TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Automatická analýza archivů dat kvality elektrické energie

Bakalářská práce

Studijní program: B2612 – Elektrotechnika a informatika
Studijní obor: 2612R011 – Elektronické informační a řídicí systémy
Autor práce: Tomáš Trdla
Vedoucí práce: Ing. Jan Kraus, Ph.D.
Konzultant: Ing. Tomáš Bedrník

Liberec 2016

Technická univerzita v Liberci
Fakulta mechatroniky, informatiky a mezioborových studií
Akademický rok: **2016/2017**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Tomáš Trdla**
Osobní číslo:	**M14000128**
Studijní program:	**B2612 Elektrotechnika a informatika**
Studijní obor:	**Elektronické informační a řídicí systémy**
Název tématu:	**Automatická analýza archivů dat kvality elektrické energie**
Zadávající katedra:	**Ústav mechatroniky a technické informatiky**

Zásady pro vypracování:

1. Seznamte se s používanými postupy a typickými veličinami pro popis kvality elektrické energie (v ustáleném stavu, nezabývejte se přechodovými jevy).
2. Navrhněte aplikaci pro zpracování vstupních souborů s daty tak, aby bylo možné analyzovat velké objemy měřených veličin a zkoumat jejich charakteristiky v různých časových intervalech.
3. Na ukázkových datech demonstrujte správnou funkci Vaší aplikace a prezentujte vhodným způsobem svá zjištění ohledně rozdělení hodnot jednotlivých veličin v různých záznamech.
4. Shrňte dosažené výsledky a uveďte možnosti využití Vašich zjištění pro inovaci automatizovaného zpracování archivů dat kvality elektrické energie.

Rozsah grafických prací: **dle potřeby dokumentace**

Rozsah pracovní zprávy: **30–40 stran**

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

[1] **BOLLEN, M.H.J. What is power quality? Electric Power Systems Research. 2003, 66(1): 5-14. DOI: 10.1016/S0378-7796(03)00067-1. ISSN 03787796. Dostupné také z: http://linkinghub.elsevier.com/retrieve/pi**

[2] **IEEE Transactions on Smart Grid. 2014, 5(1). ISSN 1949-3053. Dostupné také z: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6684605**

[3] **MAA, Heiko, Hüseyin Kemal CAKMAK, Felix BACH, Ralf MIKUT, Aymen HARRABI, Wolfgang SÜ, Wilfried JAKOB, Karl-Uwe STUCKY, Uwe G. KÜHNAPFEL, et al. Data processing of high-rate low-voltage distribution**

Vedoucí bakalářské práce: **Ing. Jan Kraus, Ph.D.**
Ústav mechatroniky a technické informatiky

Konzultant bakalářské práce: **Ing. Tomáš Bedrník**
Ústav mechatroniky a technické informatiky

Datum zadání bakalářské práce: **10. října 2016**

Termín odevzdání bakalářské práce: **15. května 2017**

prof. Ing. Zdeněk Plíva, Ph.D.
děkan

L.S.

doc. Ing. Milan Kolář, CSc.
vedoucí ústavu

V Liberci dne 10. října 2016

Prohlášení

Byl(a) jsem seznámen(a) s tím, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé bakalářské práce pro vnitřní potřebu TUL.

Užiji-li bakalářskou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Bakalářskou práci jsem vypracoval(a) samostatně s použitím uvedené literatury a na základě konzultací s vedoucím bakalářské práce a konzultantem.

Současně čestně prohlašuji, že tištěná verze práce se shoduje s elektronickou verzí, vloženou do IS STAG.

Datum: 14.5.2017

Podpis: 

Poděkování

Tímto děkuji rodině a svým blízkým za obrovskou podporu při psaní bakalářské práce a studiu samotném. Děkuji také vedoucímu práce Ing. Janu Krausovi Ph.D. za pomoc s touto prací od samotného začátku až do konce i v osobním volnu.

Abstrakt

Práce se zabývá řešením aplikace pro automatickou analýzu elektrických veličin na základě statistického rozdělení pravděpodobnosti. Aplikace byla vytvořena za pomoci jazyka C#, .NET prostředí a MATLAB. K realizaci bylo využito programovací prostředí Microsoft Visual Studio Community 2015. Výsledkem je aplikace, která načítá uživatelem zvolené soubory typu CSV a uživatelem vybrané veličiny a analyzuje jejich statistické rozdělení pravděpodobnosti v různých časových intervalech. V experimentální a testovací části jsou aplikací otestovány reálně naměřená data na intervalech od 1 s až po 1 h a jsou zde popsány některé výsledky těchto testů.

Klíčová slova:

C#, .NET, CSV, Energy Management, statistika, rozdělení pravděpodobnosti, elektrické veličiny, testování hypotézy

Abstrakt

The thesis deals with the solution of the application for automatic analysis of electrical units based on statistical probability distributions. The application was created in C# language, .NET Framework and MATLAB. For realization the Microsoft Visual Studio Community 2015 was used. The result is an application that loads user-selected CSV files and user-selected variables and analyzes their statistical probability distribution at different time intervals. In the experimental and testing sections, the real-time measured data is tested at intervals from 1 s to 1 h, and the results of some of these tests are described.

Key words:

C#, .NET, CSV, Energy Management, statistic, probability distribution, electrical units, hypothesis tests

Obsah

Seznam zkratek

CSV	Comma Separated Values, formát souboru
CEA	speciální formát souboru prostředí ENVIS
PDF	Portable Document Format, formát souboru
RAM	Random-access memory, druh počítačové paměti
VS	Microsoft Visual Studio
XML	Extensible Markup Language, rozšiřitelný značkovací jazyk, formát soubor
KS	Kolmogorov-Smirnov, test statistické hypotézy
SW	Shapiro-Wilk, test statistické hypotézy
SMC	označení zdrojového souboru dat
SMZ	označení zdrojového souboru dat
Atd.	a tak dále
Např.	například

Seznam obrázků

Seznam tabulek

1 Úvod

V dnešní době mají dodavatelé elektřiny obrovské množství archivů dat – soubory se záznamy o vlastnostech a kvalitě dodávané elektřiny, jejích jednotlivých veličinách a jak se tyto hodnoty vyvíjí v čase na daném místě na zemi.

Zkoumání těchto časových sérií a jejich vlastností pomocí informatiky není tak rozšířené, a právě tím se zde v této práci budu zabývat – analýzou za pomoci propojení informatiky a statistiky.

Mým cílem je naprogramovat aplikaci, která bude automaticky analyzovat velké objemy vstupních souborů měřených veličin, zkoumat jejich charakteristiky v různých časových intervalech a sledovat jaké charakteristické vlastnosti mají dané veličiny. Tyto experimenty budou prováděny s využitím statistických rozdělení.

Výsledkem práce budou soubory s grafy, ve kterých budou analyzovány veličiny popisující kvalitu elektrické energie z pohledu statistických rozdělení pravděpodobnosti. Tyto grafy veličin budou mezi sebou porovnány a při dostatečném počtu zdrojových dat bude možné stanovit určité standardní chování pro tyto veličiny. Výsledné standardní chování by mohlo být využito ke kontrole, zda některé měřicí stanice a objekty neporušují pravidla.

Tato bakalářská práce se tedy zabývá testováním průběhů veličin, reprezentujících kvalitu elektrické energie, na statistické rozdělení pravděpodobnosti průběhu veličin za pomoci testů hypotéz Shapiro-Wilk (test normality) a Kolmogorov-Smirnov. K testovaným vzorkům je přistupováno dvěma způsoby, a to buď vzorky po sobě jdoucí v čase, nebo náhodným výběrem z většího intervalu hodnot. Vliv těchto rozdílných přístupů je porovnán.

Testovány jsou dva druhy souborů. V prvním jsou aktuální hodnoty veličin zapsané s intervalem 200 ms a ve druhém průměry hodnot za poslední 1 s. Je zkoumáno, zda tento rozdíl ovlivní výsledná rozdělení pravděpodobnosti.

V dalším testu je zkoumáno, jaký je vhodný interval pro průměrování takovýchto souborů a jejich jednotlivých veličin, aby zůstalo zachováno normální rozdělení dat.

2 Teoretická část

2.1 Vybrané veličiny popisující elektrickou energii

Oblastí zájmu pro toto zkoumání jsou elektrické veličiny, jejichž chování není ovlivněno jednáním lidí uvnitř objektu. U takového výkonu můžeme například očekávat změnu ve chvíli, kdy člověk uvnitř objektu zapne rychlovarnou konvici, motor atd.

2.1.1 Frekvence

Frekvence sítě je 50 Hz. Střední hodnota kmitočtu se musí nacházet v intervalu od 49,5 do 50,5 Hz. V tomto rozmezí musí být frekvence v průběhu 99,5 % týdne a v rozmezí 47–52 v celých 100 % týdne. [9]

2.1.2 Napětí

Velikost napětí v nízkonapěťových sítích je dána 230 V ±10 % [9], což odpovídá rozmezí 207–253 V. Hodnota kolísá v důsledku připojování a odpojování zátěží, ale je vyrovnávána transformátory.

2.1.3 Harmonické složky napětí

Síťové napětí by v ideálním případě mělo mít, rovnoměrné sinusové napětí s konstantní amplitudou a frekvencí. Nicméně nelineární spotřebiče odebírají ze sítě nesinusový (neharmonický) zátěžový proud. Příliš velká deformace, případně příliš velký obsah vyšších harmonických složek vede k tomu, že citlivé elektronické stanice – zařízení, jako jsou např. počítače či senzory – mohou fungovat poruchově. [16]

2.1.4 THDU

Popisuje míru celkové deformace napětí a udává poměrný obsah harmonických složek. Čím vyšší, tím vyšší je zatížení sítě a zařízení.

2.2 Vybrané statistické rozdělení pravděpodobnosti

Zkoumaná data jsou v čase po sobě jdoucí hodnoty jednotlivých veličin, proto je nutné, aby vybraná statistická rozdělení byla jednorozměrná a v čase spojitá rozdělení pravděpodobnosti. Zájmem této práce je zjistit jakému rozdělení odpovídají některé

elektrické veličiny, respektive jakému rozdělení při různém počtu vzorků. Pojmem „spojitá“ jsou v této práci myšleny po sobě jdoucí vzorky ve zdrojových souborech odečítané s určitým intervalem (viz kapitola 3.2 a 4.1).

2.2.1 Normální (Gaussovo)

Jedná se o nejpoužívanější model rozdělení spojité náhodné veličiny. Hustota má typický zvoncovitý tvar (viz Obrázek 16), symetrický kolem střední hodnoty.

2.2.2 Log-normální

Spojité rozdělení pravděpodobnosti, jejíž logaritmus vzorků je rozdělen podle normálního rozdělení. Viz Obrázek 16.

2.2.3 Cauchyho

Jedná se o rozdělení, které nemá střední hodnotu ani konečný rozptyl. Podobá se normálnímu. Pokud jsou ale parametry formovány do symetrického rozložení hustoty, poté je možné stanovit ekvivalent střední hodnoty.

2.2.4 Exponenciální

Je to rozdělení, kde je náhodnou veličinou čas, ve kterém sledovaný jev nastává. Takovým jevem je například porucha. Hustota rozdělení viz Obrázek 16.

2.3 Testy statistické hypotézy

Statistickou hypotézou rozumíme předpoklad o typu pravděpodobnostního rozdělení, nebo jeho parametrech. Test takovéto hypotézy je rozhodovací postup, založený na výsledcích, zjištěných z pozorování a případně teoretického očekávání. Podle výsledků je daná hypotéza zamítnuta, či ne. Takový výsledek může být reprezentován i P hodnotou, která udává pravděpodobnost platnosti hypotézy.

Pro testování hypotéz vždy potřebujeme nulovou a alternativní hypotézu. V této práci je nulovou hypotézou, že pozorované vzorky jsou z určitého rozdělení pravděpodobnosti. Alternativní hypotézou je, že pozorované vzorky tomuto rozdělení neodpovídají.

Následující dva testy byly vybrány z důvodu, že SW se hodí pro testování normálního rozdělení a testem KS je možno testovat jakákoliv rozdělení.

2.3.1 Shapiro–Wilk

Tento test zkoumá, zda testovaný vzorek odpovídá normálnímu rozdělení. Nelze jím testovat ostatní rozdělení pravděpodobnosti.

2.3.2 Kolmogorov–Smirnov

Test Kolmogorov–Smirnov se snaží rozhodnout o podobnosti, respektive rozdílu, mezi dvěma rozděleními pravděpodobnosti. Výhodou je, že nezáleží, na jakém rozdělení se tento test použije. Stejně jako u SW platí, čím větší je vzorek, tím větší je statistická významnost. [3]

2.4 Parametry rozdělení a testů

- **P-value** – P-hodnota vyjadřuje pravděpodobnost platnosti nulové hypotézy. Lze ji definovat i jako nejmenší hladinu významnosti, při jaké bychom na daných datech zamítly nulovou hypotézu.
- **Test statistic** – Jedná se o parametr, který udává, jak jsou pravděpodobná naměřená data, pokud platí nulová hypotéza. Je transformací pozorovaných hodnot. Hluboce souvisí s p-value a rozhoduje tedy o výsledku zamítnutí hypotézy.
- **Statistická významnost** – Statisticky významné jsou vzorky, jejichž p-value je menší než alfa, tedy zvolená kritická hladina významnosti.
- **Mean** – Může znamenat průměr hodnot anebo střední hodnotu daného vzorku, která popisuje rozložení hodnot znaku pomocí jednoho „typického“ čísla.
- **Modus** – Definován jako hodnota souboru s největší četností a používá se v případě, kdy hlavním zájmem je zjištění nejvyšších hodnot souboru.
- **Quartil** – Konkrétní kvantily, které rozdělují uspořádaný statistický soubor na čtyři části se stejnou četností, tj. 25 %, 50 % a 75 % kvantily.

- **Medián** – Hodnota prostředního (50%) kvantilu, tzn. že je ve prostředku všech pozorování seřazených podle hodnot.
- **Rozptyl** – Je definován jako součet kvadratických odchylek od průměru dělený rozsahem výběru.
- **Variance** – Rozdíl mezi maximální a minimální hodnotou sledované proměnné. Je to také jiný výraz pro rozptyl.
- **STD** – Směrodatná odchylka. Definována jako druhá odmocnina z rozptylu náhodné veličiny (na rozdíl od rozptylu je v původních jednotkách proměnné).
- **Entropy** – Míra neuspořádanosti

2.5 Časová náročnost výpočetních procesů

Lze očekávat, že testování několika souborů s velkým obsahem dat pro různá rozdělení a časové intervaly zabere hodně času, proto bude vhodné využít výpočetní síly více procesorů – MultiThreading.

MultiThreading je vlastnost procesoru počítače, která umožňuje, aby více procesů běželo v jednu chvíli. V dnešní době se jedná již o naprostý standard pro každý počítač. Multithreading tak slouží pro zpracování úkolů na pozadí, aby na ně uživatel nemusel čekat. Zároveň ale slouží i ke zrychlení časově náročných procesů tím, že vícejádrové procesory využívají více jader pro dané procesy namísto jednoho.

