Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Hálózati Rendszerek és Szolgáltatások Tanszék

Anomáliák automatikus felismerése rázóasztalos rezgésvizsgálatokban

Diplomaterv

Készítette
Kostyál Domonkos Marcell

Konzulens
dr. Rucz Péter
Farkas Máté

2024. június 2.

Tartalomjegyzék

HALLGATÓI NYILATKOZAT

Alulírott *Kostyál Domonkos Marcell*, szigorló hallgató kijelentem, hogy ezt a szakdolgozatot meg nem engedett segítség nélkül, saját magam készítettem, csak a megadott forrásokat (szakirodalom, eszközök stb.) használtam fel. Minden olyan részt, melyet szó szerint, vagy azonos értelemben, de átfogalmazva más forrásból átvettem, egyértelműen, a forrás megadásával megjelöltem.

Hozzájárulok, hogy a jelen munkám alapadatait (szerző(k), cím, angol és magyar nyelvű tartalmi kivonat, készítés éve, konzulens(ek) neve) a BME VIK nyilvánosan hozzáférhető elektronikus formában, a munka teljes szövegét pedig az egyetem belső hálózatán keresztül (vagy autentikált felhasználók számára) közzétegye. Kijelentem, hogy a benyújtott munka és annak elektronikus verziója megegyezik. Dékáni engedéllyel titkosított diplomatervek esetén a dolgozat szövege csak 3 év eltelte után válik hozzáférhetővé.

Budapest, 2024. június 2.

Kostyál Domonkos Marcell
hallgató

Kivonat

Az NVH (Noise, Vibration and Harshness) területén gyakori, hogy egy adott szerkezet rezonanciafrekvenciáit és a megfelelő rezgésmintáit (módusalakokat) kell meghatározni. Ehhez kísérleti móduselemzésnek (Experimental Modal Analysis, EMA) nevezett mérési eljárást alkalmaznak, amelynek előkészítési és mérési ideje általában több órát vesz igénybe. Az eljárás a gerjesztés (kalapácsütés vagy rázógerjesztés) és a válaszjelek (tipikusan sebesség vagy gyorsulás) ismételt méréséből áll. Az EMA során alapvető fontosságú a pontos eredmények, jelen esetben átviteli függvények (Frequency Response Function, FRF) mérése. Még ha mindent előzetesen ellenőriznek is, a mérések során előfordulhatnak anomáliák, amelyek esetenként az egész mérést használhatatlanná tehetik.

A mért egyedi FRF-ek minőségének meghatározásához a legtöbb kereskedelmi szoftverben a koherenciafüggvényt vizuális ellenőrzés céljából megjelenítik. Nincs azonban pontos módja annak eldöntésére, hogy a koherenciafüggvény egyszerű megtekintésével eldönthető-e, hogy a mérés hibás. Ráadásul a döntések néha még a képzett technikusok számára is zavarosak lehetnek.

A jelen diplomamunka célja, hogy automatikusan felismerje az akusztikai rezgésmérések során bekövetkező viselkedésváltozást, és képes legyen eldönteni, hogy mikor hibás a mérési elrendezés, különösen, ha érintkezési probléma lép fel a csavarok meglazulása, a ragasztó meggyengülése, kábelezési problémák, törés vagy bármilyen nem szándékos változás miatt.

Az EMA mérésekhez, egy jól bevált környezet, ha egy rázóasztal és pásztázó lézer komnációjával mérjük az átvitelt. Ezt egészítettem ki egy automatizált monitorozórendszerrel, melyet egy tesztelrendezésen vizsgáltam.

Az adott elrendezés ellenőrzéséhez párhuzamos méréseket kell végezni. Az FRF-ek kiszámításával az egyes részmérések alatt, valamint a kezdeti és a ténylegesen mért FRF-ek összehasonlításával láthatóvá válik, hogyan változik a rendszer. Több értékelési módszert (koherencia, spektrális laposság, randomized dependence coefficient, FRAC) hasonlítottam össze, és a FRAC (Frequency Response Assurance Criterion) mutatta a legjobb eredményeket a fizikai elrendezésben bekövetkező eltérések kimutatására, mind a modális viselkedésben, mind a linearitásban.

Végül az utolsó fejezetben bemutatom egy olyan szoftver megvalósítását, amely a rázóasztalos rezgésmérések során felmerülő anomáliák monitorozására alkalmazható.

Abstract

In the field of NVH (Noise, Vibration and Harshness) it is common to determine resonance frequencies and the corresponding vibration patterns (mode shapes) of a given structure of interest. For this, a measurement procedure called, Experimental Modal Analysis (EMA) is used, with preparation and measurement time typically taking several hours. The procedure consists of repeated measurements of excitation (a hammer hit or a shaker excitation) and response signals (vibration velocity, acceleration, ...). In EMA, it is crucial to acquire accurate results (FRFs). Even if everything is checked beforehand, anomalies can occur during the measurements, in some cases rendering the whole measurement unusable.

To determine the quality of the measured individual FRFs, in most commercial software the coherence function is displayed for visual inspection purposes. However, there is no precise way of deciding whether the measurement is faulty by simply looking at the coherence function. Furthermore, decisions can sometimes be confusing even to trained technicians.

The objective of the present thesis is to automatically recognize any change in behavior during vibration measurements and be able to decide when the setup is faulty, especially if a contact problem occurs due to loosening of a bolt, weakening of the glue, cabling problems, breakage, or any unintended change of the setup.

For EMA measurements, a well-established environment is to measure the FRFs using a combination of a shaker exiter and a scanning laser vibrometer. I complemented this with an automated monitoring system, which I tested on a test setup.

Parallel measurements are needed to monitor the integrity of the setup. By calculating the FRFs during each period and comparing the initial and actual measured FRFs, it is possible to see how the system is changing. I compared multiple evaluation methods (coherence, spectral flatness, randomized dependence coefficient, FRAC), and FRAC (Frequency Response Assurance Criterion) has shown the best results to detect anomalies in the physical setup, both in modal behavior and linearity.

In the last chapter, I show an implementation of software that can be used to monitor such problems.

1. fejezet

Bevezetés

Napjainkban az autóipar gyökeres változásokon megy keresztül az elektromos hajtásláncok térnyerése révén. Ez a paradigmaváltás átalakítja a járműtervezést, új kihívásokat és lehetőségeket teremtve. A megjelenő új technológiáknak köszönhetően az elektromos hajtású járművek egyre jobban előtérbe kerülnek, ezáltal a tervezési folyamatok is átalakulnak. Az elektromos autók egyik előnye a hagyományos belső égésű motoros járművekkel szemben, hogy kevesebb zaj keletkezik működésük közben [5]. Ennek köszönhetően az akusztikai tervezés egyre nagyobb szerepet kap, mivel a különböző kisebb elektromos motorok is képesek gerjeszteni az autó további alkatrészeit és karosszériáját, ezzel felerősítve a kisebb mozgásokból adódó zajokat. Ezek nemcsak kellemetlenséget okozhatnak a vezetőnek és az utasoknak, hanem hatással lehetnek a jármű teljesítményére is.

A rezgésakusztikai elemzések és modellezések segítenek a probléma megoldásában. Az ilyen vizsgálatokkal lehetőség nyílik a rezgések forrásának azonosítására és azok csökkentésére. A szimulációk sosem lesznek egy az egyben megfeleltethetőek a valósággal, emiatt ezeket validálni kell a gyakorlatban, és utána frissíteni a szimulált számítógépes modellt a méréseknek megfelelően. Erre különböző módszerek léteznek, amik legfőképp a mérendő tárgy fizikai tulajdonságaitól és a végleges használati környezetétől függenek [17]. Ahhoz, hogy frissíthessék a modellt, kísérleti móduselemzést kell végezni az eszközön, melyhez az adott tárgyat a vizsgált frekvenciatartományban gerjeszteni kell és különböző pontjain elmozdulás-, sebesség- és gyorsulásszenzorokkal mérni a válaszait.

Ezek a mérések igen fontosak, így az is fontos, hogy megbízhassunk az eredményeikben. Sajnos sokszor a keletkezett hiba már a kiértékelés során tűnik fel, amikor a mérési elrendezést már szétszerelték, és sok esetben a mért tárgy már nem is áll a rendelkezésünkre (például élettartam vizsgálatoknál). Ha mégis rendelkezésünkre áll még a mért tárgy, akkor a technikusnak újra össze kell állítania a mérést és elvégeznie azt. Összehasonlíthatóság szempontjából egy szekvencia részújramérése viszont nem előnyös, ugyanis bizonyos paraméterei az elrendezésnek nem reprodukálhatóak, és ha minimálisan is, de más tulajdonságokkal fog rendelkezni (például szenzorok ragasztása esetén a ragasztóanyag mennyisége). Emiatt a későbbi újramérés, teljes visszaszereléssel kerülendő. Ezen okok miatt ideális egy monitorozó rendszer létrehozása, mely azonnal visszajelzést ad a mérést végző technikusnak, ha a mérés során a felépített elrendezésben nemkívánatos változás történik.

A diplomamunka célja olyan monitorozórendszer megvalósítása rázóasztalos rezgésakusztikai vizsgálatokhoz, mely biztosítja a mérés validációjának lehetőségét kiértékelőmérnök bevonása nélkül is.

1.1. Elvárások a szoftver funkcionalitása felé

A probléma feltérképezésének megkezdése előtt úgy gondoltam, hogy érdemes listát készíteni az elkészítendő szoftver elvárásairól. A projektet az egyik munkatársam ajánlotta, így közösen végigjártuk, mi az, ami elengedhetetlen a funkcionalitás szempontjából. Ezt a lépést nagyobb projektek esetén nagyon fontos elvégezni, hogy a megrendelő tudja, milyen eredményekre számítson. Emiatt is jó gyakorlatnak gondoltam, hogy néhány konkrétabb elemet lefektessek a legelején.

Az alapvető feladat a mérések során fellépő különféle anomáliák detektálása, emiatt belátható, hogy ezt egyfajta döntéshozatali algoritmus segítségével fogjuk tudni végrehajtani. Az ilyen típusú módszerek esetén nagyon jó kiértékelési eljárás, ha a végső szoftvert valós adatokon teszteljük, amikről már tudjuk, hogy hova kellene sorolni őket (ezeket felcímkézett adatoknak is nevezik), és ezután az algoritmus eredményeit össze lehet vetni a valós eredményekkel. Ezekből számolható az úgynevezett pontosság (az eltalált eredmények osztva az összes bevitt adat számával), ami kellően nagy és diverz tesztadathalmaz esetén reprezentatív értéket ad az algoritmus minőségéről [15]. Bizonyos esetekben viszont érdemes jobban megkülönböztetni a predikciók eredményeit, amit négy típusra bonthatunk szét [7]:

- **Igaz pozitív** (True Positive, TP): Annak az arányát adja meg, hogy az összes pozitívan címkézett eredményből mennyit talált el.
- **Hamis pozitív** (False Positive, FP): Annak az arányát adja meg, hogy az összes negatívan címkézett eredményből mennyit döntött pozitívra.
- **Igaz negatív** (True Negative, TN): Annak az arányát adja meg, hogy az összes negatívan címkézett eredményből mennyit talált el.
- **Hamis negatív** (False Negative, FN): Annak az arányát adja meg, hogy az összes pozitívan címkézett eredményből mennyit döntött negatívra.

Látható, hogy ez azért hasznos, mert a döntési helyzetekben nem mindegy, hogy a negatív vagy a pozitív döntés volt helytelen az algoritmus szempontjából. Például, egy önvezető autó esetében, ha az algoritmusnak azt kell eldöntenie, hogy sávot váltson-e, érdemes úgy optimalizálni, hogy a hamis pozitív értéke minél kisebb legyen (tehát ne váltson sávot, ha valójában ez nem indokolt), de a hamis negatív nem annyira kritikus, mert akkor csak egyszerűen a biztonságos sávjában marad. Az én problémámra ez nem kiemelkedően fontos, mivel nincsen nagyobb tétje a hamis detekciónak. Ha belegondolunk, a cél, hogy a lehető legjobban detektálni tudjuk a hibákat, emiatt, ha esetleg úgy érzékel hibát a rendszer, hogy valójában nincs, az nem akkora probléma mintha a valós hibát nem érzékelné és emiatt a későbbiekben teljes újramérést kellene végezni. Összességében elég, ha minél pontosabb a megtervezett rendszer, tehát a találati aránya, azaz $Accuracy = \frac{N_{TP}+N_{TN}}{N_{data}}$ értéke minél nagyobb.

Végül az alábbi elvárásokat fektettem le a szoftver funkcionalitását illetőleg:

- Valós idejű eredmények elérhetők legyenek a mérés során, így az anomáliák mielőbbi észlelése lehetővé válik, és gyorsabban lehet javítani a hibákat.
- Lehetőleg ne legyen nagy számításkapacitás és tárhely igényes, hogy a program akár egyszerűbb számítógépeken is futhasson.
- Legalább 90%-os pontossággal ismerje fel a rázóasztalos mérések során a keletkező hibát. A hoibaarány lefőképpen irányelvnek tekinthető a tervezés közben és majd csak a valós méréseken tesztelve lehet számolni.

- Egyszerű kezelőfelület álljon rendelkezésére a felhasználónak, ahol el tudja indítani a detekciót, és a felület visszajelzést ad, ha eltérést észlel.
- Az anomália kiértékeléséhez szükséges méréseket ne kelljen tárolni, csak az éppen aktuális eredmények legyenek elérhetők a helyi számítógépen.

1.2. Piackutatás

Az alapvető célok lefektetése után a piac feltérképezésével folytattam a munkám. Alapvetően ez egy állapotfigyelő rendszer, aminek a gyártósorokon való alkalmazása minőségbiztosítás szempontjából már sok éve bevett szokás [2]. Az ilyen felhasználási esetekben viszont nem rezgésakusztikai vizsgálatokat végeznek, hanem geometriai pontosságot vizsgálnak többnyire. Másik felhasználási területe az állapotfigyelő rendszereknek az ipari gépek különböző alkatrészei elfáradásának, nem megfelelő működésének vizsgálata működés közben (condition monitoring [27]). Ebben az esetben sokszor használnak akusztikai vizsgálatokat, viszont ezek többnyire nem valósidejűek, általában néhány hónaponként egy mérnök végez egy-egy mérést, és a trendekből következtet például egy csapágy meghibásodására. Az ilyen típusú szoftverek vagy nagyon robusztusak és komplexitásukból adódóan nagyobb tesztpadrendszerek monitorozására használhatóak, vagy pedig túl specifikusak, ami miatt nem alkalmasak a dolgozatban tárgyalt probléma megoldására.

2. fejezet

A mérési környezet

Ebben a fejezetben a vizsgált méréstípusokat mutatom be, majd egy általános elrendezést is részletezek, aminek a monitorozási lehetőségeit és nehézségeit járom körbe.

Az rezgésakusztikai méréseknek az autóiparban jelentős szerepük van mind a tervezés során, mind a szimulációk validációja során, hogy garantálhassák az adott eszköz megfelelő minőségét és megbízhatóságát. Alapvetően két felhasználási területet különböztethetünk meg ahol ezek a mérések kiemelkedő fontosságúak: a kísérleti móduselemzéshez és a terheléses tesztek elvégzéséhez.

2.1. Kísérleti móduselemzés

A kísérleti móduselemzés fő célja, hogy feltérképezzük egy adott test viselkedését és szimulációs modellünket javíthassuk [13]. Emellett a kiszámított modell segítségével megjósolhatjuk későbbi beavatkozások várható hatását is. A metódus alapvetően öt lépésből épül fel:

1. Az analízis a mérés összeállításával kezdődik. Ez magába foglalja a mérendő tárgy felfogatásának módját, szenzorok felhelyezését és kalibrálását és mérőberendezés összeállítását, konfigurálását.
2. A második lépés az adatok felvétele és átviteli függvények megmérése.
3. A következő lépés a rendszeridentifikáció, azaz a rendszer rezgéskarakterisztikájának meghatározása a mért adatokból.
4. A negyedik lépés a kiszámolt eredmények validációja.
5. Az utolsó lépés pedig a számolt eredményekből való következtetések levonása, szisztematikus modell javítása.

A diplomamunkámnak nem célja ezen lépések részletes leírása, viszont fontos, hogy a kísérleti móduselemzés során elvégzett mérések (második lépés) fontosságát megérthessük. Ehhez az első és harmadik lépést érdemes részletesebben megvizsgálni, ugyanis ezek vannak közvetlen kapcsolatban a mérés elvégzésével. A további lépések utána ezek eredményeire építenek, így közvetetten rájuk is hatással van.

2.1.1. Mérésösszeállítás, előkészületek

Az első lépés többek között amiatt fontos a mérés minőségének szempontjából, mert itt történik meg az elrendezés a hitelesítése. Emiatt feltételezzük, hogy a mérés során jó

eredményeket kapunk végig. A mérés validációja során a móduselemzés alapvető feltevéseit is mindig tesztelni kell, mivel ezekre építkeznek a további lépések:

- Időinvariancia: Akkor tekinthető egy rendszer indőinvariánsnak, ha a kimenete $y(t)$ és bemenete $x(t)$ akkor a bemenő jel bármekkora δ idő késleltetés hatására a válasz is δ idővel eltolt lesz, tehát $x(t+\delta)$ gerjesztésre $y(t+\delta)$ lesz a válasz. Ez a gyakorlatban azt jelenti, hogy a mérés alatt az egyes mérési pontokban az átvitel statikusnak tekinthető. Ez jellegzetesen hosszabb méréseknél lehet probléma, mivel például hőmérsékletváltozás hatására a mért tárgy anyagi tulajdonságai megváltozhatnak.

- Linearitás: A rendszer dinamikus viselkedése lineáris, ha $x_1(t)$ és $x_2(t)$ két gerjesztés, melyekre $y_1(t)$ és $y_2(t)$ a két válasz, akkor $c_1 \times x_1(t) + c_2 \times x_2(t)$ gerjesztésre a válasz pedig $c_1 \times y_1(t) + c_2 \times y_2(t)$, tehát a mért pontokon az átvitel nem függ a bemenő jel amplitúdójától és típusától. A mérés megkezdése előtt ennek az ellenőrzését úgy végzik el, hogy ugyanabban a pontban különböző erősségű gerjesztéseket visznek be, általában három különböző nagyságút, utána az átviteli függvényeket ellenőrzik. Ha ezek között jó az egyezés, a rendszer lineárisnak tekinthető. Kifejezetten a rezonancia csúcsok egyezése fontos a későbbi számításokhoz.

