Tartu Ülikool

Loodus- ja Tehnoloogiateaduskond

Füüsika Instituut

Indrek Ploom

AEROSOOLI TEKKE MODELLEERIMISPROGRAMMI LISATARKVARA

Bakalaureuse töö

Juhendajad: Teadur, füüs-mat knd, Jaan Salm

Teadur, PhD, Aare Luts

Tartu 2012

Sisukord

1. Sissejuhatus

1.1 Ülevaatlikult aerosoolidest

Aerosool on kahefaasiline dispersne süsteem, mis koosneb gaasilisest keskkonnast selles hõljuvate tahkete ja/või vedelate osakestega. Ka atmosfääriõhk on aerosool, kus gaasilises keskkonnas (dispersioonikeskkonnas) hõljuvad erinevad vedelad ja/või tahked osad (dispersne faas). Nende aerosooliosakeste mass võrreldes kogu atmosfääri massiga on kaunis tühine. Sellegipoolest võib atmosfääriaerosool avaldada märgatavat mõju ilmastiku ja kliima kujunemisele.

Aerosooli võib klassifitseerida näiteks agregaatoleku järgi: kui gaasis on vedelad osakesed, siis nimetatakse sellist aerosooli uduks. Kui tahked osakesed, siis suitsuks. Samuti võib aerosoole jagada tekkeviisi alusel primaarseteks (otse allikatest paiskuvad osakesed) ja sekundaarseteks (tekivad erinevate atmosfäärigaaside reageerimisel ja kondenseerumisel või jahtuva gaasi kondenseerumisel).

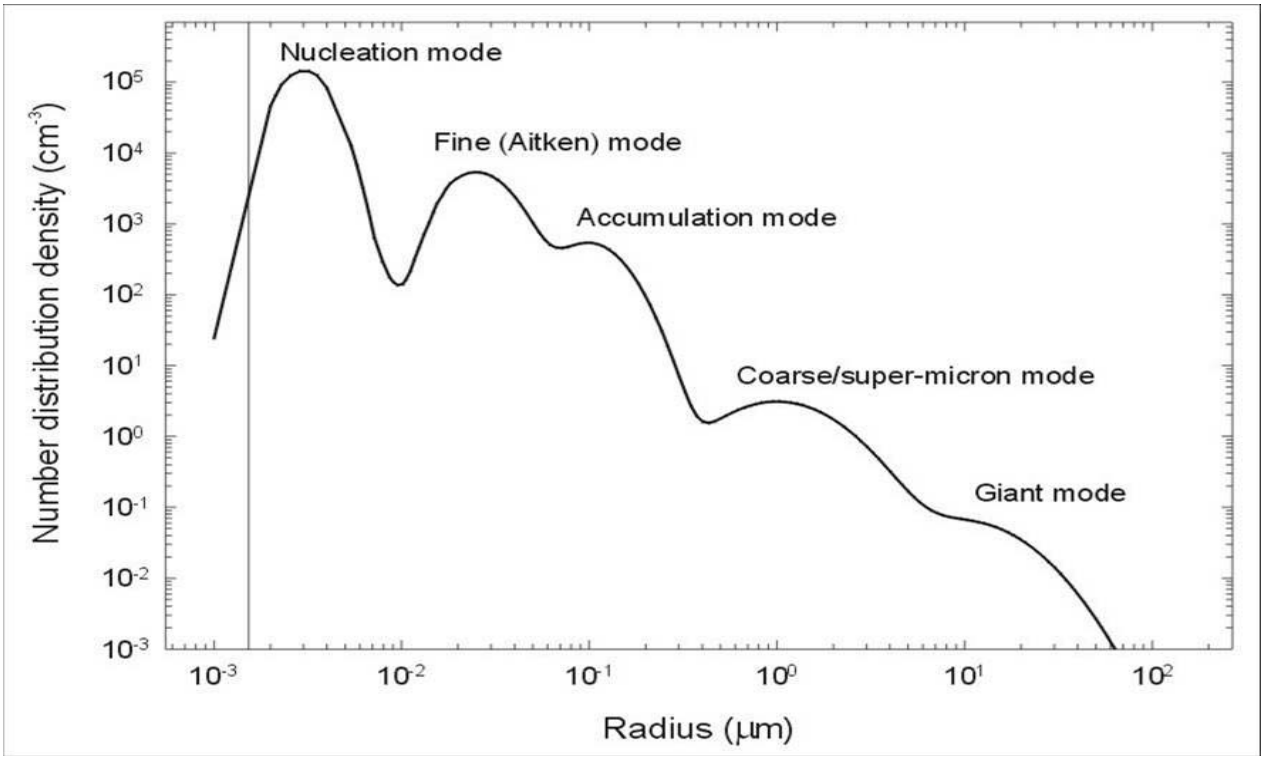
Kolmas viis aerosoole klassifitseerida on tekke-ja emiteerimisallikate alusel kaheks: looduslikud (tuule-erosioon, vulkaanipursked, merevee piiskade aurumine) ja inimtekkelised (põlemisprotsessid, keemiatööstus, metallurgia, kaevandamine, põlluharimine, transport, sõda). Õhu aerosoolid pärinevad valdavalt looduslikest allikatest ja inimtekkelisi on ca 20%.

Aerosooliosakeste mõõtmed varieeruvad nanomeetrist (10^{-9}m) kuni mõnekümne mikromeetrini (10^{-6}m). Suuruse järgi eristatakse PM10, PM2,5 ning PM1 osakesi nende keskmise diameetri alusel - vastavalt 10, 2,5 ning 1 mikromeetrit. PM1 osakesi tekib tööstusprotsessides, PM2,5 osakesi tekib valdavalt mehaaniliste protsesside tagajärjel ning PM10 osakesi näiteks sõidukite liikluse tagajärjel.

Aerosooli kontsentratsiooni väljendamiseks kasutatakse ruumalaühikus paiknevate osakeste arvu (arvkontsentratsioon) või massi (masskontsentratsioon), harvem pindala. Need näitajad on omavahel tihedalt seotud ning on üksteiseks teisendatavad osakeste kuju ja tiheduse järgi.

Aerosooli osakeste suurusjaotuse alusel ehk spektrites eristatakse nelja tekkegruppi: nukleatsioonimood (1 - 10 nm – kiirelt kaduvad osakesed ehk teatud tingimustele vastav aatomite või molekulide kogum, mis moodustuvad gaasiliste lisandite kondenseerumisel), Aitkeni mood (mõnikümmend nm – suuremad osakesed, kondensatsioonitsentrid pilvede tekkimisel), akumulatsioonimood (ca 100 nm – 1 μm – tekivad väiksemate osakeste

ühinemisel (koagulatsioonil - osakeste liitumisel või liitmisel suurematekс osakesteks), põlemisel, aurude kondenseerumisel ning on liiga suured, et edasi koaguleeruda ja liiga väikesed selleks, et välja sadestuda raskusjõul) ning mehhaanilise tekkega osakeste mood, mis jaguneb omakorda jämedaks ning hiiglaslikuks moodiks (jäme ca 1-10 μm, hiiglaslik kuni 100 μm) – tolm, lendtuhk, meresool, taimeeosed jms, mida tekib massi järgi palju, kuid mis sadestuvad massi tõttu kiiresti).

