Lappeenrannan-Lahden teknillinen yliopisto LUT

School of Energy Systems

Energiatekniikan koulutusohjelma

BH10A0202 Energiatekniikan kandidaatintyö

Kuulakekoreaktorin polttoaine-elementtien virtausdatan analysointi

Analysis of flow data of fuel elements in a pebble bed reactor

Työn tarkastaja: Apulaisprofessori Heikki Suikkanen

Työn ohjaaja: Apulaisprofessori Heikki Suikkanen

Lappeenranta 16.3.2021

Henri Rapeli

TIIVISTELMÄ

Opiskelijan nimi: Henri Rapeli

School of Energy Systems

Energiatekniikan koulutusohjelma

Opinnäytetyön ohjaaja: Apulaisprofessori Heikki Suikkanen

Kandidaatintyö 2021

35 sivua, 11 kuvaa, 1 taulukko, 4 liitettä

Hakusanat: kuulakekoreaktori, polttoaine-elementti, polttoaine-elementin virtaus

Kuulakekoreaktori on grafiittihidasteinen ja heliumjäähdytteinen ydinreaktorityyppi, jossa ydinpolttoaine on sijoitettu grafiittipäällysteisiin kuuliin pieninä rakeina. Kuulakekoreaktorin paineastia on sylinteri, jonka alaosa on kartiomainen. Polttoainekuulat virtaavat reaktorissa hitaasti alaosaa kohti, josta kuulat poistuvat reaktorista. Kuulien virtauksen ymmärtäminen eri osissa reaktoria on tärkeää muun muassa reaktoriturvallisuuden ja reaktorin tehojakauman ymmärtämisen takia.

Kandidaatintyön alussa olevassa kirjallisuusosiossa perehdytään kuulakekoreaktorin toimintaan, turvallisuusominaisuuksiin sekä kuulakekoreaktoreiden ja polttoainekuulien rakenteeseen. Tämän lisäksi tutustutaan polttoainekuulien virtauksesta tehtyihin aikaisempiin tutkimuksiin. Tässä työssä tehdyn analysoinnin tuloksia vertaillaan aikaisempien tutkimusten tuloksiin.

Kandidaatintyön päätavoitteena on kehittää Python-ohjelmointikieltä käyttäen menetelmä kuulakekoreaktorin polttoainekuulien virtausdatan analysointiin. Tutkittava virtausdata on saatu aiemmista simulaatioista, ja data sisältää muun muassa kuulien paikkakoordinaatit eri ajanhetkillä. Analysoinnin tarkoitus on muodostaa selkeä kuva kuulien virtauksesta reaktorissa, suurin mielenkiinnon kohde on muodostaa kuulien aksiaalisen nopeuden jakauma eri osissa reaktoria.

Virtausdatan analysoinnin tulokset vastaavat hyvin aikaisempien tutkimusten tuloksia. Kuulien aksiaalinen nopeus on suurimmillaan reaktorin keskilinjan läheisyydessä ja aksiaalinen nopeus on radiaalisia nopeuksia suurempi. Kuulien radiaalinen liike reaktorissa on verrattain pientä.

SISÄLLYSLUETTELO

Tiivistelmä

Sisällysluettelo

Symboli- ja lyhenneluettelo **4**

1 Johdanto **6**

2 Kuulakekoreaktori **7**
2.1 Reaktorin ja polttoainekuulien rakenne ... 7
2.2 Reaktoripiiri ... 13
2.3 Turvallisuus ... 15

3 Aikaisemmat tutkimukset **17**

4 Virtausdatan analysointi **20**
4.1 Analysoinnin lähtökohdat ... 20
4.2 Tarkasteltavan reaktorin geometria ja tarkkailualueen rajaus ... 20
4.3 Analysointimenetelmät ... 23
4.4 Python-ohjelman rakenne ja toiminta ... 25

5 Analysoinnin tulokset **26**
5.1 Aksiaalisen nopeuden jakauma ... 26
5.2 Radiaalisten nopeuksien jakaumat ... 27
5.3 Kuulien oleskeluajat ... 29
5.4 Kuulien liike reaktorin taitteessa ... 29
5.5 Kuulien radiaalinen liike ... 31

6 Johtopäätökset **33**

7 Yhteenveto **35**

Lähdeluettelo **36**

Liite 1. Polynomisovitteiden muodostaminen **39**

Liite 2. Työssä kehitetyn Python-ohjelman tiedosto: dataloader.py **42**

Liite 3. Työssä kehitetyn Python-ohjelman tiedosto: plot_funcs.py **51**

Liite 4. Työssä kehitetyn Python-ohjelman tiedosto: main.py **54**

SYMBOLI- JA LYHENNELUETTELO

Roomalaiset aakkoset

d	halkaisija	m
N	lukumäärä	[-]
r	säde, etäisyys	m
t	aika	s
v	nopeus	m/s

Kreikkalaiset aakkoset

θ	kulma	rad
ω	kulmanopeus	rad/s

Alaindeksit

rata	ratanopeus
säde	säteen suuntainen nopeus
tot	kokonais-
z	aksiaalinen nopeus

Lyhenteet

AVR	Arbeitsgemeinschaft versuchsreaktor
DEM	Discrete Element Method
HTR-10	10 MW High Temperature Gas-cooled Test Reactor
IAEA	International Atomic Energy Agency

MIT	Massachusetts Institute of Technology
PBMR	Pebble Bed Modular Reactor
THTR-300	Thorium High Temperature Reactor

1 JOHDANTO

Kuulakekoreaktori on ydinreaktorityyppi, jossa ydinpolttoaine, useimmiten uraani, on sijoitettu tuhansiin, noin tennispallon kokoisiin, grafiittivaippaisiin kuuliin (Kadak 2005, 330). Polttoainekuulat syötetään reaktoriin sen yläosasta, ja ne virtaavat hitaalla nopeudella reaktorin läpi (Li et al. 2015, 195). Polttoainekuulien virtauksen ymmärtäminen on tärkeää esimerkiksi reaktoriturvallisuuden kannalta (Kadak & Bazant 2004, 2).

Tämä kandidaatintyö koostuu kirjallisuusosista ja osiosta, jossa esitellään työssä tehdyn tutkimuksen tulokset. Kirjallisuusosiossa perehdytään kuulakekoreaktorin rakenteeseen, toimintaperiaatteeseen sekä ominaisuuksiin. Tämän lisäksi esitellään polttoainekuulien rakenne. Kirjallisuusosiossa esitellään myös aiempia polttoainekuulien virtaukseen liittyviä tutkimuksia ja niiden tuloksia.

Kandidaatintyön päätavoite on kehittää Python-ohjelmointikieltä käyttäen menetelmä polttoainekuulien jo olemassa olevan virtausdatan analysointiin. Kandidaatintyötä varten on saatu aikaisemmin tehdyistä simulaatioista dataa, joka sisältää muun muassa polttoainekuulien paikkakoordinaatit eri ajanhetkillä. Analysoinnin päämäärä on tutkia kuulien virtausnopeutta eri osissa reaktoria, eli selvittää, liikkuvatko kuulat eri nopeuksilla eri osissa reaktoria. Analysoinnin tuloksia myös verrataan aikaisempiin aiheesta tehtyihin tutkimuksiin, jotka on esitelty työn kirjallisuusosiossa. Tietojen analysoinnissa keskitytään siis pelkästään kuulien liikkeeseen. Kuulien liikettä ei myöskään tarkastella koko reaktorin alueella.

Luvussa 2 esitellään kuulakekoreaktorin ja polttoainekuulien rakenne, jonka lisäksi perehdytään reaktorin toimintaan sekä keskeisiin ominaisuuksiin. Luvussa 3 esitellään aikaisempia tutkimuksia ja niiden keskeisiä tuloksia. Luvussa 4 esitellään analysoinnin lähtökohdat, eli esimerkiksi kerrotaan työtä varten saadusta datasta, analysointitavoista sekä esitellään tarkastelun kohteena olevan reaktorin geometria. Luku 5 sisältää itse analysoinnin tulokset, ja luvussa 6 esitellään tulosten perusteella tehdyt johtopäätökset. Työn päättää luku 7, eli yhteenveto.

2 KUULAKEKOREAKTORI

Tässä luvussa perehdytään kuulakekoreaktorin rakenteeseen, toimintaan, erilaisiin reaktoripiireihin sekä reaktorin keskeisiin turvallisuusominaisuuksiin erilaisten reaktorikonseptien kautta. Tämän lisäksi esitellään myös polttoainekuulan rakenne.

Kuulakekoreaktorilla tarkoitetaan grafiittihidasteista ja heliumjäähdytteistä ydinreaktoria, jossa ydinpolttoaine on säilötty tuhansiin, pieniin grafiittipäällysteisiin kuuliin (Kadak 2005, 330).

2.1 Reaktorin ja polttoainekuulien rakenne

Kuten alla esitellyistä reaktorikonsepteista käy ilmi, suurimmalla osalla reaktoreista on monia yhteisiä piirteitä. Kuulakekoreaktorit yleisesti ottaen koostuvat grafiittisista polttoainekuulista, jotka muodostavat itse kuulakeon ja grafiittisesta tukirakenteesta, joka pitää kuulakeon kasassa. Reaktorin rakenne on sylinterimäinen sen alaosaa lukuun ottamatta, joka on kartiomainen. Polttoainekuulat syötetään reaktoriin yläkautta, ja kuulat virtaavat hitaasti reaktoria alaspäin. Alaosan kartiomainen osa ohjaa kuulat alaosan keskellä olevaan poistoputkeen, jota kautta kuulat poistuvat reaktorista. Reaktori on jäähdytetty heliumkaasulla, joka virtaa kuulakeon läpi. Reaktorin grafiittista tukirakennetta ympäröi paineastia, joka voi olla esimerkiksi betonista tai teräksestä valmistettu. (Schulten 1987, 34–35.)

Ensimmäisten kuulakekoreaktoreiden kehitys alkoi Saksassa 1950-luvulla. Tämän kehitystyön tuloksena syntyi kaksi eri reaktoria, joista ensimmäisenä vuonna 1967 valmistunut AVR (Arbeitsgemeinschaft versuchsreaktor) oli käytössä vuoteen 1988 asti (Moormann 2009, 1). AVR:n lämpöteho oli 46 MW, ja kuulakeon muodosti 100 000 polttoainekuulaa. Kuulakeon halkaisija sekä keskimääräinen korkeus oli 3 metriä. Kuulakekoa ympäröi joka suunnasta grafiittiheijastin. Kuulakeon sisällä oli neljä grafiittista rakennetta, joiden sisään pystyttiin tarvittaessa alakautta asettamaan boorikarbidia sisältäviä sammutussauvoja, joilla reaktori voitiin tarvittaessa sammuttaa. Reaktorin tehoa voitiin säätää heliumin virtausta muuttamalla. (Hecker et al. 1967; Ziermann 1987, 79.)

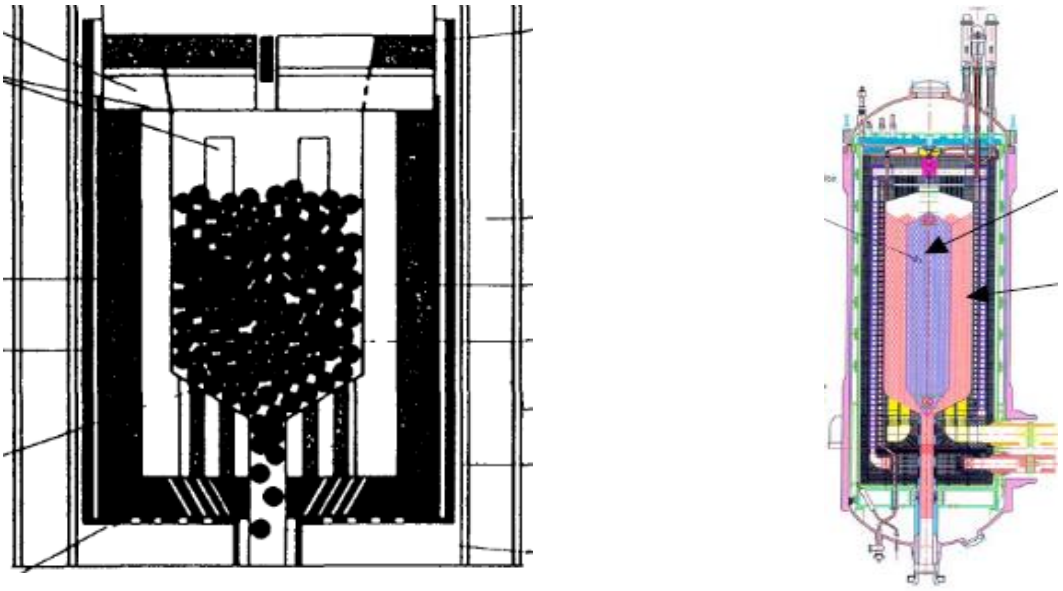
AVR:stä alkaneesta kehitystyöstä syntyi myös toinen saksalainen kuulakekoreaktori, THTR-300 (Thorium High Temperature Reactor). THTR:n elinikä oli verrattain lyhyt, sillä se oli toiminnassa vain vuodet 1985–1988 (IAEA [International Atomic Energy Agency] 2001, 209). THTR:n sähköteho oli 300 MW, ja lämpöteho 750 MW. Kuulakeko, jonka muodosti 675 000 polttoainekuulaa, halkaisija oli 5,1 metriä ja korkeus noin 6 metriä. Kuulakekoa ympäröi grafiittinen heijastin, jonka sivuosaan oli porattu kanavia, joihin voitiin asettaa säätösauvoja reaktorin tehon säätöä varten, ja reaktorin sammuttamista varten itse kuulakekoon voitiin yläpuolelta pudottaa erilliset säätösauvat. (Hecker et al. 1967; Oehme & Schöning 1970, 162, 167.)

Hieman uudempi kuulakekoreaktori on puolestaan kiinalainen HTR-10-testireaktori (10 MW High Temperature Gas-Cooled Test Reactor), joka on edelleen toiminnassa. Reaktorin lämpöteho on 10 MW, ja polttoaine on sijoitettuna 27 000 polttoainekuulaan, joiden muodostaman keon korkeus on hieman alle kaksi metriä ja keon halkaisija on 1,8 metriä. Reaktoria ympäröi tuttuun tapaan alhaalta, ylhäältä ja sivuilta grafiittiheijastin. Myös HTR-10:ssä reaktorin säätö onnistuu sivuheijastimen sisälle asetettujen säätösauvojen avulla. Tämän lisäksi sivuheijastimiin porattuihin kanaviin voidaan syöttää myös grafiittisia kuulia, jotka säätösauvojen tavoin sisältävät boorikarbidia. Näiden absorbaattorikuulien avulla reaktorin sammutus voidaan varmistaa. (Wu et al. 2002, 27, 31.)

Kiinassa on rakenteilla myös suurempikokoinen kuulakekoreaktori HTR-PM (High-Temperature Gas-Cooled Reactor Pebble-Bed Module), jonka kehitystyössä on hyödynnetty HTR-10:n käytöstä saatuja oppeja. HTR-PM-laitos koostuu kahdesta reaktorimoduulista, ja yhden moduulin lämpötehon ollessa 250 MW, koko laitoksen lämpöteho on 500 MW. Molemmat moduulit sisältävät oman kuulakekoreaktorin, joka koostuu 420 000 polttoainekuulasta. Reaktorin halkaisija on 3 metriä ja korkeus 11 metriä. Reaktorin säätö ja sammutus toimii samalla tavalla kuin HTR-10:ssä, eli sivuheijastimeen syötettyjen säätösauvojen ja absorbaattorikuulien avulla. (Zhang et al. 2016, 114; IAEA 2020, 138.)

Kuvassa 2.1 on esitetty kuulakekoreaktorin perusrakenne sekä hieman uudempi reaktorikonsepti.

(a) (b)



Kuva 2.1: Reaktorin rakenne: (a) Kuulakekoreaktorin yleinen rakenne. (Schulten 1987, 34.) (b) Uudempi reaktorikonsepti, jonka keskellä on grafiittikuulista koostuva keskiheijastin. (Kadak & Bazant 2004, 3.)

Kuvasta 2.1(a) näkyy reaktorin sylinterimäinen rakenne, joka muuttuu alaosassa kartiomaiseksi. Polttoainekuulat poistuvat reaktorista poistoputken kautta, johon kartiomainen osa kuulat ohjaa. Reaktoria ympäröi grafiittinen heijastin sekä säteilyltä suojaavat rakenteet. Kuvasta käy ilmi myös AVR:ssä olleet, kuulakeon läpi pystysuunnassa kulkevat rakenteet, joiden sisään asennetuilla pysäytyssauvoilla reaktori voitiin sammuttaa.

Kuvassa 2.1(b) on esitelty MIT:ssä (Massachusetts Institute of Technology) kehitetty reaktorikonsepti. Reaktori on 10 m korkea, ja sen halkaisija on 3,5 metriä. Reaktorin lämpöteho on 250 MW, ja polttoainekuulia reaktorissa on 360 000 kappaletta. (Kadak 2005, 336.)

MIT:n konseptissa kuulakeko muodostuu siis ikään kuin kahdesta osasta. Keskiosa koostuu pelkästään grafiittia sisältävistä kuulista, ja polttoainekuulat ovat täten rengasmaisessa muodostelmassa keskiosan ympärillä. Keskellä olevien grafiittikuulien

tehtävä on sama kuin reaktoria ympäröivällä grafiittirakenteella, eli hidastaa ja heijastaa neutroneita polttoainekuulia kohti. Tämä järjestely mahdollistaa suuremman lämpötehon siten, että reaktorin ulkoreunalla sivuheijastimessa olevat säätösauvat pystyvät edelleen sammuttamaan reaktorin tehokkaasti (Kadak 2005, 336). Niin sanottu ”dynaaminen pylväs” koostuu kuvassa 2.1(b) siis violeteista kuulista, jotka virtaavat polttoainekuulien kanssa reaktoria alaspäin.

MIT:n konseptissa grafiittikuulista koostuva keskiheijastin voi olla myös kiinteä rakenne, kuten Etelä-Afrikkalaisessa PBMR:ssä (Pebble Bed Modular Reactor) (Reitsma 2004). PBMR:n lämpöteho on 400 MW, ja reaktori koostuu keskiosan kiinteästä keskiheijastimesta, jonka ulkohalkaisija on 2 metriä sekä rengasmaisesta ulko-osasta, jonka ulkohalkaisija on 3,7 metriä. Kiinteää keskiheijastinta voidaan myös hyödyntää reaktorin hallitsemisessa tai sammuttamisessa. Tämä tapahtuu siten, että keskiheijastimen sisältämiin kanaviin syötetään myös edellä mainittuja absorbaattorikuulia.

