ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

Fakulta dopravní

Ústav letecké dopravy

Optimalizace algoritmů pro rekurentní kvantifikační analýzu

Optimization of Algorithms for Recurrent Quantification Analysis

Diplomová práca

Študijny program: Technika a technologie v dopravě a spojích

Študijny obor: Provoz a řízení letecké dopravy

Vedúci práce: Ing. Lenka Hanáková

doc. Ing. Bc. Vladimír Socha, Ph.D.

Bc. Timotej Gavura

Praha 2021

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
Fakulta dopravní
děkan
Konviktská 20, 110 00 Praha 1

K621 ..... Ústav letecké dopravy

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení studenta (včetně titulů):

Bc. Timotej Gavura

Kód studijního programu a studijní obor studenta:

N 3710 – PL – Provoz a řízení letecké dopravy

Název tématu (česky): **Optimalizace algoritmů pro rekurentní kvantifikační analýzu**

Název tématu (anglicky): Optimization of Algorithms for Recurrent Quantification Analysis

Zásady pro vypracování

Při zpracování diplomové práce se řiďte následujícími pokyny:

- Cílem práce je zvýšit výpočetní výkon pro vykonání rekurentní kvantifikační analýzy (RQA). V současnosti je tento typ analýzy využíván např. při vibrodiagnostice nebo při zpracování medicínských signálů. Největší limitací této metody je její vysoká výpočetní náročnost, což má ambici řešit právě předkládaná práce, a to tak, aby bylo možné zpracovávat relativně dlouhé signály.
- Vypracujte analýzu současného stavu v oblasti využívání RQA a základních výpočetních principů RQA.
- Navrhněte desktopovou aplikaci pro vykonání RQA.
- Optimalizujte výpočetní výkonnost pro signály (časové řady) delší než 100000 vzorků.
- Řešení validujte.
- Výsledky diskutujte a stanovte závěry práce.

Rozsah grafických prací: dle pokynů vedoucího diplomové práce

Rozsah průvodní zprávy: minimálně 55 stran textu (včetně obrázků, grafů a tabulek, které jsou součástí průvodní zprávy)

Seznam odborné literatury: Webber Jr, C. L., & Zbilut, J. P. (2005). Recurrence quantification analysis of nonlinear dynamical systems. Tutorials in contemporary nonlinear methods for the behavioral sciences, 94(2005), 26-94.

Webber, C. L., & Marwan, N. (2015). Recurrence quantification analysis. Theory and Best Practices.

Vedoucí diplomové práce: **Ing. Lenka Hanáková**

doc. Ing. Bc. Vladimír Socha, Ph.D.

Datum zadání diplomové práce: **17. července 2020**
(datum prvního zadání této práce, které musí být nejpozději 10 měsíců před datem prvního předpokládaného odevzdání této práce vyplývajícího ze standardní doby studia)

Datum odevzdání diplomové práce: **17. května 2021**
a) datum prvního předpokládaného odevzdání práce vyplývající ze standardní doby studia a z doporučeného časového plánu studia
b) v případě odkladu odevzdání práce následující datum odevzdání práce vyplývající z doporučeného časového plánu studia

L. S.

doc. Ing. Jakub Kraus, Ph.D.
vedoucí
Ústavu letecké dopravy

doc. Ing. Pavel Hrubeš, Ph.D.
děkan fakulty

Potvrzuji převzetí zadání diplomové práce.

Bc. Timotej Gavura
jméno a podpis studenta

V Praze dne 17. července 2020

Abstrakt

Dostupné aplikácie na výpočet rekurentnej kvantifikačnej analýzy sú limitované dĺžkou signálu, ktoré sú schopné spracovať, neponúkajú možnosť dynamického prahovania a vykreslenia rekurentných grafov. Ukladajú si všetky medzivýpočty do operačnej pamäte, čo zásadne obmädzuje schopnosť vypočítania dlhších vstupných signálov. Táto problematika je riešená implementáciou online algoritmov ako napríklad Welfordov online algoritmus na výpočet smerodajnej odchyľky alebo T-Digest a optimalizovaná implementácia matlabového algoritmu na výpočet hodnôt kvantilu, ktoré nahradzujú potrebu ukladať velké množstvý medzivýpočtov do operačnej pamäte. Výsledkom je aplikácia RQA s grafickým užívateľským rozhraním, ktorá je schopná analýzy dlhších vstupných signálov, vykreslovania rekurentného grafu a heatmapy. Ponúka rôzne metódy výpočtu kvantilu potrebného na výpočet rôznych metód prahovania.

Klúčové slová: optimalizácia, paralelizácia, rekurentná kvantifikačná analýza, T-Digest, Welfordov algoritmus

Abstract

Available applications for computing recurrent quantification analysis are limited by the signal length they are able to process, do not offer the possibility of dynamic thresholding and plotting recurrent graphs. They store all the intermediate calculations in operational memory, which fundamentally limits the ability to compute longer input signals. This issue is addressed by implementing online algorithms such as Welford's online algorithm for computing standard deviation or T-Digest and an optimised implementation of Matlab's algorithm for computing quantile values, which replace the need to store large amounts of intermediate calculations in operational memory. The result is an RQA application with a graphical user interface that is capable of analysing longer input signals, plotting a recurrent graph and a heatmap. It offers different methods to calculate the quantile required for different thresholding methods.

Keywords: optimisation, parallelisation, recurrent quantification analysis, T-Digest, Welford's algorithm

Poďakovanie

Týmto by som sa chcel poďakovať Ing. Lenke Hanákovej a doc. Ing. Bc. Vladimírovi Sochovi, Ph.D. za konzultácie, ochotu, cenné rady, pripomienky a zasvetenie do sveta analýzy nelineárnych dát.

Ďalej by som chcel poďakovať Milanovi Ondrášovi za dlhé večerné rozhovory, morálnu podporu v písaní diplomovej práce a zasvetenie do sveta jednotiek a núl.

V neposlednom rade by som chcel poďakovať mojej rodine za podporu aj v tých najťažších časoch, mojím kamarátom a spolužiakom za krásne chvíle strávené počas celého študentského života.

Čestné prehlásenie

Prehlasujem, že som diplomovú prácu s názvom Optimalizace algoritmů pro rekurentní kvantifikační analýzu vypracoval samostatne a použil k tomu úplný zoznam citácií použitých prameňov, ktoré uvádzam v zozname přiloženom k diplomovej práci.

Nemám závažný dôvod proti použitiu tohto školského diela v zmysle §60 Zákona č.121/2000 Sb., o autorskom práve, o právach súvisiacich s autorským právom a o zmene niektorých zákonov (autorský zákon).

V Prahe dňa 8. augusta 2021 .....

Podpis

Obsah

Zoznam obrázkov

Zoznam tabuliek

Zoznam skratiek

BPP	Bity na pixel (Bits per pixel)
CLI	Rozhranie príkazového riadku (Command Line Interface)
CLR	Common Language Runtime
CPU	Centrálna procesorová jednotka (Central processing unit)
CSV	Hodnoty oddelené čiarkou (Comma separated values)
DET	Determinizmus (Determinism)
DM	Vzdialenostná matica (Distance matrix)
DIV	Divergencia (Divergence)
EEG	Elektroencefalografia
EKG	Elektrokardiografia
ENT	Entropia (Entropy)
GC	Garbage collector
GPU	grafický procesor (Graphics processing unit)
GUI	grafické uživateľské rozhranie (Graphical user interface)
HM	Heatmapa (Heatmap)
LAM	Laminarita (Laminarity)
LOI	Hlavná diagonála (Line of identity)
MEQ	Exaktný matlabový kvantil (Matlab exact quantile)
RAM	operačná pamät (Random access memory)
RGB	červený,zelený a modrý farebný model (Red, green, blue)
RP	Rekurentný graf (Recurrence plot)
RQA	Rekurentná kvantifikačná analýza (Recurrence quantification analysis)
RR	Miera rekurencie (Recurrence rate)
TND	Trend
TT	Čas zachytenia (Trapping time)
UTF-8	Bezstratové kódovanie s variabilnou dĺžkou
WPF	Windows Presentation Foundation

Úvod

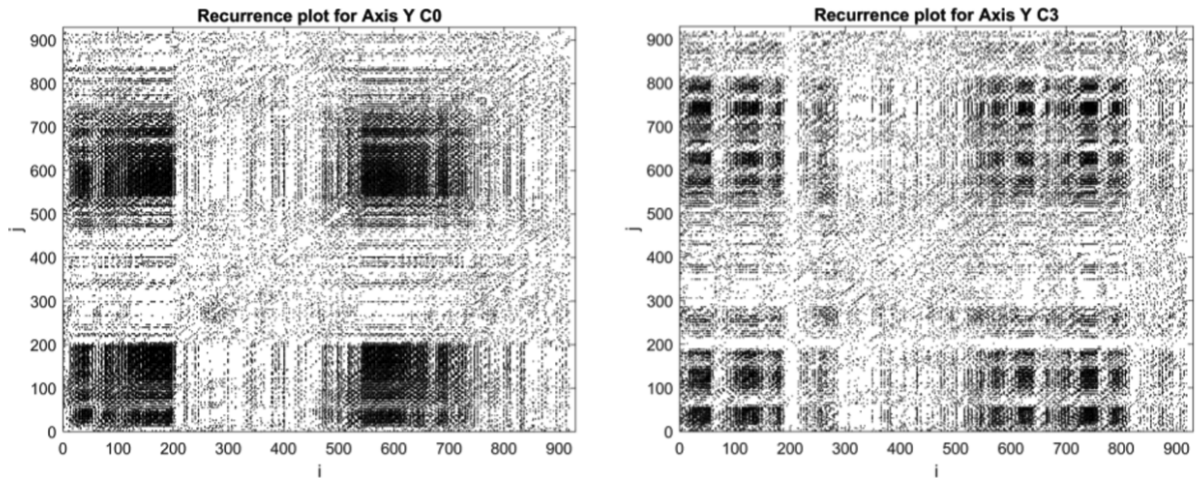
Rekurentná kvantifikačná analýza je relatívne mladou metódou, ktorá slúži na analýzu nelineárnych dát. Táto metóda sa využíva hlavne v oblasti biomedicíny pri spracovaní EKG a EEG signálov alebo vibrodiagnostike leteckých motorov. Pri hodnotení nelineárnych signálov sa využíva vizuálna stránka vo forme rekurentného grafu a vo forme jedenástich kvantifikačných parametrov, ktoré charakterizujú analyzovaný signál.

Danou problematikou je samotné spracovanie dát, ktoré je pri dlhších signáloch výpočetne náročnejšie. Limitáciou každého softvérového riešenia je nedostačujúca optimalizácia algoritmov, ktoré nie sú prispôsobené dlhším signálom. Z tohto dôvodu sa dáta, ktoré chceme analyzovať, musia rozdeliť na také časti, ktoré je konkrétne softvérové riešenie schopné spracovať. Dalším nedostatkom stávajúcich riešení je absencia dynamického prahovania vzdialenostnej matice. Väčšina dostupných softvérových riešení sú len konzolovými aplikáciami a nedisponujú grafickým uživateľským rozhraním a možnosťou nastavenia vstupných parametrov.

Táto práca sa zaoberá optimalizovaním alogoritmov pre rekurentnú kvantifikačnú analýzu s prijatelnými nárokmi na operačnú pamäť a dostatočnou rýchlosťou výpočtu. Cieľom tejto práce je návrh desktopovej aplikácie s grafickým uživateľským rozhraním schopné vykreslovať rekurentný graf a heatmapu. Hlavným cieľom optimalizácie softvérového riešenia je schopnosť spracovávať signály dlhšie ako 100 000 vzoriek.

1 Teoretické základy práce

Rekurentná kvantifikačná analýza je metóda nelineárnej analýzy dát skúmajúca dynamiku systému. Kvantifikuje počet a trvanie opakovania alebo rekurencie dynamického systému prezentovaných jeho trajektóriou vo fázovom priestore [1]. Táto metóda sa používa napríklad pri vibrodiagnostike motorov používaných v leteckom alebo v automobilovom priemysle, kde sa pomocou tejto metódy dá detekovať chybný zážih valcov [2]. Keďže sú dáta merané v každej osi, tak pomocou kombinácie RQA s lineárnou analýzou je možné identifikovať špecifický válec. Limitáciou je vykreslenie relatívne malého časového úseku, keďže všetky dostupné softvérové riešenia nedokážu spracovať dlhšie signály. Aby sa dosiahlo spracovania 4,5 sekundy chodu motora, autori museli rozdeliť dáta na 150 častí o velkosti 950 vzoriek. Na obrázku 1.1 sú zobrazené rekurentné grafy vibračných signálov motora. Naše softvérové riešenie by takto veľký signál zvládlo spracovať celý bez prerozdelenia na menšie časti. Taktiež sa používa na spracovanie dát z mikro doplerového radaru, kde pomocou rekurentnej kvantifikačnej analýzy je možné lepšie rozpoznanie cieľov vo vzdušnom priestore[3].



Obr. 1.1: Využitie rekurentnej kvantifikačnej analýzy pomocou vibrodiagnostiky na motore ROTAX 912. Rekurentný graf na ľavej strane ukazuje správny chod motora a rekurentný graf na pravej strane poukazuje na chybný zážih tretieho valca.[2]

1.1 Definícia rekurektného grafu

Rekurentný graf je pokročilá technika nelineárnej dátovej analýzy. Výsledkom je vizualizácia štvorcovej matice, v ktorej prvky matice zodpovedajú časom, v ktorých sa stav dynamického systému opakuje. Prirodzené procesy môžu mať zreteľné opakujúce sa správanie, ako aj nepravidelné.

Fázový priestor väčšinou nemá dve alebo tri dimenzie, čo by umožňovalo zobrazenie. Vyššie dimenzované fázové priestory je možné vizualizovať len premietnutím do dvoj alebo troj rozmerných priestorov. Tento problém rieši Eckmann pomocou rekurentného grafu, pomocou ktorého je možné prešetriť viac dimenzionálnu trajektóriu fázového priestoru prostredníctvom dvojrozmerného znázornenia rekurencií. Takáto rekurencia je označená v dvojrozmernej štvorcovej matici jednotkami (čiernymi bodkami) a nulami (bielymi bodkami), kde obidve osi sú časové. Toto znázornenie sa nazýva rekurentný graf (RP = Recurrence plot) [1].

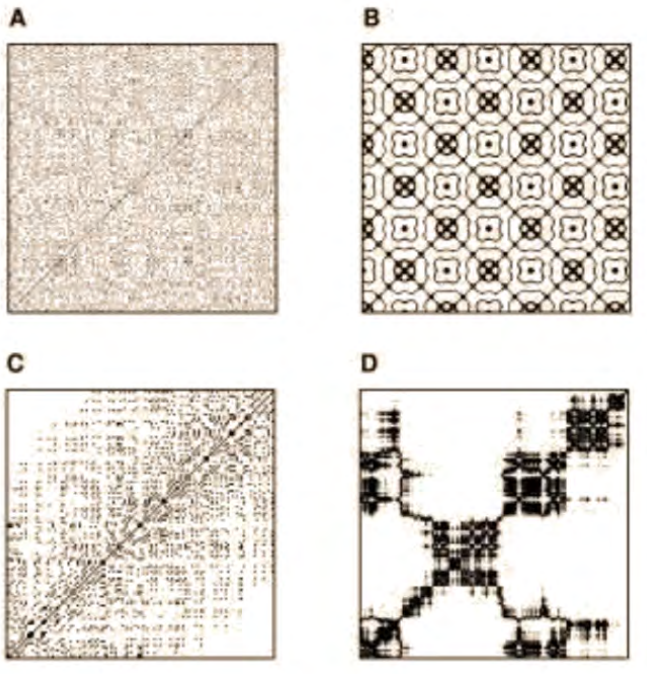
1.2 Štruktúry v rekurentných grafoch

Počiatočným účelom rekurentných grafov je kontrola viacdimenzionálnych trajektorií fázového priestoru rekurentných grafov. Výhodou rekurentných grafov je možnosť aplikácie aj na krátke alebo nestacionárne signály. Rekurentné grafy ukazujú veľké a malé charakteristické vzory [4].

1.2.1 Veľké charakteristické vzory

Homogénne rekurentné grafy (obr. 1.2A) sú typickým príkladom autonómnych a stacionárnych systémov. Tieto systémy majú krátke relaxačné časy v porovnaní s časom prekonaným v rekurentných grafoch. Príkladom môže byť náhodný časový rad. Periodické rekurentné grafy sa vyznačujú pravidelnými diagonálnymi a šachovnicovými štruktúrami (obr. 1.2B). Pri kvasiperiodicite systému sú síce diagonálne čiary od seba rôzne vzdialené a teda aj ťažšie rozpoznateľné. Avšak aj pre tieto oscilačné systémy sa dá využiť metóda

rekurentného grafu a nájsť tak osciláciu systému. Posun je spôsobený systémami s pomaly sa meniacimi parametrami. Takáto pomalá zmena sa zobrazí v rekurentom grafe na pravom dolnom rohu a ľavom hornom rohu (obr. 1.2C). Narušené rekuretné grafy, ktoré zaznamenávajú prudké zmeny v dynamike alebo extrémne udalosti, spôsobujú takzvané biele oblasti alebo pásy (obr. 1.2D) [4].



Obr. 1.2: Príklady rekurentných grafov – homogénny graf (A), periodický signál (B), posun (C) a narušený rekurentný graf (D) [4].

1.2.2 Malé charakteristické vzory

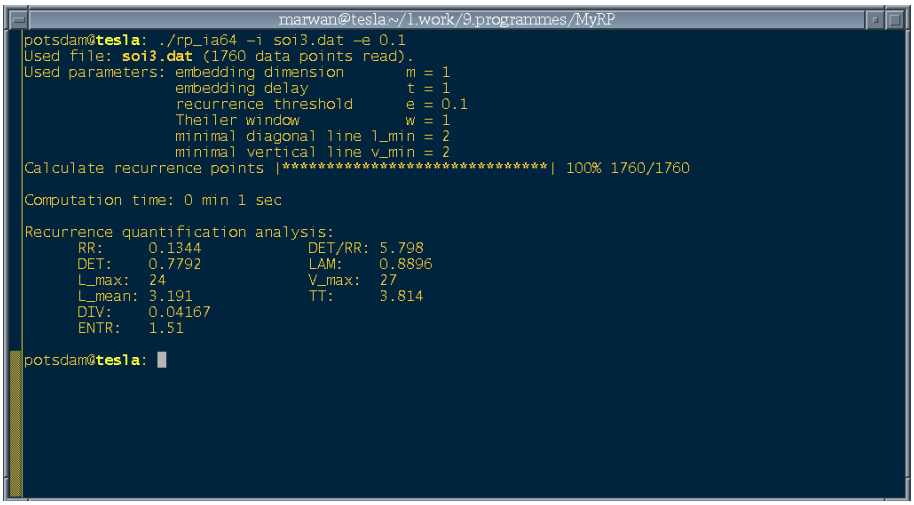
Jednotlivé body - Jednotlivé izolované body opakovania sa môžu vyskytnúť, ak sú stavy zriedkavé, ak nepretrvávajú žiadny čas alebo ak silne kolíšu. Nie sú však jedinečným znakom náhody alebo hluku. Diagonály čiary - Diagonálne čiary nastávajú v prípade, keď segment trajektórie je paralelný s iným segmentom. To znamená, že trajektória navštevuje rovnakú oblasť fázového priestoru, ale v rôznych časoch. Vertikálne a horizontálne čiary - Vertikálna

alebo horizontálna čiara označuje časovú dĺžku, v ktorej sa stav mení veľmi pomaly alebo sa nemení vôbec[4].

1.3 Softvérové riešenia

So vznikom rekurentnej kvantifikačnej analýzy vznikla aj potreba rozvoja softvérového riešenia na výpočet parametrov RQA a vytvorenie rekurentného grafu. Jeden z prvých programov RQA Software vyvinul v roku 1996 Charles Webber Jr., ktorý bol jedným z pôvodných zakladateľov rekuretnej kvantifikačnej analýzy. Aktualizovaná verzia z roku 2018 obsahuje 49 spustiteľných programov, ale slúži len na spracovanie relatívne krátkych signálov [5].

Program Commandline Recurrence plots od Norberta Marwana bol vyvinutý v roku 2006. Nevýhodou je schopnosť spracovať len relatívne krátke signály. Táto aplikácia nedisponuje grafickým uživateľským rozhraním, všetky príkazy musí uživateľ zadávať pomocou príkazov do konzole (viď obr. 1.3) [6].



Obr. 1.3: Ukážka Commandline Reccurence plots aplikácie na výpočet kvantifikačných parametrov [6].

PyRQA je sofvér, ktorý bol vyvinutý v roku 2016. Pracuje na princípe rozdeľovania matice na menšie matice a následne sa počítajú paralelne pomocou rozhrania OpenCL, ktorý tieto úlohy prerozdeľuje na grafické karty. Týmto sa im podarilo docieliť spracovanie radu dát dlhšieho ako 1 milión s použitím rozdielneho a menej výpočtovo náročného prahovania, absencie výpočtu optimálnej dimenzie vnorenia a časového oneskorenia [7].