- Megfigyelhetőség: A meghatározandó módusok legyenek mind mérhetőek, tehát például, ha az egyik mérési pontban pont egy csomópontja van a rendszernek (tehát adott frekvenciájú gerjesztésre nem keletkezik azon a ponton kitérés), akkor nem leszünk képesek megmérni az adott módust. Az ilyen szituációk felderítésére a koherencia (4.1. szakasz) vizsgálata adhat segítséget. Általában a mérési pont arrébb helyezésével megoldható ez a jelenség. A másik jellegzetes jele ennek a problémának, ha két, frekvenciában egyértelműen különböző módus alakja nagyon hasonló. Általában ezt a MAC (Modal Assurance Criterion, 4.4. szakasz) mátrix is kimutatja. Ilyenkor a mérési pontok számának növelésével javíthatunk a mérési eredményeken, például az 1 irányú gyorsulás szenzort egy három irányú gyorsulásszenzorra érdemes ilyenkor cserélni.

- Maxwell reciprocitás törvénye: Az átviteli vektor elemeinek $h_{sq}(\omega) = \frac{u_s(\omega)}{f_q(\omega)}$ definíció szerinti mérése komplikált lehet, ahol s egy mérési pont és q egy gerjesztési pont. A definíció szerint ehhez szükséges lenne annyi szenzort rátennünk a mérendő eszközre, ahány mérési pontot szeretnénk. Ez kis testek esetén amiatt lehet problematikus, mert a szenzorok össztömege összemérhető lehet a mért tárgy tömegével, ezzel jelentősen befolyásolva a test modális viselkedését. Emellett sokszor, ha több száz pontot szeretnénk megmérni, akkor lehet nem is áll rendelkezésünkre ennyi szenzor. Reprodukálható gerjesztés esetén megoldható ugyan, hogy az átviteli vektort elemenként mérjük meg, tehát a szenzort mozgatjuk, de ez is körülményes a szenzor többszöri le és felszerelésével. A reciprocitási elvet a 2.1 egyenlet írja le [6], mely azt jelenti, hogy az átvitel megegyezik, ha az s pontban gerjesztünk és a q pontban mérünk és fordítva. Így megtehetjük, hogy a rendszert egyetlen pontban mérjük és különböző pontokban gerjesztjük.

$$h_{sq}(\omega) = \frac{u_s(\omega)}{f_q(\omega)} = \frac{u_q(\omega)}{f_s(\omega)} = h_{qs}(\omega) \tag{2.1}$$

Habár ezeket a feltételeket mind tesztelik és vizsgálják a mérés kezdete előtt, ha közben történik változás, azt nem mindig fogja a mérést végző tesztmérnök észrevenni. Ezen

feltételek bármely sérülése a későbbiekben számítható módusalakok torzulását eredményezik, amik csak a negyedik lépésben lesznek észrevehetőek, de ilyenkor az adott mérési elrendezést már sokszor szétszerelték.

2.1.2. Rendszeridentifikáció

A mérések elvégzése után a rendszer identifikációja során számoljuk ki az egyes módusalakokat az adott rezonanciagörbékhez. A mért átvitelifüggvényekből az egyes modális paramétereket becsülhetjük (2.2, [6]), melyekre az elmúlt években több különböző technikát is kidolgoztak [13]. A becsült paraméterek többek között a rendszer pólusai, csillapított rezonanciafrekvenciái ω_n, csillapítási tényezői ξ_n, módusalakjai φ_n és modális részesedési tényezői.

$$h_{sq}(\omega) = \frac{u_s(\omega)}{f_q(\omega)} \approx \sum_{n=1}^{M} \frac{\varphi_{ns}\varphi_{nq}}{\omega_n^2 + j\omega 2\xi_n\omega_n - \omega^2} \tag{2.2}$$

Ahol M a mért módusok száma, φ_{ns} az n-edik módusalak értéke az s-edik megfigyelési pontban és φ_{nq} a q gerjesztési pontban. ω_n és ξ_n rendre az n-edik sajátfrekvencia és csillapítási tényező. Fontos megjegyezni, hogy az M nem a mérésből adódik, hanem a kiértékelés során állapítható meg, hogy mi az a legkisebb fokszám ami jól leírja a rendszer viselkedését.

2.2. Terheléses tesztek

A terheléses tesztek fő célja az eszköz elfáradásának és megbízhatóságának vizsgálata, valamint egy tárgy élettartamának megállapítása. Ehhez széles körű méréseket alkalmaznak, például magas vagy alacsony hőmérsékleti teszteket, hőciklusokat, magas páratartalmat, mechanikai terheléseket és ezek kombinációit, hogy a különböző termékek minél valóságközelibb körülmények között legyenek tesztelve. A rázóasztalos mérések, különösen a rezonancia-tűrésvizsgálatok, fontos szerepet játszanak a mechanikai terheléseknél.

A megvalósítandó monitorozó szoftvernek nem fő feladata az ilyen típusú mérések vizsgálata, viszont alkalmas lehet erre is, mivel hasonló az ezekhez szükséges elrendezés.

2.3. Gerjesztés

A mérések során valamilyen módon gerjesztenünk kell a tárgyat és ennek válaszát vizsgáljuk az adott célnak megfelelően. A gerjesztés típusának megválasztása több szempontból is fontos: Az általunk érdekelt frekvencia vagy frekvenciatartomány legyen gerjesztve. Itt a fő szempont, hogy időtartománybeli vagy frekvenciatartománybeli analízisre van-e szükségünk. A másik szempont pedig, hogy kellően nagy energiát vigyünk a vizsgált tárgyba. Ez is természetesen a felhasználási céltól függ. Például egy erősen csillapított tárgy esetén nagyobb energiát kell bevinni adott frekvencián, hogy a jel-zaj viszony elegendően nagy legyen. A gerjesztésre használt jeleket többféleképpen is lehet csoportosítani de alapvetően két fő típust különböztethetünk meg [13].

2.3.1. Szélessávú gerjesztés

A szélessávú gerjesztőjelek két csoportra bonthatóak: véletlenszerű zaj és impulzus jellegű gerjesztés.

- A véletlenszerű zaj normális eloszlású véletlen pontokból álló jel. A cél a megfelelő mennyiségű energiabevitel. Sokszor használnak periodikus zajt vagy álvéletlen zajokat (például Maximum Length Sequence) az ilyen mérések során. Előnye az ilyen zajoknak, hogy a spektrális szivárgás elkerülhető ablakozás nélkül is. Az ilyen típusú méréseket rázóasztal segítségével végzik és a mérési előkészületek emiatt sokkal bonyolultabbak, mivel a megfelelő gerjesztési pont sokszor nem szabadon hozzáférhető területen van. A rázóasztal a reprodukálhatóságot is garantálja.

- Impulzus jellegű gerjesztést úgynevezett impulzuskalapáccsal valósíthatunk meg. Ennek előnye, hogy a mérés rövid ideig tart, viszont a reprodukálhatósága nehézkes, mivel kétszer ugyanakkora erővel és iránnyal ütni ember számára szinte lehetetlen emiatt csak lineáris rendszerek esetén használható. A gerjesztett frekvenciatartományt a kalapácsfej anyagának a kicserélésével állíthatjuk. Minél keményebb fejet használunk, annál nagyobb sávszélességet tudunk gerjeszteni, viszont az egységnyi idő alatt bevitt energia ezzel egyenesen arányosan nő, amire figyelnünk kell a szenzorok túlvezérléseinek szempontjából.

A szélessávú gerjesztőjelek esetén szokás több mérés átlagolását használni. Ha sztochasztikus a jel, akkor ez kifejezetten kell is , ha pedig determinisztikus, akkor az átlagolás opcionálisan használható. A diplomamunkámban fehérzajos és sepertszinuszos gerjesztésű mérésekkel foglalkozok részletesebben.

2.3.2. Szinuszos gerjesztés

Az ilyen típusú gerjesztések célja, hogy megmaradjon a szélessávú gerjesztés viszont egyazon időben mindig csak egy adott frekvenciát gerjesszünk. Ezzel az adott frekvencián bevitt energia mennyisége jobban kontrollálható és a rendszer állandósult állapotú válasza is jobban vizsgálható. Ennek köszönhetően az időtartománybeli elemzés is lehetséges, így kifejezetten nemlineáris viselkedések kimutatására alkalmas. Alapvetően két típust különböztethetünk meg:

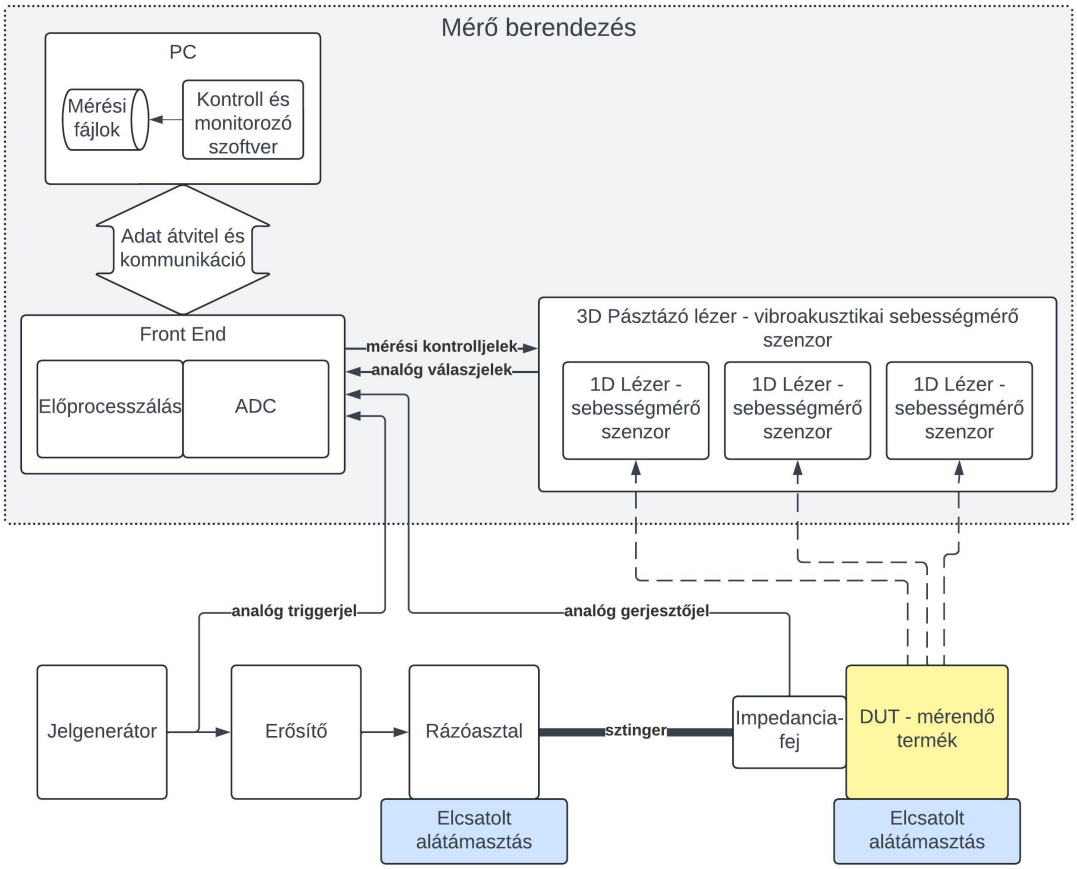
- A sepert szinusz esetén a frekvencia folytonosan változik.

- A léptetett szinusz esetén a frekvencialépték diszkrét eloszlású. Ezzel specifikusabban több energiát lehet bevinni egy adott frekvencián a rendszerbe.

2.4. Általános mérési elrendezés és a monitorozásához szükséges módosítások

Általánosságban a mérési elrendezések nagyon hasonlók. A különbségek többnyire csak a mérendő test méretétől függenek. Kisebb testeket elég lehet ugyanis impulzuskalapácsgerjesztéssel mérni viszont ez nem témája a diplomamunkámnak. A mérési berendezéseket gyártó cégek sajnos nem feltétlen teszik elérhetővé a mért jeleket további sajátcélú feldolgozásra, emiatt ezek kinyerése nehézkes. Egy általános mérési elrendezést a 2.1. ábra mutat.

A számítógépen futó szoftver felelős az egész mérés rendben lezajlásáért és itt lehet a különböző specifikus paramétereket beállítani. Ez a számítógép a mérőberendezéshez tartozik, így nagyon lekorlátozott, és csak a gyártó cég által telepített szoftverek használhatóak rajta. Az ezen futó szoftver felelős a móduselemzésért is és ennek eredményeit lehet a mérés elvégzése után exportálni. A mérés lemenete után a mért átvitelifüggvényeket vagy időjeleket (mérési beállítástól függ, hogy mit mentett ki a szoftver) már ki lehet exportálni, viszont ennek a metódusa manuális, emiatt időigényes és nem garantált



2.1. ábra. Egy általános mérési elrendezés blokkdiagramja.

például, hogy az időjelek rendelkezésünkre állnak majd. Ennek következményeképp nem ezeknek a jeleknek a feldolgozása és hibamonitorozása volt a cél, annak ellenére, hogy ez lenne a legkisseb fizikai beavatkozást igénylő megoldás a problémára.

A szoftverben egy geometria megadásával lehet a mérőpontokat definiálni, hogy a mért átvitelifüggvények a megfelelő pontokhoz legyenek hozzárendelve. Ehhez a lézerszenzorokban egy kamera segíti a geometria definiálását, ahol láthatjuk, hogy a valóságban hol fognak elhelyezkedni a felvett mérési pontok [19].

A számítógép egy frontend-del kommunikál, aminek a fő feladata a szenzorjelek megfelelő felvétele és előfeldolgozása. Ez magában foglalja a jelek mintavételezését akár túlmintavételezését, esetleges előszűrését és egyéb ezzel kapcsolatos műveleteket. Bizonyos esetekben jelgenerátorként is működhet, aminek segítségével úgynevezett *closed-loop* vezérlés is lehetséges, ha például a gerjesztő referencia erőjelre szabályoz úgy, hogy a kimenet amplitúdója garantáltan konstans legyen frekvenciától függetlenül, ezzel garantálva, hogy ugyanannyi energia kerül bele a rendszerbe a mérés minden pillanatában. Emellett ez vezérli a lézer vibrométerek működését, kontrollálva a fókuszpont térbeli elhelyezkedését.

Abban az esetben, ha a jelgenerátor külső eszköz, akkor ennek a jelét vissza szokás vezetni a frontendbe, bár ez nem kötelező, de sokkal jobb jel-zaj viszonyú jel, mint a gerjesztő pontnál mért, így a mérések triggerelését ezzel szokták vezérelni. A triggerelés a mérés szempontjából amiatt fontos, mert ideális esetben a mérési pontoknál ugyanarra a gerjesztésre mérjük a választ a 2.1. alszakaszban leírtak alapján, így, ha sepert szinuszos gerjesztést használunk akkor a jelgenerátort úgy érdemes beállítani, hogy a folyton ismétlődő szekvenciák között legyen néhány másodperc szünet, amíg a szoftver elmenti

az előtte mért átvitelt, és újra elkezdi "figyelni" a trigger csatornát, várva az újabb mérés megkezdésére. A szoftverekben általában van lehetőség úgynevezett pre-, és post-trigger idők megadására. Ez annyit jelent, hogy ha az adott csatorna elérte a megadott jelszinthatárt, akkor a pre-trigger visszamenőleg, a post-trigger pedig a mérés vége után még az itt megadott ideig mér. Ez azért hasznos, hogy garantálhassuk az egész szekvencia felvételét ezzel az esetleges spektrális szivárgást elkerülve (ha példáula felvétel úgy fejeződik be, hogy egy szinuszt "levág").

A 3D pásztázó lézer vibrométer három darab egy irányba mérő lézeres sebességmérő szenzorból épül fel, melyek térbeli elhelyezkedésükből adódóan tudnak háromdimenzióban sebességet mérni, nagy mintavételi frekvenciával (100 kHz-es tartományban). A lézerfény megfelelő visszaverődésének biztosításához, a mérendő eszközre visszaverő fóliát szokás ragasztani [19].

A gerjesztést a DUT-be (Device Under Test) a rázóasztalból úgynevezett sztingeren keresztül viszi be, ami tekinthető egy vékony rúdnak, mely hosszanti irányban végtelenül merev és keresztirányban végtelenül rugalmas. Ennek legfőképp két szerepe van a mérés során:

- Bármilyen oldalirányú erőt eliminál̈jon, ami nem a gerjesztés irányába mutat. Ezek viszonylag könnyen jelentkezhetnek, ugyanis a mérés közben a rázógép és a DUT tökéletesen egymáshoz igazítása érzékeny folyamat és könnyen kimozdulhatnak egymás síkjából. Emiatt nemkívánatos oldalirányú erők léphetnek fel.

- A rázógép egy nagyobb test, és sokszor a kiszámolt optimális gerjesztési pontban nincs annyi hely, hogy odaférjen a közelébe így szükséges egy meghosszabbítást használni, ami esetünkben a sztinger.

