Sademevee pindmise äravoolu dünaamika modelleerimine linnalistel valgaladel

Bakalaureusetöö

Üliõpilane: Iris Paalmäe
Üliõpilaskood: 213488YAFB
Juhendaja: Katrin Kaur, Konstruktsiooni- ja vedelikumehaanika uurimisrühm, teadur
Õppekava: Rakendusfüüsika

Tallinn 2024

Autorideklaratsioon

Kinnitan, et olen koostanud antud lõputöö iseseisvalt ning seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud.

Autor: Iris Paalmäe
20.05.2024
/Allkirjastatud digitaalselt/

Töö vastab bakalaureusetööle esitatavatele nõuetele.
Juhendaja: Katrin Kaur
20.05.2024
/Allkirjastatud digitaalselt/

Töö on lubatud kaitsmisele.
Kaitsmiskomisjoni esimees: [nimi]
[allkiri ja kuupäev]

Sisukord

Sisukord 3
1 Sissejuhatus 4
2 Numbriline modelleerimine 6
2.1 Navier-Stokes'i võrrandid 6
2.1.1 Rõhu-kiiruse sidususprobleem ja seda lahendavad algoritmid 7
2.2 Sisepinna meetod 8
2.3 Turbulents 9
2.4 Courant-Friedrich-Lewy tingimus 10
3 Metoodika 12
3.1 OpenFOAM 12
3.2 Eeltöötlus 13
3.2.1 Punktipilve puhastamine 13
3.2.2 Arvutusvõrgu loomine 14
3.2.3 Mudeli seadistamine 17
4 Tulemused ja arutelu 21
5 Tänuavaldused 26
6 Allikad 27
7 Annotatsioon 29
8 Abstract 30
9 Lisad 31
Lisa 1 */blockMeshDict* 31
Lisa 2 */snappyHexMeshDict* 32
Lisa 3 */alpha.water* 35
Lisa 4 */p_rgh* 36
Lisa 5 */U* 37
Lisa 6 */controlDict* 38
Lisa 8 */fvSchemes* 39
Lisa 9 */fvSolution* 40
Lisa 11 */boundary* 41
Lisa 13 */transportProperties* 42

1 Sissejuhatus

Eelmisel sajandil on Läänemere-äärsetes riikides, sealhulgas Eestis [1], täheldatud statistiliselt olulist kliimasoojenemist [2] ja aasta keskmise sademete hulga kasvu (5-15%). Sademete osas prognoositakse trendi jätkumist aastatel 2041-2070 (10-14%) ja 2070-2100 (16-19%) [3]. Samal ajal on ekstreemsetest sadudest tingitud üleujutused linnaaladel juba praegu ulatuslik probleem [4]. Linnapiirkondades on probleemiks valingvihm ehk vihm, mis langeb intensiivselt ja lühikese aja jooksul ning sageli põhjustab lokaalseid üleujutusi, seejuures ohtlikuks peetakse sademete hulka 30mm või rohkem ühe tunni või lühema aja jooksul [5]. Lisaks kliimamuutustele, tõstab sademeveesüsteemide koormust ka kõvakattega pindade osakaalu suurenemine sademeveesüsteemide valgaladel. Valingvihmade mõju linnas võimendub, sest kõvakattega pinnad takistavad vee imbumist pinnasesse ja suunavad selle lühikese kokkuvoolu ajaga (ingl. *time of concentration* – aeg, mis kulub äravooluvee jõudmiseks valgala kaugeimast punktist valitud kollektoripunkti [6]) sademeveetorustikesse, mis võib põhjustada torustike ülekoormust ja kanalisatsiooniuputusi [5], ehk kanalisatsioonivee (sh sademevee) voolamist maapinnale või hoonetesse selle ebapiisava vastuvõtuvõime tõttu [6].

Kliima, linnastumine, erinevad uurimused [7] ja direktiivid [8] suunavad digitaliseerimise ja targa linna põhimõtete rakendamisele [9] ning täpsustatud riskianalüüside läbiviimisele. Keerukamate alade ja probleemide analüüsimiseks kasutataksegi sageli modelleerimist, mis võimaldab mõista ja ennustada, kuidas erinevad protsessid toimivad ning kuidas need võivad mõjutada elukeskkonda [10]. Selleks on aga vaja suure detailsusega ruumiandmeid ning põhjalikult läbi mõeldud töövoogusid. Näiteks valingvihmadest tingitud riskide hindamiseks ja maandamiseks linnapiirkondades on vaja olemasolevate sademeveesüsteemide ja maapinna topograafia andmestikke, et defineerida võimalikke vooluteid. Enamasti määratakse selleks erinevate mudelite sisendina valgalad, kasutades selleks erinevaid meetodeid: näiteks käsitsi koostamine, matemaatilised (nt Thiesseni hulknurgad, kus ala jagatakse punktideks, mille ümber moodustatakse hulknurgad ning valgala piir on määratletud kui piirkond, kus vaadeldav punkt on lähim) ja kõrgusandmetele tuginevad poolautomaatsed meetodid (nt *arc hydrology ArcGIS Pro* tarkvaras, kus arvestatakse maapinna kallet vee voolamise suuna määramiseks). Kõigil eeltoodud meetoditel on omad miinused, näiteks käsitsi koostamine on väga ajamahukas ja kõrgusandmetele tuginevad on piiratud resolutsiooniga.

Valgaladel, ehk geograafilistes piirkondades, kus näiteks vihmaveest või lume sulamisest tekkinud voolav vesi koondub ühte punkti, on linnakeskkonnas voolamine oluliselt mõjutatud mitmesugustest ruumielementidest nagu tänavate äärekivid ja teetõkised. Sademevee pindmise äravoolu vooluteid mõjutavate linnaruumi objektide analüüs eeldab põhjalikku ning mitmekülgset andmestikku. Väljakutset pakuvad erinevad mõõtmed, kus ühelt poolt on vaja kirjeldada väikesemõõdulisi lokaalseid detaile, nagu teekünniste kõrgused (sentimeetrite skaala), ja teisalt kirjeldada terviklikke maa-alasid (meetrite ja kilomeetrite skaala), et võtta arvesse objektide omavahelist paiknemist. Seetõttu ei kirjelda laialdaselt kasutust leidev aerolaserskaneerimise (ingl. *Light Detection and Ranging* - LiDAR) tulemusel loodud digitaalne kõrgusmudel (ingl. *Digital Elevation Model* – DEM, enamasti resolutsiooniga 1, 5 ja 10 meetrit) linnalisi valgalasid piisava täpsusega, et analüüsida sademevee pindmise äravoolu tegelikke vooluteid. Üks võimalus saada piisavalt täpseid andmeid on droonimõõdistused, mis on inseneerirakendustes laialt kasutuses.

Droonimõõdistusi saab läbi viia nii LiDARi kui ka fotogrammeetria meetodil, kusjuures viimane tagab kõrgema eraldusvõime (1-4cm) ning seeläbi võimaldab läbi viia täpsustatud analüüsi.

Droonimõõdistuse alusel loodava punktipilve kasutamine pindmise äravoolu analüüsis on suurt arvutusressurssi nõudev ülesanne ning nõuab seega sobivate töövoogude välja töötamist ja praktilise rakendatavuse analüüsi. Samas võimaldab selline täpne andmestik hinnata, kuidas teetõkked voolamist mõjutavad, kuidas toimub erinevate vihma intensiivsuste korral vee jaotumine lähestikku asuvate restkaevude vahel, kuidas on tänavaruumi toimivust mõjutanud verikaalplaneeringut muutnud lokaalsed rekonstrueerimistööd ning milliseid teekondi pidi vesi hoonete keldritesse jõuda võib. Käesoleva töö eesmärk on välja töötada töövoog, mis võtab kasutusele droonimõõdistuste käigus salvestatud kõrge eraldusvõimega punktipilve ning analüüsida selle meetodi võimalusi ja kitsaskohti, et pakkuda sisendit näiteks automatiseeritud valgalade määramise täpsustamiseks. Selleks rakendati arvutusliku vedelike dünaamika (ingl. *Computational Fluid Dynamics* - CFD) mudeleid, mis võimaldavad simuleerida keerulisi füüsikalisi nähtusi. Mudelid ehitati kõrge eraldusvõimega (punktipilv kuni 1x1cm resolutsiooniga) droonifotogrammeetriaga mõõdistatud maapinna andmestike alusel.

Käesolev lõputöö on motiveeritud *Life BuildEst* projektist, mille raames tehti fotogrammeetria droonimõõdistus ja selle töö alusel on koostatud konverentsi (WDSA/CCWI 2024) artikkel pealkirjaga *„Integrating Drone-captured Sub-catchment Topography with Multiphase CFD Modelling to Enhance Urban Stormwater Management“* [11].

2 Numbriline modelleerimine

Käesolev töö põhineb CFD meetodil, mis kasutab arvutisimulatsioone keerukate vedelike ja gaaside, osakeste ning segude liikumise modelleerimiseks. Selleks kasutab CFD erinevaid diskretiseerimise meetodeid nagu lõplike elementide, lõplike vahede, Lattice Boltzmanni või lõplike mahtude meetod (ingl. *Finite Volume Method* – FVM), kusjuures viimast kasutatakse ka käesolevas töös [12]. FVM teisendab vedelike voolamist kirjeldavad Navier-Stokes'i võrrandid (osatuletistega diferentsiaalvõrrandid) lineaarvõrrandite süsteemiks [12]. Vedelike ja gaaside koosvoolamise, ehk mitmefaasiliste voolamiste lahendamiseks on arendatud mitmeid numbrilisi meetodeid nagu näiteks sisepinna meetod (ingl. *Volume of Fluid* – VoF). Mainitud meetodid on võimelised täpselt jäädvustama suuremat osa voolamise füüsikast, aga sõltuvad oluliselt aluseks loodavast arvutusvõrgust. CFD mudelid lahendavad massijäävuse, impulsi- ja energiavõrrandeid nii laminaarsele kui ka turbulentsele voolamisele. Selles peatükis on kirjeldatud CFD modelleerimiseks kasutatavaid võrrandeid ning VoF meetodit. Järgnevate võrrandite lahendamiseks on mitmeid algoritme, millest tuntumad on kolm – *SIMPLE*, *PISO* ja *PIMPLE*. Käesolevas töös kasutatakse võrrandite numbriliseks lahendamiseks *PISO* ja *PIMPLE* meetodit.

2.1 Navier-Stokes'i võrrandid

Navier-Stokes'i võrrandid kirjeldavad matemaatiliselt vedelike liikumist ning neid on vastavalt erinevatele eeldustele, näiteks kokkusurumatus, võimalik oluliselt lihtsustada. Navier-Stokes'i võrranditega on võimalik väga täpselt modelleerida kogu voolamist turbulentsest või laminaarest ühefaasilisest kokkusurumatust voolamisest kuni kokkusurutava mitmefaasilise voolamiseni [12]. Need võrrandid täielikul ja lihtsustatud kujul võimaldavad projekteerida lennukeid, autosid, elektrijaamu, analüüsida vere voolamist, reostuse levikut ja lahendada mitmeid muid probleeme. Erinevalt klassikalisest mehaanikast, kus keskendutakse punktmassi trajektoorile või pideva keskkonna liikumisele, uuritakse vedelikus pigem kiirust [12], mida näitab ka võrrandite lahendus arvutuselementides ajahetkedel. Voolujooned, mis on nagu teed, mida mööda kujuteldav vedeliku osake liigub, aitavad meil mõista vektorvälja käitumist. Need rajad on integraalkõverad, mille tuletis igas punktis on võrdne vektorväljaga ja need võivad visuaalselt kujutada vektorvälja käitumist teatud ajahetkel [12]. Kiirusvälja teades võib dünaamiliste võrrandite ja seoste abil leida muid huvipakkuvaid suurusi nagu rõhk, vooluhulk vms.

Kokkusurumatuks nimetatakse vedelikke, milles rõhu muutused ei avalda tihedusele märkimisväärset mõju [10]. Käesolevas töös modelleeritakse nii õhku kui ka vett kokkusurumatuna, eeldades, et kiirused on piisavalt väikesed. Kasutatav mudel lahendab kokkusurumatu voolamise pidevuse võrrandit:

$$\nabla \cdot \mathbf{u} = 0, \tag{1}$$

kus $\mathbf{u}$ on kiirusvektor ning kokkusurumatu voolamise impulsi võrrandit konservatiivsel kujul:

$$\frac{\partial}{\partial t}[\rho \mathbf{u}] + \nabla \cdot \{\rho \mathbf{u}\mathbf{u}\} = -\nabla \mathrm{p} + \mu \nabla^2 \boldsymbol{u} + \boldsymbol{f}_b\,, \tag{2}$$

kus ρ on tihedus, kandilistes sulgudes $[\rho \mathbf{u}]$ on skalaarse suuruse (tiheduse) ja vektorsuuruse (kiiruse) korrutis, mis on vektor, loogelistes sulgudes $\{\rho \mathbf{u}\mathbf{u}\}$ on skalaarse suuruse korrutis diaadilise

korrutisega (kahe vektori korrutis, mis on tensor), mille divergents on vektor [13], p on rõhk, μ on dünaamiline viskoossus ja $\boldsymbol{f}_b$ on raskusjõud.

2.1.1 Rõhu-kiiruse sidususprobleem ja seda lahendavad algoritmid

Kokkusurumatute vedelike puhul tekib modelleerimisel rõhu-kiiruse sidususprobleem. Pidevuse võrrandist ei saa otse arvutada impulsi võrrandis esinevat rõhuvälja, selle võrrandi puudumise tõttu, küll aga saab arvutada kiirusvälja. Rõhu võrrandi tuletuskäik pidevuse- ja impulsivõrrandite abil [14] on järgnev.

