RĪGAS TEHNISKĀ UNIVERSITĀTE
Elektronikas un Telekomunikāciju fakultāte
Transporta Elektronikas un Telemātikas katedra

Elans GRABS
Doktora studiju programmas
"Transporta Datorvadības, Informācijas un Elektroniskās Sistēmas" doktorants

SEVLĪDZĪGĀ TRAFIKA PARAMETRU ANALĪZE TĪKLA VEIKTSPĒJAS PAAUGSTINĀŠANAI, PIELIETOJOT REĀLLAIKA DISKRĒTO VEIVLETU TRANSFORMĀCIJU

Promocijas darbs

Zinātniskais vadītājs
Profesors *Dr. habil. sc. ing.*
E. PĒTERSONS

RTU Izdevniecība
Rīga 2016

Grabs E. Sevlīdzīgā trafika parametru analīze tīkla veiktspējas paaugstināšanai, pielietojot reāllaika diskrēto veivletu transformāciju. Promocijas darbs. – Rīga: RTU Izdevniecība, 2016. – 200 lpp.

Iespiests saskaņā ar ETF Promocijas Padomes "RTU P-08" 2016. gada 21. janvāra lēmumu, protokols Nr. 31.

Šis darbs izstrādāts ar Eiropas Sociālā fonda projekta "Atbalsts RTU doktora studiju īstenošanai" atbalstu.

PROMOCIJAS DARBS IZVIRZĪTS INŽENIERZINĀTŅU DOKTORA GRĀDA IEGŪŠANAI RĪGAS TEHNISKAJĀ UNIVERSITĀTĒ

Promocijas darbs inženierzinātņu doktora grāda iegūšanai tiek publiski aizstāvēts 2016. gada 5. maijā Rīgas Tehniskās Universitātes Elektronikas un Telekomunikāciju Fakultātē, Āzenes ielā 12, telpā 2-38.

OFICIĀLIE RECENZENTI:

Profesors *Dr. sc. ing.* Gunārs Lauks
Rīgas Tehniskā Universitāte, Latvija

Profesors *Dr. sc. ing.* Guntis Bārzdiņš
Latvijas Universitāte, Latvija

Profesors *Dr. sc. ing.* Aleksandrs Grakovskis
Transporta un Sakaru Institūts, Latvija

APSTIPRINĀJUMS

Apstiprinu, ka esmu izstrādājis šo promocijas darbu, kas iesniegts izskatīšanai Rīgas Tehniskajā Universitātē inženierzinātņu doktora grāda iegūšanai. Promocijas darbs zinātniskā grāda iegūšanai nav iesniegts nevienā citā universitātē.

Elans Grabs ……(Paraksts)

Datums: ……

Promocijas darbs ir uzrakstīts latviešu valodā, tajā ir ievads, četras nodaļas, secinājumi, literatūras saraksts, 16 pielikumi, 76 zīmējumi un ilustrācijas, kopā 200 lappuses (129 lappuses bez pielikumiem). Literatūras sarakstā ir 100 nosaukumi.

Satura rādītājs

Nodaļa 1

Sevlīdzīgā Trafika Modelis un Diskrētā Veivletu Transformācija

1.1. Ievads

Pašlaik tiek plaši uzskatīts, ka mūsdienu tīklos trafikam piemīt sevlīdzīguma īpašība [82], t.i. apskatot šādu datu plūsmu (trafiku) dažādos mērogos (sekundēs, minūtēs, stundās, diennaktīs, utt.) var konstatēt, ka tās raksturs saglabājas laikā neatkarīgi no apskatāmā mēroga. Tas tika konstatēts, piemēram, [46], kur autori veica trafika mērījumus dažādos *Bellcore* laboratorijas lokālajos tīklos laika posmā starp 1989. un 1992. gadu. Šajā darbā tika uzskaitītas datu paketes īsajos laika intervālos un pakešu sadalījums tika atainots grafiski atkarībā no laika. Šādi sadalījumi tika apskatīti sīkākā mērogā, samazinot pakešu uzskaitīšanas laika intervāla garumu 10 reizēs patvaļīgi izvēlētajā sadalījuma apakšintervālā.

Rezultātā, tika iegūti trafika sadalījuma grafiki atkarībā no laika dažādos mērogos, no kuriem varēja secināt, ka lokālo tīklu (*Ethernet*) trafika sadalījums izskatās gandrīz vienādi gan garajos (diennaktis), gan īsajos (sekundes) laika intervālos. Tādas pašas īpašības piemīt arī Interneta tīkla trafikam, kā tas ir aprakstīts [90] avotā. Ievērojot to, ka šis avots ir pietiekami novecojis un no tā laika Interneta tīklā multimediju pakalpojumu skaits arvien palielinās (*YouTube* un līdzīgi pakalpojumi), intuitīvi var secināt, ka mūsdienu Interneta trafika sevlīdzīgums ir vēl augstāks, nekā tika uzskatīts [90]. Tas prasa diferencēt dažādus *WEB* servisus pēc klasēm, kā tas ir piedāvāts [7].

Viena atsevišķa lietotāja pieprasījumu Interneta resursam var uzskatīt par gadījumprocesu un tādā gadījumā pieprasītā resursa apjoms arī var būt uzskatāms par gadījumrakstura procesu. Un šis pieprasītais resursa apjoms var svārstīties plašajā diapazonā, sākot ar *Twitter* ziņojumu un beidzot ar filmu *HD* kvalitātē. Turklāt abu šo notikumu varbūtības nevar būt ignorētas, salīdzinot tās savā starpā. Tieši tāda uzvedība ir raksturīga sevlīdzīgajiem procesiem, kurus labi apraksta tā saucamie **"sadalījumi ar garajām astēm"** [46, 86, 90].

Praksē šis resursa apjomu nevienmērīgums izvirza augstas prasības apstrādes mezgla veiktspējai μ un/vai buferatmiņas apjomam pieprasījuma glabāšanai līdz laika momentam, kad procesors pabeigs apstrādāt iepriekšējo pieprasījumu. Apskatot buferatmiņas pieprasījumu rindu ar garumu K atkarībā no noslodzes koeficienta ρ, kas, savukārt, ir atkarīgs no pieprasījumu intensitātes λ un apstrādes mezgla veiktspējas μ, ir iespējami vairāki gadījumi:

1. Rindas garums K samazinās. Jauni pieprasījumi atnāk reti vai nenoslogo pietiekami ilgi apstrādes mezglu, tāpēc momentos, kad procesors nav aizņemts tiek apstrādāti pieprasījumi no buferatmiņas ar turpmāko buferatmiņas atbrīvošanu. Situācija ir iespējama praksē, jo Interneta trafikā uzliesmojumi mainās ar pauzēm.

2. Rindas garums svārstās noteiktās vērtības $K = K_0$ apgabalā. Šajā gadījumā pieprasījumu intensitāte λ ir tuva[1] faktiskajai apstrādes mezgla veiktspējai μ, t.i., vidēji apstrādes

[1] Šeit ir jāatzīmē ka runa neiet par vienādību $\lambda = \mu$, praksē pieprasījumu intensitātei λ jāsastāda 50-70%

mezgls paspēj apstrādāt katru pieprasījumu. Rezultātā rindas garums K nepalielinās un nesamazinās, vai arī tā izmaiņas nav ievērojamas.

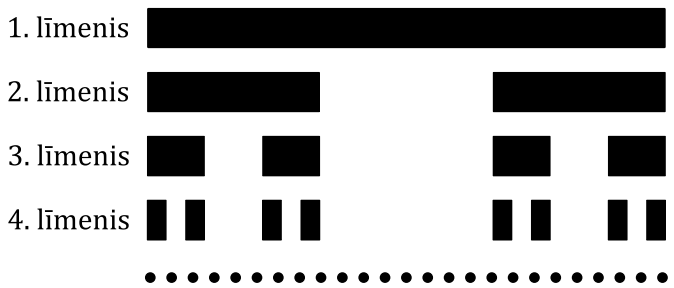
3. Rindas garums K neierobežojami pieaug laikā. Šajā gadījumā jauno pieprasījumu intensitāte λ tiecās uz vai pat pārsniedz apstrādes mezgla veiktspēju μ. Tas ir īpaši aktuāli sevlīdzīgā trafika apstākļos, kuram tāda notikuma varbūtība ir pietiekami augsta salīdzinājumā ar klasisko Puasona trafiku ar eksponenciāli sadalītajiem laika intervāliem starp pieprasījumiem [90].

Diemžēl, rindošanas teorijā labi apskatīts un matemātiski pierādīts Puasona trafika modelis nav piemērojams reālajam tīklu trafikam. Tā, piemēram, Puasona trafikam pie noslodzes koeficienta $\rho = \lambda/\mu = 0,8$ saskaņā ar [86] ir jānodrošina buferatmiņa $K = 55$ paketēm lai nodrošinātu pakešu zudumu varbūtību $P_{\text{loss}} = 10^{-6}$. Savukārt, sevlīdzīgajam trafikam novērtēt nepieciešamo buferatmiņas apjomu ir ievērojami grūtāk un ir jāsasniedz tāds pats apkalpošanas kvalitātes (QoS) līmenis. Tas ir izskaidrojams ar to, ka sevlīdzīgajam trafikam nevar prognozēt kad notiks kārtējais pieprasījumu "uzliesmojums", kā arī cik ilgajā laika intervālā tas saglabāsies. Simulācijas rezultātu analīzē tas ir paskaidrots sīkāk.

1.2. Sevlīdzīguma parametrs

Sevlīdzīguma koncepcija ir cieši saistīta ar haosa un fraktāļu teoriju, kā arī ar pakāpju rindām. Pirmais zinātnieks, kas pievērsa uzmanību šīm parādībām, bija Benuā Mandelbrots, kurš piedāvāja veidu, kā var aprakstīt dabas sarežģītās formas ar matemātikas palīdzību [50]. Konkrētāk, runa ir par fraktāļiem – sevlīdzīgajām struktūrām, kas ir plaši sastopamas apkārtesošajā pasaulē. Parādība ar sevlīdzīguma īpašībām vienādi izskatās vai uzvedās, apskatot to dažādos mērogos. Pie tam mērogu mainīt var gan telpiskajām koordinātēm, gan laika koordinātei. Visvienkāršāk ir izskaidrot sevlīdzīguma būtību ar piemēru, par pamatu paņemot tā saucamo Kantora kopu, kuru pirmoreiz 1883. gadā apskatīja G. Kantors – kopu teorijas pamatlicējs matemātikā.

Par pamatu tiek ņemts intervāls $[0, 1]$, kuru var grafiski attēlot ar nogriezni. Šajā nogrieznī tiek izgriezta vidējā trešdaļa un tā veidojās divi nogriežņi, katrs no kuriem ir 3 reizēs īsāks par sākotnējo. Šiem nogriežņiem atbilst attiecīgi kopas $[0, 1/3]$ un $[2/3, 1]$, jeb, precīzāk, šo kopu apvienojums $[0, 1/3] \cup [2/3, 1]$. Procedūru atkārto ar katru no iegūtajiem nogriežņiem, iegūstot kopu apvienojumu $[0, 1/9] \cup [2/9, 1/3] \cup [2/3, 7/9] \cup [8/9, 1]$. Sadalīšana tiek bezgalīgi atkārtota, kamēr neveidosies atsevišķo punktu kopa, kurā nav divu tādu punktu, kas būtu savienoti ar taisno līniju. Grafiski šī procesa pirmie 4 līmeņi ir parādīti 1.1. att.



1.1. att. Visvienkāršākais fraktālis, Kantora kopa.

Apskatot 1.1. att. nogriežņus kā laika skalas, varētu secināt, ka katrā nākamā līmenī mērogs tiek palielināts 3 reizēs, turklāt gan kreisā, gan labā puse pilnīgi atkārto oriģinālo nogriezni. Līdz ar to, var definēt īpašības, kas piemīt jebkurai sevlīdzīgai parādībai vai procesam:

---

no apstrādes mezgla veiktspējas μ. Apskatot gadījumu ar augstāko noslodzi, t.i., $\lambda > 0,7\mu$ parādās nopietnās grūtības tīkla veiktspējas uzvedības prognozēšanā [90].

1. Procesam piemīt struktūra, kuru var noteikt jebkurā mērogā. Samazinot mērogu turpmāk, joprojām varēs redzēt divus vienāda garuma intervālus, kas atdalīti savā starpā.

2. Šī struktūra nepārtraukti atkārtojās. Sevlīdzīgā procesa struktūru veido tās pašas struktūras samazinātās kopijas visos mērogos.

Protams, reālajā pasaulē šīs īpašības nav spēkā bezgalīgajam mērogu skaitam. Sasniedzot noteikto mērogu, šī struktūra parasti pazūd – taču daudziem procesiem sevlīdzīguma īpašības[2] izpaužas lielajā mērogu skaitā. Kā tika iepriekš atzīmēts, tas attiecas arī uz trafiku pirms diviem gadu desmitiem [46]. Patiešām, apskatot 1.1. att., var pieņemt, ka viszemāko līmeni veido atsevišķas paketes, starp kurām ir dažāda garuma pauzes. Samazinot mērogu, veidojās atsevišķās pakešu grupas, kas joprojām ir atdalāmas savā starpā ar pauzēm (t.s. *ON/OFF* trafiks). Ja turpināt samazināt mērogu, var konstatēt, ka situācija saglabās – veidojās grupu grupas, atdalāmas ar pauzēm. Šīs process atkārtojās, sekojot mēroga samazināšanai – tieši pie tādiem rezultātiem nonāca pētnieki [46], kad analizēja lokālo tīklu trafika mērījumu rezultātus.

Matemātiski, vispārīgajā gadījumā stohastisko (gadījumrakstura) procesu sauc par sevlīdzīgo, ja tas nav atkarīgs no nepārtrauktā laika mainīgā izmaiņām (jeb saka, ka process ir invariants attiecībā uz laika mērogu). Stohastiskais process ir statistiski sevlīdzīgs, un sevlīdzīguma parametrs ir vienāds ar H (turklāt $0,5 \leq H \leq 1$), ja pie jebkurām reālām pozitīvām a vērtībām procesiem $x(t)$ un $a^{-H}x(at)$ ir vienādi statistiskie raksturojumi. Šeit H ir tā saucamais *sevlīdzīguma parametrs*, jeb Hersta parametrs, kas parāda statistiskā procesa stabilitātes mēru, vai tā ilgtermiņa atkarības pakāpi. Ja $H = 0,5$, tad par tādu ilgtermiņa atkarību runāt neiet – process nav korelēts, taču, jo tuvākā ir H vērtībā 1, jo augstākā ir ilgtermiņa atkarība.

Šo sevlīdzīguma parametru nosauca par godu pētniekam H. E. Herstam, kurš veltīja savu dzīvi Nīla un citu upju līmeņa pētījumiem, kā arī ūdens uzglabāšanas problēmām. Būdams hidrologs pēc savas izglītības, Hersts pētīja dažādu upju plūsmu ilgtermiņa raksturojumus. Viņš pamanīja, ka dažām upēm ir relatīvi zemas fluktuācijas, turpretī citām upēm, tādām ka Nīla, situācija bija atšķirīga. Īstermiņa perspektīvā, kā to var sagaidīt, Nīlas ūdens līmenis svārstījās katru gadu. Varēja sagaidīt, ka ilgstošajā laikā šīs fluktuācijas dos vidējo vērtību, un Nīlas ūdens līmenis mainīsies šaurajā diapazonā ar labiem un sliktiem periodiem, kas bieži nomaina viens otru. Taču realitātē viņš konstatēja, ka Nīla uzvedas pavisam savādāk – ilgtermiņa perspektīvā ilgstoši sausuma periodi seko pēc ilgstošajiem augstā ūdens līmeņa periodiem, kuros uzpludinājumi notika gandrīz katru gadu. Hersts pamanīja, ka 800 gadu laika posmā Nīla upes ūdens līmenī var novērot sevlīdzīguma īpašības un sāka projektēt ideālo rezervuāru upes ūdens līmeņa regulēšanai pēc veiktajiem novērojumiem, lai izejas plūsma būtu vienmērīga laikā.

Hersts noteica, ka daudzām dabas parādībām sevlīdzīguma parametrs H pieņem vērtības no 0,7 līdz 0,9. Jo lielāka ir H parametra vērtība, jo straujākas datu izmaiņas var sagaidīt (protams, šeit runa ir par principiālo sagaidīšanu, nevis par prognozēšanu konkrētajam laika momentam). [86] veiktie pētījumi, parāda, ka augsta sevlīdzīguma pakāpe ($H = 0,9$) piemīt gan lokālo tīklu (*Ethernet*) trafikam, gan Interneta trafikam. Attiecībā uz sevlīdzīgo trafiku ir jāatzīmē vēl viens svarīgais moments. Spriežot pēc [60] veiktajiem pētījumiem, sevlīdzīgo trafiku var iedalīt divās lielajās kategorijās:

1. Sevlīdzīgajā lietojumprogrammu līmeņa trafikā – šajā gadījumā sevlīdzīgums piemīt pašam avotam, lietojumprogrammai, neatkarībā no konkrētās tīkla konfigurācijas uz doto brīdi. Tādā veidā, šo sevlīdzīgumu var neievērot, izvēloties piemēroto buferatmiņas apjomu [90, 60]. Pārsvarā, tādu trafiku veido video-pārraide ar mainīgo ātrumu (*VBR*).

2. Sevlīdzīgajā tīkla līmeņa trafikā – kuru veido tīkla nepastāvīgā struktūra, mainīgs lietotāju skaits un protokolu mehānismi (piemēram, *TCP* protokola sastrēgumu kontrole). Šajā gadījumā pat pie nemainīgajiem lietojumprogrammas parametriem veiktspēja var ievērojami mainīties atkarībā no konkurējošo lietotāju skaita un tīkla noslodzes.

[2] Jāatzīmē, ka realitātē runa nav par stingru atbilstību, bet tikai par līdzīgumu.

1.3. Literatūras apskats

Sevlīdzīgums ir stipri izteikts mūsdienīgajos tīklos. Daudzi pētījumi jau pierādīja, ka vecais Puasona trafika modelis vairs nav piemērojams sevlīdzīgajam trafikam, taču arī pēdējos gados jaunie pētījumi apstiprina šo faktu un pat prognozē, ka nākotnē vecais modelis nebūs piemērots nākamās paaudzes tīklos [48]. Turklāt vecos Markova ķēdes paņēmienus var izmantot papildinājumā, lai precīzāk analizētu trafiku [64]. Interesants fakts ir tāds, ka trafiks ir sevlīdzīgs ne tikai datoru tīklos – piemēram, [18] tika noteikts, ka integrēto mikroshēmu savienojuma tīklā arī ir sevlīdzīgs trafiks (runa ir par sarežģītajām mikroshēmām, tādām kā, piemēram, *DSP*). Tāpēc ir svarīgi noteikt sevlīdzīguma pakāpi, novērtējot Hersta parametru. Šajā promocijas darbā tāda mērķa sasniegšanai tiek izmantota veivletu transformācija, ko pirmoreiz piedāvāja [1]. Šis darbs rosināja pētnieku interesi par šiem jautājumiem un tika turpināti pētījumi par veivletu transformācijas Hersta parametra novērtētāju. Piemēram, darbā [47] tika parādīts, ka veivletu nulles momentu izvēle[3] nepalielina novērtēšanas precizitāti. Veivletu transformācija Hersta parametra noteikšanā ir joprojām aktuāls risinājums, par ko liecina pēdējo gadu darbi, kuros tiek meklēti jauni veivleti novērtēšanas raksturojumu uzlabošanai. Piemēram, [53] tiek izmantotas kompleksas veivletu bāzes funkcijas.

Tādam trafikam ir svarīgi veidot kontroles sistēmas, lai pārvaldītu tīkla resursus efektīvāk. Šajā jomā ir veikti daudzi pētījumi. Lai vadītu trafika plūsmu pārslogojumus, trafiku bieži iedala klasēs, katra no tām tiek apstrādāta ar dažādām *QoS* prasībām, un tādā veidā tiek veikta tīkla optimizācija, kā tas ir izdarīts, piemēram, [34]. Literatūrā ir sastopams, ka klasifikāciju ir iespējams veikt tieši pēc Hersta parametra vērtībām, piemēram, to ir piedāvāts darīt publikācijā [57]. Nopietns pētījums par trafika klasifikāciju ar diskrētās veivletu transformācijas metodi ir paveikts [61]. Turklāt tādu klašu skaits strauji pieaug, un ir nepieciešams veidot arī jaunus mehānismus savienojumu vadībai (*Connection Admission Control, CAC*) [77]. Vēl viens aktuāls jautājums ir buferatmiņas pārvaldība, lai kontrolētu rindas garumu ar dažādiem algoritmiem [3], [2].

Pētījumi tiek veikti gan stacionārajiem tīkliem, gan arī bezvadu tīkliem *WiFi*, piemēram, [76] ir veikta sevlīdzīgā trafika analīze ar mainīgo paketes garumu un uzdotām *QoS* prasībām, bet [38] – dažādu protokolu ietekmes analīze bezvadu režģtīklos (*wireless multi-hop networks*). Sevlīdzīgā trafika raksturu ievēro pat *LTE* tīklu optimizācijā [71]. Savukārt telefona zvanu pētījumi esošajos $2G/3G$ mobilajos tīklos parāda, ka balss datu trafiks joprojām atbilst Puasona modelim [13]. Heterogēnajos bezvadu tīklos sevlīdzīgais trafiks prasa arī radio-resursa pārvaldību, kā arī veiktspējas analīzi un [73] ir piedāvāts algoritms tādai vadībai.

Ir mēģinājumi izveidot Hersta parametra novērtēšanas mezglu, kurš strādātu reāllaikā, piemēram, ar *R/S* statistikas paņēmienu [6]. Rezultāti parāda, ka tas ir iespējams un, pats galvenais, ir lietojams datortīklu trafikam. Turklāt, tiek veikti pētījumi par trendu ietekmi uz Hersta novērtēšanas mezgla stabilitāti, piemēram, [78]. Šajā publikācija tika novērtēta stabilitāte dažādām Hersta parametra novērtēšanas metodēm, tajā skaitā arī veivletu transformācijas metodei. Viens no secinājumiem bija tāds, ka gadījumā ar lineārajiem trendiem veivletu transformācijas metodei ir priekšrocības Hersta parametra novērtēšanā. Vēl viens salīdzināšanas raksts dažādiem Hersta parametra novērtētājiem ir [11], kurā tiek salīdzināta to precizitāte. Cits aktuāls jautājums ir Hersta (sevlīdzīguma) parametra izmaiņu reģistrēšana [16], t.i., jākonstatē ka Hersta parametrs ir mainījies un attiecīgi jāpieskaņo sistēmas parametri, tajā skaitā arī paša Hersta parametra novērtēšanas modulim, jo novērtējuma precizitāte ir atkarīga no trafika sevlīdzīguma pakāpes.

Pastāv vēl viens mūsdienīgais transformācijas paveids, kurš paliek populārs – tā ir Empīriskā Modu Dekompozīcija (*EMD*), kurā var lietot diskrētus veivletus kā bāzes funkcijas un tādā veidā analizēt tīkla trafiku, meklējot tajā anomālijas, piemēram, [20],[40]. Šo transformāciju vēl sauc

[3] M nulles momenti veivleta bāzes funkcijai nosaka ģenerējamā ar to polinoma maksimālo pakāpi $M-1$, ar kuru bāzes funkcija būs ortogonāla. Jo vairāk nulles momentu ir bāzes veivletam, jo precīzāk un pilnīgāk tā bāze apraksta sarežģītus signālus.

par Hilberta-Huanga transformāciju (*HHT*) un to var lietot signāla izvirzīšanai tā saucamajās empīriskajās modās (signāla komponentēs), līdzīgi kā to dara jebkura cita transformācija. Taču šī transformācija nav apriori piesaistīta jebkādām bāzēm. Tīkla trafika anomālijas bija mēģināts noteikt arī ar frakcionālo Furjē transformāciju (*FRFT*) darbā [79], novērtējot Hersta parametru. Ir veikts salīdzinājums ar veivletu transformācijas un *EMD* Hersta parametra novērtētāju. Savukārt diskrēto veivletu transformāciju tādu anomāliju noteikšanai tika piedāvāts izmantot jau sen, uz ko norāda eksistējošie darbi, piemēram, [21].

Hersta parametrs tika novērtēts arī *Hadoop* trafika anomāliju meklēšanā [45]. *Hadoop* ir cieši saistīts ar lielo apjomu datiem ("big data") un sadalīto apstrādi. Pētījuma rezultāti parāda, ka trafika anomālijām parasti ir ievērojami lielāka Hersta parametra vērtība, nekā regulārajam trafikam. Kā var redzēt, daudzi pētījumi ir orientēti tieši uz trafika anomāliju noteikšanu, novērtējot Hersta parametru. Ir mēģināts pat izveidot prognozēšanas sistēmu, kas iepriekš brīdinātu par tādas situācijas rašanās iespējamību [15]. Diskrētās veivletu transformācijas papildināšana ar minimālo kvadrātu metodi [37] darbā parādīja, ka tāda prognoze ir precīzāka un ātrāka *MPEG-4 VBR* video un sevlīdzīgā tīklu trafika gadījumā un var būt veikta reāllaikā. Tāda prognozēšana var būt uzlabota ar neironu tīkliem [19].

Pētījumi parāda, ka tīkla trafika Hersta parametrs ir atkarīgs no lietotāja, viņa vajadzībām un laika sadalījuma, kuru viņš patērē tīklā. Darbā [29] ir mēģināts noskaidrot, vai dažādās dienās Hersta parametrs ir prognozējams pēc iepriekšējām dienām, pētot Interneta aiztures. Pētījuma rezultāti parāda, ka tāds periodiskums patiešām eksistē, turklāt ir stipri izteikta Hersta parametra atšķirība dienas un nakts laikā.

Lai Hersta parametru varētu novērtēt reāllaikā, ir jāizveido reāllaika diskrētās veivletu transformācijas algoritms. Tādi mēģinājumi jau ir sen veikti, piemēram, darbā [52]. Tādu transformāciju var veikt ar zemajām skaitļošanas izmaksām, izmantojot filtru bankas [66].

Apkopojot visu informāciju, saprotams, ka veivletu transformācijas paņēmiens atver plašas iespējas tīklu trafika analīzē un dod iespēju novērtēt Hersta parametru, veikt trafika klasifikāciju (tajā skaitā, izmantojot novērtēto Hersta parametru), prognozēt tīkla trafiku [69]. Turklāt diskrēto veivletu transformāciju izmanto arī citos uzdevumos, kas ir saistīti ar tīkliem – gan ar vadu tīkliem, gan ar bezvadu tīkliem. Piemēram, ir piedāvāts algoritms, kas var atklāt *WPS* uzbrukumus bezvadu tīklā [56]. Izmantojot diskrēto veivletu transformāciju ar neironu tīkliem [17], ir piedāvāts prognozēt nepieciešamo resursu apjomu mobilajos *ad-hoc* tīklos.

Kopumā, literatūras analīze parāda, ka trafika sevlīdzīgums ir joprojām aktuālā tēma un diskrēto veivletu transformāciju var lietot daudzajos uzdevumos tāda trafika klasificēšana un kontrolē.

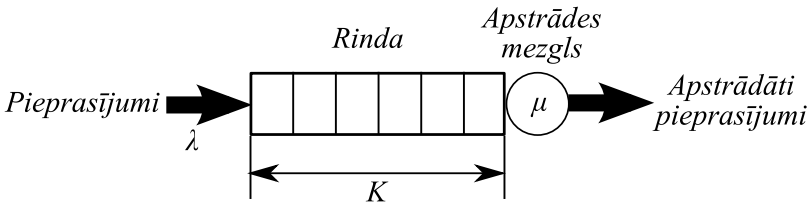
1.4. $P/M/1/K$ trafika simulācijas modelis

Datortīklu sfērā bieži nepieciešams prognozēt sistēmas noteikto izmaiņu rezultātus, piemēram, noslodzes palielināšanu. Citos gadījumos nepieciešams izveidot datortīkla projektu pēc noteiktām prasībām. Visos gadījumos visvairāk interesē tieši sistēmas veiktspēja – atkarībā no konkrētā uzdevuma runa var būt par reakcijas laiku, caurlaides spēju, vai citiem parametriem. Lai novērtētu sistēmas veiktspēju, jālieto prognozēšanas mehānismus, un tādu problēmu risināšanai var izmantot analītiskos modeļus no rindošanas teorijas [90]. Jāatzīmē, ka pati rindošanas teorijas matemātiskā sastāvdaļa ir sarežģīta, taču šīs teorijas praktiskais pielietojums daudzos gadījumos ir vienkāršs.

Rindu modelēšanā galveno pieņēmumu apzīmēšanai lieto tā saucamo **Kendala notāciju** ar šādu veidu: $X/Y/N/K$, kur X uzdod sadalījumu starp pieprasījumu intervāliem, Y uzdod apstrādes laika sadalījumu, N – apstrādes elementu skaitu un K – buferatmiņas apjomu, kas nosaka maksimālo rindas garumu. Visplašāk izmantojamus sadalījumus šajā notācijā ir pieņemts apzīmēt šādi:

- ***G*** – patvaļīgais pieprasījumu vai apstrādes laika intervālu sadalījums;
- ***GI*** – patvaļīgais pieprasījumu vai apstrādes laika intervālu sadalījums ar papildus ierobežojumu – šiem intervāliem ir jābūt neatkarīgiem savā starpā;
- ***M*** – negatīvais eksponenciālais sadalījums;
- ***P*** – Pareto sadalījums;
- ***D*** – determinētais intervāls starp pieprasījumiem vai fiksētais apstrādes laiks.

Tādā veidā modeli $P/M/1/K$ veido viens apstrādes mezgls, laika intervāliem starp pieprasījumiem ir Pareto sadalījums, apstrādes laikam ir eksponenciāls sadalījums un buferatmiņas apjoms sastāda K vienības[4]. Tāda visvienkāršākā modeļa struktūra ir parādīta 1.2. att.



1.2. att. Vienkāršāka rindošanas sistēma ar vienu apstrādes elementu.

Tādas rindošanas sistēmas ieejas informācija ir:

- Pieprasījumu intensitāte λ un tai atbilstošais vidējais laiks starp pieprasījumiem $T_\mathrm{a} = 1/\lambda$;
- Apstrādes ātrums (intensitāte) μ un tam atbilstošais vidējais apstrādes laiks $T_\mathrm{s} = 1/\mu$;
- Apstrādes elementu skaits $N = 1$;
- Buferatmiņas apjoma ierobežojums K.

Savukārt rindošanas sistēmas analīzes uzdevums reducējās uz šādu lielumu noteikšanu pēc uzdotās ieejas informācijas:

- gaidīšanas laiks rindā T_w;
- pieprasījumu skaits rindā W;
- pieprasījumu gaidīšanas laiks sistēmā T_R;
- pieprasījumu skaits sistēmā R.

Šiem lielumiem jānosaka statistiskie rādītāji – vidējās vērtības (T_ω, ω, T_r, r) un novirzes no tām (σ_{T_ω}, σ_ω, σ_{T_r}, σ_r). Cits interesējošais lielums, kuru bieži mēģina novērtēt – buferatmiņas pārpildīšanas varbūtība P_loss (pieprasījuma atteikums). Piemēram, jānosaka tādu buferatmiņas apjoma ierobežojumu K, lai pārpildīšanas varbūtība $P_\mathrm{loss} \leq 0,001$. Lai varētu noteikt visus šos lielumus, ir nepieciešamas pilnās zināšanas par pieprasījumu un apstrādes laika intervālu varbūtību sadalījumu. Taču pat tādos gadījumos kad šī informācija ir pieejama, rezultējošās formulas sanāk sarežģītas.

Klasiskās rindošanas teorijas analītiskās metodes ir spēkā tikai Puasona trafikam, t.i., tādam trafikam, kurā intervāliem starp pieprasījumiem ir eksponenciāls varbūtību sadalījums. Eksponenciālajā sadalījumā gadījumrakstura lieluma (piemēram, pieprasītā Internet-faila apjoms, baitos) varbūtība asimptotiski tuvojas nullei, pieaugot pašam gadījumrakstura lielumam. No tāda viedokļa, lielā faila lejuplāde no Interneta ir ļoti rets notikums. Pilnīgi iespējams, ka

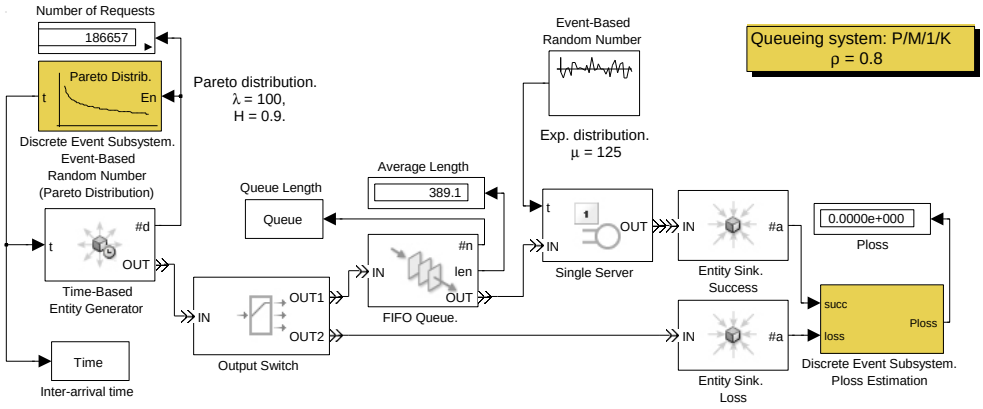
[4]Piemēram, tīkla trafika gadījumā par tādu vienību var uzskatīt paketi.

Interneta pirmsākumos tas bija atbilstošais apraksts, taču mūsdienās var pateikt gandrīz otrādi – lielo failu lejuplāde ir tikpat biežais notikums kā īso *Twitter* ziņojumu lasīšana. Līdz ar to, mainās arī varbūtību sadalījuma funkcijas veids – sadalījumam veidojās tā saucamā "garā aste". Tādos sadalījumos novirzēm no vidējās vērtības ir ievērojami augstākās varbūtības, nekā eksponenciālajā, Gausa un citos sadalījumos. Turklāt pašas vidējās vērtības, kas tiek aprēķinātas pēc datu kopuma, nav informatīvās un ir nestabilas

Un, patiešām, tādus sadalījumus izmanto sevlīdzīgo procesu statistiskajai aprakstīšanai. Piemēram, [91] autors apstrīdēja Gausa sadalījuma pielietojamību finanšu tirgū[5] un apskata šo finanšu tirgus procesus kā sevlīdzīgus, lietojot sadalījumus ar garajām astēm (piemēram, Pareto sadalījumu). Jāatzīmē, ka Pasaules krīzes apstākļos šī pieeja parādīja savu efektivitāti.

Diemžēl, klasiskajā rindošanas teorijā paredzēts izmantot sadalījumus ar "vieglajām astēm", turklāt procesam ir jābūt neatkarīgam. Tieši tāpēc tās analītiskās metodes nav izmantojamas sevlīdzīgā trafika analīzē un ir jāizmanto skaitliskā modelēšana, veidojot datoros simulācijas modeļus. Pastāv vairākas modelēšanas sistēmas, kurās no gataviem blokiem var izveidot modeli – ir iespējams veidot savu modelī arī jebkurā programmēšanas valodā. Orientējoties uz Veivletu transformācijas pielietošanu, tika izvēlēta *Matlab* datorprogramma ar *Simulink* vizuālās modelēšanas vidi, it īpaši ievērojot to, ka *Simulink* modelēšanas videi var pievienot papildus rīkus (t.s. **toolbox**), un viens no tiem ir *SimEvents*, kurš ļauj veidot rindošanas sistēmas. Jāatzīmē, ka, rindošanas teorija kopumā ir izmantojama daudz plašākajā uzdevumu lokā, neierobežojoties ar sakaru sistēmu analīzi. Piemēram, [99] autori veiksmīgi lieto rindošanas teoriju un *SimEvents* darbarīku trafika analīzei dzelzceļa posmā.

1.3. att. ir parādīts pilnais *Simulink* $P/M/1/K$ modelis sevlīdzīgā trafika sistēmas modelēšanai. Analoģiski 1.2. att. parādītajai rindošanas sistēmai, šajā modelī ir avots "**Time Based Entity Generator**", kas veido pieprasījumus kā notikumus (entities). Šie pieprasījumi tiek padoti apstrādes mezglam "**Single Server**". Ja serveris jau ir aizņemts, pieprasījums tiek pievienots rindai "**FIFO Queue**"[6]. Vienas ieejas divu izeju pārslēdzējs "**Output Switch**" kontrolē rindas garumu tā, lai kopējais pieprasījumu skaits rindā nepārsniegtu modelī iestādīto K vērtību. Kad visa buferatmiņa ir aizpildīta, jaunais pieprasījums tiek noraidīts (padots "**OUT2**" izejā) un neatgriezeniski pazaudēts. Apstrādāto pieprasījumu skaita n_{p} un noraidīto pieprasījumu skaita n_0 uzskaite notiek blokos "**Entity Sink Success**" un "**Entity Sink Loss**", attiecīgi.



1.3. att. Sevlīdzīgā trafika rindošanas sistēmas modelis *Simulink* vidē.

[5] Arī Mandelbrots [50] noteiktajā savas dzīves posmā nodarbojās ar finanšu tirgus pētījumiem.

[6] *FIFO* – F*irst In First Out*, t.i., pieprasījumu apstrāde šajā rindā notiek to atnākšanas kārtībā, bet eksistē arī citi mehānismi, tajā skaitā arī ar prioritātēm, kas arī ir pieejamas *SimEvents* modulī.

Turklāt šiem blokiem, kurus parasti izmanto kā modeļa galapunktus (terminatorus), ir ieslēgtas izejas ar mērķi aprēķināt pieprasījuma zuduma varbūtību pēc formulas:

$$P_{\text{loss}} = \frac{n_0}{n_{\text{p}} + n_0}. \tag{1.1}$$

Diskrētā apakšsistēma, t.i., lietotāja veidots bloks "**Discrete Event Subsystem, Ploss Estimation**", realizē (1.1) aprēķināšanu reāllaikā un rezultāts tiek padots indikatorā "**Ploss**". Izmantojot konteksta izvēlni, katram blokam var mainīt iestādījumus ieslēdzot/atslēdzot papildus izejas vai ieejas. Tādā veidā pieprasījumu avotam "**Time-Based Entity Generator**" tika ieslēgts izejas ports $\#d$, uz kuru tiek padots kārtējā pieprasījuma numurs – tas tiek atainots indikatorā "**Number of Requests**"; kā arī buferatmiņas blokam "**FIFO Queue**" tika ieslēgti divi izejas porti: $\#n$, no kura *MATLAB* vides mainīgajā "**Queue**" tiek veidots rindas garuma masīvs katram laika momentam un len, no kura uz indikatoru ar nosaukumu "**Average Length**" tiek padota informācija par rindas vidējo garumu.

Lai uzdotu $P/M/1/K$ modelim raksturīgus pieprasījumu intervālu un apstrādes laika varbūtību sadalījumus, avota elementam "**Time-Based Entity Generator**" un apstrādes mezglam "**Single Server**" ir jāieslēdz sinhronizācijas ieejas portus t, kuriem jāpievieno attiecīgo sadalījuma likumu gadījumskaitļu ģeneratorus – avotam Pareto sadalījuma skaitļu ģeneratoru, bet apstrādes mezglam – eksponenciāla sadalījuma skaitļu ģeneratoru. Jāatzīmē, ka avota gadījumā bloka parametros var izvēlēties vienu no iebūvētajiem ģeneratoriem, taču sadalījumu sarakstā nav neviena sadalījuma ar "garajām astēm".

Apstrādes laika gadījumrakstura lieluma ģeneratoram pietiek ievietot tā paša *SimEvents* moduļa bloku "**Event-Based Random Number**", izvēlēties sadalījuma veidu "**Exponential**" un uzdot matemātisko cerību "**Mean**" – vidējo apstrādes laiku $T_{\text{s}} = 1/\mu$. Savukārt pieprasījumu intervālu ģenerēšanai ar Pareto likumu ir jāizmanto attiecīgais gadījumskaitļu ģenerators, kurš nav iekļauts *SimEvents* modulī. Arī *Matlab* iebūvētais Pareto gadījumskaitļu ģenerators nevar būt izmantojams šajos nolūkos tiešajā veidā, jo tam ir jāuzdod sadalījuma parametri, kuri jāpārrēķina no pieprasījumu intensitātes λ un sevlīdzīguma parametra H.

Vispārīgajā gadījumā, Pareto sadalījumam ir trīs parametri [98], bet tā varbūtību sadalījuma funkcijai ir šāda forma:

$$F(x) = 1 - \left(\frac{\gamma}{x + \gamma - \beta}\right)^{\alpha}, \tag{1.2}$$

kur α ir saspiešanas (formas) koeficients, β ir nobīdes (minimālās vērtības) parametrs un γ ir mēroga parametrs. Atkarībā no šo parametru vērtībām, var pierakstīt dažādas Pareto sadalījuma likuma formas (ar 1 parametru, ar 2 parametriem un ar 3 parametriem) [98]. Visplašāk sevlīdzīgā trafika ģenerēšanai lieto vienu no 2 parametru formām [98, 90, 86], kas ir (1.2) atsevišķais gadījums ja $\gamma = \beta$. Tādā gadījumā, (1.2) izteiksmi var pierakstīt šādi:

$$F(x) = 1 - \left(\frac{\beta}{x}\right)^{\alpha}, \tag{1.3}$$

savukārt tāda gadījumprocesa Pareto varbūtību blīvuma funkcija ir pierakstāma šādi:

$$f(x) = \frac{\alpha}{\beta}\left(\frac{\beta}{x}\right)^{\alpha+1}. \tag{1.4}$$

Tāda sadalījuma α parametrs nosaka sevlīdzīgā gadījumprocesa vidējo vērtību un novirzi no tās. Ja $\alpha \leq 2$, procesam ir bezgalīgā dispersija. Savukārt, ja $\alpha \leq 1$ – tad gan procesa vidējā vērtība, gan tā dispersija būs bezgalīgie lielumi. Reālajiem sevlīdzīgajiem procesiem dispersija patiešām ir bezgalīga, savukārt vidējā vērtība ir galīgā. Tas nozīmē, ka Pareto sadalījuma α parametram ir jābūt robežās $1 < \alpha \leq 2$. Tādā veidā var novērtēt [90] šo α parametru pēc

sevlīdzīguma parametra H, ja sevlīdzīgajiem procesiem $0,5 < H \leq 1$:

$$\alpha = 3 - 2H. \tag{1.5}$$

Savukārt Pareto sadalījuma β parametru var novērtēt pēc α parametra un vidējās pieprasījumu intensitātes $\lambda = E[T_{\mathrm{a}}]$. Ievērojot to, ka $1 \leq \alpha \leq 2$, tādam sevlīdzīgajam procesam būs galīgā vidējā vērtība, kuru var aprēķināt pēc formulas:

$$\mathrm{E}[T_{\mathrm{a}}] = \frac{\alpha\beta}{\alpha - 1}, \qquad 1 \leq \alpha \leq 2. \tag{1.6}$$

Tādā gadījumā, izvēloties noteikto rindošanas sistēmas vidējo laika intervālu starp pieprasījumiem T_{a}, var izteikt Pareto sadalījuma β parametru no (1.6) izteiksmes šādi:

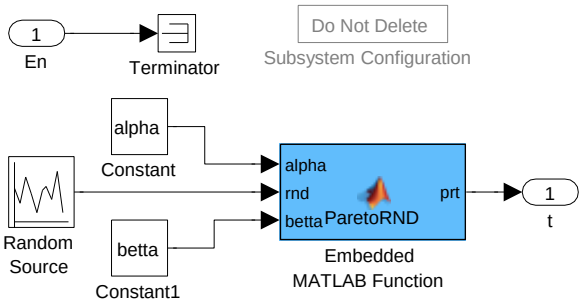
$$\beta = \frac{\alpha - 1}{\alpha\lambda}, \qquad 1 \leq \alpha \leq 2. \tag{1.7}$$

Kad ir zināmi ģenerējamā sevlīdzīgā procesa sadalījuma parametri, var aprēķināt gadījumrakstura lielumu ar Pareto sadalījuma likumu, par pamatu paņemot vienmērīgi sadalīto gadījumrakstura lielumu R, pēc šādas formulas:

$$x = \frac{\beta}{R^{1/\alpha}}, \tag{1.8}$$

kur $R \in [0, 1]$ ir vienmērīgi sadalītais gadījumrakstura lielums.

Tādā veidā, vienā blokā var samērā vienkārši realizēt gan Pareto sadalījuma parametru aprēķināšanu pēc uzdotās trafika intensitātes λ un sevlīdzīguma parametra H, gan ģenerēt gadījumskaitli, kurš atbilstu tādam sadalījumam. Šiem nolūkiem tika izveidots jaunais *Simulink* bloks "**Discrete Event Subsystem, Event-Based Random Number (Pareto Distribution)**", kuru var sīkāk apskatīt 1.4. att.



1.4. att. Pareto ģeneratora apakšsistēma rindošanas sistēmas modelī no 1.3. att.

Apakšsistēmai ir viena ieeja "**En**", kura ir nepieciešama apakšsistēmas iedarbināšanai (no 1.3. att. var redzēt, ka šajā ieejā tiek padots kārtējā pieprasījuma numurs) un nav nepieciešama nekādos citos nolūkos, un viena izeja "**t**", no kuras 1.3. att. modelī tiek padoti laika intervāli notikumu ģeneratoram, kurš veic pieprasījumus. Apakšsistēmu 1.4. att. veido vienmērīgi sadalīto gadījumskaitļu ģenerators "**Random Source**", lietotāja funkcijas bloks "**Embedded Matlab Function**" ar 3 ieejām (2 ieejas sadalījuma parametriem un 3. ieeja – vienmērīgi sadalītajam gadījumskaitlim), kā arī divi konstantes bloki "**alpha**" un "**betta**". Šīm konstantēm var piešķirt patvaļīgās vērtības, modificējot diskrētās apakšsistēmas maskas failu "**des_paretosfunmask.m**", kas atrodas vienā direktorijā ar *Simulink* modeļa failu. Šī faila beigās ir jāpievieno α un β sadalījuma parametru aprēķināšanas procedūras saskaņā ar (1.5) un (1.7):

```
function cbDistType(block)
% --- Field data
Vals = get_param(block, 'maskvalues');
lambda = str2num(Vals{1});
hurst = str2num(Vals{2});
alpha = 3 - 2*hurst; % alpha
betta = (alpha - 1)/lambda/alpha; % beta
Vals{3} = num2str(alpha); Vals{4} = num2str(betta);
set_param(block, 'maskvalues',Vals);
return;
```

Pēc sadalījuma α un β parametru aprēķināšanas, šīs vērtības tiek padotas lietotāja funkcijas bloka attiecīgajās ieejās "**alpha**" un "**betta**", savukārt trešajā ieejā "**rnd**" tiek padots gadījumskaitlis intervālā $R \in [0, 1]$ no ģeneratora ar vienmērīgo sadalījumu. Lietotāja funkcija, saskaņā ar (3.7), ģenerē jauno gadījumskaitli *prt*, kura sadalījuma blīvuma funkcija atbilst (3.6) izteiksmei. Šīs lietotāja programmas teksts *Matlab* valodā ir dots zemāk:

```
function prt = ParetoRND(alpha,rnd,betta)
prt = betta/(rnd^(1/alpha));
```

Šis jaunais gadījumskaitlis ar Pareto sadalījuma likumu tiek izmantots tālāk kā laika intervāls starp pieprasījumiem, kurus ģenerē avots "**Time-Based Entity Generator**". Kā redzams 1.3. att., katra šī gadījumrakstura lieluma vērtība tiek saglabāta masīvā "**Time**" *Matlab* darba vidē turpmāko aprēķinu veikšanai. Jāatzīmē, ka analoģiski var izmantot jebkuru citu sadalījumu laika intervālu veidošanai, piemēram, Veibula sadalījumu [97].

1.5. $P/M/1/K$ modelēšanas rezultāti

Izmantojot iepriekš apskatīto $P/M/1/K$[7]. *Simulink* modeli, tika izpētīts rindas garums un tā vidējā vērtība dažādām sevlīdzīguma (Hersta) parametra H un noslodzes koeficienta ρ vērtībām. Kopējais pieprasījumu skaits rindošanas sistēmā visos eksperimentos sasniedza apmēram $2 \cdot 10^6$ vienības. Sevlīdzīguma parametra vērtība tika mainīta robežās no $H = 0,5$ līdz $H = 0,9$ ar soli 0,1 un pēdējo vērtību $H = 0,99$, kas atbilst procesam ar visaugstāko sevlīdzīgumu. Jāatzīmē, ka formāli procesam ar visaugstāko sevlīdzīguma pakāpi $H = 1,0$, taču šajā izveidotajā modelī tādu vērtību lietot nevar, jo tādā gadījumā ģenerēto intervālu gadījumprocesam būtu ne tikai bezgalīgā dispersija, bet arī bezgalīgā vidējā vērtība, jo saskaņā ar (1.5) pie $H = 1,0$ arī $\alpha = 1$, taču tādā gadījumā Pareto gadījumprocesam būs bezgalīgā vidējā vērtība un dispersija, ja $\alpha \leq 1$. Noslodzes koeficienta vērtība tika mainīta robežās no $\rho = 0,5$ līdz $\rho = 1,0$ ar soli 0,1. Noslodzes koeficienta vērtības tika sasniegtas ar pieprasījumu intensitātes λ izmaiņām no 50 ($\rho = 0,5$) līdz 100 ($\rho = 1,0$) pie nemainīgas apstrādes mezgla (servera) veiktspējas $\mu = 100$.

Pārbaudes nolūkos, tika izveidota analoģiskā simulācijas programma *GPSS World Student* [100] imitēšanai. Neskatoties uz to, ka šis rīks ir izveidots pietiekami sen, tajā ir iespēja veidot pieprasījumu secību, tā, lai laika intervāli starp šiem pieprasījumiem būtu sadalīti pēc Pareto likuma. Jāatzīmē, ka arī pats modelis ir veidojams ievērojami vienkāršāk, faktiski visu programmu var uzrakstīt 13 īsajās rindās, kuras ir dotas zemāk:

[7] Atzīmēsim, ka šajos eksperimentos buferatmiņas apjoma K vērtība tika iestādīta tāda, lai pieprasījuma atteikums nenotiktu simulācijas laikā. Konkrēti, visās simulācijās tika iestādīta vērtība $K_0 = 10^6$. Visaugstākais rindas garums simulāciju laikā sasniedza $K_{\max} \approx 8 \cdot 10^5$.

```
H EQU 0.99
Ta EQU 1/50
Ts EQU 1/100
pa EQU 3-H#2
pb EQU Ta#(pa-1)/pa
START 2000000
GENERATE (pareto(1,pb,pa))
QUEUE Rinda
SEIZE Serveris
DEPART Rinda
ADVANCE (exponential(2,0,Ts))
RELEASE Serveris
TERMINATE 1
```

Šajā programmā **Ta** un **Ts** ir attiecīgi vidējais pieprasījumu intervālu un apstrādes laiks, bet H – sevlīdzīguma (Hersta) parametrs. Savukārt **pa** un **pb** ir Pareto sadalījuma α un β parametri, kas tiek attiecīgi aprēķināti pēc (1.5) un (1.7). Kopējais pieprasījumu skaits tika iestādīts vienāds ar $2 \cdot 10^6$, t.i., tādās pašās robežās kā tas tika darīts *Simulink* modelī. Arī noslodzes koeficienta vērtības tika nodrošinātas attiecīgi mainot $T_\text{a} = 1/\lambda$ un $T_\text{s} = 1/\mu$, līdzīgi *Simulink* modeļa parametriem. Maksimālais rindas garums netika ierobežots, kas nozīmē, ka buferatmiņas apjoms bija uzskatāms par bezgalīgu – atšķirībā no *Simulink* modeļa, kur tas bija galīgs, taču pietiekami liels skaitlis, lai buferatmiņa nebūtu aizpildīta pilnībā – par ko liecināja pieprasījuma atteikuma varbūtības indikatora *Ploss* vērtība (skat. 1.3. att.), kas bija vienāda ar 0.

Modelēšanas rezultātā tika uzkrāti *Simulink* dati katrai noslodzes koeficienta ρ vērtībai no 0,5 līdz 1,0 visām H parametra vērtībām. Datu iegūšanai tika izveidota *Matlab* programma, kuras kods ir 1.pielikums.pielikumā. Apstrādes rezultātā tika veidots tabulu masīvs ar šādām vērtībām vienā rindā (kurai atbilst noteiktā H vērtība, piemēram, pirmajai rindai $H = 0,5$, otrajai $H = 0,6$ utt.):

- vidējais rindas garums K_0;
- maksimālais rindas garums K_max;
- rekomendētais buferatmiņas apjoms K_buf, kas aprēķināts saskaņā ar (1.9) no [90]:

$$K_\text{buf} = \frac{\rho^{\frac{1}{2(1-H)}}}{(1-\rho)^{\frac{H}{(1-H)}}}. \tag{1.9}$$

Šīs vērtības ir apkopotas 1.1. – 1.6. tabulās kopā ar *GPSS World* modelēšanas rezultātiem: vidējo rindas garumu un maksimālo rindas garumu. Rindas garuma izmaiņu dinamika visiem eksperimentiem ir atainota 1.5. – 1.10. att. Ir aprēķināts vidējais rindas garums pēc pilniem rindas garuma dinamikas datiem masīvos, kas tika saglabāti failos, izmantojot *Matlab* vidi.

1.1. tabula. Rindas garuma K salīdzinājums, noslodzes koeficients $\rho = 0,5$.

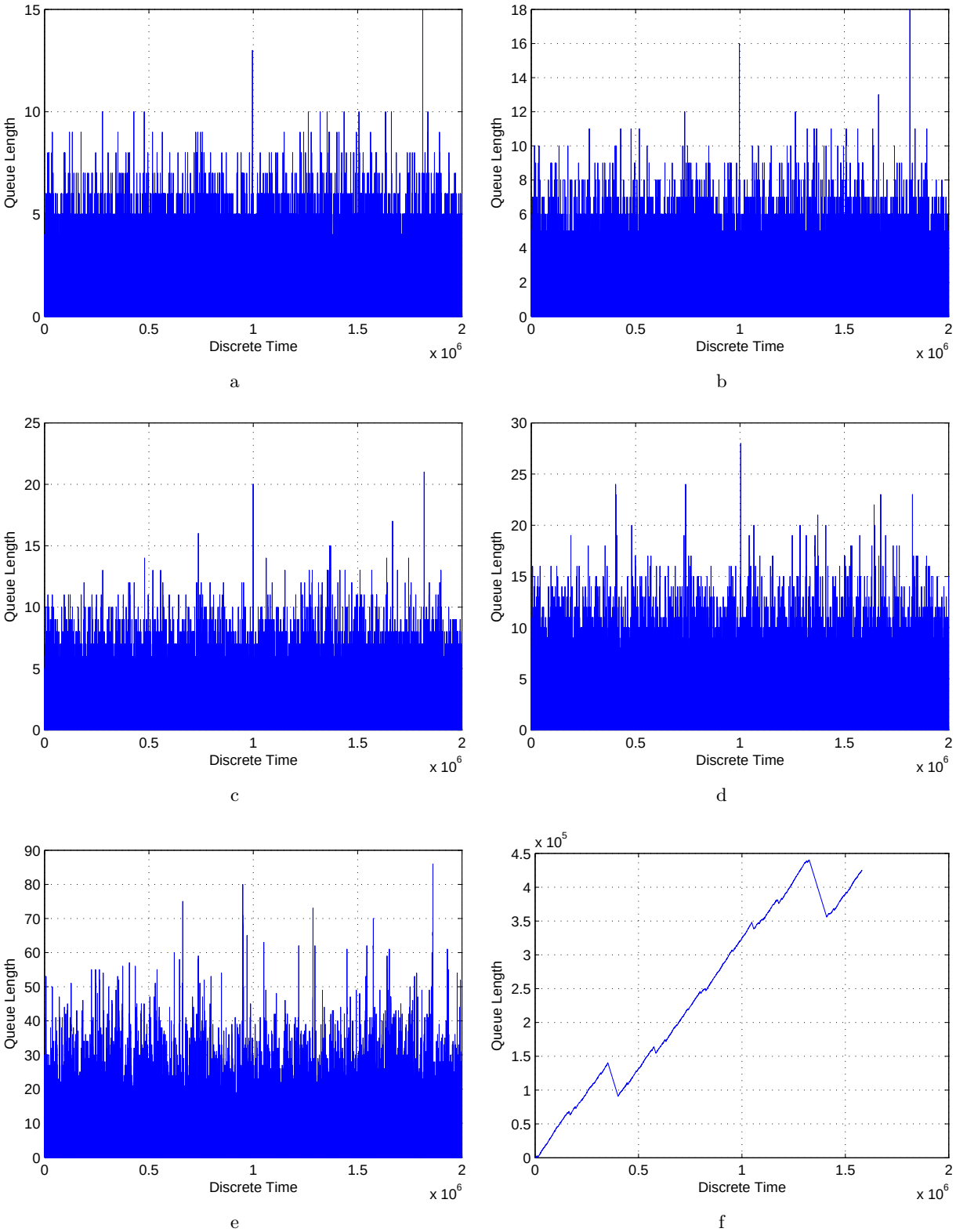
	Simulink	***GPSS***	**Vid. vērt.**	**Aprēķins**
$\boldsymbol{H = 0,5}$	0,29 (15)	0,29 (12)	0,39	1,00
$\boldsymbol{H = 0,6}$	0,35 (18)	0,35 (15)	0,5	1,19
$\boldsymbol{H = 0,7}$	0,49 (21)	0,49 (17)	0,73	1,59
$\boldsymbol{H = 0,8}$	0,88 (28)	0,87 (25)	1,41	2,83
$\boldsymbol{H = 0,9}$	4,02 (86)	3,94 (81)	7,08	16,00
$\boldsymbol{H = 0,99}$	$2,46 \cdot 10^5$ ($4,41 \cdot 10^5$)	$9,91 \cdot 10^5$ ($2,004 \cdot 10^6$)	$2,38 \cdot 10^5$	$5,63 \cdot 10^{14}$

Noslodzes koeficientam $\rho = 0,5$, kā var redzēt no 1.1. tabulas, rindas vidējais garums nesasniedz lielās vērtības, izņemot gadījumu $H = 0,99$, kad trafikam stipri izpaužas sevlīdzīguma īpašības. No 1.1. tabulas datiem var skaidri redzēt, ka visstraujākais vidējā rindas garuma pieaugums ir novērojams posmā $0,8 < H < 0,99$. Jāatzīmē, ka tieši šajā intervālā ir Interneta trafika sevlīdzīguma koeficients [86, 90]. Gan *Simulink*, gan *GPSS* šeit parāda ļoti tuvus rezultātus – vidējais rindas garums sakrīt gandrīz pilnīgi, maksimālā rindas garumā (iekavās) novērojamas atšķirības, kas veidojās simulācijas nelielo atšķirību dēļ un var būt uzskatāmas par pieņemamām. Izņēmums ir gadījums pie $H = 0,99$, kur *GPSS* simulācijā sanāca ievērojami augstākas vērtības. Maksimālais rindas garums pārsniedz uzstādīto kopējo pieprasījumu skaitu (pie noslodzes $\rho = 0,5$ – neievērojami, bet palielinot noslodzes koeficientu, šis lielums paliek arvien ievērojamāks.). Jāatzīmē, ka *Simulink* modelī pie tādām augstām Hersta parametra vērtībām arī veidojās grūtības, kā to var redzēt no 1.5f. att. – kopējais diskrētā laika momentu skaits nav sasniedzis $2 \cdot 10^6$ vienības. Savukārt buferatmiņas nepieciešamais apjoms K_{buf}, kas tika noteikts saskaņā ar (1.9), visos gadījumos ir ievērojami mazāks par iespējamajiem rindas garuma lēcienveidīgajiem pieaugumiem. To pašu var redzēt arī 1.5a. – 1.5f. att. Aprēķinātais pēc pilnās *Simulink* realizācijas vidējais garums atšķiras no simulācijas datiem pat pie zemās sevlīdzīguma pakāpes.

No 1.5a. – 1.5f. att. grafikiem ir skaidra intuitīvi saprotama tendence – rindas garums K pieaug līdz ar sevlīdzīguma koeficienta H pieaugumu, un šis pieaugums nav lineārs. Var konstatēt arī citas īpatnības grafikos, piemēram, 1.5a. – 1.5d. att. var saskatīt, ka rindas garuma dinamikā veidojās noteikti diskrēti līmeņi – un jo zemāka ir sevlīdzīguma parametra vērtība H, jo labāk tos var saskatīt. Otrkārt, no tiem pašiem grafikiem kopā ar 1.5e. att., var redzēt, ka lēcienveidīgie buferatmiņas apjoma pieaugumi paliek arvien regulārāki (sevlīdzīgāki). Šī situācija zināmā mērā var būt uzskatīta par "katastrofu", negaidīto notikumu. [91] autors tādus notikumus sauc par "**melnajiem gulbjiem**"[8]. Īpaši jāatzīmē 1.5f. att. rezultāti – apskatot šo grafiku, var bez šaubām konstatēt, ka rindas garums ir sevlīdzīgais process pie $H = 0,99$, kas ir gandrīz maksimālais sevlīdzīguma mērs. Grafika forma atgādina kalnus dabā un ir fraktālis gandrīz tīrajā veidā[9].

[8] Šeit būtu precīzāk pateikt "**pelēkajiem gulbjiem**", jo tie var būt noteiktajā veidā kontrolēti, prognozēti.

[9] Šeit tas nav parādīts, bet turpinot palielināt 1.5f. att. parādīto grafiku *Matlab* vidē, var redzēt atkārtojošo līknes raksturu



1.5. att. Rindas garuma dinamika simulācijas laikā pie noslodzes koeficienta $\rho = \lambda/\mu = 0,5$: a) $H = 0,5$, b) $H = 0,6$, c) $H = 0,7$, d) $H = 0,8$, e) $H = 0,9$, f) $H = 0,99$.

1.2. tabula. Rindas garuma K salīdzinājums, noslodzes koeficients $\rho = 0,6$.

	Simulink	***GPSS***	**Vid. vērt.**	**Aprēķins**
$\boldsymbol{H = 0,5}$	0,61 (21)	0,61 (18)	0,77	1,50
$\boldsymbol{H = 0,6}$	0,76 (22)	0,76 (22)	0,99	2,09
$\boldsymbol{H = 0,7}$	1,08 (28)	1,07 (25)	1,47	3,62
$\boldsymbol{H = 0,8}$	2,07 (39)	2,03 (44)	3,03	10,89
$\boldsymbol{H = 0,9}$	13,53 (181)	12,43 (172)	20,70	296,63
$\boldsymbol{H = 0,99}$	$2,94 \cdot 10^5$ ($5,24 \cdot 10^5$)	$1,40 \cdot 10^6$ ($2,90 \cdot 10^6$)	$2,83 \cdot 10^5$	$2,01 \cdot 10^{28}$

Līdzīgi rezultāti ir novērojami 1.2. tabulā un 1.6a. – 1.6f. att. grafikos noslodzes koeficientam $\rho = 0,6$. Jāatzīmē, ka sevlīdzīguma parametram vērtība $H = 0,99$ *GPSS* maksimālais rindas garums sastāda gandrīz $K_{\max} = 3 \cdot 10^6$, kas ir 1,5 reizēs vairāk par kopējo pieprasījumu skaitu $M = 2 \cdot 10^6$ – tas nerāda šaubu par to, ka pie tik augstajām H parametra vērtībām modelēšana *GPSS* dod nekorektu rezultātu, taču to pašu var pateikt arī par *Simulink* modeli[10].

Interesants rezultāts novērojams buferatmiņas apjoma novērtējumam – tas nav pietiekams gadījumos $0,5 \leq H \leq 0,8$ (un gadījumā ar $H = 0,99$, kad rindas garums neierobežoti aug), savukārt vērtībai $H = 0,9$ ar aprēķināto buferatmiņas apjomu varētu pietikt maksimālās rindas glabāšanai, gan pēc *Simulink*, gan pēc *GPSS* datiem. Protams, šeit runa ir tikai par $2 \cdot 10^6$ pieprasījumiem – un šie rezultāti nedod nekādu garantiju, ka nenotiks "melnais gulbis" [91]. Tieši otrādi – tādu notikumu var sagaidīt, jo pieprasītajam resursam (diskrētais laiks, šajā gadījumā) ir Pareto sadalījums, kuram ir raksturīga t.s. "garā aste". Tāpēc, buferatmiņas ierobežojums joprojām noved pie pieprasījumu atteikumiem, taču šī konkrētā pakešu zudumu varbūtības P_{loss} vērtība ir atsevišķo pētījumu jautājums.

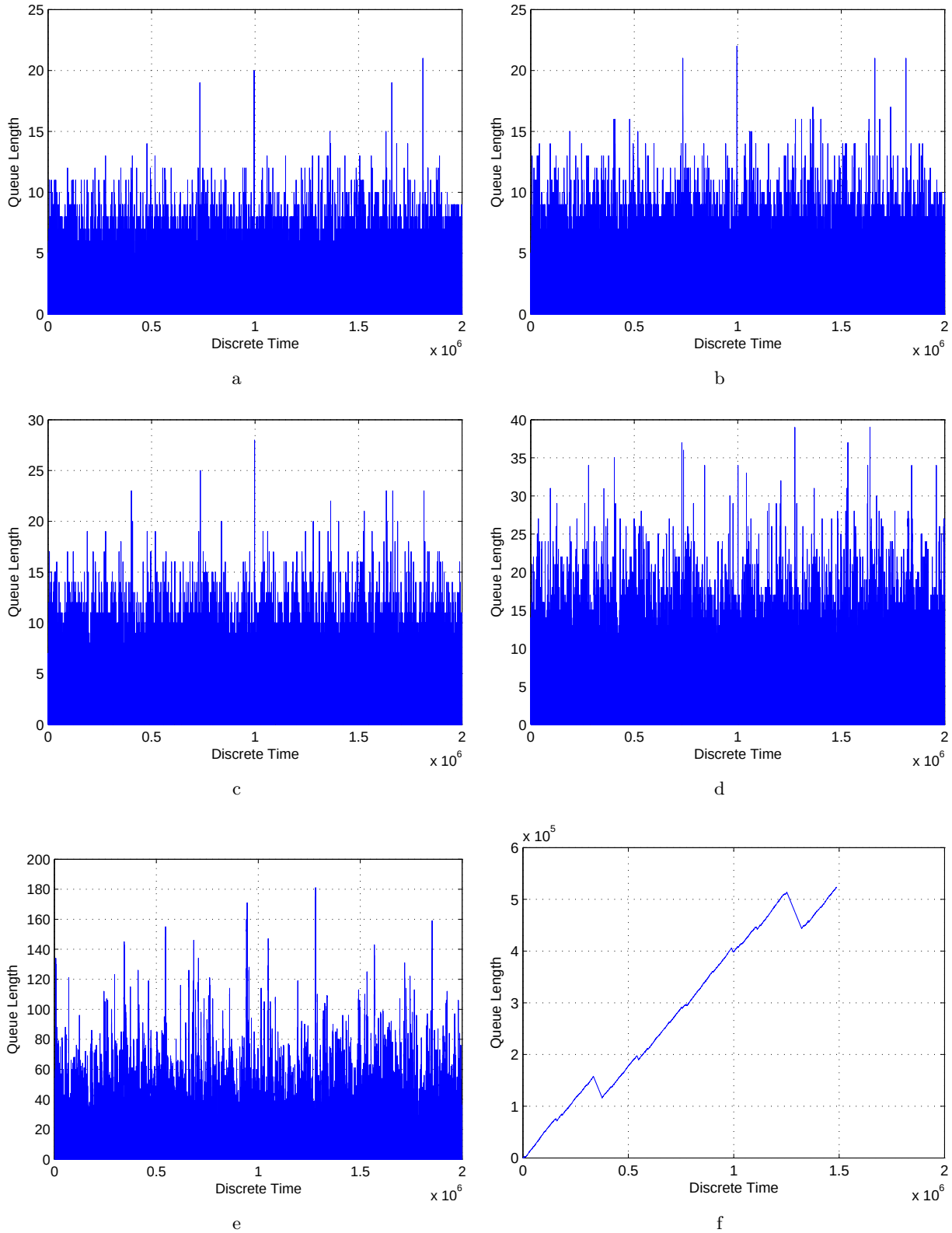
Apskatot 1.6a. – 1.6f. att. grafikus, var secināt, ka ne tikai rindas garuma vērtības pieaug, bet arī manāmas atšķirības dinamikas raksturā. Kopumā, veidojās lielākas neregularitātes, turklāt tās ir sastopamas jau pie zemākajām sevlīdzīguma H parametra vērtībām, salīdzinot ar gadījumu, kad noslodzes koeficients bija $\rho = 0,5$. Salīdzinot 1.6f. att. ar 1.5f. att., var redzēt, ka process palika "gludāks" un atkārtošanas posmu garumi arī ir tuvāki viens otram.

1.3. tabula. Rindas garuma K salīdzinājums, noslodzes koeficients $\rho = 0,7$.

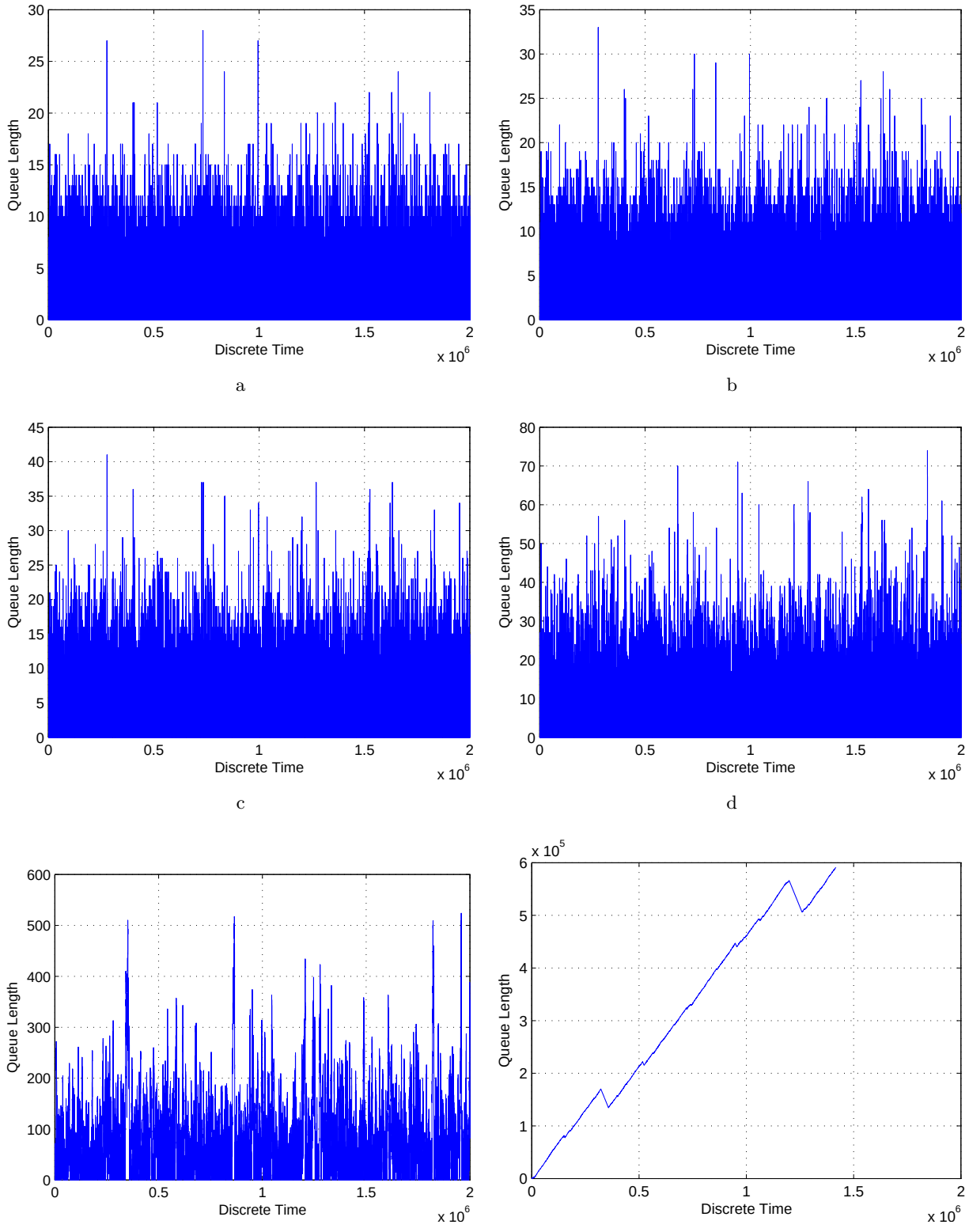
	Simulink	***GPSS***	**Vid. vērt.**	**Aprēķins**
$\boldsymbol{H = 0,5}$	1,29 (28)	1,27 (25)	1,52	2,33
$\boldsymbol{H = 0,6}$	1,63 (33)	1,59 (33)	1,97	3,90
$\boldsymbol{H = 0,7}$	2,41 (41)	2,33 (43)	305	9,16
$\boldsymbol{H = 0,8}$	5,17 (74)	4,85 (74)	6,88	50,61
$\boldsymbol{H = 0,9}$	58,71 (524)	52,87 (532)	78,54	8538,80
$\boldsymbol{H = 0,99}$	$3,27 \cdot 10^5$ ($5,91 \cdot 10^5$)	$1,81 \cdot 10^6$ ($3,6 \cdot 10^6$)	$2,83 \cdot 10^5$	$1,05 \cdot 10^{44}$

Palielinot noslodzes koeficientu līdz $\rho = 0,7$, var nonākt pie rezultātiem, kas apkopoti 1.3. tabulā un 1.7a. – 1.7f. att. Rindas garuma vidējās un maksimālās vērtības pieaug – turklāt pieaugums paliek straujāks līdz ar H vērtības pieaugumu: salīdzinot gadījumu $H = 0,5$, kur $K_{\max}$ vērtība pieauga no 18 līdz 25 ar gadījumu $H = 0,7$, kur $K_{\max}$ vērtība mainījās no 25 līdz 43. Arī vidējais rindas garums K_0 pieaug straujāk, salīdzinot pieaugumu no 2,03 līdz 4,85 ($H = 0,8$) ar pieaugumu no 12,43 līdz 52,87 ($H = 0,9$). Šie pieaugumi ir straujāki, salīdzinot ar pieaugumiem pie noslodzes koeficienta izmaiņām no $\rho = 0,5$ līdz $\rho = 0,6$. Tas ļauj secināt, ka turpmākajam noslodzes koeficienta pieaugumam līdz $\rho = 0,8$ pieaugums paliks vēl straujāks.

[10] Patiesībā, nevar apgalvot, ka modelis strādā pilnīgi nekorekti – ir novērojamas tikai grūtības ar pieprasījumu ģenerēšanu, veidojās garās pauzes starp pieprasījumiem, kas varētu atbilst procesam ar augsto sevlīdzīgumu.



1.6. att. Rindas garuma dinamika simulācijas laikā pie noslodzes koeficienta $\rho = \lambda/\mu = 0,6$: a) $H = 0,5$, b) $H = 0,6$, c) $H = 0,7$, d) $H = 0,8$, e) $H = 0,9$, f) $H = 0,99$.



1.7. att. Rindas garuma dinamika simulācijas laikā pie noslodzes koeficienta $\rho = \lambda/\mu = 0,7$: a) $H = 0,5$, b) $H = 0,6$, c) $H = 0,7$, d) $H = 0,8$, e) $H = 0,9$, f) $H = 0,99$.

Analizējot 1.7a. – 1.7f. att. grafikus, var konstatēt, ka rindas garums vērtībām $H = 0,5$ un $H = 0,6$ uzvedas gandrīz vienādi – pat "uzliesmojumu" maksimumi ir ļoti tuvi. Neregularitātes pieaug pēc amplitūdas, savukārt 1.7e. att. var skaidri redzēt, ka šīs neregularitātes ievērojami pārsniedz vidējo līmeni līdz kuram grafiks ir "nokrāsots", turklāt parādās "baltie plankumi" ar regulāriem intervāliem, kuri norāda uz augsto sevlīdzīgumu. 1.7f. att. atšķirības no 1.6f. att. un 1.5f. att. ir minimālas, salīdzinot ar atšķirībām 1.1. – 1.3. tabulās, kur $H = 0,99$ aprēķinātais buferatmiņas apjoms pieaug eksponenciāli: $K_{\text{buf}} \approx 10^{14}$ pie $\rho = 0,5$, $K_{\text{buf}} \approx 10^{28}$ pie $\rho = 0,6$ un $K_{\text{buf}} \approx 10^{44}$ pie $\rho = 0,7$. Tas norāda uz pieprasījumu kopējā skaita $M = 2 \cdot 10^6$ nepietiekamību, pētot šādu augsti sevlīdzīgo procesu.

Visas augstāk aprakstītas tendences saglabājās, palielinot noslodzes koeficientu līdz $\rho = 0,8$ un turpmāk līdz $\rho = 0,9$, bet attiecīgie rezultāti ir pieejami 1.4. tabulā un 1.5. tabulā, ka arī 1.8. att. un 1.9. att. grafikos.

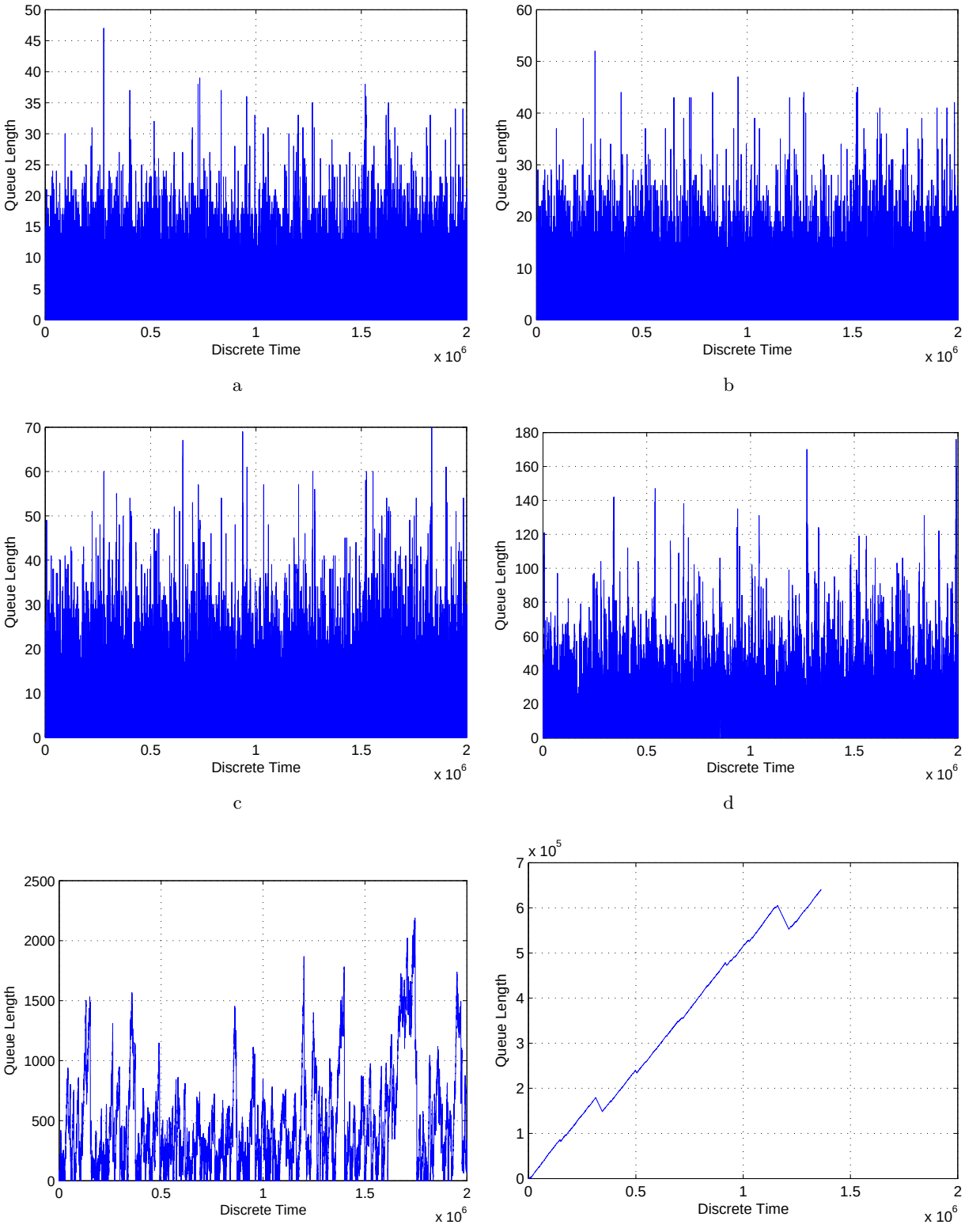
1.4. tabula. Rindas garuma K salīdzinājums, noslodzes koeficients $\rho = 0,8$.

	Simulink	***GPSS***	**Vid. vērt.**	**Aprēķins**
$\boldsymbol{H = 0,5}$	2,92 (47)	2,77 (44)	3,25	4,00
$\boldsymbol{H = 0,6}$	3,82 (52)	3,57 (53)	4,36	8,46
$\boldsymbol{H = 0,7}$	6,02 (70)	5,49 (69)	7,07	29,47
$\boldsymbol{H = 0,8}$	16,33 (176)	13,78 (163)	19,78	357,77
$\boldsymbol{H = 0,9}$	388,10 (2189)	418,84 (2621)	456,64	$6,40 \cdot 10^5$
$\boldsymbol{H = 0,99}$	$3,52 \cdot 10^5$ ($6,41 \cdot 10^5$)	$2,22 \cdot 10^6$ ($4,56 \cdot 10^6$)	$3,38 \cdot 10^5$	$2,25 \cdot 10^{64}$

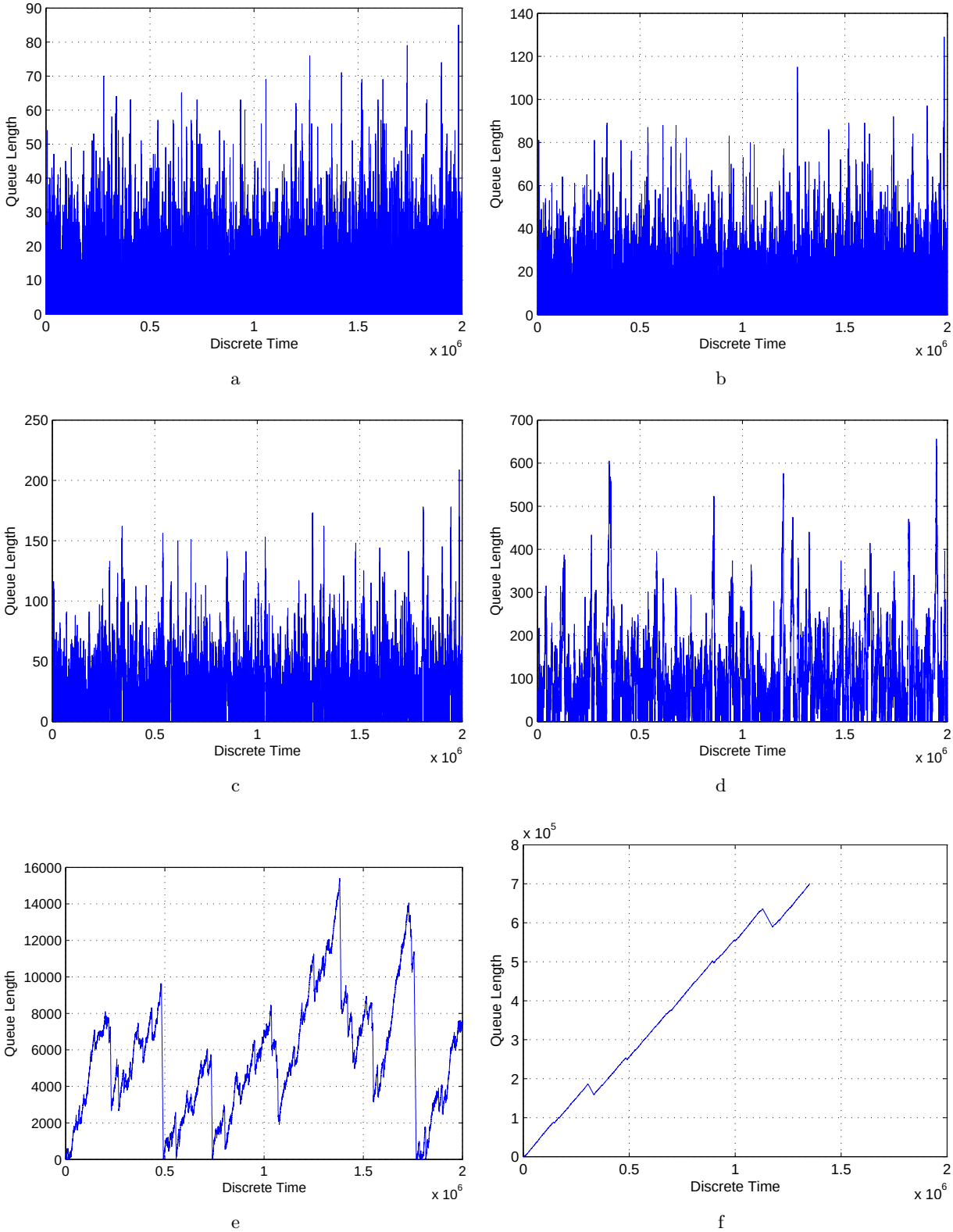
1.5. tabula. Rindas garuma K salīdzinājums, noslodzes koeficients $\rho = 0,9$.

	Simulink	***GPSS***	**Vid. vērt.**	**Aprēķins**
$\boldsymbol{H = 0,5}$	8,73 (85)	7,30 (68)	9,25	9,00
$\boldsymbol{H = 0,6}$	12,41 (129)	9,98 (88)	13,33	27,72
$\boldsymbol{H = 0,7}$	23.,69 (209)	17,99 (205)	25,82	180,75
$\boldsymbol{H = 0,8}$	93,23 (656)	78,16 (572)	102,50	7684,30
$\boldsymbol{H = 0,9}$	5232,00 (15394)	7536,09 (20909)	5548,50	$5,90 \cdot 10^8$
$\boldsymbol{H = 0,99}$	$3,76 \cdot 10^5$ ($7,00 \cdot 10^5$)	$2,64 \cdot 10^6$ ($5,49 \cdot 10^6$)	$3,63 \cdot 10^5$	$5,15 \cdot 10^{96}$

Šeit ir jāatzīmē, ka Hersta parametra vērtībai $H = 0,9$ un noslodzes koeficienta vērtībai $\rho = 0,9$ rindas garums var būt milzīgs. Atšķirība starp *Simulink* un *GPSS* rezultātiem maksimālajā rindas garumā šeit sastāda lielumu ap 5000 pieprasījumiem. 1.9e. att. grafikā novērojamā dinamika pēc sava rakstura jau sāk tuvoties dinamikai no 1.9f. att. grafika, taču rindas garums vēl neuzsāk neierobežoti pieaugt – grafikā skaidri redzams, ka aiz rindas pieauguma intervāliem seko gandrīz lēcienveidīgā visu rindā esošo pieprasījumu apstrāde, bet acīmredzams, ka apstrādes mezgls tādā režīmā nevarēs strādāt neierobežoti ilgi, ja noslodze netiks samazināta.



1.8. att. Rindas garuma dinamika simulācijas laikā pie noslodzes koeficienta $\rho = \lambda/\mu = 0,8$: a) $H = 0,5$, b) $H = 0,6$, c) $H = 0,7$, d) $H = 0,8$, e) $H = 0,9$, f) $H = 0,99$.



1.9. att. Rindas garuma dinamika simulācijas laikā pie noslodzes koeficienta $\rho = \lambda/\mu = 0,9$: a) $H = 0,5$, b) $H = 0,6$, c) $H = 0,7$, d) $H = 0,8$, e) $H = 0,9$, f) $H = 0,99$.