2.6 Očekávané výsledky

Můžeme očekávat, že se zvyšujícím se testovaným intervalem (počtem vzorků) bude narůstat rozptyl jejich hodnot a tím pádem klesat shoda mezi teoretickým rozdělením a reálnými hodnotami. Nevíme ale, okolo jakého intervalu se tento jev projeví.

Podle normy EN 50160 a článku [9] se frekvence musí pohybovat ±1 % (49,5 – 50,5 Hz) po dobu 99,5 % týdne a -6 %/+4 % (47–52 Hz) po dobu 100 % týdne. Napětí ±10 % po dobu 95 % týdne. Harmonické napětí 6 %-pátá, 5 %-sedmá, 3.5 %-jedenáctá, 3 %-třináctá složka. THD <8 %. Tabulka s jednotlivými harmonickými viz Obrázek 16.

3 Návrh řešení

3.1 Nástroje

Ze zadání práce a zkušeností jsem se rozhodl, že aplikaci budu koncipovat jako desktopovou aplikaci běžící v prostředí Windows.

3.1.1 Jazyk a vývojové prostředí

S programováním desktopových aplikací mám již nějaké zkušenosti z bakalářského projektu, školních předmětů, kde jsme využívali jazyk C#, a také jsem se tomuto jazyku věnoval ve volném čase. Právě proto jsem se rozhodl pro tento jazyk a mně známé vývojové prostředí Microsoft Visual studio 2015, které mi vyhovuje ze všech, prozatím vyzkoušených, nejvíce. Důvodem pro toto rozhodnutí je, že C# je pro mě přehledný jazyk s jednoduchou, a především logickou syntaxí. VS umožňuje přehledné řešení a ladění programu, stejně tak jako prostou tvorbu grafického (uživatelského) rozhraní aplikace, které je převážně „klikací".

3.1.2 Grafická reprezentace výsledků

S jistotou mohu očekávat, že tato práce je především o grafických výstupech v podobě grafů a jejich významech, proto bylo nutné najít řešení pro jednoduché vykreslování různých druhů grafů a dat do nich. K tomu jsem se rozhodl využít program MATLAB, ve kterém sice nemám takové zkušenosti, ale vím o jeho možnostech programovatelného a flexibilního vykreslování dat do grafů a jejich následnou manipulaci. VS sice umožňuje grafické kreslení grafů, nicméně v době tohoto rozhodnutí jsem předpokládal, že dat, která budou vykreslována, bude velké množství a ze zkušeností vím, že grafy ve VS nejsou ideálním řešením pro větší objemy dat z důvodu časové náročnosti vykreslení, manipulace a flexibility.

3.1.3 Použité knihovny

3.1.3.1 Accord

Jedná se o knihovnu [2], která má široké využití jako například pro strojové učení, regresi, práci s audio signálem a mnoho dalších, pro mě je však hlavní statistické

rozdělení a testy hypotéz. Tato knihovna je volně dostupná, zdarma a nevyžaduje žádnou instalaci či nákup licence. Její použití je velice jednoduché a intuitivní, navíc má velice přehlednou dokumentaci, která je dostupná online. Zkoušel jsem i další knihovny pro práci se statistickými rozděleními a testy hypotéz, ale žádná nebyla takto kvalitní [1], nebo postrádala dokumentaci [3], případně se v ní nedalo vyznat [14], jelikož byla nepřehledná.

3.1.3.2 iTextSharp

Tato knihovna [11] je nástavbou pro C# pro programovou tvorbu, úpravu a práci s PDF soubory. Její dokumentace není moc informativní, ale zato lze na internetových fórech najít převážnou většinu informací a rad, které uživatel potřebuje. Pro použití právě této knihovny jsem se rozhodl opět kvůli předchozím zkušenostem a práci s ní.

3.1.3.3 MATLAB knihovna – MLApp

Knihovna je nainstalována spolu s MATLABem a je jeho součástí, tudíž nebylo nutné instalovat a hledat jiné knihovny. Díky této knihovně je možné vytvořit MATLAB prostředí virtuálně přímo z programu napsaném v C# a využívat jeho funkcí. Je jednoduché na ovládání, kdy pro MATLAB vytvoříme funkci v podobě m-filu a ten pak z programu otevřeme a předáme mu příslušné parametry. Daná funkce se pak provede a pochopitelně v závislosti na tom, k čemu je určena. Může nám např. vykreslit grafy, spočítat různé matematické operace a zkrátka vše, co MATLAB umožňuje.

3.2 Vstupní soubory a jejich struktura

Archivní data se mohou nacházet v různých typech. Pro tuto práci jsem měl od vedoucího práce k dispozici soubory formátu CSV a CEA. Je absolutní nutností, aby tyto vstupní soubory dodržovaly určitou strukturu, pokud by byla narušena, pak není možné, aby je program načetl.

CEA formát souborů je binární formát pro ukládání dat z měřicích stanic. Jedná se o formát používaný prostředím ENVIS, které mi poskytl vedoucí práce. I přes poskytnuté zdrojové kódy je čtení ze souborů tohoto typu značně pomalejší oproti CSV, proto je tento formát nevhodný pro analýzu dat z měřicích stanic a bylo nutno z něj všechna data exportovat v prostředí ENVIS do CSV.

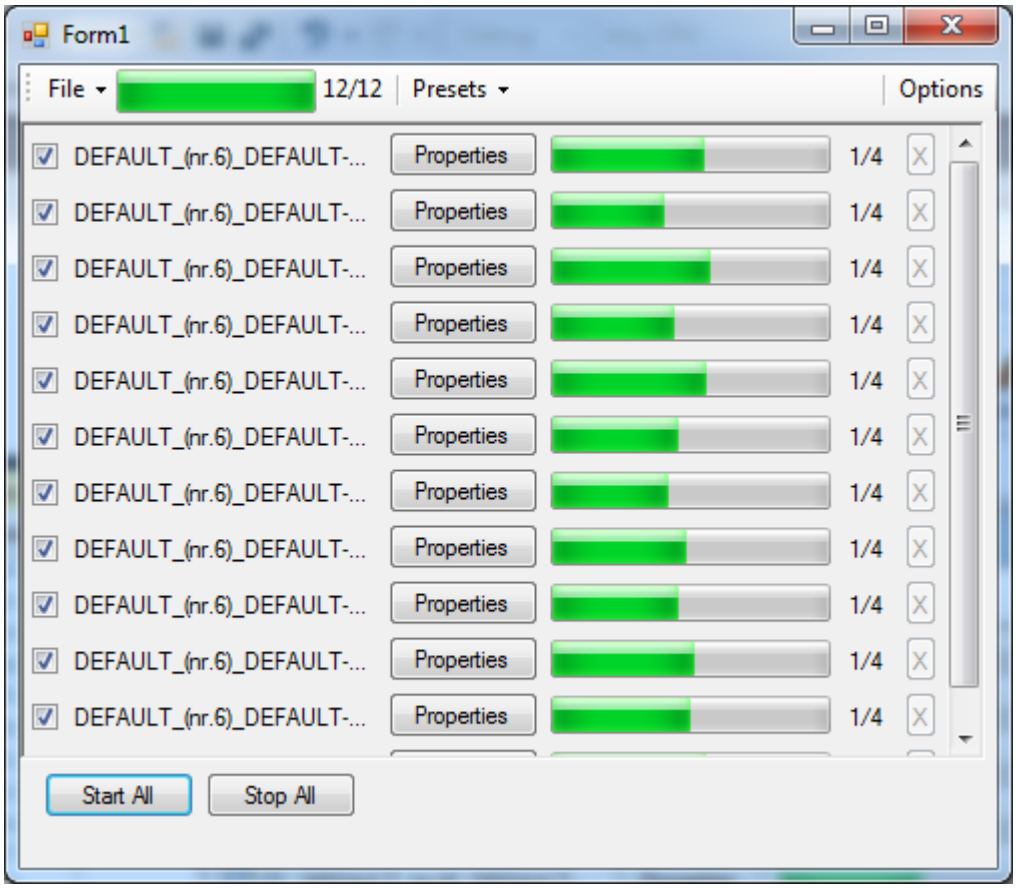
CSV znamená comma-separated values. Jak už sám název napovídá, jedná se o hodnoty oddělené čárkou. Je to souborový formát pro tabulková data, ve kterém jsou jednotlivé řádky odděleny znakem pro odřádkování, tudíž strojové načítání těchto souborů může využít čtení řádek po řádku. Pro čtení tohoto typu souborů byla využita knihovna, která se již nachází v prostředí VS – StreamReader.

V souborech jsou exportována data z elektronické měřicí stanice spolu s datem a časem měření. I po exportu z CEA je struktura CSV souborů stejná a celá aplikace je koncipována podle jejich struktury, ze které jsem vycházel. Pokud je taková struktura dodržena, je možné analyzovat jakýkoliv takový CSV soubor. Na prvním řádku je název tabulky, na druhém pak v každém sloupci zvlášť název veličiny. V prvním sloupci najdeme datum a čas měření a od třetího řádku dále se nacházejí samotné naměřené hodnoty pro danou veličinu (sloupec). Pořadí veličin, jejich počet, počet řádků už je poté různé.

Od vedoucího práce jsem měl k dispozici dva typy souborů, a to SMC a SMZ. SMC jsou soubory, do kterých jsou hodnoty zapisovány každých X ms. Oproti tomu SMZ jsou soubory, do kterých jsou zapisovány průměry hodnot za posledních X ms.

3.3 Návrh programu a jak funguje

3.3.1 Uživatelské rozhraní



Obrázek 1: Uživatelské rozhraní

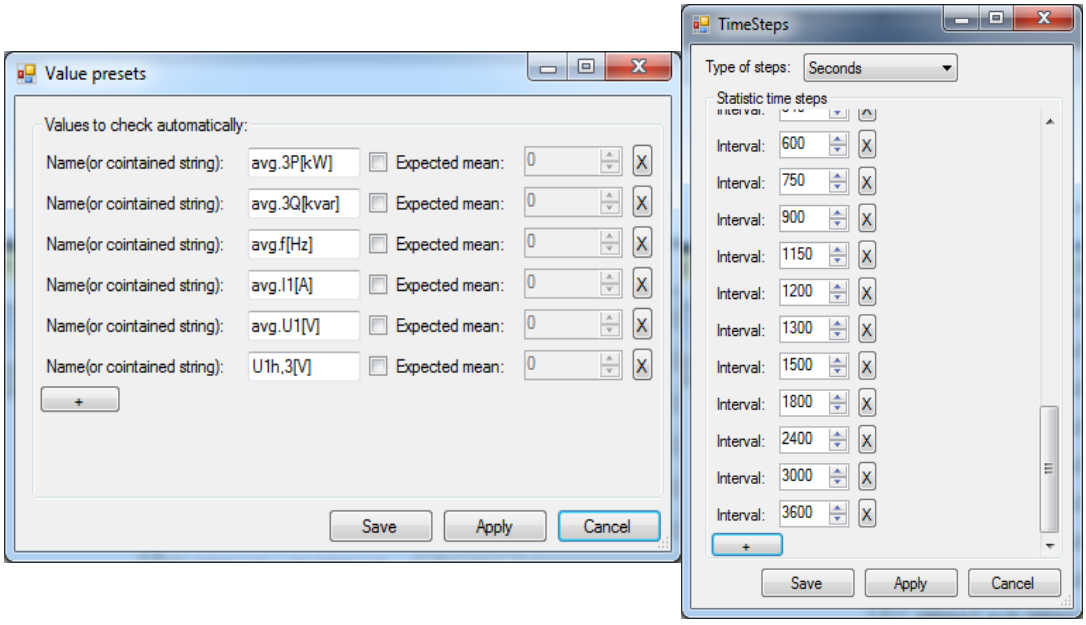
3.3.1.1 Horní panel

V horní části okna se nachází panel, který poskytuje základní ovládací prvky aplikace. Vlevo se nachází menu s dvěma tlačítky. První umožňuje uživateli výběr souborů, které mají být načteny pro pozdější analyzování a druhý slouží pro načtení rozpracované analýzy ze zálohy. Aktuální stav načítání reprezentuje tzv. progress bar, nacházející se vedle daného menu. Vedle něj je zobrazeno, který soubor z vybraných je právě načítán. Vedle se nachází další menu, které obsahuje dvě tlačítka pro zjednodušení výběru veličin a intervalů pro každý načtený soubor. Poslední prvek na této liště se nachází vpravo pod názvem „Options“. Jedná se o tlačítko, které zobrazí okno s nastavením aplikace.

3.3.1.2 Soubor

Uživatelské rozhraní je kromě horní lišty generováno dynamicky podle souborů, které uživatel chce načíst pro zpracování. Každý z načtených souborů je reprezentován svým vlastním řádkem, ve kterém se nachází checkbox, který při zaškrtnutí spouští proces a při odškrtnutí ho pozastaví. Vedle checkboxu je název souboru, progress bar reprezentující stav procesu, ve kterém se zrovna soubor nachází, což je podpořeno následným číslem „x/Y“ kde x označuje část, která zrovna probíhá a Y značí celkový počet částí. Dále má každý řádek tlačítko, po jehož nakliknutí se zobrazí okno s informacemi o daném souboru – jeho jméno, podnapis uvedený v tabulce, datum rozmezí dat od a do, interval zapsaných dat, počet veličin a řádků (záznamů). Pod tímto souhrnem jsou dvě okna – první obsahuje tabulku se všemi veličinami, které soubor obsahuje. Uživatel si zde zaškrtnutím checkboxu vybírá, které z nich chce analyzovat a může jim přidělit i očekávanou střední hodnotu (například pro frekvenci 50 Hz), což ale není vyžadováno. Druhé okno obsahuje seznam vybraných intervalů, které budou testovány. Uživatel si je zde přidává, odebírá a upravuje podle vlastního uvážení. Posledním komponentem řádku je tlačítko s křížkem, které daný řádek smaže a ostatní se tak posunou tak, aby po něm nezůstalo volné místo.

3.3.1.3 Automatické nastavení pro soubory



Obrázek 2: Přednastavení pro soubory

Ve výše zmíněném menu se nachází dvě tlačítka – první je pro výběr veličin, druhé pro výběr intervalu. Tím, že si zde uživatel dopředu navolí tyto parametry, je aplikace u každého souboru automaticky vybere a intervaly přiřadí. Pokud se ovšem daná veličina v souboru nachází a intervaly jsou v souladu s intervalem zapsaných hodnot. Uživatel díky tomu nemusí tyto parametry zdlouhavě naklikávat u každého souboru. Toto nastavení je uloženo do Application.Settings a tudíž je při každém novém spuštění načteno tak, jak bylo použito při posledním použití aplikace.

3.3.2 Načtení souborů

Načtení souboru je první a dá se říci jediná věc (krom stisknutí tlačítka „Start all“), kterou uživatel musí udělat, vše ostatní je už zautomatizované. Po stisknutí tlačítka v menu horní lišty se zobrazí známé okno pro výběr souborů ve Windows prostředí. Zde si uživatel může naklikat libovolné množství souborů typu csv nebo cea. Každý soubor je po potvrzení výběru načítán zvlášť a během toho je nutné zjistit určité informace o něm.