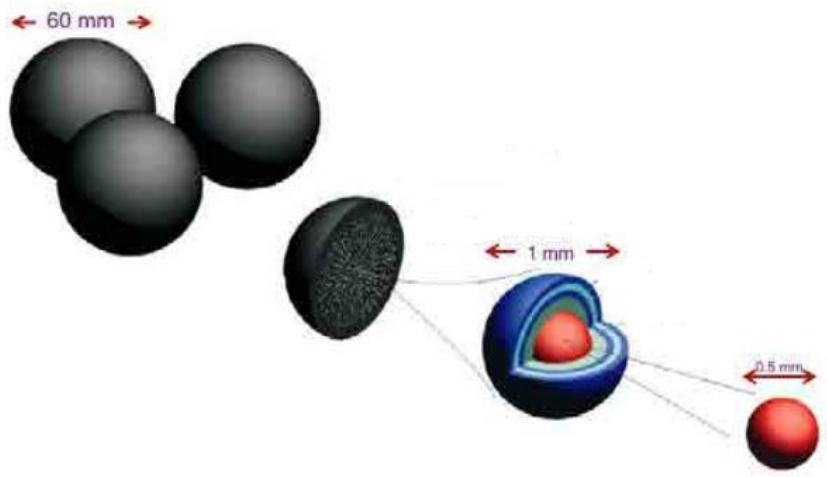
PBMR:n korkeus on 11 metriä, ja sen lämpöteho on 400 MW. Polttoainekuulia reaktorissa on reilut 451 000 kappaletta. (IAEA 2013, 209.)

PBMR:n tulevaisuus tosin on jokseenkin epäselvä. Hanke oli jo kertaalleen kaatunut, mutta voi olla, että se on tarkoitus käynnistää uudelleen lähitulevaisuudessa. (Nuclear Engineering International 2020.)

Grafiittikuulista koostuvan dynaamisen keskiheijastimen ja kiinteän grafiittirakenteen välillä on tiettyjä eroja. HTR-PM:ää suunnitellessa havaittiin muun muassa, että dynaaminen keskiheijastin johtaa radiaalisessa suunnassa epätasaiseen heliumin lämpötilajakaumaan. Tämä johtuu siitä, että reaktorin keskiosa on kylmempi kuin ulko-osa, jossa lämpöä tuottavat polttoainekuulat sijaitsevat, ja osa heliumista virtaa tämän keskiosan läpi. Lisäksi sivuheijastimeen sijoitettavat säätösauvat eivät toimineet tällä rakenteella yhtä tehokkaasti. Sen sijaan kiinteän keskiheijastimen rakenteellinen kestävyys aiheuttaa huolta, ja se pitäisi vaihtaa reaktorin käyttöiän aikana. Lisäksi se tekee polttoainekuulien virtauksesta monimutkaisempaa, ja monimutkaistaa kuulien poistoa reaktorista. (Zhang et al. 2016, 114.)

Kuulakekoreaktoreissa käytettyjen polttoainekuulien määrä vaihtelee reaktorin koon ja mallin mukaan, mutta kuulien rakenne on kuitenkin verrattain samanlainen. Polttoaine on sijoitettu kuuliin hyvin pienikokoisina rakeina, jotka ovat päällystetty esimerkiksi hiilellä (grafiitilla) ja piikarbidilla. Kuulat on vuorattu uudelleen grafiitilla, joka toimii samalla hidasteena. (Kadak 2005, 334; Rintala 2011, 32.)

Kuvassa 2.2 on esitetty polttoainekuulan rakenne.



Kuva 2.2: Polttoainekuulan rakenne. (IAEA 2013, 211.)

Kuvasta nähdään yksittäistä polttoaineraetta ympäröivät eri kerrokset. Ensimmäinen, kuvassa vaaleanvihreäksi väritetty, kerros on huokoinen hiilikerros, jonka päällä on tummansininen kerros, joka on pyrolyyttistä hiiltä. Vaaleansinisellä merkitty kerros on piikarbidia, jonka jälkeen on vielä toinen kerros pyrolyyttistä hiiltä. Näiden kerrosten paksuus on suuruusluokaltaan kymmeniä mikrometrejä. Päällystetyt polttoainerakeet ovat sijoitettu grafiittiseen rakenteeseen, jonka halkaisija on 5 cm. Tämän grafiittisen rakenteen päällä on vielä 0,5 cm paksu grafiittikerros, joka ei sisällä ollenkaan polttoainepartikkeleita. (Wols 2015, 3, 13.)

Polttoainepartikkelien monikerroksisen rakenteen tarkoitus on se, että kaikki radioaktiiviset fissiotuotteet pysyvät polttoainepartikkelin ja täten myös polttoainekuulan sisällä, joten normaaleissa toimintaolosuhteissa radioaktiivisia aineita ei pääse vuotamaan reaktoriin tai reaktorirakennukseen (Wu et al. 2002, 25).

Kuvassa 2.2 esitetty polttoainekuulan rakenne kuvaa modernia polttoainekuulaa. AVR:ssä käytettiin myös polttoainekuulia, joista puuttui pyrolyyttisten hiilikerrosten välissä oleva piikarbidikerros (Moormann 2008, 3). Polttoaine koostui uraanista ja toriumista, jota oli niin karbidina tai vaihtoehtoisesti oksidimuodossa. Myös polttoaineen tarkempi koostumus, ja täten myös väkevyys (uraani-235:n osuus), vaihteli; kuulat sisälsivät uraani-235:ttä gramman ja vaihtelevia määriä toriumia ja uraani-238:aa. (Ziermann 1987, 74–75.)

Jo AVR:n aikana alettiin tehdä testejä moderneilla polttoainekuulilla, joissa polttoaine on oksidimuodossa, polttoaine ei sisällä toriumia, pyrolyyttisten hiilikerrosten väliin on lisätty piikarbidikerros, ja polttoaineen väkevyys on matala (Krüger & Ivens 1985, 64–65).

THTR-300:ssa käytetyt polttoainekuulat sisälsivät korkeasti väkevöityä uraania (uraani-235:n osuus 93 %) noin gramman ja toriumia noin kymmenen grammaa (IAEA 2001, 209). Polttoainekuulat olivat siinä mielessä modernin kuulan tapaisia, että polttoaine oli oksidimuodossa, mutta piikarbidikerros vielä puuttui (Krüger & Ivens 1985, 64).

Uudemmissa reaktorimalleissa polttoaine on uraanidioksidia, ja ainoa ero kuulien välillä on polttoaineen väkevyydessä sekä polttoaineen määrässä per kuula. HTR-10:ssä jokainen kuula sisältää 5 grammaa uraania, jonka väkevöimisprosentti on 17 (Wu et al. 2002, 27). HTR-PM:ssä yksi kuula sisältää 7 grammaa uraania, jonka väkevyys on 8,5 %. (IAEA 2020, 138). MIT:n reaktorikonseptissa jokainen kuula sisältäisi 7 grammaa uraanidioksidia 11 000 rakeessa, ja polttoaineen väkevyys olisi 8 % (Kadak 2005, 336). PBMR:ssä sen sijaan polttoainetta on 9 grammaa, ja sen väkevyys on 9,6 % (IAEA 2013, 209).

Kuulakekoreaktoreihin on mahdollista ladata uutta polttoainetta jatkuvasti, eli kun polttoainekuulat poistuvat reaktorista alaosan poistoputken kautta, jo reaktorissa olleita, mutta edelleen käyttökelpoisia polttoainekuulia voidaan kokonaan uusien polttoainekuulien kanssa syöttää reaktoriin uudelleen reaktorin yläosasta. Esimerkiksi HTR-10:ssä kuulat kulkevat reaktorin läpi keskimäärin viisi kertaa. (Wu et al. 2002, 27–28). Kuulien virtausnopeus HTR-10:ssä on suuruusluokaltaan noin 10^{-4} – 10^{-3} m/h, joten

kuulan matka parimetrisen kuulakeon läpi voi kestää useammankin sata päivää (Gui et al. 2014, 295).

2.2 Reaktoripiiri

Kuulakekoreaktoreilla voidaan saavuttaa verrattain korkeita käyttölämpötiloja. Esimerkiksi MIT:n konseptissa heliumin lämpötila sen poistuessa reaktorista olisi 900 °C (Kadak 2005, 338). Tämä korkea lämpötila mahdollistaa sen, että kuulakekoreaktoreita voidaan käyttää energiantuotannossa useilla eri tavoilla.

AVR:ää käytettiin perinteisen lämpölaitoksen tavoin vesihöyryn tuottamiseen tavalliselle höyryturbiiniprosessille. Reaktoripiiri koostuu täten siis primääri- (heliumkierto) ja sekundääripiiristä (höyryturbiiniprosessi). AVR:n rakenne oli integroitu, eli reaktori ja höyrystin sijaitsivat saman paineastian sisällä. Helium (paine 1,08 MPa, massavirta 13 kg/s) virtaa reaktorissa alhaalta ylöspäin. Heliumin lämpötila reaktoriin tullessa on 270 °C, ja poistumislämpötila on 950 °C. Alun perin heliumin poistumislämpötila oli 850 °C, mutta lämpötilaa pystyttiin nostamaan korkeammaksi vuonna 1974. Reaktorin läpi virrattuaan helium kulkee höyrystimelle, joka sijaitsee reaktorin yläpuolella. Höyrystimellä syntyvän höyryn massavirta on 15,56 kg/s, lämpötila 506 °C, ja paine 7,2 MPa. Höyryllä pyöritetään höyryturbiinia, joka on liitetty sähkögeneraattoriin, jonka tuottama sähköteho on 15 MW. (Krüger & Ivens 1985, 62–64.)

THTR-300:n reaktoripiiri oli niin primääri- kuin sekundääripiirin osalta pitkälti samanlainen kuin AVR:ssä kokoa lukuun ottamatta. Helium virtasi reaktorissa ylhäältä alas, jonka jälkeen helium ohjattiin höyrystimelle, jossa tuotettu höyry käytettiin höyryturbiinin ja generaattorin pyörittämiseen. Heliumin paine oli noin 4 MPa, tulolämpötila 250 °C, ja poistumislämpötila 750 °C. Höyrystimessä tuotetun höyryn massavirta oli 42,5 kg/s, paine hieman yli 19 MPa, ja lämpötila 535 °C. Nimen mukaisesti reaktorin sähköteho on 300 MW. (Oehme & Schöning, 166.)

Myös HTR-10:ssä reaktoria käytetään tuottamaan sähköä höyryturbiiniprosessin avulla. HTR-10:n primääripiiri eroaa siten AVR:stä ja THTR-300:sta, että reaktori ja höyrystin ovat erotettu omiin paineastioihinsa. Toisaalta reaktoria voidaan käyttää myös kaukolämmön tuottamiseen (Wu et al. 2002, 26). Heliumin paine reaktorissa on 3 MPa, heliumin tulolämpötila on 250 °C, ja lähtölämpötila on 750 °C. Heliumin massavirta on

4,32 kg/s. Höyrystimeen syötetään vettä 3,47 kg/s, jonka lämpötila höyrystimelle tullessa on 104 °C. Höyry poistuu höyrystimeltä 4 MPa:n paineessa, lämpötilan ollessa 440 °C. Turbiinin ja generaattorin tuottama sähköteho on 3 MW. (IAEA 2013, 3–4.)

HTR-10:lle on suunniteltu myös toisenlainen sekundääripiiri yksinkertaisen höyryturbiiniprosessin lisäksi. Tässä vaihtoehdossa on yhdistetty kaasu- ja höyryturbiiniprosessi, ja heliumin poistumislämpötila reaktorista on nostettu 900 celsiusasteeseen. Ensin helium kulkee lämmönsiirtimen läpi kuumentaen typpeä (typen tulolämpötila 483 °C, poistolämpötila 850 °C, paine 3,2 MPa, massavirta 11,17 kg/s) kaasuturbiiniprosessia varten. Tässä prosessissa helium jäähtyy 600 asteeseen, ja lämmönsiirtimen lämpöteho on kokonaislämpötehosta puolet, eli 5 MW. Tämän jälkeen viilennyt helium kulkee höyrystimelle, jossa käytetään kokonaislämpötehosta jäljelle jäänyt puolikas veden höyrystämiseen. Tuotetun höyryn paine on 3,43 MPa ja lämpötila 435 °C. Kaasuturbiiniprosessin sähköteho on 2,08 MW, ja höyryturbiinin 1,36 MW. (IAEA 2001, 135–136.)

MIT:n konseptissa on tarkoitus tuottaa sähköä epäsuoralla kaasuturbiiniprosessilla. Tässä tapauksessa epäsuora prosessi tarkoittaa sitä, että prosessi koostuu kahdesta eri kiertopiiristä, joissa molemmissa kiertävänä aineena on helium. Molemmissa piireissä heliumin paine on korkeimmillaan noin 8 MPa, ja primääripiirissä heliumin lämpötila ennen reaktoria on 400–500 °C, ja poistumislämpötila on yllä mainittu 900 °C. Sekundääripiirissä helium poistuu lämmönsiirtimeltä noin 800 asteen lämpötilassa. Sekundääripiirissä on kolme kaasuturbiinivaihetta, joita käytetään kompressorien pyörittämiseen, ja neljäs turbiinivaihe pyörittää generaattoria sähkön tuottamiseen. Prosessin sähköteho on 110–120 MW. MIT:n konseptin on tarkoitus olla mahdollisimman modulaarinen, jolloin eri komponentit voitaisiin valmistaa tehtaalla, tuoda esimerkiksi maanteitä pitkin laitosalueelle, ja koota laitos paikan päällä. (Kadak 2005, 338–339.)

PBMR:ssä sen sijaan sähköä on tarkoitus tuottaa suoralla kaasuturbiiniprosessilla. Tällöin kuuma helium virtaa suoraan reaktorista kaasuturbiinille, ilman toista kiertopiiriä. Heliumin paine on korkeimmillaan 9 MPa. Sisäänmenolämpötila reaktoriin on 500 °C ja poistumislämpötila 900 °C. Prosessin sähköteho on 165 MW. (IAEA 2013, 206.)

2.3 Turvallisuus

Kuulakekoreaktoreilla on turvallisuuden kannalta monia sitä edistäviä ominaisuuksia. Oleellinen turvallisuustekijä on polttoaineen negatiivinen reaktiivisuuden lämpötilakerroin, joka sellaisenaan pystyy katkaisemaan fissioketjureaktion esimerkiksi jäähdytevirtauksen lakatessa (Wu et al. 2002, 31). Tämä ilmiö perustuu Doppler-ilmiöstä johtuvan resonanssiabsorption kasvuun, joka yksinkertaistettuna johtaa siihen, että neutroneiden absorptio polttoaineessa olevaan uraani-238:aan kasvaa lämpötilan kasvaessa, joka taas johtaa reaktiivisuuden, ja täten myös tehon, laskuun (Duderstadt & Hamilton 1976, 557–559).

Toinen tärkeä, enemmän kuulakekoreaktoreille olennainen, turvallisuusominaisuus liittyy kuulakekoreaktorin matalaan tehotiheyteen, eli siihen, että reaktoreiden lämpöteho on pieni verrattuna reaktorin kokoon. Esimerkiksi HTR-10:n tehotiheys on 2 MW/m^3 (Wu et al. 2002, 27) ja MIT:n reaktorin 3,54 MW/m^3 (Kadak 2005, 336). Tyypillisen painevesireaktorin tehotiheys voi sen sijaan olla noin 100 MW/m^3 (Shirvan et al. 2016, 1306). Matala tehotiheys, sekä grafiitin verrattain hyvä lämmönjohtavuus, mahdollistavat sen, että ilman aktiivistakin jäähdytevirtausta reaktorin lämpötila nousee hitaasti, ja toisekseen korkeimmatkaan lämpötilat reaktorissa eivät nouse niin korkeiksi, että polttoainekuulat tai reaktorirakenteet alkaisivat hajota. Luonnolliset, passiiviset lämmönsiirtomekanismit, kuten lämmön johtuminen ja lämpösäteily reaktorin sisällä ja lopulta reaktorista ulos riittävät pitämään sammutetun reaktorin lämpötilan turvallisella tasolla. (Kadak 2005, 343.)

Matalasta tehotiheydestä johtuen kuulakekoreaktorit ovat tehoonsa nähden suurikokoisia, joka johtaa siihen, että polttoainekuulien ja reaktorirakenteiden lämpökapasiteetti on verrattain suuri. Tämä tarkoittaa sitä, että jäähdytyksenkin lakatessa polttoainekuulien ja reaktorirakenteiden lämpötila nousee hitaasti, ja voi kestää päiviä, että lämpötila nousee huippuarvoonsa. (IAEA 2020, 139.)

Polttoainekuula sekä reaktorirakenteet ovat rakennettu grafiitista, joten ne kestävät korkeampia lämpötiloja ilman vaurioitumista kuin mitä reaktoreissa voi poikkeustilanteissa syntyä (Wu et al. 2002, 25). Grafiitti tosin voi reagoida hapen kanssa, jos ilmaa pääsee reaktoriin jostain syystä. Hapettumisreaktio voi olla joko endoterminen

tai eksoterminen prosessiolosuhteista riippuen, eli oikeissa olosuhteissa lisälämpöä voi vapautua tästä hapettumisreaktiosta. Grafiitin nopeampaan ”palamiseen” vaadittavaa usean tuhannen asteen lämpötilaa reaktorissa ei kuitenkaan pääse syntymään. (Kadak 2005, 343.)

Myös se, että kuulakekoreaktoreihin voidaan jatkuvasti, reaktorin ollessa käynnissäkin, syöttää uusia polttoainekuulia ja poistaa vanhempia polttoainekuulia on turvallisuutta lisäävä tekijä. Se mahdollistaa sen, että ydinpolttoaineen kulumista ajan kuluessa ei tarvitse juurikaan kompensoida, esimerkiksi uuden polttoaineen korkeammalla väkevyysasteella. Tällöin niin sanottua ylimääräistä positiivista reaktiivisuutta ei reaktorissa polttoaineen takia juurikaan ole. Myös se, että helium ei vaikuta itse neutronien liikkeeseen, eikä täten fissioreaktioiden tapahtumiseen, sekä heliumin kemiallinen neutraalius ovat turvallisuutta lisääviä tekijöitä. (IAEA 2013, 215.)

Kuulakekoreaktoreilla on myös niille ominaisia turvallisuushaasteita. Esimerkiksi reaktorin lämpötilaa on hankala tietää tarkasti, joten reaktorissa mahdollisesti syntyvät poikkeuksellisen kuumat alueet jäävät helposti huomaamatta. Monikerroksisesta rakenteesta huolimatta tiettyjä fissiotuotteita voi päästä polttoainekuulan ulkopuolelle esimerkiksi diffuusion kautta. Polttoainekuulien hankautuessa toisiaan vasten niistä irtoaa pölyä, johon fissiotuotteet voivat tarttua ja täten kulkeutua mahdollisesti myös reaktorista ulos. Lisäksi niissä reaktoreissa, joissa on vesikiertopiiri (esim. AVR:n ja HTR-10:n sekundääripiiri), on vaara, että vettä pääsee reaktoriin. Tämä voi johtaa huomattavaankin reaktiivisuuden lisäykseen. (Moormann et al. 2018, 1911–1912.)

3 AIKAISEMMAT TUTKIMUKSET

Kuten edellä mainitaan, polttoainekuulien virtauksen ymmärtäminen on tärkeää muun muassa reaktoriturvallisuuden kannalta. Erot polttoainekuulien virtauksessa eri osissa reaktorisydäntä voivat johtaa siihen, että reaktorin lämpöteho on jakautunut epätasaisesti keon eri osiin. Reaktorin tehon jakautumista ei myöskään kyetä sisäisesti mittaamaan, joten tästäkin syystä polttoainekuulien virtauksen ymmärtäminen reaktorisydämessä on tärkeää. (Bazant & Kadak 2004, 2.)

