Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra kybernetiky

Diplomová práce

Vývoj vizualizačního prostředí pro úlohu vibrační diagnostiky lopatek

Plzeň 2020 Ota Hofmann

Místo této strany bude zadání práce.

Prohlášení

Prohlašuji, že jsem diplomovou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 29. června 2020

Ota Hofmann

Abstract

The diploma thesis deals with the monitoring of vibrations of steam turbine blades. Based on the knowledge associated with this issue, software was created for offline analysis of these vibrations. The resulting application was developed with the support of Logic Elements s.r.o. and it is planned to continue with support in a future. The application was created in Visual Studio in C#. In the end the individual calculations were optimized for minimum time and memory requirements.

Key words: vibration, steam turbine, tip-timing, Visual Studio, C#, signal processing

Abstrakt

Diplomová práce se zabývá monitorováním vibrací lopatek parních turbín. Na základě poznatků s touto problematikou spojených byl vytvořen software pro offline analýzu těchto vibrací. Výsledná aplikace byla vyvíjena za podpory firmy Logic Elements s.r.o. a bude i dále podporována. Aplikace byla vytvořena v prostředí Visual Studio v jazyce C#. Jednotlivé výpočty byly v konečné fázi optimalizovány pro minimální časovou a paměťovou náročnost.

Klíčová slova: vibrace, parní turbína, tip-timing, Visual Studio, C#, zpracování signálu

Poděkování

Velice rád bych poděkoval panu Ing. Jindřichu Liškovi, Ph.D za výborné vedení, ochotu a podnětné připomínky při plnění diplomové práce.

Dále bych rád poděkoval panu Ing. Janu Jaklovi, Ph.D za poskytnutí jeho znalostí v oblasti zpracování signálu detekce vibrací lopatek parních turbín.

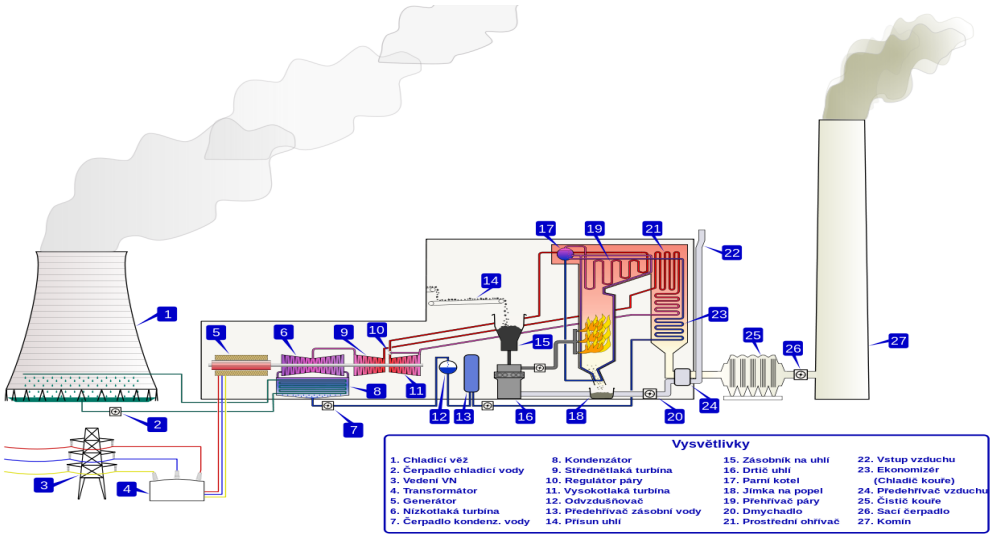
Výše uvedeným bych dále poděkoval za přátelský přístup a zpětnou odezvu při testování výsledné aplikace.

V poslední řadě bych rád poděkoval mé rodině za jejich podporu v době mého studia a firmě Logic Elements s.r.o. za podporu a poskytnutí potřebného softwaru.

Obsah

Úvod

Moderní rotační stroje vybavené lopatkami mají široké využití v oblasti leteckého, automobilového či energetického průmyslu. Havárie těchto strojů v případě poruchy mohou vést k nemalým ekonomickým ztrátám nebo v horších případech ztrátám na životech. Z tohoto důvodu jsou tyto systémy v pravidelných intervalech odstavovány, aby bylo možné odhalit jejich případné nastávající problémy, které by mohly vést k neplánovanému omezení jejich budoucího provozu.



Obrázek 1: Stavba tepelné elektrárny

V případě parních turbín je nutné odstavit celý systém generování páry (kotel, reaktor) spolu s turbínou. Tato odstávka systému stojí vlastníka elektrárny nemalé finanční prostředky, které se uplatní jako nezbytné pouze v případě nalezení konkrétní závady, která by vedla k následným škodám. Z důvodu finanční náročnosti těchto pravidelných odstávek je aktuálně velkým trendem stále častější využití moderních nástrojů technické diagnostiky a metod zpracování signálu. Na jejichž základu se osazují turbíny sofistikovanými řešeními, která umožní včasné odhalení závady již v raném stádiu i za běžného provozu turbíny.

Včasná detekce stavu, který by mohl vést k havárii systému, umožňuje naplánovat výměny konkrétních dílů, u nichž bylo odhaleno nastávající poškození a případně predikovat životnost těchto turbinových součástí s ohle-

dem na plánovaný budoucí provoz. Jednou z nejkritičtějších součástí turbín je oběžná lopatka, která přenáší tlak proudícího média (páry) na rotor a svojí vhodnou konstrukcí způsobuje rotaci rotoru, která je následně v generátoru přeměněna na elektrický výkon.

Lopatka a její velikost, tvar, použitý materiál a celá řada dalších hledisek je tak klíčovou komponentou z pohledu funkčnosti, účinnosti a spolehlivosti turbíny. Není divu, že na vývoj nových, či vylepšení stávajících typů lopatek věnují přední světoví výrobci turbín obrovské prostředky. Tato skutečnost je také způsobena tím, že výrobci turbín jsou provozovateli nuceni k tomu, aby zvyšovali účinnost a snižovali pořizovací náklady dodávek a provozu turbín. To má za následek tendenci v prodlužování a zeštíhlování lopatek, s čímž souvisí vyšší náchylnost ke kmitání. Vyšší výkon dodávaných turbín je dosahován vyšším zatěžováním jednotlivých turbínových stupňů, důsledkem čehož je vznikající vyšší budící síla od průtočného množství páry působící na jednotlivé lopatky. Tyto a další parametry (např. zvyšování vstupní teploty média) mají za následek četnější pozorování jevů, které se v minulosti objevovaly jen zřídka. K posouzení toho, zda vyvinutý typ lopatky v provozu turbíny uspěl nebo vykazuje odchylky od navrhovaných vlastností, je potřeba tyto vlastnosti v různých provozních stavech lopatek měřit a porovnávat je s navrhovanými hodnotami.

V posledních letech je pro monitorování lopatek čím dál více využívána vibro-diagnostika. Ať se už jedná o výrobce či provozovatele turbíny, jejich požadavky na chování lopatek jsou totožné. Vibrace zmíněných lopatek nesmějí překračovat limitní úrovně, které jsou udávány výrobcem. Překročením této hranice dochází ke snižování životnosti a hrozil by vznik trhlin a únavových lomů v materiálu lopatek. Lopatky jsou měřeny kontaktním nebo bezkontaktním způsobem a použitému způsobu odpovídají i metody, kterými jsou naměřené signály zpracovány.

V poslední době je nejrozšířenější metodou pro dlouhodobé monitorování a diagnostiku vibrací oběžných lopatek metoda BTT (Blade Tip Timing), jejímuž popisu je věnována i část této práce. Signály měřené touto metodou jsou specifickým způsobem zpracovávány a výsledky zpracování je možné vizualizovat několika způsoby. Ty mají své využití pro konkrétní analýzu určitých vlastností kmitajících lopatek.

Cíle práce

Tato diplomová práce se zaměřuje na zpracování signálů ze systémů diagnostiky vibrací oběžných lopatek založených na principu metody BTT - především na implementaci vhodných softwarových řešení v oblasti offline zpracování naměřených dat a jejich následnou vizualizaci. Cílem práce je vyvinout softwarovou aplikaci s uživatelsky přívětivým prostředím pro offline analýzu dat. Důraz je kladen na rychlost výpočtů a minimální paměťové nároky v prováděném zpracování signálu i při jejich následné vizualizaci.

Seznam použitých pojmů

1. Kompilovaný kód - Kód, který musí být před jeho spuštěním kompletně přeložen kompilátorem
2. Parsování - Rozdělení celku na jednotlivé části
3. Release - Vydání softwarového řešení
4. Refresh - Aktualizace (obvykle grafů)
5. NT - Nízkotlaké
6. ST - Středotlaké
7. VT - Vysokotlaké
8. TOA - Time of arrival - čas příchodu
9. RPM - Revolutions per minute - otáčky za minutu
10. t_i - Časový okamžik i
11. $\tilde{t_i}$ - Očekávaný časový okamžik i

Seznam použité řecké abecedy

1. ω [Hz] Otáčková frekvence
2. ω_{vl} [Hz] Vlastní frekvence disku
3. Ω_{krit} [Hz] Kritická frekvence otáčení disku
4. Ω_{P_+} [Hz] Úhlová frekvence dopředné vlny
5. Ω_{P_-} [Hz] Úhlová frekvence zpětné vlny

1 Úvod do problematiky

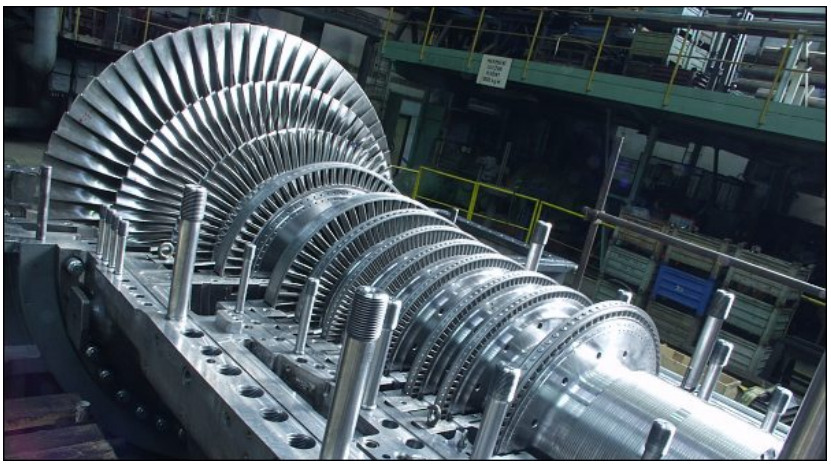
V této kapitole jsou popsány způsoby měření dynamických vlastností lopatkových disků. Diplomová práce se zaměřuje na způsoby měření vibrací rotačních turbín a prostředky v této oblasti používané. Další části se věnují zpracování získaného signálu.

1.1 Rotační turbína

Jedná se o mechanický rotační stroj, který se skládá z jednoho nebo více pohyblivých lopatkových kol umístěných na společné hřídeli. Dle řešené úlohy prochází mezi těmito koly kapalina nebo plyn, kde je následně jejich kinetická, tepelná a tlaková energie přeměňována na rotační pohyb hřídele stroje. Využívají se v oblasti letectví (pohonné jednotky), ale jedno z nejvýznamnějších využití je v oblasti energetiky, kde se využívají především jako primární poháněcí stroje pro výrobu elektrické energie. [2]

1.2 Parní turbína

Generování elektrické energie vzniká pomocí roztočené turbíny, na kterou je pod tlakem přiváděna pára, která tuto turbínu roztáčí. Stroj tedy získává mechanickou energii pomocí expanze proudící vodní páry v jednotlivých stupních turbíny, které se rozdělují na NT, ST a VT díly. [10]



Obrázek 1.1: Parní turbína

1.2.1 Lopatky parní turbíny

Jedním z klíčových prvků parní turbíny jsou její lopatky. Vzniklá pára, produkovaná spalovacím systémem elektrárny, o vysokém tlaku je vedena parním potrubím do turbíny, kde následně prochází skrze lopatky. Tímto způsobem je transformovaná tepelná energie na mechanickou a následně v generátoru na elektrickou energii. Lopatky jsou voleny na základě umístění v systému parní turbíny. Jedná se především o jejich délku. Ve VT dílu jsou lopatky krátké z důvodu působení páry o vysokém tlaku a nejsou tedy příliš namáhány. Oproti tomu lopatky v NT dílu jsou kvůli působení nižšího tlaku, a tedy působení většího objemu páry, delší. Z důvodu působení velkých odstředivých sil je na tyto lopatky kladen vyšší požadavek na jejich pevnost.

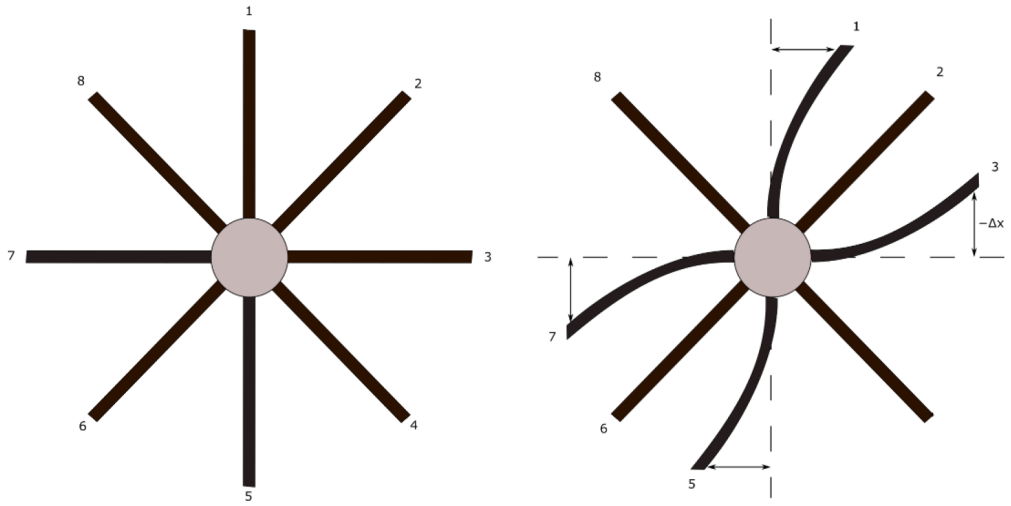
Obrázek 1.2: Lopatky parní turbíny

1.3 Vlastní frekvence a rezonance

Pojem rezonance značí stav systému, který má snahu kmitat na větší amplitudě při určité frekvenci v porovnání s jinými. Tyto frekvence nazýváme pojmem vlastní frekvence a nastávají v případě, kdy je frekvence budící síly (v případě parní turbíny se jedná o budící sílu rotoru) blízká těmto vlastním frekvencím. V případě změny otáček stroje (např. na své provozní otáčky) dochází ke změně budící frekvence a může nastat případ, že budící frekvence přechází přes zmíněné pásmo vlastních frekvencí. Z tohoto důvodu může dojít k výraznému nárůstu vibrací a je nutné tyto frekvence rychle překonat. [11]

1.4 Vibrace lopatek

Chvění je kmitavý pohyb kolem rovnovážné polohy. Počet cyklů kmitavého pohybu za jednotku času (obvykle vteřina) nazýváme pojmem frekvence. Tuto fyzikální veličinu uvádíme v jednotkách Hz [hertz]. Pokud je lopatka vystavována vibracím, může dojít k namáhání materiálu lopatky, a tedy čerpání její zbytkové životnosti.



Obrázek 1.3: Vibrace lopatek

Pro demonstraci této problematiky může sloužit systém 2. řádu, který je popsán skrze přenosovou funkci $F(p)$.

$$F(p) = \frac{Y(p)}{U(p)} = \frac{{\omega_n}^2}{p^2 + 2\xi\omega_n p + {\omega_n}^2}, \xi \in [0, 1) \tag{1.1}$$

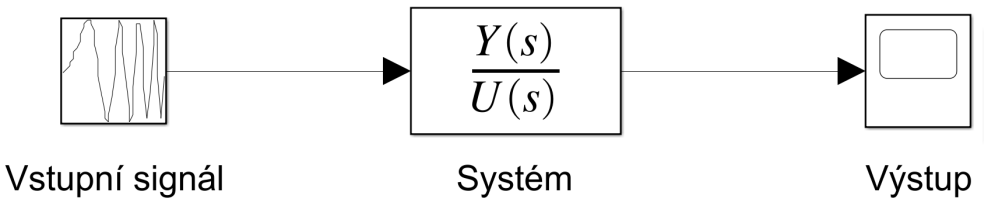
Kde ξ označuje relativní činitel tlumení a ω_n netlumenou frekvenci systému.
Pro zvolený systém byly zvoleny parametry:

$$\omega_n = 2\pi \cdot 30 \approx 188,49$$
$$\xi = 0,05$$

Výsledný systém je roven:

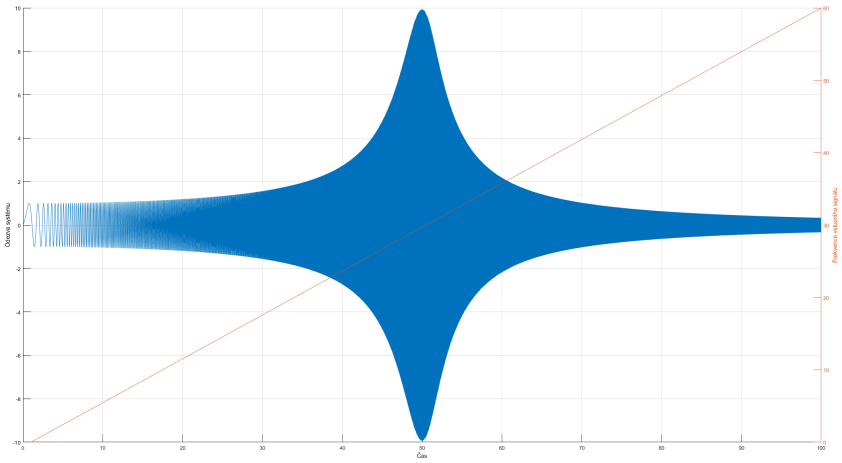
$$F(p) = \frac{Y(p)}{U(p)} = \frac{{\omega_n}^2}{p^2 + 2\xi\omega_n p + {\omega_n}^2} = \frac{188,49^2}{p^2 + 18,84 + 188,49^2} \tag{1.2}$$

Na takto navržený systém $F(p)$ byl přiveden vstupní signál $u(t)$ o zvyšující se frekvenci 0.1 - 60 Hz v časovém intervalu 100 vteřin. Systém byl simulován v prostředí Simulink [1.4].



Obrázek 1.4: Simulink schéma

Výstup tohoto systému je zobrazen na následujícím grafu [1.5].

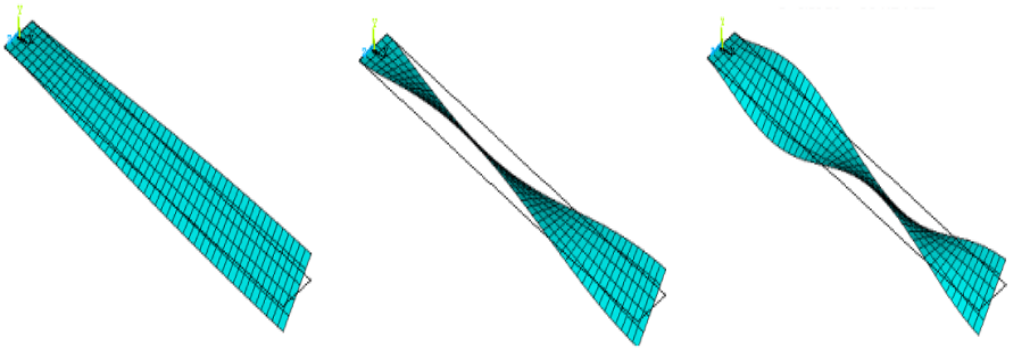


Obrázek 1.5: Odezva systému $F(p)$

Z grafu můžeme vypozorovat odezvu systému (modrý signál) a frekvenci vstupního signálu (oranžový signál) v průběhu 100 vteřin. Můžeme si všimnout, že v oblasti 50 vteřin je systém vybuzen větší amplitudou. Je tomu tak z důvodu působení vstupního signálu na frekvenci 30 Hz, které odpovídají vlastní frekvenci systému.