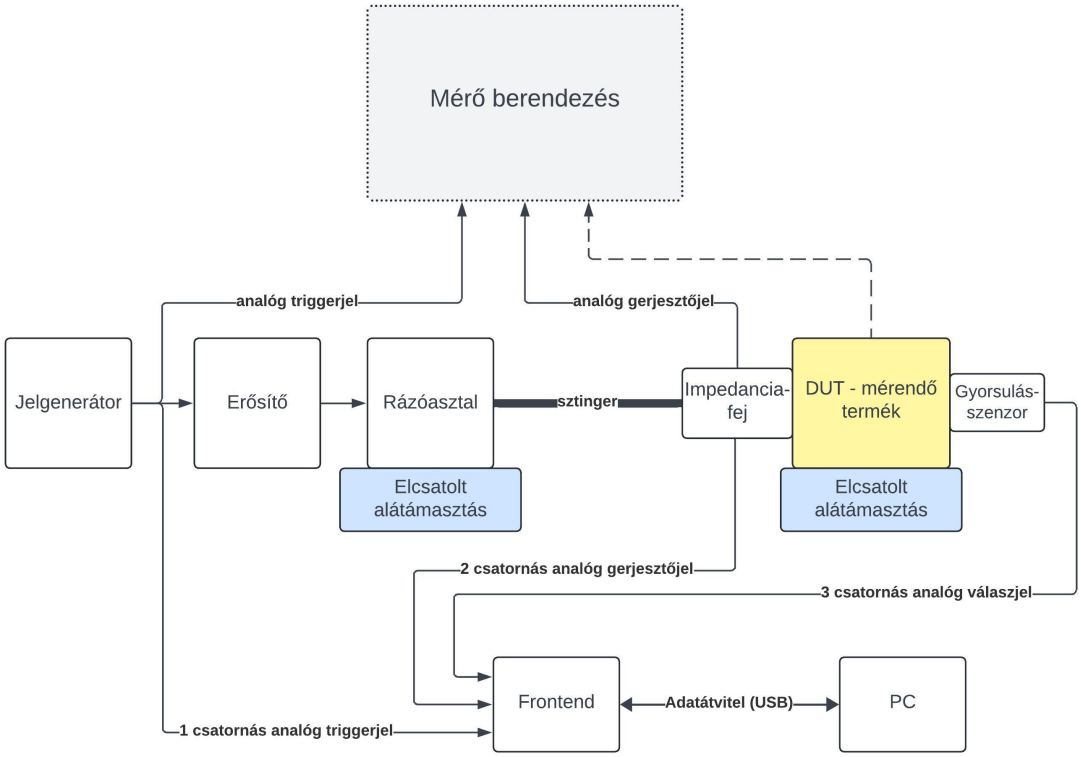
Mind a shakert és a mérendő eszközt akusztikailag el kell csatolni a környezetétől, hogy egyértelműen csak a DUT legyen a vizsgálat tárgya a megadott peremfeltételekkel. A rázógép oldaláról ez úgy van megvalósítva, hogy gumitalpakon áll a berendezés, a terméket pedig vagy gumikötél felfogatásokkal, vagy pedig szivacs alátámasztással csatolják el.

A rázóasztalnak a fizikai szerkezetéből adódóan a kiadott energiája frekvenciafüggő, aminek a főbb paramétereit a dokumentációjában fel szoktak tüntetni. Ezt figyelembe kell venni egy mérés során. A jelgenerátor és a rázóasztal között egy teljseítményerősítő van bekötve, aminek szintén van egy karakterisztikája, bár ez a mérés során tekinthető ideálisnak.

Az impedanciafej a sztinger végére van felcsavarozva és a DUT-re van ráerősítve viasszal vagy pillanatragasztóval. Ez a mérőeszköz egy egyirányú gyorsulás- és egy erőszenzor kombinációja. Az átvitel számításához az erőjelet veszik a rendszer bemenetének. Emiatt is fontos, hogy közvetlenül a tárgyon helyezkedjen el, mivel így pontosan mérhető a bevitt erő és ebből az átvitel.

A mérőberendezés viszonylag zárt rendszer, így a monitorozást a még kívülről elérhető jelekből kell valahogy elvégezni. Ennek a kiegészítéséhez elfogadható, ha egy triaxiális gyorsulásszenzort még felszerelünk a DUT-re, és ezzel a mérési elrendezést nem befolyásoljuk számottevően. Ennek az elhelyezése kritikus a monitorozás szempontjából, ugyanis csak ennek a szenzornak a mért jele és a gerjesztő erőjel közötti anomáliákat fogjuk tudni kimutatni. Ettől függetlenül, ha valahol nemlineáris torzulás lép fel a rendszerben, az valószínűleg minden jelen kimutatható lesz, mert hatni fognak egymásra. A 2.2. ábra a monitorozó rendszer bekötését mutatja. Látható, hogy egy egyszerű jel lecsatlakozással a triggerjel és az impedanciafej mérhető emellett a gyorsulás szenzor adja a válasz jelet. Ez összesen pont 6 csatorna, (1 db: trigger, 2 db: impedanciafej – gyorsulás és erő – és 3 db: triaxiális gyorsulásszenzor) ami a rendelkezésre álló frontendek közül a legegyszerűbb felső határa, emiatt ideális a monitorozásra.

Összességében fontos megjegyezni, hogy ez az összeállított monitorozórendszer egyedül a gerjesztéstől a mérendő eszközig képes hibákat detektálni a mérőpad zártsága miatt. Emiatt például, ha a lézernyaláb reflexiója nem elegendő vagy a fókuszpontja nincs jól beállítva, akkor ezeket nem fogjuk tudni detektálni. Ezeknek a monitorozására viszont nincs is feltétlen szükség, ugyanis, ha az elrendezés megfelelően validálva lett, akkor – ha a test nem mozdul el a helyéről – akkor nem is fordulhatnak elő ilyen típusú anomáliák.



2.2. ábra. A monitorozó rendszer elrendezés blokkdiagramja.

3. fejezet

Anomáliák és típusaik

A következő fejezetben az anomáliák típusait, előfordulásuk természetét és esetleges okait részletezem. A hibatípusokat két nagyobb csoportra lehet osztani: modális viselkedés változásából fakadó, és nemlineáris jellegű hibák. A valóságban ezek nem mindig különíthetőek el és sok esetben ennek a kettőnek valamilyen kombinációja fordul elő.

Ahogy azt a 2.1. szakaszban korábban említettem, a kísérleti móduselemzés során a linearitás alapvető feltevés, amit mindig kritikusan vizsgálni kell a mérés megkezdése előtt. Ezt a vizsgálatot nevezik linearitásvizsgálatnak, amit úgy valósíthatunk meg, hogy különböző bemenő energiaszintű gerjesztéseket viszünk be a rendszerbe és a test különböző pontjain vizsgáljuk a válaszát. Ezt követően átviteli függvényeket számolunk és a vizsgált néhány ponton a különböző erejű görbéket összehasonítjuk. A pontokat érdemes úgy megválasztani, hogy az a későbbi mérést a lehető legjobban reprezentálja. Kritikus helyeken is érdemes ezt elvégezni, például két csatlakozási pont között, vagy anyaghatárok között. Ha a mért görbék egybeesnek, akkor a rendszer tekinthető lineárisnak. Abban az esetben, ha a test csak közel-lineáris, de a kívülről beható zajok miatt az összehasonlított átviteli görbékben esetlegesen eltérés lelhető fel (például azt látjuk, hogy a rezonanciacsúcsok amplitúdói eltérnek), akkor is elvégezhető a mérés, viszont ilyenkor egy görbét több ugyanazon mérésből kell átlagolva számolni. Ez többnyire öt és tíz közötti mérés átlagolását jelenti [13].

Ez a vizsgálat a kezdeti kontakthibák nagy százalékát képes kiküszöbölni, mégis a mérés során keletkezhetnek különféle nemkívánt eltérések, amik észrevétele nem triviális.

3.1. Modális viselkedés változása

Az ilyen esetek során a rendszer linearitása megmarad, viszont egyéb fizikai paraméterei megváltoznak. Az ilyen változások ritkábban előforduló események és nehezebben is detektálhatóak. Lineáris eltérés alapvetően a rendszer modális viselkedésének megváltozását jelenti. Ez legfőképp úgy tud előfordulni, ha például valami ráerősített tömeg leesik róla, vagy hőváltozás hatására a csillapítási tényezője eltolódik. Ezekben az esetekben a lineáris kapcsolat megmarad, viszont az előbbi példában a rezonanciafrekvenciák változhatnak meg, az utóbbi példa esetén, a rezonanciagörbék szélesednek ki, és az amplitúdójuk is lecsökken. Másik eshetőség, ha a rendszer reciprocitása változik, pontosabban nem lesz érvényes tovább már. Ez például két mérés közötti külső behatás eredményeképpen jöhet létre, például, ha egy villanymotor rotorja elfordul. Ebben az esetben a tömegeloszlása is megváltozhat a rendszernek, ezzel hibát víve a mérésbe.

Ezek kimutatása időtartománybeli analízissel nehézkes, szinte lehetetlen. Frekvenciatartománybeli analízis során sem ragadható ki egyértelmű metódus. Ha kiszámoljuk az adott átvitelifüggvény rezonanciagörbéinek modális paramétereit, majd ezeknek az el-

téréseit vizsgálva lehetne következtetéseket levonni. Például egy gépi tanuló algoritmus segítségével különválaszthatóak lehetnek a bizonyos eseteket [16].

3.2. Nemlineáris torzulás

A nemlineáris eltérések sokkal jellegzetesebben előforduló események a méréstechnikában. Az ilyen torzulások esetén a lineáris viselkedése változik meg a rendszernek. Nemlinearitás többféle formában jelentkezhet, de a három leggyakoribb típus a geometriai, anyagi és kontaktus nemlinearitás [9].

3.2.1. Geometriai nemlinearitás

Geometriai nemlinearitásról akkor beszélünk, ha egy szerkezetre ható erő jelentős alakváltozáshoz vezet. Ez olyan testeknél fordulhat elő, amelyek nagy terhelésnek vannak kitéve és emiatt az alakváltozás meghaladja az 5% körüli tipikus küszöbértéket. Ha a deformáció elég nagy, az a szerkezet dinamikus merevségének és dinamikus tömegének megváltozásához vezethet. Ez jelentős hatással lehet a móduselemzés eredményeire, mivel a linearitás feltételezései már nem érvényesek ezen a területen. Többnyire különböző rudak és lapok esetén fordulhat elő geometriai nemlinearitás, így a technikusnak minden mérés előtt mérlegelnie kell, hogy a gerjesztés iránya és nagysága eleget tesz-e a feltételeknek. Például egy két végén kifeszített kötél esetén láthatjuk, ha oldalirányú erő behatása alá kerül, akkor meg fog nyúlni, ahogy azt a 3.1. egyenlet mutatja, nemlinearitás kerül be a kitérés és a bevitt erő közé.

$$F = k(L - L_0)^2 \tag{3.1}$$

Ahol F az oldalirányú erő, L a megnyúlt hossza, L_0 a közel kezdeti hossza és k pedig a merevsége. Ide sorolható még az anyagi elfáradásból származó nemlinearitás, ami annak a következménye, hogy egy apró repedés elindul az anyagban. Ez tovább terjedve később kontakt nemlinearitáshoz vezethet. Általában a mérés megkezdése előtt tisztában vagyunk azzal, hogy nagyságrendileg mekkora erőt fogunk bevinni a testbe, emiatt a linearitásvizsgálattal kiderül, ha geometriai nemlinearitás lép fel.

3.2.2. Anyagi nemlinearitás

Anyagi nemlinearitásról akkor beszélünk, amikor egy szerkezet anyagi tulajdonságai az alakváltozással megváltoznak. Ez a jelenség minden anyagban valamilyen szinten jelen van, viszont többnyire elhanyagolható. Kifejezetten a gumi és bizonyos fémek esetén ezzel számolni kell. Az anyagi nemlinearitás olyan anyagoknál is előfordulhat, amelyek magas hőmérsékletnek vannak kitéve. Alapvetően ez a jelenség úgy keletkezhet, ha erő hatására az anyag feszültség-nyúlás aránya a deformáció közben megváltozik. Ez az anyag merevségének megváltozásához vezethet. Ez a fajta nemlinearitás is többnyire kiszűrhető az elején, így a mérésünket nem fogja ez befolyásolni.

3.2.3. Kontaktus nemlinearitás

Végül a kontakt vagy súrlódási hibákból származó nemlinearitás a mérés során keletkező leggyakoribb hibafajta [29]. Ennek előfordulása a test különböző anyagi vagy mechanikai kapcsolatainál jelentkezhet, amikor ezek az elemek kölcsönhatásba lépnek egymással. Ez olyan tárgyaknál fordulhat elő, amelyek mozgó alkatrészekkel rendelkeznek, mint például

fogaskerekek és csapágyak, vagy olyan szerkezetekben, amelyek több, egymással összekapcsolt alkatrészből állnak, mint például egy autó karosszériája és alváza. A kontakt nemlinearitás fő oka a hosszan tartó erőhatásoknak kitett alkatrészek apró mozgásai, ami a két alkatrész közötti kölcsönhatásból fakad. Sok esetben ezt egyszerűbb csavarmeglazulás vagy ragasztó gyengülés is okozhatja. Egyik következménye, hogy az ideálisan kapcsolt elemek meglazulnak, ezzel egyfajta ütközés fog fellépni a gerjesztés során a két elem között.

3.3. Egyéb lehetséges anomáliák

A korábban felsorolt tipikusabb eltéréseken kívül előfordulhatnak ritkán egyéb esetek is. Legfőképpen ezek külső behatás miatt keletkeznek például, ha véletlenül hozzáér valaki és elmozdítja az elrendezést vagy áramszünet lép fel. Ezek álltalában észre is vehetőek, így ezeket nem kifejezetten szükséges vizsgálni.

Másik lehetséges hibajelenség, ha régebbi kábelt vagy csatlakozót használunk és emiatt kontakthiba jelentkezik valamelyik szenzor jelén. Ennek a fő tünete, ha véletlenszerű időközönként impulzusok jelennek meg a jelünkön, esetenként nem is látjuk a jelet az adott csatornán és ezek a jelenségek sűrűbben jelentkeznek, ha mozgatjuk a kábeleket. Ennek a fennállását általában a mérés megkezdése előtt már felderíti a mérést végző mérnök, és a hiba lokalizációja után kicseréli a meghibásodott kábelt. Ha mégsem derül erre fény időben, a kiértékelés alatt az időjelek vizsgálatával feltűnhetnek ezek a tüskék, vagy a frekvenciatartományban vizsgálva, a spektrogramon több szélessávú impulzusra lehetünk figyelmesek.

4. fejezet

Detekciós eljárások

A következő fejezetben különböző anomália-detekciós eljárásokat mutatok be. A célja ezeknek a metódusoknak, hogy a 3. fejezetben részletezett eltéréseket minél jobban leírják. Ezek segítségével egy mérés minőségéről lehetőleg minél reprezentatívabb értéket szeretnénk kapni, aminek a felhasználásával eldönthetőnek kell lennie, hogy a mérés folyamán végig megfelelő-e az összeállításunk. A modális viselkedés változásainak vizsgálatára korlátozottabb metódusok léteznek, viszont ennek az anomáliatípusnak az előfordulása is jelentősen ritkább.

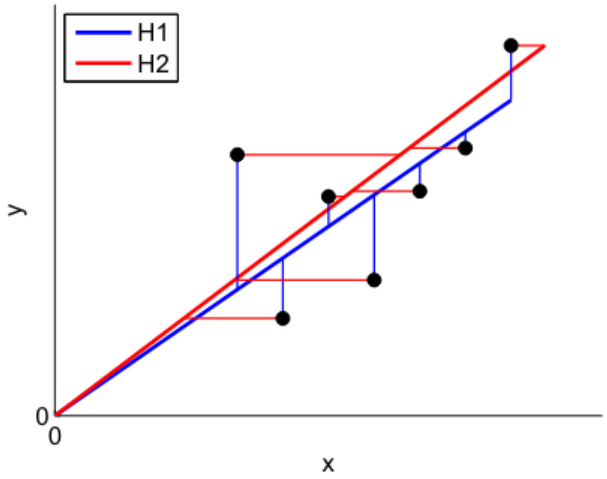
A következőkben felsorolt analitikai eljárások többnyire két adathalmaz közötti összefüggéseket vizsgálnak, mind statisztikai (RDC), mind jefleldolgozási (koherencia vagy FRAC) eljárásokat is bemutatok. Ezek mellett kitérek egy olyan metódusra is, amihez elegendő csak egy jelet vizsgálni (spektrális laposság). Mindegyik kiértékelési algoritmusnak megvan a saját előnye és hátránya is, amiket mérlegelni kell a szoftverben való megvalósításhoz.

Alapvetően kétféle megközelítésre bonthatóak fel ezek az eljárások. Az egyik esetén nincs szükség referenciajelre, tehát a mérési adatokból kimutatható a mérés minősége bármyilen külső összehasonlítás nélkül. Ebben az esetben szükséges megadni egy döntéshozatali határértéket, emiatt közvetetten szükséges lehet referenciamérés. A másik típusú csoport referenciaméréshez hasonlítja az adott eredményeket és a különböző eltéréseket vizsgálja. Ebben az esetben is szükséges a küszöbérték meghatározása, ugyanis el kell dönteni, hogy hány százalékos eltérés a maximum megengedett. Ennek az esetnek a gyakorlati megvalósítása azon a feltételezésen alapszik, hogy a mérés kezdetekor még az elrendezés helyes, így ez alatt az időtartam alatt referenciának tekinthető a mérés.

4.1. Koherencia

Az akusztikai méréseknél (például kísérleti móduselemzés) legtöbbször koherenciavizsgálatot végeznek anomália detekcióra, a referenciajel és az egyes megfigyelési pontok között [10]. A koherencia frekvenciafüggően határozza meg, hogy mennyire zajmentes a mért átvitel, pontosabban mennyire tekinthető a mért rendszer lineárisnak az adott frekvenciatartományban.

A számítása a $H1$ és a $H2$ átviteli függvény becslőkből származtatható, mivel a $H1$ becslő a válasz függvényében számolja a minimális hibát az egyenesillesztéshez, míg a $H2$ pedig a gerjesztés függvényében (4.1. ábra). A kettő aránya adja a koherenciát (4.1. egyenlet). Más szóval a koherencia adott frekvencián, a gerjesztés amplitúdójától függően mért válaszra illesztett egyenes determinációs együtthatója. A statisztikában is használatos, ahol R^2 a jelölése.



4.1. ábra. H1- és H2-becslők. A H1-becslő a kék, a H2-becslő a piros vonalakkal jelzett távolságok négyzetösszegeit minimalizálja.[6]

Belátható, hogy 0 és 1 közötti intervallumon veheti fel értékeit. Ha a rendszer lineáris, akkor az értéke 1, ellenkező esetben pedig 1-nél kisebb számot kapunk. Az értéke abban az esetben közelít a nullához, ha a mérést zaj terheli vagy nem írható le lineáris modellel. Érdemes megjegyezni, hogy a koherencia értéke ott is lecsökkenhet, ahol gerjesztőjel nem képes a testbe energiát bevinni, tehát ahol a rendszernek zérusa van. A koherenciát legfőképpen átviteli függvény számításkor használják, mivel ez visszajelzést ad, hogy mely frekvenciákon tekinthető az átvitel értéke megbízhatónak.

$$Coh(f) = \frac{H1(f)}{H2(f)} = \frac{S_{ab}(f)/S_{aa}(f)}{S_{bb}(f)/S_{ba}(f)} \tag{4.1}$$

$$Coh(f) = \gamma^2(f) = \frac{|\sum_{i=1}^{n} S_{ab_i}(f)|^2}{\sum_{i=1}^{n} S_{aa_i}(f) \sum_{i=1}^{n} S_{bb_i}(f)} \tag{4.2}$$

Ahol az $S_{aa}(f)$ és az $S_{bb}(f)$ az egyes jelek auto-teljesítményspektrumai, az $S_{ab}(f)$ pedig a kereszt-teljesítményspektrumuk. Az egyenletben látható, hogy a teljesítmény spektrumok szummáját kell venni, ez az ami lehetővé teszi az átlagolást, mert ha csak egy pontra vennénk fel az egyenest, akkor könnyen belátható, hogy a koherencia értéke mindenképp 1 lesz. Hogy több teljesítményspektrumot kaphassunk, több mérésiciklust fel kell vennünk és ezekkel fogunk valós koherenciát kapni.