Posledným spomenutým softvérom je RQAcalc od autorov Socha V., Schlenker J. a Hanáková H., ktorý je založený na programovacom jazyku Matlab. Softvér ponúka grafické uživatelské rozhranie s možnosťou jednoduchého nastavenia jednotlivých parametrov, ako aj vykreslenie rekurentného grafu. Jednou z nevýhod je, že tento softvér nedokáže spracovať dlhšie signály ako približne 12 000 vzoriek. Dĺžka signálu, ktorý tento softvér dokáže spracovať, je podmienená výpočetnou silou daného hardvérového riešenia, najmä od množstva dostupnej operačnej pamäte RAM na ukladanie všetkých vektorov a matíc na ďalšie výpočty [8].

1.4 Zhrnutie kapitoly

Rekurentná kvantifikačná analýza sa používa na analýzu nelineárnych dynamických systémov. Problematikou súčasných softvérových riešení na spracovanie nelineárneho signálu pomocou rekurentnej kvantifikačnej analýzy je nespôsobilosť spracovania dlhších signálov z dôvodu slabej optimalizácie na modernú dobu, požiadaviek na dostupnosť veľkého množstva operačnej pamäte RAM a absencie grafického rozhrania.

2 Metódy

Na vytvorenie rekurentného grafu a výpočet rekurentných kvantifikačných parametrov je potrebných niekoľko krokov, aby sa dosiahlo požadovaných výsledkov. Prvým krokom je zvolenie správneho časového oneskorenia a následne aj dimenzie vnorenia. Ďalším krokom je výpočet vzdialenostnej matice a rekurentného grafu, z ktorého sú počítané kvantifikačné parametre.

2.1 Rekonštrukcia trajektórie v časovom priestore

Dynamika celého systému je vyjadrená trajektóriou vo fázovom priestore. Niektoré metódy dovoľujú rekonštrukciu trajektórie pomocou jedného skalárneho časového radu. Najčastejšie je používaná metóda dimenzie vnorenia a časového oneskorenia, ktorá bola navrhnutá v 80. rokoch 20. storočia a je daná vzťahom:

$$x(t_i) = [x(t_i), x(t_i + \tau), ..., x(t_i + (m-1)\tau)], \tag{2.1}$$

kde $i = 1, ..., M$, m je dimenzia vnorenia, τ je časové oneskorenie a $M = N - (m-1)\tau$, kde N je počet vzoriek [4].

Základnými podmienkami rekonštrukcie fázoveho priestoru sú správna voľba dimenzie vnorenia a časového oneskorenia. Nesprávny výber týchto dvoch parametrov môže ovplyvniť výsledok parametrov rekurentnej kvantifikačnej analýzy a rekurentného grafu. Bolo dokázané, že pre atraktor o dimenzií D je postačujúca dimenzia $m \geq 2D + 1$. Fázovy priestor, zrekonštruovaný pomocou vzťahu uvedeného vyššie, nie je totožný s fazovým priestorom, ale za istých predpokladov je dynamika oboch systémov zhodná.

2.2 Časové oneskorenie

Časové oneskorenie by malo byť zvolené tak, aby sa minimalizovala interakcia medzi bodmi časového radu. Pri voľbe časového oneskorenia môžu nastať tri prípady. Ak je zvolené časové oneskorenie príliš malé, jedná sa o tzv. redundantný stav, kde nedochádza k výraznému

navýšeniu informacií o dynamike systému. Ak je zvolené časové oneskorenie príliš vysoké, jedná sa o tzv. irelevantný stav, kde sa dynamika systému javí ako chaotická. Tretím stavom je zvolenie optimálneho časového oneskorenia, na ktoré boli navrhnuté rôzne metódy. Dvomi najčastejšími sú autokorelačná funkcia a minimum vzájomnej informácie.

Softvérové riešenie využíva práve druhú metódu minima vzájomnej informácie. Táto vzájomná informácia I udáva vzájomnú závislosť medzi dvomi veličinami, a teda čím je táto závislosť vyššia, tým vyššiu informáciu zo systému získame. Tento výpočet vychádza z entropie a je daný vzťahom:

$$I(A, B) = H(A) + H(B) - H(A, B), \tag{2.2}$$

kde A a B sú jednotlivé premenné, $H(A)$ a $H(B)$ sú entropie jednotlivých premenných A a B a $H(A, B)$ je združená entropia A a B. [4].

2.3 Dimenzia

Výber správnej dimenzie vnorenia je dôležitou súčasťou hlavne z výpočetného hladiska. Cieľom výberu správnej dimenzie vnorenia je, aby vo fázovom priestore nedochádzalo k pretínaniu trajektórií. Pri tomto hľadaní prichádza k trom stavom, ktoré môžu nastať. Ak je dimenzia vnorenia príliš nízka, trajektória križuje samú seba. Ak je dimenzia zvolená správne, ku kríženiu trajektórií neprichádza vôbec. v tomto prípade je zbytočné hľadanie inej dimenzie, pretože množstvo informacií o systéme dosiahlo maximum. Pokiaľ je dimenzia vyššia než tá optimálna, kríženie trajektórií sa znižuje, ale neposkytuje viac informacií o systéme. Väčšinou sa v biologických systémoch nepoužíva vyššia dimenzia vnorenia ako 20. Pri používaní vyšších dimenzií by teoreticky nemala nastať žiadna zmena, ale nastávajú v rekurentných grafoch isté artefaktové vzory rekurencie.

Metóda použitá na voľbu správnej dimenzie vnorenia, a jednou z najpoužívanejších je metóda falošných susedov, ktorej nevýhoda spočíva v potrebe volenia prahovej hodnoty, v ktorej dva body považujeme ešte za susedné. v roku 1997 bola táto metóda falošných susedov upravená

Caom a je daná vzťahom:

$$a(i,m) = \frac{\|x_{m+1}(i) - x_{m+1}^{NN}(i)\|}{\|x_m(i) - x_m^{NN}(i)\|}. \tag{2.3}$$

kde $\|\cdot\|$ je Euklidová vzdialenosť, $x_m(i)$ je i rekonštruovaný vektor s dimenziou m a $x_m^{NN}(i)$ je jeho najbližší sused.[4].

Z tohto vzťahu je odvodený priemer všetkých hodnôt $E(m)$, ktorý je daný vzťahom[4]:

$$E(m) = \frac{1}{N - m\tau} \sum_{i=i}^{N-m\tau} a(i,m). \tag{2.4}$$

Z tohto vzťahu je zrejmé, že pred výpočtom optimálnej dimenzie vnorenia je potrebné zvolenie časového oneskorenia. Podielom hodnoty $E(m)$ a $E(m+1)$ je charakterizovaná zmena počtu falošných susedov medzi dvomi susediacimi dimenziami.

2.4 Vzdialenostná matica a rekurentný graf

V roku 1987 Eckmann et al. predstavil rekurentné grafy, ktoré umožňujú vizualizáciu trajektórie vo fázovom priestore s vyššími dimenziami. Základom rekurentného grafu je vzdialenostná matica. Táto matica je štvorcová, symetrická podľa hlavnej diagonály. Z tejto vzdialenostnej matice je následne prahovaním získaná rekurentná matica, resp. rekurentný graf.

Pre výpočet vzdialenostnej matice je potrebné určiť normu, podľa ktorej sa budú počítať vzdialenosti medzi jednotlivými bodmi. v praxi najčastejšie používanou normou je Euklidovská norma, ktorá je počítaná Euklidovou vzdialenosťou. Vzdialenostná matica je daná vzťahom:

$$DM(i,j) = \| x(i) - x(j) \| \tag{2.5}$$

kde $\|\cdot\|$ je Euklidová vzdialenosť a $x(i)$ a $x(j)$ sú stavy systému v čase i [4].

Po naprahovaní vzniká zo vzdielnostnej matice rekurentná matica, resp. rekurentný graf, ktorý je daný vzťahom:

$$R(i,j) = \theta(\epsilon \| \ x(i) - x(j) \ \|) \tag{2.6}$$

kde θ je Heavisideova funkcia a ϵ je prahová vzdialenosť [4].

Vykreslovanie rekurentného grafu, ktorý má väčšie rozlíšenie ako rozlíšenie zariadenia, na ktorom je prezeraný. Prípadné znižovanie mierky a zmena velkosti môže spôsobovať artefakty a zlú interpretáciu vizuálnej stránky rekurentného grafu [9].

2.5 Kvantifikácia

Rekurentná kvantifikačná analýza sa zaoberá jedenástimi hlavnými parametrami alebo kvantifikáciami, ktoré vychádzajú zo samotného spracovania signálu. Keďže je rekurentný graf vždy symetrický podľa diagonály, parametre sa počítaju z vrchného pravouhlého trojuholníka rekurentného grafu.

Prvým parametrom je miera rekurencie (REC alebo RR - recurrence rate), ktorý vyjadruje percento rekurentných bodov, ktoré spadajú do zadaného rádiusu. Zjedodušene povedané, miera rekurencie udáva hustotu rekurenčných bodov v rekurentnom grafe. Miera opakovania zodpovedá pravdepodobnosti, že sa konrétny stav bude opakovať. Parameter RR je daný vzťahom [4]:

$$RR = \frac{1}{N^2} sum_{i,j=1}^{N} R_{i,j}. \tag{2.7}$$

Druhým parametrom je determinizmus (DET), ktorý udáva percento po sebe následujúcich rekurentných bodov, ktoré spoločne formujú diagonálne čiary paralelne s hlavnou diagonálou (LOI - line of identity). DET je daný vzťahom [4]:

$$DET = \frac{\sum_{l=l_{min}}^{N} lP(l)}{\sum_{l=1}^{N} lP(l)}. \tag{2.8}$$

kde $P(l)$ je histogram dĺžiek l diagonálnych čiar [4].

Tretím parametrom je laminarita (LAM), ktorá je analogická k parametru DET a udáva percento rekurentných bodov, ktoré spoločne formujú vertikálne čiary. Názov laminarita je odvodený od hromadenia viacerých paralelných trajektorií a z nehybností v časových radoch. Parameter LAM je daný vzťahom [4]:

$$LAM = \frac{\sum_{v=v_{min}}^{N} vP(v)}{\sum_{v=1}^{N} vP(v)}. \tag{2.9}$$

kde $P(v)$ je histogram dĺžiek v vertikálnych čiar [4].

Z parametrov DET a LAM sa následne počíta štvrtý paramater, pomer (RATIO), ktorý je daný vzťahom [4]:

$$RATIO = N^2 \frac{\sum_{l=l_{min}}^{N} lP(l)}{(\sum_{l=1}^{N} lP(l))^2}. \tag{2.10}$$

Štvrtým parametrom je priemerná dlžka diagonálnych čiar (L), ktorý je daný vzťahom [4]:

$$L = \frac{\sum_{v=v_{min}}^{N} vP(v)}{\sum_{v=v_{min}}^{N} P(v)}. \tag{2.11}$$

a súvisí s časovou predvídatelnosťou dynamického systému so šiestym parametrom času zachytenia (TT), ktorý meria priemernú dĺžku vertikálnych čiar. TT je daný vzťahom [4]:

$$TT = \frac{\sum_{v=v_{min}}^{N} vP(v)}{\sum_{v=1}^{N} P(v)}. \tag{2.12}$$

Siedmym a ôsmym parametrom je dĺžka najdlhšej diagonálnej čiary (Lmax) a dĺžka najdlhšej vertikálnej čiary (Vmax). Tieto parametre su dané vzťahom:

$$Lmax = max([l_i; i = 1, ..., N_l]), \tag{2.13}$$

$$Vmax = max([v_i; i = 1, ..., N_v]), \tag{2.14}$$

kde $N(v)$ je počet vertikálnych čiar v rekurentnom grafe a $N(l)$ je počet diagonálnych čiar v rekurentnom grafe [4].

Deviatym parametrom je divergencia (DIV), ktorá je mierou toho, ako rýchlo sa súbežné trajekórie rozbiehajú. Divergencia je inverzná k parametru Lmax. Čím je kratšia najdlhšia čiara, tým su trajekórie odlišnejšie. Paramter DIV je daný vzťahom [4]:

$$DIV = \frac{1}{L_{max}}. \tag{2.15}$$

Desiatym parametrom je Shannonova informačná entropia (ENT), ktorá je mierou komplexnosti systému a môže nadobúdať od 0 (žiadnej komplexnosti) až po ENT-max (maximálnu hodnotu entropie) [4].

$$ENT = -\sum_{l=l_{min}}^{N} p(l) ln p(l). \tag{2.16}$$

kde $p(l)$ je rozdelenie pravdepodobností dĺžok diagonálnych čiar [4].

Posledným parametrom je trend (TND), ktorý kvantifikuje stupeň stacionárnosti systému. Ak sú rekurentné body homogénne rozložené po rekurentom grafe, hodnota parametru TND sa bude pohybovať v blízkosti nuly. Ak sú rekuretné body rozložené heterogénne, hodnoty TND sa budú odchylovať od nuly. Parameter TND je daný vzťahom[4]:

$$TND = \frac{\sum_{i=1}^{\tilde{N}} (i - \tilde{N}/2))(RR_i - \langle RR_i \rangle}{\sum_{i=1}^{\tilde{N}} (i - \tilde{N}/2))^2}. \tag{2.17}$$

kde $\tilde{N}$ je maximálny počet diagonál paralelných k stredovej diagonálnej čiare LOI, RRi je lokálna rekurencia a $\langle RR_i \rangle$ je priemerná lokálna rekurencia [4].

2.6 Priemer a smerodajná odchylka

Welfordov alogirtmus z roku 1962 rieši otázku výpočtu priemeru a smerodajnej odchylky na streamovaných dátach. Celý cyklus prebehne len raz a tieto parametre sa počítajú priebežne. Celý algoritmus je daný vzťahmi pre priemer a rozptyl:

$$\bar{x}_n = \frac{(n-1)\bar{x}_{n-1} + x_n}{n} = \bar{x}_{n-1} + \frac{x_n - \bar{x}_{n-1}}{n} \tag{2.18}$$

$$\sigma^2{}_n = \frac{(n-1)\sigma^2{}_{n-1} + (x_n - \bar{x}_{n-1})(x_n - \bar{x}_n)}{n} = \sigma^2{}_{n-1} + \frac{(x_n - \bar{x}_{n-1})(x_n - \bar{x}_n) - \sigma^2{}_{n-1}}{n} \tag{2.19}$$

kde $\bar{x}_n$ je priemer predchádzajúcich prvkov, x_n je dalšia hodnota prvku a $\sigma^2{}_n$ je rozptyl [10].

2.7 Programovací jazyk C#

Programovací jazyk C# je obiektovo-orientovaný programovací jazyk vyvinutý spoločnosťou Microsoft. C# vznikol zo základov programovacích jazykov C++ a Java. Aplikácie vytvorené pomocou C# fungujú na ekosystéme .NET. Programy v jazyku C# sa spúšťajú v prostredí .NET, virtuálnom spúšťacom systéme nazývanom common language runtime (CLR). CLR od spoločnosti Microsoft je implementáciou common language interface(CLI). CLI je základom pre vytváranie vykonávacích a vývojových prostredí, v ktorých jazyky a knižnice bez problémov spolupracujú[11].

2.7.1 Microsoft Visual Studio

Microsoft Visual Studio je vývojové prostredie od spoločnosti Microsoft. Slúži na vytváranie konzolových aplikácií ako aj aplikácií s grafickým uživateľským rozhraním. Microsoft Visual Studio podporuje niekoľko programovacích jazykov ako C/C++, VB.NET (Visual Basic) ale aj C#. Základným nástrojom každej lepšej vývojovej platformy je nástroj, pomocou ktorého môžu vývojári vytvárať, zdieľať a využívať užitočný kód. To má za úlohu správca balíkov NuGet, pomocou ktorého sa dá implementovať do kódu knižnica Math.Net alebo T-digest, ktoré sú používané pri vývoji aplikácie[12].

2.7.2 Knižnica T-digest

Knižnica T-Digest je implementácia relatívne nového algoritmu T-digest. Slúži na výpočet veľmi presných kvantilov alebo percentuálnych podielov na streamovaných dátach. Konštrukčný algoritmus t-digest používa variant jednorozmerného k-rozmerného zhlukovania na vytvorenie veľmi kompaktnej štruktúry údajov, ktorá umožňuje presný odhad kvantilov. T-digest kontroluje veľkosť zhluku pomocou škálovacej funkcie, ktorá je daná vzťahom:

$$k(q, \delta) = \frac{\delta}{2\pi} sin^{-1}(2q - 1). \tag{2.20}$$

kde k je velkosť zhlukov, q je kvantil a δ je kompresný parameter [13].

2.7.3 Knižnica Windows Forms

Windows Forms je bezplatná a open-source knižnica, ktorá umožňuje jednoduchú tvorbu a vývoj grafického uživateľského rozhrania pomocou grafického designeru. Vývoj aplikácií funguje na princípe drag and drop, kde si uživateľ môže z knižnice nástrojov pretiahnuť potrebný blok do grafického rozhrania a upravovať jeho vizuálnu stránku, čo sa líši od knižnice wpf, kde sa užívateľské rozhranie píše v zdrojovom kóde [14].

2.7.4 Knižnica WPF

Windows Presentation Foundation je knižnica, ktorá slúži primárne na vývoj užívateľského rozhrania. WPF je modernejším nástupcom knižnice Windows Forms, zároveň ale obsahuje aj možnosti na vykreslovanie veľkých bitmáp. Táto knižnica je schopná vykresliť jednofarebné bitmapy o veľkosti 130 tisíc x 130 tisíc [15].

2.7.5 Math.NET

Knižnica Math.NET poskytuje metódy a algoritmy na numerické výpočty. Knižnica zahŕňa algoritmy linearnej algebry na prácu s maticami, pravdepodobnostnými hodnotami a interpoláciou na prácu s kvantilmi, ktoré sú používané v softvérovom riešení.

2.8 Zhrnutie kapitoly

Výpočet rekurentnej kvantifikačnej analýzy pozostáva z výpočtu optimálnej dimenzie vnorenia, časového oneskorenia a následného vytvorenia rekurentného grafu a výpočtu kvantifikačných parametrov. na vývoj aplikácie bol zvolený objektovo-orientovaný programovací jazyk C# v prostredí Microsoft Visual Studio, v ktorom je možné využitie knižníc tretej strany.

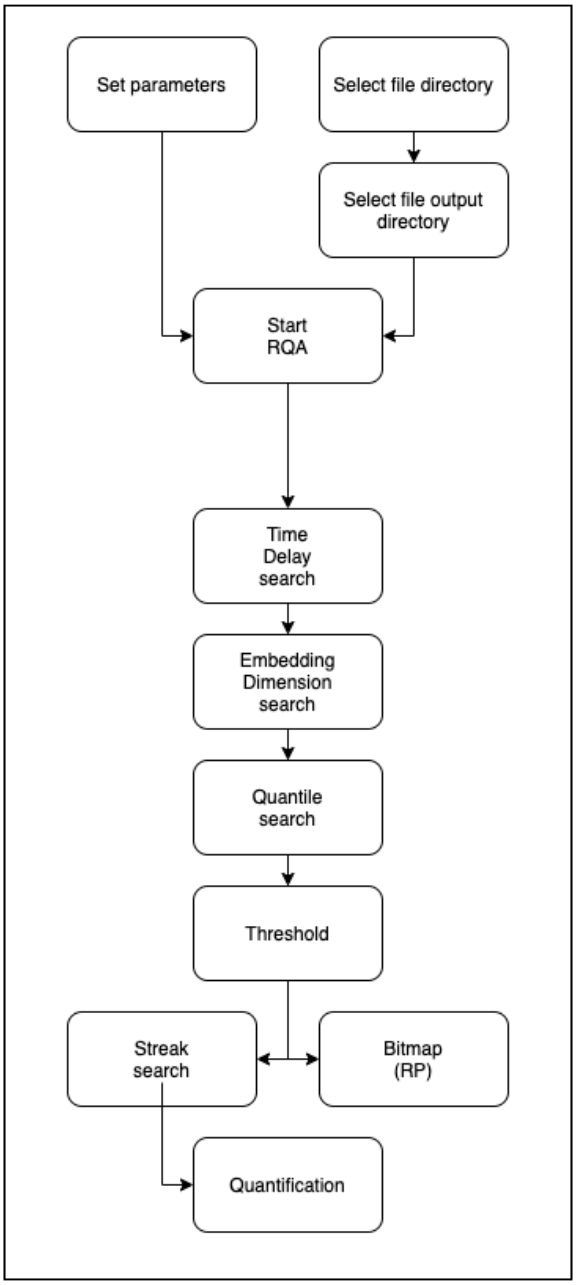
3 Realizácia a implementácia

Táto kapitola sa venuje samotnej realizácií softvéru a optimalizácií algoritmov pre rekurentnú kvantifikačnú analýzu. Softvér bol písaný za účelom dosiahnutia funkčnosti aj na menej výkonných počítačoch s ohľadom na operačnú pamäť RAM. Program bol robený podľa predlohy softvéru RQAcalc v prostredí Matlab. Z dôvodu optimalizácie bolo potrebné celý softvér prepísať do programovacieho jazyku C#. Po úspešnom prepísaní programu boli následne vykonavané kroky k optimalizovaniu celého softvérového riešenia.

