849                 double Cx = (((double)iMtxJ6.GetValue(9) * (double) iMatrix2.GetValue(10)) -
                    ((double)iMatrix2.GetValue(9) * (double)iMtxJ6.GetValue(10))); // Theta 4 otsimine.
850                 double Cy = -(((double)iMtxJ6.GetValue(8) * (double) iMatrix2.GetValue(10)) -
                    ((double)iMatrix2.GetValue(8) *
                    (double)iMtxJ6.GetValue(10)));
851                 double Cz = (((double)iMtxJ6.GetValue(8) * (double) iMatrix2.GetValue(9)) -
                    ((double)iMatrix2.GetValue(8) *
                    (double)iMtxJ6.GetValue(9)));
852                 double Cmod = Math.Sqrt((Cx * Cx) + (Cy * Cy) + (Cz * Cz));
853                 double Y3mod = Math.Sqrt(((double)iMatrix2.GetValue(4) *
                    (double)iMatrix2.GetValue(4)) + ((double)iMatrix2.GetValue
                    (5) * (double)iMatrix2.GetValue(5)) + ((double)
                    iMatrix2.GetValue(6) * (double)iMatrix2.GetValue(6)));
854                 double CxY3 = ((Cx * (double)iMatrix2.GetValue(4)) + (Cy *
                    (double)iMatrix2.GetValue(5)) + (Cz * (double) iMatrix2.GetValue(6)));
855                 double rJ4 = Math.Acos(CxY3 / (Y3mod * Cmod));
856                 double r2J4 = -rJ4;
857                 double dJ4 = 180 - (180 / Math.PI * rJ4);
858                 double d2J4 = -dJ4;
859                 this.Controls["iTh14"].Text = String.Format("{0:N3}", Convert.ToDouble(dJ4));
860                 this.Controls["iTh24"].Text = String.Format("{0:N3}", Convert.ToDouble(d2J4));
861
862                 // Theta 6 otsimine.
863                 var idhMtx41 = Array.CreateInstance(typeof(double), 16);
864                 idhMtx41.SetValue(Math.Cos(rJ4), 0); idhMtx41.SetValue(-
                    (Math.Sin(rJ4)) * Math.Cos(ial4), 4); idhMtx41.SetValue
                    (Math.Sin(rJ4) * Math.Sin(ial4), 8); idhMtx41.SetValue(ial4 * Math.Cos(rJ4), 12);
865                 idhMtx41.SetValue(Math.Sin(rJ4), 1); idhMtx41.SetValue
                    (Math.Cos(rJ4) * Math.Cos(ial4), 5); idhMtx41.SetValue(-
                    (Math.Cos(rJ4)) * Math.Sin(ial4), 9); idhMtx41.SetValue(ial4
                      * Math.Sin(rJ4), 13);
866                 idhMtx41.SetValue(0, 2); idhMtx41.SetValue(Math.Sin(ial4), 6);
                    idhMtx41.SetValue(Math.Cos(ial4), 10); idhMtx41.SetValue (izs4, 14);
867                 idhMtx41.SetValue(0, 3); idhMtx41.SetValue(0, 7); idhMtx41.SetValue(0, 11);
                    idhMtx41.SetValue(1, 15);
```

```
868
869             var idhMtx42 = Array.CreateInstance(typeof(double), 16);
870             idhMtx42.SetValue(Math.Cos(r2J4), 0); idhMtx42.SetValue(-
                   (Math.Sin(r2J4)) * Math.Cos(ial4), 4); idhMtx42.SetValue
                   (Math.Sin(r2J4) * Math.Sin(ial4), 8); idhMtx42.SetValue(ial4
                     * Math.Cos(r2J4), 12);
871             idhMtx42.SetValue(Math.Sin(r2J4), 1); idhMtx42.SetValue
                   (Math.Cos(r2J4) * Math.Cos(ial4), 5); idhMtx42.SetValue(-
                   (Math.Cos(r2J4)) * Math.Sin(ial4), 9); idhMtx42.SetValue
                   (ial4 * Math.Sin(r2J4), 13);
872             idhMtx42.SetValue(0, 2); idhMtx42.SetValue(Math.Sin(ial4), 6);
                idhMtx42.SetValue(Math.Cos(ial4), 10); idhMtx42.SetValue (izs4, 14);
873             idhMtx42.SetValue(0, 3); idhMtx42.SetValue(0, 7); idhMtx42.SetValue(0, 11);
                idhMtx42.SetValue(1, 15);
874
875             var idhMtx51 = Array.CreateInstance(typeof(double), 16);
876             idhMtx51.SetValue(Math.Cos(rJ5), 0); idhMtx51.SetValue(-
                   (Math.Sin(rJ5)) * Math.Cos(ial3), 4); idhMtx51.SetValue
                   (Math.Sin(rJ5) * Math.Sin(ial3), 8); idhMtx51.SetValue(ial5 * Math.Cos(rJ5), 12);
877             idhMtx51.SetValue(Math.Sin(rJ5), 1); idhMtx51.SetValue
                   (Math.Cos(rJ5) * Math.Cos(ial3), 5); idhMtx51.SetValue(-