Tässä luvussa esitellään aiheeseen liittyviä aiemmin tehtyjä tutkimuksia. Tutkimusta on tehty niin fyysisinä kokeina kuin laskennallisia menetelmiä hyödyntävinä tietokonesimulaatioinakin. Simulaatiot on pääasiassa tehty DEM-laskentamenetelmää (Discrete Element Method) käyttäen, joka mahdollistaa polttoainekuulien virtauksen mallintamisen kuulakekoreaktorissa. DEM-simulaatiossa jokainen kuula on eritelty omaksi yksittäiseksi elementikseen, ja niiden liikettä ja vuorovaikutusta toisiinsa nähden simuloidaan käyttäen liikettä kuvaavia fysiikan lakeja (Jiang et al. 2012, 279).

MIT:ssä tehtiin kokeita kuulien virtaukseen liittyen (Kadak & Bazant 2004). Mittausten tavoitteena oli tutkia liikkuvatko polttoainekuulat reaktorisydämessä virtaviivaisesti omissa kanavissaan vai satunnaisesti sekä tutkia, miten kuulien aksiaalinen nopeus muuttuu eri etäisyyksillä reaktorin keskilinjasta. Mittaukset tehtiin kahdella pienoismallilla, jotka oli rakennettu mittasuhteella 1:10 todelliseen reaktoriin verrattuna. Toinen pienoismalli oli poikkileikattu pystysuunnassa ja varustettu ikkunalla, josta kuulien virtausta pystyttiin seuraamaan visuaalisesti. Kokeissa havaittiin, että kuulat liikkuvat pystysuunnassa verrattain suoraan alaspäin. Virtauskanavien poikkeama vaakasuunnassa oli suuruusluokaltaan samansuuruinen kuin kuulien halkaisija. Kokeissa havaittiin myös, että kuulien nopeus oli suurin pienoismallin keskilinjan lähellä ja nopeus laski pienoismallien reunoja lähestyttäessä, mutta erot nopeudessa olivat verrattain pieniä.

Choi et al. (2004) tutkivat kuulien liikettä ja sekoittumasta radiaalisessa suunnassa kaksiulotteisella pienoismallilla, jonka läpinäkyvän seinän läpi kuulien liikettä pystyttiin kameralla seuraamaan. Kokeissa havaittiin, että kuulien radiaalinen liike oli kuulien halkaisijan mittaista matkaa suurempaa, ja radiaalinen liike oli samansuuruista eri

virtausnopeuksilla. Lisäksi havaittiin, että aksiaalinen nopeus oli suurinta reaktorin keskilinjan lähellä, ja sen maksimi oli reaktorin alaosassa.

Rycroft et al. (2006) tekivät kattavan simulaation käyttäen DEM:ää. Simulaatio suoritettiin täydessä mittakaavassa. Simuloitavassa reaktorissa oli esimerkiksi kuvassa 2.1(b) esitetty grafiittipylväs keskellä, ja tärkeä tutkittava asia olikin se, kuinka paljon erityyppiset kuulat sekoittuvat keskenään. Simulaatiossa havaittiin, että reaktorin sylinterimäisessä osassa kuulien virtauskanavat olivat lähes pystysuoria, ja että kuulien nopeus on suurempi lähellä reaktorin radiaalista keskilinjaa ja pienempi reunojen lähellä. Nopeuserot tosin eivät ole kovin suuria. Lähes pystysuorat virtauskanavat johtivat myös siihen, että grafiitti- ja polttoainekuulat eivät juurikaan sekoittuneet. Geometriassa tosin oli reaktorisylinterin yläosassa lyhyt ohjausrengas, joka esti grafiitti- ja polttoainekuulien sekoittumisen keon yläosassa, mutta eri kuulat eivät sekoittuneet myöskään ohjausrenkaan jälkeen.

Li et al. (2009) suorittivat myös kokeita kuulien virtaukseen liittyen. Kokeiden tavoitteena oli tutkia, miten käytännössä suoritetut kokeet vastaavat tietokonesimulaatioita. Kokeet suoritettiin poikkileikatun sylinterin muotoisella pienoismallilla, ja kokeissa tutkittiin kuulien pystysuuntaista virtausta. Kokeita tehtäessä havaittiin jälleen, että kuulien vertikaalinen nopeus oli suurempi pienoismallin keskilinjan lähellä ja pienempi reunoilla. Tulosten havaittiin myös vastaavan hyvin tietokonesimulaatioita.

Yang et al. (2009) tekivät pienoismallikokeita selvittääkseen, onko kuvassa 2.1(b) esitetyn kaksiosaisen keon muodostaminen mahdollista. Toisin sanoen, pysyvätkö keskelle ja reunoille asetetut kuulat erossa toisistaan omilla alueillaan, vai sekoittuvatko ne keskenään. Kokeet tehtiin 2-D mallilla, joka oli ikään kuin sylinterimäisen mallin keskilinjalta otettu aksiaalinen poikkileikkaus. Kokeissa havaittiin, että kaksiosainen keko on lähtökohtaisesti mahdollista muodostaa, eli eri kuulista muodostetut alueet pysyivät ehjinä. Pienoismalli oli tarkoitus rakentaa 1:5 mittakaavassa oikeaan reaktoriin verrattuna, ja niin tehtiinkin, paitsi korkeuden osalta, joka oli liian pieni. Korkeuden epäsuhteellisuus ja mallin kaksidimensionaalinen luonne tarkoitti tosin sitä, että osaa kokeiden tuloksista ei voi pitää oikeaa tilannetta kuvaavina.

Jiang et al. (2012) jatkoivat siitä, mihin he edellisessä kokeessa (Yang et al. 2009) jäivät. Nyt tehtyjen kokeiden oli tarkoitus vahvistaa edellisessä kokeessa saadut tulokset parempaa pienoismallia ja tietokonesimulaatioita käyttäen. Uudellakin mallilla tehdyissä kokeissa havaittiin, että kaksiosaisen keon muodostaminen on mahdollista, tämän lisäksi havaittiin sama ilmiö kuin muissakin kokeissa; kuulat liikkuvat nopeammin alaspäin lähellä keon keskilinjaa. Samat tulokset havaittiin myös simulaatioissa.

Yang et al. (2014) tutkivat kuulien liikettä niin kokeellisesti kuin myös tietokonesimulaatioilla. Tutkimuksissa tutkittiin muun muassa kuulien liikettä radiaalisessa suunnassa sekä säteen suuntaisen nopeuden suuruutta. Tutkimuksissa havaittiin, että kuulat virtaavat reaktorissa alaspäin kohtalaisen vertikaalisia virtauskanavia pitkin, mutta reaktorin alaosassa kuulat alkavat virtaamaan enemmän myös keskelle päin. Tämä johtaa myös suurempaan säteen suuntaiseen nopeuteen reaktorin alaosassa. Reaktorin ulkoreunan lähellä kuulien säteen suuntainen liike oli suurempaa kuin reaktorin keskiosassa.

Li et al. (2015) tekivät tutkimuksia liittyen reaktoriastian pinnan rakenteeseen ja sen vaikutuksesta kuulien liikkeeseen. Pelkkä sileä pinta saattaa johtaa seinämän lähellä olevien kuulien liikkeen pysähtymiseen, mikä olisi riski turvallisuuden kannalta. Tutkimuksessa tutkittiin pintoja, joihin oli mallinnettu erilaisia lovia kuulien virtausta ohjaamaan. Simulointeja tehtiin reaktorilla, jonka pinta sisälsi sahalaidan, puolikaaren tai kolmion mallisia rakenteita. Näistä parhaimmaksi vaihtoehdoksi havaittiin kolmion mallinen rakenne. Tällöin hitaammin liikkuvia kuulia siis pakkautui seinämän lähelle vähiten. Tutkimuksessa havaittiin myös, että kuulia pakkautui jonkin verran joka tapauksessa varsinkin reaktorin alaosassa alueelle, missä kartiomainen osa alkaa.

4 VIRTAUSDATAN ANALYSOINTI

Tässä osiossa esitellään työssä tarkasteltavan reaktorin ja siinä virtaavien kuulien olennaisimmat mitat sekä käydään läpi sitä, miten kuulien virtauksen analysointi tehdään.

4.1 Analysoinnin lähtökohdat

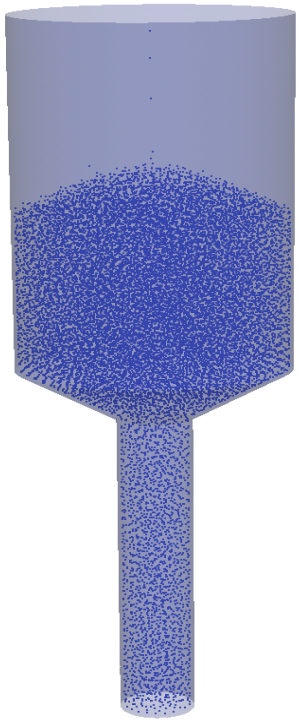
Tässä työssä tutkitaan siis aikaisemmista DEM-simulaatioista saatua virtausdataa polttoainekuulille. Data oli tallennettu kaikkiaan 670 tekstitiedostoon, jokainen tiedosto sisältäen dataa eri ajanhetkiltä. Tiedostot oli muodostettu siten, että niiden välillä aikaa kului 0,03333 sekuntia, eli kokonaisuudessaan tietoa oli noin 22,3 sekunnilta. Tiedostot sisälsivät muun muassa kaikkien kuulien paikkakoordinaatit, x-, y-, ja z-suuntaiset nopeudet ja kuulien säteen. Tarkalleen ottaen, tiedostoissa olevat paikkakoordinaatit koskevat kuulien keskipistettä. Kuulien säde on kolme senttimetriä.

Simuloitava reaktori perustui aiemmin esiteltyyn kiinalaiseen HTR-10-testireaktoriin. Simulaatiot oli tehty siten, että kuulien liikettä oli nopeutettu huomattavasti. Kuten luvussa 2.1 mainittiin, todellisessa reaktorissa kuulien matka reaktorin läpi voi kestää useita satoja päiviä, kun taas simulaatioissa osa kuulista kerkesi kulkemaan reaktorin läpi kokonaisuudessaan simuloidun parin kymmenen sekunnin ajanjakson aikana. Kuulien liikettä oli nopeutettu, jotta kuulien liikettä pystyttiin seuraamaan pidemmän matkan verran ajallisesti lyhyemmänkin simulaation aikana. Tämä kuulien nopeutettu liike johtaa siis siihen, että esimerkiksi nopeuksille lasketut lukuarvot ovat todellisia huomattavasti suurempia, mutta esimerkiksi kuulien virtausnopeuden jakauman on havaittu pysyvän jokseenkin samanlaisena, vaikka simulaatiota onkin nopeutettu (Rycroft et al. 2006, 2). Täten nopeutetuistakin simulaatioista voidaan tehdä johtopäätöksiä kuulien virtauksesta.

Virtausdatan analysoinnin tavoitteena oli selvittää esimerkiksi kuulien erisuuntaisten nopeuksien jakaumat eri osissa reaktoria. Lisäksi tahdottiin selvittää kuinka kauan kuulat viettävät eri osissa reaktoria sekä arvioida kuulien radiaalisen liikkeen suuruutta niiden liikkuessa alaspäin reaktorissa.

4.2 Tarkasteltavan reaktorin geometria ja tarkkailualueen rajaus

Tarkasteltavan reaktorin geometria siis perustuu HTR-10-testireaktoriin. Simuloinneissa käytetyn reaktorin geometria on esitetty kuvassa 4.1.

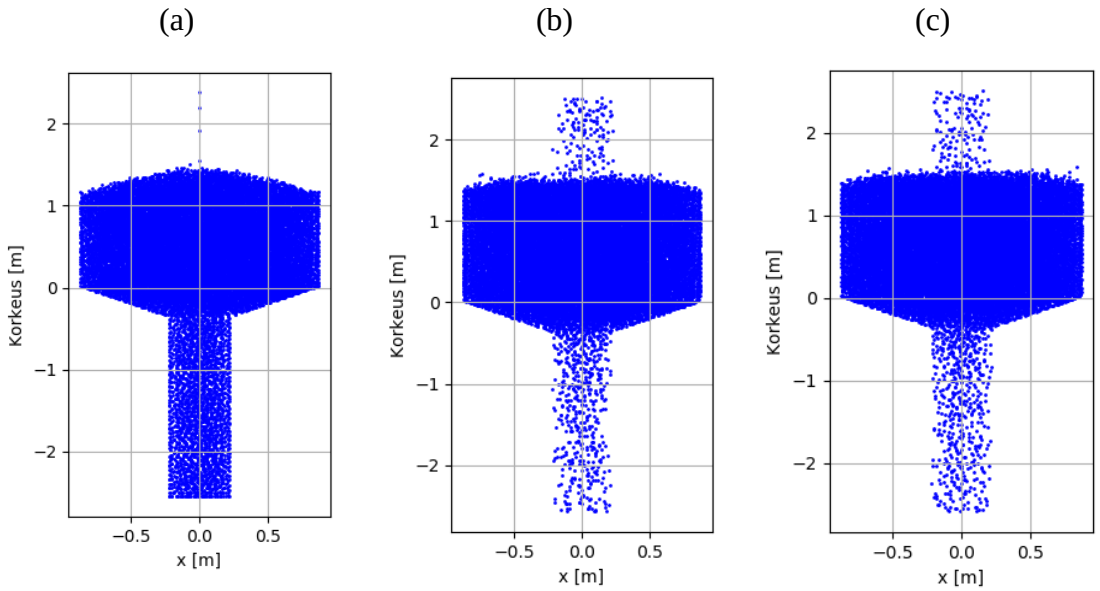


Kuva 4.1: Tarkasteltavan reaktorin geometria.

Simuloinneissa käytetyn geometrian sylinterimäisen osan halkaisija on 1,8 metriä, ja kokonaisuudessaan sen korkeus on 5,1 metriä. Tosin kuten kuvastakin nähdään, kuulakeko ei korkeussuunnassa yletä reaktorin sylinterimäisessä osassa kuin korkeintaan sen puoleenväliin. Siniset pisteet kuvassa ovat siis kuulia. Kuulia oli simuloinneissa (22 137) hieman vähemmän kuin oikeassa HTR-10:ssä (27 000).

Tutkittavaksi alueeksi valittiin vain sylinterimäinen osa reaktoria, jossa reaktoriastian muoto ei vaikuta kuulien liikkeeseen niin paljon kuin esimerkiksi reaktorin alaosan kartiomaisessa osassa, jossa kartio ohjaa kuulia poistoputkea kohti, eikä kuulien liikettä tarkastella myöskään poistoputkessa. Kuten kuvasta on erotettavissa, kuulat syötetään kuulakeon päälle reaktorin yläosasta yksittäin kuulakeon keskiosaan. Myöskään itse kuulakeon yläosaa ei tarkastella, sillä kuulat pomppivat satunnaisesti keon päälle pudotessaan.

Kuvasta 4.1 on hieman hankala määrittää tarkasteltavan alueen korkeuden tarkkoja koordinaatteja. Kuvassa 4.2 on esitetty kuulakeon poikkileikkaus muutamana eri ajankohtana.



Kuva 4.2: Poikkileikkaus kuulakeosta eri ajanhetkinä: (a) t = 0 s. (b) t = 11,15 s. (c) t = 22,3 s.

Kuvasta voidaan havaita, että kuulakeon yläosa elää jonkin verran simulaation aikana. Aivan simulaation alussa kuulakeko on hieman korkeampi keskeltä kuin reunoilta, kun taas simuloinnin aikana kuulakeon yläreuna on jokseenkin tasainen. Lisäksi kuvasta nähdään, miten simuloinnin alussa kuulia on poistoputkessa enemmän, sillä silloin poistoputki oli vielä kiinni. Sen jälkeen poistoputki avattiin, ja kuulat alkoivat virtaamaan reaktorista ulos, jonka jälkeen ne syötettiin takaisin reaktorin yläosaan.

Kuvasta 4.2 nähdään, että reaktorin kartiomainen alaosa alkaa ja sylinterimäinen osa loppuu korkeuden ollessa 0 m. Tämä on siis tarkastelualueen alaraja. Kuvista havaitaan, että kuulien poistoputken säde on noin 0,25 m, ja kuulia myös pudotetaan reaktorin yläosaan samansuuruiselle alueelle. Reaktorin sylinterimäisen osan kokonaishalkaisija on 1,8 metriä, joten sen alueen pinta-ala, johon kuulia ei pudoteta on

$$A_2 = \pi(r_2^2 - r_1^2) = \pi \cdot ((0{,}9\ \mathrm{m})^2 - (0{,}25\ \mathrm{m})^2) = 2{,}348\ \mathrm{m}^2 \qquad (4.1)$$

Kokonaispinta-ala sen sijaan on

$$A_{\mathrm{tot}} = \pi r^2 = \pi \cdot (0{,}9\ \mathrm{m})^2 = 2{,}545\ \mathrm{m}^2 \qquad (4.2)$$

Kuulakeon yläosan ulompi osa, johon kuulia ei pudoteta, käsittää täten kokonaispinta-alasta noin 92,3 %. Voidaan olettaa, että kuulat jakautuvat radiaalisessa suunnassa tasaisesti koko poikkipinta-alan alueelle, eli noin 92 % prosenttia kuulista sijaitsee ulkoreunalla alueella, jossa r > 0,25. Tarkastelualueen yläraja valittiin siten, että etsittiin korkeus, jolla simulaation aikana kuulien osuus alueella r > 0,25 laski alle 90 %:n. Tämä korkeus oli 1,25 m, eli se valittiin tarkastelualueen ylärajaksi. Kokonaisuudessaan tarkasteltava alue oli siis täten $0 \leq z < 1{,}25$ m.

4.3 Analysointimenetelmät

Kuulien virtauksen tutkimiseksi reaktori jaettiin viiteen yhtä suureen osaan aksiaalisessa suunnassa. Yhden aksiaalisen alueen korkeus on siis 25 cm. Aksiaalisten lohkojen rajat on esitetty taulukossa 4.1.

Taulukko 4.1: Aksiaalisten lohkojen rajat.

Alue	Rajat [m]
1	$0 \leq z < 0{,}25$
2	$0{,}25 \leq z < 0{,}5$
3	$0{,}5 \leq z < 0{,}75$
4	$0{,}75 \leq z < 1$
5	$1 \leq z < 1{,}25$

Kuulien liikkeen tarkastelua radiaalisessa, eli horisontaalisessa, suunnassa varten kuulien x- ja y-koordinaatit muutettiin napakoordinaattimuotoon, jolloin kuulan sijainti origoon nähden ilmoitetaan säteen (etäisyys origosta) sekä säteen ja positiivisen x-akselin välisen kulman avulla. Origo sijaitsee radiaalisessa suunnassa keskellä reaktoria, ja kuten kuvasta 4.2 havaittiin, korkeussuunnassa (z = 0) origo sijaitsee reaktorin alaosassa kartiomaisen ja sylinterimäisen osan taitteessa.