1.6. tabula. Rindas garuma K salīdzinājums pie noslodzes koeficienta $\rho = 1,0$.

	Simulink	***GPSS***	**Vid. vērt.**	**Aprēķins**
$\boldsymbol{H = 0,5}$	529,30 (1930)	1083,25 (2235)	529,63	∞
$\boldsymbol{H = 0,6}$	846,07 (2806)	1769,00 (3956)	846,95	∞
$\boldsymbol{H = 0,7}$	1873,20 (5807)	3999,59 (9352)	1876,10	∞
$\boldsymbol{H = 0,8}$	6270,60 (16757)	13754,10 (31295)	6277,30	∞
$\boldsymbol{H = 0,9}$	37015,00 (80154)	82519,51 (176287)	36879,00	∞
$\boldsymbol{H = 0,99}$	$4,18 \cdot 10^5$ $(8,05 \cdot 10^5)$	$3,07 \cdot 10^6$ $(6,35 \cdot 10^6)$	$4,10 \cdot 10^5$	∞

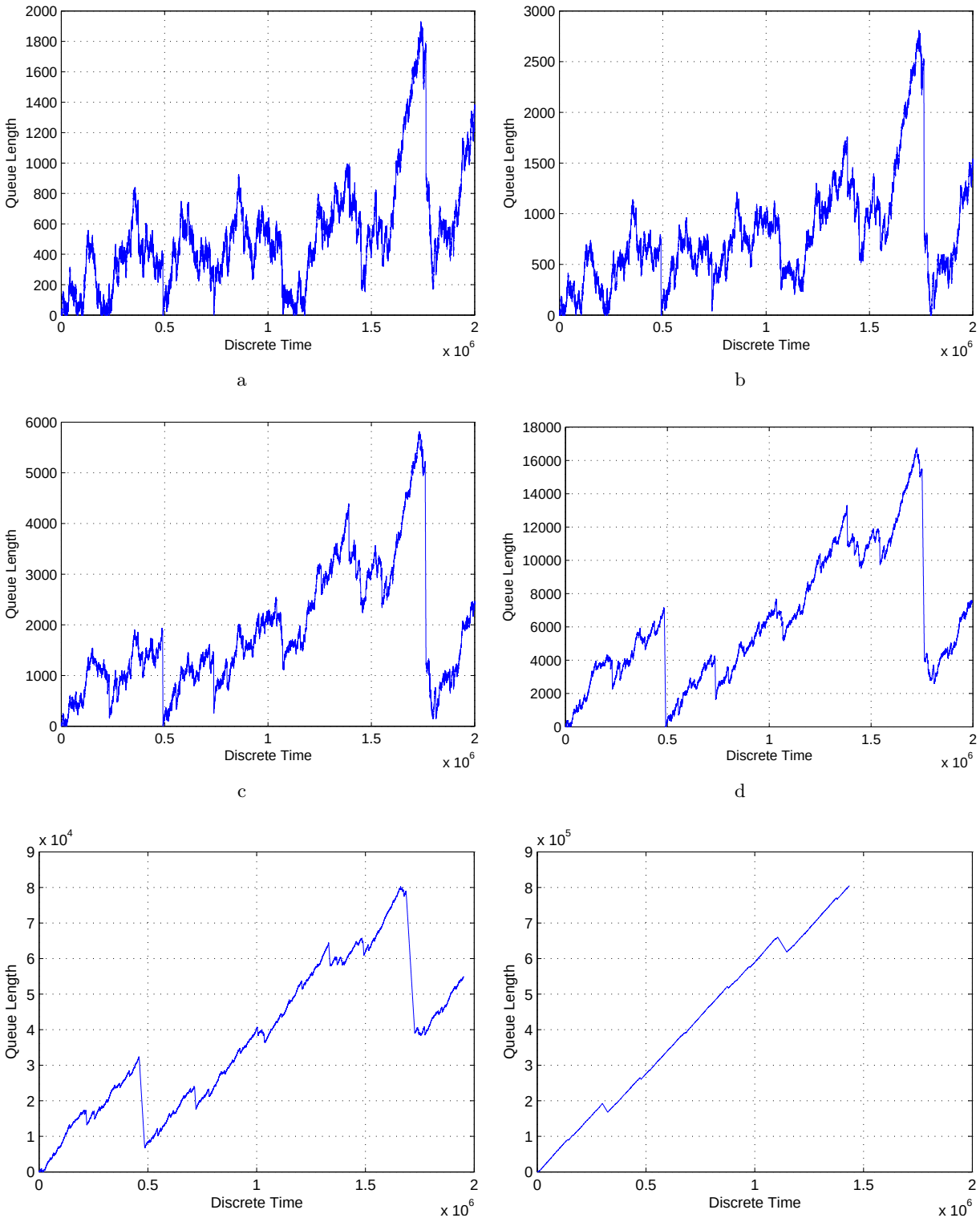
Pēdējā eksperimentu sērijā tika apskatīts gadījums ar 100 % apstrādes mezgla noslodzi, t.i., $\lambda = \mu$. Šajā gadījumā apstrādes mezgls nevar apstrādāt visus pieprasījumus un rindas garums neierobežoti pieaug. Ja buferatmiņas apjoms ir galīgs lielums, noteiktajā momentā veidojās pieprasījumu atteikumi. Šie paši atteikumi dod iespēju serverim apstrādāt rindā esošus pieprasījumus, tādā veidā atbrīvojot buferatmiņu jaunajiem pieprasījumiem. Praksē noslodzes koeficients nevar ilgstoši pieņemt vērtības virs $\rho = 0,8$ [90], taču īslaicīgas pārslodzes varētu būt pieņemamas atsevišķajos gadījumos, kaut arī tās rādīs neērtības sistēmas lietotājiem. Ja noslodzes koeficients stabili tuvojas savam maksimumam $\rho = 1,0$ – jāpalielina apstrādes mezgla veiktspēju vai tādu mezglu skaitu.

Simulāciju rezultāti ir apkopoti 1.6. tabulā un 1.10a. – 1.10f. att. grafikos. Nepieciešamais buferatmiņas apjoms pēc (1.9) tiecās uz bezgalību, aprēķinot robežu:

$$\lim_{\rho \to 1,0} \left(\frac{\rho^{\frac{1}{2(1-H)}}}{(1-\rho)^{\frac{H}{(1-H)}}} \right) \to \infty. \tag{1.10}$$

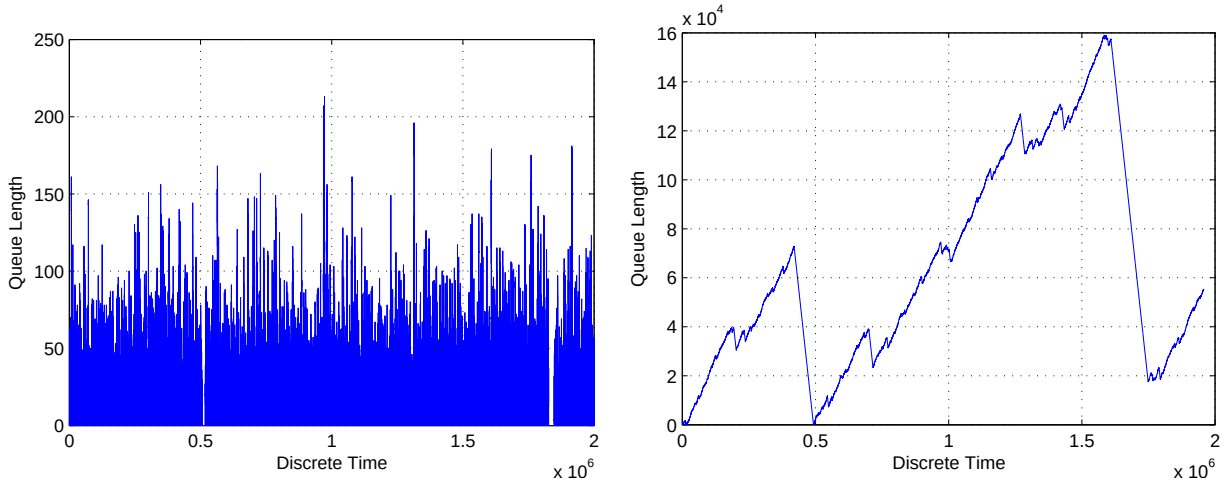
Visām H parametra vērtībām rindas garums neierobežoti pieaug. Atkarībā no H vērtības, pieaugumam būs dažāds raksturs, kā to var redzēt no 1.10a. – 1.10f. att. grafikiem. Vērtībām $H = 0,5$, $H = 0,6$ un, noteiktajā mērā, $H = 0,7$ grafikos ir grūti saskatīt sevlīdzīguma īpašības. Tas ir likumsakarīgi, jo arī šie Hersta parametri norāda uz ne pārāk augsto sevlīdzīguma pakāpi. Savukārt sākot ar 1.10d. att. un turpinot ar 1.10e. att., var redzēt ka process paliek arvien vairāk sevlīdzīgs.

1.10f. att. ir parādīts rindas garuma izmaiņu grafiks sevlīdzīguma parametra vērtībai $H = 0,99$. Kā var redzēt, šis grafiks maz atšķiras no 1.5f. – 1.9f. att. parādītajiem grafikiem. Visos sešos gadījumos $(0,5 \leq \rho \leq 1,0)$ rindas garums neierobežoti aug, turklāt maksimālā vērtība simulācijas laikā būtiski nemainās pieaugot noslodzes koeficientam (ja $\rho = 0,5$ tā sastāda apmēram $4,50 \cdot 10^5$, bet $\rho = 1,0$ apmēram $8,00 \cdot 10^5$).



1.10. att. Rindas garuma dinamika simulācijas laikā pie noslodzes koeficienta $\rho = \lambda/\mu = 1,0$: a) $H = 0,5$, b) $H = 0,6$, c) $H = 0,7$, d) $H = 0,8$, e) $H = 0,9$, f) $H = 0,99$.

Tika veiktas papildus simulācijas ar noslodzes koeficienta ρ vērtībām $0,1$ un $0,3$. Rezultāti ir parādīti 1.11. att. Kā var redzēt no 1.11b. att. grafika, rindas garumā joprojām var



a b

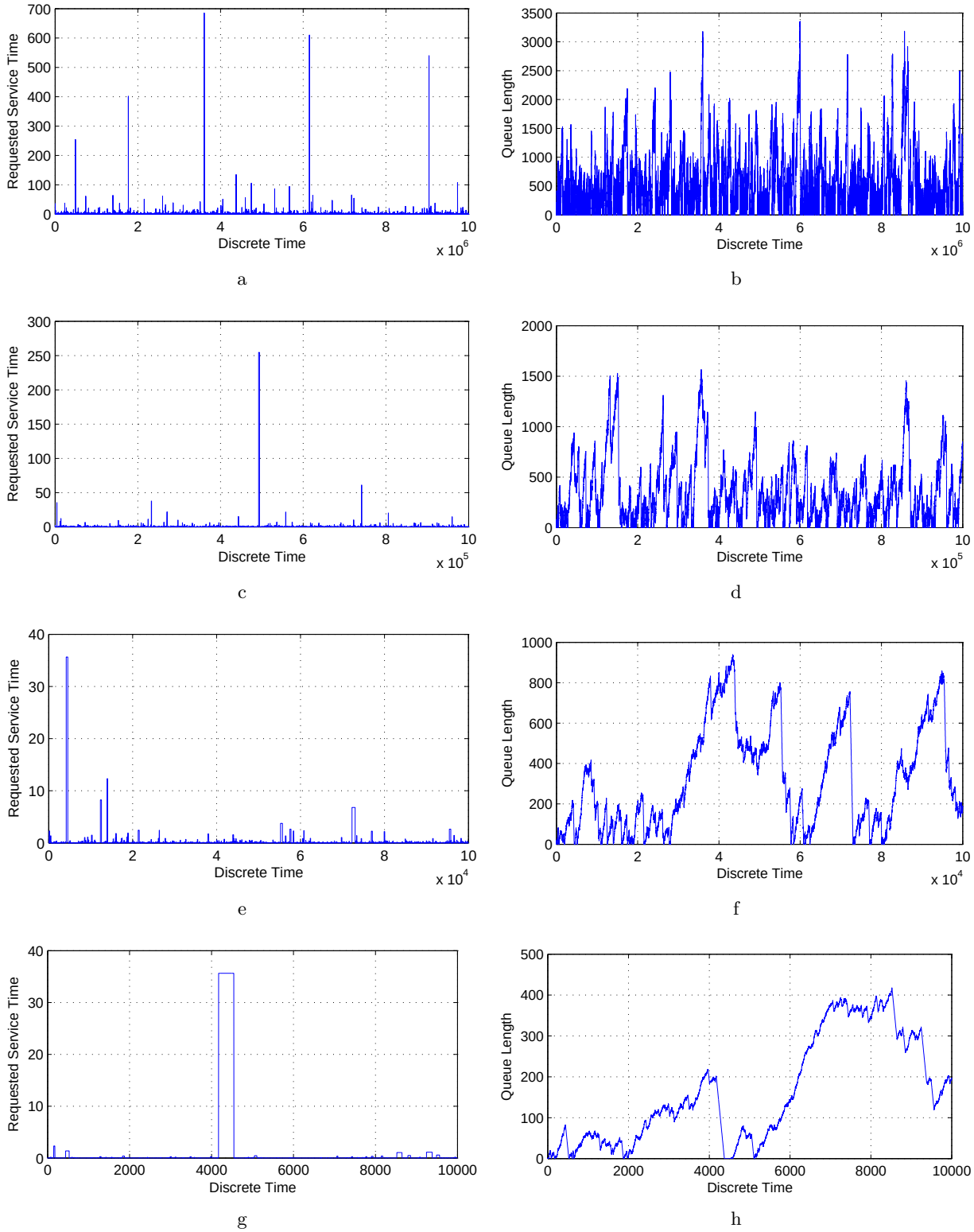
1.11. att. Rindas garuma dinamika simulācijas laikā pie sevlīdzīguma koeficienta $H = 0,99$: a) $\rho = 0,1$, b) $\rho = 0,3$.

saskatīt sevlīdzīgo procesu īpatnības, bet maksimālais rindas garums simulācijas ierobežotajā laikā sasniedz vērtību $1,60 \cdot 10^5$, t.i., vērtības kārta nav samazinājusies, salīdzinot ar 1.5f. – 1.10f. att. parādīto grafiku maksimālo vērtību kārtām. Savukārt 1.11a. att. grafikā var redzēt, ka situācija mainījās – tagad maksimālā vērtība nesasniedz 250 buferatmiņas vienības un tikai vienu reizi pārsniedz 200 vienības. Tomēr arī šāda situācija nevar būt uzskatāma par "labu", jo pārējos eksperimentos ir nepieciešams noslodzes koeficients $\rho = 0,9$ un $H = 0,7$, lai sasniegtu līdzīgo efektu. Šeit ir jāievēro, ka noslodzes koeficienta ρ vērtības $0,1$ un $0,3$ ir ļoti zemas, kas savukārt parāda, cik lielajā mērā sevlīdzīguma parametra pieaugums var veicināt pārslodzes (situācijas, kad apstrādes mezgls nepaspēj apstrādāt pieprasījumus un veidojās garās rindas.)

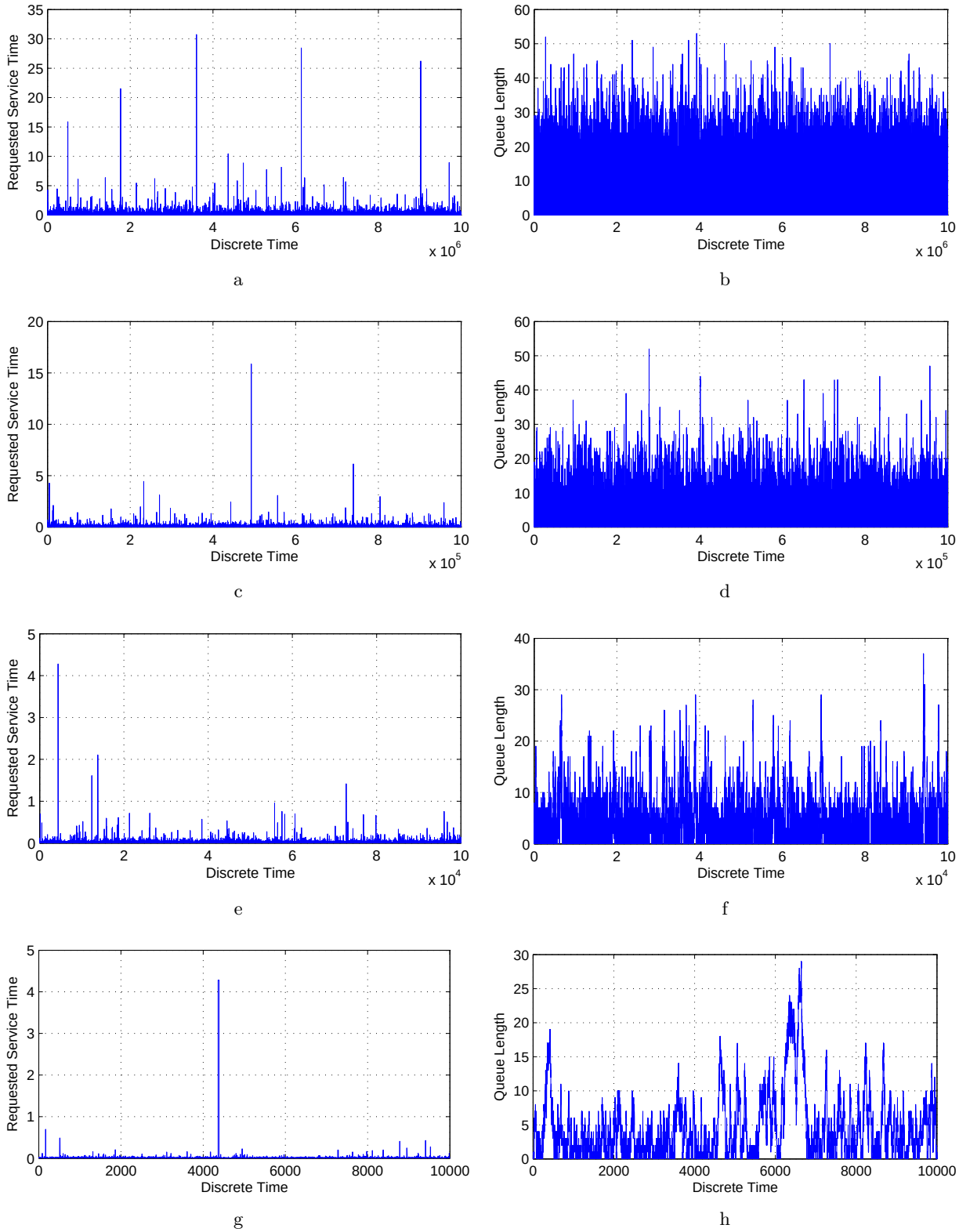
Analizējot 1.5. – 1.10. att. grafikus, varēja vairākas reizes konstatēt, ka rindas garumam augstajām Hersta parametra H vērtībām piemīt sevlīdzīgums. Šī jautājuma turpmākajai izpētei tika veiktas papildus simulācijas ar ilgāku simulācijas laiku – ja iepriekšējās simulācijās tika veidoti $2 \cdot 10^6$ pieprasījumi, tad šajās divās simulācijās to skaits bija palielināts 5 reizēs, t.i., līdz lielumam 10^7. Simulāciju laikā tika saglabāts gan tekošais pieprasītais laika intervāls no apstrādes mezgla, gan tekošais rindas garums. Šie lielumi tika attēloti grafiski dažādos laika mērogos un ir parādīti 1.12. att. un 1.13. att. attiecīgajos grafikos. Šajos grafikos kreisajā kolonnā ir pieprasītais apstrādes laiks, bet labajā – rindas garums. Katrā nākamajā rindā mērogs tiek samazināts 10 reizēs.

Salīdzinot 1.12. att. un 1.13. att. grafikus, var redzēt, ka šīm divām grafiku grupām ir atšķirīgais raksturs, un īpaši labi to var redzēt rindas garuma gadījumā. Pieprasīto apstrādes intervālu sevlīdzīgums tīklos ir jau sen pierādīts [46], šajā gadījumā interesanti rezultāti ir rindas garuma grafikos. 1.12b.,1.12d.,1.12f.,1.12h. att. samazinot mērogu, rindas garuma raksturs ir līdzīgs, bet maksimālais rindas garums samazinās eksponenciāli. Konkrēti – samazinās 2 reizēs mēroga izmaiņām, izņemot pāreju no 1.12d. att. pie 1.12f. att.

Savukārt 1.13b.,1.13d.,1.13f.,1.13h. att. grafikos saskatīt līdzīgumu ir ievērojami grūtāk. Kaut arī šajā gadījumā Hersta parametra vērtība $H = 0,6$ norāda uz noteikto sevlīdzīguma mēru, šis mērs ir pārāk mazs, lai to varētu saskatīt vizuāli grafikos. Tomēr, rezultāti saskaņojas ar līdzīgajiem rezultātiem [90], pēc kuriem 1.12b.,1.12d.,1.12f.,1.12h. att. grafiki vairāk atbilst sevlīdzīgajam procesam, nekā 1.13b.,1.13d.,1.13f.,1.13h. att. grafiki, kuriem ir vairāk kopējā ar t.s. **"Brouna procesu"** (šajā gadījumā runa ir par $H = 0,5$). Jāatzīmē, ka šis sevlīdzīgums netiek pierādīts matemātiski, uzmanība tiek pievērsta tikai tā vizuālajai izpausmei.



1.12. att. Rindas garuma un pieprasītā apstrādes laika dinamika simulācijas laikā pie noslodzes koeficienta $\rho = \lambda/\mu = 0,8$ un sevlīdzīguma parametra $H = 0,9$: a,b) pirmās 10^7 nolases, c,d) pirmās 10^6 nolases, e,f) pirmās 10^5 nolases un g,h) pirmās 10^4 nolases.



1.13. att. Rindas garuma un pieprasītā apstrādes laika dinamika simulācijas laikā pie noslodzes koeficienta $\rho = \lambda/\mu = 0,8$ un sevlīdzīguma parametra $H = 0,6$: a,b) pirmās 10^7 nolases, c,d) pirmās 10^6 nolases, e,f) pirmās 10^5 nolases un g,h) pirmās 10^4 nolases.

Visu veikto eksperimentu rezultātu apkopojums parāda kopējo tendenci: rindas garums pieaug līdz ar sevlīdzīguma parametra H un/vai noslodzes koeficienta ρ pieaugumu. Ja pieaug abi šie sistēmas parametri, tad arī rindas garuma pieaugums attiecīgi novērojams lielākajā mērā, nekā gadījumā, kad pieaug tikai viens parametrs. Taču pēc dotajiem rezultātiem ir sarežģīti pateikt, kurš no šiem diviem parametriem ietekmē rindas garumu vairāk. Visticamākais pieņēmums ir tāds, ka līdz noteiktajai sevlīdzīguma parametra H vērtībai vairāk ietekmē noslodzes koeficients ρ, savukārt pēc šīs robežas vērtības lielāka ietekme ir jau sevlīdzīguma parametram H.

Piemēram, apskatot gadījumu ar parametriem $H = 0,5$ un $\rho = 0,5$ (skat. 1.5a. att.), var palielināt noslodzes koeficientu līdz vērtībai $\rho = 0,6$ (skat. 1.6a. att.) vai sevlīdzīguma parametru līdz vērtībai $H = 0,6$ (skat. 1.5b. att.). Šajos gadījumos var redzēt, ka noslodzes koeficienta ρ palielināšanai ir lielāks efekts, jo maksimālais rindas garums mainās no 15 līdz 21, ja palielināt noslodzes koeficientu līdz $\rho = 0,6$. Savukārt ja palielināt sevlīdzīguma parametru līdz $H = 0,6$, tad maksimālais rindas garums mainīsies no 15 līdz 18. Turklāt otrajā gadījumā maksimālais rindas garums pārsniegs vai būs vienāds ar veco līmeni (15) divos punktos, savukārt pie palielinātas noslodzes – jau 6 punktos. Acīmredzams, ka sevlīdzīguma parametra vērtībai $H = 0,99$ noslodzes koeficientam nav īpašas ietekmes, mainot to robežās $0.3 \leq \rho \leq 1.0$. No otras puses, mainot noslodzes koeficientu ρ no 0.1 līdz 0.3 ir novērojams izteiktais pieaugums rindas garumā. Tātad varētu teikt, ka augsta sevlīdzīguma parametra H vērtība apstiprina efektu, ko veido noslodzes pieaugums.

1.6. Diskrētā veivletu transformācija

1.4. sadaļā izveidotais $P/M/1/K$ *Simulink* modelis, kas parādīts 1.3. att., var būt paplašināts ar sevlīdzīguma parametra H novērtēšanas bloku, kas pēc pieprasījumu plūsmas novērtē šo pieprasījumu avota sevlīdzīguma pakāpi, aprēķinot Hersta parametru H ar veivletu transformāciju. Bloka pamatā ir algoritms, kurš ir aprakstīts [1] un uzlabots [70]. Par pašiem veivletiem ir pietiekami daudz literatūras, piemēram, īss ievads pieejams [93], to pielietošana *Matlab* un *Mathcad* vidēs aprakstīta [94], bet teorētiskā bāze, piemēram, [92]. Ļoti interesants (un perspektīvs) veivletu apstrādes virziens – realizācija ar filtrācijas paņēmieniem, kura ir apskatīta [66].

Veivletu var definēt kā funkciju $\psi(t)$ ar galīgo enerģiju, kas apmierina "pieņemamības nosacījumu":

$$c_\psi = 2\pi \int_{-\infty}^{\infty} \frac{\left|\hat{\psi}(\omega)\right|^2}{|\omega|} d\omega < \infty, \tag{1.11}$$

kur $\hat{\psi}(\omega)$ ir funkcijas $\psi(t)$ spektrālais attēlojums. Praksē apskatāmajiem signāliem (1.11) to var pierakstīt vienkāršākā formā [93]:

$$\int_{-\infty}^{\infty} \psi(t) dt = 0. \tag{1.12}$$

Tas nozīmē ka $\psi(t)$ funkcija oscilē ap laika t asi, bet, ievērojot galīgo reālo signālu enerģiju, šī funkcija ātri dilst kad $t \to \infty$. Šie divi fakti deva funkcijām to nosaukumu – **veivleti** (angl.**wavelets**, "īsie vilnīši"). Vispārinot (1.12), var veidot **"veivletus $\psi(t)$ ar M pirmajiem nulles momentiem"**. Pieņemsim, ka ir uzdota funkcija $\Phi(t)$, kas ir vienmērīga atsevišķajos posmos un kurai eksistē M kārtas atvasinājums. Tādā gadījumā veivletu var definēt šādi:

$$\psi(t) = \frac{d^M \Phi(t)}{dt^M}. \tag{1.13}$$

Tādam veivletam $\psi(t)$ ir M pirmie "nulles momenti", t.i.,:

$$\int_{-\infty}^{\infty} \psi(t) t^m dt = 0 \qquad \text{visiem } m = 0, 1, 2, ..., M-1. \tag{1.14}$$

Līdz ar to, nepārtrauktās veivletu transformācijas izteiksmi var pierakstīt šādi:

$$L_\psi f(a,t) = \frac{1}{\sqrt{c\psi}} \frac{1}{\sqrt{|a|}} \int_{-\infty}^{\infty} \bar{\psi}\left(\frac{u-t}{a}\right) f(u) du \quad (a \neq 0), (t \in R), \tag{1.15}$$

kur t ir laiks, bet a – mēroga koeficients, kas nosaka meklējamo detaļu izmērus[11]. Ar augšsvītru ir apzīmēts kompleksi saistītais lielums, ja veivleta[12] $\psi(t)$ funkcija ir kompleksā. Pēc (1.15) var veikt **diskrēto veivletu transformāciju** ja diskretizēt laikā ar intervālu T_D, tādā veidā iegūstot **diskrētā laika veivletu transformāciju**. Papildus tam var diskretizēt arī mēroga koeficientu a pēc M līmeņiem:

$$L_\psi(a_i, kT_\mathrm{D}) \qquad (k = 0, ..., N-1;\ i = 1, ..., M), \tag{1.16}$$

kur T_D ir laika diskretizācijas periods, N ir signāla nolašu skaits, M ir diskrētā mēroga koeficienta a_i līmeņu skaits. Tātad, var izveidot matricu ar dimensijām $M \times N$, kurā i norāda rindas numuru, bet k – kolonnas numuru. Līdz ar to, tādas transformācijas rezultātā sākotnējais signāls ar N reālām nolasēm tiek pārveidots $M \cdot N$ skaitļos (kuri var būt arī kompleksie, ja izvēlētais bāzes $\psi(t)$ veivlets ir kompleksā funkcija) – bet tas nozīmē, ka informācijas daudzums palielinās M reizēs.

Tā rādījās jautājums – vai ir iespējams atrast tādus veivletus ψ, lai aprēķinot nepārtraukto veivletu transformāciju saskaņā ar (1.15) tikai atsevišķajos punktos $t - a_i$ tā, lai kopējais pārveidoto vērtību skaits pēc (1.16) būtu vienāds ar N un turklāt būtu iespējams rekonstruēt sākotnējo funkciju $f(t)$ pēc šiem skaitļiem? Ja izvēlēties a_i vērtības un veivletu $\psi(kT_D)$ tā, lai jebkurš $A_k = F\{\psi_{a_i}(kT_D)\} \neq 0$ visām k vērtībām[13], tad diskrētā signāla rekonstruēšanā nav jāizmanto visus $M \times N$ matricas skaitļus, bet pietiks tikai ar vienu rindu, kas atbilst izvēlētajai a_i vērtībai. Tas norāda uz to, ka parēja informācija šajā matricā ir pārmērīga – uz tā arī balstās saspiešanas princips ar veivletiem.

Viens no vienkāršākajiem tādu diskrēto veivletu piemēriem – **Hāra veivleti** [93, 92]. Visvienkāršāk tos var apskatīt uz konkrētā piemēra, izvēloties konkrētu diskrēto signālu:

$$\{f_k\} = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}, \qquad 0 \leq k \leq 9.$$

Bez tam, nepieciešama arī t.s. **"mēroga funkcija"**, kas eksistē un atšķiras katram bāzes veivletam un turklāt tā ir vienīga. Hāra veivletam $\psi_\mathrm{H}(t)$ ir šāda mēroga funkcija:

$$\phi_\mathrm{H} = \begin{cases} 1, & \text{ja } 0 \leq t < 1, \\ 0, & \text{pretējā gadījumā.} \end{cases} \tag{1.17}$$

Izmantojot šo mēroga funkciju (1.17) var rekonstruēt nepārtraukto signālu $f(t)$ pēc tā diskrēta-

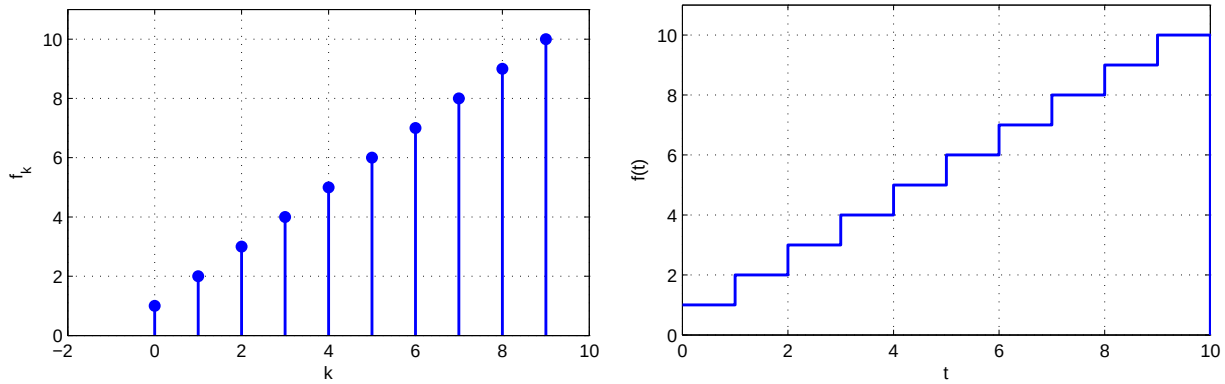
[11] Dažreiz šo koeficient tā arī sauc – "detaļu izmērs", "detalizācijas koeficients".
[12] Šo veivletu sauc par **bāzes**, jeb **mātes** veivletu.
[13] Šeit ar $F\{\}$ ir apzīmēts diskrētās Furjē transformācijas (DFT) operators.

jām nolasēm $\{f_k\}$:

$$f(t) = \sum_{k=0}^{N-1} f_k \phi_{\mathrm{H}}(t-k), \tag{1.18}$$

kur $N = 10$ ir $\{f_k\}$ diskrēto nolašu kopējais skaits.

Piemēram, 1.14. att. ir parādīts nepārtrauktā signāla $f(t)$ rekonstruēšanas process pēc tā diskrētajām laika nolasēm $\{f_k\}$. Kā redzams no 1.14b. att., signāls tiek uzdots intervālos ar soli $T_{\mathrm{D}} = 1$ (diskretizācijas periods), un šajos intervālos signāla līmenis nemainās. Savukārt nolasīšanas momentos kT_{D}, $(k = 0, 2, ..., N-1)$ notiek signāla lēciens no iepriekšējās nolases vērtības līdz jaunās nolases vērtības lielumam, kas sastāda tieši vienīnieku (skat. 1.14a. att.) – ar to ir izskaidrojams "kāpņveidīgais" funkcijas f(t) veids.



a b

1.14. att. Signāls f: a) kā diskrētā secība $\{f_k\}$, b) kā laika funkcija $f(t)$.

Galvenā veivletu transformācijas un saistītas ar to apstrādes ideja ir aproksimēt funkciju vai nolašu secību, atsevišķi saglabājot visas aproksimācijas kļūdas. Šo aproksimāciju var atkārtot iteratīvi līdz momentam, kad rezultāts apmierinās uzdevuma prasības. Hāra veivletu gadījumā aproksimāciju[14] f^1 veic, aprēķinot divu blakus esošo nolašu vidējo vērtību, t.i.,:

$$\{f_k^1\}_{k=0}^{N/2-1} = \left\{ \frac{f_0+f_1}{2}, \frac{f_2+f_3}{2}, \frac{f_4+f_5}{2}, \frac{f_6+f_7}{2}, \frac{f_8+f_9}{2} \right\}, \tag{1.19}$$

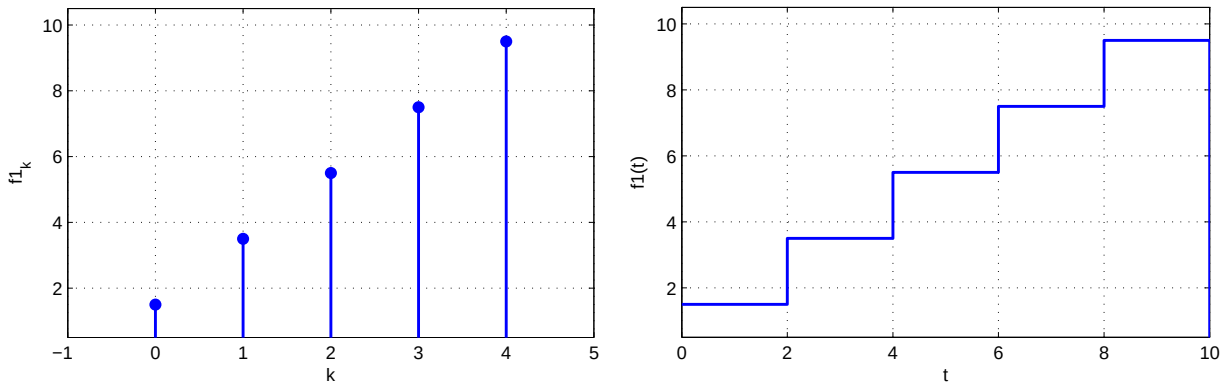
turklāt kopējais nolašu skaits samazinās līdz $N/2$ ja N bija pārskaitlis vai $(N-1)/2$, ja N bija nepārskaitlis. Tas ir skaidri redzams no 1.15a. att., kurā var redzēt 5 signāla aproksimējošās nolases $\{f_k^1\} = \{1,5\ 3,5\ 5,5\ 7,5\ 9,5\}$. Jāatzīmē, ka 1.15b. att. parādītajā attiecīgajā nepārtrauktajā signālā nemainīgais līmenis tiek saglabāts divos diskretizācijas periodos $2T_{\mathrm{D}}$, kā arī "kāpnītes" lēciens palika divreiz lielāks. Šis fakts ir pilnīgi loģisks, jo divas blakus esošās nolases izvēlētajā signālā atšķiras par 1. Tādā veidā, "pārlecot" vienai nolasei pāri citai, šīs nolases vērtība palielinās par 2 vienībām.

Tādā veidā (1.18) var pārrakstīt $f^1(t)$ rekonstruēšanai šādi:

$$f^1(t) = \sum_{k=0}^{N/2-1} f_k^1 \phi_{\mathrm{H}} \left(\frac{t}{2} - k \right). \tag{1.20}$$

Lai saglabātu informāciju par sākotnējo diskrēto nolašu secību $\{f_k\}$, aprēķina "detalizācijas

[14] Funkciju $f^1(t)$ sauc par aproksimējošo funkciju, savukārt $\{f_k^1\}$ secības elementus sauc par aproksimācijas koeficientiem.



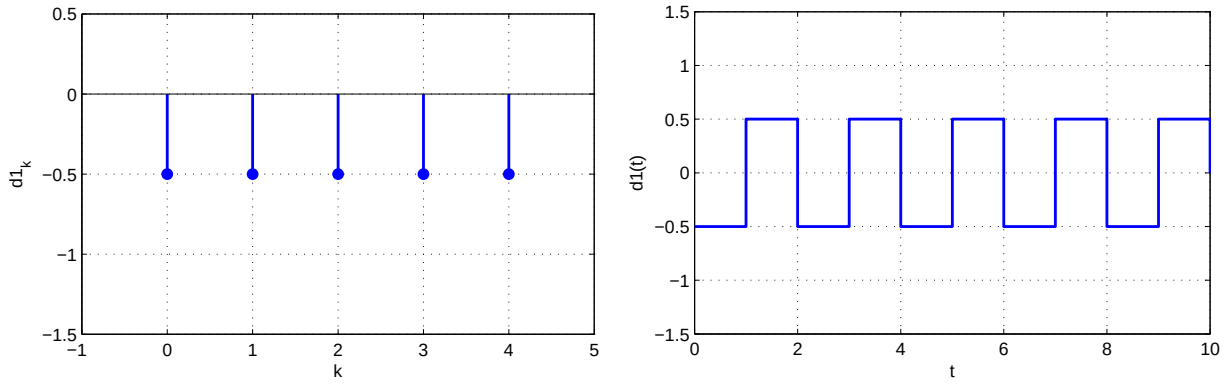
a b

1.15. att. Signāla f aproksimācija f^1: a) kā diskrētā secība $\{f_k^1\}$, b) kā laika funkcija $f^1(t)$.

signālu". Pēc būtības, šis signāls ir sākotnējā signāla $f(t)$ un tā aproksimācijas $f^1(t)$ starpība:

$$d^1(t) = f(t) - f^1(t). \tag{1.21}$$

Šī funkcija $d^1(t)$ ir parādīta 1.16b. att., bet tās diskrētās nolases[15] $\{d_k^1\}$ – 1.16a. att. Tā satur visu informāciju, lai varētu precīzi atjaunot sākotnējo signālu $f(t)$ no tā aproksimācijas $f^1(t)$ bez jebkādiem zudumiem, jo šī detalizācijas funkcija pēc savas būtības ir atšķirība starp signālu $f(t)$ un tā aproksimāciju $f^1(t)$.



a b

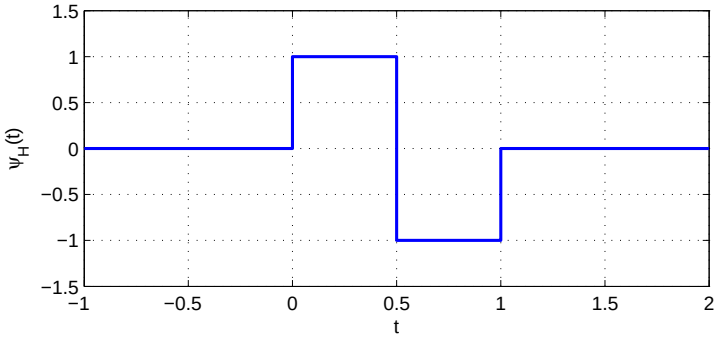
1.16. att. Detalizācijas signāls d^1: a) kā diskrētā secība $\{d_k^1\}$, b) kā laika funkcija $d^1(t)$.

Apskatot 1.16b. att. detalizācijas signāla vidējā vērtība intervālos $[0,2]$, $[2,4]$, $[4,6]$, $[6,8]$ un $[8,10]$ ir vienāda ar nulli: "kāpnītēm" ir vienāds augstums un platums, bet to zīmes ir pretējas. Tas izriet no pašas aproksimācijas procedūras – 1.17. att. ir parādīts tīrais Hāra veivlets (1.22), no kura var izveidot 1.16b. att. parādīto detalizācijas funkciju $d^1(t)$.

$$\psi_{\mathrm{H}}(t) = \begin{cases} 1, & \text{ja } 0 \le t < \frac{1}{2}, \\ -1, & \text{ja } \frac{1}{2} \le t < 1, \\ 0, & \text{pretējā gadījumā.} \end{cases} \tag{1.22}$$

Tādā veidā, detalizācijas funkciju $d^1(t)$ var aprēķināt kā Hāra veivletu superpozīciju (summa) ar attiecīgajiem detalizācijas koeficientiem $\{d_k^1\}$ un nobīdēm laika asī k. Tādā gadījumā, ievērojot no 1.16a. att. to, ka $d_0^1 = d_1^1 = d_2^1 = d_3^1 = d_4^1 = -0.5$, būs spēkā:

[15] Šīs nolases sauc par "detalizācijas koeficientiem".



1.17. att. Hāra veivleta funkcija $\psi_{\mathrm{H}}(t)$.

$$d^1(t) = -0,5 \cdot \psi_{\mathrm{H}}\left(\frac{t}{2}\right) - 0,5 \cdot \psi_{\mathrm{H}}\left(\frac{t}{2} - 1\right) - 0,5 \cdot \psi_{\mathrm{H}}\left(\frac{t}{2} - 2\right) - \\ -0,5 \cdot \psi_{\mathrm{H}}\left(\frac{t}{2} - 3\right) - 0,5 \cdot \psi_{\mathrm{H}}\left(\frac{t}{2} - 4\right). \tag{1.23}$$

Līdzīgi tam, kā no signāla nolasēm $\{f_k\}$ tika veidots nepārtrauktais signāls $f(t)$, pārrakstot (1.20) un aizvietojot tajā ϕ_{H} ar veivletu ψ_{H} un koeficientus $\{f_k^1\}$ ar detalizācijas koeficientiem $\{d_k^1\}$, var iegūt izteiksmi detalizācijas funkcijas $d^1(t)$ aprēķinam:

$$d^1(t) = \sum_{k=0}^{N/2-1} d_k^1 \psi_{\mathrm{H}}\left(\frac{t}{2} - k\right), \tag{1.24}$$

kas ir nekas cits, ka vispārinātā (1.23) izteiksmes pieraksta forma.

Ir svarīgi uzsvērt to, ka detalizācijas koeficientu secību $\{d_k^1\}$ var aprēķināt uzreiz pēc signāla nolasēm $\{f_k\}$, nevis meklējot starpību $\{d_k^1\} = \{f_k\} - \{f_k^1\}$. Aprēķins ir ļoti līdzīgs (1.19) aprēķinam ar vienīgo atšķirību – divu blakus nolašu vidējās summas vietā tiek aprēķināta vidējā starpība:

$$\{f_k^1\}_{k=0}^{N/2-1} = \left\{\frac{f_0 - f_1}{2}, \frac{f_2 - f_3}{2}, \frac{f_4 - f_5}{2}, \frac{f_6 - f_7}{2}, \frac{f_8 - f_9}{2}\right\}. \tag{1.25}$$

Ievērojot visu augstāk minēto, var izveidot likumsakarību starp apskatāmo aproksimācijas/detalizācijas procedūru un nepārtraukto veivletu transformāciju, kuru var aprakstīt ar (1.15) izteiksmi:

$$L_{\psi_{\mathrm{H}}} f(2, 2k) = \sqrt{\frac{2}{c_{\psi_{\mathrm{H}}}}} d_k^1, \qquad (k = 0, 1, ..., N_0), \tag{1.26}$$

kur $N_0 = N/2 - 1$, ja N ir pārskaitlis un $N_0 = (N-3)/2$, ja N ir nepārskaitlis. Šī formula ļauj aprēķināt nepārtraukto veivletu transformāciju ar Hāra veivletiem bez (1.15) formulas integrāļa novērtēšanas vai skaitliskās aprēķināšanas. Tomēr pastāv arī ierobežojums – $L_{\psi_{\mathrm{H}}} f(a, t)$ ir aprēķināms tikai vērtībai $a = 2$ visiem $t = 2k$, $(k = 0, 1, ...)$.

Apkopojot visu, var definēt nepārtrauktās veivletu transformācijas ātro algoritmu ar Hāra veivletu izmantošanu:

1. Diskrēto signālu $\{f_k\}$ ar mēroga funkciju ϕ_{H} (1.17), kas atbilst veivletam ψ_{H} (1.22) pārveido nepārtrauktajā signālā $f(t)$ saskaņā ar (1.18).

2. Pēc formulām (1.19) un (1.25) tiek aprēķinātas attiecīgās diskrētās secības $\{f_k^1\}$ un $\{d_k^1\}$.

3. Pēc (1.26) formulas tiek aprēķināta funkcijas $f(t)$ nepārtrauktā veivletu transformācija ar Hāra veivletu pie $a = 2$ visiem $t = 2k$, $(k = 0, 1, ...)$.

4. Signālu $f(t)$ var rekonstruēt pēc aproksimējošā signāla $f^1(t)$ (1.20) un detalizējošā signāla $d^1(t)$ (1.24), aprēķinot to summu: $f(t) = f^1(t) + d^1(t)$.

Jāatzīmē, ka veivletu transformāciju var iteratīvi atkārtot – signāla $f(t)$ vietā tiek paņemta tā aproksimācija $f^1(t)$, no kuras tiek aprēķināta vēl rupjāka aproksimācija $f^2(t)$ ar attiecīgā detalizācijas funkcija $d^2(t)$, utt. Turklāt pašas šīs procedūras var veikt ar diskrētajām nolasēm, neoperējot ar nepārtrauktajām funkcijām. Aproksimācijas un detalizācijas procedūras var aprakstīt ar (1.27) un (1.28), attiecīgi:

$$f^1 = Hf, \tag{1.27}$$

kur H procedūra iekļauj sevī divus posmus: ciparu filtrāciju ar filtra impulsa reakciju $\{h_k\} = \{1/2, 1/2\}$ un turpmāko **decimāciju**[16].

$$d^1 = Gf, \tag{1.28}$$

kur G procedūra iekļauj sevī divus posmus: ciparu filtrāciju ar filtra impulsa reakciju $\{g_k\} = \{1/2, -1/2\}$ un turpmāko decimāciju (*downsampling*). Rezultātā, gan f^1, gan d^1 ir divreiz īsāki signāli, salīdzinot ar f.

Arī signāla $f(t)$ rekonstruēšanā var lietot ciparu filtrus, nerēķinot nepārtraukto aproksimējošo signālu $f^1(t)$ (1.20) un nepārtraukto detalizējošo signālu $d^1(t)$ (1.24). Tā vietā var izmantot inversos filtrus H^* ar koeficientiem $\{h_0 = 1/2, h_1 = 1/2\}$ un G^* ar koeficientiem $\{g_0 = 1/2, g_1 = -1/2\}$. Savukārt, diskretizācijas frekvence pēc šiem filtriem tiek palielināta 2 reizēs (pēc katras nolases tiek iestarpinātas nulles vērtības, veicot t.s. "upsampling" operāciju – interpolēšanu ar nullēm). Tādā veidā, sākotnējā signāla nolases $\{f_k\}$ var rekonstruēt pēc (1.29) saskaitot divus signālus šo filtru izejās un sareizinot rezultātu ar 2:

$$\{f_k\} = 2(H^*\{f_k^1\} + G^*\{d_k^1\}). \tag{1.29}$$

Kā var redzēt, veivletu transformācijas aproksimācijas/detalizācijas procedūras, kā arī sākotnējā signāla nolašu atjaunošana pēc attiecīgajiem koeficientiem var tikt veikta ar ciparu filtrācijas paņēmieniem bez nepieciešamības rēķināt integrālo nepārtraukto veivletu transformāciju. Apskatītie Hāra veivleti ir visvienkāršākie, taču pastāv arī citi diskrētie veivleti, piemēram, Dobeši-N veivleti [93, 92], kur N ir veivleta nulles momentu skaits saskaņā ar (1.14).

1.7. *P/M/1/K* modelis ar *H* parametra novērtēšanas bloku

[1] piedāvātais algoritms var strādāt ar jebkuriem diskrētajiem veivletiem, piemēros zemāk tiks izmantoti Dobeši veivleti ar $N = 3$, kuriem ir 3 t.s. **"nulles momenti"** (1.14). Detalizācijas un aproksimācijas filtru koeficientu skaits tādā gadījumā sastādīs $2N = 6$. Aproksimācijas filtra koeficienti Dobeši 3 veivletu gadījumā ir šādi [93]:

$$\{h_k\}_{k=0}^3 = \{0,332671 \;\; 0,806892 \;\; 0,459878 \;\; -0,135011 \;\; -0,085441 \;\; 0,035226.\} \tag{1.30}$$

Savukārt detalizācijas filtra koeficientus Dobeši veivletu gadījumā var aprēķināt [93] saskaņā pēc izteiksmes:

$$g_k = (-1)^k h_{1-k} \qquad \text{visiem } k = 1, 2, ..., 2N. \tag{1.31}$$

[16] Decimācijas (*downsampling*) procesā tiek saglabāts tikai katrs otrais secības elements, t.i., tiek izlaisti secības elementi ar nepārskaitļa indeksiem. Varētu teikt arī šādi: diskretizācijas frekvence tiek samazināta 2 reizēs.

Tādā gadījumā, signāla nolašu dekompozīciju var veikt saskaņā ar 1.6.sadaļā aprakstīto filtrācijas algoritmu pēc (1.27) un (1.28). Signāla rekonstruēšana šajā uzdevumā nav vajadzīga, taču nepieciešamības gadījumā to var bez problēmām realizēt pēc (1.29) izmantojot tos pašus filtrus ar koeficientiem (3.1) un (3.2), aizvietojot decimācijas operāciju ar disktrētizācijas frekvences palielināšanu divās reizēs ("upsampling" operācija).

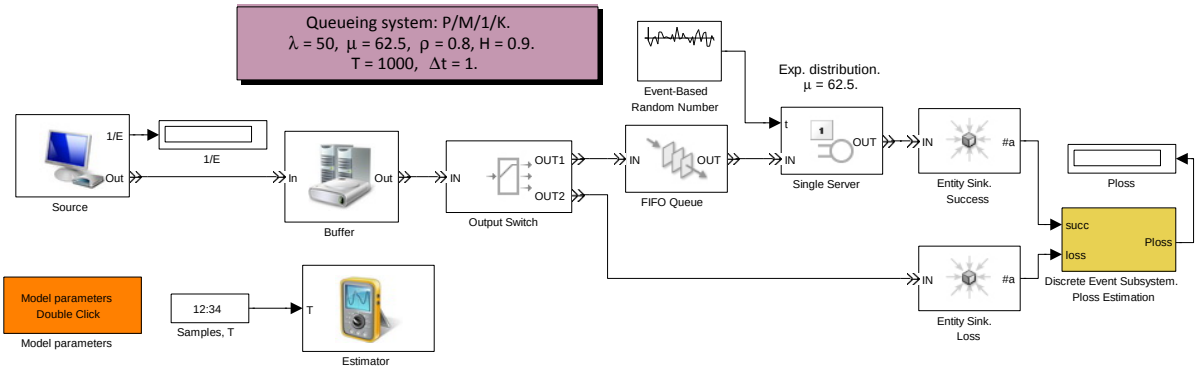
Tādā gadījumā sevlīdzīguma parametra vērtību H var novērtēt [46], aprēķinot minimālo kvadrātisko funkciju ar svariem mērogu intervālā starp j_1 un j_2 pēc šādas izteiksmes:

$$\hat{H}(j_1, j_2) = \frac{1}{2}\left[\frac{\sum_{j=j_1}^{j_2} S_j j n_j - \sum_{j=j_1}^{j_2} S_j j \sum_{j=j_1}^{j_2} S_j n_j}{\sum_{j=j_1}^{j_2} S_j \sum_{j=j_1}^{j_2} S_j j^2 - \left(\sum_{j=j_1}^{j_2} S_j j\right)^2} + 1\right], \tag{1.32}$$

kur $n_j = log_2\left(\frac{1}{n}\sum_k \left|d_k^j\right|^2\right)$, savukārt svari $S_j = (n\ln^2 2)/2^{j+1}$.

Ievērojot visu augstāk apskatīto par veivletiem, 1.4. sadaļā izveidoto $P/M/1/K$ *Simulink* modeli, kas parādīts 1.3. att., var paplašināt ar sevlīdzīguma parametra H novērtēšanas bloku tā, kā tas ir parādīts 1.18. att.[17]. Visus modeļa parametrus var uzstādīt kopējā blokā **Model Parameters**, bet konkrēti:

- pieprasījumu intensitāti λ;
- apstrādes mezgla veiktspēju μ;
- sevlīdzīguma parametru H;
- sevlīdzīguma parametra novērtēšanas intervālu periodu T;
- solis Δt, ar kuru pieprasījumi tiks nolasīti intervālā T.



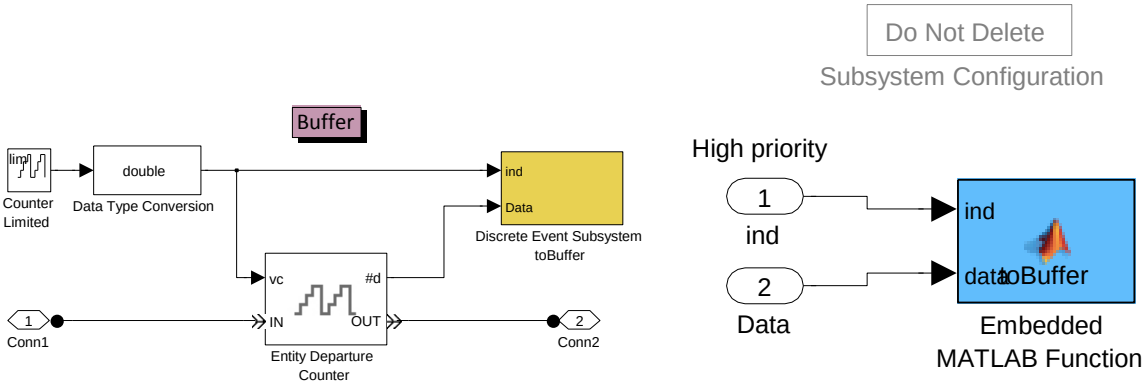
1.18. att. *Simulink* $P/M/1/K$ modelis ar sevlīdzīguma parametra H novērtēšanas bloku.

Pēdējie divi parametri T un Δt nosaka veivletu transformācijas loga garumu $M = T/\Delta t$, t.i., pieprasījumu secības elementu skaitu, pēc kuriem tiek novērtēts sevlīdzīguma parametrs H. Šo loga garumu M var nosacīti pielīdzināt T intervāla periodam, ja pieņemt soli $\Delta t = 1$, tādā gadījumā sevlīdzīguma parametra novērtēšanā tiks ievērotas visi pieprasījumu secības elementi. Šo datu saglabāšanai *Matlab* vides **Datas** masīvā $P/M/1/K$ sistēma tika papildināta ar bufera apakšsistēmu **Buffer** 1.19a. att., kuram nav nekādā sakara ar buferatmiņas apjomu K, kurā var

[17] Kā var redzēt, šajā attēlā vairāki bloki, kas veido avotu, tika apvienoti vienā kopējā blokā ar attiecīgās apakšsistēmas veidošanu.

veidoties rindas, ja apstrādes mezgls nepaspēj noteiktajā laika posmā apstrādāt visus pieprasījumus. Šis **Buffer** ir nepieciešams tikai datu uzkrāšanai un saglabāšanai *Matlab* vidē, lai pie tiem varētu lai turpmāk varētu veikt veivletu transformāciju un turpmāko sevlīdzīguma parametra H novērtējumu. 1.19b. att. parādītā *Matlab* lietotāja funkcija ieraksta *Matlab* masīvā **Datas** pieprasījumu daudzumu, kuru nosaka ar skaitītāju **Entity Departure Counter** 1.19a. att. un saglabā šo skaitu ar kārtas numuru **ind**. Šis funkcijas kods *Matlab* valodā ir dots zemāk:

```
function toBuffer(ind,data)
global datas;
if ind==0,
    ind = length(datas);
end
datas(ind) = data;
```



a b

1.19. att. *Simulink* $P/M/1/K$ modeļa bufera apakšsistēma **Buffer** pieprasījumu secības nolašu uzkrāšanai.

[1, 70] piedāvātais algoritms aprēķina tikai momentānās sevlīdzīguma parametra novērtējuma $\hat{H}[n]$ vērtības, kuras ievērojami svārstās laikā (to var redzēt tālāk, piemēram, 1.21. att.). Rezultātu var precizēt [63], iteratīvi aprēķinot vidējo vērtību $\hat{H}_{\text{avg}}[n]$ pēc formulas:

$$\hat{H}_{\text{avg}}[n] = \frac{(n-1)\cdot \hat{H}_{\text{avg}}[n-1] + \hat{H}[n]}{n}. \tag{1.33}$$

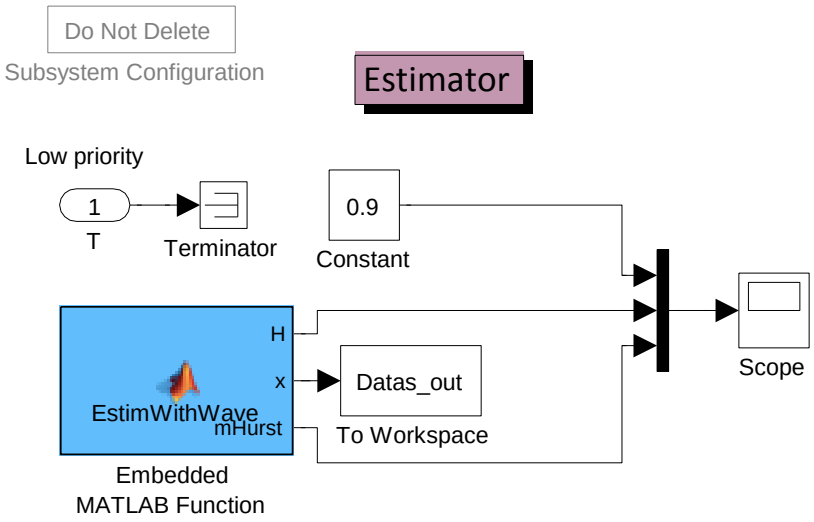
Turklāt momentānie sevlīdzīguma parametra novērtējumi $\hat{H}[n]$ var arī nebūt aprēķināti vispār, t.i., neeksistēt, (piemēram, ja nav pietiekamais pieprasījumu daudzums, pauzes posms) vai dot kļūdaino rezultātu $\hat{H}[n] < 0$. Tādā gadījumā vidējais sevlīdzīguma parametra novērtējums $\hat{H}_{\text{avg}}[n]$ tiek saglabāts laikā:

$$\hat{H}_{\text{avg}}[n] = \hat{H}_{\text{avg}}[n]. \tag{1.34}$$

Iespējamas arī situācijas, kad momentānie sevlīdzīguma parametra novērtējumi $\hat{H}[n] > 1$, tādā gadījumā tos ir jānoapaļo līdz vieniniekam:

$$\hat{H}[n] = 1, \qquad \text{ja } \hat{H}[n] > 1. \tag{1.35}$$

Visus šos nosacījumus ir jāievēro novērtēšanas blokā **Estimator** no 1.18. att., kurš ir parādīts sīkāk 1.20. att. Kā var redzēt, lielākā šī bloka funkciju daļa ir realizēta tiešajā veidā ar *Matlab*



1.20. att. *Simulink* $P/M/1/K$ modeļa sevlīdzīguma parametra H novērtēšanas apakšsistēmas bloks.

valodu blokā **Embedded Matlab Function** ar nosaukumu **EstimWaveHurst**. Šīs funkcijas kods ir dots zemāk[18]:

```
function [H,x,mHurst] = EstimWithWave
global datas meanHurst
x = datas.';
mHurst = 0;
regu = 3;
nbvoies = fix( log2(length(x)) );
xx = initDWT_discrete(x,regu);
[muj,nj]=wtspec(xx,regu,nbvoies);
j1opt = newchoosej1(regu, nj, muj, length(nj));
j1 = j1opt; j2 = length(muj);
[H,Q]=regrescomp(regu, nj, muj, j1, j2);
if ~isnan(H) && H>0
  if H>1, H=1; end
  meanHurst(2) = meanHurst(2) * meanHurst(1) + H;
  meanHurst(1) = meanHurst(1) + 1;
  meanHurst(2) = meanHurst(2) / meanHurst(1);
end
mHurst = meanHurst(2);
```

Šajā funkcijā ir pielietojama [1] autoru veidotās funkcijas:

- **xx = initDWT_discrete(x,regu)** – Dobeši N veivletu filtra inicializācijas procedūra un datu x sagatavošana, *regu* ir nulles momentu skaits $N = 3$;

[18] Bez funkciju kodiem, kuri ir attiecīgo funkciju failos vienā direktorijā ar *Simulink* modeļa failu. Šīs funkcijas realizēja [1] autori.

- **[muj,nj]=wtspec(xx,regu,nbvoies)** – Diskrētās veivletu transformācijas aprēķins pec datiem xx, $nbvoies$ ir aproksimācijas/detalizācijas procedūru iterāciju (atkārtošanu) skaits, bet rezultātā funkcija atgriež detalizācijas koeficientus $d_k^j = muj$ un $n_j = nj$ vērtības no (1.32);

- **j1opt = newchoosej1(regu, nj, muj, length(nj))** – optimālās j_1 vērtības $j1opt$ novērtēšana j diapazonam izteiksmē (1.32);

- **[H,Q]=regrescomp(regu, nj, muj, j1, j2)** – sevlīdzīguma parametra H novērtējuma aprēķināšana pēc (1.32), pieņemot par j_1 optimālo vērtību $j1opt$;

- pārējā funkcijas daļā tiek aprēķinātās iteratīvais vidējais H novērtējuma lielums pēc (1.33) ievērojot nosacījumus (1.33) un (1.33).

Pēc līdzīga algoritma tika izveidota arī *Matlab* programma tādas pašas pieprasījumu secības ģenerēšanai ar turpmāko sevlīdzīguma parametra H novērtēšanu pēc šīs pilnās realizācijas. Šīs programmas koda tekstu satur 2.pielikums.

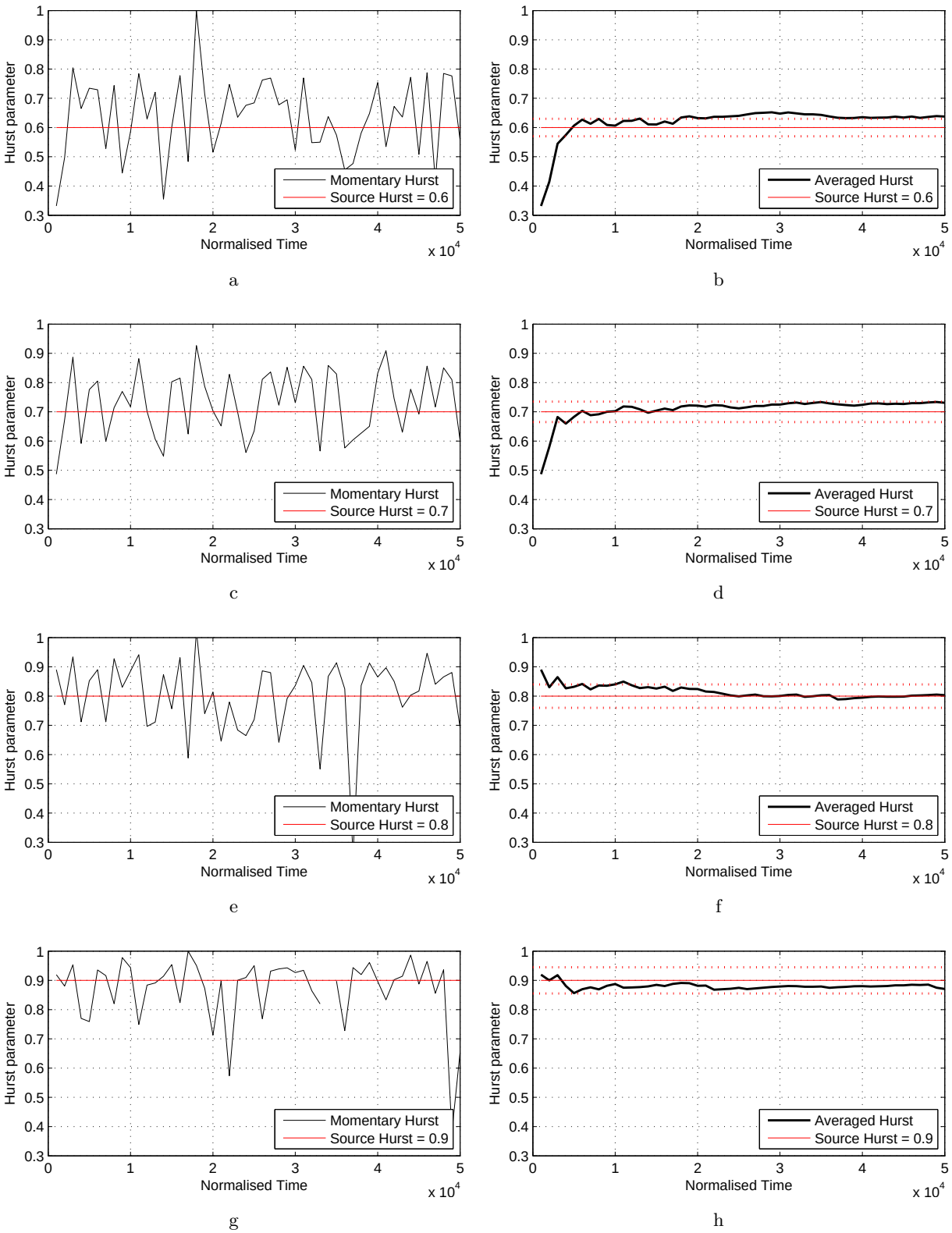
1.8. *P/M*/1/K modeļa simulācijas Hersta parametra novērtēšanas rezultāti

Simulācijas laikā sevlīdzīguma parametra novērtējumi $\hat{H}[n]$ (momentānais), $\hat{H}_{\text{avg}}[n]$ (vidējais) un H (uzstādītais) tiek saglabāti *Matlab* vidē mainīgajā **ScopeData**, kuru pēc tam var saglābāt failā vai izmantot jebkurās *Matlab* programmās, piemēram, grafiku veidošanā, statistisko raksturojumu aprēķinos utt. Visās simulācijas kopējais laiks sastāda $T_{\text{sim}} = 5 \cdot 10^4$ normētas laika vienības, neatkarībā no $T = 1000$ novērojuma perioda un diskrētā soļa $\Delta t = 1$, kas nosaka kopējo veivletu transformācijas loga garumu $M = T/\Delta t = 1000$

Simulāciju rezultāti ir parādīti 1.21. att. – kreisajā kolonnā momentānās sevlīdzīguma parametra novērtējuma $\hat{H}[n]$ vērtības, labajā – iteratīvi aprēķinātās pēc (1.33) vidējās vērtības $\hat{H}_{\text{avg}}[n]$, ievērojot nosacījumus (1.34) un (1.35). Grafikos ir atzīmēts arī avotā uzstādīta sevlīdzīguma parametra vērtība H, kā arī 5 % kļūdas diapazoni no šīs iestādītas vērtības (tikai grafikiem no labās kolonnas). Rezultāti ir apkopoti arī 1.7. tabulas veidā, kura tika papildināta arī ar sevlīdzīguma parametra novērtējumu $\hat{H}_{\text{sim}}$ pēc pilnās pieprasījumu secības realizācijas, t.i., pielietojot [46] aprakstīto algoritmu visiem ģenerētiem datiem, jeb, kas ir viens un tas pats, palielinot veivletu transformācijas logu no $T = 1000$ līdz $T = 5 \cdot 10^4$ vienībām.

1.7. tabula. H parametra novērtējumu salīdzinājums pie noslodzes koeficienta $\rho = 0,8$, veivletu transformācijas loga garuma $T = 1000$ ar soli $\Delta t = 1$.

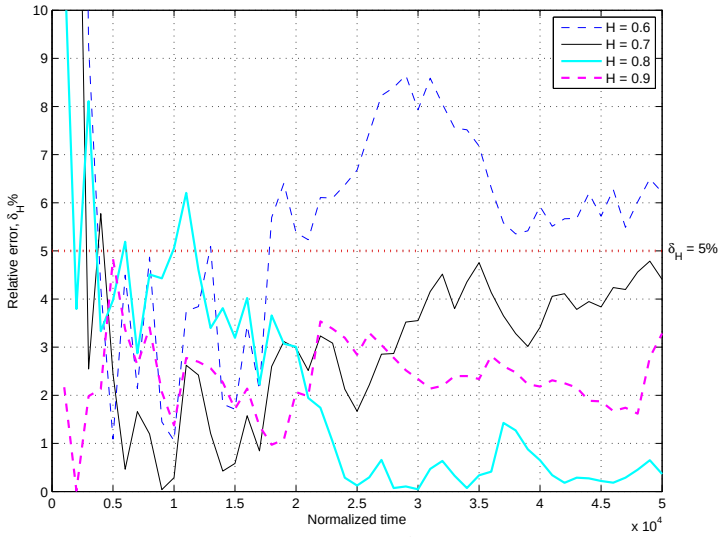
Avota H parametra vērtība	**Iteratīvais vidējais H parametra novērtējums**		**Pilnās realizācijas H parametra novērtējums**
	Vid. vērtība	**Dispersija, σ^2**	
$H = 0,6$	0,6196	0,0030	0,6410
$H = 0,7$	0,7092	0,0016	0,7287
$H = 0,8$	0,8149	0,0004	0,7992
$H = 09$	0,8803	0,0001	0,8625



1.21. att. Sevlīdzīguma parametra H novērtējumi – momentānās (kreisajā kolonnā) un vidējās (labajā kolonnā) vērtības pie noslodzes koeficienta $\rho = 0,8$, veivletu transformācijas loga garuma $T = 1000$ ar soli $\Delta t = 1$: a,b) $H = 0,6$, c,d) $H = 0,7$, e,f) $H = 0,8$, g,h) $H = 0,9$.

1.21g. att. laika intervālā $t = (32500; 35000)$ sevlīdzīguma parametru nevarēja novērtēt, kas varētu notikt nepietiekamā pieprasījumu daudzuma dēļ (trafika momentānā intensitāte ir pārāk zema). Tādā gadījumā nevar novērtēt sevlīdzīguma parametra vērtību tikai pēc loga garumā aprēķinātajiem datiem un varētu būt nepieciešams ievērot iepriekšējās novērtējuma vērtības. Tādā veidā, atbilstoši (1.34) un (1.35) attiecīgajā laika intervālā $t = (32500; 35000)$ sevlīdzīguma parametra vidējais novērtējums $\hat{H}_{\text{avg}}[n]$ paliek nemainīgs[19], kā to var redzēt no 1.21h. att. grafika.

Kā var redzēt no 1.7. tabulas datiem, sevlīdzīguma parametra novērtējuma $\hat{H}$ dispersija σ^2 samazinās ar paša Hersta parametra H pieaugumu. Savukārt pašas $\hat{H}$ novērtējuma vērtības pie visām H parametra vērtībām ir precīzākas pēc iteratīvā vidējā lieluma aprēķina, salīdzinot ar Hersta parametra novērtējumu $\hat{H_{\text{sim}}}$ un tas ir viennozīmīgi pie šādām sistēmas parametru T, Δt un ρ vērtībām. Tas skaidri norāda uz to, ka loga garumam ir noteiktā optimālā vērtība – to nedrīkst palielināt līdz bezgalībai, jo, ka var redzēt no 1.7. tabulas datiem precizitāte tādā gadījuma samazinās. Jāatzīmē arī tas, ka visos gadījumos, izņemot $H = 0,9$ novērtējumi pārsniedz faktiski esošo H parametra vērtību avotam. Visprecīzākais novērtējums (abos gadījumos) ir pie avota sevlīdzīguma parametra vērtības $H = 0,8$ – tas varētu būt izskaidrojams ar pielietojamajiem veivletiem (Dobeši N veivleti) vai to nulles momentu skaitu N.



1.22. att. Sevlīdzīguma parametra novērtējumu $\hat{H}_{\text{avg}}[n]$ relatīvās kļūdas procentos attiecībā pret avotā uzstādīto vērtību H pie noslodzes koeficienta $\rho = 0,8$.

Novērtējot precizitāti var uzdot 5 % kļūdas intervālu (vai arī jebkuru citu), lai noteiktu, cik ātri sevlīdzīguma parametra vidējais novērtējums $\hat{H}_{\text{avg}}[n]$ tiecas pie uzstādītās avotā vērtības H. Šie 5 % intervāli ir iezīmēti 1.21b.,1.21d.,1.21f.,1.21h. att., bet pašu kļūdu var aprēķināt visiem sevlīdzīguma parametra vidējiem novērtējumiem $\hat{H}_{\text{avg}}[n]$ procentos kā relatīvo kļūdu attiecībā pret uzstādīto avota vērtību H pēc formulas:

$$\delta\hat{H}[n] = \frac{\left|E[\hat{H}_{\text{avg}}[n]] - -H\right|}{H} \cdot 100\,\%. \tag{1.36}$$

Aprēķinātas pēc (1.36) kļūdas visām $H = \{0,6 \;\; 0,7 \;\; 0,8 \;\; 0,9\}$ vērtībām ir parādītas 1.22. att. kopā ar 5 % intervālu. Kopumā, var konstatēt, ka pie zemākām H parametra vērtībām novērtējumu $\hat{H}_{\text{avg}}[n]$ deviācija no uzstādītās vērtības ir augstāka, taču visos gadījumos sevlīdzīguma parametra novērtējums tiecās pie uzstādītās vērtības pēc noteiktā laika intervāla. Izskatās, ka šīs laika intervāls nav īpaši atkarīgs no sevlīdzīguma parametra H vērtības. Izņēmums

[19]Tomēr, šeit būtu jāievēro ka tieši šīs "pauzes" arī satur informāciju par gadījumprocesa sevlīdzīguma mēru, jo laika intervāli starp pieprasījumiem ir sadalīti pēc Pareto likuma ar bezgalīgo dispersiju.

ir redzams 1.21b. att., kur novērtējuma vidējā vērtība iziet no 5 % kļūdas koridora.

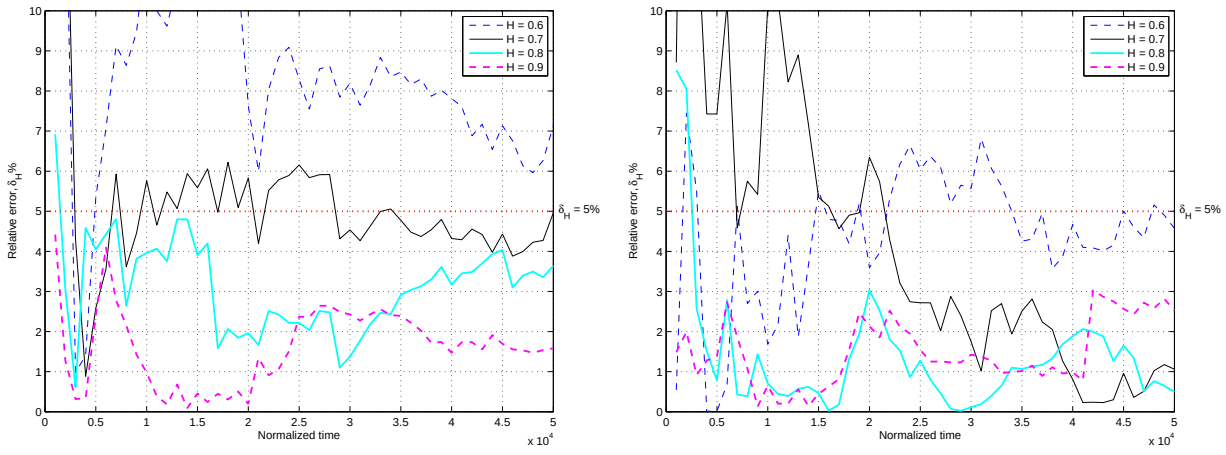
1.8. tabula. H parametra novērtējumu salīdzinājums pie noslodzes koeficienta $\rho = 0,65$, veivletu transformācijas loga garuma $T = 1000$ ar soli $\Delta t = 1$.

Avota H parametra vērtība	**Iteratīvais vidējais H parametra novērtējums**		**Pilnās realizācijas H parametra novērtējums**
	Vid. vērtība	**Dispersija, σ^2**	
$H = 0,6$	0,6421	0,0012	0,6391
$H = 0,7$	0,7262	0,0010	0,7145
$H = 0,8$	0,8218	0,0002	0,7882
$H = 0,9$	0,8875	0,0001	0,8623

1.9. tabula. H parametra novērtējumu salīdzinājums pie noslodzes koeficienta $\rho = 0,95$, veivletu transformācijas loga garuma $T = 1000$ ar soli $\Delta t = 1$.

Avota H parametra vērtība	**Iteratīvais vidējais H parametra novērtējums**		**Pilnās realizācijas H parametra novērtējums**
	Vid. vērtība	**Dispersija, σ^2**	
$H = 0,6$	0,6229	0,0003	0,6353
$H = 0,7$	0,6708	0,0010	0,7025
$H = 0,8$	0,7976	0,0003	0,7950
$H = 0,9$	0,8895	0,0001	0,8662

Turpmākajai analīzei tika mainīts noslodzes koeficients ρ un atkārtotas simulācijas. Tāpat kā 1.5.sadaļā apskatītajos rezultātos, vidējais apstrādes laiks tika saglabāts bez izmaiņām $\mu = 62,5$, bet noslodzes koeficients tika mainīts ar pieprasījumu vidējās intensitātes λ izmaiņām. Tika izpētīti divas noslodzes koeficienta ρ vērtības, kas atšķiras no 1.7. tabulas eksperimenta parametriem abās pusēs par lielumu $\Delta\rho = \pm 0,15$, t.i., sastādīja lielumus $\rho_1 = 0,65$ un $\rho_2 = 0,95$. Tāda noslodzes koeficienta nodrošināšanai ir jāizvēlas vidējās pieprasījumu intensitātes $\lambda_1 = 0,65\mu$ un $\lambda_2 = 0,95\mu$, attiecīgi. Simulācijas sēriju rezultāti ir apkopoti 1.8.–1.9. tabulās, kā arī relatīvās kļūdas $\delta_{\hat{H}_{\text{avg}}}$ grafikos 1.23. att.



a b

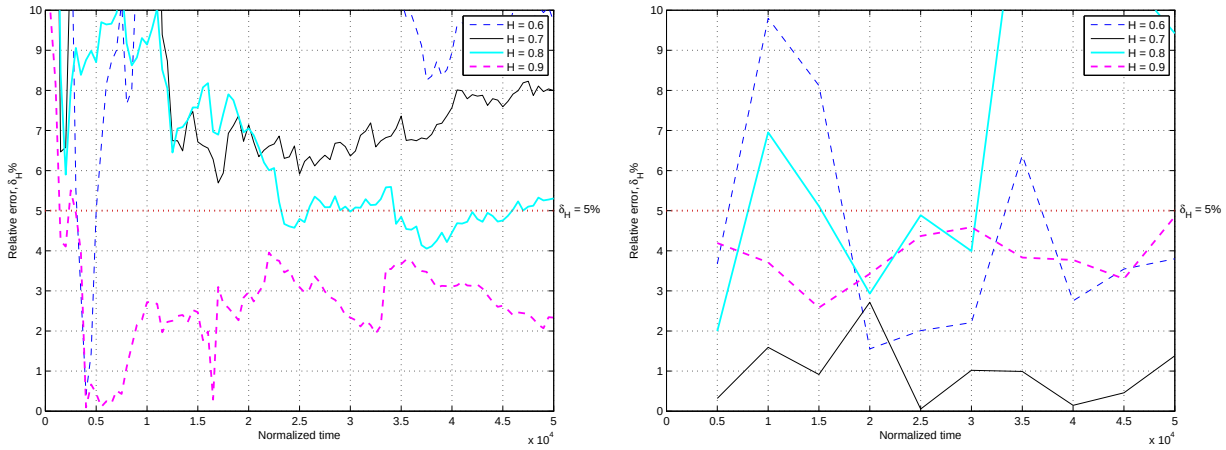
1.23. att. Sevlīdzīguma parametra novērtējumu $\hat{H}_{\text{avg}}[n]$ relatīvās kļūdas procentos attiecībā pret avotā uzstādīto vērtību H pie noslodzes koeficienta: a) $\rho = 0,65$, b) $\rho = 0,95$.

Analizējot 1.8.–1.9. tabulu datus, var nonākt pie secinājuma, ka mainoties noslodzei (abās pusēs), sevlīdzīguma parametra novērtēšanas precizitāte pārsvarā samazinās – gan iteratīvā vi-

dējā novērtējumam $\hat{H}_{\text{avg}}[n]$ gan novērtējumam pēc pilnās simulācijas datu realizācijas $\hat{H}_{\text{sim}}$. Tomēr, apskatot augstākās sevlīdzīguma pakāpes, it īpaši $H = 0,9$ gadījumā, precizitāte uzlabojas un šīs uzlabojums ir lielāks gadījumā ar $\rho = 0,95$. Šajā gadījumā pie $H = 0,8$ abi novērtējumi dod ļoti precīzas vērtības – tas ir vienīgais gadījums, kurā nav principiālās atšķirības kādu novērtējumu pielietot – pēc pilnās realizācijas $\hat{H}_{\text{sim}}$ vai iteratīvo vidējo $\hat{H}_{\text{avg}}[n]$. Pārējos gadījumos rezultāti ir pārāk neviennozīmīgi, lai varētu veikt secinājumus par loga garuma T ietekmi pie dažādām H parametra vērtībām.

Salīdzinot relatīvās H parametra kļūdas pēc 1.23a. att., 1.22. att. un 1.23b. att. grafikiem, var secināt, ka pie zemākām sevlīdzīguma parametra vērtībām, bet konkrēti pie $H = 0,6$, jo zemāks ir noslodzes koeficients, jo stiprākas ir novirzes no vidējās novērtējuma vērtības $\hat{H}_{\text{avg}}[n]$[20]. Turpretī pie augstajām sevlīdzīguma parametra vērtībām, šajā gadījumā pie $H = 0,9$, spriežot pēc augstāk minētajiem grafikiem, noslodzes koeficientam ρ nav lielās ietekmes, jo pie visām trijām noslodzes koeficienta ρ vērtībām relatīvās kļūdas līknes $\delta_{\hat{H}_{\text{avg}}}$ gadījumam $H = 0,9$ izskatās ļoti līdzīgi[21].

Noslodzes koeficienta ρ izmaiņas nenoveda pie rezultātiem, pēc kuriem varētu veikt secinājumus par tā ietekmi uz novērtējumu precizitāti. Vadoties pēc rezultātiem no 1.8.–1.9. tabulām var secināt tikai to, ka veivletu transformācijas loga garuma $M = T/\Delta t$ izmaiņas ietekmē novērtējumu precizitāti vairāk, nekā noslodzes koeficienta ρ izmaiņas. Tāpēc nākamajā simulāciju sērijā tiek mainīts veivletu transformācijas loga garums M. Stingri sakot, šīs lielums ir atkarīgs no diviem parametriem – novērošanas intervāla perioda T, kurā tiek uzkrāti dati pirms transformācijas aprēķina, un solis starp datu nolasēm Δt, ar kura palīdzību, principā, var veikt **decimācijas** ("downsampling") operāciju, t.i., izlaist nolases. Šajās simulācijās solis $\Delta t = 1$, un tādā veidā loga garums M ir skaitliski vienāds ar novērošanas intervāla periodu T. Tādā gadījumā, var uzdot divas šī perioda vērtības $T_1 = 500$ un $T_2 = 5000$, kurus arī uzskatīt par veivletu transformācijas loga garumu M.



a b

1.24. att. Sevlīdzīguma parametra novērtējumu $\hat{H}_{\text{avg}}[n]$ relatīvās kļūdas procentos attiecībā pret avotā uzstādīto vērtību H pie noslodzes koeficienta $\rho = 0,8$ un loga garuma: a) $T = 500$, b) $T = 5000$.

[20] Šeit tiek salīdzināts tikai iteratīvais vidējais novērtējums, jo novērtējums pēc pilnās realizācijas ir vienīgais skaitlis.

[21] Tas, principā arī būtu loģiski sagaidīt augsto sevlīdzīgumu pie vērtības $H = 0,9$, kura raksturīga stipri sevlīdzīgajiem procesiem.

Simulāciju sērijas rezultāti ir apkopoti 1.10.–1.11. tabulās, kuras uzrādītas novērtējuma vērtības pec pilnās realizācijas kā papildus informācija[22]. Relatīvas vidējā novērtējuma kļūdas $\delta_{\hat{H}_{\text{avg}}}$ ir parādītas 1.24a.,1.24b. att. grafikos. Informācijai par novērtējumu atkarībām, tajā skaitā momentāna sevlīdzīguma parametra novērtējumiem $\hat{H}[n]$, tika izveidoti arī grafiki 1.25. att. pie $T = 500$ un 1.26. att. pie $T = 5000$ visām četrām sevlīdzīguma parametra H vērtībām.

1.10. tabula. H parametra novērtējumu salīdzinājums pie noslodzes koeficienta $\rho = 0,8$, veivletu transformācijas loga garuma $T = 500$ ar soli $\Delta t = 1$.

Avota H parametra vērtība	**Iteratīvais vidējais H parametra novērtējums**		**Pilnās realizācijas H parametra novērtējums**
	Vid. vērtība	**Dispersija, σ^2**	
$H = 0,6$	0,6471	0,0048	0,6410
$H = 0,7$	0,7574	0,0004	0,7287
$H = 0,8$	0,8523	0,0004	0,7992
$H = 0,9$	0,8895	0,0007	0,8625

1.11. tabula. H parametra novērtējumu salīdzinājums pie noslodzes koeficienta $\rho = 0,8$, veivletu transformācijas loga garuma $T = 5000$ ar soli $\Delta t = 1$.