                   (Math.Cos(rJ5)) * Math.Sin(ial3), 9); idhMtx51.SetValue(ial5
                     * Math.Sin(rJ5), 13);
878             idhMtx51.SetValue(0, 2); idhMtx51.SetValue(Math.Sin(ial5), 6);
                idhMtx51.SetValue(Math.Cos(ial5), 10); idhMtx51.SetValue (izs5, 14);
879             idhMtx51.SetValue(0, 3); idhMtx51.SetValue(0, 7); idhMtx51.SetValue(0, 11);
                idhMtx51.SetValue(1, 15);
880
881             var idhMtx52 = Array.CreateInstance(typeof(double), 16);
882             idhMtx52.SetValue(Math.Cos(r2J5), 0); idhMtx52.SetValue(-
                   (Math.Sin(r2J5)) * Math.Cos(ial4), 4); idhMtx52.SetValue
                   (Math.Sin(r2J5) * Math.Sin(ial5), 8); idhMtx52.SetValue(ial5
                     * Math.Cos(r2J5), 12);
883             idhMtx52.SetValue(Math.Sin(r2J5), 1); idhMtx52.SetValue
                   (Math.Cos(r2J5) * Math.Cos(ial4), 5); idhMtx52.SetValue(-
                   (Math.Cos(r2J5)) * Math.Sin(ial5), 9); idhMtx52.SetValue
                   (ial5 * Math.Sin(r2J5), 13);
884             idhMtx52.SetValue(0, 2); idhMtx52.SetValue(Math.Sin(ial5), 6);
                idhMtx52.SetValue(Math.Cos(ial5), 10); idhMtx52.SetValue (izs5, 14);
885             idhMtx52.SetValue(0, 3); idhMtx52.SetValue(0, 7); idhMtx52.SetValue(0, 11);
                idhMtx52.SetValue(1, 15);
886
887             var iMatrix341 = Array.CreateInstance(typeof(double), 16);
888             var iMatrix342 = Array.CreateInstance(typeof(double), 16);
889             var iMatrix3411 = Array.CreateInstance(typeof(double), 16);
890             var iMatrix3412 = Array.CreateInstance(typeof(double), 16);
```

```
891             var iMatrix3413 = Array.CreateInstance(typeof(double), 16);
892             var iMatrix3414 = Array.CreateInstance(typeof(double), 16);
893
894             var iMatrix3421 = Array.CreateInstance(typeof(double), 16);
895             var iMatrix3422 = Array.CreateInstance(typeof(double), 16);
896             var iMatrix3423 = Array.CreateInstance(typeof(double), 16);
897             var iMatrix3424 = Array.CreateInstance(typeof(double), 16); 898
899             mulMtxByMtx(ref iMatrix2, ref idhMtx41, ref iMatrix341); //J3* J4.1 AND J3* J4.2
900             mulMtxByMtx(ref iMatrix2, ref idhMtx42, ref iMatrix342); 901
902             mulMtxByMtx(ref iMatrix341, ref idhMtx51, ref iMatrix3411); // Neli vastust.
903             mulMtxByMtx(ref iMatrix341, ref idhMtx52, ref iMatrix3412);
904             mulMtxByMtx(ref iMatrix342, ref idhMtx51, ref iMatrix3421);
905             mulMtxByMtx(ref iMatrix342, ref idhMtx52, ref iMatrix3422);
906
907             // X5*X6 iMatrix3411
908             double X5modo1 = Math.Sqrt(((double)iMatrix3411.GetValue(0) *
                   (double)iMatrix3411.GetValue(0)) + ((double)
                   iMatrix3411.GetValue(1) * (double)iMatrix3411.GetValue(1)) +
                     ((double)iMatrix3411.GetValue(2) * (double) iMatrix3411.GetValue(2)));
909             double X6modo1 = Math.Sqrt(((double)iMtxJ6.GetValue(0) *
                   (double)iMtxJ6.GetValue(0)) + ((double)iMtxJ6.GetValue(1) *
                   (double)iMtxJ6.GetValue(1)) + ((double)iMtxJ6.GetValue(2) *
                   (double)iMtxJ6.GetValue(2)));
910             double X5xX6o1 = (((double)iMatrix3411.GetValue(0) * (double) iMtxJ6.GetValue(0)) +
                ((double)iMatrix3411.GetValue(1) * (double)iMtxJ6.GetValue(1)) +
                ((double)iMatrix3411.GetValue
                   (2) * (double)iMtxJ6.GetValue(2)));
911             double rJ6o1 = Math.Acos(X5xX6o1 / (X5modo1 * X6modo1));
912             double dJ6o1 = (180 / Math.PI * rJ6o1);
913             double d_J6o1 = -dJ6o1;
914
915             // X5*X6 iMatrix3412
916             double X5modo2 = Math.Sqrt(((double)iMatrix3412.GetValue(0) *
                   (double)iMatrix3412.GetValue(0)) + ((double)
                   iMatrix3412.GetValue(1) * (double)iMatrix3412.GetValue(1)) +