4.2. Spektrális laposság

A következő analitikai metódus motivációja a nemlineáris torzulásból fakadó felharmonikusok vizsgálatából indult ki. A koherencia robusztus megoldást mutat a problémára, viszont a döntéshozatali szint nem triviális, mivel az értéke a rendszer zérusaiban jelentősen lecsökken.

Szinuszos gerjesztés esetén lineáris rendszer válasza is szinuszos, míg nemlineáris rendszer esetében a jelalak torzulása miatt valamekkora mennyiségben felharmonikusok fognak

megjelenni. A mérésből fakadóan, a laza kapcsolat miatt a szinusz csúcsainak közelében ütközések fognak fellépni, esetlegesen egy laza csavar esetén nyugalmi pozíció körül a kontaktus meg is szűnhet. Ha képesek vagyunk a felharmonikus jelenlétének detektálására, akkor ebből meghatározható a mérés linearitása is.

A spektrális laposság [12], mint a nevében is benne van, adott jel spektrumának az egyenletességére határoz meg egy arányszámot. A fehérzaj spektruma az 1 értéket közelíti és a szinusz jelé pedig 0-t eredményez. Az mérőszám a geometriai és aritmetikai közép hányadosából számolható ki (4.3. egyenlet),

$$\mathrm{SPF} = \frac{(\prod_{i=1}^{n} x_i)^{\frac{1}{n}}}{\frac{1}{n}\sum_{i=1}^{n} x_i} \tag{4.3}$$

ahol x_i a számolt spektrum egyes értékeinek abszolútértéke és n pedig a spektrum felbontása. Előnye ennek a metódusnak, hogy elég csak egy jelet felhasználni és azon elvégezni az analízist. Ahhoz, hogy a gyakorlatban a spektrális laposság hasznunkra legyen, először elő kell készítenünk a jeleket. A geometriai közép kiszámolásához a közismert képlet nem alkalmas, mivel nagy mennyiségű, kisebb mint 1 értékű elemeknél a szorzat a nullába fog tartani és a számítógép kerekítési okaiból 0 is lesz. Ennek megoldására a logaritmikus módszert kell alkalmazni, a módszert a 4.4. egyenlet mutatja.

$$\text{Geometric Mean} = \exp\left(\frac{1}{n}\sum_{i=1}^{n}\ln(x_i)\right) \tag{4.4}$$

A következő probléma, hogy csak szinuszos gerjesztésnél mérhető a nemlineáris torzulás, emiatt csak chirp vagy léptetett szinuszos gerjesztésű méréseknél alkalmazható ez a technika. Emellett a gerjesztőjelet ki kell vágni a vizsgált spektrumból, mivel ezzel együtt a spektrális laposság értéke mindig közel 0 értékű lesz. Ehhez szükségünk van a referenciajelre, de ezt többnyire mindig monitorozzák mérések során, így ez a jel probléma nélkül felhasználható a detekció során. A referenciajelből kinyert aktuális frekvenciát hasonlóan a koherenciaszámításnál, csúsztatott-ablakozó FFT (Fast Fourier Transform) algoritmussal lehet kinyerni. Ahhoz, hogy minél pontosabban lehessen meghatározni az adott frekvenciát, az ablakméret megválasztása során két fő szempontot kell figyelembe venni:

- Ha nem alkalmazunk túlmintavételezést, akkor csak az ablakméret fogja befolyásolni az FFT frekvencia felbontását. Spektrumszámításnál adott a $\delta fT >= 1$ határozatlansági reláció. Vagyis az idő- és frekvenciabeli felbontás csak egymás kárára növelhető. Így, ha az ablakméretet minél kisebbre vesszük annál pontosabban lokalizálni tudjuk a jelenség a spektrumát, viszont a felbontása csökkenni fog.

- Ha az ablakméretet növeljük az FFT frekvenciafelbontása is növekedni fog, viszont lehet, hogy a folytonos sepert szinuszos gerjesztés miatt az ablakon belül változni fog annyit a frekvencia, hogy már nem tekinthető stacionáriusnak a gerjesztés frekvenciája és emiatt nem fogunk pontos értéket kapni eredményül.

Ezeket a szempontokat figyelembe véve a megfelelően megválasztott ablakmérettel, egy maximum kereséssel viszonylag pontosan meghatározható a gerjesztés frekvenciája.

A mérés során két ciklus között általában 1-2 másodperc telik el, míg nincs gerjesztés a rendszeren. Ennek kezelésére a számolt frekvencia amplitúdó nagyságát egy határértékhez kötöm, ami fölött elfogadhatónak tekinthető az exportált érték. Ezen kívül egy alsó illetve felső frekvenciakorlátot is bevezettem„ az egy alsó és felső frekvencia korlát, amin, ha kívül esik a detektált frekvencia akkor szintén nem tekinthető érvényesnek a mért eredmény.

Ezt a gerjesztési jel ismeretével elegendő a sepert szinusz kezdő- és végfrekvenciájának választani.

A spektrális laposság értékét is adott spektrumból tudjuk kiszámolni. A számítási igények csökkentése céljából a kettő feladatot összekötöttem és így elég csak egy feldarabolást elvégezni a két jelen, ahol először a gerjesztésből kinyerjük az adott frekvenciát, utána a válaszból kivágjuk ezt és végül elvégezzük a spektrális laposság számítást. A kivágás folyamán a gerjesztés körül Δf méretű sávot érdemes kivenni, mivel teljesen pontosan lehetetlen kivágni az adott frekvenciát a korábban felsorolt okokból kifolyólag. Emellett a spektrális laposság érzékenységéből adódóan is fontos, hogy megfelelően mindent kivágjunk a spektrumból.

Eredményül egy gerjesztés-frekvencia függő spektrális laposság, egydimenziós tömbjét kapjuk, aminek az összes értéke ideális esetben az 1-hez fog közelíteni a kivágásnak köszönhetően. Bármilyen nemlineáris jelenség esetén a megjelenő felharmonikusnak köszönhetően le fog csökkenni valahol az értéke, emiatt érdemes lehet a tömb minimumát vizsgálni.

4.3. Randomized Dependence Coefficient (RDC)

A Randomized Dependence Coefficient (RDC) [14] metódus két adathalmaz lineáris és nemlineáris függőségét is képes kimutatni. Használata gépi tanuló algoritmusok területén elterjedtebb, de jelfeldolgozási feladatokra is alkalmazható. A metódus fontosabb tulajdonságai a következők:

- Nemlineáris összefüggéseket is képes detektálni két halmaz között.
- Többdimenziós vektorok összehasonlítására is alkalmas.
- A copula transzformáció használatából adódóan invariáns a skálázási és eltolási transzformációkra.
- Rényi Alfréd kritériumainak eleget tesz.
- Értékkészlete a $[0; 1]$ intervallumra korlátos.
- A számítási igénye kicsi és az algoritmus relatíve könnyen implementálható.

Rényi Alfréd 1959-ben lefektetett 7 alapvető feltételt [28], amiket teljesítenie kell az olyan mérőszámoknak, amik két adathalmaz ($X \in x; Y \in y$) egymástól való függőségét írják le $\rho^* : x \times y \to [0; 1]$. Ezek a következőek:

1. $\rho^*(X, Y)$ definiált bármely két X, Y változóra.
2. $\rho^*(X, Y) = \rho^*(Y, X)$
3. $0 \leq \rho^*(X, Y) \leq 1$
4. $\rho^*(X, Y) = 0$ ha a két változó statisztikailag független.
5. Minden bijektív leképezhető Borel-halmaz függvényre igaz, hogy $f, g : \mathbb{R} \to \mathbb{R}\rho^*(X, Y) = \rho^*(f(X), g(Y))$
6. $\rho^*(X, Y) = 1$ minden Borel-halmaz függvényre f és g ha igaz rájuk, hogy $Y = f(X)$ vagy $X = g(Y)$.
7. Ha $(X, Y) \sim N(\mu, \Sigma)$, akkor $\rho^*(X, Y) = |\rho(X, Y)|$ ahol ρ a korrelációs együttható, μ a várható érték, Σ pedig a szórás, N pedig a normális eloszlást jelenti.

Ezeknek a kitéteknek mind eleget tesz ez az algoritmus, sőt ezeken alapszik. Az 1959-es cikk publikálásakor Rényi mutatott is már egy ilyen metódust, ami a Hirschfeld-Gebelein-Renyi Maximum Korrelációs Együttható (HGR) nevet kapta, viszont ennek a számítása egyedül elméleti alapokon létezett. Emiatt fejlesztették ki az RDC metódust 2013-ban. Ez 3 fő lépésből számolható. Először a két adathalmazon úgynevezett copula-transzformációt végzünk (4.3.1. alszakasz) majd ezeken elvégzünk k darab véletlenül megválasztott nemlineáris projekciót (4.3.2. alszakasz), amit utána kanonikusan korreláltatunk a két csoport között (4.3.3. alszakasz).

4.3.1. Copula-transzformáció számítása

Az első lépés a Copula-transzformáció elvégzése. Ennek célja, hogy az algoritmus skálázásra és DC-eltolásra érzéketlen legyen. Maga a transzformáció 2 kisebb lépésből áll: Először az adathalmaz értékeit növekvő sorrendbe teszi és mindegyiknek ad egy sorszámot. Az ugyanolyan értékek az adathalmaz szerinti sorrendben egymás utáni növekvő számokat kapnak. Most már egész számokból álló halmazt kaptunk, amiknek a relítv függősége és az elemek eredeti sorrendje megmaradt. Ezt követően ezt a halmazt át kell skálázni $[0; 1]$ intervallumra, amit úgy teszünk meg, hogy leosztjuk a tömböt az elemszámmal. Így végül megkapva a két adathalmaz külön-külön áttranszformált eredményét, úgy, hogy az egyes pontok abszolút távolságai maradtak egyedül figyelmen kívül.

Az egyik adathalmaz transzformációjának megvalósítását a 4.1. kódrészlet mutatja python nyelven.

```
cx = np.column_stack([rankdata(xc, method='ordinal') for xc in x.T])/float(x.size)
```

4.1. lista. Copula-transzformáció Pythonban megvalósítva

Ahol az xc egy többdimenziós bemeneti adathalmaz (`numpy` tömb [21]), amin dimenziókként külön-külön végzem el a transzformációt. A `rankdata(a, method='average', *, axis=None, nan_policy='propagate')` pedig a *scipy.stats* könyvtárból importált függvény.

Ennek a számítási komplexitása $O(dn\log(n))$ ahol n az adathalmaz mérete és d az adathalmaz dimenzió mérete.

4.3.2. Véletlenszerű nemlineáris vetületek generálása

A következő lépés az empirikus copula-transzformációk nemlineáris vetületekkel való kiterjesztése, hogy később a lineáris módszerek felhasználhatók legyenek az eredeti adatok nemlineáris függőségeinek megtalálására. Ezt a megközelítést a regresszió területein is használják. Ebben a lépésben alapvetően a korábban előkészített adatokból létrehozunk k darab véletlenszerűen választott szinuszos projekciót. A (4.5) egyenlet mutatja az elvégzendő műveletet, ahol a véletlenszerű paraméterek az $\omega_i \sim N(0, sI), b_i \sim N(0, s)$ Normális eloszlásúak. Az s külső paraméter megválasztása a projekció szórását befolyásolja, minél nagyobbra választjuk ezt a számot, annál „diverzebb" lesz a véletlenszerűen megválasztott nemlineáris műveletsokaság.

$$\Phi(X; k, s) := \begin{pmatrix} \sin(\omega_1^T x_1 + b_1) & \dots & \sin(\omega_k^T x_1 + b_k) \\ \vdots & \ddots & \vdots \\ \sin(\omega_1^T x_n + b_1) & \dots & \sin(\omega_k^T x_n + b_k) \end{pmatrix}^T \tag{4.5}$$

Mivel van egy véletlenszerű folyamatrész ebben a pontban, így láthatjuk, hogy minden egyes lefutás eredménye eltérő lesz. A szerzője a cikknek nagyságrendileg a $k = 20$-at ele-

gendőnek találta, hogy ne legyen nagy szórása a kimenetnek és kellően stabil eredményeket kaphassunk.

```
X = np.column_stack([cx, 0])
Rx = (s/X.shape[1])*np.random.randn(X.shape[1], k)
X = np.dot(X, Rx)
fX = np.sin(X)
```

4.2. lista. Nemlineáris projekció megvalósítása Pythonban

A 4.2. kódrészlet mutatja a megvalósított projekciót. Látható, hogy először egy 0-t hozzáteszek a tömbhöz. Ez amiatt szükséges, mert így az $\omega_i^T x_j + b_i$ művelet egyszerű elemenkénti szorzással megvalósítható, ami a harmadik sorban látható is. A komplexebb matematikai műveleteket *Pythonban* a *Numpy (np)* könyvtárban [21] implementált függvények segítségével végeztem el. A komplexitása ennek a feladatnak $O(k \log(d)n)$.

4.3.3. Kanonikus korrelációanalízis

Az utolsó lépés pedig a nemlineáris transzformáción átesett két adathalmaz lineáris kombinációi közül a maximum korrelációjának a kiszámítása. Ezt kanonikus korrelációanalízisnek nevezik, ami két vektor (α, β) kiszámolásával történik meg úgy, hogy a $\alpha^T X$ és $\beta^T Y$ szorzatok eredményéül bármely két véletlenszerűen választott érték maximálisan korreláljon. A kanonikus korreláció egy sajátérték-probléma megoldása, amit a 4.6. egyenlet részletez.

$$\begin{pmatrix} 0 & C_{xx}^{-1} C_{xy} \\ C_{yy}^{-1} C_{yx} & 0 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \rho^2 \begin{pmatrix} \alpha \\ \beta \end{pmatrix}, \tag{4.6}$$

Ahol a C_{xy} a két jel kovarianciája és a C_{xx} és C_{yy} pedig a varianciájuk. Ezt a gyakorlatban úgy oldottam meg, hogy összefűztem a két adathalmazt és transzponálva őket, kiszámoltam a teljes kovarinaciamártixukat, amit utána négy negyedre bontva, megkapjuk a szükséges mátrixokat. Ahogy ezt a 4.3. kódrészlet mutatja [4], itt is a *Numpy* beépített függvényeit használtam. Az utolsó lépés egy egyszerű maximumkeresés és ennek a gyöke (szórásnégyzetekkel volt eddig számolva) a kapott arányszám.

```
C = np.cov(np.hstack([fX, fY]).T)

Cxx = C[:k, :k]
Cyy = C[k:2*k, k:2*k]
Cxy = C[:k, k:2*k]
Cyx = C[k:2*k, :k]

eigs = np.linalg.eigvals(np.dot(np.dot(np.linalg.pinv(Cxx), Cxy),
                                np.dot(np.linalg.pinv(Cyy), Cyx)))
rdc = np.sqrt(np.max(eigs))
```

4.3. lista. Kanonikus korreláció kiszámítása Python kód

4.4. Frequency Response Assurance Criterion (FRAC)

A korábbi megközelítések több oldalról járják körbe a nemlineáris jelenségeket, viszont, ha a test modális viselkedés változására kerül sor, akkor nem fogják ezeket kimutatni. A lineáris eltérések legjobban az átviteli függvények görbéinek az elváltozásából mutathatóak ki, így az egyes görbék összehasonlítása és az eltérést mennyiségének a meghatározása intuitív megközelítés a problémára.

Erre az akusztikában szerte ágazóan használatos, úgynevezett FRAC (Frequency Response Assurance Criterion) [3] arányszám nyújt megoldást. Fő felhasználási területe szimu-

lációs átviteli görbék összehasonlítása, mért átviteli görbékkel, hasonlóan a MAC (Modal Assurance Criterion) [18] mátrixhoz, bár ez a már kiszámolt módusalakok összevetésére szolgál.

Látható a 4.7. egyenletnél, hogy átviteli függvényekből számol arányszámot. Ezt úgy teszi, hogy a kettő elemenkénti szorzatának az abszolútértékét elosztja kettő amplitúdónégyzetének a szorzatával.

$$\mathrm{FRAC} = \frac{\left| \sum_{j=1}^{N_f} \left[H_1(\omega_j)^H \times H_2(\omega_j) \right] \right|^2}{\sum_{j=1}^{N_f} \left[H_1(\omega_j)^H \times H_1(\omega_j) \right] \sum_{j=1}^{N_f} \left[H_2(\omega_j)^H \times H_2(\omega_j) \right]} \tag{4.7}$$

Előnye ennek a metódusnak, hogy rezonanciacsúcsok változására érzékeny lesz, így az értéke le fog csökkenni bármilyen eltolódásra. Ez a metódus szintén $[0, 1]$ között korlátos eredményt ad, így nem kell átskálázni, vagy komplexebb limitet meghatározni, mivel egyfajta arányszámként kapjuk meg az eredményt. Előnye még, hogy konstans amplitúdóeltolódásra érzéketlen, ami más felhasználások esetén ez néha probléma lehet. Az amplitúdókülönbség kompenzálására az MFRAC (modified FRAC) metódust fejlesztették ki, ami a 4.7. egyenletet a két jel teljesítményarányával egészíti még ki.