1.4.1 Vlastní tvar kmitu lopatek

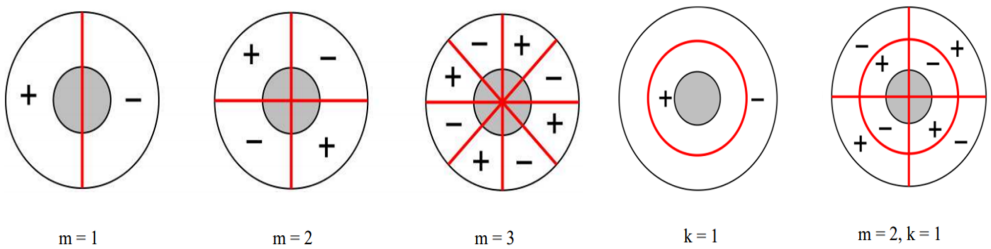
Jedná se o tvar deformace lopatek, který je tvořen na příslušné vlastní frekvenci. Každá lopatka je schopna kmitat tvary závislými na své struktuře. Tyto tvary kmitu dělíme na ohybové, torzní, ohybotorzní a vyšší tvary, které dle [11] již nelze jednoduše slovně popsat.



Obrázek 1.6: Vlastní tvary kmitu lopatek

1.4.2 Dynamické chování lopatkových disků

Lopatkový disk jakožto celek má nekonečný počet vlastních frekvencí a tvarů kmitání. U lopatkového disku (např. turbína) s konečným počtem lopatek je počet vlastních tvarů disku též konečný. Základní tvary kmitání jsou charakterizovány skrze uzlové průměry či uzlové kružnice. Jedná se o oblasti rotujícího disku, které jsou trvale v klidu, tedy místa s nulovou výchylkou [12]. Tvary kmitání disku jsou vyobrazeny na následujícím obrázku [1.7], kde je demonstrován celkový počet uzlových průměrů m a celkový počet uzlových kružnic k.



Obrázek 1.7: Tvary kmitání disku

Při rotaci se po disku šíří dvě vlny, jejichž směr je opačný (běží proti sobě). Dle jejich směru jsou označovány jako vlna dopředná a vlna zpětná. Jednotlivé úhlové rychlosti jsou definovány vztahy [1.3] a [1.4].

$$\Omega_{P+} = \Omega_{Krit} + m \cdot \omega \tag{1.3}$$

$$\Omega_{P-} = \Omega_{Krit} - m \cdot \omega \tag{1.4}$$

V případě vlny s nulovou rychlostí dochází ke vzniku stojatého vlnění, a tedy míst, která jsou trvale v klidu. Tato místa jsou označována jako uzlové či kružnicové průměry. Naopak otáčky, při kterých dochází k výraznému vybuzení, se nazývají kritické otáčky disků ω_{vl} a jsou definovány vztahem [1.5]. Tato kapitola čerpá z [11].

$$\omega_{vl} = \frac{\Omega_{krit}}{m} \tag{1.5}$$

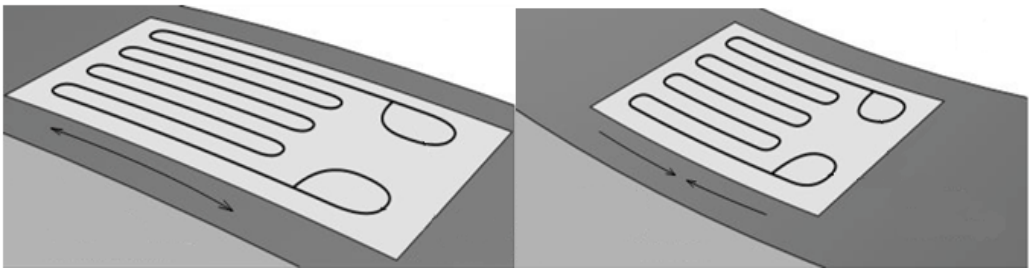
Je vhodné také uvést, že při provozu turbíny lopatkové kolo obecně kmitá všemi uzlovými průměry, avšak při jejich diagnostice mohou být díky své malé amplitudě kmitání zanedbány.

1.5 Měření vibrací lopatek

Vibrace lopatek jsou velice nežádoucí a mohou vést k mechanickému namáhání lopatek rotační turbíny. Z tohoto důvodu dochází ke ztrátě jejich předpokládané životnosti a je cílem tyto oblasti vibrací eliminovat. Monitorováním stavu lopatek lze docílit predikce jejich životnosti a na základě těchto poznatků optimalizovat provozní parametry turbín, které budou vést k maximalizaci životnosti lopatek. Tyto monitorovací systémy využívají snímače, které jsou při provozu turbíny vystaveny extrémním podmínkám (v případě VT dílů). K detekci kmitání lopatek se používají následující principy.

1.5.1 Kontaktní způsob měření vibrací lopatek

Při tomto způsobu měření vibrací se nejčastěji využívá polovodičových či kovových tenzometrů. Jedná se o pasivní elektrotechnický prvek určený jako senzor k nepřímému měření mechanického napětí na povrchu materiálu, který je deformován vnější silou. Z principu věci vyplývá, že pevně spojený tenzometr s namáhaným objektem je deformován stejnou silou, jako objekt samotný. Na základě tohoto poznatku se umístění tenzometru volí v oblasti největšího ohybu / deformace.



Obrázek 1.8: Tenzometr

Z takto naměřených dat lze snadno rozlišit časové okamžiky, kdy došlo v oblasti umístění tenzometru ke změnám v namáhání materiálu.

Kovové tenzometry

Kovové tenzometrické senzory jsou vyráběny z kovových materiálů a jejich závislost relativní změny vůči deformaci je téměř lineární. K nelineárnímu chování dochází v případě značně velké deformace senzoru.

Polovodičové tenzometry

V případě polovodičových tenzometrů je tomu oproti předchozímu typu naopak. Tento druh senzorů se vyznačuje svojí nelineární závislostí relativních změn odporu na měřené deformaci.

I přes veškeré výhody se tomuto druhu senzorů v oblasti měření otáček parních turbín nedostává velkého uplatnění. Z důvodu nežádoucích podmínek, které jsou v okolí lopatek, není možné, aby měření deformace pomocí tenzometrů mělo dlouhou životnost. Podílí se na tom velké odstředivé síly, vysoké teploty a vlhkost, která vzniká působením páry. Tato podkapitola čerpá z [5].

1.5.2 Bezkontaktní způsob měření vibrací lopatek

Měření bezkontaktní metodou patří mezi nejčastější způsoby monitorování lopatek. Umístění senzorů je ve statoru po obvodu lopatkového kola a snímá se vzdálenost lopatky v době jejího průletu v okolí senzoru. Z takto naměřených dat lze extrahovat časy průletů (TOA) a z nich pak konkrétní frekvence a amplitudy kmitání lopatek. Pokud lopatka kmitá, pak se průchod lopatky vzhledem k fázové značce v průběhu měření mění. Bezkontaktní měření využívá metoda Tip-timingu, která je popsána v následující kapitole.

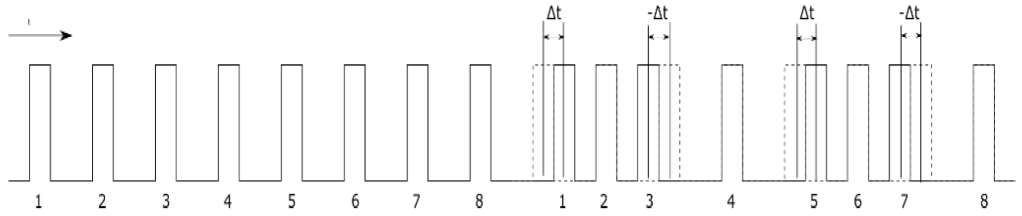
1.5.3 Blade Tip-Timing (BTT)

Blade Tip-Timing (dále jen BTT) neboli metoda bezkontaktního měření vibrací lopatek má řadu výhod oproti tradičnímu kontaktnímu měření vibrací lopatek. Metoda začala být používána v 70. letech 20. století jako náhrada kontaktního měření a je ve velké míře využívána v dnešních měřeních.

Metoda je založena na měření časů průletů hran jednotlivých lopatek a jejich následném zpracování. Využívá detekované časy průletů (TOA) a na základě rozdílu očekávaného a skutečného času průletu lopatky $\Delta t_{i,n}$ spolu se znalostí otáčkové frekvence rotoru f_{ROT} a jeho průměru r vyhodnocuje aktuální vibrace lopatky Δx_i. Pro vyhodnocení vibrací n-té otáčky lopatky i využíváme následujícího vztahu:

$$\Delta x_{i,n} = 2\pi \cdot f_{ROT} \cdot r \cdot \Delta t_{i,n}, \quad \Delta t_{i,n} = (t_{i,n} - \tilde{t}_{i,n}) \tag{1.6}$$

Pro tento výpočet je tedy nutné, aby byly známé časové hodnoty měřeného průchodu lopatky $t_{i,n}$ a jeho očekávané hodnoty $\tilde{t}_{i,n}$.

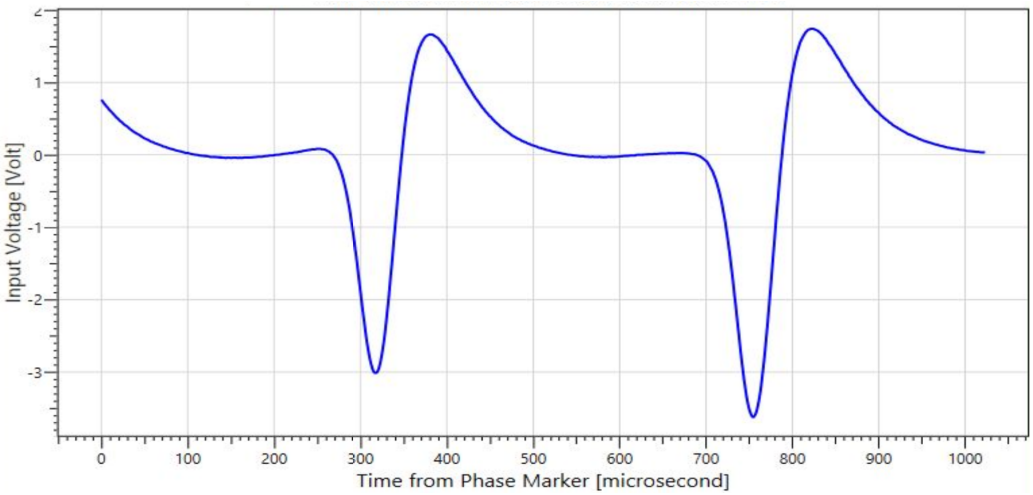


Obrázek 1.9: Očekávaná a skutečná detekce času průletu lopatky

Pro určení očekávaného časového okamžiku průletu lopatky by v ideálním případě postačovalo uvažovat ekvidistantní rozmístění lopatek po obvodu disku. Tento postup je však z důvodu konstrukčních či montážních odchylek nepoužitelný, jelikož by pro tento případ byla zanesena do signálu BTT nežádoucí chybovost.

V systému VMS (Vibration monitoring system), ze kterého byly poskytnuty data pro zpracování, je meření TOA vztaženo k otáčkám rotoru a s každou otáčkou se čítače TOA nulují. Je tomu tak z důvodu nekonstantní otáčkové frekvence rotoru, jelikož v praxi se tato frekvence mění neustále i v rámci jedné otáčky rotoru. Kromě vibrací jednotlivých lopatek jsou tak TOA ovlivněny také mezi-otáčkovými změnami otáčkové frekvence rotoru.

Samotná detekce časových událostí $t_{i,n}$ je získávána skrze triggerování signálu snímače. Níže uvedený graf [1.10] ukazuje výstupní napětí zmíněného senzoru po dobu trvání detekce dvou lopatek.



Obrázek 1.10: Výstupní signál senzoru

Z výše uvedeného grafu si můžeme všimnout dvou časových úseku v oblasti $(320-380)\mu s$ a $(760-830)\mu s$. V těchto úsecích signálu je detekována vzestupná hrana, při jejíž detekci je zaevidována časová hodnota $t_{i,n}$ detekce lopatky i při n-té otáčce.

Z důvodů uvedených v této kapitole je volena metoda zakládající se na normování signálu průletu lopatek $s(t)$ hodnotou jednotlivých délek otáček $l(t)$. Tímto způsobem získáme normované hodnoty s_{norm} v intervalu $(0, 1)$, které již nejsou závislé na otáčkách rotoru.

$$s(t)_{norm} = \frac{s(t)}{l(t)} \tag{1.7}$$

Pro získání očekávaných hodnot příchozích průletů lopatek je nutné, abychom spočetli střední hodnotu normované posloupnosti $\bar{s}(t)_{norm}$, která je rovna již zmíněné očekávané hodnotě průletu lopatky $\tilde{t}_i$. Tato hodnota je následně odečtena od normovaných hodnot, čímž je získána hodnota diference očekávané a skutečné hodnoty Δt.

$$\bar{s}(t)_{norm} = E[s(t)_{norm}] = \frac{\sum_{i=1}^{N} s(t_i)}{N} \tag{1.8}$$

$$s(t)_{norm} = s(t)_{norm} - \bar{s}(t)_{norm} \tag{1.9}$$

2 Metody a algoritmy zpracování dat z BTT

V současné době se v oblasti zpracování signálu využívají diagnostické systémy (např. VMS Analysis) umožňující výpočty základních charakteristik signálů vibrací, které mohou být užitečné při jejich zpracování. Na jejich základě může být zjištěno podrobné chování turbíny (např. skrze Campbellův diagram). Kapitola se zabývá výpočtem v časové a frekvenční oblasti spolu s výpočty modálních charakteristik a jejich významy.

2.1 Analýza v časové oblasti

2.1.1 Efektivní hodnoty (RMS)

Efektivní hodnota (RMS – Root Mean Square neboli druhá odmocnina ze střední hodnoty kvadrátu), také známá pod pojmem kvadratický průměr, je statistická hodnota měřící hodnotu měnící se veličiny.

$$x_{RMS} = \sqrt{\frac{1}{N}\sum_{i=1}^{N} x_i^2} \tag{2.1}$$

2.1.2 Výkmit (0-peak)

Výkmit (0-peak - Zero Peak neboli špičková hodnota) je maximální vzdálenost vrcholu vlny od referenčního signálu.

$$x_{0-peak} = max(x(t) - E[x(t)]) = max(x(t) - \frac{1}{n}\sum_{i=1}^{N} x(t_i)) \tag{2.2}$$

2.2 Analýza ve frekvenční oblasti

Samotný časový signál může být z důvodu rotorového chvění pod vlivem aditivního šumu, který omezuje analýzu naměřeného signálu v časové oblasti. Při správném použití frekvenční analýzy dokážeme tyto nechtěné vlivy potlačit nebo je přímo eliminovat.

Frekvenční analýza je velice schopným nástrojem pro identifikaci vznikajících poruch v jednotlivých částech sledovaného objektu (např. lopatka turbíny).

Základním kamenem tohoto typu analýzy je především diskrétní Fourierova transformace (DFT – Discrete Fourier Transform) a algoritmus rychlé Fourierovy transformace (FFT – Fast Fourier Transform). Jedná se o metodu pro popis signálu ve frekvenční oblasti, jejíž hlavní funkcí je převod z časové do zmíněné frekvenční oblasti. Pro korektní interpretaci výsledků analyzovaného signálu je předpokládána jeho stacionarita (jeho amplituda, frekvence a fáze se po dobu měření signálu nemění). V opačném případě, kdy nemůže být stacionarita zaručena a jsou tak zmíněné parametry proměnné v čase, je vhodné použít některé metody z časofrekvenčních metod. Jednou z metod, které se v těchto případech používá, je krátkodobá Fourierova transformace (STFT – Short-time Fourier transform). Kapitola se věnuje těmto metodám spolu s využitím okénkových funkcí. [1, 11]

2.2.1 Diskrétní Fourierova transformace (DFT)

Diskrétní Fourierova transformace našla velké uplatnění v oblasti výpočetní techniky. Vzhledem k celkovému počtu operací N^2, které vyžaduje tento výpočet, se spíše používá algoritmus FFT, který bude popsán v následující kapitole. S využitím této metody jsme schopni rozhodnout, jaké frekvenční složky se v daném signálu vyskytují.

Tento typ transformace se využívá pro vzorkované signály konečné délky N. Výpočet je definován následujícím vztahem:

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-j\frac{2\pi nk}{N}}, k = 0, 1, ..., N-1 \tag{2.3}$$

Posloupnost $X[k]$ zkonstruovaná dle rovnice (13) lze zpětně rekonstruovat do původního signálu x[n] pomocí inverzí DFT, definovaná vztahem:

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k] e^{j\frac{2\pi nk}{N}}, k = 0, 1, ..., N-1 \tag{2.4}$$

Z výše uvedených vztahů je zřejmé, že tato transformace původního signálu o N prvcích převádí na posloupnost jiných prvků o stejné délce.

2.2.2 Krátkodobá Fourierova transformace (STFT)

STFT (Short-time Fourier transform) je označení Fourierovy transformace, která se aplikuje na vstupní signál po krátkých úsecích o velikosti L. Délka úseku L se volí dle velikosti tzv. okénka, jejichž použití je popsáno v kapitole [2.2.4]. Jak již bylo řečeno, tak Fourierova transformace předpokládá po dobu trvání vstupního signálu $x[k]$ jeho stacionaritu. Ta však nelze vždy zaručit a z toho důvodu se používá metoda STFT, která rozdělí vstupní signál do jednotlivých segmentů o délce L. O těchto segmentech se následně předpokládá, že jsou stacionární, tudíž na každý z nich lze aplikovat Fourierovu transformaci. V případě vzorkovaném signálu $x[k]$ se využívá diskrétní STFT, která je definovaná vztahem [2.5].

$$X[m,n] = \sum_{k=0}^{L-1} x[k]w[m-k]e^{-j\frac{2\pi nk}{L}} \tag{2.5}$$

Kde w označuje váhovou funkci (např. Hannovo, Hammingovo okénko) o délce L.

2.2.3 Rychlá Fourierova transformace (FFT)

Jedná se o efektivní algoritmus pro spočtení diskrétní Fourierovy transformace a její inverze. Samotná DFT vyžaduje N^2 komplexních součinů, což může být v některých případech časově náročné. Avšak můžeme využít poznatku, že pro reálný vstupní signál o sudém počtu prvků získáváme symetrické a komplexně sdružené spektrum kolem svého středu (reálná část je kolem středu sudá a imaginární složka lichá). To znamená, že pro výsledné spektrum postačuje vypočíst pouze první polovinu spektra, kde následně druhá část identicky kopíruje první s opačným znaménkem u imaginární složky.

Rozdělíme-li DFT výpočet na dvě téměř identické rovnice, kde v první uvažujeme pouze sudé prvky g[n] a v druhé pouze liché prvky h[n] vstupní posloupnosti x[n], poté jednotlivé vztahy mají tvar [2.6] a [2.7].

$$g[n] = x[2n] \tag{2.6}$$

$$h[n] = x[2n+1] \tag{2.7}$$

Poté lze rovnici DFT zapsat v následujícím tvaru:

$$X[k] = \sum_{n=0}^{\frac{N}{2}-1} (g[n]e^{-j\frac{4\pi nk}{N}} + h[n]e^{-j\frac{2\pi(2n+1)k}{N}}) \tag{2.8}$$

Tímto postupem se celkový počet komplexních součinů zredukuje na polovinu a zároveň výsledek daný touto rovnicí bude stejný jako původní DFT.

2.2.4 Okénkové funkce

Pro korektní zpracování signálu ve spektrální oblasti je vhodné použít tzv. okénkové funkce ke spektrálnímu vyhlazení. Každá taková matematická funkce nabývá nulových hodnot mimo zvolený interval (mimo délku okénka) a její průběh je závislý na typu zvoleného okénka, kde jeho výběr je volen na typu řešené úlohy. Kapitola čerpá z [1, 11] a [7].