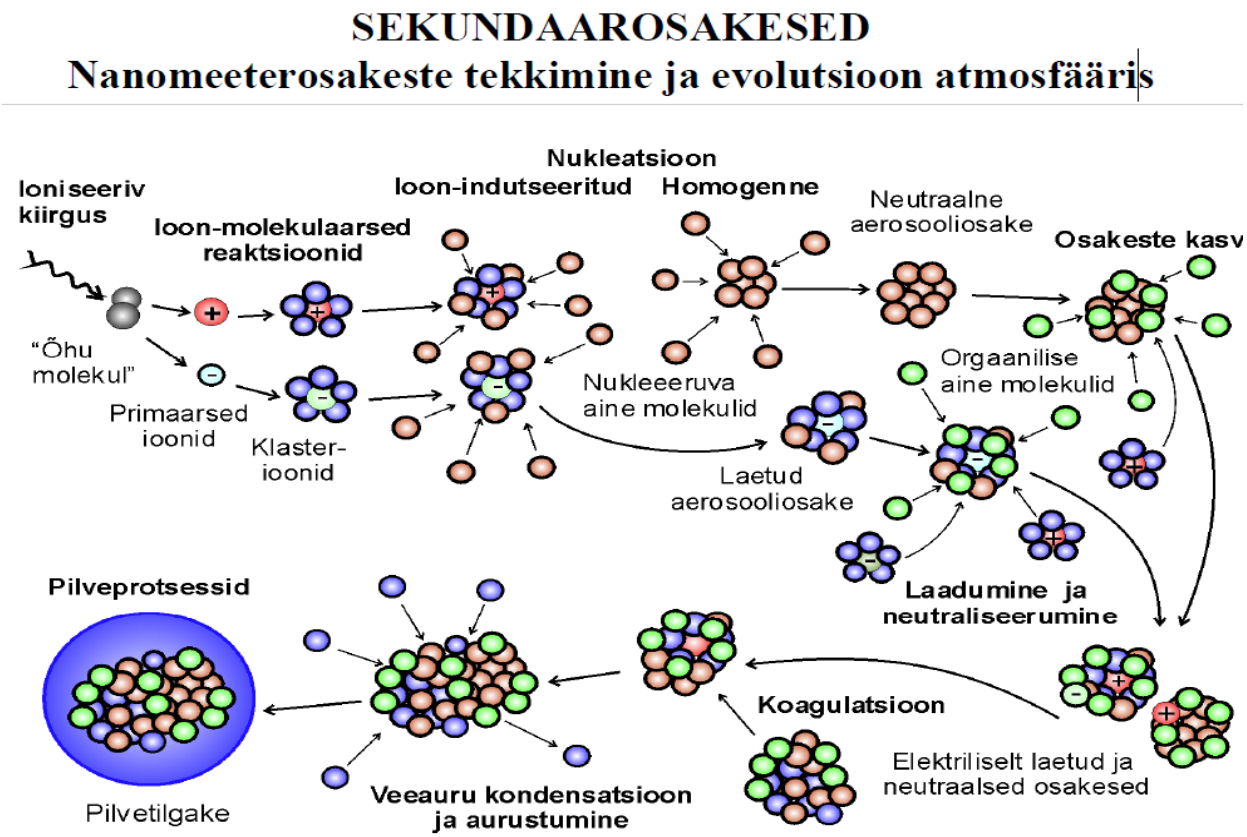


Joonis 1. Aerosooliosakeste mõõtmed ning jaotus moodidesse. **[1]**

Tänapäeval on atmosfääriaerosooli uurijate tähelepanu keskmesse tõusnud atmosfääriaerosooli sekundaarosakeste puhangute tekke uurimine (nukleatsioonipuhangud). Valdav osa tänapäeva teadmistest laetud nanomeeterosakeste puhangute kohta on saadud TÜ ja Helsingi Ülikooli teadlaste koostööna.

Nukleatsiooniks nimetatakse ca 0,35 nm diameetriga gaasimolekulide esmast ühinemist klastriks ning seejärel kasvamist aerosooliosakesteks. Kasvamisel on kõige kriitilisem moment läbida suurusjärk 1-10 nm (nukleatsioonimood). Kui aerosooliosake kannab elektrilaengut, siis nimetatakse seda aerosooliooniks. Õhus on lisaks aerosooliioonidele ka neist väiksemaid nn klasterioone. Klasterioonide tuumaks on ioniseeritud molekul, mida ümbritseb molekulaar- ja elektrijõul koos püsiv molekulide

kogum – klaster. Õhumolekulid ioniseeruvad näiteks radioaktiivsel aine lagunemisel tekkivate suure energiahulgaga elementaarosakestega põrgetel, kokkupõrgetel footonitega või kosmilise kiirguse tõttu. Ioonid kaovad vastasmärgiliste ioonidega kohtudes – rekombineeruvad, sattudes aerosooliosakesele või muudele pindadele keskkonnas.



Joonis 2. Nanomeeterosakeste teke ning areng atmosfääris. **[1]**

Vedelikes või gaasides hõljuvad osakesed võtavad osa mikroskoopilisest korrapäratust liikumisest ehk Browni liikumisest. Aerosooliosakeste Browni liikumine toimub seetõttu, et kaootiliselt liikuvad vedeliku või gaasi molekulid põrkuvad kokku aerosooliosakestega, millega muudetakse osakese kiirust ja suunda. Kokkupõrked toimuvad erinevates suundades ja seetõttu muutub osakesele üleantav impulss pidevalt. Mida väiksem on see aerosooliosake, seda rohkem ta mõjutatud on ja seda märgatavamalt ta ka liigub. Neid liikumisi iseloomustab näiteks Knudseni arv - dimensioonita arv, mis näitab molekulaarse osakese keskmise läbitud vahemaa (ennem järgmist kokkupõrget) suhet mingile füüsikalisele pikkuse skaalale (näiteks osakese enda diameetrile). Nukleatsiooniprotsessis mängib teatud rolli ka Van der Waalsi

jõud, mis on molekulide (või konkreetse molekuli enda osade) omavahelise tõmbumise või eemaletõukamise jõudude summa. Van der Waalsi jõud on võrdlemisi nõrk.

TÜ FI keskkonnafüüsika laboris on paralleelselt arendatud kahte teoreetilist mudelit, mis on eelduseks kaasaegse nukleatsiooniteooria põhiprobleemi lahendamisele. Esiteks modelleeritakse osakeste kasvu läbimõõduvahemikus 1-2 nm. Teiseks modelleeritakse nukleatsiooni ning nanomeeteraerosooli mõõtmespektri arengut erinevates tingimustes. On koostatud numbriline mudel, mis võimaldab reguleerida 94 erinevat sisendparameetrit ning simuleerida nukleatsioonipuhanguid. **[2]**

1.2 Nukleatsiooni simuleerimine

Aerosooli nukleatsiooni modelleerimiseks on loodud mitmeid tarkvarapakette. Arvutusliku efektiivsuse ja paljude füüsikaliste tegurite arvessevõtmise poolest eristub Tammeti ja Kulmala poolt loodud tarkvarapakett. **[3, 4]** Selle tarkvarapaketi sisuks on nukleatsioonipuhangute simuleerimise programm **burstsimulator.exe**, mis on kirjutatud Pascali programmeerimiskeeles ning mis modelleerib atmosfääris tekkivaid aerosooli nukleatsioonipuhanguid. Programmile tuleb ettenähtud formaadis tekstifailina ette anda arvulised lähteparameetrid ning nende baasil teostab programm arvutused. Programmi töö tulemuseks on signatuurifail ning tulemuste fail - maatriksi kujul arvud. Käesolevas töös nimetame seda **HT programm**.

HT programm on mõeldud ühe juhtumi läbiarvutamiseks. Tema automatiseeritud kasutamiseks on tarvilik kest, mis varustab HT programmi sobival viisil etteantava juhtumite reaga ja salvestab (ning vajadusel töötleb) HT programmi poolt genereeritud üksikjuhtumite arvutuste tulemused. Seni oli selle kesta rollis A.Lutsu poolt koostatud lisatarkvara.

Oleks vaja teada, milliste lähteparameetrite juures selle programmi töö tulem on kõige sarnasem reaalses elus mõõdetud tulemustele (referents väärtused). Tegemist on nn. *Monte Carlo* meetodil lähenemisega, kus tulemuste võrdlemisel referents väärtustega soovitakse teada saada need lähteparameetrid, mille juures programm väljastab referentsväärtustele kõige lähedasemad tulemuste väärtused.

Selline lähenemine tähendab aga väga palju arvutusi väga erinevate lähteparameetrite juures ning nende võrdlemist referentsväärtustega. Käesoleva töö eesmärgina on vaja luua vastav lisatarkvara, mis võimaldaks automatiseeritud tööd HT programmiga.

1.2.1 HT programmist

Programm modelleerib aerosooli nanomeeterosakeste ehk nanoosakeste puhangulist teket ja evolutsiooni. Arvesse võetakse väga palju teadaolevaid füüsikalisi protsesse ja vastasmõjusid, mis otseselt või kaudselt mõjutavad nanoosakeste evolutsiooni. **[3, 4]**

Mudelis antakse ette nanoosakese sünnisuurus d_0 ja maksimaalsed nukleatsioonikiirused J_+, J_-, J_0. Nukleatsioonikiirused antakse positiivsete ja negatiivsete ning neutraalsete osakeste jaoks funktsioonina ajast, nii vabas õhus kui ka metsas.

Nanoosakesed kasvavad kondensatsiooni abil. Mudelis vaadeldakse kahte liiki gaasilist kondenseeruvat ainet. Esimene aine, näiteks väävelhape, põhjustab kasvu osakese algstaadiumis. Teine aine, näiteks mingi orgaaniline ühend, põhjustab osakese kiirenevat kasvu hilisemas staadiumis. Mudelis antakse ette kasvuühikute diameetrid ja osakeste kasvukiirused. Kasvukiirused antakse funktsioonina ajast, nii vabas õhus kui ka metsas.

Nanoosakesed muudavad elektrilaengut kohtudes klasterioonidega ja kaovad kohtumisel taustaerosooli osakestega või puude okastega.

Positiivsed ja negatiivsed klasterioonid tekivad vastavalt paarikaupa. Tekke kiirused antakse funktsioonina ajast, puhangu algul I_0, keskel I_1 ja lõpus I_2, nii vabas õhus kui ka metsas. Klasterioonid kaovad rekombinatsiooniga, nukleatsiooniga ja sadestumisega nii nanoosakestel, taustaerosooli osakestel kui ka puude okastel. Need tekke ja kadumise protsessid määravad klasterioonide kontsentratsiooni.