Kuulan säde, eli sen etäisyys origosta, saadaan yksinkertaisesti Pythagoraan lauseesta:

$$r = \sqrt{x^2 + y^2} \qquad (4.3)$$

Kuulien kulmaa laskettaessa täytyy ottaa huomioon se, ovatko x- ja y-koordinaatit negatiivisia vai positiivisia. Kulma voidaan laskea eri tapauksissa seuraavilla yhtälöillä

$$\theta = \tan^{-1}\left(\frac{y}{x}\right), \;\; x > 0 \text{ ja } y \geq 0 \tag{4.4}$$

$$\theta = \tan^{-1}\left(\frac{y}{x}\right) + 2\pi, \;\; x > 0 \text{ ja } y < 0 \tag{4.5}$$

$$\theta = \tan^{-1}\left(\frac{y}{x}\right) + \pi, \;\; x < 0 \tag{4.6}$$

Kuulien radiaalisia nopeuksia tarkasteltiin niiden reaktorin säteen suuntaisen nopeuden ja ratanopeuden (säteen normaalin suuntainen nopeus) avulla. Reaktorin säteen suuntainen nopeus on yksinkertaisesti kuulan etäisyyden muutos jaettuna ajan muutoksella:

$$v_{\text{säde}} = \frac{\Delta r}{\Delta t} \tag{4.7}$$

Ratanopeus voidaan laskea kulmanopeuden ja etäisyyden avulla:

$$v_{\text{rata}} = \omega r = \frac{\Delta \theta}{\Delta t} r \tag{4.8}$$

Aksiaalinen, eli vertikaalinen, nopeus sen sijaan voidaan laskea kuulan korkeuden (z-koordinaatin) muutoksen ja ajan muutoksen avulla:

$$v_{\text{z}} = \frac{\Delta z}{\Delta t} \tag{4.9}$$

Kuulien oleskeluajat, eli se, miten kauan kuulat tietyllä alueella viettivät aikaa, voitiin laskea tutkimalla sitä, kuinka monta datapistettä kuulalla kyseisellä alueella on. Eli toisin sanoen laskemalla kuinka monena ajanhetkenä kuula alueella oli:

$$t = \Delta t \cdot N, \tag{4.10}$$

jossa Δt on datapisteiden, eli tiedostojen välillä kulunut aika, ja N on datapisteiden lukumäärä.

Radiaalisessa suunnassa kuulien liikettä tarkasteltiin tutkimalla kuulan etäisyyden muutosta kuulien liikkuessa reaktorissa alaspäin. Kuulan etäisyyden muutos on yksinkertaisesti

$$\Delta r = \; r_0 - r_{\text{n}}, \tag{4.11}$$

jossa r_0 on kuulan etäisyys keskilinjasta ensimmäisenä ajanhetkenä, jona se on tarkasteltavalla alueella, ja r_n on kuulan etäisyys, kun kuula on liikkunut reaktorissa alaspäin.

4.4 Python-ohjelman rakenne ja toiminta

Virtausdatan analysointia varten kehitettiin ohjelma Python-ohjelmointikieltä käyttäen. Ohjelman pääasiallinen tehtävä oli lukea virtausdata tekstitiedostoista, suorittaa tarvittavat laskutoimenpiteet, esimerkiksi nopeuksien laskemiseksi, yllä esitetyillä yhtälöillä. Vaikka tekstitiedostot sisälsivät kuulien liikkeestä paljonkin tietoa, tekstitiedostoista luettiin ainoastaan x-, y- ja z-koordinaatit eri ajanhetkiltä, sillä ne riittävät kaikkiin yllä mainittuihin laskutoimenpiteisiin. Lisäksi luettiin kuulien indeksit, jonka avulla kuulat pystyttiin erottamaan toisistaan. Ohjelmalla pystytään lukemaan tietoja miltä vain halutulta aikaväliltä, ottaen toki huomioon simulointiajan pituuden. Tässä työssä tehdyssä tarkastelussa hyödynnettiin kaikkea saatavissa olevaa dataa, eli tiedot luettiin koko simuloinnin ajalta.

Ohjelma on jaettu kolmeen eri Python-tiedostoon. Yksi tiedosto (dataloader.py) sisältää Python-objektin, jonka funktioilla virtausdataa voidaan analysoida. Objekti sisältää funktiot tekstitiedostojen lukemiselle ja kaikille yllä mainituille laskutoimenpiteille. Lisäksi funktioita on datan muuttamiselle parempaan muotoon tulosten esittämistä varten. Tämä tiedosto on esitetty liitteessä 2.

Toinen tiedosto sisältää (plot_funcs.py) sisältää tulosten esittämisessä, eli kuvaajien piirrossa, hyödyllisiä funktioita. Tämä tiedosto on esitetty liitteessä 3.

Kolmas tiedosto (main.py) sisältää ohjelman käytännön suorittamisen. Tässä tiedostossa käytetään kahden muun tiedoston sisältämiä funktioita datan analysointiin ja tulosten esittämiseen. Tiedosto on esitetty liitteessä 4.

5 ANALYSOINNIN TULOKSET

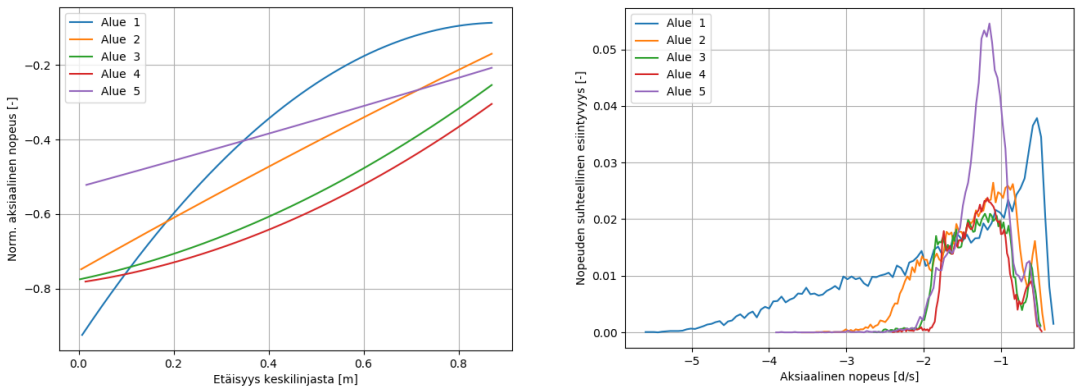
Tässä osiossa esitetään virtausdatan analysoinnin tuloksia reaktorin eri osissa. Kuulien virtausta on kuvattu muodostamalla muun muassa jakaumat aksiaaliselle ja radiaalisille nopeuksille sekä kuulien oleskeluajoille. Lisäksi on esitelty jakaumat sille, kuinka paljon kuulat liikkuivat radiaalisessa suunnassa kulkiessaan reaktorissa alaspäin. Pääasiallisen tarkastelualueen ($0 \leq z < 1{,}25$ m) lisäksi kuulien oleskeluaikoja tutkittiin myös alueella, jossa reaktorin sylinterimäinen ja kartiomainen osa kohtaavat. Tämä alue havaittiin aiemmin esitetyssä tutkimuksessa (Li et al. 2015) olevan kriittinen alue kuulien liikkeen pysähtymisen kannalta.

Kuulien erisuuntaisten nopeuksien sekä oleskeluaikojen kuvaajat etäisyyden funktiona on piirretty muodostamalla datapisteille polynomisovitteet, joten kuvaajat kertovat enemmänkin nopeuden ja oleskeluajan keskimääräisestä käyttäytymisestä eri reaktorin osissa, eivätkä välttämättä kerro nopeuden tai oleskeluaikojen täsmällisiä lukuarvoja. Tämä menetelmä on esitetty liitteessä 1.

5.1 Aksiaalisen nopeuden jakauma

Kuulien nopeuksien tarkastelussa on otettu huomioon kaikki kuulat, jotka ovat kulkeneet kokonaan tarkasteltavan alueen läpi. Kuvassa 5.1 on esitetty keskimääräinen aksiaalinen nopeus etäisyyden funktiona sekä aksiaalisen nopeuden suhteellinen jakauma eri aksiaalisilla alueilla.

(a) (b)



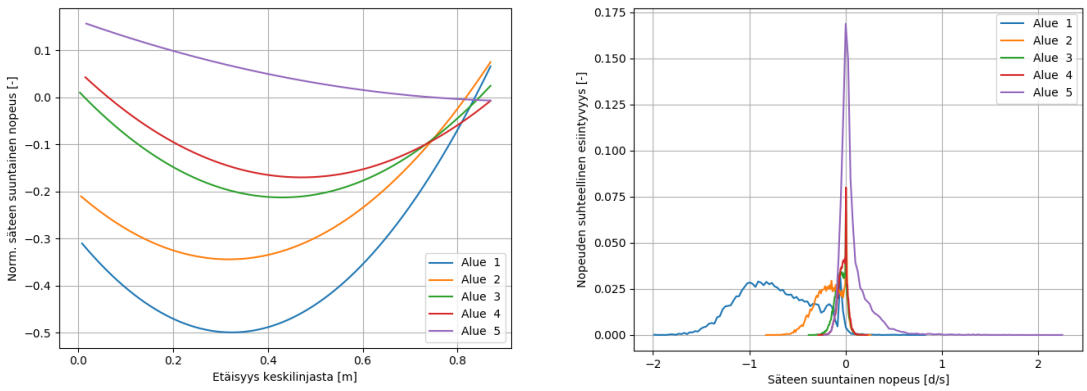
Kuva 5.1: Aksiaalinen nopeus eri alueilla: (a) Keskimääräinen aksiaalinen nopeus etäisyyden funktiona. (b) Nopeuden suhteellinen jakauma.

Aksiaalisen nopeuden kuvaajat on normalisoitu siten, että nopeuden etumerkki säilyy, jolloin nopeuden arvot ovat maksimissaan välillä -1…0. Tosin, kuten kuvasta 5.1(a) havaitaan, keskimääräiset nopeudet ovat suurimmillaankin hieman yli 0,8-kertaiset maksiminopeuteen nähden. Kaikilla alueilla on selkeästi havaittavissa, että aksiaalinen nopeus on suurimmillaan lähellä keskilinjaa. Alueella 1 ero on huomattavin, kun taas esimerkiksi alueella 5 nopeuden jakauma on huomattavasti tasaisempi. Suhteellisen jakauman kuvaajissa aksiaalinen nopeus on skaalattu kuulan halkaisijan mukaan (d = 6 cm), eli nopeuden yksikkö on kuulan halkaisija per sekunti. Suurimmilta osin kaikilla alueilla kuulien aksiaalinen nopeus on itseisarvoltaan pienempi kuin 2 d/s.

5.2 Radiaalisten nopeuksien jakaumat

Kuvassa 5.2 on esitetty säteen suuntaisen nopeuden kuvaajat sekä suhteellinen jakauma eri aksiaalisilla alueilla.

(a) (b)



Kuva 5.2: Säteen suuntainen nopeus eri alueilla: (a) Keskimääräiset säteen suuntaiset nopeudet etäisyyden funktiona. (b) Säteen suuntaisen nopeuden suhteellinen jakauma.

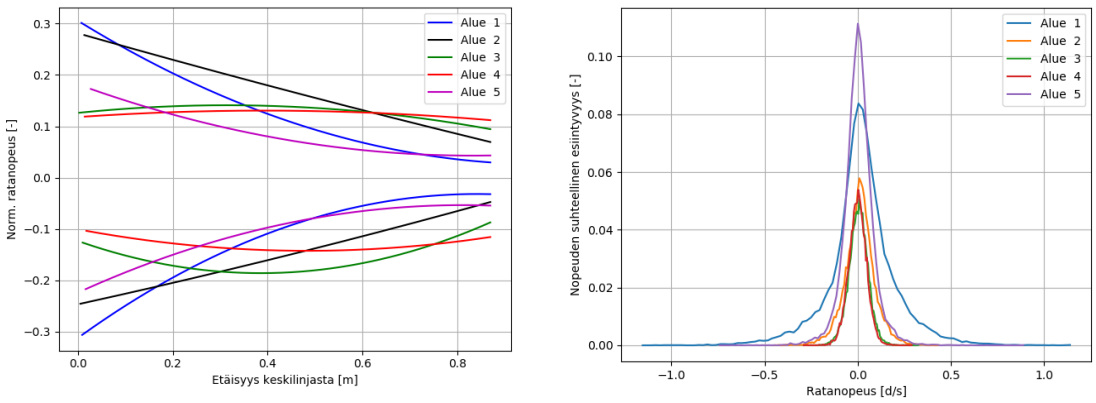
Myös säteen suuntaisen nopeuden kuvaajissa nopeuden arvot ovat normalisoitu siten, että nopeuden etumerkki säilyy. Negatiiviset nopeudet ovat siis välillä -1…0 ja positiiviset välillä 0…1. Kuvasta 5.2(a) havaitaan, että alueella 5 säteen suuntaisen nopeuden arvo on kaikkialla muualla paitsi aivan reaktorin ulkoreunalla positiivinen. Tämä tarkoittaa sitä, että vielä tällä alueella suuri osa kuulista liikkuu ulkoreunaa kohti sen jälkeen, kun ne ovat reaktorin keskiosaan pudonneet. Muilla alueilla nopeus on lähes poikkeuksetta negatiivista, ja nopeus on suurimmillaan reaktorin puolivälin kohdilla radiaalisessa

suunnassa. Havaitaan myös, että säteen suuntainen nopeus kasvaa reaktoria alaspäin mentäessä. Tämä on loogista, sillä kuvasta 5.1(a) havaittiin, että kuulat liikkuvat reaktorin alaosassa verrattain suurella nopeudella alaspäin reaktorin kartiomaista osaa ja poistoputkea kohti. Tämä antaa reaktorin ulkolaidoilla oleville kuulille tilaa liikkua keskemmälle päin reaktorin alaosassa.

Kuvasta 5.2(b) voidaan tehdä samankaltaisia havaintoja; alueella 5 verrattain suurella osalla kuulia on positiivinen säteen suuntainen nopeus. Alaspäin kuljettaessa negatiivisten säteen suuntaisten nopeuksien osuus kasvaa huomattavasti, ja alueella 1 positiivisten säteen suuntaisten nopeuksien osuus on minimaalinen.

Samat kuvaajat ratanopeudelle on esitetty kuvassa 5.3.

(a) (b)



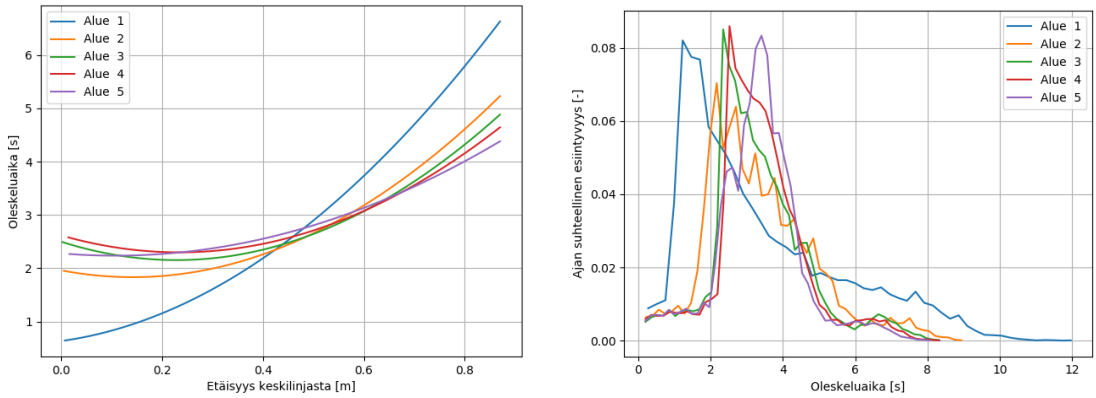
Kuva 5.3: Ratanopeus eri alueilla: (a) Keskimääräinen ratanopeus etäisyyden funktiona. (b) Ratanopeuden suhteellinen jakauma.

Kuten kuvasta 5.3(b) havaitaan, ratanopeus on jakautunut hyvin symmetrisesti negatiivisen ja positiivisen nopeuden välillä. Tästä syystä kuvassa 5.3(a) on muodostettu kuvaajat erikseen negatiivisille ja positiivisille nopeuksille. Suhteellisen jakauman kuvaajista nähdään, että ratanopeus on suurimmillaan noin 1 d/s molempiin suuntiin, mutta kuva 5.3(a) kertoo, että keskimäärin kuulien nopeus on pääasiassa vain noin kolmasosan tästä, jos sitäkään. Kaikilla alueilla ratanopeuden suhteellinen jakauma on hyvin pitkälti painottunut nollan lähelle. Hyvin symmetrinen ratanopeuden jakauma tarkoittaa sitä, että reaktorissa ei pitäisi esiintyä ainakaan suuria säännöllisiä pyörteitä, joissa kuulat liikkuisivat suurissa määrin samaan suuntaan.

5.3 Kuulien oleskeluajat

Kuulien oleskeluajan tarkastelussa on otettu huomioon kaikki kuulat, jotka ovat tarkastelun alaisella alueella olleet. Tällöin tuloksissa esiintyy myös hyvin lyhyitä aikoja, mutta myös alueella pidempään olleet, tai mahdollisesti sinne pysähtyneet kuulat pitäisi löytää, jos sellaisia on. Kuvassa 5.4 on esitetty kuvaajat kuulien oleskeluajoille eri aksiaalisilla alueilla etäisyyden funktiona sekä oleskeluaikojen suhteellinen jakauma.

(a) (b)



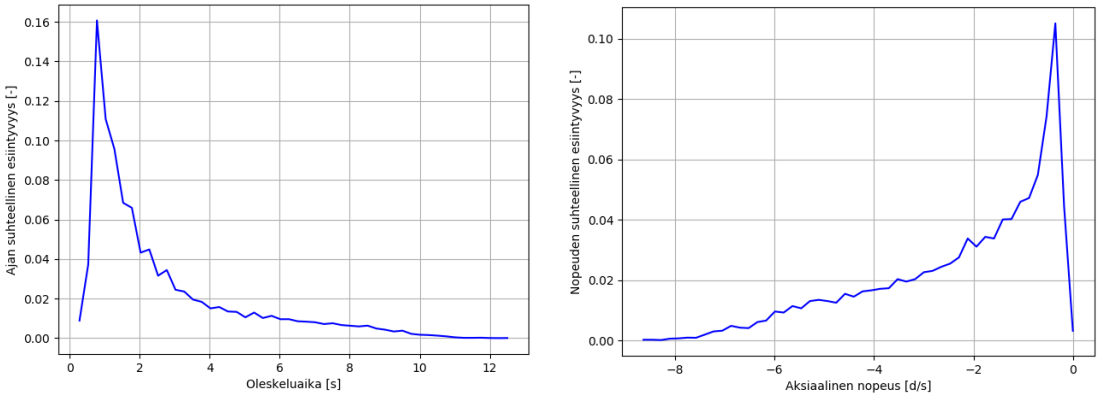
Kuva 5.4: Kuulien oleskeluajat eri alueilla: (a) Keskimääräiset oleskeluajat etäisyyden funktiona. (b) Oleskeluaikojen suhteellinen jakauma.