Pidevuse võrrand jääb samaks nagu võrrand (1) ning võrrandi (2) saab esitada üldmaatriksi kujul:

$$\mathbf{M}\mathbf{u} = -\nabla p, \tag{3}$$

kus $\mathbf{u}$ on otsitav, p on otsitav, $\mathbf{M}$ on tegurite maatriks, mis leitakse impulsivõrrandi diferentsiaalliikmete tuletamisel, kasutades FVM meetodit. Üldmaatriksi kuju (3) komponentideks lahti kirjutatult on järgmine:

$$\begin{pmatrix} M_{1,1} & M_{1,2} & \cdots & M_{1,n} \\ M_{2,1} & M_{2,2} & \cdots & M_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ M_{n,1} & M_{n,2} & \dots & M_{n,n} \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{pmatrix} = \begin{pmatrix} (\partial p/\partial x)_1 \\ (\partial p/\partial x)_2 \\ \vdots \\ (\partial p/\partial x)_n \end{pmatrix}. \tag{4}$$

Tegurite maatriksi $\mathbf{M}$ saab jagada diagonaalseteks ja mittediagonaalseteks komponentideks ($\mathbf{A}$) ja ($\mathbf{H}$) (st. maatriksist (4) võetakse peadiagonaal ja tehakse sellest maatriks $\mathbf{A}$, kus peadiagonaali ümber on nullid ning kõikidest liikmetest mis jääb üle moodustatakse mittediagonaalne maatriks $\mathbf{H}$, kus peadiagonaalis on nullid ja ümber selle on maatriksi $\mathbf{M}$ ülejäänud liikmed). Seeläbi saab võrrandi (3) vasaku poole ümber tõstmisel võrrand kuju (5):

$$\mathbf{M}\mathbf{u} = \mathbf{A}\mathbf{u} - \mathbf{H}, \tag{5}$$

asendades selle võrrandisse (3), saadakse:

$$\mathbf{A}\mathbf{u} - \mathbf{H} = -\nabla p, \tag{6}$$

mis annab diskretiseeritud impulsivõrrandi, mida nimetatakse ka algosadeks lahutatud impulsi võrrandiks. $\mathbf{A}$ pöördmaatriks näeb välja järgmine:

$$\mathbf{A}^{-1} = \begin{pmatrix} \frac{1}{A_{1,1}} & 0 & \dots & 0 \\ 0 & \frac{1}{A_{2,2}} & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & \frac{1}{A_{n,n}} \end{pmatrix} \tag{7}$$

Maatriksi (5) ümber tõstmisel, saadakse maatriks (8):

$$\mathbf{H} = \mathbf{A}\mathbf{u} - \mathbf{M}\mathbf{u}, \tag{8}$$

mida vajame rõhu võrrandi parema poole arvutamiseks. Rõhu võrrand saadakse korrutades impulsivõrrandi (6) mõlemad pooled pöördmaatriksiga (7) ning saadud tulemuse ümbertõstmisel avaldatakse kiiruse võrrand (9):

$$\mathbf{u} = \mathbf{A}^{-1}\mathbf{H} - \mathbf{A}^{-1}\nabla p. \tag{9}$$

Kuna võrrandis (9) on rõhuväli otsitav suurus, siis ei rahulda see pidevuse võrrandit. Asendades võrrandist (9) kiirusvektori $\mathbf{u}$ pidevusvõrrandisse (1) ja ümber organiseerides, saadakse rõhu Poissoni võrrand (10):

$$\nabla \cdot (\mathbf{A}^{-1}\nabla p) = \nabla \cdot (\mathbf{A}^{-1}\mathbf{H}). \tag{10}$$

Võrrandi (10) lahendamisel leitud rõhuväljaga on uuesti võimalik lahendada kiiruse võrrand (9), mis seekord rahuldab pidevuse võrrandit. Kuna aga võrrandis (10) on maatriks $\mathbf{H}$, mis sõltub kiirusest (Võrrand 9), siis ei ole rõhu võrrand enam õige ning viimane nõuab uuesti arvutamist. Selles seisnebki rõhu-kiiruse sidususprobleem, sest arvutades uue rõhuvälja väärtuse on vaja arvutada ka kiirusväli jne. Antud võrrandeid lahendavad erinevates kombinatsioonides ja erinevate korduvustega algoritmid *SIMPLE*, *PISO* ja *PIMPLE*. *SIMPLE* algoritm (ingl. *Semi-Implicit Method for Pressure Linked Equasions*) on mõeldud statsionaarse voolamise probleemidele ehk probleemidele, mis aja jooksul ei muutu ning *PISO* (ingl. *Pressure Implicit with Splitting of Operations*) on mõeldud mittestatsionaarsete probleemide lahendamiseks [14]. Mõlemad algoritmid põhinevad esialgsete lahenduste hindamisel ning seejärel paranduste tegemisel. Nende algoritmide erinevused tulenevad sellest, mis järjekorras eeltoodud võrrandeid lahendatakse ja korratakse [13]. *SIMPLE* algoritm läheb terve tsükli algusesse võrrandisse (3) ja arvutab selle kaudu maatriksi $\mathbf{H}$ ning lahendab uue $\mathbf{H}$ väärtusega taaskord rõhu ja kiiruse võrrandid - seda tsüklit korratakse. *PISO* algoritm aga lahendab ühe korra võrrandi (3) ning seejärel alustatakse tsüklit, kus arvutatakse maatriks $\mathbf{H}$ ning rõhu ja kiiruse võrrandid ehk kui arvutus jõuab tsükli lõppu ei arvutata enam uuesti impulsivõrrandit (3) vaid jätkatakse koheselt uue $\mathbf{H}$ leidmisega. *PIMPLE* algoritm ühendab endas nii *SIMPLE* kui *PISO* algoritmi ning on optimaalne suurematekss ajasammudeks, võimaldades osalist ajaintervallide vähendamist ja Courant'i arvu suurendamist. Nendes tsüklites lahendatakse ka skalaarsed suurused nagu vedeliku fraktsioon, või turbulentne kineetiline energia olenevalt sellest, mis tüüpi voolamist modelleeritakse.

2.2 Sisepinna meetod

Sisepinna meetodiga (VoF) analüüsitakse faasifraktsiooni parameetrit α mis näitab kui suur osa arvutusvõrgu ühest elemendist on vedelikuga täidetud ning selle väärtus võib varieeruda nullist üheni (näiteks vee ja õhu modelleerimisel 0 kui elemendis ei ole vett, 1 kui elemendis on ainult vesi ja 0.5 kui elemendis paikneb eralduspind) [15]. See annab ka info selle kohta, kus asub kahe vedeliku vaheline eralduspind. VoF meetodi puhul lahendatakse koos Navier-Stokes'i võrranditega faasi fraktsiooni transpordivõrrandit (11):

$$\frac{\partial \alpha}{\partial t} + \nabla \cdot (\alpha \mathbf{u}) = 0. \tag{11}$$

VoF meetodi puhul vaadeldakse erinevate vedelike segu ühise liitvedelikuna, mille omadused igas arvutuselemendis sõltuvad faasifraktsiooni α väärtusest. Pidevuse ja impulsi võrrandite arvutus

toimub samamoodi nagu võrrandites (1) ja (2). Pindpinevus modelleeritakse pideva pinnajõuna – see arvutatakse eraldi valemiga:

$$\boldsymbol{f}_\sigma = \sigma\kappa\nabla c, \tag{12}$$

kus c on pindpinevuskonstant ja κ on pinna kumerus. VoF meetodis leitakse tihedus ρ järgmiselt:

$$\rho = \alpha\rho_1 + (1-\alpha)\rho_2, \tag{13}$$

kus α muutub nullist üheni ning ρ_1 ja ρ_2 modelleeritavate vedelike tihedused. Ka arvutuselemendis paikneva liitvedeliku viskoossus ja kiirus leitakse sama loogika alusel.

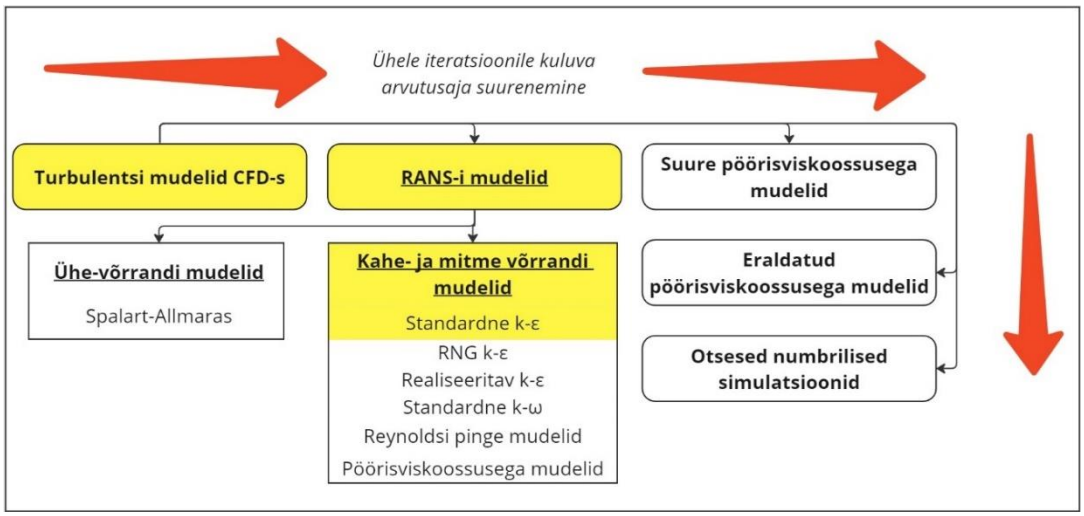
2.3 Turbulents

Vedeliku voolamisel on kaks võimalikku režiimi – laminaarne ja turbulentne [10]. Turbulentseks voolamiseks nimetatakse pööristega täidetud voolamist, mille trajektoor on keerulise kujuga ja pidevalt muutuv ning laminaarseks vedeliku kihilist liikumist, kus voolu trajektoor on korrapärase kujuga [10]. Üleminekut laminaarselt režiimilt turbulentsele iseloomustab kriitiline Reynoldsi arv. Raskusjõu toimel mööda kaldpinda voolava õhukese veekihi Reynoldsi arv leitakse valemiga (14) [16]:

$$Re = \frac{U \cdot h}{\nu}, \tag{14}$$

kus U on keskmine kiirus veekihis, h on veesamba kõrgus ja ν on vee kinemaatiline viskoossus. Selliselt leitud Reynoldsi arvu kriitilise väärtuse alampiir on vahemikus 500-1000.

CFD-s kasutatakse turbulentse voolamise modelleerimiseks erinevaid mudeleid (Joonis 1), millest käesolevas töös võeti kasutusele keskmistatud Navier-Stokes'i võrrandi (ingl. *Reynolds-Averaged Navier-Stokes* – RANS) mudelite hulka kuuluv standardne k-epsiloni ($k-\varepsilon$) turbulentsi mudel, sest see on üks laialdasemalt kasutatavaid mudeleid, mis sobib suurele osale voolamistest. RANS põhineb võrrandite keskmistamisele ajas – voolumuutujad jagatakse kaheks: keskväärtuse ning fluktuatsiooni komponendiks. Seejärel lisatakse need osad vedelike liikumist kirjeldavatesse võrranditesse ja võrrandid keskmistatakse üle aja.



Joonis 1. Turbulentsi mudelite jaotus [17]. Kollasel taustal käesolevas töös kasutatav mudel.

Valitud mudel kasutab kahte transpordivõrrandit, et kirjeldada voolu turbulentsust ja arvutada Reynoldsi keskmistamise tulemusel impulsi võrrandis tekkivat Reynoldsi pinge tensorit. Esimene transporditav muutuja on turbulentne kineetiline energia (k) ning teine on turbulentse kineetilise energia hajumise kiirus (ε), mis määrab turbulentsi ulatust. See mudel on sobiv voolamisele, kus on madal rõhugradient, eriti voolamisele, mis toimub mingi pinna lähedal [18].

Peatükis 2.1.1 kirjeldatud algoritmid lahendavad turbulentseid voolamisi nagu laminaarseidki, aga tsüklisse lisatakse veel kaks transpordivõrrandit [14]. Nendeks on:

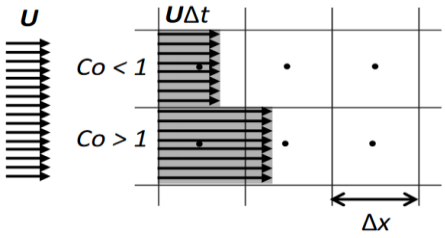
$$\mathbf{M}_k k = S_k, \tag{15}$$

$$\mathbf{M}_\varepsilon \varepsilon = S_\varepsilon, \tag{16}$$

kus S_k ja S_ε on turbulentset kineetilist energiat ja selle hajumist kontrollivad liikmed.