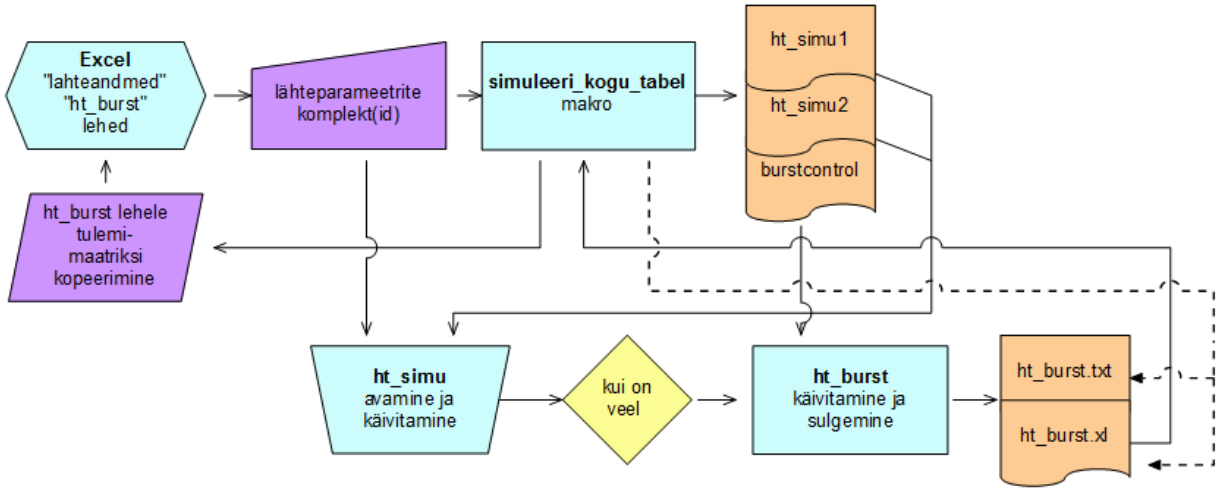
1.3 Olemasolev lisatarkvara

Varasemalt oli olemas Aare Lutsu poolt koostatud lisatarkvara, mille töö protsessi võis jagada järgmistesse etappidesse:

1. Eelvalmistus: avatakse Microsoft Excel ja määratakse lähteväärtuste komplektid (tulbad K, L, M...jne töölehel „lahteandmed“);
2. Käivitatakse makro: simuleeri kogu tabel, mis loob 3 faili:
   a) "C:\Temp\ht_burst\ht_simu1.txt – vaja ht_simu.exe programmi jaoks,
   b) "C:\Temp\ht_burst\burstcontrol.txt – vaja HT programmi jaoks,
   c) "C:\Temp\ht_burst\ht_simu2.txt – vaja ht_simu.exe programmi jaoks;

   ning kustutab (olemasolul) ht_burst.xl ja .txt failid kataloogist C:\Temp\ht_burst\;

3. Avatakse programm ht_simu.exe ja käivitatakse. Seejärel automaatselt:
   a) avatakse HT programm C:\Temp\ht_burst\ kataloogist,
   b) HT loeb lähteväärtused sisse burstcontrol.txt failist C:\Temp\ht_burst\ kataloogist ning teostab nende põhjal arvutused,
   c) HT väljastab:
      i. C:\Temp\ht_burst\ht_burst.txt - (signatuurifail),
      ii. C:\Temp\ht_burst\ht_burst.xl - (tulemuste maatriks),
      iii. (kui väljundfailid juba eksisteerivad, siis kirjutatakse need üle);
   d) Excel avab „ht_burst“ töölehe ning kopeerib ht_burst_xl tabeli sinna;
   e) kui üks lähteväärtuste komplekt on arvutatud ja tulemused kopeeritud (kirjutatakse iga kord üle), siis võetakse kasutusele järgmine,
   f) kui Exceli lähteväärtuste tulpades rohkem väärtusi ei ole, siis ht_simu väljastab teate ht_simu2.txt failist.



Plokkskeem 1. Varasem nukletasioonipuhangute simuleerimise tööskeem.

Selline süsteem on infotehnoloogilisest seisukohast ebaefektiivne:

- Tegemist on süsteemiga, kus HT programmiga töö tegemiseks on vaja installeerida mahukas lisaprogramm ja kasutada eraldi veel ka Excelit. Süsteemi käivitamisel ei saa seda arvutit muuks otstarbeks kasutada, sest vastasel korral tsükkel katkeb.

- Tulemuste tabel Excelis kirjutatakse iga arvutuse käigus üle, mistõttu ei ole hiljem enam võimalik varasemaid tulemusi analüüsida kui just ei hakata arvutusi ühe kaupa tegema ja siis eraldi lehtedele tulemuse kopeerima. Uuemate MS Office`ga (2003-10) on töölehtede arv piiratud küll arvuti mälumahuga, seetõttu oleks mõtet täiendada olemasolevaid makrosid nii, et iga uus arvutus looks automaatselt ka uue töölehe, millega saaksid kõik vahetulemused salvestatud. Samas on selge, et tuhandete töölehtede käsitlemine muutuks keeruliseks. Excel ei olegi mõeldud andmebaasi rolli täitma.

- Vanas süsteemis järgneb igale arvutusele võrdlemine referentsväärtustega, ja võrdlustulemused ka salvestatakse, kuid samas peab hakkama sobivaid (vähima erinevusega) komplekte uuesti läbi arvutama. Oleks hulga mugavam kui otsitavad arvutustulemused oleksid kohe käepärast. Nagu öeldud, takistavad sellist lahendust Exceli sisemised piirid.

- Vana süsteemi edasine täiustamine on raskendatud, seda juba Excel piiride tõttu.

1.4 Eesmärgid

Üldjoontes oli eesmärgiks luua süsteem, mis omaks senise süsteemi kõiki olulisi võimalusi, aga oleks viidud sellisele baasile, milles oleksid kõrvaldatud senise süsteemi põhilised puudused ja milles saaks teda hiljem hõlpsasti täiendada. Eesmärgiks oli luua kasutaja jaoks võimalikult lihtne ja kasutajasõbralik süsteem, mille kasutamine ei nõuaks kasutajalt erilisi teadmisi. Oluline oli see, et kasutaja saaks lihtsasti sisestada ja muuta lähteparameetreid (või neid automaatselt ettenähtud vahemikes genereerida) ning muus osas käiks töö tema jaoks täis automaatselt, vajutades vaid vastavaid nuppe. Kindlasti pidi programm võimaldama lähtekomplektide salvestamist või laadimist nimega failist. Puududa ei tohtinud ka kõikide väljade tühjendamine ühe nupuga ehk algseisu taastamise funktsioon ega vaikeväärtuste seadmiste võimalus. Pidi olema võimalus märata vahemikke, kust programm genereeriks juhuslike arvude baasil väärtusi. Tulemuste võrdlemine referentsväärtustega pidi samuti olema täisautomaatne. Kasutaja jaoks pidi kõik olema realiseeritud graafilises kasutajaliideses. Graafiline aken pidi olema inglise keeles, et seda programmi saaksid kasutada ka muukeelsed.

Programm pidi:

1. automaatselt genereerima vastava lähtefaili HT programmi jaoks kasutades etteantud väärtusi või juhuslikult genereerituid väärtusi. Samas mitte kõik parameetrid ei ole muudetavad. Need, mis on muudetavad, neid kasutaja näeb ja nende väärtusi saab ka muuta. Lisaks peaks olema võimalus lähteandmete lugemiseks hoopistükkis failist;
2. salvestama need lähteparameetrid ka andmebaasi ning uurima andmebaasist, milline id konkreetsele lähtekomplektile andmebaasi poolt määrati, et hiljem oleks võimalik seostada tulemused vastava lähteväärtuste komplektiga;
3. avama ning HT programmi ning töö lõpetades viimase sulgema;
4. lugema HT programmi töö tulemusena tekkinud tulemustemaatriksi sisse ja salvestama selle andmebaasi määratud kujul ja nii, et tekiks jääv seos tulemuste ja vastava lähteväärtuste komplekti suhtes;
5. kordama protsessi kui soovitakse korraga rohkem kordi tsüklit läbida;

6. iseseisvalt arvutama etteantud referents tabeli ja tulemuste tabelite summaarsed erinevused. Väljastama lähteväärtuste kompleti, mille tulemuste tabeli summaarsed erinevused referentsväärtustega olid kõige väiksemad.

Kõige parem lahendus oli kasutada mõnda objektorienteeritud programmeerimiskeelt, mille abil luua kõik vajalik v.a. andmebaasi osa. Seetõttu varasemast süsteemist muud peale HT programmi kasutusse ei võetud. Kindlasti pidi see keel olema populaarne, et selles keeles loodud tarkvara oleks hiljem paljudel võimalik uuendada. Kaalumisele tulid C++, Java, Python.

Kuna tehtavas programmis on kasutajal mitmeid kümneid muudetavaid parameetreid ning lisaks veel iga parameetri sisestamiseks kaks võimalust: minimaalne - maksimaalne vahemik või valikumenüü, siis oli selge, et väga oluliseks ja töömahukaks saab just graafilise kasutajaliidese realiseerimine. Võis ennustada, et valmiva programmi kogufunktsionaalsuse koodi osa ei ületa graafilise kasutajaliidese koodi mahtu. Oli väga oluline, et graafiline osa saaks tehtud võimalikult kerge vaevaga ja keskendatud funktsionaalsuse arendamisele.

Java jaoks oli Eclipse Foundation eraldi loonud Windowbuilder nimelise lisatarkvara Eclipse keskonnas programmeerimiseks, mis tegi graafilise akna joonistamise väga lihtsaks: kood genereeritakse automaatselt, kasutaja lihtsalt sisestab aknasse soovitud elemente ja elemendidki on lihtsasti valitavad. Eclipse (toetab ka C++) on abiprogramm programmeerijale, mis võimaldab koodi kompileerida ja testida mugavalt ning parandada jooksvalt vigu, mis kompileerimisel tuleksid. Päris nii lihtsat graafilise osa loomise võimalust teistes keeltes ei olnud (tulnuks leppida kehvemate variantidega) ja WindowBulderi lisa Eclipsile Java jaoks saigi otsustavaks.