Avota H parametra vērtība	**Iteratīvais vidējais H parametra novērtējums**		**Pilnās realizācijas H parametra novērtējums**
	Vid. vērtība	**Dispersija, σ^2**	
$H = 0,6$	0,5902	0,0009	0,6410
$H = 0,7$	0,7017	0,00007	0,7287
$H = 0,8$	0,7424	0,0011	0,7992
$H = 0,9$	0,8652	0,00004	0,8625

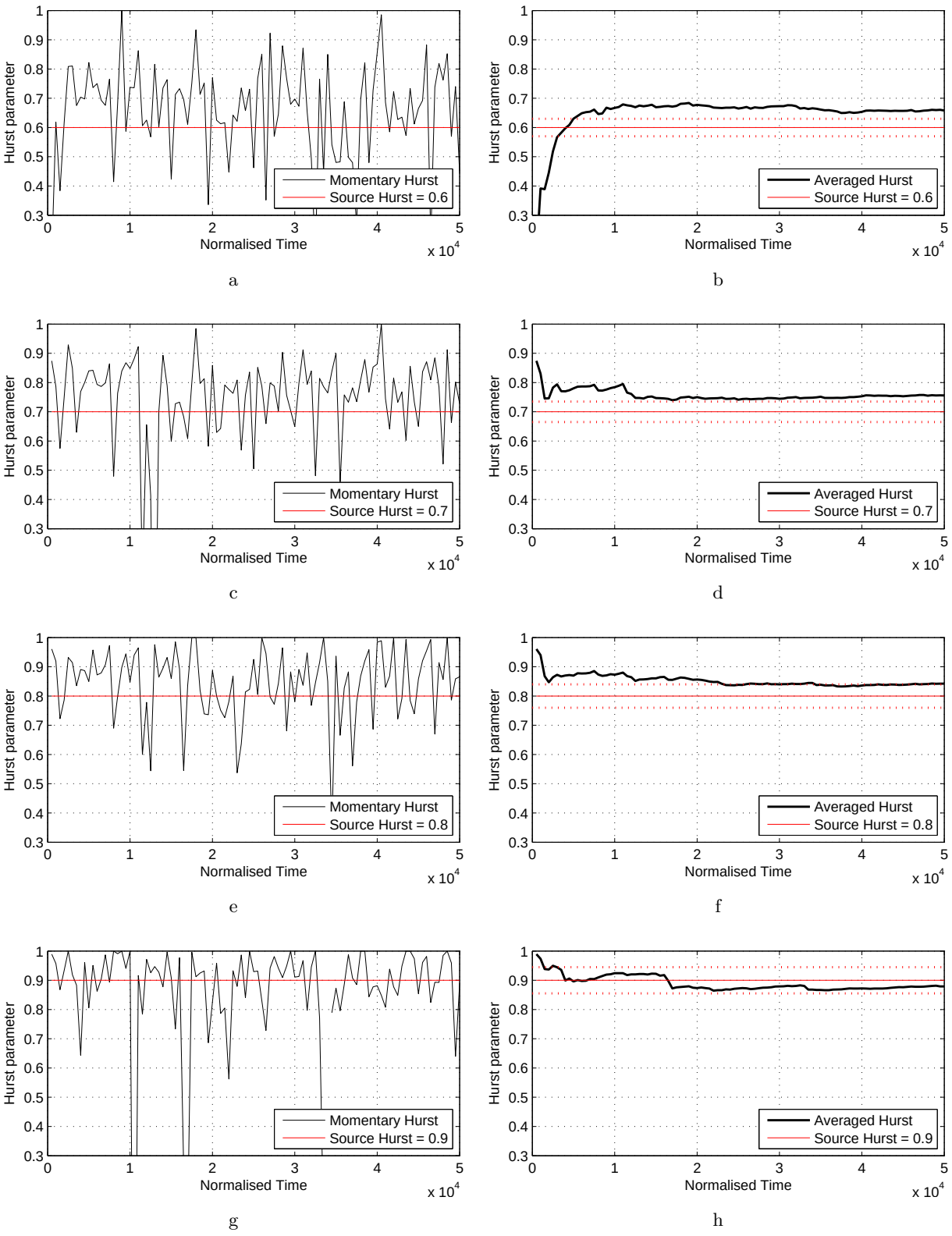
Salīdzinot 1.10. tabulas,1.11. tabulas, kā arī 1.7. tabulas datus, var skaidri redzēt, ka vidējā novērtējuma matemātiskā cerība $E[\hat{H}_{\text{avg}}]$ atšķiras vismazāk no iestādītās H parametra vērtības:

- pie $H = 0,6$ gadījumā ar $T = 5000$;
- pie $H = 0,7$ gadījumā ar $T = 5000$;
- pie $H = 0,8$ gadījumā ar $T = 1000$;
- pie $H = 0,9$ gadījumā ar $T = 500$.

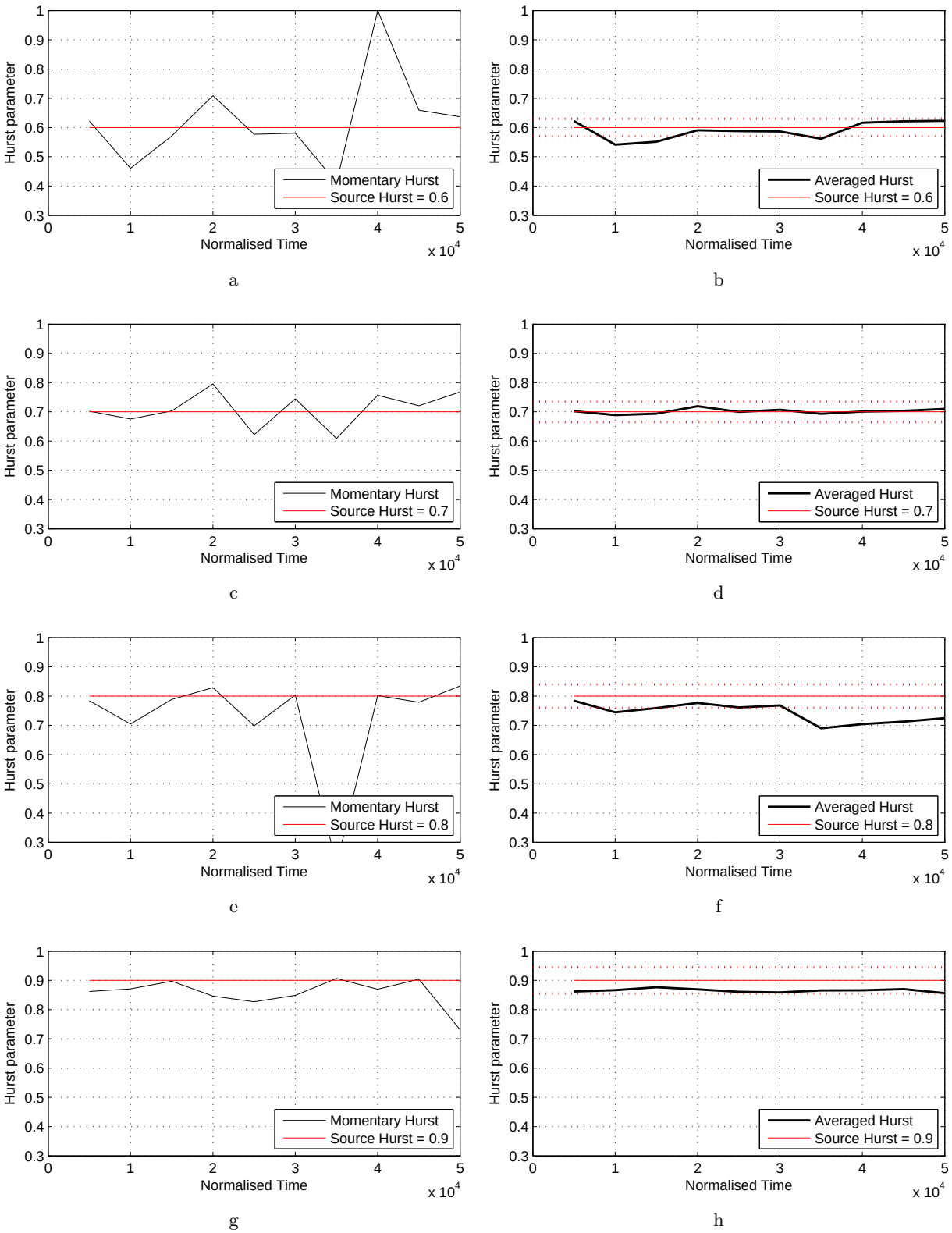
Arī attiecīgās dispersiju σ^2 vērtības mainās robežās $0,00007 \leq \sigma^2 \leq 0,0009$, kas ir vismazākie lielumi iegūtie simulāciju laikā. Līdz ar to, skaidri novērojama tendence – pie augstākām sevlīdzīguma parametra H vērtībām ir jāizvēlas īsākais logs un otrādi. Tomēr, arī pārāk garš logs nav lietderīgs, kā to var redzēt no 1.11. tabulas datiem, kur novērtējumam pēc pilnās realizācijas (šeit var uzskatīt par loga garumu $T = 5 \cdot 10^4$) precizitāte ir ievērojami zemāka, nekā vidējai novērtējumu vērtībai ar loga garumu $T = 5000$.

Līdzīgu informāciju var iegūt arī no 1.24a.,1.24b.,1.22. att. grafiku salīdzināšanas. Tā, piemēram, novērtējuma kļūdu pie $H = 0,6$ 1.24b. att. var uzskatīt par zemāku, nekā to pašu kļūdu 1.22. att., un noteikti par zemāku, nekā 1.24a. att. Arī gadījumā ar kļūdu novērtējumiem pie $H = 0,7$ situācija ir tāda pati, savukārt gadījumā ar $H = 0,9$ – pilnīgi pretēja.

[22] Šīs vērtības neatšķiras savā starpā šajās divās tabulās, kā arī no vērtībām 1.7. tabulā, jo visos šajos gadījumos ir viena un tā pati gadījumprocesa realizācija, kuru veido gadījumskaitļu ģenerators ar sākuma stāvokli 12345.



1.25. att. Sevlīdzīguma parametra H novērtējumi – momentānās (kreisajā kolonnā) un vidējās (labajā kolonnā) vērtības pie noslodzes koeficienta $\rho = 0,8$, veivletu transformācijas loga garuma $T = 500$ ar soli $\Delta t = 1$: a,b) $H = 0,6$, c,d) $H = 0,7$, e,f) $H = 0,8$, g,h) $H = 0,9$.



a b c d e f g h

1.26. att. Sevlīdzīguma parametra H novērtējumi – momentānās (kreisajā kolonnā) un vidējās (labajā kolonnā) vērtības pie noslodzes koeficienta $\rho = 0,8$, veivletu transformācijas loga garuma $T = 5000$ ar soli $\Delta t = 1$: a,b) $H = 0,6$, c,d) $H = 0,7$, e,f) $H = 0,8$, g,h) $H = 0,9$.

1.9. Nobeigums

Šajā ievada nodaļā tika aprakstīts mūsdienu trafiks, tā sevlīdzīguma parādība un veiktas vairākas ilustrējošas simulācijas. Ir veikta literatūras analīze, kas parāda ka tēma ir joprojām aktuāla un attīstās, lietojot arvien jaunākās metodes un palielinot jauno uzdevumu skaitu, apskatot vairākas metodes, kas dod sinerģiju.

Nodaļas tekstā tika parādīta un aprakstīta metode, pēc kuras tiks ģenerēts simulācijas trafiks pārējās nodaļās, kā arī parādīts ar eksistējošo metodi, ka promocijas darba uzdevums ir principiāli risināms. Ir aprakstīta metode, ar kuras palīdzību tiks noteikts Hersta parametra novērtējums $\hat{H}$ un tā vidējā vērtība $E[\hat{H}]$.

Simulāciju gaitā tika novērtēts buferatmiņās apjoms K rindošanas modelim ar sevlīdzīgo trafiku un ierobežotu buferatmiņu. Simulācijas dati nesakrīt ar pieejamo analītisko izteiksmju rezultātiem, kas norāda uz to pielietošanas neiespējamību. Šis novērojums ir apstiprināms arī citos literatūras avotos, un tas nozīmē ka buferatmiņas apjomu ir jānovērtē adaptīvi, mērot rindošanas sistēmas parametrus – Hersta parametru H un noslodzes koeficientu ρ, lai panāktu noteikto pakešu zudumu varbūtības P_{loss} vērtību.

Nodaļā ir sniegts arī ieskats diskrētās veivletu transformācijas teorijā, pēc kuras turpmākajās darba nodaļās tiks izveidotas praktiski lietojamās programmas, tajā skaitā arī Hersta parametra H vērtības novērtēšanai.

Nodaļa 2

Reāllaika diskrētā veivletu transformācija ar filtru bankām

2.1. Reāllaika veivletu transformācija literatūrā

Literatūras analīze parādīja, ka daudzmērogu diskrētās veivletu transformācijas realizācija reāllaikā nav pietiekoši labi apskatīta. Nekur neizdevās atrast šādu algoritmu, lai viņu varētu pielietot sevlīdzīguma parametra novērtēšanai reāllaikā. Šeit ir jāatzīmē, ka interesē tieši tāds veivletu transformācijas algoritms, kurš apstrādātu katru nolasi visos mērogos, nevis uzkrātu datu segmentu un apstrādātu to tālāk.

Tieši šādā veidā, pārsvarā, literatūra ir sastopams tiešās un inversās diskrētās veivletu transformācijas algoritms – pēc labi aprakstīta daudzajos teorētiskajos līdzekļos (piem., [66]) t.s. Malata piramidālā filtru bankas algoritma [49]. Saskaņā ar šo algoritmu, tiek uzkrāts datu segments, kas tālāk tiek apstrādāts ar $j = 1$ mēroga filtru banku (viss segments), pēc tam iegūtie aproksimācijas koeficienti tiek apstrādāti ar $j = 2$ mēroga filtru banku, utt. Šādu pieeju veiksmīgi izmanto pat reāllaika apstrādē, piemēram, maģistra darbā [10] audio signālu reāllaika apstrādei. Citos darbos un rakstos tiek apskatīta iedalīšana segmentos – ar pārklāšanās vai segmentu robežas efekta mazināšanos (piem., [58]).

Ir plaši aprakstītas diskrētās veivletu transformācijas aparāt-realizācijas, īpaši populāras ir realizācijas ar programmējamo loģiku *FPGA* (piem. [5], [14]). Pietiekoši izsmeļoši tādas realizācijas ir apskatītas maģistra darbā [65]. Šajā darbā tika piedāvātas dažādas realizācijas – tiešā, polifāzes, kāpņveida struktūras. Ir arī daudzi raksti, kuros ir veikti noteiktie veiktspējas uzlabojumi, piemēram, [41], [72], [89]. Ir arī uzlabojumi jaudas patēriņā reizinājumu veikšanai [54]. Tomēr, aparātrealizācija ir visai specifiska un nevar būt pielietojama visur, piemēram, simulāciju veidošanā ir vai nu jāizmanto simulācijas rīkus, kas var imitēt *FPGA* darbību, vai nu jāsimulē patstāvīgi tādu darbību.

Pastāv arī veivletu transformācijas algoritmi signālu ciparapstrādes procesoriem (*DSP*) [9], [59], tajā skaitā arī veicot paralēlo apstrādi [74] veivletu transformācijas algoritma skaitļošanas efektivitātes palielināšanai.

Ja pievērst uzmanību visu šo darbu izlaides gadiem, var konstatēt, ka šīs jautājums tiek sen pētīts un attīstīts. Parādās arī jaunākie darbi, kuros tiek palielināts dimensiju skaits un tiek meklēti efektīvākie apstrādes algoritmi, piemēram, [25], [26] darbos. Citos darbos algoritmi tiek optimizēti atkarībā no pielietojamā uzdevuma, piemēram, [62], [24]. Kā tika minēts augstāk, cits svarīgs faktors ir energoefektivitāte, kas arī uz doto brīdi tiek pētīts, par to liecina dažādas publikācijas, piemēram, neironu tīklu apstrādē [68].

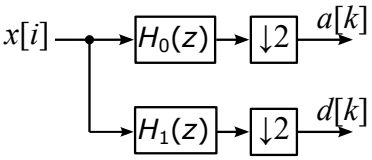
Savukārt, ja apskatīt universālos mikroprocesorus vai mikrokontrolierus, tad atrast tādu algoritmu neizdevās vispār. Kā jau tika minēts augstāk ar norādēm uz dažādiem avotiem, šajā gadījumā pat reāllaika apstrādei tiek uzkrāts noteiktā garuma segments, kurš arī tiek apstrādāts ar filtru banku, rēķinot diskrēto konvolūciju. Tādu filtru banku nav sarežģīti realizēt, bet ir iespējams uzlabot veiktspēju, ja realizēt šo filtru banku ar polifāzes vai kāpņveida filtriem.

Vistuvākais apraksts vēlamajai reāllaika daudzmērogu diskrētajai veivletu transformācijai tika atrasts darba autoreferātā [85]. Šīs darbs tika aizstāvēts 2012. gadā, kas ir salīdzinoši

nesen, ja salīdzināt ar iestrādājumiem citās arhitektūrās vai segmentu apstrādē. Tomēr pats šīs darbs nav pieejams, līdz ar to nav pieejams arī apskatītais šādā darbā algoritms. Taču no šī autoreferāta satura var secināt, ka apskatāmais algoritms atrada praktisko pielietojamību. Nav aprakstīts arī pēc kādas shēmas tika veidota filtru banka, kādi ir uzlabojumi veikti algoritma veiktspējas uzlabošanai.

2.2. Reāllaika tiešā veivletu transformācija

Pastāv vairāki daudzmērogu veivletu transformācijas aprēķina algoritmi (gan tiešajai transformācijai, gan arī inversajai). Tomēr, visos šajos algoritmos ir nepieciešams uzkrāt noteikta garuma apstrādājamo datu bloku, kura garums noteiks mērogu skaitu[1]. Ievērojot to, ka trafika sevlīdzīguma parametra H noteikšanai ir jāveic diskrēto veivletu transformāciju ievērojamajā mērogu skaitā, tad apstrādājamo datu (trafika) segmentu garums būs arī ievērojams lielums.



2.1. att. Diskrētā veivletu transformācija ar filtru banku, tiešā realizācija.

Galvenais šeit ir tas, ka rezultāts nevar būt izmantojams, kamēr nenotiks visa bloka apstrāde. Tomēr, ja veivletu transformācija tiek aprēķināta ar filtru banku pieeju (skat. 2.1. att.), reāli pirmos rezultātus varētu sākt izmantot pirms visa segmenta apstrādes, ja realizēt šo filtru darbību tiešajā veidā laikā, nerēķinot diskrēto konvolūciju no visiem datiem. Tas ir iespējams, jo pati transformācijas procedūra, pēc savas būtības, ir nekas cits kā ciparu filtrācija, kuru ir iespējams veikt reāllaikā. Turklāt veivletu transformācijas gadījumā ļoti bieži tiek pielietoti nerekursīvie ciparu filtri (2.2. att.), kuru impulsa reakcijai ir ierobežotais garums, tātad arī laika aizture līdz pirmajam rezultātam ir prognozējamā un, kas svarīgāks, nesasniedz augstās vērtības. Protams, konkrētā laika aiztures vērtība ir atkarīga no filtra impulsa reakcijas garuma un no veivletu transformācijas mērogu skaita (katrā mērogā tiek ieviesta laika aizture), bet tā attiecās uz pašu filtrācijas procesa sākumu. Tikko pirmais rezultāts ir parādījies, nākamie rezultāti tiks iegūti bez papildus laika aiztures. Vēl jo vairāk, kā tiks pāradīts vēlāk, šai laika aizturei ir noteicošā loma ideālajai signāla rekonstruēšanai, jeb sintēzei (vai tuvajai pie ideālās), bet šī operācija pilnīgi netiek pielietota sevlīdzīguma parametra novērtēšanai.

Šeit būtu jāatzīmē, ka reāllaika veivletu transformācijas aprēķins neizslēdz nepieciešamību uzkrāt rezultātu – detalizējošos koeficientus, pēc kuriem tiek novērtēts sevlīdzīguma parametrs H. Šajā gadījumā visa uzkrāšana notiks ar *FIFO* veida bufera atmiņas pielietošanu, turklāt parādās iespēja adaptīvi mainīt šīs bufera atmiņas apjomu, un, tātad apstrādājamā segmenta garumu, kas būtu 2 reizēs garāks par detalizācijas koeficientu skaitu bufera atmiņā. Vispār, jāievēro tas, ka sevlīdzīguma parametra novērtēšanai nav nepieciešami paši detalizācijas koeficienti, bet to kvadrātiskā summa:

$$D_j = \frac{1}{n_j} \sum_{k=0}^{n_j-1} d_j[k]^2, \tag{2.1}$$

kur j ir mēroga numurs, kurā tiek aprēķināta jauda D_j, $d_j[k]$ – attiecīgā mēroga detalizācijas koeficienti $d[k]$ un n_j – tādu koeficientu skaits mērogā j.

Tas dod iespēju pārrēķināt šo kvadrātisko summu reāllaikā tikai attiecībā uz jaunāko $d_j[0]^2$ vērtību, kā arī uz vecāko $d_j[K-1]^2$ vērtību (kuras kvadrātu būtu jāatņem), kur K ir detalizācijās

[1] Atgādinājumam, katrā diskrētās veivletu transformācijas mērogā datu blokam ar garumu M, veidojās $M/2$ aproksimācijas koeficienti un $M/2$ detalizācijas koeficienti. No šiem koeficientiem nākamajā veivletu transformācijas līmenī (mērogā) izmantojami tikai aproksimācijas koeficienti.

koeficientu $d_j[k]$ bufera atmiņas apjoms:

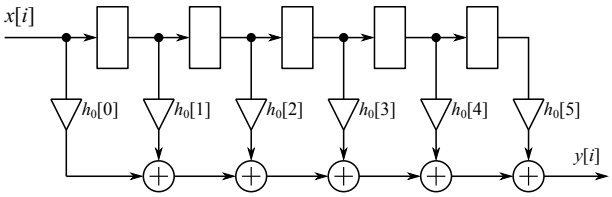
$$D_j[i] = \frac{n_j D_j[i-1] - d_j[K-1]^2 + d_j[0]^2}{n_j}, \tag{2.2}$$

kur $D_j[i]$ ir tekošā detalizācijas koeficientu jauda, savukārt $D_j[i-1]$ – šī vērtība iepriekšējam H parametra novērtējumam.

Tādā veidā, jaunās nolases filtrācija ar filtru bankām notiks jebkurā gadījumā tikai vienu reizi – ja segmenti nepārklājās, tad attiecīgajā segmentā šī vērtība tiks padota filtru bankas ieejā tikai vienu reizi, savukārt reāllaika apstrādē tā arī būs padota filtru ieejā tikai vienu reizi. Tātad, filtrācijas daļā operāciju skaits nemainās.

Savukārt, minimālo kvadrātu metodē turpmākajai H parametra novērtēšanai dažādos mērogos arī var pārrēķināt noteiktās summas, papildinot tās ar jaunā mēroga detalizācijas koeficientu jaudas logaritmu $\log_2(D_j)$. Turklāt maksimālais mērogu skaits varētu sastādīt lielumu ap 20, kas ir samērā liels lielums, ievērojot, ka $j = 20$ diskrētās veivletu transformācijas mērogā var novērtēt vienu aproksimācijas koeficientu un vienu detalizācijas koeficientu ne agrāk kā pēc 2^{20} trafika nolasēm. Tas viss, kopumā, palielinās segmentu skaitu tā, it kā apstrādājamie trafika segmenti pārklājās gandrīz pilnīgi, izņemot vienu nolasi, bet, tomēr, bez milzīgajiem apstrādes laika virstēriņiem

Tāpēc, faktiski, palielinās tieši H parametra novērtējumu skaits, taču tādi novērtējumi neprasa lielu operāciju skaitu, savukārt jo vairāk tādu novērtējumu, jo precīzāk var rēķināt vidējo sevlīdzīguma parametra H novērtējumu, ja tas ir nepieciešams[2].



2.2. att. Ciparu filtrs ar galīgo impulsa reakciju (*FIR*) Dobeši-3 veivletiem.

2.2. att. ir parādīts vispārinātais Dobeši-3 veivletu filtrs ar kārtu $N = 5$, kas tiek izmantots analīzes (dekompozīcijas) un sintēzes (rekonstruēšanas) filtru banku realizēšanā. Konkrētajiem filtriem mainās tikai impulsu reakcijas koeficienti:

- analīzes filtru bankā – zemfrekvenču filtrs ar impulsa reakcijas koeficientiem $h_0[n]$ un augstfrekvenču filtrs ar impulsa reakcijas koeficientiem $h_1[n]$, kur $n = 0, 1, 2, ..., N$;
- sintēzes filtru bankā – zemfrekvenču filtrs ar impulsa reakcijas koeficientiem $f_0[n]$ un augstfrekvenču filtrs ar impulsa reakcijas koeficientiem $f_1[n]$, kur $n = 0, 1, 2, ..., N$;

Jāatzīmē, ka ortogonālo filtru banku gadījumā (pie kura attiecināmi arī Dobeši-3 veivleti) visi šo filtru impulsa reakcijas koeficienti var būt aprēķināti no analīzes filtru bankas aproksimācijas filtra koeficientiem ar pārvades funkciju $H_0(z)$. Pārējo filtru impulsa reakcijas vai pārvades funkcijas var noteikt [66] saskaņā ar sekojošajām izteiksmēm, analīzes filtru bankas detalizācijas filtram:

$$H_1(z) = z^{-N} H_0(-z^{-1}) = \sum_{n=0}^{N} (-1)^n h_0[n] z^{-(N-n)}, \tag{2.3}$$

[2] Acīmredzami, kad H parametrs dinamiski mainās, vidējās vērtības aprēķins ienesīs papildus "inerci", jo nav iespējams droši atbildēt – vai H parametrs ir patiesībā mainījies, vai tikai tā novērtējums, apstrādājot nākamo segmentu.

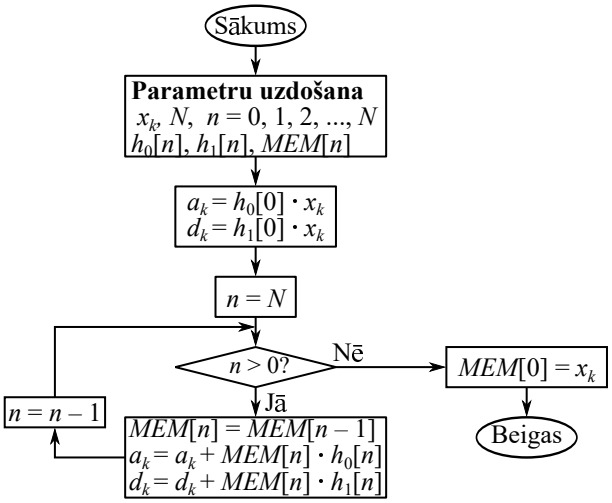
kur N ir filtra kārta, sintēzes filtru bankas aproksimācijas filtram:

$$F_0(z) = z^{-N} H_0(z^{-1}) = \sum_{n=0}^{N} h_0[n] z^{-(N-n)}, \tag{2.4}$$

un sintēzes filtru bankas detalizācijas filtram:

$$F_1(z) = -H_0(-z) = -\sum_{n=0}^{N} (-1)^n h_0[n] z^{-n}. \tag{2.5}$$

2.3. att. zemāk ir parādīts algoritms, kas realizē 3.1. att. parādīto diskrētās veivletu transformācijas analīzes filtru banku ar impulsa reakcijām $h_0[n]$ un $h_1[n]$, jeb pārvades funkcijām $H_0(z)$ un $H_1(z)$, (2.3) ar piebildi, ka šajā realizācijā decimācijas operācija (katrās nepāra skaita numura nolases izlaišana) netiek veikta, atšķirībā no 3.1. att. parādītā algoritma. Pēc būtības, šis algoritms realizē divus ciparu filtrus ar kārtu N, kuru ieejās ir padots viens un tas pats signāls un noņemtas divas signāla izejas vērtības.



2.3. att. Reāllaika tiešās veivletu transformācijas filtru bankas realizācijas algoritms.

Tā kā tiek paredzēts realizēt šo algoritmu daudzmērogu diskrētajai veivletu transformācijai *C/C++* programmēšanas valodā, ir nepieciešamība saglābāt katra filtra stāvokli katrā mērogā un izmantot tos nākošās signāla nolases apstrādē kā filtru sākumstāvokļus. Šie sākumstāvokļi $MEM_0[n]$ un $MEM_1[n]$, kuri šajā filtru bankā sakrīt un var būt aizvietoti ar vienu kopējo stāvokli $MEM[n]$, kur $n = 0, 1, 2, ..., N$. Tādā gadījumā, šo divu filtru struktūra izskatīsies kā ir parādīts zemāk 2.4. att. Šādas aiztures līnijas garums atbilst impulsa reakcijas koeficientu skaitam 2.2. att. parādītajā filtrā un sastāda $L = N + 1$. Bez sākumstāvokļa tiek uzdota arī aproksimācijas filtra impulsa reakcijas koeficienti $h_0[n]$ un detalizācijas filtra impulsa reakcijas koeficienti $h_1[n]$, pie $n = 0, 1, 2, ..., N$.

Aproksimācijas koeficienta a_k un detalizācijas koeficienta d_k vērtības tiek aprēķinātas kā ieejas signāla x_i pēdējo L nolašu un filtru stāvokļa $MEM[n]$[3] diskrētās konvolūcijas, ja $n = 0, 1, ..., N$:

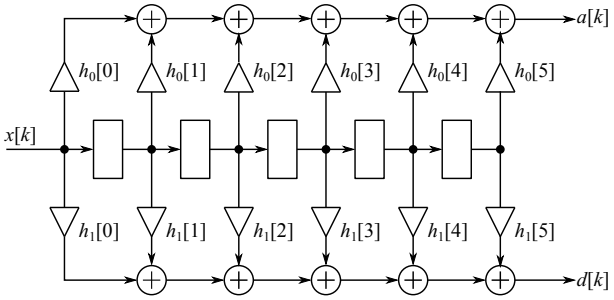
$$a_k = h_0[0] x_i + \sum_{n=1}^{N} MEM[n-1] h_0[n], \tag{2.6}$$

[3]Pēc būtības, filtra stāvoklis šajā gadījumā ir nekas cits, ka N iepriekšējās signāla x_i vērtības.

kur x_i ir tekošā signāla nolase, un

$$d_k = h_1[0]x_i + \sum_{n=1}^{N} MEM[n-1]h_1[n]. \tag{2.7}$$

Abas diskrētās konvolūcijas tiek izskaitļotas vienā ciklā, turklāt šajā pašā ciklā notiek arī filtra stāvokļa nobīde ar aiztures elementiem. Funkcijas izpildes rezultātā tiek izskaitļots aproksimācijas koeficients a_k un detalizācijas koeficients d_k uz vienu ieejas signāla nolasi x_k, kā arī tiek saglabātas iepriekšējās N signāla nolases kā filtru stāvoklis.

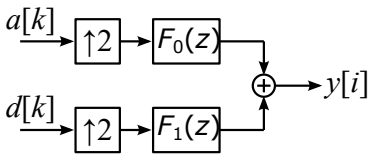


2.4. att. Ciparu filtru banka ar kopējo aiztures līniju Dobeši-3 veivletiem.

Rezultātā tika izveidota funkcija, kas realizē filtrāciju no 2.4. att. Atstājot a_k un d_k skaitļu secībās tikai katru nepārskaitļa numura nolasi (pie nosacījuma, ka numerācija sākas ar "0"), var pabeigt diskrētās veivletu transformācijas aprēķinu vienā mērogā. Iegūtos aproksimācijas koeficientus a_k var izmantot kā ieejas signālu tādai pašai filtru struktūrai, lai veiktu veivletu transformāciju $j = 2$ mērogam ar turpmāko decimācijas operāciju. Šo procesu var atkārtot līdz nepieciešamā rezultāta sasniegšanai, pie nosacījuma ka sākotnējām signālam ir pietiekoši lielais nolašu skaits, lai veivletu transformāciju varētu izpildīt kārtējam mērogam. Pretējā gadījumā vienkārši pietrūks ieejas signāla nolases šādai transformācijai.

2.3. Reāllaika inversā veivletu transformācija

2.2.sadaļā tika apskatīts viena mēroga reāllaika tiešās diskrētās veivletu transformācijas algoritms. Šajā sadaļā tiks apskatīts līdzīgs algoritms viena mēroga inversajai diskrētajai veivletu transformācijai. Inverso diskrēto veivletu transformāciju arī var veikt ar filtru bankām – šajā gadījumā runa iet par sintēzes filtru banku (2.5. att.), kuru veido aproksimācijas filtrs ar pārvades funkciju $F_0(z)$ un detalizācijas filtrs ar pārvades funkciju $F_1(z)$. Pirmā filtra ieejā tiek padoti aproksimācijas koeficienti a_k pēc diskretizācijas frekvences divkāršošanas ("upsampling" operācija, jeb interpolācija ar nullēm – tā ir nuļļu ieviešana starp nolasēm), savukārt otrajā filtrā – attiecīgi, detalizācijas koeficienti d_k.

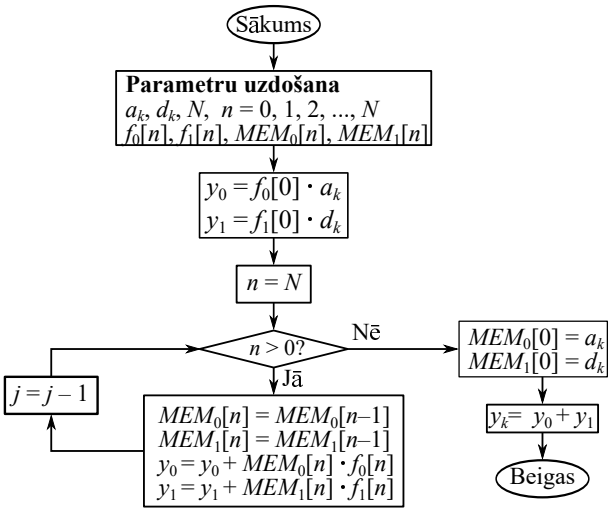


2.5. att. Inversā diskrētā veivletu transformācija ar sintēzes filtru banku.

Sintēzes filtru bankas veidojošo filtru pārvades funkcijas ir saistītas ar analīzes filtru bankas pārvades funkcijām saskaņā ar (2.4) aproksimācijas filtram un (2.5) detalizācijas filtram[4]. Pēc

[4] Jāpievērš uzmanību, ka faktiski $H_0(z)$ un $F_0(z)$ pārvades funkciju pāris, kā arī $H_1(z)$ un $F_1(z)$ pāris ir savstarpēji spoguļattēlā.

filtrācijas procedūras vērtības abu filtru izejā tiek saskaitītas un rezultātā tiek aprēķinātas signāla nolases x_i, vai lielākā mēroga $j-1$ aproksimācijas koeficienti $a_{j-1,i}$, ja runa iet par transformāciju ar mērogu j. Acīmredzami, ka inversā diskrētā veivletu transformācija tiek rēķināta pretējā kārtībā, salīdzinot ar tiešo diskrēto veivletu transformāciju, t.i., mērogos $J \geq j \geq 1$, kur J – mērogu skaits.



2.6. att. Reāllaika inversās diskrētās veivletu transformācijas filtru bankas realizācijas algoritms.

2.6. att. ir parādīts inversās diskrētās veivletu transformācijas algoritms ar filtru bankas pielietošanu. 2.6. att. parādītā algoritma realizāciju *C++* valodā satur 3.pielikums.. Atzīmēsim, ka ieejas dati filtriem atšķiras, konkrētāk, aproksimācijas filtram ar pārvades funkciju $F_0(z)$ ieejas dati ir aproksimācijas koeficienti $a[k]$, savukārt detalizācijas filtram ar pārvades funkciju $F_1(z)$ ieejas dati ir detalizācijas koeficienti $d[k]$. Tas nozīmē, ka atšķirībā no tiešas transformācijas algoritma, šeit katram filtram būs jānodrošina savs stāvoklis $MEM_0[n]$ un $MEM_1[n]$, $n = 0, 1, 2, ..., N$, katrs no tiem ar garumu L, filtru impulsa reakcijas garumu.

Tālāk, līdzīgi kā tas bija tiešās diskrētās veivletu transformācijas gadījumā, tiek aprēķinātas divas diskrētas konvolūcijas divām nolases y_k sastāvdaļām:

$$y_0 = f_0[0]a_k + \sum_{n=1}^{N} MEM_0[n-1]f_0[n] \tag{2.8}$$

un

$$y_1 = f_1[0]d_k + \sum_{n=1}^{N} MEM_1[n-1]f_1[n]. \tag{2.9}$$

Abas diskrētās konvolūcijas tiek izskaitļotas vienā ciklā, turklāt šajā pašā ciklā notiek arī filtra stāvokļa pārbīde. Funkcijas izpildes rezultātā tiek izskaitļotas divas daļas: aproksimācijas daļa y_0 un detalizācijas daļa y_1, kuras tiek apvienotas nolases vērtībā summas veidā: $y_k = y_0 + y_1$.

Jāpiezīmē, ka diskrētizācijas frekvences palielināšanu 2 reizēs ir jāveic atsevišķi pirms šādas filtru bankas izmantošanas, lai kompensētu decimācijas operāciju, kas veikta tiešās diskrētās veivletu transformācijas laikā. Pēc būtības, šādu diskretizācijas frekvences palielināšanu divās reizēs veic iestarpinot "0" vērtības starp aproksimācijas koeficientu a_k un detalizācijas koeficientu d_k vērtībām: $a_1, 0, a_2, 0, ...$ un $d_1, 0, d_2, 0, ...$

Kā tiks parādīts 2.5.sadaļā, tieši sintēzes laikā ir jāievēro laika aiztures, kas veidojās filtrācijas dēļ (nerekursīvie ciparu filtri satur laika aiztures līniju). Taču, neskatoties uz to, ka varētu sagaidīt, ka šādu laika aizturi ir jāievēro gan analīzes filtru bankā, gan sintēzes filtru bankā (jo abi

filtri ievieš laika aizturi N), īstenībā šo aiztures vērtību ir jāievēro tikai vienreiz "tiešā-inversā" transformāciju pārī, lai veiksmīgi rekonstruētu sākotnējo signālu. Daudzmērogu transformācijas gadījumā ir jāievēro arī decimācijas operāciju, kas, pēc būtības, divkāršos šādu aizturi katrā nākamajā transformācijas mērogā (līmenī). Sīkāk par laika aizturi un tās noteikšanu jebkuram mērogam aprakstīts 2.5.sadaļā.

2.4. Reāllaika tiešā J mērogu diskrētā veivletu transformācija

2.3.sadaļā tika aprakstīts reāllaika analīzes filtru bankas algoritms viena mēroga diskrētās veivletu transformācijas aprēķinam. Šajā sadaļā algoritms tiks pielietots daudzmērogu diskrētās veivletu transformācijas aprēķinam reāllaikā – t.i., uz katru ienākošo signāla (procesa) nolasi tiek aprēķināti aproksimācijas koeficienti $\{a_{j,k}\}$ un detalizācijas koeficienti $\{d_{j,k}\}$ visos mērogos[5] $j = 1, 2, ..., J$[6].

Šeit būtu jāatzīmē, ka K signāla x_k nolašu ideālajai rekonstruēšanai (sintēzei) pēc J mērogu tiešās diskrētās veivletu transformācijas ir nepieciešami šādi koeficienti:

$\{d_{1,1}, d_{1,2}, ..., d_{1,K/2}\}$	$\{d_{2,1}, ..., d_{2,K/4}\}$	...	$\{d_{J,1}, ..., d_{J,K/2^J}\}$	$\{a_{J,1}, ..., a_{J,K/2^J}\}$,

kur $a_{j,k}$ un $d_{j,k}$ – aproksimācijas un, attiecīgi, detalizācijas koeficienti ar numuru k diskrētās veivletu transformācijas mērogam j. Tas nozīmē, ka veiksmīgajai signāla nolašu x_k rekonstruēšanai ir nepieciešami visi detalizācijas koeficienti $d_{j,k}$ visos mērogos un pēdējā mēroga aproksimācijas koeficienti $a_{J,k}$. Pārējo mērogu $j = 1, 2, ..., J-1$ aproksimācijas koeficienti $a_{j,k}$ rekonstruēšanai nav nepieciešami, taču apskatāmajā algoritmā tiks saglabāti pārbaudes nolūkos[7] Pēc veiktās pārbaudes šo saglabāšanu var neveikt, lai panāktu lielāku algoritma ātrdarbību.

2.7. att. ir parādīts J mērogu reāllaika tiešās diskrētās veivletu transformācijas realizācijas algoritms, kas izveidots izmantojot 2.4. att. parādītā tipa filtru bankas. 4.pielikums. satur pārbaudes programmu, kas izveidota saskaņā ar 2.7. att. parādīto filtru bankas algoritmu. Algoritmam ir jāuzdod analīzes aproksimācijas filtra impulsa reakcijas koeficientus $h_0[n]$ un analīzes detalizācijas filtra impulsa reakcijas koeficientus $h_1[n]$, $n = 0, 1, 2, ..., N$. Bez tam, jānorāda vēlamais[8] mērogu skaits J. Šis lielums ir augšējais ierobežojums, t.i., ja nolašu skaits ir pietiekamais vēl lielāku mērogu aprēķinam, algoritms ierobežosies tikai ar uzdoto mērogu skaitu J.

Šis mērogu skaits J sakrīt ar izmantojamo 2.4. att. filtru banku skaitu. Turklāt visu šo filtru banku pārvades funkcijas $H_0(z)$ un $H_1(z)$ būs attiecīgi vienādas un katru tādu filtru banku realizē 2.3. att. parādītais algoritms. Jāatzīmē, ka katrai tādai filtru bankai filtru stāvokli (aiztures līnija) ir jāglabā atsevišķi, tāpēc, pēc būtības ir jāglabā J filtru stāvokļi vienā stāvokļu matricā $MEM[j][n]$ ar dimensijām $J \times L$, kur N – filtru kārta. Katrā tādas $MEM[j][n]$ matricas rindā ir filtra stāvoklis transformācijas mērogam ar numuru j. Pirms algoritma izpildes uzsākšanas (t.i., līdz pirmās nolases x_0 saņemšanai) visi filtru stāvokļi tiek iestādīti nullēs. Bez tam, tiek nullēm tiek pielīdzināti arī aproksimācijas un detalizācijas koeficientu indeksi $k(j)$ katrā mērogā, kur $j = 0, 1, ..., J-1$.

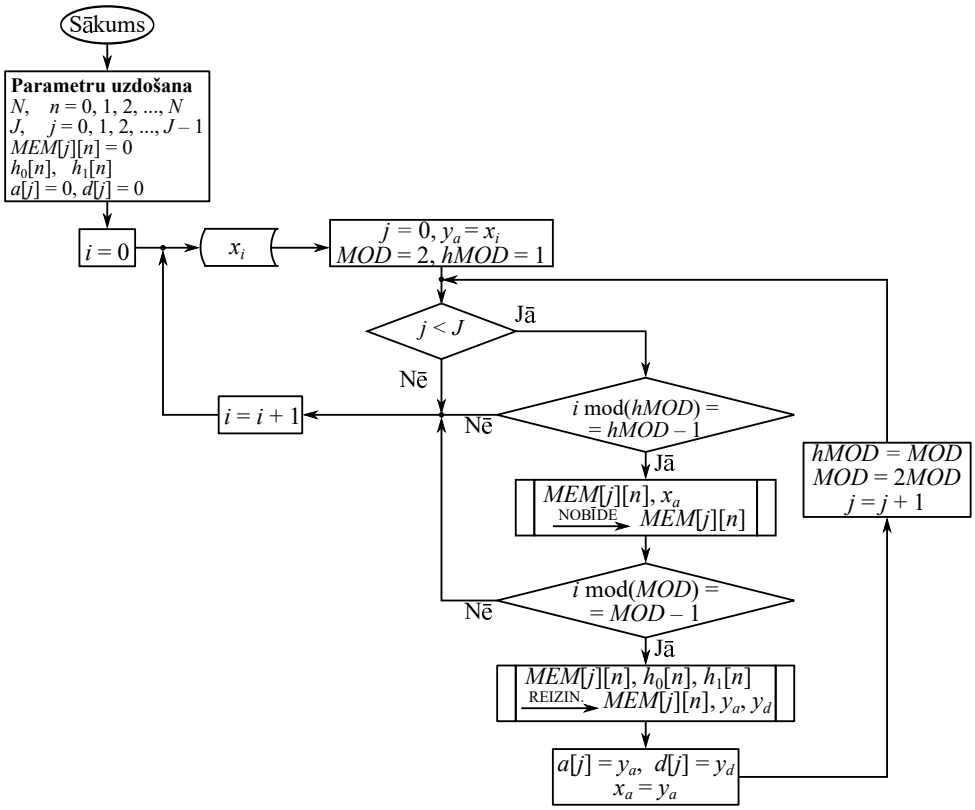
Tālāk, sākot ar laika momentu $i = 0$, pirmās filtru bankas ieejā nonāk signāla (procesa) nolases x_i, kurām ir jāveic J mērogu diskrētā veivletu transformācija. Ievērojot to, ka sākot ar

[5] Uzreiz atzīmēsim, ka tiešās diskrētās veivletu transformācijas gadījumā gan mērogs, gan līmeņa numurs pieaug, savukārt inversās transformācijas gadījumā – līmeņa numurs pieaug, mērogs – samazinās

[6] Pie nosacījuma, ka attiecīgā j mēroga koeficientu aprēķinam tika apstrādāts nepieciešamais ieejas signāla x_k nolašu skaits.

[7] Veicot J mēroga inverso veivletu transformāciju, rezultātā ir jāaprēķina iepriekšējā, t.i., mazākā, $J-1$ mēroga aproksimācijas koeficienti. Ja tādi būtu saglabāti, tas palīdzētu pārbaudīt transformācijas aprēķina pareizību.

[8] Jāatzīmē, ka šādu mērogu skaitu nevarēs aprēķināt pēc pirmajām nolasēm, tāpēc algoritms izpildīsies tikai tādiem mērogiem, kuros tas ir iespējams, un to skaitu noteiks pamatojoties uz jau apstrādāto nolašu skaitu.



2.7. att. J mērogu reāllaika tiešās diskrētās veivletu transformācijas realizācijas algoritms ar filtru bankām.

$j = 2^9$ filtru banku, tās ieejā tiek padoti iepriekšējā mēroga aproksimācijas koeficienti $a_{j-1,k}$, būtu ērti uzskatīt signālu pirms diskrētās veivletu transformācijas par $j = 0$ mēroga aproksimācijas koeficientiem $a_{0,i} = x_i$, kas, kopumā, atbilst diskrētās veivletu transformācijas būtībai filtru banku gadījumā, kad katrā nākamajā mērogā rupjajos aproksimācijas koeficientos $a_{j,k}$ tiek izdalīti vēl rupjāki aproksimācijas koeficienti $a_{j+1,k}$, kas apraksta signāla zemfrekvenču sastāvdaļas, un precīzākie detalizācijas koeficienti $d_{j+1,k}$, kuri, savukārt, apraksta signāla augstfrekvenču sastāvdaļas. Algoritmā šīs vērtības tiek saglabātas laicīgajos mainīgajos y_a un y_d, attiecīgi, un turpmāk y_a vērtība paliek arvien "rupjāka" katrā nākamajā mērogā. Līdz ar to, sākotnēji $y_a = x_i$.

Tālāk, algoritms iziet visiem mērogiem j līdz norādītajam maksimālajam J, ja tas ir iespējams tekošās nolases numuram, katrā mērogā tiek veikta decimācijas operācija par 2^{j-1} nolasēm pirms filtrēšanas[10], pēc kuras paliek tikai nolases ar numuriem, kas ir kārtni lielumam 2^{j-1}. Šīs nolases tiek padotas filtra ieejā ar turpmāko decimāciju par 2 nolasēm (paliek tikai nepārskaitļa numura nolases). Ja ir nepieciešams, tiek saglabāti aproksimācijas un detalizācijas koeficienti. Savukārt, ja x_i nolases numurs i neatbilst izvirzītajai prasībai, tāda nolase netiek apstrādāta šajā mērogā, kā arī nevienā lielākajā mērogā.

Saskaņā ar 3.1. att. parādīto filtru banku viena mēroga diskrētajai veivletu transformācijai, pēc katras filtrēšanas operācijas diskretizācijas frekvence tiek samazināta 2 reizēs, t.i., tiek saglabāta tikai katrā otrā signāla nolase, turklāt jaunā signāla ilgums pēc laika sakrīt ar vecā signāla ilgumu. Tas nozīmē, ka tajā pašā laika intervālā būs divreiz mazāks signāla nolašu skaits, kas atbilst divreiz zemākajai diskretizācijas frekvencei. Ievērojot to, ka katrā nākamajā mērogā tiek izmantots mazākā mēroga izejas signāls – aproksimācijas koeficienti, nav grūti konstatēt,

[9] Šeit vismazākajam, t.i., pirmajam, mērogam atbilst $j = 1$.

[10] Acīmredzami, vienīgais izņēmums ir mērogs $j = 1$, kurā apstrādā katru nolasi bez izlaišanas.

ka $j = 1$ mērogā tiek saglabāta katrā 2^1 nolase[11], $j = 2$ mērogā – katrā 2^2 nolase (tas pats, ka katrā 2. nolase no $j = 1$ mēroga rezultāta), $j = 3$ mērogā – katrā 2^3 nolase, utt.

Ja sekot līdzi tekošās nolases kārtas numuram i, ir iespējams uzreiz droši noteikt, cik diskrētas veivletu transformācijas mērogos ir iespējams aprēķināt aproksimācijas un detalizācijas koeficientus. Šeit ir svarīgi atzīmēt – kaut arī ne visas nolases saglabājās koeficientu veidā, tās ietekmē filtru bankas stāvokli. Decimācijas procedūra tiek veikta filtra izejā, tas nozīmē ka šādas nolases joprojām ir jārēķina. Tā ir ļoti neefektīvā pieeja, par cik ir iespējams samazināt operāciju skaitu divās reizēs (un palielināt veiktspēju). Veidojamajā algoritmā, līdz ar to tiek veikti uzlabojumi, kas dod iespēju samazināt reizinājumu skaitu divās reizēs. Bīdes operāciju skaits netiek mainīts šajā realizācijā.

Tās nolases, kas tiek noņemtas j mēroga filtru izejā vairs netiks padotas uz nākamā, $j + 1$ mēroga, filtru ieejām, t.i., vienkārši nebūs ieejas datu $j + 1$ mērogā. Taču, joprojām, tieši j mērogā ir jāveic aprēķins, par cik šī nolase x_i ietekmēs nākamās N filtra izejas vērtības (atmiņas efekts). Turklāt atzīmēsim, ka reāli apstrādāts tiek divreiz lielāks nolašu skaits, nekā tiek saglabāts (bet, kā tika minēts, ir iespējams samazināt reizinājumu skaitu). Apkopojot visu augstāk minēto, var veikt vispārinājumu:

- mērogā ar numuru $j = 1$ tiek apstrādāta katrā $\Delta_1 = 1$ nolase, bet saglabāts katrs $\Delta_2 = 2$ koeficientu pāris;
- mērogā ar numuru $j = 2$ tiek apstrādāta katrā $\Delta_1 = 2$ nolase, bet saglabāts katrs $\Delta_2 = 4$ koeficientu pāris;
- mērogā ar numuru $j = 3$ tiek apstrādāta katrā $\Delta_1 = 4$ nolase, bet saglabāts katrs $\Delta_2 = 8$ koeficientu pāris;
- mērogā ar numuru $j = J$ tiek apstrādāta katrā $\Delta_1 = 2^{J-1}$ nolase, bet saglabāts katrs $\Delta_2 = 2^J$ koeficientu pāris.

Tādā veidā, tekošās nolases x_i kārtas numurs i satur pilno informāciju par to, kādos mērogos ir jāveic aprēķins, un kādos – jāsaglabā koeficienti. 2.7. att. algoritmā šīs nolases numurs tiek uzglabāts mainīgajā i un tiek palielināts $i = i + 1$ ar kārtējo ieejas nolasi x_i.

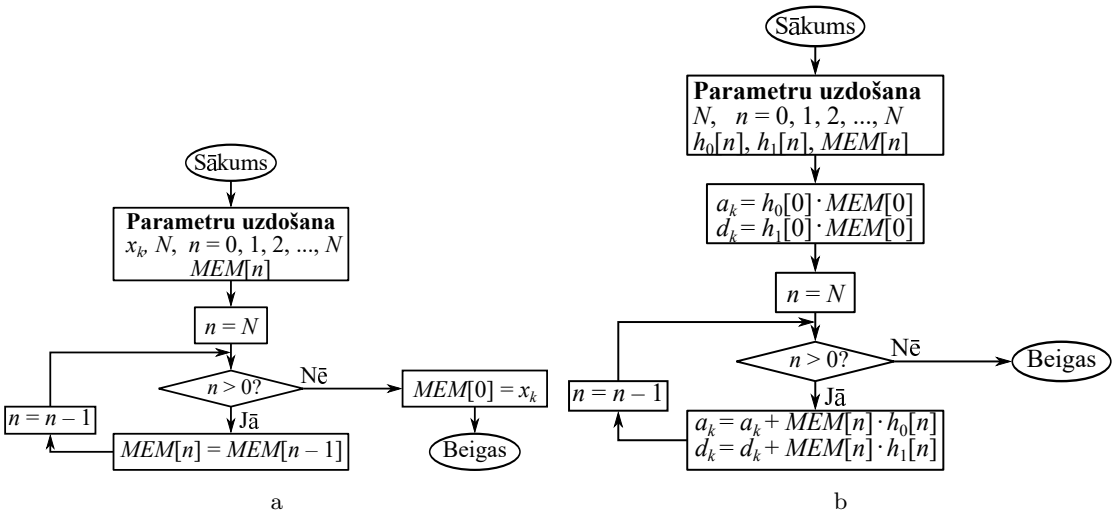
Līdz ar to, katrā mērogā $j = 1, 2, ..., J$ notiek pārbaude pēc i lieluma, vai ir jāveic filtrēšanu šajā mērogā. Pārbaude notiek, aprēķinot nolases numura i dalījuma atlikumu attiecībā pret $hMOD = 2^{j-1}$ saskaņā ar:

$$R_1 = i \bmod(hMOD), \qquad i = 0, 1, 2, ..., K, \tag{2.10}$$

kur K – ieejas signāla nolašu skaits.

Ja atlikums R_1 sasniedz savu maksimālo vērtību $\max(R_1) = hMOD-1$, nolase tiek apstrādāta pēc 2.3. att. parādītā algoritma. Pretējā gadījumā tā tiek atmesta pēc decimācijas mazākajos mērogos un filtru bankas ieejā nekas netiks padots. Tas nozīmē, ka filtru bankas izejā arī nebūs jaunās nolases, tātad, pārbaudīt lielākus mērogus j vairs nav nepieciešamības. Šajā situācijā tekošās nolases x_i apstrāde beidzās un tiek sagaidīta nākošā nolase x_{i+1}. Acīmredzami, ka mērogā $j = 1$ $hMOD = 1$ un nosacījums (2.10) vienmēr izpildās, par cik $\max(R_1) = hMOD - 1 = 0$. Tādā veidā, mērogā $j = 1$ apstrāde notiek vienmēr. Šeit būtu jāpiebilst, kaut arī apstrāde notiek – koeficienti tiek saglābāti tikai reizi uz divām nolasēm. Tas nozīmē, ka var veikt tikai nolašu bīdi aiztures līnijā un nerēķināt y_a un y_d vērtības tādai nolasei. Tādā gadījumā, būtu ērti sadalīt 2.3. att. parādīto filtrācijas algoritmu divās daļās, kā tas ir parādīts 2.8. att. – nobīdes veikšana 2.8a. att. un konvolūcijas aprēķināšana 2.8b. att., kas dod filtra izejas signālu – reakciju. Abas filtru banka algoritma daļas tika realizētas *C++* valodā. Šo funkciju programmas koda tekstus satur 3.pielikums.

[11] Jāatzīmē, ka nolašu numerācija sākās ar $i = 0$.



2.8. att. Diskrētās veivletu transformācijas filtru bankas funkciju sadalīšana: a) aiztures līnijas bīde, b) filtru reakcijas aprēķins.

Pēc nobīdes apakšprogrammas izpildes saskaņā ar 2.8a. att. algoritmu, katrai otrajai nolasei (ar nepārskaitļa numuru) tiek aprēķināta filtra reakcija pēc 2.8b. att. algoritma. Kā tika minēts iepriekš, vērtības tiek saglabātas koeficientu veidā 2 reizēs retāk, nekā notiek apstrāde attiecīgajā mērogā j. Tas nozīmē ka jāaprēķina atlikums attiecībā uz 2 reizēs lielāko dalītāju saskaņā ar:

$$R_2 = i \bmod(MOD), \qquad i = 0, 1, 2, ..., K, \tag{2.11}$$

kur K – ieejas signāla nolašu skaits.

Ja (2.11) pārbaudes rezultātā sanāk maksimālais atlikums $\max(R_2) = MOD - 1$, filtru reakcijas tiek aprēķinātas saskaņā ar 2.8b. att. algoritmu un vērtības y_a un y_d tiek attiecīgi saglabātas j mēroga koeficientos $a_{j,k} = a[j]$ un $d_{j,k} = d[j]$. Pretējā gadījumā filtru izejas reakcijas var nerēķināt, tādā veidā paātrinot apstrādes ātrdarbību.

Par cik šī pati aproksimācijas koeficienta vērtība $a[j]$ tiks izmantota nākamajā $j+1$ mērogā kā ieejas dati filtru bankām, loģiski būtu secināt, ka gadījumā kad $R_2 < MOD - 1$, t.i., koeficienti netiek saglabāti, filtru ieejās netiks padoti dati un nolases x_i apstrādi var pabeigt. Savukārt, pretējā gadījumā var garantēti veikt diskrēto veivletu transformāciju vismaz vēl vienā mērogā (vai vairākos, atkarībā no konkrētās i vērtības).

2.5. Filtru bankas ieviestās laika aiztures ievērošana

Veicot tiešo diskrēto veivletu transformāciju pēc 2.3. att. parādīta algoritma un uzreiz inverso – pēc 2.6. att. parādītā, var konstatēt, ka sintēzes filtru izejas vērtības (reakcijas) y_i veidojās ar noteikto laika aizturi attiecībā pret sākotnējo signālu x_i. Tālāk tas tiks parādīts uz konkrētā piemēra Dobeši-3 veivletu gadījumā vairākiem mērogiem $j = 1, 2, 3$.

Dobeši-3 diskrētās veivletu transformācijas analīzes aproksimācijas (ZF) filtram ir šādi impulsa reakcijas koeficienti, kas apkopoti 2.1. tabulā. Pārējo filtru impulsa reakcijas var aprēķināt pēc 2.1. tabulā uzdotām vērtībām:

- $H_1(z)$ saskaņā ar (2.3);
- $F_0(z)$ saskaņā ar (2.4);
- $F_1(z)$ saskaņā ar (2.5).

2.1. tabula. Dobeši-3 veivletu analīzes aproksimācijas filtra impulsa reakcijas koeficienti

$H_0[0]$	0,035226291882100656
$H_0[1]$	-0,085441273882241486
$H_0[2]$	-0,13501102001039084
$H_0[3]$	0,45987750211933132
$H_0[4]$	0,80689150931333875
$H_0[5]$	0,33267055295095688

Datu apstrādei tika izvēlēta šāda skaitļu secība:

$$x_i = \{255\ 224\ 192\ 159\ 127\ 95\ 3\ 32\}. \tag{2.12}$$

(2.12) dati tiek periodiski atkārtoti 5 reizes, tādā veidā iegūstot datu masīvu ar $K = 40$ nolasēm. Pielietojot 2.7. att. parādīto diskrētās veivletu transformācijas algoritmu[12] gadījumam ar maksimālo mērogu skaitu $J = 3$. Transformācijas aprēķina rezultātā tika atrasti aproksimācijas koeficienti $a_{j,k}$ un detalizācijas koeficienti $d_{j,k}$ visos mērogos. Šie rezultāti ir apkopoti 2.2. tabulā.

2.2. tabula. x_i signāla $J = 3$ mērogu diskrētās veivletu transformācijas aproksimācijas un detalizācijas koeficienti.

Mērogs $j = 1$		**Mērogs** $j = 2$		**Mērogs** $j = 3$	
Approks. koef. $a_{1,k}$	**Detal. koef.** $d_{1,k}$	**Approks. koef.** $a_{2,k}$	**Detal. koef.** $d_{2,k}$	**Approks. koef.** $a_{3,k}$	**Detal. koef.** $d_{3,k}$
-13,8968	131,2391	3,8724	-36,5702	-1,6066	15,1728
76,2226	-35,4118	-36,2166	151,3064	3,5856	45,2649
324,8999	-0,0496	168,0252	-81,5177	-4,2537	-74,6679
233,4913	-0,2974	319,9727	141,8071	377,0817	-49,9959
129,6589	120,6080	253,5273	-72,4639	405,5257	-46,9839
123,0013	-30,4584	319,9727	141,8071		
324,8999	-0,0496	253,5273	-72,4639		
233,4913	-0,2974	319,9727	141,8071		
129,6589	120,6080	253,5273	-72,4639		
123,0013	-30,4584	319,9727	141,8071		
324,8999	-0,0496				
233,4913	-0,2974				
129,6589	120,6080				
123,0013	-30,4584				
324,8999	-0,0496				
233,4913	-0,2974				
129,6589	120,6080				
123,0013	-30,4584				
24,8999	-0,0496				
233,4913	-0,2974				

Kā var redzēt no 2.2. tabulas rezultātiem, apstrādājot $K = 40$ signāla x_i nolases, $j = 1$ mērogā tika izskaitļoti tieši $K/2 = 20$ aproksimācijas koeficienti $a_{1,k}$ un $K/2 = 20$ detalizācijas koeficienti $d_{1,k}$. Izmantojot koeficientu kopu $\{a_{j,k}\}$ kā ieejas signālu $j = 2$ mēroga transformācijā,

[12] Jāatzīmē, kaut arī algoritms tika veidots ar nolūku aprēķināt diskrēto veivletu transformāciju reāllaikā, nav nekādu ierobežojumu pielietot to pēcapstrādes režīmā. Protams, top jautājums par skaitļošanas efektivitāti, salīdzinājumā ar eksistējošajiem algoritmiem, bet tāds salīdzinājums tiks apskatīts turpmāk.

tika iegūti $K/2^2 = 10$ aproksimācijas koeficienti $a_{2,k}$ un $K/2^2 = 10$ detalizācijas koeficienti $d_{2,k}$. Izpildot transformāciju pēdējā, $j = 3$ mērogā, abu koeficientu skaits sastāda $K/2^3 = 5$.

Tagad, izmantojot $j = 3$ mēroga koeficientu kopas $a_{3,k}$ un $d_{3,k}$ kā ieejas datus 2.5. att. parādītajam algoritmam[13], ar kuriem tiek veikta inversā diskrētā veivletu transformācija $j = 3$ mērogā. Rezultātā tika iegūta šāda skaitļu secība:

$$y_{3,i} = \{0\ 0\ 0\ 0\ 3,8724\ -36,2166\ 168,0252\ 319,9727\ 253,5273\ 319,9727\}. \tag{2.13}$$

Salīdzinot iegūtos rezultātus (2.13) ar 2.2. tabulas datiem, nav grūti pamanīt, ka $y_{3,i}$ secība sākās ar 4 nullēm, bet pārējās vērtības atbilst augstākā mēroga aproksimācijas koeficientiem no 2.2. tabulas, t.i., $a_{2,k}$ mērogā $j = 2$. Turklāt var skaidri redzēt, ka koeficienti no tabulas iet ar nobīdi 4, uzreiz pēc nullēm. Šīs nulles veidojās filtra darbības uzsākšanas laikā, bet konkrētais skaitlis 4 – nav nejaušība, jo filtram ir N aiztures elementi (kur N ir filtra kārta), tad aiztures vērtība inversai $j = 3$ mēroga diskrētajai veivletu transformācijai sastādīs $N - 1$ nolases[14].

Šeit ir jāatzīmē: lai veiksmīgi veiktu sākotnējā signāla x_i (2.12) rekonstruēšanu inversās transformācijas $j = 1$ mērogā, būtu jāievēro šī laika aizture tādā veidā, lai aizkavētie aproksimācijas koeficienti $a_{j,k}$ atbilstu attiecīgajiem detalizācijas koeficientiem $d_{j,k}$. Ja piemērā augstāk izmantot $y_{3,i}$ kā $a_{2,k}$, savukārt attiecīgos detalizācijas koeficientus $d_{2,k}$ paņemt no 2.2. tabulas bez to savstarpējās nobīdes ievērošanas un pielietotu inversās diskrētās veivletu transformācijas algoritmu saskaņā ar 2.5. att., vienlaikus veicot "0" iestarpināšanu pēc katra koeficienta $a_{2,k}$ un $d_{2,k}$, tiktu iegūts šāds rezultāts:

$$\begin{aligned} y_{2,i} =& \{-1,2882\ -3,1246\ 10,2674\ 29,7456\ -52,8078\ -64,3816\ 138,0890\ -0,7309 \\ & -86,1858\ -41,1622\ 118,9344\ -31,4800\ -41,2577\ 93,3054\ 316,0126\ 232,4878 \\ & 136,9644\ 119,9894\ 324,8999\ 233,4913\}. \end{aligned} \tag{2.14}$$

Kā var redzēt pēc (2.14) iegūtais rezultāts nav līdzīgs aproksimācijas koeficientu $a_{1,k}$ vērtībām no 2.2. tabulas datiem, kā to varētu sagaidīt. Ja papildināt detalizācijas koeficientus $d_{2,k}$ ar $N - 1 = 4$ nullēm pirms pirmās nolases un aprēķināt inverso diskrēto transformāciju $j = 2$ mēroga vēlreiz pēc tāda paša algoritma, tad transformācijas rezultāts būs šāds:

$$\begin{aligned} y_{2,i} =& \{0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ -13,8968\ 76,2226\ 324,8999\ 233,4913\ 129,6589 \\ & 123,0013\ 324,8999\ 233,4913\}. \end{aligned} \tag{2.15}$$

Acīmredzams, ka tādā situācijā pēdējie $N-1$ detalizācijas koeficienti $d_{2,k}$ netiek izmantoti[15], toties rezultātā patiešām sanāca mazākā mēroga aproksimācijas koeficienti $a_{1,k}$. Līdzīgi kā tas bija $j = 3$ mērogā, arī šajā gadījumā pirms aproksimācijas koeficientiem ir nuļļu secība ar garumu $M_2 = 12$, kuru var izteikt, zinot iepriekšējā mēroga aiztures lielumu $\Delta i_3 = 4$ saskaņā ar:

$$\Delta i_j = 2\Delta i_{j+1} + N - 1, \tag{2.16}$$

kur $N - 1$ ir laika aizture, ko ievieš viens sintēzes bankas filtrs ar kārtu N, savukārt Δi_{j+1} – iepriekšējā mēroga laika aizture, kas tiek sareizināta ar 2, jo pirms filtrēšanas procedūras ir

[13] Algoritms veic tikai filtrēšanas procedūru, bez diskretizācijas frekvences palielināšanas 2 reizēs, ko ir jāveic pirms filtrēšanas. Attiecīga operācija ir ekvivalenta nulles nolašu pievienošanai pēc katras vērtības.

[14] Tādā gadījumā, varētu sagaidīt laika aizturi $N = 5$, nevis $N - 1 = 4$. Tālāk tekstā tiks izskaidrots, kāpēc sanāk tieši $N - 1$.

[15] Vismaz, līdz nākošās $a_{j,k+1}$ vērtības – apskatāmais algoritms ir paredzēts nepārtrauktajai darbībai reāllaikā. Līdz ar to, šie neizmantojamie koeficienti ir jāuzglabā *FIFO* tipa buferatmiņā.

jāpalielina diskretizācijas frekvence 2 reizēs, pievienojot pēc katras nolases "0" vērtību. Par cik šajā situācijā pirmās Δi_{j+1} nolases arī ir nulles, tad rezultātā aiztures laiks divkāršojās.

Šo pašu aiztures laiku ir iespējams izteikt neatkarīgi, izmantojot zināšanas tikai par interesējošā transformācijas mēroga numuru j saskaņā ar:

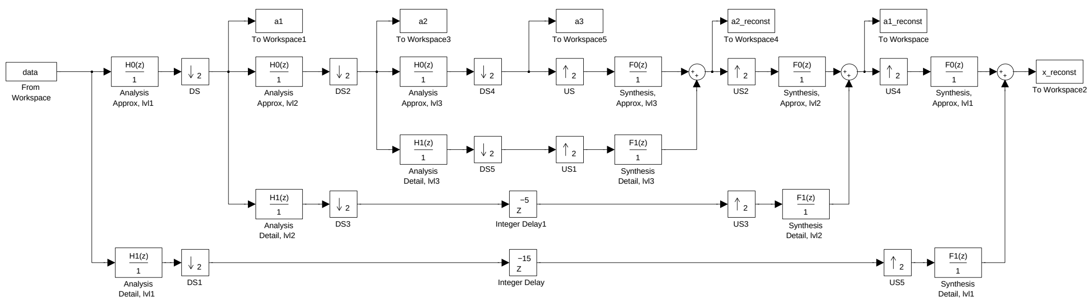
$$\Delta i_j = (N-1)\sum_{n=1}^{j} 2^{n-1}. \tag{2.17}$$

Tomēr, skaitliski izdevīgāk ir pielietot (2.16) izteiksmi, jo veicot j mērogu inverso diskrēto veivletu transformāciju nav iespējams aprēķināt tikai rezultātu mērogā j, bet ir jāveic aprēķins visos mērogos līdz tam, sākot ar pirmo, t.i., vislielāko.

Ievērojot visu iepriekš minēto, aprēķināsim inverso diskrēto veivletu transformāciju pēdējā $j = 1$ mērogā, kuras rezultātā ir jāatgriežas pie sākotnējā signāla nolasēm x_i (2.12), kas būtu aizkavētas pēc laika par $\Delta i_1 = 28$ nolasēm (saskaņā ar (2.16)):

$$\begin{aligned} y_{1,i} = \{&0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\\ &255\ 224\ 192\ 159\ 127\ 95\ 63\ 32\ 255\ 224\ 192\ 159\}. \end{aligned} \tag{2.18}$$

Kā var redzēt, rezultāts ir pareizs. Nobeigumam, tiks apskatīta šāda pati $J = 3$ mērogu tiešās-inversās diskrētās transformācijas pāris, kas tika realizēts ar filtru bankām *Simulink* modelī (skat. 2.9. att.). Modeļa ieejas dati tiek paņemti no *Matlab* mainīgo vektora, kas satur x_i nolases (2.12). Pēc katra aproksimācijas filtra (analīzes filtru bankā – pēc attiecīgās decimācijas procedūras, sintēzes filtru bankā – pēc summatora) tiek saglabāts aproksimācijas koeficientu $a_{j,k}$ vektors. Visas šo vektoru vērtības ir apkopotas 2.3. tabulā.



2.9. att. $J = 3$ mērogu tiešās un inversās diskrētās veivletu transformācijas *Simulink* modelis.

Salīdzinot savā starpā aproksimācijas koeficientus 2.1.-2.3. tabulās, var konstatēt, ka koeficienti sanāca pilnīgi atšķirīgi. Tomēr, salīdzinot aproksimācijas koeficientus $a_{j,k}$ analīzes filtra izejā ar rekonstruētajiem koeficientiem $a\_\text{reconst}_{j,k}$ sintēzes filtru bankas izejā, var secināt ka rekonstruēšana tiek veikta pareizi. Par to liecina arī x_reconst vektora nolases:

$$\begin{aligned} x\_\text{reconst} = \{&0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\\ &0\ 0\ 0\ 0\ 0\ 0\ 0\ 255\ 224\ 192\ 159\ 127\}. \end{aligned} \tag{2.19}$$

Attiecīgās atšķirības aproksimācijas koeficientu vērtībās (kā arī detalizācijas koeficientu vērtībās, kuras šeit nav parādītas) ir izskaidrojamās ar to, ka decimācijas operācija *Simulink* modeļa analīzes filtru bankās atšķiras no šīs pašas operācijas 2.7. att. parādītajā algoritmā, kurš tika pielietots piemērā augstāk. *Simulink* modelī decimācijas operācija saglabā tikai nolases ar pārskaitļa kārtas numuriem $i = 0, 2, 4, ..., K-2$, savukārt 2.7. att. piedāvātajā algoritmā tiek saglabātas tikai nolases ar nepārskaitļa kārtas numuriem $i = 1, 3, 5, 7, ..., K-1$ (abos gadījumos numerācija sākās ar $i = 0$, tiek pieņemts, ka kopējais nolašu skaits K ir pārskaitlis).

2.3. tabula. $J = 3$ mērogu tiešās-inversās diskrētās veivletu transformācijas aproksimācijas koeficienti.

$a1$	$a2$	$a3$	$a2$_reconst	$a1$_reconst
8,9827	0,3164	0,0111	0	0
-46,8032	12,4289	-3,5065	0	0
273,7364	-68,1829	-3,8306	0	0
279,5178	312,99480	40,9075	0	0
196,8011	219,5952	391,9162	0	0
50,3507	353,9048		0,3164	0
284,3818	219,5952		12,4289	0
279,5178	353,9048		-68,1829	0
196,8011	219,5952		312,9948	0
50,3507	353,9048		219,5952	0
284,3818				0
279,5178				0
96,8011				0
50,3507				0
284,3818				0
279,5178				8,9827
196,8011				-46,8032
50,3507				273,7364
284,3818				279,5178
279,5178				196,8011

Analizējot sīkāk 2.3. tabulas rezultātus, var pamanīt, ka šāda decimācijas operācijas atšķirība ietekmē ne tikai koeficientu vērtības, bet arī laika aizturi. Salīdzinot (2.18) ar (2.19), var konstatēt, ka *Simulink* modelī laika aizture līdz pirmajām signāla nolašu vērtībām ir lielāka par 7 nolasēm un sastāda $M_3 = 35$ nolases. Arī rekonstruētajiem aproksimācijas koeficientiem ir novērojamas atšķirīgas laika aiztures. Apskatot $J = 3$ mērogu transformāciju, var nonākt pie secinājuma:

- laika aizture līdz pirmā koeficienta aprēķina sintēzes filtru bankai sastāda N, ja analīzes filtru bankā decimācijas rezultātā tika saglabātas nolases ar pārskaitļa kārtas numuriem (skaitot no $i = 0$);
- laika aizture līdz pirmā koeficienta aprēķina sintēzes filtru bankai sastāda $N-1$, ja analīzes filtru bankā decimācijas rezultātā tika saglabātas nolases ar nepārskaitļa kārtas numuriem (skaitot no $i = 0$).

Tādā veidā, analoģiski (2.16) un (2.17) var noteikt laika aizturi jebkura j mēroga transformācijai:

$$\Delta i_j = 2\Delta_{j+1} + N, \tag{2.20}$$

vai, attiecīgi pilnajā veidā:

$$\Delta i_j = N \sum_{n=1}^{j} 2^{n-1}. \tag{2.21}$$

Ievērojot to, ka transformācijas procedūra un attiecīgas filtru bankas šajos gadījumos neatšķiras, bet visa atšķirība ir decimācijas operācijas izpildes kārtībā, var secināt ka ir racionālāk veikt decimāciju, saglabājot nolases ar nepārskaitļa kārtas numuriem, jo tas dod iespēju ātrāk aprēķināt pirmās inversās diskrētās veivletu transformācijas nolases[16].

[16]Decimācijas operācijas izpildes kārtībai nav noteicošās ietekmes, bet koeficientu vērtības būs atšķirīgas.

2.6. Reāllaika inversā J mērogu diskrētā veivletu transformācija

2.3.sadaļā tika apskatīts reāllaika J mērogu tiešās diskrētās veivletu transformācijas algoritms, kurš apstrādāja katru ieejas nolasi x_i visos mērogos $j = 0, 1, ..., J-1$ (J ir maksimālais mērogs, numurēšana algoritmā sākās ar $j = 0$) un, atkarībā, no nolases numura i, izejā veidojās noteiktā mēroga j aproksimācijas koeficients $a[j]$, kā arī šī paša un visu mazāko mērogu detalizācijas koeficienti $d[0], d[2], ..., d[j]$. Šajā sadaļā tiks aprakstīts līdzīgais algoritms inversās diskrētās veivletu transformācijas algoritms reālajā laikā. Šāda algoritma ieejas dati ir:

- $j = J-1$ mēroga aproksimācijas koeficients $a_{J-1}[k]$ un detalizācijas koeficients $d_{J-1}[k]$, $k = 0$, bet $KK[J-1] = 1$ – koeficientu kopējais skaits mērogā $J-1$;
- $j = J-2$ mēroga detalizācijas koeficienti $d_{J-1}[k]$, $k = 0, 1$, bet $KK[J-2] = 2$ – koeficientu kopējais skaits mērogā $J-2$;
- katra iepriekšējā mērogā j koeficientu skaits divkāršojās, līdz mēroga $j = 1$ sasniegšanai, kurā jānodrošina 2^{L-1} detalizācijas koeficienti.

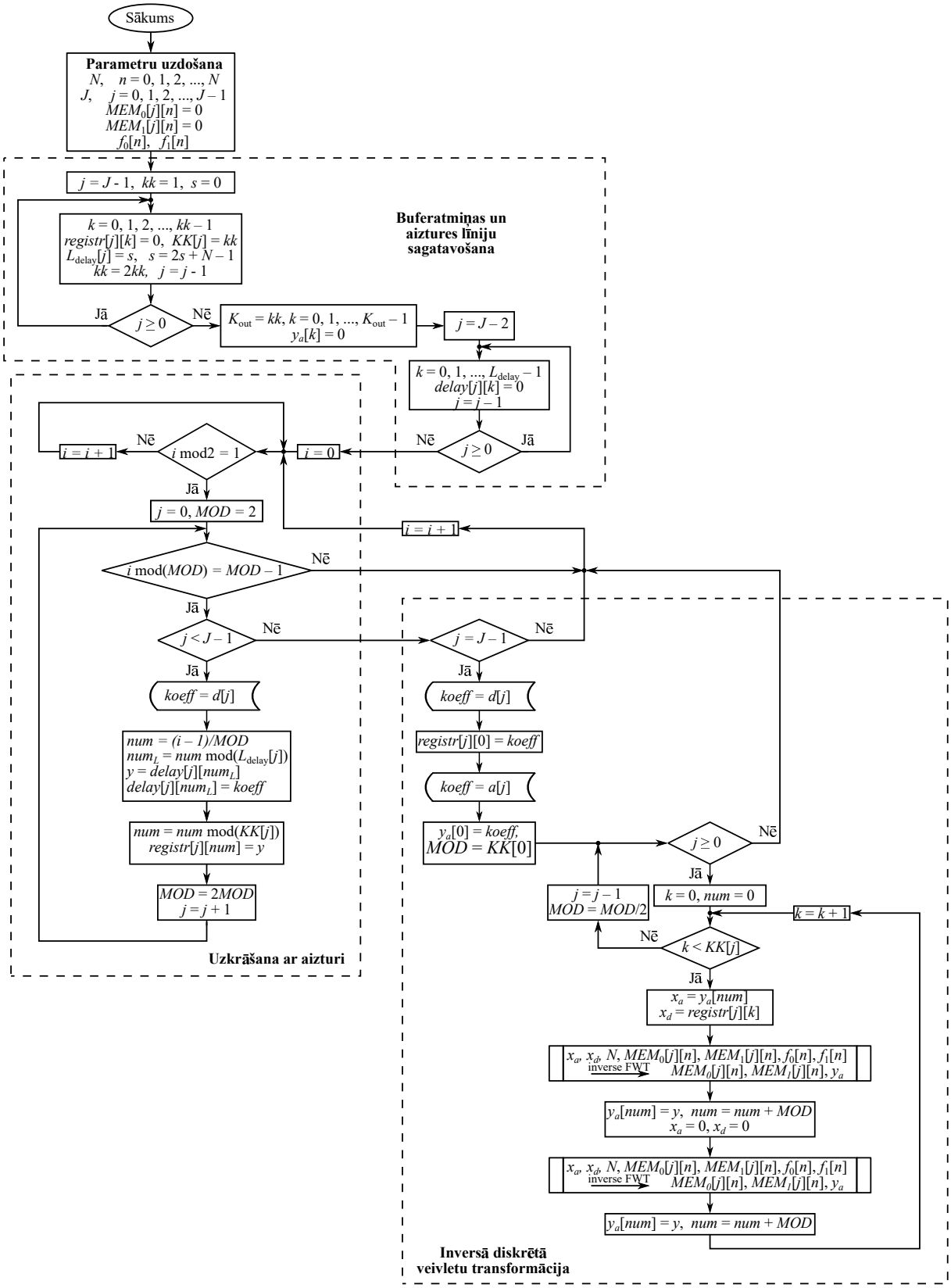
Vispārinot, lai varētu pilnīgi rekonstruēt signālu līdz mērogam $j = 0$, patvaļīgajā mērogā j ir jānodrošina $KK[j]$ detalizācijas koeficienti. Šo lielumu var novērtēt šādi:

$$KK[j] = 2^{J-j-1}, \tag{2.22}$$

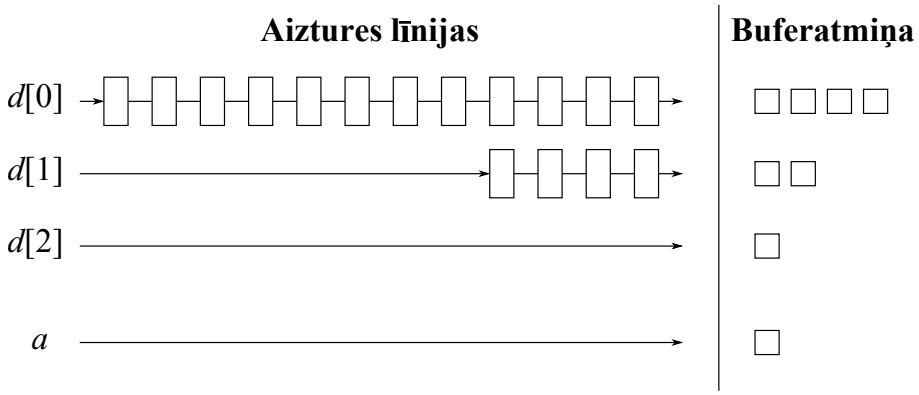
kur $j = 0, 1, ..., J-1$, bet J – diskrētās veivletu transformācijas maksimālais mērogu skaits.

Algoritma izpildes rezultātā tiek rekonstruētas $K_{\text{out}} = 2^J$ signāla nolases. Jāatzīmē, ka arvien augošais koeficientu skaits nesagādā problēmas, par cik šādi koeficientu skaitiem ir jābūt padotiem vienā un tā paša laika intervālā, ievērojot decimācijas operāciju, kas tika veikta tiešās diskrētās veivletu transformācijas laikā. T.i., visi detalizācijas koeficienti no mēroga $j = J-1$ ar 1 detalizācijas koeficientu līdz mērogam $j = 0$ ar 2^{J-1} detalizācijas koeficientiem, tiek aprēķināti vienas tiešās veivletu transformācijas iterācijas laikā, saskaņā ar algoritmu 2.7. att. Līdz ar to, inversās transformācijas laikā tie katrā taktī tiks uzkrāti, kamēr netiks saņemts vislielākā mēroga $J-1$ detalizācijas un aproksimācijas koeficients. Uz to brīdi pārējie koeficienti tiks uzkrāti un būs pieejami aprēķinu veikšanai. Vienīgais jautājums, kuru ir jāapskata šajā gadījumā būtu sistēmas veiktspējas jautājums: cik ātri sistēma veiks inverso diskrēto veivletu transformāciju? Tas ir lielajā mērā saistīts ar mērogu skaitu J – jo katrs tāds koeficientu pāris veidojās tieši vienu reizi uz 2^J nolasēm. No otrās puses, kamēr šis koeficientu pāris nav saņemts, nav iespējams sākt veikt inverso transformāciju, koeficientus no sākuma ir jāuzkrāj. Var noformulēt šādu secinājumu: jo lielāks ir mērogu skaits J, jo vairāk aprēķinu ir nepieciešams, bet vienlaicīgi arī retāk tādi aprēķini tiek veikti.

Reāllaika inversās veivletu transformācijas algoritms J mērogos ir parādīts 2.10. att. Šī algoritma pamatā ir sintēzes filtru bankas algoritms no 2.6. att., kas tika papildināts ar koeficientu uzkrāšanu, to laika aizturi (sīkāk par to aprakstīts 2.5.sadaļā) un interpolāciju ar nullēm.



2.10. att. J mērogu reāllaika inversās veivletu transformācijas realizācijas algoritms ar filtru bankām.



2.11. att. Atmiņas izdalīšana koeficientu buferim un laika aiztures ievērošanai J mērogu reāllaika inversās diskrētās veivletu transformācijas realizācijā.

Apkopojot, šo algoritmu var sadalīt šādos trīs posmos, katrs no kuriem ir iezīmēts 2.10. att.:

1. **Noskaņošana**:

   1.1. Analīzes filtru bankas aproksimācijas filtra impulsa reakcijas $h_0[n]$ uzdošana un pārējo filtru impulsa reakciju aprēķins saskaņā ar (2.3) - (2.5). Sintēzes filtru $F_0(z)$ un $F_1(z)$ aiztures līniju $MEM_0[j][n]$ un $MEM_1[j][n]$ inicializēšana un nonullēšana visos mērogos $j = 0, 1, ..., J-1$, kur J – maksimālais mērogs.

   1.2. Apstrādājamo koeficientu buferatmiņas $registr[j][KK[j]]$ apjoma $KK[j]$ noteikšana un buferatmiņas inicializēšana, nonullēšana. Koeficientu skaits mērogā j tiek noteikts saskaņā ar (2.22) un Dobeši-3 veivletu gadījumā buferatmiņai ir 2.11. att. parādītā struktūra.

   1.3. Aiztures līniju masīvu lielumu $L_{\text{delay}}[j]$ noteikšana, šo līniju $delay[j][n]$ inicializēšana un nonulēšana, $n = 0, 1, 2, ..., L_{\text{delay}}[j] - 1$ un $j = 0, 1, 2, ..., J-1$. Aiztures līnijas garums tiek novērtēts saskaņā ar (2.16) katram nakāmajām mērogam j, pie nosacījuma ka $L_{\text{delay}}[J-1] = 0$. Laika aizture var tikt veidota ar bīdes reģistru saskaņā ar 2.8a. att. parādīto algoritmu. Taču, ievērojot lielus aiztures līniju garumus šāda pieeja ir ļoti neefektīva, par cik ir jāveic pārāk daudz operāciju. Efektīvākais risinājums šajā gadījuma tiks piedāvāts zemāk. Pēc laika aiztures detalizācijas koeficienti tiek saglabāti buferatmiņā kā tas ir parādīts 2.11. att. struktūra.

   1.4. Izejas signāla $y_a[k]$ masīva inicializācija un nullēšana, $k = 0, 1, 2, ..., K_{\text{out}}$. Masīva apjoms $K_{\text{out}} = 2^J$, kur J ir mērogu skaits.

2. **Detalizācijas koeficientu nolasīšana un uzkrāšana:**

   2.1. Ievērojot to, ka pēc decimācijas veikšanas palika tikai nolases ar nepārskaitļa numuru $i \bmod 2 = 1$, nolasīšana tiek veikta tikai nepārskaitļa numuriem i. Ja numurs i ir pārskaitlis, tas tiek palielināts par vienu vienību.

   2.2. Ja nolases numurs ir nepārskaitlis, tiek veikta koeficientu nolasīšana. Koeficients ir jānolasa ja $i \bmod(MOD) = MOD - 1$, kur sākotnēji $MOD = 2$ un šī vērtība tiek divkāršota pēc katras izpildīšanas, kamēr netiek sasniegta mēroga vērtība $j = J - 1$.

   2.3. Tādā veidā, algoritms nosaka automātiski pēc nolases numura i, cik mērogu koeficientus var nolasīt un saglabā tos aiztures līnijās, pie kuru izejām ir pieslēgta koeficientu buferatmiņa, kā tas ir parādīts 2.11. att.

   2.4. punkti 2.1.-2.4. tiek atkārtoti, kamēr netiek aizpildīta visa koeficientu buferatmiņa pie $j = J - 1$, turklāt izpildās arī $i \bmod(MOD) = MOD - 1$, kur $MOD = 2^J$. Šajā gadījumā tiek veikta inversā diskrētā veivletu transformācija, pēc kuras programma atgriežas pie punkta 2.1.

3. **Inversās diskrētās veivletu transformācijas iterācija:**

   3.1. Buferatmiņā ir saglabāti detalizācijas koeficienti $d[j][k]$, $k = 0, 1, 2, ..., KK[j]$, visos mērogos, ieskaitot vislielākā $j = J - 1$ mēroga detalizācijas koeficientu $d[J-1][0]$. Bez tam, ir jānolasa arī $j = J - 1$ mēroga aproksimācijas koeficients $a[0]$, kurš tiek saglabāts izejas masīva elementā ar numuru $num = 0$: $y_a[0] = a[0]$.