2.4 Courant-Friedrich-Lewy tingimus

Courant-Friedrich-Lewy tingimuse Courant'i arv on CFD üks peamisi parameetreid, mis määrab numbrilise arvutuse stabiilsuse [12]. Selle piiramine tagab, et kujuteldav vedelikuosake ei saa ühe ajasammu jooksul liikuda rohkem kui ühe elemendi pikkuse võrra. Courant'i arv näitab, kui suure osa ühest arvutusvõrgu elemendist voolamine ühe ajasammuga läbi liigub (Valem 17). Üldiselt ei tohi Courant'i arv olla suurem kui üks (Co < 1) ja seda võib mõista nii, et kui Co < 1, jõuab informatsioon ühest võrgu elemendist järgmisesse ühe ajasammu jooksul ning kui Co > 1, jõuab informatsioon ka teise või kolmanda naaberelemendini, mis võib põhjustada vigu arvutuses (Joonis 2) [13]. Samas leidub ka mudelite seadistusi, mille puhul on võimalik stabiilsus tinglikult tagada suuremate väärtuste korral ja seadistusi, mille puhul tuleb Courant'i arv hoida ühest väiksem.



Joonis 2. Courant'i arvu põhimõtte visualiseerimine [19].

Courant'i arv on defineeritud järgmiselt:

$$Co = \frac{\mathbf{u}\Delta t}{\Delta x}, \tag{17}$$

kus $\mathbf{u}$ on kiirusvektor, Δt on ajasamm ja Δx on vahemaa, mille vedelik on ajasammu jooksul läbinud. Valemi (17) põhjal saab järeldada, et mida suurem on vaadeldavas elemendis kiirus u, seda suurem on Courant'i arv [13]. Courant'i arvu kriteerium peab olema arvutusvõrgu igas elemendis täidetud (Co < 1), sest iga üksik halvasti defineeritud element võib muuta arvutuse ebastabiilseks.

Courant-Friedrich-Lewy (CFL) tingimus ütleb, et diferentsiaalvõrrandi lahenduse lähenemiseks osatuletistega diferentsiaalvõrrandi lahendile, peab arvutusskeem kasutama kogu lahendust mõjutavat algandmetes sisalduvat informatsiooni. Lihtsamalt võib võtta CFL-i tingimust kui üht põhireeglit, mida koefitsiendid peavad rahuldama [12].

OpenFOAM'is toimub Courant'i arvu arvutus pigem elemendi ruumala V kui elementide keskpunktide kauguse baasil [13]. Ühe elemendi pindala arvutamisel võetakse arvesse kõik elemendi pinnad (ingl. *face*), mistahes kujuga antud element on, sest *OpenFOAM* võimaldab teostada arvutusi paindlikult, meelevaldse kujuga elementidel. Ideaalne element on siiski kuubik või risttahukas.

3 Metoodika

Käesolevas töös loodi droonifotogrammeetria mõõdistuse tulemusel saadud punktipilvest geomeetriline pind, mida kasutati erinevate mõõtmete ja resolutsiooniga arvutusdomeenide loomiseks arvutusliku vedelike dünaamika simulatsioonide läbiviimise eesmärgil. Simulatsioonide usaldusväärsuse hindamiseks analüüsiti erinevaid arvutusvõrkude kvaliteedi parameetreid ja võrreldi tulemusi geoinfosüsteemide-põhise analüüsi tulemustega. Töö läbiviimiseks rakendati erinevaid tarkvarasid.

CFD modelleerimine koosneb kolmest põhietapist: eeltöötlus, töötlus ja järeltöötlus. Eeltöötluse alla kuuluvad arvutusdomeeni ja -võrgu loomise tegevused ning piiritingimuste seadistamine. Eeltöötluseks kasutati põhiliselt *Salome*, *MeshLab* ja *OpenFOAM* rakendusi. Töötluseks, ehk simulatsioonide läbiviimiseks kasutati *OpenFOAM*'i tarkvara *interFoam* lahendajat, mis baseerub VoF meetodil ning järeltöötluseks, ehk andmete analüüsimiseks *ParaView* ning *ArcGIS Pro* tarkvaraga saadud tulemusi. Kuna eeltöötlus on CFD modelleerimise mahukaim etapp, on sellele pühendatud metoodika osas eraldi peatükid. Töötluse faasis lahendab tarkvara Numbrilise modelleerimise peatükis kirjeldatud võrrandeid ning järeltöötlust käsitletakse põgusalt tulemuste osas.

3.1 OpenFOAM

OpenFOAM (*Open Field Operation and Manipulation*) on tasuta ja avatud lähtekoodiga C++ programmeerimiskeelne arvutusliku vedelike dünaamika tarkvara. See võimaldab numbriliselt lahendada pideva keskkonna mehaanika probleeme ning on laialdaselt levinud inseneriteadustes ja -ettevõtetes, aga ka teadustöös. Tänu sellele, et antud tarkvara on avatud lähtekoodiga, lubab see kasutajatel põhjalikult süveneda erinevate protsesside läbiviimiseks loodud koodi ning omale vajalikul viisil rakendada, muuta ja kombineerida erinevatele ülesannete mõeldud lahendusalgoritme ja koodi. *OpenFOAM* koosneb üle 100 funktsionaalsest moodulist, mida saab kasutada üle 250 erineva rakenduse loomiseks. Lahendusalgoritmid simuleerivad CFD-probleeme ning muud insenerimehaanika rakendused teostavad andmete manipuleerimist, visualiseerimist, arvutusvõrgu töötlemist ja palju muud [20]. Tarkvaras sisalduvad ka moodulid eeltöötluse läbiviimiseks, nagu näiteks arvutusvõrgu genereerimise rakendused *blockMesh* ja *snappyHexMesh* ja selle kvaliteedi parameetreid hindav *checkMesh* ning järeltöötluse rakendused, mis võimaldavad terminali käskudega viia läbi täiendavaid analüüse, sh arvutada väljade keskmisi väärtusi jne. Kasutaja seadistab oma lahendatava ülesande defineerides konkreetse probleemi simuleerimiseks vajalikud andmesõnastikud juhtumi kataloogis (ingl. *case directory*). Iga lahendatav probleem sisaldab andmesõnastikke arvutusvõrgu, väljade, vedelike omaduste, kontrollparameetrite jms kohta. Viimased on salvestatud failidena juhtumi kataloogis. Kõik juhtumi kataloogid järgivad sama struktuuri, kus esimesed kaustad on *0/*, *constant/* ja *system/ (*Joonis 3*)*. Kataloogist *0/* leitakse esmaseid piiritingimusi sisaldavaid ajafaile erinevate väljade jaoks (nt kiirus- ja rõhuväli ning modelleeritavate faaside definitsioonid piiretel). Kaustas *constant/* on andmed alusvõrgu, geomeetria, transpordi- ja turbulentsiomaduste kohta. Kõik informatsioon, mis puudutab numbrilisi skeeme ja informatsiooni ajasammu kohta, on leitavad kaustas *system/*. Töötlusetapi käigus kirjutatakse katalooge juurde, mis kirjeldavad voolamise igat ajasammu.

Joonis 3. OpenFOAM juhtumi kataloogi struktuur enamuste alamkaustade ja -failidega.

Üks *OpenFOAM* lahendajatest on *interFoam*, mis on mõeldud kahe kokkusurumatu segunematu vedeliku jaoks, kasutades VoF meetodi faasifraktsioonil (Peatükk 2.2 Sisepinna meetod) põhinevat eralduspinna asukoha määramise meetodit (ingl. *interface-capturing method*) [14]. Käesolevas töös modelleeriti sademevee pindmist voolamist antud lahendajaga, kus eirati vedelike kokkusurutavust ja segunemist, defineerides vaid viskoossuse, tiheduse ja pindpinevusjõu. Lahendajat juhitakse kaustas */system* (Joonis 3) [14]. Failis */controlDict* kontrollitakse muuhulgas aja- ja andmete kirjutamise sammu ning seadistatakse vastavalt Courant'i arvu väärtusele arvutuse käigus kohanev ajasamm (ingl. *adjustable timeStep*). Courant'i arvu kontrollitakse simulatsiooni käigus igal ajahetkel kogu domeenile ja eralduspinnale. Sõnastikus */fvSchemes* määratakse rõhu- ja kiirusvälja numbrilise lahendamise skeemid ning määratakse arvutusmeetodid sõnastiku alamosadena. Igas alamosas on teatud tüüpi meetodid, näiteks *gradSchemes* sisaldab kõiki gradiendi lahendamiseks vajalikke meetodeid. Failis */fvSolutions* määratakse soovitud algoritm rõhu-kiiruse sidususprobleemi lahendamiseks, failis */transportProperties* defineeritakse vedeliku omadused (näiteks kas Newtoni või mitte-Newtoni vedelik), Newtoni vedelikule antakse ette kinemaatiline viskoossus ja tihedus ning määratakse pindpinevusjõud kahe segunematu vedeliku vahel.

3.2 Eeltöötlus

Numbrilise modelleerimise esimene faas on eeltöötlus, mis hõlmab geomeetria loomist ja diskretiseerimist. Simulatsiooni lahendamiseks valitakse piiritingimused ja lahendusalgoritmid. See faas on modelleerija kõige aeganõudvam etapp, sest töötluseks peab valmima korralik, minimaalsete diskretiseerimisvigadega arvutusvõrk, mille peal hakatakse simulatsiooni käivitama [15]. Peatükis on kirjeldatud eeltöötlust ette antud punktipilvega, millest loodi pinnafailid ja seejärel arvutusvõrk, võttes arvesse andmemahtu (vältimaks väga suuremahuliseks muutumist ja arvutusaja minemist nädalate või isegi kuude pikkuseks).

3.2.1 Punktipilve puhastamine

Töös kasutati Hades Geodeesia poolt *Life BuildEst* projekti raames drooniga mõõdistatud Tallinna Akadeemia tee piirkonna punktipilve. Viimasele tehti seal esialgne filtreerimine ja üleliigsete objektide eemaldamine, aga edastatud punktipilv koos loodud erineva resolutsiooniga STL (ingl. *stereolithography,* kolmnurk-elementidest koosnev pinnamudel) pindadega sisaldasid kriitilisel hulgal ebatäpsusi ja vajas edasist töötlust, milleks kasutati antud töös programmi *MeshLab*, mis on punktipilve töötlemiseks loodud vabavaraline tarkvara ning võimaldab seda parandada ja muuta vajaduspõhiselt [21]. Geomeetria loomisel kasutati ka *Salome* tarkvara, mis on inseneriteadustes laialdaselt kasutatav tasuta raalprojekteerimise tarkvara. See võimaldab paljude funktsioonidega luua nt. vedelike liikumise jms modelleerimiseks vajalikku geomeetriat ja arvutusvõret [22].

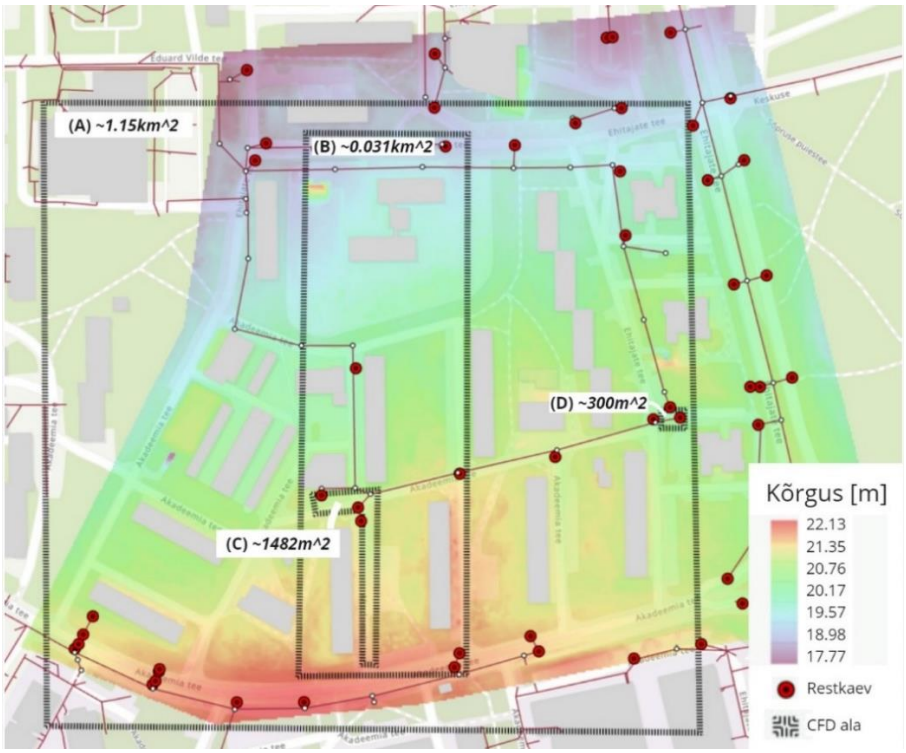
Punktipilve puhastamist alustati punktipilve parandamisega, eemaldades ebakorrapärasusi ja üleliigseid punkte, mis võisid hiljem moonutada analüüsi tulemusi või takistada lahendaja *interFOAM* rakendamist mudelil. Ebaolulisteks kohtadeks võib lugeda näiteks maapinna suhtes kõrgel olevaid objekte, nagu kujusid, rõdusid majadel jms, mis ei ole vajalikud sademevee pindmise äravoolu analüüsimiseks, aga ka punktipilve loomise faasis ekslikult sisse jäänud üksikuid punkte. Kuna droonifotogrammeetria alusel punktipilve loomisel jäävad esialgsesse tulemusse sisse kõik drooni vaateväljas asuvad objektid (sh puud ja autod) tuleb need eemaldada. Selle poolautomatiseeritud tegevuse käigus jääb andmestikku punkte, mida visuaalsel vaatlusel kohe ei tuvastata, kuid mis tekitavad ebatäpsusi punktipilvest STL pinna genereerimisel. *MeshLab* võimaldas rakendada erinevaid parandusmeetodeid nagu selekteeritud punktide ja pinnaelementide eemaldamine ning aukude sulgemine, mis olid selle töö olulised vahendid.