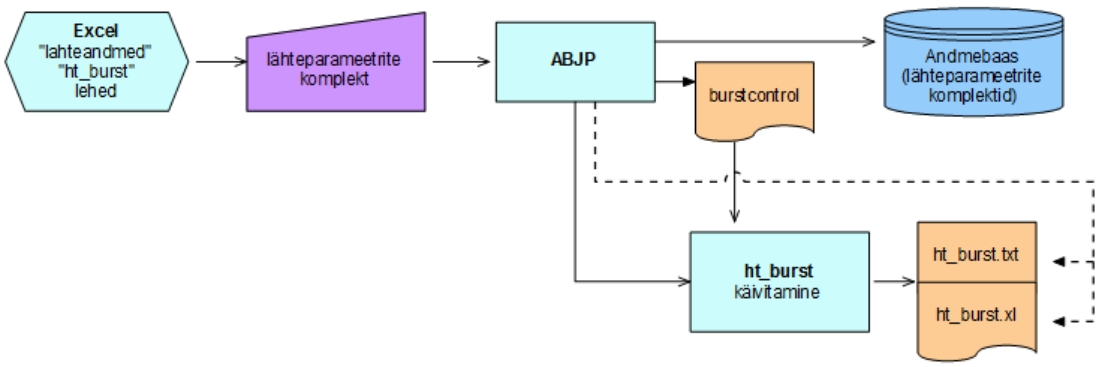
Andmebaasina sai kaalutud Sybase -i ning MySQL-i. Kuna viimasel on aga parem tugi Javaga ühildamiseks, siis Sybase vahetasin välja hiljem MySQL-i vastu. Nende kahe abil oli võimalik nüüd kõik eesmärgid täita.

2. Plaan

Töö oli planeeritud etappidesse, et järk-järgult eesmarkide poole liikuda.

I. Olemasolevale süsteemile luuakse juurde andmebaasi osa: andmebaas ja selle juhtimisprogramm (edaspidi ABJP). Tehakse ühekordne automaatne arvutamise protsess ilma Excelita ja lähteandmete komplekt(id) salvestatakse. Lähteandmed võetakse eraldi failist.

   a) Andmebaasis luuakse tabel lähteväärtuste komplektide jaoks;
   b) ABJP loeb etteantud failist lähteväärtuste komplekti ning loob selle baasil ise burstcontrol.txt faili. Kõik arvutused tehakse 5-min ajasammuga (v.t. lk 17, 3.1) ja tulemuses näidatakse kõiki võimalikke parameetreid, st HT programmi poolt tekitatava maatriksi laius on alati võrdne võimalike tulemusparameetrite maksimumarvuga. HT programmi poolt tekitatava maatriksi pikkus sõltub etteantavast ajavahemikust, kui pikalt tahetakse puhangut simuleerida;
   c) Programm kustutab ht_burst.txt ning ht_burst.xl failid C:\Temp\ht_burst kataloogist;
   d) ABJP salvestab lähteväärtuste komplekti andmebaasi, iga komplekt saab unikaalse ID;
   e) ABJP käivitab HT programmi.



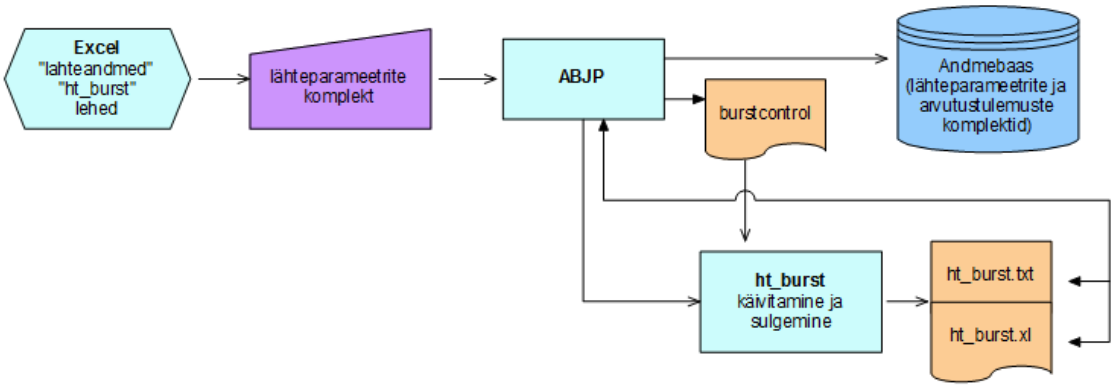
Plokkskeem 2. Esimese etapi tööskeem.

II. Lisafunktsionaalsused ühekordseks arvutamiseks ja andmete salvestamiseks.

a) ABJP käib peale HT programmi käivitamist kontrollimas, kas programm on töö lõpetanud. Kui programm leiab C:\Temp\ht_burst kataloogist eelmainitud kaks faili, siis on töö lõpetatud ja kui ei leia, siis töö veel käib;

b) Kui töö on lõpetatud, siis programm suletakse;

c) Programm loeb tulemuste faili sisse ja kirjutab need andmebaasi, kusjuures loodaks jääv seos algandmete ja tulemuste vahel (tulemustest tulenev andmebaasi uus kirje on alati ühe pikkusega: rea pikkus ehk tulemusparameetrite maksimumarv korda ridade arv ehk 50(veergu)*278(rida) - vastab juhtumile, kui oleks modelleeritud 24 tundikestvat puhangut 5-minutilise ajasammuga (v.t. lk 17, 3.1). ABJP on võimeline aru saama kui tegelikult modelleeriti lühemat ja sisestama ka nii.

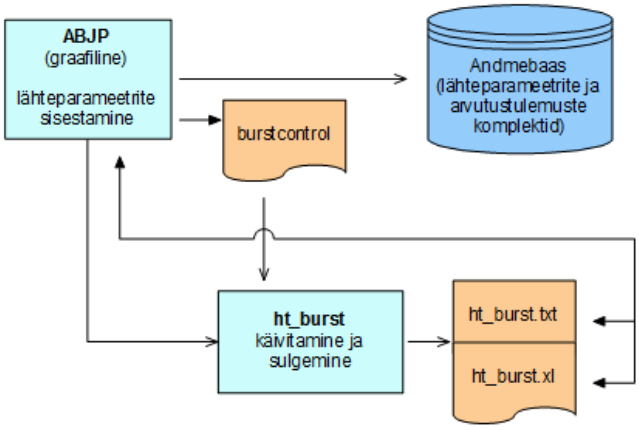


Plokkskeem 3. Teise etapi tööskeem.

III. Graafiline kasutajaliides (GUI) lähteväärtuste sisestamise jaoks. Kasutajale avaneb graafiline aken, kuhu ta saab kõik lähteväärtused sisestada ning vajutada „Simuleeri“ nuppu.

a) ABJP ei võta enam lähteandmeid failist, lähteväärtused on kasutaja poolt sisestatud. Kui kasutaja on igale poole väärtused lisanud ja vajutanud „Start“ nuppu, siis protsess jätkub nagu eelnevates etappides;

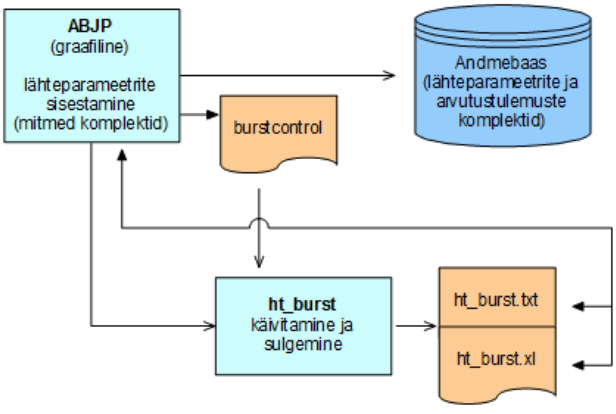
b) Kui arvutused on tehtud ning tulemused salvestatud, siis programm väljastab vastava teate.



Plokkskeem 4. Kolmanda etapi tööskeem.

IV. Täiustatud funktsionaalsus ja graafiline kasutajaliides. Kasutaja saab graafilises aknas luua lähteandmete komplekte ja jooksutada neid.

a) Kasutaja saab lähteparameetreid valida rippmenüüst või sisestada väärtusvahemikke, mille vahel genereeritakse väärtusi automaatselt;
b) Kasutaja saab ka määrata, mitu simuleerimise tsüklit tehakse;
c) Kui kasutaja vajutab „Simuleeri“, siis tehakse täis arvutuste tsükkel nii mitu korda nagu kasutaja oli määranud ja selliste väärtustega nagu kasutaja seadistas (parameetrite vahemike kohta on nõuded, millest alla- ega ülespoole ei tohi minna);
d) Iga lähteväärtuste komplekti ja tulemuste komplekti vahel luuakse andmebaasis jääv seos.