                     ((double)iMatrix3412.GetValue(2) * (double) iMatrix3412.GetValue(2)));
917             double X6modo2 = Math.Sqrt(((double)iMtxJ6.GetValue(0) *
                   (double)iMtxJ6.GetValue(0)) + ((double)iMtxJ6.GetValue(1) *
                   (double)iMtxJ6.GetValue(1)) + ((double)iMtxJ6.GetValue(2) *
                   (double)iMtxJ6.GetValue(2)));
918             double X5xX6o2 = (((double)iMatrix3412.GetValue(0) * (double) iMtxJ6.GetValue(0)) +
                ((double)iMatrix3412.GetValue(1) * (double)iMtxJ6.GetValue(1)) +
                ((double)iMatrix3412.GetValue
                   (2) * (double)iMtxJ6.GetValue(2)));
919             double rJ6o2 = Math.Acos(X5xX6o2 / (X5modo2 * X6modo2));
920             double dJ6o2 = (180 / Math.PI * rJ6o2);
```

```
921             double d_J6o2 = -dJ6o2;
922
923             // X5*X6 iMatrix3421
924             double X5modo3 = Math.Sqrt(((double)iMatrix3421.GetValue(0) *
                    (double)iMatrix3421.GetValue(0)) + ((double)
                    iMatrix3421.GetValue(1) * (double)iMatrix3421.GetValue(1)) +
                      ((double)iMatrix3421.GetValue(2) * (double) iMatrix3421.GetValue(2)));
925             double X6modo3 = Math.Sqrt(((double)iMtxJ6.GetValue(0) *
                    (double)iMtxJ6.GetValue(0)) + ((double)iMtxJ6.GetValue(1) *
                    (double)iMtxJ6.GetValue(1)) + ((double)iMtxJ6.GetValue(2) *
                    (double)iMtxJ6.GetValue(2)));
926             double X5xX6o3 = (((double)iMatrix3421.GetValue(0) * (double) iMtxJ6.GetValue(0)) +
                ((double)iMatrix3421.GetValue(1) * (double)iMtxJ6.GetValue(1)) +
                ((double)iMatrix3421.GetValue
                    (2) * (double)iMtxJ6.GetValue(2)));
927             double rJ6o3 = Math.Acos(X5xX6o3 / (X5modo3 * X6modo3));
928             double dJ6o3 = (180 / Math.PI * rJ6o3);
929             double d_J6o3 = -dJ6o3;
930
931             // X5*X6 iMatrix3422
932             double X5modo4 = Math.Sqrt(((double)iMatrix3422.GetValue(0) *
                    (double)iMatrix3422.GetValue(0)) + ((double)
                    iMatrix3422.GetValue(1) * (double)iMatrix3422.GetValue(1)) +
                      ((double)iMatrix3422.GetValue(2) * (double) iMatrix3422.GetValue(2)));
933             double X6modo4 = Math.Sqrt(((double)iMtxJ6.GetValue(0) *
                    (double)iMtxJ6.GetValue(0)) + ((double)iMtxJ6.GetValue(1) *
                    (double)iMtxJ6.GetValue(1)) + ((double)iMtxJ6.GetValue(2) *
                    (double)iMtxJ6.GetValue(2)));
934             double X5xX6o4 = (((double)iMatrix3422.GetValue(0) * (double) iMtxJ6.GetValue(0)) +
                ((double)iMatrix3422.GetValue(1) * (double)iMtxJ6.GetValue(1)) +
                ((double)iMatrix3422.GetValue
                    (2) * (double)iMtxJ6.GetValue(2)));
935             double rJ6o4 = Math.Acos(X5xX6o4 / (X5modo4 * X6modo4));
936             double dJ6o4 = (180 / Math.PI * rJ6o4);
```

```
937             double d_J6o4 = -dJ6o4;
938
939             this.Controls["iTh16"].Text = String.Format("{0:N3}", Convert.ToDouble(dJ6o1)); // Joint
                6 saadud nurgad.
940             this.Controls["iTh26"].Text = String.Format("{0:N3}", Convert.ToDouble(dJ6o2));
941             this.Controls["iTh36"].Text = String.Format("{0:N3}", Convert.ToDouble(dJ6o3));
942             this.Controls["iTh46"].Text = String.Format("{0:N3}", Convert.ToDouble(dJ6o4));
943
944             this.Controls["i2Th166"].Text = String.Format("{0:N3}", Convert.ToDouble(d_J6o1)); //
                Joint 6 saadud nurkade negatiivsed väärtused.
945             this.Controls["i2Th266"].Text = String.Format("{0:N3}", Convert.ToDouble(d_J6o2));
946             this.Controls["i2Th366"].Text = String.Format("{0:N3}", Convert.ToDouble(d_J6o3));
947             this.Controls["i2Th466"].Text = String.Format("{0:N3}", Convert.ToDouble(d_J6o4));
948             }
949             }
950             }
```