Načítání začíná vytvořením základních instancí pro aktuální a budoucí ukládání dat, otevřením instance souboru, zjištěním podnadpisu, načtením a rozdělením veličin, které soubor obsahuje a uložením jejich příslušného indexu reprezentující sloupec, na kterém se budou nacházet všechny jejich hodnoty. Tyto veličiny jsou nejprve označeny jako „nevyužité“ a až později jsou přezkoumány vůči shodám mezi automatickým výběrem a poté označeny jako „využité“. Dále je přezkoumán interval zapsaných hodnot, kde je první část dat přeskočena až do bodu, kde se nachází změna například v minutách nebo vteřinách. To proto, že začátek nemusí být přesně na počátku dané minuty, či vteřiny. Zjištěný interval je následně přezkoumán, jestli je v souladu s automatickým výběrem intervalu. Hodnoty, které takto korespondují (nelze zkoumat interval pro 10 vteřin, pokud zapisovací frekvence je každé 3 vteřiny), jsou ponechány, zbytek je vyřazen. Následně jsou spočítány všechny řádky, aby se mohl nastavit progress bar pro budoucí informování uživatele o stavu procesu.

Nakonec jsou vytvořeny příslušné „Controls“, které jsou popsány v kapitole Soubor, a ty jsou předány algoritmu, který je zobrazí a umístí v aplikaci. Tento

algoritmus na pozicování jsem vytvořil na míru a některé jeho části jsou použity při pozicování řádků v nastavení automatického výběru veličin a intervalů.

3.3.3 Univerzalita

Kladl jsem velký důraz na univerzalitu a dynamičnost celé aplikace, jelikož dopředu nevím, jak velké budou sobory a jaké veličiny budou obsahovat, kolik záznamů, jaký budou mít interval zapsaných hodnot, jaké budou časy a datumy zapsaných hodnot atd. Kromě toho také není známé, kolik souborů bude chtít uživatel analyzovat, jaké veličiny si vybere a jaké intervaly bude zkoumat. Všechny tyto otázky jsou položeny s ohledem na paměťovou náročnost. Lze říci, že při větším množství analyzovaných souborů by došlo v RAM paměti běžného domácího počítače místo, a to zejména v případě ukládání dat jako samostatných objektů. Zároveň pokud by nebylo vyřešeno využití možností výkonu procesoru, procesy výpočtů by trvaly několikanásobně déle.

Načtení veličin ze souboru je zjednodušeno tím, že jsou uloženy ve formě tabulky, tudíž příslušné hodnoty jsou na jednom řádku a rozděleny středníkem, takže jejich načtení je pouze oddělením každé „buňky“ a následným uložením seznamu do paměti. Jednotlivá data a parametry každého souboru jsou také vyřešeny individuálním přístupem ke každému ze souborů a načítáním těchto informací z nich.

3.3.3.1 RAM řešení

Složitější přístup nastává v místě, kdy začíná práce se souborem. Nelze načíst celý soubor najednou, protože pokud pracujeme s velkým objemem dat a nejistým počtem těchto souborů, tak by načtení několika z nich způsobilo nedostatek RAM paměti pro další soubory či samotné operace uvnitř programu. Proto je zde možnost přístupu k datům uvnitř souboru řádek po řádku – v případě csv i cea je to načtení jednoho řádku tabulky se všemi sloupci, což představuje mnohem menší nápor na RAM.

Testované hodnoty je třeba mít uloženy v paměti, protože pro každý interval testujeme různé množství hodnot. V hlavní výpočetní operaci je zahrnuto čištění těchto načtených hodnot, když už je jistota, že nebudou nadále potřeba.

Dalším problémem je ukládání mezivýsledků každého výpočtu a testu. Pokud by byl každý výsledek ukládán v samostatné instanci, listu či poli, tak už z prostého matematického hlediska, kdy máme 20 souborů, každý po 400000 řádcích, testujeme 20 veličin na 30 intervalech a potřebujeme uložit 40 hodnot z každého výsledku, tudíž uložit 190 miliard výsledků, navíc každý jako typ double, není rozumné řešení. Proto jsem zvolil přístup, kdy každý soubor u každé testované veličiny má uloženy instance intervalů, kde v každém z nich jsou umístěny instance jednotlivých testů určené pro výsledky a až v nich samotné výsledky v podobě čísla, kam se s každým novým výsledkem přičte jeho hodnota a do druhého čísla se přičte jednička. Takto je pak výsledek reprezentován aritmetickým průměrem výsledků a předchozích 190 miliard výsledků je zredukováno na necelý 1 milion, což už je přijatelné.

3.3.3.2 Využití výkonu procesoru

Po startu procesu zaškrtnutím checkboxu je pro každý soubor vytvořen `BackgroundWorker`, ve kterém celý proces poběží. Tím je docíleno uvolnění hlavního vlákna, ve kterém běží grafické rozhraní a aplikace se tak nejeví jako zaseknutá.

Proces je rozdělený na 2 podprocesy, o kterých je pojednáno později v kapitole 3.3.4. Každý z těchto podprocesů je přiřazen k vláknu, které je spuštěno jako „`TaskCreationOptions.LongRunning`“ – což je pro systém známka, že toto vlákno má menší prioritu, co se potřebného výkonu týče, a pokud tedy počítač potřebuje použít výkon procesoru jinde, je mu to umožněno za cenu snížení výkonu pro toto vlákno. Instance vlákna je uložena do statického listu napříč celou třídou spolu s ostatními vlákny od každého souboru, důvod je popsán v kapitole 3.3.6.

Asi nejdůležitější součástí je rozdělení do dalších vláken uvnitř výše zmíněného vlákna. Je vytvořen list, obsahující stejný počet předpřipravených instancí pro vlákna, jako je počet testovaných veličin. Po předání hodnoty veličině je vybrána instance, která buď dokončila svoji předchozí práci, nebo zatím nebyla zaúkolována. Následně je jí práce přiřazena a vlákno je označeno jako „`TaskCreationOptions.AttachedToParent`“ (v hierarchii vláken to označuje podvlákno nějakého výše postaveného vlákna) a „`TaskCreationOptions.PreferFairness`“ (vlákna vytvořená dříve než toto, jsou zpracována dříve, a naopak). Tímto je dosaženo funkce, kdy každé testování dané

veličiny probíhá nezávisle na ostatních veličinách, nicméně pro jistotu je třeba uzamčení listu s načtenými hodnotami vůči přístupu (zápisu) stejné operace s nově načteným řádkem.

```
     if (childTasks != null)
     {
      //Select(in this order) index of task that currently has no assigned task
to do, or run to completion or is running
       Repeat:
       int index = childTasks.FindIndex(x => x == null || x.Status ==
TaskStatus.RanToCompletion || x.Status == TaskStatus.Running);
       if (index == -1)
       {
              Task.WaitAny(childTasks.ToArray());
              goto Repeat;
       }

       //Assignt work for selected task
       if (childTasks[index] == null)
       {
              childTasks[index] = Task.Factory.StartNew(delegate ()
              {
                     //Save data against concurent threads
                     DateTime dt = date;
                     string val = value;
                     //Start actual work
                     this.AddPoint(dt, val);
              }, TaskCreationOptions.AttachedToParent &
TaskCreationOptions.LongRunning & TaskCreationOptions.PreferFairness);
       }
       else
       {
              childTasks[index] = childTasks[index].ContinueWith((prevTask) =>
              {
                     DateTime dt = date;
                     string val = value;

                     this.AddPoint(dt, val);
              }, TaskContinuationOptions.AttachedToParent &
TaskContinuationOptions.LongRunning &
TaskContinuationOptions.OnlyOnRanToCompletion &
TaskContinuationOptions.PreferFairness);
       }
     }
     else this.AddPoint(date, value);
```

Zdrojový kód 1 : Rozdělení do podvláken

Tento multithreading jsem testoval čtyřmi různými přístupy a ve výsledku vyšel nejlépe třetí pokus, kde využívám „`.ContinueWith((prevTask)…`“. Díky tomu je docíleno, že se nemusí čekat na dokončení předchozího úkolu před zadáním nové práce

a tohoto času se může využít k načtení nového řádku. Zároveň při použití `Task` místo `ThreadPool` mám kontrolu nad prioritou daného vlákna vůči ostatním procesům počítače.

3.3.4 Hlavní výpočetní část

Skládá se ze dvou částí a v budoucnu může být poměrně jednoduše rozšířena o další. V každé z nich je nejdříve vynulován a nastaven progress bar příslušného souboru na odpovídající počet maximální hodnoty podle počtu řádků. Následně je spuštěno již popisované vlákno a uvnitř začíná daná operace. Nejprve je otevřen soubor, zkontroluje se, zda současný řádek, na kterém se má nacházet, odpovídá poslednímu známému řádku a případně jeho dosažení (přeskočení určitého počtu). Tato kontrola je zde proto, že uživatel má možnost proces v jeho průběhu pozastavit. Tato operace však uzavře soubor, přesto ale zůstane uložena informace o posledním řádku, na kterém se proces nacházel a toho je využito k dané kontrole či přeskočení. Dále je uvnitř cyklu `while` (jehož podmínkou je, zda byl dosažen konec souboru nebo uživatel vyžaduje pozastavení) načten nový řádek jako typ `string` a ten je ihned rozdělen po střednících, reprezentující oddělení buněk tabulky, do pole. První element je převeden na datum. Dále je ve smyčce `foreach` pro všechny vybrané veličiny zavolána příslušná funkce, ve které se odehrává jádro testu a výpočtu. Té je za pomoci indexu uloženého u dané veličiny předána vybraná hodnota na příslušném místě v poli rozdělených hodnot (odpovídá sloupcům) spolu s datem a instancí listu podvláken. Po každém dokončení cyklu je inkrementován progress bar – uživatel tak má přehled kolik z aktuálně běžícího procesu je zhruba hotovo a kolik zbývá. Pokud si uživatel zvolil zálohování procesu, tak je zkontrolován počet již načtených řádků a zavolána funkce na zálohování.

3.3.4.1 Hledání intervalu pro redukci intervalu zapisování

Poté, co je podvláknu uvolněn výkon procesoru, přejde u příslušné veličiny ke zpracování hodnoty, která mu byla předána. Vzhledem k tomu, že hodnota je po celou dobu jako typ `string` už od načtení a rozdělení do pole, tak je třeba ji převést do typu double, aby se s ní mohlo pracovat jako s číslem. Zde opět hraje roli dynamičnost aplikace, protože zdroj může brát desetinné číslo s čárkou ale i s tečkou, proto je v tomto kroku zakomponováno automatické převedení čárky na tečku, což neovlivní nic, pokud tam už tečka je. Tento textový řetězec je pak pokusem převeden na double.

Jestliže se převod nepovede, tak zbytek kódu neproběhne a je přeskočen, ale opakuje se při další načtené hodnotě. V opačném případě je přistoupeno k uzamčení instance s uloženými hodnotami, dokud není dokončena práce s ní.

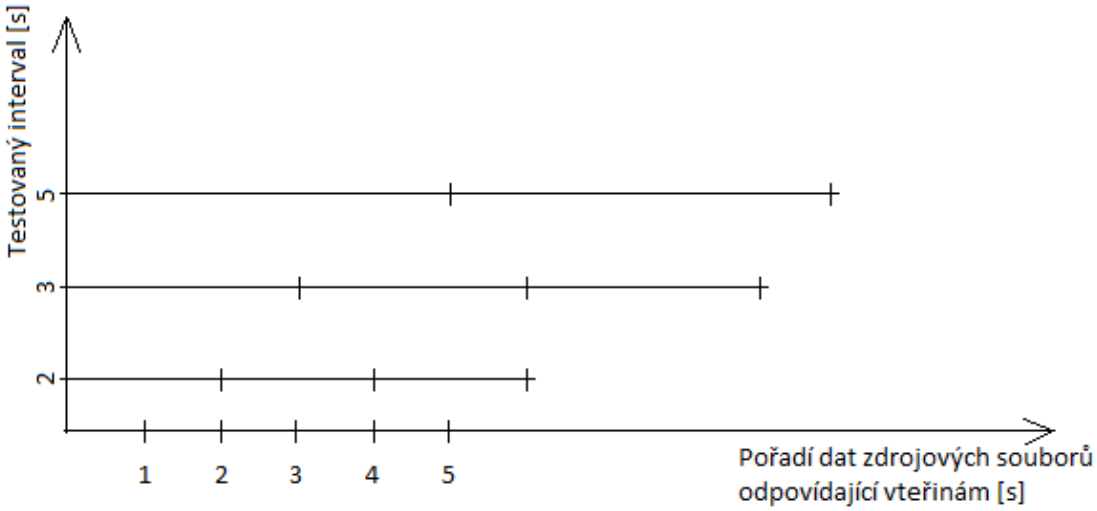
Myšlenka této části je taková, že hodnoty jsou uloženy v listu tak, jak jsou načítány. Tyto hodnoty nejprve musí splnit Sturgesovo pravidlo (požadující určitý počet mezi sebou se lišících hodnot v testovaném vzorku). Poté jsou od určitého počtu všechny otestovány, zda odpovídají normálnímu rozdělení. Pokud ano, jejich počet odpovídající intervalu je uložen a smaže se předchozí platný. Při načtení další hodnoty je pak tento počet zvýšen o jednu hodnotu a znovu otestován. Jestliže opět odpovídá, tak je vše opakováno při příštím načtení hodnoty atd., v opačném případě (test je vyhodnocen jako zamítnutý) zůstává uložena hodnota předchozího platného testu a list s načtenými hodnotami je vyčištěn. Tímto je docíleno postupného testování hodnot a uložení posledního platného počtu, na kterém test prošel do listu, jež později slouží k vykreslení histogramu a vyhodnocení průměrného platného intervalu. Ty jsou výsledkem této operace.

Samotné vyhodnocení je řešeno jako, aritmetický průměr uložených platných intervalů. Průměr je poté uložen do předpřipravené instance pro výsledky dané veličiny, kam jsou takto uloženy ostatní výsledky ze všech souborů. Finální výsledek je opět reprezentován aritmetickým průměrem těchto mezivýsledků.

3.3.4.2 Testování rozdělení

Stejně jako předchozí proces běží i tento na podvlákně. Jakmile je mu uvolněn výkon, zpracuje hodnotu příslušné veličiny stejným způsobem jako v předchozím

procesu a následně také uzamkne list.



Obrázek 3: Způsob testování

Obě metody striktně dodržují testování intervalů načtených hodnot, viz Obrázek 3, který říká, že žádná z hodnot není testována dvakrát pro stejný interval. Hodnoty jsou testovány způsobem jako by byly do zdrojového souboru zapisovány s příslušným intervalem, tudíž žádná hodnota není testována dvakrát pro ten samý interval.

Pro toto testování jsem zakomponoval i možnost náhodného výběru testovaných hodnot z načtených dat, které neodpovídá Obrázek 3. Uživatel si může tuto možnost zvolit v nastavení aplikace, o kterém je pojednáno v kapitole 3.3.12. Její vliv na výsledky je probrán v kapitole 4.6.

3.3.4.2.1 Metoda 1

Podmínkou pro vstup do výběru hodnot je, aby počet hodnot v listu s načtenými hodnotami byl vyšší než minimální počet vstupních dat pro testování rozdělení a odpovídal alespoň nejmenšímu testovanému intervalu.