Tööd jätkati punktipilve lihtsustamisega, mis oli oluline samm, et hõlbustada analüüsi ja vähendada arvutuskoormust. Selleks kasutati erinevaid algoritme nagu maapinna lihtsustamine kolmnurksete pindade tipupaaride liitmisel (ingl. *Surface simplification: Quadric error metrics*), säilitades algse mudeli igas tipus geomeetrilise vea lähenduse ruutmaatriksite abil [23]. Veel kasutati pinna rekonstrueerimist Poissoni meetodil (ingl. *Screened Poisson: Surface reconstruction*), mis aitas tekitada veekindla mudeli aluspinna, ehk automaatselt täita väikesed avad. See meetod põhineb reeglil, et uus pind peab olema ühendatud vana pinna punktide kaudu, et säilitada algne pinna kuju [24]. Need algoritmid võimaldasid säilitada pinna olulisi omadusi, samal ajal vähendades selle keerukust. Lihtsustamine võimaldas ka paremat visualiseerimist ja suuremat efektiivsust hilisemates arvutusetappides. Puhastatud punktipilvest sai luua pinnad, mis defineerivad arvutusdomeeni piirded.

3.2.2 Arvutusvõrgu loomine

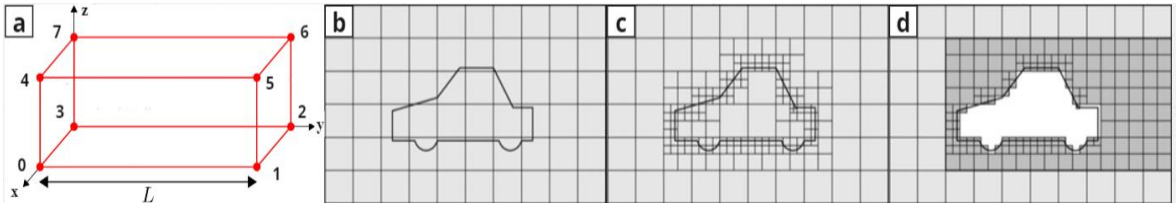
Mudel koosneb erinevatest piiretest nagu atmosfäär, seinad ja maapind ning vee haldamise piirded nagu sisse- ja väljavoolukohad, sh restkaevud. *Salome* ja *MeshLab* tarkvarades loodi need piirded, mis moodustasid ühise veekindla mudeli. Loodud piirded eksporditi STL-failidena juhtumi kausta */constant/polyMesh* (Joonis 3). Järgmine samm oli arvutusdomeeni loomine, milleks kasutati loodud STL faile koos *OpenFOAM*'i aruvutusvõrgu genereerimise rakendustega *blockMesh* (Lisa 1 */blockMeshDict*) ja *snappyHexMesh* (Lisa 2 */snappyHexMeshDict*).

Saadud punktipilv oli suur ala, millel katsetati erinevaid võimalusi äravoolu modelleerimiseks ja samuti mitmel alamalal arvutusmahu piiramiseks. Näiteks ristmikul, kus valgala moodustus ümber restkaevu (Joonis 4(D)) ja teealal, mis moodustas valgala lähestikku paiknevatele restkaevudele, (Joonis 4(C)). Viimasega jätkati edasist tööd – see oli Akadeemia tee 22 kortermaja esine ja selle küljel olev tänav, mis moodustas L tähe kujulise mudeli aluspinna. Antud piirkond valiti välja seetõttu, et teealal oli ligi meetrine kõrguste vahe. Lisaks oli antud teelõigul kolm restkaevu sügavustega 0.59m, 1.14m ja 1.23m ja läbimõõtudega vastavalt 700mm, 700mm ja 400mm millesse vedeliku voolamist eeldati.



Joonis 4. Valgalad, mida CFD-ga modelleeriti. (A) on suur majadega ala, (B) on väiksem majadega ala, (C) on Akadeemia tee 22 esine ala ja (D) ala ristmikul ümber teel paikneva restkaevu. Taustal kõrgusmudeli skaala värvides toodud droonimõõdistuste ala.

blockMesh töövahendit kontrolliva sõnastiku leiab *OpenFOAM*'i juhtumi kaustast *constant/polyMesh/blockMeshDict*. Selles sõnastikus määratakse alusvõrgu dimensioonid vajamineva arvu kastidega (ingl. *block*), mille põhjal jagab *blockMesh* domeeni korrapärasteks osadeks. Iga kast määratakse kaheksa nurga (ingl. *vertex*) koordinaatide abil (Joonis 5 (a)). Kasti minimaalse koordinaadi määrab tipp 0 ja maksimaalse tipp 6, aga kõik tipud tuleb määrata *blockMeshDict* seadistustes vastavalt vajadusele. Rakendades *blockMesh*'i alusvõrgu genereerimiseks, tuleb selle tipud määrata *snappyHexMesh*'i tarbeks nii , et ette antud geomeetria jääks igas suunas alusvõrgu sisse (Joonis 5 (b)), sest geomeetria mõne osa alusvõrgust välja jäämisel lõigatakse see osa *snappyHexMesh*'i protsessis välja ja minnakse edasi ainult alusvõrgu sees oleva geomeetriaga (Joonis 5 (d)). Kehvemal juhul ei oska *snappyHexMesh* algoritm sellisel juhul otsustada millist osa domeenist diskretiseerida ja arvutusvõrgu loomine seega ei õnnestu. Sobivate definitsioonide korral võib samas *blockMesh*'iga sihilikult pinda lõigata ja tekitada arvutusvõrk selliselt valitud osas.



Joonis 5. Piltidel (a) alusvõrgu kasti tipud, (b) alusvõrgu loomine, (c) võrgu täiustamine snappyHexMesh-ga ja (d) locationInMesh parameetri töö [14].

Alusvõrgu loomiseks antakse ette X-, Y- ja Z-telje suunalised väärtused, mis määravad mitu elementi iga koordinaattelje suunas luuakse. Siin tuleb mõelda määratav arv korralikult läbi, et loodud

elemendid oleksid enam-vähem võrdsete dimensioonidega. Antud sõnastikus defineeritakse ka eelnevalt loodud piiritingimused (sisse- ja väljavoolukohad jms).

snappyHexMesh töövahendi parameetreid seadistava sõnastiku leiab kaustast */system/snappyHexMeshDict*. Selle sõnastiku töö eesmärk on genereerida STL formaadis pinna geomeetria põhjal automaatselt 3-mõõtmeline arvutusvõrk. Viimane sisaldab kuusnurkseid (ingl. *hex*) ja jaotatud kuusnurkseid (ingl. *split-hex*) elemente. Moodustatud võrk sobitub ligikaudu geomeetriaga ja täiustab iteratiivselt lähtevõrku. Võrgu viimistlemise taseme spetsifikatsioon on väga paindlik. *snappyHexMesh* algoritm on paralleelarvutuse võimekusega ning selle töö põhineb enne käivitamist defineeritud */snappyHexMeshDict* sõnastikul [14]. Võrgu loomine toimub järgmiselt: luuakse alusvõrk, kasutades *blockMesh* käsklust (Joonis 5 (b)); defineeritakse geomeetria, mis asub kataloogis */constant/triSurface;* luuakse võrk, mis täidab kogu *blockMesh*'iga piiritletud ala ja hõlmab geomeetriat, kasutades *snappyHexMesh* käsklust; lisatakse pindade lähedale kihte (Joonis 5 (c)); kontrollitakse loodud võrgu kvaliteeti ning viimaseks visatakse välja võrgu osa (kas defineeritud geomeetria sees või väljaspool vastavalt uuritavale probleemile), mida ei soovita modelleerida (Joonis 5 (d)). CFD modellerimise oluline ülesanne on arvutusvõrgu täpsustamine, mida juhitakse */snappyHexMeshDict* sõnastikus asuvate *castellatedMeshControls* sätetega. Täpsustamine toimub mitmes etapis – viimistletakse arvutusvõrgu geomeetrilise pinna lähedal, et hõlbustada hiljem võrguelementide ja geomeetrilise pinna ühendamist ning vähendada anomaaliaid. Arvutusvõrgu elementide suurused on defineeritud alusvõrgu suhtes, näiteks alusvõrgu elemendi küljepikkuse 1m korral on täpsustasemel 1 ja 2 küljepikkused vastavalt 0,5m ja 0,25m. Sõnastikus on lisaks mitmeid võre tekitamise protsessi juhtivad parameetrid. Näiteks *maxLocalCells* ja *maxGlobalCells* piiravad elementide arvu protsessorites ja kogu võrgustiku tasandil. *nCellsBetweenLevels* lisab elemente kahe täpsustaseme vahele, vältides järske üleminekuid. *refinementSurfaces* ja *refinementRegions* määratlevad täpsustasemed geomeetria osadele, näiteks pinna lähedale või konkreetsetele aladele pinnal. Parameeter *resolveFeatureAngle* lisab viimistlust vastavalt pinnakumerusele (mida väiksem väärtus, seda rohkem viimistletakse). Viimaseks kasutatakse *locationInMesh* koordinaati, et säilitada soovitud elemendid lõpp-võrgus (elemendid, mis pole koordinaadiga ühendatud eemaldatakse).

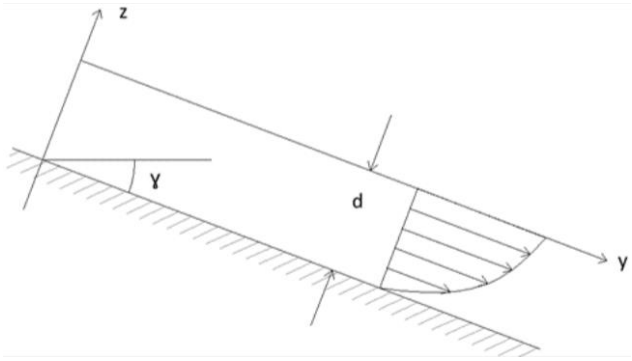
Käesolevas töös moodustati erinevate kombinatsioonide katsetamise järel Akadeemia tee 22 mudelile (A22) neli erineva suurusega alusvõrku */blockMeshDict* sõnastikus ühe bloki abil. Tippude 0 ja 6 koordinaadid valiti kõigile neljale võrgule samad ning määrati kahele võrgule koordinaattelgede suunalised parameetrid nii, et alusvõrgu ruudustiku element tekiks suurusega 1x1m ning teistele 0.8x0.8m ja 0.6x0.6m. Võrku täiustati */snappyHexMeshDict* sõnastikuga, kus määrati võrgu täpsustamise tasemed, *locationInMesh* koordinaat jms. Erineva resolutsiooniga võred loodi selleks, et oleks võimalik läbi viia võre tundlikkuse analüüsi ning määrata võre koonduvuse indeks (CGI, Peatükk 3.2.4 Tundlikkuse analüüs).

Nagu eelnevalt sai mainitud, siis eeltöötlus on kõige aeganõudvam osa numbrilisest modelleerimisest. Enne nende kolme mudelini jõudmist, katsetati läbi üle 50 erineva variandi, kuidas parim arvutusvõrk luua. Lisaks tehti eraldi alamalade peal katsetusi, et arvutusmahtu kokku hoida ning tutvuda erinevate võimalustega.

3.2.3 Mudeli seadistamine

Mudeli ettevalmistamise hetkeks loodi sellele korralik arvutusvõrk, mille arvutamiseks pidi seadistama kõik ülejäänud kaustad, et simulatsioon käivitada. Seadistust vajavad kaustad on näidatud Joonis 3 ning käesoleva töö olulisemad failid, mis seadistati mudelil 2 (vt. Peatükk 3.2.4 Tundlikkuse analüüs) on toodud Lisades 3-13.

Veevoolu kiiruse määramiseks (st. kui suure kiirusega vedelik domeeni siseneb), võeti arvesse allpool tuletatud kiirusprofiili arvutamise valemit (27), milles kasutati välisvaatlusel saadud ligikaudset veesamba kõrgust ja A22 tee kallet. Vedeliku kiiruse profiil näitab kiiruse suurust asukoha funktsioonina [10]. Joonis 6 on näidatud, milline näeb välja vedeliku laminaarne voolamine lõpmata suurel kaldtasandil, kus teljed z ja y on märgitud nagu käesolevas töös ning ɣ tähistab kaldenurka ja d veesamba kõrgust.



Joonis 6. Vedeliku kiiruse profiil [10].

Kiiruse arvutus toimub Navier-Stokes'i võrranditest lihtsustuste läbi tuletatud valemi põhjal. Valemi tuletamise eelduseks on et, voolamine toimub 2D-s ja laminaarselt ehk voolamise kiirus ei võrdu nulliga selle koordinaadi suunas, milles kiirusprofiili tahetakse arvutada. Käesoleva mudeli põhjal valiti voolamisega paralleelseks y-telg ja seega x-koordinaattelje sihiline kiirus $u = 0$, y-telje sihiline kiirus $v \neq 0$ ja z-koordinaattelje sihiline kiirus $w = 0$. Kõigepealt leiti kaldenurk, mille sai arvutada kalde kaudu. See leiti vastavalt mudeli koordinaatide algus- ja lõppkoordinaatide kaudu:

$$m = \frac{tõus}{maapinna\ pikkus} = \frac{\Delta z}{\Delta y} = \frac{z_2 - z_1}{y_2 - y_1}, \tag{18}$$

Kaldeks leiti $m = 0.0100621$ ning sellest kaldenurk $ɣ = arctan(m) = 0.57^o$.