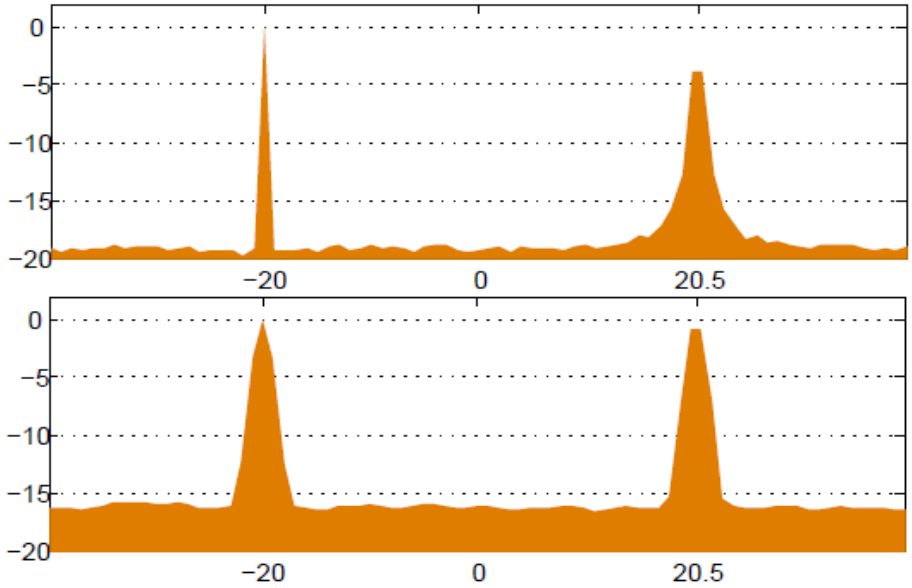
Jedním ze základních typů okénkových funkcí je vhodné uvést tzv. obdélníkové okénko.

$$X[k] = \frac{1}{N} \sum_{n=0}^{N-1} h[n]x[n]e^{-j\frac{2\pi nk}{N}} \tag{2.9}$$

Kde $h[n]$ reprezentuje zmíněnou okénkovou funkci, resp. obdélníkové okénko. Tato funkce je reprezentována následujícím vztahem:

$$h[n] = \begin{cases} 1, & 0 < n < N \quad (2.10) \\ 0, & \text{jinde} \quad (2.11) \end{cases}$$

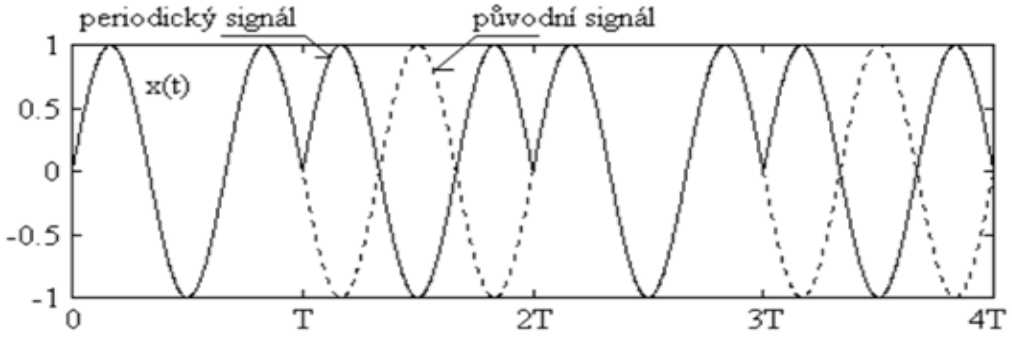
Při použití tohoto typu okénka však vznikají výrazné artefakty, jejichž zobrazení ve frekvenční oblasti přináší nechtěné frekvence, které mohou překrývat námi hledané koeficienty frekvenčního spektra. Takový efekt se nazývá spektrální prosakování (spectral leakage) a u zvoleného obdélníkového okénka k němu dochází více než u ostatních okének. [7] Pro příklad je zde uveden signál, který se skládá ze dvou sinusoid o různých frekvencích. Nechtěný efekt prosakování ve spektru může narušit schopnost odlišit jednotlivé frekvence. Tento případ je demonstrován na následujícím grafu.



Obrázek 2.1: Spektrální prosakování

Druhy tohoto narušení jsou obvykle rozděleny do dvou protichůdných skupin. V případě, že se jedná o rozdílné frekvence, pak únik ze silnější komponenty může zakrýt přítomnost slabší. Pokud jsou však frekvence více podobné, pak únik může způsobit, že jsou jednotlivé frekvence od sebe nerozpoznatelné. Nejedná se však pouze o případy s více komponenty.

Při diskrétní Fourierově transformaci dostáváme korektní výsledky pouze v situaci, kdy je analyzovaný signál periodický resp. vyznačený úsek signálu o délce $Ts[1], \ldots, s[T]$ obsahuje celočíselný počet period, které jsou obsaženy ve vstupním signálu. Výše uvedený požadavek vede na obecnou problematiku určení velikosti časového okénka, kterým se poté původní signál hodnotí. Tato problematika je zobrazena na následujícím grafu.



Obrázek 2.2: Volba časového okénka

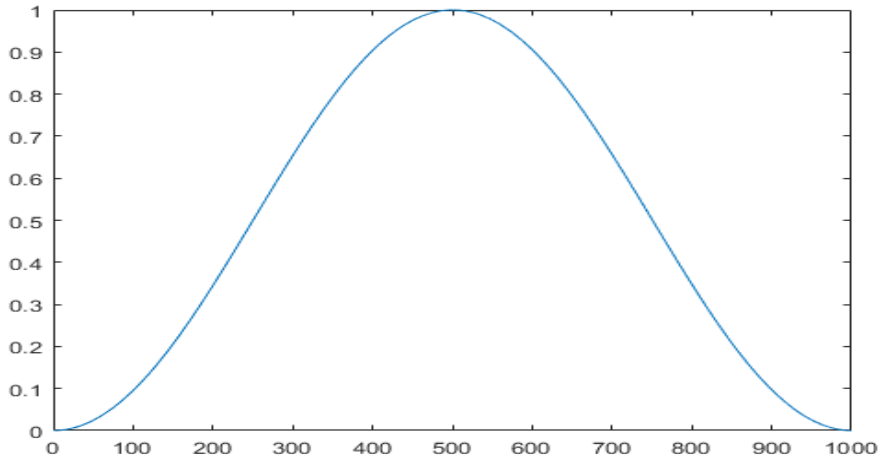
Jelikož DFT předpokládá periodicitu signálu, pak harmonické signály s neceločíselným násobkem své frekvence jsou zaznamenány jako výsek, o kterém se implicitně předpokládá, že je jednou celistvou periodou. Můžeme si všimnout, že původní a předpokládaný periodický signál si jsou navzájem odlišné, což se projeví i ve složení frekvenčních spekter. Toto chování lze do určité míry potlačit snížením hodnot v okolí okrajů okna, tedy jeho typem.

Po součinu části signálu a vhodně zvolené okénkové funkce získáme signál, který obsahuje jednu periodu. Jak již bylo zmíněno, okénkových funkcí je nezměrné množství a výběr je zvolen na základě typu úlohy. Mezi často používané funkce můžeme zařadit Hammingovo a Hanningovo okénko, které budou blíže popsány v následujícím textu.

Hanningova váhová funkce

Hanningovo okénko se hojně využívá v oblasti digitálního zpracování signálu. Je tomu tak z důvodu tvaru Hanningovy křivky [2.12], která má za důsledek dobrý kompromis mezi ostrostí spektra a potlačení falešných frekvencí. [14]

$$h[n] = \begin{cases} \frac{1}{2}[1 - cos(\frac{2\pi n}{N})] & 0 < n < N \\ 0 & ,\text{jinde} \end{cases} \tag{2.12}$$

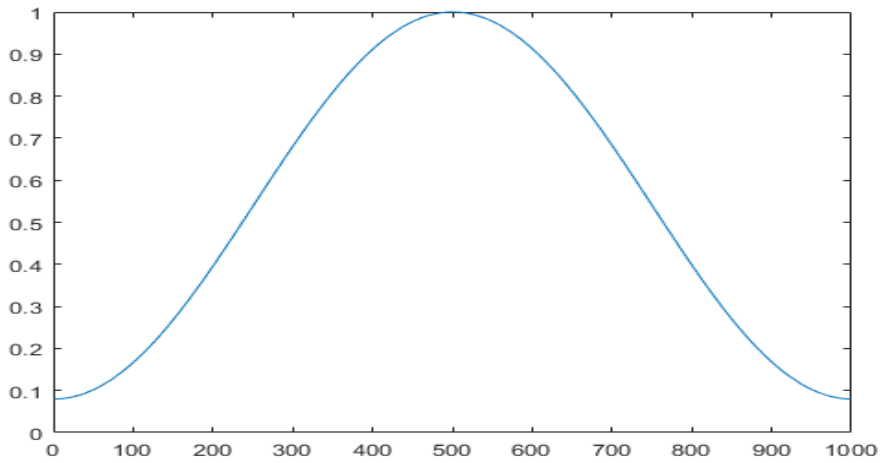


Obrázek 2.3: Hanningova váhová funkce

Hammingova váhová funkce

Jedná se o modifikaci Hanningovy váhové funkce, která se vyznačuje větším útlumem bočních laloků a širším hlavním vrcholem [13]. Jeho průběh je vyznačen na grafu [2.4]

$$h[n] = \begin{cases} 0,54 - 0,46cos(\frac{2\pi n}{N}) & ,0 < n < N \\ 0 & ,\text{jinde} \end{cases} \tag{2.13}$$



Obrázek 2.4: Hammingova váhová funkce

2.2.5 Frekvenční spektrum

Výsledek transformace z časové do frekvenční oblasti je označován jako frekvenční spektrum. V praxi se můžeme setkat s pojmy amplitudové $|X(\omega)|$ a fázové $\Phi(\omega)$ spektrum. Jednotlivá spektra jsou definována pomocí následujících vztahů [2.14], [2.15].

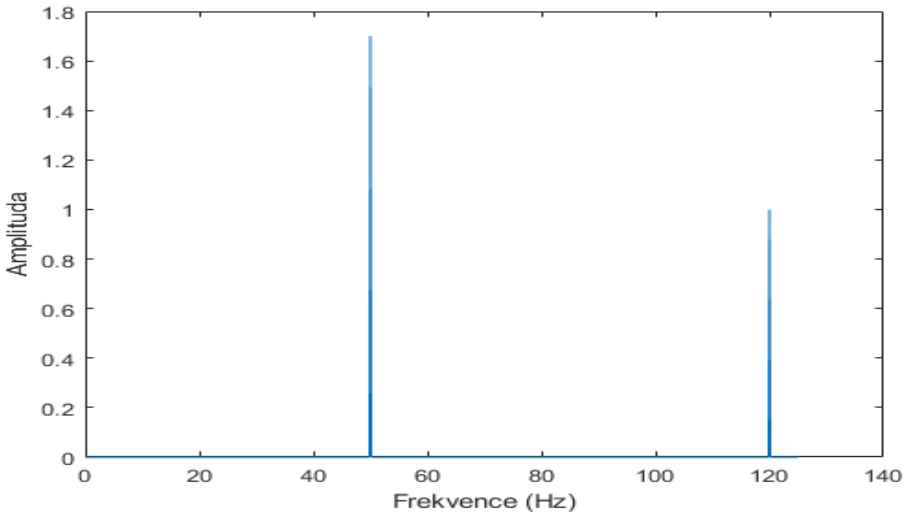
$$|X(\omega)| = \sqrt{Re^2\{X(\omega)\} + Im^2\{X(\omega)\}} \tag{2.14}$$

$$\Phi(\omega) = \arctan(\frac{Im\{X(\omega)\}}{Re\{X(\omega)\}}) \tag{2.15}$$

Tato spektra jsou označována jako oboustranná spektra, jelikož jsou definována jak pro kladné, tak i záporné frekvence ω. V praxi postrádá používání záporné frekvence význam a z tohoto důvodu je používáno jednostranné amplitudové spektrum, které je dáno vztahem $A(\omega)$.

$$A(\omega) = \begin{cases} |X(\omega)| & ,\omega = 0 \\ |X(\omega)| \cdot 2 & ,\omega > 0 \end{cases} \tag{2.16}$$

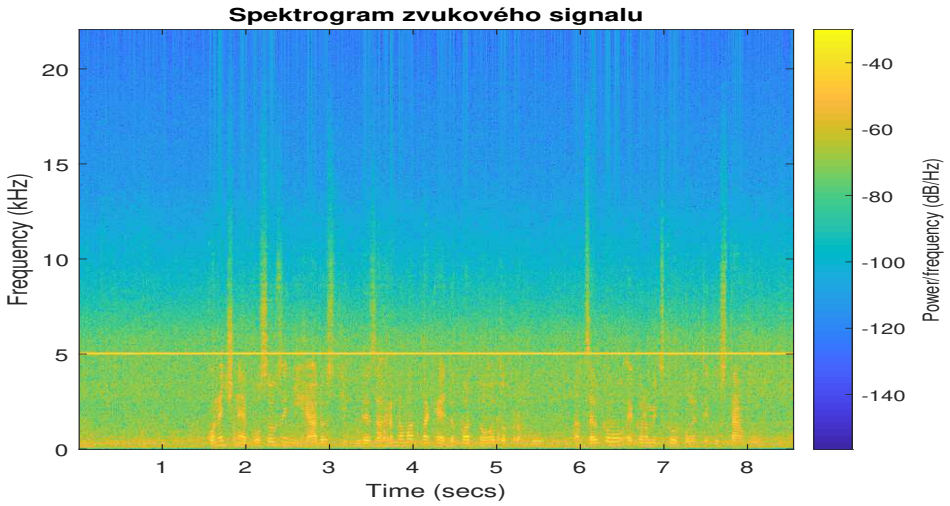
Graf amplitudového spektra slouží k identifikaci jednotlivých frekvenčních pásem obsažených ve zdrojovém signálu a z tohoto důvodu může být označen jako jeden ze základních nástrojů ve vibrační, ale např. i akustické diagnostice. Pro demonstraci je níže uveden graf frekvenčního spektra signálu $x(t)$ obsahující dvě frekvenční složky (50 a 120 Hz).



Obrázek 2.5: Frekvenční spektrum

2.2.6 Spektrogram

Většina reálných signálů, se kterými se v praxi setkáváme, jsou obvykle nestacionární. Pro účely zobrazení těchto nestacionárních dějů slouží spektrogram. V praxi se můžeme setkat s 3D spektrogramy nebo jejich 2D verzí, kde jsou jednotlivé amplitudy zobrazeny ve formě barevné mapy. Zmíněná barva je volena ze zvolené škály a reprezentuje již zmíněnou amplitudu. Pro demonstraci je zde uveden spektrogram zvukové nahrávky, která obsahuje rušivou frekvenci 5 kHz.



Obrázek 2.6: Spektrogram

Jednotlivá frekvenční spektra jsou vypočtena skrze již zmíněnou STFT, avšak je nutné podotknout závislost zvolených parametrů na výsledný tvar spektrogramu. Při větší délce okénka je rozlišení spektrogramu lepší ve frekvenci a v opačném případě lepší rozlišení v čase (princip neurčitosti). V praxi obvykle dochází ke kompromisům při volbě těchto parametrů. V aplikaci, jak bude zmíněno v následujících kapitolách, se uživatel může setkat s pojmy Single-blade spektrum a All-blade spektrum. Jedná se též o spektrogramy s rozdílnými vstupními signály. All-blade spektrum uvažuje za vstupní signál data z celého senzoru X. Tedy výchylky jednotlivých lopatek $b_i(t)$ v pořadí, ve kterém byly senzorem X detekovány. Výsledný signál $s(t)$ má tvar [2.17].

$$s(t) = \{b_1(t_1)), b_2(t_1), ..., b_N(t_1), ..., b_1(t_T), b_2(t_1), ..., b_N(t_T)\} \tag{2.17}$$

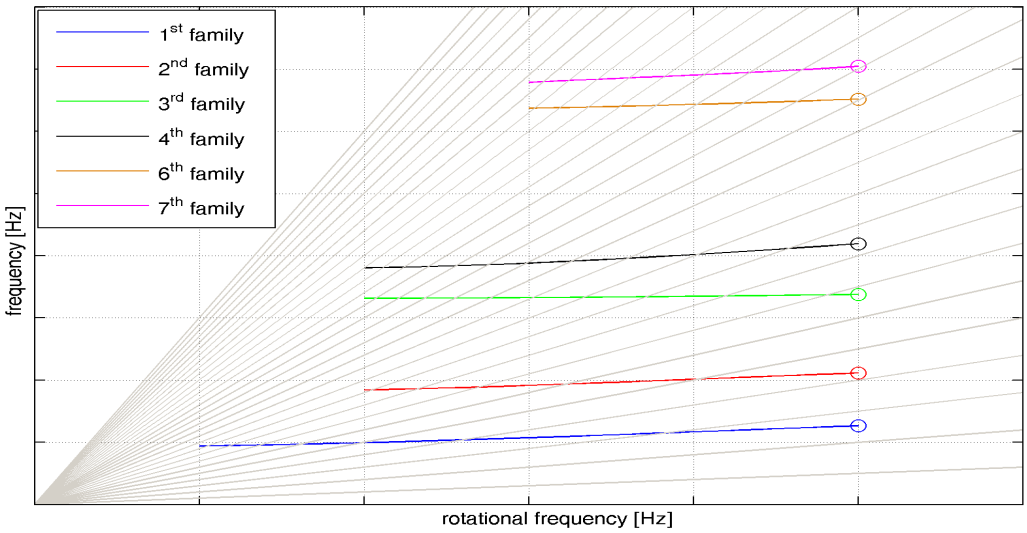
Kde N značí celkový počet lopatek a T konečný časový úsek signálu. V případě single-blade spektra se uvažuje vstupní signál jako pouhé výchylky N_{BLD}-té lopatky detekované senzorem X. V tomto případě je výsledný signál $s(t)$ ve tvaru [2.18].

$$s(t) = \{b_{N_{BLD}}(t_1), b_{N_{BLD}}(t_2), ..., b_{N_{BLD}}(t_T)\} \tag{2.18}$$

Kde N_{BLD} značí index vybrané lopatky z intervalu $\langle 1, N \rangle$ a T uvažujeme opět jako konečnou hodnotu časového intervalu měřeného signálu výchylky lopatky $b_{N_{BLD}}$(t). K výslednému signálu tedy dojdeme decimací signálu senzoru X faktorem N_{BLD}.

2.2.7 Campbellův diagram

Vlastní frekvence lopatek se zpravidla mění s frekvencí budicí síly (otáčková frekvence). V případě, že se tato síla mění, mění se i vlastní frekvence kmitání pro daný uzlový průměr. Při zvyšující se otáčkové frekvenci je zvýšena tuhost lopatek a to zapříčiní i změnu frekvence kmitání. Campbellův diagram zobrazuje závislost vlastních frekvencí lopatek na otáčkové frekvenci.

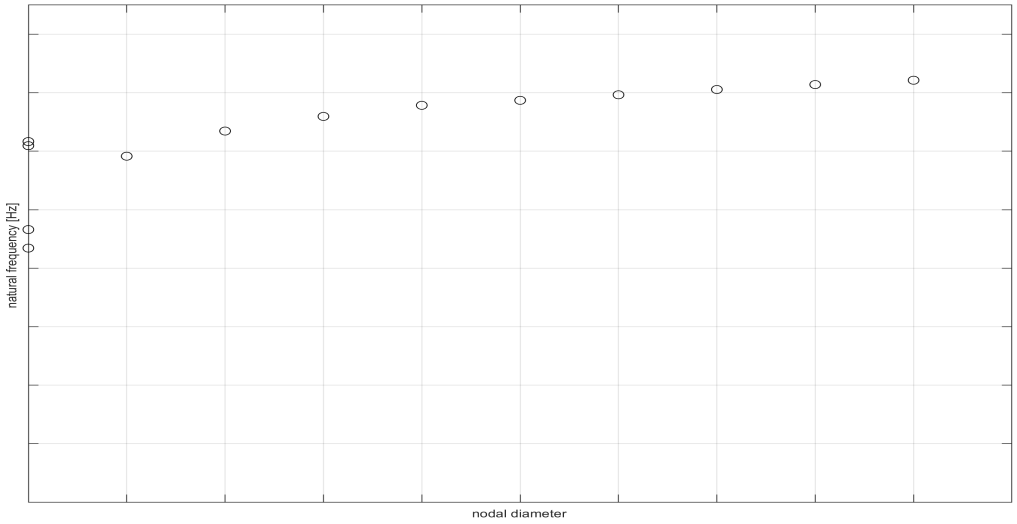


Obrázek 2.7: Campbellův diagram

Vlastní frekvence kmitání a otáčkové frekvence pro n-tý vlastní tvar a všechny uvažované uzlové průměry tvoří dohromady n-tou rodinu (family) [11]. Na výše uvedeném grafu jsou uvedeny rodiny 1 - 7.