   3.2. Tiek veikta inversā diskrētā veivletu transformācija saskaņā ar 2.6. att. parādīto filtrācijas algoritmu šādus ieejas datiem:

   - $x_d = registr[j][k]$ detalizācijas koeficientam, kur $k = 0, 1, 2, ..., KK[j] - 1$ ir koeficienta numurs;
   - $x_a = y_a[num]$ aproksimācijas koeficientam, kur num ir aproksimācijas koeficienta numurs, kas mainās ar soli $MOD = 2^j$, $j = 0, 1, 2, ..., J - 1$ – mērogs.

   3.3. Inversās diskrētās veivletu transformācijas rezultātā tiek aprēķināts mazākā $j - 1$ mēroga aproksimācijas koeficients $y_a[num]$ ar tādu pašu numuru, ar kādu tas bija ieejas datos. Pēc rezultāta ierakstīšanas izejas signāla buferī, koeficienta numurs tiek palielināts par soli MOD, t.i., $num = num + MOD$.

   3.4. Filtrācijas procedūra tiek uzreiz atkārtota nulles vērtībām, kas tika pievienotas pēc interpolācijas ar nullēm, operācijas, kas atjauno sākotnējo nolašu skaitu pēc decimācijas. Šajā gadījumā filtrācija tiek veikta šādām ieejas koeficientu vērtībām:

   - $x_d = 0$;
   - $x_a = 0$,

   bet rezultāts līdzīgi tiek saglābāts izejas signāla buferī $y_a[num]$, pēc saglabāšanas koeficienta numurs tiek palielināts $num = num + MOD$.

   3.5. Punkti 3.2.-3.4. tiek atkārtoti visiem detalizācijas koeficientiem $registr[j][k]$ ar numuriem $k = 0, 1, 2, ..., KK[j] - 1$, kur $KK[j]$ ir tādu koeficientu skaits mērogā j.

   3.6. Punkti 3.2.-3.5. tiek atkārtoti visiem detalizācijas koeficientiem $registr[j][k]$ visos mērogos $j = J - 1, ..., 2, 1, 0$. Kad tiek sasniegts mērogs $j = 0$ un koeficienta numurs ir $k = KK[0] - 1$, inversās diskrētās veivletu transformācijas iterācija ir paveikta un tiek veikta nākamās iterācijas koeficientu uzkrāšana un aizture (punkts 2.1.).

Vislielākais operāciju skaits šajā algoritmā ir saistīts ar laika aiztures veidošanu, pie nosacījuma, ka tās veikšanai tiks izmantots bīdes reģistrs. Tādā gadījumā, nobīde tiek veikta saskaņā ar 2.8a. att. parādīto algoritmu, pēc kura:

1. No pēdēja masīva elementa ar numuru $num_L = L_{\text{delay}}[j]-1$ tiek nolasīta koeficienta vērtība y;

2. Masīva elementa numurs tiek samazināts: $num_L = num_L - 1$ un ja $num_L > 0$, tad vērtība no masīva elementa $delay[j][num_L]$ tiek nolasīta un ierakstīta masīva elementā $delay[j][num_L - 1]$;

3. Punkts 2. tiek atkārtots, kamēr $num_L > 0$, savukārt ja $num_L = 0$, tad ieejas lielums $koeff$ tiek ierakstīts pirmajā masīva elementā $delay[j][0]$ un bīdes operācija ir pabeigta.

Tādai pieejai ir būtiskais trūkums, par cik saskaņā ar (2.17) izteiksmi, vismazākā mēroga $j = 1$ laika aizture var sasniegt lielumu: $L_{\text{delay}} = 4092$ kopējam mērogu skaitam $J = 10$. Savukārt ja mērogu kopējais skaits ir $J = 20$, tad maksimālā laika aizture sasniegs: $L_{\text{delay}} = 4194300$. Veicot aizturi, būs jāveic L_{delay} nolasīšanas operācijas un L_{delay} ierakstīšanas operācijas.

Galvenais šeit ir tas, ka interesē tikai ieejas un izejas lielums, nevis pārējie lielumi. Daudz efektīvāka pieeja būtu izmantot buferatmiņu, nevis aiztures līniju. Šajā gadījuma buferatmiņas elementu skaits paliek nemainīgs – $L_{\text{delay}}[j]$ elementi katrā mērogā. Šajā gadījumā, koeficienti tiek apstrādāti līdzīgi kā tas tiek darīts koeficientu buferatmiņā no 2.11. att.:

- Tiek noteikts koeficienta kārtas numurs[17] $num = (i-1)/MOD$;
- Šis numurs $num_L = num$ tiek palielināts ar katru saņemto detalizācijas koeficientu šajā mērogā j par vienību. Lai nonullētu šo vērtību pēc $num_L = L_{\text{delay}}[j] - 1$ vērtības sasniegšanas, koeficienta numura vietā ir jāizmanto dalīšanas atlikums no koeficienta numura num dalījuma ar aiztures līnijas garumu $L_{\text{delay}}[j]$. Tādā veidā, atliek nolasīt vērtību no masīva elementa $y = delay[j][num_L]$ un uzreiz ierakstīt tās vietā jauno vērtību $delay[j][num_L] = koeff$. Pēc L_{delay} koeficientiem šis numurs num_L atkal norādīs uz šo elementu, un tas tiks nolasīts ar laika aizturi $L_{\text{delay}}[j]$.

Secinot, tāda pieeja ļauj samazināt nolasīšanas un ierakstīšanas operāciju skaitu no $L_{\text{delay}}[j]$ skaita līdz vienīgajai operācijai ar papildus dalījuma aprēķinu. Būtiskākais ir, ka algoritma skaitļošanu ir apjoms ir minimāls un vairs nav atkarīgs no aiztures līnijas garuma $L_{\text{delay}}[j]$. Saskaņā ar 2.10. att. parādīto algoritma shēmu tika izveidota programma ar visiem augstāk minētiem uzlabojumiem, kuras tekstu satur 5.pielikums.

2.7. Reāllaika tiešā J mērogu veivletu transformācija ar polifāzes filtriem

Ātrā veivletu transformācija, kuru veic ar filtru bankām kā tas ir parādīts 3.1. att. nav pilnīgi racionāla, jo puse no aprēķinātajām nolasēm netiks izmantota. Veikt decimāciju pirms filtrācijas, kā tika radīts iepriekš, tiešajā veidā nevar. Tomēr, pastāv iespēja sadalīt filtru $H(z)$ uz pāra/nepāra daļām $H_{\text{ev}}(z)$ un $H_{\text{odd}}(z)$ ar kurām arī apstrādāt signāla pāra/nepāra komponentes $x_{\text{ev}}[n]$ vai $x_{\text{odd}}[n]$. Atkarībā no tā, kādā veidā kombinēt šos lielumus ir iespējams veikt filtrāciju tikai pārskaitļa numura nolasēm:

$$\left(H(z)X(z)\right)_{\text{ev}} = H_{\text{ev}}(z)X_{\text{ev}}(z) + z^{-1}H_{\text{odd}}(z)X_{\text{odd}}(z), \tag{2.23}$$

vai arī tikai nepārskaitļa numura nolasēm:

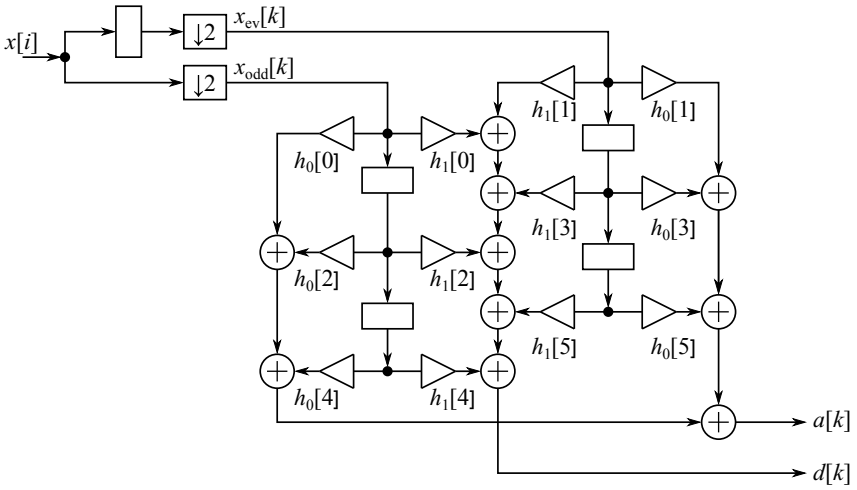
$$\left(H(z)X(z)\right)_{\text{odd}} = z^{-1}\left(H_{\text{ev}}(z)X_{\text{odd}}(z) + H_{\text{odd}}(z)X_{\text{ev}}(z)\right). \tag{2.24}$$

Konkrētā izteiksme tiek izvēlēta atkarībā no tā, kādas nolases ir jāatstāj, veicot decimāciju – ar pārskaitļa numuriem, vai nepārskaitļa numuriem. Acīmredzami, ka attiecīgie aproksimācijas un detalizācijas koeficienti šajos gadījumos atšķirsies, bet tomēr tie abos gadījumos raksturo vienu un to pašu signālu – un var būt izmantoti ideālajai rekonstruēšanai.

Lietojot polifāzes filtrus, ir iespējams samazināt aprēķinu skaitu, salīdzinājumā ar augstāk apskatītajām shēmām. Tiešās ātrās reāllaika veivletu transformācijas algoritmā tika veikti uzlabojumi, kas ievēro nolases numura paritāti un samazina veicamo reizinājumu skaitu 2 reizēs. Tomēr, nobīdes operācija tiek veikta filtrā visām nolasēm un ir iespējams uzlabot šo shēmu izvairoties no tādas operācijas nolasēm, kuras pēc filtrācijas tiek atmestas.

Polifāzes filtra realizācijā viens filtrs ar impulsa reakciju $h_0[n]$, kur $n = 0, 1, 2, ..., N$, šeit N ir filtra kārta tiek aizvietots ar diviem (vai vairākiem, ($\downarrow M$) decimācijai) filtriem, kuru impulsa reakcijās ir pāra nolases $h_{\text{ev}} = h_0[2k]$ vai nepāra nolases $h_{\text{odd}} = h_0[2k+1]$, kur $k = 0, 1, 2, ..., NN$, šeit NN ir vienas fāzes filtra kārta. Pie tam, šādu nolašu skaits sastāda pusi no sākotnējā nolašu skaita $LL = (N+1)/2$. Tādā veidā tiks veikti $L = 2LL$ reizinājumi un $N = 2NN$ nobīdes ar aiztures elementiem uz katrām divām nolasēm ar 1 papildus aiztures elementu pāra un nepāra nolašu savietošanai laikā (2.12. att. tā ir pirms pāra nolašu decimācijas operācijas). Tiešajā

[17] Šis lielums tiek aprēķināts koeficienta numura noteikšanai buferatmiņā *num* 2.11. att. un šo lielumu var uzreiz izmantot.



2.12. att. Filtru bankas polifāzes realizācija, pēc decimācijas paliek nolases ar nepārskaitļa nolašu numuriem.

filtru bankas filtrācijā, kas parādīta 2.4. att.[18] operāciju skaits ir nepieciešams vienas nolases apstrādei, un tiek apstrādātas visas nolases.

Filtru banka polifāzes realizācija Dobeši-3 ātrās veivletu transformācijas aprēķinam ir parādīta 2.12. att. Zemfrekvenču (aproksimācijas) filtra impulsa reakcija ir $h_0[n]$ un augstfrekvenču filtra impulsa reakcija ir $h_1[n]$. Tā kā pēc decimācijas veikšanas ir jāpaliek tikai nepārskaitļa numura nolasēm, filtrācija ir jāveic saskaņā ar (2.24) izteiksmi. Pie tam, 2.12. att. shēmā izmantotie decimācijas elementi $x_{\text{ev}}[k]$ un $x_{\text{odd}}[k]$ formēšanai arī saglabā tikai nepārskaitļa numura nolases.

Gadījumā, ja decimācijai ir jāatstāj pārskaitļa numura nolases (kā tas ir plaši izplatīts) ir jāveic šādi shēmas papildinājumi:

- pāra komponentes $x_{\text{ev}}[k]$ un nepāra komponentes $x_{\text{odd}}[k]$ ir jāpadod filtru ieejās otrādi:
  - $x_{\text{ev}}[k]$ jāpadod aiztures līnijas ieejā ar koeficientiem $h[0], h[2], h[4]$;
  - $x_{\text{odd}}[k]$ jāpadod aiztures līnijas ieejā ar koeficientiem $h[1], h[3], h[5]$;
- aiztures elements pirms decimācijas veikšanas ir jāpārnes $x_{\text{odd}}[k]$ kanālā.

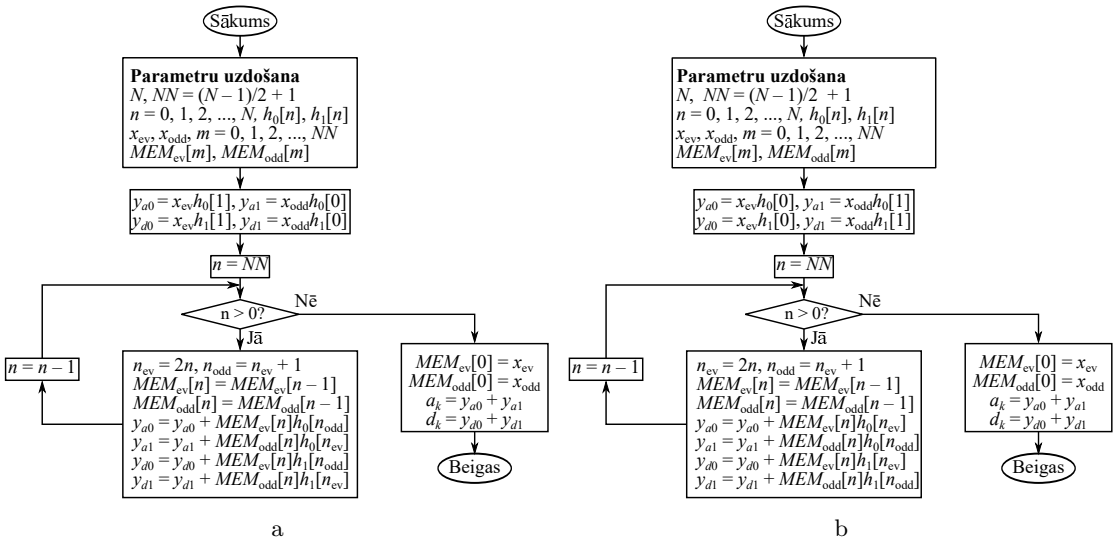
Ja izmantotie decimācijas operācijas elementi $x_{\text{ev}}[k]$ un $x_{\text{odd}}[k]$ formēšanai arī saglabā tikai pārskaitļa numura nolases, tad visi šie labojumi tiks veikti automātiski un nekas cits, izņemot decimācijas operācijas elementus, šajā shēmā nav jāmaina.

Salīdzinot šīs divas realizācijas, kas atšķiras tikai ar to, kādas nolases būs filtru bankas izejā – ar pārskaitļa vai nepārskaitļa numuriem, var secināt, ka principā algoritms nemainās. 3.pielikums. satur abu filtru bankas realizācijas funkciju tekstus, kuras tika izveidotas *C++* valodā. Ja izveidotājā apakšprogrammā (funkcijā) $x_{\text{ev}}[k]$ un $x_{\text{odd}}[k]$ argumentus mainīt vietām, tiks sasniegts tāds pats rezultāts. Ne ar ko citu šie divi algoritmi neatšķiras, izņemot to, ka signāla un filtra impulsa reakcijas koeficientu pāra/nepāra komponentes tiek reizinātas, attiecīgi, 2.13a. att. saskaņā ar (2.24), bet 2.13b. att. – saskaņā ar (2.23). Šo filtru izejas komponentes tiek saskaitītas, lai noteiktu aproksimācijas koeficientu a_k un detalizācijas koeficientu d_k.

2.13. att. parādīto funkciju algoritmos ir jāuzdod šādi argumenti:

- aproksimācijas (ZF) filtra impulsa reakciju $h_0[n], n = 0, 1, 2, ..., N$, kur N ir filtra kārta;
- detalizācijas (AF) filtra impulsa reakciju $h_1[n], n = 0, 1, 2, ..., N$, kur N ir filtra kārta;
- pārskaitļa numura signāla (vai cita mēroga aproksimācijas koeficienta) nolasi x_{ev};

[18] Šeit ir apvienotas filtru aiztures līnijas, jo tās ir vienādas un strādā ar vieniem un tiem pašiem datiem.



2.13. att. Polifāzes filtru bankas realizēšana: a) nepārskaitļa numura nolasēm, b) pārskaitļa numura nolasēm.

- nepārskaitļa numura signāla (vai cita mēroga aproksimācijas koeficienta) nolasi x_{odd};
- polifāzes filtra sastāvdaļu impulsa reakcijas garumu $NN = (N+1)/2-1$, kur N – oriģinālā filtra impulsa reakcijas garums;
- polifāzes filtra pāra daļas aiztures līnija $MEM_{\text{ev}}[m], m = 0, 1, 2, ..., NN$;
- polifāzes filtra nepāra daļas aiztures līnija $MEM_{\text{odd}}[m], m = 0, 1, 2, ..., NN$.

Algoritma izpildes rezultātā funkcija atgriež šādas vērtības:

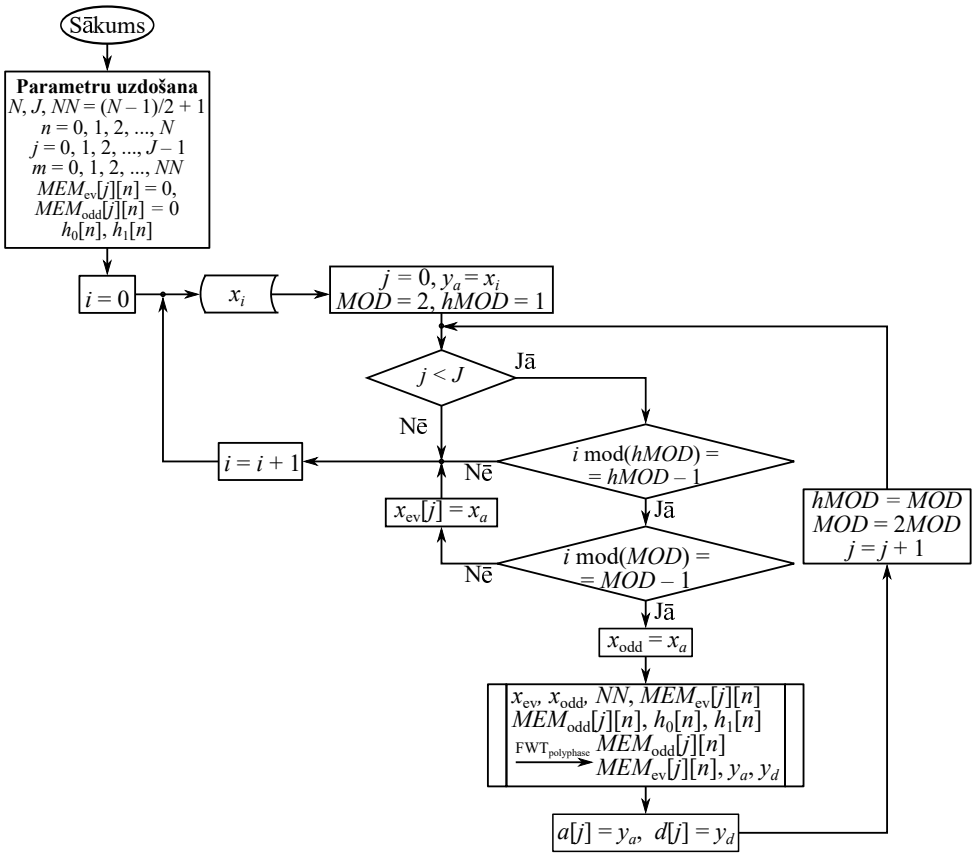
- aproksimācijas koeficienta vērtība a_k;
- detalizācijas koeficienta vērtība d_k.

Zemāk, 2.14. att., ir parādīts ātrās diskrētas veivletu transformācijas algoritms patvaļīgajām mērogam J, kas strādā reāllaikā un ir realizēts ar polifāzes filtru bankām. Šis algoritms ir realizēts nepārskaitļa nolašu apstrādei un dod pilnīgi tādu pašu rezultātu kā apskatītais 2.7. att. algoritms, bet veicot mazāku aprēķinu skaitu.

Saskaņā ar 2.14. att. parādīto algoritma shēmu tika izveidota testēšanas programma *C++* valodā, kura pārbaudīja tā darbību un novērtēja patērēto laiku apstrādei. Šīs programmas tekstu iekļauj sevī 6.pielikums.

Šis algoritms ir praktiski analoģisks apskatītajam 2.4.sadaļas 2.7. att. algoritmam. Galvenā atšķirība ir izsaucamajās apakšprogrammās (funkcijās).Tiešajā filtru bankas realizācijā bija divas apakšprogrammas: laika aizture, kas apstrādā gan pārskaitļa numura nolases, gan arī nepārskaitļa numura nolases, un reizināšanas apakšprogramma, kas veic nolašu reizinājumu no laika aiztures līnijas ar filtra impulsa reakcijas koeficientiem saskaņā ar ciparu filtra darbības principu.

Filtru bankas polifāzes realizācijā, kas parādīta 2.14. att. un saglabā tikai nepārskaitļa numura nolases, apakšprogramma apstrādā katras divas nolases paralēli $x_{2k} = x_{\text{ev}}$ un $x_{2k+1} = x_{\text{odd}}$ – vienu pāra un vienu nepāra nolasi. Līdz ar to, ja tekošā nolase x_i ir ar nepārskaitļa nolases numuru un pie tam atlikums $R_2 = imod(MOD) = MOD - 1$ pēc (2.11), tad jāveic apstrādi kopā ar iepriekšējo pārskaitļa numura nolasi. Tieši atlikums $MOD - 1$, kur $MOD = 2^j$, norāda uz to, ka mērogā j nolase x_{odd} ir ar nepārskaitļa numuru un to ir jāapstrādā. Detalizētāk tas bija aprakstīts 2.4.sadaļā. Savukārt, lai kopā ar šo nolasi būtu attiecīgā pārskaitļa



2.14. att. J mērogu reāllaika ātrās diskrētās veivletu transformācijas algoritms ar polifāzes filtru banku nepārskaitļa nolasēm.

numura nolase x_{ev}, pirms atlikuma R_2 pārbaudes ir jāpārbauda saskaņā ar (2.10) atlikumu $R_1 = imod(hMOD) = hMOD - 1$, kur $hMOD = MOD/2 = 2^{j-1}$. Patiešām, var pierādīt, ka gadījumā ja ir spēkā (2.11), tad būs spēkā arī (2.10). Dalītāja divkāršošana šajā gadījumā atbilst decimācijas operācijai, kas no divām nolasēm ar numuru i, tādam, lai $R_1 = hMOD - 1$ saglabā katru otro (nepārskaitļa numura) nolasi ar numuru i, kuram ir spēkā $R_2 = MOD - 1$.

Tādā veidā, pamatojoties uz visu augstāk minēto, var secināt, ka apskatot mēroga j ieejas signāla nolasi x_k (kurš pie $j > i$ ir iepriekšējā mēroga aproksimācijas koeficients $a_{j-1,k}$)) gadījumā kad izpildās (2.10), t.i., $R_1 = hMOD - 1$, bet neizpildās (2.11), t.i. $R_2 \neq MOD - 1$ iet runa par šī signāla nolases pārskaitļa numuru k. Tādu nolasi ir jāsaglabā atmiņā līdz nākamajai nolasei $k + 1$ ar nepārskaitļa numuru.

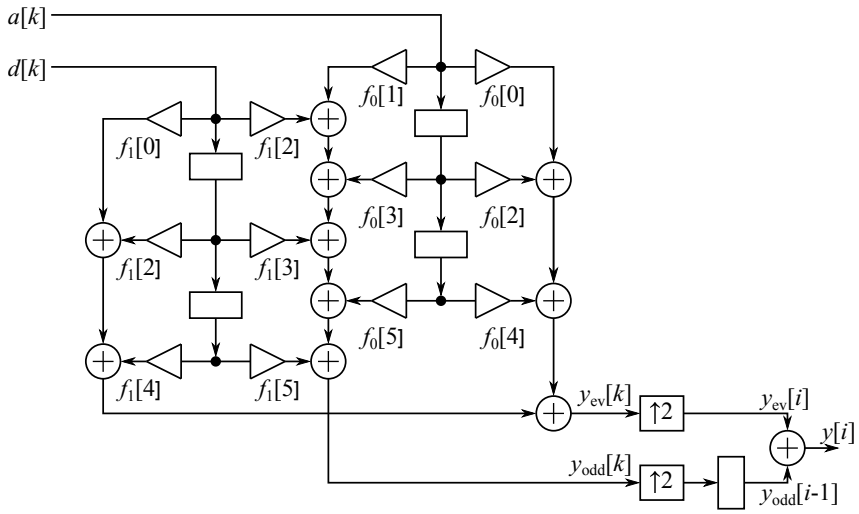
Ievērojot to, ka algoritms reāllaikā aprēķina aproksimācijas koeficientus $a_{j,k}$ un detalizācijas koeficientus $d_{j,k}$ visos mērogos $j = 1, 2, ..., J$ pēc kārtas, tad ir nepieciešams saglābāt pārskaitļa numura nolasi $x_{\text{ev}}[j]$ katrā no mērogiem tā, lai jaunā mēroga vērtība nepārrakstītu iepriekšējā mēroga vērtību. Acīmredzami, ka tādā gadījumā $x_{\text{ev}}[j]$ nolases ir jāsaglabā masīvā. Nepārskaitļa numura nolases nav jāsaglabā visos mērogos, pietiek tikai ar tekošā mēroga vērtību x_{odd}, jo šī nolase tiek nekavējoties apstrādāta. Programmas izpildes sākumā pārskaitļa numuru nolašu masīvs tiek aizpildīts ar nullēm $x_{\text{ev}}[j], j = 0, 1, 2, ..., J - 1$, kur J – analīzes mērogu skaits.

Lai 2.14. att. parādītais algoritms veiktu decimāciju atstājot pārskaitļa numura nolases, ir jāsalīdzina atlikumi R_1 un R_2 ar nulli, nevis lielumiem $hMOD - 1$ un $MOD - 1$, attiecīgi. Bez tam, apakšprogrammas izsaukšanā ir jāmaina vietām argumenti $x_{\text{ev}}[j]$ un x_{odd}. Acīmredzami, ka tādā gadījumā sanāks citas aproksimācijas un detalizācijas koeficientu vērtības.

2.8. Reāllaika inversā J mērogu veivletu diskrētā transformācija ar polifāzes filtriem

Līdzīgi tam, kā varēja uzlabot 2.3.sadaļā aprakstīto tiešās veivletu diskrētās transformācijas algoritmu J mērogos, pielietojot polifāzes filtrus, var uzlabot arī 2.6.sadaļā aprakstīto inversās veivletu diskrētās transformācijas algoritmu. Tiešās veivletu transformācijas gadījumā ieguvums bija saistīts ar to, ka varēja neveikt filtrāciju nolasēm, kuras netiks izmantotas, bet atmestas decimācijas procesā. Inversās diskrētās transformācijas gadījumā, līdzīgi, nav jāfiltrē nuļļu nolases, kas tiek pievienotas, interpolējot signālu ar nullēm pirms filtrācijas. Tā vietā, interpolācija ar nullēm tiek veikta pēc filtrācijas. Arī filtri, ar kuriem notiek vienas (nevis divu) nolašu apstrāde ir vienkāršāki skaitļošanas ziņā – koeficientu skaits nemainās, taču aiztures elementu skaits samazinās.

Zemāk, 2.15. att., ir parādīta tādas sintēzes filtru bankas struktūra ar polifāzes realizāciju Dobeši-3 veivletu gadījumā (filtru kārta $N = 5$). Kā tam jābūt saskaņā ar 2.5. att. parādīto filtru bankas struktūru, šādai filtru bankai ir divas ieejas un viena izeja.



2.15. att. Sintēzes filtru bankas polifāzes realizācija.

2.15. att. filtru bankas ieejās tiek padoti aproksimācijas koeficienti $a[k]$ un detalizācijas koeficienti $d[k]$, turklāt tos nesadala pāra un nepāra fāzēs – tiek padoti visi koeficienti. Filtrācijas rezultātā veidojās divas signāla komponentes. Viena no tām ir signāla pāra fāze $y_{a0}[k]$, kura tiek formēta saskaņā ar izteiksmi:

$$y_{a0}[k] = a[k] \cdot F_0[0] + a[k-1] \cdot F_0[2] + a[k-2] \cdot F_0[4] = \sum_{n=0}^{NN} a[k-n] F_0[2n], \qquad (2.25)$$

kur NN ir vienas fāzes filtra kārta, kā arī tiek veidota signāla nepāra fāze $y_{a1}[k]$ saskaņā ar šādu izteiksmi:

$$y_{a1}[k] = a[k] \cdot F_0[1] + a[k-1] \cdot F_0[3] + a[k-2] \cdot F_0[5] = \sum_{n=0}^{NN} a[k-n] F_0[2n+1], \qquad (2.26)$$

kur NN ir vienas fāzes filtra kārta. Līdzīgajā veidā tiek apstrādāti detalizācijas koeficienti $d[k]$

ar filtru, kura pārvades funkcija ir $F_1(z)$. Veidojās pāra signāla fāze:

$$y_{d0}[k] = d[k] \cdot F_1[0] + d[k-1] \cdot F_1[2] + d[k-2] \cdot F_1[4] = \sum_{n=0}^{NN} d[k-n] F_1[2n], \tag{2.27}$$

un nepāra signāla fāze:

$$y_{d1}[k] = d[k] \cdot F_1[1] + d[k-1] \cdot F_1[3] + d[k-2] \cdot F_1[5] = \sum_{n=0}^{NN} d[k-n] F_1[2n+1], \tag{2.28}$$

kur NN ir vienas fāzes filtra kārta.

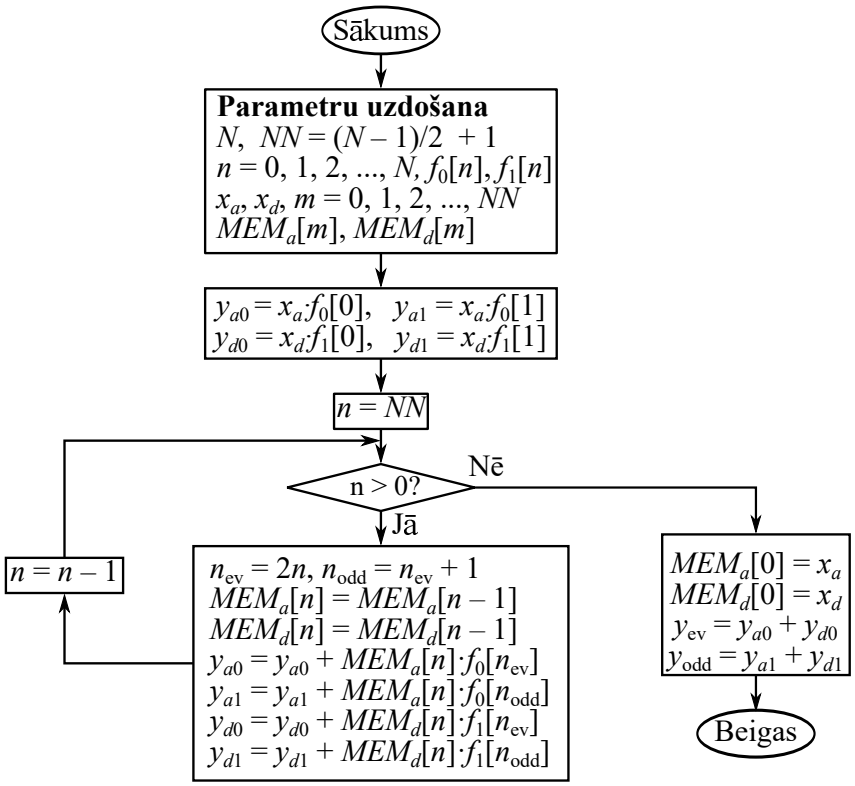
Rezultātā, saskaitot (2.25) ar (2.27) un (2.26) ar (2.28), tiek iegūtas izejas signāla $y[k]$ pāra fāze:

$$y_{\text{ev}}[k] = y_{a0}[k] + y_{d0}[k] = \sum_{n=0}^{NN} a[k-n] F_0[2n] + \sum_{n=0}^{NN} d[k-n] F_1[2n] \tag{2.29}$$

un izejas signāla $x[k]$ nepāra fāze:

$$y_{\text{odd}}[k] = y_{a1}[k] + y_{d1}[k] = \sum_{n=0}^{NN} a[k-n] F_0[2n+1] + \sum_{n=0}^{NN} d[k-n] F_1[2n+1]. \tag{2.30}$$

Šīs divas fāzes tiek interpolētas ar nullēm, bet $y_{\text{odd}}[i]$ fāze vēl tiek aizkavēta par 1 laika vienību, lai kompensētu analīzes filtru bankā ieviesto laika aizturi pāra signāla fāzē (skat. 2.12. att.). Šīs divas signāla fāzes tiek saskaitītas, un tā veidojās izejas signāla nolases: $y[i] = y_{\text{ev}}[i] + y_{\text{odd}}[i-1]$.



2.16. att. Polifāzes sintēzes filtru bankas realizēšanas algoritms.

2.16. att. parādītās algoritma funkcijas realizāciju *C++* valodā satur 3.pielikums. Augstāk aprakstīto filtru banku var realizēt ar 2.16. att. parādīto algoritmu. Šī algoritma ieejas dati ir:

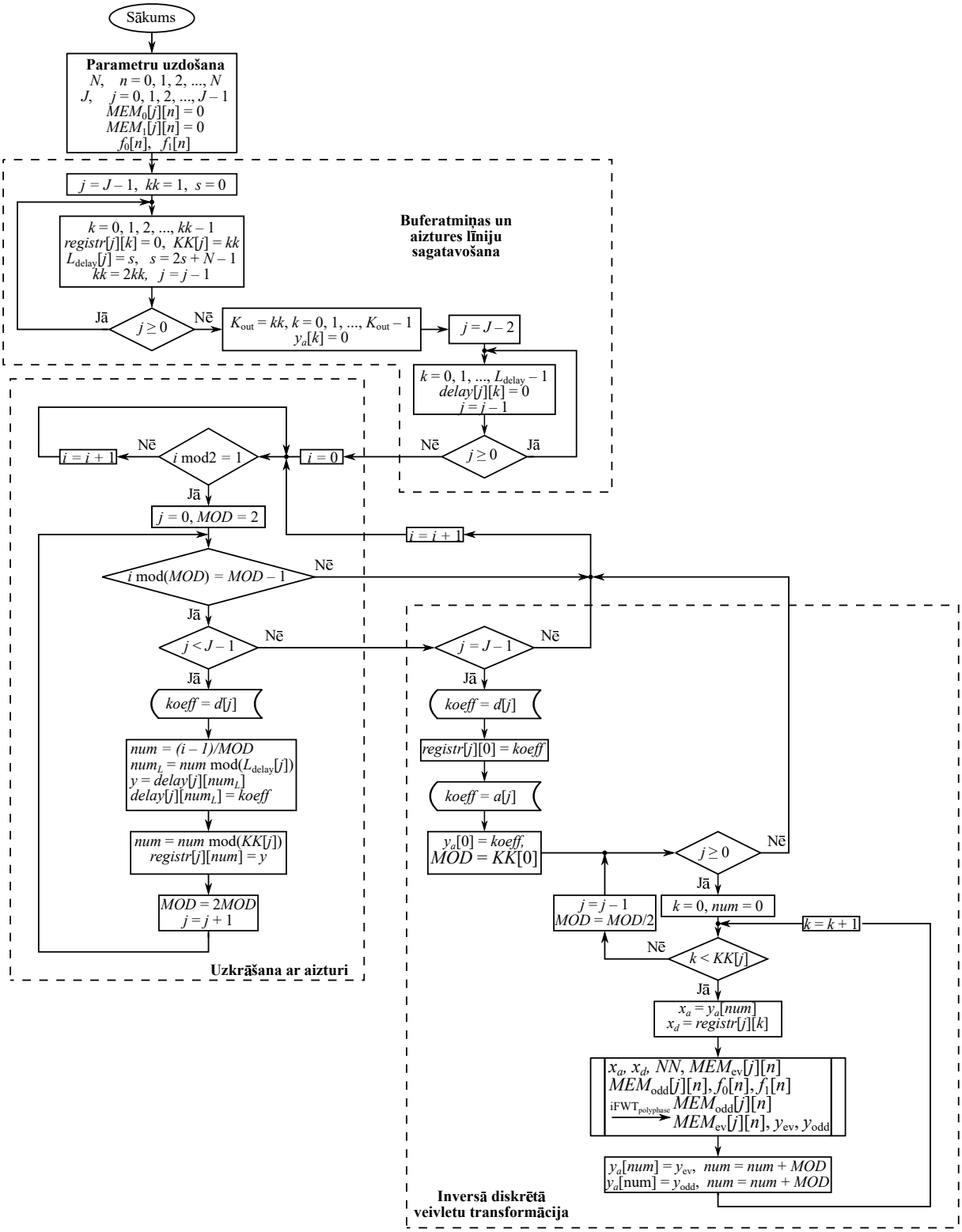
- pilnās filtru impulsa reakcijas $f_0[n]$ un $f_1[n]$, kur $n = 0, 1, 2, ..., N$, bet N – filtra kārta;
- viens aproksimācijas koeficients x_{a} un viens detalizācijas koeficients x_{d};
- pilnā filtra kārta N un vienas fāzes filtra kārta NN, kas var būt aprēķināta pēc N;
- aproksimācijas un detalizācijas filtru stāvokļi $MEM_a[n]$ un $MEM_d[n]$, attiecīgi, kur $n = 0, 1, 2, ..., NN$, bet NN – vienas fāzes filtra kārta.

Algoritms apstrādā aproksimācijas koeficientus x_{a} un detalizācijas koeficientus x_{d} saskaņā ar 2.15. att. parādīto filtru struktūru un aprēķina vienādojumus (2.25), (2.26), (2.27) un (2.28). Vienīgā atšķirība šajā datorapstrādes algoritmā ir tāda, ka izpildes rezultātā tiek veidotas divas vērtības y_{ev} un y_{odd}, kuras tiek paralēli nolasītas, nevis sakārtotas vienā secībā, izmantojot interpolāciju ar nullēm, laika aizturi un saskaitīšanu.

Šāds reāllaika inversās diskrētās veivletu transformācijas algoritms ar polifāzes filtru bankām tiek izmantots J mērogu sintēzes filtru bankas formēšanā. Algoritms, kas realizē tādu filtru banku ir parādīts 2.17. att. un gandrīz pilnīgi sakrīt ar 2.6.sadaļā aprakstīto tiešās realizācijas inverso veivletu diskrētās transformācijas algoritmu. Tas ir izskaidrojams ar to, ka šajā algoritmā tiek mainīts tikai filtrs, un tādai izmainai nav jāmaina buferatmiņas apjomu, aiztures līnijas vai apstrādes kārtību.

Tā kā 2.10. att. parādītajā algoritmā interpolācija ar nullēm tiek veikta apstrādājot katru nolasi, t.i. uzreiz pēc pašas nolases tiek apstrādāta arī pievienotā pēc tās nulle, tad ir iespējams aizvietot šo divu nolašu filtrācijas procedūru pēc 2.6. att. ar vienas nolases filtrāciju ar polifāzes filtru banku pēc 2.16. att. algoritma. Rezultātā arī tiek veidotas divas nolases ar tādām pašām vērtībām, taču tas tiek panākts viena cikla izpildes laikā, turklāt filtru aiztures līnijas ir divās reizēs īsākas.

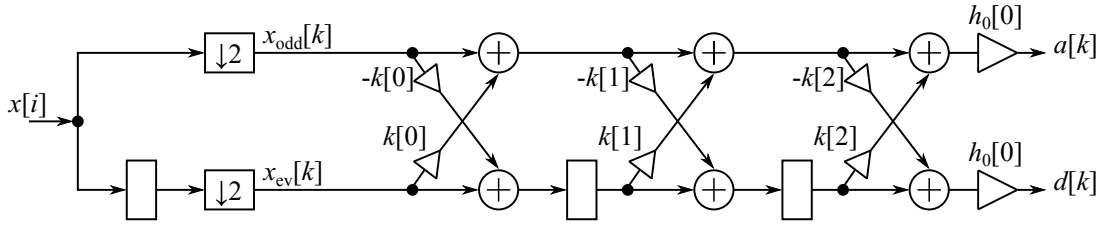
Šīs divas vērtības y_{ev} un y_{odd} pēc kārtas tiek saglabātas izejas bufera reģistrā $y_a[num]$ pozīcijā, kur $num = num + MOD$ ir nolases numurs reģistrā, bet MOD – solis, ar kuru šis numurs tiek palielināts, atkarībā no mēroga j. Turklāt jāatzīmē, ka šajā gadījumā interpolācija ar nullēm nav vispār jāveic. Pārbaudes programma ar diskrētās inversās daudzmērogu veivletu transformācijas realizāciju, izmantojot polifāzes filtru bankas saskaņā ar 2.17. att. parādīto algoritma shēmu, satur 7.pielikums..



2.17. att. J mērogu inversās reāllaika ātrās diskrētās veivletu transformācijas algoritms ar polifāzes filtru bankām.

2.9. Reāllaika J mērogu veivletu diskrētā transformācija ar kāpņveida filtriem

Pastāv vēl viena efektīvā filtru bankas shēma, kurā ir samazināts atmiņas elementu skaits un tajā pašā laikā decimācija tiek veikta pirms filtrēšanas. Tā ir kāpņveida filtru struktūra. Līdzīgi kā tas ir polifāzes filtru realizācijā, ieejas signāls tiek sadalīts pārskaitļa numura nolasēs $x_{\mathrm{ev}}[k]$ un nepārskaitļa numura nolasēs $x_{\mathrm{odd}}[k]$. Tādas shēmas piemērs Dobeši-3 veivletiem ir parādīts zemāk 2.18. att.



2.18. att. Dobeši-3 veivletu analīzes filtru banka ar kāpņveida filtriem, pēc decimācijas paliek nolases ar nepārskaitļa nolašu numuriem.

Salīdzinot shēmas 2.18. att. un 2.12. att., var konstatēt, ka shēmas strādā vienādi, taču sastāv no dažāda elementu skaita. Šīs atšķirības ir apkopotas 2.4. tabulā:

2.4. tabula. Polifāzes un kāpņveida filtru elementu skaita salīdzinājums Dobeši-3 veivletu filtru bankām.

	Polifāzes	Kāpņveida
Reizinātāji	12	8
Aiztures elementi	5	3
Sumatori	10	6

Kā var redzēt no 2.4. tabulas datiem, Dobeši-3 veivletu gadījumā ar filtru koeficientu skaitu $L = 6$ kāpņveida struktūras filtru banka ir pēc visu elementu skaita labāks risinājums, nekā vienkāršā polifāzes filtru banka. Tas tiek panākts ar to, ka filtru impulsa reakcija $H_0(z)$ un $H_1(z)$ tiek sadalīta reizinātājos ar koeficientiem $k[i]$, kur $i = 0, 1, ...LL - 1$. Šajā gadījumā $LL = L/2$ ir puse no tiešās realizācijas filtra impulsu reakcijas koeficientu $h_0[n]$ skaita.

Līdz ar to, lai tāds filtrs strādātu, ir nepieciešams pārrēķināt filtru bankas impulsu reakcijas koeficientus uz $k[n]$, kā arī noteikt jaunākas pakāpes koeficientu $h_0[0]$, kurš abos kanālos (pāra un nepāra) ir vienāds un tiek pareizināts filtrācijas beigās. Šis koeficients sakrīt ar analīzes filtru bankas aproksimējošā filtra jaunākās pakāpes koeficientu $h_0[0]$, savukārt pēdējais $k[LL]$ koeficients ir vienāds ar aproksimējošā filtra vecākās pakāpes koeficientu $h_0[N]$. Izmantojot iteratīvo algoritmu, ir iespējams noteikt arī pārējus $k[n]$ koeficientus. Vispirms ir jāpārrēķina impulsa reakcijas koeficientus $h_0[n]$:

$$\tilde{h}_0[n]^{(l-1)} = \tilde{h}_0[n]^{(l)} - k[l]\tilde{h}_1[n]^{(l)}, \tag{2.31}$$

Turklāt $\tilde{h}_0[n]^{(LL)} = h_0[n]$ un $\tilde{h}_1[n]^{(LL)} = h_1[n]$, bet koeficients $k[LL] = h_0[N]$. Šajā gadījumā koeficients $k[l-1]$ var būt novērtēts pēc iegūtas reakcijas $\tilde{h}_0[n]^{(l-1)}, n = 0, 1, 2..., N$, kur N ir filtru bankas kārta. Konkrēti, to var noteikt saskaņā ar:

$$k[l-1] = \tilde{h}_0[n_{\max}]^{(l-1)}, \tag{2.32}$$

kur $n_{\max} = \max(n)$ kuram $\tilde{h}_0[n]^{(l-1)} \neq 0$. Savukārt, $\tilde{h}_1[n]^{(l)}$ ir iespējams aprēķināt, izmantojot likumsakarību starp ortogonālo filtru banku aproksimācijas un detalizācijas filtru impulsa reakcijām (2.3):

$$\tilde{h}_1[n]^{(l-1)} = (-1)^n \tilde{h}_0[n_{\max} - n]^{(l-1)}, \tag{2.33}$$

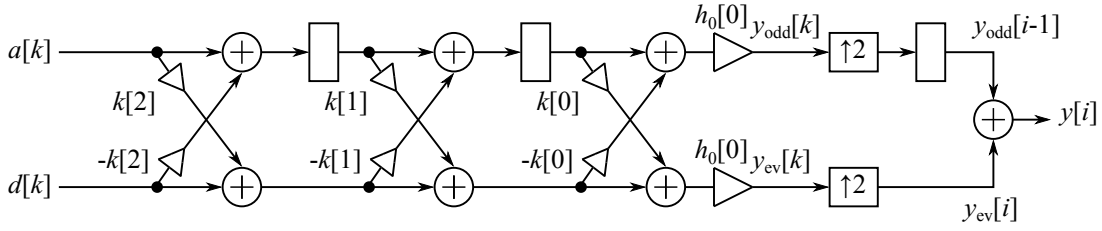
visām $n = 0, 1, 2, ...n_{\max}$ vērtībām, kurām $\tilde{h}_0[n] \neq 0$.

Iteratīvi atkārtojot (2.31), (2.32) un (2.33) visiem $LL \geq l \geq 1$, tiek noteikti kāpņveida filtra koeficienti filtru bankai no 2.18. att. Automatizētajam aprēķinam tika izveidots *Matlab* programmas skripts, kas pārrēķina tiešās filtru bankas impulsa reakcijas koeficientus $h_0[n]$ un $h_1[n]$ kāpņveida filtru bankas koeficientos $k[n]$. Šīs programmas kods ir dots zemāk:

```
clc, clear all, format compact
%Program is used to calculate k quotients for orthogonal filter bank from
%impulse response h0 (analysis, lowpass)

%Impulse Responses
load db3; [h0,h1,f0,f1]=orthfilt(db3);
L=length(h0); A=h0(1); B=f0(1);    %Normalization, z^0 quotient equal to 1
LL=L/2; n=1:L;

hh0(LL,:)=h0/A; hh1(LL,:)=h1/A; k(LL)=h0(L)/A;      %Analysis
for i=LL:-1:2
hh0(i-1,:) = (hh0(i,:)-k(i)*hh1(i,:))/(1+k(i)^2);  %Equation to calculate factor
nn = sum(abs(hh0(i-1,:))>10^-8);             %Count the number of nonzero samples
n=1:nn; k(i-1) = hh0(i-1,nn);               %Read maximum power quotient as k(i-1)
hh1(i-1,n) = fliplr(hh0(i-1,n)).*(-1).^n;      %h1, via alternating flip
end
```



2.19. att. Dobeši-3 veivletu sintēzes filtru banka ar kāpņveida filtriem.

Tādā veidā ir iespējams izveidot tiešās diskrētās veivletu transformācijas filtru banku ar kāpņveida filtriem. Savukārt, lai izveidotu inversās diskrētās veivletu transformācijas filtru banku ar kāpņveida filtriem, ir nepieciešams pārveidot 2.18. att. parādīto shēmu:

1. Jāmaina vietā izejas $a[k]$ un $d[k]$, kurās tiks ar ieeju $x[i]$, kurā formēsies izejas signāl nolases $y[i]$;

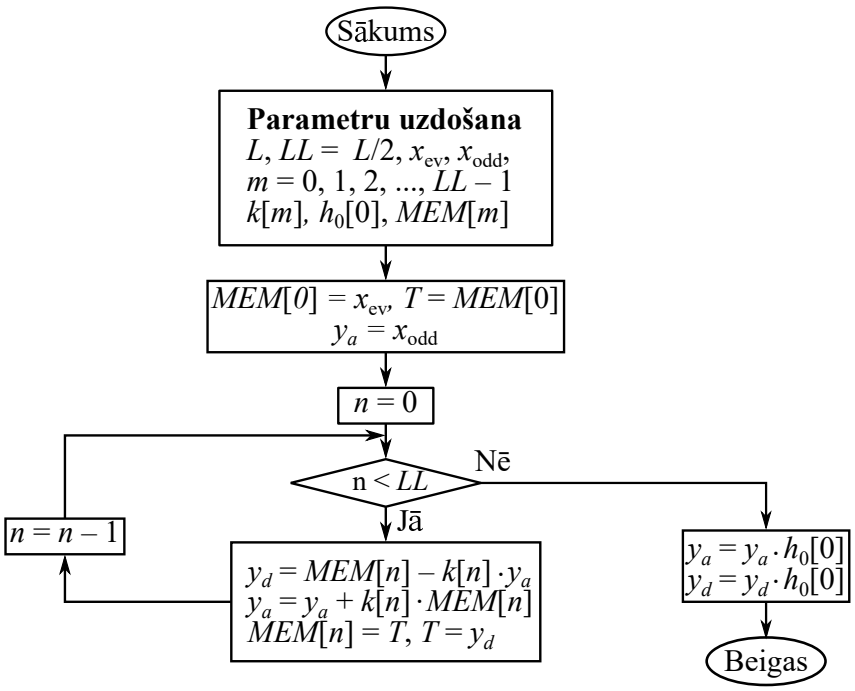
2. Izejas signāla $y[i]$ formēšanai no divām komponentēm – pāra $y_{\text{ev}}[k]$ un y_{odd} pēc interpolēšanas ar nullēm jāpieliek summators tā lai $y[i] = y_{\text{odd}}[i] + y_{\text{ev}}[i]$;

3. Analīzes filtru bankas aiztures kompensēšanai jāpārnes visi aiztures elementi no pāra kanāla $d[k]$ nepāra kanālā $a[k]$, tādā veidā abi kanāli tiks aizkavēti par vienādu lielumu;

4. Brīvais koeficients $h_0[0]$ abos kanālos jāpārnes pirms interpolācijas ar nullēm;

5. Visu koeficientu $k[l]$ zīmes abos kanālos tiek mainītas uz pretējām.

Rezultātā tiek iegūta sintēzes filtru bankas shēma, kas ir parādīta 2.19. att. Pēc būtības, tas nozīmē, ka lai veiktu inverso diskrēto veivletu transformāciju ar kāpņveida filtru banku, pietiek padot analīzes filtru bankas izejās koeficientus $d[k]$ un $a[k]$ (t.i., izejas kļūst par ieejām), taču to ir jādara pretējā kārtībā ($d[k]$ jāpadod ieejā $a[k]$, un otrādi), kā arī jānomaina decimāciju uz interpolāciju ar nullēm un jāpievieno sumators. Faktiski, tas ļauj izveidot universālo filtru bankas realizēšanas funkciju, kuru var izmantot gan tiešās, gan inversās diskrētās transformācijas aprēķinā. Tādas funkcijas algoritms ir parādīts 2.20. att., bet tās realizācijas kodu *C++* valodā satur 3.pielikums.



2.20. att. *DWT* filtru bankas realizācijas algoritms ar kāpņveida filtriem.

Salīdzinot 2.20. att. parādīto algoritmu ar polifāzes filtru bankas algoritmu 2.13. att., var secināt, ka tie ir ļoti līdzīgi un praktiski savstarpēji aizvietojami J mērogu algoritma shēmās 2.14. att. (tiešā transformācija) un 2.17. att. (inversā transformācija). Tas arī tika pārbaudīts realizējot šos algoritmus divās programmās:

- J mērogu tiešās diskrētās transformācijas programma ar kāpņveida filtriem, kuras kodu satur 8.pielikums.
- J mērogu inversās diskrētās transformācijas programma ar kāpņveida filtriem, kuras kodu satur 9.pielikums.

Abas programmas strādā analoģiski polifāzes filtru bankas programmām un transformācijas rezultātā sanāk tādi paši aproksimācijas koeficienti $a[k]$ un detalizācijas koeficienti $d[k]$. Vienīga atšķirība ir tāda, ka polifāzes filtru bankā jāuzdod filtru impulsa reakcijas koeficientus $h_0[n]$ un $h_1[n]$, $n = 0, 1, ..., N$, savukārt kāpņveida filtru bankai jāuzdod tā saucamie atstarošanas koeficienti $k[l]$, $l = 0, 1, ...L/2 - 1$, kur $L = N + 1$ ir filtru impulsa reakcijas koeficientu skaits.

2.10. Nobeigums

Šajā nodaļā tika apskatīta diskrētās veivletu transformācijas algoritmu realizācija ar filtru bankām programmēšanas valodā (šajā promocijas darbā tika izvēlēta *C++* valoda). Tas dod iespēju lietot algoritmu jebkuros uzdevumos, kuros var veikt tādu programmēšanu. Tika izveidoti gan tiešās diskrētās veivletu transformācijas algoritmi, gan inversās veivletu transformācijas algoritmi. Tika panākts, lai tie strādātu reāllaika bez datu segmenta uzkrāšanas. Tomēr, šeit ir jāprecizē, ka inversajā tranformācija datu uzkrāšana notiek un transformāciju nevar veikt, kamēr netiks saņemts pēdējā mēroga koeficients. Ja mērogu skaits ir liels, aiztures var būt ievērojamas. Kas attiecas uz tiešo diskrēto veivletu transformāciju, tur tādas aiztures nav un katra signāla nolase tiek analizēta visos mērogos J līdz nākamās nolases saņemšanai.

Tika izveidotas *C++* funkcijas un aprakstīti to algoritmi, kā arī tās tika izmantotas testēšanas programmās ar sekojošajām filtru bankām:

- Tiešās realizācijas filtru banka[19];
- Polifāzes filtru banka, kas veic decimāciju pirms filtrācijas un sadala nolases pāra un nepāra numura nolasēs, paralēli apstrādājot tos;
- Kāpņveida filtru banka, kas ir uzlabota polifāzes filtru banka ar samazināto reizinātāju un summatoru skaitu.

Tika konstatēts, ka viszemākā efektivitāte ir tiešajai filtru bankas realizācijai un jāizmanto polifāzes vai kāpņveida filtru bankas ātrākajai algoritma veiktspējai. Savukārt polifāzes un kāpņveida filtru bankas ir pielietojamas ļoti līdzīgi. To funkciju izsaukšanā ir vienīgā atšķirība impulsa reakcijas koeficientu izvēlē. Tas dod iespēju elastīgi mainīt algoritmus, izvēloties labāku pēc dažādiem kritērijiem (veiktspēja, precizitāte, utt.)

Apskatot inverso diskrēto veivletu transformāciju tika konstatēts sekojošais:

- No decimācijas operācijas ($\downarrow 2$) izpildes kārtības (atkarībā no tā, paliek nolases ar pāra vai nepāra numuriem) ir atkarīga inversās diskrētās veivletu transformācijas aizture katrā no mērogiem un kopējā aizture;
- Ja decimācijas operācijas rezultātā signālā paliek nepāra numura nolases, kopējā aizture inversajā diskrētajā veivletu transformācijā būs mazāka, nekā gadījumā kad pēc decimācijas operācijas paliek pārskaitļa numura nolases;
- Inversās diskrētās transformācijas aiztures līnijas tiešā realizācija programmēšanā dod pārāk lielus skaitļošanas zaudējumus un ir iespējams tos novērst, ja neveikt pārbīdi tiešajā nozīmē bet cikliski mainīt masīva nolasāmā elementa numuru;
- Jo lielāks ir kopējais mērogu skaits, jo vairāk datu tiek uzkrāts aiztures līnijā un jo vēlāk notiks signāla nolases rekonstruēšana.

Iegūtās programmas un rezultāti tiks izmantoti nākamajā nodaļā, Hersta parametra H novērtēšanas elementa izveidē. Tiks izmantota tikai tiešā diskrētā veivletu transformācija ar polifāzes vai kāpņveida filtru banku.

[19] Jāpiezīmē, ka veiktspējas uzlabošanai pusei no koeficientu, kuri netiek rēķināti, tika veikta tikai bīdes operācija bez koeficienta vērtības aprēķināšanas. Formāli tiešajā realizācijā filtrs tās aprēķina un pēc tam decimācijas gaitā tās tiek atmestas.

Nodaļa 3

Trafika sevlīdzīguma parametra novērtējums, tā kļūda un realizācija ar filtru bankām

3.1. Sevlīdzīguma parametra novērtēšana un veivletu transformācija

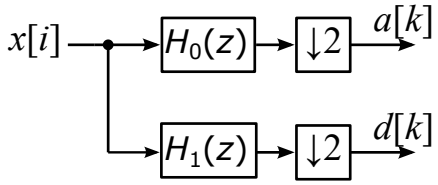
[1] piedāvātais algoritms var strādāt ar jebkuriem diskrētajiem veivletiem, piemēros zemāk tiks izmantoti Dobeši-3 veivleti, kuriem ir 3 nulles momenti. Detalizācijas un aproksimācijas filtru impulsu reakciju koeficientu skaits tādā gadījumā sastādīs $L = 6$. Aproksimācijas filtra impulsu reakcijas koeficienti Dobeši-3 (DB-3) veivletu gadījumā ir šādi [93]:

$$h_0[n] = \{0,035226\ -0,085441\ -0,135011\ 0,459878\ 0,806892\ 0,332671\}, \tag{3.1}$$

kur $n = 0, 1, 2, ..., N$ un N ir filtra kārta. Savukārt detalizācijas filtra koeficientus ortogonālo veivletu filtru bankas gadījumā var aprēķināt [93] saskaņā pēc izteiksmes:

$$h_1[n] = -(-1)^n h_0[N-n] \qquad \text{visiem } n = 0, 1, 2, ..., N. \tag{3.2}$$

Tādā gadījumā, signāla nolašu dekompozīciju var veikt ar ciparu filtrāciju, lietojot filtru bankas [66], kā tas ir parādīts 3.1. att. zemāk:



3.1. att. Diskrētā veivletu transformācija ar filtru banku.

Šeit $H_0(z)$ ir aproksimācijas filtra pārvades funkcija, kas atbilst impulsa reakcijai (3.1), bet $H_1(z)$ – detalizācijas filtra pārvades funkcija, kura atbilst (3.2). Diskrētās veivletu transformācijas rezultātā tiek izskaitļoti aproksimācijas koeficienti $\{a_k\}$ un detalizācijas koeficienti $\{d_k\}$. Abu šo koeficientu skaits būs vienāds ar ieejas nolašu skaitu, tāpēc, lai transformācijas rezultātā punktu skaits nebūtu divreiz lielāks, pēc katras filtrācijas procedūras diskretizācijas frekvence tiek pazemināta divreiz ("decimācija"). Tādā veidā, ja ieejas signāla nolašu skaits ir K, tad tiek aprēķināti $K/2$ aproksimācijas koeficienti $\{a_k\}$, kuri satur informāciju par signāla zemfrekvenču sastāvdaļām, un $K/2$ detalizācijas koeficienti, kas, savukārt, satur informāciju par signāla augstfrekvenču sastāvdaļām.

Ļoti bieži diskrēto veivletu transformāciju veic vairākkārtīgi. Tādā gadījumā nākamās veivletu transformācijas mērogā $WT^{(j)}$ ieejas signāls ir iepriekšējās veivletu transformācijas mēroga

$WT^{(j-1)}$ aproksimācijas koeficientu kopa $\{a_{j-1,k}\}$ un šo procesu var rekursīvi atkārtot vai nu līdz pieņemamam rezultātam vai arī tikmēr aproksimācijas koeficientu kopa $\{a_{j,k}\}$ nav pārāk īsa turpmākajai dekompozīcijai. Tādā veidā, pēc J mērogu diskrētās veivletu transformācijas aprēķina kopējo rezultātu veido:

- vislielākā mēroga J aproksimācijas koeficientu kopa $\{a_{J,k}\}$;
- visu mērogu detalizācijas koeficientu kopas: $\{d_{J,k}\}$, $\{d_{J-1,k}\}$, ..., $\{d_{1,k}\}$.

Tādā veidā, diskrētās veivletu transformācijas mērogā J kopējais punktu skaits joprojām paliek vienāds ar K. Tieši tāda vairākkārtīga diskrētā veivletu transformācija ir trafika sevlīdzīguma parametra H novērtēšanas pamatā: kārtējā mērogā analizējamās frekvences tiek samazinātas divreiz. Ja analizējamais process ir sevlīdzīgs, t.i., uzvedās līdzīgi dažādos laika mērogos, tad detalizācijas koeficientiem to ir jāatspoguļo.

Šis sevlīdzīgums tiek atspoguļots detalizācijas koeficientu jaudā D_j, kur j ir mēroga numurs, t.i., tiešās diskrētās veivletu transformācijas līmeņa numurs. Šī jauda tiek aprēķināta saskaņā ar (3.3) un eksponenciāli pieaug katrā mērogā, turklāt pieauguma ātrums ir konstants.

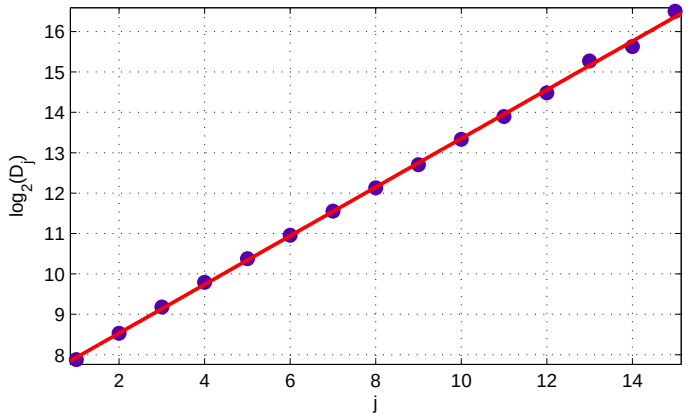
$$D_j = \frac{1}{n_j} \sum_{k=1}^{n_j} d_{j,k}^2, \tag{3.3}$$

kur j ir mēroga numurs, kurā tiek aprēķināta jauda D_j, $d_{j,k}$ – attiecīgā mēroga detalizācijas koeficienti d_k un n_j – tādu koeficientu skaits mērogā j.

3.2. att. ir parādīta $\log_2(D_j)$ atkarība no mēroga ar atsevišķiem punktiem, katrs no tiem atbilst detalizācijas koeficientu jaudas logaritmam mērogā j. Veicot šīs atkarības lineāro aproksimāciju, var noteikt tās lineārās atkarības slīpumu a, pēc kura, ņemot vērā ka:

$$a = 2H - 1, \tag{3.4}$$

var izteikt sevlīdzīguma parametru H. 3.2. att. ir parādīta detalizācijas koeficientu jaudas logaritma $\log_2(D_j)$ atkarība no mēroga Pareto gadījumrakstura procesam ar sevlīdzīguma parametru $H = 0,8$. Šīs atkarības punktu lineāras aproksimācijās rezultātā tika iegūta izteiksme: $\log_2(D_j) = 0,6024j + 7,329$, pēc kuras, izmantojot slīpuma parametru $a = 0,6024$, saskaņā ar (3.4) var noteikt sevlīdzīguma parametru $\hat{H} = (a+1)/2 = 0,8012$. Šī vērtība ir ļoti tuva uzstādītajai vērtībai $H = 0,8$.



3.2. att. Detalizācijas koeficientu jaudas logaritma $\log_2(D_j)$ atkarība no mēroga j.

Atkarības slīpuma parametra a novērtēšanai ir labi piemērota minimālo kvadrātu metode (3.5):

$$a = \frac{\sum_{j=1}^{J} j^2 \sum_{j=1}^{J} \log_2(D_j) - \sum_{j=1}^{J} j \sum_{j=1}^{J} j\log_2(D_j)}{J \sum_{j=1}^{J} j^2 - \left(\sum_{j=1}^{J} j\right)^2}, \tag{3.5}$$

kur j ir mēroga numurs, kurā tiek aprēķināta jauda D_j, bet J – tādu mērogu skaits.

Pēc aprakstītā sevlīdzīguma parametra H novērtēšanas algoritma tika izveidotas *Matlab* funkcijas, kuras veic veivletu transformāciju ar Dobeši-3 veivletiem, izmantojot filtru bankas (3.1. att.) pieeju, kā arī funkcija Hersta parametra H novērtēšanai. Ar šo funkciju var novērtēt sevlīdzīguma parametra $\hat{H}$ vidējo vērtību, turklāt ir iespējams norādīt kopējo mērogu skaitu J, un šis parametrs tiks novērtēts visos mērogos līdz norādītajam[1].

3.2. Sevlīdzīguma parametra novērtēšana modelētajam trafikam

Veivletu transformācijas mērogu skaita, transformācijas loga garuma un šo parametru atkarības izpētei no trafika novērtējamās sevlīdzīguma pakāpes H *Matlab* vidē tika izveidota trafika apstrādes programma. Šī programma novērtē sevlīdzīguma parametru $\hat{H}$ un tā absolūto kļūdu $\Delta H = |\hat{H} - H|$ dažādos transformācijas loga garumos un ievērojot dažādu mērogu skaitu. Mērķis – noskaidrot, kādā mērā šie parametri ietekmē novērtēšanas precizitāti un kā šī precizitāte ir atkarīga no tiem, ja mainās iestādītā novērtējamā parametra H vērtība.

Jāatzīmē, ka sevlīdzīgo trafiku ar uzdoto Hersta parametru H var ģenerēt ar dažādām metodēm, tajā skaitā arī ar veivletu transformāciju [80]. Var lietot neironu tīklus, kā tas ir panākts [51], kur aprakstīts kā apmācīts pēc reālā trafika neironu tīkls, kurš ģenerē trafiku ar tādu pašu Hersta parametru (kuru vēl ir jānosaka reālajam trafikam). Dažādi trafika ģenerēšanas modeļi tika pētīti arī [39] darbā. Mūsdienās plaši izmanto frakcionālos Gausa trokšņus ar veivletiem (*WBFGN*) [67]. Tomēr šajā eksperimentu sērijā tiek veidots visvienkāršākais sevlīdzīgā trafika modelis ar Pareto sadalījuma likumu. Trafiks šādai apstrādei arī tika ģenerēts *Matlab* vidē, izmantojot Pareto gadījumskaitļu ģeneratoru ar varbūtību blīvuma funkciju:

$$f(x) = \frac{\alpha}{\beta}\left(\frac{\beta}{x}\right)^{\alpha+1}, \tag{3.6}$$

kur $\alpha = 3 - 2H$, $\beta = \frac{\alpha-1}{\alpha\lambda}$, $\quad 1 \leq \alpha \leq 2$ un λ – trafika pieprasījumu intensitāte.

Tādu gadījumskaitli var ģenerēt šādi:

$$x = \frac{\beta}{R^{1/\alpha}}, \tag{3.7}$$

kur $R \in [0, 1]$ ir vienmērīgi sadalītais gadījumrakstura lielums.

Sevlīdzīgais trafiks tika veidots kā pieprasījumu skaits laika intervālos Δt, savukārt intervāli starp pieprasījumiem tiek ģenerēti saskaņā ar (3.7). Ja kārtējais pieprasījums pārsniegs mērījumu intervālu Δt, tas tiks uzskaitīts nākamajā mērījumu periodā. Tāds trafiks tika saglabāts failos turpmākajai pēcapstrādei. Visās simulācijās tika izmantots trafiks ar mērījumu intervālu $\Delta t = 1$ un intensitāti $\lambda = 50$, bet trafika kopējais ilgums sastādīja $T = 2^{24}$ Δt vienības. Tika veikta divu eksperimentu sērija:

- trafika analīze sevlīdzīguma parametra vērtībām $0,5 < H < 0,99$ ar pieauguma soli 0,1;

[1] Šeit, protams, būtu jāatzīmē, ka minimālais mērogu skaits tādai novērtēšanai ir 2, t.i., jārēķina vismaz 2 mērogu veivletu transformāciju. Tas ir pamatojams ar to, ka vienu punktu nevar aproksimēt ar lineāro funkciju.

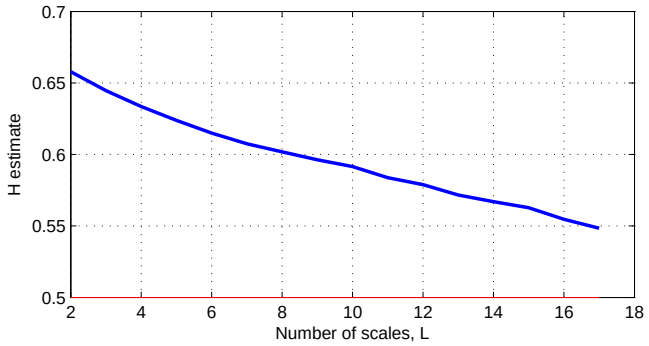
- trafika analīze sevlīdzīguma parametra vērtībām $0,8 < H < 0,99$ ar pieauguma soli 0,05.

Trafiks tika apstrādāts segmentos kuru garumi ir no $M_{\min} = 2^{10} = 1024$ līdz $M_{\max} = 2^{24} = 16\,777\,216$, turklāt pēc katras segmenta garuma palielināšanas 2 reizēs, pieauga arī maksimālais mērogu skaits par vienu. Attiecīgi, atkarībā no segmenta garuma sevlīdzīguma parametrs $\hat{H}$ tika novērtēts $J_{\min} = 2$ līdz $J_{\max} = 22$ mērogos, attiecīgi. Šeit ir jāatzīmē, ka tiek novērtēta tieši vidējā $\hat{H}$ parametra vērtība, savukārt segmenta garums ietekmēs novērtējumu skaitu, un, tātad arī šo vidējo vērtību. Tā, piemēram, segmentam ar maksimālo garumu $T = 2^{24}$ vidējā vērtība tiks aprēķināta pēc vienīgā novērtējuma līdz pat $J_{\max} = 22$ mērogam. Bez tam, tika novērtēta arī absolūta kļūda $\Delta H = |\hat{H} - H|$.

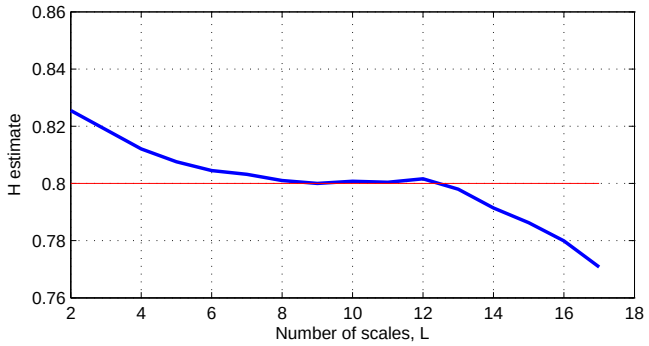
3.3. Modelētā trafika apstrādes rezultāti

3.2.sadaļā aprakstīto eksperimentu rezultātus tabulu formā satur 10.pielikums., 11.pielikums., 12.pielikums. un 13.pielikums. Pēc tām var konstruēt grafikus, kuros ir parādītas sevlīdzīguma parametra novērtējuma vidējās vērtības $\hat{H}$ un vidējā absolūtā kļūda ΔH atkarībā no mērogu skaita J pie dažādām segmenta garuma vērtībām (segmenta garums palielinās no kreisās puses uz labo, ejot no augšas uz leju).

Analizējot rezultātus, var pamanīt ka ir sarežģīti precīzi novērtēt sevlīdzīguma parametru zemās (piem., $H = 0,5$, skat. 3.3. att.) un augstās vērtības ($H > 0,9$, 3.5. att.), savukārt vērtības $H = 0,8$ (3.4. att.) robežās var panākt ļoti augstu precizitāti.

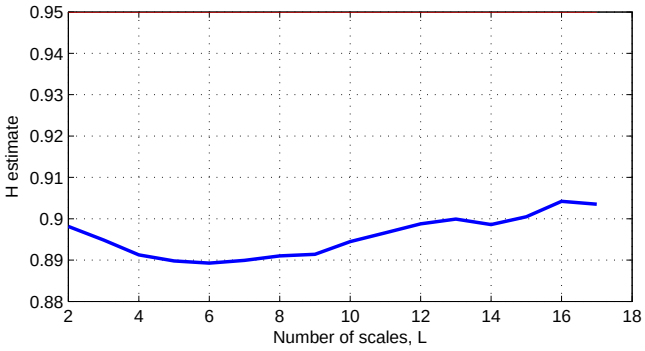


3.3. att. $\hat{H}$ novērtējums pie $H = 0,5$ un segmenta garuma $M = 2^{21}$.



3.4. att. $\hat{H}$ novērtējums pie $H = 0,8$ un segmenta garuma $M = 2^{21}$.

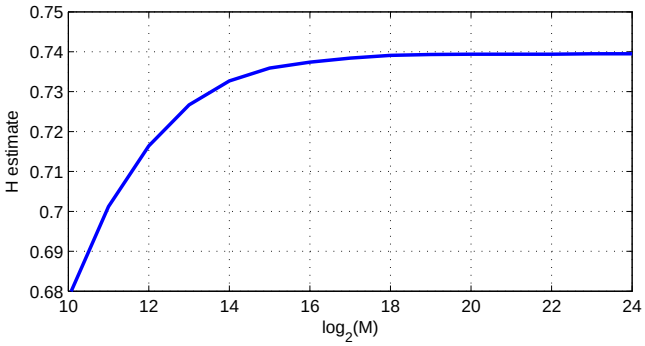
Interesanti, ka vērtībām $0,7 < H < 0,8$ novērojamas optimālās mērogu vērtības, turklāt $H = 0,8$ gadījumā šo optimālo vērtību intervāls ir plašāks (skat. 3.4. att.). Tas liecina par to, ka izmantot maksimālo mērogu skaitu, t.i., aprēķināt diskrēto veivletu transformāciju tādā mērogu skaitā, nav vienmēr vislabākais risinājums. Tomēr, kā tiks parādīts turpmāk, no otras



3.5. att. $\hat{H}$ novērtējums pie $H = 0,95$ un segmenta garuma $M = 2^{21}$.

puses, segmenta garuma palielināšanās gadījumā ir jāizmanto lielākais mērogu skaits precizitātes uzlabošanai.

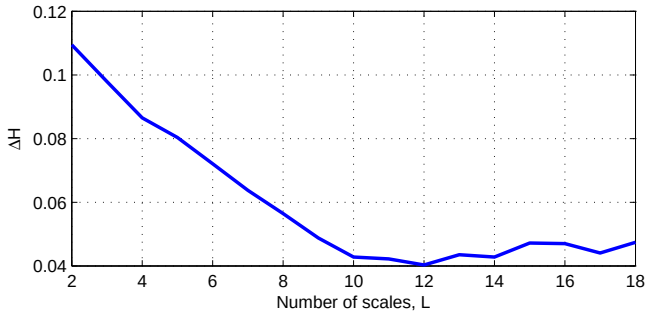
Tālāk tiek apskatīts gadījums, kad mērogu skaits J paliek konstants dažādos segmenta garumos. Tas nozīmē, ka, palielinot segmenta garumu M, jaunie mērogi tiek ignorēti un iepriekšējais mērogu skaits tiek atstāts. Ļoti interesanti, ka tādā gadījumā neatkarīgi no iestādītā H parametra un no segmenta garuma, t.i., visos gadījumos bez izņēmumiem, rezultējošās atkarības raksturs ir tāds, kā parādīts 3.6. att. Tas nozīmē, ka katrā mērogā pastāv precizitātes ierobežojums un, lai palielinātu precizitāti, nepietiek tikai ar segmenta pagarināšanu, bet jāievēro lielāks mērogu skaits. Jāatzīmē, ka tajos eksperimentos, kuri tika veikti divas reizes, šīs ierobežojums atšķiras nelielās robežās (skat. 3.1.tabulu), kas dod pamatojumu uzskatīt, ka šīs ierobežojums ir atkarīgs no konkrētās analizējamā trafika realizācijas.



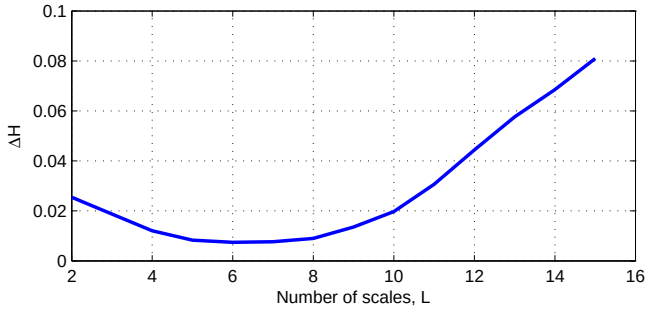
3.6. att. Sevlīdzīguma parametra novērtējums $\hat{H}$ atkarībā no segmenta garuma logaritma pie $H = 0,7$ ievērojot $J = 7$ mērogus.

Analizējot sevlīdzīguma parametra $\hat{H}$ novērtējuma vidējo kļūdu ΔH, var secināt, ka sākot ar noteikto segmenta garumu M_0 vērtībai $H = 0,8$ kļūdas atkarībai no mērogu skaita J ir optimums[2] (skat. 3.8. att.), turklāt šāds atkarības raksturs saglabājās visām segmenta garuma vērtībām $M > M_0$. Vērtībām $H < 0,8$ kļūdas atkarībai ir, kopumā, dilstošais raksturs (piem. 3.7. att.), turpretī vērtībām $H > 0,8$ kļūdas atkarības raksturs, pārsvarā, ir augošais (piem. 3.9. att.).

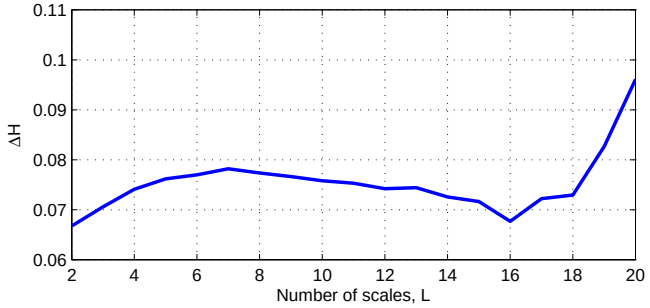
[2]Trafiks ar $H = 0.8$ tika ģenerēts divas reizes, un abās reizēs rezultāti bija analoģiski, interesanti, ka pat minimālais segmenta garums M_0 šajos eksperimentos neatšķiras (skat. pielikumu)



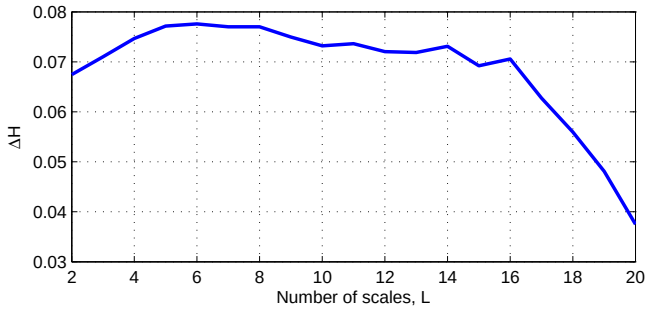
3.7. att. Vidējā absolūtā kļūda ΔH pie $H = 0,6$ un segmenta garuma $M = 2^{21}$.



3.8. att. Vidējā absolūtā kļūda ΔH pie $H = 0,8$ un segmenta garuma $M = 2^{15}$.



3.9. att. Vidējā absolūtā kļūda ΔH pie $H = 0,99$ un segmenta garuma $M = 2^{23}$.



3.10. att. Vidējā absolūtā kļūda ΔH pie $H = 0,99$ un segmenta garuma $M = 2^{23}$ (2. eksperiments).

3.1. tabula. $\hat{H}$ maksimumu salīdzinājums dažādām H vērtībām pie konstanta mērogu skaita J.

Avota H vērtība	**Mērogu skaits, J**	**Eksperimentālais $\hat{H}$ maksimums**	
		1.eksperiments	**2.eksperiments**
$H = 0,8$	3	0,8255	0,8257
	7	0,8045	0,8051
$H = 0,9$	3	0,876	0,8756
	7	0,8663	0,8661
$H = 0,99$	3	0,9226	0,9232
	7	0,9124	0,913

Tomēr, pie ļoti augstajām H vērtībām, ievērojot iepriekš aprakstīto precizitātes ierobežojumu katrā mērogā, lielākajā mērogu skaitā aprēķinot vidējo $\hat{H}$ novērtējuma vērtību pēc garākiem segmentiem, ir iespējams, ka kļūda atkal uzsāks samazināties. Tāda gadījuma piemērs ir parādīts 3.10. att., kurā kļūdas atkarības raksturs ievērojami atšķiras no 3.9. att. parādītā. Taču, tas, pirmkārt ir atkarīgs no konkrētām trafika realizācijām. Otrkārt, šādam mērogu skaitam veidojās tikai 2 segmenti, tātad arī vidējā novērtējuma vērtība tiek aprēķināta tikai pēc 2 novērtējumiem. Lai aprēķinātu vidējo vērtību pēc lielākā novērtējumu skaita ir nepieciešams izanalizēt lielāku trafika apjomu.

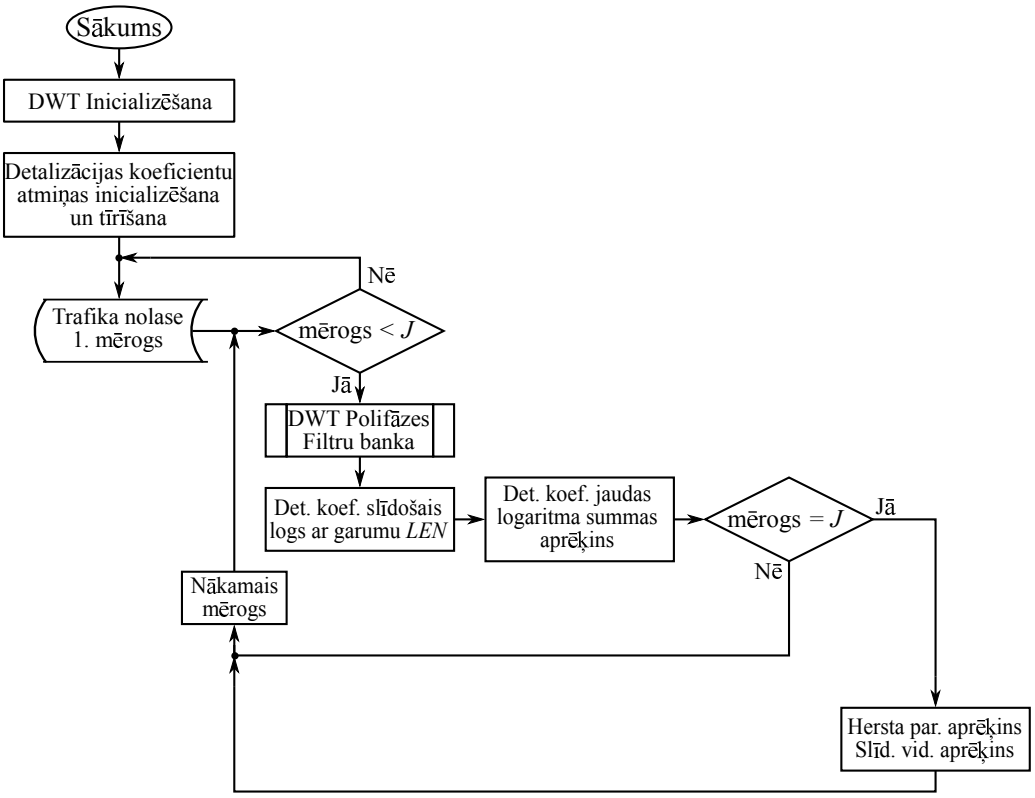
Šeit būtu jāpiezīmē, ka tādam lielajam mērogu skaitam $J = 20$ ir nepieciešami trafika segmenti ar garumu $M = 2^{23}$. Tas ir salīdzinoši liels garums, piemēram, ne pārāk jaunajam datoram ir nepieciešamas 7–10 stundas, lai simulācijā ģenerētu divu tādu segmentu trafika apjomu. Turpretī, divu tādu segmentu analīze notiek daudz ātrāk: 10–15 sekunžu laikā. Turklāt, jārēķinās ar vēl vienu apsvērumu – reālajās sistēmās tāda trafika daudzuma uzkrāšanas laikā situācija var ievērojami mainīties un uz to brīdi, kad tiks aprēķināts šīs novērtējums, tas var būt jau neaktuāls. Nemaz nerunājot par to, ka nav nekādas garantijas, ka tāds novērtējums būs precīzāks, nekā novērtējot pēc īsākajiem trafika segmentiem.

3.4. Trafika sevlīdzīguma parametra novērtēšanas realizācija ar filtru bankām

Vienkāršotā Hersta parametra novērtēšanas algoritma shēma ir parādīta 3.11. att. Par pamatu tiek paņemts diskrētās daudzmērogu veivletu transformācijas algoritms, kas ir aprakstīts 2.nodaļā. Šajā realizācijā tiks izmantota polifāzes filtru banka, taču nepieciešamības gadījumā to var viegli pārveidot arī par kāpņveida filtru banku.

Parādītais 3.11. att. algoritms reāllaikā uz katru trafika nolasi (piemēram, tas varētu būt pakešu skaits laika vienībā) veic diskrēto veivletu transformāciju un saglabā katrā mērogā j detalizācijas koeficientu vidējās jaudas logaritmu. Vidējā vērtība tiek aprēķināta pēc noteiktā detalizācijas koeficientu skaita LEN vienā mērogā. Kā tika iepriekš parādīts, mērogu skaitam J ir liela ietekme uz Hersta parametra novērtēšanas precizitāti un šīs lielums algoritmā var būt uzdots un pat mainīts izpildes gaitā, lai pieskaņotos trafika Hersta parametram. Ja tika noteikts detalizācijas koeficientu vidējās jaudas logaritms, t.i., algoritms apstrādāja pēdējo analizējamo mērogu $j = J$, tad tiek aprēķināta Hersta parametra vērtība. Lai to izdarītu, kā tika iepriekš minēts šajā nodaļā, var pielietot minimālo kvadrātu metodi, lai novērtētu slīpumu vidējās jaudas logaritma atkarībai no mēroga numura.

$$a = \frac{J \sum_{j=1}^{J} j \log_2(D_j) - \sum_{j=1}^{J} j \sum_{j=1}^{J} \log_2(D_j)}{J \sum_{j=1}^{J} j^2 - (\sum_{j=1}^{J} j)^2}. \tag{3.8}$$



3.11. att. Hersta parametra novērtēšanas algoritms ar polifāzes filtru banku, vienkāršotā shēma.