Kuna tegemist on tasapinnalise kaldel voolamisega, siis gravitatsioonivälja tegur leitakse järgmiselt (gravitatsioonivektor jagatakse y- ja z-telgede sihilisteks komponentideks), võttes arvesse kaldenurka ɣ:

$$\mathbf{g} = \mathbf{g} \cdot sin(ɣ) - \mathbf{g} \cdot cos(ɣ)\,. \tag{19}$$

Nüüd vaatame pidevuse võrrandit (1), mis komponentide lahti kirjutamisel näeb välja järgmine:

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} = 0. \tag{20}$$

2D eelduse puhul lihtsustub võrrand (20) nii, et kiirusväli on ainult y-telje suunas nullist erinev ja üleliigsed komponendid taanduvad välja ning tulemus ruutu tõstes on järgmisel kujul:

$$\frac{\partial^2 v}{\partial y^2} = 0. \tag{21}$$

Liikumishulga võrrandit vaadatakse samuti vaid y-koordinaattelje suunas ehk:

$$\rho\left(\frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} + w\frac{\partial v}{\partial z}\right) = -\frac{\partial p}{\partial y} + \rho g_y + \mu\left[\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} + \frac{\partial^2 v}{\partial z^2}\right], \tag{22}$$

kus võrrandi vasak pool võrdub nulliga, sest $a_y = 0$ ehk kiirusväli ei muutu ajas. Kuna rõhk ei muutu piki y-telge taandub ka see välja ning alles jääb:

$$0 = \rho g * sin\gamma + \mu\left[\frac{\partial^2 v}{\partial z^2}\right], \tag{23}$$

kus μ on dünamiline viskoossus ning komponentide ümber tõstmisel saadakse:

$$\frac{\partial^2 v}{\partial z^2} = -\frac{\rho g * sin\gamma}{\mu}. \tag{24}$$

Alles jäi *z*-telje suunaline komponent, sest antud töös arvutatakse kiirusprofiil piki z-telge. Võrrandit (24) integreeritakse kaks korda ning saadakse:

$$v(z) = -\frac{\rho g * sin\gamma}{2\mu} z^2 + C_1 z + C_2. \tag{25}$$

Piiritingimused määratakse järgmiselt: kui z=0, siis v=0 ning kui z=h, siis nihkepinge τ=0, sest õhus seda eiratakse:

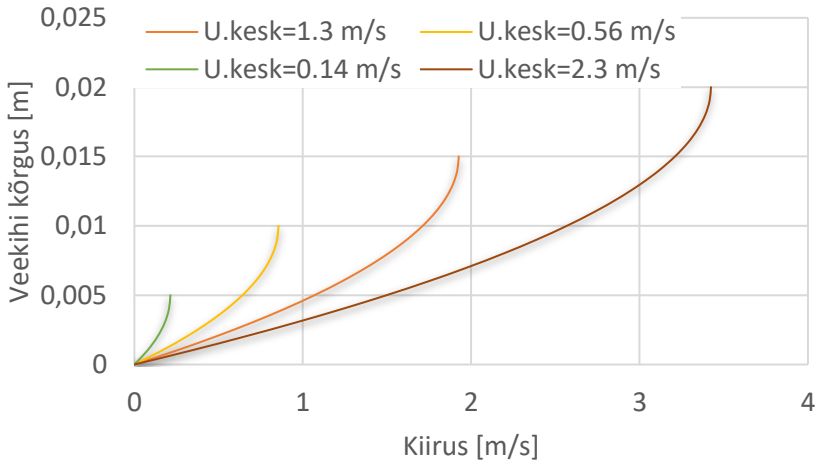
$$\mu\frac{\partial u}{\partial z} = 0. \tag{26}$$

Asendades piiritingimused võrrandisse (25), saadaksegi kiirusprofiili arvutamise valem ajahetkel:

$$v(z) = \frac{\rho g * sin\gamma}{2\mu}(2hz - z^2), \tag{27}$$

kus *h* tähistab veesamba kõrgust ja *z* koordinaati.

Välisvaatluse alusel hinnati samba kõrguseks 0.5 – 2 cm, mis vastas antud asukohas vihma intensiivsusele umbes 5mm tunnis (mõõdukas vihm). Sellele tehti arvutus ning leiti, et veekiirus domeenis võiks olla 1m/s.



Joonis 7. Kiirusprofiilid erinevate veesamba kõrguste juures.

Vastavalt hinnatud vedeliku samba kõrgusele ja selle põhjal valemit (27) kasutades leitud kiirusprofiili keskmisele hinnati mudeli ettevalmistamiseks ka Reynoldsi arvu väärtust vastavalt valemile (14), et selgitada mis tüüpi mudeliga peaks simulatsioonid läbi viima. Leiti, et kui veesamba kõrgus muutub vahemikus 0.5 – 2 cm, siis Reynoldi arvu väärtus varieerub vastavalt vahemikus 670 – 45 000. Vastavalt kriitilise Reynoldsi arvu väärtusele ei olnud seega simulatsioonide läbiviimise eel selge, kas õigem oleks rakendada laminaarse või turbulentse voolamise mudelit ning nende lahendusi otsustati võrrelda.

3.2.4 Tundlikkuse analüüs

Tundlikkuse analüüs tehti antud töös selleks, et võrrelda diskretiseerimisvigu ning valida välja sobivaim arvutusvõrk. Võre valimisel lähtuti CFD eesmärgist saavutada võimalikult täpne tulemus võimalikult väikese arvutusmahuga. Analüüsi tehti võre koonduvuse indeksi (ingl. *grid convergence index* – GCI) abil. GCI näitab protsendina, kui palju erineb arvutatud väärtus eeldatavast teoreetilisest väärtusest. See aitab kasutajal mõista, kui suur võib olla tema lahenduse viga võrreldes eeldatavaga ning aitab hinnata kui palju lahendus võib muutuda kui arvutusvõrk muudetakse täpsemaks [15]. See aitab avastada diskretiseerimisvigu, mis tekivad seetõttu, et arvutustes kasutatakse ligikaudsust. GCI arvutamiseks on vaja kolme erineva resolutsiooniga korrapäraselt täpsustatud arvutusvõrku ning mingit valitud parameetrit, mille lahendusi võrreldakse. Käesolevas töös valiti selleks parameetriks keskmine veekiirus ja veekihi kõrgus. GCI arvutamiseks on vaja leida vea parandamise järk p, mille saab arvutada järgmise valemiga:

$$p = \frac{log\left(\frac{\Phi_2 - \Phi_1}{\Phi_3 - \Phi_2}\right)}{log\, r}, \tag{28}$$

kus Φ tähistab lahenduse väärtust vastavas võrgustikus (alaindeksid 1, 2 ja 3 vastavad kõrgeima, keskmise ja madalaima resolutsiooniga võrkudele) ning r on parameeter, mis näitab mitu korda võrgu kvaliteeti parandati [15]. Soovituslik on genereerida võrk nii, et parameeter tuleks vähemalt r *=1.3…2*. Nende parameetrite abil sai arvutada GCI järgmise valemi põhjal [15]:

$$GCI_{32} \approx \frac{\Phi_3 - \Phi_2}{r_{32}^p - 1} \text{ ja } GCI_{21} \approx \frac{\Phi_2 - \Phi_1}{r_{21}^p - 1}. \tag{29}$$

Need arvutused tehti läbi vaadeldes kogu mudeli elementide arvu ja ühe elemendi pikkust (domeeni osas, kus vesi voolab, ehk maapinna lähedal) ning mõlema puhul arvutati lahenduste põhjal GCI keskmise kiiruse ja veekihi kõrguse alusel. Vastavad andmed, millega tundlikkuse analüüs läbi viidi ja tulemused on toodud Tabel 1. Seal on lisaks toodud ka turbulentse mudeli käivitamiseks loodud arvutusvõrgu (*) vastavad näitajad. Kuna viimane loodi nii, et ei järgitud sama süstematiseeritud täpsustamist ning ka kiirusväli ei ole arvutatud samadel alustel kui ülejäänud kolmel juhul, siis ei kasutatud selle võre parameetreid GCI arvutuses.

*Tabel 1. Võre parameetrid (Δy – maapinnaga külgneva elemendi küljepikkus, N – elementide arv arvutusdomeenis) ja vastavad keskmised kiirused (U) ja veesamba kõrgused (h) ning võre kvaliteedi järk (r), vea parandamise järk (p) ja võre koonduvuse indeksi (GCI) väärtused . Mudel * vastab turbulentse mudeli arvutusdomeeni võrele, mida ei kasutatud võre koonduvuse indeksi arvutusel.*

Mudel	***Δy,*** $[m]$	***N,*** $[-]$	***U,*** $[m/s]$	***h,*** $[m]$
1	0.068	2759685	1.61	0.0052
2	0.086	1443089	1.58	0.0068
3	0.108	761901	1.42	0.0073
*****	0.001	1621829	0.909	0.0092

Võre kvaliteedi järk	
$r_{21}=\Delta y_2/\Delta y_1$, $r_{32}=\Delta y_3/\Delta y_2$	1.3
$r_{21}=N_1/N_2$, $r_{32}=N_2/N_3$	1.9
Vea parandamise järk	
p (Δy,U)	-6.38
p (Δy,h)	4.43
p (N,U)	-2.61
p (N,h)	1.81

Võre koonduvuse indeks	
GCI₂₁ (Δy,U)	**0.037**
GCI₃₂ (Δy,U)	0.20
GCI₂₁ (Δy,h)	**0.00073**
GCI₃₂ (Δy,h)	0.00023
GCI₂₁ (N,U)	**0.037**
GCI₃₂ (N,U)	0.20
GCI₂₁ (N,h)	**0.00073**
GCI₃₂ (N,h)	0.00023

GCI arvutamisel kasutati nii elemendi küljepikkuse kui ka elementide arvude alusel arvutatud võre kvaliteedi parandamise järku p. Kuna töös tõsteti võre resolutsiooni süstemaatiliselt on ühe parameetri (kas *Δy* või *N*) alusel arvutatud erinevate parandamise järgud (r_{21} ja r_{32}) ligikaudu võrdsed. Tabelist 1 on näha, et GCI väheneb märgatavalt ja väärtus suurema resolutsiooni jaoks (muutus mudelite 2 ja 1 vahel) varieerub 0.01 – 4% (ehk viitab kuni 4% dikretiseerimisest tingitud veale) vahel. Kui lugeda keskmise resolutsiooniga võre sobivaks (ümardatult täpsematel võredel *U=1.6m/s*) siis leitud GCI vastab *~0.06 m/s* ja veesamba kõrgustest vastab GCI $5.92 \cdot 10^{-7}$ *m* kuni $4.36 \cdot 10^{-6}$ *m*. Kuna viga on kahe täpseima võre vahel piisavalt väike, siis võib GCI alusel otsustada, et edasiseks analüüsiks on sobiv mudel 2, et arvutusmahtu kokku hoida.

4 Tulemused ja arutelu

Järeltöötlust tehti *ParaView* abil, mis on juhtiv visualiseerimistarkvara, võimaldades andmete analüüsi läbiviimist, graafikute loomist ja andmete eksportimist vastavalt vajadusele [25]. Analüüsi käigus vaadeldi veekihi arenemist arvutusdomeenis läbi ajasammude. Lisaks eksporditi *ParaView*'st erinevalt filtreeritud andmestikke konkreetsetel ajahetkedel. Näiteks kiirusvälja andmestik, kus domeenist filtreeriti eelnevalt vedeliku fraktsiooni parameetri α abil välja ainult veekihile vastav väli, et arvutada ruumiliselt keskmistatud kiiruse väärtus. Lisaks pakkus huvi modelleerimise tulemusena saadud keskmine veekihi kõrgus. Viimast on lihtne erinevate punkti määramise või graafikute loomise võtetega määrata ühes konkreetses x ja y koordinaadiga määratud punktis. Samal ajal kuna maapind mudelis on langusega ning erinevate maapinna kumeruste tõttu varieeruv on keskmise kihi kõrguse määramine üle domeeni eriülesanne. Selleks töötati välja variant, mis tuleneb *OpenFOAM*'i eripärasest rõhu definitsioonist mitmefaasilise voolamise mudelite korral. *OpenFOAM* kasutab rõhu arvutamisel alternatiivset rõhu definitsiooni, lüües lahku hüdrostaatilise rõhu osa [14]:

$$p' = p - \rho(g \cdot h), \tag{30}$$

kus p' on hüdrostaatiline rõhk, mis tarkvaras kannab nime *p_rgh*. Avaldades valemist (30) kõrgus, leiti veekihi kõrguse arvutamise valem iga võreelemendi jaoks:

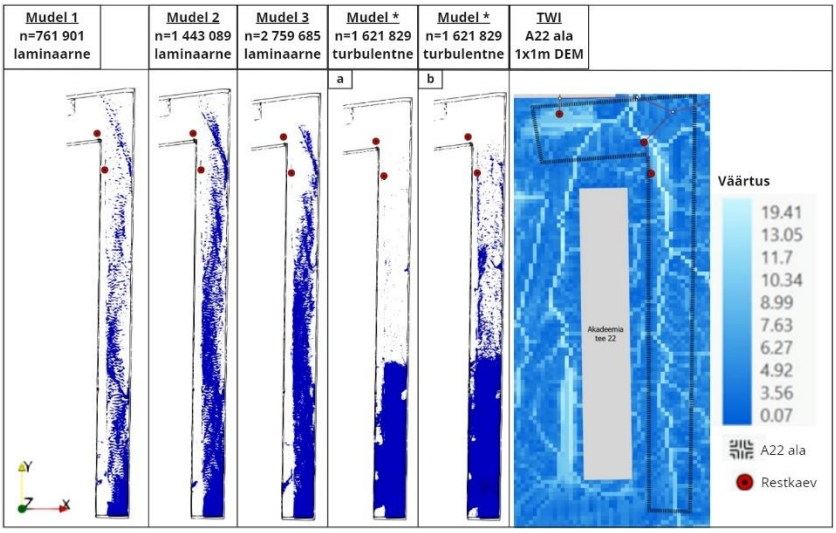
$$h = \frac{(p - p')}{\rho g}. \tag{31}$$

Arvutusteks valemi (31) põhjal pidi eraldi leidma elemendi sees olevate segunenud vedelike tiheduse:

$$\rho = \alpha \cdot p + (1 - \alpha) \cdot 1, \tag{32}$$

kus α on faasifraktsioon. Vastavalt valemitele (31) ja (32) mudeli 30. arvutussekundil arvutatud veekihi kõrguse andmed tee alal on leitavad tabelist 1. Keskmise resolutsiooniga arvutusvõrgul (*~1.4* miljonit elementi) andsid valemid tulemuseks, et domeeni keskmine veekihi kõrgus on 0.0068 *m*, mis langeb kokku välisvaatluste põhjal mudeli seadistamisel tehtud eeldusega, et veepinna kõrgus muutus vahemikus 0.5 – 2 *cm*.