2.2.8 Safe diagram

Samotný Campbellův diagram zobrazuje závislost vlastních frekvencí lopatek na otáčkové frekvenci. Není v něm však zahrnuta informace, o jaký ND (uzlový průměr) se jedná. K tomuto účelu lze použít tzv. Safe diagram, který pro vybranou frekvenci budící síly zobrazuje závislost vlastní frekvence na hodnotě ND.



Obrázek 2.8: Safe diagram

3 Použité technologie

V následující kapitole jsou uvedeny technologie, které byly použity při vývoji softwarového řešení diplomové práce.

3.1 Visual Studio

Jedná se o vývojové prostředí (IDE – Integrated Development Environment) vytvořené společností Microsoft. Možnosti využití tohoto vývojového studia jsou velice široké. Lze vytvořit konzolové aplikace nebo aplikace s grafickým prostředím. Tvorba těchto aplikací probíhá v editoru kódu obsahující funkci zvanou IntelliSense. Tento nástroj usnadňuje tvorbu kódu ve formě automatického dokončování psaného výrazu. Návrhy dokončování se nabízí ve formě vyskakovacího seznamu. Díky této funkcionalitě je psaní kódu přívětivější a mnohem rychlejší. Dalším velice užitečným nástrojem je tzv. Debugger, který v průběhu chodu aplikace umožní zkontrolovat její stav nebo krokovat jednotlivé funkce.

Visual studio podporuje jazyky prostřednictvím jazykových služeb, což umožňuje plnou podporu jakéhokoliv programovacího jazyka. Mezi vestavěné jazyky patří C/C++ nebo například C#. Samozřejmostí je však možnost doinstalace podpory dalších programovacích jazyků, jako je například Python nebo Ruby skrze rozšiřovací balíčky.

Další funkcí je Designer, díky které je možné vytvářet grafický obsah aplikace. Tento obsah může být tvořen přetažením (funkce Drag and Drop, která bude blíže popsána v následujících kapitolách) ovládacích prvků z příslušného seznamu na pracovní plochu, kdy je následně skrze Designer vygenerován XAML kód pro UI (User Interface), nebo druhou metodou v podobě přímého vepsání kódu do XAML struktury. Podporuje všechny funkce WPF včetně propojení dat skrze Binding (širší rozprava v následujících kapitolách)

3.2 WPF

Windows Presentation Foundation (WPF) framework, který je součástí .NET frameworku, slouží k tvorbě moderních GUI aplikací. Disponuje širokou paletou ovládacích prvků (Controls), jako jsou například popisky

(Label) či různé typy tlačítek (Button), díky které je možná tvorba graficky bohatých aplikací.

Předchůdcem WPF jsou Windows Forms (WinForms), avšak i přes technologicky vyspělejší framework WPF jsou ve vývojovém prostředí Visual Studio stále přítomny. Je tomu tak z důvodu, že některé dosavadní aplikace jsou tvořeny zmíněným WinForms frameworkem. Oba zmíněné frameworky jsou firmou Microsoft v době tvorby diplomové práce podporovány a aktualizovány.

3.3 XAML

Značkovací jazyk XAML (eXtensible Application Markup Language) využívaný k tvorbě prezentační vrstvy grafické aplikace. Struktura XAML dokumentu je stromová a skládá se z tzv. elementů. Základem dokumentu je však jeden kořenový element, ve kterém jsou umístěny všechny ostatní elementy. Pod pojmem element si můžeme představit tag zapsaný v lomených závorkách. Elementy dělíme na párové a nepárové, jejichž vlastnosti jsou určeny ukončovacím tagem (viz. Následující ukázky kódu)

```
</Label>
```

Kód 3.1: Nepárový element

```
<Grid></Grid>
```

Kód 3.2: Párový element

Mezi obrovskou výhodu WPF patří, že XAML rozděluje do dvou souborů, které obsahují zvlášť uživatelské rozhraní (.xaml) a programovou část (.cs).

3.4 MVVM

Model-View-ViewModel (MVVM) je návrhový vzor pro WPF aplikace, který nabízí celkem jednoduché řešení jak od sebe oddělit logiku aplikace od uživatelského rozhraní. Důsledkem při implementaci je méně obsáhlý a přehledný kód, při jehož změně dochází k méně modifikacím.

Hlavním principem je vytvoření třídy, která drží stav aplikace. Ta je označována jako ViewModel a v aplikaci zastává pozici prostředníka mezi

načtenými daty a uživatelským rozhraním (dále jako View). Veškerá logika aplikace by měla fungovat právě ve zmíněném ViewModelu. View by se mělo pouze na aktuální stav dotazovat nebo být upozorněno ViewModelem na jejich změnu a tu dle předem daných pravidel vykreslit. V opačném případě, kdy uživatel zadá některé informace do View (např. zadání textu), by se měly dané údaje zpropagovat do příslušného ViewModelu, kde by se měly zpracovat. Oba uvedené případy jsou ve WPF lehce zpracovatelné díky procesu DataBinding.

3.5 DataBinding

Jedná se o mechanismus, který poskytuje automatickou synchronizaci dat a uživatelského rozhraní, resp. datového zdroje a jeho cílem. V případě vývoje softwaru se nejedná o nezbytnou implementaci, avšak při změně zdroje či cíle by se musela vyvolat logika, která by musela být předem vytvořena. Což by bylo v případě rozsáhlého softwaru velice časově náročné. Nejlepší volbou je tedy použít zmíněný DataBinding, který má zmíněnou logiku již implementovanou. Ve WPF můžeme tímto způsobem provázat jakýkoliv objekt nebo jeho vlastnost s příslušným grafickým prvkem, o jehož změně bude vždy příslušná protistrana informována.

```
<DataGridTextColumn Binding="{Binding Label, Mode=OneWay}"/>
```

Kód 3.3: DataBinding

3.6 DataTemplate

V moderních aplikacích je běžné, že se data nezobrazují výchozím způsobem. Tento způsob i přes snahu moderního vzhledu WPF prvků působí velice jednoduše a pro uživatele se takto definovaný vzhled stává nepřívětivý. Z tohoto důvodu nabývá na své platnosti zmíněný pojem DataTemplate, díky kterému lze specifikovat přesnou podobu vykreslení jednotlivých prvků.

3.7 Drag and Drop

Velkým důrazem při implementaci byl požadavek na funkcionalitu Drag and Drop. Jedná se o operaci, která se v široké míře uplatňuje v moderních aplikacích, kdy uživatel tzv. „uchopí“ grafický objekt a přesune ho na cílové místo, od kterého následně očekává interakci. Obrovskou výhodou je využití této operace na dotykových zařízeních. Software je tedy možné využít i na tabletu či notebooku s dotykovou složkou a operačním systémem Windows. V oblasti aplikace byl tento proces využit hlavně v oblasti výpočtů [5.3.2].

3.8 Programovací jazyk C#

Jedná se o vysokoúrovňový objektově orientovaný jazyk (širší rozprava v následující kapitole) vyvinutý firmou Microsoft. Samotná syntaxe je velmi blízká jazyku C/C++, ze kterého sama čerpá. Překladače jazyka C# jsou case sensitive, rozlišují tedy velká a malá písmena uvnitř zdrojového kódu.

```
namespace Application
{
    /// <summary>
    /// Interaction logic for MainWindow.xaml
    /// </summary>
    public partial class MainWindow : Window
    {
        public MainWindow()
        {
            InitializeComponent();
        }
    }
}
```

Kód 3.4: Struktura programovacího jazyka C#

3.9 .NET Framework

.NET Framework je rozsáhlá softwarová platforma, která je určena pro vývoj různých druhů aplikací. Skrze tento framework se vyvíjí mnoho jejich typů od klasických pro Windows, až po webové aplikace a aplikace pro mobilní zařízení. Obsahem frameworku je velká škála knihoven, mezi kterou řadíme i Base Class Library (BCL), což je označení sady základních knihoven. Dalším obsahem jsou funkce, ale i běhové prostředí CLR (Common Language Runtime), které zajišťuje běh a kompilaci aplikace.

3.10 LINQ

Language INtegrated Query (LINQ) je integrovaný jazyk .NET frameworku, jehož přínosem je uvedení jednotné syntaxe pro přístup k datům bez ohledu na jejich typ. Jedná se tak o soubor nástrojů, které usnadňují a zobecňují práci s dotazováním se na data, která jsou součástí kolekce více prvků. Samotná syntaxe je velmi přívětivá a vývojář při její znalosti ušetří čas díky zredukování výsledného kódu. Jako příklad je zde uvedeno jednoduché přenásobení částí signálu okénkovou funkcí hanning [2.2.4], na kterém lze přímo demonstrovat zjednodušení kódu.

```
for (int index = 0; index < part.Length; index ++)
{
        part[index] = part[index] * hanning[index];
}
```

Kód 3.5: Klasické použití For cyklu

```
part = part.Zip(hanning, (d1, d2) => (d1 * d2)).ToArray();
```

Kód 3.6: Použití jazyka LINQ

Využití tohoto nástroje není však jen v přenásobení prvků jedné kolekce jinou. Je možné použít například tyto funkce (mimo jiných):

- Select
  - Výběr hodnoty z kolekce (ekvivalent Map funkce z jiných jazyků)
- Where
  - Výběr prvků dle specifických podmínek (ekvivalent Filter z jiných jazyků)
- Empty / Repeat / Range
  - Metody vytvářející kolekce daného typu
- Join
  - Spojení dvou kolekcí do jedné
- Reverse
  - Změna pořadí prvků v kolekci
- ToArray, ToList, ToDictionary
  - Převedení kolekce na vybraný typ

Tento dotazovací jazyk byl integrovaný přímo do syntaxe jazyka C# od verze 3.0 a .NET frameworku 3.5. Od novějších verzí je tato knihovna vylepšena o vícevláknové zpracování, což zvyšuje efektivitu této technologie. Nejedná se však jen o nástroj pracující s objekty kolekce. Lze jej využít i v oblasti SQL nebo XML parsování.

3.11 Math.Net Numerics

Jedná se o open-source knihovnu pro .NET framework napsaná v jazyce C# a F#. Tato knihovna si klade za cíl poskytnout metody a algoritmy pro numerické výpočty. Je několik oblastí, kterým se tato knihovna věnuje, sahajících od pravděpodobnostních modelů, až po integrální transformace a mnohá další odvětví.

Využití této knihovny bylo v oblasti signálů a jejich transformací. Při optimalizaci poměru časových a paměťových nároků jednotlivých operací byly využity matice (podrobnější popis problematiky optimalizace je uveden v kapitole č. 7) a v oblasti výpočtů komplexní čísla a okénkové funkce [2.2.4].

3.12 OOP

Object-Oriented Programming (OOP) neboli objektově orientované programování je metoda programování, při které se využívají objekty. Pro širší porozumění tomuto typu programování je vhodné definovat pojem třída.

Třída(Class) je abstraktní vzor (šablona), která definuje vlastnosti a metody, které objekt příslušný k této třídě obsahuje. Avšak neobsahuje jeho konkrétní hodnoty (ty obsahuje až vytvořený objekt)

```
public class Dialog
{
    public string Title {get; private set;}
    public string Message { get; private set; }

    public Dialog(String title, String message)
    {
        this.Title = title;
        this.Message = message;
    }
}
```

Kód 3.7: Třída (Class)

Objekt poté můžeme definovat jako konkrétního jedince dané třídy (typu) s konkrétními hodnotami.

```
private void Application_Loaded(object sender, RoutedEventArgs e)
{
    Dialog modal = new Dialog("Information", "Hello World" );
}
```

Kód 3.8: Objekt (Object)

S těmito pojmy úzce souvisí pojem polymorfismus neboli dědičnost. Je to jedna ze základních vlastností OOP a slouží k vytvoření nových datových struktur, jejichž vlastnosti jsou shodné s prvkem, ze kterého je podděděné. Zároveň může obsahovat vlastnosti rozdílné od zdrojového prvku. Případně může určitá třída pevně nastavit některé hodnoty (v níže uvedeném případě třída $InfoDialog$ vždy nastaví hodnotu dialogu $Title$ na "Info")

```
public class InfoDialog : Dialog
{
    Public InfoDialog(String message) : base( "Info", message)
    {

    }
}
```

Kód 3.9: Dědičnost (Polymorphism)

3.13 JSON

JavaScript Object Notation je otevřený standart způsobu zápisu dat, resp. jeho datové struktury. Jeho obrovskou výhodou je, že je nezávislý na počítačové platformě, a je ho tedy možné použít napříč různými systémy. Samotný zápis datových struktur je platným zápisem jazyka Javascript a lze do něj uložit následující výčet datových typů.

- JSONString - textový řetězec
- JSONNumber - číslo (celočíselné, reálné, včetně zápisu s exponentem)
- JSONBoolean - logická hodnota
- JSONNull - hodnota null
- JSONArray - pole
- JSONObject - objekt

Ostatní datové typy nelze vkládat přímo, například datum pro vložení do JSON se musí převést do JSONString. Tento typ zápisu se uplatňuje například při uložení dat aplikace do perzistentní paměti a jejich následnému vyčtení při opětovném startu aplikace nebo v době jejich potřeby v případě smazání (opětovné načtení).

```
{
  "File": " C:\\source_data.sql",
  "TimeFrom": "2019-07-25 11:55:03",
  "TimeTo": "2019-07-25 15:54:59",
  "SignalType": "BTT",
  "BladesNo": 45,
  "SamplFreq": 100000000,
  "SensorNo": 8
}
```

Kód 3.10: JSON struktura

3.13.1 Parsování v jazyce C#

Pro práci s JSON soubory byla v aplikaci využita knihovna Json.NET od společnosti Newtonsoft. Jedná se o populární a vysoce výkonnou softwarovou knihovnu s intuitivním použitím.

```
        string Content = GetTextFileContent(FilePath);
Header result = JsonConvert.DeserializeObject<Header>(Content);
```

Kód 3.11: Použití knihovny Json.NET

3.14 SQL

Structured Query Language (Strukturovaný dotazovací jazyk) SQL – je obecný nástroj pro manipulaci, správu a organizování dat uložených v databázi. Skrze SQL lze definovat strukturu dat (pole záznamů) a následně se na ně skrze tento jazyk dotazovat, mazat nebo upravovat pomocí jednotlivých dotazů. Samotné uložení dat můžeme chápat ve formě tabulek, ve kterých jsou předem nadefinované sloupce. Jednotlivé záznamy uložené v těchto databázích mají své uplatnění v oblasti aplikací určené právě k selekci či modifikaci dat.

Obrovská kvanta dat, jejich selekce a následné zpracování může vést při neoptimalizaci databáze na dlouhodobé zpracování. Avšak i přes optimalizaci databázové struktury může být pouze samotný přenos dat k cílovému stroji, jenž je chce vyčíst, velice zdlouhavý. Z tohoto důvodu se při offline analýze dat využívá tzv. exportů těchto dat. Samotný export je díky dnešním uživatelským rozhraním databázových systémů velice jednoduchý. Je možné vyexportovat data například ve formě JSONu, CSV nebo přímo SQL formátu.

```
2 3 1 0 1564055703209 45 45 1330725 1999315 39811
3 4 1 0 1564055703209 45 45 1330725 1999315 22388
```

Kód 3.12: Struktura SQL souboru

Tento formát byl zvolen pro zdrojové soubory naměřených signálů časových průletů lopatek. Výhodou takového typu exportu je minimalizace velikosti výsledného souboru, jelikož neobsahuje informace o významnosti jednotlivých sloupců (viz. JSON struktura). Je tedy nutné mít tuto informaci ve formě metadat, které budou obsaženy v příslušném souboru nebo v aplikaci, která bude tyto soubory zpracovávat.

3.15 Multithreading

Operační systém Windows je víceúlohový operační systém, tedy systém s možností spuštění více aplikací v jednom čase (Multitasking). Aplikaci můžeme tedy definovat jako proces, v němž může být spuštěno více vláken (Multithreading). Každá aplikace má minimálně jeden proces, ve kterém běží jeho hlavní vlákno. Zbylá vlákna si volí vývojář sám. Pokud by tomu tak neučinil a spustil časově náročný výpočet v hlavním vlákně, zasekl by tak hlavní vlákno a aplikace by zamrzla. To by způsobilo nemožnost ovládání grafických prvků aplikace a samotný operační systém by na to zareagoval jako na špatně fungující aplikaci.

3.15.1 ThreadPool

Samotné vytvoření nového vlákna je z pohledu času, ale i paměťových nároků dosti náročná operace. Z tohoto důvodu není vhodné vytvářet nová vlákna, ale recyklovat již ta vytvořená. Tímto způsobem jsou redukovány nároky na samotný systém. ThreadPool tedy zajišťuje, aby neběželo příliš mnoho vláken a jednotlivé procesy byly neefektivní.

3.16 SciChart

Pro možnost vizualizace bylo potřeba výběru vhodné knihovny, díky které by bylo možné daná data vykreslit, a následně by byla uživateli umožněna práce s grafem, kterou by požadoval. Po podrobnějším zkoumání několika knihoven byla vybrána knihovna SciChart.

Obrázek 3.1: SciChart logo

Společnost (SciChart Ltd) spravující zmíněnou knihovnu pravidelně vydává nové verze a má vyčleněný tým specialistů, kteří pracují na technické podpoře, což bylo v případě problematických situací velice vítané. Zároveň tuto knihovnu využívají pro svá softwarová řešení velké firmy, jako jsou Samsung, Microsoft, Intel nebo například Siemens. Tato knihovna získala několikanásobná ocenění za rychlost vykreslování dat. Knihovna mimo jiné funkcionality podporuje:

- Vykreslení XY grafů
- Vykreslení HeatMap (využito pro spektrogramy)
- Přiblížení grafů
- Škálogramy pro zmíněnou HeatMap

Mezi další aspekty, díky kterým byla tato knihovna vybrána, je možnost stylizace výsledného grafu. Jelikož grafické prostředí aplikace bylo specificky navrženo, bylo nutné k tomu uzpůsobit i samotné prvky vizualizace dat.

3.17 NuGet

Jedná se o mechanismus pro moderní vývojové platformy, jehož prostřednictvím mohou vývojáři vytvářet, sdílet a využívat užitečný kód. Tento kód je distribuován skrze tzv. balíčků, které obsahují kompilovaný kód spolu s dalším obsahem, které jsou potřebné pro tyto balíčky [20].