Tādā gadījumā, var novērtēt Hersta parametra vērtību saskaņā ar sekojošo izteiksmi:

$$\hat{H} = \frac{|a+1|}{2}, \tag{3.9}$$

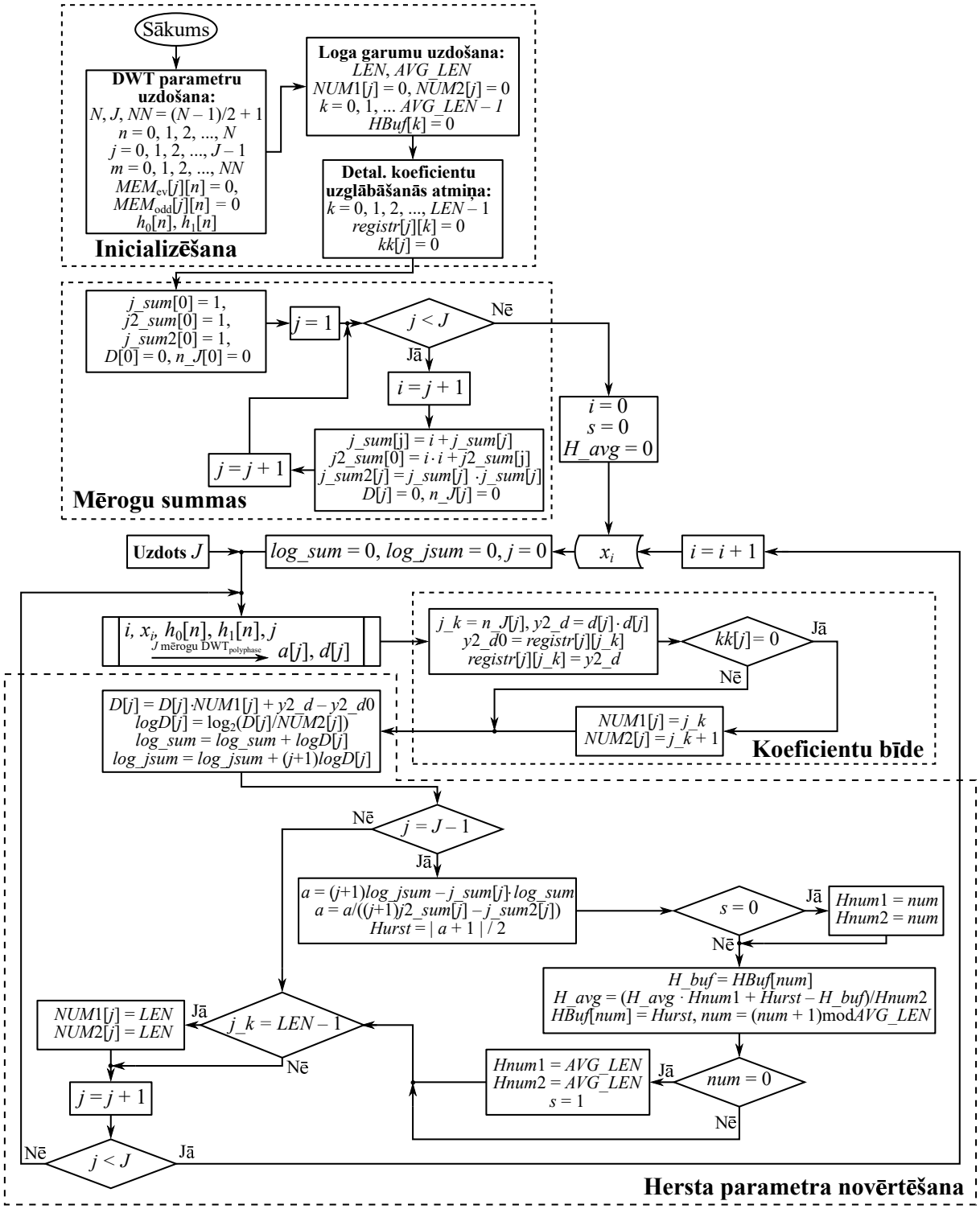
kur a tika novērtēts ar algoritmu saskaņā ar (3.8).

Bez Hersta parametra momentānās vērtības tiek novērtēta arī vidējā vērtība ar slīdošo logu, kura garums sastāda AVG_LEN, saskaņā ar izteiksmi[3]:

$$\mathrm{E}[\hat{H}] = \frac{AVG\_LEN \cdot \mathrm{E}[\hat{H}] + \hat{H}[k] - \hat{H}[k-1]}{AVG\_LEN}, \tag{3.10}$$

kur $\hat{H}$ ir Hersta parametra novērtējums, bet $\mathrm{E}[\hat{H}]$ – šī parametra novērtējuma vidējā vērtība. Šāda Hersta parametra momentānās un vidējās vērtības aprēķināšana tiek atkārtota katrai trafika nolasei un šie divi lielumi arī ir gala mērķis izveidotajam novērtēšanas elementam. Izvērstā algoritma shēma ir parādīta 3.12. att. Programmas kodu $C++$ valodā, kurš realizē šo algoritmu, satur 14.pielikums.

[3]Kā tiks parādīts tālāk, apskatot detalizēto algoritma shēmu, algoritmam uzsākot darbu loga garums palielinās pakāpeniski līdz savai ierobežotajai vērtībai AVG_LEN, un formula nedaudz atšķirsies



3.12. att. Hersta parametra novērtēšanas algoritms ar polifāzes filtru banku, detalizētā shēma.

Algoritma sākumā ir jāveic inicializēšana un dažu lielumu pirmsaprēķinu:

- Diskrētās veivletu transformācijas filtru bankas parametru un atmiņas (aiztures) elementu inicializēšana, kas tiek veikta tieši tādā pašā veidā, kā tas tika darīts J mērogu diskrētajai veivletu transformācijai ar polifāzes filtru banku 2.14. att.;
- Detalizācijas koeficientu un Hersta parametra buferatmiņa un tās garuma uzdošana, kā arī šīs buferatmiņas nullēšana:
  - $Hbuf[k]$ ir Hersta parametra vērtību buferatmiņa vidējās vērtības aprēķinam ar slīdošā loga garumu AVG_LEN;
  - $registr[j][k]$ ir detalizācijas koeficientu kvadrātu buferatmiņa vidējās jaudas aprēķinam pēc LEN pēdējiem koeficientiem;
  - $NUM1[j]$ un $NUM2[j]$ ir divi detalizācijas koeficientu skaitītāji katram mērogam j detalizācijas jaudas pārrēķinam kad $NUM1[j] < NUM2[j] < LEN$;
  - $kk[j]$ – stāvokļa karodziņš katram mērogam, $kk[j] = 1$ kad j mērogā detalizācijas koeficientu skaits pārsniedz lielumu LEN.
- Mērogu summas, mērogu kvadrātu summas un mērogu summas kvadrāta aprēķināšana pēc J uzdotajiem mērogiem (3.8) izteiksmē.

Tā kā (uzskatot ka mērogu skaits J programmas izpildes gaitā paliek nemainīgs[4]) summas no j izteiksmē (3.8) ir konstantas, ir lietderīgi aprēķināt tās iepriekš. Mērogu summa tiek apzīmēta un aprēķināta sekojošajā veidā:

$$j\_sum = \sum_{j=1}^{J} j, \tag{3.11}$$

kur j ir tekošais mērogs intervālā $1 \leq j \leq J$. Nākamā summa tiek aprēķināta no mēroga numuru kvadrātiem j^2 un apzīmēta šādi:

$$j\_sum = \sum_{j=1}^{J} j^2 \tag{3.12}$$

un, pēdējais, summas kvadrāts ir (3.11) vērtības kvadrāts:

$$j\_sum2 = \left(\sum_{j=1}^{J} j \right)^2 = j\_sum^2. \tag{3.13}$$

Tādā veidā notiek sagatavošanās pirms pirmās trafika nolases. Trafika nolases ir iespējams padod no jebkura avota, šajā piemērā tās tiek nolasītas no datorfaila un atspoguļo pieprasījumu skaitu laika vienībā (trafiks tika ģenerēts *Simulink* vidē, kurā laika vienības ir normētas). Padodot trafika nolases x_i ir iespējams arī mainīt mērogu skaitu J, pamatojoties uz $\hat{H}$ novērtējumu ar mēģinājumu palielināt novērtēšanas precizitāti.

Katra trafika nolase x_i tiek apstrādāta ar J mērogu diskrētās veivletu transformācijas polifāzes filtru banku saskaņā ar algoritmu no 2.14. att. Rezultātā tiek saglabāti detalizācijas koeficienti $d[j]$ (kā arī aproksimācijas koeficienti $a[j]$, kuri nav vajadzīgi). Šiem koeficientiem tiek aprēķināta jauda, kāpinot tos kvadrātā un iegūstot:

$$y2\_d = d[j]^2, \tag{3.14}$$

[4]Ja tā nav, ir nepieciešams vai nu aprēķināt visas summas visiem starpmērogiem iepriekš, vai nu jāpārrēķina tas pie katras mērogu skaita J izmaiņas.

kura tiek saglabāta buferatmiņā $registr[j][j\_k]$ veicot bīdes operāciju. Līdzīgi kā tas bija inversās diskrētās veivletu transformācijas gadījumā, šeit ir jāzina tikai pirmais un pēdējais bīdes atmiņas elementi, un līdz ar to, var izmantot tādu pašu pieeju, kāda tika apskatīta 2.6..sadaļā, saglabājot iepriekšējo vērtību mainīgajā $y2\_d0$. Šīs lielums ir jāatņem, lai detalizācijas koeficientu jauda būtu aprēķināta tikai pēc pēdējiem LEN koeficientiem. Ja stāvokļa karodziņš analizējamajā mērogā j ir $kk[j] = 0$, tad LEN koeficienti vēl nav uzkrāti un vidējās slīdošā loga aprēķinā ir jānosaka iepriekšējās vērtības loga garumu $NUM1[j]$ un jaunās vērtības loga garumu $NUM2[j]$. Ja $kk[j] = 1$, tad $NUM1[j] = NUM2[j] = LEN$, pretējā gadījumā jāizmanto $n_J[j]$ koeficientu skaitītāju un $NUM2[j] = NUM1[j] + 1$. Tādā gadījumā vidējās koeficientu jaudas logaritms tiek aprēķināts atbilstoši izteiksmei:

$$logD[j] = \log_2 \left(\frac{D[j] \cdot NUM1[j] + y2\_d - y2\_d0}{NUM2[j]} \right). \tag{3.15}$$

Saskaņā ar (3.15) izteiksmes rezultātu tiek aprēķinātas vēl divas summas, kas nepieciešamas, lai veiksmīgi novērtēt $\log_2 D[j]$ atkarības no j slīpumu saskaņā ar (3.8):

$$log\_sum = \sum_{j=1}^{J} logD[j] \tag{3.16}$$

un

$$log\_jsum = \sum_{j=1}^{J} j \cdot logD[j]. \tag{3.17}$$

Tādā gadījumā, pēdējā mērogā J tiek novērtēts slīpums $\log_2 D[j]$ atkarībai no j saskaņā ar (3.8), izmantojot (3.11)-(3.13), (3.16) un (3.17). Pēc (3.9) tiek noteikts Hersta parametra momentānais novērtējums $\hat{H}$, kuram turpmāk tiek aprēķināta vidējā vērtība pēc pēdējām AVG_LEN novērtējuma vērtībām. Slīdošais logs tiek realizēts analoģiski detalizācijas koeficientu slīdošajam logam, aprēķinot nolases numura dalījuma atlikumu pēc modAVG_LEN. Ja $\hat{H}$ novērtējumu skaits ir mazāks par AVG_LEN lielumu, par ko liecina karodziņš $s = 0$, ir jāizmanto divi mainīgie: $Hnum1$ un $Hnum2$, kur $Hnum1$ ir iepriekšējās vidējās vērtības $\mathrm{E}[\hat{H}]$ novērtējumu skaits un $Hnum2 = Hnum1 + 1$ ir tekošo novērtējumu skaits. Tādā gadījumā, vidējā vērtība tiek noteikta saskaņā ar izteiksmi:

$$\mathrm{E}[\hat{H}] = H\_avg = \frac{H\_avg \cdot Hnum1 + Hurst - H\_buf}{Hnum2}, \tag{3.18}$$

kur $Hurst$ ir tekošais novērtējums saskaņā ar (3.9) un (3.8), savukārt H_buf ir novērtējums ar aizturi AVG_LEN, kuru ievieš buferatmiņas masīvs $HBuf[num]$.

Turpretī, ja $\hat{H}$ novērtējumu skaits $num \geq AVG\_LEN$, tad karodziņš $s = 1$ un $Hnum1 = = Hnum2 = AVG\_LEN$, savukārt vidējās vērtības aprēķins notiek saskaņā ar izteiksmi (3.10). Algoritms tiek atkārtots katrai trafika nolasei x_i.

3.5. Nobeigums

Šajā nodaļā tika izveidots un izpētīts Hersta parametra H novērtēšanas elements un tika veikti divi eksperimenti sevlīdzīguma pakāpes novērtēšanai:

- ja Hersta parametrs mainījās robežās $0,5 < H < 0,99$ ar soli 0,1;
- ja Hersta parametrs mainījās robežās $0,8 < H < 0,99$ ar soli 0,05.

Apstrāde tika veikta izmantojot *Matlab* ar līdzīgu algoritmu, kas tika aprakstīts .sadaļā. Dati tika apstrādāti "segmentos", kur viens segments atbilst lielumam $2^{J_{\max}}$, $J_{\max}$ – maksimālais analizējamo mērogu skaits. Kaut arī diskrētā veivletu transformācija tiek aprēķināta reāllaikā saskaņā ar .sadaļā aprakstīto algoritmu, Hersta parametra H novērtēšanai ir nepieciešams veikt J mēroga diskrēto veivletu transformāciju, kura var tikt aprēķināta tikai uz katru pēdējo no 2^J nolasēm. Tāpēc pārāk lielas analizējamo mērogu skaita vērtības $J_{\max}$ nav vēlamas, jo Hersta parametra H novērtēšana notiks retāk.

Tajā pašā laikā tika konstatēts, ka izmantot visu $J_{\max}$ mērogu koeficientus nav lietderīgi Hersta parametra novērtējuma $\hat{H}$ absolūtās kļūdas ziņā. Pastāv noteiktais mērogu skaits $J < J_{\mathbf{max}}$, pie kura novērtēšanas kļūda ΔH ir vismazākā. Tas norāda uz to, ka izmantojamo mērogu skaitu J novērtēšanai ir adaptīvi jāpieskaņo atkarībā no Hersta parametra novērtējuma $\hat{H}$ vērtības. Tāda iespēja izveidotajā novērtēšanas algoritmā pastāv.

Nodaļas beigās tika aprakstīts Hersta parametra H novērtēšanas elementa algoritms un piedāvāta detalizēta algoritma shēma, saskaņā ar kuru tika izveidota testēšanas programma, kuras tekstu satur 14.pielikums. Programma tika pārbaudīta ar tām pašam trafika realizācijām, kas tika analizētas nodaļas tekstā. Analīzes mērogu izvēles jautājums saglabājās, par to, piemēram, liecina 3.2.tabulas dati, kuros apkopotas Hersta parametra novērtējuma $\hat{H}$ un vidējās novērtējuma vērtības $\mathrm{E}[\hat{H}]$ dažādam mērogu skaitam pie maksimālā mērogu skaita $J_{\max} = 8$ trafikam ar sevlīdzīguma parametru $H = 0,9$. Diskrētās veivletu transformācijas veikšanai ar filtru bankām tika izvēlēti Dobeši-3 veivleti. Detalizācijas koeficientu loga garums $LEN = 1000$, vidējo vērtību loga garums $AVG\_LEN = 1000$. Trafika masīva garums sastāda 5000 vērtības, 3.2. tabulā ir dots pēdējais novērtējums.

3.2. tabula. Sevlīdzīguma parametra novērtējums dažādiem mērogiem j, $J_{\max} = 8$.

Mērogs, j	$\hat{H}$	$\mathrm{E}[\hat{H}]$	**Izpildes laiks**	**DWT izpildes laiks**
2	0,735	0,808	0,021 s	0,0197 s
3	0,814	0,817	0,0203 s	0,0198 s
4	0,860	0,844	0,02 s	0,0195 s
5	0,854	0,872	0,0202 s	0,0203 s
6	0,873	0,919	0,021 s	0,0198 s
7	0,839	0,882	0,021 s	0,02 s
8	0,853	0,859	0,021 s	0,0204 s

Kā var redzēt no tabulas, ja tiek novērtēta $H = 0,9$ vērtība, tad visprecīzākais rezultāts būtu $J = 6$ mērogos no $J_{\max} = 8$ analizētajiem mērogiem.

3.2. tabulas dati tika iegūti personālajam datoram ar *Core i5-4690* procesoru, takts frekvence 3,2 GHz. Kā var redzēt no rezultātiem, salīdzinājumā ar vienkāršo diskrētās veivletu transformācijas aprēķinu, izpildes laiks palielinās neievērojami.

Tomēr, šeit ir jārēķinās ar to, ka trafika vērtības tika nolasītas no cietā diska nesēja, kura veiktspēja ir daudz zemāka par procesora veiktspēju (dators bija aprīkots ar ātrdarbīgo *SSD* disku). Lai novērtētu paša algoritma veiktspēju un laika zudumus, kuri patērēti ar diska nolasīšanas operācijām tika veikta diskrētu veivletu transformācija no nolases numura i (mainīgais reģistru atmiņā) un novērtēts tāda "trafika" sevlīdzīguma koeficients. Rezultāti ir šādi:

- Diskrētās veivletu transformācijas izpildes laiks sastāda 0,000239273 s;
- Hersta parametra novērtēšanas izpildes laiks[5] sastāda 0,000564183 s.

Kas attiecās uz nepieciešamo saglabāto koeficientu skaitu LEN, tad šeit arī ir saskatāma atkarība no Hersta parametra H vērtības, kura raksturo trafiku. Dažas rekomendācijas ir pieejamas literatūrā, piemēram, [55] piedāvā šādus nolašu skaitļus dažādiem trafika veidiem:

[5]Kurš, acīmredzami, iekļauj sevī arī diskrētas veivletu transformācijas izpildes laiku kā sastāvdaļu.

- *VBR* video plūsmām – vismaz 20 000 nolases;
- *LAN* trafikam – vismaz 100 000 nolases;
- frakcionālajiem Gausa trokšņiem – vismaz 20 000 nolases.

No šiem rezultātiem skaidri redzams, ka patiešām lielāka laika daļa no 3.2. tabulas mērījumiem tiek patērēta tieši uz trafika vērtību nolasīšanas no cietā diska. Ja šīs vērtības nāktu no cita avota, piemēram, tās paziņotu maršrutētāja monitoringa sistēma, aprēķins notiktu daudz ātrāk. Rezultāti parāda, ka, kopumā, Hersta parametra novērtēšanai ir nepieciešams gandrīz tāds pats laiks, kāds nepieciešams diskrētās veivletu transformācijas veikšanai ar polifāzes filtru banku.

Nodaļa 4

Pakešu zudumu varbūtības analīze ģenerētajam sevlīdzīgajam trafikam $P/M/1/K$ un $G/M/1/K$ simulācijas modeļiem

Šajā nodaļā aprakstīti vairāki eksperimenti, kuros tika veiktas simulācijas ar trafiku pie dažādām noslodzes koeficientu ρ un Hersta parametra H vērtībām. Galvenais eksperimentu sērijas mērķis ir noskaidrot $G/M/1$ un $P/M/1$ modeļa vidējo $E[R]$ un maksimālo $\max(R)$ rindas garumu, salīdzināt šos divus lielumus un izpētīt to atkarību no Hersta parametra H vērtības. Turpmāk šīs iegūtās vidējā rindas garuma vērtības $E[R]$ tiks izmantotas $G/M/1/K$ un $P/M/1/K$ modeļos ar ierobežotu buferatmiņu. Buferatmiņas apjomu ierobežo vidējais rindas garums $E[R]$ vai tā kārtnie skaitļi, piemēram, trīskārtējais rindas vidējais garums $3E[R]$, utt. Simulācijas procesā tiek novērtēta pakešu zudumu varbūtība P_{loss} un pēc eksperimentiem tiek analizēta tās atkarība no Hersta parametra H.

4.1. $P/M/1$ simulācijas modeļa vidējā un maksimālā rindas garuma novērtēšana

Kā tika iepriekš minēts, viens no piemērotākajiem varbūtības blīvuma sadalījuma likumiem intervāliem starp pieprasījumiem ir Pareto sadalījuma likums. Izmantojot dažādus simulācijas rīkus, tādus kā, piemēram, *GPSS* vai *Matlab Simulink*, var veidot $P/M/1$ modeli ar neierobežotu sistēmas bufeartmiņas apjomu. Tādas simulācijas tika izveidotas abās vidēs ar nolūku izpētīt rindas garuma atkarību no Hersta parametra un noslodzes koeficienta.

Izveidotā modeļa parametri ir šādi:

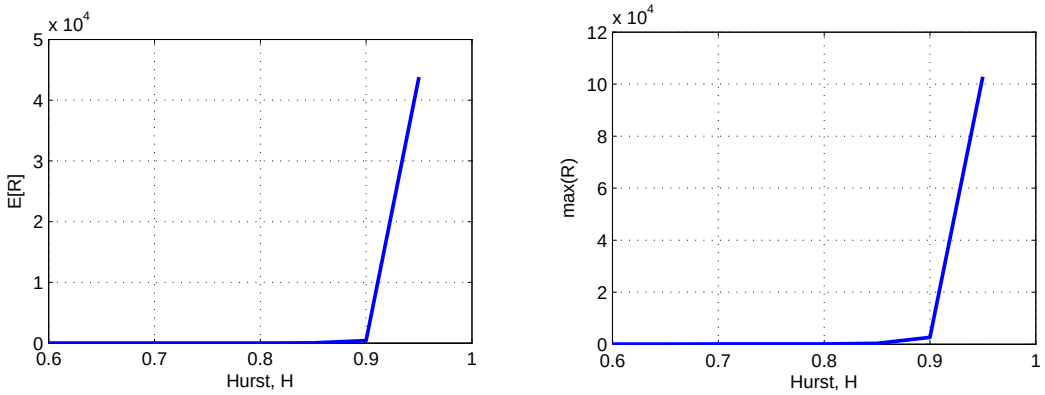
- Trafika intensitāte $\lambda = 100$ paketes/s, jeb vidējais laika intervāls starp pieprasījumiem ir $T_{\text{a}} = 1/120$ s. Intervāli sadalīti pēc Pareto likuma.
- Trafika sevlīdzīguma (Hersta) parametrs H tiek uzstādīts robežās no 0,6 līdz 0,95, mainot to katrā simulācijā ar soli 0,05.
- Kopējais pieprasījumu skaits simulācijā sastāda 1 000 000.
- Apstrādes mezgla veiktspēja $\mu = 125$ paketes/s, jeb vidējais apstrādes laiks sastāda $T_{\text{s}} = 1/125$ s. Apstrādes intervāli sadalīti pēc eksponenciālā likuma.

Tādā veida apskatāmais modelis tika pētīts pie noslodzes koeficienta $\rho = \lambda/\mu = 0,8$ vērtības. Šāda vērtība tika izvēlēta tāpēc, ka tā atbilst reālo sistēmu noslodzes koeficientu intervāla viduspunktam. Noslodzes koeficientam reālām sistēmām, lai tās strādātu efektīvi, ir jābūt $0,7 \leq \rho \leq 0,9$ robežās. Rezultātā pie katras Hersta parametra H vērtības tika nomērīts vidējais rindas garums $E[R]$ un maksimālais rindas garums $\max(R)$. Eksperimentu rezultāti ir apkopoti 4.1. tabulā.

4.1. tabula. *GPSS* $P/M/1$ modeļa simulācijas dati noslodzes koeficientam $\rho = 0,8$ un dažādām H parametra vērtībām.

Hersta parametrs	**Rindas vid. garums**	**Rindas maks. garums**
0,6	3,45	46
0,65	4,13	52
0,7	5,28	59
0,75	7,54	69
0,8	13,23	103
0,85	36,75	305
0,9	410,68	2621
0,95	43810,35	102830

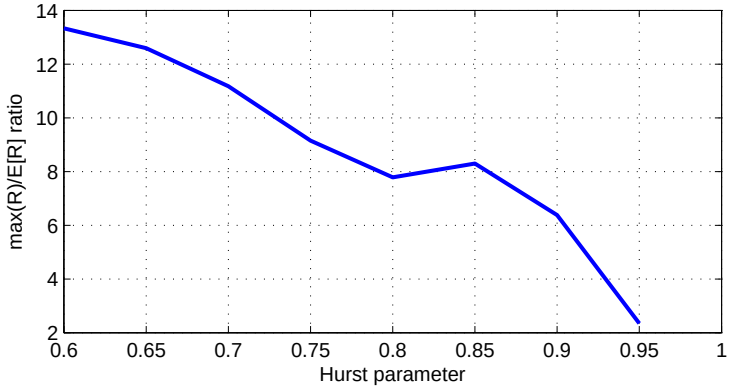
Zemāk, 4.1. att., ir grafiski parādītas rindas garuma atkarības no Hersta parametra. Kopējais raksturs šādām atkarībām ir eksponenciālais. Turpmāk, veidojot tādu pašu simulāciju *Simulink* vidē ar mazāko Hersta parametra izmaiņu soli tiks iegūti līdzīgi rezultāti.



a b

4.1. att. *GPSS* $P/M/1$ modeļa rindas garums pie $\rho = 0,8$: a) vidējais, b) maksimālais.

Lai uzzinātu, cik vienai vai otrai Hersta parametra vērtībai vidējais rindas garums $E[R]$ pārsniedz maksimālo $\max(H)$ un noskaidrotu, vai šī attiecība ir atkarīga no Hersta parametra, tika konstruēts tādas attiecības grafiks atkarībā no Hersta parametra lieluma, kurš ir parādīts 4.2. att.



4.2. att. *GPSS* $P/M/1$ modeļa maksimālā rindas garuma $\max(R)$ attiecība pret vidējo rindas garumu $E[R]$.

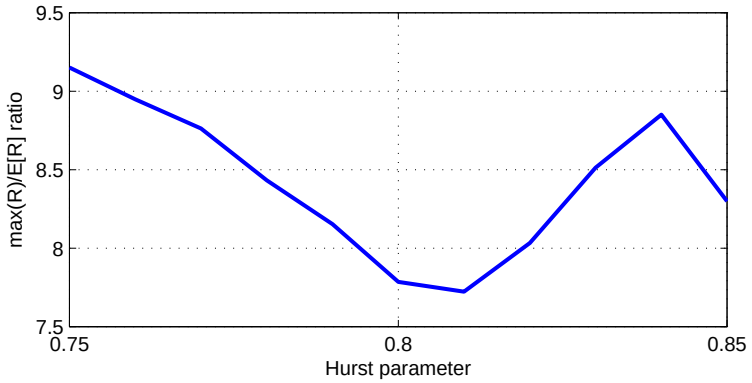
4.2. att. parādītā attiecības atkarība no Hersta parametra samazinās pie augstākām Hersta parametra vērtībām. No tā var secināt, ka jo augstāka ir trafika sevlīdzīguma pakāpe, jo tuvāks ir rindas vidējais garums maksimālajam[1]. Tomēr šī atkarība nav monotona, kā redzams $H = 0,8$ parametra apgabalā, līknes raksturs mainās un attiecība sāk atkal pieaugt. Lai sīkāk izpētītu šo parādību, tika atkārtots eksperiments Hersta parametru intervālā $0,75 \leq H \leq 0,85$, mainot šo lielumu ar soli 0,01. Mērījumu rezultāti tika apkopoti 4.2. tabulā un rindas garumu attiecības grafiks ir parādīts 4.3. att.

4.2. tabula. *GPSS* $P/M/1$ modeļa simulācijas dati noslodzes koeficientam $\rho = 0,8$ un dažādām H parametra vērtībām.

Hersta parametrs	**Rindas vid. garums**	**Rindas maks. garums**
0,75	7,54	69
0,76	8,27	74
0,77	9,13	80
0,78	10,2	86
0,79	11,53	94
0,8	13,23	103
0,81	15,41	119
0,82	18,3	147
0,83	22,32	190
0,84	28,02	248
0,85	36,75	305

Kā var redzēt no grafika 4.3. att., līknes dilstošais raksturs mainās uz augošo apmēram $H = 0,81$ vērtībā un atpakaļ uz dilstošo $H = 0,84$ vērtībā. Tieši tāda anomālā uzvedība $H = 0,8$ apgabalā tiks pētīta turpmākajos eksperimentos vairākiem trafika veidiem pie dažādām noslodzes koeficienta ρ vērtībām. Galu galā tiks izpētīta arī tādas parādības ietekme modelī ar ierobežoto buferatmiņu, $P/M/1/K$ un $G/M/1/K$, novērtējot pakešu zudumu varbūtību P_{loss}.

[1] Šajā eksperimentā tika veikti 1 000 000 pieprasījumi. Turpmākajos eksperimentos šis skaitlis tika palielināts, taču tendence saglabājās.

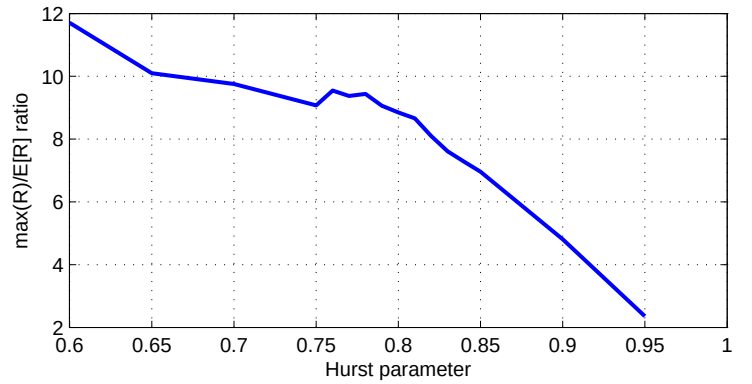


4.3. att. *GPSS* $P/M/1$ modeļa maksimālā rindas garuma $\max(R)$ attiecība pret vidējo rindas garumu $E[R]$ Hersta parametra vērtībām intervālā $0,75 \leq H \leq 0,85$ ar soli 0,01.

Sastādot līdzīgo *Simulink* modeli, tika veiktas tādas pašas simulācijas kā *GPSS* vidē un arī šajā gadījumā novērtēts vidējais rindas garums $E[R]$ un maksimālais rindas garums $\max(R)$. Šajā simulāciju sērijā Hersta parametrs tika mainīts ar dažādu soli:

- intervālā $0,6 \leq H \leq 0,75$ solis sastāda 0,05;
- intervālā $0,75 \leq H \leq 0,85$ solis sastāda 0,01;
- intervālā $0,85 \leq H \leq 0,95$ solis sastāda 0,05.

Rezultāti tika apkopoti līdzīgajā veidā, kā tas tika darīts 4.1. un 4.2. tabulās. Grafiskā atkarība starp maksimālo/vidējo rindas garumu attiecību un Hersta parametru ir parādīta 4.4. att. visām Hersta parametra vērtībām. Salīdzinot 4.2. att., 4.3. att. un 4.4. att. grafikus, var konstatēt, ka arī *Simulink* modelī Hersta parametra intervālā $0,75 \leq H \leq 0,85$ mainās līknes raksturs līdzīgajā veidā. Kopējā tendence samazināties attiecībai maksimālais/vidējais rindas garums saglabājās, kas nozīmē ka *Simulink* un *GPSS* $P/M/1$ modeļi ir vienlīdzīgi, un turpmākajos pētījumos tiks izmantots tieši *Simulink* modelis[2]. Tas dod iespēju samazināt Hersta parametra izmaiņu soli, kā arī mainīt trafika ģenerācijas modeli, piemēram, izmantojot Veibula sadalījumu intervāliem starp pieprasījumiem vai arī *ON/OFF* Pareto trafika modeli – un jebkuru citu trafika ģenerēšanas modeli, izveidojot attiecīgo funkciju.



4.4. att. *Simulink* $P/M/1$ modeļa maksimālā rindas garuma $\max(R)$ attiecība pret vidējo rindas garumu $E[R]$ Hersta parametra vērtībām intervālā $0,6 \leq H \leq 0,95$ ar mainīgo soli.

[2]Tas ir pamatojams ar to, ka *Matlab* dod iespēju veidot veselas simulācijas programmas, kas var mainīt nepieciešamus parametrus pēc uzdotā skripta bez turpmākās lietotāja iejaukšanās simulācijas procesā. *GPSS* simulācijās ir jāveic liels roku darbs, uzstādot simulācijas parametrus katrā eksperimentā. Arī rezultātu apkopošana *Matlab* vidē ir veicama ar programmas palīdzību.

4.2. $P/M/1$ modelis ar *ON/OFF* trafiku un $G/M/1$ modelis ar Veibula sadalījumu

Nākamajā eksperimentu sērijā tika izpētīts trafika modelis ar tādiem pašiem parametriem, t.i., ar noslodzes koeficientu $\rho = 0,8$. Savukārt Hersta parametrs tika mainīts ar soli 0,005, kas dod iespēju precīzāk novērtēt maksimālā/vidējā rindas garuma attiecības atkarību no Hersta parametra vērtības. Tika izpētīti šādi *Simulink* modeļi:

- $P/M/1$ modelis, kas tika aprakstīts iepriekš;
- $P/M/1$ modelis ar *ON/OFF* trafika raksturu, kur trafika vienas grupas (burst) laika intervāliem ir eksponenciālais sadalījums ar vidējo laiku $E[T_\mathrm{p}] = 2$ s;
- $G/M/1$ modelis ar Veibula sadalījumu intervāliem starp pieprasījumiem.

Simulāciju rezultāti ir parādīti 4.5. att. grafikos visām apskatītajām simulacijām. Kā var redzēt no grafikiem, visos gadījumos $H = 0,8$ apgabalā attiecībai maksimālais/vidējais rindas garums mainās līknes raksturs.

Vislabāk tas ir redzams 4.5b. att. ar *ON/OFF* veida trafiku, kuram intervāli starp pieprasījumiem mainās saskaņā ar Pareto sadalījumu. Tāds trafiks ir līdzīgs trafikam tīklos, kurā ir gan pauzes, gan veselas pakešu grupas (burst). Kā var redzēt no 4.5b. att., sākot ar vērtību $H = 0,8$, likumsakarības izmaiņas notiek biežāk, nekā līdz šai vērtībai. Tomēr, kopējā tendence samazināties paliek – izņemot intervālu $0,8 \leq H \leq 0,85$, kurā ir ievērojams pieaugums. Pārējos gadījumos šādai likumsakarībai raksturs ir tuvāks monotonās samazināšanas raksturam. Tas dod pamatojumu uzskatīt, ka tieši *ON/OFF* trafika struktūra var ievērojami mainīt attiecību starp maksimālo rindas garumu $\max(R)$ un vidējo rindas garumu $E[R]$. Bez tam, salīdzinot 4.5a. att. un 4.5c. att. grafikus, var redzēt, ka gadījumā ar Veibula sadalītajiem intervāliem attiecība samazinās lēnāk, nekā ar Pareto sadalītajiem intervāliem.

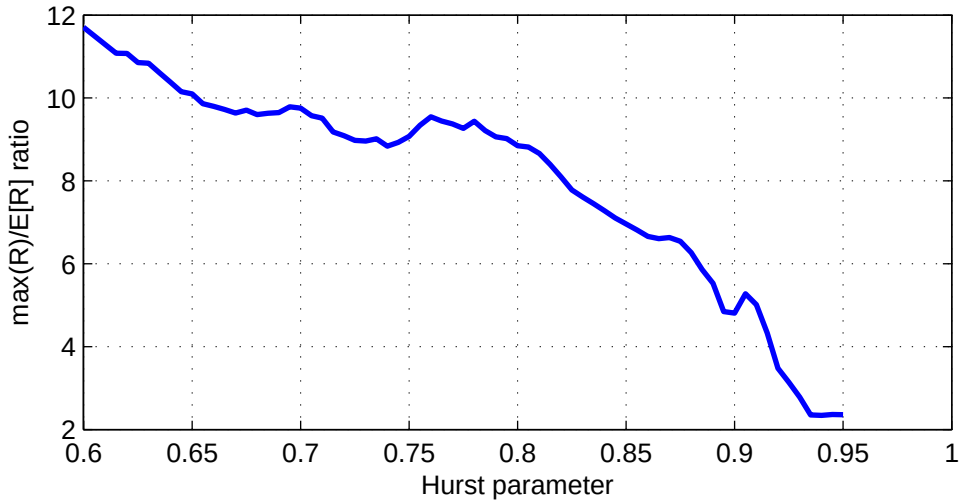
Aprakstītie eksperimenti tika veikti nelielajam pieprasījumu skaitam, apmēram 100 000, katrā grafika punktā. Tas ir pamatojams ar lielu kopējo datu apjomu pie tāda nelielā soļa kā 0,005. Tādi paši rezultāti tika iegūti arī dažādiem noslodzes koeficientiem ρ, kura vērtības tika mainītas no 0,6 līdz 0,9 ar soli 0,1. Šajā gadījumā Hersta parametra H solis atkal bija mainīgs un mainījās līdzīgi kā pirmajos eksperimentos, t.i., šādi:

- intervālā $0,6 \leq H \leq 0,75$ solis sastāda 0,05;
- intervālā $0,75 \leq H \leq 0,85$ solis sastāda 0,01;
- intervālā $0,85 \leq H \leq 0,95$ solis sastāda 0,05.

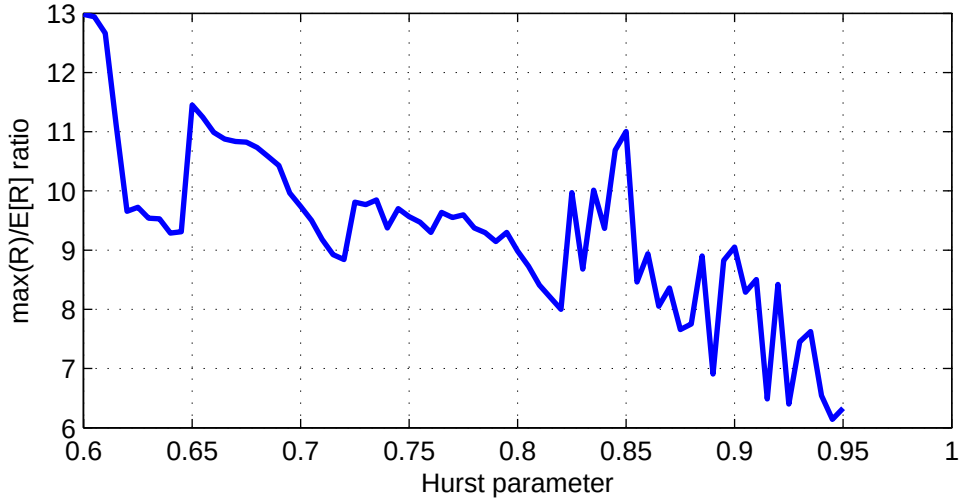
Savukārt noslodzes koeficients $\rho = \lambda/\mu$ tika mainīts, iestādot trafika intensitātes λ vērtību. Apstrādes mezgla veiktspēja μ visos eksperimentos bija nemainīga un sastāda $\mu = 125$ paketes/s. Trafika intensitātei šajos četros eksperimentos bija uzstādītās šādas vērtības:

- 1. simulācijā $\lambda = 75$ paketes/s, bet noslodzes koeficients $\rho = 0,6$;
- 2. simulācijā $\lambda = 87,5$ paketes/s, bet noslodzes koeficients $\rho = 0,7$;
- 3. simulācijā $\lambda = 100$ paketes/s, bet noslodzes koeficients $\rho = 0,8$;
- 4. simulācijā $\lambda = 112,5$ paketes/s, bet noslodzes koeficients $\rho = 0,9$.

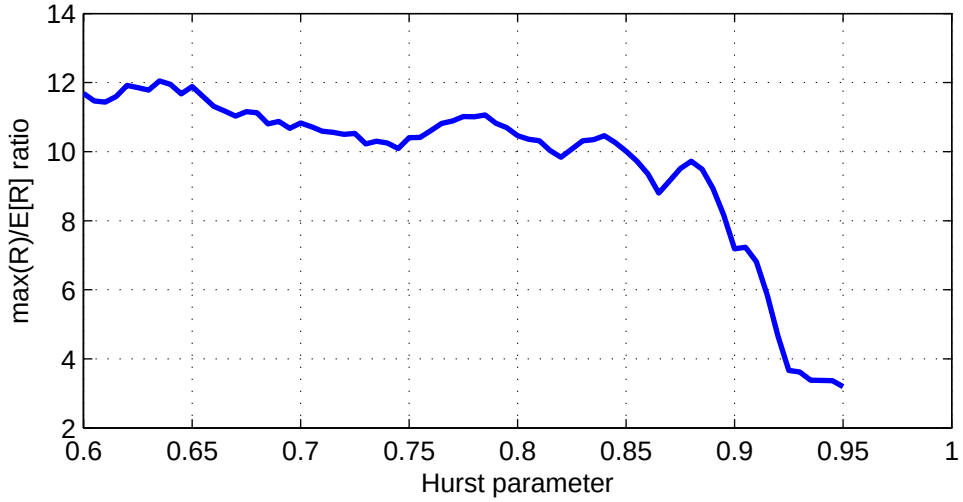
Visas simulācijas tika veiktas $P/M/1$, $P/M/1$ ar *ON/OFF* trafiku un $G/M/1$ modeļiem, izmantojot *Matlab* sastādīto programmu. Šajā programmā tika mainītas parametru vērtības, tika veiktas visu modeļu simulācijas un saglabāti rezultāti failā, lai veiktu turpmāko analīzi. Pēc šiem rezultātiem tika konstruēti grafiki, kas parādīti 4.6. att.



a

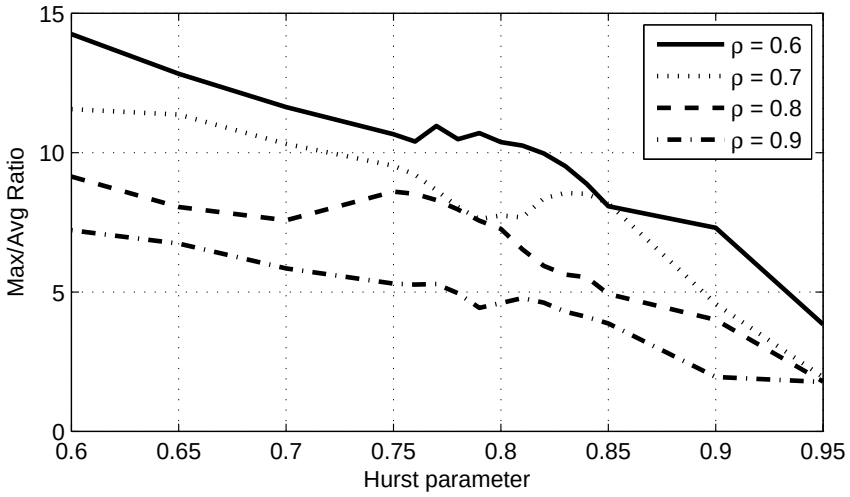


b

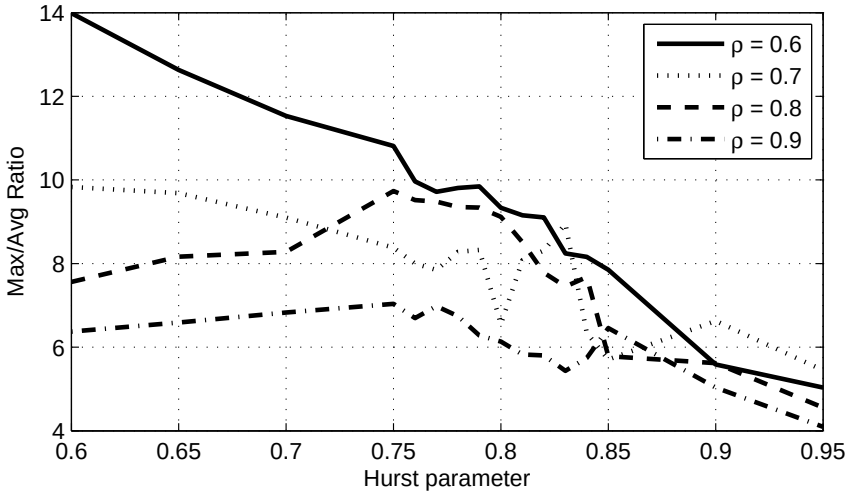


c

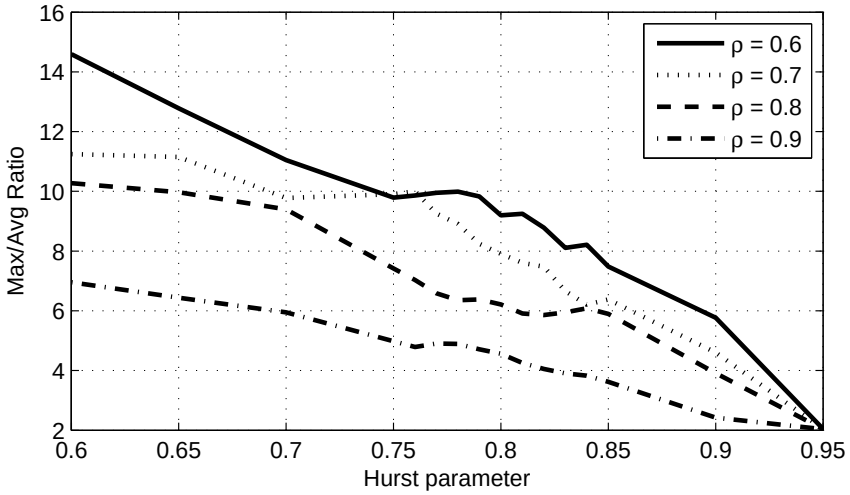
4.5. att. Simulink modeļa maksimālā rindas garuma $\max(R)$ attiecība pret vidējo rindas garumu $E[R]$ Hersta parametra vērtībām intervālā $0.6 \leq H \leq 0.95$ ar soli 0.005: a) $P/M/1$, b) $P/M/1$ ON/OFF, c) $G/M/1$.



a



b



c

4.6. att. Simulink modeļa maksimālā rindas garuma max(R) attiecība pret vidējo rindas garumu $E[R]$ atkarībā no Hersta parametra H dažādiem noslodzes koeficientiem ρ: a) $P/M/1$, b) $P/M/1$ *ON/OFF*, c) $G/M/1$.

4.3. Simulācijas modeļi ar ierobežotu buferatmiņu

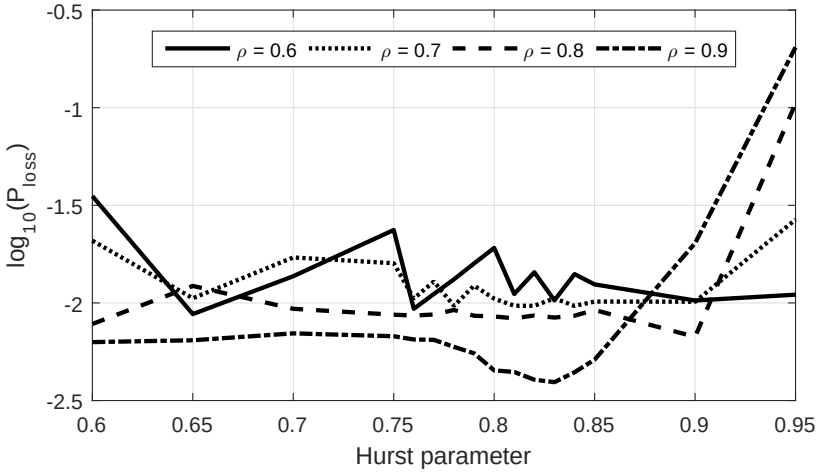
Lai varētu novērtēt pakešu zudumu varbūtību P_{loss}, modelī ir jāierobežo buferatmiņas apjoms. Šo apjomu ir jāiestāda simulācijā kā konkrēto skaitli. Veiktie pētījumi norāda uz to, ka buferatmiņas apjoms ir atkarīgs gan no Hersta parametra H, gan no noslodzes koeficienta ρ. [90] piedāvā formulu buferatmiņas apjoma novērtēšanai, taču kā parāda iepriekšējo eksperimentu rezultāti, tā nav pielietojama. Darbā [42] ir sniegtas rekomendācijas attiecībā uz buferatmiņas apjoma izvēli, taču šajā eksperimentu sērijā buferatmiņas apjoms tiks izvēlēts vienāds ar vidējo rindas garumu $E[R]$, kas tiek sareizināts ar konstantu reizinātāju k saskaņā ar (4.1), piemēram, eksperimentos zemāk tas būs vienāds ar 1, 3, vai 5:

$$K = kE[R], \text{ kur } k = 1, 2, 3, ... \tag{4.1}$$

Jāatzīmē, ka apskatāmās vērtības ir tuvas faktiski rekomendējamajām, piemēram, [35] piedāvāts $M/M/1/K$ un $M_x/M/1/K$ modeļiem ņemt $4E[R]$ buferatmiņas apjomu.

Spriežot pēc iepriekšējiem rezultātiem 4.4. att., tādai buferatmiņas apjoma palielināšanai vajadzētu samazināt pakešu zudumu varbūtību P_{loss}. Jo lielāka ir Hersta parametra vērtība H, jo mazāk atšķiras vidējais rindas garums $E[R]$ no maksimālā $\max(R)$. Piemēram, $G/M/1$ modelim, 4.5c. att. grafikā, pie $H = 0,95$ tāda attiecība sastāda apmēram 4. Ja izvēlētais buferatmiņas apjoms sastāda $K = 5E[R]$, tad pakešu zudumu varbūtībai pie noslodzes koeficienta $\rho = 0,8$ un Hersta parametra $H = 0,95$ jābūt nelielai[3].

Eksperimentu sērija tika veikta *Matlab Simulink* vidē. Tomēr, lai pārliecinātos par rezultātu pareizību, viens no eksperimentiem tika atkārtots arī *GPSS World* simulācijas vidē. Buferatmiņas apjoma uzdošanai ir jānoskaidro vidējais rindas garums $E[R]$ visām H un ρ vērtībām, pie kurām tiks veikti eksperimenti. Ar šo mērķi tika veiktas simulācijas $P/M/1$, $P/M/1$ ar *ON/OFF* trafiku un $G/M/1$ modeļiem *Simulink* vidē. Rezultāti ir apkopoti 4.3. tabulā un izmantoti buferatmiņas apjoma K uzdošanai saskaņā ar (4.1) gan *GPSS*, gan *Simulink* modeļos.



4.7. att. *GPSS* $P/M/1/K$ pakešu zudumu varbūtības logaritma $\lg P_{\text{loss}}$ atkarība no Hersta parametra H dažādām noslodzes koeficienta ρ vērtībām.

[3]Saprotams, ka šeit tiek modelēti gadījumrakstura procesi un atteikumi apstrādē ir joprojām iespējami, kaut arī it kā ir paņemta buferatmiņa ar apjomu, kas pārsniedz maksimālo rindas garumu. Īpaši tas attiecās uz $P/M/1/K$ modeli ar *ON/OFF* veida trafiku.

4.3. tabula. Simulink $P/M/1$, $P/M/1$ *ON/OFF* un $G/M/1$ modeļu vidējais garums dažādām Hersta parametra H un noslodzes koeficienta ρ vērtībām.

Hersta parametrs	**P/M/1**				***P/M/1 ON/OFF***				***G/M/1***			
	$\rho = 0,6$	$\rho = 0,7$	$\rho = 0,8$	$\rho = 0,9$	$\rho = 0,6$	$\rho = 0,7$	$\rho = 0,8$	$\rho = 0,9$	$\rho = 0,6$	$\rho = 0,7$	$\rho = 0,8$	$\rho = 0,9$
0,6	0,98	1,9	4,05	11,23	1	1,93	3,84	9,57	1,78	2,85	5,06	12,37
0,65	1,17	2,29	4,97	14,39	1,19	2,27	4,65	11,38	2,27	3,59	6,32	15,38
0,7	1,46	2,91	6,47	19,83	1,47	2,75	5,92	14,35	3,08	4,8	8,51	20,34
0,75	1,97	3,99	9,41	32,84	1,94	3,82	8,32	21,18	4,6	7,17	12,67	29,73
0,76	2,12	4,34	10,33	37,62	2,11	4,13	9,25	23,29	5,07	7,9	13,94	33,01
0,77	2,28	4,74	11,45	43,48	2,26	4,46	10,02	24,07	5,63	8,75	15,49	36,92
0,78	2,48	5,22	12,82	52,46	2,45	4,83	10,91	26,25	6,31	9,77	17,46	41,72
0,79	2,71	5,78	14,55	65,43	2,64	5,28	11,78	29,45	7,12	11,04	19,9	47,76
0,8	2,99	6,45	16,67	84,12	2,89	5,8	13,27	31,79	8,16	12,65	22,85	56,11
0,81	3,32	7,3	19,46	108,46	3,17	6,93	14,92	34,84	9,41	14,6	26,57	66,74
0,82	3,71	8,37	22,91	145,73	3,51	7,58	17,22	38,96	10,93	17,12	31,23	80,2
0,83	4,21	9,71	27,55	194,22	4	8,38	20,8	46,56	12,95	20,58	37,5	98,91
0,84	4,84	11,5	33,48	255,42	4,41	9,31	24,1	52,24	15,59	25,45	46,21	129,15
0,85	5,69	13,82	41,79	353,2	5,1	11,92	28,52	57,32	19,11	32,44	58,66	181,58
0,9	18,34	59,46	382,25	4155	14,14	36,38	60,69	126,18	109,78	199,34	491,13	1779,4
0,95	691,37	6438,2	13195	20626	99,95	196,19	230,94	475,01	9443,6	12736	15230	19958

Salīdzinājuma veikšanai *GPSS World* simulāciju dati tika apkopoti tabulās, kuras satur 15.pielikums. Katrā no tabulām apkopo datus vienai noslodzes koeficienta ρ vērtībai: 15 p. 23. tabula pie $\rho = 0,6$, 15 p. 24. tabula pie $\rho = 0,7$, 15 p. 25. tabula pie $\rho = 0,8$ un 15 p. 26. tabula pie $\rho = 0,9$. Tabulās ir dots kopējais pieprasījumu skaits, kas atbilst *Simulink* modeļa pieprasījumu skaitam pie tādiem pašiem H un ρ parametriem un apstrādāto pieprasījumu skaits[4]. Papildus tam, tabulā ir apkopots rindas vidējais garums $E[R]$, kas tika noapaļots līdz tuvākajam lielākajam veselajam skaitlim un buferatmiņas apjoms, kas šajā eksperimentā tika uzdots ar lielumu $K = 3E[R]$.

Pēc šīm tabulām tika konstruētas 4 līknes vienā grafikā, kas parāda pakešu zudumu varbūtības logaritma $\lg P_{\text{loss}}$ atkarību no Hersta parametra H pie dažādām noslodzes koeficienta ρ vērtībām. Šis grafiks ir parādīts 4.7. att. Buferatmiņas apjomam tika paņemta vērtība $K = 3E[R]$, savukārt $E[R]$ vērtība tika noapaļota līdz tuvākajam lielākajam veselajam skaitlim.

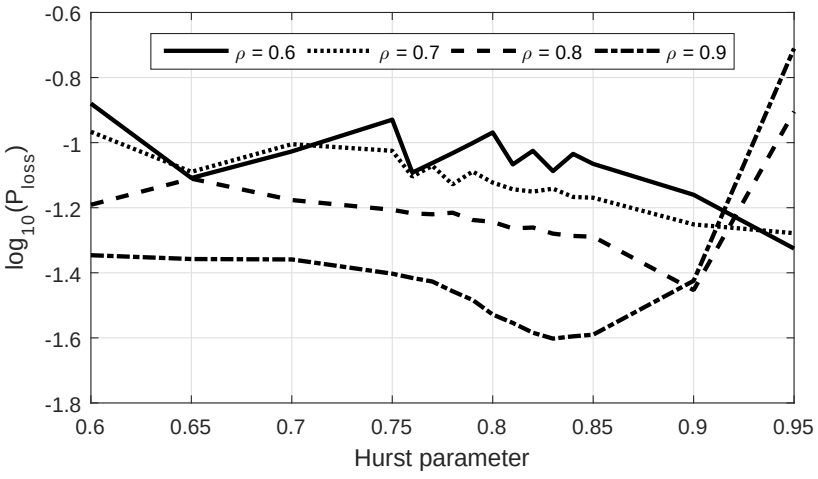
4.7. att. parādītajām līknēm līdzīgās līknes tika konstruētas *Simulink* modeļiem ar dažādiem trafika avotiem un buferatmiņas apjomiem. Trafikam ar Veibula sadalījumu starp intervāliem līknes ir parādītas 4.8. att., modelējot ar dažādiem buferatmiņas apjomiem.

Iepriekš veiktajai simulācijai *GPSS* vidē, kuras rezultāti ir parādīti 4.7. att. *Simulink* modelim atbilst grafiki 4.8b. att. Kā var redzēt, rezultāti ir gandrīz neatšķirami un tas nozīmē, ka *Simulink* izmantošana tādiem modeļiem dod ekvivalentu rezultātu, bet pati modelēšanas vide ir labāk piemērota simulācijas procesa automatizēšanai. Turklāt, *Simulink* vidē ir arī Hersta parametra H novērtēšanas elements, kurš tika veidots patstāvīgi un nav pieejams *GPSS* vidē.

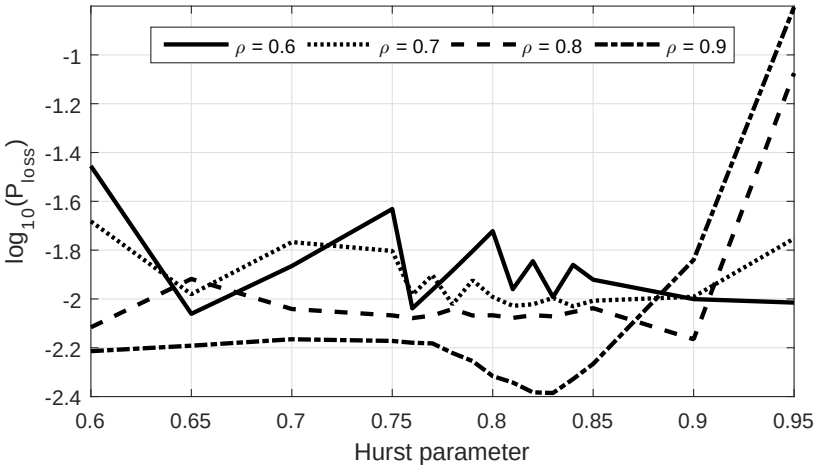
Pārējiem apskatāmajiem darbā modeļiem: $P/M/1/K$ ar *ON/OFF* trafiku un $G/M/1/K$ ar Veibula sadalījumu arī tika konstruēti tādi paši grafiki, kuri ir parādīti 4.9. att. un 4.10. att., attiecīgi. Kā var redzēt no 4.8.–4.10. att., pakešu zudumu varbūtība samazinās līdz ar buferatmiņas apjoma palielināšanu. Turklāt grafiku forma daudzajos gadījumos saglabājās, kā arī novērojamas izmaiņas līknēs pie vērtībām intervālā $0,8 \leq H \leq 0,85$[5].

[4] Attiecīgi, pie ierobežotās buferatmiņas ne visi pieprasījumi tiks apstrādāti, bet daļa no tiem pazūd. Pēc šiem diviem lielumiem var noteikt pieprasījumu (pakešu) zudumu varbūtību P_{loss}.

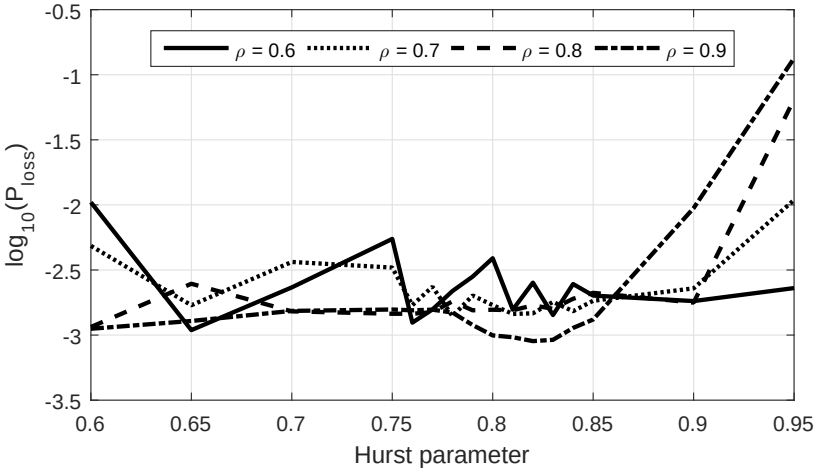
[5] Savukārt $G/M/1/K$ modelim ar Veibula sadalījumu ir novērojama viszemākā pakešu zudumu varbūtība pie vērtības $H = 0,9$.



a

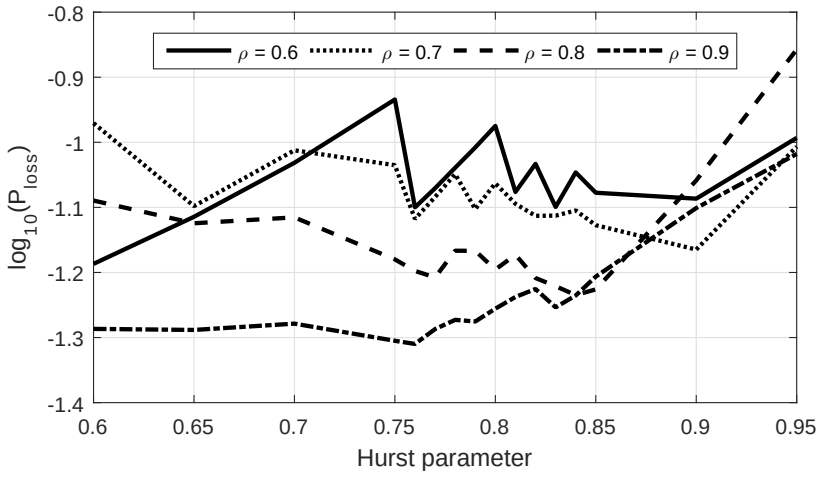


b

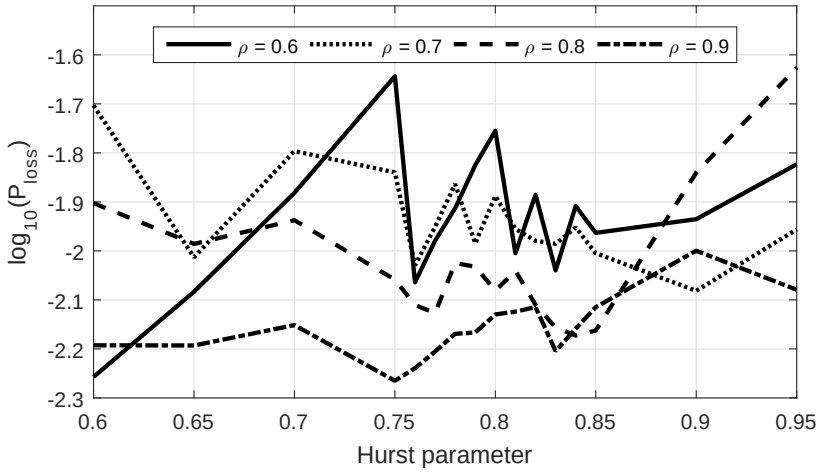


c

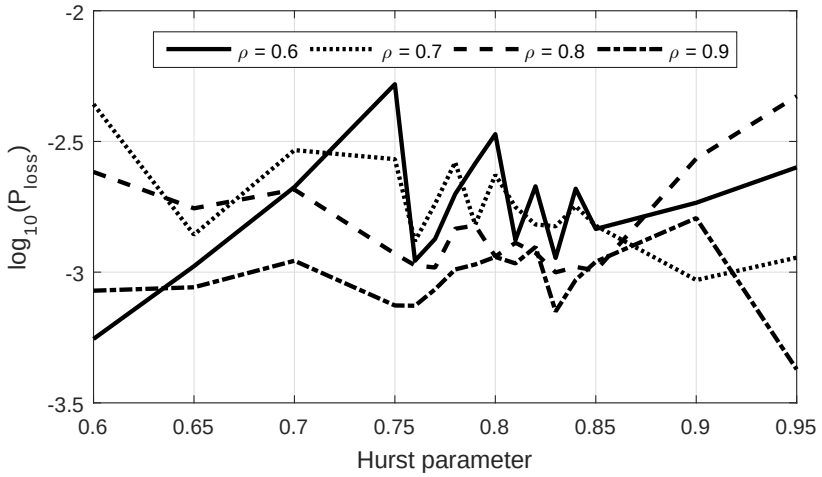
4.8. att. *Simulink* $P/M/1/K$ modeļa pakešu zudumu varbūtības logaritma $\lg P_{\text{loss}}$ atkarība no Hersta parametra H dažādām noslodzes koeficienta ρ vērtībām un buferatmiņas apjomu: a) $E[R]$, b) $3E[R]$, c) $5E[R]$.



a

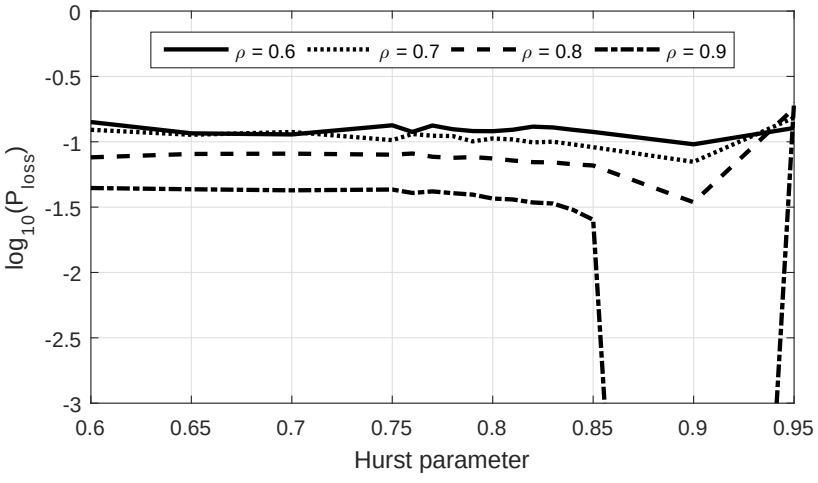


b

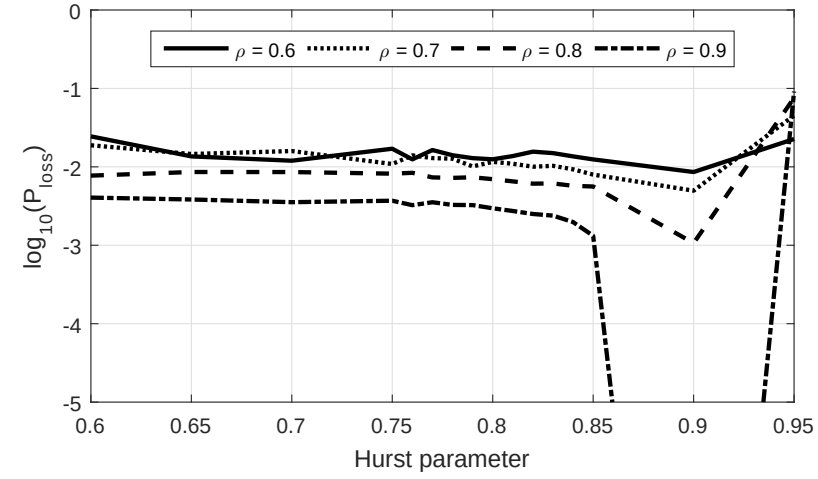


c

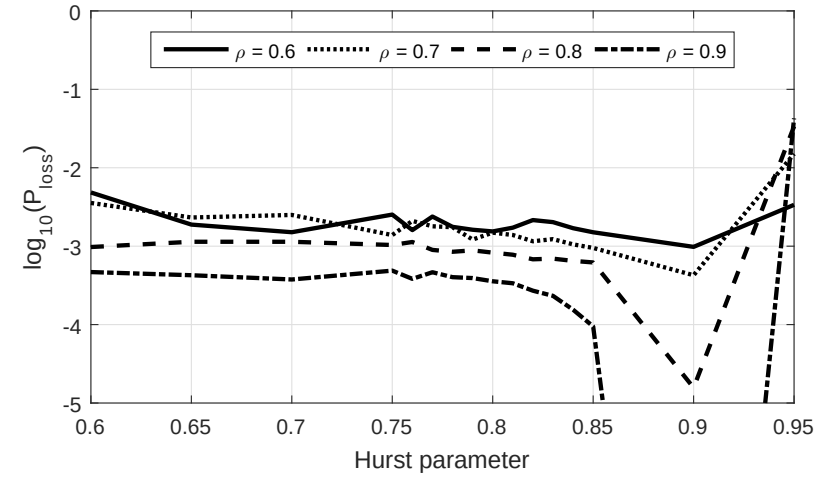
4.9. att. *Simulink* $P/M/1/K$ ar *ON/OFF* modeļa pakešu zudumu varbūtības logaritma $\lg P_{\text{loss}}$ atkarība no Hersta parametra H dažādām noslodzes koeficienta ρ vērtībām un buferatmiņas apjomu: a) $E[R]$, b) $3E[R]$, c) $5E[R]$.



a



b



c

4.10. att. *Simulink* $G/M/1/K$ modeļa pakešu zudumu varbūtības logaritma $\lg P_{\text{loss}}$ atkarība no Hersta parametra H dažādām noslodzes koeficienta ρ vērtībām un buferatmiņas apjomu: a) $E[R]$, b) $3E[R]$, c) $5E[R]$.

4.4. Modelēšanas rezultāti

Pēc eksperimentu sērijas veikšanas un rezultātu analīzes var secināt, ka buferatmiņas apjoma izmaiņas noved pie pakešu zudumu varbūtības P_{loss} izmaiņām. Taču šajā gadījumā, salīdzinot iegūtās vērtības vienādā Hersta parametra intervālā $0,6 \leq H \leq 0,95$ pie vienādiem noslodzes koeficientiem, var konstatēt to, ka:

- mainās P_{loss} vērtības (bet precīzāk – logaritmiskās vērtības $\lg P_{\text{loss}}$);
- kopējais līknes raksturs saglabājās – un to var skaidri redzēt no 4.8.–4.10. att. grafikiem.

Tātad, eksperimentu rezultāti norāda uz to, ka buferatmiņas apjoma izvēle ietekmē pašas vērtības P_{loss} vairāk, nekā to izmaiņu dinamiku atkarībā no Hersta parametra H. Noslodzes koeficientam ρ arī ir liela ietekme uz šādu izmaiņu raksturu, tomēr vienādiem noslodzes koeficientiem un dažādiem buferatmiņas apjomiem, līknes forma ir ļoti līdzīga. Šādi rezultāti ir spēkā visiem 3 apskatītajiem modeļiem: $P/M/1/K$, $P/M/1/K$ ar *ON/OFF* trafiku un $G/M/1/K$ ar Veibula sadalījuma likumu.

Attiecībā uz pēdējo eksperimentu, kura grafiki ir parādīti 4.10. att., var konstatēt, ka trafika modelim ar Veibula sadalījuma likumu ir izteikta minimālā P_{loss} vērtība pie Hersta parametra $H = 0,9$.

Savukārt Pareto trafika modeļos (gan $P/M/1/K$, gan $P/M/1/K$ ar *ON/OFF* trafiku) pakešu zudumu varbūtības P_{loss} (vai tās logaritma) atkarībai no Hersta parametra H mainās līknes raksturs pie Hersta parametra vērtībām $H \approx 0,8$ intervāla robežās.

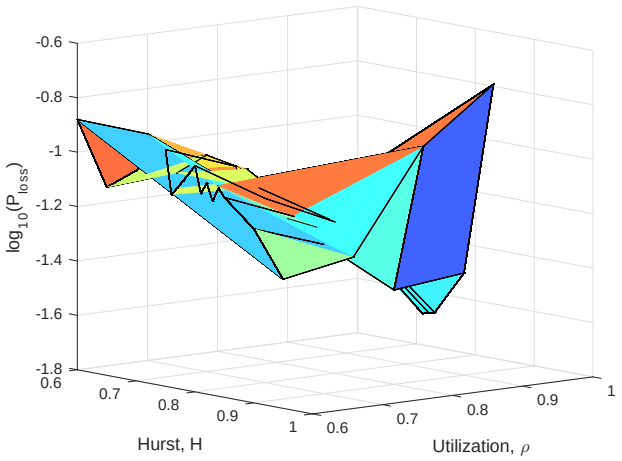
Uzskatāmības nolūkos, šos pašus rezultātus var attēlot arī ar trīsdimensiju grafiku – virsmu, kas parāda pakešu zudumu varbūtības logaritma $\lg P_{\text{loss}}$ atkarību no Hersta parametra H un noslodzes koeficienta ρ. Tas dod iespēju vizuāli konstatēt to, ka šādai atkarībai ir novērojama minimālā vērtība pie noteiktām Hersta parametra H un noslodzes koeficienta ρ vērtībām.

Šie rezultāti ir parādīti zemāk attēlos:

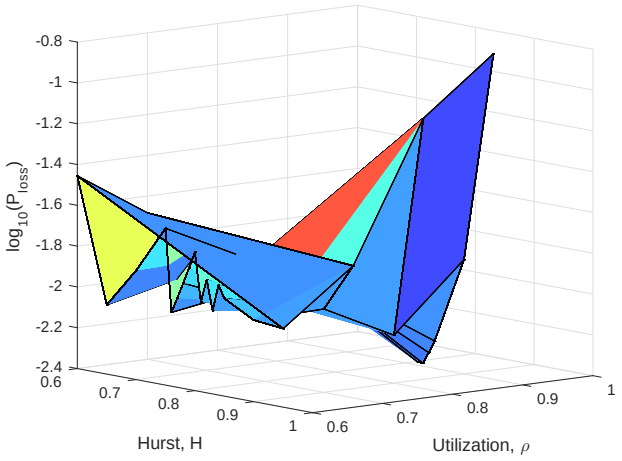
- 4.11. att. ir $P/M/1/K$ modeļa grafiki $E[R]$, $3E[R]$ un $5E[R]$ buferatmiņas apjomiem;
- 4.12. att. ir $P/M/1/K$ *ON/OFF* modeļa grafiki $E[R]$, $3E[R]$ un $5E[R]$ buferatmiņas apjomiem;
- 4.13. att. ir $P/M/1/K$ modeļa grafiki $E[R]$, $3E[R]$ un $5E[R]$ buferatmiņas apjomiem.

No šiem attēliem redzams, ka visiem trafika modeļiem pastāv ekstrēms[6] pakešu zudumu varbūtības P_{loss} atkarībā no Hersta parametra H un noslodzes koeficienta ρ. Intuitīvi saprotams, ka palielinot H vērtību rindas garums arī pieaugs, tomēr pastāv pretējie gadījumi kad H parametra pieaugums samazina vidējo rindas garumu (un, tātad arī buferatmiņas apjomu). Tādus rezultātus var sastapt arī literatūrā, piemēram, [4]. Katram konkrētajam modelim šis ekstrēms (minimums) ir atšķirīgs, tomēr visos gadījumos parametru vērtības ir tuvākas 1. Tā, piemēram, $P/M/1/K$ modelim tādas parametru vērtības ir $H \approx 0,9$ un $\rho \approx 0,8$.

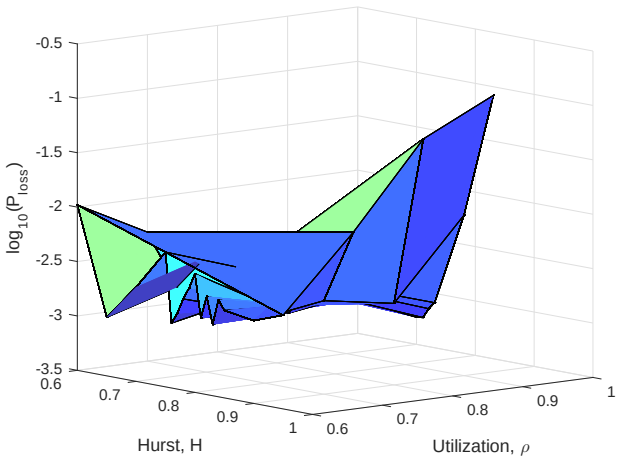
[6] Šeit ir vēlreiz jāuzsver, ka tādu rezultātu iegūšanai trafika modelī ir jāievieš adaptīvā kontrole, kas mainīs buferatmiņas apjomu atkarībā no H un ρ parametriem saskaņā ar 4.3. tabulas datiem.



a

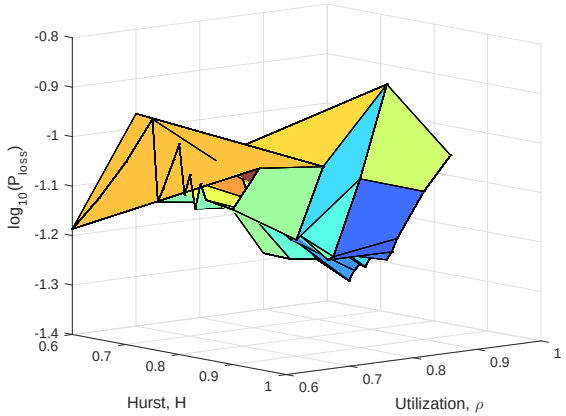


b

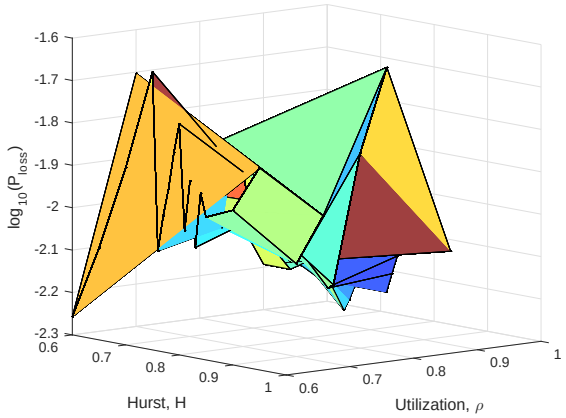


c

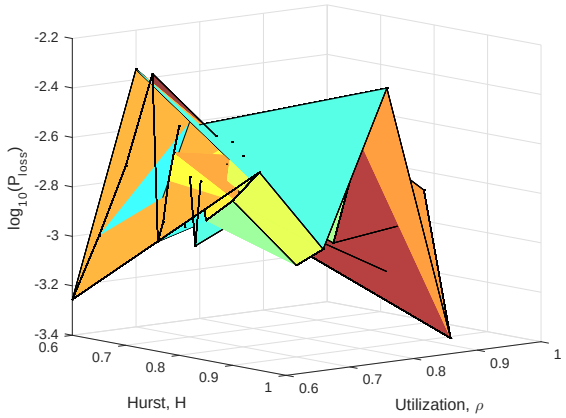
4.11. att. *Simulink* $P/M/1/K$ modeļa pakešu zudumu varbūtības logaritma $\lg P_{\text{loss}}$ atkarība no Hersta parametra H un noslodzes koeficienta ρ vērtībām dažādiem buferatmiņas apjoma lielumiem: a) $E[R]$, b) $3E[R]$, c) $5E[R]$.



a

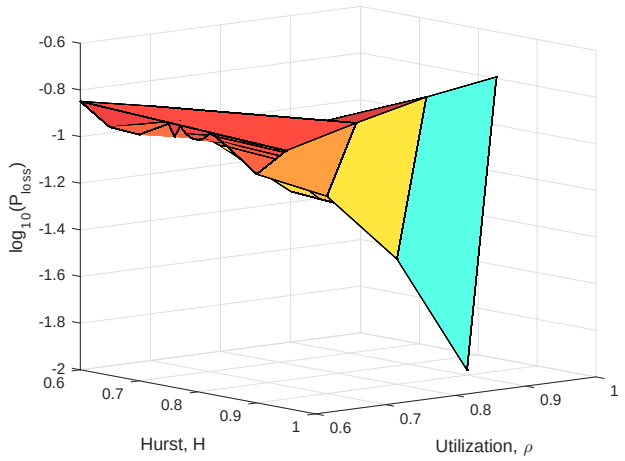


b

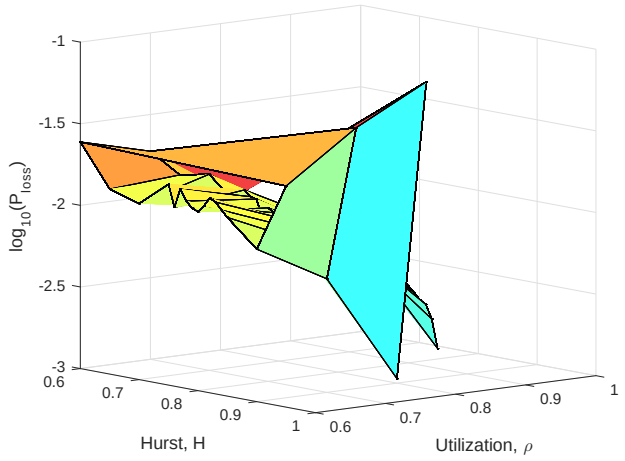


c

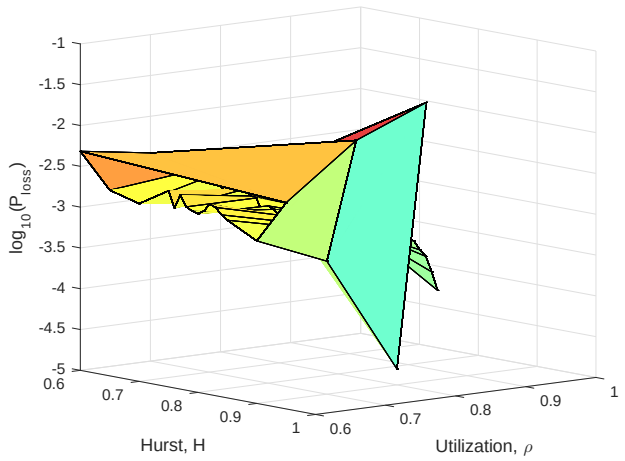
4.12. att. *Simulink* $P/M/1/K$ ar *ON/OFF* modeļa pakešu zudumu varbūtības logaritma $\lg P_{\text{loss}}$ atkarība no Hersta parametra H un noslodzes koeficienta ρ vērtībām dažādiem buferatmiņas apjoma lielumiem: a) $E[R]$, b) $3E[R]$, c) $5E[R]$.



a



b



c

4.13. att. *Simulink* $G/M/1/K$ modeļa pakešu zudumu varbūtības logaritma $\lg P_{\text{loss}}$ atkarība no Hersta parametra H dažādām noslodzes koeficienta ρ vērtībām un buferatmiņas apjomu: a) $E[R]$, b) $3E[R]$, c) $5E[R]$.

4.5. Simulācija ar fiksēto buferatmiņas apjoma ierobežojumu

Lai pamatotu nepieciešamību mainīt trafika apjomu adaptīvi, šajā sadaļā ir apkopoti rezultāti eksperimentiem, kuros buferatmiņas apjoms tika uzturēts nemainīgs visām H vērtībām[7] Tas nozīmē, ka atšķirībā no 4.3.sadaļā aprakstītajiem eksperimentiem, kuros buferatmiņas apjoms tika izvēlēts kā veselais vidējo rindas garumu skaits, šajos eksperimentos ir jāuzdod buferatmiņas apjoms K.

Lai to izdarītu, tiks izmantots klasiskais $M/M/1/K$ modelis, kuram pastāv analītiskās izteiksmes. Tādā sistēmā pakešu zudumu varbūtību P_{loss} konkrētajam buferatmiņas apjomam var novērtēt saskaņā ar izteiksmi [33]:

$$P_{\text{loss}} = \frac{(1-\rho)\rho^K}{1-\rho^{K+1}}, \tag{4.2}$$

kur ρ ir sistēmas noslodzes koeficients. Lai varētu novērtēt nepieciešamo buferatmiņas apjomu K pie noteiktā noslodzes koeficienta ρ un pakešu zudumu varbūtības P_{loss} vērtības, pārveidosim izteiksmi (4.2), atverot iekavas:

$$\begin{aligned}(1-\rho^{K+1})P_{\text{loss}} &= (1-\rho)\rho^K,\\ P_{\text{loss}} - P_{\text{loss}}\rho\rho^K &= \rho^K - \rho\rho^K.\end{aligned}$$

Pēc iekavu atvēršanas var sagrupēt locekļus ar ρ^K kreisajā vienādojuma pusē, bet pārējus – labajā, un iznest lielumu ρ^K pirms iekavām:

$$\begin{aligned}&- P_{\text{loss}}\rho\rho^K - \rho^K + \rho\rho^K = -P_{\text{loss}},\\ &- \rho^K(P_{\text{loss}}\rho + 1 - \rho) = -P_{\text{loss}},\\ &\rho^K(1 - \rho + P_{\text{loss}}) = P_{\text{loss}}.\end{aligned}$$

Izsakot ρ^K var atrast lielumu K, aprēķinot logaritmu pēc bāzes ρ:

$$\begin{aligned}\rho^K &= \frac{P_{\text{loss}}}{1-\rho+P_{\text{loss}}},\\ K &= \log_\rho\left(\frac{P_{\text{loss}}}{1-\rho+P_{\text{loss}}}\right).\end{aligned}$$

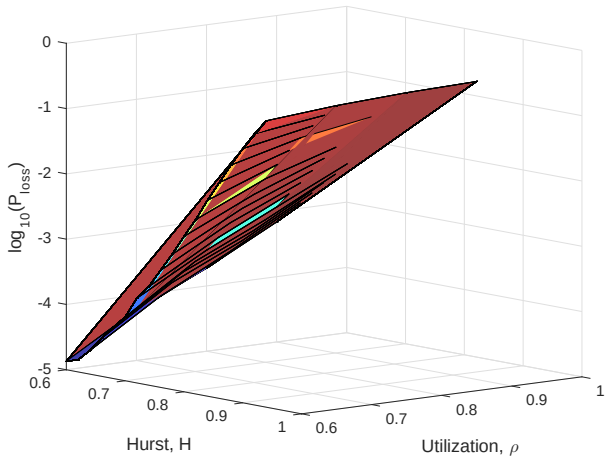
Lai izmantotu šo formulu dažādās programmēšanas valodās, būtu noderīgi nomainīt logaritma bāzi. Piemēram, naturālajam logaritmam pēdējā izteiksme pierakstāma šādi:

$$K = \frac{\ln\left(\frac{P_{\text{loss}}}{1-\rho+P_{\text{loss}}}\right)}{\ln\rho}. \tag{4.3}$$

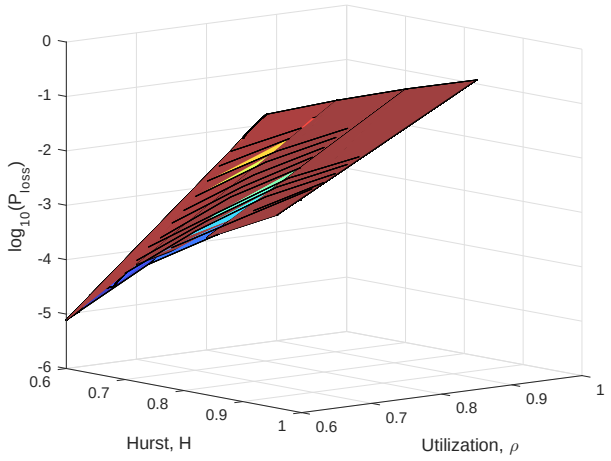
Pēc (4.3) izteiksmes ir iespējams novērtēt $M/M/1/K$ modeļa buferatmiņas apjomu K pie noslodzes koeficienta ρ. Ja saskaņā ar (4.3) K vērtība ir daļskaitlis, to ir jānoapaļo līdz tuvākajam lielākajam veselam skaitlim.

Līdzīgi kā 4.3.sadaļā veiktajās simulācijās, tika atkārtoti palaista programma tādām pašām noslodzes koeficientu ρ un Hersta parametra H vērtībām, kā iepriekš. Buferatmiņas apjoms tika novērtēts saskaņā ar (4.3) izteiksmi divām P_{loss} vērtībām: $P_{\text{loss}} = 10^{-4}$ (4.14. att.) un $P_{\text{loss}} = 10^{-6}$ (4.15. att.). Aprēķinātās buferatmiņas apjoma vērtības ir apkopotas 4.4. tabulā.