Ahhoz, hogy ez a mérési eljárás alkalmazható legyen az adott problémára, a korábbi algoritmusokkal ellentétben szükséges egy referencia átviteli függvény. Intuitívan, ehhez feltételeznünk kell, hogy a mérés kezdetekor az elrendezés még hibamentes. Így az első mérésekből számítható egy referencia, amit a további részeredmények összevetésére lehet majd használni. Ezen felül a mérőpad belső jeleinek hiányában, arra sem kell figyelnünk, hogy a különböző mérési pontok eltérő átviteli függvényei következtében a FRAC arányszáma lecsökkenne. A 2.4. alszakaszban leírtak alapján a monitorozórendszer ugyanazt az átvitelt vizsgálja egy mérés alatt, így ez a metódus is alkalmasnak bizonyulhat az anomália detektálására.

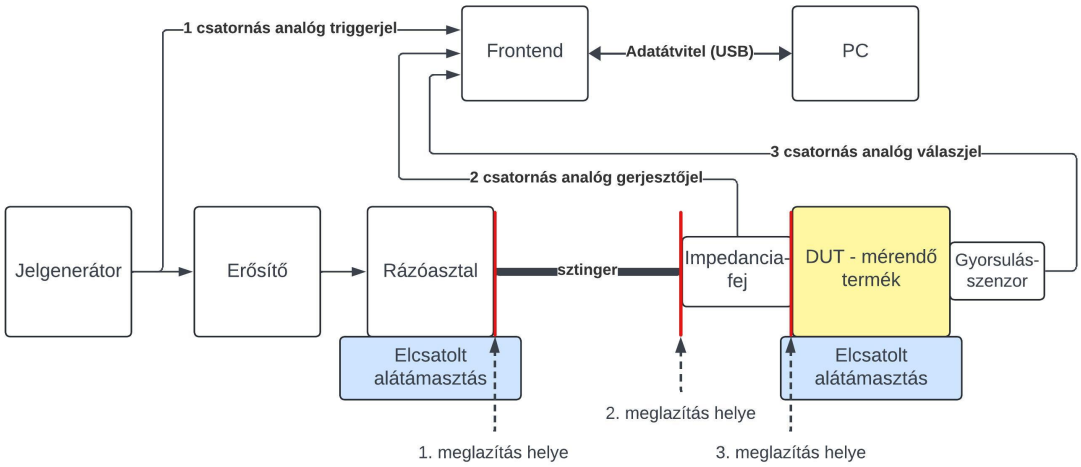
5. fejezet

Detekciós eljárások összevetése valós méréseken

A következőkben a 4. fejezet különböző algoritmusait hasonlítom össze, egy általam felépített mérési környezet segítségével, ahol háromféle mérési hibát viszek bele ebbe a rendszerbe és ezek kimutathatóságát vizsgálom.

5.1. Tesztmérések

A tesztmérések alapvető célja, hogy reprezentatív környezetet hozzunk létre, ahol a közel valós esetek a lehető legjobban reprodukálhatóak és vizsgálhatóak. Ahogy azt korábban a 2.4. szakasz részletezte, egy ehhez hasonló elrendezést alkottam meg (5.1. ábra), annyi változtatással, hogy a külső mérőpadot nem építettem bele, tekintve, hogy fizikailag nincs befolyással a monitorozó rendszerre és a laborban gyakran használatos eszköz, így ritkán is elérhető.



5.1. ábra. Tesztmérés elrendezésének blokkdiagramja és bevitt hibák helyei.

A választott forntend egy hordozható Head Acoustics hatcsatornás berendezés, ami tökéletesen megfelelt erre a célra. Másik előnye, hogy a laborban ennek a cégnek a termékei széleskörűen használatosak, emiatt nem szükséges újabb betanulás a használójának. További előnye, hogy a mérési fájlokhoz hozzá lehet férni és fel lehet dolgozni, mivel készült

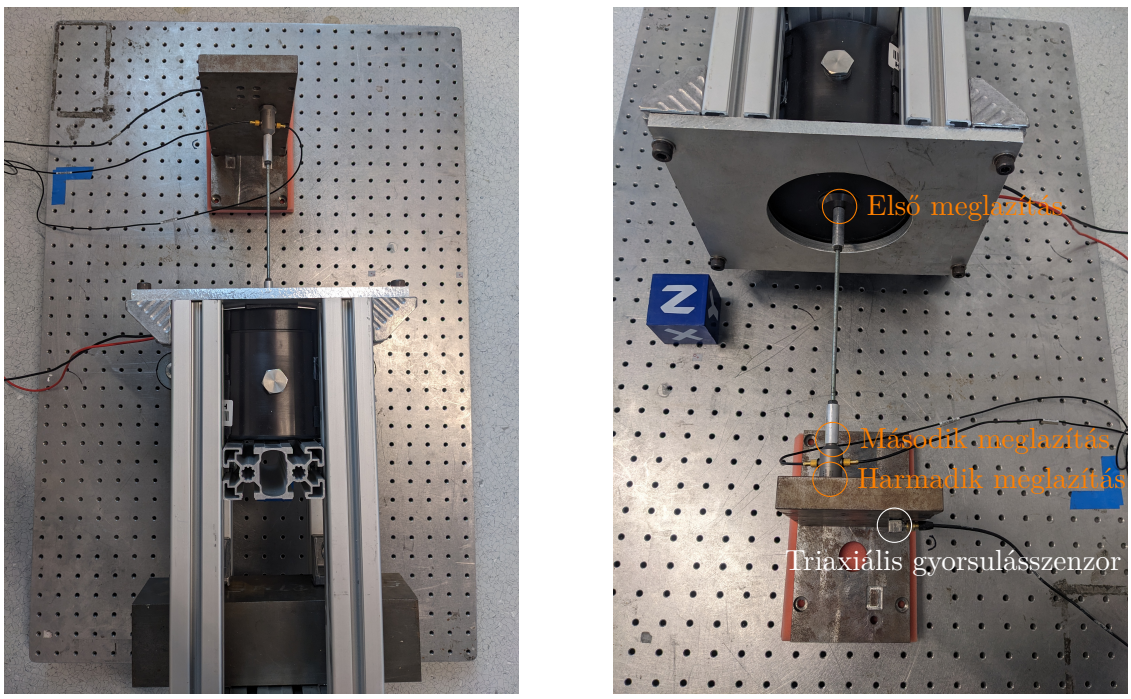
hozzá Python API (Application Programming Interface), aminek segítségével a további jelfeldolgozás elvégezhető.

Az ideális mérés mellé három hibatípust vettem fel, amik a jellegzetesen előforduló helyekre összpontosultak. A meglazításokat külön-külön végeztem el, így sosem volt egyszerre kettő eset aktív, hogy jobban lehessen lokalizálni az eredményeket.

1. Ideális esetben minden szorosan kapcsolódva van és az átvitel tekinthető lineárisnak a vizsgált tartományban.
2. Első meglazítás: közvetlenül a rázóasztal és a sztinger közötti menet kilazítása.
3. Második meglazítás: a sztinger és az impedanciafej közötti menet meglazítása.
4. Harmadik meglazítás: az impedanciafej és a mérendő test közötti ragasztás meglazítása, megtörése.

Nem minden vizsgálandó metódus független a gerjesztő jel típusára emiatt ebben az esetben csak chirp, azaz sepert szinuszos gerjesztést használtam, 100 Hz és 10000 Hz között lineáris frekvencia-eloszással és 5 másodperces lefutási időintervallummal. Sok esetben használnak fehérzajos gerjesztést és ezen belül is reprodukálható fehérzajt is, amit szintén vizsgálni fogok, viszont az a spektrális laposság esetén nem ad használható eredményt.

Az 5.2. ábra mutatja a véglegesen megvalósított elrendezést, amin bejelöltem a lazítási pontokat is. A jobb ábrán látható még egy kék műanyag kocka x-y-z feliratozással. Ennek az orientációja mutatja a globális koordinátarendszert. Ezt azért szokták használni, mert a felerősített szenzoroknak is van saját, kialakításukból adódó koordinátarendszerük. Hogy a különböző jelek összemérhetőek legyenek, kell választani egy globális koordinátarendszert, amihez a felvevőszoftverben állítani kell az egyes jeleket. Jelen esetben az impedanciafejhez állítottam az X irányt, viszont a triaxiális gyorsulásszenzornak ez a Z iránynak feleltethező meg, amit módosítottam is a szoftveren belül ($X \to Y$, $Y \to Z$ és $Z \to X$ módosítások megtartva a jobbkéz szabályt).



5.2. ábra. Mérési elrendezés két oldalról a mért tárggyal.

5.2. Teszttárgy

A hiba lokalizálhatóság szempontjából a teszttágynak egy egyszerű testnek kell lennie. A célja az egyszerűségnek, hogy a test lineáris viselkedésű legyen és ne befolyásolja a mérési eredményeket komplex viselkedése miatt ismeretlen tényezők. Így, amikor a hibát belevisszük a rendszerbe, akkor pontosan tudni fogjuk, hogy a jelenség miből adódik. Rövid körbenézés után a laborban, két nagyobb nem használt vaslapot találtam, amiket kényelmi okokból összecsavaroztam (hogy megálljon magába és az imedanciafej könnyen ráerősíthető lehessen) amit a 5.3. ábra mutat. Először két menetet kellett fúrnom az egyik vaslap végébe, ahol a másikon két lyuk már található volt. Az összecsavarozás esetleges hibapont is lehet, így fontos, hogy jól meghuzzuk ezeket. Végül összecsavarozva egy 'T' alakú testet kaptunk, mely két öntöttvas elemből épül fel. Ennek következményeképpen a tárgy tekinthető lineárisnak és anyagi minőségéből fakadóan a csillapítási tényezője igen kicsi, tehát az első módusa is várhatóan a kHz-es tartományba esik majd. A tömege 4.9 kg, ami a különböző karosszéria elemek nagyságrendjébe esik.

5.3. ábra. A mért tárgy oldalról.

5.3. Mérések menete

5.3.1. Előkészítés

Először a fizikai elrendezést állítottam össze, ami a rázóasztal egy gumitalpas vázba helyezésével kezdődött. Ez lehetővé tette, hogy vízszintes irányba stabilan lehessen gerjeszteni egy testet. Ez a váz ehhez a rázóasztalhoz készült (a laborban), még korábbi mérési elrendezésekhez. A váz végébe beleerősítettem egy nagy tömegű vastömböt is, ami azt segítette, hogy a shaker stabilabban, közel ideálisan merev, de környezetétől elcsatolt testnek legyen tekinthető. Ezt követően a mérendő tárgy alá helyeztem egy kellően stabil szivacsot. Mivel viszonylag nehéz a test, emiatt egy relatív keményebbet használtam stabilitás növelésének céljából. Ezt követően a sztinger egyik végét beleerősítettem a rázóasztal mechanikai kimenetére, a másik végére pedig rászereltem az impedanciafejet a megfelelő orientációval, tehát a nagyobb csatlakozási felülete a mérendő test irányába nézett. Végül a mérendő testet óvatosan odahelyeztem az impedanciafejhez, amit olyan pozícióban erősítettem fel,

hogy lehetőleg semmilyen szimmetriapontba ne kerüljön a gerjesztési pont. Ezzel próbáltam elérni, hogy minél több módusalak gerjeszthető legyen. A testet és az impedanciafejet pillanatragasztóval erősítettem fel, amihez ragasztó-aktivátort használtam a gyorsabb és erősebb megragadás érdekében. Ezalatt a procedúra alatt, a legfontosabb amire figyelnem kellett, hogy oldalirányú erők ne lépjenek fel ragasztás után, tehát nyugalmi állapotban kell lennie mind a sztingernek, mind a mérendő tárgynak. Végül a triaxiális gyorsulásszenzort felerősítettem a tárgy másik oldalára. Mivel kicsi a tömege, ezért ehhez viasz elegendő volt.

Az elektronikai eszközök és kábelek elrendezésével folytattam a munkálatokat. Az erősítőt és a frontendet konnektorba dugtam, a mobil jelgenerátor kimenetét pedig az erősítőre csatlakoztattam. A szenzorokat a frontend bemenetére kötöttem és a jelgenerátor jelét egy 'T' csatlakozóval szintén bekötöttem. Végül a számítógépet, amin a vezérlő szoftver fut, USB-vel összekapcsoltam a frontenddel.

Miután a fizikai elrendezés felépült, már csak a szoftveres beállítások és kalibrációk maradtak hátra. Először a számítógépen futó vezérlésért és felvételért felelős szoftveren be kellett állítanom, hogy milyen típusú bemeneteket várjon. Ehhez a konkrét mérőeszköz megadható és ebből már tudni fogja, hogy milyen mennységekkel kell számolnia ami rendre [V], [N], [m/s^2] $\times$ 4. A hozzájuk tartozó nominális érzékenységek is automatikusan betáplálódtak. Ezt követően a jelgenerátoron be kellett állítanom a korábban részletezett sepert szinuszos gerjesztés paramétereit, majd a megfelelő erőjelek beállításához az erősítő kimeneti jelét elkezdtem növelni úgy, hogy közben végig figyeltem a szenzorok csúcsértékeit. Nagyjából 2 N csúcsértékre állítottam be a kimenetet.

Végül ahhoz, hogy a szenzorokat kalibrálni tudjuk a mérés előtt, le kell szerelni őket és a kalibráló eszközre helyezni. Majd a kalibráció után vissza kell szerelni őket az eredeti helyükre.

5.3.2. Kalibráció

A szenzorkalibráció fontos része a mérés előkészületeknek. Az erő- és gyorsulásszenzorok gyártói általában adnak a termékről egy megbízható kalibrálási információt tartalmazó dokumentumot, ami vagy egy érzékenységi értéket, vagy pedig egy grafikont tartalmaz, amin látható, hogy a frekvencia függvényében miként változik az érzékenysége az adott szenzornak. Normális körülmények között a szenzorok jellemzői meglehetősen stabilak maradnak, viszont szélsőséges behatások vagy durva kezelés (egy gyorsulásmérő leejtése, erőmérő cellák túlterhelése) miatt előfordulhatnak eltérések a viselkedésükben. Ezenkívül az erősítők és szűrők befolyásolják a számítógépbe táplált végső jelet, így egy jó, megbízható kalibrációnak ezeket a berendezéseket is tartalmaznia kell. Ezen okok miatt érdemes, hogy minden mérés előtt újra kalibráljuk a szenzorokat. Sokfelé kalibrációs technika létezik, a szenzor típusától és felhasználási területtől függően. Én csak a mérés szempontjából fontosabbakat részletezem a következő alfejezetekben.

5.3.2.1. Gyorsulásszenzor kalibráció

A gyorsulásszenzor alapvető kalibrálásához szükségünk van egy másik frissen kalibrált eszközre, melynek tudjuk a pontos érzékenységét. Szokványosabb metódus [13], ha a kalibrált eszköz egy jelgenerátor, ami adott frekvencián adott jelerősségű szinuszt tud produkálni. Ennek értéke többnyire 1000 Hz-en 1g szokott lenni, de fajtája válogatja. Egy másik hasonló megvalósítás, ha rendelkezésünkre áll egy másik gyorsulásszenzor, amiről tudjuk, hogy kalibrált. Ha mindkettőt ráhelyezzük egy szinuszos jelgenerátorra és leolvassuk a két értékét, átszorozva megkaphatjuk a valós érzékenységét a számunkra kérdéses szenzornak. Ezek a metódusok egy fontos feltételezésen alapulnak, mégpedig, hogy a szenzor állapo-

ta frekvenciafüggetlenül változik és így elegendő csak egy adott frekvencián vizsgálni a jeleiket.

5.3.2.2. Erőszenzor kalibráció

Az erőszenzor kalibrációja is hasonló típusú, mint a fent említett metódusok, annyi különbséggel, hogy ezeket szintén gyorsulásszenzorral kalibrálják, aminek az érzékenysége kalibrálva van és ismert [13]. Az erőszenzorokat többnyire impulzuskalapácsokban helyezik el, így ennek a mérési metódusa Newton második törvényén alapszik ($F(t) = ma(t)$). Ezt úgy oldják meg, hogy egy ismert tömeget felkötnek valahova lazán és ráteszik a kalibrált gyorsulásszenzort. Ezután a kalapáccsal megütik ezt a tömeget lehetőleg a gyorsulásszenzor irányába, ami után leolvasható a csúcsértéke mind az erőjelnek, mind pedig a gyorsulásjelnek. Az ismert tömeg segítségével könnyedén kiszámolható a szenzor valós érzékenysége.

5.3.3. Mérések elvégzése

A kivezérlés beállítását és szenzorok kalibrációját követően a mérés megkezdődhet. A szoftverben beállítottam az úgynevezett pre-trigger értéket ami a jelgenerátor jelét vizsgálja és ha elér egy bizonyos amplitúdó értéket akkor onnan még 1 másodpercet visszaszámol és beleméri, így garantáltan az egész periódust fel tudom venni. A korábban felsorolt (5.1. szakasz) hiba típusokból mind mértem 5-5 felfutást.

A különböző meglazításokat sajnos nem volt egyszerű tökéletesen reprodukálhatóan megvalósítani, így pont csak annyit lazítottam mindenhol, hogy az anyacsavar épphogy ne feszüljön már. Ezzel is próbáltam törekedni a minél valósabb eredmények szimulálására. A ragasztott rész volt a legnehezebb, ugyanis, ha megtörtem a ragasztót, akkor tulajdonképpen a kapcsolat teljesen megszűnt, emiatt ez inkább egy boolean-i esetnek volt tekinthető. Végül sikerült egy köztes állapotot elérnem, ahol még ragasztottnak volt tekinthető az impedanciafej, de mégis laza volt már a kapcsolat, így már nem volt tökéletesen kontaktusban. A mérési folyamat ezek miatt hasszadalmas volt, de a végén sikerült a megfelelő eredményeket elérnem.

5.4. Mérési eredmények

Először kiválasztottam egy-egy reprezentatív méréseredményt minden esetből, így négy esetet hasonlítottam össze. Mivel az RDC és a FRAC kiértékelési metódusoknál csak egy mérőszámot kapunk egy adott mérésre, emiatt ezek eredményeit statisztikailag, az összes elvégzett méréssel együtt értékeltem.