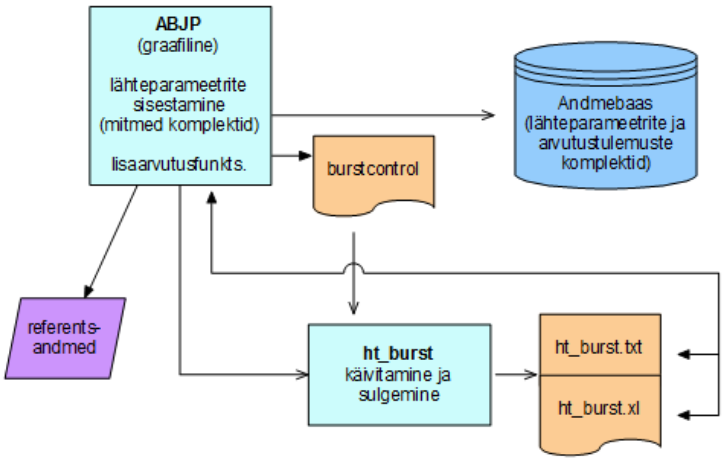


Plokkskeem 5. Neljanda etapi tööskeem.

V. Võrdlus referents väärtustega.

a) ABJP on võimeline sisse lugema HT programmi väljundmaatriksile sarnasel kujul etteantud referentsandmed (mõõdetud tulemused) koos iga väljundparameetri (tehniliselt: maatriksi veeru) kaaluga (0 kuni 10).

b) Käima läbi andmebaasi tulemuste komplektid, arvutama summaarsed erinevused referentsandmetega ning väljastama väikseima leitud väärtuse.



Plokkskeem 6. Viienda etapi tööskeem.

3. Lahendus

Tervikliku lahenduse pakuvad kaks osapoolt: Java programm (mis jooksutab endas ka HT programmi) ja MySQL andmebaas. Rohkemat vaja ei ole.

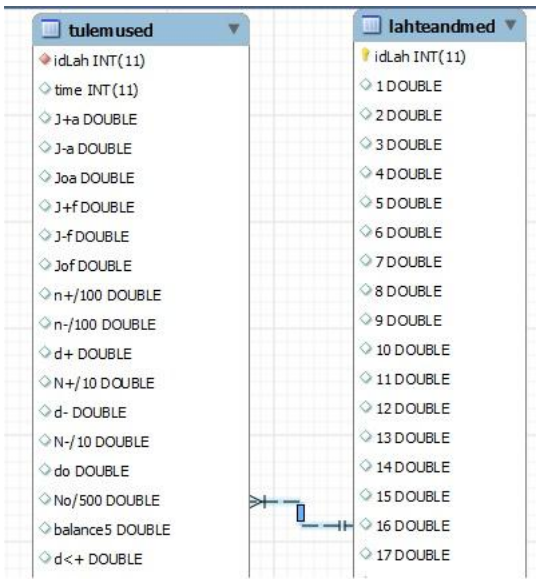
3.1 Andmebaas

Andmebaasi planeerimisel sai läbi töötatud palju variante. Kõige suurem murepunkt oli see, et ühe mõõtmelise (realise) lähtekomplekti kohta tuli tulemuseks maatriksi kujul (mitmerealine) tulemuste komplekt. Lisakeerukust lisas see, et sõltuvalt lähteparameetritest muutus tulemuste maatriksi pikkus ja elemendid (spektri osa). Oli vaja leida standard, et tulemustemaatriks oleks sõltumata muudetavatest lähteparameetritest alati ühesuguse struktuuriga, s. t. arvulised väärtused muidugi erinesid, aga tulpades on alati neil sama füüsikalise suuruse vastav väärtus.

I. Variandina sai kaalutud tulemustemaatriksi tükeldamist ridadeks ja nende ridade põimimist üksteise otsa üheks pikaks reaks. Sellisel juhul oleks saanud lähteparameetrite rea kohta ka ühe rea tulemustemaatriksisse. Maksimaalsel juhul oleks tulemustemaatriksi pikkus läinud aga tugevasti üle lubatud veergude arvu tabeli kohta (maksimaalselt MySQLis 4096 tulpa, aga reaalselt vähem - sõned, komakohaga tüüpi väärtused). **[3]**
II. Variandina sai kaalutud iga tulemustemaatriksi kohta eraldi tulemustetabeli loomist vastava lähtekomplekti identifitseeriva numbriga tabeli nimes. Selle vastu rääkis asjaolu, et paljude tabelite korral läheks tabelite võrdlemise süntaksi loomine väga pikaks, sest iga tabeli nime peab ära mainima. See läheks andmebaasi ideega - võrreldavad väärtused võiksid olla samas tabelis samades tulpades – vastuollu. Ka siin pidanuks tabelite üldine struktuur ühesugune olema.
III. Parimaks kujunes variant, kus andmebaasi tulemuste tabelisse pannakse tulemustemaatriks mitme reana. Just sellisena nagu nad tulemustemaatriksis on, iga rida on ka tulemuste tabelis omaette real. Esimeseks tulbaks saab aga vastava lähteväärtuste

tulba id number. Nii on need ühe tulemustemaatriksi read seotud sama id numbriga. Siin ei teki ka vajadust osa andmebaasist täita tühiväärtustega, sest tulemustetabelisse pannakse täpselt nii palju infot (ridu) kui on vastavas tulemustemaatriksis (veergude arv ja struktuur on kokku lepitud). Ei ole ka vaja teist tulpa selle jaoks , et tulemuste tabeli ridu määratleda eraldi numbritega 1, 2 ,3 jne. Selleks sobib juba tulemustemaatriksi esimene rida ehk „time“ väärtus.

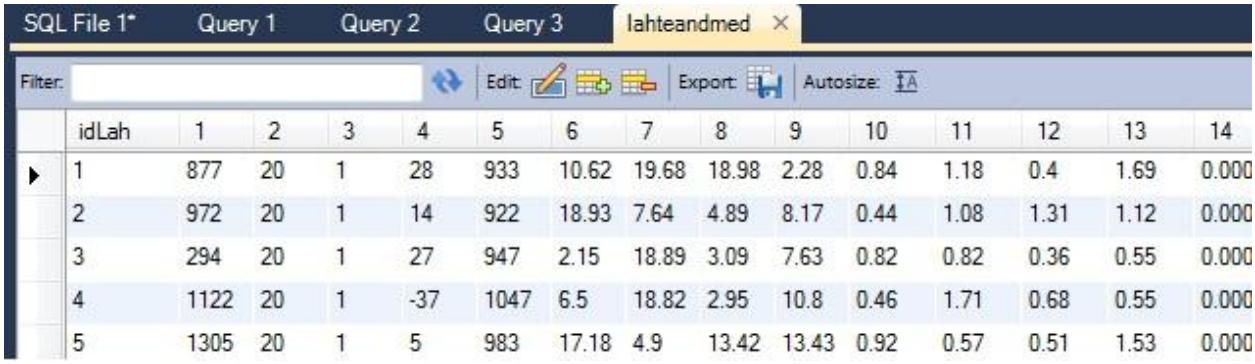
Andembaas sai realiseeritud nii, et on vaid kaks tabelit: tabel „lahtevaartused“ ning tabel „tulemused“. Rohkem ei olegi vaja. Nende vahel kehtib üks mitmele (1:n) seos, kus „lahtevaartused“ tulba „idLah“ väärtus seob „tulemused“ tabeli vastavaid ridu. Lisaks sai kasutusmugavuse huvides paigaldatud MySQL Workbench, et graafiliselt andmebaasi kasutada.



Joonis 2. Andmebaasi EER mudel.

- Andmebaasi loomine: **CREATE SCHEMA `htsupport`;** (kasutaja: root ja parooliks 00pi00ir).
- Andmebaasi kasutamiseks: **use htsupport;.**
- Lähtudes ülesande spetsiifilisusest tuli kaotada ka tabelite vaikeväärtused maksimaalse ridade arvu jaoks (oli 1000).

Lähteväärtusi on tabelis 105 tk (erinevaid 94), ning lisaks esimene veerg on idLah, unikaalne väärtus, mida suurendatakse automaatselt kui tabelisse uus rida sisestatakse.



SQL File 1* | Query 1 | Query 2 | Query 3 | lahteandmed

Filter: | Edit: | Export: | Autosize:

idLah	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	877	20	1	28	933	10.62	19.68	18.98	2.28	0.84	1.18	0.4	1.69	0.000
2	972	20	1	14	922	18.93	7.64	4.89	8.17	0.44	1.08	1.31	1.12	0.000
3	294	20	1	27	947	2.15	18.89	3.09	7.63	0.82	0.82	0.36	0.55	0.000
4	1122	20	1	-37	1047	6.5	18.82	2.95	10.8	0.46	1.71	0.68	0.55	0.000
5	1305	20	1	5	983	17.18	4.9	13.42	13.43	0.92	0.57	0.51	1.53	0.000

Joonis 3. Andmebaasi lähteandmete tabeli osa näidis.

Tulemuste tabeli puhul sai kaalutud mitmeid formaate ning lõpuks sai kasutusele võetud variant, kus veerge on 50 ning lisaks esimene veerg vastava lähtekomplekti id numbriga. Veergude nimed on valitud nii, et nad viitaksid mis väärtus selles veerus on. Lähteparameetrid, mille muutmisel tulemustetabeli pikkus ei oleks 50 või ei oleks kõik veerud sama sisuga, sai muudetud kasutaja jaoks mitte muudetavaks ja neid ta Java programmi aknas näe.