Visualiseeritud vooluteed laminaarse ja turbulentse voolamise puhul olid täiesti erinevad. Joonisel 8 on kuvatud 30. ajasammul arenenud vooluteed erineva resolutsiooniga modelleeritud laminaarsel voolamisel ja kõrgema resolutsiooniga modelleeritud turbulentsel voolamisel. Esimeste mudelite puhul valiti järeltöötluse tarkvaras faasifraktsiooniks α=0.5 (Joonis 8 Mudelid 1-3, *a), et visualiseerida ainult veevoolu. Selleks, et turbulentsel voolamisel veevoolu elementides paremini kuvada, valiti võrdluseks faasifraktsiooniks α=0.1, mis võimaldas visualiseerida veevoolu ka elementides, kus oli ainult *10%* vett (Joonis 8 Mudel *b).

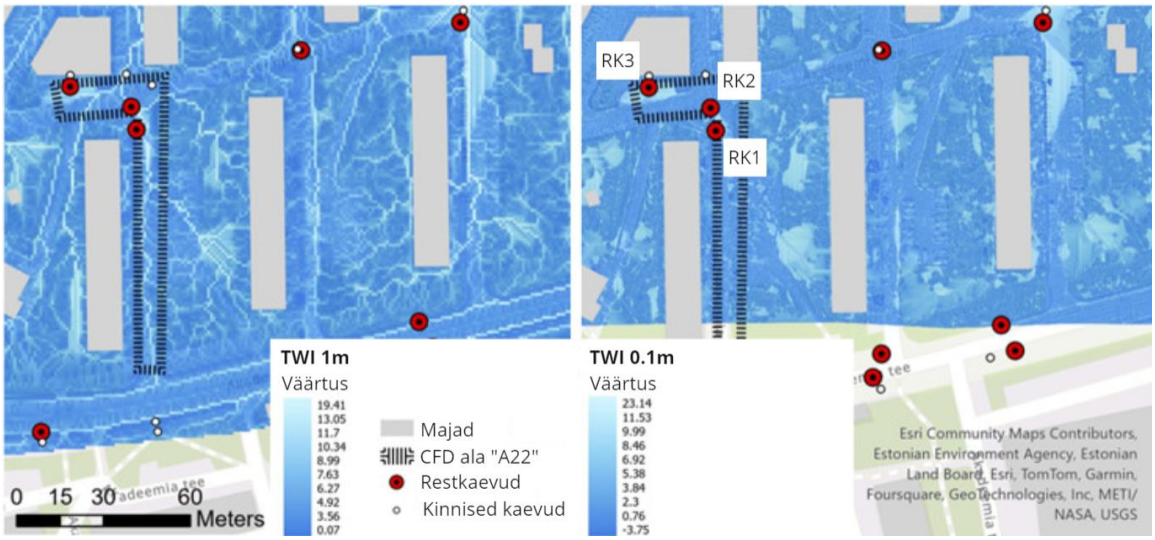


Joonis 8. Vooluteed mudelitel 1-3 visualiseeritud faasifraktsioonil α=0.5, mudelil 4a ja 4b vastavalt α=0.5 ja α= 0.1. Paremal TWI A22 alal.

Modelleerimistulemuste hindamiseks võrreldi tulemusi üleujutusriskide hindamisel laialdaselt kasutatava topograafilise niiskusindeksiga. Selleks arvutati *ArcGIS Pro* tarkvara abil topograafiline niiskusindeks (ingl. *topographic wetness index* – TWI). *ArcGIS Pro* on geoinfosüsteemide tarkvara, mis võimaldab hallata ruumiandmeid, visualiseerida neid kaartidel ning analüüsida vajalikke andmekogumeid [26]. TWI näitab suurtest vihmasadudest potentsiaalselt tekkivaid üleujutuskohti ja vooluteid vastavalt maapinna topograafiale [27] ning see leitakse digitaalse kõrgusmudeli andmete alusel. Aluseks võeti Maa-ameti geoportaalist leitav aerolaserskanneerimisega loodud maapinnamudel resolutsiooniga 1x1 m ja Hades Geodeesia droonilennu tulemusel loodud maapinnamudel 0.1x0.1 m resolutsiooniga (aluseks olnud punktipilve lahutus minimaalselt 1x1 cm võimaldas üleliigsete objektide eemaldamisel luua just sellise resolutsiooniga maapinnamudeli) ning nende põhjal arvutatud topograafilisi niiskusindekseid võrreldi omavahel. TWI leiti filtrite kombinatsiooniga mõlemale DEM-le järgneva valemi abil:

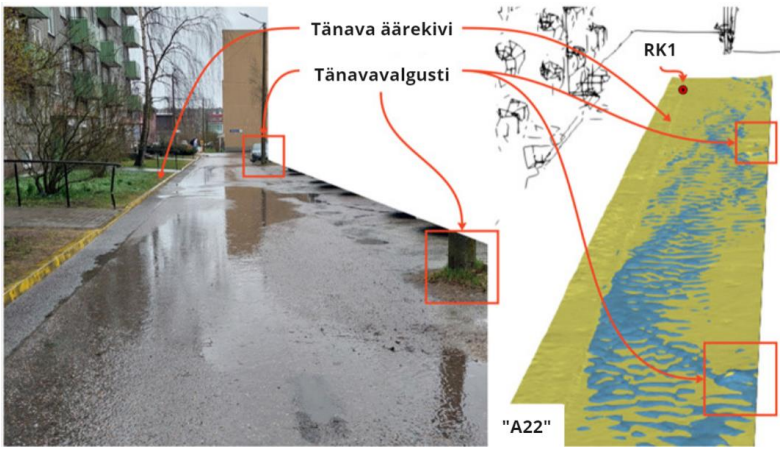
$$TWI = ln\frac{a}{tan\beta}, \tag{33}$$

kus a on ülesvooluala näitaja, mis näitab rastriruutude arvu mida mööda vesi voolab kindlasse ruutu ning $tan\beta$ on iga ruudu kalle [28]. Maa-ameti lehelt saadud 1x1m resolutsiooniga alal on vooluteed selgelt eristatavad (Joonis 9). Vooluteed näitavad vee voolamist restkaevudesse ja teistesse äravoolualadesse, nt muruplatsid. Tihedama resolutsiooniga 0.1x0.1m alal pole vooluteed eristatavad. See langeb kokku teadmisega, et maapinna mudelit järjest täpsustades ei ole võimalik TWI arvutust täpsustada, kuna teatud kriitilisest piirist alates (mis sõltub konkreetsest maa alast) muutuvad tulemused liiga killustatuks [28]. Siiski langevad madalpunktidele viitavad veekogunemise alad erinevate resolutsioonide vahel suures osas kokku ning täpsemal juhul on neid lihtsam eristada.



Joonis 9. TWI erineva eraldusvõimega kõrgusandmete jaoks. RK1-RK3 restkaevud, mida käesolevas töös modelleeriti. Vasakul TWI arvutatud resolutsiooniga 1x1m DEM-il, paremal 0.1x0.1m [11].

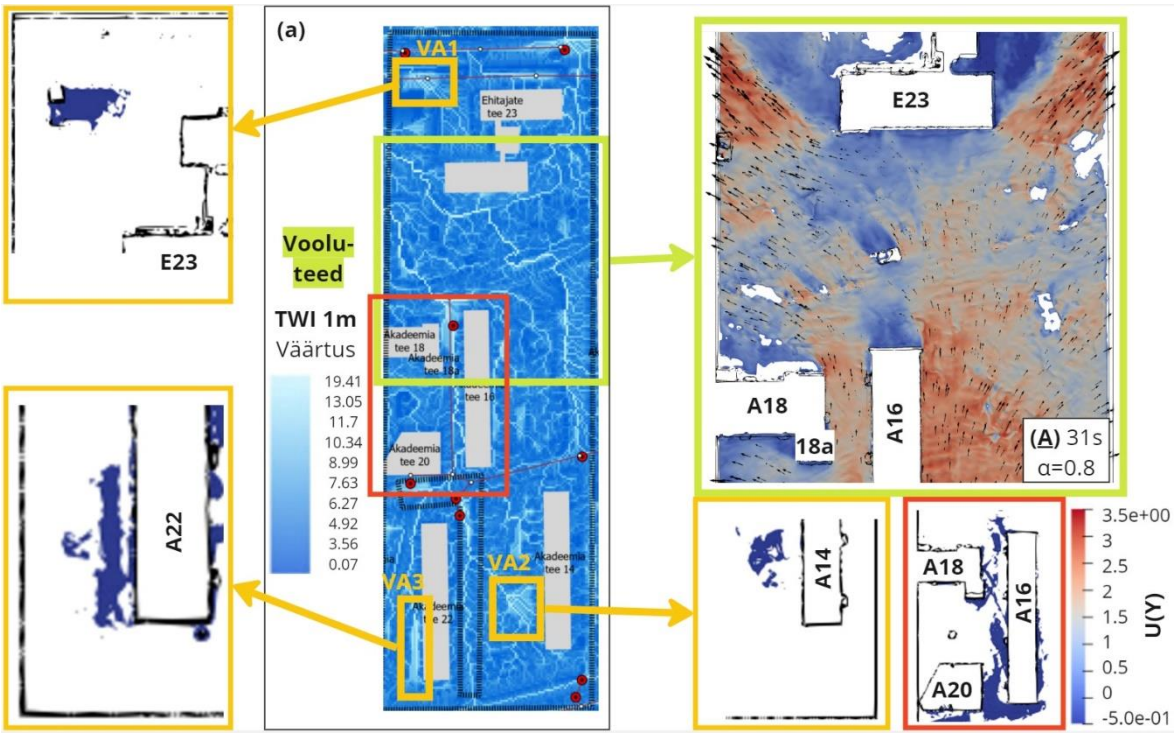
Välisvaatlusel pildistatud Akadeemia tee 22 tänavat ja A22 mudelit võrreldes on võimalik visuaalselt hinnata, et vooluteed on sarnased. Selle põhjal võib järeldada, et koostatud CFD mudel on sellel alal usaldusväärne. Kui 1x1m DEM-ile arvutatud TWI näitab, et voolutee on suunatud otse restkaevu (Joonis 9), siis CFD mudel näitab vooluteed, mis möödub looklevalt restkaevust, mida kinnitab ka välisvaatlus mõõduka vihma korral (Joonis 10) ning restkaevuni jõuab vesi alles siis kui üle tee paiknevas madalpunktis on moodustunud kriitilise sügavusega lomp, kust hakkab vesi üle kõrgema tee keskosa restkaevu voolama. Selle põhjal saab järeldada, et CFD mudelid annavad reaalsetest protsessidest õigema ülevaate kui TWI. Samas võib see ka olla seotud pinnase vajumisega või esialgset vertikaalplaneeringut mõjutanud tee parandustöödega DEM loomisele aluseks olnud aerolaserskaneerimise mõõdistuse hetke ja vaatluse hetke vahepealsel ajal.



Joonis 10. Vasakul pilt tänavast (sademetel 1.7-2.5mm/h), mida CFD-ga modelleeriti A22-l (Akadeemia tee 22), paremal ParaView vaade A22 koos restkaevuga 1 (RK1) [11].

Suurema maa-ala simulatsiooni tulemusi visualiseerides nähti sarnasusi *ArcGIS Pro* abil tehtud niiskusindeksiga. Näiteks määrasid mõlemad lähenemised küllaltki täpselt madalpunktid ning CFD modelleerimine kinnitas TWI voolusuundi osadel aladel. Eriti hästi joonistusid välja vee kogunemise alad VA(1-3) (Joonis 11), kus simulatsiooni 90. ja 120. sekundil need vastavatel faasifraktsioonidel α selgemini jälgitavaks muutusid. Samuti on võimalik tuvastada vooluteed kahel pool Ehitajate tee

23 (E23) maja nii simulatsioonides kui ka TWI-l. Kaarti analüüsides näeb, et VA1 paikneb kõvakattega alal, kus ei ole restkaevu ehk ei ole teada, kuidas vesi sealt kanalisatsiooni jõuab. Veel leiti, et näiteks Akadeemia tee 16, 18, 20 (A16, A18, A20) ja E23 hoonete soklite äärtes või vahetus läheduses toimub oluline voolamine (Joonis 11 VA3 suuremal joonisel ja ala punases kastis), mida TWI-lt ei ole võimalik tuvastada.