3.1 RQA softvér

Softvér pre rekurentnú kvantifikačnú analýzu bol písaný v programovacom jazyku C# hneď z niekoľkých dôvodov. Keďže väčšina dostupných softvérov je založená na ukladaní dlhých vektorov a veľkých matíc do vyrovnávacej pamäte RAM, sú limitované práve dostupnou kapacitou v určitom počítači, na ktorom je analýza vykonávaná. Jedna z výhod programovacieho jazyka C# je tzv. garbage collector, ktorý slúži ako automatický manažér pre pamäť RAM. To znamená, že nie je až tak potrebné sledovať objekty, ktoré už nie su potrebné v ďalších procesoch programu a sú týmto garbage collectorom uvoľnené.

Jednotlivé hlavné časti algoritmu sú uvedené na diagrame na obrázku 3.1, ktoré tvoria celkové softvérové riešenie. Prvým krokom je zadanie parametrov v grafickom užívateľskom rozhraní pre rekurentnú kvantifikačnú analýzu, nastavenie cesty k cieľovému súboru, na ktorom sa bude vykonávať samotná analýza a cestu k zložke pre uloženie výsledkov, ako aj samotné vykreslenie rekurentného grafu. Po spustení softvéru je prvým krokom hľadanie optimálneho časového oneskorenia, ktoré je potrebné na výpočet optimálnej dimenzie vnorenia. Pre prahovanie vzdialenostnej matice, je potrebné zistiť paramtre jednotlivých metód prahovania, ako kvantil alebo smerodajnú odchýlku. Následne prebehne výpočet prahov a samotné prahovanie vzdialenostnej matice. Po tomto procese prebieha vykreslenie rekurentného grafu zároveň s počítaním diagonálnych a vertikálnych čiar. Vypočítané kvantifikačné paramtre sú potom s rekurentným grafom uložené do užívateľom zvolenej zložky.



Obr. 3.1: Zjednodušený diagram aplikácie RQA.

3.2 Signál a vstupné parametre

Podporovaným súborom softvérového riešenia je comma-separated values UTF-8 alebo .csv, kde sú hodnoty usporiadané v stĺpcoch a oddelené ”,”, desatinné číslo je oddeľované bodkou.

Všetky vstupné parametre sú zvolené uživateľom v grafickom rozhraní. Paramater dooneskorenia slúži na ohraničenie, do akého oneskorenia má byť optimálne časové oneskorenie hľadané. Taktiež to platí pre parameter dodimenzie, ktorým sa udáva, koľko dimenzií má softvér preskúmať a z týchto zvoliť tú najoptimálnejšiu. Program má v sebe implementovaných 7 rozdielných výpočtov prahu, kde si uživateľ môže zvoliť vyhovujúce prahovanie.

3.3 Časové oneskorenie

Metóda oneskorenie slúži na vyhľadanie optimálneho oneskorenia. Parameter oneskorenie je následne používaný na výpočet dimenzie vnorenia. Vstupnými parametrami pre metódu oneskorenia sú: Signál, ktorý analyzujeme a od akého oneskorenia chceme vyhladávať. Metóda beží v cykle ktorý na začiatku každej iterácie inkrementuje vstupný parameter dooneskorenia o jedna a cyklus končí až v prípade keď sa nájde validné minimum. Pre výpočet minima je potrebné vytvorenie dvoch vektorov, pre ktoré platí, že sa rovnajú vstupnému signálu, ktorý je pre jeden skratený o parameter tau od konca a pre druhý skratený o parameter tau na začiatku. Pomocou týchto vektorov môžeme následne vytvoriť histogramy, ktoré nám umožňujú vypočítať entropiu pomocou uvedeného zdrojového kódu. Vo vektore sa hľadajú nenulové čísla, ktoré sa vynásobia logaritmom o základe 2 toho čísla.

```
1 Vector<double> indices1 = Vector<double>.Build.Dense(n1.Count);
2 n11.Map(element =>
3                 {
4                     if (element != 0)
5                     {
6                         return element * Math.Log2(element);
7                     }
8                     else
9                     {
```

```
                        return 0;
                    }
                }, indices1, Zeros.Include);
Vector<double> indices2 = Vector<double>.Build.Dense(n2.Count)
n22.Map(element =>
                {
                    if (element != 0)
                {
                    return element * Math.Log2(element);
                }
                else
                    {
                    return 0;
                    }
                }, indices2, Zeros.Include);
```

Ďalším krokom je vytvorenie 3D histogramu, z ktorého sa počíta združená entropia. Sčítaním dvoch entropií z 2D histogramov a odčítanie združenie entropie sa vypočíta vzájomná informácia, ktorá sa pri každej iteráciou zapisuje do listu mif. Validné minimum je posledné minimum v liste miním, ktoré spĺňa podmienku, že je menšie ako predchadzajúce a menšie ako nadchádzajúce.

```
for (int i = 1; i < mif_m.Count - 1; i++)
                {
                    minVal = mif_m[i];
                    if (minVal < mif_m[i - 1] && minVal < mif_m[i + 1])
                    {
                        minimum = i;
                    }
                }
                if (minimum != -1)
                {
                    isFinished = true;
                }
            }
            return minimum + 1;
        }
```

Index tohto minima sa vyhľadá iteráciou uvedenou hore a je výstupom metódy oneskorenia, ktoré sa následne používa vo výpočte dimenzie vnorenia.

Optimalizácia výpočtu validného minima nebola potrebná, keďže v tejto metóde aplikácia pracuje len s vektormi, ktorých maximálna velkosť nepresahuje velkosť vstupného signálu. Samotnou limitáciou v prípade hľadania validného minima je dĺžka poľa, ktorého maximálna veľkosť nemôže presiahnuť 4 miliardy prvkov. Keďže používame číselné typy double, kde každý prvok má 8 bajtov, výslednou limitáciou metódy časového oneskorenia je dĺžka samotného vstupného signálu. Limitáciou tejto metódy je velkosť poľa, ktorá musí byť menšia ako 150 miliónov, aby sme zistili validné minimum.

3.4 Dimenzia vnorenia

Metóda dimenzia slúži na vyhľadanie vhodnej dimenzie vnorenia. Vstupnými parametrami pre metódu dimenzie sú: signál, ktorý analyzujeme, do akej dimenzie chceme vyhladávať a parameter časové oneskorenie. Metóda beží v cykle, ktorý na začiatku každej iterácie inkrementuje parameter m o jedna do momentu, kde sa tento parameter rovná parametru dodimenzie. Cieľom tohto algoritmu je vyhľadanie najmenšej absolútnej percentuálnej zmeny medzi vstupným signálom, ktorý chce užívateľ analyzovať a jednotlivými dimenziami, ktoré sú posunuté o parameter časového oneskorenia. Posledná časť algoritmu hľadania optimálnej dimenzie vnorenia je ukázaná nižšie.

```
1  for (int i = 0; i < percento.Length - 1; i++)
2  {
3      percento[i + 1] = Math.Abs((e_1[i + 1] - e_1[i]) / e_1[i]) * 100;
4  }
5  for (int i = 0; i < percento.Length; i++)
6  {
7      if (percento[i] <= 10)
8      {
9          dimenze.Add(i);
10     }
11 }
```

```
if (dimenze.Count != 0)
{
    return dimenze[0] + 1;
}
else
{
    return dimenze.Min()+1;
}
```

Na to, aby sme našli vzdialenosti vo fázovom priestore medzi dvomi signálmi, je potrebných niekoľko krokov. Matlabové riešenie využíva funkcie repmat a reshape. Funkcia repmat zoberie vektor xe a N2-krát ho nakopíruje pod seba, čím vytvára dlhé pole. Funkcia reshape zase pretransformuje daný vektor do matice.

```
x1 = repmat(xe, N2, 1);
x2 = reshape(repmat(xe(:), 1, N2)', N2 * N2, m);
```

Pri vývoji softvérového riešenia v programovacom jazyku C# sme museli zvoliť inú cestu ukladania tak veľkých matíc. Matice dimenzií si neukladáme v operačnej pamäti RAM, ale ak sú tieto hodnoty potrebné na další výpočet, pomocou algoritmu si vždy tieto hodnoty vieme spätne dopočítať. Táto metóda je síce časovo náročnejšia, ale pri realizácií sa kládol veľký dôraz na šetrenie operačnej pamäte RAM. Tento algoritmus sa používa nielen pri počítaní optimálnej dimenzie vnorenia, ale aj kvantilu, vykreslovaní rekurentného grafu a kvantifikácií, kde sú tieto matice potrebné.

```
for (int i = 0; i < n2; i++)
{
    for (int j = i; j < length1; j++)
    {
        riadokSum = 0;
        if (j == i)
        {
            // Hlavna diagonala
        }
        else
        {
            // Pod/nad diagonalou
```

```
13              for (int k = 0; k < length0; k++)
14              {
15                  value = (xearr[k, j] - xearr[k, i]);
16                  riadokSum += value * value;
17              }
18              riadokSum = Math.Sqrt(riadokSum);
19              // help je hodnota matice S na~indexe i,j
20              help = (riadokSum - minSum) / (maxSum - minSum);
21          }
22      }
23  }
```

Z dôvodu optimalizácie sa využíva metóda subdimenzie, pomocou ktorej hľadáme minimá v každom riadku. Výsledkom sú parametre aritmetického priemeru, ktoré sa používajú na vyhodnotenie optimálnej dimenzie vnorenia.

Tento proces je pri vačších signáloch naročnejší na operačnu pamäť RAM. Softvérové riešenie v matlabe si všetky tieto signály posunuté o parameter časového oneskorenia ukladá do pamäte RAM. Jednou z výhod takéhoto postupu je rýchlejší výpočet, keďže sú tieto dáta ihneď dostupné. Pri dlhších signáloch dochádza k exponencialnemu rastu objemu dát, ktoré sa musia ukladať do operačnej pamäte a zaberajú veľa miesta.

3.5 Výpočet prahu a prahovanie

Program disponuje siedmimi metódami výpočtu prahu, ktoré sú označené pomocou písmena "v". na každú metódu výpočtu prahu je potrebný výpočet rozdielnych parametrov. na výpočet niektorých metód je potrebný výpočet kvantilu, priemeru a smerodajnej odchylky.

- Prvá metóda prahovania vychádza z p-kvantilu priemeru fázového priestoru (v=1).
- Druhá metóda prahovania je percento, ktoré je počítané vynásobením parametra p so strednou hodnotou (v=2).
- Tretia metóda prahovania je prahový násobok maximálneho priemeru fázového priestoru (v=3).

- Štvrtá metóda prahovania je násobkom štandardnej odchýlky časového radu (v=4).
- Piata metóda prahovania je pomer štandardnej odchýlky priemerov fázového priestoru (v=5).
- Šiesta metóda prahovania sa rovná hodnote, ktorú uživateľ v grafickom rozhraní nastavil do textového poľa p (v=6).
- Siedma metóda prahovania je násobkom štandardnej odchýlky priemerov fázového priestoru (v=7).

Prahovanie následne prebieha porovnávaním hodnoty metódy prahovania s hodnotami vo vzdialenostnej matici. Keď je hodnota prahovania väčšia alebo rovná ako hodnota vo vzdialenostnej matici na konkretnom indexe, nastaví sa jednotka. Ak je hodnota prahovania menšia, nastavi sa nula. Práve z tohto prahovania vznika sammotný rekurentný graf, ktorý sa vykresluje a počítajú sa na ňom kvantifikačné parametre.

3.5.1 Priemer a smerodajná odchylka

Welfordov online algoritmus je používany na výpočet priemeru a smerodajnej odchýlky v jednom kroku bez ukladania predošlých dát. To sa používa napriklad pri dátach, kde nie je dostatok pamäte na udržiavanie všetkých hodnôt. Keďže toto softvérové riešenie musí dokázať spracovať aj dlhšie signály, bola zvolená práve táto metóda, keďže používame metódu streamovacích dát.

```
for (int i = 0; i < n2; i++)
{
    for (int j = i; j < length1; j++)
    {
        riadokSum = 0;
        double help;

        for (int k = 0; k < length0; k++)
        {
            value = (xearr[k, j] - xearr[k, i]);
            riadokSum += value * value;
```

```
12          }
13          riadokSum = Math.Sqrt(riadokSum);
14          help = (riadokSum - minSum) / (maxSum - minSum);
15
16          //Výpočet rozptylu a~priemeru
17          //pomocou Welfordovej metody
18
19          oldM = varMean;
20          varMean += ((help - varMean) / count);
21          S += ((help - varMean) * (help - oldM));
22          sum += help;
23          count++;
24      }
25  }
26  double variance = S / count; //Smerodatna odchylka
27  double mean = sum / (n2^2)/2; //Priemer
```

Výsledkom tohto algoritmu sú priemer a rozptyl. Pomocou odmocnenia rozptylu získame práve smerodajnú odchylku, ktorá je potrebná pri prahovaní pomocou štvrtej a piatej metódy prahovania.

3.5.2 Kvantil

Výpočet kvantilu je v matlabovom riešení počítaný na vektore, ktorého velkosť je dĺžka signálu umocneného na druhú. Toto riešenie sa hodí na kratšie signály, ktoré sa zmestia do operačnej pamäte RAM. Pri dlhších signáloch bolo potrebné zvoliť inú metódu výpočtu kvantilu. Softvérové riešenie disponuje 3 voľbami výpočtu kvantilu:

Metóda T-Digest - používa sa na výpočet kvantilu zhluku dát, v ktorých počíta kvantilové odhady pre každú časť osobitne a následne tieto odhady skombinuje pomocou lineárnej interpolácie, pričom zaberá konštatnú pamäť. T-Digest používa veľké zhluky dát na oblasti, ktoré sú blízko q=0,5, malé a presnejšie zhluky dát pre q blížiace sa k hodnotám q=0 alebo q=1, čo spôsobuje väčšiu presnosť pri hľadaní malých kvantilov. Počet týchto zhlukov (compression) sa dá nastaviť v grafickom užívateľskom rozhraní. Pri vyššom počte zhlukov narastá presnosť výsledkov s dlhšiou dobou trvania výpočtu kvantilu.

```
    //Metoda T-Digest
    %
    MergingDigest digest = new MergingDigest(compression);
    k_min = digest.Quantile(p_min);
    k_max = digest.Quantile(p_max);
    k_10 = digest.Quantile(p_10);
    k_p = digest.Quantile(p);
```

Metóda ME-quantile - Matlab exact quantile funguje na princípe funkcie kvantilu v matlabe. Hodnoty si tento algoritmus ukladá do tzv. SortedDictionary. Nevýhodou tejto metódy môžu byť nevhodné dáta s nízkym opakovaním, kde si táto metóda musí ukladať všetky špecifické hodnoty, čím narastá využitie operačnej pamäte RAM. Ak chceme robiť lineárnu interpoláciu, potrebujeme hodnoty z ľavej a pravej strany. Matlab počíta kvantil zoradením dát, kde každému datapointu patrí hodnota určitého kvantilu. Pre jednotlivý datapoint sa dá vypočítať hodnota kvantilu pomocou vzorca (n-0,5)/n. Najskôr potrebujeme vypočítať indexy, ktore nesú hodnotu datapointov kvantilu na ľavej a na pravej strane od nášho kvantilu. Realkvantilsize je veľkosť vektoru, RealH je hodnota prvého datapointu a Realkrok je hodnota, o koľko bude rásť ďalšia hodnota kvantilu od toho prvého. Kindex-min je číslo, ktoré určuje, koľko hodnôt bude SortedDictionary udržiavať. Toto sa urobi pre každý kvantil, ktorý chceme vypočítať.

```
//Metoda ME-quantile

double realKvantilSize = (double)n2 * n2;
double realH = 0.5 / realKvantilSize;
double realKrok = realH * 2;

//Pre každý kvantil
if (p_min > 0.5) //p, p_min, p_10
{
    double indexVpravo_min = Math.Ceiling((p_min -realH) / realKrok);
    pVpravo_min =realH + (indexVpravo_min * realKrok);
    kIndex_min = realKvantilSize - indexVpravo_min +1;
    kth_min = new KthLargest(kIndex_min, false, q_round, useRounding);
}
else
{
    double indexVpravo_min = Math.Ceiling((p_min -realH) / realKrok);
    pVpravo_min =realH + (indexVpravo_min * realKrok);
```

```
    kIndex_min = indexVpravo_min + 1;
    kth_min = new KthLargest(kIndex_min, true, q_round, useRounding);
}
```

Pomocou metódy ADD pridávame hodnoty do SortedDictionary a udržiavame jeho veľkosť. Algoritmus najskôr zistí, či velkosť SortedDictionary nie je väčšia ako hľadaný index. Následne hodnota, ktorá má byť priadená, sa hľadá v SortedDictionary. Ak záznam o tejto hodnote existuje, tak sa na ten zaznam pridá, čím sa zvýši aj veľkosť celého záznamu. Ak sa nenachadza v zázname, tak sa táto hodnota pridá a celková velkosť sa inkrementuje. Ak je celková velkosť záznamu väčšia ako hľadaný index, prebieha odstraňovanie hodnôt, ktoré nás v tomto prípade nezaujímajú.

```
public void Add(double num)
    if (useRounding) num = System.Math.Round(num, q_round);
    if (actualSize == kth)
    {
        if (minHeap.First().Key < num && smallest) return;
        if (minHeap.First().Key > num && !smallest) return;
    }
    if (minHeap.ContainsKey(num))
    {
        minHeap[num] += count;
        actualSize += count;
    }
    else
    {
        minHeap.Add(num, count);
        actualSize += count;
    while (actualSize > kth)
    {
        KeyValuePair<double, int> minKV;
        minKV = minHeap.First();
        if (minKV.Value == 1)
        {
            minHeap.Remove(minKV.Key);
            actualSize--;
        }
        else if (actualSize - minKV.Value > kth)
        {
            actualSize -= minKV.Value;
            minHeap.Remove(minKV.Key);
        }
```

```
31          else
32          {
33              minHeap[minKV.Key]--;
34              actualSize--;
35          }
36      }
```

Po dokončení a naplnení SortedDictionary si pomocou metódy GetIndex vytiahneme hodnoty vlavo-min a vpravo-min, ktoré sú reálnymi hodnotami kvantilu v týchto indexoch. Keďže nás zaujíma hodnota kvantilu medzi týmito hodnotami, posledným krokom je lineárna interpolácia.

```
1  double vlavo_min;
2  double vpravo_min;
3  if (kth_min.smallest) (vlavo_min, vpravo_min)
4      = trueK.GetAtIndex((int)kth_min.kth);
5  else (vlavo_min, vpravo_min) =
6      falseK.GetAtIndexFalse((int)kth_min.kth);
7  k_min = vlavo_min +
8      ((p_min - (pVpravo_min - realKrok)) / realKrok) *
9          (vpravo_min - vlavo_min);
```

Metóda Math.NET - funguje len na dátach do dĺžky približne 60 000. Táto limitácia je spôsobená tým, že si musí táto metóda všetky hodnoty uložiť do operačnej pamäte RAM a nesmie presiahnuť maximálnu veľkosť poľa.

```
1  //Metoda Math.NET
2  k_min = MathNet.Numerics.Statistics.Statistics.Quantile(S2, p_min);
3  k_max = MathNet.Numerics.Statistics.Statistics.Quantile(S2, p_max);
4  k_10 = MathNet.Numerics.Statistics.Statistics.Quantile(S2, p_10);
5  k_p = MathNet.Numerics.Statistics.Statistics.Quantile(S2, p);
```

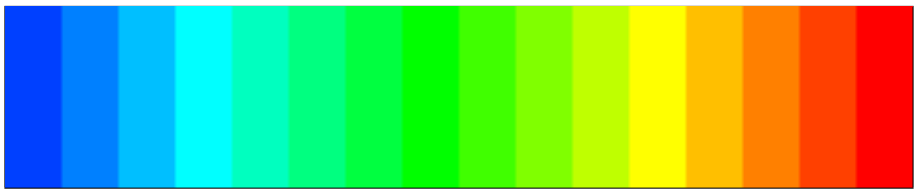
Podľa voľby metódy výpočtu kvantilu zaberá proces počítania kvantilu najviac operačnej pamäte RAM v porovnaní s počítaním smerodajnej odchýlky, priemeru a samotného prahovania. Dĺžka vstupného signálu a rozmanitosť dát priamo ovplyvňujú výpočetnú náročnosť ME-quantile metódy. Táto metóda je odzrkadlením funkcie metódy v Matlabe, kde je využívaná na týchto dátach. v našom softvérovom riešení je tento algoritmus poupravený práve metódou SortedDictionary, aby mohla fungovať na dlhšiom signále.