Ridade arv sai sõltuvaks simuleerimise sessiooni pikkusest jagatuna mõõtmiste intervalliga. Kokku sai lepitud, et mõõtmised toimuvad alati iga 5 minuti järel, sest siis on väärtused tulemuste vastavates ridades alati võrreldavad.

SQL File 1* | lahteandmed | tulemused

Filter: | Export: | Autosize:

idLah	time	J+a	J-a	Joa	J+f	J-f	Jof	n+/100	n-/100	d+	N+/
1	0	0	0	0	0	0	0	14.68	4.09	0	0
1	5	0.21	0.08	0.79	0.02	0.04	2.03	14.54	4.02	1.69	0.47
1	10	0.42	0.15	1.57	0.03	0.07	4.06	14.34	3.94	1.81	1.95
1	15	0.63	0.23	2.36	0.05	0.11	6.09	14.12	3.86	1.93	4.14
1	20	0.77	0.28	2.87	0.06	0.13	7.41	13.9	3.78	2.05	6.85
1	25	0.77	0.28	2.87	0.06	0.13	7.41	13.76	3.71	2.2	9.12
1	30	0.77	0.28	2.87	0.06	0.13	7.41	13.66	3.64	2.38	10.7

Joonis 4. Andmebaasi tulemuste tabeli osa näidis.

Et otsimist tulemustetabelis märgatavalt kiirendada, sai kasutusele võetud indekseerimine: indeksi nimega “lahteandmedidLah” tulemuste tabeli veeru idLah kohta:

```
CREATE INDEX lahteandmedidLah ON tulemused (idLah);
```

Tabelite vahel seose loomiseks:

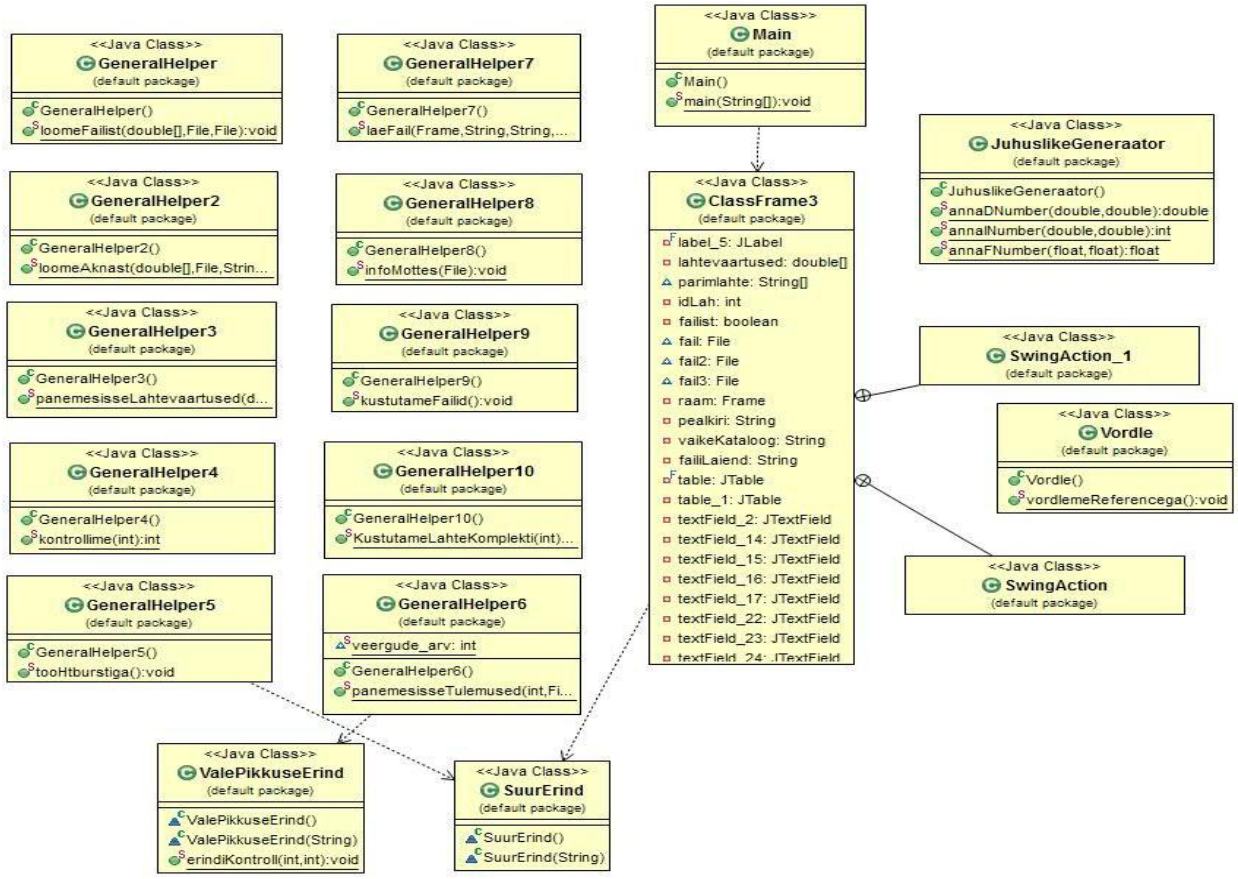
```
ALTER TABLE tulemused ADD CONSTRAINT
fk_tulemused_2_lahteandmed FOREIGN KEY (idLah) REFERENCES
lahteandmed(idLah) ON DELETE CASCADE ON UPDATE CASCADE;
```

Kui kasutaja kustutab lähteparameetrite tabelist ridu, siis kustutatakse ka vastavad tulemuste tabeli read, kus on sama IdLah, sest puudub mõte nende edaspidiseks hoidmiseks andmebaasis.

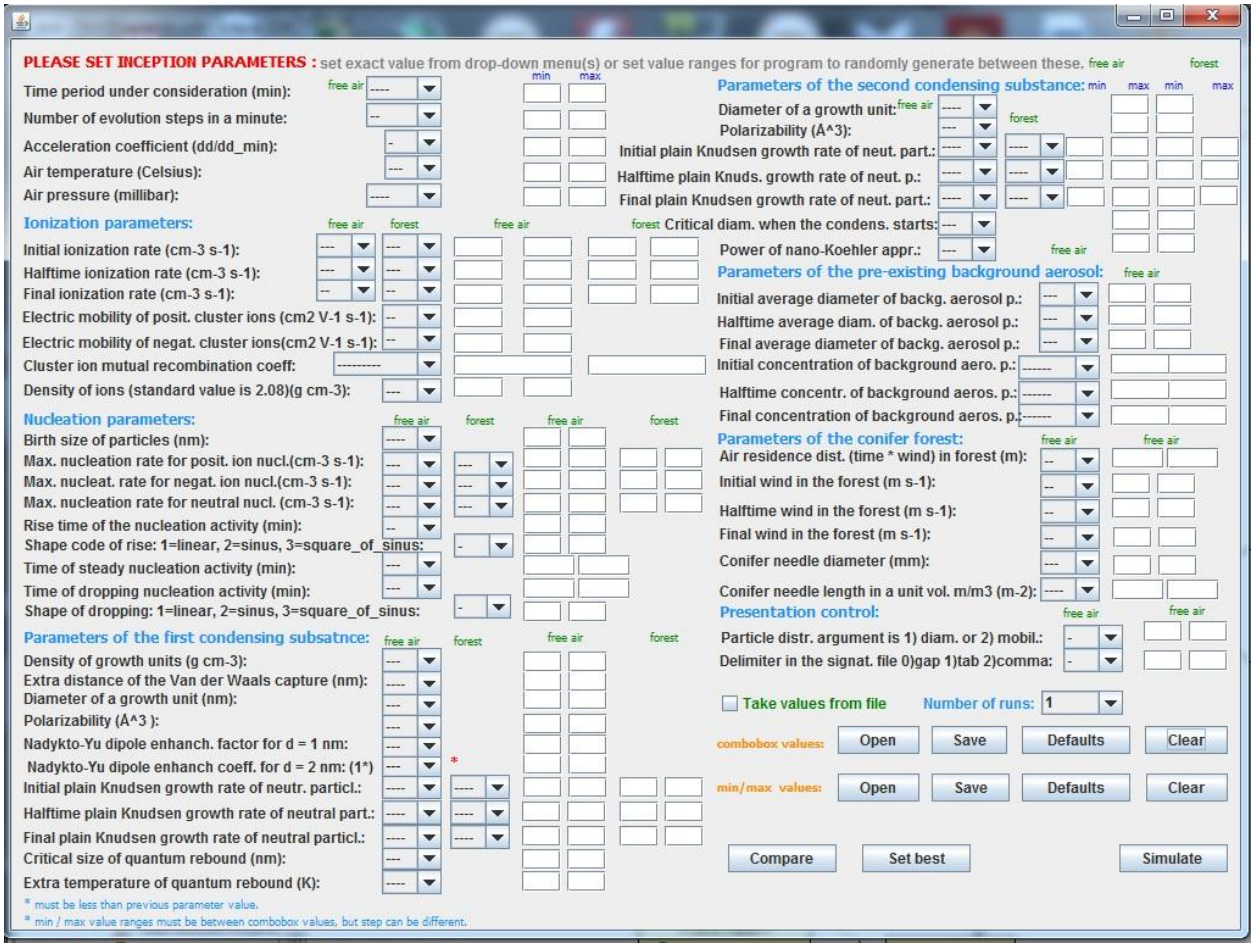
3.1 Java programm

Kogu programm on jagatud seitsmeteistkümnesse klassi. Peaklass on nimega „Main“. Klassi „ClassFrame3“ lühendamiseks on meetodid sellest klassist tõstetud eraldi „GeneralHelper“ klassidesse. Need klassid täidavad spetsiifilisi ülesandeid. „ClassFrame3“ loob kasutajale graafilise tööakna.