Kuulien oleskeluaika tarkastelualueilla on kääntäen verrannollinen aksiaaliseen nopeuteen, joten on luonnollista, että lähellä keskilinjaa olevien kuulien oleskeluajat ovat lyhyempiä. Alueella 1 on siis, kuten aksiaalisenkin nopeuden tapauksessa, jälleen suurimmat erot, niin lyhyimmät kuin pisimmät oleskeluajat löytyvät kyseiseltä alueelta. Muilla alueilla jakauma on tasaisempi. Oleskeluaikojen suhteellinen jakauma kertoo, että kuulien oleskeluajat ovat painottuneet vahvasti välille $1 < t < 6$ s.

5.4 Kuulien liike reaktorin taitteessa

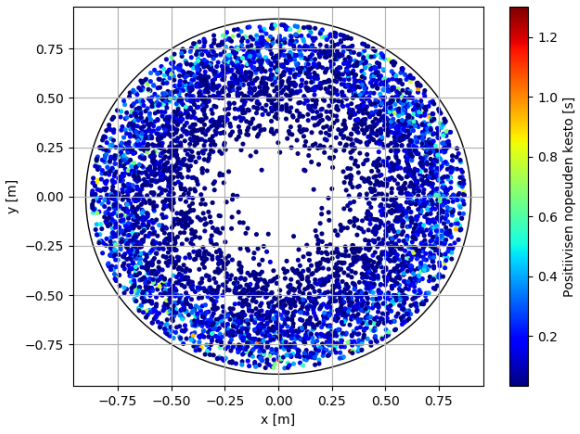
Kuulien virtaus reaktorin sylinterimäisen ja kartiomaisen osan taitteessa oli aiemmassa tutkimuksessa havaittu olevan sellainen alue, johon kuulia voi päästä pakkautumaan, ja niiden liike mahdollisesti jopa pysähtymään. Täten kuulien oleskeluaikaa ja aksiaalista nopeutta tutkittiin vielä erikseen alueella $-0{,}1 \leq z \leq 0{,}1$ m. Kuvassa 5.5 on esitetty tällä alueella olleiden kuulien oleskeluajan ja aksiaalisen nopeuden suhteelliset jakaumat.

(a) (b)



Kuva 5.5: Kuulien liike reaktorin alaosan taitteessa: (a) Oleskeluajan suhteellinen jakauma. (b) Aksiaalisen nopeuden suhteellinen jakauma.

Tässäkin tarkastelussa otettiin huomioon kaikki kuulat, jotka alueella sattuivat olemaan siitä huolimatta, kulkivatko ne kokonaan alueen läpi vai eivät. Oleskeluajan jakaumasta voidaan nähdä, että läheskään koko simulaation kestoa mikään kuula ei ainakaan ole alueella jumissa ollut. Samaten nopeuden jakaumasta nähdään, että aksiaalinen keskinopeus on ollut kaikilla kuulilla negatiivista, joskin mahdollisesti hyvin hidasta. Kuulien pakkautuessa reaktorin alaosan taitteeseen on odotettavissa, että kuulilla voi aksiaalinen nopeus hetkellisesti olla nollan lisäksi myös positiivista. Kuvassa 5.6 on esitetty kaikki ne kuulat reaktorin alaosan taitteessa, joilla on edes hetkellisesti ollut aksiaalinen nopeus ≥ 0 d/s.

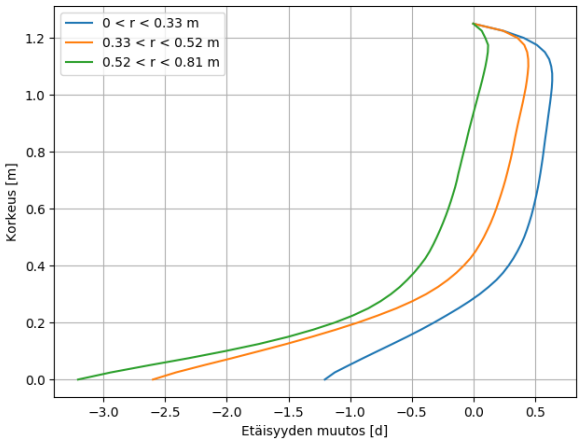


Kuva 5.6: Kuulat, joilla on ollut positiivinen aksiaalinen nopeus reaktorin alaosassa.

Kaiken kaikkiaan 5635 kuulalla (hieman alle 30 % kaikista alueella olleista kuulista) aksiaalinen nopeus oli positiivinen (tai nolla) ainakin hetkellisesti reaktorin alaosassa, ja kuulat sijaitsivat pääasiassa alueella, jossa r > 0,25 m, eli muualla kuin suoraan poistoputken yläpuolella. Mutta kuten kuvasta nähdään, kuulilla oli pääsääntöisesti positiivinen aksiaalinen nopeus yhteensä (ei välttämättä yhtäjaksoisesti) reilusti alle sekunnin ajan. Täten kuvasta 5.5 ja kuvasta 5.6 voidaan päätellä, että vaikka sylinterimäisen ja kartiomaisen osan taitteessa varsinkin ulkoreunaa lähellä olevien kuulien aksiaalinen nopeus hidastuu, ja hetkellisesti voi kääntyä positiiviseksikin, pidempiaikaisempaa pakkautumista tai alaspäin suuntautuvan liikkeen pysähtymistä ei pääse tapahtumaan.

5.5 Kuulien radiaalinen liike

Kuulien radiaalista liikettä tarkasteltiin vain reaktorin säteen suuntaisesti. Kuten kuvasta 5.3 havaittiin, ratanopeus on hidasta verrattuna säteen suuntaiseen nopeuteen ja symmetristä. Tämä tarkoittaa, että ratanopeuden suuntainen liikekin on verrattain vähäistä, ja symmetristä sekin. Säteen suunnassa paikan muutos on myös siinä mielessä kiinnostavaa, että se vaikuttaa esimerkiksi kuvassa 2.1(b) esitetyn dynaamisen keskiheijastimen toimintaan. Pieni säteen suuntainen liike mahdollistaa dynaamisen keskiheijastimen käytön, sillä silloin polttoaine- ja heijastinkuulat pysyisivät erillään. Kuvassa 5.7 on esitetty säteen suuntaisen liikkeen suuruus koko reaktorin matkalla (0 ≤ z < 1,25 m).



Kuva 5.7: Säteen suuntaisen liikkeen suuruus.

Tarkastelussa otettiin huomioon kuulat, jotka liikkuivat yhtäjaksoisesti koko reaktorin läpi, ja tällaisia kuulia oli 4521 kappaletta, eli noin viidesosa kaikista kuulista. Kuulat jaettiin kolmeen radiaaliseen alueeseen siten, että kuulien lukumäärä joka alueella oli sama. Uloimmaisen alueen rajoista havaitaan, että reaktorin ulkoreunalla ($0{,}8 < r < 0{,}9$ m) ei kokonaan reaktorin läpi menneitä kuulia juurikaan ollut.

Kuvaajat on muodostettu siten, että kuulan etäisyyttä ensimmäisenä ajanhetkenä, jolloin se tarkkailualueelle saapuu ($z \approx 1{,}25$ m), on käytetty vertailuarvona ja kuulan etäisyyden muutos tähän arvoon on laskettu eri korkeuksilla 2,5 cm:n välein. Täten saadaan kuvaaja kuulan etäisyyden muutokselle korkeuden funktiona. Negatiivinen säteen muutos kuvaa etäisyyden pienenemistä, ja positiivinen muutos etäisyyden kasvua.

Kuten kuvasta havaitaan, kuulien liikkeen luonne on hyvin samankaltaista eri radiaalisissa lohkoissa. Ensin tapahtuu pieni liike positiiviseen suuntaan, jonka jälkeen etäisyys alkaa pienenemään. Mitä lähempänä reaktorin ulkolinjaa ollaan, sitä suurempaa on etäisyyden muutos. Kuvaajista on havaittavissa myös, miten reaktorin alaosaa lähestyttäessä etäisyyden muutosnopeus kasvaa huomattavasti, kun lähellä keskilinjaa olevien kuulien aksiaalinen nopeus kasvaa, ja kuulille syntyy tilaa siirtyä keskemmälle. Osaltaan myös kuulien aksiaalisen nopeuden hidastuminen ja pakkautuminen sylinterimäisen ja kartiomaisen osan taitteeseen johtaa siihen, että kuulien täytyy siirtyä keskemmälle, sillä niillä ei välttämättä ole tilaa liikkua suoraan alaspäin.

Kuulien säteen suuntainen liike on suurelta osin sen verran pientä verrattuna radiaalisen lohkon leveyteen, että korkeussuunnassa suuressa osassa reaktoria kuulat pysyvät omissa lohkoissaan, ainakin tällä lohkojaolla.

6 JOHTOPÄÄTÖKSET

Kuulien aksiaalinen nopeus oli suurempaa reaktorin keskilinjan läheisyydessä, ja nopeus keskilinjan lähellä kasvoi reaktorin alaosaa lähestyttäessä. Aksiaalisen nopeuden jakauma etäisyyden funktiona oli jyrkempi reaktorin alaosassa, eli kun aksiaalinen nopeus kasvoi keskilinjan lähellä, nopeus hidastui reaktorin seinämän läheisyydessä. Kuulien keskimääräinen aksiaalinen nopeus oli valitulla tarkkailualueella poikkeuksetta negatiivista, eli kuulat liikkuivat jatkuvasti alaspäin. Aksiaalinen nopeus oli myös suurempaa kuin radiaaliset nopeudet, eli alaspäin suuntautuva liike oli dominoivin liikesuunta, kuten painovoiman vaikutuksesta voi olettaakin.

Eri radiaalisten nopeuksien suuruudessa ja luonteessa oli eroja. Säteen suuntainen nopeus oli esimerkiksi pääsääntöisesti suurempaa kuin ratanopeus. Säteen suuntainen nopeus oli ylintä aksiaalista tarkkailualuetta lukuun ottamatta lähes poikkeuksetta negatiivista, eli kuulien liike suuntautui tässä suunnassa reaktorin keskiosaa kohti. Tämä johtuu siitä, että kuulien suurempi aksiaalinen nopeus keskiosassa antaa ulompana oleville kuulille tilaa liikkua keskelle päin. Ratanopeus sen sijaan oli hyvin symmetristä, eli jokaisella tarkkailualueella negatiivisten ja positiivisten nopeuksien määrä ja suuruus oli samaa luokkaa. Tästä voidaan päätellä, että säännöllisiä pyörteitä reaktorissa ei synny, sillä silloin suurella osalla kuulista olisi samansuuntainen ratanopeus. Myöskään sitä ei havaittu, että etäisyydellä olisi vaikutusta ratanopeuden symmetrisyyteen. Niin aksiaalinen nopeus, säteen suuntainen nopeus kuin ratanopeus pieneni etäisyyden kasvaessa reaktorin keskilinjasta. Tosin varsinkin ratanopeuden tapauksessa ero on verrattain pieni.

Kuulien oleskeluajat eri aksiaalisilla alueilla ovat kääntäen verrannollisia aksiaaliseen nopeuteen, joten oleskeluaikojen kuvaajat ja suhteelliset jakaumat ovat hyvin saman luonteisia kuin aksiaalisen nopeudenkin kuvaajat; lähellä keskiosaa olevien kuulien oleskeluajat ovat lyhyempiä kuin lähempänä reunaa olevien kuulien.

Reaktorin sylinterimäisen ja kartiomaisen osan taite tiedettiin aiemmista tutkimuksista alueeksi, johon kuulat voivat pakkautua, jolloin niiden alaspäin suuntautuva liike voi myös pysähtyä. Tätä ilmiötä ei tässä tutkimuksessa havaittu, sillä kuulien oleskeluajat taitteessa eivät olleet juurikaan esimerkiksi aluetta 1 ($0 \leq z < 0{,}25$ m) pidempiä. Vaikka

kuulien aksiaalinen nopeus taitteessa hieman hidastuikin, ja paikoitellen kääntyi positiiviseksikin, positiivinen aksiaalinen nopeus oli hyvin lyhytkestoista. Kaikkien kuulien keskinopeus oli myös negatiivinen, joka sekin on merkki siitä, että kuulat liikkuivat silläkin alueella vakaasti alaspäin.

Kuulien liike radiaalisessa suunnassa koko reaktorin matkalla oli suurempaa reaktorin ulkoreunalla kuin lähempänä keskilinjaa. Aivan tarkkailualueen yläosassa kuulat liikkuivat vielä keskimäärin keskilinjasta poispäin sen jälkeen, kun ne olivat keskiosaan pudonneet. Kuulien liike kuitenkin kääntyi nopeasti reaktorin keskilinjaa kohti. Reaktorin alaosaa lähestyttäessä, noin korkeudella $z \approx 0{,}4$ m, kuulien säteen suuntaisen liikkeen muutosnopeus kasvoi huomattavasti. Tällä korkeudella siis lähellä keskilinjaa olevien kuulien suurempi aksiaalinen nopeus, ja reaktorin alaosan taitteen takia hidastuvat kuulat alkavat vaikuttaa kuulien radiaaliseen liikkeeseen enemmän.

Virtausdatan analysoinnissa saavutetut tulokset ovat pääsääntöisesti yhteneviä luvussa kolme esitettyjen tutkimusten tulosten kanssa. Virtausnopeuden jakauma on samanlainen kuin mitä aiemmissa tutkimuksissa oli havaittu. Kuulien radiaalinen liike havaittiin sen verran pieneksi, että suurta kuulien sekoittumista radiaalisessa suunnassa ei suuressa osassa reaktoria pääse välttämättä tapahtumaan. Tämäkin havainto on linjassa aikaisempien tutkimusten kanssa.

7 YHTEENVETO

Kuulakekoreaktori on kaasujäähdytteinen, grafiittihidasteinen ydinreaktorityyppi, jossa ydinpolttoaine on sijoitettu tuhansiin grafiittipäällysteisiin kuuliin pieninä rakeina. Työssä selvitettiin kuulakekoreaktorin ja polttoainekuulan keskeiset ominaisuudet ja rakenne. Tämän lisäksi selvitettiin aikaisemmin tehtyjä kuulien virtaukseen liittyviä tutkimuksia ja niiden tuloksia. Osittain aikaisempien tutkimusten perusteella päätettiin, mitä tässä työssä tehtävä virtausdatan analysointi pitää sisällään ja mitä ilmiöitä analysoinnilla kannattaa selvittää. Täten analysoinnin tuloksia voitiin verrata muiden tutkimusten tuloksiin.

Työssä tehdyn polttoainekuulien virtausdatan analysoinnin pääasiallisena tutkimuskohteena oli selvittää polttoainekuulien aksiaalisen nopeuden jakauma eri osissa reaktoria. Tämän lisäksi aiempien tutkimusten innoittamana päätettiin esimerkiksi tutkia tarkemmin kuulien liikettä reaktorin alaosan sylinterimäisen ja kartiomaisen osan taitteessa. Myös kuulien radiaalista liikettä ja nopeutta tutkittiin, joka mahdollisti kattavamman ymmärryksen kuulien virtauksesta.

Polttoainekuulien liike ei pysähdy missään osassa reaktoria, ja kuulien aksiaalinen nopeus on suurempi reaktorin keskilinjan läheisyydessä ja pienempi seinämän lähellä. Reaktorin yläosassa aksiaalinen nopeus oli hitaampaa kuin alaosassa. Aksiaalinen nopeus oli reaktorissa pääsääntöisesti radiaalisia nopeuksia suurempi, joten alaspäin suuntautuva liike on dominoiva liikesuunta. Reaktorin keskilinjan läheisyydessä, suurempi aksiaalinen nopeus mahdollisti sen, että reaktorissa ulompana liikkuvat kuulat pääsevät liikkumaan keskemmälle päin, eli negatiivinen säteen suuntainen nopeus kasvoi, ja kuulat siirtyivät keskelle päin. Lisäksi reaktorin alaosan kartiomainen rakenne saa kuulat liikkumaan keskiosaa kohti. Ratanopeudesta eri osissa reaktoria nähtiin, että kuulat eivät muodosta reaktorissa suurempia säännöllisiä pyörteitä. Kuulien radiaalinen liike reaktorissa on verrattain pientä, maksimissaan kuulan etäisyys reaktorin keskilinjasta muuttui noin kolmen halkaisijan verran.

Tässä tutkimuksessa tehdyt havainnot ovat yhteneviä aiheesta aiemmin tehtyjen tutkimusten tulosten kanssa.

LÄHDELUETTELO

Choi Jaehyuk et al. 2004. Diffusion and mixing in gravity-driven dense granular flows. Physical Review Letters, 2004. Vol. 92. Nro. 17.

Duderstadt James J., Hamilton Louis J. 1976. Nuclear Reactor Analysis. John Wiley & Sons, Inc. Yhdysvallat. 650 s. ISBN: 0-471-22363-8

Gui Nan et al. 2014. Effect of bed configuration on pebble flow uniformity and stagnation in the pebble bed reactor. Nuclear Engineering and Design. Vol. 270. s. 295–301.

Hecker R., Rausch W., Schulten R. 1967. Development of High Temperature Thermal Reactors in Germany. Nuclear Energy Meeting 1966, Milan (Italy) December 15–17, 1966.

International Atomic Energy Agency. 2001. Current status and future development of modular high temperature gas cooled reactor technology. Nuclear Power Technology Development Section. Wien, Itävalta. 259 s. ISSN: 1011-4289

International Atomic Energy Agency. 2013. Evaluation of High Temperature Gas Cooled Reactor Performance: Benchmark Analysis Related to the PBMR-400, PBMM, GT-MHR, HTR-10 and the ASTRA Critical Facility. Nuclear Power Technology Development Sector. Wien, Itävalta. 676 s. ISBN: 978-92-0-137610-7

International Atomic Energy Agency. 2020. Advances in Small Modular Reactor Technology Developments. A Supplement to: IAEA Advanced Reactors Information System (ARIS) 2020 Edition. Itävalta.

Jiang S.Y. et al. 2012. Experimental and numerical validation of a two-region-designed pebble bed reactor with dynamic core. Nuclear Engineering and Design. Vol. 246. s. 277-285.

Kadak A.C. 2005. A future for nuclear energy: pebble bed reactors. Int. J. Critical Infrastructures, Vol. 1. Nro. 4. s. 330-345.

Kadak A.C. Bazant Martin. 2004. Pebble Flow Experiments for Pebble Bed Reactors. 2nd International Topical Meeting on High Temperature Reactor Technology, September 22-24, 2004, Peking, Kiina.

Krüger K.J., Ivens G.P. 1985. Safety-Related Experiences with the AVR Reactor. Teoksessa: International Atomic Energy Agency. Gas-Cooled Reactor Safety and Accident Analysis. Proceedings of a Specialist's Meeting on Safety and Accident Analysis for Gas-Cooled Reactors Organized by the International Atomic Energy Agency and Held at Oak Ridge National Laboratory Oak Ridge, 13-15 May 1985. Wien, Itävalta. s. 61-70.