3.6 Vykreslenie rekurentného grafu a heatmapy

Vykreslenie prebieha medzi poslednými procesmi aplikácie z dôvodu potreby naprahovanej vzdialenostnej matice. Keďže na väčšinu metód prahovania je potrebný výpočet kvantilu, priemeru a smerodajnej odchylky, je potrebné počkať na dokončenie výpočtov týchto procesov. Vykreslenie rekurentného grafu prebieha paralelne s výpočtom kvantifikačných parametrov, aby bola čo najviac využitá kapacita procesoru. Heatmapa sa vykresluje zo vzdialenostnej matice, kde sa hľadá minimálna a maximálna hodnota, podľa ktorej sa určí, aká farba bude v enkóderi priradená jednotlivým pixelom.

3.6.1 Nastavenie palety farieb

Na základe nastavenia uživateľa a dĺžky signálu prebehne jeden z algoritmov na generáciu farebnej palety, ktorá bude následne použitá na vykreslenie heatmapy. Uživateľ má na výber dve farebné palety podľa nastavenia v grafickom rozhraní. Prvá farebná paleta (obr. 3.2) sa skladá, z dôvodu bitového obmädzenia, len zo 16 farieb. Maximálna veľkosť heatmapy využívajúcej 4 bitov na pixel je nastavená na signál dlhý 60 000. na vykreslenie rekurentného grafu stačí len čierna a biela farba, ktorá využíva len 1 bit na pixel. Vykreslenie rekurentného grafu je limitované knižnicou WPF. Maximálna veľkosť rekurentného grafu využívajúcej 1 bit na pixel je nastavená na signál dlhý 130 000. Výpočet dlhšieho signálu prebehne bez vykreslenia rekurentného grafu a výstupom sú len kvantifikačné parametre.



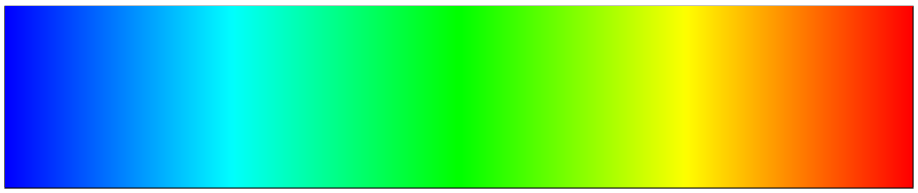
Obr. 3.2: 4bitová farebná paleta zložená zo 16 farieb.

```
1 //4 bity na~pixel
2
3 pfHM = System.Windows.Media.PixelFormats.Indexed4;
4 colors.Add(Media.Color.FromRgb(0, 64, 255));
```

```
colors.Add(System.Windows.Media.Color.FromRgb(0, 128, 255));
colors.Add(System.Windows.Media.Color.FromRgb(0, 191, 255));
colors.Add(System.Windows.Media.Color.FromRgb(0, 255, 255));
colors.Add(System.Windows.Media.Color.FromRgb(0, 255, 191));
colors.Add(System.Windows.Media.Color.FromRgb(0, 255, 128));
colors.Add(System.Windows.Media.Color.FromRgb(0, 255, 64));
colors.Add(System.Windows.Media.Color.FromRgb(0, 255, 0));
colors.Add(System.Windows.Media.Color.FromRgb(64, 255, 0));
colors.Add(System.Windows.Media.Color.FromRgb(128, 255, 0));
colors.Add(System.Windows.Media.Color.FromRgb(191, 255, 0));
colors.Add(System.Windows.Media.Color.FromRgb(255, 255, 0));
colors.Add(System.Windows.Media.Color.FromRgb(255, 191, 0));
colors.Add(System.Windows.Media.Color.FromRgb(255, 128, 0));
colors.Add(System.Windows.Media.Color.FromRgb(255, 64, 0));
colors.Add(System.Windows.Media.Color.FromRgb(255, 0, 0));

BitmapPalette myPalette = new BitmapPalette(colors);
```

Druhá farebná paleta využíva 8 bitov na pixel a celá paleta sa skladá zo 256 farieb (obr. 3.3). Kvôli väčšiemu počtu bitov na pixel táto heatmapa zaberá viac miesta a limit vykreslenia signálu je nižší. Maximálna veľkosť heatmapy využívajúcej 8 bitov na pixel je nastavená na signál dlhý 45 000.



Obr. 3.3: 8bitová farebná paleta zložená z 256 farieb.

```
//8 bitov na~pixel

pfHM = System.Windows.Media.PixelFormats.Indexed8;
for (int i = 0; i < 64; i++)
{
    colors.Add(System.Windows.Media.Color.FromRgb(0, (byte)(i*4), 255));
}
for (int i = 63; i >= 0; i--)
{
    colors.Add(System.Windows.Media.Color.FromRgb(0, 255, (byte)(i * 4)));
}
```

```
for (int i = 0; i < 64; i++)
{
    colors.Add(System.Windows.Media.Color.FromRgb((byte)(i * 4), 255, 0));
}
for (int i = 63; i >= 0; i--)
{
    colors.Add(System.Windows.Media.Color.FromRgb(255, (byte)(i * 4), 0));
}

BitmapPalette myPalette = new BitmapPalette(colors);
```

3.6.2 Enkóder

Táto časť zdrojového kódu má za úlohu uložiť vykreslené grafy. na ukážke je vidieť príklad ukladania rekurentného grafu, ktorý ma inheretne nastavenú čiernobielu farebnú paletu na základe jeho pixelformátu. v prípade vykreslovania heatmapy, kde sa používa pixel formát Indexed4 alebo Indexed8, je potrebné miesto null nastaviť nami vygenerovanú farebnú paletu. Výsledný rekurentný graf alebo heatmapa sa uloží do uživateľom zvoleného priečinka.

```
Mi.BitmapSource image = Mi.BitmapSource.Create
    (width, height, 96, 96, pf, null, rawImage, rawStride);

Mi.TiffBitmapEncoder encoder = new();
FileStream stream = new(
    string.Format("{0}/RP-{1}v{2}.tiff",logFilePath, filePath, v),
    FileMode.Create);
encoder.Compression = Mi.TiffCompressOption.Zip;
encoder.Frames.Add(Mi.BitmapFrame.Create(image));
encoder.Save(stream);

stream.Close();
```

3.7 Kvantifikácia

Výpočet kvantifikačných parametrov je jedným z poslendých krokov celého programu. Hlavnou časťou je výpočet parametru RVD a RVV. Výpočet pola RVV funguje na pricnípe iterovania po vertikálnych čiarach, na ktorých počítame tzv. streaky. Táto iterácia zároveň prebieha aj na diagonálnych čiarach. To znamená, že sa pomocou týchto iterácií počíta dĺžka

čiar zložených z rekurentných bodov a ich početnosť. Zároveň s vypočtom pola RVV sa môžu vykreslovať rekurentné grafy, ak si túto možnosť uživateľ zvolí v grafickom rozhraní.

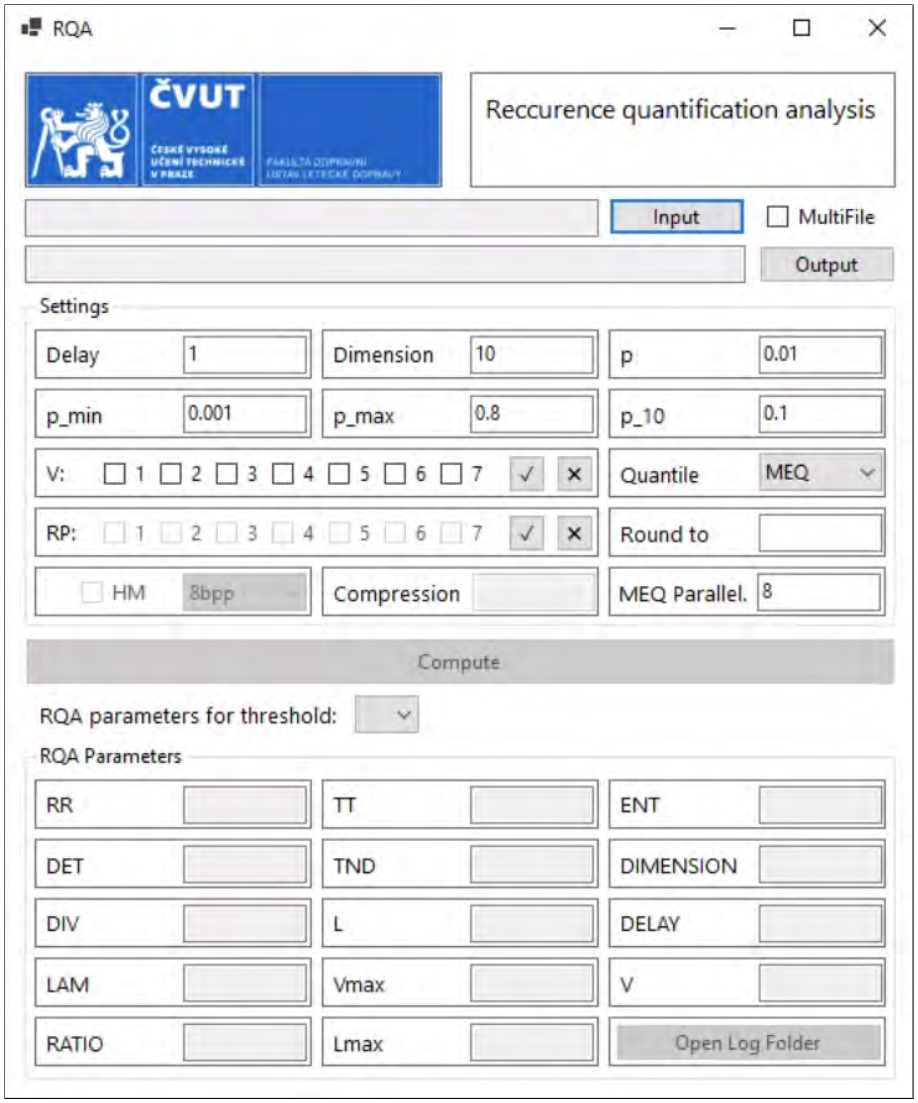
Algoritmus na parameter RVD a RVV fungujú prakticky na rovnakom princípe. Nižšie je uvedený algoritmus pre výpočet listu RVD. Parameter lastDi si udržiava informáciu o predošlom bode a diCount udržiava tzv. streak alebo počet rekurentných bodov, ktoré majú hodnotu 1. Paramter i je v tomto prípade každa diagonála a parameter j je ako hlboko sme do diagonály. Ak hodnota boolu lastDi bola true, tak sa diCount inkrementuje. Následne sa lastDi nastavi na true. na šiestom riadku algoritmu je podmienka, kedy sa nachádzame na konci diagonály.

```
1  if (help <= e && help >= 0)
2  {
3      if (!lastDi) diCount = 1;
4      else diCount++;
5      lastDi = true;
6      if (j == (n2 - i) - 1 && diCount > 1)
7      {
8          rvd[diCount - 1]++;
9          diCount = 0;
10         lastDi = false;
11     }
12 }
13 else
14 {
15     if (lastDi && diCount > 1)
16     {
17         rvd[diCount - 1]++;
18     }
19     diCount = 0;
20     lastDi = false;
21 }
```

3.8 Grafické uživatelské rozhranie

Grafické uživatelské rozhranie (obr. 3.4) bolo vytvorené pomocou knižnice Windows Forms. Vzhľad aplikácie je rozdelený na tri časti:

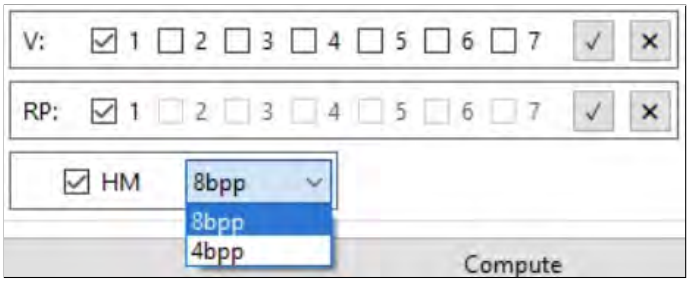
Prvá časť je zameraná na upresnienie cesty k súborom. Prvý riadok slúži na zadanie cesty k vstupnému signálu, ktorý chce užívateľ analyzovať. Vedľa tlačítka Input sa nachádza aj možnosť zvolenia MultiFile. Táto funkcia slúži na spracovanie viacerých signálov za sebou, ak sa nachádzajú v jednom priečinku. Táto funkcionalita bola pridaná, aby užívateľ mohol nechať program bežať a spracovali sa všetky signály, bez potreby opakovaného spúšťania programu. Druhý riadok slúži na upresnenie cesty k priečinku, do ktorého sa budú ukladať výsledné kvantifikačné parametre, vykreslené rekurentné grafy, heatmapy a súbor timer.



Obr. 3.4: Vzhľad grafického uživatelského rozhrania.

Súbor timer obsahuje údaje o dĺžke signálu, kedy program začal signál analyzovať, kedy skončila analýza signálu a výsledná doba trvania analýzy. Kým nie sú zvolené vstupné a výstupné cesty, tlačidlo na spustenie analýzy sa nedá stlačiť.

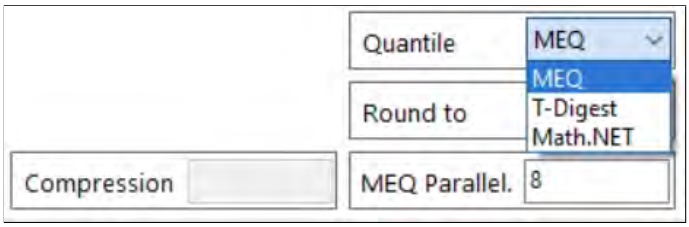
Druhá časť alebo Settings umožňuje užívateľovi zvoliť niekoľko parametrov. v prvom riadku sa nastavujú parametre delay a dimenzia, ktoré určujú, do akého časového oneskorenia a do akej dimenzie sa bude počítať. Parametre p, p-min, p-max a p-10 určujú kvantilové hodnoty, ktoré sa používajú v každej metóde výpočtu kvantilu. v ďalšom riadku si užívateľ môže zvoliť metódu prahovania označenú písmenom „V“ pomocou zaškrtávacích políčok (obr. 3.5). Aplikácia je naprogramovaná tak, že dokáže vypočítať a prahovať pomocou každej metódy prahovania v jednom compute cykle. Pod týmto riadkom sa nachádza identické pole zložené zo zaškrtávacích políčok, označené „RP“ (obr. 3.5), kde si uživateľ môže zvoliť, či chce vykresliť rekurentný graf pre danú metódu prahovania alebo výsledné kvantifikačné parametre sú postačujúce. Poslednými možnostiami vykreslenia je vytvorenie heatmapy a pomocou akej farebnej palety sa bude heatmapa vykreslovať. v rozbalovacej ponuke je možnosť výberu 16 farebnej palety (4bpp) alebo 256 farebnej palety (8bpp), viď obrázok 3.5.



Obr. 3.5: Rozbaľovacia ponuka s možnosťou výberu farebnej palety s 8 bitmi na pixel (8bpp) alebo 4 bitmi na pixel (4bpp).

Posledná sekcia druhej časti sa zameriava na nastavenie metódy výpočtu kvantilu. Predvolenou nastavenou metódou je ME-qunatile metóda. Táto metóda funguje veľmi presne na kratších signáloch. Ak je zvolená táto metóda, užívateľovi sa odomknú textové polia Round to a MEQ Parallel (obr. 3.6). Textové pole ”Round to”slúži na nastavenie

zaokrúhľovania čísel, na ktorých sa vykonáva kvantil. Štandardne sa žiadne zaokrúhľovanie nepoužíva. Keďže je táto metóda kvantilu paralelizovaná, užívateľ si podľa svojho systému môže alokovať počet jadier procesora v textovom poli MEQ Parallel. Štandardne sa používajú všetky jadrá procesora. Druhou metódou je T-Digest (obr. 3.6), pri ktorej sa odomkne textové pole "Compression". Compression určuje počet zhlukov, ktoré táto metóda používa na aproximáciu kvantilu. Treťou metódou je metóda Math.NET (obr. 3.6).



Obr. 3.6: Rozbaľovacia ponuka s možnosťou výberu metódy kvantilu, nastavenia parametrov Compression, Round to a MEQ Parallel.

Tretia časť po dokončení analýzy zobrazuje jedenásť kvantifikačných parametrov a 3 informačné parametre. Ak si užívateľ zvolil viac metôd prahovania, medzi výsledkami sa dá prepínať pomocou rozbaľovacieho zoznamu. Tlačidlo Öpen Log Folder"slúži na priame otvorenie priečinka, kde sa nachádzajú výsledky s vykreslenými rekurentnými grafmi.

3.8.1 Konzola

Aj napriek tomu, že pre konzolovú aplikáciu RQA bolo navrhnuté a vyvinuté grafické užívateľské rozhranie, konzola sa taktiež vždy zapína pri spustení programu. Konzola v tomto prípade funguje ako indikátor priebehu. Pri každom kroku sa vypisuje WorkingSet, ktorý údáva množstvo operačnej pamäte RAM využitej programom RQA práve v danom momente. Konzola najskôr vypíše začiatok analýzy, názov signálu a čas. Následne sa prevedú všetky spomínané iterácie. Nižšie je uvedená komprimovaná verzia výpisu konzoly.

```
Starting RQA for file.csv of length xxx -at: dd.mm.yyyy hh:mm:ss

//Hladanie casoveho oneskorenia
Starting delay search -at: dd.mm.yyyy hh:mm:ss
Returned delay: X -at: dd.mm.yyyy hh:mm:ss and -took: hh:mm:ss.xxxx

//Hladanie dimenzie vnorenia
Starting dimension search -at: dd.mm.yyyy hh:mm:ss
//Pre kazdu dimenziu
Starting maxSum search for dim Y -at: dd.mm.yyyy hh:mm:ss
maxSum for dim Y found -at: dd.mm.yyyy hh:mm:ss and -took: hh:mm:ss.xxxx
Dimension Y -completed at: dd.mm.yyyy hh:mm:ss and -took: hh:mm:ss.xxxx
Returned dimension: Y -at: dd.mm.yyyy hh:mm:ss and -took: hh:mm:ss.xxxx

Starting MinMax cycle (xxx) -at: dd.mm.yyyy hh:mm:ss
MinMax iteration: xxx -at: dd.mm.yyyy hh:mm:ss and -took: hh:mm:ss.xxxx
MinMax cycle finished at: dd.mm.yyyy hh:mm:ss and -took: hh:mm:ss.xxxx
Starting mean and variance search -at: dd.mm.yyyy hh:mm:ss
Mean and variance search iteration: xxx -at:
dd.mm.yyyy hh:mm:ss and -took: hh:mm:ss.xxxx
//Vypočet kvantilu
MEQ gen iteration: xxx -at:
dd.mm.yyyy hh:mm:ss and -took: hh:mm:ss.xxxx
MEQ gen finished -at: dd.mm.yyyy hh:mm:ss and -took: hh:mm:ss.xxxx
Starting MEQ extraction -at: dd.mm.yyyy hh:mm:ss
MEQ extraction finished at: dd.mm.yyyy hh:mm:ss and -took: hh:mm:ss.xxxx

//Vykreslovanie a~kvantifikácia
Starting streak cycle for v1 -at: dd.mm.yyyy hh:mm:ss
Streak iteration for v1: 0 -at: dd.mm.yyyy hh:mm:ss and -took: hh:mm:ss.xxxx
Streak finished for v1 -at: dd.mm.yyyy hh:mm:ss and -took: hh:mm:ss.xxxx
RQA for v1 finished at: dd.mm.yyyy hh:mm:ss and -took: hh:mm:ss.xxxx

Finished RQA for file.csv of length xxx -at:
dd.mm.yyyy hh:mm:ss and -took: hh:mm:ss.xxxx
```

3.9 Zhrnutie kapitoly

Pri vývoji aplikácie nastalo niekoľko problémov spojených s výpočtom dlhších signálov. Hľadanie optimálnej dimenzie vnorenia bolo optimalizované pomocou paralelizácie a rozdelenia výpočtov na menšie časti. Z dôvodu dynamického prahovania na dlhých signáloch aplikácia disponuje tromi výpočtami kvantilu. na vykreslenie rekurentných grafov

a heatmapy je použitá knižnica WPF. Grafické uživateľské rozhranie bolo vytvorené pomocou programovacieho prostredia Microsfot Visual Studio použitím knižnice Windows Forms.

4 Validácia

Validácia prebehla porovnaním výsledných parametrov a rekurentných grafov na 45 signáloch o rôznych dĺžkach. Maximálna dĺžka validačných signálov bola 11 000 z dôvodu limitácie softvérového riešenia RQAcalc. K porovnaniu parametrov bol použitý softvér RQAcalc vo vývojovom prostredí matlab. Tento softvér je schopný spracovávať signály o dlžke 10-12 tisíc vzoriek a následne vykresliť rekurentný graf. Validácia prebehla na prvom systéme s osemjadrovým procesorom i7-9600k (Intel), ktorý je stabilne pretaktovaný na 4,8 Ghz. Systém disponuje 32GB operačnej pamäte RAM s frekvenciou 3200 Mhz.