Lisaks on klassid juhuslike arvude genereerimiseks nimega „JuhuslikeGeneraator“, kaks erindite klassi „ValePikkuseErind“ ja „SuurErind“ ning klass „Vordle“. „JuhuslikeGeneraator“ genereerib etteantud arvude vahemikes juhuslikult arvu ja tagastab selle kas täisarvuna, kahe komakohaga arvuna või ujukoma arvuna. Erindi klassid on spetsiifiliste erindite leidmiseks ja klass „Vordle“ teostab referentsväärtuste ja tulemuste vahel vajalikud arvutused.



Joonis 5. Ülevaatlikult programmi klassid ja meetodid.



Joonis 6. Programmi graafiline kasutajaliides.

Kasutaja saab liugmenüü abil valida igale füüsikalisele suurusele väärtuse määratud vahemikes ja sammudega või kasutaja võib sisestada juhuslikuks väärtuse genereerimiseks samas vahemikus minimaalse ja maksimaalse väärtusvahemiku. Kui on täidetud mõlemad väljad, siis kasutusse läheb liugmenüüst määratud väärtus.

Nupuga „Clear“ saab kasutaja korraga kõik sisestatud väärtused kaotada ja nupuga „Defaults“ seada vaikeväärtused. Kasutaja saab ka salvestada hetkeväärtusi („Save“) või laadida varem salvestatud väärtusi („Open“). Lisaks on võimalus väärtusi võtta hoopiski eraldi failist ja aknas neid mitte määrata („take values from file“). „Set best“ väärtus seab miinimumi ja maksimumi väljadele failist väärtused nii, et miinimum ja maksimum on ühesugused ja antud väärtustega on võimalik uuesti simuleerimine läbi viia või teha neis ennem lihtsasti mõni muudatus.

Nupust „Simulate“ algab tsükkel:

- kus lähteväärtustest luuake vastavav lähteväärtuste fail;
- lähteväärtused pannakse andmebaasi;
- avatakse HT programm ja kui see on töö lõpetanud, siis see suletakse;
- kui programmi töös on tekkinud vigu ja tulemustefaile ei loodud, siis andmebaasist kustutatakse ka vastav lähteväärtuste komplekt;
- uuritakse andmebaasist, millise id väärtuse sai vastav läheväärtuste komplekt;
- loetakse tulemused;
- pannakse tulemused andmebaasi;
- korratakse protsessi kui jooksutamiste arvuks oli rohkem kui üks määratud.

Konsooli aknast saab jälgida töö protsessi.

```
Tekstiväljad tühjendatud: Defaults

ring: 1

  >>>> Hakkame simuleerima! <<<<
Valiste failist lähteväärtuste võtmise? false
Loeme sisse lähteväärtuste komplekti aknast
Tekitame burstcontrol.txt faili..
Korras!
Sisestan lähteväärtused Mysql andmebaasi lahteandmete tabelisse!
korras.
***
Fail ht_burst.txt on olemas, kustutan.
Faili ei ole olemas, korras.
Fail ht_burst.xl on olemas, kustutan.
Faili ei eksisteeri, korras.
***
Avame ht_bursti..
Burstsimulator HT20060912

23900 time steps and 3827 size sections up to 212.27 nm

ooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooo
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
01234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456
fffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
01234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456

ht_burst.xl & ht_burst.txt are successfully saved.
Processing time 14.42 seconds.
Press ENTER for exit!
ht_burst on selleks korraks töö lõpetanud, sulgeme.
***
Loeme sisse tulemuste komplekti..
Fail leitud!
Ridu on seekord: 241
Hakkame tulemusi andmebaasi panema!
Uurime eelneva lähtekomplekti idLah väärtust..
Käes! idLah eelmisele komplektile oli: 6
Sisestame..
Sisestamine lõpetatud.
  >>>> Lõpetasime selle simuleerimise ringi.  <<<<  1
```

Joonis 7. Programmi konsooli näidiskirjed.

Nupust „Compare“ nupust algab protsess:

- kus uuritakse kui pikk on võrdlusesse minev referentsmaatriks;
- referentsmaatriksi esimesest reast loetakse vastavate tulpade kaalud;
- referentstabeli ülejäänud osa loetakse võrdluseks sisse;
- uuritakse andmebaasist milline on minimaalne ja maksimaalne tulemuste komplektide id;
- algab tsükkel minimaalsemast leitud komplektist kuni maksimaalseni:
    - kui vastav tulemuste komplekt ei ole vähemalt nii pikk kui on referents, siis seda komplekti ei võrrelda;
    - kui vastava id numbriga komplekti ei eksisteeri, siis minnakse järgmist otsima;
    - kui tulemuste komplekt on olemas, siis loetakse sealt referentsmaatriksiga võrdne osa ja teostatakse iga maatriksi komponentide vahel kaalude vahe arvutamise operatsioon. Kõik erinevuste summad liidetakse kokku;
    - kui suma on väiksem kui seni parim, siis tunnistatakse see komplekt parimaks;
- kui kõik tulemused on andmebaasist läbi käidud, siis väljastatakse teade, milline lähteväärtuste komplekti id oli parim, milline oli kogusumma ja vastavad lähteväärtused, millega saadi parim tulemuste vahe, kirjutatakse faili.

4. Kokkuvõte

Antud tööna valminud lahendus täitab väga hästi seatud eesmärke. Kasutaja jaoks on programmi kasutamine väga lihtne – vaid üks programmi aken võimaldab kõiki eesmärkideks seatud ülesandeid täita. Kasutaja saab seada lähteväärtusi, anda ette väärtusvahemikke juhuslike arvude genereerimiseks või lugeda neid failist. Ei puudu ka mugavuse jaoks vaikeväärtuste seadmise ega väljade tühjendamise funktsioon. Realiseeritud on ka väärtuste salvestamine faili või nende lugemine failist.

Lisana mugavuse mõttes on andmevood java programmis realiseeritud märgivoogudena, mistõttu kasutaja saab kõiki failides (kasutaja poolt salvestatud lähteväärtuste komplektid, vaikeväärtustega failid jms) olevaid väärtusi vaadata ja modifitseerida ilma java programmi kasutamata mõne lihtsat tekstifaile lugeva programmiga.

Arvutamine HT programmiga ning koostöö andmeaasiga käib täisautomaatselt. Kasutaja vajutab vaid ühte nuppu. Ka arvutustulemuste võrdlemine referentsväärtustega on lahendatud kasutaja jaoks täisautomaatselt.

Edasi võiks seda süsteemi täiendada lisades referentsväärtustega võrdlemisel veergude kaaludele lisaks ka ridadel kaalud. Samuti võiks lisada ka keeratavad nupud lähteparameetrite muutmiseks ja vastavate lähteparameetrite baasil arvutatud tulemused võiksid olla esitatud graafikuna. Kui lähteparameetrit muudetakse, siis programm otsiks automaatselt tulemused andmebaasist, mis on nupu all oleva väärtuse jaoks juba arvutatud ja joonistaks automaatselt tulemused graafikule. Nii oleks lähteparameetri(te) jooksval muutmisel hea graafikult vaadata, kuidas tulemused muutuvad.

5. Summary

Subsidiary software for the modeling of aerosol genesis

Indrek Ploom

Summary

There were many possible solutions to provide support for atmospheric aerosol nucleation simulating tool for extensive calculations in search of best inceptions parameters – closest to real life measurements.

It was found in this work, that the best way to support extensive calculations is to use a database and additional specific software to control the process. One software to perform all required actions.

This solution has a lot of advantages. First of all, it is simple. It uses a database to bind and hold results and according pre-parameters. As database is meant for holding extensive amount of data, it has no issues to provide information about previous calculations. As the matter of fact, it only has two tables so the structure is also very primitive.

The other good side of this solution is the software to control and execute processes. As it is written in object oriented programming language – Java in this case, it is easy to upgrade or maintenance code. There are no requirements for user. Everything is pretty simply bought to a user via graphical interface and all the work can be done by that one window. User does not need to know how the system works or interact in the middle of the process. All the sophisticated part, such as communication with the database or executing comparison is done fully automatically, triggered by a user.

Third positive part of this solution is the ability to work automatically. User can set system to auto generate random values and set number of cycles system to perform. Later on user can find out what were the best pre-parameters set that produced closest calculation results for real life measurements.

Fourth, this system lets user to set precise values for every single parameter (within limits and if parameter is changeable) or combine drop-down menu values among random generations. This way user can freely start with best results and towards even better or do very random simulations.

Fifth, the system is fast. As soon as prerequisites are set for simulation program to work, it will be run inside Java and closed when program is done. This is very time effective.

There were also some problems arising when designing database structure. One thing was the fact that the simulating tool produced two-dimensional result matrixes from pre-parameters, which were one-dimensional. The other thing was related to specific pre-parameters, which changed the result matrix formation. We had to find a solution how to keep output in one format so results can be compared one to one and how to store them most efficiently.