Pro ne-náhodný výběr je pro každý testovaný interval vybráno X hodnot na základě uloženého indexu pozice v listu načtených hodnot poslední testované hodnoty. Následuje podmínka, která zkontroluje, zda vybraný počet souhlasí s počtem pro daný interval. Pokud ano, tak jsou vybrané hodnoty předány na otestování na jednotlivá

rozdělení a příslušnému testovanému intervalu je připsán index poslední vybrané hodnoty.

Jakmile je předchozí krok proveden pro všechny vybrané intervaly, tak je zkontrolováno, zda počet hodnot načtených v listu není příliš velký (odpovídající druhému násobku maximálního testovaného intervalu). Pokud je velký, je první polovina listu vymazána a indexy všech posledních indexu testovaných intervalů jsou sníženy o vymazaný počet.

Ještě před vymazáním první poloviny listu s načtenými hodnotami se nachází funkce pro náhodný výběr testovaných dat, která se provede pouze pokud je zvoleno náhodné testování. Její princip je podobný, pouze jsou hodnoty vybrány náhodně a přeposlány na otestování tolikrát, aby počet opakování těchto testů pro daný interval odpovídal opakování testu, jako by se jednalo o ne-náhodný výběr.

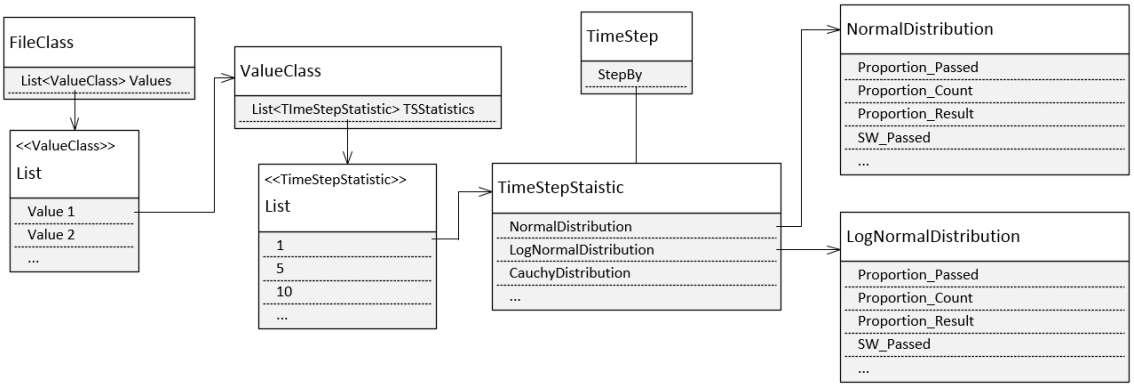
3.3.4.2.2 Metoda 2

Prvotní podmínka je stejná jako u první metody. Následně jsou vybrány intervaly, které odpovídají svým intervalem počtu načtených hodnot anebo je zbytek po dělení počtu načtených hodnot tímto intervalem roven nule. Dále je pro každý, takto vybraný interval vypočítán počet hodnot „`SampleAmount`“, který má být z listu s načtenými hodnotami extrahován. V dalším kroku je otestováno, zda vypočítané číslo je větší nebo rovno nejmenšímu počtu hodnot pro testování rozdělení, pokud ano, tak je přistoupenou k samotnému výběru hodnot. Pokud je v listu s načtenými hodnotami stejný počet jako je potřeba tak jsou tyto hodnoty rovnou vybrány, pokud ale ne, tak jsou v případě ne-náhodného výběru přeskočeny všechny hodnoty, krom posledních, odpovídající číslu „`SampleAmount`“.

Pro náhodný výběr je nejprve načteno množství hodnot odpovídající maximálnímu testovanému intervalu. Hodnoty nejsou přeskakovány, ale jsou vybírány náhodně z načtených.

Vymazání listu je řešeno podobně jako u první metody. Pokud počet načtených hodnot odpovídá dvounásobku maximálního testovaného intervalu, je první polovina z nich smazána.

3.3.4.3 Uložení mezivýsledků



Obrázek 4: Diagram ukládání mezivýsledků

Jak už jsem naznačil v kapitole 3.3.3.1, jednotlivé mezivýsledky jsou uloženy jako průměry výsledků testů, aby nezabíraly příliš RAM paměti. Součet všech testů daného parametru v jedné instanci a celkový počet testů ve druhé. Výsledek je zpětně získáván `readonly` funkcí, která součet vydělí počtem. Po předání k otestování je vybrána instance, která byla přidělena danému intervalu, veličině a souboru. Následně po samotném otestování, o němž je pojednáno v kapitole 3.3.5, je daný výsledek předán příslušné přidělené instanci třídy. Zde je vybráno, o jaké rozdělení se jedná a výsledek a jeho parametry jsou uloženy v instanci daného rozdělení a ihned poté uvolněny z paměti.

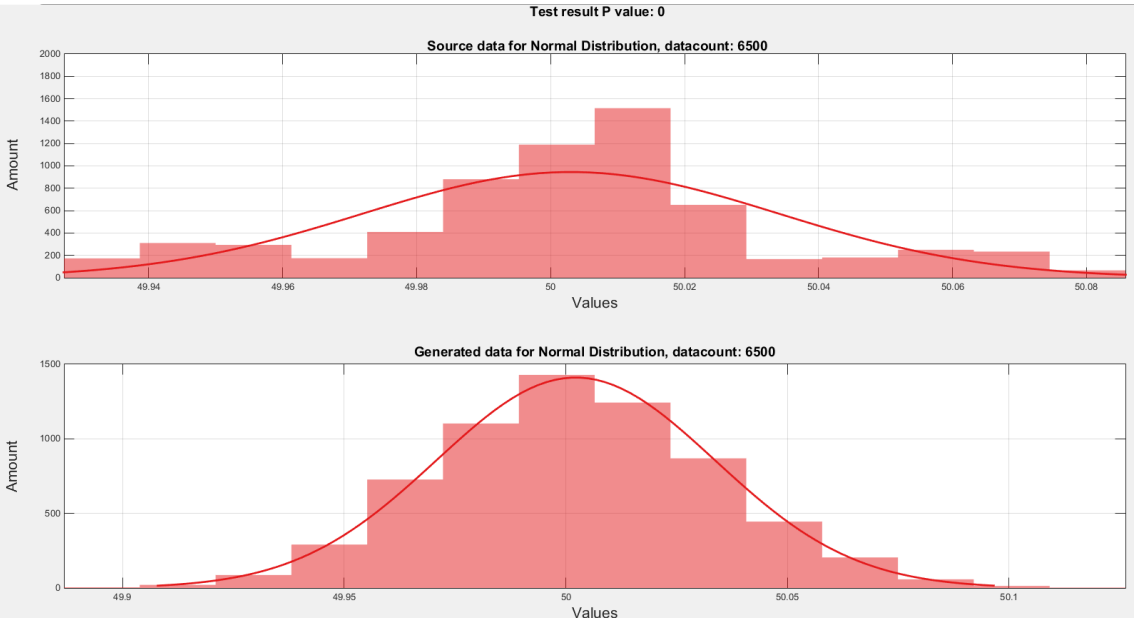
3.3.5 Testování rozdělení využitím knihovny Accord

```
      this._NormalDistribution = new NormalDistribution();
      _NormalDistribution.Fit(data);
      this._SWTest = new ShapiroWilkTest(data);
      NormalDistribution test = new NormalDistribution(_NormalDistribution.Mean,
_NormalDistribution.StandardDeviation);
      this._KSTest = new KolmogorovSmirnovTest(data, test);
      _2sampleKSTest = (!KS2s)? null: new TwoSampleKolmogorovSmirnovTest(data,
test.Generate(data.Count()));
      _ChSTest = (!ChS) ? null : new ChiSquareTest(data, test);
```

Zdrojový kód 2: Testování rozdělení

Instance rozdělení je vytvořena ve chvíli, kdy jí jsou předána nějaká data. Počet vzorků musí být větší než 4, jinak nelze testy provádět a zároveň nelze testovat hodnoty, které jsou naprosto identické, je nutno, aby se mezi nimi nacházela alespoň jedna rozdílná od ostatních.

Teoretická myšlenka, za funkcí algoritmu pro otestování, zda lze o testovaných hodnotách říci, že jsou daného rozdělení nebo ne, je taková, že vytvoříme rozdělení z těchto hodnot, Accord knihovna přesně pro tento účel nabízí funkci .Fit(double[] data). Vezmeme parametry takto vytvořeného rozdělení a ty použijeme pro vytvoření hypotetického rozdělení stejného druhu. Některá rozdělení nemohla být otestována, protože vyžadovala parametr, který se takto získat nedal. Například Weibullovo rozdělení pravděpodobnosti vyžadovalo již v prvním konstruktoru neznámé parametry, jež právě zjišťujeme pro vytvoření hypotetického. Následně otestujeme, zda se tyto dvě instance podobají pomocí testu Kolmogorov–Smirnov a Shapiro–Wilk. Z těchto testů získáme hodnotu p_value nulové hypotézy, která vyjadřuje pravděpodobnost shody. Pokud je p_value<0,05 (5 %), je hypotéza shody zamítnuta, pokud je větší, není vyvrácena. Roli alternativní hypotézy zde hrají testy na ostatní rozdělení. Vizuálně interpretovaný test viz Obrázek 5, jedná se vlastně o porovnání histogramů.



Obrázek 5: Testování rozdělení

Rozhodnutí o výsledku testu je určeno kombinací výše zmíněných testů hypotéz. Oba výsledky musí mít p_value vyšší než 0,05 a Significant (zda má být nulová hypotéza zamítnuta) rovno false.

3.3.6 Synchronizace procesů

```
      lock (FinalizingTask)
      {
       if (!ct.IsCancellationRequested && Computing_parts_current >= 1 &&
Computing_parts_current < 2)
      {
                  List<Task[]> wt = new List<Task[]>();
              //Selecting active files
                  foreach (FileClass fc in Form1.Files)
                  {
                           if (fc.Checkbox.Checked) wt.Add(fc.ComputingTask);
                  }

                  List<Task> waitingTasks = wt.SelectMany(x =>
x.ToArray()).ToList();
              if (!Recovering)
                  {
                    //Waiting for other threads to finish
                           while (waitingTasks.Any(x => x == null))
                    {
                           Thread.Sleep(1000);
                                  waitingTasks = wt.SelectMany(x =>
x.ToArray()).ToList(); //Refreshing list
                                 }
                  }
                  Task.WaitAll(waitingTasks.ToArray());
              //Start finalization proces
              FinalizingTask[0] = Task.Factory.StartNew(delegate ()
                  {
                           Computing_part_ONE_finalize(rep);
                  });
      Task.WaitAll(FinalizingTask); //Waiting for finalization to end
       }
      }
```

Zdrojový kód 3: Synchronizace procesů

Samotné výpočetní procesy jednotlivých souborů běží nezávisle na ostatních. Synchronizace je vyžadována pouze pro shrnutí výsledků a jejich předání do MATLABu k vykreslení výsledných grafů a to tak, že je nutno, aby veškeré testy pro jednotlivé soubory byly dokončeny, než se přejde k tomuto shrnutí.

Jak již bylo zmíněno, každý výpočetní proces jednotlivých souborů běží v samostatném vlákně. Instance tohoto vlákna je uložena ve statickém listu, který slouží ke kontrole, zda všechny nebo nějaké vlákno dokončilo svůj proces. Po spuštění procesu výpočtu každý soubor čeká, až nějaké vlákno procesu dokončí svou práci. Jakmile se tak stane, uzamkne instanci vlákna, které slouží ke shrnutí výsledků vůči

ostatním souborům. Následně počká na všechny ostatní procesy, než dokončí svou práci, až teprve poté spustí proces, ve kterém dojde ke shrnutí výsledků.

Uvnitř tohoto procesu je zvýšeno číslo aktuálního procesu pro všechny aktivní soubory (procesy jsou totiž rozděleny na výpočetní a shrnovací části a patřičně očíslovány). K tomuto číslu se vztahuje kontrola uvnitř uzamčeného čekání na ostatní procesy, díky ní je pro všechny aktivní soubory zajištěno, že po dokončení shrnutí výsledků ostatní soubory, čekající na uvolnění přístupu k uzamčené instanci vlákna, přeskočí samotný proces shrnování a mohou přejít k dalšímu výpočetnímu procesu.

3.3.7 Příprava/shrnutí dat pro MATLAB

Řešení přípravy dat pro přesun do prostředí MATLAB je lehce komplexnější, protože jde ruku v ruce s vytvořenými funkcemi, které jsou později zavolány. Bylo třeba zjistit v jakém formátu, jak indexované a zkrátka jak daná data vypadají před zavoláním funkce a jak vypadají uvnitř, když jsou jí načteny. MATLAB je spíše maticově orientované prostředí, což jsem se snažil využít ve svůj prospěch. Problém ale nastal v přesunu dat. Konkrétně se jednalo o matici vytvořenou v C#, kterou se nedařilo správně přesunout při zachování stejného formátu. Proto jsem se rozhodl ji rozdělit na dvě části, v prvním parametru předat data jako hodnoty na ose X, ve druhém pak seznam počtu, kde jednotlivé elementy odpovídají počtu hodnot pro daný celek dat na ose X a stejně tak i pro osu Y. Jednotlivé popisky jsou pak umístěny v dalším seznamu, který má pokaždé stejný počet dat pro daný soubor. Výhodou návrhu tohoto řešení je, že obsahuje další prvek univerzality. Místo aby každý budoucí graf byl jako jeden řádek v matici pro X, Y a popisky, jsou jejich data za sebou a informace o délce jednoho řádku je nesena v dalším parametru. Základní struktura je zachována bez ohledu na vstupní data, mění se pouze počet těchto dat pro osu X a Y, který je ale sledován a daná informace je také předána.

Třída pro přípravu si všechny potřebné informace získá z předaných dat sama. Samotné předání dat je pouze uložení daných dat do listu. Tato data jsou připravena od každého souboru a veličiny zvlášť, ale stejné veličiny jsou shrnuty do jedné instance přípravné třídy.