4.1 Porovnanie kvantifikačných parametrov

Hlavným výstupom aplikácie RQA sú kvantifikačné parametre a rekurentný graf. Cieľom bolo porovnať parametre rôznych signálov pri počítaní s tromi metódami kvantilov. Dáta z aplikácie RQA, ktoré sú ukladané do .json súborov, obsahujúce všetky kvantifikačné parametre, hodnotu kvantilu, metódu prahovania a použitú metóda výpočtu kvantilu, časovú hodnotu trvania celej aplikáciae. Dáta z programu sa v originálnej forme vypisovali do textového súboru. Z dôvodu zjednotenia typu súboru bol zdrojový kód aplikácie RQAcalc poupravený, aby boli parametre ukladané do súboru .json. Pre všetky signály, ktoré sú spomenuté v kapitole validácia, boli použité vstupné parametre: Dodimenzie: 10, Dooneskorenia: 1, p: 0.01, p-min: 0.001, p=max: 0.8 a p=10: 0.1.


```
"S1": [
   {
        "RR": 0.010012200565392055,
        "DET": 0.7904324106547254,
        "DIV": 0.0051813471502590676,
        "LAM": 0.8663106725580719,
        "RATIO": 78.94692135781979,
        "TT": 3.832111137080753,
        "TND": -0.20084404760975158,
        "L": 3.670398086241796,
        "Vmax": 44,
        "Lmax": 193,
```

```
        "ENT": 1.4911750411052909,
        "DIMENZIA": 6,
        "ONESKORENIE": 5,
        "v": 1,
        "e": 0.022978378139952144,
        "Time": 30.8519864
    }
],
```


Každý dátový list pre jednotlivý signál obsahuje 7 metód prahovania, kde nás pre každú metódu zaujímalo 11 kvantifikačných parametrov a hodnota kvantilu. Časový údaj sme pri validácií dát nebrali do úvahy. Z dôvodu jednoduchšieho spracovania a porovnávania týchto parametrov bol využitý externý program. Dáta boli spracované v programovom prostredí PyCharm s programovacím jazykom python. Jedná sa o jednoduchý kód, ktorý zlučuje všetky metódy prahovania pre všetky signály do spoločných listov.

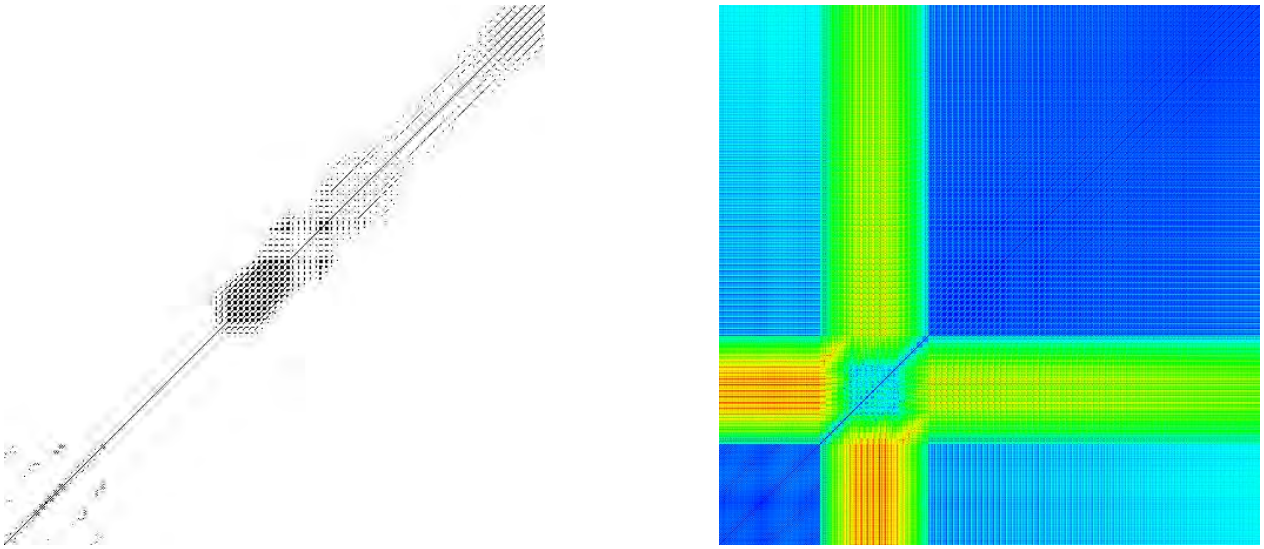
Nižšie je uvedených 5 konrétnych prípadov analýzy výsledných kvantifikačných parametrov a hodnôt kvantilov. Ku každému signálu je priložený rekurentný graf a heatmapa. Rekurentný graf alebo heatmapa nemusia vyzerať v tejto podobe totožne s výsledným vykreslením z dôvodu konvertovania súboru tiff do súboru jpeg. Pri kompresií môžu nastať artefakty na obrázkoch. Parametre dimenzia a oneskorenie nie sú vo výsledných tabuľkách uvedené. Pri všetkých validačných signáloch hodnota dimenzie vnorenia a časového oneskorenia presne zodpovedali Matlabovým výsledkom.

4.1.1 Signál S1

Signál S1 sú EKG dáta o veľkosti 485 vzoriek. Z tabuľky 4.1 je vidieť, že jediná odchýlka nastala pri metóde výpočtu T-digest, kde sa hodnota kvantilu líšila až na šiestom desatinnom čísle. Presnosť tejto metódy je pri nižších dĺžkach signálu menšia ako metódy MEQ a Math.NET. V tabuľke 4.2 je porovnanie výsledných parametrov so softvérom RQAcalc, označeným Matlab. Odchýlky sme zaznamenali len na metóde T-digest pri šiestich parametroch, kde najväčšia odchýlka nastala pri parametri RATIO. Na obrázku 4.1 sú zobrazené vykreslenia rekurentného grafu a heatmapy.

Tabuľka 4.1: Rozdiel kvantilov pri rôznych metódach výpočtu pre signál S1.

	v = 1	Rozdiel hodnôt pri rôznych metódach kvantilu					
Signál	Matlab	MEQ	(%)	T-Digest	(%)	Math.NET	(%)
S1	1,14E-02	0,00	0,00	-1,19E-06	-7,80E-03	0,00	0,00



(a) Rekurentný graf

(b) Heatmapa

Obr. 4.1: Rekurentný graf (a) a heatmapa (b) pre signál S1.

Tabuľka 4.2: Rozdiel kvantifikačných parametrov pre signál S1.

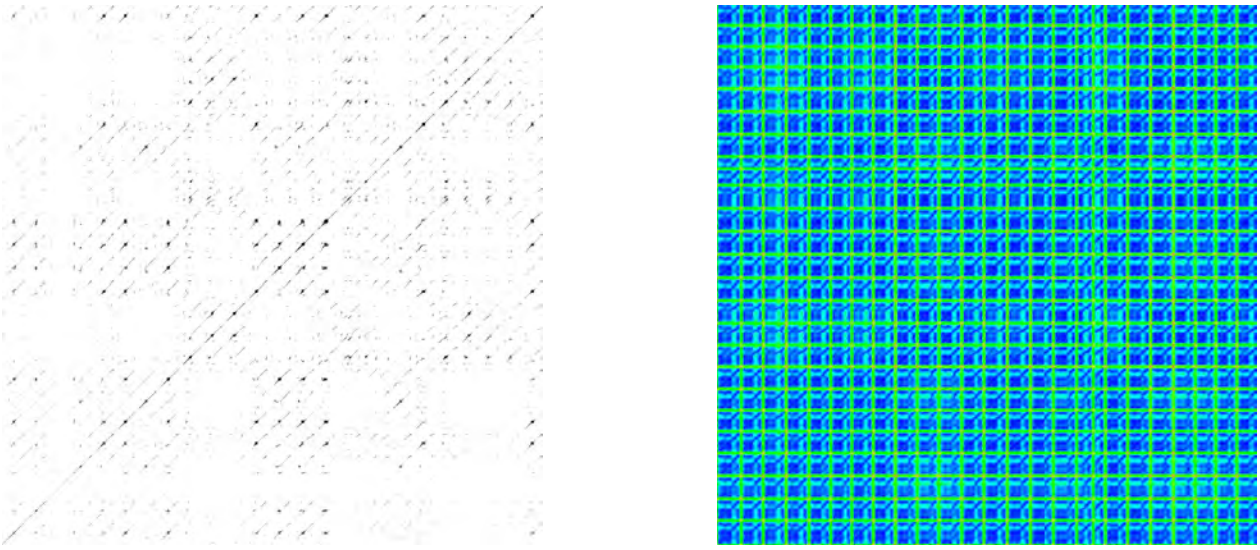
Signál S1 (485 vz.)	v = 1	Metódy kvantilu		
Parameter	Matlab	MEQ	T-Digest	Math.NET
RR	0,0100502 036563417	0,0000000 000000000	-0,0000378 895519560	0,0000000 000000000
DET	0,6201696 512723840	0,0000000 000000000	0,0023469 050190060	0,0000000 000000000
DIV	0,0200000 000000000	0,0000000 000000000	0,0000000 000000000	0,0000000 000000000
LAM	0,4622997 172478790	0,0000000 000000000	0,0003303 678987620	0,0000000 000000000
RATIO	61,707172 558742300	0,0000000 000000000	0,4679200 123854980	0,0000000 000000000
TT	2,7711864 406779600	0,0000000 000000000	-0,0006481 970519001	0,0000000 000000000
TND	-7,764061 757890260	0,0000000 000000000	0,0279579 309227600	0,0000000 000000000
L	3,8934911 242603500	0,0000000 000000000	0,0000000 000000000	0,0000000 000000000
Vmax	3,0000000 000000000	0,0000000 000000000	0,0000000 000000000	0,0000000 000000000
Lmax	50,000000 000000000	0,0000000 000000000	0,0000000 000000000	0,0000000 000000000
ENT	0,9317373 760899260	0,0000000 000000000	0,0000000 000000000	0,0000000 000000000

4.1.2 Signál S2

Signál S2 je dlhší EKG signál o dĺžke 5 525 vzoriek. V tabuľke 4.3 je uvedená nepresnosť v metóde T-Digest, kde sa kvantil líši až na šiestej desatinnej čiarke. Táto odchýlka ale nemala žiadny vplyv na prahovaní vzdialenostnej matice. V tabuľke 4.4 sa nenachádzajú žiadne nenulové hodnoty. Všetky parametre pre všetky metódy kvantilu boli totožné s matlabovým riešením. Na obrázku 4.2 sú zobrazené vykreslenia rekurentného grafu a heatmapy pre signál S2.

Tabuľka 4.3: Rozdiel kvantilov pri rôznych metódach výpočtu pre signál S2.

	v = 1	Rozdiel hodnôt pri rôznych metódach kvantilu					
Signál	Matlab	MEQ	(%)	T-Digest	(%)	Math.NET	(%)
S2	2,30E-02	0,00	0,00	2,74E-05	1,80E-01	0,00	0,00



(a) Rekurentný graf

(b) Heatmapa

Obr. 4.2: Rekurentný graf (a) a heatmapa (b) pre signál S2.

Tabuľka 4.4: Rozdiel kvantifikačných parametrov pre signál S2.

Signál S2 (5525 vz.)	v = 1	Metódy kvantilu		
Parameter	Matlab	MEQ	T-Digest	Math.NET
RR	0,010012200 5653920550	0,0000000 000000000	0,0000000 000000000	0,0000000 000000000
DET	0,790432410 6547254000	0,0000000 000000000	0,0000000 000000000	0,0000000 000000000
DIV	0,005181347 1502590676	0,0000000 000000000	0,0000000 000000000	0,0000000 000000000
LAM	0,866310672 5580719000	0,0000000 000000000	0,0000000 000000000	0,0000000 000000000
RATIO	78,94692135 7819790000	0,0000000 000000000	0,0000000 000000000	0,0000000 000000000
TT	3,832111137 0807530000	0,0000000 000000000	0,0000000 000000000	0,0000000 000000000
TND	-0,20084404 7609751580	0,0000000 000000000	0,0000000 000000000	0,0000000 000000000
L	3,670398086 2417960000	0,0000000 000000000	0,0000000 000000000	0,0000000 000000000
Vmax	44,00000000 0000000000	0,0000000 000000000	0,0000000 000000000	0,0000000 000000000
Lmax	193,0000000 0000000000	0,0000000 000000000	0,0000000 000000000	0,0000000 000000000
ENT	1,491175041 1052909000	0,0000000 000000000	0,0000000 000000000	0,0000000 000000000

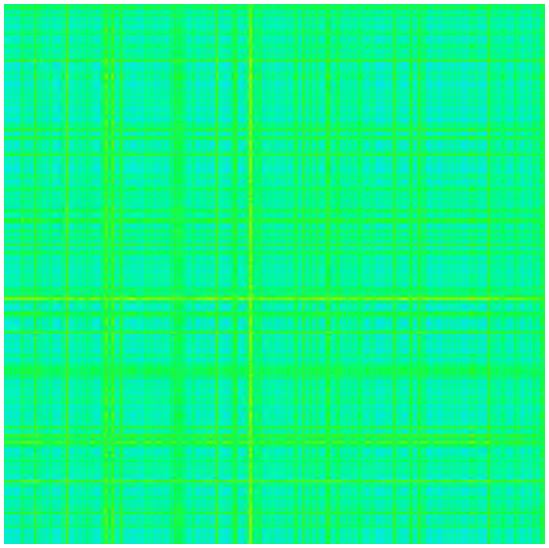
4.1.3 Signál S3

Signál S3 je chaotický signál o dĺžke 9 000 vzoriek. V tabuľke 4.5 sa líšia všetky hodnoty kvantilu od matlabového riešenia. Najväčšiu odchýlku sme zaznamenali v prípade metódy T-Digest, kde sa kvantil líši na šiestej desatinnej čiarke. Najnižšia odchýlka nastala v prípade metódy Math.NET, kde sa kvantil líši až na dvanástej desatinnej čiarke. V tabuľke 4.6 metóda Math.NET nemá žiadnu odchýlku od matlabového riešenia. Odchýlka metódy MEQ sa pohybuje na hranici rozdielu na siedmom desatinnom čísle, pri parametri RATIO na piatom desatinnom čísle. Najväčšiu odchýlku sme zaznamenali na metóde T-Digest, kde sa odchýlka pohybuje na piatom desatinnom čísle, pri parametri RATIO na štvrtom desatinnom čísle. Vykreslenie rekurentného grafu a heatmapy je zobrazené na obrázku 4.3.

Tabuľka 4.5: Rozdiel kvantilov pri rôznych metódach výpočtu pre signál S3.

	v = 1	Rozdiel hodnôt pri rôznych metódach kvantilu					
Signál	Matlab	MEQ	(%)	T-Digest	(%)	Math.NET	(%)
S3	1,50E-01	1,63E-07	1,10E-03	-3,20E-06	-2,10E-02	9,79E-12	6,40E-08

(a) Rekurentný graf



(b) Heatmapa

Obr. 4.3: Rekurentný graf (a) a heatmapa (b) pre signál S3.

Tabuľka 4.6: Rozdiel kvantifikačných parametrov pre signál S3.

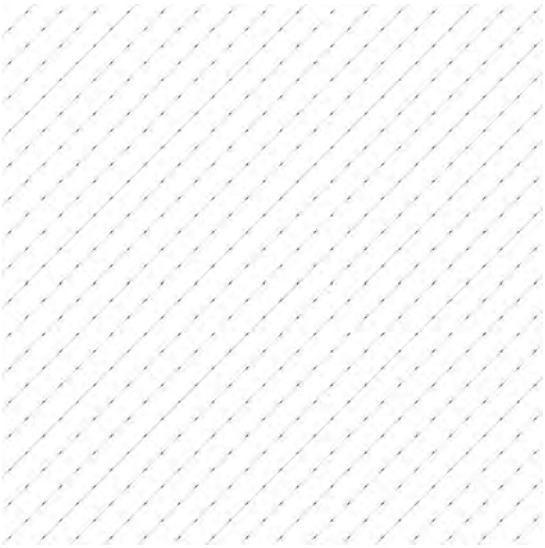
Signál S3 (9000 vz.)	v = 1	Metódy kvantilu		
Parameter	Matlab	MEQ	T-Digest	Math.NET
RR	0,01000112 8642154869	2,4892857407 3103980E-08	-1,593142874 0990905e-06	0,0000000 000000000
DET	0,08314769 5057085330	-2,069544987 6040328E-07	-1,164678261 3894402e-05	0,0000000 000000000
DIV	0,02000000 0000000000	0,0000000000 000000	0,000000000 0000000	0,0000000 000000000
LAM	0,18630076 6364584450	-4,637023515 1638184E-07	-2,680465300 6640144e-06	0,0000000 000000000
RATIO	8,31383117 1675652000	-4,138617722 3046160E-05	0,0001598412 5201029542	0,0000000 000000000
TT	2,12870428 3032819300	0,0000000000 000000	-3,675355747 3456766e-05	0,0000000 000000000
TND	0,02939337 1159600182	-9,846458289 3448280E-07	1,3386690708 924731e-05	0,0000000 000000000
L	2,04894504 4160942300	0,0000000000 000000	1,5014738376 617487e-05	0,0000000 000000000
Vmax	8,00000000 0000000000	0,0000000000 000000	0,000000000 0000000	0,0000000 000000000
Lmax	5,00000000 0000000000	0,0000000000 000000	0,000000000 0000000	0,0000000 000000000
ENT	0,19763123 9498894250	0,0000000000 000000	4,6020202403 296295e-05	0,0000000 000000000

4.1.4 Signál S4

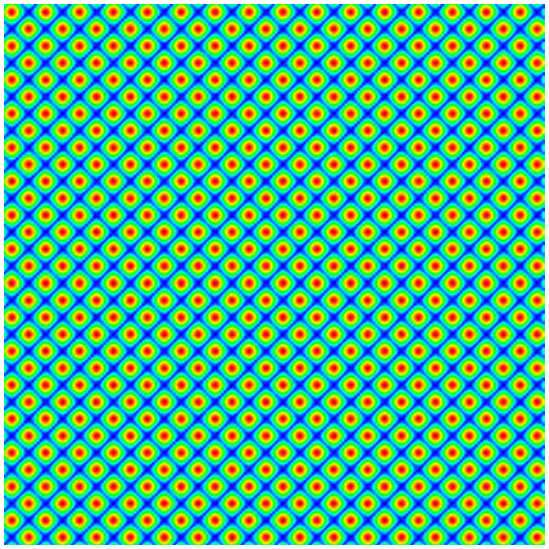
Signál S4 je periodický signál o dĺžke 10 054 vzoriek. Odchýlku kvantilov v tabuľke 4.7 sme zaznamenali pri každej metóde kvantilu. Najmenšiu odchýlku sme zaznamenali pri metóde MEQ na jedenástej desatinnej čiarke. Najväčšiu odchýlku mala metóda T-Digest na siedmom desatinnom čísle v porovnaní s matlabovým riešením. Aj napriek odchýlkám vo výpočte kvantilov sa parametre líšili len pre metódu T-Digest (Tabuľka 4.8). Odchýlka parametrov sa v tomto prípade pohybovala v okolí tretieho desatinného čísla. Metódy MEQ a Math.NET nezaznamenali žiadnu odchýlku vo výsledkoch. Na obrázku 4.4 je zobrazený rekurentný graf a heatmapa pre signál S4.

Tabuľka 4.7: Rozdiel kvantilov pri rôznych metódach výpočtu pre signál S4

	v = 1	Rozdiel hodnôt pri rôznych metódach kvantilu					
Signál	Matlab	MEQ	(%)	T-Digest	(%)	Math.NET	(%)
S4	6,83E-03	4,43E-11	2,90E-07	5,19E-07	3,40E-03	-9,97E-09	-6,50E-05



(a) Rekurentný graf



(b) Heatmapa

Obr. 4.4: Rekurentný graf (a) a heatmapa (b) pre signál S4.