Li Yanjie, Yong Xu, Jiang Shengyao. 2009. DEM simulations and experiments of pebble flow with monosized spheres. Powder Technology. Vol. 193. s. 312-318.

Li Yu et al. 2015. Effect of wall structure on pebble stagnation behavior in pebble bed reactor. Annals of Nuclear Energy, Vol. 80. s. 195-202.

Moormann Rainer. 2008. Fission Product Transport and Source Terms in HTRs: Experience from AVR Pebble Bed Reactor. Science and Technology of Nuclear Installations. Vol. 2008. s. 1-14.

Moormann R. 2009. AVR prototype pebble bed reactor: a safety re-evaluation of its operation and consequences for future reactors. Kerntechnik. Vol. 74. s. 8-21.

Moormann Rainer, Kemp Scott R., Li Ju. 2018. Caution Is Needed in Operating and Managing the Waste of New Pebble-Bed Nuclear Reactors. Joule. Vol. 2. s. 1911-1914.

Nuclear Engineering International. 2020. South Africa seeks to revive PBMR project. [verkkojulkaisu] [viitattu 22.10.2020] Saatavissa: https://www.neimagazine.com/news/newssouth-africa-seeks-to-revive-pbmr-project-7765057

Oehme H., Schöning J. 1970. Design, Features and Engineering Status of the THTR 300 MWe Prototype Power Station. Proceedings of the gas-cooled reactor information meeting at the Oak Ridge National Laboratory, April 27-30, 1970, Oak Ridge, Tennessee, USA. s. 161-179.

Reitsma Frederik. 2004. The Pebble Bed Modular Reactor Layout and Neutronics Design of the Equilibrium Cycle. PHYSOR 2004 – The Physics of Fuel Cycles and Advanced Nuclear Systems: Global Developments. 2004.

Rintala Ville. 2011. Monte Carlo -reaktorifysiikkalaskennan ja laskennallisen virtausmekaniikan kytkentä kuulakekoreaktorissa. Diplomityö. Lappeenrannan teknillinen yliopisto. Ydintekniikan laboratorio. 115 s.

Rycroft Chris H. et al. 2006. Analysis of granular flow in a pebble-bed nuclear reactor. Physical Review E 74, 2006.

Schulten R. 1987. High Temperature Gas-Cooled Reactors. Teoksessa: International Atomic Energy Agency. Gas-Cooled Reactors and Their Applications (Proceedings of a Technical Committee Meeting, Juelich 20-23 October 1986). Wien, Itävalta. s. 29-39.

Shirvan Koroush et al. 2016. Technology Selection for Offshore Underwater Small Modular Reactors. Nuclear Engineering and Technology. Vol. 48. s. 1303-1314.

Wols Frank. 2015. Conceptual Design Studies of a Passively Safe Thorium Breeder Pebble Bed Reactor. Väitöskirja. Delft University of Technology, department of Radiation Science and Technology (RST). Delft, Alankomaat. 147 s.

Wu Zongxin, Lin Dengcai, Zhong Daxin. 2002. The design features of the HTR-10. Nuclear Engineering and Design. Vol. 218. s. 25-32.

Yang Xingtuan, Hu Wenping, Jiang Shengyao. 2009. Experimental Investigation on Feasibility of Two-Region-Designed Pebble-Bed High-Temperature Gas-Cooled Reactor. Journal of NUCLEAR SCIENCE and TECHNOLOGY. Vol 46. Nro. 4, s. 374-381.

Yang Xingtuan et al. 2014. Some Movement Mechanisms and Characteristics in Pebble Bed Reactor. Science and Technology of Nuclear Installations. Vol. 2014.

Zhang Zuoyi et al. 2016. The Shangdong Shidao Bay 200 ME_e High-Temperature Gas-Cooled Reactor Pebble-Bed Module (HTR-PM) Demonstration Power Plant: An Engineering and Technological Innovation. Engineering. Vol 2. Nro. 1. s. 112-118.

Ziermann E. 1987. AVR Experience. Teoksessa: International Atomic Energy Agency. Gas-Cooled Reactors and Their Applications (Proceedings of a Technical Committee Meeting, Juelich 20-23 October 1986). Wien, Itävalta. s. 71-97.

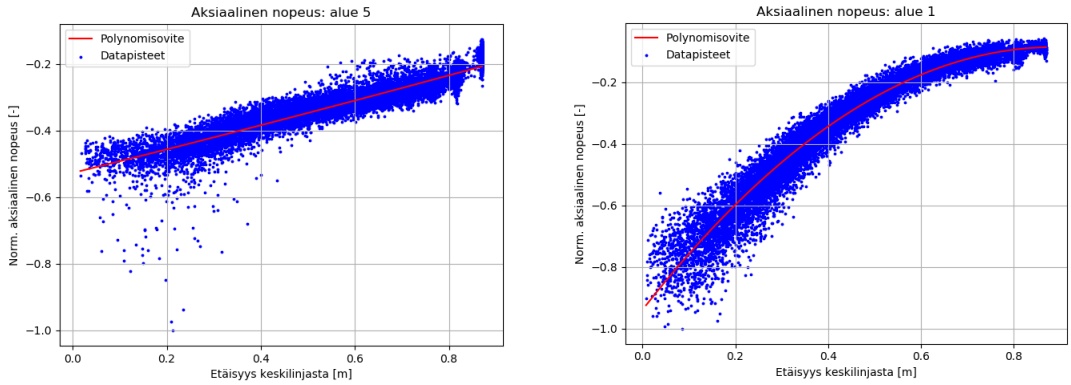
LIITE 1. POLYNOMISOVITTEIDEN MUODOSTAMINEN

Virtausnopeuksien ja oleskeluaikojen kuvaajien muodostamiseen käytettiin siis toisen asteen polynomisovitetta, joka oli muotoa

$$y(x) = ax^2 + bx + c \tag{1.1}$$

Parametrien *a*, *b*, ja *c* arvojen määrittämiseen käytettiin Pythonin scipy.optimize-moduulista löytyvää curve_fit-funktiota. Funktio siis palauttaa parametreille sellaiset arvot, joilla sovite mahdollisimman hyvin sopii datapisteisiin. Alla on muutama esimerkki siitä, miten sovite ja datapisteet asettuvat kuvaan. Kuvassa 1.1 on esitetty aksiaalisen nopeuden datapisteet ja sovitteet reaktorin ala- ja yläosassa.

(a) (b)

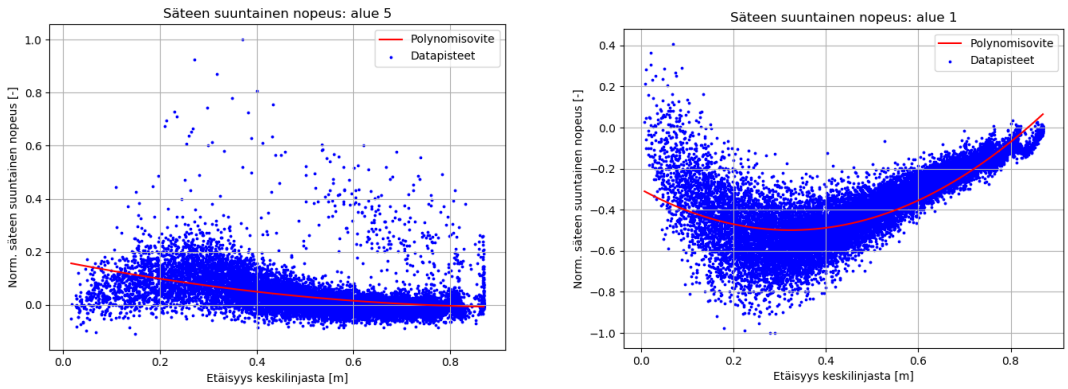


Kuva 1.1: Aksiaalisen nopeuden sovitteet ja datapisteet: (a) Reaktorin yläosassa. (b) Reaktorin alaosassa.

Kuvassa 1.1(a) datapisteet ovat hieman enemmän hajaantuneita, mutta kuten nähdään, sovite kuitenkin aika hyvin asettuu sinne, missä suurin osa datapisteistä sijaitsee. Alaosassa sen sijaan data on hyvin nipussa, ja sovite kuvaa nopeuden käyttäytymistä aika hyvin.

Kuvassa 1.2 on esitetty säteen suuntaiselle nopeudelle datapisteiden ja sovitteen suhde reaktorin ala- ja yläosassa.

(a) (b)

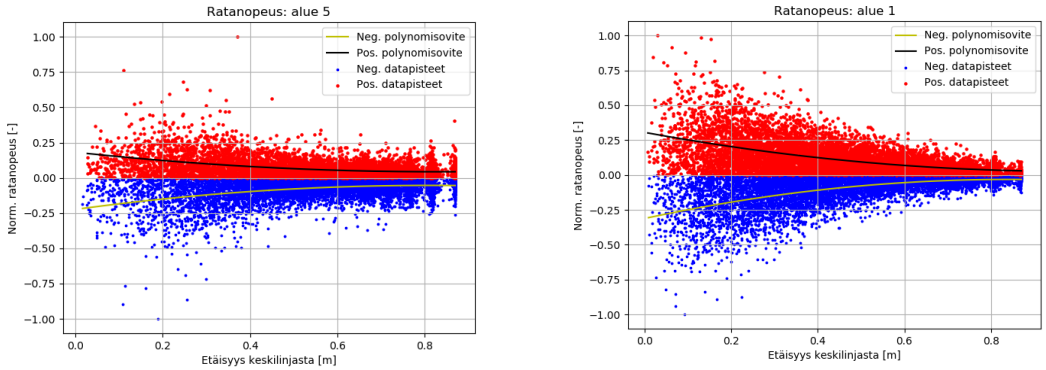


Kuva 1.2: Säteen suuntaisen nopeuden sovitteet ja datapisteet: (a) Reaktorin yläosassa. (b). Reaktorin alaosassa

Myös säteen suuntaisessa nopeudessa on enemmän vaihtelua reaktorin yläosassa. Sovite kuitenkin kuvaa kohtalaisella tarkkuudella keskinopeuden käyttäytymistä eri etäisyyksillä keskilinjasta. Sovite on jälleen tarkempi alaosassa, missä kuulien liike muutenkin on hieman säännöllisempää, tai vähemmän satunnaista.

Kuvassa 1.3 on esitetty kuulien ratanopeuden datapisteet ja sovitteet, jälleen reaktorin ylä- ja alaosassa.

(a) (b)

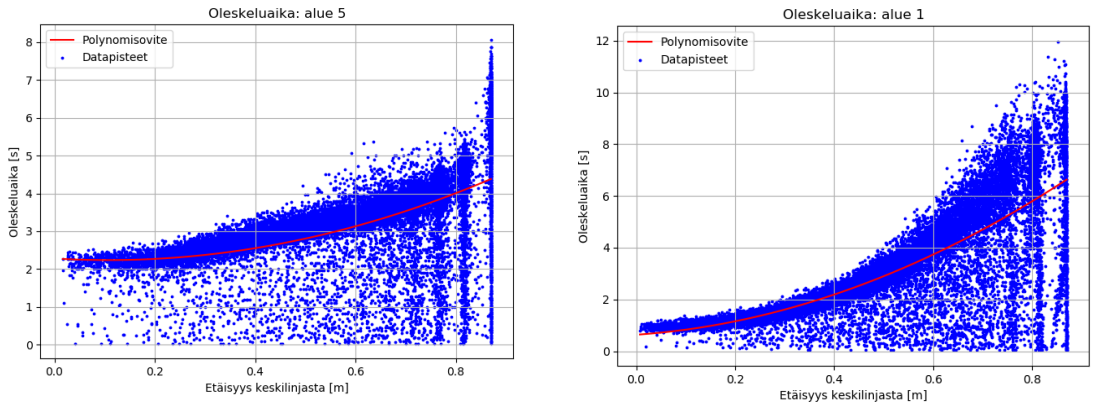


Kuva 1.3: Ratanopeuden datapisteet ja sovitteet: (a) Reaktorin yläosassa. (b) Reaktorin alaosassa.

Ratanopeuden jakauma on hyvin samanlainen niin reaktorin ylä- kuin alaosassa. Ratanopeus on myös hyvin symmetristä, jonka takia oli parempi tehdä erilliset sovitteet positiiviselle ja negatiiviselle nopeudelle. Muuten sovite olisi vain hyvin lähellä nollaa koko matkalla. Ratanopeuden datapisteissä on myös jonkin verran hajontaa, mutta sovite kuitenkin kuvaa sen keskimääräistä käyttäytymistä kohtalaisen hyvin.

Kuulien oleskeluaika on suoraan verrannollinen vertikaaliseen nopeuteen, joten sen datapisteet ja sovitteet ovat käytännössä samat kuin tässä liitteessä yläpuolella esitetyssä kuvassa 1.1. Kuvassa 1.4. on vielä oleskeluajallekin esitetty datapisteet ja sovitteet reaktorin ylä- ja alaosassa.

(a) (b)



Kuva 1.4: Kuulien oleskeluajan datapisteet ja sovitteet: (a) Reaktorin yläosassa. (b) Reaktorin alaosassa.

Kuulien oleskeluajan datapisteet ovat hieman enemmän hajaantuneet kuin vertikaalisen nopeuden datapisteet, sillä oleskeluajan tarkastelussa huomioitiin kaikki tarkkailualueella olleet kuulat. Tällöin tarkasteluun tulevat mukaan myös kuulat, jotka ovat alueella olleet lyhyenkin ajan, jolloin datapisteitä on enemmän myös lähempänä nollaa. Sovite kuitenkin seuraa aika hyvin sitä, missä suurin osa datapisteistä on, eli kuvaa keskimääräistä oleskeluaikaa jokseenkin tarkasti.