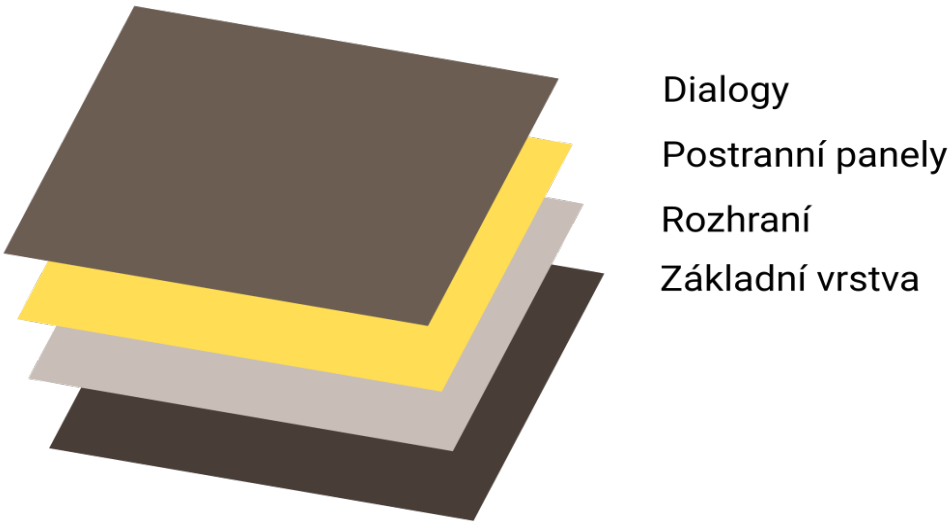
Tento mechanismus je podporován firmou Microsoft a lze jej jednoduše využít ve vývojovém prostředí Microsoft Visual Studio [3.1]. Můžeme ho tedy využít k distribuci našeho kódu nebo k jednoduché instalaci požadovaných knihoven (např. Json.NET) do projektu aplikace. Při releasu nové verze knihovny je vývojář informován o nové verzi softwaru, který je v projektu použitý a lze jej i snadno aktualizovat.

4 Grafický návrh aplikace

Samotný návrh aplikace vychází z požadavků na jednoduché a intuitivní ovládání s využitím veškerého výkonu, kterým koncová stanice disponuje. Zároveň byl důraz na jednoduchou práci s grafy a využití plné plochy aplikace.

4.1 Uživatelské rozhraní

Specifikace uvedené v předchozí kapitole vedly k návrhu uživatelského rozhraní, které obsahuje více překrývajících se vrstev (viz. následující obrázek)

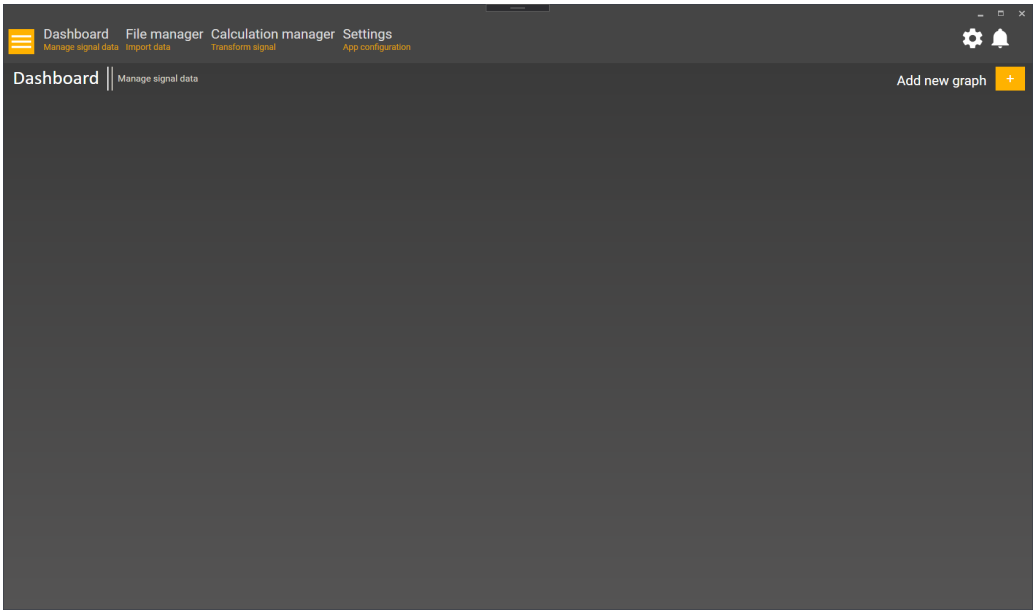


Obrázek 4.1: Návrh grafického rozhraní

Z výše uvedeného grafického rozložení je zřejmé, že se aplikace skládá ze 4 vrstev, jejichž překrytí je dáno pořadím ve vertikálním umístění grafické struktury (Dialogy se vykreslí před vrstvou Rozhraní). Zmíněné pořadí bylo zvoleno dle důležitosti informace, jakou můžou jednotlivé vrstvy nabývat.

4.1.1 Základní vrstva

Jedná se o základní kámen grafického rozhraní aplikace. Hlavním prvkem, který se v této vrstvě nachází, je menu. Obsahem menu jsou grafické prvky reagující na uživatelskou interakci v podobě zobrazení prvků jiné vrstvy.



Obrázek 4.2: Struktura uživatelského rozhraní

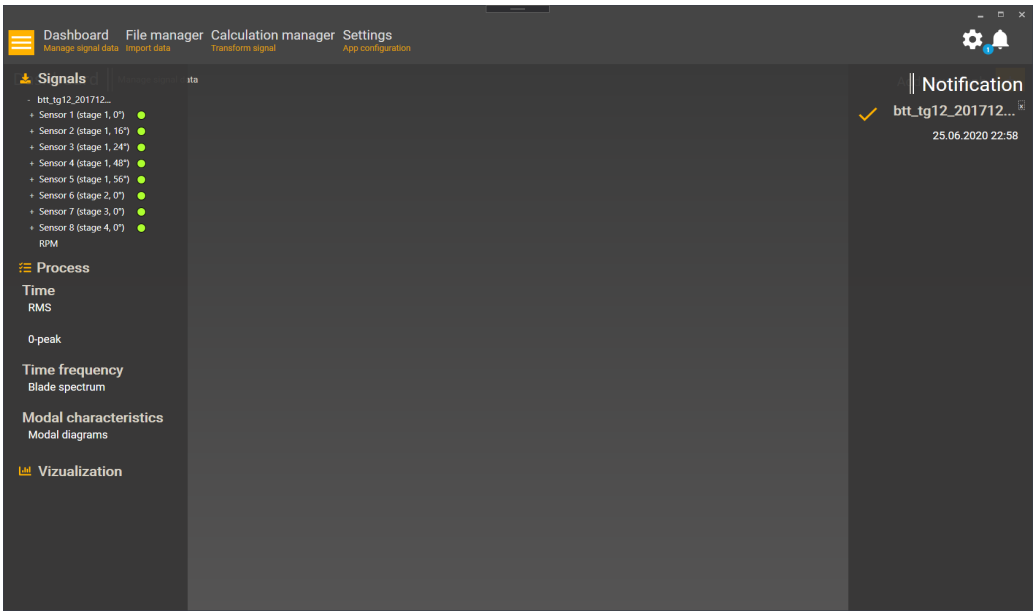
4.1.2 Rozhraní

Pro uživatele pravděpodobně nejdůležitější a nejpoužívanější část celé aplikace. Obsahem je uživatelem vybraná sekce Dashboard, File manager, Calculation manager nebo Settings, jejichž popis bude popsán v následujících kapitolách.

4.1.3 Postranní panely

V aplikaci je možné zobrazit celkem 4 postranní panely.

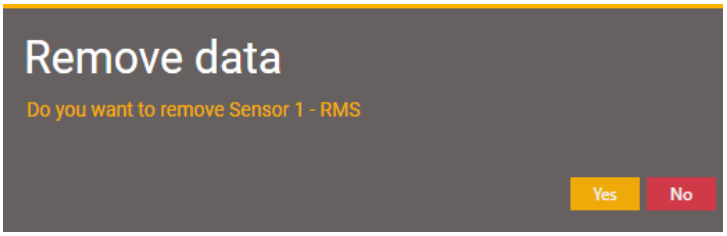
- Panel načtených souborů a výpočtů (dále jen Stromová struktura)
- Panel notifikací
- Panel procesů, které se aktuálně zpracovávají



Obrázek 4.3: Postranní panely

4.1.4 Dialogy

Samotný dialog můžeme prezentovat jako samostatné okno se zobrazenou informací určenou pro uživatele, od kterého se očekává, že s ním bude určitým způsobem interagovat. Tato interakce může být například ve formě kliknutí. Toto kliknutí může znamenat odsouhlasení nebo zamítnutí některého tvrzení, které bylo před vykreslením dialogu vyvoláno.

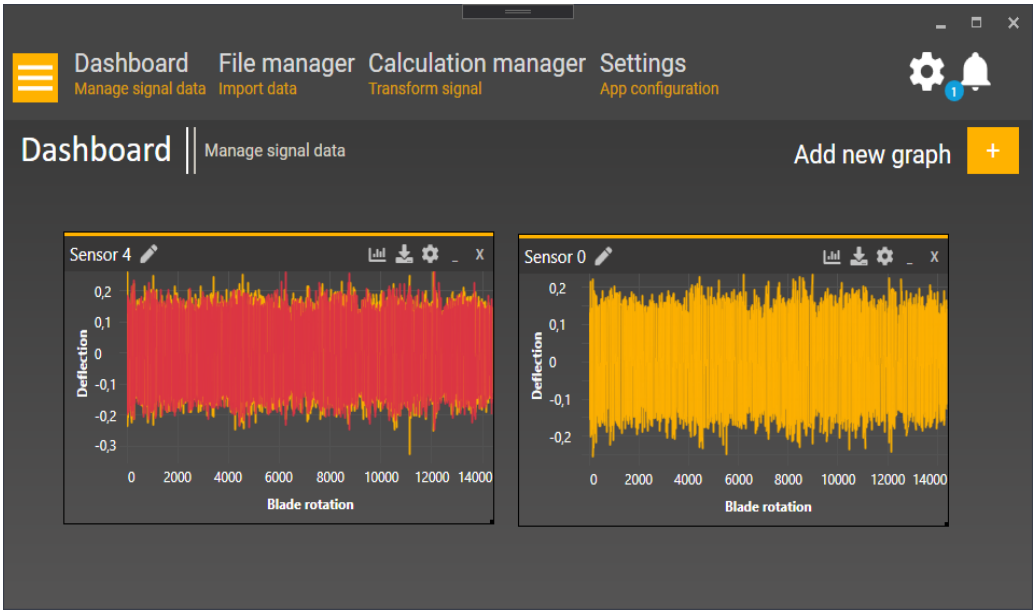


Obrázek 4.4: Dialog aplikace

Další možností je například nastavení některých hodnot skrze tento dialog. Samozřejmostí je, že se tyto dialogy musí vykreslit na nejvyšší vrstvě aplikace a zároveň toto vykreslení nemůže pozastavit jiné procesy.

4.2 Dashboard

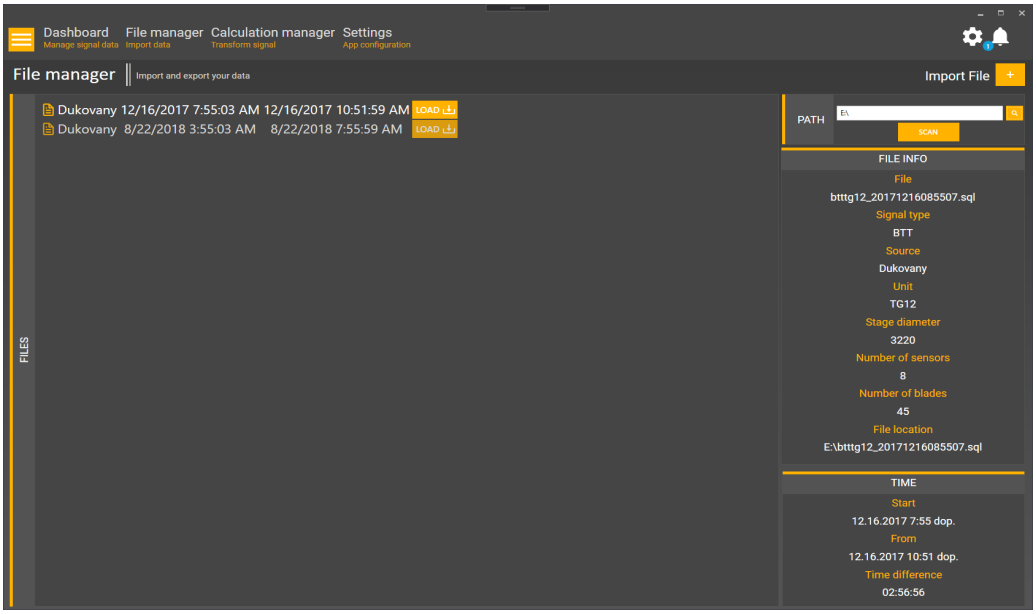
Řídící panel (Dashboard) je typ grafického uživatelského rozhraní, které poskytuje informace ve formě různých grafů. Uživatel si tak může vizualizovat napočtená data a dále s nimi pracovat. Práce s grafy je možná pomocí kurzoru myši, kde je lze například přesouvat, zvětšovat, zmenšovat nebo z pracovní plochy odstranit. Dále lze pojmenovat celý graf nebo jednotlivé osy, měnit režimy grafu, zobrazovat legendu nebo exportovat aktuálně načtenou vizualizaci. Jednotlivé funkce budou podrobněji rozepsány v následujících kapitolách.



Obrázek 4.5: Dashboard

4.3 File manager

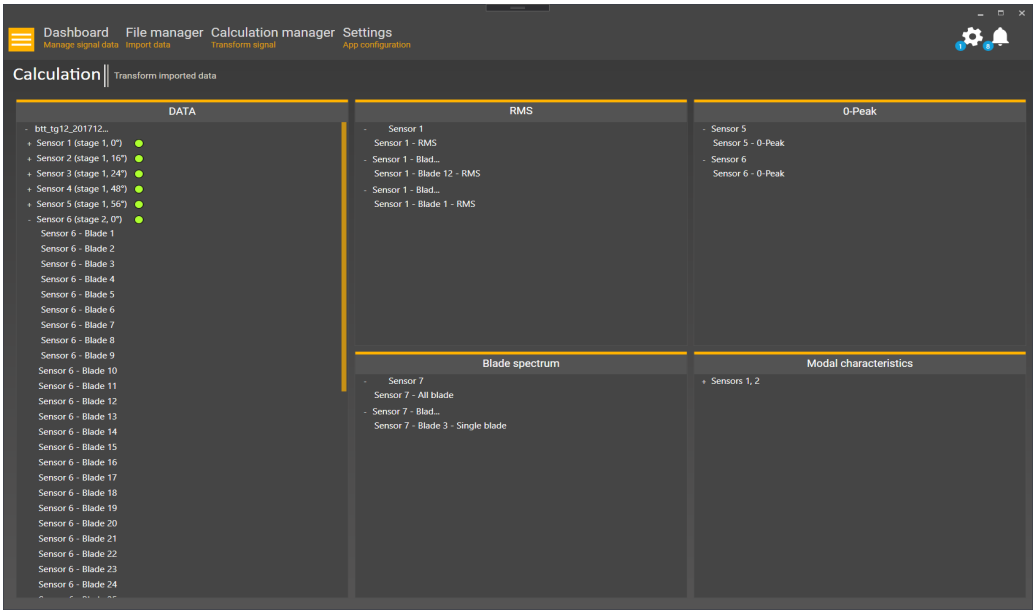
Panel, který je určen pro práci s hlavičkovými a zdrojovými soubory. Poskytuje uživateli informaci o jednotlivých hlavičkových souborech, které jsou již načteny. Další funkcí je možnost importování nového hlavičkového souboru skrze funkce pro import samostatného souboru nebo jejich vyhledávání ve vybrané složce souborů. Tyto funkce jsou podrobněji popsány v kapitole [5.1].



Obrázek 4.6: File manager

4.4 Calculation manager

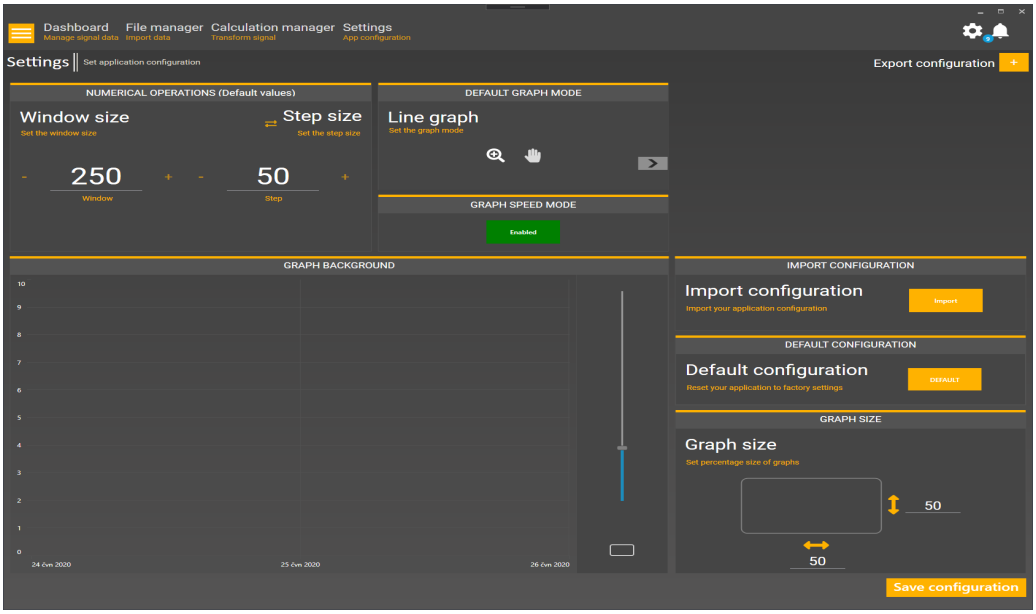
V případě načtených dat s velkým počtem lopatek / senzorů může nastat situace, kdy se stane stromová struktura nepřehledná. Z toho důvodu byl vytvořen samostatný panel, který tento nedostatek eliminuje a poskytuje lepší přehlednost výpočtů (viz. Obrázek [4.7]).



Obrázek 4.7: Calculation manager

4.5 Settings

Posledním panelem je rozhraní pro ovládání a konfiguraci aplikace. Uživatel má zde možnost nastavit výchozí hodnoty nastavení jako jsou hodnoty pro výpočty, jednotlivé módy grafů, jejich pozadí, velikost a boost mód (tzn. zrychlený mód). Dále je možné aktuální konfiguraci ukládat do perzistentní paměti aplikace, importovat nebo exportovat pro použití na jiné klientské stanici. Již modifikovanou konfiguraci lze též změnit do původních hodnot.



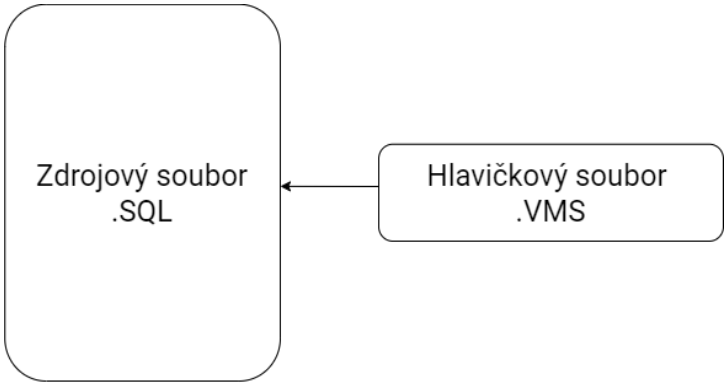
Obrázek 4.8: Settings

5 Funkce aplikace

Následující kapitola obsahuje výčet funkcí, které jsou v aplikaci obsažené. Dále obsahuje jejich popis a možné cíle, ke kterým lze dojít za různých situací.

5.1 Importování hlavičkového souboru

Časy detekce průletu jednotlivých lopatek byly vyexportovány do SQL souborů [3.14]. Bohužel tyto soubory nabývají obvykle velikosti několik TB (Terabyte) a není možné zjistit všechny informace jinou cestou než celý tento soubor zpracovat. Zpracování však vyžaduje určitou dobu, jejíž cena nemusí vést k hledanému výsledku. Z tohoto důvodu byly zavedeny tyto hlavičkové soubory, které obsahují metadata o vybraném .SQL souboru. Zdrojový soubor nelze tedy do aplikace importovat jiným způsobem než pomocí hlavičkového souboru.



Obrázek 5.1: Vztah hlavičkového a zdrojového souboru

Výše uvedený graf ukazuje vzájemný vztah mezi zmíněnými soubory. Samotný hlavičkový soubor je typu .VMS (souborová přípona) a jeho obsah je tvořen skrze JSON formát [3.13]. V té jsou uvedeny podrobné informace o daném souboru (metadata).