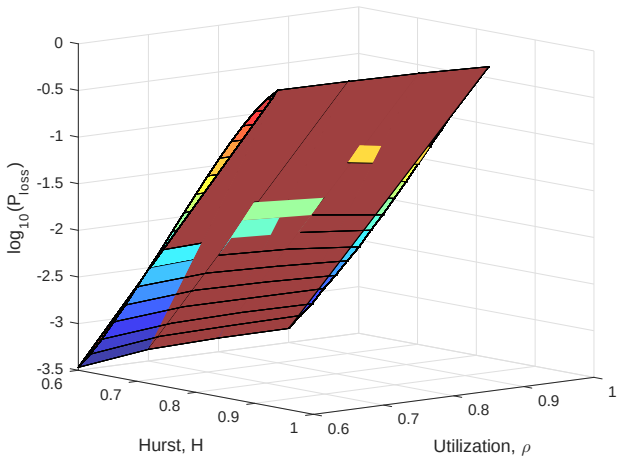
[7] Toties noslodzes koeficientam ρ, kurš ietekmēts ar kanāla caurlaides spēju un servera veiktspēju, buferatmiņas apjoms tiek aprēķināts atbilstoši.



a

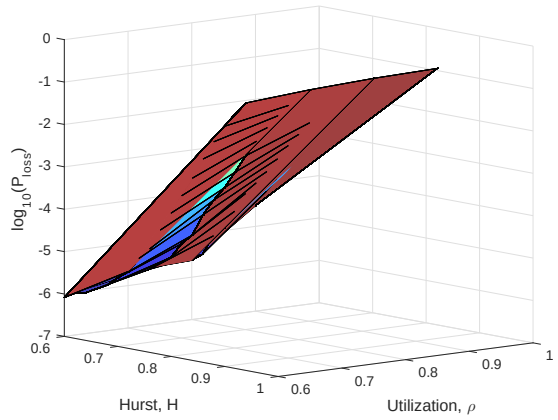


b

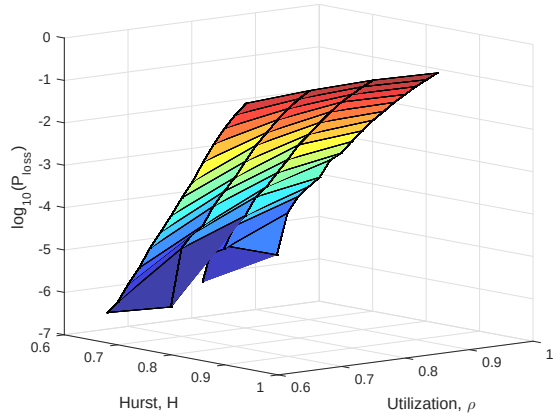


c

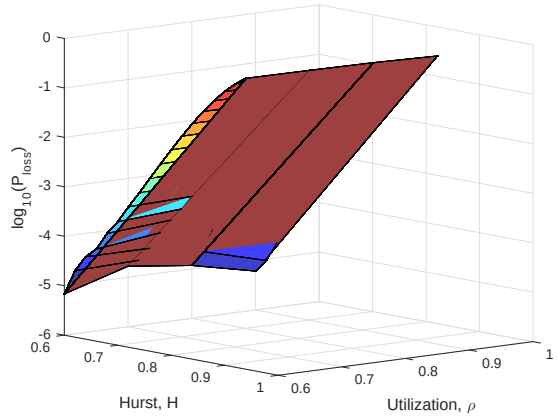
4.14. att. Pakešu zudumu varbūtības logaritma $\lg P_{\text{loss}}$ atkarība no Hersta parametra H un noslodzes koeficienta ρ buferatminas apjomam, kas atbilst $M/M/1/K$ modelim ar uzdoto $P_{\text{loss}} = 10^{-4}$ dažādiem *Simulink* modeļiem a) $P/M/1/K$, b) $P/M/1/K$ ar *ON/OFF* trafiku, c) $G/M/1/K$ ar Veibula sadalījumu.



a



b



c

4.15. att. Pakešu zudumu varbūtības logaritma lgP_{loss} atkarība no Hersta parametra H un noslodzes koeficienta ρ buferatminas apjomam, kas atbilst $M/M/1/K$ modelim ar uzdoto $P_{\text{loss}} = 10^{-6}$ dažādiem *Simulink* modeļiem a) $P/M/1/K$, b) $P/M/1/K$ ar *ON/OFF* trafiku, c) $G/M/1/K$ ar Veibula sadalījumu.

4.4. tabula. Klasiskā $M/M/1/K$ modeļa buferatmiņas apjoms simulāciju veikšanai.

Noslodzes koeficients	**Vērtībai** $P_{\text{loss}} = 10^{-4}$	**Vērtībai** $P_{\text{loss}} = 10^{-6}$
0,6	17	26
0,7	23	36
0,8	35	55
0,9	66	110

Kaut arī buferatmiņas apjoms tika rēķināts $M/M/1/K$ modelim, šāds ierobežojums tika iestādīts $P/M/1/K$, $P/M/1/K$ ar *ON/OFF* trafiku un $G/M/1/K$ modelī ar Veibula sadalījumu. Visi šie modeļi tuvāk apraksta reālo trafiku tīklos. Eksperimentu mērķis ir noteikt P_{loss} vērtības šādiem modeļiem visām H un ρ parametru vērtībām.

Kā var redzēt no trīsdimensiju grafikiem 4.14.–4.15. att., visos gadījumos pakešu zudumu varbūtība[8] P_{loss} pieaug līdz ar H un ρ parametru pieaugumu. Trīsdimensiju virsmas, kas atspoguļo tādas atkarības ir ļoti tuvas plaknēm, kas nozīmē ka parametru ietekme uz pakešu zudumu varbūtības logaritmu $\lg P_{\text{loss}}$ ir tuva eksponenciālajai.

Salīdzinot savā starpā 4.14.–4.15. att. ar 4.11.-4.13. att., var konstatēt, ka buferatmiņas apjoma novērtēšana pēc klasiskās formulas saskaņā ar (4.3) dod nepamatoti optimistiskus novērtējumus, kas noved pie pakešu zudumu varbūtības pieauguma līdz ar Hersta parametra H palielināšanu. [83] ir veikts salīdzinājums eksistējošajām buferatmiņas apjoma novērtēšanas metodēm un konstatēts, ka tās dod dažādus rezultātus, kuri ne vienmēr ir apstiprināmi ar modelēšanu. Tāpēc ir nepieciešama adaptīvā kontrole, kura ievēros Hersta parametra vērtību H un noslodzes koeficientu ρ. Atkarībā no nepieciešamās pakešu zudumu varbūtības P_{loss} vērtības tāda sistēma izvēlēsies piemēroto buferatmiņas apjomu. Šādu buferatmiņas apjoma atkarību no pakešu zudumu varbūtības P_{loss} var formēt ar dominējošo secību, izmantojot dinamiskās programmēšanas principu (Belmana algoritms).

4.6. Dinamiskās programmēšanas paņēmiena lietošana dominējošās secības veidošanai

Lai veiktu optimizācijas uzdevuma risināšanu, var pielietot R. Belmana algoritmu [8], ar kura palīdzību var izveidot lēmuma pieņemšanas secību par to, kādus sistēmas parametrus K un ρ ir jāizvēlas pie noteiktās H parametra vērtības, lai nodrošinātu nepieciešamo pakešu zudumu varbūtību P_{loss} un tajā skaitā lai risinājums būtu optimāls pēc izmaksām, kas ir ļoti svarīgi praktiskajos nolūkos. Pašam dinamiskās programmēšanas principam nav nekā kopīga ar programmas koda rakstīšanu, t.i., programmēšanas plašo jēdzienu. Tajā vietā runa ir par optimizācijas procedūru, kurā ar katru tuvinājumu tiek atrasts arvien labāks risinājums, kamēr prasības pēc viena no parametriem netiek pārsniegtas.

Neskatoties uz to, ka ideja ir ļoti veca, tā joprojām vēl tiek apskatīta un pielietota daudzajos uzdevumos. Pati pieeja ir izklāstīta vairākās grāmatās, piemēram, [27]. To var izmantot daudzajos uzdevumos, bet tie tuvāki, kas attiecās uz šajā darba apskatāmajiem jautājumiem, ir saistīti ar resursu rezervāciju, kā ir apskatīts, piemēram, [88] un izmaksu minimizāciju ar vairākiem ierobežojošajiem parametriem, kā ir aprakstīts [84]. Arī drošuma jautājums ir ļoti svarīgs, jo sistēmai sakaru kvalitātes uzturēšanai ir jānodrošina noteiktā pakešu zudumu varbūtība P_{loss}, un dinamiskās programmēšanas paņēmiens ir pielietojams arī līdzīgajos uzdevumos, piemēram, [87].

Dinamiskās programmēšanas pieeja tiek izmantota arī tīklu optimizācijas problēmās. Piemēram, [32] runa ir par datortīkla paplašināšanas problēmām, kuras atrisināšanai tiek piedāvāts

[8] Uzskatāmībai z-asī tika atlikts tās logaritms, $\lg P_{\text{loss}}$.

izmantot dinamisko programmēšanu. Savukārt, [23] tiek piedāvāts izmantot dinamiskās programmēšanas paņēmienu *ARMA* procesam ar Hersta parametra izmaiņām, lai noteiktu straujas izmaiņas procesā.

[30] ir piedāvāts izmantot dinamiskās programmēšanas paņēmienu, lai sadalītu sakaru resursus heterogēnajos tīklos. Savukārt [31] ir veikts salīdzinājums starp dinamiskās programmēšanas paņēmienu un ģenētisko algoritmu trafika plūsmas kontrolei. Abas metodes dod atšķirīgus rezultātus, bet var būt pielietotas tā, lai noslogotajā tīklā neveidotos garās rindas.

Lai pielietotu dinamiskās programmēšanas Belmana algoritmu, ir jāsastāda tabula vienai konkrētajai H parametra vērtībai, kurai tiks veikts optimizācijas uzdevums. Šīs vērtības tika iegūtas no eksperimentu rezultātiem, kas iekļauj sevī pakešu zudumu varbūtību vērtības P_{loss} dažādām Hersta parametra H, noslodzes koeficienta ρ un buferatmiņas apjoma K vērtībām. Lai pēdējais lielums mainītos nepārtraukti, tika veiktas papildus simulācijas ar buferatmiņas apjomiem $2E[R]$ un $4E[R]$, kā arī papildus ar $6E[R]$, $7E[R]$, $8E[R]$, $9E[R]$ un $10E[R]$.

Katrai H parametra vērtībai tiek sastādīta atsevišķa tabula, kas parāda pakešu zudumu P_{loss} atkarību no noslodzes koeficienta ρ un buferatmiņas apjoma (vai tā reizinātāja). Tādas tabulas piemērs $P/M/1/K$ modelim ir parādīts 4.16. att. Katrā tabulas ailītē bez pakešu zudumiem P_{loss} tiek ievadītas izmaksas – gan pēc noslodzes koeficienta ρ, gan pēc buferatmiņas apjoma K.

	ρ = 0,6	ρ = 0,7	ρ = 0,8	ρ = 0,9
$K = 1E[R]$	1 6 0.10748	1 7 0.075375	1 8 0.057081	1 9 0.029627
$K = 2E[R]$	2 6 0,042893	2 7 0,026055	2 8 0,020595	2 9 0,011168
$K = 3E[R]$	3 6 0,018967	3 7 0,010185	3 8 0,0085718	3 9 0,0048284
$K = 4E[R]$	4 6 0,0084389	4 7 0,0039853	4 8 0,0038761	4 9 0,0023069
$K = 5E[R]$	5 6 0,0038934	5 7 0,0016982	5 8 0,0015656	5 9 0,00099385
$K = 6E[R]$	6 6 0,0018009	6 7 0,00066002	6 8 0,00068674	6 9 0,00045967
$K = 7E[R]$	7 6 0,00080708	7 7 0,00031331	7 8 0,00036305	7 9 0,00021448
$K = 8E[R]$	8 6 0,0004042	8 7 0,00013092	8 8 0,00018028	8 9 0,00011016
$K = 9E[R]$	9 6 0,00019813	9 7 $3{,}9377 \cdot 10^{-5}$	9 8 $7{,}5242 \cdot 10^{-5}$	9 9 $5{,}1062 \cdot 10^{-5}$
$K = 10E[R]$	10 6 $8{,}9547 \cdot 10^{-5}$	10 7 $1{,}3647 \cdot 10^{-5}$	10 8 $1{,}6265 \cdot 10^{-5}$	10 9 $3{,}0703 \cdot 10^{-5}$

4.16. att. Dominējošās secības formēšana ar Belmana algoritmu $P/M/1/K$ modelim ar Hersta parametru $H = 0,8$.

Parametru izmaksas 4.16. att. parādītajā tabulā tiek novērtētas šādi[9]:

- noslodzes koeficienta ρ izmaksas ir pati noslodzes koeficienta vērtība, kas tika sareizināta ar 10;
- buferatmiņas apjoma K izmaksas ir reizinātājs, kas parāda cik vidējie rindas garumi tika paņemti buferatmiņas apjoma iegūšanai (no 1 līdz 10).

Tālāk, izveidotajā tabulā, sākot ar vismazākajām parametru vērtībām, tiek formēta dominējošā secība, izvēloties vismazāko iespējamo pakešu zudumu varbūtību P_{loss}, kas būtu pieņemams lielums (atkarībā no prasībām). Šajā piemērā, uzskatīsim ka pirmā pieņemamā vērtība būtu $P_{\text{loss}} = 0,029627$ pie $\rho = 0,9$ un $K = 1E[R]$. Citos gadījumos var uzdot savādāku vērtību. Kopējās izmaksas sastāda $9 + 1 = 10$. Tālāk, pie $\rho = 0,7$ un $K = 2E[R]$, vērtība $P_{\text{loss}} = 0,026055$, bet izmaksas ir $7 + 2 = 9$, kuras ir zemākas un ceļš pāriet uz šo ailīti. Tālāk ceļš turpina pakešu zudumu varbūtības P_{loss} samazināšanas virzienā, priekšroku dodot ailītēm ar zemākām izmaksām. Rezultātā formējas dominējošā secība, kā tas ir parādīts 4.16. att.

Pēc šīs secības ir iespējams veikt optimizācijas uzdevuma risināšanu, nosakot parametrus ρ un K, kuri nodrošinātu pakešu zudumu varbūtību P_{loss} (vai mazāku par to) pie viszemākajām izmaksām. Lai to panāktu, ir jāuzdod:

- buferatmiņas apjoma izmaksas C_1 gadā;
- kanāla pārraides ātruma izmaksas C_2 gadā.

Šeit ir jāatzīmē, ka izvēlētajām izmaksām nav principiālās nozīmes šajos piemēros, kuri tiks parādīti zemāk. Reālas izmaksas ir atkarīgas no atrašanās vietas (valsts) un veidotās sistēmas. Šajā sadaļā tika izvēlēti vidējie lielumi uz 2013. gadu, lai novērtētu kopējo situāciju un veiktu optimizāciju pēc izmaksām.

Tādā veidā, saskaņā ar pētījumu [96], kanāla caurlaides spējas izmaksas Eiropā sastādīja apmerām (vidēji) 2.5 \$/mēn par 1 Mbps 2013. gada beigās[10]. Kopumā, izmaksām par informācijas pārraidi ir izskatāma tendence samazināties.

Savukārt buferatmiņas apjoma izmaksas ir atkarīgas no izmantojamās atmiņas veida. Tā, piemeram, saskaņa ar [12] varēja izvēlēties statisko atmiņu *SRAM* ar piekļuves laiku 0.45 ms un izmaksām 27 \$/MB vai dinamisko atmiņu *DRAM* ar piekļuves laiku 55 ms un izmaksām 0.016 \$/MB.

Lai pārietu no abstraktajām mērvienībām, kas tiek izmantotas simulācijās, ir jāveic vairāki pieņēmumi un pārrēķini. Pirmkārt, tiek pieņemts, ka runa ir par *TCP* trafiku, kuram paketes garums sastāda $L_{\text{pak}} = 1500$ baitus, [90]. Tādā veidā, zinot trafika intensitāti λ paketes/s mērvienībās un buferatmiņas apjomu K, kurš visās simulācijās arī tika norādīts pakešu mērvienības, ir iespējams novērtēt nepieciešamo buferatmiņas apjomu K_{buf} un kanāla pārraides ātrumu R_{kan}.

Lai noteiktu buferatmiņas apjomu MB vienībās, ir jāsareizina buferatmiņas apjoms paketēs K ar vienas paketes garumu L_{pak} un jāizsaka to MB mērvienībās, izejot no ta, ka $1MB = 2^{10}KB = 2^{20}B$:

$$K_{\text{buf}} = \frac{K \cdot 1500}{2^{20}} \text{ [MB]}. \tag{4.4}$$

Tālāk, zinot 1MB izmaksas, pēc (4.4) var aprēķināt kopējās buferatmiņas apjoma izmaksas. Zemāk apskatāmajos piemēros tika paņemta dārgāka, statiskā *SRAM* atmiņa, kas nodrošina maršrutētājos nepieciešamu augstu veiktspēju:

$$C_1 = K_{\text{buf}} \cdot 27 \text{ [\$]}. \tag{4.5}$$

[9] Šeit ir jāatzīmē, ka pareizāk būtu ievērot izmaksas cenas vienības, turklāt vēl jānoved abi parametri pie salīdzināmajām pakāpēm. Tas tiks izdarīts tālāk.

[10] Šeit vidējās izmaksas tika noteiktas vadoties gan pēc vara vadu tehnoloģijām, tādām kā *xDSL*, gan pēc kabeļa pieslēgumiem, gan pēc optiskās šķiedras pieslēgumam

Aprēķinātās pēc (4.5) izmaksas ir paredzētas maršrutētāja veidošanai uz 5 gadiem, savukārt izmaksu salīdzināšanai ir jānoved visas izmaksas uz vienādu laika periodu, piemēram, 1 kalendāro gadu. Tādā gadījumā izmaksas sastādīs:

$$C_1 = \frac{K_{\text{buf}} \cdot 27}{5} = K_{\text{buf}} \cdot 5.4 \text{ [\$/gadā]}. \tag{4.6}$$

Līdzīgi jānosaka arī kanāla pārraides ātruma R izmaksas C_2. Pēc būtības, nepieciešamais kanāla pārraides ātrums R ir atkarīgs no noslodzes koeficienta $\rho = R/C$, kur C – kanāla caurlaides spēja, t.i., maksimālais pārraides ātrums $\max R$. Šo caurlaides spēju maršrutētājs izmanto datu pārraidei un tā ir saistīta ar apstrādes veiktspēju μ šādā veidā:

$$\mu = \frac{C}{L_{\text{pak}} \cdot 8} \text{ [paketes/s]}, \tag{4.7}$$

Līdz ar to, var izteikt lielumu C saskaņā ar šādu izteiksmi:

$$C = \mu \cdot L_{\text{pak}} \cdot 8 \text{ [bps]}. \tag{4.8}$$

Saskaņā ar izteiksmi (4.8), pie simulācijas parametra $\mu = 125$ tika iegūta vērtība $C = 1500000$ bps. Tā kā apskatāmais noslodzes koeficients $\rho = R/C$ ir relatīvs lielums, tad gan kanāla ātrdarbība R, gan kanāla caurlaides spēja C var būt izteiktas jebkurās mērvienībās, kamēr saglabāsies to attiecība un tiks nodrošināts uzdotais noslodzes koeficients ρ. Lai to pierādītu, tika veiktas simulācijas ar vērtībām $\mu = 800$ paketes/s (lai $C \approx 10$ Mbps) un $\mu = 125$ paketes/s. Savukārt trafika intensitāte tika noteikta kā daļa no caurlaides spējas μ, izmantojot noslodzes koeficientu: $\lambda = \rho \cdot \mu$. Simulāciju rezultāti $P/M/1/K$ modelim ir apkopoti 4.5. tabulā un parāda ka atšķirības ir ļoti neievērojamās. Pārējiem apskatītajiem nodaļā modeļiem arī bija tāda pati sakritība.

4.5. tabula. Vidējā rindas garuma salīdzinājums dažādām intensitātēm λ.

Noslodzes koeficients	**Intensitāte paketes/s**	**Vidējais rindas garums** $E[R]$			
		$H = 0,6$	$H = 0,7$	$H = 0,8$	$H = 0,9$
$\rho = 0,6$	$\lambda = 75$	0,76	1,08	2,07	13,81
	$\lambda = 480$	0,76	1,08	2,07	13,81
$\rho = 0,7$	$\lambda = 87,5$	1,63	2,41	5,21	60,54
	$\lambda = 560$	1,63	2,41	5,21	60,54
$\rho = 0,8$	$\lambda = 100$	3,83	6,12	16,56	458,49
	$\lambda = 640$	3,83	6,12	16,56	458,49
$\rho = 0,9$	$\lambda = 112,5$	12,58	24,09	102,53	16031,53
	$\lambda = 720$	12,58	24,09	102,53	16031,43

Tādā veidā, paņemot noteikto lielumu $C = 9,6$ Mbps mērvienībās[11]. Savukārt faktisko pieslēguma ātrumu R var noteikt kā daļu no caurlaides spējas C, izmantojot noslodzes koeficientu:

$$R = \rho \cdot C \text{ [Mbps]}. \tag{4.9}$$

Galu galā, lai noteiktu gada izmaksas par pieslēgumu, ir jāsareizina pārraides ātrums, kas aprēķināts saskaņā ar (4.9), 1 Mbps izmaksas, kas vienādas ar 2.5 \$/mēn un 12, lai noteiktu gada izmaksas:

$$C_2 = 12R \cdot 2,5 \text{ } \textbf{[\$/gadā]}. \tag{4.10}$$

[11] Tāda vērtība paņemta tikai reālistiskākajam gadījumam, jo 1,5 Mbps mūsdienās būtu pārāk zems pārraides ātrums priekš parastā lietotāja, bet 96 Mbps varētu būt pārāk liels ātrums, ja apskatīt vidējo situāciju Eiropā. Tādā veidā, tika paņemts vidējais lielums.

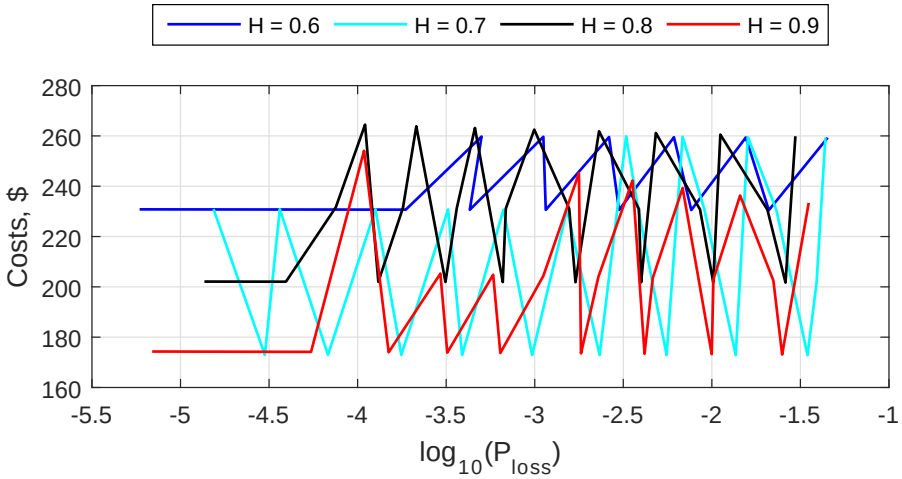
Pēc izmaksu aprēķina parādītais 4.16. att. algoritms var atkal būt pielietots, tikai šajā reizē tiks ievērotas cenas izmaksas, nevis abstraktie lielumi, kā tas bija darīts pirmajā reizē. Tabulas ar izveidotajām dominējošajām secībām visiem apskatītajiem šajā nodaļā trafika modeļiem: $P/M/1/K$, $P/M/1/K$ ar *ON/OFF* un $G/M/1/K$ ar Veibula sadalījumu satur 16.pielikums.. Hersta parametrs mainās robežās $0,6 \leq H \leq 0,95$ ar soli $0,05$. Līdzīgi 4.16. att. parādītajai dominējošajai secībai, tika sastādīta dominējošā secība Hersta parametra vērtībai $H = 0,8$, ievērojot izmaksas. Ja eksperimenta laikā visas paketes tika apstrādātas, t.i., P_{loss} novērtējums ir 0, tāds rezultāts tiek atmests. Izveidotā dominējošā secība ir parādīta 4.17. att.

	ρ = 0,6	ρ = 0,7	ρ = 0,8	ρ = 0,9
K = 1$E[R]$	172,8 $ un 0,02 $ 0,10748 172,82 $	201,6 $ un 0,05 $ 0,075375 201,65 $	230,4 $ un 0,13 $ 0,057081 230,53 $	259,2 $ un 0,66 $ 0,029627 259,86 $
K = 2$E[R]$	172,8 $ un 0,04 $ 0,042893 172,84 $	201,6 $ un 0,1 $ 0,026055 201,7 $	230,4 $ un 0,26 $ 0,020595 230,66 $	259,2 $ un 1,32 $ 0,011168 260,52 $
K = 3$E[R]$	172,8 $ un 0,06 $ 0,018967 172,86 $	201,6 $ un 0,15 $ 0,010185 201,75 $	230,4 $ un 0,39 $ 0,0085718 230,79 $	259,2 $ un 1,98 $ 0,0048284 261,18 $
K = 4$E[R]$	172,8 $ un 0,08 $ 0,0084389 172,88 $	201,6 $ un 0,2 $ 0,0039853 201,8 $	230,4 $ un 0,52 $ 0,0038761 230,92 $	259,2 $ un 2,64 $ 0,0023069 261,84 $
K = 5$E[R]$	172,8 $ un 0,1 $ 0,0038934 172,9 $	201,6 $ un 0,25 $ 0,0016982 201,85 $	230,4 $ un 0,65 $ 0,0015656 231,05 $	259,2 $ un 3,3 $ 0,00099385 262,5 $
K = 6$E[R]$	172,8 $ un 0,12 $ 0,0018009 172,92 $	201,6 $ un 0,3 $ 0,00066002 201,9 $	230,4 $ un 0,78 $ 0,00068674 231,18 $	259,2 $ un 3,96 $ 0,00045967 263,16 $
K = 7$E[R]$	172,8 $ un 0,14 $ 0,00080708 172,94 $	201,6 $ un 0,35 $ 0,00031331 201,95 $	230,4 $ un 0,91 $ 0,00036305 231,31 $	259,2 $ un 4,62 $ 0,00021448 263,82 $
K = 8$E[R]$	172,8 $ un 0,16 $ 0,0004042 172,96 $	201,6 $ un 0,4 $ 0,00013092 202 $	230,4 $ un 1,04 $ 0,00018028 231,44 $	259,2 $ un 5,28 $ 0,00011016 264,48 $
K = 9$E[R]$	172,8 $ un 0,18 $ 0,00019813 172,98 $	201,6 $ un 0,45 $ $3,9377 \cdot 10^{-5}$ 202,05 $	230,4 $ un 1,17 $ $7,5242 \cdot 10^{-5}$ 231,57 $	259,2 $ un 5,94 $ $5,1062 \cdot 10^{-5}$ 265,14 $
K = 10$E[R]$	172,8 $ un 0,2 $ $8,9547 \cdot 10^{-5}$ 173 $	201,6 $ un 0,5 $ $1,3647 \cdot 10^{-5}$ 202,1 $	230,4 $ un 1,3 $ $1,6265 \cdot 10^{-5}$ 231,7 $	259,2 $ un 6,6 $ $3,0703 \cdot 10^{-5}$ 265,8 $

4.17. att. Dominējošās secības formēšana ar Belmana algoritmu $P/M/1/K$ modelim ar Hersta parametru $H = 0,8$ pie uzdotajām parametru izmaksām.

Katrā 4.17. att. ailītē ir ierakstītas atsevišķās izmaksas: kanāla pārraides ātruma izmaksas C_2 un buferatmiņas izmaksas C_1; pakešu zudumu varbūtība P_{loss} un kopējās izmaksas $C_2 + C_1$. Dominējošā secība tika veidota saskaņā ar Belmana dinamiskās programmēšanas algoritmu un, salīdzinot 4.16. att. ar 4.17. att., var secināt ka precīzākā izmaksu pievienošana nemainīja dominējošo secību, jo abos šajos gadījumos kanāla pārraides ātruma izmaksas ievērojami pārsniedz buferatmiņas apjoma izmaksas. Pēc datiem no tabulām, kuras satur 16.pielikums., var konstruēt līknes, kas parāda izmaksu $C_2 + C_1$ atkarību no pakešu zudumu varbūtības logaritma $\lg P_{\text{loss}}$ dažādām Hersta parametra H vērtībām. Tādu grafiku piemērs ir parādīts 4.18. att.

Pēc grafikiem 4.18. att. var redzēt, ka pastāv minimālās izmaksas, turklāt vērtības atšķiras dažādiem Hersta parametriem H. Šiem minimumiem atbilst noteikti sistēmas parametri, kuriem tika novērtētas izmaksas. Tādas minimālās izmaksas ir apkopotas 4.6. tabulā.



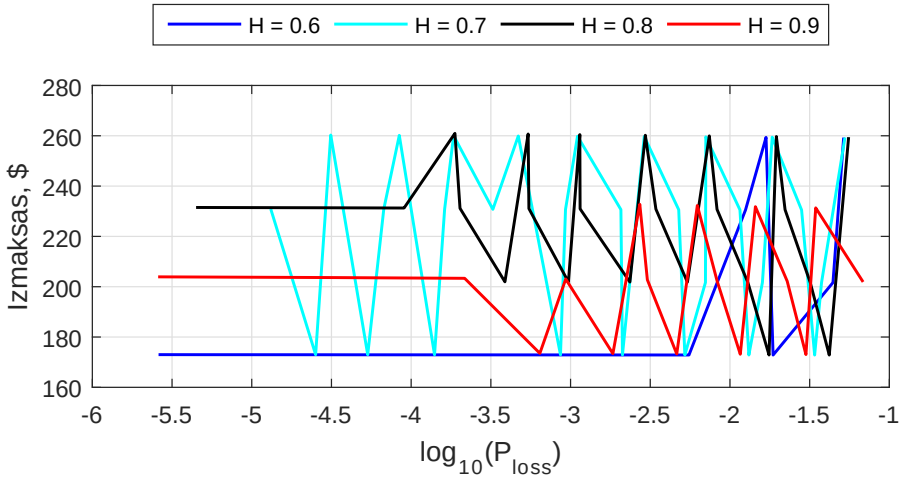
4.18. att. Kopējās $P/M/1/K$ modeļa izmaksas atkarībā no pakešu zudumu varbūtības dažādām Hersta parametra vērtībām.

4.6. tabula. Sistēmas parametri minimālajām parametru izmaksām $P/M/1/K$ modelim un attiecīgās P_{loss} vērtības.

Hersta parametrs	**Kopējās izmaksas**	**Pārraides ātrums**	**Buferatmiņas apjoms**	**Varbūtība P_{loss}**
$H = 0,6$	230,48 $	6,72 Mbps	14,9 KB	0,020972
	230,52 $	6,72 Mbps	22,3 KB	0,0076529
	230,56 $	6,72 Mbps	29,8 KB	0,0029914
	230,60 $	6,72 Mbps	37,2 KB	0,0011542
	230,64 $	6,72 Mbps	44,7 KB	0,00042973
	230,68 $	6,72 Mbps	52,1 KB	0,00018692
	230,72 $	6,72 Mbps	59,6 KB	$7,3643 \cdot 10^{-5}$
	230,76 $	6,72 Mbps	67 KB	$2,1255 \cdot 10^{-5}$
	230,80 $	6,72 Mbps	74,5 KB	$5,8765 \cdot 10^{-6}$
$H = 0,7$	172,84 $	5,76 Mbps	7,4 KB	0,034693
	172,86 $	5,76 Mbps	11,2 KB	0,013641
	172,88 $	5,76 Mbps	14,9 KB	0,0055544
	172,90 $	5,76 Mbps	18,6 KB	0,0023246
	172,92 $	5,76 Mbps	22,3 KB	0,00096637
	172,94 $	5,76 Mbps	26,1 KB	0,00038904
	172,96 $	5,76 Mbps	29,8 KB	0,00017647
	172,98 $	5,76 Mbps	33,5 KB	$6,8028 \cdot 10^{-5}$
	173 $	5,76 Mbps	37,2 KB	$2,9939 \cdot 10^{-5}$

$H = 0,8$	201,70 $	6,72 Mbps	19 KB	0,026055
	201,75 $	6,72 Mbps	28,4 KB	0,010185
	201,80 $	6,72 Mbps	37,9 KB	0,0039853
	201,85 $	6,72 Mbps	47,4 KB	0,0016982
	201,90 $	6,72 Mbps	56,9 KB	0,00066002
	201,95 $	6,72 Mbps	66,4 KB	0,00031331
	202,00 $	6,72 Mbps	75,9 KB	0,00013092
	202,05 $	6,72 Mbps	85,3 KB	$3,9377 \cdot 10^{-5}$
	202,10 $	6,72 Mbps	94,8 KB	$1,3647 \cdot 10^{-5}$
$H = 0,9$	173,10 $	5,76 Mbps	56,9 KB	0,024942
	173,25 $	5,76 Mbps	85,3 KB	0,0099848
	173,40 $	5,76 Mbps	113,8 KB	0,004169
	173,55 $	5,76 Mbps	142,2 KB	0,0018247
	173,70 $	5,76 Mbps	170,7 KB	0,00063947
	173,85 $	5,76 Mbps	199,1 KB	0,00032084
	174,00 $	5,76 Mbps	227,6 KB	0,00014972
	174,15 $	5,76 Mbps	256 KB	$5,4574 \cdot 10^{-5}$
	174,30 $	5,76 Mbps	284,4 KB	$6,92 \cdot 10^{-6}$

Līdzīgi, var konstruēt tādus pašus grafikus arī citiem trafika modeļiem, piemēram, $P/M/1/K$ modelim ar *ON/OFF* trafika raksturu. Izmaksu atkarības līknes ir parādītas 4.19. att. dažādām sevlīdzīguma (Hersta) parametra vērtībām.



4.19. att. Kopējās izmaksas atkarībā no pakešu zudumu varbūtības dažādām Hersta parametra vērtībām.

Salīdzinot savā starpā 4.18. att. un 4.19. att. grafikus, var konstatēt, ka Hersta parametram H nav izteikta likumsakarība uz izmaksām (un, tātad arī buferatmiņas apjomu). Kaut arī, kopumā, buferatmiņas apjoms palielinās, tomēr tas nenoved pie lielajām izmaksām pat gadījumā ar statisko atmiņu SRAM, kura ir dārgāka. Tas ir spēkā visām dominējošajām secībām, kas tika konstruētas 4.18. att. un 4.19. att.. Līdzīgi, kā tika sastādīta minimumu 4.6. tabula, tika apkopoti dati par minimālajām izmaksām un attiecīgajiem sistēmas parametriem 4.7. tabulā.

4.7. tabula. Sistēmas parametri minimālajām parametru izmaksām $P/M/1/K$ modelim ar *ON/OFF* trafiku un attiecīgās P_{loss} vērtības.

Hersta parametrs	**Kopējās izmaksas**	**Pārraides ātrums**	**Buferatmiņas apjoms**	**Varbūtība P_{loss}**
$H = 0,6$	172,84 $	5,76 Mbps	7,8 KB	0,018646
	172,86 $	5,76 Mbps	11,4 KB	0,0055316
	172,88 $	5,76 Mbps	14,9 KB	0,0017569
	172,90 $	5,76 Mbps	18,6 KB	0,00055529
	172,92 $	5,76 Mbps	22,3 KB	0,00016706
	172,94 $	5,76 Mbps	26,1 KB	$4,4236 \cdot 10^{-5}$
	172,96 $	5,76 Mbps	29,8 KB	$2,6021 \cdot 10^{-5}$
	172,98 $	5,76 Mbps	33,5 KB	$5,2042 \cdot 10^{-6}$
	173,00 $	5,76 Mbps	37,2 KB	$2,6021 \cdot 10^{-6}$
$H = 0,7$	172,84 $	5,76 Mbps	7,6 KB	0,033973
	172,86 $	5,76 Mbps	11,4 KB	0,013138
	172,88 $	5,76 Mbps	15,2 KB	0,0052122
	172,90 $	5,76 Mbps	18,9 KB	0,0021165
	172,92 $	5,76 Mbps	22,8 KB	0,00086468
	172,96 $	5,76 Mbps	30,3 KB	0,00014018
	172,98 $	5,76 Mbps	34,1 KB	$5,3401 \cdot 10^{-5}$
	173,00 $	5,76 Mbps	37,9 KB	$2,516 \cdot 10^{-5}$
$H = 0,8$	172,84 $	5,76 Mbps	7,4 KB	0,04197
	172,86 $	5,76 Mbps	11,1 KB	0,017583
	201,80 $	6,72 Mbps	37,2 KB	0,0054083
	201,85 $	6,72 Mbps	46,5 KB	0,002356
	201,90 $	6,72 Mbps	55,9 KB	0,00096937
	201,95 $	6,72 Mbps	65,2 KB	0,00038749
	231,28 $	7,68 Mbps	163,8 KB	$9,0256 \cdot 10^{-5}$
	231,39 $	7,68 Mbps	184,3 KB	$2,4988 \cdot 10^{-5}$
	231,50 $	7,68 Mbps	204,8 KB	$4,4755 \cdot 10^{-6}$
$H = 0,9$	173,04 $	5,76 Mbps	44,7 KB	0,029965
	173,16 $	5,76 Mbps	67 KB	0,011608
	173,28 $	5,76 Mbps	89,4 KB	0,0046361
	173,40 $	5,76 Mbps	111,7 KB	0,0018431
	173,52 $	5,76 Mbps	134,1 KB	0,00064173
	203,63 $	6,72 Mbps	377,9 KB	$3,3634 \cdot 10^{-5}$
	203,92 $	6,72 Mbps	431,9 KB	$2,5872 \cdot 10^{-6}$

Salīdzinot savā starpā šīs divas tabulas, var konstatēt, ka, lielākoties, grupveidīgs *ON/OFF* trafiks ar Pareto sadalījuma likumu (garās astes varbūtību blīvuma funkcijā) nav dārgāks par vienkāršo sevlīdzīgo trafiku un atsevišķajos gadījumos pat lētāks, ja runāt par gala izmaksām attiecībā uz pakešu zudumu varbūtību P_{loss}. Tajā pašā laikā nepieciešamais buferatmiņas apjoms palielinās gadījumā ar grupveidīgo trafiku pie augstākajām sevlīdzīguma parametra H vērtībām. Tas noved pie lielākām aizturēm, t.i., nepieciešams ilgāks laiks, kamēr paketes aizies līdz adresātam.

4.7. Nobeigums

Šajā nodaļā tika veiktas simulācijas trafikam ar dažādu sevlīdzīguma pakāpi (Hersta parametrs) un ģenerēšanas modeli. Tika apskatīti trīs rindošanas modeļi ar ierobežotu buferatmiņas apjomu:

- $P/M/1/K$ modelis ar Pareto sadalījuma likumu starp pieprasījumiem;
- $P/M/1/K$ *ON/OFF* modelis ar Pareto sadalījuma likumu starp pieprasījumiem un *ON/OFF* rakstura trafiku;
- $G/M/1/K$ modelis ar Veibula sadalījuma likumu starp pieprasījumiem.

Visiem modeļiem tika noteikts buferatmiņas apjoms kā vidējais rindas garums tādam pašam modelim bez buferatmiņas apjoma ierobežojuma. Šādos eksperimentos tika noteikts vidējais $E[R]$ un maksimālais rindas garums $\max[R]$ dažādām Hersta parametra H un noslodzes koeficienta ρ vērtībām. Tika izpētīta šo divu rindas garumu attiecība un bija konstatēts, ka tādas attiecības atkarībai no Hersta parametra H ir dilstošais raksturs, bet pie $0,8 \leq H \leq 0,85$ vērtībām šī likumsakarība strauji mainās ar turpmāko atgriešanos pie vecā rakstura.

Iegūtās vidējo rindas garumu vērtības $E[R]$ tika izmantotas augstāk norādītajos modeļos buferatmiņas apjoma ierobežošanai, ievērojot Hersta parametra H un noslodzes koeficienta ρ vērtības, pie kurām tās tika iegūtas. Tas nozīmē, ka faktiski būtu jānosaka Hersta parametru H, izmantojot 3.5..sadaļā izveidoto novērtēšanas bloku un tad jāizvēlas buferatmiņas apjoms, ievērojot iepriekš veiktas simulācijas. Reālajai aparatūrai tāda pieeja varētu būt līdzīga. Simulāciju laikā tika izvēlēts kārtns E$[R]$ buferatmiņas apjoms K, kurš vairākās reizēs (no 1 līdz 10) pārsniedz vidējo rindas garumu. Visām simulācijām tika novērtēta pakešu zudumu varbūtība P_{loss} un konstruētas 3-dimensiju virsmas.

Šajās 3-dimensiju virsmās tika konstatēts, ka pastāv ekstrēmi pakešu zudumu varbūtības atkarībā no Hersta parametra H un noslodzes koeficienta ρ vērtībām. Turklāt ja novērtēt buferatmiņas apjomu saskaņā ar $M/M/1/K$ formulu, tādi ekstrēmi vairs nav novērojami.

Lai sīkāk izpētītu šos ekstrēmus, tika izmantota Belmana dinamiskās programmēšanas metode, ar kuras palīdzību tika risināts optimizācija uzdevums ar izmaksu minimizēšanu pie uzdotās P_{loss} vērtības. Tika konstatēts, ka izvēloties no kanāla caurlaides spējas izmaksām un buferatmiņas apjoma izmaksām, sevlīdzīgajam trafikam ar Hersta parametru $H \leq 0,9$ lielāka izmaksu daļa ir saistīta ar kanāla caurlaides spējas izmaksām un ir iespējams minimizēt tās, pievienojot papildus buferatmiņas apjomu.

Nodaļā iegūtie rezultāti norāda uz to, ka dažādām Hersta parametra vērtībām H ir jānodrošina dažāds buferatmiņas apjoms nepieciešamā pakešu zudumu varbūtības P_{loss} lieluma sasniegšanai. Pie tam, pastāv vairāki sistēmas uzbūves varianti ar iespējamajiem optimalitātes kritērijiem – veiktspēju, izmaksām, noslodzi, drošumu utt. Ievērojot to, ka dažādiem trafika veidiem (protokoliem, lietojumprogrammām) Hersta parametrs atšķiras un ir iespējams to nomērīt, var apstrādāt dažādus trafika veidus atšķirīgi, atkarībā no *QoS* prasībām. Tāda pieeja, pēc būtības ir *PBAC* (*Policy Based Admission Control*), kas jau tiek izmantota lielajās kompānijās, tādās kā *Cisco* [22], *Microsoft* [75]. *Cisco* piedāvā izmantot arī imitācijas rīku *GNS3* [95], kurš imitē kompānijas maršrutētājus ar visu operacionālo sistēmu un ļauj sīki konfigurēt visus elementus. Diplomdarbā [81] ir izmantots šis rīks *PBAC* maršrutētāju konfigurēšanai un tīkla analīzei.

Galvenie secinājumi

Šī promocijas darba mērķi tika veiksmīgi sasniegti un atrisināti visi uzdevumi. Darba rezultātus var apkopot un veikt šādus secinājumus:

- Eksistējošās buferatmiņas apjoma novērtēšanas metodes sevlīdzīgajam trafikam nedod adekvātus apjoma novērtējumus un ir adaptīvi jāmaina sistēmas parametrus, tajā skaitā, arī buferatmiņas apjomu, veicot trafika mērījumus;
- Dažādām Hersta (sevlīdzīguma) parametra vērtībām atbilst dažāds buferatmiņas apjoms, un buferatmiņas apjoma palielināšanās tendence ir novērojama gan atkarībā no Hersta parametra vērtības, gan no noslodzes koeficienta vērtības;
- Reāllaika diskrētās veivletu transformācijas aprēķināšanai var izmantot filtru bankas pieeju (arī gadījumos kad runa nav par *FPGA* vai *DSP* sistēmām) un veikt uzlabojumus, izvēloties sarežģītākas filtru struktūras, kā polifāzes filtru vai kāpņveida filtru bankas realizācija;
- Diskrētās veivletu transformācijas algoritms *C++* valodā strādā pietiekami ātri, lai varētu to izmantot sevlīdzīgā trafika parametru novērtēšanai reāllaikā;
- Apskatot inverso diskrēto veivletu transformāciju, tika konstatēts, ka ātrākajai signāla rekonstruēšanai ir lietderīgi veikt signāla decimāciju analīzes filtru bankā tādā veidā, lai tā atstātu nepāra numura nolases – tas samazina aizturi katrā mērogā un šie lielumi uzkrājas vairākos mērogos;
- Var izveidot tādu Hersta parametra novērtēšanas mezglu, kurš atjaunotu Hersta parametra novērtējumu ar katru trafika nolasi reāllaikā, analizējot vairākus transformācijas mērogus, turklāt programmas izpildes laiks ir salīdzināms ar diskrētās veivletu transformācijas aprēķina laiku un tāds Hersta parametra novērtēšanas mezgls ir praktiski lietojams;
- Hersta parametra mērīšanas precizitātes uzlabošanai ir jāizvēlas piemērots analīzes mērogu skaits, kurš ir mazāks par diskrētās veivletu transformācijas maksimālo mērogu skaitu un ir atkarīgs no paša Hersta parametra lieluma. Izveidotajā programmā pastāv iespēja adaptīvi mainīt šo lielumu programmas izpildes gaitā;
- Novērtējot pakešu zudumu varbūtību sistēmā ar ierobežotu buferatmiņu, tika konstatēts, ka gadījumā, kad buferatmiņas apjoms tiek izvēlēts adaptīvi, Hersta parametra palielināšanās ne vienmēr noved pie pakešu zudumu varbūtības pieauguma. Piemēram, Hersta parametriem intervālā $0,8 \leq H \leq 0,9$ tika novērota pretēja aina;
- Sevlīdzīgajam trafikam ar Hersta parametru $H = 0,8$ vai tuvu pie tā ir novērojams anomāls raksturs vairākās likumsakarībās, piemēram, vidējais/maksimālais rindas garums atkarībā no Hersta parametra, vai arī Hersta parametra novērtēšanas kļūda atkarībā no Hersta parametra. Daudzajos darbos piemēri tiek doti tieši ar $H = 0,8$ vērtību, kura, pēc būtības, atbilst vidēji izteiktajai sevlīdzīguma pakāpei;
- Iegūtos Hersta parametra, buferatmiņas apjoma un pakešu zudumu varbūtības mērījumus var izmantot, lai veiktu daudzparametru optimizāciju ar dinamiskās programmēšanas

Belmana algoritmu pēc dažādiem kritērijiem: kopējo izmaksu, pakešu zudumu varbūtības minimuma utt.;

- Belmana algoritma lietošana sevlīdzīgajam trafikam, ievērojot reālās caurlaides spējas un buferatmiņas izmaksas 2013. gada beigās, parādīja, ka izmaksu minimizēšanai ir jāpalielina buferatmiņas apjoms un jāsamazina noslodzes koeficients;
- Veidojot dominējošo secību grafikus, kas parāda kopējo izmaksu atkarību no pakešu zudumu varbūtības, Hersta parametra vērtībai $H = 0,8$ līknes raksturs attiecībā uz pārējām Hersta parametra vērtībām arī atšķiras un uzrādīja pārāk augstas izmaksas.

Galvenie promocijas darba rezultāti tika aprakstīti publikācijās un paziņoti starptautiskajās konferencēs.

Promocijas darba aizstāvēšanai tiek izvirzītas šādas tēzes:

1. Diskrētās veivletu transformācijas decimācijas procedūrā nepāra numuru koeficientu atstāšana samazina rekonstruējamā signāla laika aizturi, veicot inverso veivletu transformāciju;
2. Hersta parametra novērtēšanā ar diskrēto veivletu transformāciju eksistē optimālais mērogu skaits, kurš ir atkarīgs no mērītā Hersta parametra lieluma un veivletu transformācijas uzdotā maksimālā mērogu skaita;
3. Trafikam, kura Hersta parametrs $H = 0,8$, piemīt mazākā zudumu varbūtība un zemākā novērtēšanas kļūda, salīdzinot ar trafikiem, kuriem ir atšķirīgas Hersta parametra vērtības.

Galvenie promocijas darba rezultāti tika aprakstīti publikācijās un paziņoti starptautiskajās konferencēs.

LITERATŪRA

[1] P. Abry, D. Veitch. Wavelet-Analysis of Long-Range-Dependent Traffic // IEEE Transactions on Information Theory. – IEEE Computer Society Washington, DC, USA, 1998 – vol. 44, no. 1 – pp. 2–15. DOI: 10.1109/18.650984

[2] Amin, F.; Mizanian, K. Buffer management for self-similar network traffic. // Telecommunications (IST), 2012 Sixth International Symposium, 6–8 Nov. 2012 – pp. 737–742

[3] Amin, F.; Mizanain, K.; Mirjalily, G. Active queue management for self-similar network traffic // Electrical Engineering (ICEE), 2013 21st Iranian Conference on , vol., no., 14–16 May 2013 – pp. 1–5

[4] Arnold L. Neidhardt, Jonathan L. Wang. he Concept of Relevant Time Scales and Its Application to Queuing Analysis of Self-Similar Traffic (or Is Hurst Naughty or Nice?). // in Proc. SIGMETRICS '98/PERFORMANCE '98, 1998 – pp. 222–232

[5] M. Bahoura, H. Ezzaidi. Real-time implementation of discrete wavelet transform on FPGA // 2010 IEEE 10th International Conference on Signal Processing (ICSP). – IEEE Press Piscataway, 2010 – pp. 191–194.

[6] Xiaoming Bai; Daohuan Liu; Taorong Qiu. Research and Application of an Online Real time Hurst Parameter Measure Algorithm. // Intelligent Information Technology Application, 2009. IITA 2009. Third International Symposium on , vol. 1, no., 21–22 Nov. 2009 – pp. 15–17

[7] Balamash, A.; Krunz, M. Application of multifractals in the characterization of WWW traffic. / Communications, 2002. ICC 2002. IEEE International Conference on , vol. 4, no., 2002 – pp. 2395–2399

[8] Richard Bellman. Dynamic Programming. // Princeton University Press. – NY, USA, 1957 – 340 p.

[9] I. Ben Hnia Gazzah, C. Souani, K. Besbes. DSP implementation and performances evaluation of 1D and 2D DWT using the lifting scheme. // Design and Test Workshop, 2008. IDT 2008. 3rd International, 20–22 Dec. 2008 – pp. 166–172. doi: 10.1109/IDT.2008.4802490

[10] Florian Bomers. Wavelets in real time digital audio processing: Analysis and sample implementations. Master's Thesis. // University of Mannheim, May 2000 – 119 p.

[11] Bregni, S. Compared accuracy evaluation of estimators of traffic long-range dependence. // Communications (LATINCOM), 2014 IEEE Latin-America Conference on , vol., no., 5–7 Nov. 2014 – pp. 1–5

[12] Gabriel M.de Brito, Pedro B.Velloso, Igor M.Moraes. Information Centric Networks: A New Paradigm for the Internet // Willey-ISTE. – April 2013 – 144 p.

[13] Buyukcorak, S.; Kurt, G.K.; Toprakkiran, G. An empirical study on call arrivals in cellular networks: Incoming and outgoing calls. // Signal Processing and Communication Systems (ICSPCS), 2013 7th International Conference, 16–18 Dec. 2013 – pp. 1–5

[14] M. A. Cavuslu, F. Karakaya. Hardware implementation of Discrete Wavelet Transform and Inverse Discrete Wavelet Transform on FPGA // Signal Processing and Communications Applications Conference (SIU), 2010 IEEE 18th. – IEEE Press Piscataway, 2010 – pp. 141–144.

[15] Can Ye; Huiyun Li; Guoqing Xu. An early warning model of traffic accidents based on fractal theory. // Intelligent Transportation Systems (ITSC), 2014 IEEE 17th International Conference on , vol., no., 8–11 Oct. 2014 – pp. 2280–2285

[16] Chatterjee, S.; MacGregor, M.H.; Bates, S. Detecting changes in the Hurst parameter. // Local Computer Networks, 2008. LCN 2008. 33rd IEEE Conference on , vol., no., 14–17 Oct. 2008 – pp. 876–883

[17] Chaudhari, S.S.; Biradar, R.C. Resource prediction based routing using wavelet neural network in mobile ad-hoc networks. // Circuits, Communication, Control and Computing (I4C), 2014 International Conference on , vol., no., 21–22 Nov. 2014 – pp. 273–276

[18] Yiou Chen; Jianhao Hu; Jianrong Wang. Self-similar traffic study of on-chip interconnection networks. // Communications, Circuits and Systems (ICCCAS), 2013 International Conference on , vol. 1, no., 15–17 Nov. 2013 – pp. 381–385

[19] Chen Di; Feng Hai-Hang; Lin Qing-jia; Chen Chun-xiao. Multi-scale Internet Traffic Prediction Using Wavelet Neural Network Combined Model. // Communications and Networking in China, 2006. ChinaCom '06. First International Conference on , vol., no., 25–27 Oct. 2006 – pp. 1–5

[20] X. Cheng, K. Xie, D. Wang. Network Traffic Anomaly Detection Based on Self-Similarity Using HHT and Wavelet Transform. // IAS '09. Proceedings of the Fifth International Conference on Information Assurance and Security. – IEEE Press Piscataway, 2009 – pp. 710–713. DOI: 10.1109/IAS.2009.219

[21] Chin-Tser Huang; Thareja, S.; Yong-June Shin. Wavelet-based Real Time Detection of Network Traffic Anomalies. // Securecomm and Workshops, 2006, vol., no., Aug. 28 2006 – Sept. 1 2006 – pp. 1–7

[22] Cisco Identity Services Engine User Guide, Release 1.2 // Cisco Systems, April 7, 2015 – 788 p.

[23] Coulon, M.; Chabert, M.; Swami, A., Detection of Multiple Changes in Fractional Integrated ARMA Processes // Signal Processing, IEEE Transactions on, vol. 57, no. 1, – Jan. 2009 – pp. 48–61. doi: 10.1109/TSP.2008.2007313

[24] Cutajar, M.; Gatt, E.; Grech, I; Casha, O.; Micallef, J., Design of a hardware-based discrete wavelet transform architecture for phoneme recognition // Communications, Control and Signal Processing (ISCCSP), 2014 6th International Symposium, 21–23 May 2014 – pp. 554–557. doi: 10.1109/ISCCSP.2014.6877935

[25] Darji, AD., Shashikanth, Konale, Limaye, Ankur; Merchant, S.N.; Chandorkar, AN. Flipping-based high speed VLSI architecture for 2-D lifting DWT. // Circuits and Systems (MWSCAS), 2014 IEEE 57th International Midwest Symposium, 3–6 Aug. 2014 – pp. 193–196

[26] Darji, A; Shukla, S.; Merchant, S.N.; Chandorkar, AN. Hardware Efficient VLSI Architecture for 3-D Discrete Wavelet Transform // VLSI Design and 2014 13th International Conference on Embedded Systems, 2014 27th International Conference, 5–9 Jan. 2014 – pp. 348–352. doi: 10.1109/VLSID.2014.66

[27] S. Dasgupta, C.H.Papadimitriou, U.V. Vazirani. Algorithms. // McGraw-Hill Science/Engineering/Math – 2006 – 336 p.

[28] I. Daubechies and W. Sweldens. Factoring Wavelet Transforms into Lifting Steps. // Fourier Analysis and Applications, 4(3), 1998 – pp. 247–269.

[29] DeVirgilio, M.; Pan, W.D.; Joiner, L.L.; Dongsheng Wu. Internet delay statistics: Measuring Internet feel using a dichotomous Hurst parameter. // Southeastcon, 2012 Proceedings of IEEE , vol., no., 15–18 March 2012 – pp. 1–6

[30] Elsherif, A.R.; Zhi Ding; Xin Liu, A Resource Allocation Scheme for Heterogeneous Networks Using Dynamic Programming Approach // Vehicular Technology Conference (VTC Spring) 2014 IEEE 79th. – 18–21 May 2014 – pp. 1–6. doi: 10.1109/VTCSpring.2014.7022794

[31] Elshqeirat, B.; Soh, S.; Rai, S.; Lazarescu, M., A Dynamic Programming Algorithm for Reliable Network Design // Reliability, IEEE Transactions on, vol. 63, no. 2, June 2014 – pp. 443–454. doi: 10.1109/TR.2014.2314597

[32] Olaf E. Flippo, Antoon W.J. Kolen, Arie M.C.A. Koster, Robert L.M.J. van de Leensel. A dynamic programming algorithm for local access telecommunication network expansion problem // European Journal of Operational Research 127 – 2000 – pp. 189–202

[33] F. N. Gouweleeuw and H. C. Tijms. Computing loss probabilities in discrete-time queues. // Operations Research, 46(1) – January 1998 – pp. 149–154

[34] Geyong Min; Xiaolong Jin. Analytical Modelling and Optimization of Congestion Control for Prioritized Multi-Class Self-Similar Traffic. // Communications, IEEE Transactions on, vol. 61, no. 1, January 2013 – pp. 257–265

[35] A. Girard, B. Sanso, F. Vazquez-Abad. Performance Evaluation and Planning Methods for the Next Generation Internet. // Springer US, 2005 – 365 p.

[36] E. Grab, S. Sarkovskis. Real-Time Estimation of Traffic Self-Similarity Parameter in Simulink with Wavelet Transform // Electronics and Electrical Engineering. – Kaunas: Technologija, 2013 – no.3(129) – pp. 88–91

[37] Hong Zhao, Nirwan Ansari. Wavelet Transform-based Network Traffic Prediction: A Fast On-line Approach. // Journal of Computing and Information Technology – CIT 20, 2012 – pp. 15–25

[38] Jain, K.; Baras, J.S. Impact of protocols on traffic burstiness at large timescales in wireless multi-hop networks. // Ad Hoc Networking Workshop (MED-HOC-NET), 2013 12th Annual Mediterranean , vol., no., 24–26 June 2013 – pp. 59–66

[39] Romans Jerjomins. Research of Wireless Local Area Network in Non-stationary Mode. // PhD Thesis, Program "Computer control, information and electronic systems of transport", Riga Technical University – 2012 – 157 p.

[40] Jieying Han; Zhang, J.Z. Network traffic anomaly detection using weighted self-similarity based on EMD. // Southeastcon, 2013 Proceedings of IEEE , vol., no., 4–7 April 2013 – pp. 1–5

[41] C. Jing, H. Yuan Bin. Efficient Wavelet Transform on FPGA Using Advanced Distributed Arithmetic // Electronic Measurement and Instruments, 2007. ICEMI '07. 8th International Conference on, Aug. 16 2007 – pp. 2–515 to 5–515. doi: 10.1109/ICEMI.2007.4350730

[42] M. Kulikovs. Research and development of effective managing algorithms in admission control systems for telecommunication networks. Doctoral Diploma Thesis. // Riga: RTU, 2010 – 200 p.

[43] M. Kulikovs, E. Petersons, S. Sharkovsky. Integral measurement process of incoming traffic for measurement-based admission control // Proceeding of the 2010 IEEE Region 8 International Conference on Computational Technologies in Electrical and Electronics Engineering – IEEE Press Piscataway, 2010 – pp. 183–186. DOI:10.1109/SIBIRCON.2010.5555296

[44] M. Kulikovs, S. Sharkovsky, E. Petersons. Comparative studies of methods for accurate Hurst parameter estimation // Electronics and Electrical Engineering. – Kaunas: Technologija, 2010 – no. 7(103) – pp. 113–116.

[45] Lee, J.R.; Sang-Kug Ye; Jeong, H.-D.J. Detecting Anomaly Teletraffic Using Stochastic Self-Similarity Based on Hadoop. // Network-Based Information Systems (NBiS), 2013 16th International Conference on , vol., no., 4–6 Sept. 2013 – pp. 282–287

[46] W. Leland, M. Taqqu, W. Willinger, D. Wilson. On the Self-Similar Nature of Ethernet Traffic // IEEE/ACM Transactions on Networking – IEEE Press Piscataway, vol. 2, no. 1, 1994 – pp. 203–213. DOI: 10.1109/90.282603

[47] Li, Yongli; Guizhong Liu; Hongliang Li; Hou, Xingsong. Wavelet-based analysis of hurst parameter estimation for self-similar traffic. // Acoustics, Speech, and Signal Processing (ICASSP), 2002 IEEE International Conference on , vol. 2, no., 13–17 May 2002 – pp. 2061–2064

[48] Liji, P.I.; Dipin, A. Real time data traffic analysis using poisson process in next generation network. // Control, Instrumentation, Communication and Computational Technologies (ICCICCT), 2014 International Conference on , vol., no., 10–11 July 2014 – pp. 289–293

[49] Mallat, Stéphane. A wavelet tour of signal processing. 2nd edition. // Academic press, 1999 – 668 p.

[50] B. Mandelbrot. The Fractal Geometry of Nature. // Henry Holt and Company, 1982 – 480 p.

[51] Mirzaei, M.M.; Mizanian, K.; Rezaeian, M. Modeling of self-similar network traffic using artificial neural networks. // Computer and Knowledge Engineering (ICCKE), 2014 4th International eConference on , vol., no., 29–30 Oct. 2014 – pp. 741–746

[52] Mota, Hd.O.; Vasconcelos, F.H.; da Silva, R.M. Real-time wavelet transform algorithms for the processing of continuous streams of data. // Intelligent Signal Processing, 2005 IEEE International Workshop on , vol., no., 1–3 Sept. 2005 – pp. 346–351

[53] Nafornita, C.; Isar, A.; Nelson, J.D.B. Regularised, semi-local hurst estimation via generalised lasso and dual-tree complex wavelets. // Image Processing (ICIP), 2014 IEEE International Conference on , vol., no., 27–30 Oct. 2014 – pp. 2689–2693

[54] A. O'Brien, R. Conway. Lifting scheme discrete Wavelet Transform using Vertical and Crosswise multipliers. // Signals and Systems Conference, 208. (ISSC 2008). IET Irish, 18–19 June 2008, pp. 331–336

[55] Ramirez Pacheco, J.C.; Torres Roman, D. Local and Cumulative Analysis of Self-similar Traffic Traces. // Electronics, Communications and Computers, 2006. CONIELECOMP 2006. 16th International Conference on , vol., no., 01 Feb. 2006

[56] Petiz, I.; Rocha, E.; Salvador, P.; Nogueira, A. Using multiscale traffic analysis to detect WPS attacks. // Communications Workshops (ICC), 2013 IEEE International Conference on , vol., no., 9–13 June 2013

[57] Premarathne, U.; Premaratne, U.; Samarasinghe, K. Network traffic self similarity measurements using classifier based Hurst parameter estimation. // Information and Automation for Sustainability (ICIAFs), 2010 5th International Conference, 17–19 Dec. 2010 – pp. 64–69

[58] Z. Prusa and P. Rajmic. Real-Time lifting wavelet transform algorithm // Elektrorevue. – Brno: International Society for Science and Engineering, o.s., 2011 – vol. 2, no. 3 - pp. 53–59

[59] QiWei Lin, Zhenhui Liu, Gui Feng. DWT based on watermarking algorithm and its implementing with DSP. // Anti-counterfeiting, Security, and Identification in Communication, 2009. ASID 2009. 3rd International Conference, 20–22 Aug. 2009 – pp. 131–134

[60] B. Ryu, S. Lowen. Point Process Approaches for Modeling and Analysis of Self Similar Traffic, Part II – Applications. // Proceedings, International Conference on Telecommunications Systems, Modeling and Analysis. – March 1997.

[61] David Rincon Rivera. Contributions to the Wavelet-based Characterization of Network Traffic. // PhD Thesis, Program on Telematics Engineering – Universitat Politechnica de Catalunya – Barcelona, July 2007 – 315 p.

[62] Sardar, S.; Babu, K.A, Hardware Implementation of Real-Time, High Performance, RCE-NN Based Face Recognition System // VLSI Design and 2014 13th International Conference on Embedded Systems, 2014 27th International Conference, 5–9 Jan. 2014 – pp. 174–179. doi: 10.1109/VLSID.2014.37

[63] O. Sheluhin, S. Smolskiy, A. Osin. Self-Similar Processes in Telecommunications – John Wiley & Sons Ltd, 2007 – 314 p.

[64] Shu-Yan Chen; Wei Wang; Gao-Feng Qu. Combining wavelet transform and Markov model to forecast traffic volume. // Machine Learning and Cybernetics, 2004. Proceedings of 2004 International Conference on, vol. 5, no., 26–29 Aug. 2004 – pp. 2815–2818

[65] Deepika Sripath. Efficient Implementations of Discrete Wavelet Transforms Using FPGAs. Master's Thesis. // The Florida State University, DigiNole Commons, 2003 – 71 p.

[66] G. Strang, T. Ngueyen. Wavelets and filter banks. // Wellesley: Wellesley-Cambridge Press, 1996 – 495 p.

[67] Syed, A.R.; Burney, S.M.A.; Saleemi, A. Simulation and Analysis of Self-Similar Network Traffic Using WBFGN. // Environmental and Computer Science, 2009. ICECS '09. Second International Conference on , vol., no., 28–30 Dec. 2009 – pp. 148–151

[68] Tang-Hsuan Wang; Po-Tsang Huang; Kuan-Neng Chen; Jin-Chern Chiou; Kuo-Hua Chen; Chi-Tsung Chiu; Ho-Ming Tong; Ching-Te Chuang; Wei Hwang, Energy-efficient configurable discrete wavelet transform for neural sensing applications, // Circuits and Systems (ISCAS), 2014 IEEE International Symposium, 1–5 June 2014 – pp. 1841–1844. doi: 10.1109/ISCAS.2014.6865516

[69] Tereshchenko, T.; Yamnenko, Y.; Veretiuk, A.; Veretiuk, S. Wavelet transform at oriented basis for network traffic forecasting. // Electronics and Nanotechnology (ELNANO), 2013 IEEE XXXIII International Scientific Conference, vol., no., 16–19 April 2013 – pp. 450–454

[70] D. Veitch, P. Abry, M. Taqqu. On the Automatic Selection of the Onset of Scaling // FRACTALS. – World Scientific Publishing Company, 2003 – Vol. 11, No. 4 – P.377-390.

[71] Wang, K.; Li, X.; Ji, H.; Du, X. Modeling and Optimizing the LTE Discontinuous Reception Mechanism under Self-Similar Traffic. // Vehicular Technology, IEEE Transactions on, vol. PP, no. 99 – pp. 1–1

[72] F. Wenbing, G. Yingmin. FPGA Design of Fast Lifting Wavelet Transform // Image and Signal Processing, 2008. CISP '08. Congress, vol.4, 27-30 May 2008 – pp. 362–365. doi: 10.1109/CISP.2008.239

[73] Wen Juan; Sheng Min; Zhang Yan; Wang Xijun; Li Yuzhou. Traffic characteristics based dynamic radio resource management in heterogeneous wireless networks. // Communications, China , vol. 11, no. 1, Jan. 2014 – pp. 1–11

[74] V. C. Wilburn, W. E. Alexander. A parallel implementation of the discrete wavelet transform. // System Theory, 1994., Proceedings of the 26th Southeastern Symposium, 20–22 Mar 1994 – pp. 260–264. doi: 10.1109/SSST.1994.287872

[75] Ted Wobber, Thomas L. Rodeheffer and Douglas B. Terry. Policy-based Access Control for Weakly Consistent Replication // Microsoft Research Technical Report MSR-TR-2009-15, – Published February 2009, revised July 2009 – 17 p.

[76] Xiaolong Li; Hancheng Lu; Hao Lu. QoS Analysis of Self-Similar Multimedia Traffic with Variable Packet Size in Wireless Networks. // Vehicular Technology Conference (VTC Fall), 2013 IEEE 78th, vol., no., 2–5 Sept. 2013 – pp. 1–5

[77] Yanpu Hu; Wenfu Yang; Yong Wang; Ying Dong. The study of a new Call Admission Control method based on self-similar traffic. // Intelligent Control and Automation (WCICA), 2012 10th World Congress on, vol., no., 6–8 July 2012 – pp. 15–19

[78] Ye, X.; Xia, X.; Zhang, J.; Chen, Y. Effects of trends and seasonalities on robustness of the Hurst parameter estimators. // Signal Processing, IET , vol. 6, no. 9, Dec. 2012 – pp. 849–856

[79] Ye Xiaolong; Julong Lan; Wanwei Huang. Network traffic anomaly detection based on self-similarity using FRFT // Software Engineering and Service Science (ICSESS), 2013 4th IEEE International Conference on, vol., no., 23–25 May 2013 – pp. 837–840

[80] Zhong Fan; Mars, P. Self-similar traffic generation and parameter estimation using wavelet transform. // Global Telecommunications Conference, 1997. GLOBECOM '97., IEEE, vol. 3, no., 3–8 Nov 1997 – pp. 1419–1423

[81] Jānis Lošaks. Tīkla simulēšana un projektēšana izmantojot "Policy-Based" maršrutēšanu. // Bakalaura darbs ar projekta daļu, Rīgas Tehniskā Universitāte – 2014 – 80 p.

[82] E. Pētersons. Sevlīdzīgs trafiks telekomunikāciju un datoru tīklos. // Rīga: RTU Izdevniecība, 2005 – 61 lpp.

[83] Э.В. Афонцев, М.К. Гребенкин, С.В. Поршнев. О выборе размера буфера маршрутизатора компьютерной сети, нагруженного интенсивным трафиком реального времени // Известия Томского политехнического университета, Т.313 №5 – 2008 – с. 106–109

[84] Дж. Кеттель. Увеличение надежности при минимальных затратах. // Оптимальные задачи надежности. Под редакцией кандидата технических наук И.А.Ушакова. – Москва, 1968 – с. 29–43

[85] В.С.Коновалова. Измерение параметров локального сигнала методом дискретного вейвлет-преобразования в режиме реального времени. Автореферат диссертации на соискание ученой степени кандидата технических наук. // Санкт-Петербургский государственный электротехнический университет «ЛЭТИ» им. В.И. Ульянова (Ленина), 2012 – 18 с.

[86] Ю. Лосев, К. Руккас. Анализ моделей вероятности потери пакетов в буфере маршрутизатора с учетом фрактальности трафика. // Вестник Харьковского Национального Университета. Серия «Математическое моделирование. Информационные технологии. Автоматизированные системы управления». – Харьковский национальный университет им. В.Н. Каразина, 2008. – no. 10 – с. 163–169

[87] Э.Я.Петерсон, Н.Д.Путинцев. Прикладные задачи теории надежности автоматов. // Вопросы надежности дискретных автоматов. – Рига, 1970 – с. 121–132

[88] Ф.Прошан, Т.Брей. Оптимальное резервирование при наличии нескольких ограничений. // Оптимальные задачи надежности. Под редакцией кандидата технических наук И.А.Ушакова – Москва, 1968 – с. 92–109

[89] М. М. Родионов. Двухканальный банк фильтров на основе лестничных структур и арифметики с фиксированной запятой переменного формата. / М.М.Родионова, А.А.Петровский // Цифровая обработка сигналов. no. 2, Москва, 2010 – с. 12–18

[90] В. Столлингс. Современные компьютерные сети. 2-е издание. // Санкт-Петербург: Питер, 2003 – 783 с.

[91] Н. Н. Талеб. Черный лебедь. Под знаком непредсказуемости. / пер. с англ.: В. Сонькина, А. Бердичевского, М. Костионовой и др.; под. ред. М. Тюнькиной // Москва: Издательство КоЛибри, 2009 – 528 .

[92] Ч. Чуи. Введение в вейвлеты. / пер. с англ.: Я. М. Жилейкина // Москва: Издательство Мир, 2001 – 412 .

[93] Г.-Г. Штарк. Мир цифровой обработки. Применение вейвлетов для ЦОС. / пер. с англ.: Н. И. Смирновой; под. ред. д. ф.-м. н. А. Г. Кюркчана // Москва: Издательство Техносфера, 2007 – 192 .

[94] А. Н. Яковлев. Введение в вейвлет-преобразования: Учеб. пособие. // Новосибирск: Издательство НГТУ, 2003 – 104 .

[95] Cisco GNS3 Network Simulator // Internets: `http://www.gns3.com/`

[96] Mark Jackson. The Average Cost Per Megabit by Broadband Tech and Country for Q3 2013 // October 31-st, 2013 – Internet resource: http://www.ispreview.co.uk/index.php/2013/10/average-cost-per-megabit-broadband-tech-country-q3-2013.html.

[97] S. Sharkovsky,E. Grab. Modeling Self-Similar Traffic in Networks // RTU 52nd International Scientific Conference, Latvia, Riga, 13–14 Oct 2011 – pp. 19–19 – Internet: http://issuu.com/elan.grab/docs/modelling_self-similar_traffic_in_networks?mode=window&viewMode=singlePage

[98] J. Shortle, M. Fischer, D. Gross, D. Masi. Using the Pareto distribution in queueing modeling // submitted to Journal of Probability and Statistical Science – Mar. 2005 – Internet: http://www.noblis.org/missionareas/es/thoughtleadership/papers/documents/shortle_pareto.pdf

[99] K. Valigura, M. Foltin, M. Blaho. Transport system realization in SimEvents tool // Slovak University of Technology in Bratislava, Faculty of electrical engineering and information technology. – Internet: http://logi.upce.cz/issues/2011-01/brumercik.pdf

[100] GPSS World imitācijas datorprogrammas mājaslapa. – Internets: http://www.minutemansoftware.com/

Pielikumi

1.pielikums. Simulācijas datu apstrādes programma *Matlab* vidē

```
clc, clear all, format short g
%Parameters
H=[0.5; 0.6; 0.7; 0.8; 0.9; 0.99]; ro=0.8;

%Estimation of buffer length
K=H./(1-H); K=(1-ro).^K; Tab(1:6,4)=1./(1-H)/2;
Tab(:,4)=ro.^Tab(:,4)./K;

%First filename selection
filename='H05\data_ro05_H05.mat';
if ro==1.0, a=int2str(1); b=int2str(0); else
    a=int2str(0); b=int2str(ro*10); end
filename(12:13)=[a,b];

for i=5:10
%Next file selection
    if i==10, a=[filename(1:3),int2str(9)];
        a=[a,filename(4:17),int2str(9)];
        filename=[a,filename(18:21)];
    else a=int2str(i);
        filename(3)=a;
        filename(17)=a;
    end
%Data extraction and calculation
    load(filename,'K','Queue');
    Tab(i-4,1)=K;
    Tab(i-4,2)=mean(Queue);
    Tab(i-4,3)=max(Queue);
end
```

2.pielikums. Sevlīdzīguma parametra H novērtēšana pēc pilnās trafika realizācijas *Matlab* vidē

```
clc, format long, clear all
rand('state',12345);

H = 0.9; alfa = 3 - 2*H;
lambda = [1 10 100 1000]; lambda = 50;
for k=1:length(alfa)
    betta(k,:) = (alfa(k) - 1)./lambda.'./alfa(k);
end
filename = 'TrafficOutNewH_2012_'; fileext = '.mat';

for Hind = 1:length(H);
    alf = alfa(Hind);
    file = [filename num2str(H(Hind)) fileext];

    N = 1; % Number of starts (for averaging)

    % Time - 1 day
    T = 3600*24; dt = [0.001 0.01 0.1 1 10];
    T = 50000;
    dt = 1;
    Hem = zeros(length(dt),length(lambda));
    clear packets;
    for n=1:N
        disp(['n = ' num2str(n)]);
        for i = 1:length(dt), for k=1:length(lambda),
                %rand('state',lambda(k)*k*n);
                bt = betta(Hind,k);
                packets{n,i}(k,:) = SelfSimilarOut(alf,bt,T,dt(i));
                He(i,k) = EstimHbyWave_Abry(packets{n,i}(k,:));
        end, end
        He
        Hem = Hem + He;
    end
    Hema{Hind} = Hem / N % mean!
    Hem = 0;
end
```

3.pielikums. Tiešās un inversās diskrētās veivletu transformācijas filtru bankas realizācijas *C++* valodas funkciju veidā: tiešā, ar polifāzes un kāpņveida filtriem

```
//Full filtering bank without downsampling
//H0 - lowpass filter IR
//H1 - highpass filter IR
//MEM - shift register (delay line)
//L - impulse response length
//N - filter order
//LL - downsampled impulse response length
//NN - downsampled impulse response filter order

void rt_filbank(double x, double *H0, double *H1, double *MEM,
double &y_a, double &y_d, int N){
int j;
y_a=H0[0]*x;                        //1-st sample of IR
y_d=H1[0]*x;
for(j=N;j>0;j--){
MEM[j]=MEM[j-1];                    //Shift of memory
y_a=y_a+MEM[j]*H0[j];               //1-st channel filtering
y_d=y_d+MEM[j]*H1[j];}               //2-nd channel filtering
MEM[0]=x;
return;
}

//This function makes shift only - useful for skipping out downsampled outputs
void filbank_delay(double x, double *MEM, int N){
int j;
for(j=N;j>0;j--) MEM[j]=MEM[j-1];                    //Shift of memory
MEM[0]=x;
return;}

//Multiplication to calculate y_a and y_d quotients (no shift, to be made separately)
void filbank_multiply(double *H0, double *H1, double *MEM,
double &y_a, double &y_d, int N){
int j; double x=MEM[0];
y_a=H0[0]*x;                          //1-st sample of IR
```

```
y_d=H1[0]*x;
for(j=N;j>0;j--){
y_a=y_a+MEM[j]*H0[j];             //1-st channel filtering
y_d=y_d+MEM[j]*H1[j];}            //2-nd channel filtering
return;
}

//Inverse direct for for fast DWT with two filters
void inv_rt_filbank(double x_a, double x_d, double *F0, double *F1,
double *MEM0, double *MEM1, double &y_a, int N){
int j; double y_0, y_1;
y_0=F0[0]*x_a;                      //1-st sample of IR
y_1=F1[0]*x_d;
for(j=N;j>0;j--){
MEM0[j] = MEM0[j-1];                //Shift of memory
MEM1[j] = MEM1[j-1];
y_0 = y_0 + MEM0[j]*F0[j];            //1-st channel filtering
y_1 = y_1 + MEM1[j]*F1[j];}           //2-nd channel filtering
MEM0[0] = x_a; MEM1[0] = x_d; y_a = y_0 + y_1;
return;
}

//Direct transform, Polyphase implementation, odd samples saved
void filbank_polyphase_odd(double x_ev, double x_odd, double *H0, double *H1,
double *MEM_ev, double *MEM_odd, double &y_a, double &y_d, int NN){
int j, n_ev, n_odd;
double y_a0, y_a1, y_d0, y_d1;
y_a0 = x_ev*H0[1]; y_a1 = x_odd*H0[0]; //Initial values of lowpass components
y_d0 = x_ev*H1[1]; y_d1 = x_odd*H1[0]; //Initial values of highpass components
for(j=NN;j>0;j--)
{
n_ev=2*j; n_odd=n_ev+1;
MEM_ev[j] = MEM_ev[j-1]; MEM_odd[j] = MEM_odd[j-1]; //Memory shifting
y_a0 = y_a0 + MEM_ev[j]*H0[n_odd]; //Lowpass Even*Odd component
y_a1 = y_a1 + MEM_odd[j]*H0[n_ev]; //Lowpass Odd*Even component
y_d0 = y_d0 + MEM_ev[j]*H1[n_odd]; //Highpass Even*Odd component
y_d1 = y_d1 + MEM_odd[j]*H1[n_ev]; //Highpass Odd*Even component
}
MEM_ev[0] = x_ev; MEM_odd[0] = x_odd; //Finalizing shift for MEM[0]
y_a = y_a0 + y_a1; y_d = y_d0 + y_d1; //Adding components
```

```
}

//Direct transform, Polyphase implementation, even samples saved
void filbank_polyphase_even(double x_ev, double x_odd, double *H0, double *H1,
double *MEM_ev, double *MEM_odd, double &y_a, double &y_d, int NN){
int j, n_ev, n_odd;
double y_a0, y_a1, y_d0, y_d1;
y_a0 = x_ev*H0[0]; y_a1 = x_odd*H0[1]; //Initial values of lowpass components
y_d0 = x_ev*H1[0]; y_d1 = x_odd*H1[1]; //Initial values of highpass components
for(j=NN;j>0;j--)
{
n_ev=2*j; n_odd=n_ev+1;
MEM_ev[j] = MEM_ev[j-1]; MEM_odd[j] = MEM_odd[j-1]; //Memory shifting
y_a0 = y_a0 + MEM_ev[j]*H0[n_ev]; //Lowpass Even*Even component
y_a1 = y_a1 + MEM_odd[j]*H0[n_odd]; //Lowpass Odd*Odd component
y_d0 = y_d0 + MEM_ev[j]*H1[n_ev]; //Highpass Even*Even component
y_d1 = y_d1 + MEM_odd[j]*H1[n_odd]; //Highpass Odd*Odd component
}
MEM_ev[0] = x_ev; MEM_odd[0] = x_odd; //Finalizing shift for MEM[0]
y_a = y_a0 + y_a1; y_d = y_d0 + y_d1; //Adding components
}

//Inverse transform, Polyphase implementation
void filbank_inverse_polyphase(double y_a, double y_d, double *F0, double *F1,
double *MEM_a, double *MEM_d, double &y_ev, double &y_odd, int NN){
int j, n_ev, n_odd;
double y_a0, y_a1, y_d0, y_d1;
y_a0 = y_a*F0[0]; y_a1 = y_a*F0[1]; //Initial values of lowpass components
y_d0 = y_d*F1[0]; y_d1 = y_d*F1[1]; //Initial values of highpass components
for(j=NN;j>0;j--)
{
n_ev=2*j; n_odd=n_ev+1;
MEM_a[j] = MEM_a[j-1]; MEM_d[j] = MEM_d[j-1]; //Memory shifting
y_a0 = y_a0 + MEM_a[j]*F0[n_ev]; //Lowpass Even component
y_a1 = y_a1 + MEM_a[j]*F0[n_odd]; //Lowpass Odd component
y_d0 = y_d0 + MEM_d[j]*F1[n_ev]; //Highpass Even component
y_d1 = y_d1 + MEM_d[j]*F1[n_odd]; //Highpass Odd component
}
MEM_a[0] = y_a; MEM_d[0] = y_d; //Finalizing shift for MEM[0]
y_ev = y_a0 + y_d0; y_odd = y_a1 + y_d1; //Adding components
```

```
}


//Lattice filter bank, polyphase
void filbank_lattice(double x_ev, double x_odd, double *k, double A,
double *MEM, double &y_a, double &y_d, int NN){
int j; double T;
MEM[0] = x_ev; T = MEM[0]; y_a = x_odd;
for(j=0; j<NN; j++)
{
y_d = MEM[j] - k[j]*y_a;
y_a = y_a + k[j]*MEM[j];
MEM[j] = T; T = y_d;
}
y_a *= A; y_d *= A;
}
```

4.pielikums. Tiešās diskrētās J-mērogu veivletu transformācijas filtru bankas realizācija *C++* valodā

```
#include <math.h>
#include <iostream>
#include <windows.h>
#include <stdlib.h>
#include "filters.cpp"
#include <fstream>
using namespace std;
#define K 128          //Number of data points (16777216)
#define L 6           //Length of filters impulse response
#define J 3           //Number of DWT levels

int main(){
// Input file
ifstream file1("info.txt");
if(!file1.is_open()) {cout<<"Input file error!\n"; system("PAUSE"); return 0;}

/*
//Output files
ofstream file2("approx_lvl8.txt");
if(!file2.is_open()) {cout<<"Output file error!\n"; system("PAUSE"); return 0;}
ofstream file3("detail_lvl8.txt");
if(!file3.is_open()) {cout<<"Output file error!\n"; system("PAUSE"); return 0;} */

int i,j,s=-1, MOD, hMOD; double y_a,y_d,x_a,yy;
double MEM[J][L];
for(i=0; i<L; i++) for(j=0;j<J;j++) MEM[j][i]=0;     //Memory
int N=L-1;


//Impulse Responses
double H1[L],F0[L],F1[L],t0,t1;
double H0[L]={0.035226291882100656, -0.085441273882241486,
-0.13501102001039084,0.45987750211933132,0.80689150931333875,
0.33267055295095688};
for(j=0;j<L;j++){
```

```
F0[j]=H0[L-j-1];        //Synthesis Approx. Filter
H1[j]=F0[j]*s;        //Analysis Detail. Filter
F1[j]=-H0[j]*s;       //Synthesis Detail. Filter
s=-s;}

//Time measurement
LARGE_INTEGER StartingTime, EndingTime, Frequency;
double ElapsedTime;

QueryPerformanceFrequency(&Frequency);
QueryPerformanceCounter(&StartingTime); //Start Timer


//Processing L-level DWT
for(i=0;i<K;i++){
MOD=2; hMOD=1; x_a=i; j=0;
while(i%hMOD == hMOD-1 && j<J){ //Checking the rem(i,MOD/2).
Skip level, if not MAXed (leaving odd numbered)
filbank_delay(x_a, MEM[j], N); //If processing level, shifting in all cases

//Calculating quotients only if sample is left after downsampling
if(i%MOD == MOD-1)
{
filbank_multiply(H0, H1, MEM[j], y_a, y_d, N);
if ((j+1)==1) cout<<"y_a = "<<y_a<<"\t\t"<<"y_d = "<<y_d<<"\n";
}
hMOD=MOD; MOD*=2; x_a=y_a; j++;
}
}

QueryPerformanceCounter(&EndingTime); //Stop Timer
//
// We now have the elapsed number of ticks, along with the
// number of ticks-per-second. We use these values
// to convert to the number of elapsed miliseconds.
// To guard against loss-of-precision, we convert
// to microseconds *before* dividing by ticks-per-second.
//
ElapsedTime=(EndingTime.QuadPart-StartingTime.QuadPart)*1000.0/Frequency.QuadPart;
```

```
//Finalization
cout<<"Total time: "<<ElapsedTime/1000<<" sec\n";
cout<<"Single transformation average time: "<<ElapsedTime*1000/K<<" microsec\n\n\n";
file1.close();
//    file2.close(); file3.close();
return 0;
}
```