Tabuľka 4.8: Rozdiel kvantifikačných parametrov pre signál S4

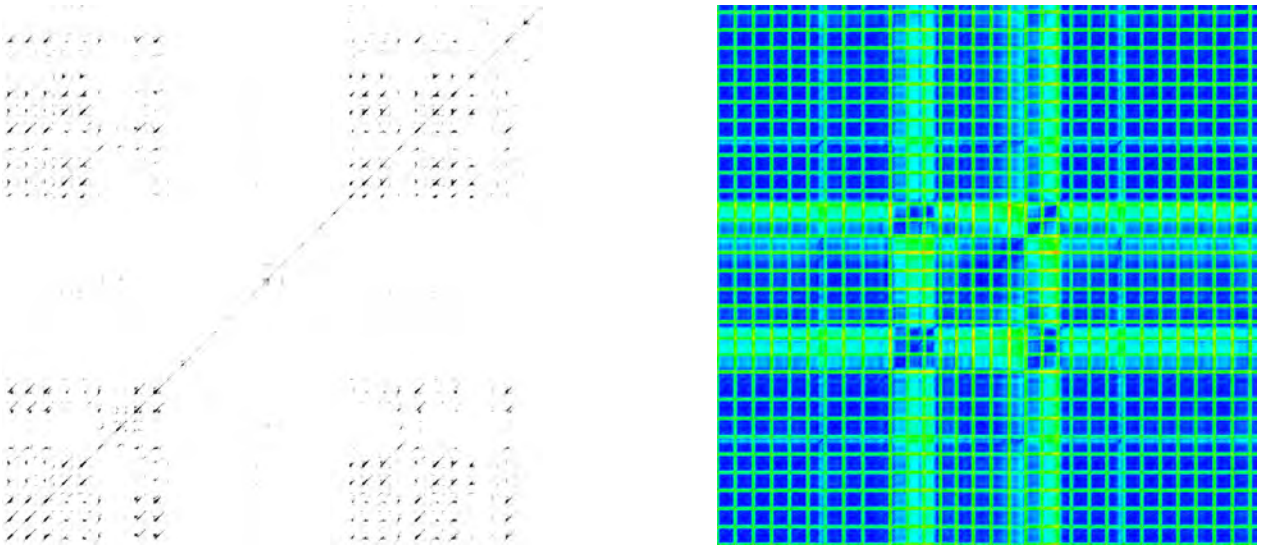
Signál S4 (10054 vz.)	v = 1	Metódy kvantilu		
Parameter	Matlab	MEQ	T-Digest	Math.NET
RR	0,01000099 8898506540	0,0000000 000000000	7,160562770 9192540e-07	0,000000 000000000
DET	0,99996817 8455223100	0,0000000 000000000	1,990982371 3026853e-06	0,0000000 000000000
DIV	9,97307270 370001e-05	0,0000000 000000000	0,000000000 0000000000	0,0000000 000000000
LAM	0,98951678 9842562900	0,0000000 000000000	2,739232291 992799e0-06	0,0000000 000000000
RATIO	99,9868301 7598866000	0,0000000 000000000	-0,00695932 8006985288	0,0000000 000000000
TT	6,23545261 9969671000	0,0000000 000000000	0,000385559 2964212562	0,0000000 000000000
TND	0,00860658 3181077734	0,0000000 000000000	8,849774066 1462600e-06	0,0000000 000000000
L	85,4500339 9048267000	0,0000000 000000000	-0,00823280 1017925340	0,0000000 000000000
Vmax	23,0000000 0000000000	0,0000000 000000000	0,000000000 0000000000	0,0000000 000000000
Lmax	10027,0000 0000000000	0,0000000 000000000	0,000000000 0000000000	0,0000000 000000000
ENT	4,72374118 0407347000	0,0000000 000000000	0,001089765 7084791845	0,0000000 000000000

4.1.5 Signál S5

Signál S5 je chaotický signál o dĺžke 10 329 vzoriek. V tabuľke 4.9 je rozdiel medzi kvantilmi len v prípade metódy T-Digest na siedmom desatinnom čísle. To odrzkadluje odchýlky v kvantifikačných parametroch v tabuľke 4.10. Pomocou tejto metódy na tomto signále boli najväčšie odchýlky, ktoré sa pohybovali na druhom desatinnom čísle, pri parametri RATIO na prvom desatinnom čísle. Na obrázku 4.5a je zobrazený vykreslený rekurentný graf a heatmapa pre signál S5

Tabuľka 4.9: Rozdiel kvantilov pri rôznych metódach výpočtu pre signál S5

	v = 1	Rozdiel hodnôt pri rôznych metódach kvantilu					
Signál	Matlab	MEQ	(%)	T-Digest	(%)	Math.NET	(%)
S5	1,53E-02	0,00	0,00	-3,75E-07	2,60E-03	0,00	0,00



(a) Rekurentný graf

(b) Heatmapa

Obr. 4.5: Rekurentný graf (a) a heatmapa (b) pre signál S6. Nenulové hodnoty sú zvýraznené červenou farbou.

Tabuľka 4.10: Rozdiel kvantifikačných parametrov pre signál S5

Signál S5 (10329 vz.)	v = 1	Metódy kvantilu		
Parameter	Matlab	MEQ	T-Digest	Math,NET
RR	0,01005127 1558328561	0,0000000 000000000	-0,000103784 05414276579	0,0000000 000000000
DET	0,89404052 1513270600	0,0000000 000000000	-0,000661732 62448754320	0,0000000 000000000
DIV	0,00347222 2222222222	0,0000000 000000000	0,0000000000 00000000000	0,0000000 000000000
LAM	0,93226057 5685207100	0,0000000 000000000	-0,000519139 07380451980	0,0000000 000000000
RATIO	88,9480018 8464332000	0,0000000 000000000	0,8614890559 44332400000	0,0000000 000000000
TT	7,17605153 5771803000	0,0000000 000000000	-0,030881529 34544297000	0,0000000 000000000
TND	0,09596577 3632824920	0,0000000 000000000	-0,001317171 92976937850	0,0000000 000000000
L	6,40693213 7454074000	0,0000000 000000000	-0,024465541 54148295000	0,0000000 000000000
Vmax	113,000000 0000000000	0,0000000 000000000	0,0000000000 00000000000	0,0000000 000000000
Lmax	288,000000 0000000000	0,0000000 000000000	0,0000000000 00000000000	0,0000000 000000000
ENT	2,26566264 5385392000	0,0000000 000000000	-0,004281581 70984316300	0,0000000 000000000

4.2 Porovnanie metód výpočtu kvantilu

Pri prvej metóde prahovania je potrebný výpočet kvantilu na veľmi veľkých dátach. Výsledná hodnota kvantilu priamo ovplyvňuje väčšinu výsledných kvantifikačných parametrov, keďže sa podľa tejto hodnoty prahuje vzdialenostná matica. Preto bol kladený dôraz na čo najvyššiu presnosť, než len rýchlosť. Tabuľka 4.11 poukazuje na výsledky kvantilov pre každý validačný signál s použitím jednotlivých metód.

Tabuľka 4.11: Hodnoty kvantilov s použítím rôznych metód výpočtu na všetkýc validačných signáloch. Nenulové hodnoty sú zvýraznené červenou farbou.

	v = 1	**Rozdiel hodnôt pri rôznych metódach kvantilu**					
Signál	**Matlab**	**MEQ**	[%]	**T-Digest**	[%]	**M.NET**	[%]
AS1	2.51E-02	0.00E+00	0.0E+00	-2.88E-05	-1.9E-01	0.00E+00	0.0E+00
AS2	2.30E-02	0.00E+00	0.0E+00	2.74E-05	1.8E-01	0.00E+00	0.0E+00
AS3	2.39E-02	0.00E+00	0.0E+00	1.59E-07	1.0E-03	0.00E+00	0.0E+00
AS4	3.50E-02	0.00E+00	0.0E+00	-1.10E-05	-7.2E-02	0.00E+00	0.0E+00
AS5	2.35E-02	0.00E+00	0.0E+00	-1.55E-05	-1.0E-01	0.00E+00	0.0E+00
M1	2.90E-02	0.00E+00	0.0E+00	-1.03E-05	-6.7E-02	0.00E+00	0.0E+00
M2	2.62E-02	0.00E+00	0.0E+00	-9.50E-07	-6.2E-03	0.00E+00	0.0E+00
M3	1.50E-01	1.63E-07	1.1E-03	-3.20E-06	-2.1E-02	9.79E-12	6.4E-08
M4	2.49E-02	0.00E+00	0.0E+00	0.00E+00	0.0E+00	0.00E+00	0.0E+00
M5	1.19E-01	2.30E-08	1.5E-04	-2.47E-06	-1.6E-02	1.46E-11	9.6E-08
M6	7.23E-03	4.72E-11	3.1E-07	-2.56E-06	-1.7E-02	-2.44E-08	-1.6E-04
M7	8.96E-02	2.46E-11	1.6E-07	2.25E-05	1.5E-01	-1.47E-05	-9.6E-02
NS1	1.70E-02	0.00E+00	0.0E+00	1.14E-05	7.5E-02	0.00E+00	0.0E+00
NS2	1.54E-02	0.00E+00	0.0E+00	0.00E+00	0.0E+00	0.00E+00	0.0E+00
NS3	1.86E-02	0.00E+00	0.0E+00	0.00E+00	0.0E+00	0.00E+00	0.0E+00
NS4	1.58E-02	0.00E+00	0.0E+00	-3.19E-05	-2.1E-01	0.00E+00	0.0E+00
NS5	1.75E-02	0.00E+00	0.0E+00	0.00E+00	0.0E+00	0.00E+00	0.0E+00
RN1	2.20E-01	1.75E-07	1.1E-03	-2.93E-06	-1.9E-02	0.00E+00	0.0E+00

RN2	2.12E-01	6.09E-08	4.0E-04	-5.83E-06	-3.8E-02	0.00E+00	0.0E+00
S01	2.95E-03	0.00E+00	0.0E+00	0.00E+00	0.0E+00	0.00E+00	0.0E+00
S02	1.14E-02	0.00E+00	0.0E+00	-1.19E-06	-7.8E-03	0.00E+00	0.0E+00
S03	2.91E-02	0.00E+00	0.0E+00	0.00E+00	0.0E+00	0.00E+00	0.0E+00
S04	1.83E-02	0.00E+00	0.0E+00	0.00E+00	0.0E+00	0.00E+00	0.0E+00
S05	3.56E-03	0.00E+00	0.0E+00	0.00E+00	0.0E+00	0.00E+00	0.0E+00
S06	1.83E-02	0.00E+00	0.0E+00	0.00E+00	0.0E+00	0.00E+00	0.0E+00
S07	1.03E-02	-1.23E-16	-8.1E-13	2.25E-05	1.5E-01	-1.23E-16	-8.1E-13
S08	6.64E-03	0.00E+00	0.0E+00	6.98E-07	4.6E-03	0.00E+00	0.0E+00
S09	6.39E-02	-8.33E-17	-5.5E-13	-3.35E-06	-2.2E-02	-3.79E-05	-2.5E-01
S10	7.81E-03	0.00E+00	0.0E+00	0.00E+00	0.0E+00	0.00E+00	0.0E+00
Sn1	6.25E-03	5.88E-07	3.9E-03	-4.83E-06	-3.2E-02	-1.99E-13	-1.3E-09
Sn2	5.39E-03	-1.60E-10	-1.0E-06	-4.88E-07	-3.2E-03	-1.71E-07	-1.1E-03
Sn3	8.96E-03	-7.97E-12	-5.2E-08	6.60E-07	4.3E-03	-3.04E-07	-2.0E-03
Sn4	7.86E-03	-1.26E-10	-8.2E-07	-3.75E-07	-2.5E-03	-1.98E-09	-1.3E-05
Sn5	7.43E-03	1.39E-11	9.1E-08	1.25E-06	8.2E-03	-9.78E-09	-6.4E-05
Sn6	7.13E-03	3.33E-08	2.2E-04	-3.44E-07	-2.3E-03	-6.81E-11	-4.5E-07
Sn7	7.13E-03	6.01E-09	3.9E-05	-3.57E-07	-2.3E-03	-7.46E-11	-4.9E-07
Sn8	6.97E-03	2.77E-09	1.8E-05	5.57E-07	3.6E-03	1.56E-10	1.0E-06
Sn9	6.98E-03	-1.65E-10	-1.1E-06	-2.33E-07	-1.5E-03	-2.47E-09	-1.6E-05
Sn10	6.83E-03	4.43E-11	2.9E-07	5.19E-07	3.4E-03	-9.97E-09	-6.5E-05
T1	8.65E-05	5.67E-13	3.7E-09	-5.40E-09	-3.5E-05	-4.29E-10	-2.8E-06
WP1	2.90E-03	0.00E+00	0.0E+00	0.00E+00	0.0E+00	0.00E+00	0.0E+00
WP2	1.26E-02	0.00E+00	0.0E+00	6.99E-06	4.6E-02	0.00E+00	0.0E+00
WP3	5.32E-03	0.00E+00	0.0E+00	0.00E+00	0.0E+00	0.00E+00	0.0E+00
WP4	3.05E-03	0.00E+00	0.0E+00	0.00E+00	0.0E+00	0.00E+00	0.0E+00
WP5	1.53E-02	0.00E+00	0.0E+00	0.00E+00	0.0E+00	0.00E+00	0.0E+00

Pomocou porovnania výsledkov z jednotlivých metód výpočtu kvantilu s hodnotami matlabového riešenia vyšla metóda MEQ ako najpresnejšia z týchto troch. Medzi 19 signálmi zo všetkých validačných signálov bol nenulový rozdiel hodnôt kvantilu s najvyššou odchýlkou na siedmom desatinnom čísle. Pri metóde Math.NET bola síce nenulová hodnota len na 18 signáloch, ale s najvyššou odchýlkou na piatom desatinnom čísle.

4.3 Zhrnutie kapitoly

Z vyššie uvedených dát vyplýva, že odchýlky výsledných parametrov pri všetkých metódach kvantilu sú dostatočne malé na to, aby sme mohli považovať aplikáciu RQA za presnú. Avšak niektoré metódy sa prejavujú presnejšie. Pomocou metódy MEQ sme dosiahli najväčšej presnosti výsledných parametrov.

5 Optimalizácia

Cieľom bolo dosiahnuť spracovatelnosť signálu dlhšieho ako 100 000. Pri takto dlhom signále bolo potrebné myslieť na zaťaženie operačnej pamäte RAM a využitie procesora. Kvôli vyššej výpočetnej náročnosti bola zvolená vyššia trieda procesorov. Pri výkonnostných porovnávaniach boli použité 3 rozdielne osobné počítače s operačným systémom Windows 10 a jeden serverový počítač s operačným systémom Windows Server 2019:

1. VRPC - Osemjadrový procesor i7-9600k (Intel, Santa Clara, USA) disponujúci 8 logickými procesormi, ktorý je stabilne pretaktovaný na 4,8Ghz. Systém disponuje 32GB operačnej pamäte RAM s frekvenciou 3200 Mhz.

2. MPC - Osemjadrový procesor i7-10700k (Intel, Santa Clara, USA) disponujúci 16 logickými procesormi, ktorý je stabilne pretaktovaný na 4,7Ghz. Systém disponuje 64GB operačnej pamäte RAM s frekvenciou 2666 Mhz.

3. AMD - Osemjadrový procesor AMD Ryzen 2700 (AMD, Sunnyvale, USA) disponujúci 16 logickými procesormi, ktorého základná rýchlosť je 3,3Ghz. Systém disponuje 16GB operačnej pamäte RAM s frekvenciou 2666 Mhz.

4. SRVR - Dva šesťjadrové procesory Xeon E5-2620 v3 (Intel, Santa Clara, USA) disponujúce 24 logickými procesormi, ktorých základná rýchlosť je 2,4Ghz. Systém disponuje 128GB operačnej pamäte RAM s frekvenciou 2133 Mhz.

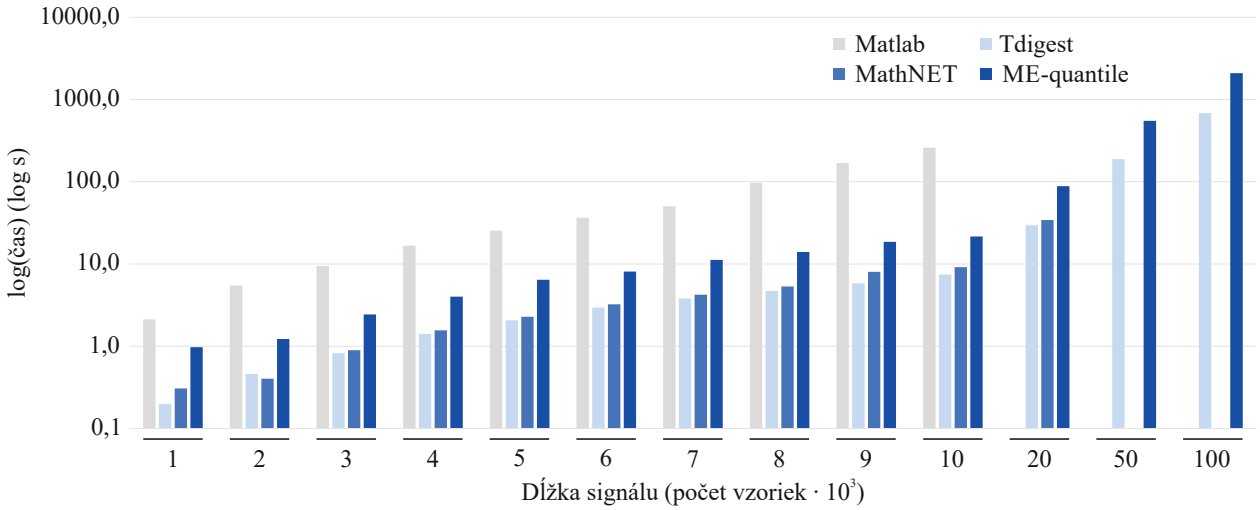
Treba podotknúť, že štvrtý systém je server, ktorého matičná doska disponuje dvomi slotmi na procesor. Pri testovaní aplikácie RQA sme zistili, že paralelizácia, ktorá je použítá v aplikácií RQA, v momentálnom stave nedokáže prerozdeliť prácu medzi dvomi soketmi a teda aplikácia využívala len polovicu dostupného výkonu.

Na optimalizačnú validáciu bolo použitých 13 signálov o dĺžke tisíc až stotisíc vzoriek. Pre všetky signály, ktoré sú spomenuté v kapitole optimalizácia, boli použité vstupné parametre: Dodimenzie: 10, Dooneskorenia: 1, p: 0.01, p-min: 0.001, p=max: 0.8 a p=10: 0.1.

5.1 Rýchlostná optimalizácia

5.1.1 Kvantily

Najväčšie časové rozdiely spôsobuje voľba metódy výpočtu kvantilu. Každá z týchto metód funguje na inom princípe. Časy boli porovnané s dĺžkou trvania matlabového riešenia. Keďže matlab dokáže naprahovať vzdialenostnú maticu len pre jednu metódu prahovania, použité bolo prahovanie číslo jedna, kde je na samotné prahovanie porebný výpočet kvantilu. Časové porovnanie prebehlo na systéme VRPC s osemjadrovým procesorom i7-9600k s taktovaciou frekvenciou 4,8 Ghz a 32GB operačnej pamäte RAM.



Obr. 5.1: Graf v závislosti času (logaritmická mierka o základe 10) na dĺžke signálu pri použití rôznych metód výpočtu kvantilu v porovnaní s Matlabom. Použitý bol systém VRPC - Osemjadrový procesor i7-9600k s taktovaciou frekvenciou 4,8 Ghz a 32GB operačná pamäť.

Graf na obrázku 5.1 poukazuje na čas, ktorý je potrebný na spracovanie jednotlivých dĺžok signálov. Najrýchlejšiou metódou výpočtu kvantilu je v každom prípade T-Digest, ktorá je priemerne 10-krát rýchlejšia ako matlabové riešenie. Pri vyšších signáloch porovnávame len metódy, ktoré sú schopné danú dĺžku signálu spracovať. Aj v prípade metódy MEQ je čas spracovania približne o polovicu kratší s porovnaním matlabového riešenia.

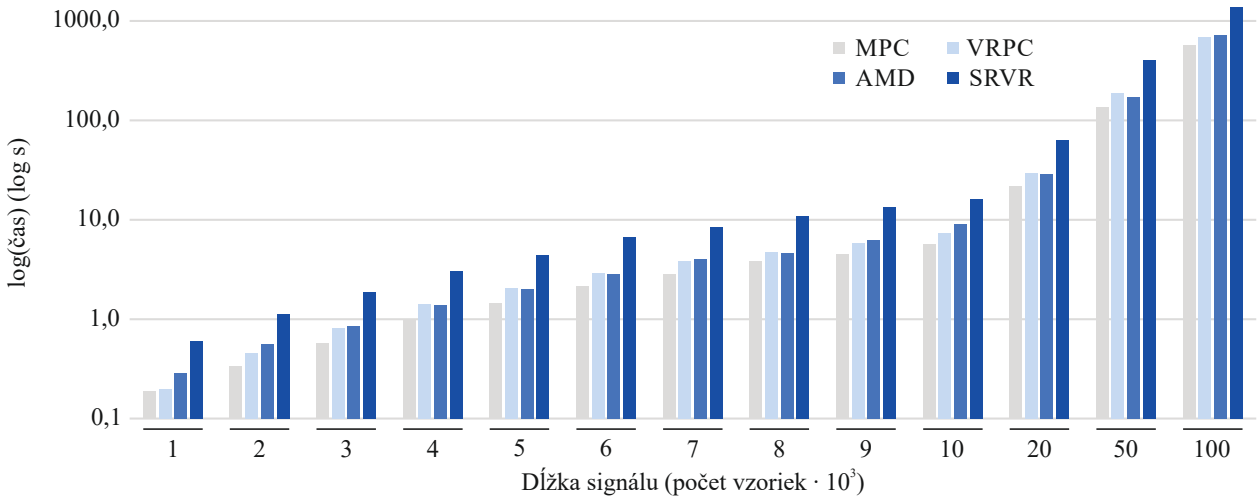
5.1.2 Kvantily na rôznych počítačoch

Analýza 13 signálov prebehla na štyroch systémoch s rôznou konfiguráciou (vyššie spomínané konfigurácie: VRPC, MPC, AMD a SRVR). Na grafoch sa porovnávali výhody a nevýhody jednotlivých metód vypočítavania kvantilu. Dobu behu programu avšak ovplyvňujú aj iné faktory, ako je napríklad paralelný výpočet optimálnej dimenzie vnorenia alebo kvantifikácia a vykreslovanie rekurentných grafov. Časti programu, ktoré nie sú paralelizované, môžu spôsobiť dlhšiu dobu behu programu na systémových konfiguráciach s nižšiou taktovaciou frekvenciou, aj napriek väčšiemu počtu logických procesorov.