LIITE 2. TYÖSSÄ KEHITETYN PYTHON-OHJELMAN TIEDOSTO: DATALOADER.PY

```
# Tekijä: Henri Rapeli
# Päivämäärä: 16.3.2021
# Energiatekniikan kandidaatintyö
# Kuulakekoreaktorin polttoaine-elementtien virtausdatan analysointi

# Tämä tiedosto sisältää kuulien virtausdatan lataamiseen ja
# käsittelyyn käytettävän objektin ja sen funktiot

import numpy as np


class PebbleDataLoader:

    # Funktio tekstitiedostojen lukemiseen. Palauttaa sanakirjan, jossa
    # joka kuulalle on oma osio, joka sisältää numpy-arrayna
    # x-, y- ja z-koordinaatit halutulta aikaväliltä.
    @staticmethod
    def read_files(t_start, t_end, timestep, steps_per_file, n_headers):

        # Muutetaan valittu aikaväli tiedostojen numeroinnin mukaiseksi
        first_file = round((t_start / timestep), 0) * steps_per_file
        last_file = round((t_end / timestep), 0) * steps_per_file
        n_files = int((last_file - first_file) / steps_per_file)
        files_to_open = np.linspace(first_file, last_file, num=n_files + 1)

        # Luetaan tiedot koko reaktorin alueelta valitulta aikaväliltä ja
        # tallennetaan ne sanakirjaan
        pebbles = {}
        file_prefix = 'htr_10_animation_'
        for i in range(len(files_to_open)):
            filename = f'{file_prefix}{str(int(files_to_open[i]))}'
            try:
                with open(filename, 'r') as file:
                    line_count = 0
                    while True:

                        # Luetaan tiedosto rivi riviltä
                        line = file.readline()

                        # Skipataan/poistetaan otsikkorivit
                        if line_count < n_headers:
                            line_count += 1
                            del line
                            continue

                        if not line:
                            break
                        line = line.split()
                        index = int(line[0])
```

```
                    x = float(line[3])
                    y = float(line[4])
                    z = float(line[5])
                    if index not in pebbles.keys():
                        pebbles[index] = np.array([x, y, z])
                    else:
                        pebbles[index] = np.vstack((pebbles[index],
                                                    np.array([x, y, z])))

            except FileNotFoundError:
                print(f'Tiedoston {filename} avaus epäonnistui. '
                      f'Valittu aikaväli lienee epäsopiva.')
                quit(-1)

        return pebbles

    # Funktio, joka laskee eri koordinaattien perusteella joka kuulalle
    # joka ajanhetkenä etäisyyden keskilinjasta, korkeuden ja kulman
    # positiiviseen x-akseliin nähden. Palauttaa yllämainituille
    # suureille listan, jonka sisällä on joka kuulalle oma array.
    @staticmethod
    def separate_values(data_dict, bot_bound=0):
        radii = []
        heights = []
        angles = []
        for data in data_dict.values():
            radii_temp = []
            heights_temp = []
            angles_temp = []
            valid_data = data[data[:, 2] >= bot_bound]
            for x, y, z in valid_data:
                angle = None
                radius = np.sqrt(x ** 2 + y ** 2)
                if y >= 0 and x > 0:
                    angle = np.arctan(y / x)
                elif y < 0 < x:
                    angle = np.arctan(y / x) + 2 * np.pi
                elif x < 0:
                    angle = np.arctan(y / x) + np.pi

                radii_temp.append(radius)
                heights_temp.append(z)

                # Varmistetaan, että kulmallekin saatiin jokin arvo
                assert angle is not None, 'Kulma jäi ilman arvoa.'
                angles_temp.append(angle)

            radii.append(np.array(radii_temp))
            heights.append(np.array(heights_temp))
            angles.append(np.array(angles_temp))

        return radii, heights, angles
```

```
    # Funktio etsii tarkastelualueelle sopivan ylärajan sen mukaan,
    # että kuinka suuri osuus kuulista on tietyllä alueella
    # radiaalisessa suunnassa. Palauttaa kyseisen arvon.
    @staticmethod
    def find_top_boundary(r_data, h_data, bot_lim=0.9, top_lim=1.5,
                          dz=0.05, peb_lim=0.9, area=0.25):
        n_int = int((top_lim - bot_lim) / dz + 1)
        heights_top = np.linspace(bot_lim, top_lim, n_int)
        top_boundaries = []
        for i in range(1, len(heights_top)):
            pebble_amount = 0
            pebble_amount_total = 0
            for j in range(len(h_data)):
                valid_inds = np.where((heights_top[i - 1] <= h_data[j]) &
                                      (h_data[j] < heights_top[i]))
                valid_r = r_data[j][valid_inds[0]]
                if valid_r.size > 0:
                    pebble_amount_total += 1
                    if np.mean(valid_r) > area:
                        pebble_amount += 1

            top_boundaries.append(pebble_amount / pebble_amount_total)

        top_boundary = 0
        for i in range(len(top_boundaries)):
            if top_boundaries[i] < peb_lim:
                top_boundary = heights_top[0] + i * dz
                break

        return top_boundary

    # Funktio aksiaalisen nopeuden, säteen suuntaisen nopeuden
    # ja ratanopeuden laskemiseen. Tarkasteluun otetaan mukaan kuulat,
    # jotka yhtäjaksoisesti kulkivat koko aksiaalisen tarkkailualueen läpi.
    # Palauttaa listan, jossa on joka aks. alueella joka kuulalle
    # keskimääräinen etäisyys ja keskimääräiset nopeudet.
    @staticmethod
    def calc_velocities(r_data, h_data, a_data, bot_bound, top_bound,
                        dt, margin=0, ax_split=5):

        diff_limit = 0.1

        def ax_and_rad_vel(data, delta_t):
            vel = np.array([(data[k] - data[k - 1]) / delta_t
                            for k in range(1, len(data))])
            return vel

        def circ_vel(rads, angles, delta_t):
            vel = []
            for k in range(1, len(angles)):
                a_0 = angles[k - 1]
                a_1 = angles[k]
                r = rads[k]
```

```
            if abs(a_1 - a_0) > np.pi and a_1 > a_0:
                vel.append(((a_1 - 2 * np.pi - a_0) / delta_t) * r)

            elif abs(a_1 - a_0) > np.pi and a_1 < a_0:
                vel.append(((a_1 + 2 * np.pi - a_0) / delta_t) * r)

            else:
                vel.append(((a_1 - a_0) / delta_t) * r)

        return np.array(vel)

    heights = np.linspace(bot_bound, top_bound, ax_split + 1)
    vels_avg = []
    for i in range(1, len(heights)):
        vels_avg_temp = [[], [], [], []]
        for j in range(len(h_data)):
            valid_inds = np.where((heights[i - 1] <= h_data[j]) &
                                  (h_data[j] < heights[i]))
            valid_z = h_data[j][valid_inds[0]]
            valid_r = r_data[j][valid_inds[0]]
            valid_a = a_data[j][valid_inds[0]]
            if valid_z.size > 1:
                max_ind = np.where(valid_z == max(valid_z))[0][0]
                end = None
                z_from_max = valid_z[max_ind:]
                r_from_max = valid_r[max_ind:]
                a_from_max = valid_a[max_ind:]
                z_diff = np.diff(z_from_max)
                try:
                    # Kymmentä senttiä suurempi peräkkäisten
                    # korkeusarvojen ero viittaa siihen, että kuula on
                    # tullut alueelle uudemman kerran
                    diff_ind = np.where(z_diff > diff_limit)[0][0]
                    end = diff_ind + 1
                except IndexError:
                    pass

                z_split = z_from_max[:end]
                r_split = r_from_max[:end]
                a_split = a_from_max[:end]
                dist = max(z_split) - min(z_split)

                # Jos kuula on kulkenut reaktorissa tietyn matkan
                # (aks. tarkkailualueen korkeus miinus marginaali),
                # niin kuulan lasketaan kulkeneen koko alueen läpi.
                # Marginaali == kuulan halkaisija esimerkiksi.
                h_diff = heights[i] - heights[i - 1] - margin
                if dist >= h_diff:
                    vels_avg_temp[0].append(np.mean(r_split[1:]))

                    vz = ax_and_rad_vel(z_split, dt)
                    vels_avg_temp[1].append(np.mean(vz))
```

```
                    vrad = ax_and_rad_vel(r_split, dt)
                    vels_avg_temp[2].append(np.mean(vrad))

                    vcirc = circ_vel(r_split, a_split, dt)
                    vels_avg_temp[3].append(np.mean(vcirc))

            vels_avg.append(np.array(vels_avg_temp))

        return vels_avg

    # Funktio oleskeluajan laskemiseen aksiaalisilla tarkkailualuiella.
    # Tarkastelussa huomioidaan se, että kuulat ovat voineet olla
    # samalla alueella kaksi kertaa siten, että toista kertaa ei lasketa,
    # jotta ajasta ei tule virheellisen suuri.
    # Palauttaa listan, jossa on joka aks. alueella joka kuulalle
    # keskimääräinen etäisyys ja oleskeluaika.
    @staticmethod
    def calc_times(r_data, h_data, bot_bound, top_bound, dt,
                   ax_split=5):

        diff_limit = 0.1
        heights = np.linspace(bot_bound, top_bound, ax_split + 1)
        times_avg = []
        for i in range(1, len(heights)):
            times_temp = [[], []]
            for j in range(len(h_data)):
                valid_inds = np.where((heights[i - 1] <= h_data[j]) &
                                      (h_data[j] < heights[i]))
                valid_r = r_data[j][valid_inds[0]]
                valid_z = h_data[j][valid_inds[0]]
                if valid_z.size > 0:
                    max_ind = np.where(valid_z == max(valid_z))[0][0]
                    end = None
                    r_from_max = valid_r[max_ind:]
                    z_from_max = valid_z[max_ind:]
                    z_diff = np.diff(z_from_max)
                    try:
                        diff_ind = np.where(z_diff > diff_limit)[0][0]
                        end = diff_ind + 1
                    except IndexError:
                        pass

                    r_split = r_from_max[:end]
                    times_temp[0].append(np.mean(r_split))
                    times_temp[1].append(r_split.size * dt)

            if times_temp[0]:
                times_avg.append(np.array(times_temp))

        return times_avg

    # Funktio, jolla voidaan järjestää nopeus- ja oleskeluaikojen
    # arrayt ensimmäisen arrayn (etäisyys) arvojen mukaan sovitteen
```

```
    # muodostamista varten.
    # Palauttaa listan, jossa arrayt on uudessa järjestyksessä.
    @staticmethod
    def sort_by_x(data):
        sort_arrlist = []
        for i in range(len(data)):
            arr_sorted = data[i][:, data[i][0, :].argsort()]
            sort_arrlist.append(arr_sorted)

        return sort_arrlist

    # Funktio nopeuksien normalisoimiseksi. Palauttaa arrayn
    # normalisoiduista nopeuksista. Jos neg. ja pos. nopeudet
    # erotellaan, palauttaa kaksi listaa, joissa omat arrayt etäisyyden
    # ja nopeuden arvoille.
    @staticmethod
    def norm_vels(v_data, split_neg_pos=False, r_data=None):
        if not split_neg_pos:

            # Varmistetaan, että normalisointi tapahtuu itseis-
            # arvoltaan suurimmalla nopeudella
            max_vel = min(v_data)
            if abs(min(v_data)) < abs(max(v_data)):
                max_vel = max(v_data)
                norm = np.array([v / max_vel for v in v_data])
            else:
                norm = np.array([v / max_vel * -1 for v in v_data])

            return norm

        else:

            # Tarkistetaan, että etäisyyksistäkin on dataa, jotta
            # neg. ja pos. nopeudet voidaan yhdistää oikeille
            # x-akselin arvoille
            assert r_data is not None, 'Etäisyysdata puuttuu.'

            # Eritellään neg. ja pos. nopeudet
            v_neg_temp = [[], []]
            v_pos_temp = [[], []]
            for a in range(len(v_data)):
                r = r_data[a]
                v = v_data[a]
                if v >= 0:
                    v_pos_temp[0].append(r)
                    v_pos_temp[1].append(v)
                else:
                    v_neg_temp[0].append(r)
                    v_neg_temp[1].append(v)

            max_neg = min(v_neg_temp[1])
            max_pos = max(v_pos_temp[1])
```

```
            norm_neg = np.array([v / max_neg * -1 for v in v_neg_temp[1]])
            norm_pos = np.array([v / max_pos for v in v_pos_temp[1]])
            neg = [np.array(v_neg_temp[0]), norm_neg]
            pos = [np.array(v_pos_temp[0]), norm_pos]

            return neg, pos

    # Funktio radiaalisen liikkeen suuruuden laskemiseksi. Tarkastelussa
    # mukana kuulat, jotka kulkivat yhteen putkeen koko reaktorin läpi.
    # Palauttaa listan, jossa on joka kuulalle array, joka sisältää
    # etäisyyden muutokset referenssiarvoon (arvo aivan reaktorin
    # yläosassa) nähden.
    @staticmethod
    def calc_radial_movement(r_data, h_data, bot_bound, top_bound,
                             h_int, margin=0, scale_results=True,
scale_factor=1):

        diff_limit = 0.1
        pebbles_fullpass = []
        for i in range(len(h_data)):
            valid_inds = np.where((bot_bound <= h_data[i]) &
                                  (h_data[i] < top_bound))
            valid_z = h_data[i][valid_inds[0]]
            valid_r = r_data[i][valid_inds[0]]
            if valid_z.size > 1:
                max_ind = np.where(valid_z == max(valid_z))[0][0]
                end = None
                z_from_max = valid_z[max_ind:]
                r_from_max = valid_r[max_ind:]
                z_diff = np.diff(z_from_max)
                try:
                    diff_ind = np.where(z_diff > diff_limit)[0][0]
                    end = diff_ind + 1
                except IndexError:
                    pass

                z_split = z_from_max[:end]
                r_split = r_from_max[:end]
                dist = max(z_split) - min(z_split)
                h_diff = top_bound - bot_bound - margin
                if dist >= h_diff:
                    pebbles_fullpass.append([r_split, z_split])

        # Tehdään etäisyyden muutoksen tarkastelu, jos kokonaan
        # reaktorin läpi menneitä kuulia löytyy
        if pebbles_fullpass:

            # Tarkastellaan säteen muutoksia niiden kulkiessa alaspäin
            # reaktorissa
            # Tulokset on skaalattu kuulien halkaisijalla
            delta_rad = []
            n_int = int((top_bound - bot_bound) / h_int)
            heights_deltarad = np.linspace(top_bound, bot_bound,
```

```
                                        n_int + 1)
            for i in range(len(pebbles_fullpass)):
                ref_rad = pebbles_fullpass[i][0][0]
                delta_rad_temp = [ref_rad, 0]
                height_matches = []
                inds = []
                for j in heights_deltarad:
                    height_matches.append(min(pebbles_fullpass[i][1],
                                              key=lambda h: abs(h - j)))
                for k in height_matches:
                    inds.append(np.where(pebbles_fullpass[i][1] == k)[0][0])
                for n in inds[1:]:
                    rad = pebbles_fullpass[i][0][n]
                    if scale_results:
                        delta_rad_temp.append((rad - ref_rad) / scale_factor)
                    else:
                        delta_rad_temp.append((rad - ref_rad))
                delta_rad.append(np.array(delta_rad_temp))

            return delta_rad

        else:
            print('Kokonaan reaktorin läpi menneitä kuulia ei ole.')
            return None

    # Lasketaan kuulien keskimääräinen aksiaalinen nopeus ja oleskeluaika
    # reaktorin taitteessa. Lisäksi tutkitaan sitä, että onko aksiaalinen
    # nopeus nolla tai positiivinen. Palauttaa kolme listaa. Yksi sisältää
    # joka kuulan oleskeluajan, toinen joka kuulalle keskimääräisen
    # aks. nopeuden. Kolmas sisältää niille kuulille, joilla oli pos. aks.
    # nopeus, keskimääräiset x- ja y-koordinaatit ja pos. nopeuden keston.
    @staticmethod
    def vel_and_time_turn(peb_data, bot_lim, top_lim, dt):
        pebble_diam = 0.06

        def ax_vel(z_data, delta_t):
            vz = np.array([(z_data[k] - z_data[k - 1]) / delta_t
                           for k in range(1, len(z_data))])
            return vz

        times_turn = []
        vel_turn = []
        vel_pos_turn = [[], [], []]
        for data in peb_data.values():
            valid_inds = np.where((bot_lim <= data[:, 2]) &
                                  (data[:, 2] <= top_lim))
            valid_x = data[valid_inds[0], 0]
            valid_y = data[valid_inds[0], 1]
            valid_z = data[valid_inds[0], 2]
            if valid_x.size > 0:
                max_ind = np.where(valid_z == max(valid_z))[0][0]
                end = None
                x_from_max = valid_x[max_ind:]
```

```
            y_from_max = valid_y[max_ind:]
            z_from_max = valid_z[max_ind:]
            z_diff = np.diff(z_from_max)
            try:
                diff_ind = np.where(z_diff > pebble_diam)[0][0]
                end = diff_ind + 1
            except IndexError:
                pass

            x_split = x_from_max[:end]
            y_split = y_from_max[:end]
            z_split = z_from_max[:end]
            if z_split.size > 1:
                times_turn.append(z_split.size * dt)
                vel = ax_vel(z_split, dt)
                vel_turn.append(np.mean(vel))
                pos_ind = np.where(vel >= 0)[0]
                pos_x = x_split[pos_ind]
                pos_y = y_split[pos_ind]
                if pos_x.size > 0:
                    vel_pos_turn[0].append(np.mean(pos_x))
                    vel_pos_turn[1].append(np.mean(pos_y))
                    vel_pos_turn[2].append(pos_x.size * dt)

    return times_turn, vel_turn, vel_pos_turn
```

LIITE 3. TYÖSSÄ KEHITETYN PYTHON-OHJELMAN TIEDOSTO: PLOT_FUNCS.PY

```python
# Tekijä: Henri Rapeli
# Päivämäärä: 16.3.2021
# Energiatekniikan kandidaatintyö
# Kuulakekoreaktorin polttoaine-elementtien virtausdatan analysointi

# Tämä tiedosto sisältää virtausdatan analysoinnin tulosten esittämisessä
# hyödyllisiä funktioita

import matplotlib.pyplot as plt
from scipy.optimize import curve_fit
import numpy as np


# Funktio sovitteiden piirtoon
def fit_plot(xdata, ydata, fit_func, **kwargs):
    graphs = []

    # Datan käsittely ja sovitteen määritys
    for i in range(len(xdata)):
        popt, _ = curve_fit(fit_func, xdata[i], ydata[i])
        ydata_opt = fit_func(xdata[i], *popt)

        # Määritetään muut kuvaajan parametrit
        lab, color = None, None
        if 'label' in kwargs.keys():
            if len(xdata) > 1:
                prefix = str(kwargs['label'])
                lab = f'{prefix} {i + 1}'
            else:
                lab = str(kwargs['label'])
        if 'color' in kwargs.keys():
            color = str(kwargs['color'][i])

        graphs.append(plt.plot(xdata[i], ydata_opt, label=lab, color=color))

        if 'xlabel' in kwargs.keys():
            plt.xlabel(str(kwargs['xlabel']))
        if 'ylabel' in kwargs.keys():
            plt.ylabel(str(kwargs['ylabel']))
        if 'legend' in kwargs.keys() and kwargs['legend'] is True:
            plt.legend()
        if 'title' in kwargs.keys():
            plt.title(str(kwargs['title']))
        if 'grid' in kwargs.keys() and kwargs['grid'] is True:
            plt.grid()
        if 'xticks' in kwargs.keys():
            plt.xticks(kwargs['xticks'])

    return graphs
```

```python
# Funktio datapisteiden piirtoon
def scatter_plot(x_data, y_data, **kwargs):
    lab, c, cmap = None, None, None
    ms = 5
    if 'label' in kwargs.keys():
        lab = str(kwargs['label'])
    if 'c' in kwargs.keys():
        c = kwargs['c']
    if 'ms' in kwargs.keys():
        ms = int(kwargs['ms'])
    if 'cmap' in kwargs.keys():
        cmap = str(kwargs['cmap'])

    sc = plt.scatter(x_data, y_data, label=lab, s=ms, c=c,
                     cmap=cmap)

    # Tsekataan muut parametrit
    if 'xlabel' in kwargs.keys():
        plt.xlabel(str(kwargs['xlabel']))
    if 'ylabel' in kwargs.keys():
        plt.ylabel(str(kwargs['ylabel']))
    if 'legend' in kwargs.keys() and kwargs['legend'] is True:
        plt.legend()
    if 'title' in kwargs.keys():
        plt.title(str(kwargs['title']))
    if 'grid' in kwargs.keys() and kwargs['grid'] is True:
        plt.grid()
    if 'xticks' in kwargs.keys():
        plt.xticks(kwargs['xticks'])
    if cmap is not None:
        cbar = plt.colorbar()
        cbar.set_label(str(kwargs['cbar_label']))

    return sc


# Suhteellisten jakaumien piirto
def relative_plot(data, n_int, scale_results=True, scale_factor=1,
                  **kwargs):
    graphs = []

    # Skaalataan tuloksia haluttaessa
    if scale_results:
        factor = scale_factor
    else:
        factor = 1

    # Muodostetaan suhteellinen jakauma
    for i in range(len(data)):
        interval = np.linspace(min(data[i]) / factor,
                               max(data[i]) / factor, n_int + 1)
```

```
        fractions = []
        for j in range(1, len(interval)):
            counter = 0
            for k in range(len(data[i])):
                val = data[i][k]
                if interval[j - 1] < (val / factor) <= interval[j]:
                    counter += 1

            fractions.append(counter / len(data[i]))

        # Tarkistetaan piirtoparametrit
        lab, color = None, None
        if 'label' in kwargs.keys():
            prefix = str(kwargs['label'])
            lab = f'{prefix} {i + 1}'
        if 'color' in kwargs.keys():
            color = str(kwargs['color'][i])

        graphs.append(plt.plot(interval[1::], fractions, label=lab,
                               color=color))

        if 'xlabel' in kwargs.keys():
            plt.xlabel(str(kwargs['xlabel']))
        if 'ylabel' in kwargs.keys():
            plt.ylabel(str(kwargs['ylabel']))
        if 'legend' in kwargs.keys() and kwargs['legend'] is True:
            plt.legend()
        if 'title' in kwargs.keys():
            plt.title(str(kwargs['title']))
        if 'grid' in kwargs.keys() and kwargs['grid'] is True:
            plt.grid()
```