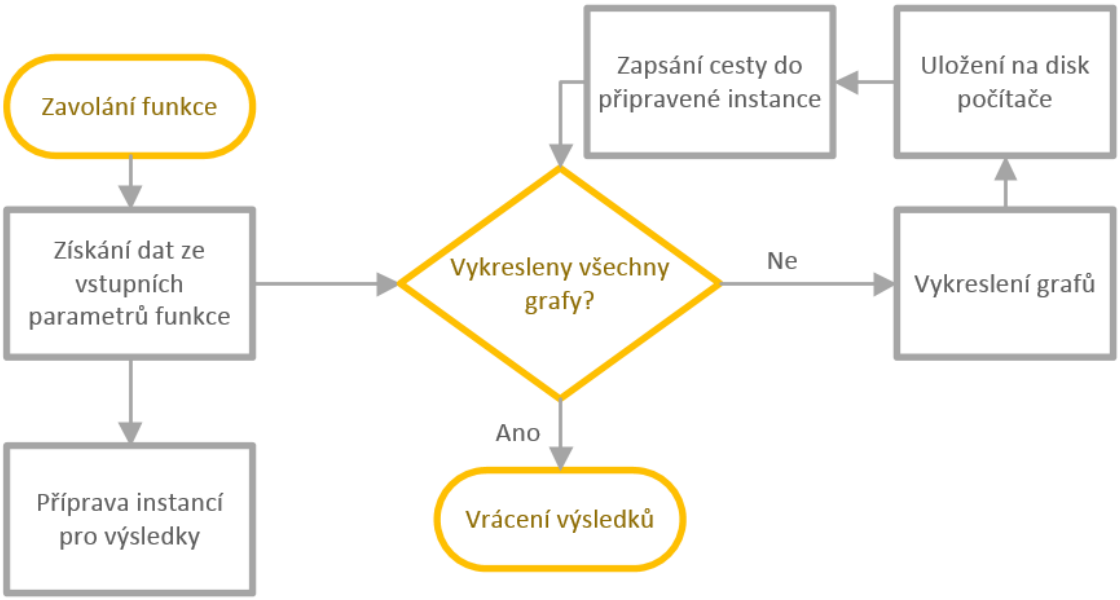
```
      List<double> times = new List<double>(); //osa X
      List<double> NormProportion = new List<double>(); // 1. Osa Y
      List<double> NormSWP_value = new List<double>(); // 2. Osa Y

      foreach (TimeStepStatistic ts in this.TSStatistic)
       {
       times.Add(Convert.ToDouble(ts.timeStep.StepBy));
       NormProportion.Add(ts.NormalDistribution.NormProportion);
       NormSWP_value.Add(ts.NormalDistribution.NormSWP_value);
       .
       .
       .
       }
      string d = " Distribution";
      string DistName = "Normal";
      Distribution dist = CheckForDist(DistName + d);

      if (AppSettings.Default.Testing_NormalDist)
      {
       dist.PreparePicture(matlabMatrix.Add_All(times, NormProportion, DistName +
Picture.Separator + "Proportion", "interval [s]", "passed/notpassed [%]",
this.FileOwner.FileName, dist.Name));
       dist.PreparePicture(matlabMatrix.Add_All(times, NormSWP_value, DistName +
Picture.Separator + "Shapiro-Wilk P value", "interval [s]", "P value",
this.FileOwner.FileName, dist.Name));
```

Zdrojový kód 4: Příprava pro vykreslení grafů

Nejprve jsou uloženy do listu testované intervaly, které budou sloužit jako osa X. Následně jsou uloženy data pro osu Y s tím, že každý graf zvlášť. Poslední jsou popisky os, jméno grafu, rozdělení a souboru, který je původcem předchozích dat. Zároveň je připraveno místo pro uložení výstupu z MATLABu – umístění uloženého grafu.

3.3.8 MATLAB skripty

Jakmile jsou připravena shrnutá data pro vykreslení grafů, jsou předány MATLABu k vykreslení grafů. Vytvořil jsem tři hlavní m-fily, první pro vykreslení grafů s logaritmickým měřítkem na ose X, druhý pro vykreslení sloupcových grafů a poslední pro vykreslení jednoho nebo dvou (pro porovnání) histogramů.



Obrázek 6: Diagram zjednodušené funkce MATLAB scriptů

Všechny tři m-fily mají společný princip. V první řadě si zpětně převedou do matice vstupní data tak, že vytvoří prázdnou matici o rozměrech, které jsou známy a předem zjištěny v C#, a následně ji naplní daty. Hlavní vykreslovací část poté funguje tak, že na základě známé hodnoty počtu grafů jsou vybírány data pro vykreslení do daného grafu. Hlavní cyklus `for` prochází různými typy grafů, uvnitř něj je vnořený další cyklus `for`, který podle výše zmíněné hodnoty počtu grafů listuje mezi daty jednotlivých souborů.

Grafy jsou ukládány do složky automaticky a pojmenovány podle typu veličiny, statistického rozdělení a jeho parametru, případně svého vlastního jména. Cestu k takto uloženému obrázku je nutno předat jako výstupní informaci z m-filu, aby mohl být později vložen do výstupního PDF souboru v C#, což je řešeno předpřipraveným, prázdným polem, do něhož jsou tyto cesty ukládány v průběhu a těsně po uložení daného grafu. Stejně tak jsou vráceny i průměry mezi daty na osách X a Y jako výstup z m-filu pomocí MATLAB funkce mean

Z důvodu uložení v co nejlepším rozlišení je těsně před uložením celá obrazovka s grafem zvětšena do maximální velikosti. Nicméně grafy jsou vytvářeny na pozadí, takže neruší uživatele při jeho práci.

Ve výchozím nastavení se legenda nachází v pravé horní části grafu, kde poměrně často překáží a zakrývá tak samotný graf. V MATLABu sice existují funkce, které ji přesunou vedle daného grafu, ale za cenu jeho zmenšení horizontální či vertikální velikosti. Proto jsem navrhl řešení, kdy na konci do PDF vypisuji legendu zvlášť od ostatních grafů, protože všechny grafy ji mají stejnou. Kvůli tomu je nutné stejně jako samotné grafy ji uložit, ale „vystřiženou“ z obrazovky, bez grafu samotného a také předat cestu, kam a pod jakým jménem byla uložena. Toho jsem docílil přesunutím grafu mimo viditelnou oblast na obrazovce a posunutím legendy na jeho místo. Zároveň je nutno upravit velikost okna na velikost dané legendy, jinak je uložený obrázek z velké části prázdný, ale v dokumentu zbytečně zabírá místo. Po jejím uložení je vše navráceno do původních pozic a velikostí a legenda schována.

3.3.9 Uložení výsledků z MATLABu

Bylo nutné přijít s návrhem, jak ukládat data z MATLABu, aby byla později snadno dosažitelná a extrahovatelná při vypisování finálního PDF souboru. Proto jsem vytvořil knihovnu sloužící pro ukládání těchto výsledků, která je strukturovaná tak, že nejvíce nadřazenou roli hraje interval zapsaných dat ve zdrojových souborech, protože na základě pozorování vím, že hraje největší roli ve výsledcích a ty se pak mezi těmito různými intervaly liší. Pod tento interval pak patří název zkoumané veličiny, jež obsahuje jednotlivá zkoumaná rozdělení. Tato rozdělení obsahují listy, do kterých byly připraveny instance pro budoucí cesty k uloženým grafům na základě jejich jmen.

Po získání výstupu z MATLABu jsou dané výsledky rozděleny podle jmen souborů a zapsány do výše zmíněných připravených instancí právě na základě jejich jmen. U některých obrázků jsou důležitá i data os X a Y, například protože podle nich je rozhodnuto, kterému rozdělení tato veličina odpovídá porovnáním stejného typu grafu různých rozdělení. Tyto hodnoty jsou pak uloženy v listech.

3.3.10 Výpočet chyb pro průměrování

Pro výpočet chyby je nutné alespoň deset naměřených hodnot. Zvolil jsem si, že pokud bude méně než 10 testovaných souborů, bude výpočet opakován. Naopak když jich je alespoň 10, tak to pro výpočet stačí. Samotná hodnota je vypočtena z průměrů

výsledků výpočtu intervalu pro průměrování (kapitola 3.3.4.1) každého souboru podle Rovnice 1. Implementace zdrojového kódu viz Zdrojový kód 6: Výpočet chyby.

$$u_A = \sqrt{\frac{\sum_{i=1}^{n}(x_i-\bar{x})^2}{n(n-1)}} \qquad u_B = \frac{max-min}{\sqrt{6}} \qquad Error = \sqrt{(u_A)^2 - (u_B)^2}$$

Rovnice 1: Výpočet chyby

Zaokrouhlování chyby a výsledku podle norem zatím vyřešeno není. Tato funkce byla přidána na poslední chvíli a na samotné naprogramování správného zaokrouhlování již nezbyl čas.

3.3.11 Mechanismus zálohování

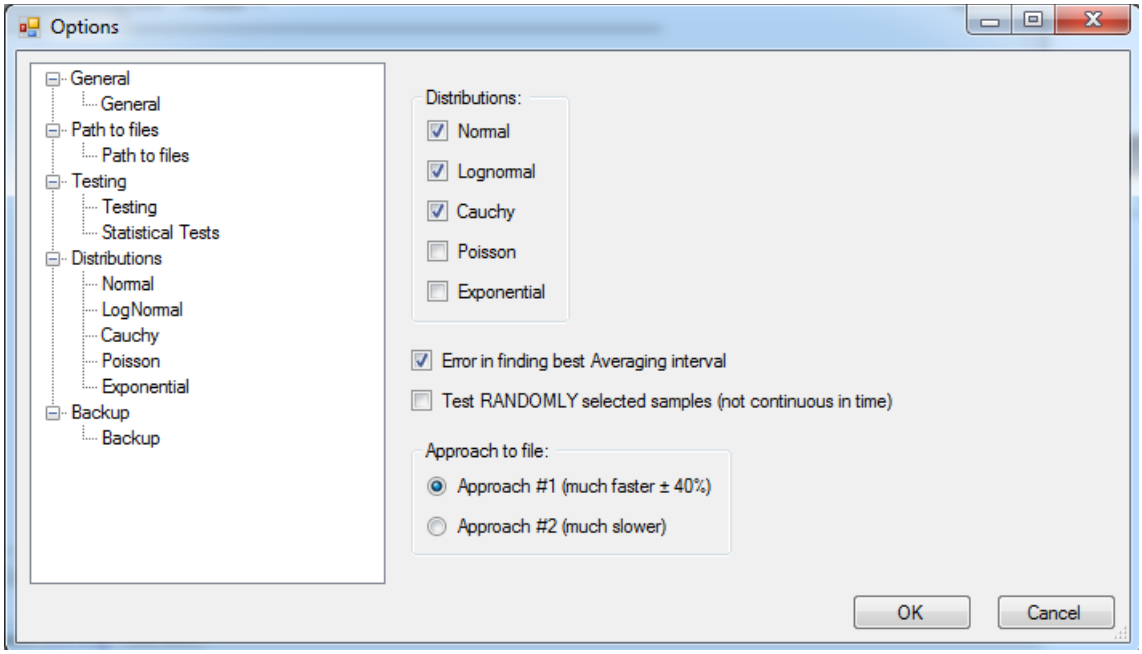
Z důvodu vysoké časové náročnosti výpočetních procesů (trvajících i několik desítek hodin) jsem považoval za nutné zakomponovat do programu zálohovací mechanismus, který bude průběžně ukládat výsledky a mezivýsledky, které bude možné načíst v případě výpadku elektřiny, modré smrti či jiných externích záležitostí, které by proces přerušily

Toto zálohování jsem vyřešil ukládáním mezivýsledků do CSV souboru po zpracování každých n řádů ze zdrojového souboru případně po vykreslení výsledných grafů jednotlivých procesů. Takto jsou ukládána nezbytná data, o jaký soubor se jedná, v jaké části procesu se právě nachází, výsledky v podobě cest k vykresleným grafům, a především samotné mezivýsledky nutné pro výsledné shrnutí.

Zpětné načítání ze souboru zálohy načte k příslušným souborům a veličinám jejich mezivýsledky a samotné výsledky do tříd, které jsou určeny pro shrnování výsledků. Po načtení se aplikace prakticky vrátí do posledního známého stavu se všemi daty, jako by žádné přerušení nenastalo. Opětovné spuštění pak pokračuje od tohoto bodu.

Opět zde hraje roli univerzalita, která se setkala s problémem maximálního počtu znaků v CSV buňce, to ale bylo vyřešeno rozdělením na více buněk a uložením daného počtu.

3.3.12 Uživatelské nastavení (Options)



Obrázek 7: Uživatelské nastavení

Do aplikace jsem se rozhodl zakomponovat i formu uživatelského nastavení. Zde si uživatel může vybrat, co chce testovat, jaká rozdělení a jaké parametry vypisovat do výsledného PDF atd. Protože jsem nenašel žádný již hotový plugin nebo knihovnu, která by něco takového zprostředkovala, musel jsem si vytvořit vlastní.

Na ukládání nastavení do paměti počítače využívám AppSettings, které již VS obsahuje právě pro takové případy. Samotný návrh poté pracuje na principu přiřazení eventu ke Controls, které to tak vyžadují, a vyvolání daného eventu poté změní AppSettings také přiřazené k tomuto Controls. Tvorba okna je řízena vlastním algoritmem, který řádky pozicuje. Řádek je vytvořen přidáním Controls do příslušné, tomu určené, třídy a instance takto vytvořeného řádku a může (ale také nemusí) být umístěna v samostatném GroupBoxu.

AppSettings je uloženo až v případě, kdy uživatel potvrdí změny přes tlačítko OK, v opačném případě jsou změny ztraceny. Aplikace na dané AppSettings reaguje a řídí se jimi bez nutnosti změny zdrojového kódu.

3.4 Výstupní PDF soubor

Výsledkem všech procesů této aplikace je PDF soubor, do kterého jsou vypsány všechny výsledky ve formě grafů. Ty jsou seřazeny podle veličin a testovaných rozdělení. První část souboru obsahuje jednotlivé veličiny s jejich grafy a rozděleními, druhá část obsahuje porovnání všech veličin mezi sebou pro stejný typ grafu.

Pořadí informací je vždy stejné, neboť PDF je generováno automaticky aplikací. První část je rozdělena do sekcí, rozdělených po intervalu, jakým jsou zapisovány hodnoty do zdrojových souborů (pokud se tedy nějaký liší od ostatních). V těchto sekcích jsou poté testované veličiny. Prvním vykresleným obrázkem jsou legendy histogramu a logaritmických čárových grafů. Důvod je uveden v kapitole 3.3.8. Následuje vypočtený ideální interval pro průměrování dané veličiny a hned pod ní je histogram výskytu prošlých testů, ze kterých byl tento interval vypočten. Pod histogramem se nachází sloupcový graf porovnání, jak daná veličina odpovídá příslušným, testovaným, statistickým rozdělením v závislosti na výsledcích hypotézy o shodě mezi testovanými vzorky a příslušným rozdělením. Následují testovaná rozdělení s patřičným nadpisem, pod nimiž se nachází shrnutí grafů parametrů a výsledků daného rozdělení.

4 Výsledky testů a experimentů

4.1 Použité soubory

Pro testy bylo použito 20 souborů typu CMS ve formátu CEA, ze kterých bylo v prostředí ENVIS vyexportováno pouze několik, pro testy užitečných, veličin do lépe zpracovatelného formátu CSV. Jejich struktura zůstala stejná spolu s intervalem zapsaných hodnot, a to po 200 ms. Všechny testované soubory jsou z jednoho místa měření, o kterém víme, že má poměrně dobrou kvalitu elektrické energie.

Pro testy sudých harmonických bylo použito 11 souborů typu SMZ, jejichž data jsou ukládána po jedné vteřině a jedná se o průměr této vteřiny.

4.2 Zvolené testované intervaly

Zvolil jsem přibližně 30 intervalů od 0 do 3600 s s narůstajícím rozdílem mezi hodnotami, aby odpovídaly logaritmickému měřítku. Volit vyšší interval bylo zbytečné, protože testy KS a SW selhávají pro vysoké počty testovaných vzorků, o čemž jsem se přesvědčil před spuštěním hlavních testů.