5.pielikums. Inversās diskrētās J-mērogu veivletu transformācijas filtru bankas realizācija $C++$ valodā

```
#include <math.h>
#include <iostream>
#include <windows.h>
#include <stdlib.h>
#include <stdio.h>
#include <iomanip>
#include "filters.cpp"
using namespace std;
#define K 2097152   //Number of data points (16777216)
#define L 6           //Length of filters impulse response
#define J 20          //Number of DWT levels

int main(){
int i,j,k,KK[J],L_delay[J],kk,MOD,num,num_L,s;
double koeff, x_a, x_d, y;
int N=L-1;

double **registr = new double* [J]; kk=1; s=0;
//Initialize array
for(j=J-1;j>=0;j--){
registr[j] = new double [kk]; //registr - buffer to store quotients
for(k=0;k<kk;k++) registr[j][k]=0;
KK[j]=kk; kk=2*kk; //KK - array length for each register
L_delay[j]=s; s=2*s+N-1;} //Array for filter delays at each scale

int K_out = kk;    //Length of output register
double *y_a = new double [K_out]; //For output data samples storage
for(i=0;i<K_out;i++) y_a[i]=0; //Setting to zeroes

double **delay = new double* [J-1];
for(j=J-2;j>=0;j--) {
delay[j] = new double [L_delay[j]]; //Delay lines for perfect reconstruction
for(k=0;k<L_delay[j];k++) delay[j][k] = 0; } //setting to zeroes

double MEM0[J][L], MEM1[J][L];
```

```
for(i=0; i<L; i++) for(j=0;j<J;j++) {MEM0[j][i]=0; MEM1[j][i]=0;}   //Memory cleaning

//Impulse Responses
double H1[L],F0[L],F1[L],t0,t1; s=-1;
/* double H0[L]={-0.010597401784993, 0.032883011666979, 0.030841381835980,
-0.187034811718880, -0.027983769416982, 0.630880767929596,
0.714846570552543, 0.230377813308853}; */
double H0[L]={0.035226291882100656, -0.085441273882241486,
-0.13501102001039084,0.45987750211933132,0.80689150931333875,
0.33267055295095688};
// double H0[L] = {0.707106781186548, 0.707106781186548};
for(j=0;j<L;j++){
F0[j]=H0[L-j-1];        //Synthesis Approx. Filter
H1[j]=F0[j]*s;        //Analysis Detail. Filter
F1[j]=-H0[j]*s;       //Synthesis Detail. Filter
s=-s;}

//Creating file for write/read
FILE *data;
if((data=fopen("coefficients.bin", "rb"))==NULL) {printf("File error!\n"); exit(1);
}

//Time measurement
LARGE_INTEGER StartingTime, EndingTime, Frequency;
double ElapsedTime;

QueryPerformanceFrequency(&Frequency);
QueryPerformanceCounter(&StartingTime); //Start Timer

for(i=0;i<K;i++){
if(i%2==1)  //Skipping all even-numbered samples, to be interpolated by zeroes
{
j=0; MOD=2;
while(i%MOD==MOD-1 && j<J-1) { //Defining number of scales based on //sample number,
limited to maximum of J
//koeff=i;
fread(&koeff,sizeof(double),1,data); //Data read, one detail quotient
// y = koeff; //For Haar wavelets, also to comment the next 2 lines!
num=(i-1)/MOD;
num_L=num%L_delay[j];
```

```
y=delay[j][num_L]; delay[j][num_L]=koeff;   //Delay of detail quotient
num=(i-1)/MOD; num=num%KK[j]; registr[j][num] = y; //Save delayed d[j][i]
//quotient in buffer
MOD=MOD*2; j++;}
if(j==J-1 && i%MOD==MOD-1){ //Perform DWT on maximum scale j = J
fread(&koeff,sizeof(double),1,data); //Data read, detail quotient at scale j = J
registr[j][0] = koeff;
fread(&koeff,sizeof(double),1,data); //Data read, one (and only) approximation quotient
y_a[0] = koeff; MOD=KK[0];
for(j=J-1;j>=0;j--){ num=0; //Starting with scale J
for(k=0;k<KK[j];k++){ //Processing all samples over scale
//Filter bank for x_a and x_d
x_a = y_a[num]; x_d = registr[j][k];
inv_rt_filbank(x_a, x_d, F0, F1, MEM0[j], MEM1[j], y, N);
//Delay implemented at buffering of quotients
y_a[num] = y; num = num + MOD;  //Advancing Num in y_a[Num]
//Filter bank for zero sample
x_a = 0; x_d = 0;
inv_rt_filbank(x_a, x_d, F0, F1, MEM0[j], MEM1[j], y, N);
y_a[num] = y; num = num + MOD;} //Advancing Num in y_a[Num]
MOD = MOD/2;
}
//printf("\n"); for(k=0; k<K_out; k++) printf("%f\n",y_a[k]);
}
}
}

QueryPerformanceCounter(&EndingTime); //Stop Timer
//
// We now have the elapsed number of ticks, along with the
// number of ticks-per-second. We use these values
// to convert to the number of elapsed miliseconds.
// To guard against loss-of-precision, we convert
// to microseconds *before* dividing by ticks-per-second.
//
ElapsedTime=(EndingTime.QuadPart-StartingTime.QuadPart)*1000.0/Frequency.QuadPart;
printf("Total time: %f sec\n", ElapsedTime/1000);
printf("Single transformation point average time:
%f microsec\n\n\n", ElapsedTime*1000/K);
```

```
//Clear array
for (i = 0; i < J; i++) delete [] registr[i];
for (i = 0; i < J-1; i++) delete [] delay[i];
delete [] y_a;
fclose(data);
return 0;
}
```

6.pielikums. Tiešās diskrētās J-Mērogu veivletu transformācijas filtru bankas realizācija $C++$ valodā ar polifāzes filtriem

```
#include <math.h>
#include <iostream>
#include <windows.h>
#include <stdlib.h>
#include "filters.cpp"
#include <fstream>
using namespace std;
#define K 128          //Number of data points (16777216)
#define L 6           //Length of filters impulse response
#define J 3           //Number of DWT levels

int main(){
// Input file
ifstream file1("info.txt");
if(!file1.is_open()) {cout<<"Input file error!\n"; system("PAUSE"); return 0;}

/*
//Output files
ofstream file2("approx_lvl8.txt");
if(!file2.is_open()) {cout<<"Output file error!\n"; system("PAUSE"); return 0;}
ofstream file3("detail_lvl8.txt");
if(!file3.is_open()) {cout<<"Output file error!\n"; system("PAUSE"); return 0;}
*/

int N=L-1, LL=L/2, NN=LL-1;
int i,j,s=-1, MOD, hMOD; double y_a,y_d,x_a,yy,x_odd;
double MEM_ev[J][LL], MEM_odd[J][LL], x_ev[J];
for(i=0; i<LL; i++) for(j=0;j<J;j++) {MEM_ev[i][j]=0; MEM_odd[i][j]=0;}    //Memory


//Impulse Responses
double H1[L],F0[L],F1[L],t0,t1;
double H0[L]={0.035226291882100656, -0.085441273882241486,
-0.13501102001039084,0.45987750211933132,0.80689150931333875,
0.33267055295095688};
```

```
for(j=0;j<L;j++){
F0[j]=H0[L-j-1];        //Synthesis Approx. Filter
H1[j]=F0[j]*s;        //Analysis Detail. Filter
F1[j]=-H0[j]*s;       //Synthesis Detail. Filter
s=-s;
// cout<<"H0["<<j<<"] = "<<H0[j]<<" ";
// cout<<"H1["<<j<<"] = "<<H1[j]<<" ";
// cout<<"F0["<<j<<"] = "<<F0[j]<<" ";
// cout<<"F1["<<j<<"] = "<<F1[j]<<"\n";
}

//Time measurement
LARGE_INTEGER StartingTime, EndingTime, Frequency;
double ElapsedTime;

QueryPerformanceFrequency(&Frequency);
QueryPerformanceCounter(&StartingTime); //Start Timer


//Processing L-level DWT
for(i=0;i<K;i++){
MOD=2; x_a=i; hMOD=1;
for(j=0;j<J;j++){
if(i%hMOD==hMOD-1){ //Checking the rem(i,MOD/2). Skip level, if not MAXed
// (leaving odd numbered)
//Calculating quotients only if sample is left after downsampling
if(i%(MOD)==MOD-1)
{
x_odd=x_a; //Being processed immediately, no need to save
filbank_polyphase_odd(x_ev[j], x_odd, H0, H1, MEM_ev[j], MEM_odd[j], y_a, y_d, NN);
//Output here y_a - approx., y_d - detail

if ((j+1)==1) cout<<"y_a = "<<y_a<<"\t\t"<<"y_d = "<<y_d<<"\n";
//if ((j+1)==9) {file2<<x_a<<"\n"; file3<<y_d<<"\n";}
}
else x_ev[j]=x_a; //Otherwise, saving even sample. Needs to be saved between
// samples for all scales (levels)!
}
else break;
hMOD=MOD; MOD*=2; x_a=y_a;
```

```
}
}


QueryPerformanceCounter(&EndingTime); //Stop Timer
//
// We now have the elapsed number of ticks, along with the
// number of ticks-per-second. We use these values
// to convert to the number of elapsed miliseconds.
// To guard against loss-of-precision, we convert
// to microseconds *before* dividing by ticks-per-second.
//
ElapsedTime=(EndingTime.QuadPart-StartingTime.QuadPart)*1000.0/Frequency.QuadPart;

//Finalization
cout<<"Total time: "<<ElapsedTime/1000<<" sec\n";
cout<<"Single transformation average time: "<<ElapsedTime*1000/K<<" microsec\n\n\n";
file1.close();
//    file2.close(); file3.close();
return 0;
}
```

7.pielikums. Inversās diskrētās J-mērogu veivletu transformācijas filtru bankas realizācija *C++* valodā ar polifāzes filtriem

```
#include <math.h>
#include <iostream>
#include <windows.h>
#include <stdlib.h>
#include <stdio.h>
#include <iomanip>
#include "filters.cpp"
using namespace std;
#define K 16777216   //Number of data points (16777216)
#define L 6           //Length of filters impulse response
#define J 15           //Number of DWT levels

int main(){
int i,j,k,KK[J],L_delay[J],kk,MOD,num,num_L,s,LL=L/2,NN=LL-1;
double koeff, x_a, x_d, y, y_ev, y_odd;
int N=L-1;

double **registr = new double* [J]; kk=1; s=0;
//Initialize array
for(j=J-1;j>=0;j--){
registr[j] = new double [kk]; //registr - buffer to store quotients
for(k=0;k<kk;k++) registr[j][k]=0;
KK[j]=kk; kk=2*kk; //KK - array length for each register
L_delay[j]=s; s=2*s+N-1;} //Array for filter delays at each scale

int K_out = kk;    //Length of output register
double *y_a = new double [K_out]; //For output data samples storage
for(i=0;i<K_out;i++) y_a[i]=0; //Setting to zeroes

double **delay = new double* [J-1];
for(j=J-2;j>=0;j--) {
delay[j] = new double [L_delay[j]]; //Delay lines for perfect reconstruction
for(k=0;k<L_delay[j];k++) delay[j][k] = 0; } //setting to zeroes

double MEM0[J][LL], MEM1[J][LL];
```

```
for(i=0; i<LL; i++) for(j=0;j<J;j++) {MEM0[j][i]=0; MEM1[j][i]=0;}   //Memory cleaning

//Impulse Responses
double H1[L],F0[L],F1[L],t0,t1; s=-1;
/* double H0[L]={-0.010597401784993, 0.032883011666979, 0.030841381835980,
-0.187034811718880, -0.027983769416982, 0.630880767929596,
0.714846570552543, 0.230377813308853}; */
double H0[L]={0.035226291882100656, -0.085441273882241486,
-0.13501102001039084,0.45987750211933132,0.80689150931333875,
0.33267055295095688};
// double H0[L] = {0.707106781186548, 0.707106781186548};
for(j=0;j<L;j++){
F0[j]=H0[L-j-1];        //Synthesis Approx. Filter
H1[j]=F0[j]*s;        //Analysis Detail. Filter
F1[j]=-H0[j]*s;       //Synthesis Detail. Filter
s=-s;}

//Creating file for write/read
FILE *data;
if((data=fopen("coefficients.bin", "rb"))==NULL) {printf("File error!\n"); exit(1);
}

//Time measurement
LARGE_INTEGER StartingTime, EndingTime, Frequency;
double ElapsedTime;

QueryPerformanceFrequency(&Frequency);
QueryPerformanceCounter(&StartingTime); //Start Timer

for(i=0;i<K;i++){
if(i%2==1)  //Skipping all even-numbered samples, to be interpolated by zeroes
{
j=0; MOD=2;
while(i%MOD==MOD-1 && j<J-1) { //Defining number of
//scales based on sample number, limited to maximum of J
koeff=i;
// fread(&koeff,sizeof(double),1,data); //Data read,
//one detail quotient
// y = koeff; //For Haar wavelets, also to comment the next 2 lines!
num=(i-1)/MOD;
```

```
num_L=num%L_delay[j];
y=delay[j][num_L]; delay[j][num_L]=koeff;   //Delay of detail quotient
num=num%KK[j]; registr[j][num] = y; //Save delayed d[j][i] quotient in buffer
MOD=MOD*2; j++;}
if(j==J-1 && i%MOD==MOD-1){ //Perform DWT on
//maximum scale j = J
// fread(&koeff,sizeof(double),1,data); //Data read, detail
//quotient at scale j = J
registr[j][0] = koeff;
// fread(&koeff,sizeof(double),1,data); //Data read,
//one (and only) approximation quotient
y_a[0] = koeff; MOD=KK[0];
for(j=J-1;j>=0;j--){ num=0; //Starting with scale J
for(k=0;k<KK[j];k++){ //Processing all samples over scale
//Filter bank for x_a and x_d
x_a = y_a[num]; x_d = registr[j][k];
filbank_inverse_polyphase(x_a, x_d, F0, F1, MEM0[j], MEM1[j], y_ev, y_odd, NN);
//Delay implemented at buffering of quotients
y_a[num] = y_ev; num = num + MOD;  //Advancing Num in y_a[Num]
y_a[num] = y_odd; num = num + MOD;} //Advancing Num in y_a[Num]
MOD = MOD/2;
}
// printf("\n"); for(k=0; k<K_out; k++) printf("%f\n",y_a[k]);
}
}
}

QueryPerformanceCounter(&EndingTime); //Stop Timer
//
// We now have the elapsed number of ticks, along with the
// number of ticks-per-second. We use these values
// to convert to the number of elapsed miliseconds.
// To guard against loss-of-precision, we convert
// to microseconds *before* dividing by ticks-per-second.
//
ElapsedTime=(EndingTime.QuadPart-StartingTime.QuadPart)*1000.0/Frequency.QuadPart;
printf("Total time: %f sec\n", ElapsedTime/1000);
printf("Single transformation point average time: %f microsec\n\n\n", ElapsedTime*1000/K);

//Clear array
```

```
for (i = 0; i < J; i++) delete [] registr[i];
for (i = 0; i < J-1; i++) delete [] delay[i];
delete [] y_a;
fclose(data);
return 0;
}
```

8.pielikums. Tiešās diskrētās J-mērogu veivletu transformācijas filtru bankas realizācija $C++$ valodā ar kāpņveida filtriem

```
#include <math.h>
#include <iostream>
#include <windows.h>
#include <stdlib.h>
#include "filters.cpp"
#include <fstream>
using namespace std;
#define K 16777216          //Number of data points (16777216)
#define L 6           //Length of filters impulse response
#define J 15           //Number of DWT levels

int main(){
// Input file
ifstream file1("info.txt");
if(!file1.is_open()) {cout<<"Input file error!\n"; system("PAUSE"); return 0;}

/*
//Output files
ofstream file2("approx_lvl8.txt");
if(!file2.is_open()) {cout<<"Output file error!\n"; system("PAUSE"); return 0;}
ofstream file3("detail_lvl8.txt");
if(!file3.is_open()) {cout<<"Output file error!\n"; system("PAUSE"); return 0;}
*/

int N=L-1, LL=L/2, NN=LL-1;
int i,j,s=-1, MOD, hMOD; double y_a,y_d,x_a,yy,x_odd;
double MEM[J][LL], x_ev[J];
for(i=0; i<LL; i++) for(j=0;j<J;j++) MEM[i][j]=0;     //Memory


//Impulse Responses
double H1[L],F0[L],F1[L],t0,t1;
double k[L/2] = {-2.425497243932123,0.546096402964684,9.443814127935182};
double H0[L]={0.035226291882100656, -0.085441273882241486,
-0.13501102001039084,0.45987750211933132,0.80689150931333875,
```

```
0.33267055295095688};
for(j=0;j<L;j++){
F0[j]=H0[L-j-1];        //Synthesis Approx. Filter
H1[j]=F0[j]*s;         //Analysis Detail. Filter
F1[j]=-H0[j]*s;        //Synthesis Detail. Filter
s=-s;
// cout<<"H0["<<j<<"] = "<<H0[j]<<" ";
// cout<<"H1["<<j<<"] = "<<H1[j]<<" ";
// cout<<"F0["<<j<<"] = "<<F0[j]<<" ";
// cout<<"F1["<<j<<"] = "<<F1[j]<<"\n";
}

//Time measurement
LARGE_INTEGER StartingTime, EndingTime, Frequency;
double ElapsedTime;

QueryPerformanceFrequency(&Frequency);
QueryPerformanceCounter(&StartingTime); //Start Timer


//Processing L-level DWT
for(i=0;i<K;i++){
MOD=2; x_a=i; hMOD=1;
for(j=0;j<J;j++){
if(i%hMOD==hMOD-1){ //Checking the rem(i,MOD/2). Skip level, if not MAXed
// (leaving odd numbered)
//Calculating quotients only if sample is left after downsampling
if(i%(MOD)==MOD-1)
{
x_odd=x_a; //Being processed immediately, no need to save
filbank_lattice(x_ev[j], x_odd, k, H0[0], MEM[j], y_a, y_d, LL);
//Output here y_a - approx., y_d - detail

//if ((j+1)==3) cout<<"y_a = "<<y_a<<"\t\t"<<"y_d = "<<y_d<<"\n";
//if ((j+1)==9) {file2<<x_a<<"\n"; file3<<y_d<<"\n";}
}
else x_ev[j]=x_a; //Otherwise, saving even sample. Needs to be saved between
// samples for all scales (levels)!
}
else break;
```

```
hMOD=MOD; MOD*=2; x_a=y_a;
}
}


QueryPerformanceCounter(&EndingTime); //Stop Timer
//
// We now have the elapsed number of ticks, along with the
// number of ticks-per-second. We use these values
// to convert to the number of elapsed miliseconds.
// To guard against loss-of-precision, we convert
// to microseconds *before* dividing by ticks-per-second.
//
ElapsedTime=(EndingTime.QuadPart-StartingTime.QuadPart)*1000.0/Frequency.QuadPart;

//Finalization
cout<<"Total time: "<<ElapsedTime/1000<<" sec\n";
cout<<"Single transformation average time:
"<<ElapsedTime*1000/K<<" microsec\n\n\n";
file1.close();
//    file2.close(); file3.close();
return 0;
}
```

9.pielikums. Inversās diskrētās J-mērogu veivletu transformācijas filtru bankas realizācija $C++$ valodā ar kāpņveida filtriem

```
#include <math.h>
#include <iostream>
#include <windows.h>
#include <stdlib.h>
#include <stdio.h>
#include <iomanip>
#include "filters.cpp"
using namespace std;
#define K 16777216   //Number of data points (16777216)
#define L 6           //Length of filters impulse response
#define J 15           //Number of DWT levels

int main(){
int i,j,k,KK[J],L_delay[J],kk,MOD,num,num_L,s,LL=L/2,NN=LL-1;
double koeff, x_a, x_d, y, y_ev, y_odd;
int N=L-1;

double **registr = new double* [J]; kk=1; s=0;
//Initialize array
for(j=J-1;j>=0;j--){
registr[j] = new double [kk]; //registr - buffer to store quotients
for(k=0;k<kk;k++) registr[j][k]=0;
KK[j]=kk; kk=2*kk; //KK - array length for each register
L_delay[j]=s; s=2*s+N-1;} //Array for filter delays at each scale

int K_out = kk;    //Length of output register
double *y_a = new double [K_out]; //For output data samples storage
for(i=0;i<K_out;i++) y_a[i]=0; //Setting to zeroes

double **delay = new double* [J-1];
for(j=J-2;j>=0;j--) {
delay[j] = new double [L_delay[j]]; //Delay lines for perfect reconstruction
for(k=0;k<L_delay[j];k++) delay[j][k] = 0; } //setting to zeroes

double MEM[J][LL];
```

```
for(i=0; i<LL; i++) for(j=0;j<J;j++) MEM[j][i]=0;    //Memory cleaning

//Impulse Responses
double H1[L],F0[L],F1[L],t0,t1; s=-1;
double k_inv[L/2] = {9.443814127935182,0.546096402964684,-2.425497243932123};
/* double H0[L]={-0.010597401784993, 0.032883011666979, 0.030841381835980,
-0.187034811718880, -0.027983769416982, 0.630880767929596,
0.714846570552543, 0.230377813308853}; */
double H0[L]={0.035226291882100656, -0.085441273882241486,
-0.13501102001039084,0.45987750211933132,0.80689150931333875,
0.33267055295095688};
// double H0[L] = {0.707106781186548, 0.707106781186548};
for(j=0;j<L;j++){
F0[j]=H0[L-j-1];        //Synthesis Approx. Filter
H1[j]=F0[j]*s;        //Analysis Detail. Filter
F1[j]=-H0[j]*s;       //Synthesis Detail. Filter
s=-s;}

//Creating file for write/read
FILE *data;
if((data=fopen("coefficients.bin", "rb"))==NULL) {printf("File error!\n"); exit(1);
}

//Time measurement
LARGE_INTEGER StartingTime, EndingTime, Frequency;
double ElapsedTime;

QueryPerformanceFrequency(&Frequency);
QueryPerformanceCounter(&StartingTime); //Start Timer

for(i=0;i<K;i++){
if(i%2==1)  //Skipping all even-numbered samples, to be interpolated by zeroes
{
j=0; MOD=2;
while(i%MOD==MOD-1 && j<J-1) { //Defining number of scales
//based on sample number, limited to maximum of J
koeff=i;
// fread(&koeff,sizeof(double),1,data); //Data read,
//one detail quotient
// y = koeff; //For Haar wavelets, also to comment the next 2 lines!
```

```
num=(i-1)/MOD;
num_L=num%L_delay[j];
y=delay[j][num_L]; delay[j][num_L]=koeff;   //Delay of detail quotient
num=num%KK[j]; registr[j][num] = y; //Save delayed d[j][i] quotient in buffer
MOD=MOD*2; j++;}
if(j==J-1 && i%MOD==MOD-1){ //Perform DWT on maximum
//scale j = J
// fread(&koeff,sizeof(double),1,data); //Data read,
//detail quotient at scale j = J
registr[j][0] = koeff;
// fread(&koeff,sizeof(double),1,data); //Data read,
//one (and only) approximation quotient
y_a[0] = koeff; MOD=KK[0];
for(j=J-1;j>=0;j--){ num=0; //Starting with scale J
for(k=0;k<KK[j];k++){ //Processing all samples over scale
//Filter bank for x_a and x_d
x_a = y_a[num]; x_d = registr[j][k];
filbank_lattice(x_a, x_d, k_inv, H0[0], MEM[j], y_ev, y_odd, LL);
//Delay implemented at buffering of quotients
y_a[num] = y_ev; num = num + MOD;  //Advancing Num in y_a[Num]
y_a[num] = y_odd; num = num + MOD;} //Advancing Num in y_a[Num]
MOD = MOD/2;
}
// printf("\n"); for(k=0; k<K_out; k++) printf("%f\n",y_a[k]);
}
}
}

QueryPerformanceCounter(&EndingTime); //Stop Timer
//
// We now have the elapsed number of ticks, along with the
// number of ticks-per-second. We use these values
// to convert to the number of elapsed miliseconds.
// To guard against loss-of-precision, we convert
// to microseconds *before* dividing by ticks-per-second.
//
ElapsedTime=(EndingTime.QuadPart-StartingTime.QuadPart)*1000.0/Frequency.QuadPart;
printf("Total time: %f sec\n", ElapsedTime/1000);
printf("Single transformation point average time:
%f microsec\n\n\n", ElapsedTime*1000/K);
```

```
//Clear array
for (i = 0; i < J; i++) delete [] registr[i];
for (i = 0; i < J-1; i++) delete [] delay[i];
delete [] y_a;
fclose(data);
return 0;
}
```

10.pielikums. Hersta parametra vērtību novērtēšana 1. eksperimentā

10 p. 1. tabula. Hersta parametra novērtējums, pie $H = 0,5$.

Mērogu skaits	**Punktu skaits 1. mērogā**														
	2^{10}	2^{11}	2^{12}	2^{13}	2^{14}	2^{15}	2^{16}	2^{17}	2^{18}	2^{19}	2^{20}	2^{21}	2^{22}	2^{23}	2^{24}
2															
3	0,648	0,6527	0,6551	0,6566	0,6572	0,6575	0,6577	0,6578	0,6578	0,6578	0,6578	0,6578	0,6578	0,6578	0,6578
4	0,6278	0,6354	0,6397	0,6422	0,6434	0,644	0,6444	0,6445	0,6446	0,6446	0,6446	0,6447	0,6447	0,6447	0,6447
5	0,6064	0,6171	0,6239	0,6283	0,6307	0,632	0,6328	0,6331	0,6333	0,6334	0,6334	0,6335	0,6335	0,6335	0,6335
6	0,5848	0,5979	0,6072	0,6139	0,618	0,6205	0,6221	0,623	0,6235	0,6237	0,6238	0,6239	0,6239	0,6239	0,624
7	0,5637	0,5792	0,5902	0,5984	0,6044	0,6085	0,6113	0,6131	0,614	0,6146	0,6149	0,615	0,6151	0,6151	0,6151
8	0,5424	0,5617	0,5743	0,5838	0,591	0,5965	0,6004	0,6033	0,6052	0,6065	0,6071	0,6074	0,6077	0,6077	0,6077
9		0,5425	0,5589	0,5698	0,5781	0,5845	0,5895	0,5938	0,597	0,5995	0,6007	0,6018	0,6025	0,6025	0,6026
9		0,5425	0,5589	0,5698	0,5781	0,5845	0,5895	0,5938	0,597	0,5995	0,6007	0,6018	0,6025	0,6025	0,6026
10			0,5429	0,557	0,5672	0,5742	0,5802	0,5852	0,5892	0,5925	0,5947	0,5963	0,5974	0,5975	0,5975
11				0,5427	0,5549	0,5634	0,5702	0,5755	0,5804	0,5851	0,5887	0,5916	0,5946	0,5951	0,5953
12					0,5431	0,5533	0,561	0,567	0,5722	0,5771	0,5809	0,5838	0,5867	0,5872	0,5873
13						0,5403	0,5498	0,557	0,5633	0,5692	0,5743	0,5789	0,5834	0,5841	0,5842
14							0,5417	0,5498	0,557	0,5632	0,5679	0,5717	0,5755	0,5763	0,5767
15								0,5434	0,5511	0,5584	0,5629	0,567	0,5707	0,5713	0,5717
16									0,5464	0,5544	0,5594	0,5628	0,5659	0,5666	0,5669
17										0,5431	0,5502	0,5547	0,558	0,5585	0,5589
18											0,5457	0,5484	0,5528	0,5553	0,5563
19												0,5397	0,5428	0,5446	0,5458
20													0,5364	0,5385	0,5428
21														0,5452	0,5484
22															0,5491

10 p. 2. tabula. Hersta parametra novērtējums, pie $H = 0,6$.

Mērogu skaits	**Punktu skaits 1. mērogā**														
	2^{10}	2^{11}	2^{12}	2^{13}	2^{14}	2^{15}	2^{16}	2^{17}	2^{18}	2^{19}	2^{20}	2^{21}	2^{22}	2^{23}	2^{24}
2															
3	0,7011	0,7053	0,7074	0,7084	0,709	0,7092	0,7093	0,7094	0,7094	0,7094	0,7094	0,7094	0,7094	0,7094	0,7094
4	0,6827	0,6901	0,6938	0,6959	0,6969	0,6974	0,6976	0,6978	0,6978	0,6979	0,6979	0,6979	0,6979	0,6979	0,6979
5	0,6599	0,6715	0,6786	0,6824	0,6844	0,6855	0,686	0,6863	0,6864	0,6865	0,6865	0,6865	0,6865	0,6865	0,6865
6	0,6364	0,6527	0,6642	0,6715	0,6756	0,678	0,6792	0,6798	0,6801	0,6802	0,6803	0,6804	0,6804	0,6804	0,6804
7	0,6105	0,6304	0,6452	0,6555	0,6624	0,6669	0,6694	0,6709	0,6716	0,6718	0,6719	0,6721	0,6721	0,6721	0,6721
8	0,5836	0,6076	0,6252	0,6379	0,6473	0,6538	0,6584	0,6611	0,6624	0,6632	0,6636	0,6637	0,6638	0,664	0,664
9		0,5844	0,6054	0,6201	0,6316	0,6399	0,6464	0,6509	0,6538	0,6552	0,6561	0,6564	0,6565	0,6567	0,6567
10			0,5839	0,6018	0,6146	0,6242	0,632	0,6376	0,642	0,6448	0,6471	0,6488	0,6491	0,6502	0,6503
11				0,5842	0,5993	0,6103	0,6191	0,626	0,632	0,6366	0,6407	0,6428	0,6446	0,6476	0,6485
12					0,5825	0,5968	0,607	0,6155	0,6234	0,6302	0,638	0,6422	0,6481	0,6571	0,6579
13						0,5821	0,5947	0,6044	0,6141	0,623	0,634	0,6383	0,6477	0,6639	0,674
14							0,5802	0,5922	0,6035	0,6135	0,6271	0,6335	0,6457	0,6679	0,6826
15								0,5797	0,5924	0,604	0,6175	0,6238	0,6341	0,6551	0,6681
16									0,5811	0,5945	0,6117	0,619	0,6299	0,6548	0,6693
17										0,5865	0,6036	0,6117	0,6237	0,6494	0,6641
18											0,5985	0,6064	0,6206	0,6461	0,6648
19												0,5968	0,6102	0,6344	0,6501
20													0,6009	0,6244	0,642
21														0,6154	0,6289
22															0,6307

10 p. 3. tabula. Hersta parametra novērtējums, pie $H = 0,7$.

Mērogu skaits	**Punktu skaits 1. mērogā**														
	2^{10}	2^{11}	2^{12}	2^{13}	2^{14}	2^{15}	2^{16}	2^{17}	2^{18}	2^{19}	2^{20}	2^{21}	2^{22}	2^{23}	2^{24}
2															
3	0,7616	0,765	0,7666	0,7674	0,7678	0,768	0,7681	0,7682	0,7682	0,7682	0,7682	0,7682	0,7682	0,7682	0,7682
4	0,7482	0,7537	0,7566	0,758	0,7587	0,759	0,7593	0,7594	0,7594	0,7594	0,7594	0,7595	0,7595	0,7595	0,7595
5	0,7297	0,7396	0,745	0,7477	0,7491	0,7499	0,7502	0,7504	0,7505	0,7506	0,7506	0,7506	0,7506	0,7506	0,7506
6	0,7062	0,7224	0,732	0,7378	0,7407	0,7422	0,7429	0,7433	0,7436	0,7437	0,7437	0,7438	0,7438	0,7438	0,7438
7	0,6784	0,7012	0,7164	0,7267	0,7327	0,7359	0,7374	0,7384	0,7391	0,7393	0,7394	0,7394	0,7394	0,7395	0,7395
8	0,6468	0,676	0,6958	0,7108	0,7206	0,7267	0,7299	0,7321	0,7331	0,7334	0,7337	0,7338	0,7338	0,7339	0,7339
9		0,6479	0,6732	0,6921	0,7063	0,7164	0,7227	0,7267	0,7291	0,73	0,7305	0,731	0,7311	0,7312	0,7312
10			0,6486	0,6716	0,6888	0,7022	0,7114	0,7183	0,7231	0,7251	0,7263	0,727	0,7275	0,7276	0,7276
11				0,6489	0,6699	0,6864	0,6996	0,7103	0,7187	0,7225	0,7258	0,7278	0,7294	0,7302	0,7303
12					0,6489	0,6688	0,6843	0,6991	0,7127	0,7205	0,7274	0,7325	0,7345	0,736	0,7363
13						0,6487	0,6677	0,683	0,6984	0,7071	0,715	0,7212	0,7239	0,7266	0,7271
14							0,6523	0,6707	0,6878	0,6982	0,7075	0,7151	0,7208	0,7242	0,725
15								0,655	0,6756	0,6882	0,6993	0,7073	0,7139	0,7179	0,7185
16									0,6609	0,6737	0,6853	0,693	0,7002	0,7043	0,7054
17										0,6606	0,6744	0,6831	0,6894	0,693	0,6938
18											0,6552	0,6681	0,6726	0,6771	0,6781
19												0,6525	0,6609	0,6672	0,6682
20													0,6427	0,6501	0,6509
21														0,635	0,6347
22															0,629

10 p. 4. tabula. Hersta parametra novērtējums, pie $H = 0,8$.

Mērogu skaits	**Punktu skaits 1. mērogā**														
	2^{10}	2^{11}	2^{12}	2^{13}	2^{14}	2^{15}	2^{16}	2^{17}	2^{18}	2^{19}	2^{20}	2^{21}	2^{22}	2^{23}	2^{24}
2															
3	0,8168	0,82	0,8221	0,8237	0,8244	0,8254	0,8255	0,8255	0,8255	0,8255	0,8255	0,8255	0,8255	0,8255	0,8255
4	0,8082	0,8125	0,815	0,8168	0,8176	0,8186	0,8187	0,8188	0,8188	0,8188	0,8188	0,8188	0,8188	0,8188	0,8188
5	0,7962	0,8031	0,8069	0,8093	0,8105	0,8117	0,8119	0,812	0,812	0,812	0,8121	0,8121	0,8121	0,8121	0,8121
6	0,7803	0,7926	0,7996	0,8034	0,8053	0,8069	0,8072	0,8074	0,8075	0,8075	0,8076	0,8076	0,8076	0,8076	0,8076
7	0,7576	0,7783	0,7908	0,7974	0,8006	0,803	0,8038	0,8041	0,8044	0,8045	0,8045	0,8045	0,8045	0,8045	0,8045
8	0,7271	0,7571	0,7776	0,7892	0,7956	0,7999	0,8017	0,8025	0,8029	0,8031	0,8032	0,8032	0,8032	0,8032	0,8032
9		0,73	0,758	0,7755	0,7867	0,794	0,7974	0,7994	0,8004	0,8007	0,8009	0,801	0,8011	0,8011	0,8011
10			0,7326	0,7569	0,7738	0,7859	0,7919	0,7962	0,7982	0,7991	0,7997	0,8	0,8001	0,8001	0,8001
11				0,7345	0,7572	0,774	0,7837	0,7912	0,7964	0,7989	0,8001	0,8008	0,8009	0,8009	0,8009
12					0,7351	0,7572	0,7713	0,7824	0,7904	0,7954	0,798	0,8004	0,8006	0,8007	0,8007
13						0,7371	0,7558	0,7697	0,7823	0,7902	0,7961	0,8016	0,8021	0,8024	0,8025
14							0,736	0,7527	0,769	0,7812	0,7908	0,798	0,7995	0,8001	0,8003
15								0,7344	0,754	0,7677	0,7821	0,7914	0,7974	0,8009	0,8011
16									0,7406	0,7576	0,7735	0,7863	0,7932	0,7995	0,8002
17										0,7435	0,7628	0,7799	0,7903	0,7988	0,8018
18											0,7501	0,7708	0,7826	0,7953	0,8013
19												0,7628	0,7773	0,7937	0,8039
20													0,7616	0,7767	0,7839
21														0,7622	0,7696
22															0,7662

10 p. 5. tabula. Hersta parametra novērtējums, pie $H = 0,9$.

Mērogu skaits	**Punktu skaits 1. mērogā**														
	2^{10}	2^{11}	2^{12}	2^{13}	2^{14}	2^{15}	2^{16}	2^{17}	2^{18}	2^{19}	2^{20}	2^{21}	2^{22}	2^{23}	2^{24}
2															
3	0,8466	0,8564	0,864	0,8706	0,874	0,8758	0,8759	0,876	0,876	0,876	0,876	0,876	0,876	0,876	0,876
4	0,8429	0,8524	0,8598	0,8663	0,8694	0,8713	0,8714	0,8715	0,8715	0,8715	0,8715	0,8715	0,8715	0,8715	0,8715
5	0,8384	0,8487	0,8565	0,863	0,8662	0,8681	0,8681	0,8682	0,8682	0,8683	0,8683	0,8683	0,8683	0,8683	0,8683
6	0,8326	0,8454	0,8542	0,8613	0,8645	0,8665	0,8667	0,8668	0,8668	0,8669	0,8669	0,8669	0,8669	0,8669	0,8669
7	0,8217	0,84	0,8514	0,8595	0,8634	0,8656	0,866	0,8662	0,8662	0,8663	0,8663	0,8663	0,8663	0,8663	0,8663
8	0,8015	0,8298	0,8464	0,8575	0,8624	0,865	0,8656	0,8659	0,866	0,866	0,8661	0,8661	0,8661	0,8661	0,8661
9		0,8118	0,8373	0,8535	0,8608	0,8646	0,8656	0,8661	0,8664	0,8665	0,8665	0,8666	0,8666	0,8666	0,8666
10			0,8203	0,8447	0,8568	0,863	0,8651	0,866	0,8665	0,8668	0,8669	0,8669	0,867	0,867	0,8671
11				0,8293	0,848	0,8584	0,8635	0,866	0,8671	0,8677	0,868	0,8681	0,8682	0,8682	0,8683
12					0,8324	0,85	0,8589	0,8641	0,8669	0,8679	0,8683	0,8684	0,8687	0,8687	0,8688
13						0,8362	0,85	0,8595	0,8662	0,8683	0,869	0,8692	0,8697	0,8697	0,87
14							0,837	0,8524	0,8634	0,8689	0,8703	0,8707	0,8714	0,8718	0,8723
15								0,8368	0,8546	0,8636	0,8682	0,8707	0,8721	0,8724	0,8728
16									0,8393	0,8521	0,861	0,8654	0,8668	0,8674	0,8681
17										0,8348	0,848	0,8544	0,8564	0,8575	0,8588
18											0,8303	0,8427	0,8468	0,8479	0,8491
19												0,8298	0,8334	0,8351	0,8389
20													0,8133	0,8173	0,8239
21														0,7956	0,8029
22															0,7867

10 p. 6. tabula. Hersta parametra novērtējums, pie $H = 0,99$.

Mērogu skaits	**Punktu skaits 1. mērogā**														
	2^{10}	2^{11}	2^{12}	2^{13}	2^{14}	2^{15}	2^{16}	2^{17}	2^{18}	2^{19}	2^{20}	2^{21}	2^{22}	2^{23}	2^{24}
2															
3	0,5367	0,559	0,5798	0,5953	0,6137	0,6288	0,6523	0,685	0,7202	0,8057	0,8072	0,9223	0,9223	0,9225	0,9226
4	0,5368	0,558	0,5783	0,5929	0,6111	0,6265	0,6495	0,6819	0,7182	0,8041	0,8041	0,9191	0,9188	0,919	0,919
5	0,5356	0,5562	0,5764	0,5909	0,6086	0,6237	0,6467	0,6786	0,7147	0,7999	0,8006	0,9147	0,9149	0,9153	0,9153
6	0,5336	0,5547	0,5749	0,5896	0,6074	0,6222	0,6452	0,6774	0,7131	0,7991	0,7986	0,913	0,9125	0,9128	0,9128
7	0,5307	0,5536	0,5745	0,5893	0,6073	0,6219	0,6454	0,6777	0,7132	0,7994	0,7985	0,9127	0,9124	0,9124	0,9124
8	0,5245	0,5516	0,5742	0,5893	0,6075	0,6221	0,6456	0,678	0,714	0,8005	0,7991	0,9136	0,9133	0,913	0,913
9		0,5457	0,5718	0,5887	0,6073	0,622	0,6456	0,6782	0,7137	0,7998	0,7994	0,9135	0,9134	0,913	0,913
10			0,5677	0,5881	0,6078	0,6233	0,6468	0,6798	0,7154	0,8018	0,8013	0,9162	0,9154	0,915	0,915
11				0,5843	0,6071	0,6241	0,6483	0,6814	0,7169	0,8038	0,8029	0,9183	0,9173	0,9168	0,9168
12					0,6027	0,6228	0,6479	0,6807	0,7168	0,8033	0,8029	0,9182	0,917	0,9164	0,9163
13						0,6193	0,648	0,6818	0,7193	0,8068	0,8041	0,92	0,919	0,918	0,9179
14							0,6443	0,6804	0,7183	0,8078	0,8048	0,92	0,9188	0,9181	0,9182
15								0,6764	0,7157	0,8085	0,8029	0,9172	0,9173	0,9169	0,9167
16									0,7144	0,8109	0,8049	0,9219	0,9206	0,9208	0,9202
17										0,8063	0,8047	0,9231	0,9188	0,9194	0,9186
18											0,8055	0,9286	0,9272	0,9273	0,9252
19												0,9357	0,9333	0,9341	0,9303
20													0,9391	0,9419	0,9394
21														0,9525	0,9456
22															0,9502

11.pielikums. Hersta parametra kļūdas novērtēšana 1. eksperimentā

11 p. 7. tabula. Hersta parametra vidējā kļūda, pie $H = 0,5$.

Mērogu skaits	**Punktu skaits 1. mērogā**														
	2^{10}	2^{11}	2^{12}	2^{13}	2^{14}	2^{15}	2^{16}	2^{17}	2^{18}	2^{19}	2^{20}	2^{21}	2^{22}	2^{23}	2^{24}
2	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5
3	0,1497	0,1528	0,1552	0,1566	0,1572	0,1575	0,1577	0,1578	0,1578	0,1578	0,1578	0,1578	0,1578	0,1578	0,1578
4	0,1295	0,1355	0,1397	0,1422	0,1434	0,144	0,1444	0,1445	0,1446	0,1446	0,1446	0,1447	0,1447	0,1447	0,1447
5	0,1094	0,1175	0,1239	0,1283	0,1307	0,132	0,1328	0,1331	0,1333	0,1334	0,1334	0,1335	0,1335	0,1335	0,1335
6	0,0908	0,0988	0,1072	0,1139	0,118	0,1205	0,1221	0,123	0,1235	0,1237	0,1238	0,1239	0,1239	0,1239	0,124
7	0,0762	0,0819	0,0906	0,0984	0,1044	0,1085	0,1113	0,1131	0,114	0,1146	0,1149	0,115	0,1151	0,1151	0,1151
8	0,0676	0,0687	0,0755	0,0838	0,091	0,0965	0,1004	0,1033	0,1052	0,1065	0,1071	0,1074	0,1077	0,1077	0,1077
9		0,0598	0,063	0,0702	0,0782	0,0845	0,0895	0,0938	0,097	0,0995	0,1007	0,1018	0,1025	0,1025	0,1026
10			0,0543	0,0591	0,0672	0,0742	0,0802	0,0852	0,0892	0,0925	0,0947	0,0963	0,0974	0,0975	0,0975
11				0,0503	0,0559	0,0635	0,0702	0,0755	0,0804	0,0851	0,0887	0,0916	0,0946	0,0951	0,0953
12					0,0478	0,0539	0,061	0,067	0,0722	0,0771	0,0809	0,0838	0,0867	0,0872	0,0873
13						0,044	0,0505	0,057	0,0633	0,0692	0,0743	0,0789	0,0834	0,0841	0,0842
14							0,0444	0,0499	0,057	0,0632	0,0679	0,0717	0,0755	0,0763	0,0767
15								0,0448	0,0513	0,0584	0,0629	0,067	0,0707	0,0713	0,0717
16									0,0473	0,0544	0,0594	0,0628	0,0659	0,0666	0,0669
17										0,0439	0,0502	0,0547	0,058	0,0585	0,0589
18											0,0457	0,0484	0,0528	0,0553	0,0563
19												0,0397	0,0428	0,0446	0,0458
20													0,0364	0,0385	0,0428
21														0,0452	0,0484
22															0,0491

11 p. 8. tabula. Hersta parametra vidējā kļūda, pie $H = 0,6$.

Mērogu skaits	**Punktu skaits 1. mērogā**														
	2^{10}	2^{11}	2^{12}	2^{13}	2^{14}	2^{15}	2^{16}	2^{17}	2^{18}	2^{19}	2^{20}	2^{21}	2^{22}	2^{23}	2^{24}
2	0,6	0,6	0,6	0,6	0,6	0,6	0,6	0,6	0,6	0,6	0,6	0,6	0,6	0,6	0,6
3	0,1088	0,1067	0,1074	0,1084	0,109	0,1092	0,1093	0,1094	0,1094	0,1094	0,1094	0,1094	0,1094	0,1094	0,1094
4	0,0955	0,0934	0,0943	0,0959	0,0969	0,0974	0,0976	0,0978	0,0978	0,0979	0,0979	0,0979	0,0979	0,0979	0,0979
5	0,0836	0,0801	0,0807	0,0827	0,0844	0,0855	0,086	0,0863	0,0864	0,0865	0,0865	0,0865	0,0865	0,0865	0,0865
6	0,0774	0,0718	0,0709	0,0729	0,0758	0,078	0,0792	0,0798	0,0801	0,0802	0,0803	0,0804	0,0804	0,0804	0,0804
7	0,0742	0,0653	0,0616	0,0613	0,0636	0,0669	0,0694	0,0709	0,0716	0,0718	0,0719	0,0721	0,0721	0,0721	0,0721
8	0,0779	0,0636	0,0559	0,0526	0,0523	0,0548	0,0585	0,0611	0,0624	0,0632	0,0636	0,0637	0,0638	0,064	0,064
9		0,0674	0,0554	0,0489	0,0455	0,0451	0,0474	0,0515	0,0538	0,0552	0,0561	0,0564	0,0565	0,0567	0,0567
10			0,0598	0,048	0,0405	0,0375	0,0372	0,0394	0,0421	0,0448	0,0471	0,0488	0,0491	0,0502	0,0503
11				0,0523	0,0417	0,0369	0,0329	0,0323	0,0339	0,0366	0,0407	0,0428	0,0446	0,0476	0,0485
12					0,0477	0,0385	0,0323	0,03	0,0295	0,0314	0,038	0,0422	0,0481	0,0571	0,0579
13						0,0442	0,0359	0,0312	0,0289	0,0306	0,0387	0,0403	0,0477	0,0639	0,074
14							0,0427	0,0366	0,0324	0,036	0,0437	0,0435	0,0516	0,0679	0,0826
15								0,0441	0,0389	0,0398	0,0449	0,0428	0,0488	0,0551	0,0681
16									0,0481	0,0488	0,0509	0,0472	0,0538	0,0548	0,0693
17										0,0524	0,0523	0,047	0,0547	0,0494	0,0641
18											0,0505	0,044	0,0531	0,0472	0,0648
19												0,0474	0,051	0,0412	0,0501
20													0,0524	0,0417	0,042
21														0,0398	0,0289
22															0,0307

11 p. 9. tabula. Hersta parametra vidējā kļūda, pie $H = 0,7$.

Mērogu skaits	**Punktu skaits 1. mērogā**														
	2^{10}	2^{11}	2^{12}	2^{13}	2^{14}	2^{15}	2^{16}	2^{17}	2^{18}	2^{19}	2^{20}	2^{21}	2^{22}	2^{23}	2^{24}
2	0,7	0,7	0,7	0,7	0,7	0,7	0,7	0,7	0,7	0,7	0,7	0,7	0,7	0,7	0,7
3	0,0788	0,0706	0,0678	0,0675	0,0678	0,068	0,0681	0,0682	0,0682	0,0682	0,0682	0,0682	0,0682	0,0682	0,0682
4	0,0733	0,0633	0,0591	0,0584	0,0588	0,059	0,0593	0,0594	0,0594	0,0594	0,0594	0,0595	0,0595	0,0595	0,0595
5	0,0753	0,0607	0,0527	0,0495	0,0493	0,0499	0,0502	0,0504	0,0505	0,0506	0,0506	0,0506	0,0506	0,0506	0,0506
6	0,0828	0,0646	0,0524	0,0451	0,0428	0,0426	0,043	0,0433	0,0436	0,0437	0,0437	0,0438	0,0438	0,0438	0,0438
7	0,0938	0,0734	0,0586	0,0474	0,041	0,0388	0,0382	0,0386	0,0391	0,0393	0,0394	0,0394	0,0394	0,0395	0,0395
8	0,1082	0,0844	0,0672	0,0533	0,0427	0,0368	0,0344	0,0328	0,0331	0,0334	0,0337	0,0338	0,0338	0,0339	0,0339
9		0,0991	0,0782	0,0621	0,0493	0,0403	0,0351	0,0314	0,0298	0,03	0,0305	0,031	0,0311	0,0312	0,0312
10			0,0911	0,0725	0,0577	0,046	0,039	0,033	0,0272	0,0258	0,0263	0,027	0,0275	0,0276	0,0276
11				0,0855	0,0681	0,0554	0,0464	0,0394	0,0306	0,0276	0,0278	0,0278	0,0294	0,0302	0,0303
12					0,0814	0,0659	0,0561	0,0469	0,0391	0,0365	0,0352	0,0334	0,0345	0,036	0,0363
13						0,0792	0,0653	0,0536	0,0435	0,0413	0,0343	0,0276	0,0239	0,0266	0,0271
14							0,0762	0,0616	0,0499	0,0457	0,036	0,0274	0,0217	0,0242	0,025
15								0,073	0,0571	0,0495	0,0367	0,0261	0,0203	0,0179	0,0185
16									0,0617	0,0526	0,0383	0,0258	0,0184	0,0096	0,0054
17										0,0582	0,0408	0,0302	0,0216	0,0077	0,0062
18											0,0507	0,039	0,0316	0,0229	0,0219
19												0,0525	0,0391	0,0328	0,0318
20													0,0573	0,0499	0,0491
21														0,065	0,0653
22															0,071

11 p. 10. tabula. Hersta parametra vidējā kļūda, pie $H = 0,8$.

Mērogu skaits	**Punktu skaits 1. mērogā**														
	2^{10}	2^{11}	2^{12}	2^{13}	2^{14}	2^{15}	2^{16}	2^{17}	2^{18}	2^{19}	2^{20}	2^{21}	2^{22}	2^{23}	2^{24}
2	0,8	0,8	0,8	0,8	0,8	0,8	0,8	0,8	0,8	0,8	0,8	0,8	0,8	0,8	0,8
3	0,058	0,0438	0,0347	0,0292	0,0268	0,0254	0,0255	0,0255	0,0255	0,0255	0,0255	0,0255	0,0255	0,0255	0,0255
4	0,056	0,0409	0,0312	0,0247	0,021	0,019	0,0187	0,0188	0,0188	0,0188	0,0188	0,0188	0,0188	0,0188	0,0188
5	0,0619	0,0441	0,0318	0,0234	0,0183	0,0142	0,0127	0,0121	0,012	0,012	0,0121	0,0121	0,0121	0,0121	0,0121
6	0,076	0,0531	0,0365	0,0262	0,0191	0,0138	0,0111	0,0092	0,0083	0,0079	0,0076	0,0076	0,0076	0,0076	0,0076
7	0,0971	0,0678	0,0456	0,0323	0,0228	0,0161	0,0118	0,0095	0,0074	0,006	0,0048	0,0045	0,0045	0,0045	0,0045
8	0,122	0,0888	0,0616	0,0435	0,0303	0,0204	0,0144	0,0103	0,0077	0,0061	0,0047	0,0032	0,0032	0,0032	0,0032
9		0,1114	0,0807	0,058	0,04	0,0276	0,0193	0,0134	0,009	0,0066	0,0049	0,0027	0,0017	0,0011	0,0011
10			0,1031	0,0772	0,0553	0,0388	0,028	0,0192	0,0136	0,0093	0,0062	0,0023	0,0015	0,0016	0,0001
11				0,0968	0,0727	0,0537	0,0394	0,0276	0,0198	0,0123	0,0089	0,002	0,0019	0,001	0,0009
12					0,0918	0,0697	0,0536	0,0396	0,0307	0,02	0,014	0,0025	0,0013	0,0007	0,0007
13						0,0873	0,0693	0,0555	0,0444	0,0297	0,021	0,0049	0,004	0,0024	0,0025
14							0,0874	0,0724	0,0576	0,0387	0,026	0,0089	0,005	0,0026	0,0003
15								0,0865	0,0685	0,0507	0,0365	0,0205	0,0143	0,0009	0,0011
16									0,081	0,0603	0,0427	0,0276	0,0211	0,005	0,0002
17										0,0745	0,053	0,0342	0,0255	0,0124	0,0018
18											0,0638	0,0459	0,0369	0,0196	0,0013
19												0,0567	0,0467	0,0276	0,0039
20													0,0518	0,0254	0,0161
21														0,0378	0,0304
22															0,0338

11 p. 11. tabula. Hersta parametra vidējā kļūda, pie $H = 0,9$.

Mērogu skaits	**Punktu skaits 1. mērogā**														
	2^{10}	2^{11}	2^{12}	2^{13}	2^{14}	2^{15}	2^{16}	2^{17}	2^{18}	2^{19}	2^{20}	2^{21}	2^{22}	2^{23}	2^{24}
2	0,9	0,9	0,9	0,9	0,9	0,9	0,9	0,9	0,9	0,9	0,9	0,9	0,9	0,9	0,9
3	0,0783	0,0581	0,0434	0,0322	0,0266	0,0243	0,0241	0,024	0,024	0,024	0,024	0,024	0,024	0,024	0,024
4	0,0763	0,0576	0,0448	0,0351	0,0308	0,0287	0,0286	0,0285	0,0285	0,0285	0,0285	0,0285	0,0285	0,0285	0,0285
5	0,0798	0,0608	0,0476	0,0381	0,034	0,0319	0,0319	0,0318	0,0318	0,0317	0,0317	0,0317	0,0317	0,0317	0,0317
6	0,0876	0,0656	0,0506	0,0399	0,0356	0,0335	0,0333	0,0332	0,0332	0,0331	0,0331	0,0331	0,0331	0,0331	0,0331
7	0,1022	0,0743	0,0558	0,0428	0,0371	0,0345	0,034	0,0338	0,0338	0,0337	0,0337	0,0337	0,0337	0,0337	0,0337
8	0,1256	0,0885	0,0637	0,0466	0,0389	0,0352	0,0344	0,0341	0,034	0,034	0,0339	0,0339	0,0339	0,0339	0,0339
9		0,1113	0,0771	0,0537	0,0422	0,0358	0,0344	0,0339	0,0336	0,0335	0,0335	0,0334	0,0334	0,0334	0,0334
10			0,0989	0,0668	0,0488	0,0384	0,035	0,034	0,0335	0,0332	0,0331	0,0331	0,033	0,033	0,0329
11				0,0871	0,0611	0,0455	0,0374	0,0343	0,0331	0,0323	0,032	0,0319	0,0318	0,0318	0,0317
12					0,0805	0,0572	0,0431	0,0361	0,0333	0,0321	0,0317	0,0316	0,0313	0,0313	0,0312
13						0,0742	0,0552	0,0424	0,034	0,0317	0,031	0,0308	0,0303	0,0303	0,03
14							0,0718	0,0525	0,0373	0,0311	0,0297	0,0293	0,0286	0,0282	0,0277
15								0,0707	0,0474	0,0372	0,0318	0,0293	0,0279	0,0276	0,0272
16									0,0634	0,0483	0,039	0,0346	0,0332	0,0326	0,0319
17										0,0656	0,052	0,0456	0,0436	0,0425	0,0412
18											0,0697	0,0573	0,0532	0,0521	0,0509
19												0,0702	0,0666	0,0649	0,0611
20													0,0867	0,0827	0,0761
21														0,1044	0,0971
22															0,1133

11 p. 12. tabula. Hersta parametra vidējā kļūda, pie $H = 0,99$.

Mērogu skaits	**Punktu skaits 1. mērogā**														
	2^{10}	2^{11}	2^{12}	2^{13}	2^{14}	2^{15}	2^{16}	2^{17}	2^{18}	2^{19}	2^{20}	2^{21}	2^{22}	2^{23}	2^{24}
2	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99
3	0,4547	0,4317	0,4105	0,3947	0,3764	0,3612	0,3377	0,305	0,2698	0,1843	0,1828	0,0677	0,0677	0,0675	0,0674
4	0,4542	0,4325	0,4118	0,3971	0,3789	0,3635	0,3405	0,3081	0,2718	0,1859	0,1859	0,0709	0,0712	0,071	0,071
5	0,4553	0,4341	0,4137	0,3991	0,3814	0,3663	0,3433	0,3114	0,2753	0,1901	0,1894	0,0753	0,0751	0,0747	0,0747
6	0,4574	0,4357	0,4152	0,4005	0,3826	0,3678	0,3448	0,3126	0,2769	0,1909	0,1914	0,077	0,0775	0,0772	0,0772
7	0,4604	0,437	0,4157	0,4008	0,3828	0,3681	0,3446	0,3123	0,2768	0,1906	0,1915	0,0773	0,0776	0,0776	0,0776
8	0,467	0,4392	0,4162	0,4008	0,3826	0,3679	0,3444	0,312	0,276	0,1895	0,1909	0,0764	0,0767	0,077	0,077
9		0,4455	0,4188	0,4015	0,3828	0,368	0,3444	0,3118	0,2763	0,1902	0,1906	0,0765	0,0766	0,077	0,077
10			0,4232	0,4023	0,3823	0,3667	0,3432	0,3102	0,2746	0,1882	0,1887	0,0738	0,0746	0,075	0,075
11				0,4064	0,3833	0,3661	0,3419	0,3087	0,2731	0,1862	0,1871	0,0717	0,0727	0,0732	0,0732
12					0,3878	0,3675	0,3423	0,3094	0,2732	0,1867	0,1871	0,0718	0,073	0,0736	0,0737
13						0,371	0,3422	0,3083	0,2707	0,1832	0,1859	0,07	0,071	0,072	0,0721
14							0,3461	0,3098	0,2717	0,1822	0,1852	0,07	0,0712	0,0719	0,0718
15								0,3139	0,2745	0,1819	0,1871	0,0728	0,0727	0,0731	0,0733
16									0,2759	0,1797	0,1851	0,0681	0,0694	0,0692	0,0698
17										0,1851	0,1853	0,0669	0,0712	0,0706	0,0714
18											0,1845	0,0614	0,0628	0,0627	0,0648
19												0,0543	0,0567	0,0559	0,0597
20													0,0509	0,0481	0,0506
21														0,0375	0,0444
22															0,0398

12.pielikums. Hersta parametra vērtību novērtēšana 2. eksperimentā

12 p. 13. tabula. Hersta parametra novērtējums, pie $H = 0,8$.

Mērogu skaits	**Punktu skaits 1. mērogā**														
	2^{10}	2^{11}	2^{12}	2^{13}	2^{14}	2^{15}	2^{16}	2^{17}	2^{18}	2^{19}	2^{20}	2^{21}	2^{22}	2^{23}	2^{24}
2															
3	0,817	0,8204	0,8219	0,8231	0,8246	0,8255	0,8256	0,8256	0,8256	0,8257	0,8257	0,8257	0,8257	0,8257	0,8257
4	0,8092	0,8135	0,8156	0,817	0,8186	0,8197	0,8197	0,8198	0,8198	0,8198	0,8198	0,8198	0,8198	0,8198	0,8198
5	0,7971	0,8041	0,8076	0,8097	0,8116	0,8128	0,813	0,8131	0,8131	0,8131	0,8132	0,8132	0,8132	0,8132	0,8132
6	0,7807	0,7929	0,7993	0,8028	0,8056	0,8071	0,8076	0,8077	0,8078	0,8079	0,8079	0,8079	0,8079	0,8079	0,8079
7	0,7573	0,7781	0,79	0,7967	0,8011	0,8033	0,8042	0,8046	0,8049	0,805	0,805	0,8051	0,8051	0,8051	0,8051
8	0,7266	0,7565	0,7756	0,7873	0,7949	0,799	0,8006	0,8015	0,8021	0,8022	0,8023	0,8024	0,8024	0,8024	0,8024
9		0,7285	0,7551	0,773	0,7852	0,7928	0,7962	0,7983	0,7993	0,7998	0,8	0,8	0,8001	0,8001	0,8001
10			0,7297	0,7541	0,7721	0,7848	0,7916	0,7958	0,7981	0,7989	0,7993	0,7995	0,7997	0,7998	0,7998
11				0,7301	0,7543	0,7719	0,7826	0,7907	0,7949	0,7967	0,7975	0,7978	0,7986	0,7987	0,7987
12					0,7322	0,7542	0,7689	0,7806	0,7891	0,7931	0,7951	0,7962	0,7976	0,7979	0,7979
13						0,7341	0,7535	0,7692	0,7816	0,7893	0,7935	0,7961	0,8	0,8003	0,8007
14							0,7342	0,7532	0,7684	0,7789	0,7866	0,7913	0,7982	0,7985	0,7989
15								0,7354	0,7541	0,7675	0,7774	0,7867	0,7974	0,7977	0,8012
16									0,7401	0,7555	0,7674	0,7821	0,7992	0,8045	0,8104
17										0,7431	0,7587	0,7773	0,7959	0,8007	0,8053
18											0,7445	0,7673	0,7882	0,7964	0,8015
19												0,7551	0,7804	0,7868	0,7908
20													0,7638	0,7714	0,7751
21														0,768	0,7724
22															0,7354

12 p. 14. tabula. Hersta parametra novērtējums, pie $H = 0,85$.

Mērogu skaits	**Punktu skaits 1. mērogā**														
	2^{10}	2^{11}	2^{12}	2^{13}	2^{14}	2^{15}	2^{16}	2^{17}	2^{18}	2^{19}	2^{20}	2^{21}	2^{22}	2^{23}	2^{24}
2															
3	0,8204	0,825	0,8276	0,8298	0,8313	0,8339	0,8389	0,8456	0,8522	0,8522	0,8522	0,8522	0,8522	0,8522	0,8522
4	0,8146	0,8196	0,8222	0,8245	0,8261	0,8287	0,8337	0,8403	0,847	0,8469	0,8469	0,8469	0,8469	0,8469	0,847
5	0,806	0,8128	0,8161	0,8187	0,8205	0,8231	0,8282	0,8348	0,8413	0,8414	0,8413	0,8414	0,8414	0,8414	0,8414
6	0,7954	0,8061	0,8114	0,8148	0,817	0,82	0,8251	0,8318	0,8383	0,8383	0,8384	0,8384	0,8384	0,8384	0,8384
7	0,7791	0,7975	0,8066	0,8122	0,8154	0,819	0,8243	0,8309	0,8376	0,8377	0,8377	0,8378	0,8378	0,8378	0,8378
8	0,7528	0,782	0,7973	0,807	0,8125	0,8169	0,8227	0,8294	0,8364	0,8365	0,8365	0,8367	0,8367	0,8367	0,8367
9		0,7587	0,7834	0,7993	0,8089	0,8154	0,8224	0,8296	0,8366	0,837	0,8371	0,8372	0,8372	0,8372	0,8372
10			0,7623	0,7855	0,8012	0,8112	0,8203	0,8284	0,8358	0,8364	0,8369	0,837	0,8371	0,8372	0,8371
11				0,7649	0,7875	0,8028	0,8157	0,8257	0,8343	0,8353	0,8359	0,8359	0,836	0,8361	0,8361
12					0,7693	0,7903	0,8083	0,822	0,8331	0,8346	0,8358	0,8363	0,8364	0,8366	0,8366
13						0,7706	0,7953	0,8149	0,8303	0,8329	0,8349	0,8356	0,8362	0,8369	0,8369
14							0,7775	0,8018	0,8188	0,8229	0,8275	0,8295	0,8307	0,8334	0,8334
15								0,7857	0,8074	0,8138	0,8222	0,8257	0,8283	0,8353	0,8358
16									0,7913	0,8003	0,8129	0,8173	0,8194	0,8292	0,8311
17										0,7877	0,8073	0,816	0,8248	0,8452	0,8467
18											0,7989	0,8138	0,8223	0,8511	0,8601
19												0,8105	0,8244	0,8667	0,8821
20													0,8091	0,8558	0,8736
21											.			0,8518	0,8682
22															0,8726

12 p. 15. tabula. Hersta parametra novērtējums, pie $H = 0.9$.

Mērogu skaits	**Punktu skaits 1. mērogā**														
	2^{10}	2^{11}	2^{12}	2^{13}	2^{14}	2^{15}	2^{16}	2^{17}	2^{18}	2^{19}	2^{20}	2^{21}	2^{22}	2^{23}	2^{24}
2															
3	0.8273	0.8376	0.846	0.8539	0.86	0.8704	0.8721	0.8757	0.8756	0.8756	0.8756	0.8756	0.8756	0.8756	0.8756
4	0.8249	0.8347	0.8429	0.8506	0.8567	0.867	0.8686	0.8721	0.872	0.872	0.872	0.872	0.8721	0.8721	0.8721
5	0.8202	0.8308	0.8394	0.8469	0.8532	0.8634	0.8651	0.8685	0.8685	0.8685	0.8686	0.8686	0.8686	0.8686	0.8686
6	0.8137	0.8269	0.8366	0.8445	0.8509	0.8613	0.863	0.8665	0.8665	0.8665	0.8665	0.8665	0.8665	0.8665	0.8665
7	0.8034	0.8223	0.8345	0.8433	0.85	0.8609	0.8626	0.8661	0.866	0.8661	0.8661	0.8661	0.8661	0.8661	0.8661
8	0.7835	0.8121	0.8301	0.8417	0.8494	0.8606	0.8626	0.8663	0.8663	0.8663	0.8663	0.8663	0.8663	0.8664	0.8664
9		0.7956	0.8217	0.8382	0.8485	0.8607	0.8629	0.867	0.8671	0.8672	0.8672	0.8672	0.8672	0.8672	0.8672
10			0.8052	0.83	0.8454	0.8599	0.8633	0.8678	0.8683	0.8685	0.8686	0.8686	0.8687	0.8687	0.8687
11				0.815	0.8376	0.8567	0.8622	0.8679	0.8688	0.8693	0.8695	0.8696	0.8696	0.8697	0.8697
12					0.8229	0.8488	0.8584	0.8661	0.8679	0.869	0.8694	0.8694	0.8696	0.8697	0.8698
13						0.8364	0.8522	0.8631	0.8673	0.8695	0.871	0.8713	0.8716	0.8717	0.8718
14							0.84	0.8568	0.8642	0.871	0.873	0.8733	0.8739	0.874	0.8743
15								0.8481	0.8603	0.8728	0.8756	0.8762	0.8776	0.8777	0.8779
16									0.8506	0.8717	0.8772	0.8801	0.8851	0.8854	0.8857
17										0.8658	0.8759	0.8811	0.8906	0.8912	0.8916
18											0.8649	0.8745	0.8889	0.889	
19												0.8583	0.8698	0.8774	0.8789
20													0.8648	0.8707	0.8718
21														0.855	0.8563
22															0.8217

12 p. 16. tabula. Hersta parametra novērtējums, pie $H = 0,95$.

Mērogu skaits	**Punktu skaits 1. mērogā**														
	2^{10}	2^{11}	2^{12}	2^{13}	2^{14}	2^{15}	2^{16}	2^{17}	2^{18}	2^{19}	2^{20}	2^{21}	2^{22}	2^{23}	2^{24}
2															
3	0,7603	0,7786	0,7935	0,8096	0,8242	0,8367	0,8525	0,8632	0,87	0,87	0,8981	0,8981	0,8981	0,8981	0,8981
4	0,7595	0,7766	0,7909	0,807	0,8215	0,8337	0,8493	0,8598	0,8668	0,8669	0,8948	0,8948	0,8948	0,8949	0,8949
5	0,7565	0,7736	0,7879	0,8039	0,8184	0,8305	0,846	0,8564	0,8634	0,8634	0,8912	0,8912	0,8913	0,8913	0,8913
6	0,7529	0,7714	0,786	0,802	0,8165	0,8289	0,8444	0,8548	0,8619	0,862	0,8898	0,8898	0,8898	0,8898	0,8898
7	0,7475	0,769	0,7848	0,8011	0,8157	0,8283	0,8437	0,8543	0,8613	0,8615	0,8893	0,8893	0,8893	0,8892	0,8893
8	0,7361	0,7648	0,7837	0,801	0,8164	0,829	0,8446	0,855	0,8621	0,8622	0,89	0,8899	0,8899	0,8899	0,89
9		0,7544	0,7801	0,8006	0,8169	0,8298	0,8458	0,8562	0,8631	0,8632	0,8912	0,891	0,891	0,891	0,891
10			0,7707	0,7965	0,8158	0,8294	0,846	0,8564	0,8633	0,8635	0,8916	0,8914	0,8913	0,8913	0,8913
11				0,7891	0,8131	0,8302	0,8485	0,8593	0,8662	0,8666	0,8946	0,8945	0,8944	0,8944	0,8944
12					0,8061	0,828	0,8489	0,8608	0,868	0,8684	0,8968	0,8966	0,8966	0,8967	0,8967
13						0,8204	0,8464	0,861	0,8695	0,8705	0,8991	0,8988	0,8988	0,8988	0,899
14							0,8395	0,8572	0,8673	0,8697	0,9001	0,8999	0,8998	0,9002	0,9003
15								0,848	0,8623	0,8676	0,8989	0,8986	0,8985	0,8989	0,899
16									0,8568	0,8672	0,9001	0,9005	0,9003	0,9007	0,9007
17										0,8641	0,9012	0,9042	0,9062	0,9065	0,9067
18											0,8996	0,9035	0,9079	0,909	0,9099
19												0,9026	0,9076	0,9126	0,9126
20													0,9027	0,9076	0,9076
21														0,9076	0,9174
22															0,9146

12 p. 17. tabula. Hersta parametra novērtējums, pie $H = 0,99$.

Mērogu skaits	**Punktu skaits 1. mērogā**														
	2^{10}	2^{11}	2^{12}	2^{13}	2^{14}	2^{15}	2^{16}	2^{17}	2^{18}	2^{19}	2^{20}	2^{21}	2^{22}	2^{23}	2^{24}
2															
3	0,7468	0,7771	0,8027	0,8258	0,8509	0,8761	0,8978	0,9087	0,9231	0,9231	0,9232	0,9232	0,9232	0,9232	0,9232
4	0,7467	0,7754	0,8002	0,8227	0,8475	0,8727	0,8943	0,9053	0,9194	0,9193	0,9193	0,9194	0,9194	0,9194	0,9194
5	0,7449	0,7731	0,7973	0,8199	0,8446	0,8696	0,8911	0,9019	0,9159	0,9158	0,9159	0,9159	0,9159	0,9159	0,9159
6	0,7423	0,7712	0,7957	0,8182	0,8428	0,8678	0,8892	0,8999	0,9139	0,9137	0,9138	0,9138	0,9138	0,9138	0,9138
7	0,7382	0,7696	0,7946	0,8174	0,8419	0,8671	0,8884	0,8991	0,9131	0,913	0,913	0,913	0,913	0,913	0,913
8	0,7284	0,765	0,7925	0,8158	0,8403	0,8657	0,8871	0,8974	0,9119	0,9118	0,9118	0,9118	0,9118	0,9118	0,9118
9		0,7578	0,7899	0,8157	0,8408	0,8663	0,8879	0,8981	0,9126	0,9126	0,9126	0,9126	0,9126	0,9126	0,9127
10			0,7837	0,8141	0,8406	0,867	0,8887	0,8989	0,9132	0,9133	0,9133	0,9133	0,9134	0,9134	0,9134
11				0,808	0,839	0,8669	0,8894	0,8998	0,9141	0,9141	0,9141	0,9142	0,9142	0,9142	0,9142
12					0,8343	0,8652	0,8893	0,9003	0,9148	0,9147	0,9146	0,9146	0,9146	0,9147	0,9147
13						0,8615	0,8883	0,9008	0,9157	0,9155	0,9157	0,9158	0,9157	0,9158	0,916
14							0,8815	0,8984	0,9145	0,9144	0,9151	0,9154	0,9155	0,9156	0,9158
15								0,8932	0,9133	0,9138	0,9164	0,9168	0,9172	0,9174	0,9176
16									0,9105	0,913	0,9169	0,9178	0,9182	0,9183	0,9185
17										0,9113	0,9172	0,9196	0,9215	0,9223	0,9224
18											0,9107	0,9146	0,9177	0,9178	0,918
19												0,9112	0,9158	0,917	0,9204
20													0,9053	0,9073	0,9113
21														0,8939	0,902
22															0,877

13.pielikums. Hersta parametra kļūdas novērtēšana 2. eksperimentā

13 p. 18. tabula. Hersta parametra vidējā kļūda, pie $H = 0,8$.

Mērogu skaits	**Punktu skaits 1. mērogā**														
	2^{10}	2^{11}	2^{12}	2^{13}	2^{14}	2^{15}	2^{16}	2^{17}	2^{18}	2^{19}	2^{20}	2^{21}	2^{22}	2^{23}	2^{24}
2	0,8	0,8	0,8	0,8	0,8	0,8	0,8	0,8	0,8	0,8	0,8	0,8	0,8	0,8	0,8
3	0,0586	0,0442	0,0355	0,0306	0,0269	0,0256	0,0256	0,0256	0,0256	0,0257	0,0257	0,0257	0,0257	0,0257	0,0257
4	0,0568	0,0422	0,0321	0,0259	0,0221	0,0201	0,0198	0,0198	0,0198	0,0198	0,0198	0,0198	0,0198	0,0198	0,0198
5	0,0625	0,0448	0,0324	0,0245	0,0185	0,0148	0,0133	0,0131	0,0131	0,0131	0,0132	0,0132	0,0132	0,0132	0,0132
6	0,0763	0,0538	0,0378	0,0271	0,0187	0,0133	0,0102	0,0088	0,008	0,0079	0,0079	0,0079	0,0079	0,0079	0,0079
7	0,0977	0,0689	0,0481	0,0342	0,0233	0,0163	0,0127	0,0099	0,007	0,0054	0,0051	0,0051	0,0051	0,0051	0,0051
8	0,1222	0,0893	0,0629	0,0447	0,0301	0,0204	0,0151	0,0107	0,0072	0,0049	0,0033	0,0031	0,0024	0,0024	0,0024
9		0,1124	0,083	0,0603	0,0419	0,0285	0,0208	0,0139	0,0092	0,0057	0,0034	0,0029	0,0011	0,0006	0,0001
10			0,1058	0,0801	0,0586	0,0409	0,0295	0,0191	0,012	0,008	0,0063	0,005	0,0013	0,0004	0,0002
11				0,1006	0,0764	0,0555	0,0413	0,0279	0,0179	0,0127	0,0096	0,0081	0,0024	0,0013	0,0013
12					0,095	0,0716	0,0549	0,0391	0,0262	0,0191	0,0119	0,0099	0,005	0,0021	0,0021
13						0,0892	0,0693	0,0535	0,0388	0,0266	0,0182	0,0164	0,005	0,0049	0,0007
14							0,0856	0,0676	0,0509	0,0387	0,0298	0,0248	0,007	0,0055	0,0011
15								0,0837	0,0648	0,0518	0,0416	0,0338	0,015	0,0151	0,0012
16									0,0785	0,0655	0,0567	0,0428	0,0113	0,0045	0,0104
17										0,0777	0,0661	0,0467	0,0121	0,0011	0,0053
18											0,0778	0,0535	0,0208	0,0058	0,0015
19												0,0589	0,0246	0,0132	0,0092
20													0,0362	0,0286	0,0249
21														0,032	0,0276
22															0,0646

13 p. 19. tabula. Hersta parametra vidējā kļūda, pie $H = 0,85$.

Mērogu skaits	**Punktu skaits 1. mērogā**														
	2^{10}	2^{11}	2^{12}	2^{13}	2^{14}	2^{15}	2^{16}	2^{17}	2^{18}	2^{19}	2^{20}	2^{21}	2^{22}	2^{23}	2^{24}
2	0,85	0,85	0,85	0,85	0,85	0,85	0,85	0,85	0,85	0,85	0,85	0,85	0,85	0,85	0,85
3	0,0757	0,0596	0,0475	0,0387	0,0325	0,0268	0,0193	0,0113	0,0036	0,0027	0,0022	0,0022	0,0022	0,0022	0,0022
4	0,0741	0,058	0,0467	0,0384	0,0322	0,0267	0,0199	0,0119	0,0042	0,0036	0,0033	0,0031	0,0031	0,0031	0,003
5	0,0798	0,0618	0,0499	0,0408	0,0348	0,0293	0,0229	0,0158	0,0089	0,0086	0,0087	0,0086	0,0086	0,0086	0,0086
6	0,0915	0,0693	0,055	0,0451	0,0383	0,0326	0,0259	0,0185	0,0118	0,0117	0,0116	0,0116	0,0116	0,0116	0,0116
7	0,1115	0,0821	0,0635	0,0507	0,0422	0,0348	0,027	0,0196	0,0124	0,0123	0,0123	0,0122	0,0122	0,0122	0,0122
8	0,1382	0,1011	0,0769	0,0592	0,0471	0,0384	0,03	0,0217	0,0137	0,0135	0,0135	0,0133	0,0133	0,0133	0,0133
9		0,1264	0,0951	0,071	0,0544	0,0429	0,0316	0,0219	0,0137	0,013	0,0129	0,0128	0,0128	0,0128	0,0128
10			0,1175	0,0885	0,0658	0,0514	0,0369	0,025	0,0158	0,0144	0,0131	0,013	0,0129	0,0128	0,0129
11				0,1107	0,0826	0,0623	0,0438	0,0297	0,0175	0,0153	0,0141	0,0141	0,014	0,0139	0,0139
12					0,1026	0,0786	0,0558	0,0373	0,0218	0,0172	0,0143	0,0137	0,0136	0,0134	0,0134
13						0,0985	0,0714	0,0485	0,0278	0,0207	0,0155	0,0144	0,0138	0,0131	0,0131
14							0,0896	0,0641	0,0423	0,0311	0,0245	0,0225	0,0209	0,0166	0,0166
15								0,0825	0,06	0,0473	0,035	0,0322	0,03	0,0147	0,0142
16									0,078	0,0626	0,0458	0,0425	0,0409	0,0208	0,0189
17										0,0832	0,0617	0,0544	0,0435	0,0088	0,0033
18											0,0762	0,0676	0,0568	0,0226	0,0101
19												0,0877	0,0688	0,0336	0,0321
20													0,0813	0,0384	0,0236
21														0,0378	0,0182
22															0,0226

13 p. 20. tabula. Hersta parametra vidējā kļūda, pie $H = 0,9$.

Mērogu skaits	**Punktu skaits 1. mērogā**														
	2^{10}	2^{11}	2^{12}	2^{13}	2^{14}	2^{15}	2^{16}	2^{17}	2^{18}	2^{19}	2^{20}	2^{21}	2^{22}	2^{23}	2^{24}
2	0,9	0,9	0,9	0,9	0,9	0,9	0,9	0,9	0,9	0,9	0,9	0,9	0,9	0,9	0,9
3	0,0963	0,0758	0,0608	0,0487	0,0407	0,0298	0,0279	0,0243	0,0244	0,0244	0,0244	0,0244	0,0244	0,0244	0,0244
4	0,094	0,0754	0,0614	0,0507	0,0435	0,0331	0,0314	0,0279	0,028	0,028	0,028	0,028	0,0279	0,0279	0,0279
5	0,0982	0,0787	0,065	0,0544	0,0471	0,0366	0,0349	0,0315	0,0315	0,0315	0,0314	0,0314	0,0314	0,0314	0,0314
6	0,1069	0,0846	0,0688	0,0573	0,0498	0,0387	0,037	0,0335	0,0335	0,0335	0,0335	0,0335	0,0335	0,0335	0,0335
7	0,1201	0,092	0,0726	0,0595	0,051	0,0396	0,0374	0,0339	0,034	0,0339	0,0339	0,0339	0,0339	0,0339	0,0339
8	0,1438	0,1063	0,0806	0,063	0,0522	0,0402	0,0374	0,0337	0,0337	0,0337	0,0337	0,0337	0,0337	0,0336	0,0336
9		0,1276	0,0933	0,0696	0,0546	0,0408	0,0373	0,0332	0,0329	0,0328	0,0328	0,0328	0,0328	0,0328	0,0328
10			0,1147	0,083	0,0608	0,0431	0,037	0,0325	0,0317	0,0315	0,0314	0,0314	0,0313	0,0313	0,0313
11				0,1025	0,0727	0,049	0,0386	0,0323	0,0312	0,0307	0,0305	0,0304	0,0304	0,0303	0,0303
12					0,0919	0,0608	0,0448	0,0344	0,0321	0,031	0,0306	0,0306	0,0304	0,0303	0,0302
13						0,0782	0,0545	0,0398	0,0334	0,0306	0,029	0,0287	0,0284	0,0283	0,0282
14							0,0715	0,0488	0,0389	0,0304	0,027	0,0267	0,0261	0,026	0,0257
15								0,0628	0,0461	0,0291	0,025	0,0238	0,0224	0,0223	0,0221
16									0,0608	0,0337	0,0282	0,0226	0,0149	0,0146	0,0143
17										0,0424	0,032	0,0262	0,0105	0,0088	0,0084
18											0,044	0,0353	0,012	0,011	0,0093
19												0,0453	0,0302	0,0226	0,0211
20													0,0352	0,0293	0,0282
21														0,045	0,0437
22															0,0783

13 p. 21. tabula. Hersta parametra vidējā kļūda, pie $H = 0,95$.

Mērogu skaits	**Punktu skaits 1. mērogā**														
	2^{10}	2^{11}	2^{12}	2^{13}	2^{14}	2^{15}	2^{16}	2^{17}	2^{18}	2^{19}	2^{20}	2^{21}	2^{22}	2^{23}	2^{24}
2	0,95	0,95	0,95	0,95	0,95	0,95	0,95	0,95	0,95	0,95	0,95	0,95	0,95	0,95	0,95
3	0,1985	0,1756	0,1576	0,1406	0,1258	0,1133	0,0975	0,0868	0,08	0,08	0,0519	0,0519	0,0519	0,0519	0,0519
4	0,1967	0,1758	0,1597	0,1431	0,1286	0,1163	0,1007	0,0902	0,0832	0,0831	0,0552	0,0552	0,0552	0,0551	0,0551
5	0,1993	0,1786	0,1626	0,1463	0,1317	0,1195	0,104	0,0936	0,0866	0,0866	0,0588	0,0588	0,0587	0,0587	0,0587
6	0,2034	0,1814	0,1647	0,1482	0,1336	0,1211	0,1056	0,0952	0,0881	0,088	0,0602	0,0602	0,0602	0,0602	0,0602
7	0,2108	0,1849	0,1664	0,1492	0,1344	0,1217	0,1063	0,0957	0,0887	0,0885	0,0607	0,0607	0,0607	0,0608	0,0607
8	0,2244	0,1908	0,1683	0,1496	0,1339	0,121	0,1054	0,095	0,0879	0,0878	0,06	0,0601	0,0601	0,0601	0,06
9		0,2036	0,1733	0,1504	0,1336	0,1202	0,1042	0,0938	0,0869	0,0868	0,0588	0,059	0,059	0,059	0,059
10			0,1851	0,1562	0,1356	0,1209	0,1043	0,0936	0,0867	0,0865	0,0584	0,0586	0,0587	0,0587	0,0587
11				0,1661	0,1394	0,1207	0,102	0,0909	0,0838	0,0834	0,0554	0,0555	0,0556	0,0556	0,0556
12					0,1484	0,124	0,1024	0,0895	0,082	0,0816	0,0532	0,0534	0,0534	0,0533	0,0533
13						0,1331	0,1059	0,0891	0,0805	0,0795	0,0509	0,0512	0,0512	0,0512	0,051
14							0,116	0,0947	0,0827	0,0803	0,0499	0,0501	0,0502	0,0498	0,0497
15								0,1053	0,0879	0,0824	0,0511	0,0514	0,0515	0,0511	0,051
16									0,0947	0,0828	0,0499	0,0495	0,0497	0,0493	0,0493
17										0,0859	0,0488	0,0458	0,0438	0,0435	0,0433
18											0,0504	0,0465	0,0421	0,041	0,0401
19												0,0474	0,0424	0,0374	0,0374
20													0,0473	0,0424	0,0424
21														0,0424	0,0326
22															0,0354

13 p. 22. tabula. Hersta parametra vidējā kļūda, pie $H = 0,99$.

Mērogu skaits	**Punktu skaits 1. mērogā**														
	2^{10}	2^{11}	2^{12}	2^{13}	2^{14}	2^{15}	2^{16}	2^{17}	2^{18}	2^{19}	2^{20}	2^{21}	2^{22}	2^{23}	2^{24}
2	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99
3	0,2452	0,2139	0,1877	0,1643	0,1391	0,1139	0,0922	0,0813	0,0669	0,0669	0,0668	0,0668	0,0668	0,0668	0,0668
4	0,2447	0,2152	0,19	0,1674	0,1425	0,1173	0,0957	0,0847	0,0706	0,0707	0,0707	0,0706	0,0706	0,0706	0,0706
5	0,2465	0,2174	0,1928	0,1701	0,1454	0,1204	0,0989	0,0881	0,0741	0,0742	0,0741	0,0741	0,0741	0,0741	0,0741
6	0,2491	0,2193	0,1944	0,1718	0,1472	0,1222	0,1008	0,0901	0,0761	0,0763	0,0762	0,0762	0,0762	0,0762	0,0762
7	0,2534	0,2212	0,1957	0,1727	0,1482	0,1229	0,1016	0,0909	0,0769	0,077	0,077	0,077	0,077	0,077	0,077
8	0,2637	0,2261	0,198	0,1744	0,1497	0,1243	0,1029	0,0926	0,0781	0,0782	0,0782	0,0782	0,0782	0,0782	0,0782
9		0,2338	0,2008	0,1746	0,1493	0,1237	0,1021	0,0919	0,0774	0,0774	0,0774	0,0774	0,0774	0,0774	0,0773
10			0,2073	0,1763	0,1494	0,123	0,1013	0,0911	0,0768	0,0767	0,0767	0,0767	0,0766	0,0766	0,0766
11				0,1829	0,1513	0,1231	0,1006	0,0902	0,0759	0,0759	0,0759	0,0758	0,0758	0,0758	0,0758
12					0,1563	0,125	0,1008	0,0897	0,0752	0,0753	0,0754	0,0754	0,0754	0,0753	0,0753
13						0,1291	0,1019	0,0894	0,0743	0,0745	0,0743	0,0742	0,0743	0,0742	0,074
14							0,1088	0,0918	0,0755	0,0756	0,0749	0,0746	0,0745	0,0744	0,0742
15								0,0973	0,0767	0,0762	0,0736	0,0732	0,0728	0,0726	0,0724
16									0,0795	0,077	0,0731	0,0722	0,0718	0,0717	0,0715
17										0,0787	0,0728	0,0704	0,0685	0,0677	0,0676
18											0,0793	0,0754	0,0723	0,0722	0,072
19												0,0788	0,0742	0,073	0,0696
20													0,0847	0,0827	0,0787
21														0,0961	0,088
22															0,113

14.pielikums. Hersta parametra novērtēšanas programma *C++* valodā ar polifāzes filtru banku

```
#include <math.h>
#include <iostream>
#include <windows.h>
#include <stdlib.h>
#include "filters.cpp"
#include <fstream>
using namespace std;
#define K 5000          //Number of data points (16777216)
#define L 6           //Length of filters impulse response
#define J 8           //Number of DWT levels
#define LEN 1000
#define AVG_LEN 1000

int main(){
// Input file
ifstream file1("data09.txt");
if(!file1.is_open()) {cout<<"Input file error!\n"; system("PAUSE"); return 0;}

/*
//Output files
ofstream file2("approx_lvl8.txt");
if(!file2.is_open()) {cout<<"Output file error!\n"; system("PAUSE"); return 0;}
ofstream file3("detail_lvl8.txt");
if(!file3.is_open()) {cout<<"Output file error!\n"; system("PAUSE"); return 0;}
*/

int N=L-1, LL=L/2, NN=LL-1, n_J[J],num=0,j_k,NUM1[J],NUM2[J],kk[J],Hnum1,Hnum2;
int i,j,s=-1, MOD, hMOD; double y_a,y_d,x_a,yy,x_odd,D[J],logD[J],j_sum[J];
int j2_sum[J],j_sum2[J];
double MEM_ev[J][LL], MEM_odd[J][LL], x_ev[J], log_sum, log_jsum, a, Hurst;
double H_avg=0, y2_d, y2_d0, H_buf;
for(i=0; i<LL; i++) for(j=0;j<J;j++) {MEM_ev[i][j]=0; MEM_odd[i][j]=0;} //Memory

//Initialize j-sums for Hurst Estimator
j_sum[0]=1; j2_sum[0]=1; j_sum2[0]=1; D[0]=0; n_J[0]=0;
for(j=1;j<J;j++){
i=j+1; D[j]=0; n_J[j]=0;
```

```
j_sum[j]=i+j_sum[j-1];
j2_sum[j]=i*i+j2_sum[j-1];
j_sum2[j]=j_sum[j]*j_sum[j];
}

//Initialize register memory to store detail coefficients
double **registr = new double* [J];
for(j=0;j<J;j++){
registr[j] = new double [LEN]; //registr - buffer to store quotients
for(i=0;i<LEN;i++) registr[j][i]=0;
kk[j]=0;} //pointer to element of buffer (number of element)
double *HBuf = new double [AVG_LEN];
for(i=0;i<AVG_LEN;i++) HBuf[i]=0;

//Impulse Responses
double H1[L],F0[L],F1[L],t0,t1;
double H0[L]={0.035226291882100656, -0.085441273882241486,
-0.13501102001039084,0.45987750211933132,0.80689150931333875,
0.33267055295095688};
for(j=0;j<L;j++){
F0[j]=H0[L-j-1];        //Synthesis Approx. Filter
H1[j]=F0[j]*s;        //Analysis Detail. Filter
F1[j]=-H0[j]*s;       //Synthesis Detail. Filter
s=-s;
}
s=0;

//Time measurement
LARGE_INTEGER StartingTime, EndingTime, Frequency;
double ElapsedTime;

QueryPerformanceFrequency(&Frequency);
QueryPerformanceCounter(&StartingTime); //Start Timer


//Processing L-level DWT
for(i=0;i<K;i++){
file1>>x_a;
MOD=2; hMOD=1; //x_a=i;
log_sum=0; log_jsum=0;
```

```
for(j=0;j<J;j++){
if(i%hMOD==hMOD-1){ //Checking the rem(i,MOD/2). Skip level, if not
 //MAXed (leaving odd numbered)
//Calculating quotients only if sample is left after downsampling
if(i%(MOD)==MOD-1)
{
x_odd=x_a; //Being processed immediately, no need to save
filbank_polyphase_odd(x_ev[j], x_odd, H0, H1, MEM_ev[j], MEM_odd[j], y_a, y_d, NN);
//Output here y_a - approx., y_d - detail

//if ((j+1)==6) cout<<"y_a = "<<y_a<<"\t\t"<<"y_d = "<<y_d<<"\n";
//if ((j+1)==9) {file2<<x_a<<"\n"; file3<<y_d<<"\n";}

// ********************** Here comes Hurst parameter estimation
// Saving and shiftin quotients
j_k = n_J[j]; //Coeff. number
y2_d=y_d*y_d; //Square
y2_d0=registr[j][j_k]; //first element of sum
registr[j][j_k]=y2_d;

//Calculate length of coefficient data
if(kk[j]==0) {NUM1[j]=j_k; NUM2[j]=j_k+1;}
//Coefficient power logarithm calculation
D[j]=(D[j]*NUM1[j]+y2_d-y2_d0)/NUM2[j]; logD[j]=log(D[j])/log(2.0);
log_sum+=logD[j]; log_jsum+=(j+1)*logD[j];
if((j+1)==6) {
if(s==0) {Hnum1=num; Hnum2=num+1;}
a = (j+1)*log_jsum-j_sum[j]*log_sum;
a=a/((j+1)*j2_sum[j]-j_sum2[j]); Hurst=fabs(a+1)/2;
H_buf=HBuf[num];
H_avg=(H_avg*Hnum1+Hurst-H_buf)/(Hnum2); HBuf[num]=Hurst;
num=(num+1)%AVG_LEN; if(num==0) {Hnum1=AVG_LEN; Hnum2=AVG_LEN; s=1;}
cout<<Hurst<<"\t\t"<<H_avg<<"\n";}
if(j_k==LEN-1) {NUM1[j]=LEN; NUM2[j]=LEN; kk[j]=1;} //When register filled,
 //length maxed
n_J[j]=(n_J[j]+1)%LEN;
}
else x_ev[j]=x_a; //Otherwise, saving even sample. Needs to be saved
 //between samples for all scales (levels)!
}
```

```
 else break;
 hMOD=MOD; MOD*=2; x_a=y_a;
 }
 }

 QueryPerformanceCounter(&EndingTime); //Stop Timer
 //
 // We now have the elapsed number of ticks, along with the
 // number of ticks-per-second. We use these values
 // to convert to the number of elapsed miliseconds.
 // To guard against loss-of-precision, we convert
 // to microseconds *before* dividing by ticks-per-second.
 //
ElapsedTime=(EndingTime.QuadPart-StartingTime.QuadPart)*1000.0/Frequency.QuadPart;

 //Finalization
 cout<<"Total time: "<<ElapsedTime/1000<<" sec\n";
cout<<"Single transformation average time: "<<ElapsedTime*1000/K<<" microsec\n\n\n";
 file1.close();
 //    file2.close(); file3.close();
 delete [] HBuf;
 for (i = 0; i < J; i++) delete [] registr[i];
 return 0;
 }
```

15.pielikums. *GPSS P/M/1/K* modeļa simulācijas dati par vidējo rindas garumu un buferatmiņas apjomu

15 p. 23. tabula. *GPSS* $P/M/1/K$ modeļa simulācijas dati noslodzes koeficientam $\rho = 0,6$.