Metóda T-Digest benefituje z väčšieho počtu logických procesorov, avšak taktovacia frekvencia nie je zanedbateľná pri behu programu. Z porovnania výsledných časov na Obr 5.2 vyplýva, že MPC s 16 logickými procesormi a taktovaciou frekvenciou 4,7 Ghz bol rýchlejší než VRPC s 8 logickými procesormi a taktovaciou frekvenciou 4,8 Ghz. Napriek tomu konfigurácia SRVR s 12 logickými procesormi bola znateľne pomalšia ako VRPC, z dôvodu výrazne nižšej taktovacej frekvencie.

Metóda MEQ využívala pri tejto analýze maximálny počet logických procesorov. Z obrázku 5.3 je vidieť, že rýchlosť behu aplikácie závisí na taktovacej frekvencií ako počte dostupných logických procesorov. Pri porovnaní konfigurácií MPC a VRPC s približne rovnakou taktovaciou frekvenciou, kde konfigurácia MPC disponuje dva krát väčším počtom logických procesorov, počet jadier nezohráva veľku rolu. Konfigurácie MPC a AMD disponujú rovnakým počtom logických procesorov, ale inou taktovaciou frekvenciou, čo zapričiňuje približne dvojnásobnú dobu behu programu.

Metóda Math.NET nie je paralelizovaná, všetky hodnoty si ukladá do operačnej pamäte RAM a pomocou jedného logického procesora počíta hodnotu kvantilu. Preto časť výpočtu kvantilu nezávisí na počte logických procesorov, ale taktovacej frekvencií daného procesora. Porovnaním konfigurácií MPC a VRPC je z obrázku 5.4 vidieť, že aj napriek dvojnásobnému počtu logických jadier sú časy doby behu programu takmer totožné.



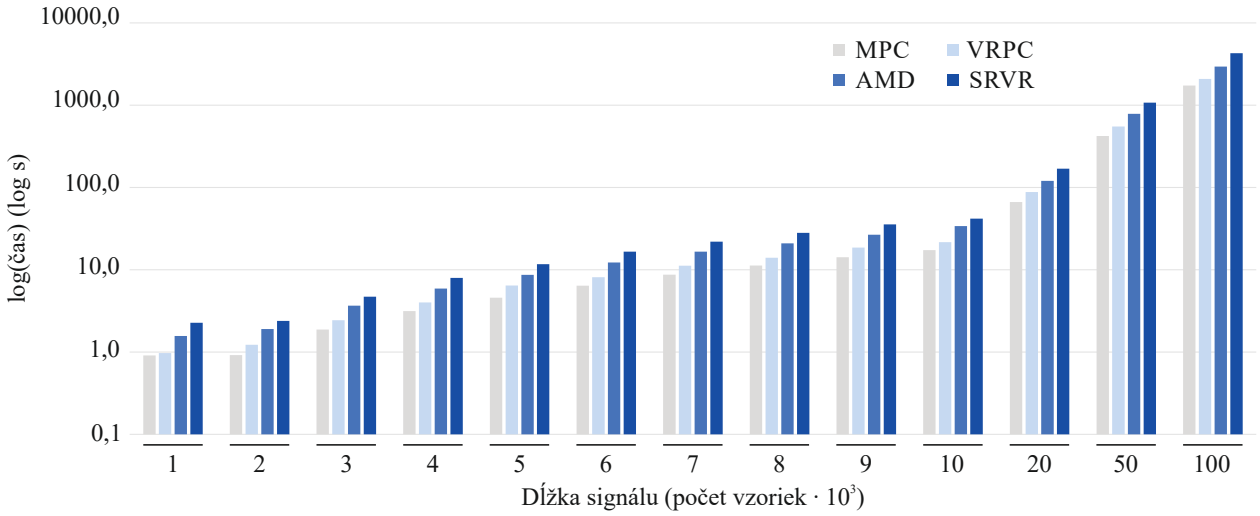
Obr. 5.2: Graf metódy T-Digest v závislosti času (logaritmická mierka o základe 10) na dĺžke signálu použitím štyroch rôznych systémových konfiguráciach: VRPC - 8 jadrový procesor i7-9600k s taktovaciou frekvenciou 4,8 Ghz a 32GB operačná pamäť. MPC - 8/16 jadrový procesor i7-10700k s taktovaciou frekvenciou 4,7 Ghz a 64GB operačná pamäť. AMD - 8/16 jadrový procesor AMD Ryzen 2700 s taktovaciou frekvenciou 3,3 Ghz a 16GB operačná pamäť a SRVR - dva 6/12 jadrové procesory Xeon E5-2620 v3 s taktovaciou frekvenciou 2,4 Ghz a 128GB operačná pamäť

5.1.3 Dĺžka signálu 100 000 a viac

Pomocou metód T-Digest a MEQ je možné spracovanie dát väčších ako 100 000. Na obrázku 5.5 je zobrazená časová naročnosť metódy T-Digest na signáloch, kde prvý začína na dĺžke 62 500 a každý další má dvojnásobnú dĺžku toho predchádzajúceho. Z grafu vieme predpokladať, že pri dvojnásobnej dĺžke signálu bude približne štvornásobná doba behu programu.

5.2 Operačná pamäť RAM

Využitie operačnej pamäte RAM zavisí hlavne od metódy výpočtu kvantilu a rôznorodosti dát. Porovnanie prebehlo na dvoch signáloch, ktoré mali dĺžku 20 000. Signál č.1 sa skladal z hodnôt v rozmedzí 1 až 1000. Signál č.2 sa skladal z hodnôt 1 až 100 000.



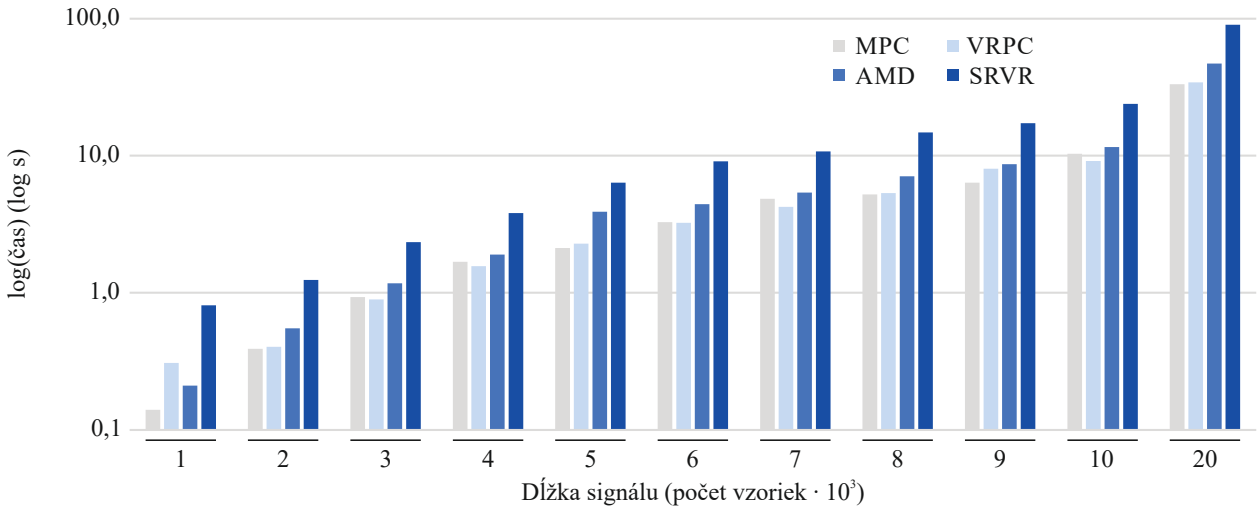
Obr. 5.3: Graf metódy MEQ v závislosti času (logaritmická mierka o základe 10) na dĺžke signálu použitím štyroch rôznych systémových konfiguráciach: VRPC - 8 jadrový procesor i7-9600k s taktovaciou frekvenciou 4,8 Ghz a 32GB operačná pamäť. MPC - 8/16 jadrový procesor i7-10700k s taktovaciou frekvenciou 4,7 Ghz a 64GB operačná pamäť. AMD - 8/16 jadrový procesor AMD Ryzen 2700 s taktovaciou frekvenciou 3,3 Ghz a 16GB operačná pamäť a SRVR - dva 6/12 jadrové procesory Xeon E5-2620 v3 s taktovaciou frekvenciou 2,4 Ghz a 128GB operačná pamäť.

Porovnanie prebehlo na systémovej konfigurácií VRPC s maximálnym nastavením paralelizácie. Nevýhodou výkonostnej analýzy v platforme Microsoft Visual Studio, že graf operačnej pamäte ovplyvňuje garbage collector. Môže sa stať, že dva rovnaké signály budú mať malé odchyľky v grafe využitia operačnej pamäte RAM.

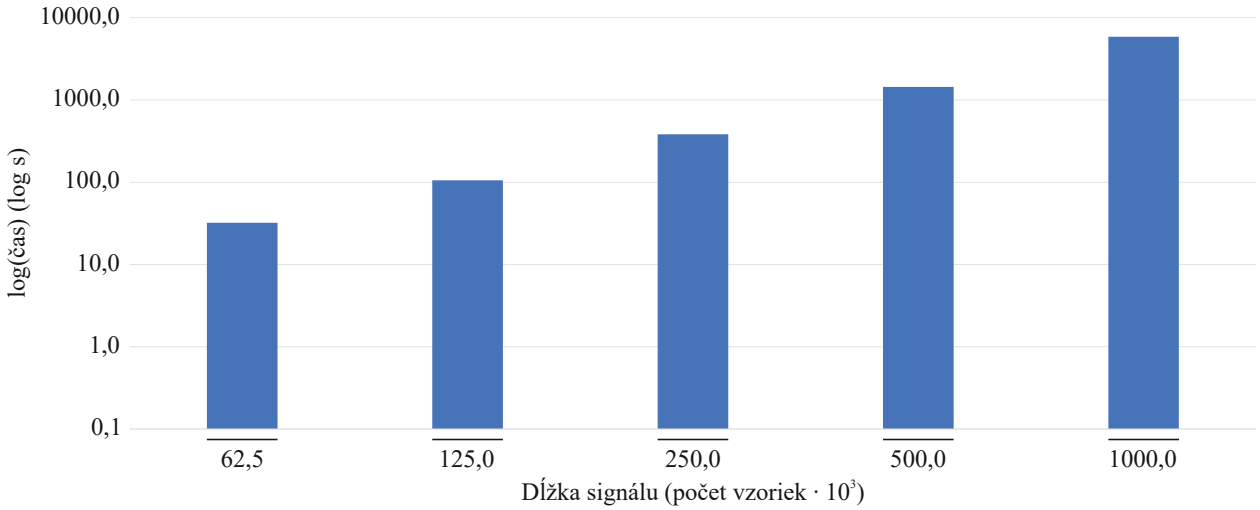
Pri porovnaní obrázku 5.6 a obrázku 5.7 je rozdiel v grafoch využitia operačnej pamäte taký, že je ho možné pripísať k iným rozhodnutiam garbage collectora. Pri porovnaní obrázku 5.8 a obrázku 5.9 sa ukazuje významný rozdiel využitia operačnej pamäte RAM. V prípade nedostatku operačnej pamäte v systémovej konfigurácií je možné pred spustením programu zvoliť nižšiu paralelizáciu alebo nastaviť hodnotu zaokruhlovania. Tento rozdiel je spôsobený väčším počtom unikátných dát v signále č.2, ktorý spôsobuje väčšie množstvo

klúčov SortedDictionary. Táto skutočnosť spôsobuje nie len väčšie využitie operačnej pamäte, ale aj dlhšiu dobu behu programu. Spomalenie zapríčíňuje hlavne metódy ContainsKey a Add triedy SortedDictionary, ktoré sú v priebehu online algoritmu MEQ volané pri každej iterácií. Väčšie množstvo klúčov v SortedDictionary znamená, že metódy ContainsKey a Add musia prehladávať väčšie množstvo dát. Toto prehladávanie má výpočetnú zložitosť $O(log\ n)$.

Pri metóde T-Digest sa neprejavuje žiadny rozdiel v porovnaní signálu č.1 (Obr. 5.10) a signálu č.2 (Obr. 5.11). Hlavný parameter ovplyvňujúci využitie operačnej pamäte je paramter *compression*, ktorý sa volí pred začiatkom výpočtu.



Obr. 5.4: Graf metódy Math.NET v závislosti času (logaritmická mierka o základe 10) na dĺžke signálu použitím štyroch rôznych systémových konfiguráciach: VRPC - 8 jadrový procesor i7-9600k s taktovaciou frekvenciou 4,8 Ghz a 32GB operačná pamäť. MPC - 8/16 jadrový procesor i7-10700k s taktovaciou frekvenciou 4,7 Ghz a 64GB operačná pamäť. AMD - 8/16 jadrový procesor AMD Ryzen 2700 s taktovaciou frekvenciou 3,3 Ghz a 16GB operačná pamäť a SRVR - dva 6/12 jadrové procesory Xeon E5-2620 v3 s taktovaciou frekvenciou 2,4 Ghz a 128GB operačná pamäť.



Obr. 5.5: Graf v závislosti času (logaritmická mierka o základe 10) na dĺžke signálu pomocou metódy T-Digest pri hodnotách vyšších ako 62 500. Použitý bol systém VRPC - Osem jadrový procesor i7-9600k s taktovaciu frekvenciou 4,8 Ghz a 32GB operačná pamäť.

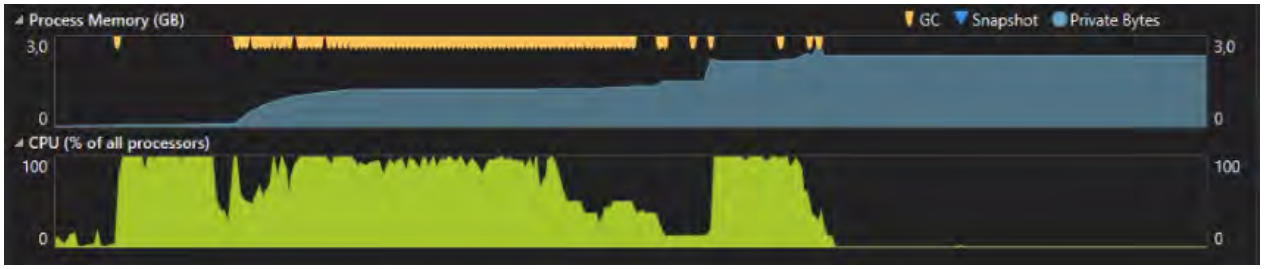
Obr. 5.6: Využitie operačnej pamäte s použitím metódy Math.NET na signále č.1 dlhom 20 000, ktorý sa skladá z hodnôt 1 až 1000.

5.2.1 Vykreslenie rekurentného grafu a heatmapy

Každé vykreslenie zaberá maximálne 2GB operačnej pamäte RAM. To znamená, že ak si užívateľ zvolí vykreslenie všetkých siedmich rekurentných grafov a vykreslenie heatmapy buď v 16 farebnej alebo 256 farebnej palete, celková operačná pamäť nepresiahne 16GB operačnej pamäte RAM. Taketé množstvo operačnej pamäte budú tieto vykreslenia potrebovať len pri najvyšších možných vykresleniach. Ak užívateľ nedisponuje takým množstvom operačnej pamäte, môže rekurentné grafy vykreslovať po jednom na úkor dlhšieho času spracovania.

Obr. 5.7:]

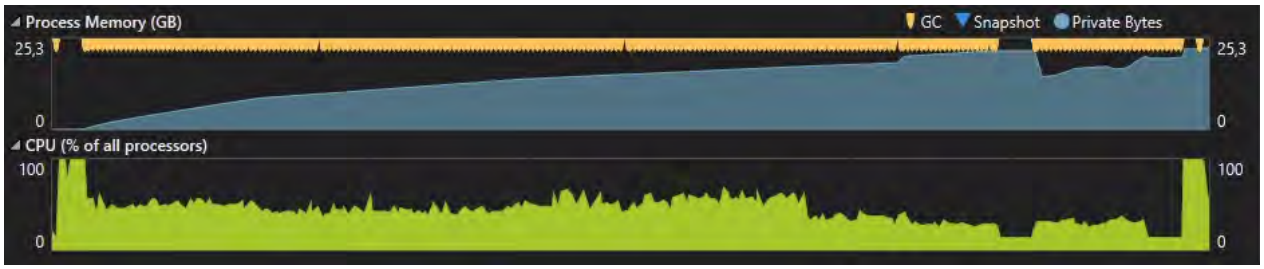
Využitie operačnej pamäte s použitím metódy Math.NET na signále č.1 dlhom 20 000, ktorý sa skladá z hodnôt 1 až 100 000.



Obr. 5.8: Využitie operačnej pamäte s použitím metódy MEQ na signále č.1 dlhom 20 000, ktorý sa skladá z hodnôt 1 až 1000.

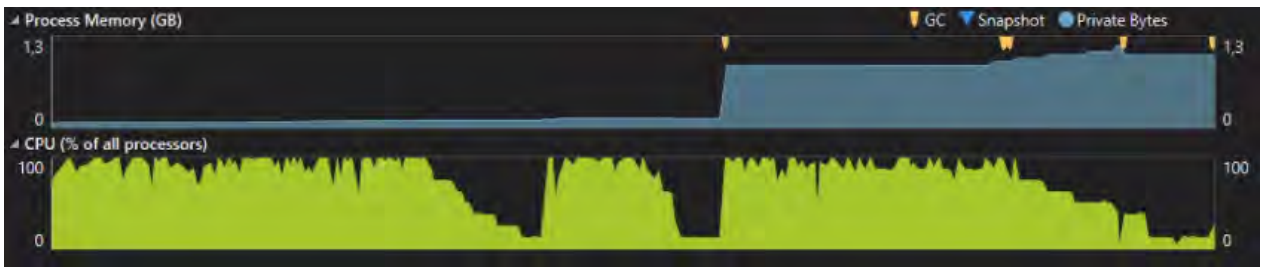
5.3 Zhrnutie kapitoly

Metóda Math.NET je obmädzená rozsahom indexovania v programovacom jazyku C#, ktorý nepodporuje výpočet pre vstupné signály dlhšie ako 40 000. Metóda MEQ podporuje výpočty signálov dlhších ako 100 000, avšak je možné naraziť na nedostatok operačnej pamäte v prípade velkého počtu unikátnych dát vo vstupnom signále. Najrýchlejšiou metódou je T-Digest, ktorá okrem veľkej rýchlosti výpočtu nezaberá velké množstvo operačnej pamäte. Na základe týchto vlastností je možné pomocou metódy T-Digest analyzovať aj vstupné signály o dĺžkach väčších ako 1 milión.



Obr. 5.9: Využitie operačnej pamäte s použitím metódy MEQ na signále č.1 dlhom 20 000, ktorý sa skladá z hodnôt 1 až 100 000.

Obr. 5.10: Využitie operačnej pamäte s použitím metódy T-Digest na signále č.1 dlhom 20 000, ktorý sa skladá z hodnôt 1 až 1000.



Obr. 5.11: Využitie operačnej pamäte s použitím metódy T-Digest na signále č.2 dlhom 20 000, ktorý sa skladá z hodnôt 1 až 100 000.

6 Diskusia

Pri plánovaní paralelizácie aplikácie sme sa rozhodovali medzi použitím procesora alebo grafického procesora na výpočet. Grafický procesor má výhodu v rýchlosti paralelných výpočtov. Cieľom bolo ale vytvorenie aplikácie, ktorá bude schopná analyzovať dáta aj na prenosných počítačoch. Väčšina prenosných počítačov nedisponuje grafickým procesorom, čo by limitovalo zrovna týchto užívateľov. Použitie procesora na výpočet RQA je síce pomalšie oproti grafickému procesoru, ale hneď pri nainštalovaní je pripravené k spusteniu. Výrobcov grafických procesorov je niekoľko a každý ma vlastný ovládač s odlišným rozhraním, ktorý by si užívateľ musel nainštalovať spoločne s našou aplikáciou.