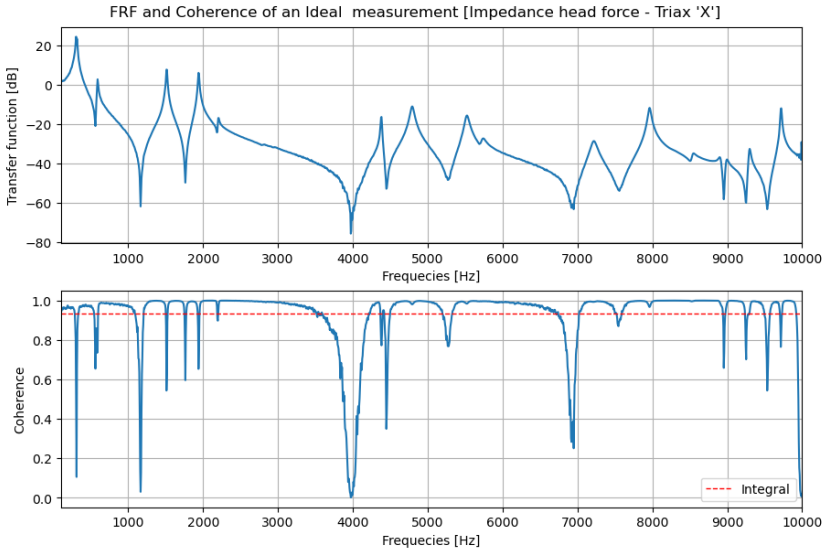
5.4.1. Koherencia

A koherencia vizsgálatával kezdtem a kiértékelést. A gerjesztőjel miatt 100 Hz és 10 kHz között vizsgáltam az eredményeit a négy különböző esetre. A koherenciát az impedanciafejben levő erőjel és a testre helyezett gyorsulásszenzor megegyező irányba néző komponense között számoltam ki. Emellett átviteli függvényeket is ábrázoltam, mivel a kettő eredménye összefügg. Végül intuitívan egy mérőszámot számoltam ki egy-egy mérésre és ábrázoltam az eredmények függvényében:

- Integrál: A koherencia fölötti terület az 1 étrék alatt, átskálázva $[0, 1]$ intervallumra. Ez valójában a koherencia értékek átlagát határozza meg.

Az 5.4. ábra mutatja az ideális mérés eredményét. Látható, hogy a koherencia azokon a helyeken lecsökken, ahol az átviteli függvénynek minimumai vannak (például 4 kHz-en vagy 7 kHz környékén). Ez elvárt viselkedése a metódusnak a korábban részletezett okoból. 4 kHz körül látható egy nagyobb beesés és ekörül kicsit zajosabb is a görbe, viszont nincs rezonanciája ezen a területen emiatt ez nem is problematikus. Bizonyos helyeken a rezonanciákban is láthatók kisebb beesések (például 2 kHz-nél), ennek mértéke nem jelentős és az átlagolás növelésével csökkenthető.

Látható még„ hogy a számolt átlag is 1 közeli értéket vesz fel, ami az elvárásoknak szintén eleget tesz.



5.4. ábra. Az ideális elrendezés mért átviteli függvénye és koherenciája.

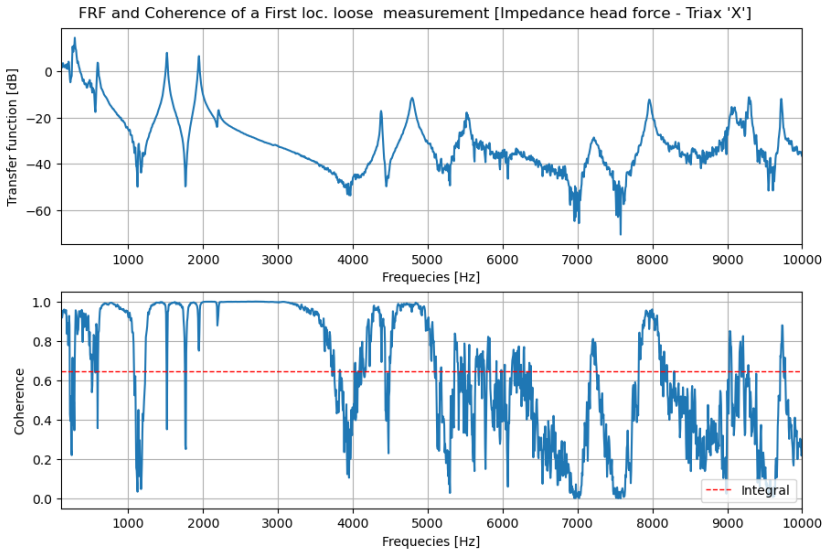
Az 5.5. ábrán látható az első meglazítás eredménye. A koherencia 5kHz-től jelentősen beesik, ami nemilearitás jelenlétére enged következtetni, tehát a szintetikus hiba bevitele sikeres volt. Ezen a szakaszon az átviteli függvény eredményei már nem tekinthetőek reprezentatívnak a rendszer szempontjából. Így, ha ilyen típusú méréssel találkozunk, meg kell ismételni a kontaktus hiba kijavítása után. Ha erre nincs lehetőség, akkor csak 5 kHz-ig lehet vizsgálni az eredményeket. Például látható, hogy 9 kHz környéki rezonanciák már nem mind fellelhetőek az átviteli függvényen.

Ez az eredmény nagyon mérési elrendezés függő, emiatt konkrétabb általánosítást nem lehet levonni más testek vizsgálatához. Egyedül azt látjuk, hogy a vizsgált spektrum fele nem tekinthető relevánsnak. A számolt átlag értéke is jelentősen lecsökken.

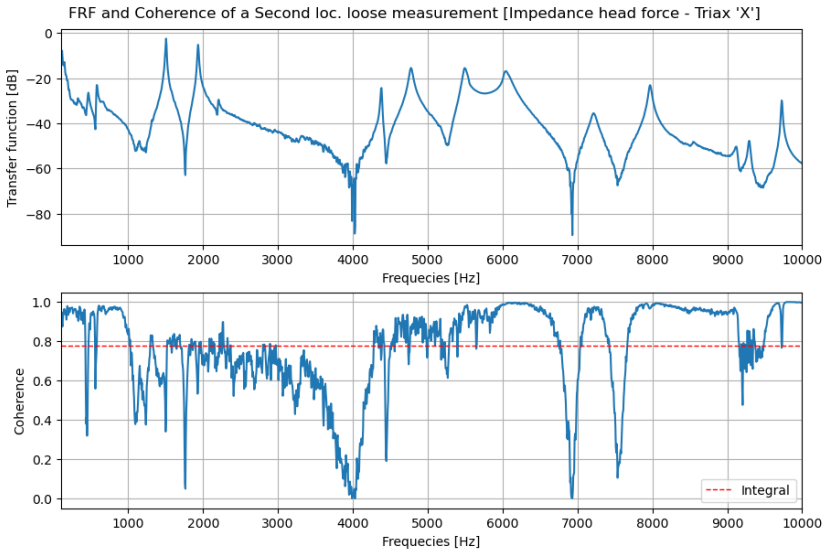
Az 5.6. ábra és az 5.7. ábra hasonló eredményeket mutat. Mindkét bevitt hiba az impedanciafej körüli meglazításokból ered és látható, hogy ez inkább az alsó frekvencia tartományt befolyásolja, tehát használható eredményeink inkább 5 kHz-től vannak. Kifejezetten az 1000 Hz alatti rezonanciák jelentősen torzulnak, így ez a tartomány nem használható későbbi kiértékelésre.

Az átlag ezekben az esetekben is lecsökken, viszont látható, hogy az eredmények meglehetősen eltérőek. Érdekes még, hogy a második helyen meglazított eset átviteli függvényén látható, hogy az 5800 Hz-en levő antirezonancia eltűnik és helyette 6 kHz-en egy rezonancia megjelenik. Ez a test valamilyen szintű modális viselkedés változására enged következtetni, mivel a koherencia nagy ebben a sávban.

Összességében a koherencia beszakadás-helye a meglazítások hatására dirverz, és konkrétabb következtetést nehéz levonni. Általánosságban kijelenthető, hogy az átlag fel-



5.5. ábra. Az első helyen meglazított elrendezés mért átviteli függvénye és koherenciája.

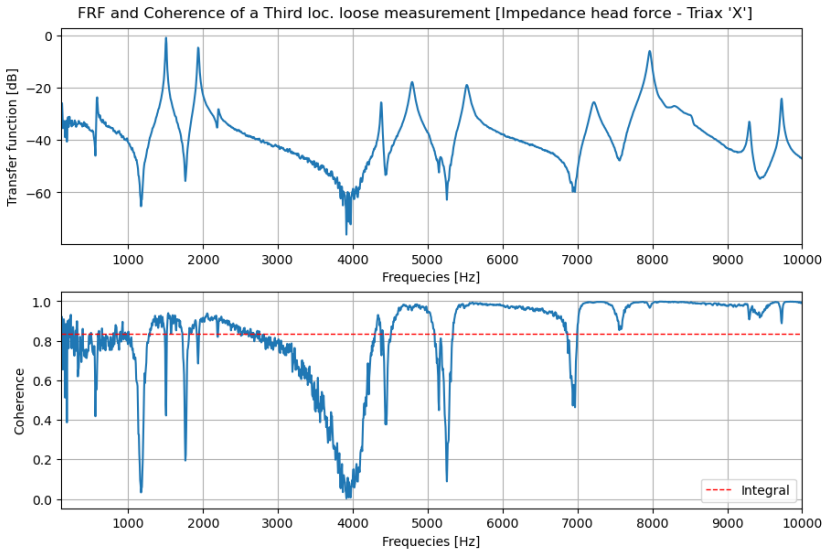


5.6. ábra. A második helyen meglazított elrendezés mért átviteli függvénye és koherenciája.

használható lehet a rendszer teljes állapotára adott linearitás mérőszámának. A határérték meghatározása viszont nehézkes lehet, mivel minden mérési elrendezés esetén más mértékű lecsökkenés jelentkezett, aminek a hibahatára más és más volt. Ennek a metódusnak a fő előnye, hogy nem gerjesztésspecifikus, így mind fehérzajra, mind szinuszos gerjesztésre is értelmezhető eredményt fog adni.

5.4.2. Spektrális laposság

Folytatva a spektrális laposság vizsgálatával, a végleges metódus ennek kiszámolására a következő: csak a gyorsulásszenzor gerjesztés irányú komponensét vizsgáltam, és a felvett bemeneti feszültségjelből pedig az adott gerjesztő frekvenciát számoltam. Ezután a



5.7. ábra. A harmadik helyen meglazított elrendezés mért átviteli függvénye és koherenciája.

gerjesztő frekvencia körüli tartományt kivágtam az adott spektrumból, és az így kapott spektrumra kiszámoltam a spektrális laposságot.

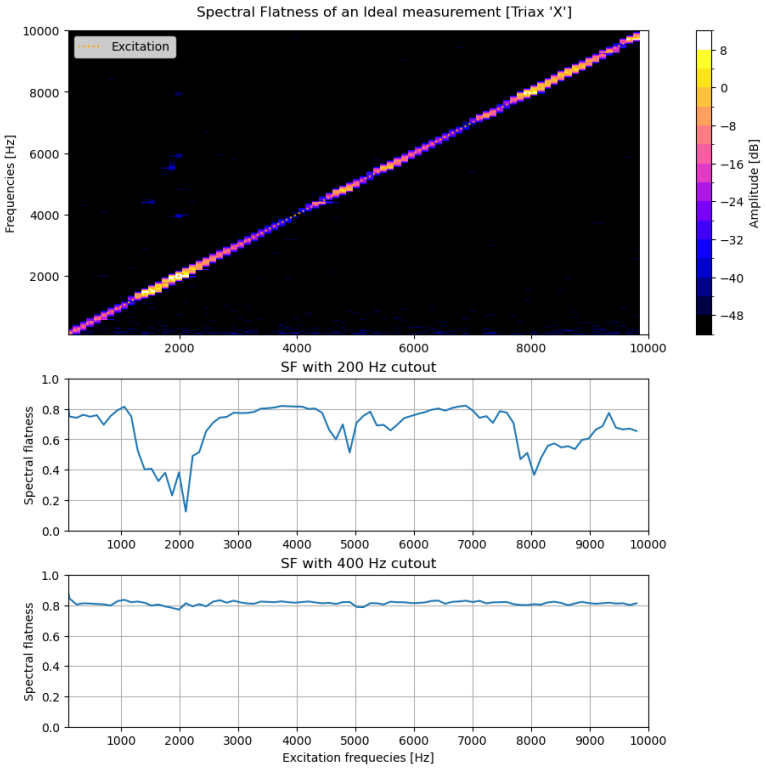
Hogy jobban értelmezhető legyen a kapott eredmény, kiszámoltam és megjelenítettem a jel spektrogramját is. Így minden egyes ablaknak láthatjuk a spektrumát és a hozzá tartozó spektrális laposság értékét is. A kiszámolt gerjesztés frekvenciájának függvényében ábrázoltam az eredményeket, mert a gerjesztés lineáris eloszlású chirp jel volt, emiatt a gerjesztőjel frekvenciája egyenes arányos a mérés kezdete óta eltelt idővel.

Az 5.8. ábrán láthatjuk, az ideális esetet, ahol semmilyen meglazítás nincs még jelen. Először 200 Hz-es tartományt vágtam ki a gerjesztés körül, tehát fölötte és alatta is 100 Hz-et nem számoltam bele. Látható, hogy ez nem volt elegendő és a gerjesztés még átszivárog a válaszban. Ezután megdupláztam a kivágás mennyiségét, így 400 Hz-et vágtam ki és ez már elegendőnek bizonyult. Ez is azt mutatja, hogy igen érzékeny ez az elemzési metódus. Végül, így végig 0.8 körüli értéket kapunk, ami nagyságrendileg kellően lapos spektrumra ad következtetést.

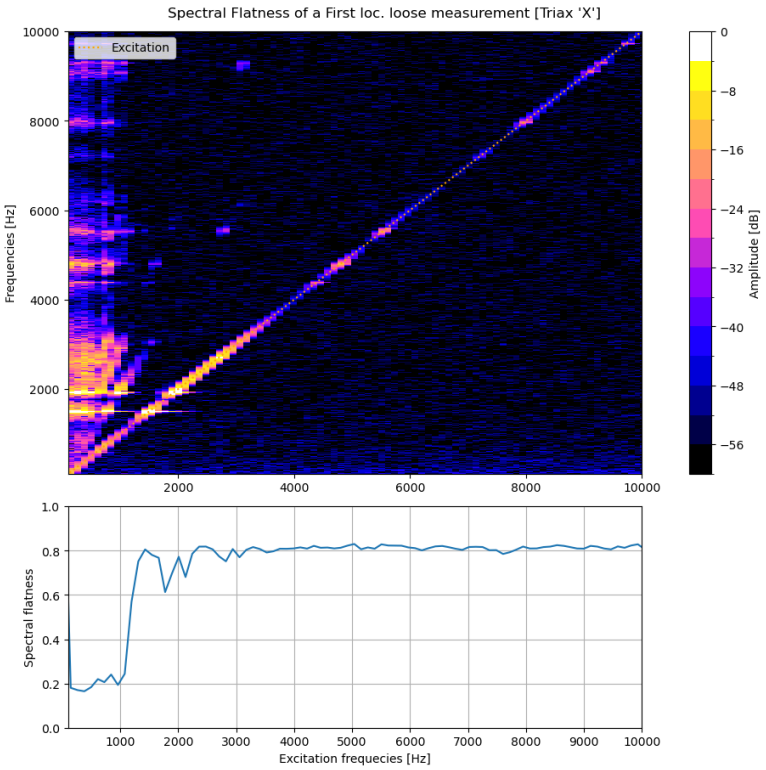
Az első és második helyű meglazításnál hasonló eredményeket kapunk, ahogy azt az 5.9 és 5.10. ábra mutatja. Körülbelül 1000 Hz-ig a bemenő erő olyan tartományt gerjeszt, ahol kontakt nemlinearitásból származó viselkedés figyelhető meg. Ez jelentősen torzítja a válasz jelalakját, ezzel felharmonikusokat víve bele. A spektrális laposság ennek a jelenlétét kimutatja, és látható, hogy egészen 0.2 körüli értékre lecsökken ezen a tartományon.

Az utolsó, harmadik ponton bevitt meglazítás spektrális laposság minimuma a korábbiakkal ellentétben 7500 Hz-en figyelhető meg (5.11. ábra). Hogy jobban megértsük ezt a viselkedést egészen a Nyquist frekvenciahatárig bővítettem a spektrogram limitjét (a mintavételi frekvencia 48 kHz volt, így 24 kHz a felső frekvenciakorlát). Látható, hogy ezzel a határral épphogy rögzíteni tudtam a létrejött felharmonikus egyik komponensét, így látható, hogy sok esetben érdemes minél nagyobb mintavételi frekvenciát használni. Mivel a spektrális laposság kifejezetten a periodikus jelekre érzékeny, így érthető, hogy az arányszám értéke egészen 0.1-ig lecsökken.