Bylo by zajímavé zjistit, jak se pravděpodobnost, že veličina odpovídá určitému rozdělení, bude chovat pro intervaly daleko větší, než je 3600 s, například 12 hodin, 1 den, týden atd. To ale není dostupnými nástroji možné zjistit z důvodu velkého počtu testovaných vzorků a samotných testů hypotéz.

4.3 Testovaná rozdělení pravděpodobnosti

Pro testy jsem vybral normální (Gaussovo), lognormální a Cauchyho rozdělení pravděpodobnosti. Exponenciální rozdělení sice testovat šlo, ale p value výsledků se pohybovala na celém testovaném intervalu okolo 0, takže jsem ho z testu také vyřadil.

Některá rozdělení jako například gamma nešlo otestovat z důvodu nemožného vytvoření teoretického rozdělení z parametrů testovaného (viz kapitola 3.3.5). Vyzkoušel jsem i nespojité Poissonovo rozdělení, ale už podle jeho histogramu (viz Obrázek 18) je zřejmé, že nemá smysl ho testovat i nadále.

4.4 Časové nároky

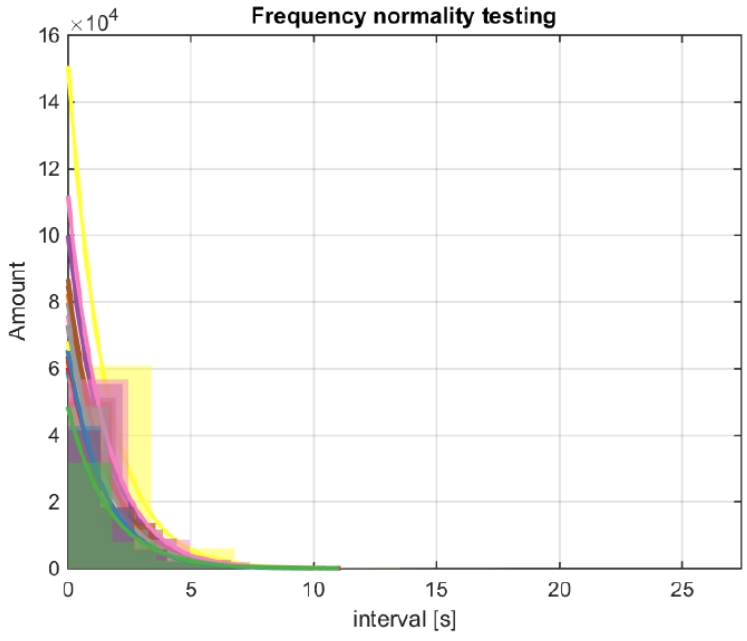
I přes stálé využívání 95–100 % výkonu procesoru trvala většina výpočtů v rozsahu od několika minut až po tři dny. Výpočetní čas narůstal lineárně s počtem testovaných souborů a počtem jejich řádků, ale exponenciálně s počtem testovaných veličin. Velice užitečným se ukázal záložní mechanismus, díky kterému bylo možné doopravit některé závažnější i menší chyby ve zdrojovém kódu, stejně tak jako pokračovat po výpadku elektřiny bez nutnosti restartování celého procesu a ztráty cenného času.

4.5 Obecný význam výsledků v PDF

Jak již bylo uvedeno v kapitole 3.3.8, legendy odpovídají všem grafům dané veličiny, protože v grafech samotných by svou velikostí zakrývaly důležité informace, proto je uvedena zvlášť. Příklad legendy viz Obrázek 12: Příklad legendy.

4.5.1 Hledání intervalu pro průměrování

```
Best interval for averaging (based on seconds): 7,575±0,467 s
Best interval for averaging (based on file): 1,515±0,093 s
```



Obrázek 8: Příklad výsledku hledání intervalu pro průměrování

Hledání intervalu pro průměrování souboru tak, aby odpovídal normálnímu rozdělení, je řešeno funkcí popsanou v kapitole 3.3.4.1. Výsledkem je histogram zobrazující četnost (osa Y) vypočtených intervalů (osa X). V převážné většině výsledků měl tento histogram exponenciální spád, proto jsou vyneseny i exponenciální křivky.

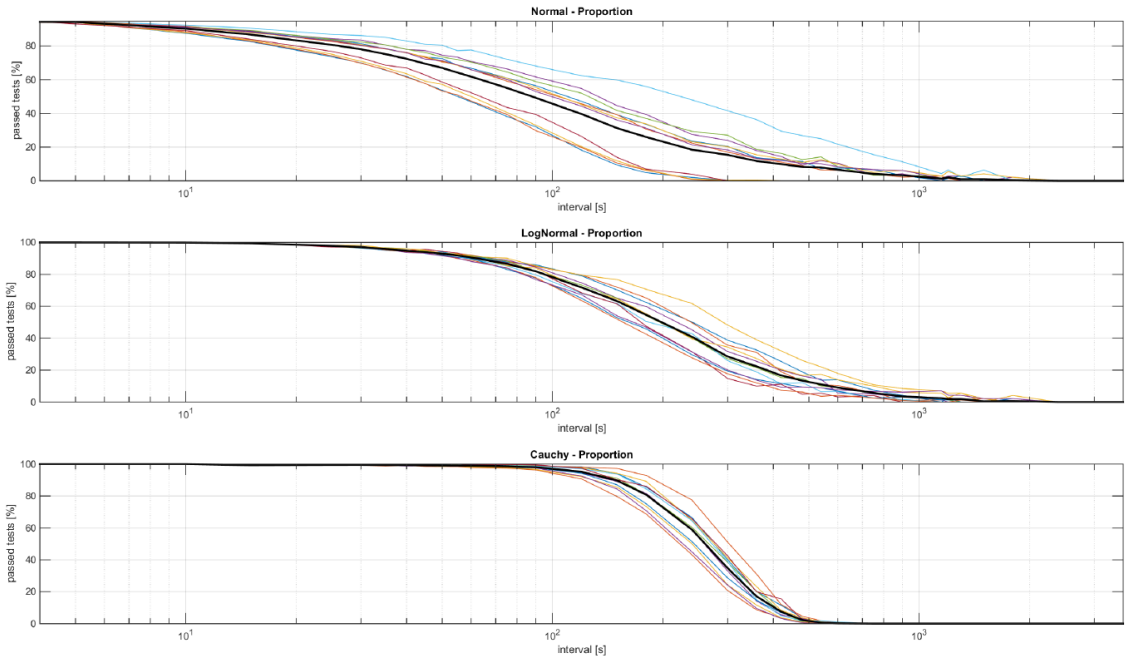
Tento graf hluboce závisí na intervalu zapsaných hodnot ve zdrojovém souboru. Pokud má daný soubor vysoký interval zápisu, např. 60 s poté začátek osy X nemůže být v 0 s, ale bude posunut do času, který odpovídá nejmenšímu možného počtu vzorků pro otestování (Accord knihovna vyžaduje čtyři), tím pádem do 240 s. Výpočet je tudíž ovlivněn intervalem zápisu a lze využít průměrování do nekonečna, kdy pokud zprůměrujeme soubor a otestujeme ho znovu, získáme výsledek, podle kterého ho můžeme zprůměrovat znovu a tak dále. Tím ale přijdeme o kritické hodnoty a po nekonečném průměrování bychom dostali souvislou neměnnou čáru, která už neodpovídá původnímu časovému průběhu dané veličiny.

Z důvodu popsaného v předchozím odstavci se nad histogramem nachází dva číselné výsledky:

- **„Based on seconds“** odpovídá průměru všech výsledků jednotlivých testovaných souborů. Jedná se o výsledek „based on file“, který byl přepočítán číslem, získaným z intervalu zapsaných hodnot ve zdrojovém souboru. Tento výsledek odpovídá vhodnému intervalu nehledě na interval zápisu zdrojového souboru a může být i menší.
- **„Based on file“** odpovídá průměru výsledků jednotlivých testovaných souborů na základě jejich intervalu zápisu. Vždy bude větší, než je původní interval.

Výpočet nejistot je uveden v kapitole 3.3.10.

4.5.2 Testování rozdělení pravděpodobnosti



Obrázek 9: Příklad porovnání výsledků rozdělení

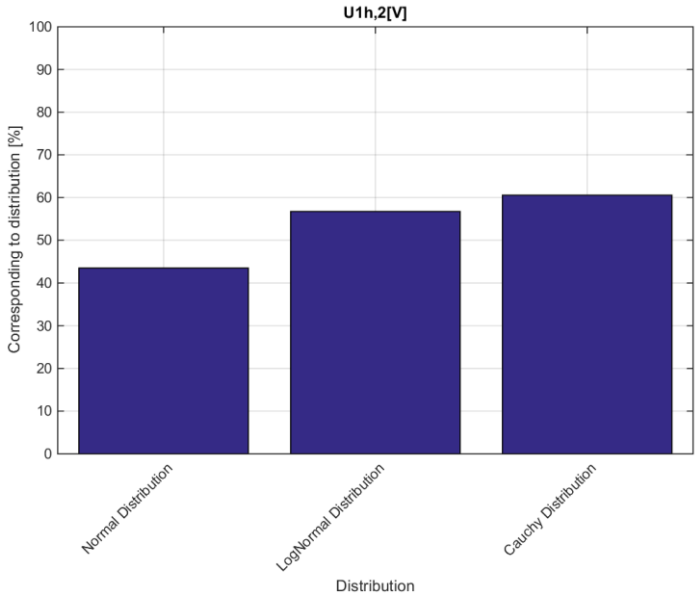
Způsob testování rozdělení je popsán v kapitole 3.3.4.2. O výsledku rozhodují grafy pojmenované „Proportion“. Příklad takového porovnání viz Obrázek 9. Tyto grafy vycházejí z procentuálního výsledku testů na daném intervalu (prošlé děleno celkovým počtem).

$$R[\%] = \frac{\int_{\ln(\min(interval))}^{\ln(\max(interval))} f(x)dx}{\int_{\ln(\min(interval))}^{\ln(\max(interval))} (100)dx} \cdot 100$$

Rovnice 2: Výpočet odpovědnosti rozdělení

Odpovědnost k rozdělení v grafu viz Obrázek 10 je spočítána podle Rovnice 2, kde $f(x)$ reprezentuje výslednou (avarage) funkci příslušného rozdělení v „Proportion“ grafech (viz Obrázek 9) a limity jsou přepočítány, aby odpovídaly logaritmickému měřítku. R[%] je vyneseno na ose Y pro příslušné rozdělení. Implementace integrálu ve zdrojovém kódu (viz Zdrojový kód 5: Způsob Integrace) je řešena pro každou inkrementaci o velikosti rozdílu předchozího a současného intervalu jako obsah trojúhelníku + obsah obdélníku. Dělitel reprezentuje celkovou plochu grafu. Výsledkem

je procentuální odpovědnost veličiny danému rozdělení na intervalu od nejmenšího zvoleného po největší zvolený.



Obrázek 10: Příklad výsledků pravděpodobnostního rozdělení

Tento výsledek se může lišit v závislosti na zvolených testovaných intervalech. Platí, že čím menší rozestupy mezi nimi se zvolí, tím přesnější bude výsledek. Logicky lze očekávat, že pokud by byl zvolen maximální interval větší než 3600 s, potom bude procentuální odpovědnost klesat z důvodu zvětšení plochy grafu.

Stejně jako při hledání intervalu pro průměrování (kapitola 4.5.1) zde platí velká závislost na intervalu zápisu zdrojových dat. V tomto případě je to způsobeno statistickými testy a způsobem, jakým fungují. Po předchozím zprůměrování budou příslušným rozdělením odpovídat i na intervalech, kde dříve neodpovídaly a to proto, že se změní počet testovaných vzorků a je pravděpodobné, že menší počet vzorků hypotetickým testem projde oproti většímu počtu z předchozího testování. Nicméně by měl zůstat zachován výsledek, kterému rozdělení daná veličina odpovídá.

4.5.3 Ostatní grafy

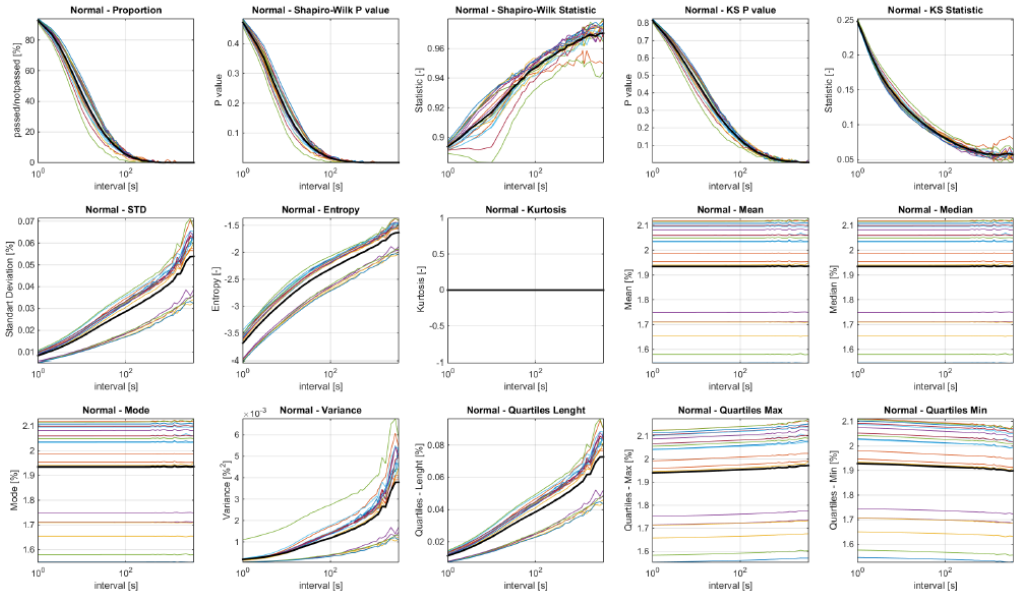
V PDF se dále kromě „Proportion“ grafů nacházejí i další grafy, které zobrazují, jak se chovají různé parametry (popis viz kapitola 2.4) daných rozdělení, které obsahuje knihovna Accord, na příslušných intervalech. V každé kapitole, respektive pro každou

veličinu, je vykresleno pouze shrnutí těchto grafů viz Obrázek 11. Jejich detaily jsou ve spodní části dokumentu a jsou zobrazeny jako přehled daného parametru pro všechny testované veličiny.

Mezi nejvýznamnější grafy patří ty, které zobrazují chování „P_value“ na testovaných intervalech pro daná rozdělení. Od těchto grafů se totiž odvíjí grafy „Proportion“, protože p value rozhoduje o výsledcích testů.

Všechny grafy jsou vykreslovány s osou X v logaritmickém měřítku, MATLAB ale toto vykreslení nedodržuje od určité velikosti grafu, a proto v některých grafech nejsou osy X vykresleny tak, jak by měli být. V detailech poté už ano.