Hersta parametrs	**Ģenerēti pieprasījumi**	**Rindas vid. garums**	**Buferatmiņas apjoms**	**Apstrādāti pieprasījumi**
0,6	6003255	1	3	5791663
0,65	6006252	2	6	5953593
0,7	6012192	2	6	5929825
0,75	6019685	2	6	5877395
0,76	6023422	3	9	5967219
0,77	6023435	3	9	5956483
0,78	6028449	3	9	5948809
0,79	6034517	3	9	5938408
0,8	6041531	3	9	5925963
0,81	6050120	4	12	5982902
0,82	6059654	4	12	5972652
0,83	6074123	5	15	6011512
0,84	6092192	5	15	6006605
0,85	6115945	6	18	6039855
0,9	6358367	19	57	6292927
0,95	7586121	692	2076	7502513

15 p. 24. tabula. *GPSS* $P/M/1/K$ modeļa simulācijas dati noslodzes koeficientam $\rho = 0,7$.

Hersta parametrs	**Ģenerēti pieprasījumi**	**Rindas vid. garums**	**Buferatmiņas apjoms**	**Apstrādāti pieprasījumi**
0,6	6996597	2	6	6850732
0,65	6997730	3	9	6923759
0,7	7001094	3	9	6881404
0,75	7009837	4	12	6897875
0,76	7013106	5	15	6939212
0,77	7016998	5	15	6926685
0,78	7022153	6	18	6954318
0,79	7027146	6	18	6941009
0,8	7034635	7	21	6960638
0,81	7043400	8	24	6975300
0,82	7055406	9	27	6987221
0,83	7071377	10	30	6996651
0,84	7091504	12	36	7023061
0,85	7116801	14	42	7044481
0,9	7388051	60	180	7313113
0,95	9064437	6439	19317	8822146

15 p. 25. tabula. *GPSS* $P/M/1/K$ modeļa simulācijas dati noslodzes koeficientam $\rho = 0,8$.

Hersta parametrs	**Ģenerēti pieprasījumi**	**Rindas vid. garums**	**Buferatmiņas apjoms**	**Apstrādāti pieprasījumi**
0,6	7997999	5	15	7935598
0,65	8000165	5	15	7902398
0,7	8005588	7	21	7930899
0,75	8018234	10	30	7948445
0,76	8022975	11	33	7953842
0,77	8028301	12	36	7958250
0,78	8034838	13	39	7960881
0,79	8043349	15	45	7974197
0,8	8053976	17	51	7985348
0,81	8066722	20	60	7999426
0,82	8082920	23	69	8013198
0,83	8103461	28	84	8035262
0,84	8130343	34	102	8060407
0,85	8162730	42	126	8087633
0,9	8545458	383	1149	8488196
0,95	10302634	13195	39585	9219814

15 p. 26. tabula. *GPSS* $P/M/1/K$ modeļa simulācijas dati noslodzes koeficientam $\rho = 0,9$.

Hersta parametrs	**Ģenerēti pieprasījumi**	**Rindas vid. garums**	**Buferatmiņas apjoms**	**Apstrādāti pieprasījumi**
0,6	9005232	12	36	8948505
0,65	9010482	15	45	8952477
0,7	9020436	20	60	8957434
0,75	9040641	33	99	8979616
0,76	9047390	38	114	8988603
0,77	9055778	44	132	8997194
0,78	9065720	53	159	9011709
0,79	9072929	66	198	9022858
0,8	9087061	85	255	9046041
0,81	9103052	109	327	9062739
0,82	9123274	146	438	9086389
0,83	9149932	195	585	9113983
0,84	9180927	256	768	9140380
0,85	9222106	354	1062	9174858
0,9	9635652	4156	12468	9440614
0,95	11417269	20626	61878	9089631

16.pielikums. Dominējošās secības dažādām H parametra vērtībām.

	ρ = 0,6	ρ = 0,7	ρ = 0,8	ρ = 0,9
$K = 1E[R]$	172,8 $ un 0,01 $ 0,13188 172,81 $	201,6 $ un 0,02 $ 0,10799 201,62 $	230,4 $ un 0,04 $ 0,064416 230,44 $	259,2 $ un 0,09 $ 0,045084 259,29 $
$K = 2E[R]$	172,8 $ un 0,02 $ 0,066126 172,82 $	201,6 $ un 0,04 $ 0,045503 201,64 $	230,4 $ un 0,08 $ 0,020972 230,48 $	259,2 $ un 0,18 $ 0,015549 259,38 $
$K = 3E[R]$	172,8 $ un 0,03 $ 0,035057 172,83 $	201,6 $ un 0,06 $ 0,020802 201,66 $	230,4 $ un 0,12 $ 0,0076529 230,52 $	259,2 $ un 0,27 $ 0,0061104 259,47 $
$K = 4E[R]$	172,8 $ un 0,04 $ 0,01903 172,84 $	201,6 $ un 0,08 $ 0,0099107 201,68 $	230,4 $ un 0,16 $ 0,0029914 230,56 $	259,2 $ un 0,36 $ 0,0026278 259,56 $
$K = 5E[R]$	172,8 $ un 0,05 $ 0,010476 172,85 $	201,6 $ un 0,1 $ 0,0048461 201,7 $	230,4 $ un 0,2 $ 0,0011542 230,6 $	259,2 $ un 0,45 $ 0,0011183 259,65 $
$K = 6E[R]$	172,8 $ un 0,06 $ 0,0058272 172,86 $	201,6 $ un 0,12 $ 0,0023946 201,72 $	230,4 $ un 0,24 $ 0,00042973 230,64 $	259,2 $ un 0,54 $ 0,00050093 259,74 $
$K = 7E[R]$	172,8 $ un 0,07 $ 0,0032816 172,87 $	201,6 $ un 0,14 $ 0,001191 201,74 $	230,4 $ un 0,28 $ 0,00018692 230,68 $	259,2 $ un 0,63 $ 0,00020488 259,83 $
$K = 8E[R]$	172,8 $ un 0,08 $ 0,0018155 172,88 $	201,6 $ un 0,16 $ 0,00060329 201,76 $	230,4 $ un 0,32 $ $7{,}3643 \cdot 10^{-5}$ 230,72 $	259,2 $ un 0,72 $ $9{,}4168 \cdot 10^{-5}$ 259,92 $
$K = 9E[R]$	172,8 $ un 0,09 $ 0,0010218 172,89 $	201,6 $ un 0,18 $ 0,00031272 201,78 $	230,4 $ un 0,36 $ $2{,}1255 \cdot 10^{-5}$ 230,76 $	259,2 $ un 0,81 $ $3{,}8422 \cdot 10^{-5}$ 260,01 $
$K = 10E[R]$	172,8 $ un 0,1 $ 0,00056886 172,9 $	201,6 $ un 0,2 $ 0,00015836 201,8 $	230,4 $ un 0,4 $ $5{,}8765 \cdot 10^{-6}$ 230,8 $	259,2 $ un 0,9 $ $8{,}7727 \cdot 10^{-6}$ 260,1 $

16 p. 1. att. Dominējošās secības formēšana ar Belmana algoritmu $P/M/1/K$ modelim ar Hersta parametru $H = 0,6$ un uzdotajām parametru izmaksām.

	ρ = 0,6	ρ = 0,7	ρ = 0,8	ρ = 0,9
$K = 1E[R]$	172,8 $ un 0,02 $ 0,077867 172,82 $	201,6 $ un 0,02 $ 0,08134 201,62 $	230,4 $ un 0,04 $ 0,077502 230,44 $	259,2 $ un 0,12 $ 0,043883 259,32 $
$K = 2E[R]$	172,8 $ un 0,04 $ 0,025243 172,84 $	201,6 $ un 0,04 $ 0,027892 201,64 $	230,4 $ un 0,08 $ 0,028958 230,48 $	259,2 $ un 0,24 $ 0,01536 259,44 $
$K = 3E[R]$	172,8 $ un 0,06 $ 0,0086911 172,86 $	201,6 $ un 0,06 $ 0,010459 201,66 $	230,4 $ un 0,12 $ 0,012079 230,52 $	259,2 $ un 0,36 $ 0,0064343 259,56 $
$K = 4E[R]$	172,8 $ un 0,08 $ 0,0030598 172,88 $	201,6 $ un 0,08 $ 0,0040497 201,68 $	230,4 $ un 0,16 $ 0,0053839 230,56 $	259,2 $ un 0,48 $ 0,0027395 259,68 $
$K = 5E[R]$	172,8 $ un 0,1 $ 0,0010917 172,9 $	201,6 $ un 0,1 $ 0,0016953 201,7 $	230,4 $ un 0,2 $ 0,0024792 230,6 $	259,2 $ un 0,6 $ 0,0012833 259,8 $
$K = 6E[R]$	172,8 $ un 0,12 $ 0,00038893 172,92 $	201,6 $ un 0,12 $ 0,00062177 201,72 $	230,4 $ un 0,24 $ 0,0011151 230,64 $	259,2 $ un 0,72 $ 0,00053671 259,92 $
$K = 7E[R]$	172,8 $ un 0,14 $ 0,00015134 172,94 $	201,6 $ un 0,14 $ 0,00026694 201,74 $	230,4 $ un 0,28 $ 0,00047512 230,68 $	259,2 $ un 0,84 $ 0,00027968 260,04 $
$K = 8E[R]$	172,8 $ un 0,16 $ $5{,}4943 \cdot 10^{-5}$ 172,96 $	201,6 $ un 0,16 $ 0,00011046 201,76 $	230,4 $ un 0,32 $ 0,00025962 230,72 $	259,2 $ un 0,96 $ 0,00012075 260,16 $
$K = 9E[R]$	172,8 $ un 0,18 $ $1{,}898 \cdot 10^{-5}$ 172,98 $	201,6 $ un 0,18 $ $4{,}0299 \cdot 10^{-5}$ 201,78 $	230,4 $ un 0,36 $ $9{,}9123 \cdot 10^{-5}$ 230,76 $	259,2 $ un 1,08 $ $5{,}4271 \cdot 10^{-5}$ 260,28 $
$K = 10E[R]$	172,8 $ un 0,2 $ 9,6566·10-06 173 $	201,6 $ un 0,2 $ $1{,}7434 \cdot 10^{-5}$ 201,8 $	230,4 $ un 0,4 $ $3{,}9499 \cdot 10^{-5}$ 230,8 $	259,2 $ un 1,2 $ $1{,}7313 \cdot 10^{-5}$ 260,4 $

16 p. 2. att. Dominējošās secības formēšana ar Belmana algoritmu $P/M/1/K$ modelim ar Hersta parametru $H = 0,65$ un uzdotajām parametru izmaksām.

	ρ = 0,6	ρ = 0,7	ρ = 0,8	ρ = 0,9
$K = 1E[R]$	172,8 $ un 0,02 $ 0,09396 172,82 $	201,6 $ un 0,02 $ 0,098922 201,62 $	230,4 $ un 0,05 $ 0,066611 230,45 $	259,2 $ un 0,15 $ 0,043756 259,35 $
$K = 2E[R]$	172,8 $ un 0,04 $ 0,034693 172,84 $	201,6 $ un 0,04 $ 0,039011 201,64 $	230,4 $ un 0,1 $ 0,023156 230,5 $	259,2 $ un 0,3 $ 0,015969 259,5 $
$K = 3E[R]$	172,8 $ un 0,06 $ 0,013641 172,86 $	201,6 $ un 0,06 $ 0,017099 201,66 $	230,4 $ un 0,15 $ 0,0090936 230,55 $	259,2 $ un 0,45 $ 0,0068367 259,65 $
$K = 4E[R]$	172,8 $ un 0,08 $ 0,0055544 172,88 $	201,6 $ un 0,08 $ 0,0077786 201,68 $	230,4 $ un 0,2 $ 0,0038459 230,6 $	259,2 $ un 0,6 $ 0,0032804 259,8 $
$K = 5E[R]$	172,8 $ un 0,1 $ 0,0023246 172,9 $	201,6 $ un 0,1 $ 0,0036454 201,7 $	230,4 $ un 0,25 $ 0,0015213 230,65 $	259,2 $ un 0,75 $ 0,0015337 259,95 $
$K = 6E[R]$	172,8 $ un 0,12 $ 0,00096637 172,92 $	201,6 $ un 0,12 $ 0,001699 201,72 $	230,4 $ un 0,3 $ 0,00066541 230,7 $	259,2 $ un 0,9 $ 0,00068034 260,1 $
$K = 7E[R]$	172,8 $ un 0,14 $ 0,00038904 172,94 $	201,6 $ un 0,14 $ 0,00078831 201,74 $	230,4 $ un 0,35 $ 0,00032402 230,75 $	259,2 $ un 1,05 $ 0,00034788 260,25 $
$K = 8E[R]$	172,8 $ un 0,16 $ 0,00017647 172,96 $	201,6 $ un 0,16 $ 0,00035337 201,76 $	230,4 $ un 0,4 $ 0,00012604 230,8 $	259,2 $ un 1,2 $ 0,00017959 260,4 $
$K = 9E[R]$	172,8 $ un 0,18 $ $6{,}8028 \cdot 10^{-5}$ 172,98 $	201,6 $ un 0,18 $ 0,0001914 201,78 $	230,4 $ un 0,45 $ $3{,}635 \cdot 10^{-5}$ 230,85 $	259,2 $ un 1,35 $ $8{,}3921 \cdot 10^{-5}$ 260,55 $
$K = 10E[R]$	172,8 $ un 0,2 $ $2{,}9939 \cdot 10^{-5}$ 173 $	201,6 $ un 0,2 $ $8{,}4415 \cdot 10^{-5}$ 201,8 $	230,4 $ un 0,5 $ $1{,}5364 \cdot 10^{-5}$ 230,9 $	259,2 $ un 1,5 $ $3{,}359 \cdot 10^{-5}$ 260,7 $

16 p. 3. att. Dominējošās secības formēšana ar Belmana algoritmu $P/M/1/K$ modelim ar Hersta parametru $H = 0,7$ un uzdotajām parametru izmaksām.

	ρ = 0,6	ρ = 0,7	ρ = 0,8	ρ = 0,9
$K = 1E[R]$	172,8 $ un 0,02 $ 0,11764 172,82 $	201,6 $ un 0,03 $ 0,094369 201,63 $	230,4 $ un 0,08 $ 0,062161 230,48 $	259,2 $ un 0,25 $ 0,039561 259,45 $
$K = 2E[R]$	172,8 $ un 0,04 $ 0,050363 172,84 $	201,6 $ un 0,06 $ 0,036534 201,66 $	230,4 $ un 0,16 $ 0,021578 230,56 $	259,2 $ un 0,5 $ 0,015163 259,7 $
$K = 3E[R]$	172,8 $ un 0,06 $ 0,023361 172,86 $	201,6 $ un 0,09 $ 0,015728 201,69 $	230,4 $ un 0,24 $ 0,0085612 230,64 $	259,2 $ un 0,75 $ 0,0067291 259,95 $
$K = 4E[R]$	172,8 $ un 0,08 $ 0,011205 172,88 $	201,6 $ un 0,12 $ 0,0069956 201,72 $	230,4 $ un 0,32 $ 0,0036826 230,72 $	259,2 $ un 1 $ 0,0032132 260,2 $
$K = 5E[R]$	172,8 $ un 0,1 $ 0,0054792 172,9 $	201,6 $ un 0,15 $ 0,0033009 201,75 $	230,4 $ un 0,4 $ 0,0014578 230,8 $	259,2 $ un 1,25 $ 0,0015719 260,45 $
$K = 6E[R]$	172,8 $ un 0,12 $ 0,0026694 172,92 $	201,6 $ un 0,18 $ 0,0014622 201,78 $	230,4 $ un 0,48 $ 0,00062408 230,88 $	259,2 $ un 1,5 $ 0,00072517 260,7 $
$K = 7E[R]$	172,8 $ un 0,14 $ 0,0012876 172,94 $	201,6 $ un 0,21 $ 0,00066421 201,81 $	230,4 $ un 0,56 $ 0,00030293 230,96 $	259,2 $ un 1,75 $ 0,00035241 260,95 $
$K = 8E[R]$	172,8 $ un 0,16 $ 0,0006188 172,96 $	201,6 $ un 0,24 $ 0,00033367 201,84 $	230,4 $ un 0,64 $ 0,00013881 231,04 $	259,2 $ un 2 $ 0,00019932 261,2 $
$K = 9E[R]$	172,8 $ un 0,18 $ 0,00032228 172,98 $	201,6 $ un 0,27 $ 0,00016677 201,87 $	230,4 $ un 0,72 $ $4{,}9138 \cdot 10^{-5}$ 231,12 $	259,2 $ un 2,25 $ $9{,}4905 \cdot 10^{-5}$ 261,45 $
$K = 10E[R]$	172,8 $ un 0,2 $ 0,00015682 173 $	201,6 $ un 0,3 $ $8{,}2028 \cdot 10^{-5}$ 201,9 $	230,4 $ un 0,8 $ $1{,}534 \cdot 10^{-5}$ 231,2 $	259,2 $ un 2,5 $ $3{,}6502 \cdot 10^{-5}$ 261,7 $

16 p. 4. att. Dominējošās secības formēšana ar Belmana algoritmu $P/M/1/K$ modelim ar Hersta parametru $H = 0,75$ un uzdotajām parametru izmaksām.

	ρ = 0,6	ρ = 0,7	ρ = 0,8	ρ = 0,9
$K = 1E[R]$	172,8 $ un 0,02 $ 0,10748 172,82 $	201,6 $ un 0,05 $ 0,075375 201,65 $	230,4 $ un 0,13 $ 0,057081 230,53 $	259,2 $ un 0,66 $ 0,029627 259,86 $
$K = 2E[R]$	172,8 $ un 0,04 $ 0,042893 172,84 $	201,6 $ un 0,1 $ 0,026055 201,7 $	230,4 $ un 0,26 $ 0,020595 230,66 $	259,2 $ un 1,32 $ 0,011168 260,52 $
$K = 3E[R]$	172,8 $ un 0,06 $ 0,018967 172,86 $	201,6 $ un 0,15 $ 0,010185 201,75 $	230,4 $ un 0,39 $ 0,0085718 230,79 $	259,2 $ un 1,98 $ 0,0048284 261,18 $
$K = 4E[R]$	172,8 $ un 0,08 $ 0,0084389 172,88 $	201,6 $ un 0,2 $ 0,0039853 201,8 $	230,4 $ un 0,52 $ 0,0038761 230,92 $	259,2 $ un 2,64 $ 0,0023069 261,84 $
$K = 5E[R]$	172,8 $ un 0,1 $ 0,0038934 172,9 $	201,6 $ un 0,25 $ 0,0016982 201,85 $	230,4 $ un 0,65 $ 0,0015656 231,05 $	259,2 $ un 3,3 $ 0,00099385 262,5 $
$K = 6E[R]$	172,8 $ un 0,12 $ 0,0018009 172,92 $	201,6 $ un 0,3 $ 0,00066002 201,9 $	230,4 $ un 0,78 $ 0,00068674 231,18 $	259,2 $ un 3,96 $ 0,00045967 263,16 $
$K = 7E[R]$	172,8 $ un 0,14 $ 0,00080708 172,94 $	201,6 $ un 0,35 $ 0,00031331 201,95 $	230,4 $ un 0,91 $ 0,00036305 231,31 $	259,2 $ un 4,62 $ 0,00021448 263,82 $
$K = 8E[R]$	172,8 $ un 0,16 $ 0,0004042 172,96 $	201,6 $ un 0,4 $ 0,00013092 202 $	230,4 $ un 1,04 $ 0,00018028 231,44 $	259,2 $ un 5,28 $ 0,00011016 264,48 $
$K = 9E[R]$	172,8 $ un 0,18 $ 0,00019813 172,98 $	201,6 $ un 0,45 $ $3{,}9377 \cdot 10^{-5}$ 202,05 $	230,4 $ un 1,17 $ $7{,}5242 \cdot 10^{-5}$ 231,57 $	259,2 $ un 5,94 $ $5{,}1062 \cdot 10^{-5}$ 265,14 $
$K = 10E[R]$	172,8 $ un 0,2 $ $8{,}9547 \cdot 10^{-5}$ 173 $	201,6 $ un 0,5 $ $1{,}3647 \cdot 10^{-5}$ 202,1 $	230,4 $ un 1,3 $ $1{,}6265 \cdot 10^{-5}$ 231,7 $	259,2 $ un 6,6 $ $3{,}0703 \cdot 10^{-5}$ 265,8 $

16 p. 5. att. Dominējošās secības formēšana ar Belmana algoritmu $P/M/1/K$ modelim ar Hersta parametru $H = 0,8$ un uzdotajām parametru izmaksām.

	ρ = 0,6	ρ = 0,7	ρ = 0,8	ρ = 0,9
$K = 1E[R]$	172,8 $ un 0,05 $ 0,086042 172,85 $	201,6 $ un 0,11 $ 0,067711 201,71 $	230,4 $ un 0,32 $ 0,051398 230,72 $	259,2 $ un 2,73 $ 0,025692 261,93 $
$K = 2E[R]$	172,8 $ un 0,1 $ 0,03065 172,9 $	201,6 $ un 0,22 $ 0,024191 201,82 $	230,4 $ un 0,64 $ 0,020594 231,04 $	259,2 $ un 5,46 $ 0,010782 264,66 $
$K = 3E[R]$	172,8 $ un 0,15 $ 0,01199 172,95 $	201,6 $ un 0,33 $ 0,0098338 201,93 $	230,4 $ un 0,96 $ 0,0091642 231,36 $	259,2 $ un 8,19 $ 0,00542 267,39 $
$K = 4E[R]$	172,8 $ un 0,2 $ 0,004881 173 $	201,6 $ un 0,44 $ 0,0042259 202,04 $	230,4 $ un 1,28 $ 0,0043704 231,68 $	259,2 $ un 10,92 $ 0,0027531 270,12 $
$K = 5E[R]$	172,8 $ un 0,25 $ 0,0020113 173,05 $	201,6 $ un 0,55 $ 0,0018347 202,15 $	230,4 $ un 1,6 $ 0,0021113 232 $	259,2 $ un 13,65 $ 0,0013178 272,85 $
$K = 6E[R]$	172,8 $ un 0,3 $ 0,00079726 173,1 $	201,6 $ un 0,66 $ 0,00073545 202,26 $	230,4 $ un 1,92 $ 0,0010569 232,32 $	259,2 $ un 16,38 $ 0,00061818 275,58 $
$K = 7E[R]$	172,8 $ un 0,35 $ 0,0003571 173,15 $	201,6 $ un 0,77 $ 0,00033569 202,37 $	230,4 $ un 2,24 $ 0,00052372 232,64 $	259,2 $ un 19,11 $ 0,00020277 278,31 $
$K = 8E[R]$	172,8 $ un 0,4 $ 0,00015631 173,2 $	201,6 $ un 0,88 $ 0,00016918 202,48 $	230,4 $ un 2,56 $ 0,00025065 232,96 $	259,2 $ un 21,84 $ $6{,}6253 \cdot 10^{-5}$ 281,04 $
$K = 9E[R]$	172,8 $ un 0,45 $ $7{,}9301 \cdot 10^{-5}$ 173,25 $	201,6 $ un 0,99 $ $7{,}4191 \cdot 10^{-5}$ 202,59 $	230,4 $ un 2,88 $ 0,00010989 233,28 $	259,2 $ un 24,57 $ $1{,}6265 \cdot 10^{-6}$ 283,77 $
$K = 10E[R]$	172,8 $ un 0,5 $ $1{,}8476 \cdot 10^{-5}$ 173,3 $	201,6 $ un 1,1 $ $1{,}5597 \cdot 10^{-5}$ 202,7 $	230,4 $ un 3,2 $ $4{,}4593 \cdot 10^{-5}$ 233,6 $	259,2 $ un 27,3 $ – 286,5 $

16 p. 6. att. Dominējošās secības formēšana ar Belmana algoritmu $P/M/1/K$ modelim ar Hersta parametru $H = 0,85$ un uzdotajām parametru izmaksām.

	ρ = 0,6	ρ = 0,7	ρ = 0,8	ρ = 0,9
K = 1$E[R]$	172,8 $ un 0,15 $ 0,069146 172,95 $	201,6 $ un 0,46 $ 0,056062 202,06 $	230,4 $ un 2,96 $ 0,035225 233,36 $	259,2 $ un 32,1 $ 0,037615 291,3 $
K = 2$E[R]$	172,8 $ un 0,3 $ 0,024942 173,1 $	201,6 $ un 0,92 $ 0,022247 202,52 $	230,4 $ un 5,92 $ 0,014434 236,32 $	259,2 $ un 64,2 $ 0,02268 323,4 $
K = 3$E[R]$	172,8 $ un 0,45 $ 0,0099848 173,25 $	201,6 $ un 1,38 $ 0,010205 202,98 $	230,4 $ un 8,88 $ 0,0068384 239,28 $	259,2 $ un 96,3 $ 0,014458 355,5 $
K = 4$E[R]$	172,8 $ un 0,6 $ 0,004169 173,4 $	201,6 $ un 1,84 $ 0,0046351 203,44 $	230,4 $ un 11,84 $ 0,0035641 242,24 $	259,2 $ un 128,4 $ 0,010936 387,6 $
K = 5$E[R]$	172,8 $ un 0,75 $ 0,0018247 173,55 $	201,6 $ un 2,3 $ 0,0022862 203,9 $	230,4 $ un 14,8 $ 0,0017763 245,2 $	259,2 $ un 160,5 $ 0,0094481 419,7 $
K = 6$E[R]$	172,8 $ un 0,9 $ 0,00063947 173,7 $	201,6 $ un 2,76 $ 0,0011186 204,36 $	230,4 $ un 17,76 $ 0,0008491 248,16 $	259,2 $ un 192,6 $ 0,0081626 451,8 $
K = 7$E[R]$	172,8 $ un 1,05 $ 0,00032084 173,85 $	201,6 $ un 3,22 $ 0,00058243 204,82 $	230,4 $ un 20,72 $ 0,000363 251,12 $	259,2 $ un 224,7 $ 0,0070178 483,9 $
K = 8$E[R]$	172,8 $ un 1,2 $ 0,00014972 174 $	201,6 $ un 3,68 $ 0,00029332 205,28 $	230,4 $ un 23,68 $ 0,00010871 254,08 $	259,2 $ un 256,8 $ 0,0061713 516 $
K = 9$E[R]$	172,8 $ un 1,35 $ $5{,}4574 \cdot 10^{-5}$ 174,15 $	201,6 $ un 4,14 $ 0,00013103 205,74 $	230,4 $ un 26,64 $ – 257,04 $	259,2 $ un 288,9 $ 0,0053092 548,1 $
K = 10$E[R]$	172,8 $ un 1,5 $ $6{,}92 \cdot 10^{-6}$ 174,3 $	201,6 $ un 4,6 $ $5{,}7121 \cdot 10^{-5}$ 206,2 $	230,4 $ un 29,6 $ – 260 $	259,2 $ un 321 $ 0,004448 580,2 $

16 p. 7. att. Dominējošās secības formēšana ar Belmana algoritmu $P/M/1/K$ modelim ar Hersta parametru $H = 0,9$ un uzdotajām parametru izmaksām.

	ρ = 0,6	ρ = 0,7	ρ = 0,8	ρ = 0,9
K = 1$E[R]$	172,8 $ un 5,35 $ 0,047263 178,15 $	201,6 $ un 49,74 $ 0,052686 251,34 $	230,4 $ un 101,93 $ 0,12446 332,33 $	259,2 $ un 159,33 $ 0,19471 418,53 $
K = 2$E[R]$	172,8 $ un 10,7 $ 0,019553 183,5 $	201,6 $ un 99,48 $ 0,029392 301,08 $	230,4 $ un 203,86 $ 0,098489 434,26 $	259,2 $ un 318,66 $ 0,17307 577,86 $
K = 3$E[R]$	172,8 $ un 16,05 $ 0,0096603 188,85 $	201,6 $ un 149,22 $ 0,017686 350,82 $	230,4 $ un 305,79 $ 0,08418 536,19 $	259,2 $ un 477,99 $ 0,15699 737,19 $
K = 4$E[R]$	172,8 $ un 21,4 $ 0,0047397 194,2 $	201,6 $ un 198,96 $ 0,012872 400,56 $	230,4 $ un 407,72 $ 0,07385 638,12 $	259,2 $ un 637,32 $ 0,14396 896,52 $
K = 5$E[R]$	172,8 $ un 26,75 $ 0,0022965 199,55 $	201,6 $ un 248,7 $ 0,010911 450,3 $	230,4 $ un 509,65 $ 0,06483 740,05 $	259,2 $ un 796,65 $ 0,134 1055,85 $
K = 6$E[R]$	172,8 $ un 32,1 $ 0,0010078 204,9 $	201,6 $ un 298,44 $ 0,0094927 500,04 $	230,4 $ un 611,58 $ 0,055907 841,98 $	259,2 $ un 955,98 $ 0,12677 1215,18 $
K = 7$E[R]$	172,8 $ un 37,45 $ 0,00041537 210,25 $	201,6 $ un 348,18 $ 0,0080878 549,78 $	230,4 $ un 713,51 $ 0,04822 943,91 $	259,2 $ un 1115,31 $ 0,12223 1374,51 $
K = 8$E[R]$	172,8 $ un 42,8 $ 0,00010981 215,6 $	201,6 $ un 397,92 $ 0,0066797 599,52 $	230,4 $ un 815,44 $ 0,041824 1045,84 $	259,2 $ un 1274,64 $ 0,12041 1533,84 $
K = 9$E[R]$	172,8 $ un 48,15 $ – 220,95 $	201,6 $ un 447,66 $ 0,0052576 649,26 $	230,4 $ un 917,37 $ 0,03544 1147,77 $	259,2 $ un 1433,97 $ 0,1186 1693,17 $
K = 10$E[R]$	172,8 $ un 53,5 $ – 226,3 $	201,6 $ un 497,4 $ 0,0038522 699 $	230,4 $ un 1019,3 $ 0,030818 1249,7 $	259,2 $ un 1593,3 $ 0,117 1852,5 $

16 p. 8. att. Dominējošās secības formēšana ar Belmana algoritmu $P/M/1/K$ modelim ar Hersta parametru $H = 0,95$ un uzdotajām parametru izmaksām.

]

	ρ = 0,6	ρ = 0,7	ρ = 0,8	ρ = 0,9
$K = 1E[R]$	172,8 $ un 0,02 $ 0,065047 172,82 $	201,6 $ un 0,02 $ 0,10702 201,62 $	230,4 $ un 0,03 $ 0,081375 230,43 $	259,2 $ un 0,08 $ 0,051679 259,28 $
$K = 2E[R]$	172,8 $ un 0,04 $ 0,018646 172,84 $	201,6 $ un 0,04 $ 0,044273 201,64 $	230,4 $ un 0,06 $ 0,030066 230,46 $	259,2 $ un 0,16 $ 0,016839 259,36 $
$K = 3E[R]$	172,8 $ un 0,06 $ 0,0055316 172,86 $	201,6 $ un 0,06 $ 0,019809 201,66 $	230,4 $ un 0,09 $ 0,012516 230,49 $	259,2 $ un 0,24 $ 0,0064181 259,44 $
$K = 4E[R]$	172,8 $ un 0,08 $ 0,0017569 172,88 $	201,6 $ un 0,08 $ 0,0091451 201,68 $	230,4 $ un 0,12 $ 0,0053033 230,52 $	259,2 $ un 0,32 $ 0,002421 259,52 $
$K = 5E[R]$	172,8 $ un 0,1 $ 0,00055529 172,9 $	201,6 $ un 0,1 $ 0,0043907 201,7 $	230,4 $ un 0,15 $ 0,0024166 230,55 $	259,2 $ un 0,4 $ 0,00084942 259,6 $
$K = 6E[R]$	172,8 $ un 0,12 $ 0,00016706 172,92 $	201,6 $ un 0,12 $ 0,0021395 201,72 $	230,4 $ un 0,18 $ 0,00099133 230,58 $	259,2 $ un 0,48 $ 0,00032197 259,68 $
$K = 7E[R]$	172,8 $ un 0,14 $ $4{,}4236 \cdot 10^{-5}$ 172,94 $	201,6 $ un 0,14 $ 0,0010436 201,74 $	230,4 $ un 0,21 $ 0,0004281 230,61 $	259,2 $ un 0,56 $ 0,00014186 259,76 $
$K = 8E[R]$	172,8 $ un 0,16 $ $2{,}6021 \cdot 10^{-5}$ 172,96 $	201,6 $ un 0,16 $ 0,000539 201,76 $	230,4 $ un 0,24 $ 0,00017821 230,64 $	259,2 $ un 0,64 $ $5{,}737 \cdot 10^{-5}$ 259,84 $
$K = 9E[R]$	172,8 $ un 0,18 $ $5{,}2042 \cdot 10^{-6}$ 172,98 $	201,6 $ un 0,18 $ 0,00025565 201,78 $	230,4 $ un 0,27 $ $8{,}7344 \cdot 10^{-5}$ 230,67 $	259,2 $ un 0,72 $ $2{,}4686 \cdot 10^{-5}$ 259,92 $
$K = 10E[R]$	172,8 $ un 0,2 $ $2{,}6021 \cdot 10^{-6}$ 173 $	201,6 $ un 0,2 $ 0,00012961 201,8 $	230,4 $ un 0,3 $ $5{,}7576 \cdot 10^{-5}$ 230,7 $	259,2 $ un 0,8 $ $8{,}3447 \cdot 10^{-6}$ 260 $

16 p. 9. att. Dominējošās secības formēšana ar Belmana algoritmu $P/M/1/K$ *ON/OFF* trafika modelim ar Hersta parametru $H = 0,6$ un uzdotajām parametru izmaksām.

	ρ = 0,6	ρ = 0,7	ρ = 0,8	ρ = 0,9
$K = 1E[R]$	172,8 $ un 0,02 $ 0,076829 172,82 $	201,6 $ un 0,02 $ 0,079804 201,62 $	230,4 $ un 0,04 $ 0,075094 230,44 $	259,2 $ un 0,09 $ 0,051487 259,29 $
$K = 2E[R]$	172,8 $ un 0,04 $ 0,024573 172,84 $	201,6 $ un 0,04 $ 0,02673 201,64 $	230,4 $ un 0,08 $ 0,02675 230,48 $	259,2 $ un 0,18 $ 0,017389 259,38 $
$K = 3E[R]$	172,8 $ un 0,06 $ 0,0082525 172,86 $	201,6 $ un 0,06 $ 0,0097066 201,66 $	230,4 $ un 0,12 $ 0,010334 230,52 $	259,2 $ un 0,27 $ 0,0064134 259,47 $
$K = 4E[R]$	172,8 $ un 0,08 $ 0,0028823 172,88 $	201,6 $ un 0,08 $ 0,0036979 201,68 $	230,4 $ un 0,16 $ 0,0042093 230,56 $	259,2 $ un 0,36 $ 0,0024341 259,56 $
$K = 5E[R]$	172,8 $ un 0,1 $ 0,0010537 172,9 $	201,6 $ un 0,1 $ 0,0013917 201,7 $	230,4 $ un 0,2 $ 0,0017547 230,6 $	259,2 $ un 0,45 $ 0,00087536 259,65 $
$K = 6E[R]$	172,8 $ un 0,12 $ 0,00035504 172,92 $	201,6 $ un 0,12 $ 0,00054414 201,72 $	230,4 $ un 0,24 $ 0,00064956 230,64 $	259,2 $ un 0,54 $ 0,00033857 259,74 $
$K = 7E[R]$	172,8 $ un 0,14 $ 0,00012232 172,94 $	201,6 $ un 0,14 $ 0,00021188 201,74 $	230,4 $ un 0,28 $ 0,00030648 230,68 $	259,2 $ un 0,63 $ 0,00013099 259,83 $
$K = 8E[R]$	172,8 $ un 0,16 $ $4{,}613 \cdot 10^{-5}$ 172,96 $	201,6 $ un 0,16 $ 0,0001035 201,76 $	230,4 $ un 0,32 $ 0,00013474 230,72 $	259,2 $ un 0,72 $ $2{,}7377 \cdot 10^{-5}$ 259,92 $
$K = 9E[R]$	172,8 $ un 0,18 $ $2{,}6434 \cdot 10^{-5}$ 172,98 $	201,6 $ un 0,18 $ $4{,}442 \cdot 10^{-5}$ 201,78 $	230,4 $ un 0,36 $ $7{,}438 \cdot 10^{-5}$ 230,76 $	259,2 $ un 0,81 $ $3{,}4654 \cdot 10^{-6}$ 260,01 $
$K = 10E[R]$	172,8 $ un 0,2 $ $5{,}7014 \cdot 10^{-6}$ 173 $	201,6 $ un 0,2 $ $5{,}3304 \cdot 10^{-6}$ 201,8 $	230,4 $ un 0,4 $ $2{,}3365 \cdot 10^{-5}$ 230,8 $	259,2 $ un 0,9 $ – 260,1 $

16 p. 10. att. Dominējošās secības formēšana ar Belmana algoritmu $P/M/1/K$ *ON/OFF* trafika modelim ar Hersta parametru $H = 0,65$ un uzdotajām parametru izmaksām.

	ρ = 0,6	ρ = 0,7	ρ = 0,8	ρ = 0,9
$K = 1E[R]$	172,8 $ un 0,02 $ 0,093011 172,82 $	201,6 $ un 0,02 $ 0,09721 201,62 $	230,4 $ un 0,05 $ 0,076675 230,45 $	259,2 $ un 0,12 $ 0,052648 259,32 $
$K = 2E[R]$	172,8 $ un 0,04 $ 0,033973 172,84 $	201,6 $ un 0,04 $ 0,03754 201,64 $	230,4 $ un 0,1 $ 0,028141 230,5 $	259,2 $ un 0,24 $ 0,018409 259,44 $
$K = 3E[R]$	172,8 $ un 0,06 $ 0,013138 172,86 $	201,6 $ un 0,06 $ 0,015978 201,66 $	230,4 $ un 0,15 $ 0,01155 230,55 $	259,2 $ un 0,36 $ 0,0070573 259,56 $
$K = 4E[R]$	172,8 $ un 0,08 $ 0,0052122 172,88 $	201,6 $ un 0,08 $ 0,0070181 201,68 $	230,4 $ un 0,2 $ 0,0047703 230,6 $	259,2 $ un 0,48 $ 0,0029 259,68 $
$K = 5E[R]$	172,8 $ un 0,1 $ 0,0021165 172,9 $	201,6 $ un 0,1 $ 0,00293 201,7 $	230,4 $ un 0,25 $ 0,0020694 230,65 $	259,2 $ un 0,6 $ 0,0011056 259,8 $
$K = 6E[R]$	172,8 $ un 0,12 $ 0,00086468 172,92 $	201,6 $ un 0,12 $ 0,0013237 201,72 $	230,4 $ un 0,3 $ 0,0009296 230,7 $	259,2 $ un 0,72 $ 0,0004701 259,92 $
$K = 7E[R]$	172,8 $ un 0,14 $ 0,00032913 172,94 $	201,6 $ un 0,14 $ 0,00061108 201,74 $	230,4 $ un 0,35 $ 0,00032548 230,75 $	259,2 $ un 0,84 $ 0,00018467 260,04 $
$K = 8E[R]$	172,8 $ un 0,16 $ 0,00014018 172,96 $	201,6 $ un 0,16 $ 0,00032276 201,76 $	230,4 $ un 0,4 $ 0,00016216 230,8 $	259,2 $ un 0,96 $ $8{,}4253 \cdot 10^{-5}$ 260,16 $
$K = 9E[R]$	172,8 $ un 0,18 $ $5{,}3401 \cdot 10^{-5}$ 172,98 $	201,6 $ un 0,18 $ 0,00013467 201,78 $	230,4 $ un 0,45 $ $6{,}734 \cdot 10^{-5}$ 230,85 $	259,2 $ un 1,08 $ $3{,}1294 \cdot 10^{-5}$ 260,28 $
$K = 10E[R]$	172,8 $ un 0,2 $ $2{,}516 \cdot 10^{-5}$ 173 $	201,6 $ un 0,2 $ $8{,}5215 \cdot 10^{-5}$ 201,8 $	230,4 $ un 0,5 $ $1{,}3158 \cdot 10^{-5}$ 230,9 $	259,2 $ un 1,2 $ $1{,}5475 \cdot 10^{-5}$ 260,4 $

16 p. 11. att. Dominējošās secības formēšana ar Belmana algoritmu $P/M/1/K$ *ON/OFF* trafika modelim ar Hersta parametru $H = 0,7$ un uzdotajām parametru izmaksām.

	ρ = 0,6	ρ = 0,7	ρ = 0,8	ρ = 0,9
$K = 1E[R]$	172,8 $ un 0,02 $ 0,11635 172,82 $	201,6 $ un 0,03 $ 0,092217 201,63 $	230,4 $ un 0,07 $ 0,066087 230,47 $	259,2 $ un 0,17 $ 0,049549 259,37 $
$K = 2E[R]$	172,8 $ un 0,04 $ 0,049448 172,84 $	201,6 $ un 0,06 $ 0,03456 201,66 $	230,4 $ un 0,14 $ 0,022306 230,54 $	259,2 $ un 0,34 $ 0,015919 259,54 $
$K = 3E[R]$	172,8 $ un 0,06 $ 0,022715 172,86 $	201,6 $ un 0,09 $ 0,014461 201,69 $	230,4 $ un 0,21 $ 0,0087396 230,61 $	259,2 $ un 0,51 $ 0,0054333 259,71 $
$K = 4E[R]$	172,8 $ un 0,08 $ 0,010652 172,88 $	201,6 $ un 0,12 $ 0,0061643 201,72 $	230,4 $ un 0,28 $ 0,0033591 230,68 $	259,2 $ un 0,68 $ 0,0019889 259,88 $
$K = 5E[R]$	172,8 $ un 0,1 $ 0,0052346 172,9 $	201,6 $ un 0,15 $ 0,0027055 201,75 $	230,4 $ un 0,35 $ 0,0011765 230,75 $	259,2 $ un 0,85 $ 0,00074578 260,05 $
$K = 6E[R]$	172,8 $ un 0,12 $ 0,0025103 172,92 $	201,6 $ un 0,18 $ 0,0012365 201,78 $	230,4 $ un 0,42 $ 0,00046929 230,82 $	259,2 $ un 1,02 $ 0,00029267 260,22 $
$K = 7E[R]$	172,8 $ un 0,14 $ 0,0012764 172,94 $	201,6 $ un 0,21 $ 0,00053753 201,81 $	230,4 $ un 0,49 $ 0,00017488 230,89 $	259,2 $ un 1,19 $ $8{,}0901 \cdot 10^{-5}$ 260,39 $
$K = 8E[R]$	172,8 $ un 0,16 $ 0,00063238 172,96 $	201,6 $ un 0,24 $ 0,00025853 201,84 $	230,4 $ un 0,56 $ $9{,}4315 \cdot 10^{-5}$ 230,96 $	259,2 $ un 1,36 $ $2{,}0735 \cdot 10^{-5}$ 260,56 $
$K = 9E[R]$	172,8 $ un 0,18 $ 0,0003258 172,98 $	201,6 $ un 0,27 $ 0,00012709 201,87 $	230,4 $ un 0,63 $ $4{,}4676 \cdot 10^{-5}$ 231,03 $	259,2 $ un 1,53 $ $1{,}0198 \cdot 10^{-6}$ 260,73 $
$K = 10E[R]$	172,8 $ un 0,2 $ 0,00015632 173 $	201,6 $ un 0,3 $ $8{,}5743 \cdot 10^{-5}$ 201,9 $	230,4 $ un 0,7 $ $2{,}1765 \cdot 10^{-5}$ 231,1 $	259,2 $ un 1,7 $ – 260,9 $

16 p. 12. att. Dominējošās secības formēšana ar Belmana algoritmu $P/M/1/K$ *ON/OFF* trafika modelim ar Hersta parametru $H = 0,75$ un uzdotajām parametru izmaksām.

	ρ = 0,6	ρ = 0,7	ρ = 0,8	ρ = 0,9
$K = 1E[R]$	172,8 $ un 0,02 $ 0,10596 172,82 $	201,6 $ un 0,05 $ 0,08649 201,65 $	230,4 $ un 0,11 $ 0,063745 230,51 $	259,2 $ un 0,25 $ 0,055511 259,45 $
$K = 2E[R]$	172,8 $ un 0,04 $ 0,04197 172,84 $	201,6 $ un 0,1 $ 0,031876 201,7 $	230,4 $ un 0,22 $ 0,022164 230,62 $	259,2 $ un 0,5 $ 0,019552 259,7 $
$K = 3E[R]$	172,8 $ un 0,06 $ 0,017583 172,86 $	201,6 $ un 0,15 $ 0,012946 201,75 $	230,4 $ un 0,33 $ 0,0082961 230,73 $	259,2 $ un 0,75 $ 0,0074208 259,95 $
$K = 4E[R]$	172,8 $ un 0,08 $ 0,0079025 172,88 $	201,6 $ un 0,2 $ 0,0054083 201,8 $	230,4 $ un 0,44 $ 0,0034308 230,84 $	259,2 $ un 1 $ 0,0029457 260,2 $
$K = 5E[R]$	172,8 $ un 0,1 $ 0,0033717 172,9 $	201,6 $ un 0,25 $ 0,002356 201,85 $	230,4 $ un 0,55 $ 0,0011476 230,95 $	259,2 $ un 1,25 $ 0,0011434 260,45 $
$K = 6E[R]$	172,8 $ un 0,12 $ 0,0014697 172,92 $	201,6 $ un 0,3 $ 0,00096937 201,9 $	230,4 $ un 0,66 $ 0,00054788 231,06 $	259,2 $ un 1,5 $ 0,00054193 260,7 $
$K = 7E[R]$	172,8 $ un 0,14 $ 0,000706 172,94 $	201,6 $ un 0,35 $ 0,00038749 201,95 $	230,4 $ un 0,77 $ 0,00020252 231,17 $	259,2 $ un 1,75 $ 0,00018818 260,95 $
$K = 8E[R]$	172,8 $ un 0,16 $ 0,00033722 172,96 $	201,6 $ un 0,4 $ 0,00019651 202 $	230,4 $ un 0,88 $ $9{,}0256 \cdot 10^{-5}$ 231,28 $	259,2 $ un 2 $ $9{,}3424 \cdot 10^{-5}$ 261,2 $
$K = 9E[R]$	172,8 $ un 0,18 $ 0,0001484 172,98 $	201,6 $ un 0,45 $ $9{,}9957 \cdot 10^{-5}$ 202,05 $	230,4 $ un 0,99 $ $2{,}4988 \cdot 10^{-5}$ 231,39 $	259,2 $ un 2,25 $ $3{,}9564 \cdot 10^{-5}$ 261,45 $
$K = 10E[R]$	172,8 $ un 0,2 $ $6{,}1627 \cdot 10^{-5}$ 173 $	201,6 $ un 0,5 $ $4{,}1259 \cdot 10^{-5}$ 202,1 $	230,4 $ un 1,1 $ $4{,}4755 \cdot 10^{-6}$ 231,5 $	259,2 $ un 2,5 $ $1{,}7288 \cdot 10^{-5}$ 261,7 $

16 p. 13. att. Dominējošās secības formēšana ar Belmana algoritmu $P/M/1/K$ *ON/OFF* trafika modelim ar Hersta parametru $H = 0,8$ un uzdotajām parametru izmaksām.

	ρ = 0,6	ρ = 0,7	ρ = 0,8	ρ = 0,9
$K = 1E[R]$	172,8 $ un 0,05 $ 0,083656 172,85 $	201,6 $ un 0,09 $ 0,074588 201,69 $	230,4 $ un 0,22 $ 0,059431 230,62 $	259,2 $ un 0,45 $ 0,062187 259,65 $
$K = 2E[R]$	172,8 $ un 0,1 $ 0,028663 172,9 $	201,6 $ un 0,18 $ 0,026012 201,78 $	230,4 $ un 0,44 $ 0,019404 230,84 $	259,2 $ un 0,9 $ 0,02106 260,1 $
$K = 3E[R]$	172,8 $ un 0,15 $ 0,010884 172,95 $	201,6 $ un 0,27 $ 0,0098943 201,87 $	230,4 $ un 0,66 $ 0,0068809 231,06 $	259,2 $ un 1,35 $ 0,0076802 260,55 $
$K = 4E[R]$	172,8 $ un 0,2 $ 0,0039729 173 $	201,6 $ un 0,36 $ 0,0039592 201,96 $	230,4 $ un 0,88 $ 0,0028465 231,28 $	259,2 $ un 1,8 $ 0,0027948 261 $
$K = 5E[R]$	172,8 $ un 0,25 $ 0,001463 173,05 $	201,6 $ un 0,45 $ 0,0015056 202,05 $	230,4 $ un 1,1 $ 0,001015 231,5 $	259,2 $ un 2,25 $ 0,0010968 261,45 $
$K = 6E[R]$	172,8 $ un 0,3 $ 0,00062404 173,1 $	201,6 $ un 0,54 $ 0,00074606 202,14 $	230,4 $ un 1,32 $ 0,00040659 231,72 $	259,2 $ un 2,7 $ 0,00037507 261,9 $
$K = 7E[R]$	172,8 $ un 0,35 $ 0,00026643 173,15 $	201,6 $ un 0,63 $ 0,00031887 202,23 $	230,4 $ un 1,54 $ 0,00019936 231,94 $	259,2 $ un 3,15 $ 0,00010876 262,35 $
$K = 8E[R]$	172,8 $ un 0,4 $ 0,00011101 173,2 $	201,6 $ un 0,72 $ 0,0001462 202,32 $	230,4 $ un 1,76 $ $8{,}539 \cdot 10^{-5}$ 232,16 $	259,2 $ un 3,6 $ $1{,}6585 \cdot 10^{-5}$ 262,8 $
$K = 9E[R]$	172,8 $ un 0,45 $ $5{,}8578 \cdot 10^{-5}$ 173,25 $	201,6 $ un 0,81 $ $7{,}0045 \cdot 10^{-5}$ 202,41 $	230,4 $ un 1,98 $ $2{,}3938 \cdot 10^{-5}$ 232,38 $	259,2 $ un 4,05 $ – 263,25 $
$K = 10E[R]$	172,8 $ un 0,5 $ $1{,}8896 \cdot 10^{-5}$ 173,3 $	201,6 $ un 0,9 $ $2{,}4027 \cdot 10^{-5}$ 202,5 $	230,4 $ un 2,2 $ $6{,}7884 \cdot 10^{-6}$ 232,6 $	259,2 $ un 4,5 $ – 263,7 $

16 p. 14. att. Dominējošās secības formēšana ar Belmana algoritmu $P/M/1/K$ *ON/OFF* trafika modelim ar Hersta parametru $H = 0,85$ un uzdotajām parametru izmaksām.

	ρ = 0,6	ρ = 0,7	ρ = 0,8	ρ = 0,9
$K = 1E[R]$	172,8 $ un 0,12 $ 0,081923 172,92 $	201,6 $ un 0,29 $ 0,068424 201,89 $	230,4 $ un 0,47 $ 0,087383 230,87 $	259,2 $ un 0,98 $ 0,079248 260,18 $
$K = 2E[R]$	172,8 $ un 0,24 $ 0,029965 173,04 $	201,6 $ un 0,58 $ 0,022865 202,18 $	230,4 $ un 0,94 $ 0,034509 231,34 $	259,2 $ un 1,96 $ 0,026863 261,16 $
$K = 3E[R]$	172,8 $ un 0,36 $ 0,011608 173,16 $	201,6 $ un 0,87 $ 0,0082895 202,47 $	230,4 $ un 1,41 $ 0,014442 231,81 $	259,2 $ un 2,94 $ 0,010009 262,14 $
$K = 4E[R]$	172,8 $ un 0,48 $ 0,0046361 173,28 $	201,6 $ un 1,16 $ 0,0030326 202,76 $	230,4 $ un 1,88 $ 0,0062418 232,28 $	259,2 $ un 3,92 $ 0,0039842 263,12 $
$K = 5E[R]$	172,8 $ un 0,6 $ 0,0018431 173,4 $	201,6 $ un 1,45 $ 0,00093288 203,05 $	230,4 $ un 2,35 $ 0,0027184 232,75 $	259,2 $ un 4,9 $ 0,0016101 264,1 $
$K = 6E[R]$	172,8 $ un 0,72 $ 0,00064173 173,52 $	201,6 $ un 1,74 $ 0,00021659 203,34 $	230,4 $ un 2,82 $ 0,0012745 233,22 $	259,2 $ un 5,88 $ 0,00086076 265,08 $
$K = 7E[R]$	172,8 $ un 0,84 $ 0,00031463 173,64 $	201,6 $ un 2,03 $ $3{,}3634 \cdot 10^{-5}$ 203,63 $	230,4 $ un 3,29 $ 0,00056972 233,69 $	259,2 $ un 6,86 $ 0,00054466 266,06 $
$K = 8E[R]$	172,8 $ un 0,96 $ 0,00016033 173,76 $	201,6 $ un 2,32 $ $2{,}5872 \cdot 10^{-6}$ 203,92 $	230,4 $ un 3,76 $ 0,0002457 234,16 $	259,2 $ un 7,84 $ 0,00034401 267,04 $
$K = 9E[R]$	172,8 $ un 1,08 $ $7{,}6509 \cdot 10^{-5}$ 173,88 $	201,6 $ un 2,61 $ – 204,21 $	230,4 $ un 4,23 $ $8{,}1249 \cdot 10^{-5}$ 234,63 $	259,2 $ un 8,82 $ 0,00023613 268,02 $
$K = 10E[R]$	172,8 $ un 1,2 $ $3{,}4816 \cdot 10^{-5}$ 174 $	201,6 $ un 2,9 $ – 204,5 $	230,4 $ un 4,7 $ $1{,}7294 \cdot 10^{-5}$ 235,1 $	259,2 $ un 9,8 $ 0,0001265 269 $

16 p. 15. att. Dominējošās secības formēšana ar Belmana algoritmu $P/M/1/K$ *ON/OFF* trafika modelim ar Hersta parametru $H = 0,9$ un uzdotajām parametru izmaksām.

	ρ = 0,6	ρ = 0,7	ρ = 0,8	ρ = 0,9
$K = 1E[R]$	172,8 $ un 0,77 $ 0,10159 173,57 $	201,6 $ un 1,52 $ 0,098467 203,12 $	230,4 $ un 1,78 $ 0,13888 232,18 $	259,2 $ un 3,68 $ 0,095958 262,88 $
$K = 2E[R]$	172,8 $ un 1,54 $ 0,038314 174,34 $	201,6 $ un 3,04 $ 0,032972 204,64 $	230,4 $ un 3,56 $ 0,055809 233,96 $	259,2 $ un 7,36 $ 0,028133 266,56 $
$K = 3E[R]$	172,8 $ un 2,31 $ 0,015023 175,11 $	201,6 $ un 4,56 $ 0,01106 206,16 $	230,4 $ un 5,34 $ 0,023651 235,74 $	259,2 $ un 11,04 $ 0,0083319 270,24 $
$K = 4E[R]$	172,8 $ un 3,08 $ 0,0061041 175,88 $	201,6 $ un 6,08 $ 0,0033442 207,68 $	230,4 $ un 7,12 $ 0,010952 237,52 $	259,2 $ un 14,72 $ 0,0023453 273,92 $
$K = 5E[R]$	172,8 $ un 3,85 $ 0,0025174 176,65 $	201,6 $ un 7,6 $ 0,0011366 209,2 $	230,4 $ un 8,9 $ 0,004709 239,3 $	259,2 $ un 18,4 $ 0,00042468 277,6 $
$K = 6E[R]$	172,8 $ un 4,62 $ 0,00090646 177,42 $	201,6 $ un 9,12 $ 0,00033958 210,72 $	230,4 $ un 10,68 $ 0,0019039 241,08 $	259,2 $ un 22,08 $ – 281,28 $
$K = 7E[R]$	172,8 $ un 5,39 $ 0,0002878 178,19 $	201,6 $ un 10,64 $ $3{,}0514 \cdot 10^{-5}$ 212,24 $	230,4 $ un 12,46 $ 0,00071161 242,86 $	259,2 $ un 25,76 $ – 284,96 $
$K = 8E[R]$	172,8 $ un 6,16 $ 0,00011747 178,96 $	201,6 $ un 12,16 $ – 213,76 $	230,4 $ un 14,24 $ 0,00033683 244,64 $	259,2 $ un 29,44 $ – 288,64 $
$K = 9E[R]$	172,8 $ un 6,93 $ $5{,}2861 \cdot 10^{-5}$ 179,73 $	201,6 $ un 13,68 $ – 215,28 $	230,4 $ un 16,02 $ 0,00018456 246,42 $	259,2 $ un 33,12 $ – 292,32 $
$K = 10E[R]$	172,8 $ un 7,7 $ $1{,}9578 \cdot 10^{-6}$ 180,5 $	201,6 $ un 15,2 $ – 216,8 $	230,4 $ un 17,8 $ $9{,}4372 \cdot 10^{-5}$ 248,2 $	259,2 $ un 36,8 $ – 296 $

16 p. 16. att. Dominējošās secības formēšana ar Belmana algoritmu $P/M/1/K$ *ON/OFF* trafika modelim ar Hersta parametru $H = 0,95$ un uzdotajām parametru izmaksām.

	ρ = 0,6	ρ = 0,7	ρ = 0,8	ρ = 0,9
$K = 1E[R]$	172,8 $ un 0,02 $ 0,14156 172,82 $	201,6 $ un 0,02 $ 0,12351 201,62 $	230,4 $ un 0,05 $ 0,076005 230,45 $	259,2 $ un 0,1 $ 0,044277 259,3 $
$K = 2E[R]$	172,8 $ un 0,04 $ 0,057052 172,84 $	201,6 $ un 0,04 $ 0,046334 201,64 $	230,4 $ un 0,1 $ 0,022631 230,5 $	259,2 $ un 0,2 $ 0,012325 259,4 $
$K = 3E[R]$	172,8 $ un 0,06 $ 0,024467 172,86 $	201,6 $ un 0,06 $ 0,018841 201,66 $	230,4 $ un 0,15 $ 0,0077287 230,55 $	259,2 $ un 0,3 $ 0,0040527 259,5 $
$K = 4E[R]$	172,8 $ un 0,08 $ 0,010879 172,88 $	201,6 $ un 0,08 $ 0,0080927 201,68 $	230,4 $ un 0,2 $ 0,0026012 230,6 $	259,2 $ un 0,4 $ 0,0013999 259,6 $
$K = 5E[R]$	172,8 $ un 0,1 $ 0,0048407 172,9 $	201,6 $ un 0,1 $ 0,0035579 201,7 $	230,4 $ un 0,25 $ 0,00097532 230,65 $	259,2 $ un 0,5 $ 0,0004676 259,7 $
$K = 6E[R]$	172,8 $ un 0,12 $ 0,0021375 172,92 $	201,6 $ un 0,12 $ 0,0015057 201,72 $	230,4 $ un 0,3 $ 0,00040204 230,7 $	259,2 $ un 0,6 $ 0,00017951 259,8 $
$K = 7E[R]$	172,8 $ un 0,14 $ 0,0010512 172,94 $	201,6 $ un 0,14 $ 0,00070145 201,74 $	230,4 $ un 0,35 $ 0,00014426 230,75 $	259,2 $ un 0,7 $ $4{,}8181 \cdot 10^{-5}$ 259,9 $
$K = 8E[R]$	172,8 $ un 0,16 $ 0,00049202 172,96 $	201,6 $ un 0,16 $ 0,00032189 201,76 $	230,4 $ un 0,4 $ $5{,}5329 \cdot 10^{-5}$ 230,8 $	259,2 $ un 0,8 $ $1{,}0436 \cdot 10^{-5}$ 260 $
$K = 9E[R]$	172,8 $ un 0,18 $ 0,00022902 172,98 $	201,6 $ un 0,18 $ 0,00014888 201,78 $	230,4 $ un 0,45 $ $1{,}3489 \cdot 10^{-5}$ 230,85 $	259,2 $ un 0,9 $ $1{,}4432 \cdot 10^{-6}$ 260,1 $
$K = 10E[R]$	172,8 $ un 0,2 $ $9{,}827 \cdot 10^{-5}$ 173 $	201,6 $ un 0,2 $ $6{,}0382 \cdot 10^{-5}$ 201,8 $	230,4 $ un 0,5 $ $6{,}6195 \cdot 10^{-6}$ 230,9 $	259,2 $ un 1 $ – 260,2 $

16 p. 17. att. Dominējošās secības formēšana ar Belmana algoritmu $G/M/1/K$ ar Veibula sadalījumu trafika modelim ar Hersta parametru $H = 0,6$ un uzdotajām parametru izmaksām.

	ρ = 0,6	ρ = 0,7	ρ = 0,8	ρ = 0,9
$K = 1E[R]$	172,8 $ un 0,02 $ 0,11602 172,82 $	201,6 $ un 0,03 $ 0,11311 201,63 $	230,4 $ un 0,05 $ 0,080683 230,45 $	259,2 $ un 0,12 $ 0,043311 259,32 $
$K = 2E[R]$	172,8 $ un 0,04 $ 0,038332 172,84 $	201,6 $ un 0,06 $ 0,038631 201,66 $	230,4 $ un 0,1 $ 0,024434 230,5 $	259,2 $ un 0,24 $ 0,011943 259,44 $
$K = 3E[R]$	172,8 $ un 0,06 $ 0,013593 172,86 $	201,6 $ un 0,09 $ 0,014578 201,69 $	230,4 $ un 0,15 $ 0,0085947 230,55 $	259,2 $ un 0,36 $ 0,0038347 259,56 $
$K = 4E[R]$	172,8 $ un 0,08 $ 0,0051239 172,88 $	201,6 $ un 0,12 $ 0,0057647 201,72 $	230,4 $ un 0,2 $ 0,0032033 230,6 $	259,2 $ un 0,48 $ 0,0013568 259,68 $
$K = 5E[R]$	172,8 $ un 0,1 $ 0,0018847 172,9 $	201,6 $ un 0,15 $ 0,0023214 201,75 $	230,4 $ un 0,25 $ 0,0011385 230,65 $	259,2 $ un 0,6 $ 0,00042611 259,8 $
$K = 6E[R]$	172,8 $ un 0,12 $ 0,00069628 172,92 $	201,6 $ un 0,18 $ 0,00096194 201,78 $	230,4 $ un 0,3 $ 0,00044043 230,7 $	259,2 $ un 0,72 $ 0,00016006 259,92 $
$K = 7E[R]$	172,8 $ un 0,14 $ 0,00027994 172,94 $	201,6 $ un 0,21 $ 0,00040561 201,81 $	230,4 $ un 0,35 $ 0,00015684 230,75 $	259,2 $ un 0,84 $ $3{,}2522 \cdot 10^{-5}$ 260,04 $
$K = 8E[R]$	172,8 $ un 0,16 $ 0,00010925 172,96 $	201,6 $ un 0,24 $ 0,00016342 201,84 $	230,4 $ un 0,4 $ $7{,}455 \cdot 10^{-5}$ 230,8 $	259,2 $ un 0,96 $ $7{,}6587 \cdot 10^{-6}$ 260,16 $
$K = 9E[R]$	172,8 $ un 0,18 $ $4{,}6296 \cdot 10^{-5}$ 172,98 $	201,6 $ un 0,27 $ $6{,}3226 \cdot 10^{-5}$ 201,87 $	230,4 $ un 0,45 $ $1{,}998 \cdot 10^{-5}$ 230,85 $	259,2 $ un 1,08 $ 2,2199·10-07 260,28 $
$K = 10E[R]$	172,8 $ un 0,2 $ $2{,}1483 \cdot 10^{-5}$ 173 $	201,6 $ un 0,3 $ $3{,}6394 \cdot 10^{-5}$ 201,9 $	230,4 $ un 0,5 $ $5{,}994 \cdot 10^{-6}$ 230,9 $	259,2 $ un 1,2 $ – 260,4 $

16 p. 18. att. Dominējošās secības formēšana ar Belmana algoritmu $G/M/1/K$ ar Veibula sadalījumu trafika modelim ar Hersta parametru $H = 0,65$ un uzdotajām parametru izmaksām.

	ρ = 0,6	ρ = 0,7	ρ = 0,8	ρ = 0,9
$K = 1E[R]$	172,8 $ un 0,03 $ 0,11378 172,83 $	201,6 $ un 0,04 $ 0,11883 201,64 $	230,4 $ un 0,07 $ 0,081109 230,47 $	259,2 $ un 0,16 $ 0,042516 259,36 $
$K = 2E[R]$	172,8 $ un 0,06 $ 0,035564 172,86 $	201,6 $ un 0,08 $ 0,041027 201,68 $	230,4 $ un 0,14 $ 0,024732 230,54 $	259,2 $ un 0,32 $ 0,011318 259,52 $
$K = 3E[R]$	172,8 $ un 0,09 $ 0,011972 172,89 $	201,6 $ un 0,12 $ 0,015903 201,72 $	230,4 $ un 0,21 $ 0,0085819 230,61 $	259,2 $ un 0,48 $ 0,003543 259,68 $
$K = 4E[R]$	172,8 $ un 0,12 $ 0,0041621 172,92 $	201,6 $ un 0,16 $ 0,0064112 201,76 $	230,4 $ un 0,28 $ 0,0030429 230,68 $	259,2 $ un 0,64 $ 0,0012639 259,84 $
$K = 5E[R]$	172,8 $ un 0,15 $ 0,0015065 172,95 $	201,6 $ un 0,2 $ 0,0025009 201,8 $	230,4 $ un 0,35 $ 0,001138 230,75 $	259,2 $ un 0,8 $ 0,00037607 260 $
$K = 6E[R]$	172,8 $ un 0,18 $ 0,00053046 172,98 $	201,6 $ un 0,24 $ 0,0010981 201,84 $	230,4 $ un 0,42 $ 0,00043933 230,82 $	259,2 $ un 0,96 $ 0,00014603 260,16 $
$K = 7E[R]$	172,8 $ un 0,21 $ 0,00021228 173,01 $	201,6 $ un 0,28 $ 0,00045561 201,88 $	230,4 $ un 0,49 $ 0,00015506 230,89 $	259,2 $ un 1,12 $ $3{,}1737 \cdot 10^{-5}$ 260,32 $
$K = 8E[R]$	172,8 $ un 0,24 $ $7{,}176 \cdot 10^{-5}$ 173,04 $	201,6 $ un 0,32 $ 0,00020904 201,92 $	230,4 $ un 0,56 $ $6{,}8665 \cdot 10^{-5}$ 230,96 $	259,2 $ un 1,28 $ $5{,}7703 \cdot 10^{-6}$ 260,48 $
$K = 9E[R]$	172,8 $ un 0,27 $ $4{,}4455 \cdot 10^{-5}$ 173,07 $	201,6 $ un 0,36 $ $9{,}1179 \cdot 10^{-5}$ 201,96 $	230,4 $ un 0,63 $ $2{,}0225 \cdot 10^{-5}$ 231,03 $	259,2 $ un 1,44 $ $1{,}1097 \cdot 10^{-6}$ 260,64 $
$K = 10E[R]$	172,8 $ un 0,3 $ $1{,}4818 \cdot 10^{-5}$ 173,1 $	201,6 $ un 0,4 $ $3{,}7955 \cdot 10^{-5}$ 202 $	230,4 $ un 0,7 $ $4{,}9938 \cdot 10^{-6}$ 231,1 $	259,2 $ un 1,6 $ – 260,8 $

16 p. 19. att. Dominējošās secības formēšana ar Belmana algoritmu $G/M/1/K$ ar Veibula sadalījumu trafika modelim ar Hersta parametru $H = 0,7$ un uzdotajām parametru izmaksām.

	ρ = 0,6	ρ = 0,7	ρ = 0,8	ρ = 0,9
$K = 1E[R]$	172,8 $ un 0,04 $ 0,13365 172,84 $	201,6 $ un 0,06 $ 0,10293 201,66 $	230,4 $ un 0,1 $ 0,07965 230,5 $	259,2 $ un 0,23 $ 0,043106 259,43 $
$K = 2E[R]$	172,8 $ un 0,08 $ 0,045611 172,88 $	201,6 $ un 0,12 $ 0,03162 201,72 $	230,4 $ un 0,2 $ 0,023819 230,6 $	259,2 $ un 0,46 $ 0,011422 259,66 $
$K = 3E[R]$	172,8 $ un 0,12 $ 0,017022 172,92 $	201,6 $ un 0,18 $ 0,010866 201,78 $	230,4 $ un 0,3 $ 0,0081871 230,7 $	259,2 $ un 0,69 $ 0,0036983 259,89 $
$K = 4E[R]$	172,8 $ un 0,16 $ 0,0065311 172,96 $	201,6 $ un 0,24 $ 0,0038711 201,84 $	230,4 $ un 0,4 $ 0,002803 230,8 $	259,2 $ un 0,92 $ 0,0013299 260,12 $
$K = 5E[R]$	172,8 $ un 0,2 $ 0,0025281 173 $	201,6 $ un 0,3 $ 0,001386 201,9 $	230,4 $ un 0,5 $ 0,0010353 230,9 $	259,2 $ un 1,15 $ 0,00048763 260,35 $
$K = 6E[R]$	172,8 $ un 0,24 $ 0,00098601 173,04 $	201,6 $ un 0,36 $ 0,0005161 201,96 $	230,4 $ un 0,6 $ 0,00034259 231 $	259,2 $ un 1,38 $ 0,00018857 260,58 $
$K = 7E[R]$	172,8 $ un 0,28 $ 0,00045239 173,08 $	201,6 $ un 0,42 $ 0,00020912 202,02 $	230,4 $ un 0,7 $ 0,00015151 231,1 $	259,2 $ un 1,61 $ $5{,}6239 \cdot 10^{-5}$ 260,81 $
$K = 8E[R]$	172,8 $ un 0,32 $ 0,00015829 173,12 $	201,6 $ un 0,48 $ $8{,}8157 \cdot 10^{-5}$ 202,08 $	230,4 $ un 0,8 $ $4{,}8674 \cdot 10^{-5}$ 231,2 $	259,2 $ un 1,84 $ $1{,}3977 \cdot 10^{-5}$ 261,04 $
$K = 9E[R]$	172,8 $ un 0,36 $ $6{,}2083 \cdot 10^{-5}$ 173,16 $	201,6 $ un 0,54 $ $2{,}4393 \cdot 10^{-5}$ 202,14 $	230,4 $ un 0,9 $ $1{,}3229 \cdot 10^{-5}$ 231,3 $	259,2 $ un 2,07 $ $3{,}1059 \cdot 10^{-6}$ 261,27 $
$K = 10E[R]$	172,8 $ un 0,4 $ $4{,}1444 \cdot 10^{-5}$ 173,2 $	201,6 $ un 0,6 $ $8{,}2736 \cdot 10^{-6}$ 202,2 $	230,4 $ un 1 $ $4{,}2434 \cdot 10^{-6}$ 231,4 $	259,2 $ un 2,3 $ – 261,5 $

16 p. 20. att. Dominējošās secības formēšana ar Belmana algoritmu $G/M/1/K$ ar Veibula sadalījumu trafika modelim ar Hersta parametru $H = 0,75$ un uzdotajām parametru izmaksām.

	ρ = 0,6	ρ = 0,7	ρ = 0,8	ρ = 0,9
$K = 1E[R]$	172,8 $ un 0,07 $ 0,12042 172,87 $	201,6 $ un 0,1 $ 0,10596 201,7 $	230,4 $ un 0,18 $ 0,074531 230,58 $	259,2 $ un 0,44 $ 0,036778 259,64 $
$K = 2E[R]$	172,8 $ un 0,14 $ 0,037121 172,94 $	201,6 $ un 0,2 $ 0,033252 201,8 $	230,4 $ un 0,36 $ 0,021619 230,76 $	259,2 $ un 0,88 $ 0,0096181 260,08 $
$K = 3E[R]$	172,8 $ un 0,21 $ 0,012482 173,01 $	201,6 $ un 0,3 $ 0,011573 201,9 $	230,4 $ un 0,54 $ 0,0069624 230,94 $	259,2 $ un 1,32 $ 0,0029631 260,52 $
$K = 4E[R]$	172,8 $ un 0,28 $ 0,0044441 173,08 $	201,6 $ un 0,4 $ 0,0041031 202 $	230,4 $ un 0,72 $ 0,0022977 231,12 $	259,2 $ un 1,76 $ 0,00096075 260,96 $
$K = 5E[R]$	172,8 $ un 0,35 $ 0,0015419 173,15 $	201,6 $ un 0,5 $ 0,0014872 202,1 $	230,4 $ un 0,9 $ 0,00082721 231,3 $	259,2 $ un 2,2 $ 0,00035676 261,4 $
$K = 6E[R]$	172,8 $ un 0,42 $ 0,00054285 173,22 $	201,6 $ un 0,6 $ 0,0005667 202,2 $	230,4 $ un 1,08 $ 0,00029553 231,48 $	259,2 $ un 2,64 $ 0,00013847 261,84 $
$K = 7E[R]$	172,8 $ un 0,49 $ 0,00022243 173,29 $	201,6 $ un 0,7 $ 0,00021932 202,3 $	230,4 $ un 1,26 $ 0,00011801 231,66 $	259,2 $ un 3,08 $ $4{,}7339 \cdot 10^{-5}$ 262,28 $
$K = 8E[R]$	172,8 $ un 0,56 $ $8{,}5512 \cdot 10^{-5}$ 173,36 $	201,6 $ un 0,8 $ $8{,}5846 \cdot 10^{-5}$ 202,4 $	230,4 $ un 1,44 $ $3{,}0314 \cdot 10^{-5}$ 231,84 $	259,2 $ un 3,52 $ $2{,}2173 \cdot 10^{-5}$ 262,72 $
$K = 9E[R]$	172,8 $ un 0,63 $ $3{,}0112 \cdot 10^{-5}$ 173,43 $	201,6 $ un 0,9 $ $2{,}3244 \cdot 10^{-5}$ 202,5 $	230,4 $ un 1,62 $ $9{,}2314 \cdot 10^{-6}$ 232,02 $	259,2 $ un 3,96 $ $9{,}8669 \cdot 10^{-6}$ 263,16 $
$K = 10E[R]$	172,8 $ un 0,7 $ $1{,}0814 \cdot 10^{-5}$ 173,5 $	201,6 $ un 1 $ $1{,}0267 \cdot 10^{-5}$ 202,6 $	230,4 $ un 1,8 $ $1{,}497 \cdot 10^{-6}$ 232,2 $	259,2 $ un 4,4 $ $2{,}5499 \cdot 10^{-6}$ 263,6 $

16 p. 21. att. Dominējošās secības formēšana ar Belmana algoritmu $G/M/1/K$ ar Veibula sadalījumu trafika modelim ar Hersta parametru $H = 0,8$ un uzdotajām parametru izmaksām.

	ρ = 0,6	ρ = 0,7	ρ = 0,8	ρ = 0,9
$K = 1E[R]$	172,8 $ un 0,15 $ 0,11894 172,95 $	201,6 $ un 0,25 $ 0,090883 201,85 $	230,4 $ un 0,46 $ 0,065826 230,86 $	259,2 $ un 1,41 $ 0,025319 260,61 $
$K = 2E[R]$	172,8 $ un 0,3 $ 0,036852 173,1 $	201,6 $ un 0,5 $ 0,02619 202,1 $	230,4 $ un 0,92 $ 0,018046 231,32 $	259,2 $ un 2,82 $ 0,0055137 262,02 $
$K = 3E[R]$	172,8 $ un 0,45 $ 0,012439 173,25 $	201,6 $ un 0,75 $ 0,0079628 202,35 $	230,4 $ un 1,38 $ 0,0056154 231,78 $	259,2 $ un 4,23 $ 0,0013223 263,43 $
$K = 4E[R]$	172,8 $ un 0,6 $ 0,0042578 173,4 $	201,6 $ un 1 $ 0,0027368 202,6 $	230,4 $ un 1,84 $ 0,0019849 232,24 $	259,2 $ un 5,64 $ 0,00028177 264,84 $
$K = 5E[R]$	172,8 $ un 0,75 $ 0,0015041 173,55 $	201,6 $ un 1,25 $ 0,00094998 202,85 $	230,4 $ un 2,3 $ 0,00062032 232,7 $	259,2 $ un 7,05 $ $9{,}4735 \cdot 10^{-5}$ 266,25 $
$K = 6E[R]$	172,8 $ un 0,9 $ 0,00058884 173,7 $	201,6 $ un 1,5 $ 0,00028467 203,1 $	230,4 $ un 2,76 $ 0,00024793 233,16 $	259,2 $ un 8,46 $ $2{,}77 \cdot 10^{-5}$ 267,66 $
$K = 7E[R]$	172,8 $ un 1,05 $ 0,00020521 173,85 $	201,6 $ un 1,75 $ 0,00010825 203,35 $	230,4 $ un 3,22 $ $8{,}7049 \cdot 10^{-5}$ 233,62 $	259,2 $ un 9,87 $ – 269,07 $
$K = 8E[R]$	172,8 $ un 1,2 $ $8{,}0319 \cdot 10^{-5}$ 174 $	201,6 $ un 2 $ $3{,}7651 \cdot 10^{-5}$ 203,6 $	230,4 $ un 3,68 $ $3{,}9534 \cdot 10^{-5}$ 234,08 $	259,2 $ un 11,28 $ – 270,48 $
$K = 9E[R]$	172,8 $ un 1,35 $ $2{,}1784 \cdot 10^{-5}$ 174,15 $	201,6 $ un 2,25 $ $4{,}1359 \cdot 10^{-6}$ 203,85 $	230,4 $ un 4,14 $ $2{,}1575 \cdot 10^{-5}$ 234,54 $	259,2 $ un 12,69 $ – 271,89 $
$K = 10E[R]$	172,8 $ un 1,5 $ $3{,}4921 \cdot 10^{-6}$ 174,3 $	201,6 $ un 2,5 $ – 204,1 $	230,4 $ un 4,6 $ $1{,}4217 \cdot 10^{-5}$ 235 $	259,2 $ un 14,1 $ – 273,3 $

16 p. 22. att. Dominējošās secības formēšana ar Belmana algoritmu $G/M/1/K$ ar Veibula sadalījumu trafika modelim ar Hersta parametru $H = 0,85$ un uzdotajām parametru izmaksām.

	ρ = 0,6	ρ = 0,7	ρ = 0,8	ρ = 0,9
$K = 1E[R]$	172,8 $ un 0,85 $ 0,095619 173,65 $	201,6 $ un 1,54 $ 0,070228 203,14 $	230,4 $ un 3,8 $ 0,034457 234,2 $	259,2 $ un 13,75 $ 0,010507 272,95 $
$K = 2E[R]$	172,8 $ un 1,7 $ 0,026958 174,5 $	201,6 $ un 3,08 $ 0,018221 204,68 $	230,4 $ un 7,6 $ 0,00606 238 $	259,2 $ un 27,5 $ 0,00059239 286,7 $
$K = 3E[R]$	172,8 $ un 2,55 $ 0,0085909 175,35 $	201,6 $ un 4,62 $ 0,004963 206,22 $	230,4 $ un 11,4 $ 0,001065 241,8 $	259,2 $ un 41,25 $ – 300,45 $
$K = 4E[R]$	172,8 $ un 3,4 $ 0,0029734 176,2 $	201,6 $ un 6,16 $ 0,0013732 207,76 $	230,4 $ un 15,2 $ 0,00026739 245,6 $	259,2 $ un 55 $ – 314,2 $
$K = 5E[R]$	172,8 $ un 4,25 $ 0,00097972 177,05 $	201,6 $ un 7,7 $ 0,00042621 209,3 $	230,4 $ un 19 $ $1{,}5662 \cdot 10^{-5}$ 249,4 $	259,2 $ un 68,75 $ – 327,95 $
$K = 6E[R]$	172,8 $ un 5,1 $ 0,00032863 177,9 $	201,6 $ un 9,24 $ 0,00018635 210,84 $	230,4 $ un 22,8 $ – 253,2 $	259,2 $ un 82,5 $ – 341,7 $
$K = 7E[R]$	172,8 $ un 5,95 $ 0,00014428 178,75 $	201,6 $ un 10,78 $ $8{,}3062 \cdot 10^{-5}$ 212,38 $	230,4 $ un 26,6 $ – 257 $	259,2 $ un 96,25 $ – 355,45 $
$K = 8E[R]$	172,8 $ un 6,8 $ $7{,}7483 \cdot 10^{-5}$ 179,6 $	201,6 $ un 12,32 $ $4{,}3898 \cdot 10^{-5}$ 213,92 $	230,4 $ un 30,4 $ – 260,8 $	259,2 $ un 110 $ – 369,2 $
$K = 9E[R]$	172,8 $ un 7,65 $ $5{,}4271 \cdot 10^{-5}$ 180,45 $	201,6 $ un 13,86 $ $1{,}5206 \cdot 10^{-5}$ 215,46 $	230,4 $ un 34,2 $ – 264,6 $	259,2 $ un 123,75 $ – 382,95 $
$K = 10E[R]$	172,8 $ un 8,5 $ $3{,}5903 \cdot 10^{-5}$ 181,3 $	201,6 $ un 15,4 $ – 217 $	230,4 $ un 38 $ – 268,4 $	259,2 $ un 137,5 $ – 396,7 $

16 p. 23. att. Dominējošās secības formēšana ar Belmana algoritmu $G/M/1/K$ ar Veibula sadalījumu trafika modelim ar Hersta parametru $H = 0,9$ un uzdotajām parametru izmaksām.

	ρ = 0,6	ρ = 0,7	ρ = 0,8	ρ = 0,9
$K = 1E[R]$	172,8 $ un 72,95 $ 0,12749 245,75 $	201,6 $ un 98,39 $ 0,15294 299,99 $	230,4 $ un 117,66 $ 0,1802 348,06 $	259,2 $ un 154,18 $ 0,20066 413,38 $
$K = 2E[R]$	172,8 $ un 145,9 $ 0,052088 318,7 $	201,6 $ un 196,78 $ 0,074862 398,38 $	230,4 $ un 235,32 $ 0,11057 465,72 $	259,2 $ un 308,36 $ 0,13361 567,56 $
$K = 3E[R]$	172,8 $ un 218,85 $ 0,022645 391,65 $	201,6 $ un 295,17 $ 0,045262 496,77 $	230,4 $ un 352,98 $ 0,07593 583,38 $	259,2 $ un 462,54 $ 0,091323 721,74 $
$K = 4E[R]$	172,8 $ un 291,8 $ 0,0096068 464,6 $	201,6 $ un 393,56 $ 0,026128 595,16 $	230,4 $ un 470,64 $ 0,049506 701,04 $	259,2 $ un 616,72 $ 0,060214 875,92 $
$K = 5E[R]$	172,8 $ un 364,75 $ 0,0033685 537,55 $	201,6 $ un 491,95 $ 0,015049 693,55 $	230,4 $ un 588,3 $ 0,033776 818,7 $	259,2 $ un 770,9 $ 0,042613 1030,1 $
$K = 6E[R]$	172,8 $ un 437,7 $ 0,0014459 610,5 $	201,6 $ un 590,34 $ 0,010032 791,94 $	230,4 $ un 705,96 $ 0,024606 936,36 $	259,2 $ un 925,08 $ 0,032423 1184,28 $
$K = 7E[R]$	172,8 $ un 510,65 $ – 683,45 $	201,6 $ un 688,73 $ 0,0057316 890,33 $	230,4 $ un 823,62 $ 0,017889 1054,02 $	259,2 $ un 1079,26 $ 0,022407 1338,46 $
$K = 8E[R]$	172,8 $ un 583,6 $ – 756,4 $	201,6 $ un 787,12 $ 0,0022451 988,72 $	230,4 $ un 941,28 $ 0,011918 1171,68 $	259,2 $ un 1233,44 $ 0,014745 1492,64 $
$K = 9E[R]$	172,8 $ un 656,55 $ – 829,35 $	201,6 $ un 885,51 $ 9,5037·10-05 1087,11 $	230,4 $ un 1058,94 $ 0,0074957 1289,34 $	259,2 $ un 1387,62 $ 0,0090253 1646,82 $
$K = 10E[R]$	172,8 $ un 729,5 $ – 902,3 $	201,6 $ un 983,9 $ – 1185,5 $	230,4 $ un 1176,6 $ 0,0051128 1407 $	259,2 $ un 1541,8 $ 0,0061711 1801 $

16 p. 24. att. Dominējošās secības formēšana ar Belmana algoritmu $G/M/1/K$ ar Veibula sadalījumu trafika modelim ar Hersta parametru $H = 0,95$ un uzdotajām parametru izmaksām.