- Cesta ke zdrojovému souboru
- Zdroj naměřených dat
- Časové údaje měření

- Typ měřeného signálu
- Počet lopatek
- Měřená frekvence
- Počet senzorů a jejich pozice

Samotná aplikace dokáže zpracovat pouze soubory .VMS s předepsanou strukturou.

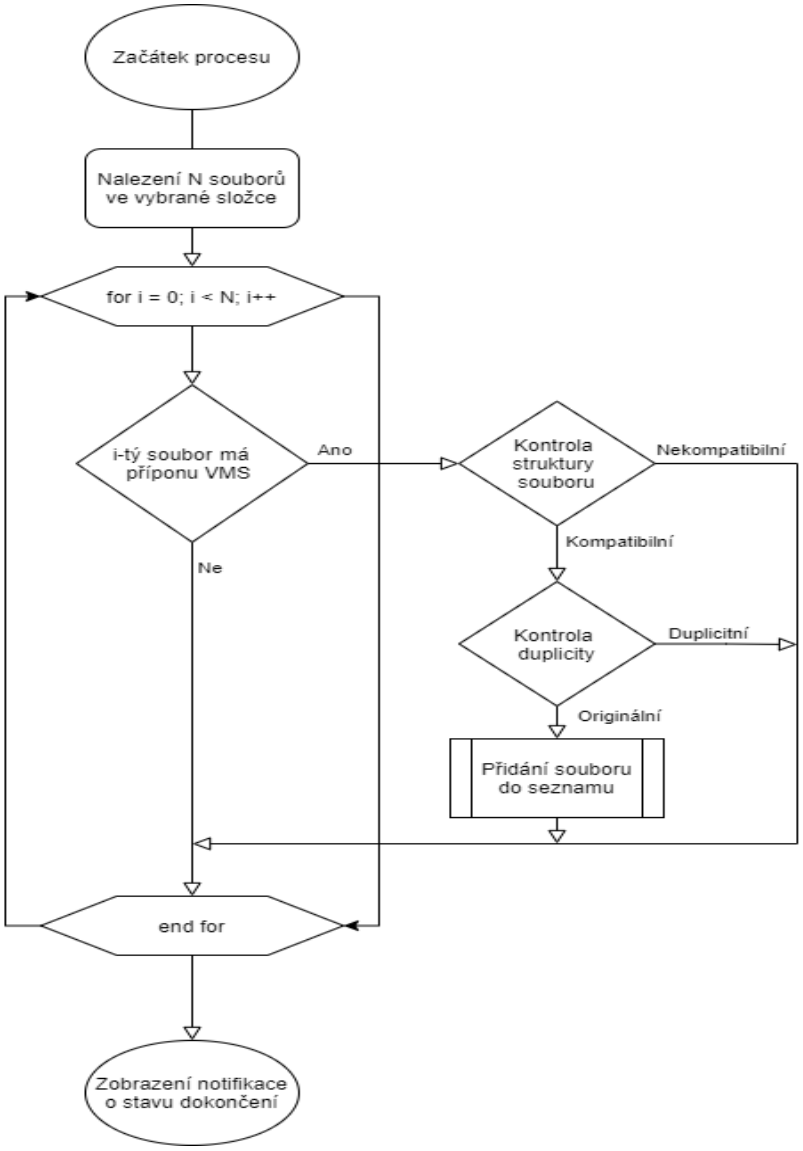
5.1.1 Importování souboru do aplikace

Uživatel má možnost importovat jednotlivé hlavičkové soubory. Při tomto importu je však důležité, aby aplikace zkontrolovala jeho strukturu, a v případě jeho duplicity s již zaevidovaným souborem tento soubor nepoužila.

Obrázek 5.2: Importování hlavičkového souboru

5.1.2 Automatické vyhledávání hlavičkových souborů

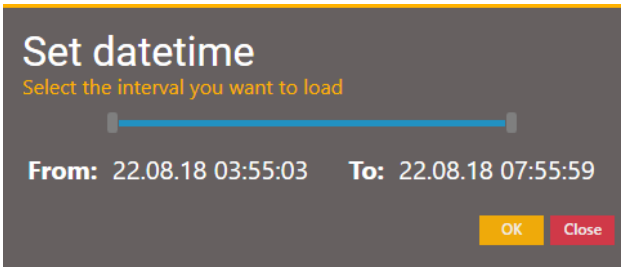
Další funkcí je možnost vyhledávání hlavičkových souborů ve vybrané souborové složce. Algoritmus v tomto případě načte názvy všech souborů ve vybrané složce a následně kontroluje jejich strukturu a duplicitu jako v předchozím případě [5.2]. Rozdíl při kontrole těchto zmíněných vlastností oproti importování samostatného souboru je zobrazení dialogů o chybovém stavu. Jelikož může nastat, že se ve vybrané složce vyskytuje více nekompatibilních souborů nebo souborů, které jsou již v aplikaci zaevidovány, nejsou zobrazovány chybové hlášky a soubor není brán v potaz.



Obrázek 5.3: Vyhledávání hlavičkových souborů

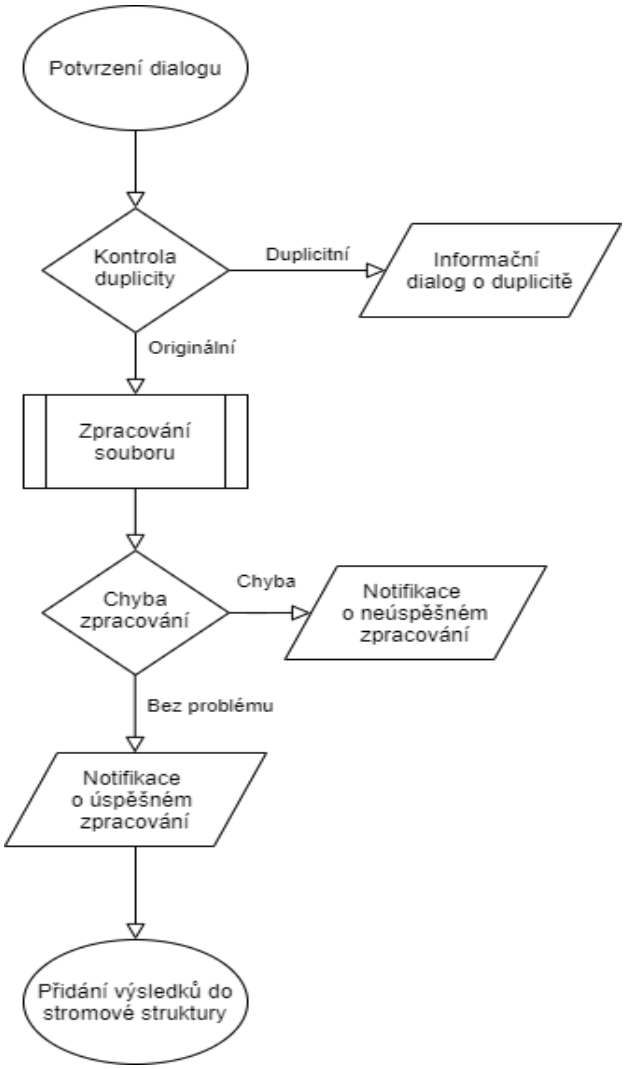
5.2 Načtení zdrojového souboru

V seznamu importovaných hlavičkových souborů je ke každému prvku přidáno tlačítko. Jelikož může nastat situace, kdy hlavičkový soubor odkazuje na zdrojový .SQL soubor, který již neexistuje (byl přesunut nebo vymazán až po načtení hlavičkového souboru), zahajuje se po kliknutí na příslušné tlačítko kontrola, zda přidružený soubor existuje. V případě, že tomu tak není, je vykreslen informační dialog s chybovou zprávou. V opačném případě se zobrazí dialog pro výběr časového úseku zdrojového souboru. Uživatel má možnost potvrdit vybraný úsek nebo dialog uzavřít. Pokud byl dialog potvrzen, vytvoří se proces s parametry z hlavičkového souboru a délky časového úseku.



Obrázek 5.4: Dialog výběru časového úseku

V případě potvrzení dialogu dochází opět k následné kontrole duplicity, tedy kontroly, zda již v načtených datech vybraný soubor s vybraným časovým intervalem neexistuje. Pokud duplicitní objekt není v aplikaci evidován, dochází k přečtení zdrojového souboru po řádcích, ale k načtení hodnot dochází pouze v případě, že časová informace náleží do vybraného časového intervalu. Pokud aplikace vyčte časovou značku, která je již za konečnou hodnotou časového intervalu, pročítání se ukončí a následně se zpracují načtená data.



Obrázek 5.5: Potvrzení dialogu

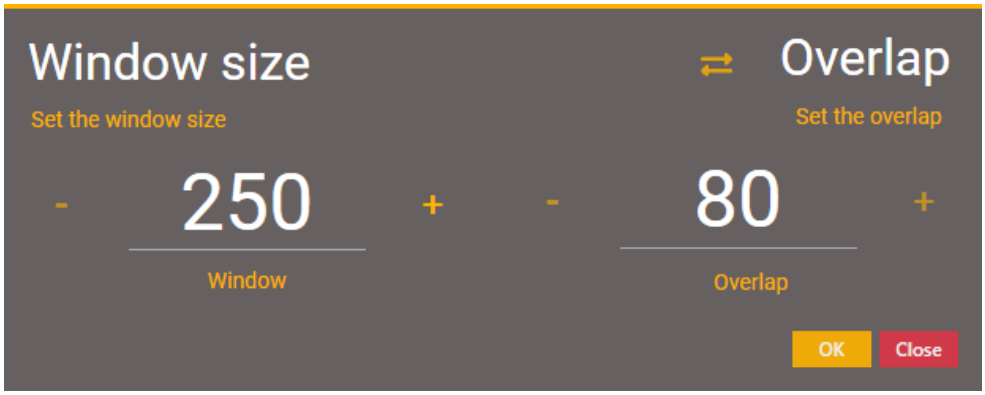
5.3 Výpočty

Importované soubory lze ve stromové struktuře přesunout do sekce libovolného výpočtu. Po přenosu zdrojového signálu do sekce se opět zobrazí dialogové okno, jehož parametry jsou délka okénka a velikost kroku jednotlivých iterací.

5.3.1 Volba parametrů

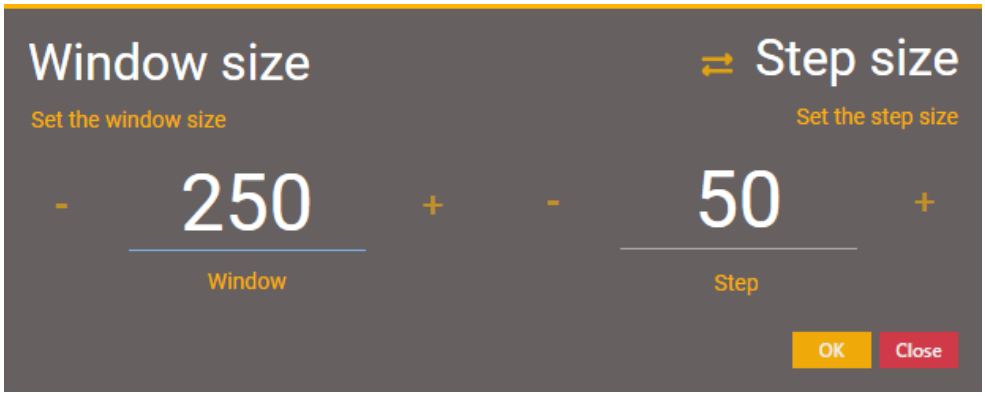
Mezi požadavky na aplikaci byla možnost volby parametrů jednotlivých výpočtů. Mezi nejintuitivnější možnosti volby patřil samotný dialog, který by umožňoval nejen potřebné parametry nastavit, ale zároveň dal uživateli

možnost volby, zda chce daný proces (výpočet) zahájit.



Obrázek 5.6: Dialog nastavení parametrů skrze Overlap

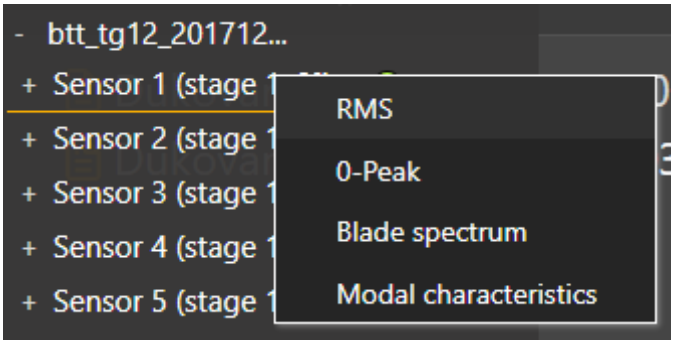
Skrze dialog lze tedy nastavit parametry, jako jsou velikost okénka (Window size) a krok jednotlivých iterací (Step size). K nastavení těchto parametrů může uživatel využít textová pole, do kterých požadovanou hodnotu zapíše, případně může k aktuálně zapsané hodnotě využít funkce inkrementace/dekrementace skrze tlačítka +/-. Hodnota kroku je zde možná nastavit i skrze využití funkcionality přepočtu na procentuální překrytí okének (Overlap). Změna hodnoty overlap vede na okamžitý přepočet hodnoty step. V případě změny hodnoty step je tomu obdobným způsobem.



Obrázek 5.7: Dialog nastavení parametrů skrze Step

5.3.2 Typy výpočtů

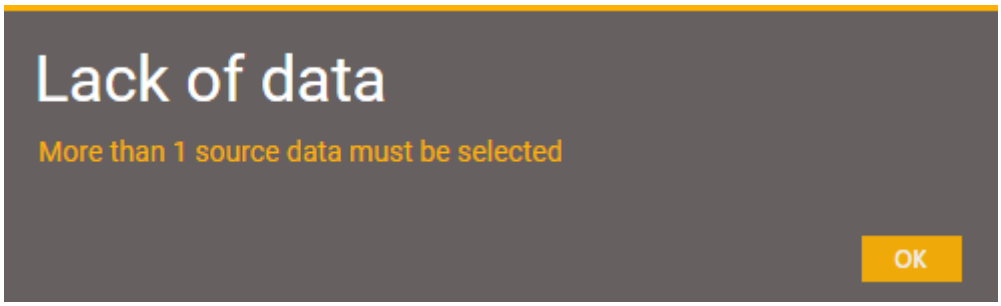
Aplikace podporuje výpočty, jejichž popis je uveden v kapitole 2. Jednotlivé operace výpočtů může uživatel dosáhnout skrze přetažení naimportovaných dat do příslušné sekce (např. sekce RMS) nebo skrze kontextové menu (viz. [5.8])



Obrázek 5.8: Kontextové menu stromové struktury

5.3.3 Možnost více výpočtů

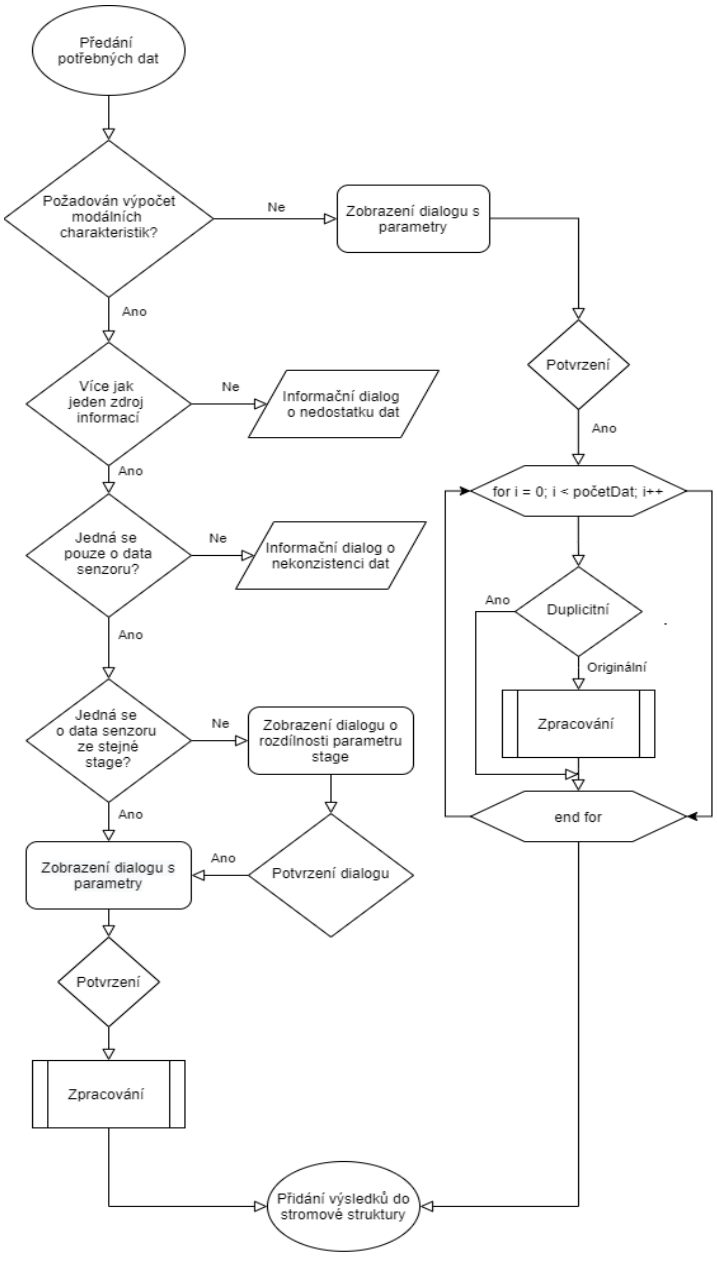
Výpočet modálních charakteristik (Safe a Campbell diagram) vyžadují více zdrojů informací (data z více senzorů). V tomto případě je možné tedy z vybraných dat vypočíst požadované hodnoty pro jejich vizualizaci. Ve zmíněném výpočtu modálních charakteristik je však nutné, aby se data počítala ze stejných dat, co se jejich původu týče (jeden konkrétní soubor měření). Jedná se především o umístění senzorů a z tohoto důvodu je nutné tyto informace před výpočtem zkontrolovat a zároveň zdetekovat, zda bylo předloženo více jak jeden datový balíček senzorových dat. V opačném případě vykreslí informační dialog, který je zobrazen na následujícím obrázku.



Obrázek 5.9: Informační dialog o nedostatku dat

V případě, kdy se nejedná o výpočty modálních charakteristik, by se bez specifické implementace zobrazilo k X výpočtům X dialogových okének,

které by uživatel musel jednotlivě potvrdit a v případě odlišnosti od defaultních hodnot dialogů i jednotlivě nastavit. Z tohoto důvodu byla do aplikace implementována logika, která obsluhuje výpočty pro více jak jeden označený prvek. Pokud uživatel předá data do sekce, pro jehož výpočet postačuje pouze samotná informace, vykreslí se mu jeden dialog, po jehož odsouhlasení aplikace vytvoří separované výpočty s totožným nastavením parametrů ze zobrazeného dialogu. Tento postup je zobrazen na následujícím diagramu.

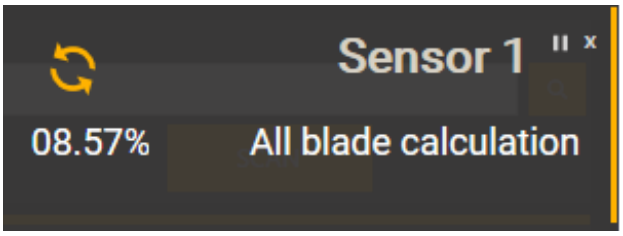


Obrázek 5.10: Postup při požadavku na výpočet

5.4 Procesy

Některá softwarová řešení (např. výpočty v prostředí Matlab), podporující potřebné výpočty, nezobrazují aktuální stav, ve kterém se operace nachází. Uživatel v tomto případě netuší, kdy bude požadovaný výpočet dokončen, což může být pro časově náročné operace, které mohou trvat několik vteřin či minut, nežádoucí. Z toho důvodu byl v rámci aplikace vytvořen mechanismus (dále jen autorita), jehož funkcí je zobrazení aktuálního stavu jednotlivých operací. Je však zapotřebí, aby každá operace svůj stav hlásila objektu, který se stará o jeho vizualizaci.