A) Normal Distribution (Value: avg.THD-RU2[%])



Obrázek 11: Příklad shrnutí parametrů

4.6 Experiment s náhodným výběrem

Ukázalo se, že náhodný výběr testovaných vzorků má velký vliv na to, kterému rozdělení daná veličina odpovídá nejlépe. Otestovány byly sudé harmonické třetí fáze napětí a 3 fáze spolu se sdruženými napětími. Porovnání výsledných grafů viz Obrázek 13 a Obrázek 14.

Zajímavým výsledkem experimentu je parametr statistic, který se při spojitém testování chová pro každou veličinu různě, ale při náhodném testování má pokaždé velice podobný průběh viz Obrázek 15.

4.7 Konkrétní výsledky testů

Soubory se všemi výslednými PDF dokumenty jsou na přiloženém CD ve složce „Výsledné PDF soubory“. Některé z výsledků jsou uvedeny v tabulkách. Pro všechny veličiny a rozdělení platí, že p value klesá se zvyšujícím se testovaným intervalem.

4.7.1 Soubory SMC

Většina veličin odpovídá z testovaných rozdělení pravděpodobnosti nejvíce log-normálnímu rozdělení.

Liché harmonické neodpovídají ani jednou normálnímu. První a třetí harmonická se od ostatních vyšších liší, ty už odpovídají velice podobně log-normálnímu. Třetí jako jediná odpovídá Cauchyho rozdělení. Pro druhou a třetí fázi odpovídají Cauchyho rozdělení první a třetí harmonická, zbytek je rovněž log-normální. Tento soubor neobsahoval sudé harmonické, proto nemohly být otestovány.

Veličina	avg.U1 [V]	avg.U2 [V]	avg.U3 [V]	avg.U12 [V]	avg.U23 [V]	avg.U13 [V]
Výsledné rozdělení	Log-norm	Cauchy	Cauchy	Log-norm	Log-norm	Log-norm

Veličina	U1h,1 [V]	U1h,3 [V]	U1h,5 [V]	U1h,7 [V]	U1h,9 [V]	U1h,11 [V]
Výsledné rozdělení	Log-norm	Cauchy	Log-norm	Log-norm	Log-norm	Log-norm

Tabulka 1: Výsledky testů rozdělení SMC souborů

	Veličina	avg.U1 [V]	avg.U2 [V]	avg.U3 [V]	avg.U12 [V]	avg.U23 [V]	avg.U13 [V]
Interval průmě-rování	BoS [s]	9,5	8,5	7,5	12,1	9,4	13,6
	BoF [s]	1,9	1,7	1,5	2,4	1,9	2,7

	Veličina	U1h,1 [V]	U1h,3 [V]	U1h,5 [V]	U1h,7 [V]	U1h,9 [V]	U1h,11 [V]
Interval průmě-rování	BoS [s]	9,5	12,4	19,1	17,8	18,4	17
	BoF [s]	1,9	2,5	3,8	3,6	3,7	3,4

Průměry	BoS [s]	13,992857
	BoF [s]	2,8

BoS - Based on seconds

BoF - Based on file

Tabulka 2: Výsledky hledání intervalu průměrování SMC souborů

Výsledek výpočtu pro průměrování se pro „Based on seconds“ rovná 14 s a pro „Based on file“ 2,8 s.

4.7.2 Soubory SMZ

Pro fáze a první liché harmonické první fáze vychází ve všech případech log-normální rozdělení. Sudé harmonické všech třech fází vychází nejvíce Cauchyho rozdělení. U testů tohoto typu souboru mají výsledky vyšší procentuální zastoupení z důvodu intervalu zápisu dat zdrojových souborů 1 s oproti zdrojovým souborům SMC s 200 ms zápisem aktuální hodnoty.

Veličina	avg.U1 [V]	avg.U2 [V]	avg.U3 [V]	avg.U12 [V]	avg.U23 [V]	avg.U13 [V]
Výsledné rozdělení	Log-norm	Log-norm	Log-norm	Log-norm	Log-norm	Log-norm

Veličina	U1h,1 [V]	U1h,3 [V]	U1h,5 [V]	U1h,7 [V]	U1h,9 [V]	U1h,11 [V]
Výsledné rozdělení	Log-norm	Log-norm	Log-norm	Log-norm	Log-norm	Log-norm

Tabulka 3: Výsledky testů rozdělení SMZ souborů

	Veličina	avg.U1 [V]	avg.U2 [V]	avg.U3 [V]	avg.U12 [V]	avg.U23 [V]	avg.U13 [V]
Interval průměrování	BoS [s]	11,9	8,5	9,6	10,9	16,2	17,8
	BoF [s]	11,9	8,5	9,6	10,9	16,2	17,8

	Veličina	U1h,1 [V]	U1h,3 [V]	U1h,5 [V]	U1h,7 [V]	U1h,9 [V]	U1h,11 [V]
Interval průměrování	BoS [s]	11,9	29,8	19,8	19,3	22	20,7
	BoF [s]	11,9	29,8	19,8	19,3	22	20,7

Průměry	BoS [s]	17,6
	BoF [s]	17,6

BoS - Based on seconds

BoF - Based on file

Tabulka 4: Výsledky hledání intervalu průměrování SMZ souborů

Výsledek výpočtu pro průměrování se pro „Based on seconds“ rovná 17,6 s a pro „Based on file“ stejně. Ačkoliv se může zdát, že by BoS mělo vyjít stejně nebo podobně jako pro SMC soubory, je nutno brát v potaz, že se jedná o jiný způsob ukládání dat do zdrojových souborů, ne pouze jiný interval.

4.7.3 Vlastnosti grafů

Při spojitém testování se směrodatná odchylka, entropie, variance a quartily chovají pro všechny testované veličiny stejně. Jejich křivka nabývá na hodnotě se zvyšujícím se počtem testovaných vzorků, respektive velikosti testovaného intervalu. Obzvlášť proměnlivý je parametr statistic, jehož průběh je pro každou veličinu jiný. Průměry a poloha se naopak nemění.

Při náhodném testování jsou pouze směrodatná odchylka, entropie a 50 % kvartil rozdílné od ostatních, a to pouze na počátku testovaného intervalu přibližně do intervalu 10 s. Zvláštně se ale chovají pro Cauchyho rozdělení, kde nejprve jejich hodnota narůstá a poté začne klesat. Ostatní parametry jsou na všech testovaných intervalech téměř neměnné. Parametr statistic se v tomto případě chová také pro všechny veličiny velice podobně.

Parametr statistic má tudíž zajímavé chování viz Obrázek 15. Zachycuje hodnotu testové statistiky a je vidět, že při spojitém testování se její hodnota mění různě, zatímco při náhodném je to přibližně stejné. Tím je pravděpodobně dokázáno, že náhodný výběr má pro takovéto testování větší smysl než spojitý, protože se průběh na různých časových intervalech chová mnohem stabilněji.

5 Závěr

Naprogramoval jsem aplikaci, která je schopna automatické analýzy velkých objemů dat s efektivním využitím dostupných možností procesoru počítače díky rozdělení jednotlivých procesů do synchronizovaných vláken s minimálním nárokem na RAM paměť za pomoci průměrování výsledků jednotlivých výpočtů podprocesů. Byl kladen důraz na univerzálnost aplikace (kapitola 3.3.3) pro různé soubory, jejich obsažené veličiny, množství záznamů a množství vstupních souborů. Zároveň ale uživateli umožňuje vlastnoruční výběr a nastavení, jaké veličiny chce testovat a na jakých intervalech.

Výsledky testů každého souboru jsou aplikací přeneseny do prostředí MATLAB, kde skripty zajistí vykreslení do příslušných grafů a jejich shrnutí, které je po dokončení všech testů vypsáno jako přehled výsledků do PDF souboru.

Experimentálně jsem zjistil, že obrovský vliv na výsledky testů má, pokud jsou testované vzorky, při dodržení stejného testovaného množství, načítány a posílány k testu jako v čase po sobě jdoucí hodnoty, nebo zda jsou tyto hodnoty extrahovány jako náhodný výběr z určitého intervalu. Tento fakt se odráží v naprosté změně výsledné odpovědnosti danému rozdělení a není výjimkou, že poté odpovídá zcela jinému. Viz kapitola 4.6. Na základě chování testového parametru statistic lze říci, že při tomto testování je vhodnější testovat náhodně načítané vzorky.

Rozhodnutí o průměrování, či kratším snímacím intervalu při zapisování hodnot veličin do souborů, nelze s jistotou definovat. Tento test závisel na intervalu zapsaných dat v souboru. Je pochopitelně možné průměrovat hodnoty stále dokola, respektive zvětšovat interval, ale tím se pouze vyhlazuje časová křivka a zanikají kritické momenty v čase, na kterých záleží a mohou mít neblahé dopady na zařízení. Tvrzení, že je možné zkrátit, případně zvýšit, interval zapisování na daném místě, proto záleží především na tom, jaká kvalita elektrické energie je zde vyžadována a jak moc je na ni náchylné prostředí. Normy [5] stanovují průměrování na 10 minut, výsledky mých testů se pohybovaly od 10 s do 20 s. Tyto testy vycházely z udržení normálního (Gaussova) rozdělení pravděpodobnosti, ne rozdělení odpovídajícího dané veličině.

Výsledky tohoto testu na průměrování se liší veličinu od veličiny, a proto myslím, že pokud by se redukoval interval zapisování, měl by být podle veličiny, kterou to ovlivňuje nejvíce, jinak řečeno má nejnižší výsledek průměrování. Ovšem také musí být brán zřetel na to, jak moc tato daná veličina ovlivňuje kvalitu elektrické energie.

Všechny testované veličiny se shodují v tom, že se zvyšujícím se intervalem (množství) testovaných hodnot klesá jejich odpovědnost pro všechna rozdělení, pro která byla testována, což platí i pro náhodný výběr, ne pouze po sobě jdoucí vzorky.

Samotné výsledné rozdělení pravděpodobnosti, kterým dané veličiny odpovídají, se liší. Sudé harmonické Cauchyho a log-normálnímu zatímco liché harmonické log-normálnímu. Všechny tři fáze napětí spolu se sdruženými napětími a THDU odpovídají také nejvíce log-normálnímu. Frekvence také odpovídá nejvíce log-normálnímu.

Použitá literatura

[1] A computational statistics class – CodeProject. *CodeProject – For those who code* [online]. Copyright © [cit. 10.05.2017]. Dostupné z: https://www.codeproject.com/KB/cs/csstatistics.aspx

[2] Accord.NET Machine Learning Framework. *Accord.NET Machine Learning Framework* [online]. Copyright © 2008 [cit. 06.05.2017]. Dostupné z: http://accord-framework.net/

[3] AForge.NET :: Framework. *AForge.NET :: Computer Vision, Artificial Intelligence, Robotics* [online]. Copyright © 2008 [cit. 10.05.2017]. Dostupné z: http://www.aforgenet.com/framework/

[4] BALATKA, Sláva a Olga KUTNOHORSKÁ. *Inženýrská statistika pro ekonomy*. 2014. Praha. ISBN 978-80-7080-894-8.

[5] *ČSN EN 50160 ed. 3: Charakteristiky napětí elektrické energie dodávané z veřejných distribučních sítí*. Praha: Český normalizační institut, 2011.

[6] Framework modules. *Accord.NET Machine Learning Framework* [online]. Copyright © 2009 [cit. 06.05.2017]. Dostupné z: http://accord-framework.net/docs/html/R_Project_Accord_NET.htm

[7] *ING. NOVOTNÝ, Radovan. Weibullovo rozdělení při analýzách bezporuchovosti [online]. Ústav mikroelektroniky FEKT VUT Brno [cit. 2017-05-10]. Dostupné z: http://www.elektrorevue.cz/clanky/02017/index.html*

[8] Kvalita elektrické energie. *OEnergetice.cz* [online]. Copyright © 2017 [cit. 06.05.2017]. Dostupné z: http://oenergetice.cz/technologie/elektroenergetika/kvalita-elektricke-energie/

[9] MARKIEWICZ, Henryk a Antoni KLAJN. *Voltage Disturbances: Standard EN 50160 – Voltage Characteristics in Public Distribution Systems* [online]. [cit. 2017-05-10]. Dostupné z: http://www.cdtechnics.be/542-standard-en-50160-voltage-characteristics-in.pdf

[10] MATLAB Dokumentation. *MathWorks – Makers of MATLAB and Simulink* [online]. Dostupné z: https://www.mathworks.com/help/matlab/index.html

[11] NuGet Gallery | iTextSharp 5.5.11. NuGet Gallery | Home [online]. Copyright © 2017 .NET Foundation [cit. 06.05.2017]. Dostupné z: https://www.nuget.org/packages/iTextSharp/

[12] *Statistické metody a demografie*. Praha: Vysoká škola ekonomie a managementu, 2009. ISBN 978-80-86730-43-1.

[13] Statistika a pravděpodobnost | Přírodovědecká fakulta Masarykovy univerzity. *Veřejné služby Informačního systému* [online]. Copyright © 2012 [cit. 06.05.2017]. Dostupné z: https://is.muni.cz/do/rect/el/estud/prif/ps15/statistika/web/index2.html

[14] SuanShu | Numerical Method Inc.. *Math, Quantitative Investment, Algorithmic, Trading* [online]. Dostupné z: http://numericalmethod.com/suanshu/

[15] Výpočet nejistot měření [online]. Copyright © [cit. 06.05.2017]. Dostupné z: https://vscht.cz/ufmt/cs/pomucky/uhrovah/docs/kapitola4.pdf

[16] Vyšší harmonické a jejich působení na síť – Časopis Elektro – Odborné časopisy. *Odborné časopisy* [online]. Copyright © 2014 [cit. 06.05.2017]. Dostupné z: http://www.odbornecasopisy.cz/elektro/casopis/tema/vyssi-harmonicke-a-jejich-pusobeni-na-sit--11949

Příloha

A Dodatečné obrázky k textu

Příklad legendy

Obrázek 12: Příklad legendy

```
double A = 0;
double B = 0;
double lastY = 100;
double lastX = X[0];

for (int i = 0; i < Y.Count(); i++)
{
	if (!double.IsNaN(Y[i]) && !double.IsNaN(X[i]))
	{
		double d = Math.Log10(X[i] / lastX);
		A += Integrate(lastY, Y[i], (d));
		B += Integrate(100, 100, (d));
		lastY = Y[i];
		lastX = X[i];
	}
}

double r = (A / B) * 100;
```