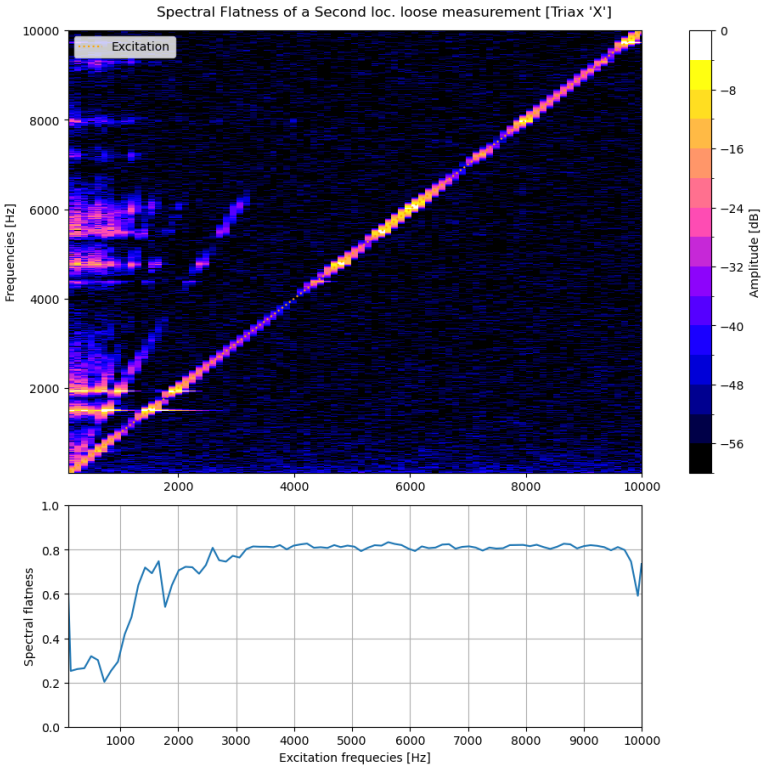
Összességében, ha a minimumokat vizsgáljuk, akkor könnyedén kimutatható a nemlinearitás, viszont mivel nagyon érzékeny mérőszámról van szó, emiatt, ha nincs kellően kivágva vagy nem kellően nagy felbontású az ablak, akkor esetlegesen lecsökkenhet az értéke akkor is, ha a mérés teljesen jó volt. A gerjesztést teljesen figyelmen kívül hagyásából



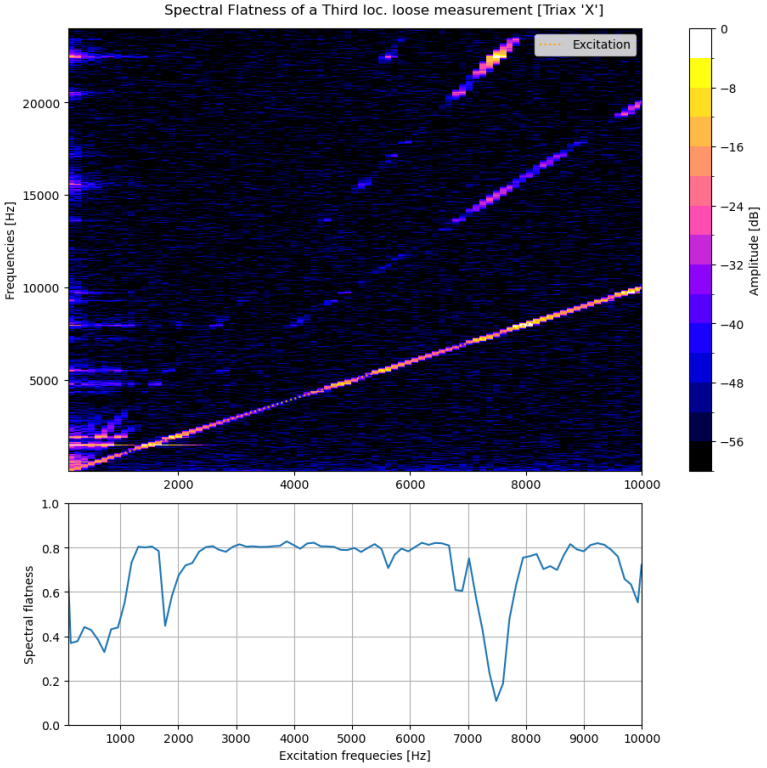
5.8. ábra. Egy ideális mérés spektrogramja és spektrális laposssága.



5.9. ábra. Egy első helyen meglazított mérés spektrogramja és spektrális laposssága.



5.10. ábra. Egy második helyen meglazított mérés spektrogramja és spektrális laposssága.



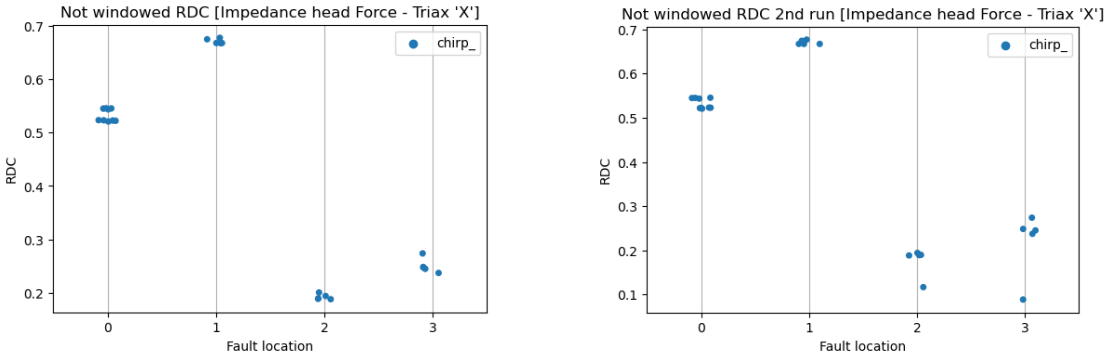
5.11. ábra. Egy harmadik helyen meglazított mérés spektrogramja és spektrális laposssága.

adódóan, a modális viselkedés változására teljesen érzéketlen. Másik hátránya ennek a metódusnak, hogy csak szinuszos (léptetett vagy sepert) jelekre lehet használni, emiatt a felhasználási területe korlátozott.

5.4.3. Randomized Dependence Coefficient

Az RDC kiértékeléséhez a korábbiakhoz képest inkább a statisztikai megközelítést választottam. Mivel ez egy viszonylag új eljárás (2013 [14]) két jel hasonlóságvizsgálatához, emiatt nincs sok irodalom a felhasználási területeiről. Végül három esetet vizsgáltam meg és ezeknek az eredményeit ábrázolva hasonlítottam össze, hogy melyik lehet a legalkalmasabb az anomáliák azonosítására.

Először a teljes mérések jelein végeztem el az RDC kalkulációját, bármilyen előprocesszálás nélkül. Az 5.12. ábra bal oldali grafikonja mutatja az első futtatás eredményeit. Az egyes pontok az egyes mérések eredményeit mutatják és az RDC-t egy mérésen belül hasonlóan a koherenciához a gerjesztő erő és a testen levő gyorsulásszenzor között számoltam. A négy csoport a négy mérési eset. Látható, hogy a legnagyobb egyezést nem is az ideális eset (0 oszlop) adja, hanem az első helyen levő meglazítás. Ezt érdekesnek találtam, ugyanis ez nem az elvárt eredményt adta, és mivel tudtam, hogy minden lefuttatásra más véletlenszerű projekciókat vesz, amit utána korreláltat, emiatt kiszámoltattam még egyszer az eredményeket. Az 5.12. ábra jobb oldali grafikonján látható, hogy tényleg eltértek az eredmények (például a második és harmadik helyen levő meglazításnál kaptunk olyan értékeket a mérésekből, ahol az érték leesett egészen 0.1-ig), de nem jelentősen, és ezt el is vártam, ugyanis a felhasznált cikkben arra hivatkoztak, legalább 20 eleműre választott projekciók esetén az eredmény kellően stabil lesz már. Sajnos ez nem oldotta meg annak a problematikáját, hogy a legnagyobb egyezést nem az ideális eset adta.



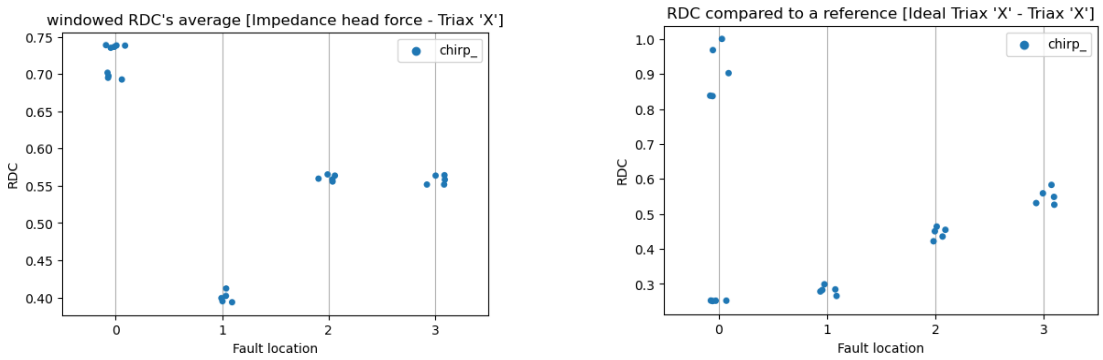
5.12. ábra. RDC kétszeres kiszámolása ugyanazokon a jeleken.

Ezt a gondolatmenetet folytatva, a második esetben hasonlóan álltam a jelek kiértékeléséhez mint a koherenciánál vagy a spektrális laposság számításánál és feldaraboltam kisebb ablakokra a mérendő jeleket, majd ezek eredményeinek átlagát vettem. Ennek az eredményeit az 5.13. bal oldali ábrája mutatja. Látható, hogy ebben az esetben a legnagyobb egyezést már az ideális esetek mutatták.

Látható még, hogy az ideális esetnél két csoportra oszlanak az értékek. Ez amiatt jöhetett létre mert az ideális esetből kétszer annyit mértem úgy, hogy ezeket kétszer 5 mérési sorozatra bontottam. Elviekben ugyanazt a mérést végeztem el mindkétszer, annyi különbséggel, hogy egy nap eltelt a kettő között. Ebből látható, hogy egy mérés tökéletes reprodukciója szinte lehetetlen.

Végül megpróbáltam különböző mérések ugyanazon jelein elvégezni a hasonlósági vizsgálatot. Ennek a motivációja a későbbi FRAC kiértékelésből ered, ahol szintén szükségünk van referenciajelre, amihez képest vizsgáljuk az eltéréseket. Az 5.13. jobb oldali ábra mu-

tatja ennek az eredményeit. Ebben az esetben a testre helyezett triaxiális gyorsulásszenzor jeleit hasonlítottam össze a legelső ideális mérés jelével. Összességében ez adta a legminőségibb értékeket. Az ideális esetnél a legnagyobb egyezést a korábbi kettő metódushoz képest itt találta a legnagyobbnak, bár ez abból a szempontból is érthető, mivel itt az egyik mérésnél saját magával lett összehasonlítva. Vágül látható, hogy a két külön időpontban elvégzett ideális méréscsoport még jelentősebben eltér egymástól, egyezést szinte nem is talál az RDC.



5.13. ábra. RDC eredményei ugyanazonok a jeleken átlagolást (bal) vagy referenciát (jobb) használva.

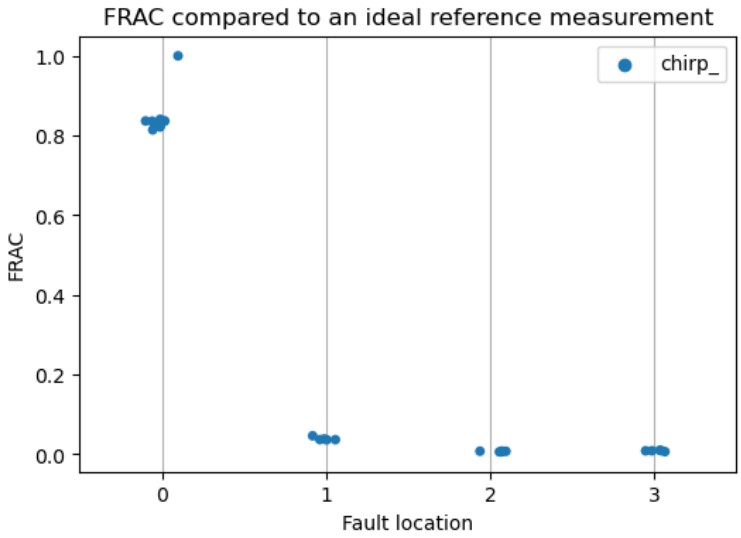
Összegezve, az RDC legfőképpen zajra érzékeny, és mind lineáris és nemlineáris összefüggéseket is képes kimutatni, ami sajnos a mi esetünkben hátránynak bizonyult, ugyanis ezekre emiatt érzéketlenebb. Ami szintén érdekes volt, hogy nagyon érzékenynek bizonyult bizonyos esetekben, így amikor két nagyon hasonló elrendezésű mérést hasonlítottam össze, nem talált egyezést, ami felettébb különös, és ez nem előnyös tulajdonság az anomáliák detektálásához. Azt is érdemes megjegyezni, hogy hasonlóan a spektrális laposséghoz, ez is csak szinuszos jelekre működik, ami felhasználás szempontjából jelentősen korlátoz.

5.4.4. Frequency Response Assurance Criterion

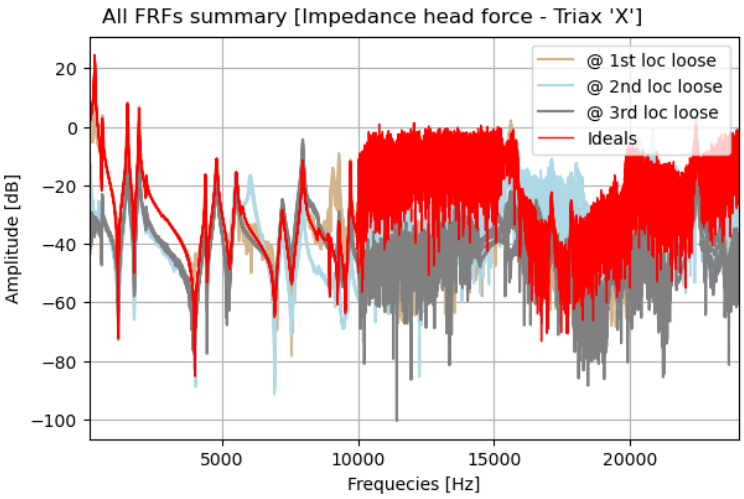
A FRAC kiértékelési módjához hasonlóan álltam hozzá, mint az RDC kiértékeléséhez, mivel ez is egy teljes mérésre, pontosabban két átviteli függvényre ad hasonlósági arányszámot. Látható az 5.14. ábrán, hogy az önmagával összehasonlított mérés egyedül, ami teljes egyezést mutat. A többi ideális mérésre viszont már csak átlagosan 0.8 értéket adott. Ellenben az RDC-vel, itt nem volt megfigyelhető az a két csoportra oszlás. Emellett az is látható, hogy bármelyik hibára már olyan jelentősen lecsökken az értéke, hogy 0 közeli értéket vesz fel.

A 5.15. ábrán - hogy jobban lássuk az eltéréseket - az összes mérés átviteli függvényeit ábrázoltam. A négy különböző esetet színezéssel választottam szét, hogy jobban megérthessük a kapott értékeket. Az ábráról leolvasható azonnal, hogy miért nem kaptam nagy százalékú egyezést: nem csak azt a tartományt vizsgáltam, ami gerjesztve volt, hanem a teljes rendelkezésre álló frekvenciatartományt. Ez időjeli összehasonlításnál (RDC) vagy azokban az esetekben, ahol csak egy jel frekvencia spektrumát vizsgálok (SF) hasznos, viszont átvitel számolásánál a nem gerjesztett területeken csak zaj lesz látható. Ez egyedül még a koherencia számolásánál adhat eltérő eredményeket, azonban abban az esetben az integrál és átlag értékek kiszámolása során határoztam meg ezeket a limiteket, így ez a hiba nem befolyásolta azok eredményeit.

Ezt követően a H1 becslő függvényemet úgy módosítottam, hogy csak a kiválasztott frekvenciatartományban adja vissza az értékeit (5.1). A *self* paraméter annyit jelent a limitek előtt, hogy ez a függvény egy osztály belső függvénye. Az általam megvalósított



5.14. ábra. Az egyes mérési lokációk FRAC analízise egy ideális méréshez véve.



5.15. ábra. Az összes mérés átvitelifüggvényei.

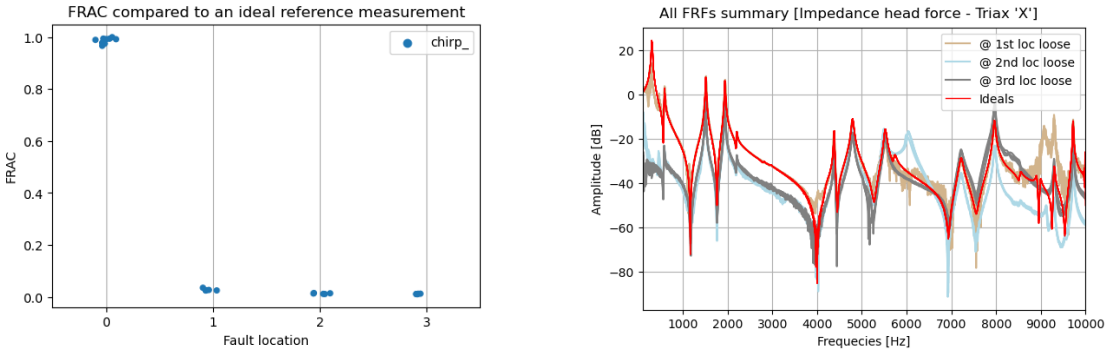
osztály fő motivációja, hogy a különböző kiértékelési metódusok során ne kelljen az általánosabb paramétereket folyton megadni, így ennek konstruáláskor például elég egyszer megadni a mintavételi frekvenciát, alsó és felső frekvencia limiteket, ablakméretet és egyéb belső paramétereket. Ezt követően a kiértékelő függvények ezeket szükség szerint fel tudják használni. A `valid_freqs` egy kételemű, úgynevezett *Tuple* típusú változó, aminek az első paramétere az alsó, a második pedig a felső frekvenciahatár. A mi esetünkben ez 100 és 10000 Hz. Látható, hogy két visszatérési értéke van a függvénynek (f és h1). Mindkettő azonos méretű *Numpy* tömb, ahol *h1* a kiszámolt komplex értékeket tartalmazza, az *f* pedig a hozzájuk tartozó frekvencia értékeket.

```
return f[(f>self.valid_freqs[0]) & (f<self.valid_freqs[1])],
      h1[(f>self.valid_freqs[0]) & (f<self.valid_freqs[1])]
```

5.1. lista. H1 becslő függvény frissített visszatérése, Python kód

Újra kiszámolva a FRAC értékeket, az ideális esetre sokkal nagyobb egyezést kaptunk (5.16. ábra), ami az elvárásoknak már eleget tesz. A többi bevitt hibák eseteire ugyanúgy

nagyon alacsony egyezést talált. Érdekes még, ha az átviteli görbéket jobban szemügyre vesszük, a második helyen meglazított eset átviteli függvényeit 6000 Hz-en újabb rezonanciát figyelhetünk meg úgy, hogy a koherencia is jó értékeket mutat ezen a területen (5.6. ábra). Az ideális esetnél 5800 Hz környékén figyelhetünk meg egy kisebb rezonanciát és valószínűleg ez mozdult el a 6000 Hz felé. Ez modális viselkedés változásra enged következtetni, és a koherencia elemzése során is felfigyeltem már erre.



5.16. ábra. Javított FRAC eredmények és hozzájuk tartozó FRF görbék, csak a gerjesztett frekvenciákon kiértékelve.