Java in general tends to use a lot of computer memory and therefore there might be need to reopen application sometimes. Very extensive communication between Java program and database might sometimes also create issues, e.g. connection limit.

One more thing is related to simulation program. If it fails to calculate because of "faulty" pre-parameters, it can end up "freezing".' And Java program does not know that. In this case the cycle is at dead stop. But this can be avoided by testing and seems now "problematic" pre-parameters have been found.

6. Kasutatud kirjandus

1. Urmas Hõrrak, „ATMOSFÄÄRIAEROSOOLIDE FÜÜSIKA JA SEIRE, “ http://meteo.physic.ut.ee/kkfi/index_files/oppetoo/tutvustavad-materjalid/Atmosfaariaerosoolide-fyysika-ja-seire.pdf .
2. H. Tammet, E. Tamm, J. Salm, E. Realo, „Aerosoolid ja radioaktiivsus Keskkonnas,“ *Teadusmõte Eestis, täppisteadused,* Eesti Teaduste Akadeemia, Tallinn, 127-134, http://www.fyysika.ee/doc/akad_96_aero.pdf , (2006).
3. H. Tammet, M. Kulmala, “Simulation tool for atmospheric aerosol nucleation Bursts, ” *J. Aerosol Sci.* **36**, 173-196, (2005).
4. H. Tammet, M Kulmala, „Simulating aerosol nucleation bursts in a coniferous forest,“ *Boreal Env. Res.*, **12**, 421-430, (2007).
5. http://et.wikipedia.org/wiki/Aerosool 08.09.2011.
6. http://et.wikipedia.org/wiki/Browni_liikumine 17.03.2012.
7. http://en.wikipedia.org/wiki/Brownian_motion 13.05.2012.
8. http://en.wikipedia.org/wiki/Knudsen_number 16.05.2012.
9. http://en.wikipedia.org/wiki/Van_der_Waals_force 26.05.2012.
10. http://dev.mysql.com/doc/refman/5.0/en/column-count-limit.html 2012.
11. E. Tamm, „Aerosoolifüüsika,“ http://meteo.physic.ut.ee/kkfi/index_files/eduard_tamm/AEROSOOLIFYYSIKA_internetis_2008.pdf , (2008).

Lisa 1. Programmi koodid

Lisa 2. Andmebaasi tabelite loomise käsud

```
CREATE  TABLE `htsupport`.`lahteandmed` (
  `idlah` INT NOT NULL AUTO_INCREMENT,
  `1` DOUBLE NULL ,
  `2` DOUBLE NULL ,
  `3` DOUBLE NULL ,
  `4` DOUBLE NULL ,
  `5` DOUBLE NULL ,
  `6` DOUBLE NULL ,
  `7` DOUBLE NULL ,
  `8` DOUBLE NULL ,
  `9` DOUBLE NULL ,
  `10` DOUBLE NULL ,
  `11` DOUBLE NULL ,
  `12` DOUBLE NULL ,
  `13` DOUBLE NULL ,
  `14` DOUBLE NULL ,
  `15` DOUBLE NULL ,
  `16` DOUBLE NULL ,
  `17` DOUBLE NULL ,
  `18` DOUBLE NULL ,
  `19` DOUBLE NULL ,
  `20` DOUBLE NULL ,
  `21` DOUBLE NULL ,
  `22` DOUBLE NULL ,
  `23` DOUBLE NULL ,
  `24` DOUBLE NULL ,
  `25` DOUBLE NULL ,
  `26` DOUBLE NULL ,
  `27` DOUBLE NULL ,
  `28` DOUBLE NULL ,
  `29` DOUBLE NULL ,
  `30` DOUBLE NULL ,
  `31` DOUBLE NULL ,
  `32` DOUBLE NULL ,
  `33` DOUBLE NULL ,
  `34` DOUBLE NULL ,
  `35` DOUBLE NULL ,
  `36` DOUBLE NULL ,
  `37` DOUBLE NULL ,
  `38` DOUBLE NULL ,
  `39` DOUBLE NULL ,
  `40` DOUBLE NULL ,
  `41` DOUBLE NULL ,
  `42` DOUBLE NULL ,
  `43` DOUBLE NULL ,
  `44` DOUBLE NULL ,
  `45` DOUBLE NULL ,
  `46` DOUBLE NULL ,
  `47` DOUBLE NULL ,
  `48` DOUBLE NULL ,
  `49` DOUBLE NULL ,
  `50` DOUBLE NULL ,
  `51` DOUBLE NULL ,
  `52` DOUBLE NULL ,
  `53` DOUBLE NULL ,
  `54` DOUBLE NULL ,
  `55` DOUBLE NULL ,
  `56` DOUBLE NULL ,
  `57` DOUBLE NULL ,
  `58` DOUBLE NULL ,
  `59` DOUBLE NULL ,
  `60` DOUBLE NULL ,
  `61` DOUBLE NULL ,
```

```
  `62` DOUBLE NULL ,
  `63` DOUBLE NULL ,
  `64` DOUBLE NULL ,
  `65` DOUBLE NULL ,
  `66` DOUBLE NULL ,
  `67` DOUBLE NULL ,
  `68` DOUBLE NULL ,
  `69` DOUBLE NULL ,
  `70` DOUBLE NULL ,
  `71` DOUBLE NULL ,
  `72` DOUBLE NULL ,
  `73` DOUBLE NULL ,
  `74` DOUBLE NULL ,
  `75` DOUBLE NULL ,
  `76` DOUBLE NULL ,
  `77` DOUBLE NULL ,
  `78` DOUBLE NULL ,
  `79` DOUBLE NULL ,
  `80` DOUBLE NULL ,
  `81` DOUBLE NULL ,
  `82` DOUBLE NULL ,
  `83` DOUBLE NULL ,
  `84` DOUBLE NULL ,
  `85` DOUBLE NULL ,
  `86` DOUBLE NULL ,
  `87` DOUBLE NULL ,
  `88` DOUBLE NULL ,
  `89` DOUBLE NULL ,
  `90` DOUBLE NULL ,
  `91` DOUBLE NULL ,
  `92` DOUBLE NULL ,
  `93` DOUBLE NULL ,
  `94` DOUBLE NULL ,
  `95` DOUBLE NULL ,
  `96` DOUBLE NULL ,
  `97` DOUBLE NULL ,
  `98` DOUBLE NULL ,
  `99` DOUBLE NULL ,
  `100` DOUBLE NULL ,
  `101` DOUBLE NULL ,
  `102` DOUBLE NULL ,
  `103` DOUBLE NULL ,
  `104` DOUBLE NULL ,
  `105` DOUBLE NULL ,
  PRIMARY KEY (`idlah`) ,
  UNIQUE INDEX `idlah_UNIQUE` (`idLah` ASC) );

CREATE  TABLE `htsupport`.`tulemused` (
  `idLah` INT NOT NULL ,
  `time` INT NULL ,
  `J+a` DOUBLE NULL ,
  `J-a` DOUBLE NULL ,
  `Joa` DOUBLE NULL ,
  `J+f` DOUBLE NULL ,
  `J-f` DOUBLE NULL ,
  `Jof` DOUBLE NULL ,
  `n+/100` DOUBLE NULL ,
  `n-/100` DOUBLE NULL ,
  `d+` DOUBLE NULL ,
  `N+/10` DOUBLE NULL ,
  `d-` DOUBLE NULL ,
  `N-/10` DOUBLE NULL ,
  `do` DOUBLE NULL ,
  `No/500` DOUBLE NULL ,
  `balance5` DOUBLE NULL ,
  `d<+` DOUBLE NULL ,
```

```
`N<+/10` DOUBLE NULL ,
`d<-` DOUBLE NULL ,
`N<-/10` DOUBLE NULL ,
`d<o` DOUBLE NULL ,
`N<o/500` DOUBLE NULL ,
`balance<` DOUBLE NULL ,
`100q` DOUBLE NULL ,
`Nq/-10` DOUBLE NULL ,
`Nd/100` DOUBLE NULL ,
`p1.00-1.32` DOUBLE NULL ,
`p1.32-1.73` DOUBLE NULL ,
`p1.73-2.28` DOUBLE NULL ,
`p2.28-3.00` DOUBLE NULL ,
`p3.00-3.95` DOUBLE NULL ,
`p3.95-5.20` DOUBLE NULL ,
`p5.20-6.84` DOUBLE NULL ,
`p6.84-9.00` DOUBLE NULL ,
`n1.00-1.32` DOUBLE NULL ,
`n1.32-1.73` DOUBLE NULL ,
`n1.73-2.28` DOUBLE NULL ,
`n2.28-3.00` DOUBLE NULL ,
`n3.00-3.95` DOUBLE NULL ,
`n3.95-5.20` DOUBLE NULL ,
`n5.20-6.84` DOUBLE NULL ,
`n6.84-9.00` DOUBLE NULL ,
`z1.00-1.32` DOUBLE NULL ,
`z1.32-1.73` DOUBLE NULL ,
`z1.73-2.28` DOUBLE NULL ,
`z2.28-3.00` DOUBLE NULL ,
`z3.00-3.95` DOUBLE NULL ,
`z3.95-5.20` DOUBLE NULL ,
`z5.20-6.84` DOUBLE NULL ,
`z6.84-9.00` DOUBLE NULL );
```