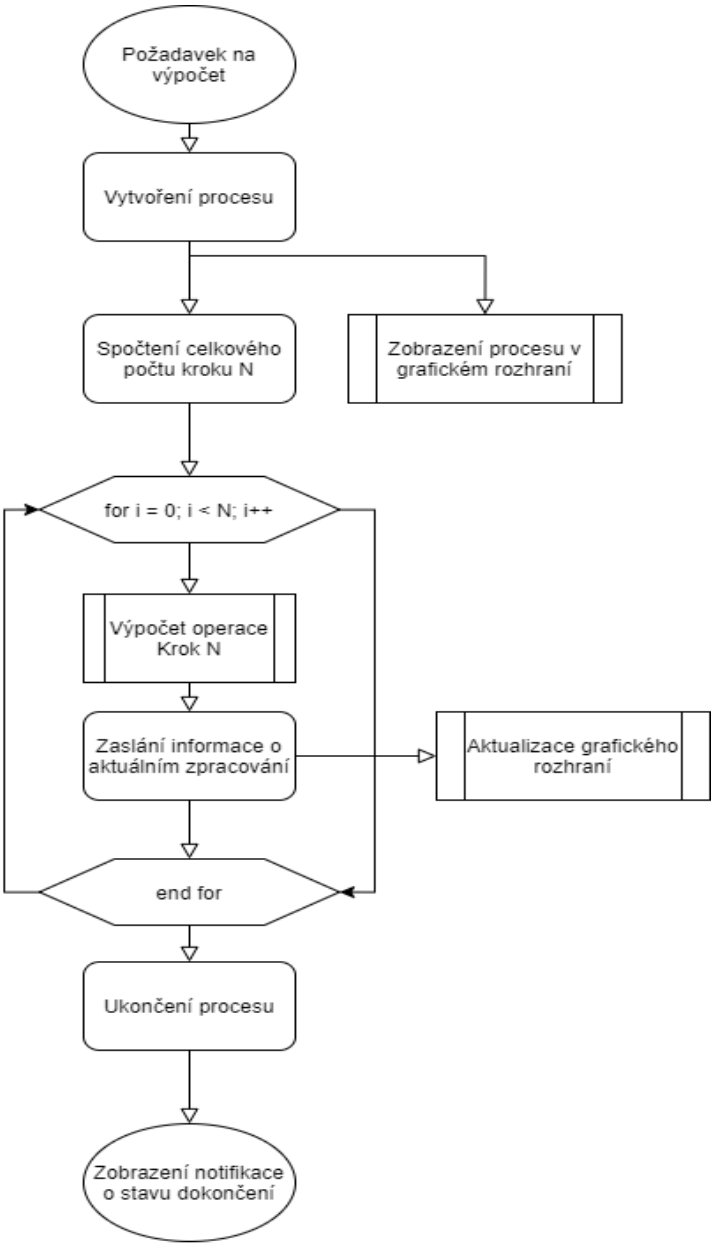
Všem procesům, které byly vytvořeny, je přiřazena notifikační záložka v procesech. Tato záložka informuje uživatele o jejím aktuálním stavu. Pokud je alespoň jeden proces aktivní, je nastaveno upozornění ve formě rotujícího kolečka.



Obrázek 5.11: Proces

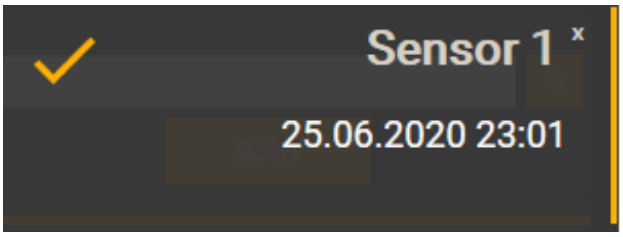
Každý proces tedy funguje takovým způsobem, že si předem vypočte potřebné kroky, které bude absolvovat (část inicializace). Dalším krokem (část výpočtu) je samotný výpočet, který samozřejmě probíhá v cyklu. Na konci každého cyklu předá informaci zmíněné autoritě, jejíž obsahem je krok, ve kterém se aktuálně operace nachází, a celkový počet kroků, který byl vypočten ve fázi inicializace. Všechny tyto operace probíhají ve vlákně, které je určeno pro výpočet.

Pokud autorita detekuje informaci od některého vlákna, pak z údajů (aktuální krok a celkový počet kroků) vypočte procentuální hodnotu na stupnici 0-100%, která prezentuje právě vývoj aktuálního stavu výpočtu.



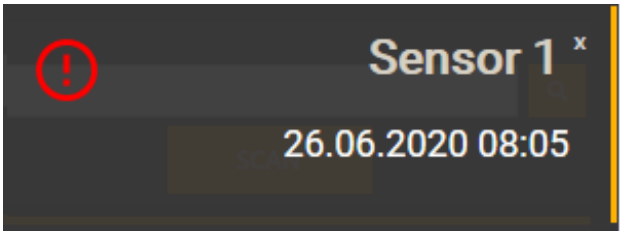
Obrázek 5.12: Aktualizace grafického rozhraní

Po dokončení cyklu se vypočtené hodnoty předají zmíněné autoritě, která má za úkol ukončit vizualizaci procesu a vytvořit notifikaci o dokončení výpočtu.



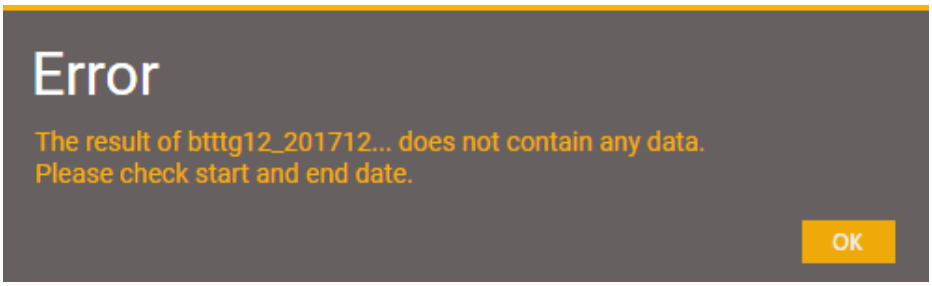
Obrázek 5.13: Úspěšně dokončení procesu - Notifikace

Pokud by však byl z některých důvodů výpočet ukončen, ať už z důvodu nedostatku paměti nebo jiné systémové chyby, bude uživatel informován skrze chybovou notifikaci.

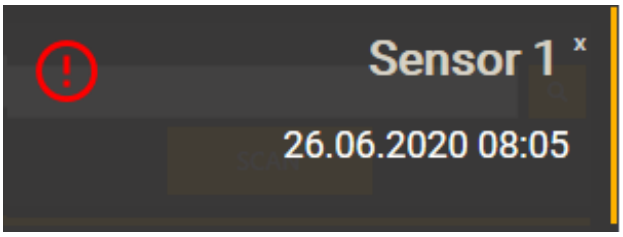


Obrázek 5.14: Neúspěšně dokončení procesu - Notifikace

V případě procesu načítání zdrojového souboru může nastat situace, kdy je definovaný časový interval v hlavičkovém souboru odlišný od dat, které jsou součástí zdrojového souboru. V tomto případě by výsledkem byla prázdná množina dat. Z toho důvodu byla implementována kontrola detekce délky výsledné množiny. Pokud tedy nastane, že výsledná množina je prázdná, pak aplikace upozorní uživatele skrze dialog [5.15] a varovnou notifikaci [5.16].



Obrázek 5.15: Nedostatek dat - dialog

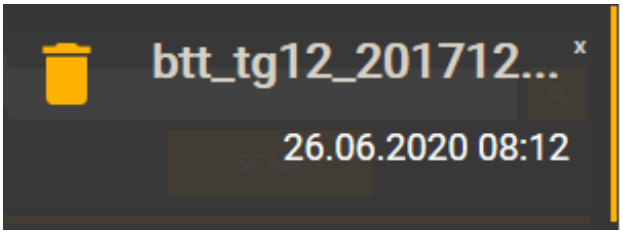


Obrázek 5.16: Varovná notifikace

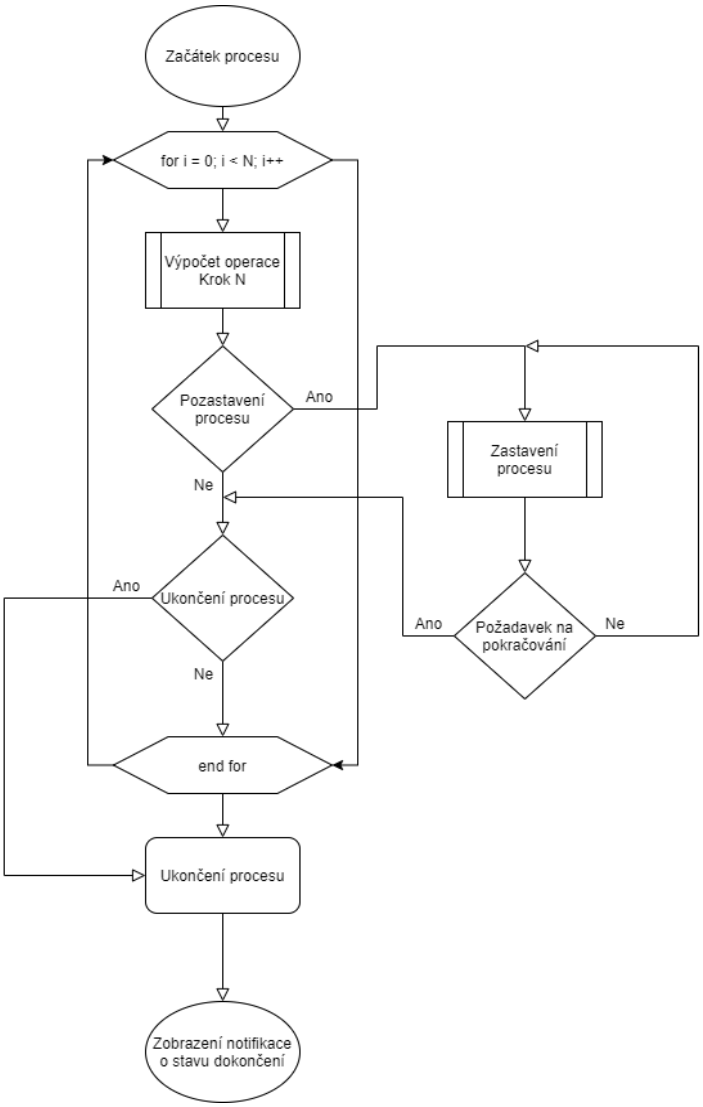
5.4.1 Pozastavení a ukončení procesu

Každý proces, implementovaný v aplikaci, lze skrze jednotlivá tlačítka v panelu procesů ukončit nebo pozastavit. Kontrola těchto požadavků se provádí na konci každého cyklu, jak uvádí následující diagram [5.18]. Pokud je proces ukončen, pak je vygenerovaná varovná notifikace evidující ukončení uživatelem a zpracovaná data jsou zahozena.

V případě, kdy je proces pozastaven skrze dřívější požadavek a následně uživatel odešle požadavek na ukončení procesu, aplikace na tento stav zareaguje zaevidováním požadavku na ukončení a okamžitým odesláním požadavku na pokračování procesu. Tímto způsobem se ukončí vyčkávací smyčka na pozastavení procesu a následně se proces ukončí skrze následující podmínku. Podrobný popis je zobrazen na diagramu [5.18]. V takovém případě se vytvoří notifikace, která eviduje ukončení procesu uživatelským vstupem (viz následující obrázek).



Obrázek 5.17: Notifikace definující ukončení procesu ze strany uživatele



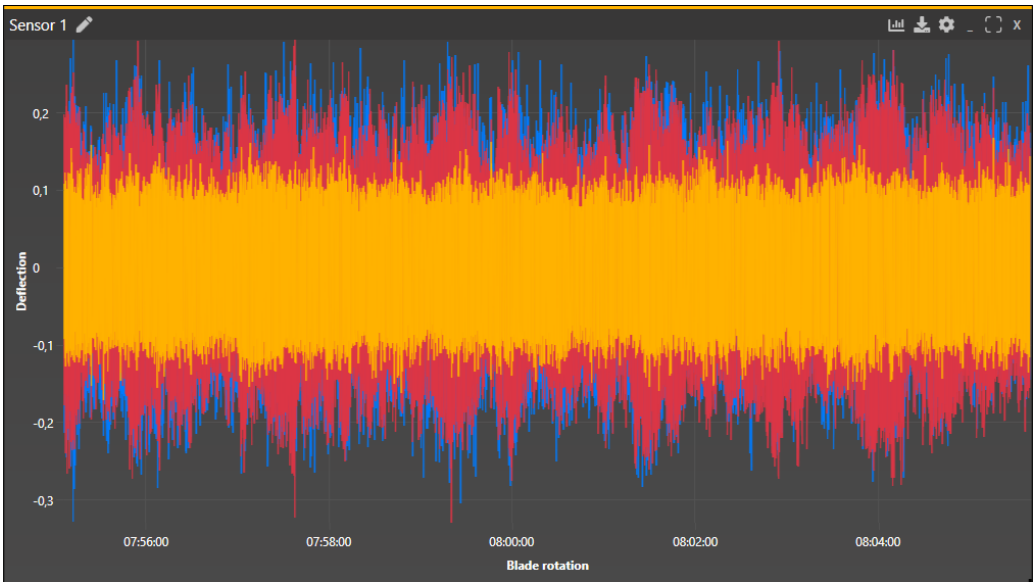
Obrázek 5.18: Kontrola požadavku ukončení a pozastavení procesu

5.5 Grafy

Jednotlivé výpočty lze vizualizovat na kartě Dashboard, kde se grafy dělí na tři typy. Na základě typu jsou uživateli umožněny různé operace a funkce. Krom funkcí, které jsou pro některé grafy specifické, může uživatel graf maximalizovat nebo minimalizovat, zavřít, přejmenovat nebo vyexportovat ve formátu .PNG. Dále je možné graf přesouvat po pracovní ploše Dashboardu [5.18]. Mezi další funkcionality patří funkce zoom (přibližování), drag (přesouvání) a click (kliknutí - výběr). Funkce click je využívána pouze u posledních 2 typů grafu.

5.5.1 XY Graf (Line graph)

Tento objekt slouží pro reprezentaci časového vývoje signálu. Může jím být např. BTT, RMS nebo 0-Peak signál. Do jednoho grafu lze přidat více signálů, které lze od sebe rozlišit barevnou složkou.



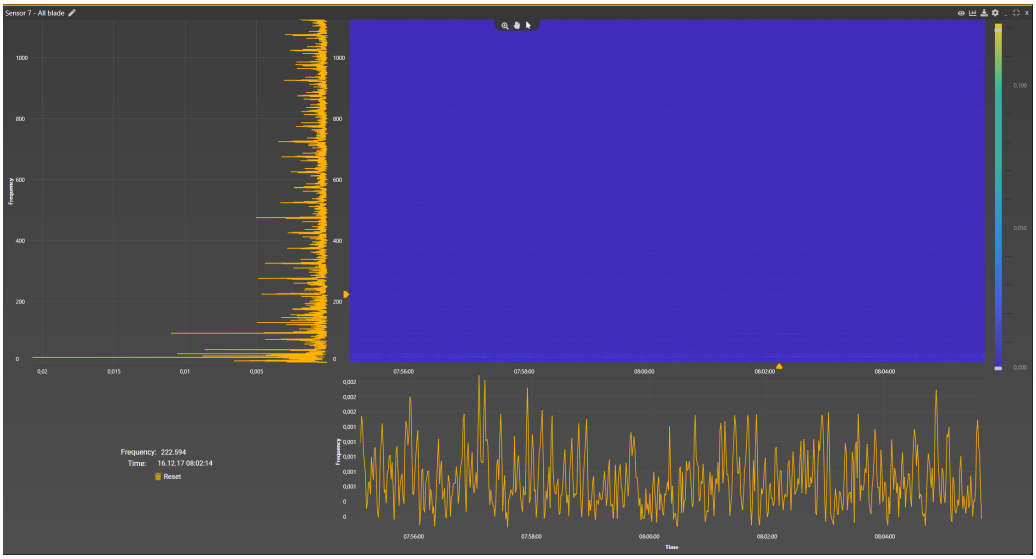
Obrázek 5.19: XY graf

Graf lze vytvořit jako samostatný graf nebo přetažením ze stromové struktury na pracovní plochu, kdy se vytvoří prvek zobrazující vybraná data. Dále je zde možnost přidání signálu do již existujícího grafu. Ve všech těchto případech je před přidáním signálu zahájena kontrola jeho duplicity.

Dále je možné pro jednotlivé signály měnit jejich hodnotu Z, tedy hodnotu, která eviduje, jak je signál zobrazen před či za jiným signálem. Problematiku můžeme demonstrovat na výše uvedeném grafu [5.19], kde jsou obsaženy 3 signály (červený, oranžový, modrý). Z-hodnoty jednotlivých signálů jsou: Oranžový = 1, Červený = 2, Modrý = 3.

5.5.2 Spectrogram

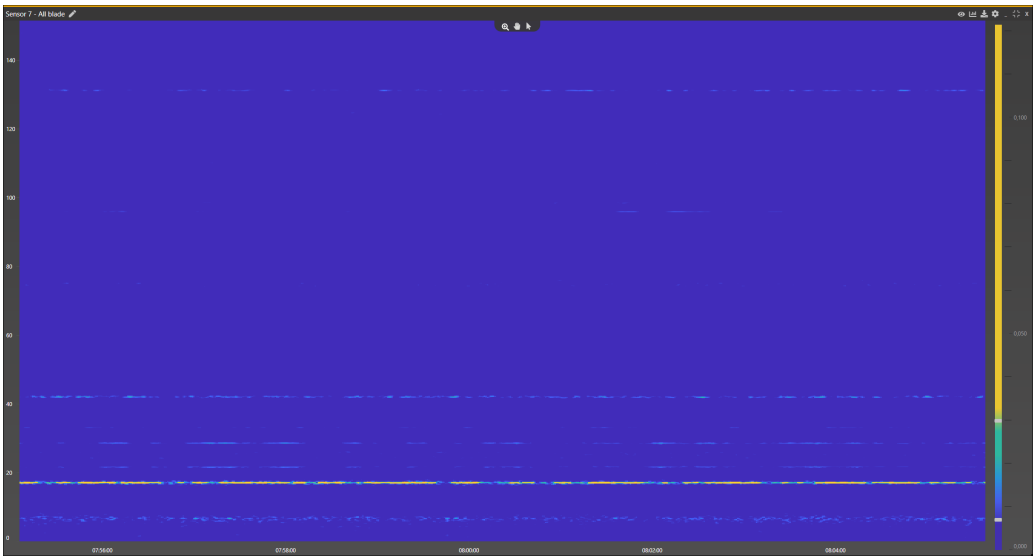
K vizualizaci výpočtů v časo-frekvenční oblasti lze využít objekt Spectrogram, jehož vlastnosti byly popsány v kapitole [2.2.6].