LIITE 4. TYÖSSÄ KEHITETYN PYTHON-OHJELMAN TIEDOSTO: MAIN.PY

```
# Tekijä: Henri Rapeli
# Päivämäärä: 16.3.2021
# Energiatekniikan kandidaatintyö
# Kuulakekoreaktorin polttoaine-elementtien virtausdatan analysointi

# Tämä tiedosto sisältää virtausdatan käsittelyyn käytettävän objektin
# käytön ja tulosten esittämisen

from dataloader import PebbleDataLoader
from plot_funcs import fit_plot, relative_plot, scatter_plot
import matplotlib.pyplot as plt
import numpy as np

# Tiedostoihin liittyviä parametrejä
file_timestep = 0.03333
timesteps_per_file = 10000
header_lines = 9

# Kuulan halkaisija mm. tulosten skaalaamista varten
pebble_diam = 0.06

# Valitaan aikaväli (0 ... 22,3 s)
start_time, end_time = 0, 22.3

# Alustetaan/luodaan PebbleDataLoader-objekti
data_loader = PebbleDataLoader()

# Luetaan tiedot halutulta aikaväliltä
pebbles = data_loader.read_files(start_time, end_time, file_timestep,
                                 timesteps_per_file, header_lines)

# Asetetaan tarkastelun alarajaksi 0 m
bottom_boundary = 0

# Eritellään kuulien säde (etäisyys keskilinjasta), korkeus ja
# kulma pos. x-akseliin nähden omiksi dataseteiksi
radius, height, angle = data_loader.separate_values(pebbles,
                                                    bottom_boundary)

# Tehdään sanakirja, johon laskettuja tietoja voidaan tallentaa
pebbledata = {'radii': radius, 'heights': height, 'angles': angle}

# Etsitään sopiva yläraja sen mukaan kuinka suuri osuus kuulista
# on reaktorin ulkoreunalla ja lähellä keskiosaa
bot_lim, top_lim, interval = 0.9, 1.5, 0.05
pebble_limit = 0.9
area = 0.25
top_boundary = data_loader.find_top_boundary(pebbledata['radii'],
                                             pebbledata['heights'],
```

```
                                            bot_lim, top_lim, interval,
                                            pebble_limit, area)
print(f'Tarkastelualueen yläraja: {top_boundary}')

# Valitaan, kuinka moneen aksiaaliseen alueeseen reaktori jaetaan
ax_split = 5

# Lasketaan nopeudet aksiaalisilla alueilla
vels_avg = data_loader.calc_velocities(pebbledata['radii'],
                                       pebbledata['heights'],
                                       pebbledata['angles'],
                                       bottom_boundary, top_boundary,
                                       file_timestep, margin=pebble_diam,
                                       ax_split=ax_split)

# Lasketaan oleskeluajat aksiaalisilla alueilla
times_avg = data_loader.calc_times(pebbledata['radii'],
                                   pebbledata['heights'], bottom_boundary,
                                   top_boundary, file_timestep,
                                   ax_split=ax_split)

# Järjestellään nopeus-arrayt x-arvojen (etäisyyden mukaan)
vels_sort = data_loader.sort_by_x(vels_avg)

# Sama järjestely oleskeluajoille
times_sort = data_loader.sort_by_x(times_avg)

# Eritellään nopeudet (ja etäisyys) omiksi osioiksi sanakirjaan
pebbledata['rad_avg'] = [vels_lst[0]
                         for vels_lst in vels_sort]
pebbledata['vz_avg'] = [vels_lst[1]
                        for vels_lst in vels_sort]
pebbledata['vrad_avg'] = [vels_lst[2]
                          for vels_lst in vels_sort]
pebbledata['vcirc_avg'] = [vels_lst[3]
                           for vels_lst in vels_sort]

# Eritellään niiden kuulien, jotka ovat aikatarkastelussa mukana,
# etäisyyksille ja ajoille omat osionsa sanakirjaan
pebbledata['times_rad'] = [np.array(times_lst[0])
                           for times_lst in times_sort]
pebbledata['times'] = [np.array(times_lst[1])
                       for times_lst in times_sort]

# Normalisoidaan eri nopeudet
pebbledata['vz_norm'] = []
pebbledata['vrad_norm'] = []
pebbledata['vcirc_neg'] = []
pebbledata['rad_avg_neg'] = []
pebbledata['vcirc_pos'] = []
pebbledata['rad_avg_pos'] = []
for i in range(len(pebbledata['vz_avg'])):
    pebbledata['vz_norm'].append(data_loader.norm_vels(
```

```
        pebbledata['vz_avg'][i]))

    pebbledata['vrad_norm'].append(data_loader.norm_vels(
        pebbledata['vrad_avg'][i]))

    varc_data = data_loader.norm_vels(pebbledata['vcirc_avg'][i],
                                      split_neg_pos=True,
                                      r_data=pebbledata['rad_avg'][i])

    pebbledata['rad_avg_neg'].append(varc_data[0][0])
    pebbledata['rad_avg_pos'].append(varc_data[1][0])
    pebbledata['vcirc_neg'].append(varc_data[0][1])
    pebbledata['vcirc_pos'].append(varc_data[1][1])

# Lasketaan kuulien radiaalisen liikkeen suuruus
# Mukana tarkastelussa on vain sellaiset kuulat, jotka kulkivat
# koko reaktorin läpi yhteen putkeen
height_interval = 0.025
delta_rad = data_loader.calc_radial_movement(pebbledata['radii'],
                                             pebbledata['heights'],
                                             bottom_boundary,
                                             top_boundary, height_interval,
                                             margin=pebble_diam,
                                             scale_results=True,
                                             scale_factor=pebble_diam)

# Jaotellaan kuulia muutamalle radiaaliselle alueelle
# etäisyyden muutoksien kuvaajien piirtoa varten
if delta_rad is not None:

    # Lasketaan kuinka monta kuulaa per alue siten,
    # että kuulat jakautuvat tasan
    n_areas = 3
    n_pebs = len(delta_rad)
    print(f'\nKokonaan reaktorin läpi menneiden kuulien lkm.: '
          f'{n_pebs}')
    mod = n_pebs % n_areas
    if mod == 0:
        pebs_per_area = [n_pebs // n_areas for _ in range(n_areas)]
    else:
        pebs_per_area = [n_pebs // n_areas for _ in range(n_areas - mod)]
        pebs_per_area += [n_pebs // n_areas + 1 for _ in range(mod)]
    inds = np.cumsum(pebs_per_area)

    # Etsitään sellaiset rajat alueille, että yllä lasketut kuulien
    # alueelliset lukumäärät toteutuvat
    delta_rad_sort = sorted(delta_rad, key=lambda r: r[0])
    delta_rad_split = [np.array(delta_rad_sort[0:inds[i]]) if i == 0
                       else np.array(delta_rad_sort[inds[i - 1]:inds[i]])
                       for i in range(len(inds))]

    bounds = [0]
    bounds += [round(sublist[0][0], 2) for sublist in delta_rad_split[1:]]
```

```
    bounds += [round(delta_rad_split[-1][-1][0], 2)]

    for i in range(len(delta_rad_split)):
        rad_area = f'{bounds[i]} < r < {bounds[i + 1]} m'
        print(f"Kuulien lkm. alueella {rad_area}: "
              f"{len(delta_rad_split[i])}")

    # Lasketaan keskimääräinen etäisyyden muutos eri alueilla
    delta_rad_avg = [
        np.array([np.mean(delta_rad_split[i][:, j])
                  for j in range(1, len(delta_rad_split[i][0]))])
        for i in range(len(delta_rad_split))
    ]
    pebbledata['d_rad_avg'] = delta_rad_avg

    # Etäisyyden muutoksen kuvaajat
    plt.figure()
    n_int = int((top_boundary - bottom_boundary) / height_interval)
    heights_deltarad = np.linspace(top_boundary, bottom_boundary,
                                   n_int + 1)
    for i in range(len(pebbledata['d_rad_avg'])):
        lab = f'{bounds[i]} < r < {bounds[i + 1]} m'
        plt.plot(pebbledata['d_rad_avg'][i], heights_deltarad,
                 label=lab)

    plt.xlabel('Etäisyyden muutos [d]')
    plt.ylabel('Korkeus [m]')
    plt.grid()
    plt.legend()

# Tarkastellaan kuulien aksiaalista nopeutta ja oleskeluaikaa
# reaktorin alaosan taitteessa (taite alkaa kohdassa z = 0 m)
bottom = -0.1
top = 0.1
times_turn, vel_turn, vel_pos_turn = data_loader.vel_and_time_turn(
    pebbles, bottom, top, file_timestep
)
pebbledata['times_turn'] = np.array(times_turn)
pebbledata['vel_turn'] = np.array(vel_turn)
pebbledata['vel_pos_turn'] = np.array(vel_pos_turn)

# Tulostetaan joitain mahdollisesti mielenkiintoisia tietoja
print()
for i in range(len(pebbledata['rad_avg'])):
    print(f"Kuulien lkm. (nopeus), Alue {i + 1}:"
          f" {len(pebbledata['rad_avg'][i])}")

print()
for i in range(len(pebbledata['times_rad'])):
    print(f"Kuulien lkm. (aika), Alue {i + 1}:"
          f" {len(pebbledata['times_rad'][i])}")

print()
```

```
print(f"Kuulien lkm. taite: {len(pebbledata['times_turn'])}")
print(f"Pos. nop. lkm. taite: {len(pebbledata['vel_pos_turn'][0])}")


# Määritellään toisen asteen polynomin funktio, jonka voi curve_fitin
# avulla sovittaa datapisteisiin
# Tätä varten kaikki arrayt piti järjestää x-akselin arvojen mukaan
def fun_polynom(x_val, a, b, c):
    return a * np.square(x_val) + b * x_val + c


# Norm. aksiaalisen nopeuden kuvaajat
plt.figure()
graphs_vz = fit_plot(pebbledata['rad_avg'], pebbledata['vz_norm'],
                     fun_polynom, xlabel='Etäisyys keskilinjasta [m]',
                     ylabel='Norm. aksiaalinen nopeus [-]',
                     legend=True, label='Alue ', grid=True)

# Määritellään kuinka monella alueella (binillä) suhteellinen
# jakauma muodostetaan
# Vaikuttaa lähinnä siihen, minkänäköinen käyrästä tulee, ei siis
# itse jakaumaan tai sen muotoon
n_bins_vel = 100

# Aksiaalisen nopeuden suhteellinen jakauma
plt.figure()
graphs_rel_vz = relative_plot(
    pebbledata['vz_avg'], n_bins_vel,
    scale_results=True, scale_factor=pebble_diam,
    xlabel='Aksiaalinen nopeus [d/s]',
    ylabel='Nopeuden suhteellinen esiintyvyys [-]',
    label='Alue ', grid=True, legend=True
)

# Norm. säteen suuntainen nopeus eri alueilla
plt.figure()
graphs_vrad = fit_plot(pebbledata['rad_avg'], pebbledata['vrad_norm'],
                       fun_polynom, xlabel='Etäisyys keskilinjasta [m]',
                       ylabel='Norm. säteen suuntainen nopeus [-]',
                       legend=True, label='Alue ', grid=True)

# Säteen suuntaisen nopeuden suhteellinen jakauma
plt.figure()
graphs_rel_vrad = relative_plot(
    pebbledata['vrad_avg'], n_bins_vel,
    scale_results=True, scale_factor=pebble_diam,
    xlabel='Säteen suuntainen nopeus [d/s]',
    ylabel='Nopeuden suhteellinen esiintyvyys [-]',
    label='Alue ', grid=True, legend=True
)

# Norm. ratanopeus eri alueilla
plt.figure()
```

```
colors = ['b', 'k', 'g', 'r', 'm']
graphs_vcirc = [
    fit_plot(pebbledata['rad_avg_pos'], pebbledata['vcirc_pos'],
             fun_polynom, xlabel='Etäisyys keskilinjasta [m]',
             ylabel='Norm. ratanopeus [-]',
             legend=True, label='Alue ', grid=True,
             color=colors),

    fit_plot(pebbledata['rad_avg_neg'], pebbledata['vcirc_neg'],
             fun_polynom, xlabel='Etäisyys keskilinjasta [m]',
             ylabel='Norm. ratanopeus [-]',
             legend=True, color=colors)
]

# Ratanopeuden suhteellinen jakauma
plt.figure()
graphs_rel_vcirc = relative_plot(
    pebbledata['vcirc_avg'], n_bins_vel,
    scale_results=True, scale_factor=pebble_diam,
    xlabel='Ratanopeus [d/s]',
    ylabel='Nopeuden suhteellinen esiintyvyys [-]',
    label='Alue ', grid=True, legend=True
)

# Oleskeluajan kuvaaja
plt.figure()
graphs_time = fit_plot(pebbledata['times_rad'], pebbledata['times'],
                       fun_polynom, xlabel='Etäisyys keskilinjasta [m]',
                       ylabel='Oleskeluaika [s]',
                       legend=True, label='Alue ', grid=True)

# Ajan suht. jakaumille 50 aluetta/biniä tuntuu olevan
# parempi kuin 100
n_bins_time = 50

# Sama taitteen suht. jakaumille
n_bins_turn = 50

# Oleskeluajan suhteellinen jakauma
plt.figure()
graphs_rel_time = relative_plot(pebbledata['times'], n_bins_time,
                                scale_results=False,
                                xlabel='Oleskeluaika [s]',
                                ylabel='Ajan suhteellinen esiintyvyys [-]',
                                is_time=True, label='Alue ', grid=True,
                                legend=True)

# Aks. nopeuden suhteellinen jakauma taitteessa
plt.figure()
graphs_vz_turn = relative_plot(
    [pebbledata['vel_turn']], n_bins_turn,
    scale_results=True, scale_factor=pebble_diam,
    xlabel='Aksiaalinen nopeus [d/s]',
```

```
    ylabel='Nopeuden suhteellinen esiintyvyys [-]',
    grid=True, color='b'
)

# Oleskeluajan suhteellinen jakauma taitteessa
plt.figure()
graphs_time_turn = relative_plot([pebbledata['times_turn']], n_bins_turn,
                                 scale_results=False,
                                 xlabel='Oleskeluaika [s]',
                                 ylabel='Ajan suhteellinen esiintyvyys [-]',
                                 is_time=True, grid=True, color='b')

# Kuulat, joilla on ollut positiivinen aksiaalinen nopeus taitteessa
plt.figure()
col = pebbledata['vel_pos_turn'][2]
pos_vels_turn = scatter_plot(pebbledata['vel_pos_turn'][0],
                             pebbledata['vel_pos_turn'][1],
                             ms=7, c=col, cmap='jet',
                             cbar_label='Positiivisen nopeuden kesto [s]',
                             xlabel='x [m]', ylabel='y [m]', grid=True)
circle = plt.Circle((0, 0), 0.9, color='k', fill=False)
plt.gca().add_patch(circle)

# Sivukuvat simulaation eri ajan hetkiltä lähinnä kuulakeon
# yläosan havainnoimiseksi
s = int(round((start_time / file_timestep), 0))
e = int(round((end_time / file_timestep), 0))
n_pics = 3
time_points = np.linspace(s, e, n_pics)
for i in time_points:
    x = []
    z = []
    plt.figure()
    for data in pebbles.values():
        x.append(data[int(i)][0])
        z.append(data[int(i)][2])

    plt.scatter(x, z, s=1, c='b')
    plt.xlabel('x [m]')
    plt.ylabel('z [m]')
    plt.grid()
    plt.gca().set_aspect(0.6)

# Aks. nopeuden datapisteet ja sovite reaktorin alaosassa
plt.figure()
fit_data_vz_bottom = [
    scatter_plot(pebbledata['rad_avg'][0], pebbledata['vz_norm'][0],
                 label='Datapisteet', c='b', ms=3),

    fit_plot([pebbledata['rad_avg'][0]], [pebbledata['vz_norm'][0]],
             fun_polynom, label='Polynomisovite', color='r',
             xlabel='Etäisyys keskilinjasta [m]',
             ylabel='Norm. aksiaalinen nopeus [-]',
```

```
            legend=True, grid=True,
            title='Aksiaalinen nopeus: alue 1')
]

# Aks. nopeuden datapisteet ja sovite reaktorin yläosassa
plt.figure()
fit_data_vz_top = [
    scatter_plot(pebbledata['rad_avg'][-1], pebbledata['vz_norm'][-1],
                 label='Datapisteet', c='b', ms=3),

    fit_plot([pebbledata['rad_avg'][-1]], [pebbledata['vz_norm'][-1]],
             fun_polynom, label='Polynomisovite', color='r',
             xlabel='Etäisyys keskilinjasta [m]',
             ylabel='Norm. aksiaalinen nopeus [-]',
             legend=True, grid=True,
             title=f'Aksiaalinen nopeus: alue {ax_split}')
]

# Säteen suuntaisen nopeuden datapisteet ja sovite reaktorin alaosassa
plt.figure()
fit_data_vrad_bottom = [
    scatter_plot(pebbledata['rad_avg'][0], pebbledata['vrad_norm'][0],
                 label='Datapisteet', c='b', ms=3),

    fit_plot([pebbledata['rad_avg'][0]], [pebbledata['vrad_norm'][0]],
             fun_polynom, label='Polynomisovite', color='r',
             xlabel='Etäisyys keskilinjasta [m]',
             ylabel='Norm. säteen suuntainen nopeus [-]',
             legend=True, grid=True,
             title='Säteen suuntainen nopeus: alue 1')
]

# Säteen suuntaisen nopeuden datapisteet ja sovite reaktorin yläosassa
plt.figure()
fit_data_vrad_top = [
    scatter_plot([pebbledata['rad_avg'][-1]], [pebbledata['vrad_norm'][-1]],
                 label='Datapisteet', c='b', ms=3),

    fit_plot([pebbledata['rad_avg'][-1]], [pebbledata['vrad_norm'][-1]],
             fun_polynom, label='Polynomisovite', color='r',
             xlabel='Etäisyys keskilinjasta [m]',
             ylabel='Norm. säteen suuntainen nopeus [-]',
             legend=True, grid=True,
             title=f'Säteen suuntainen nopeus: alue {ax_split}')
]

# Ratanopeuden datapisteet ja sovite reaktorin alaosassa
plt.figure()
fit_data_varc_bottom = [
    scatter_plot(pebbledata['rad_avg_neg'][0], pebbledata['vcirc_neg'][0],
                 label='Neg. datapisteet', c='b', ms=3),

    scatter_plot(pebbledata['rad_avg_pos'][0], pebbledata['vcirc_pos'][0],
```

```
                label='Pos. datapisteet', c='r', ms=5),

    fit_plot([pebbledata['rad_avg_neg'][0]], [pebbledata['vcirc_neg'][0]],
             fun_polynom, label='Neg. polynomisovite', color='y'),

    fit_plot([pebbledata['rad_avg_pos'][0]], [pebbledata['vcirc_pos'][0]],
             fun_polynom, label='Pos. polynomisovite', color='k',
             xlabel='Etäisyys keskilinjasta [m]',
             ylabel='Norm. ratanopeus [-]',
             legend=True, grid=True,
             title='Ratanopeus: alue 1')
]

# Ratanopeuden datapisteet ja sovite reaktorin yläosassa
plt.figure()
fit_data_varc_top = [
    scatter_plot(pebbledata['rad_avg_neg'][-1], pebbledata['vcirc_neg'][-1],
                 label='Neg. datapisteet', c='b', ms=3),

    scatter_plot(pebbledata['rad_avg_pos'][-1], pebbledata['vcirc_pos'][-1],
                 label='Pos. datapisteet', c='r', ms=5),

    fit_plot([pebbledata['rad_avg_neg'][-1]], [pebbledata['vcirc_neg'][-1]],
             fun_polynom, label='Neg. polynomisovite', color='y'),

    fit_plot([pebbledata['rad_avg_pos'][-1]], [pebbledata['vcirc_pos'][-1]],
             fun_polynom, label='Pos. polynomisovite', color='k',
             xlabel='Etäisyys keskilinjasta [m]',
             ylabel='Norm. ratanopeus [-]',
             legend=True, grid=True,
             title=f'Ratanopeus: alue {ax_split}')
]

# Oleskeluajan datapisteet ja sovite reaktorin alaosassa
plt.figure()
fit_data_time_bottom = [
    scatter_plot(pebbledata['times_rad'][0], pebbledata['times'][0],
                 label='Datapisteet', c='b', ms=3),

    fit_plot([pebbledata['times_rad'][0]], [pebbledata['times'][0]],
             fun_polynom, label='Polynomisovite', color='r',
             xlabel='Etäisyys keskilinjasta [m]',
             ylabel='Oleskeluaika [s]',
             legend=True, grid=True,
             title='Oleskeluaika: alue 1')
]

# Oleskeluajan datapisteet ja sovite reaktorin yläosassa
plt.figure()
fit_data_time_top = [
    scatter_plot(pebbledata['times_rad'][-1], pebbledata['times'][-1],
                 label='Datapisteet', c='b', ms=3),
```

```
    fit_plot([pebbledata['times_rad'][-1]], [pebbledata['times'][-1]],
             fun_polynom, label='Polynomisovite', color='r',
             xlabel='Etäisyys keskilinjasta [m]',
             ylabel='Oleskeluaika [s]',
             legend=True, grid=True,
             title=f'Oleskeluaika: alue {ax_split}')
]

plt.show()
```