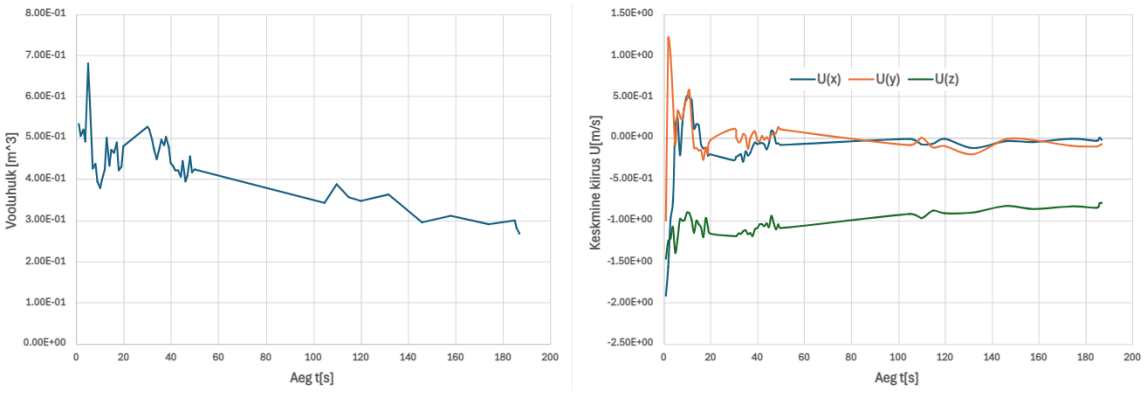


Joonis 11. Kastis (a) TWI 1x1m resolutsiooniga DEM-i peal, kollastes kastides näidatud vee-alad 1-3 (VA1-VA3), rohelises kastis näidatud voolusuunad ühel piirkonnal, punases kastis näidatud sokli äärtesse tekkivaid vee-alasid.

Kõige väiksemal modelleeritud alal (Joonis 4 (D)) arvutati kokkuvoolu aeg *T* vastavalt standardile. Kuni 4ha aladele võib kasutada näiteks Kerby (1959) valemit:

$$T_c = 1.44\left(\frac{Lr}{S^{\frac{1}{2}}}\right)^{0.467}, \tag{34}$$

kus *L* on voolutee pikkus valgalal [m], *r* on valgala tegur (kõvakattega pinna jaoks on nt. 0.02 ja kõva mullapinna jaoks 0.1), *S* on valgala kalle [m/m] [29]. Kerby valemile vastavalt on vaadeldud ristmikualal hinnanguline kokkuvoolu aeg 2min. CFD simulatsioon näitas, et arvutusdomeeni algajahetkel vabastatud $15m^3$ vett väljus domeeni keskosas paiknenud kaevu kaudu umbes 190 sekundi jooksul, kuid 100. sekundiks oli kogu veemaht kokku voolanud domeeni madalpunkti kaevu läheduses. Seega toimub simulatsioonis kokkuvoolamine kiiremini kui Kerby valemi järgi, kuid täielik väljavool domeenist võtab viimasest kauem aega. Domeeni alguses vabastatud suur kogus vett langes maapinnale, mis tekitas lainetuse ning umbes 50. sekundist on voolamine rahunenud ja muutub stabiilsemaks (Joonis 12). Tuleb märkida, et Joonisel 12 esitatud vooluhulk hõlmab nii vett kui õhku, seega mõjutab graafikut ka õhuvool läbi domeeni väljavoolu.



Joonis 12. Vasakul vooluhulk, mis väljub domeenist läbi restkaevu (nii vesi kui õhk), paremal koordinaattelgede suunalised voolukiirused, kusjuures z-telje suunaline kiirus on negatiivne, sest liigub domeenist välja.

Kuna diskretiseerimisviga näiteks veesamba kõrgusel oli kordades väiksem kui 1mm, siis võib järeldada, et kasutatud VoF meetod toimib väga hästi. Siiski tuleb suhtuda viimasesse kriitiliselt, sest veesamba kõrguse tuvastamine võis olla raskendatud, sest element oli kohati suurem kui veepinna kõrgus.

Tehtud töö põhjal võib järeldada, et CFD simulatsioonid on väikeste alade, näiteks üksikute teealade või kinnistute uurimisel kasulikud kuna annavad üksikasjalikku teavet vee voolamise kohta keerulistel pindadel, võimaldades saada informatsiooni, mida lihtsustatud mudelid ei võimalda. Suurte alade puhul ei ole CFD simulatsioonid aga praktilised, sest need nõuavad suuri arvutusvõimsusi. Näiteks arvutati Joonisel 11 toodud tulemusi TalTechi teadusarvutuse keskuse klastril paralleelarvutusega 50 tuuma peal mitu päeva. Samas ristmikuala mudeli arvutusel võttis ühe arvutussekundi loomine 17. tuumal aega umbes ühe sekundi reaalajas. Viimasele lisandub arvutuse lõpus paralleelarvutuse tulemusel loodud tuumakataloogide rekonstrueerimine terviklikeks ajakataloogideks.

Käesoleva töö üheks olulisemaks tulemuseks on töövoo koostamine, kirjeldamaks kuidas kasutada droonimõõdistustel saadud kõrge eraldusvõimega punktipilve arvutusliku vedelike dünaamikaga modelleerimiseks. Antud töö annab olulistest sammudest ja metoodikast põhjaliku ülevaate.

5 Tänuavaldused

Käesoleva töö autor tänab eelkõige töö juhendajat Katrin Kauri, kes oli alati toeks ja abivalmis selgitamaks tekkivaid küsimusi ning valmis arutlema erinevate võimaluste üle, kuidas tööd paremaks muuta. Samuti tänab autor ka konstruktsiooni- ja vedelikumehaanika uurimisrühma liikmeid Ivar Annust ja Murel Truud, kellelt saadi ideid ja tuge takistuste ületamiseks ning probleemide lahendamiseks.

Simulatsioonid viidi osaliselt läbi Tallinna Tehnikaülikooli teadusarvutuste keskuse arvutusklastril. Uurimistöö läbiviimist on toetanud Euroopa Komisjon grandilepinguga Life 20 IPC/EE/000010 ja Eesti Teadusagentuur grandilepinguga PRG667.

6 Allikad

[1] J. Jaagus, „Climatic changes in Estonia during the second half of the 20th century in relationship with changes in large-scale atmospheric circulation“, *Theor. Appl. Climatol.*, kd 83, nr 1, lk 77–88, jaan 2006, doi: 10.1007/s00704-005-0161-0.

[2] T. Tamm, O. Tamm, ja E. Saaremäe, „Sademeveesüsteemide projekteerimise aluste kaasajastamine : SA Keskkonnainvesteeringute Keskuse ja Eesti Maaülikooli vahel 28.05.2019 sõlmitud lepingu nr. 3-2_3/8691-6/2018 lõpparuanne“, 2020, [Online]. http://hdl.handle.net/10492/6288

[3] A. Luhamaa, A. Männik, A. Kallis, K. Mändla, T. Pedusaar, ja K. Rosin, „Eesti tuleviku kliimastsenaariumid aastani 2100“, 2015.

[4] IPCC, „Climate Change 2022: Impacts, Adaptation, and Vulnerability. Contribution of Working Group II to the Sixth Assessment Report of the Intergovernmental Panel on Climate Change“, 2022.

[5] J. Laanearu ja A. Piirsalu, *Numbrilised arvutusskeemid voolamise hüdraulikas*. Tallinn: Tallinna Tehnikaülikool, 2022.

[6] „EVS 848:2021, Väliskanalisatsioonivõrk“, EVS. [Online]. https://www.evs.ee/et/evs-848-2021

[7] M. Truu, I. Annus, J. Roosimägi, N. Kändler, A. Vassiljev, ja K. Kaur, „Integrated Decision Support System for Pluvial Flood-Resilient Spatial Planning in Urban Areas“, *Water*, kd 13, nr 23, lk 3340, nov 2021, doi: 10.3390/w13233340.

[8] Research Executive Agency., *Digitalisation in the water sector recommendations for policy developments at EU Level.* LU: Publications Office, 2022. [Online]. https://data.europa.eu/doi/10.2848/915867

[9] M. Abily, C. M. Duluc, J. B. Faes, ja P. Gourbesville, „Performance assessment of modelling tools for high resolution runoff simulation over an industrial site“, *J. Hydroinformatics*, kd 15, nr 4, lk 1296–1311, okt 2013, doi: 10.2166/hydro.2013.063.

[10] K. Laigna, *Hüdromehaanika*. Eesti Merehariduskeskus, Tallinn, 1997.

[11] K. Kaur, I. Annus, M. Truu, N. Kändler, ja I. Paalmäe, „Integrating Drone-captured Sub-catchment Topography with Multiphase CFD Modelling to Enhance Urban Stormwater Management“, esitatud 3rd International Joint Conference on Water Distribution Systems Analysis ja Computing and Control for the Water Industry (WDSA/CCWI 2024), Ferrara, Italy, Esitatud.

[12] F. Moukalled, L. Mangani, ja M. Darwish, *The Finite Volume Method in Computational Fluid Dynamics: An Advanced Introduction with OpenFOAM® and Matlab*, kd 113. Fluid Mechanics and Its Applications, vol. 113. Cham: Springer International Publishing, 2016. doi: 10.1007/978-3-319-16874-6.

[13] T. Holzmann, *Mathematics, numerics, derivations and OpenFOAM®*, 7.0. 2019. [Online]. https://holzmann-cfd.de

[14] „User Guide“. [Online]. https://www.openfoam.com/documentation/user-guide

[15] J. H. Ferziger ja M. Perić, *Computational methods for fluid dynamics*, Third, rev. Edition. Berlin Heidelberg New York: Springer, 2002.

[16] Y. Iso ja X. Chen, „Flow Transition Behavior of the Wetting Flow Between the Film Flow and Rivulet Flow on an Inclined Wall“, *J. Fluids Eng.*, kd 133, nr 9, lk 091101, sept 2011, doi: 10.1115/1.4004765.

[17] „Turbulence modeling -- CFD-Wiki, the free CFD reference“. [Online]. https://www.cfd-online.com/Wiki/Turbulence_modeling

[18] B. E. Launder ja D. B. Spalding, „The numerical computation of turbulent flows“, *Comput. Methods Appl. Mech. Eng.*, kd 3, nr 2, lk 269–289, märts 1974, doi: 10.1016/0045-7825(74)90029-2.

[19] K. Kaur, „Dynamic Processes of Air-Water Flows in Urban Water Systems“.

[20] „OpenFOAM“. [Online]. https://www.openfoam.com/

[21] „MeshLab“. [Online]. https://www.meshlab.net/

[22] „SALOME PLATFORM - The open-source platform for numerical simulation“, SALOME PLATFORM. [Online]. https://www.salome-platform.org/

[23] M. Garland ja P. S. Heckbert, „Surface Simplification Using Quadric Error Metrics“.

[24] M. Kazhdan ja H. Hoppe, „Screened poisson surface reconstruction“, *ACM Trans. Graph.*, kd 32, nr 3, lk 1–13, juuni 2013, doi: 10.1145/2487228.2487237.

[25] „ParaView - Open-source, multi-platform data analysis and visualization application“. [Online]. https://www.paraview.org/

[26] „Desktop GIS Software | Mapping Analytics | ArcGIS Pro“. [Online]. https://www.esri.com/en-us/arcgis/products/arcgis-pro/overview

[27] M. Kopecký, M. Macek, ja J. Wild, „Topographic Wetness Index calculation guidelines based on measured soil moisture and plant species composition“, *Sci. Total Environ.*, kd 757, lk 143785, veebr 2021, doi: 10.1016/j.scitotenv.2020.143785.

[28] A. O. Altunel, „The effect of DEM resolution on topographic wetness index calculation and visualization: An insight to the hidden danger unraveled in Bozkurt in August, 2021“, *Int. J. Eng. Geosci.*, kd 8, nr 2, lk 165–172, juuli 2023, doi: 10.26833/ijeg.1110560.

[29] W. S. Kerby ja J. M. Asce, „Time of concentration for overland flow“. 1959.

7 Annotatsioon

Käesolevas töös tegeleti sademevee modelleerimisega linnalistel valgaladel, milleks kasutati droonifotogrammeetria tulemusel loodud kõrge eraldusvõimega punktipilve sisendina arvutusliku vedelike dünaamika mudeli geomeetria loomiseks. Eesmärk oli kaardistada võimalused loomaks töövoog linna sademevee pindmise äravoolu kõrge eraldusvõimega modelleerimiseks, et anda sisend linna sademeveemajanduse parandamiseks. Mudeli simulatsiooni tulemuste usaldusväärsuse hindamiseks kasutati võrdlust erinevate meetoditega ja arvutusvõrgu tundlikkuse analüüsiga.