Obrázek 5.20: Spektrogram

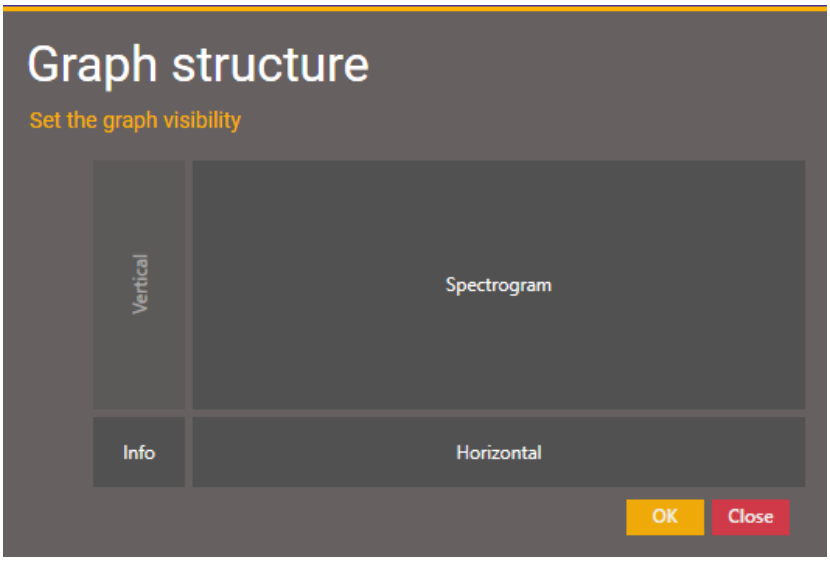
Objekt se skládá celkem z pěti dílčích částí (Spektrogram, Vertikální graf, Horizontální graf, Informační blok, Frekvenční škála). Samotný spektrogram lze přibližovat nebo oddalovat (zoom), případně se v něm lze pohybovat (drag). Další funkcí je možnost kliknutí na vybranou oblast (časovou a frekvenční). Časový vývoj vybrané frekvence se následně zobrazí v horizontálním grafu pod spektrogramem a frekvenční spektrum vybraného času ve vertikálním grafu, který je umístěn vlevo od spektrogramu. Vybrané hodnoty se zobrazí v informačním bloku.

Další možností je volba barevné škály frekvenčního spektra. Uživatel má tedy možnost nastavit různé barevné složky pro vybraný úsek frekvenčních amplitud. Tyto změny lze jednoduše nastavit na původní hodnoty. Funkcionalita je zobrazena na následujícím grafu. Díky změně barevné škály můžeme na zmíněném grafu lépe detekovat větší frekvenční amplitudy, které jsou v oblasti 17 Hz.



Obrázek 5.21: Změna škály spektrogramu

Mezi další funkcionalitu tohoto typu grafu patří výběr zobrazení zmíněných částí objektu. Volba se provádí skrze následující dialog.

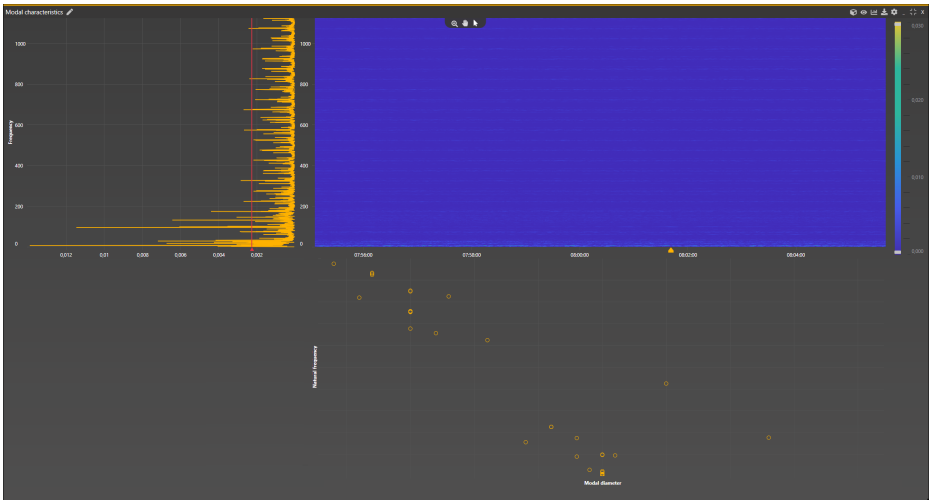


Obrázek 5.22: Dialog výběru sekcí

Je tedy možné například zobrazit pouze horizontální graf (časový průběh frekvence) spolu se spektrogramem. Tohoto stavu docílíme odstraněním sekce Vertical a Info. Ostatní stavy zobrazení se provádí obdobným způsobem.

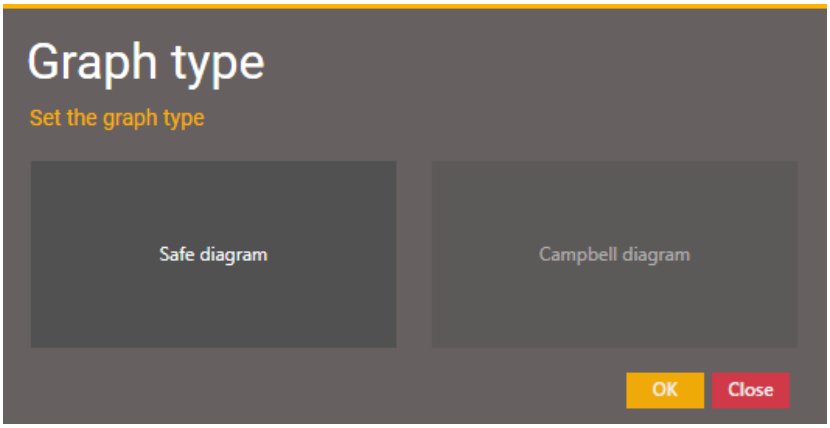
5.5.3 Graf modálních charakteristik

Objekt, který slouží k zobrazení Safe a Campbellova diagramu, je svou strukturou obdobný objektu Spectrogram, který byl popsán v předchozí kapitole. Změna oproti předchozímu grafu je v možnosti zobrazení uzlových průměrů v horizontálním grafu.



Obrázek 5.23: Graf modálních charakteristik - Safe diagram

Na graf lze aplikovat pouze vypočtená data v podobě modálních charakteristik. Uživatel si poté může zobrazit vypočtená data v podobě Campbellova nebo Safe diagramu. Volba je opět volena skrze dialogové okno (viz. [5.24])



Obrázek 5.24: Volba typu grafu

Výsledná data uzlových průměrů se zobrazí v horizontálním grafu. Pro Safe diagram je možnost uživatelského vstupu ve formě mezního prahu volby frekvenční amplitudy (viz. vertikální graf na Obrázku [5.23])

6 Optimalizace

Aplikace byla optimalizována za účelem zrychlení jednotlivých výpočtů a redukce jejich paměťových nároků. Při optimalizaci byly brány v potaz omezené hardwarové zdroje uživatelského zařízení (PC). Byl kladen důraz na minimální paměťovou náročnost a maximalizace výpočetního výkonu.

6.1 Optimalizace paměťových nároků

Před úvodem do problematiky optimalizace paměťových nároků uveďme význam Garbage collectoru. Tento pojem můžeme definovat jako samostatně běžící program, který běží paralelně s naší aplikací v samostatném vlákně. Jeho úkolem je kontrolovat zda na data uložená v operační paměti RAM není vedena některá reference. V případě, že zmíněná data nejsou žádným objektem referována (žádný jiný objekt v paměti se na ně neodkazuje), jsou automaticky z paměti uvolněna. [19]

Optimalizace paměťových nároků byla nutná v oblasti zpracování zdrojového signálu. Při prvotní implementaci bylo zvoleno uložení jednotlivých signálů senzorů a lopatek do samostatných objektů, což vedlo na redundantní informaci v operační paměti. Pro optimalizaci tohoto řešení byla zvolena forma uložení zpracovaných dat senzorů ve formě vícerozměrného pole (matice) M x N, kde M značí počet detekovaných průletů a N počet lopatek. Pro případ využití vizualizace senzoru či lopatky je však nutné, aby byly předloženy ve formě jednorozměrného pole (vektoru) o velikosti L, která značí velikost výsledného signálu. V případě jedné lopatky se jedná o hodnotu L = M, jelikož se jedná o jeden sloupec vícerozměrného pole. Pokud bychom chtěli vizualizovat data z celého senzoru, tedy data jednotlivých lopatek, které senzor detekoval. Byla by délka výsledného signálu L = M x N.

V takovém případě se vždy v operační paměti vytvoří nový objekt o velikosti L, který by v případě vícenásobného využití (např. v případě vizualizace dvou grafů odkazující se na stejný objekt) duplikoval již existující prvek a paměťová náročnost by tedy s N odkazy N-krát narůstala. Z toho důvodu byl využit fakt, že výsledná data již nejsou po jejich zpracování žádným způsobem modifikována a v případě vícenásobné reference se může odkazovat pouze na jeden vytvořený objekt.

Pro demonstraci zvolené metody je uveden výčet hodnot samostatné lopatky senzoru.

```
private IDataSeries selectedData = null;
public override IDataSeries Data
{
   get
   {
     if (selectedData == null)
     {
        selectedData = new XyDataSeries<DateTime, float>
        (SignalDate.ToArray(), DataMatrix.Row(N).ToArray());
     }
     return selectedData;
   }
}
```

Kód 6.1: Výběr signálu lopatky z matice dat

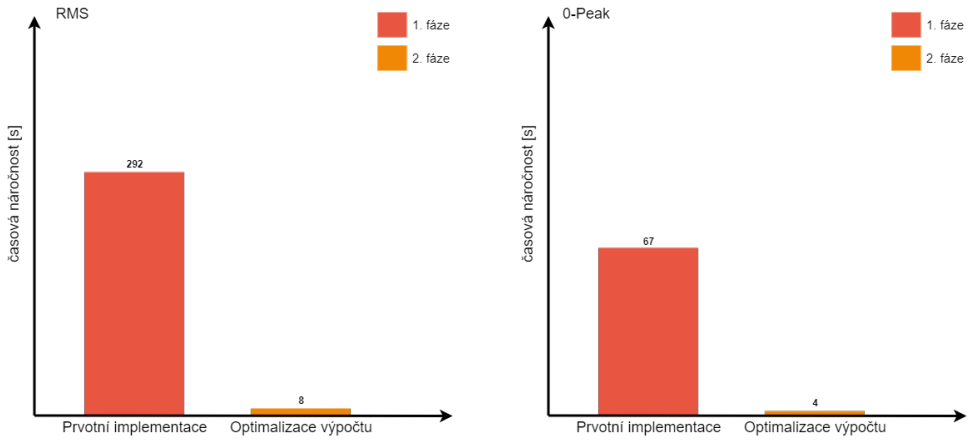
Objekt lopatky zpočátku jeho vytvoření neodkazuje na žádný vytvořený objekt, pouze obsahuje interní referenci na již vytvořenou matici (reference zmíněné matice je uložena v každém objektu signálu lopatky a k nim přidruženého senzoru). V případě dotazu na data se zkontroluje, zda objekt privátně obsahuje již vytvořená požadovaná data, a v případě nesplnění této podmínky data vytvoří a předá na ně referenci. V případě dalšího dotazu se předá rovnou reference. Tento postup je vhodný v situacích, kdy uživatel využívá tato data víceúčelově (např. více vizualizací, výpočty), ale i v situacích, kdy data vůbec nevyužívá, jelikož není nadbytečně vytvořen datový objekt, který by zabíral místo v operační paměti.

6.2 Optimalizace výkonu

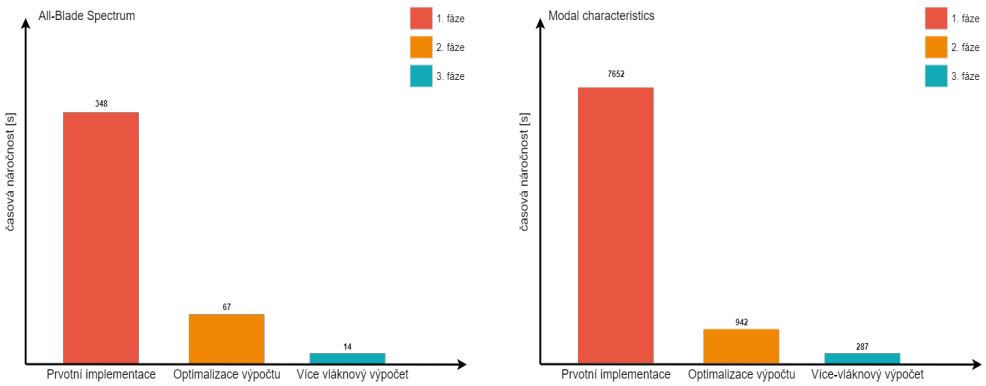
Každý algoritmus výpočtu byl optimalizován pro minimalizaci časových a paměťových nároků. Vývoj výpočetních operací můžeme zahrnout do tří fází, přičemž při každé z nich bylo dosaženo správných výsledků.

1. fáze - Získání správných výpočtů
2. fáze - Optimalizace výpočtů - jednovláknové operace
3. fáze - Nasazení vícevláknových operací

Ukončení první fáze bylo při dosažení správných výsledků (bez ohledu na časovou náročnost operace). Při druhé fázi se dbalo na minimalizaci počtu kroků, které vedou k výsledku, avšak v této fázi se pracovalo s jednovláknovými výpočty, aby výsledný čas nebyl ovlivněn výkonem zařízení, na kterém se výpočty provádí (porovnání časů jednotlivých operací, které jsou následně demonstrovány, vychází z výsledku z jednoho zařízení). Poslední fáze byla na základě implementace vícevláknového zpracování výpočtů. Časové náročnosti optimalizací jsou uvedeny v následujících grafech [6.1, 6.2]. Grafy demonstrují průměrné hodnoty pěti výpočtů ze zdrojového signálu, který trval 2 hodiny a 57 minut (10 620 vteřin).



Obrázek 6.1: Časové náročnosti výpočtů



Obrázek 6.2: Časové náročnosti výpočtů

U výše uvedených grafů je možné vypozorovat nasazení vícevláknových výpočtů pouze u dvou posledních grafů [6.2]. Je tomu tak z důvodu nízké časové náročnosti i v případě jednovláknového výpočtu.

7 Závěr

Práce se věnovala problematice měření lopatkových vibrací. V první kapitole byly popsány možné stavy lopatek a lopatkových disků, které nejsou z hlediska čerpání zbytkové životnosti vhodné. Byly blíže popsány pojmy, jako jsou vlastní tvary lopatek a uzlový průměr lopatkového disku. Na základě těchto poznatků byla popsána možná měření signálu těchto vibrací a jeho následné zpracování.

Ve druhé kapitole, které se tato práce věnovala, byly uvedeny jednotlivé algoritmy zpracování a to jak v časové, tak ve frekvenční oblasti. Dále byl popsán Campbellův a Safe diagram a jeho možná využití.

Třetí část se věnuje popisu použitých technologií, které vedly k hlavní části této diplomové práce - vytvoření softwaru pro zpracování signálu vibrací lopatek. V následujících dvou kapitolách jsou blíže přiblíženy funkce této aplikace a její grafický návrh.

Poslední kapitola se zabývá optimalizací výpočtů a jejich časovou náročností. Jsou zde uvedeny jednotlivé časové náročnosti operací při určitých fází jejich implementace. Algoritmy byly v konečné fázi upraveny do takové podoby, že jejich časová náročnost je ve všech případech pouze zlomkem předchozích náročností.

Výsledná aplikace obsahuje sadu uživatelských funkcí, které usnadňují práci s naměřenými signály a jsou v některých případech lepší oproti stávajícím řešením, které jsou používány (např. zlepšené vykreslení grafů oproti prostředí Matlab nebo vícevláknové výpočty). Avšak je zde určitě prostor k jejich vylepšení či modifikaci. Zároveň by bylo v budoucnu vhodné doplnit funkce, které nebyly v rámci diplomové práce implementovány. Může se tak jednat o funkcionality z jiných softwarových řešení (jejich totožná nebo modifikovaná verze) nebo funkcionality, které by cílová skupina uživatelů, pro který je tento software určen, požadovala. V budoucnu bude brána v potaz odezva od těchto uživatelů, na jejíž základě se bude software dále vyvíjet.

Při implementaci byla zahrnuta možnost, že zdrojová data mohou v budoucnu pocházet i z jiného zdroje měření (např. data od jiných výrobců) a mohou mít tedy i jinou strukturu. Z tohoto důvodu byla aplikace navržena

takovým způsobem, aby bylo možné jednoduše implementovat další funkcionality v podobě parsování dat nebo jejich transformace bez nutnosti měnit stávající řešení její struktury.

Literatura

[1] TŮMA, Jiří. *Zpracování signálů získaných z mechanických systémů užitím FFT.* Praha: Sdělovací technika, 1997. ISBN 80-901936-1-7.

[2] Parní turbína – Cez [online].
Plzeň: Západočeská univerzita v Plzni

[3] Století páry pokračuje. Ani to nevíte | VTM.cz. [online]. Dostupné z: *http://vtm.e15.cz/stoleti-pary-pokracuje-ani-to-nevite*

[4] KOUTSKÝ, Jaroslav. *Materiály tepelně-energetických zařízení. 1. vyd.* Plzeň: Západočeská univerzita v Plzni

[5] Tenzometr – Wikipedie. [online].
Dostupné z: https://cs.wikipedia.org/wiki/Tenzometr

[6] *Geometrie a materiály lopatkových strojů. Články – Transformační technologie* [online]. Copyright ©Jiří Škorpík, [cit. 23.05.2020].
Dostupné z: *https://www.transformacni-technologie.cz/15.html*

[7] FFT and Spectral Leakage – GaussianWaves. GaussianWaves – Signal Processing Simplified [online]. Copyright © 2020 GaussianWaves [cit. 27.05.2020]. Dostupné z: *https://www.gaussianwaves.com/2011/01/fft-and-spectral-leakage-2/*

[8] FIEDLER, Jan. *Parní turbíny: návrh a výpočet. 1. vyd.* Brno: Akademické nakladatelství CERM, 2004, 66 s. ISBN 80-214-2777-9.

[9] Zimmermann, Pavel. *Identifikace poměrného útlumu lopatek z experimentálně naměřených dat.* Plzeň, 2012. Diplomová práce. Západočeská univerzita v Plzni.

[10] Šuma, Jiří. *Modernizace parní turbíny.* Plzeň, 2015. Bakalářská práce. Západočeská univerzita v Plzni.

[11] Vašíček, Vojtěch. *Analýza vibrací lopatek turbín ze signálů relativního rotorového chvění.* Plzeň, 2015. Diplomová práce. Západočeská univerzita v Plzni

[12] Drahý, Jan. *Modální analýza turbínového kola pro letecký motor.* Brno, 2010. Diplomová práce. Vysoké učení technické v Brně, Fakulta strojního inženýrství

[13] Hamming Window - an overview | ScienceDirect Topics. ScienceDirect.com | Science, health and medical journals, full text articles and books. [online]. Copyright © 2020 Elsevier B.V. or its licensors or contributors. [cit. 27.05.2020].
Dostupné z: *https://www.sciencedirect.com/topics/engineering/hamming-window*

[14] Hanning Window - an overview | ScienceDirect Topics. ScienceDirect.com | Science, health and medical journals, full text articles and books. [online]. Copyright © 2020 Elsevier B.V. or its licensors or contributors. [cit. 27.05.2020].
Dostupné z: *https://www.sciencedirect.com/topics/engineering/hanning-window*

[15] JAKL, Jan. *Výzkum a vývoj metod a algoritmů pro detekci a lokalizaci rubbingu na parních turbínách.* Plzeň, 2014. Disertační práce. ZČU.

[16] SciChart | WPF Charts, iOS Charts, Android Charts and Xamarin Charts. SciChart | WPF Charts, iOS Charts, Android Charts and Xamarin Charts [online].
Dostupné z: *https://www.scichart.com/*

[17] Threads and threading | Microsoft Docs. [online]. Copyright © Microsoft 2020 [cit. 28.05.2020]. *Dostupné z: https://docs.microsoft.com/en-us/dotnet/standard/threading/threads-and-threading*

[18] Json.NET - Newtonsoft. [online]. Copyright © 2020 Newtonsoft [cit. 28.05.2020]. Dostupné z: *https://www.newtonsoft.com/json*

[19] Lekce 4 - Odkazy na objekty, jejich kopírování a garbage collector. ITnetwork.cz [online]. Copyright © 2020 itnetwork.cz [cit. 26.06.2020]. Dostupné z: *https://www.itnetwork.cz/dart/oop/odkazy-na-objekty-jejich-kopirovani-a-garbage-collector*

[20] What is NuGet and what does it do? | Microsoft Docs. [online]. Copyright © Microsoft 2020 [cit. 28.06.2020]. Dostupné z: *https://docs.microsoft.com/en-gb/nuget/what-is-nuget*