Implementáciou metódy T-Digest sa rapídne zvýšila maximálna dĺžka spracovania dát, a to niekoľkonásobne. Nevieme s presnosťou určiť limitáciu použitím tejto metódy. Najdlhšie otestovaný signál mal dĺžku jeden milión vzoriek a celá analýza trvala približne 16 hodín pre všetky metódy prahovania bez vykreslenia rekurentných grafov. Pri využití metódy MEQ záleží na rôznorodosti dát. Presný predpoklad nie je teda možné určiť, platí avšak pravidlo, že čím majú dáta väčší rozptyl a viac unikátnych hodnôt, tým dlhší signál je táto metóda schopná spracovať. Metóda Math.NET je limitovaná takou dĺžkou signálu, ktorá sa zmestí do jedného listu. Preto sa odporúča používať ju na signály do 40 000 vzoriek.

6.1 Systémové požiadavky

Medzi hlavné systémové požiadavky patrí hlavne operačný systém Windows, na ktorom je podporovaný framework .NET 4.5 a viac, aby bolo možné program spustiť. Program bol testovaný na systémoch s operačným systémom Windows 10 a Windows Server 2019, na ktorých spustenie aplikácie nerobilo žiadny problém. Kvôli paralelizácií odporúčame aspoň osem logických procesorov s 8GB operačnej pamäte RAM. Čím vyšší počet logických procesorov a vyššia taktovacia frekvencia, tým rýchlejší je výpočet jednotlivých signálov.

6.2 Ďalšie možnosti paralelizácie

Na grafoch v kapitole optimalizácie je vidieť priebeh využitia procesora pri výpočte RQA. Snahou bolo využitie plného potenciálu procesora v každej časti výpočtu. Pri hľadaní optimálnej dimenzie vnorenia sa pomocou rozdelenia výpočtu na subdimenzie sa podarilo docieliť plného využitia procesora. Časť vytvárania rekurentných grafov je paralelizovaná pomocou prerozdelenia vykreslenia jednotlivých metód prahovania. Ak si uživateľ zvolí len jednu metódu prahovania a vykreslenia, neodzrkadlí sa to na rýchlejšom vykreslení. Časť výpočtu kvantilu je paralelizovaná v rámci možností jednotlivých metód. K dosiahnutiu plného využitia výpočtového výkonu by bola potrebná implementácia inej metódy výpočtu kvantilu, ktorá by bola schopná bežať paralelne v plnom rozsahu.

7 Záver

Hlavným cieľom tejto práce bolo vytvoriť desktopovú aplikáciu a zvýšiť výpočetný výkon pre vykonanie rekurentnej kvantifikačnej analýzy tak, aby bolo možné spracovať signály dlhšie ako 100 000 vzoriek.

Pri realizácií a optimalizácií sa naskytlo niekoľko problémov. Pri prvom probléme sme sa venovali počítaniu optimálnej dimenzie vnorenia, kde sme pomocou rozdielneho indexovania a paralelizácie umožnili vyhľadávať relatívne rýchlo dimenziu vnorenia. Druhým problémom bola štatistika, konkrétnejšie výpočet smerodajnej odchýlky, priemeru a kvantilu. Smerodajná odchýlka je počítana pomocou implementácie Welfordovho algoritmu na streamovaných dátach. Na hľadanie hodnoty kvantilu sme implementovali tri metódy výpočtu kvantilu. Výsledkom je aplikácia RQA v programovacom jazyku C#, ktorej schopnosť spracovania dlhších dát zavisí na zvolenej metóde počítania hodnoty kvantilu. Validačné výsledky porovnávané pomocou programovacieho jazyka python vykazujú, že najpresnejšiou metódou je MEQ, čo je C# implementáciou matlabového kvantilu. Z časového a optimalizačného hľadiska sa osvedčila metóda T-Digest, pomocou ktorej je možné spracovať signály aj 10-krát väčšie, než bolo cieľom optimalizácie.

Aplikácia disponuje užívateľsky prívetivým grafickým rozhraním s možnosťou nastavenia počiatočných parametrov, metódy kvantilu, výberu prahu, vykreslenia rekurentného grafu a heatmapy. Na vykreslenie rekurentného grafu sa používa tiff indexovaného typu, kde nás pri vykreslovaní obmedzuje počet indexovaných farieb Aplikácia dokáže vykresliť rekurentné grafy s maximálnou dĺžkou signálu 130 000. Na vykreslenie heatmapy je potrebné aj použitie farieb a tým pádom aj viac bitov na pixel. Na výber sú 4 bity na pixel s limitáciou dĺžky signálu 65 000 a 8 bitov na pixel s limitáciou dĺžky signálu 40 000. V prílohe je užívateľská príručka, ktorá sprevádza uživateľa celým procesom analýzy, od nastavenia parametrov až po otvorenie súboru s výsledkami.

Zoznam použitej literatúry

[1] Joseph P. Zbilut and Charles L. Webber. Recurrence quantification analysis: Introduction and historical context. 17.

[2] Piotr Jakliński Arkadius Syta, Jacek Czarnigowski. Detection of cylinder misfire in an aircraft engine using linear and non-linear signal analysis.

[3] Mohammad Mahdi Johari and Mohammad Mahdi Nayebi. Robust airborne target recognition based on recurrence plot quantification of micro-doppler radar signatures. In *2016 17th International Radar Symposium (IRS)*, pages 1–4. IEEE.

[4] Charles L. Webber and Norbert Marwan, editors. *Recurrence Quantification Analysis: Theory and Best Practices*. Understanding Complex Systems. Springer International Publishing.

[5] Charles L. Webber. RQA software. [online]. Cit. [2021-06-15]. Dostupné z: `http://cwebber.sites.luc.edu/`.

[6] Norbert Marwan. Commandline recurrence plots. [online]. Cit. [2021-03-11]. Dostupné z: `https://tocsy.pik-potsdam.de/commandline-rp.php`.

[7] Tobias Rawald, Mike Sips, and Norbert Marwan. PyRQA—conducting recurrence quantification analysis on very long time series efficiently. 104:101–108.

[8] Schlenker J. Hanáková L., Socha V. RQAcalc. [online]. Cit. [2021-06-15]. Dostupné z: `http://uld.fd.cvut.cz/stazeni/vedecke_vystupy/RQAcalc.rar`.

[9] Norbert Marwan. How to avoid potentail pitfalls in recurrence plot based data analysis. 21(4):1003–1017.

[10] B. P. Welford. Note on a method for calculating corrected sums of squares and products. 4(3):419–420.

[11] Bill Wagner. A tour of C# - C# guide. Microsoft Docs [online]. Cit. [2021-04-02]. Dostupné z: https://docs.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/.

[12] Terry Glee. Overview of Visual Studio. Microsoft Docs [online]. Cit. [2021-04-02]. Dostupné z: https://docs.microsoft.com/en-us/visualstudio/get-started/visual-studio-ide?view=vs-2019.

[13] Ted Dunning and Otmar Ertl. Computing extremely accurate quantiles using t-digest.

[14] Terry Glee. What is windows forms - windows forms .NET. Microsoft Docs [online]. Cit. [2021-04-02]. Dostupné z: https://docs.microsoft.com/en-us/dotnet/desktop/winforms/overview/.

[15] Terry Glee. What is WPF? - visual studio (windows). Microsoft Docs [online]. Cit. [2021-04-02]. Dostupné z: https://docs.microsoft.com/en-us/visualstudio/designers/getting-started-with-wpf.

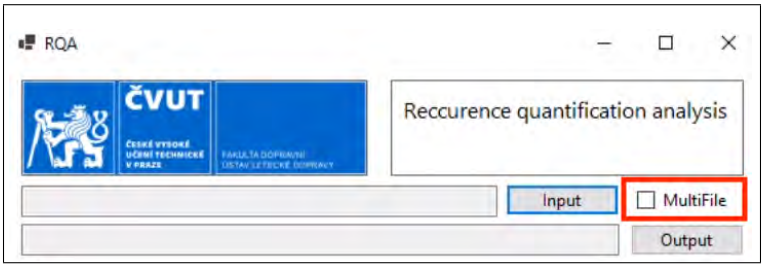
Príloha - Používateľská príručka

Súčasťou prílohy je krok za krokom nastavenie aplikácie RQA pre úspešnú analýzu signálu. Celé grafické užívateľské rozhranie je navrhnuté tak, aby užívateľ postupoval zhora nadol.

1. Prvým krokom je načítanie signálu do aplikácie pomocou tlačítka *Input* (Obr. 7.1). Súbor musí byť vo formáte .csv, kde desatinné čísla sú bodky a oddelovač je stredník.

Obr. 7.1

2. Ak je zaškrknuté tlačitko možnosti *MultiFile* (Obr. 7.2), aplikácia RQA bude analyzovať všetky súbory .csv v príslušnom prečinku zvoleného súboru. Toto slúži na analýzu viacerých dát.

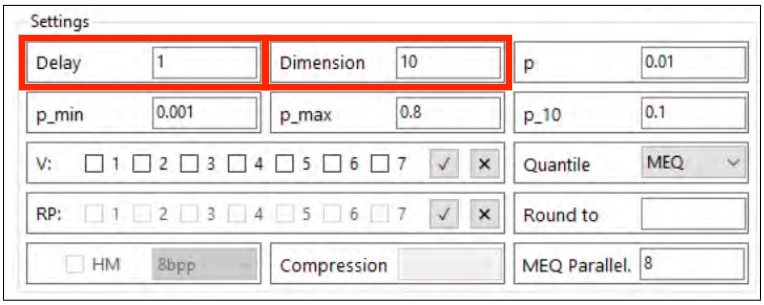


Obr. 7.2

3. Následne stačí zvoliť cieľový priečinok pomocou tlačítka *Output* (Obr. 7.3). Pokiaľ nie sú zvolené obe cesty vstupu a výstupu, nie je možné analýzu spustiť.

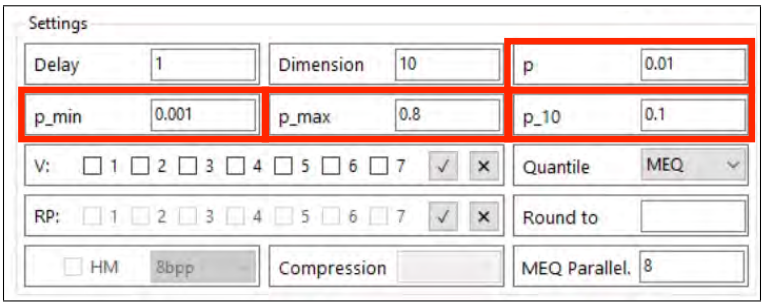
Obr. 7.3

4. V prvých dvoch textových poliach je možné zvoliť, do akého časového oneskorenia a akej dimenzie budeme hľadať ich optimálne hodnoty (Obr. 7.4).



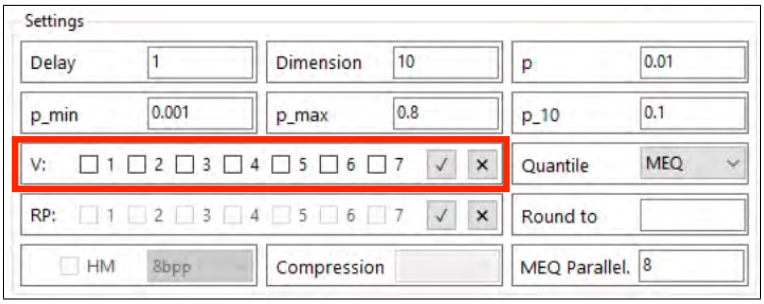
Obr. 7.4

5. V parametroch p zobrazených na obrázku (Obr. 7.5) je možné si nastaviť iné ako prednastavené hodnoty kvantilov.



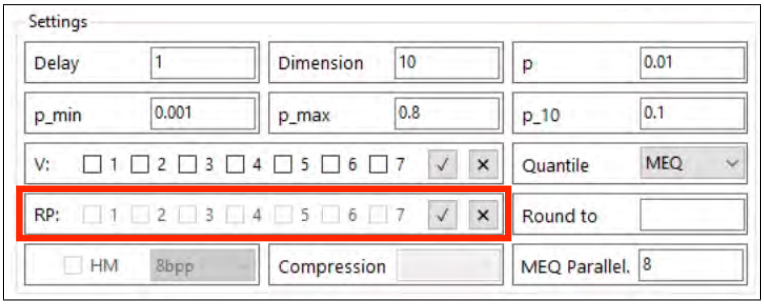
Obr. 7.5

6. V riadku V je možné vybrať si jednotlivé metódy prahovania pomocou zaškrtávacích políčok, pre ktoré chce užívateľ vypočítať kvantifikačné parametre (Obr. 7.6). Pre zjednodušenie vypočítania každej metódy prahovania su pridané na pravej strane dve tlačítka na zaškrtnutie a odškrtnutie všetkých metód prahovania naraz.



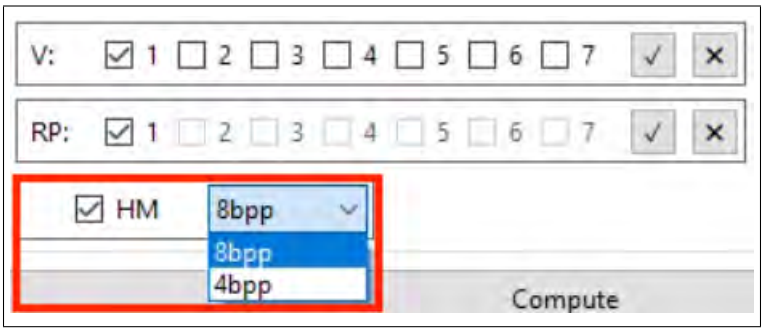
Obr. 7.6

7. Pod riadkom výberu metódy prahovania sa nachádzajú zaškrtávacie políčka pre vykreslenie rekurentného grafu (Obr. 7.7). Tieto políčka nie je možné stlačiť bez zvolenej metódy prahovania v predošlom kroku.



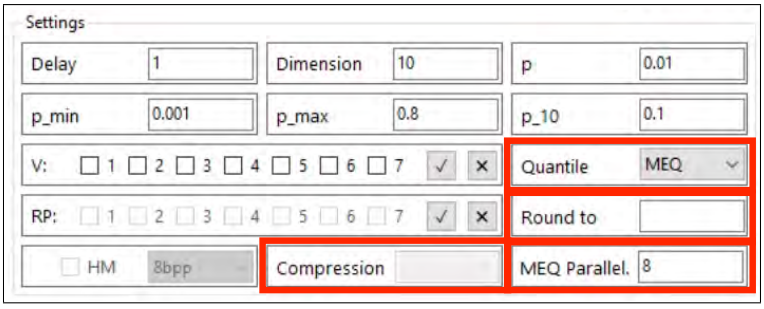
Obr. 7.7

8. Posledným výberom vykreslovania je možnosť vykreslenia heatmapy (Obr. 7.8). Ak užívateľ zaškrtne políčko, odomkne sa výber rozbaľovacieho listu, v ktorej si užívateľ môže vybrať jednu z dvoch dostupných farebných paliet. Farebná paleta 4bpp používa na vykreslenie 16 farieb. Farebná paleta 8bpp používa na vykreslenie 256 farieb.

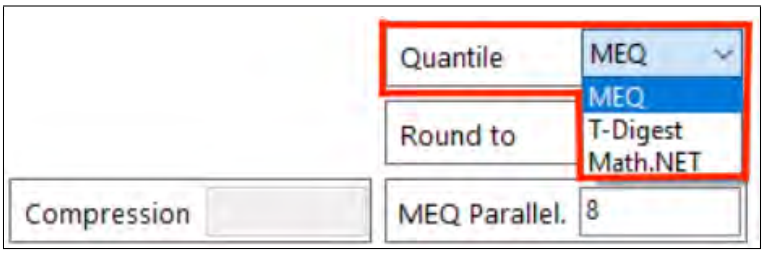


Obr. 7.8

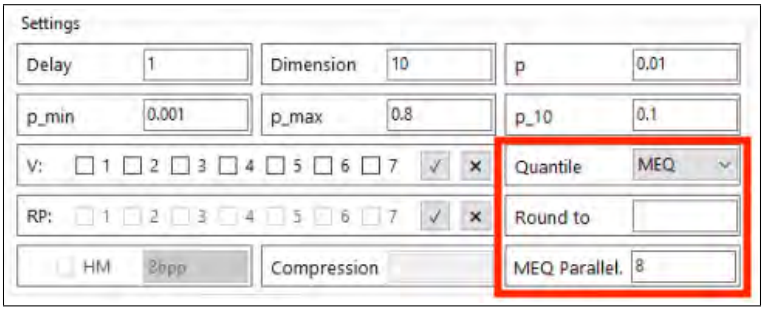
9. Posledné textové polia zobrazené na obrázku sa používajú na zvolenie metódy výpočtu kvantilu a nastavenie ich parametrov (Obr. 7.9). Pomocou rozbaľovacieho zoznamu (Obr. 7.10) sú na výber tri metódy výpočtu kvantilu. Prvou metódou je MEQ, pri ktorej sa odomknú textové polia *Round to* a *MEQ Parallel* (Obr. 7.11). Užívateľ vpísaním čísla do textového poľa *Round to* dokáže nastaviť zaokrúhľovanie kvantilu. Ak zostane toto pole prázdne, žiadne zaokruhľovanie hodnôt nebude použité. Textové pole *MEQ Parallel* vypíše na začiatku dostupný počet logických procesorov. Uživateľ môže nastaviť nižší počet logických procesorov. Druhou metódou je T-Digest, pri ktorom je možné nastaviť parameter kompresie, tzv. koľko zhlukov bude používať pri výpočte kvantilu (Obr. 7.12). Treťou metódou je Math.NET, ktorý neponúka žiadne ďalšie nastavenia. Po zvolení metódy a kontrole zadania každého parametra v oblasti Settings sa analýza spustí tlačítkom *Compute*.



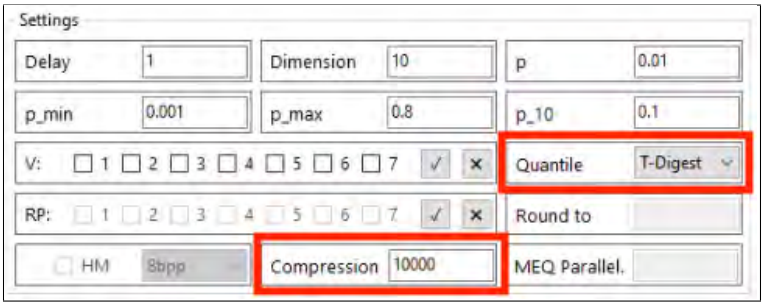
Obr. 7.9



Obr. 7.10



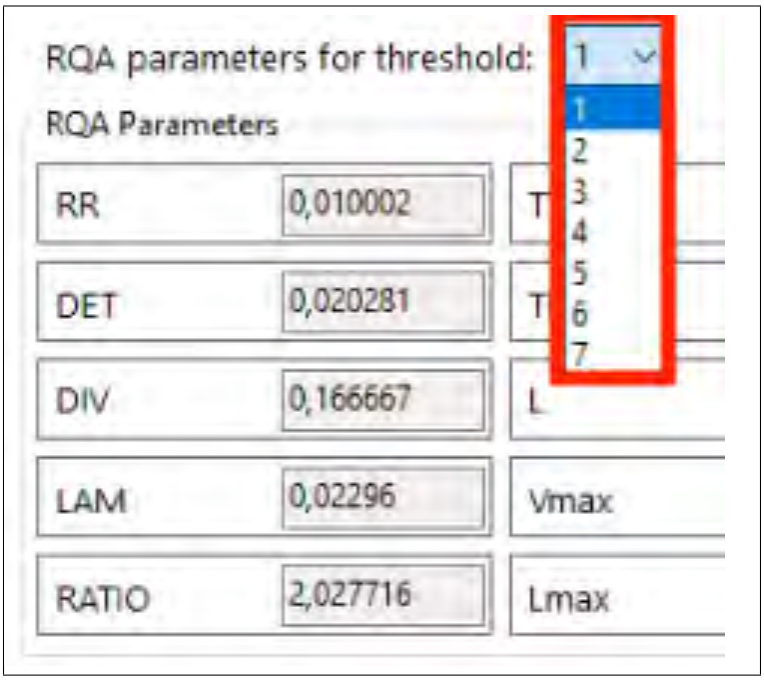
Obr. 7.11



Obr. 7.12

10. Po dokončení analýzy sa zobrazia kvantifikačné parametre v oblasti RQA Parameters pre zvolenú prahovaciu metódu. Medzi metódami prahovania je možné prepínať pomocou rozbaľovacieho listu (Obr. 7.13).

11. K rýchlemu náhľadu rekurentného grafu a heatmapy po analýze bolo pridané tlačítko *Open Log Folder*, ktorý otvorí priečinok, do ktorého boli exportované výsledky, rekuretné grafy a heatmapa (Obr. 7.14).



Obr. 7.13

Obr. 7.14

Výsledný priečinok po dokončení analýzy je nazvaný pomocou názvu signálu, za ktorý sa pridá časová značka. V priečinku sa, podľa prvotného nastavenia, bude nachádzať niekoľko súborov. Súbor *results.json*, v ktorom sú uložené výsledné kvantifikačné parametre pre zvolené metódy prahovania, súbor *timer.txt*, v ktorom sú uložené časové hodnoty a dĺžka analyzovaného signálu.