Töö esimene peatükk kirjeldas numbrilise modelleerimise aluseid ja selgitas kasutatavaid võrrandeid ning tutvustas algoritme ja parameetreid, millega probleeme lahendati. Teises peatükis anti ülevaade kasutatud metoodikast, tutvustati *OpenFOAM* tarkvara, kirjeldati eeltöötlust, valiti simulatsiooniks sobiv arvutusvõre ning leiti domeeni sisendiks sobiv kiirus. Kolmandas peatükis analüüsiti CFD tulemusi võrdluses muude meetoditega, sh kõrgusmudelil baseeruv topograafiline niiskusindeks, välisvaatlused ja levinud väikeste valgalade kokkuvoolu aega kirjeldav seos.

Tulemuste põhjal järeldati, et sademevee äravoolu on arvutusliku vedelike dünaamika meetodil mõistlik modelleerida kinnistupõhiselt või üksikutel kinnistute gruppidel, et täpsustada lokaalsed sademeveesüsteemide probleemid, aga mitte suurtel aladel linnapõhiselt. Töö käigus töötati välja töövoog erinevate tarkvarade, protsesside ja valemite kombineerimiseks droonimõõdistuste alusel CFD mudeli seadistamiseks. Töö eesmärk loetakse saadud tulemuse põhjal täidetuks.

8 Abstract

This thesis focused on modelling stormwater runoff in urban catchment areas using a high-resolution point cloud generated by drone photogrammetry as an input for creating geometry of a computational fluid dynamics (CFD) model. The aim was to explore the possibilities of establishing a workflow for high-resolution modelling of urban stormwater surface runoff to provide input for improving urban stormwater management. To evaluate the reliability of the model simulation results, comparisons with different methods and sensitivity analysis of the computational grid were used.

The first chapter described the fundamentals of numerical modelling, explained the equations in use and introduced the algorithms and parameters used to solve the problems. The second chapter provided an overview of the methodology, introduced the *OpenFOAM* software, described the preprocessing steps, selected an appropriate computational grid for the simulation, and determined a suitable velocity for the domain input. The third chapter analyzed the CFD results in comparison with other methods, including the topographic wetness index based on digital elevation model, field observations and a common empirical relationship describing the time of concentration for small catchment areas.

Based on the results, it was concluded that stormwater runoff can be reasonably modelled using the CFD method on small properties or for individual groups of properties to refine local stormwater system issues, but not on a large-scale urban areas. During the work, a workflow was developed for combining different software, processes, and formulas to set up a CFD model based on drone survey data. The purpose of the work is considered fulfilled based on the results obtained.

9 Lisad

Lisa 1 */blockMeshDict*

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.0.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      blockMeshDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

convertToMeters 1;

vertices   // (x y z)
(
    (-155 -237 37)  // 0
    (-106 -237 37)  // 1
    (-106  -129 37) // 2
    (-155 -129 37)  // 3
    (-155 -237 44)  // 4
    (-106 -237 44)  // 5
    (-106  -129 44) // 6
    (-155 -129 44   // 7
);

blocks (hex (0 1 2 3 4 5 6 7) (61 125 9) simpleGrading (1 1 1));

edges ();

patches ();

mergePatchPairs ();

// ************************************************************************* //
```

Lisa 2 */snappyHexMeshDict*

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.2.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile {version     2.0;
          format      ascii;
          class       dictionary;
          object      snappyHexMeshDict;}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
castellatedMesh true; snap true; addLayers false;

geometry // Siia kirjutatakse kõik STL failid ning määratakse piirkonnad, mis vajavad täpsustamist
{
   teeWallLeft.stl {type triSurfaceMesh; name teeWallLeft;}
   wallLow.stl {type triSurfaceMesh; name wallLow;}
   wallRight.stl {type triSurfaceMesh; name wallRight;}
   atmosphere.stl {type triSurfaceMesh; name atmosphere;}
   inlet.stl {type triSurfaceMesh; name inlet;}
   outlet1.stl {type triSurfaceMesh; name outlet1;}
   outlet2.stl {type triSurfaceMesh; name outlet2;}
   outlet3.stl {type triSurfaceMesh; name outlet3;}

   refinementBox1 {type searchableBox; min (-156 -238 36); max (-105 -128 40.3);}
   refinementBox2 {type searchableBox; min (-156 -238 36); max (-105 -128 41.3);}
   refinementBox3 {type searchableBox; min (-156 -238 36); max (-105 -128 45);}
   refinementBox4 {type searchableBox; min (-112.5 -236 39.5); max (-107.5 -234 40.2);}
   refinementBox5 {type searchableBox; min (-135.86 -134.8 36); max (-135.66 -134.6 39);}
};

castellatedMeshControls
{
          maxLocalCells 20000000;
          maxGlobalCells 30000000;
          minRefinementCells 10;
          maxLoadUnbalance 0.10;
          nCellsBetweenLevels 2;

features ({file "teeWallLeft.eMesh"; level 1;}
            {file "wallLow.eMesh"; level 0;}
            {file "wallRight.eMesh"; level 0;}
            {file "atmosphere.eMesh"; level 0;}
```

```
        {file "inlet.eMesh"; level 1;}
        {file "outlet1.eMesh"; level 1;}
        {file "outlet2.eMesh"; level 1;}
        {file "outlet3.eMesh"; level 1;});

refinementSurfaces {teeWallLeft {level (1 1);}
                    wallLow {level (1 1);}
                    wallRight {level (1 1);}
                    atmosphere {level (1 1);}
                    inlet {level (3 3);}
                    outlet1 {level (0 0);}
                    outlet2 {level (0 0);}
                    outlet3 {level (0 0);}}

 resolveFeatureAngle 30;

 refinementRegions  {refinementBox1 {mode inside; levels ((3 3));}
                     refinementBox2 {mode inside; levels ((1 1));}
                     refinementBox3 {mode inside; levels ((0 0));}
                     refinementBox4 {mode inside; levels ((5 5));}
                     refinementBox5 {mode inside; levels ((4 4));}}

  locationInMesh (-111 -220 40.2);

  allowFreeStandingZoneFaces true;
  }

snapControls {
        nSmoothPatch 3; tolerance 4.0; nSolveIter 30; nRelaxIter 5; nFeatureSnapIter 15;
        implicitFeatureSnap false;
        explicitFeatureSnap true;
        multiRegionFeatureSnap false;
            }

addLayersControls
{
         relativeSizes false;
         layers { }
         expansionRatio 1.3;
         finalLayerThickness 0.00016;
         minThickness 0.00008;
         nGrow 0;
         featureAngle 80;
         nRelaxIter 3;
         nSmoothSurfaceNormals 1;
```

```
        nSmoothNormals 3;
        nSmoothThickness 10;
        maxFaceThicknessRatio 0.5;
        maxThicknessToMedialRatio 0.3;
        minMedianAxisAngle 130;
        nBufferCellsNoExtrude 0;
        nLayerIter 50;
}

meshQualityControls
{
        maxNonOrtho 65;
        maxBoundarySkewness 20;
        maxInternalSkewness 4;
        maxConcave 80;
        minFlatness 0.5;
        minVol 1e-13;
        minTetQuality 1e-9;
        minArea -1;
        minTwist 0.02;
        minDeterminant 0.001;
        minFaceWeight 0.02;
        minVolRatio 0.01;
        minTriangleTwist -1;
        nSmoothScale 4;
        errorReduction 0.75;
}

debug 0;

// ************************************************************************* //
```

Lisa 3 */alpha.water*

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  v1812                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      alpha.water;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 0 0 0 0 0 0];

internalField   nonuniform 0;

boundaryField {teeWallLeft    {type zeroGradient;}
                    wallLow         {type inletOutlet; inletValue uniform 0; value uniform 0;}
                    wallRight       {type inletOutlet; inletValue uniform 0; value uniform 0;}
                    atmosphere      {type inletOutlet; inletValue uniform 0; value uniform 0;}
                    inlet           {type inletOutlet; inletValue uniform 1; value uniform 1;}
                    outlet1         {type inletOutlet; inletValue uniform 0; value uniform 0;}
                    outlet2         {type inletOutlet; inletValue uniform 0; value uniform 0;}
                    outlet3         {type inletOutlet; inletValue uniform 0; value uniform 0;}
                    }


// ************************************************************************* //
```

Lisa 4 */p_rgh*

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  v1812                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile { version     2.0;
           format      ascii;
           class       volScalarField;
           location    "0";
           object      p_rgh;}

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [1 -1 -2 0 0 0 0]; // OpenFOAM defineerib ühikud SI-süsteemi põhiühikute kaudu [kg m s K mol A cd]

internalField   uniform 0;

boundaryField {
        teeWallLeft     {type fixedFluxPressure; gradient uniform 0; value uniform 0;}

        wallLow         {type totalPressure; rho rho; psi none; gamma 1;
                         p0 uniform 0; value uniform 0;}

        wallRight       {type fixedFluxPressure; gradient uniform 0; value uniform 0;}

        atmosphere      {type totalPressure; rho rho; psi none; gamma 1;
                         p0 uniform 0; value uniform 0;}

        inlet           {type zeroGradient;}

        outlet1         {type totalPressure; rho rho; psi none; gamma 1;
                         p0 uniform 0; value uniform 0;}

        outlet2         {type totalPressure; rho rho; psi none; gamma 1;
                         p0 uniform 0; value uniform 0;}

        outlet3         {type totalPressure; rho rho; psi none; gamma 1;
                         p0 uniform 0; value uniform 0;}
            }

// ************************************************************************* //
```

Lisa 5 */U*

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  v1812                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
   version     2.0;
   format      ascii;
   class       volVectorField;
   location    "0";
   object      U;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 1 -1 0 0 0 0]; // [kg m s K mol A cd]

internalField   uniform (0 0 0);

boundaryField {
        teeWallLeft  {type fixedValue;                       value uniform (0 0 0);}

        wallLow      {type pressureInletOutletVelocity; value uniform (0 0 0);}

        wallRight    {type fixedValue;                       value uniform (0 0 0);}

        atmosphere {type pressureInletOutletVelocity; value uniform (0 0 0);}

        inlet        {type fixedValue;                       value uniform (0 1 0);}

        outlet1      {type pressureInletOutletVelocity;  value uniform (0 0 0);}

        outlet2      {type pressureInletOutletVelocity;  value uniform (0 0 0);}

        outlet3      {type pressureInletOutletVelocity;  value uniform (0 0 0);}
                }


// ************************************************************************* //
```

Lisa 6 */controlDict*

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  v1812                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{   version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      controlDict;   }

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

application     interFoam;
startFrom       latestTime;
startTime       0;
stopAt          endTime;
endTime         30;
deltaT          0.001;
writeControl    adjustableRunTime;
writeInterval   0.1;
purgeWrite      0;
writeFormat     ascii;
writePrecision  6;
writeCompression off;
timeFormat      general;
timePrecision   6;
runTimeModifiable yes;
adjustTimeStep  yes;
maxCo           1;
maxAlphaCo      1;
maxDeltaT       1;

// ************************************************************************* //
```

Lisa 8 */fvSchemes*

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  v1812                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvSchemes;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

ddtSchemes              {default        Euler;}

gradSchemes             {default        Gauss linear;}

divSchemes              {div(rhoPhi,U)  Gauss linearUpwind grad(U);
                         div(phi,alpha)  Gauss vanLeer;
                         div(phirb,alpha) Gauss linear;
                         div(((rho*nuEff)*dev2(T(grad(U))))) Gauss lineaar;}

laplacianSchemes        {default        Gauss linear corrected;}

interpolationSchemes {default        linear;}

snGradSchemes           {default        corrected;}


// ************************************************************************* //
```

Lisa 9 */fvSolution*

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  v1812                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvSolution;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

solvers {"alpha.water.*" {nAlphaCorr 2; nAlphaSubCycles 1; cAlpha 1; MULESCorr yes;
                          nLimiterIter 5; solver smoothSolver; smoother symGaussSeidel;
                          tolerance 1e-8; relTol 0;}

         "pcorr.*"        {solver PCG; preconditioner DIC; tolerance 1e-5; relTol 0;}

         p_rgh            {solver PCG; preconditioner DIC; tolerance 1e-07; relTol 0.05;}

         p_rghFinal       {$p_rgh; relTol 0;}

         U                {solver smoothSolver; smoother symGaussSeidel;
                           tolerance 1e-06; relTol 0;}
        }

PIMPLE {momentumPredictor no; nOuterCorrectors 1;
        nCorrectors 3; nNonOrthogonalCorrectors 0;}

relaxationFactors {equations {".*" 1;}}


// ************************************************************************* //
```

Lisa 11 */boundary*

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  v1812                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       polyBoundaryMesh;
    location    "constant/polyMesh";
    object      boundary;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

8
(   teeWallLeft    {type wall; inGroups 1(wall); Faces 151724; startFace 4296758;}

    wallLow         {type patch; nFaces 8749; startFace 4448482;}

    wallRight       {type wall; inGroups 1(wall); nFaces 13931; startFace 4457231;}

    atmosphere    {type patch; nFaces 7408; startFace 4471162;}

    inlet           {type patch; nFaces 1434; startFace 4478570;}

   outlet1          {type patch; nFaces 16; startFace 4480004;}

   outlet2          {type patch; nFaces 16; startFace 4480020;}

   outlet3          {type patch; nFaces 9; startFace 4480036;} )

// ************************************************************************* //
```

Lisa 13 */transportProperties*

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  v1812                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      transportProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

phases (water air);

water {transportModel Newtonian; nu 1e-06; rho 1000;}

air       {transportModel Newtonian; nu 1.48e-05; rho 1;}

sigma  0.07; //pindpinevus

// ************************************************************************* //
```