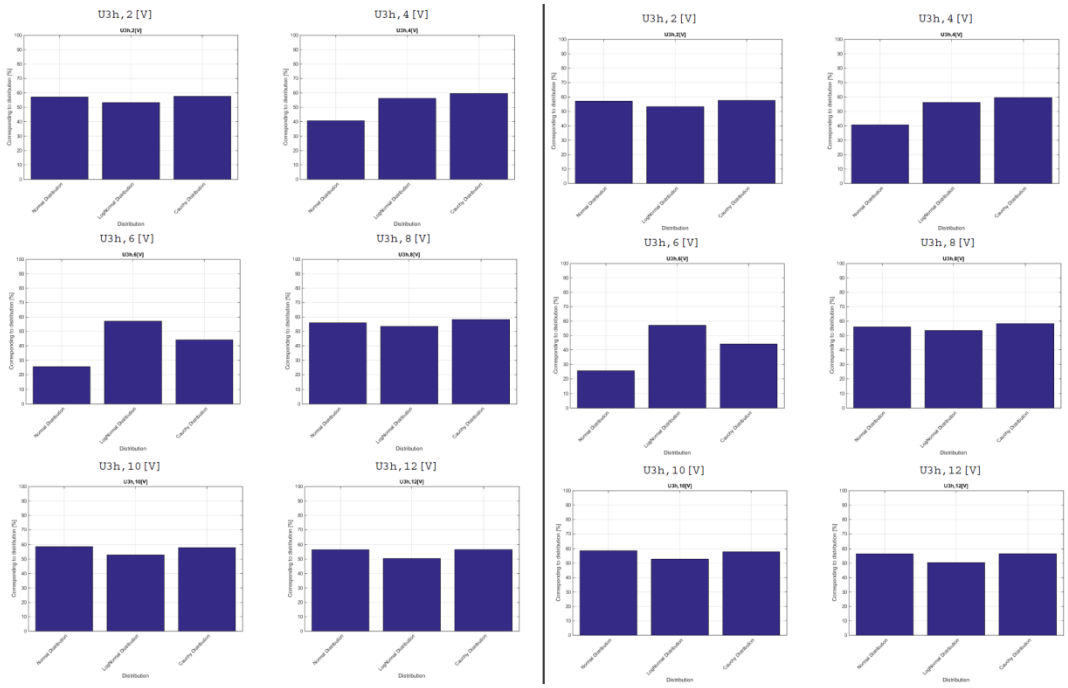
```
private double Integrate(double startY, double endY, double sizeX)
{
	double lowEnd = (startY >= endY) ? endY : startY;
	double highstart = (startY >= endY) ? startY : endY;

	double triangle = Math.Abs(highstart - lowEnd);
	triangle = (triangle * sizeX) / 2;
	double square = lowEnd * sizeX;
      return triangle + square;
}
```

Zdrojový kód 5: Způsob Integrace

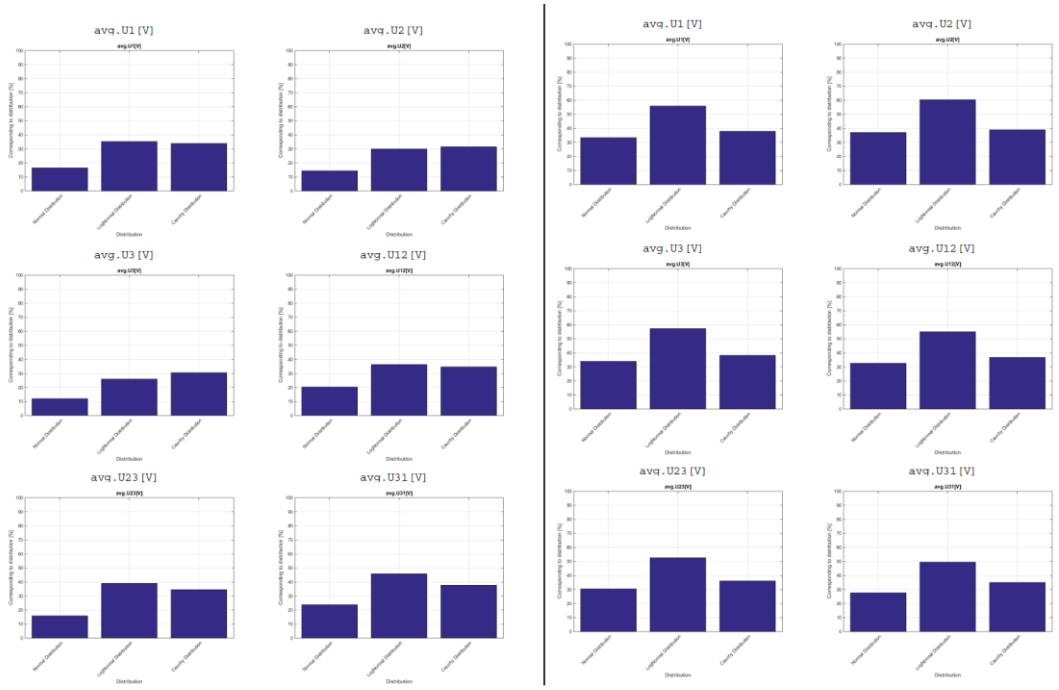
```
private double ComputateError(List<FreqResult> res,double avg)
{
	foreach(FreqResult fr in res)
	{
		fr.GetPower(avg, 2);
	}

	double ErrorA = Math.Sqrt(((double)1/(res.Count() * (res.Count() - 1))) *
res.Sum(x => x.Power));
	double Difference = res.Max(x=> x.Interval)- res.Min(x => x.Interval);
	double ErrorB = Difference/Math.Sqrt(6);
	return Math.Sqrt(Math.Pow(ErrorA,2) + Math.Pow(ErrorB, 2));
}


public double GetPower(double average, int power)
{
	this.Power = Math.Pow(this.Interval - average, power);
	return Power;
}
```

Zdrojový kód 6: Výpočet chyby

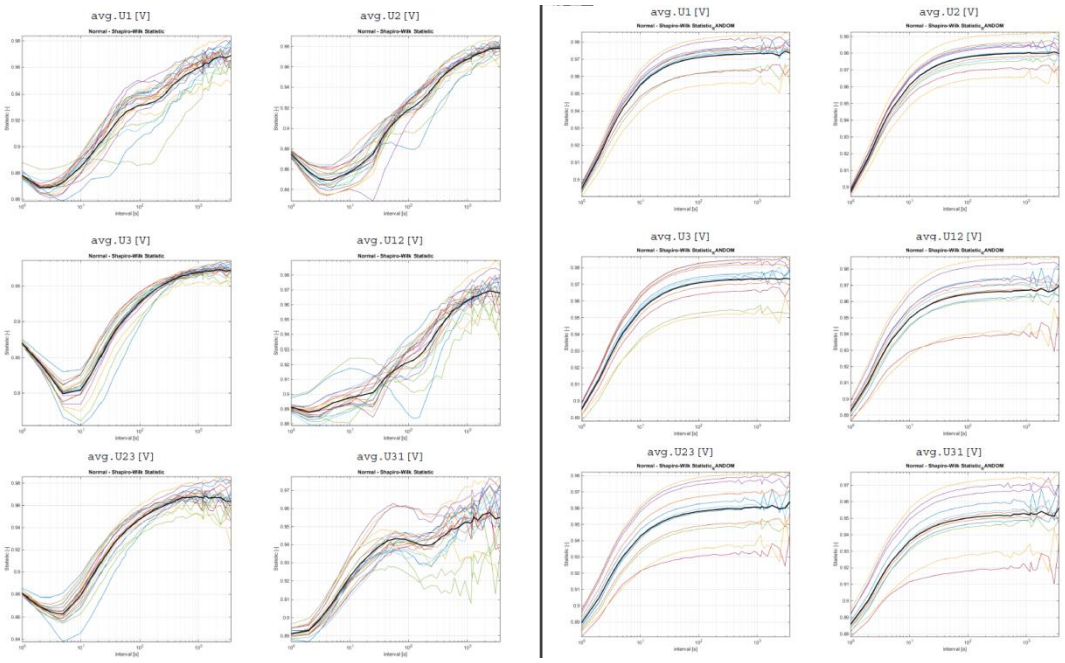
Porovnání výsledků náhodného testování



Obrázek 13: Spojité (L) vs Náhodné (P) testování sudých harmonických

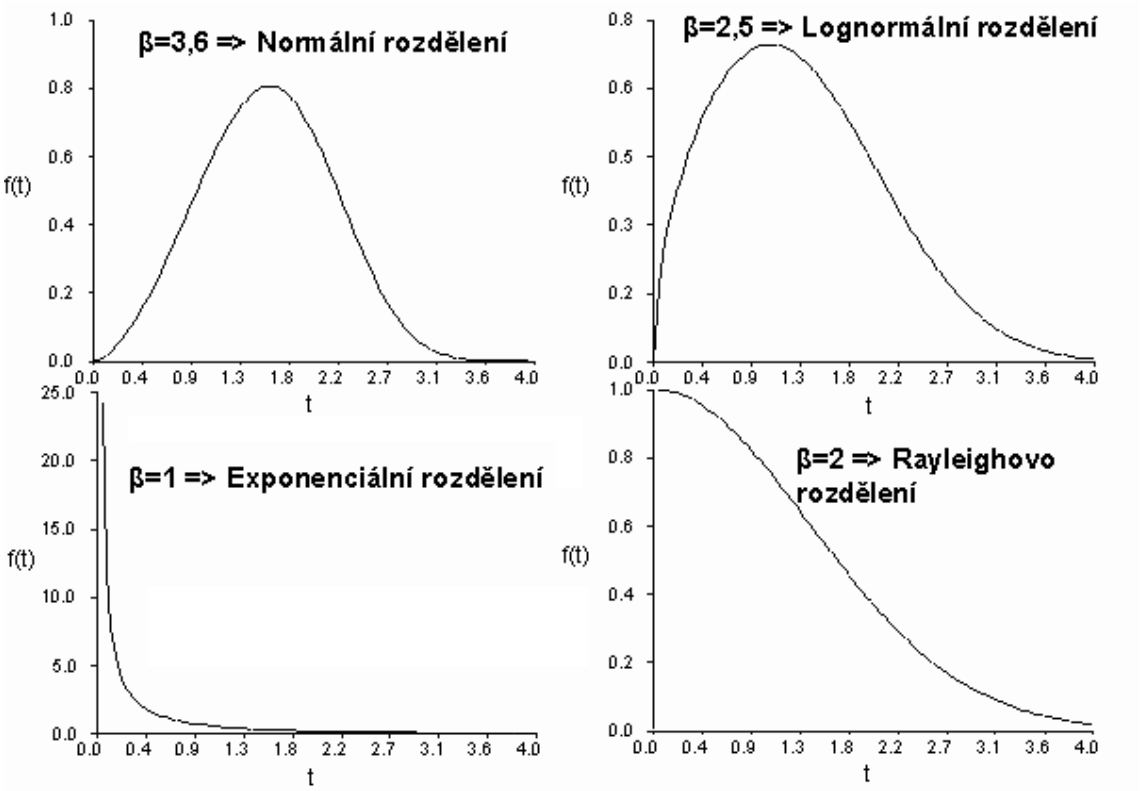


Obrázek 14: Spojité (L) vs Náhodné (P) testování fází napětí



Obrázek 15: Spojité (L) vs Náhodné (P) testování fází, parametr Statistic

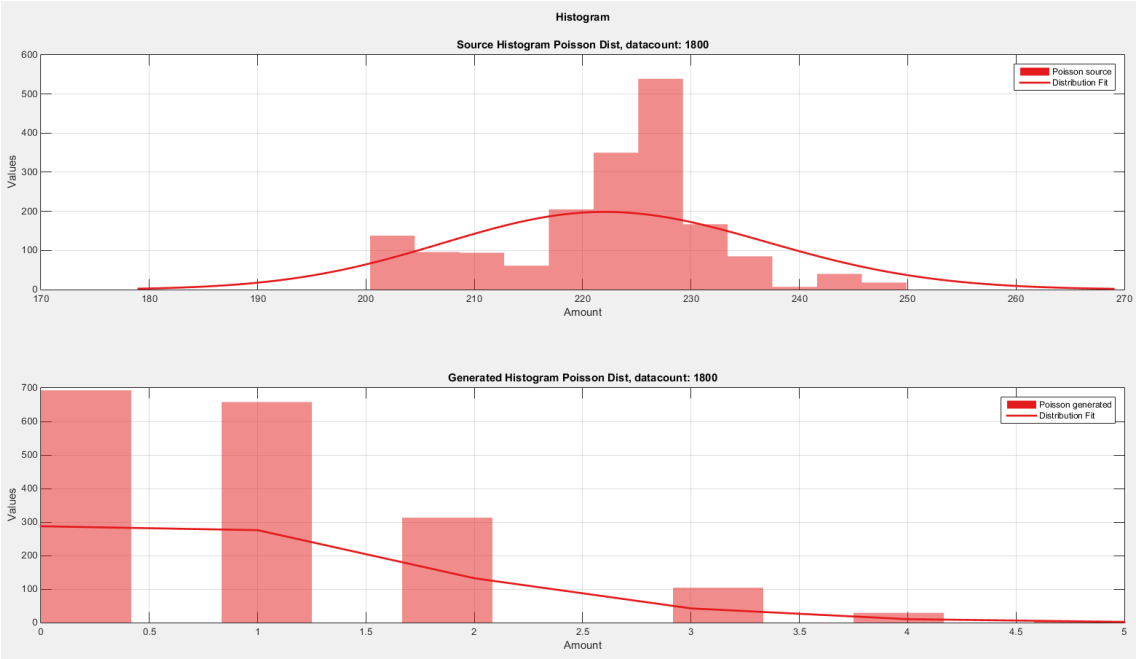
Další obrázky



Obrázek 16: Typy rozdělení pravděpodobnosti. Zdroj viz [7]

Odd harmonics				Even harmonics	
Not multiples of 3		Multiples of 3			
Order *h*	Relative voltage (%)	Order *h*	Relative voltage (%)	Order *h*	Relative voltage (%)
5	6	3	5	2	2
7	5	9	1.5	4	1
11	3.5	15	0.5	6 24	0.5
13	3	21	0.5		
17	2				
19	1.5				
23	1.5				
25	1.5				

Obrázek 17: Hodnoty jednotlivych harmonických napětí. Zdroj viz [9]



Obrázek 18: Test Poissonova rozdělení

B Návod k instalaci a spuštění aplikace

Spustitelné soubory aplikace se nachází na CD ve složce Aplikace – Spustitelné. Soubor jménem Bakalarka.exe. Dále je zde i VS projekt ve složce Aplikace - Visual_Studio_Project. M-fily jsou ve složce Aplikace – MATLAB skripty.

Po spuštění aplikace je uživatel vyzván k vyhledání složky s m-fily, jedná se o podstatnou součást programu, bez kterého není možné vykreslit grafy a aplikace nebude správně fungovat. Uživateli je doporučeno před startem analýzy otevřít „Options“ a zvolit si vyhovující nastavení i přes již přednastavené výchozí nastavení.

K výběru souborů lze pokračovat stisknutím tlačítka „File“ v levém horním rohu okna a stisknutím „Add file“ a vyberejeden nebo označí více souborů. Pokud chce uživatel obnovit ztracený proces, pak místo „Add file“ stiskne „Load from backup“.

Po vybrání souborů pro analýzu je třeba vybrat testované veličiny, to je možno buď u každého souboru zvlášť po rozkliknutí tlačítka „Properties“ a zaškrtnutím jednotlivých veličin, nebo stisknutím tlačítka „Presets“ v horní liště a poté tlačítko „Values“. Výběr testovaných intervalů je taktéž možný v „Properties“ každého souboru, a zároveň v „Presets“ -> „TimeSteps“.

Poté už stačí stisknout „Start All“ a čekat na dokončení všech procesů, které jsou monitorovány progressbary pro každý soubor, nebo zaškrtnout checkbox každého souborů zvlášť. Po dokončení bude výsledný PDF soubor automaticky otevřen.