Összegezve, a FRAC nagyon pontos eredményeket ad, és a hibás esetek jól elválaszthatóak az ideális esettől. Az is kiderült, hogy az jelentősen befolyásolja az eredményt, ha nem a megfelelő frekvenciatartományt értékeljük ki. Ebből a szempontból érdemes lehet a kezelőfelületen az alsó és felső frekvenciahatárt megadhatóvá tenni, ha ezt a metódust választjuk a monitorozó rendszer fő metódusának. Végül látható, hogy csak átviteli függvények kiszámolása szükségesek a FRAC eredményekhez, így hasonlóan a koherenciához, ennek az eredménye is független a bemeneti jel típusától.

5.5. Eredmények összegzése

A vizsgált metódusokból a koherencia és a FRAC mérőszám bizonyult a leghasznosabbnak a pontosságát és a felhasználási területeit illetőleg egyaránt, ahogy azt az 5.1. táblázat is mutatja. A koherencia viszont a modális viselkedés megváltozására érzéketlen. Egyedül a FRAC, ami képes ennek kimutatására viszont az adott mérési elrendezésre is érzékeny, emiatt erre figyelni kell a későbbiekben. Annak ellenére, hogy kifejezetten a meglazításokból fakadó nemlinearitásokra fókuszáltam a próbamérések során, egyéb tulajdonságai is kiderültek az egyes metódusoknak felhasználhatóság szempontjából.

Egyik metódus sem igényel igazán komplex, sok időt igénybe vevő számolást, és skálázhatóságuk is könnyedén megoldható (tehát nagyobb adathalmazokon is használható). Emiatt a végleges metódus kiválasztásában ez a tényező, nem számottevő jelentőségű. Megjegyzésként, az eredmények kiszámolása során a leghosszabb időt az RDC vette idénybe, ami érthető is, mert egy lineáris egyenletrendszer sajátérték problémáját kell megoldani az utolsó lépésben, ezzel minimálisan növelve a futási időt. Emellett az összes közül az RDC adta a legkevésbé konzisztens eredményeket, emiatt ezt a metódust nem érdemes a végleges szoftverbe beépíteni. A táblázatból végül az is kiderül, hogy a legtöbb hibatípusra a FRAC metódus ad kimutatható eltéréseket, így erre ezt választottam a végső kiértékelési algoritmusnak.

Metódus	Alkalmazható jeltípusok	Modális változásra érzékeny	Nemlineáris torzulásra érzékeny	külső zaj jelenlétére érzékeny
Koherencia	független	nem	igen	nem jelentősen
Spektrális Laposság	csak szinuszos	nem	igen	nem
RDC	csak szinuszos	nem	valamennyire igen	igen
FRAC	független	igen	igen	nem

5.1. táblázat. Az egyes kiértékelési metódusok főbb tulajdonságai.

6. fejezet

Elfogadási küszöbértékek meghatározása, vizsgálat nagyobb adathamazon

Ebben a fejezetben a döntéshozatali szint meghatározásával foglalkozok, a kiválasztott metódusra a döntéshozatal miként oldható meg optimálisan és milyen megközelítések léteznek erre a problémára.

Ez a téma már széleskörűen körbejárt több mérnöki területen is, ugyanis minden automatizálási rendszerben van olyan fázis, amiben valamilyen szintű döntéshozatalt meg kell valósítani. Legyen ez egy gyártó sor minőségvizsgálata, egy útvonaltervező algoritmus optimális út megtalálása, egy zenefelismerő szoftver egyezés találása, vagy csak egy egyszerű PID szabályozó kör paramétereinek behangolása.

Mindegyiknek más-más szinten történik a döntéshozatala, sok esetben egy arra alkalmas ember hozza meg ezeket a döntéseket (például a szabályozó kör esetén), vagy hibafüggvények globális minimumát keresik meg különféle algoritmusok, máskor pedig korábban meghozott döntésekből, historikus adathalmazból, statisztikai következtetéseket vonnak le (akár gépi tanuló algoritmus segítségével).

Én ezek közül az adataim természetéből adódóan a statisztikai megközelítést választottam, amihez először a mérések mennyiségét kellett bővítenem, hogy áttekintőbb képet kaphassunk az eredményekről.

6.1. Mérési adatok bővítése

A mérési adatok bővítése során a fő szempont, hogy minél diverzebb elrendezésekből szerezzek adatokat, hogy elkerülhető legyen az, hogy az adott mérési elrendezésre specifikus legyen a döntési limit.

A FRAC metódus nagyon pontos eredményeket adott chirp jelre (amiből már egy használható adathalmaz el is készült, 6.1. ábra), emiatt fehérzaj gerjesztésre összpontosítottam. Az elrendezést különféle módszerekkel módosítottam, volt, hogy egy másik sztingert használtam, aminek a hossza és az átmérője is változott, volt, hogy alátétet tettem a csavarozási pontokra, volt, hogy egy nagyobb tömegű gyurmát tettem a mért testre és olyan is volt, hogy befogtam a test alját ezzel módosítva a peremfeltételeket. A pontos változtatásokat nem jegyzeteltem le, de nem is ez volt a lényeges szempont, hanem hogy minél több és sokfélébb méréseket állítsak elő.

Ezeken a különféle méréseken mind előállítottam és megmértem a korábban definiált négy esetet (5.1. szakasz) majd a kiválasztott metódussal (FRAC (4.4. szakasz) és kohe-

rencia (4.1. szakasz)) kiértékeltem az eredményeket. Végül összesen 165 mérést végeztem el és értékeltem ki.

6.2. Statikus döntési fa

Az első és egyben legegyszerűbb algoritmus a döntési fa [1]. Az alapvető működése egy komplexeb eldöntendő kérdésekből álló úgynevezett fa-rendszer, mely a mi esetünkben elegendő, ha egyetlenegy kérdésből áll. A kérdéssel pedig egy elegendően jónak tűnő határértéket határozunk meg ahol a limit fölött tekintjük elfogadhatónak a mérésünk állapotát. Ezt én 80%-os FRAC és átlag-koherencia hasonlósági szintnek választottam. Az eredmények függvényében természetesen ezt lehet tovább hangolni, de ez időigényes és nem garantált, hogy minden esetet lefedünk, vagy javulni fog az eredményünk.

6.3. Statisztikai megközelítés

A következő limitmeghatározáshoz a statisztikai megközelítést választottam, ahol feltételezzük, hogy a mérések minősége standard normális eloszlást követ. A gyártósorokon ezt sokszor használják és a termékek elfogadási határát az átlagtól a szórás háromszoros eltéréséhez teszik [8]. Ha ennél jobban eltér az adott termék, akkor azt hibásnak tekintik. Ez azt jelenti, hogy ideális esetben az összes termék (jelen esetben mérési pontok) 99.7%-a ezen a területen belül kell esnie. Ehhez a limitmeghatározáshoz csak a jó mérések átlagát és szórását számoltam ki és ennek az alsó limitét határoztam meg.

Ennek az előnye, hogy mindig specifikus lesz az adott mérésre, viszont ehhez átlagot és szórást kell számolni az első néhány mérésből. A vizsgált tesztméréseken a különböző csoportokból az első 5 ideálisat választottam ennek.

6.4. Instance-based learning, k-nearest-neighbors (k-NN)

Az első gépi-tanuló algoritmus, amit alkalmasnak találtam limit definiálásra, az a *k-nearest-neighbor (k-NN)* metódus [1]. A fő gondolata, hogy a döntéshozatalt úgy végzi el, hogy az éppen megkapott eredménynek kiszámolja a historikus adatoktól vett távolságait, és a hozzá legközelebbi csoportba sorolja. Ehhez látható, hogy szükséges egy felcímkézett korábbi adathalmaz, így ez inkább a statikus döntésben fog segíteni minket, hogy merre érdemes finomhangolni. Az algoritmusnak egy változóját lehet megadni (k), ami azt adja meg, hogy mennyi legyen a legközelebbi pont, ami alapján a döntést meghozza. Ennek növelésével legfőképpen az adathalmaz zaját lehet redukálni. Én egy tipikus értéket választottam, $k = 3$-at, tehát a legközelebbi 3 mérésipontból dönt az algoritmus. Abszolút limit kinyerése nem egyértelmű magasabb dimenziók esetén, de az én esetemben csak egy egydimenziós teret kell szétválasztani, így a limit definiálható ebből.

6.5. Klaszterezés, k-means algoritmus

A korábbi metódussal ellentétben, a klaszterezés [1] egy tipikus példája a felügyelet nélküli tanulásnak (unsupervised learning). Itt nem szükséges előre felcímkéznünk az adathalmazunkat, elég, ha megadjuk, hogy hány csoportra szeretnénk felosztani a teret (ez a k paraméter). Az algoritmus pedig a következő lépéseket hajtja végre, ha két osztályra szeretnénk bontani az adatainkat:

1. Először két pontot véletlenszerűen kiválaszt az adathalmazból.

2. Majd az összes többi pontot a hozzá közelebbi kategóriájába besorol, így létrejön az első szétválasztás és mindkét csoportnak kialakul a számtani közepe.

3. A kapott két számtani közép lesz az új viszonyítási pont és a csoportok ennek függvényében újra lesznek osztva. Iteratívan így egy újabb számtani közepe lesz az új kiosztásnak és ezekhez újraosztják a távolság függvényében a címkéket.

4. Ez a procedúra addig folytatódik, amig az újrakiosztásban változás van, egyébként pedig véget ér és létrejön a két osztály.

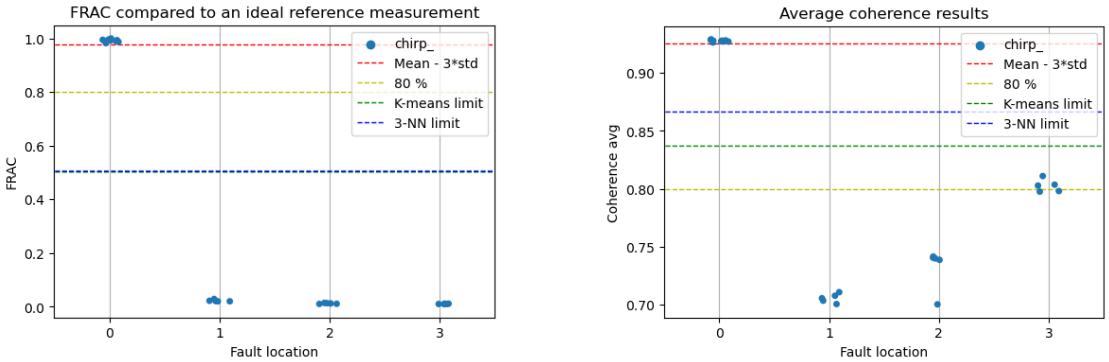
Könnyen belátható, hogy abban az esetben, ha a két osztály pontjainak a szórása kicsi, hasonló eredményt fogunk kapni a k-NN algoritmushoz, mivel a döntési limitet a két létrejött csoport (klaszterek) számtani közepétől vett távolság alapján fogja meghatározni.

6.6. Eredmények

Összesen négy különböző mérési elrendezést mértem meg fehérzajjal, kisebb-nagyobb módosításokkal. A fentebb leírt limiteket szintén ábrázoltam a diagramokon. Statisztikai vizsgálatot korábban az átlag-koherenciákon nem végeztem, és kíváncsi voltam, hogy milyen eredményeket ad a FRAC-hoz képest, emiatt annak az eredményeit is kiszámoltam.

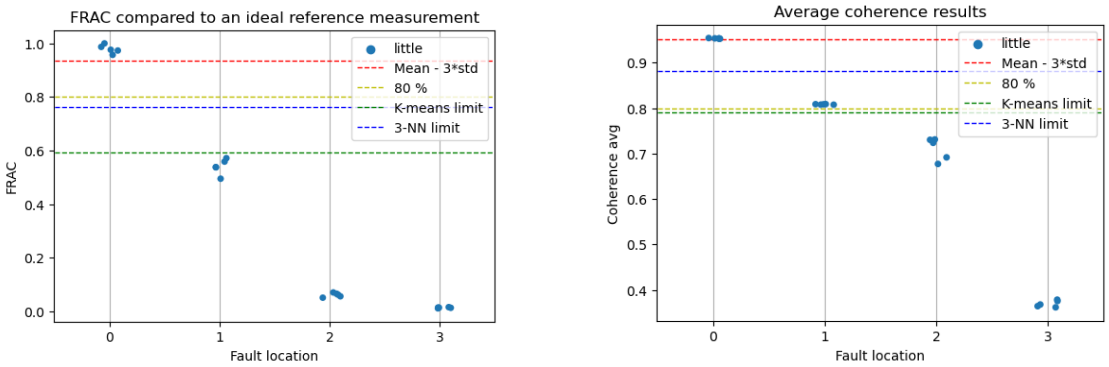
Összegezve látható, hogy a nagyobb adathalmazon az átlag-koherencia nem bizonyult megbízhatónak, kifejezetten a 6.3. ábra elrendezésen, ahol a legnagyobb egyezést nem is az ideális esetre érzékelte. Ezzel ellentétben a FRAC minden mérésre pontos eredményt adott, és nem elég, hogy az összes valamilyen szintű gépi tanulást alkalmazó algoritmus, de a statisztikai és a statikus limitek is 100% pontosságot adtak.

Tovább vizsgálva a határértékeket, a statisztikai megközelítés nagyon kis eltérést engedett, és azonnal nem is tud döntést hozni, emiatt a statikus jobban preferáltnak bizonyult. Esetleg ha nem háromszoros szórással számolunk hanem ezt megnöveljük, akkor jobb határértéket kaphatunk, viszont ez nem tűnt hasznosnak a korábban felsorolt okokból. Emellett a 6.1. ábra és 6.4. ábra 3-NN és klaszterező limitjei közeli értéket vesznek fel, ami az elvárásainknak eleget tesz a korábban részletezett okokból (6.5. szakasz). Néhány esetben a klaszterező algoritmus pedig jelentősen kicsire állította a limitet (6.2. ábra), ami azért lehetett, mert a hibás mérések szórása nagy volt. a 3-NN algoritmus minden esetben 80% alá számolta a limitet, és ez az érték az ideális eseteknek is elegendő eltérést enged, ezért ez optimális határértéknek tekinthető.

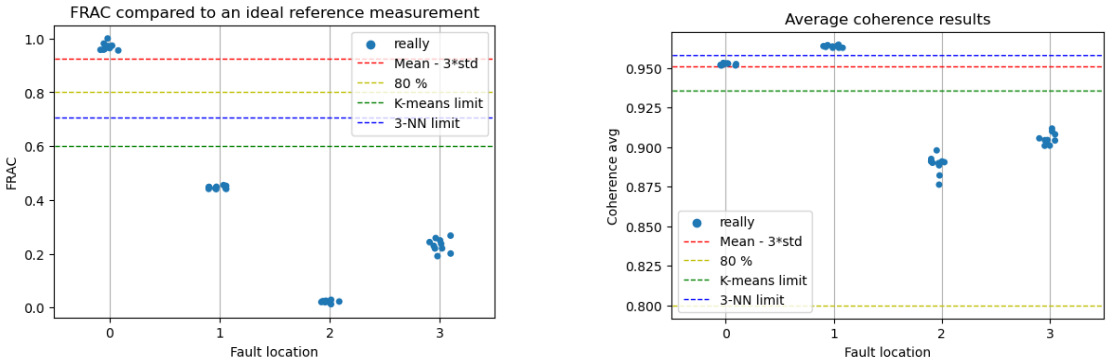


6.1. ábra. Chirp gerjesztés eredményei és különböző metódusok limitjei.

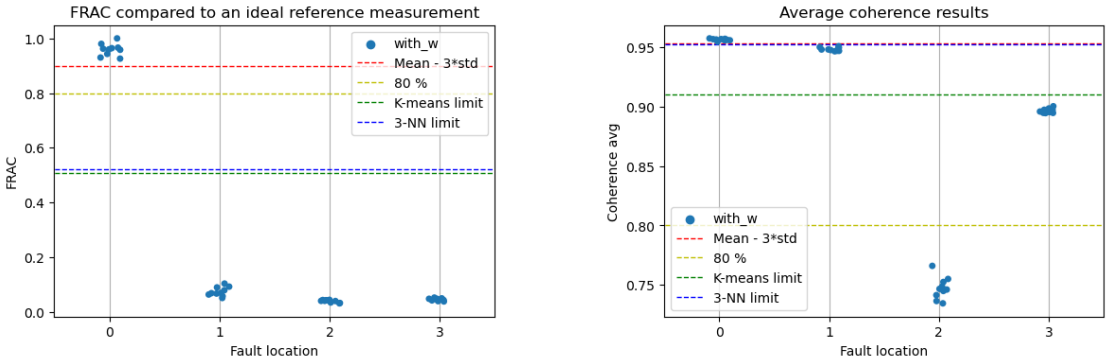
Végül érdekességként összehasonlítottam egy ideális mérést az összes elrendezéssel és egyértelműen látható (6.6. ábra), hogy eredő, a szoftverbe konstansként előre betáplált



6.2. ábra. Fehérzaj gerjesztés eredményei, "kis" meglazítással és különböző metódusok limitjei.

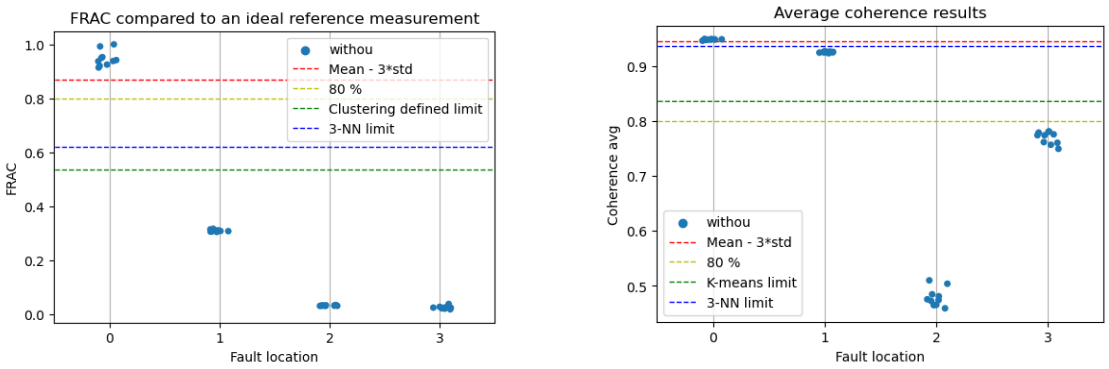


6.3. ábra. Fehérzaj gerjesztés eredményei, "nagy" meglazítással és különböző metódusok limitjei.

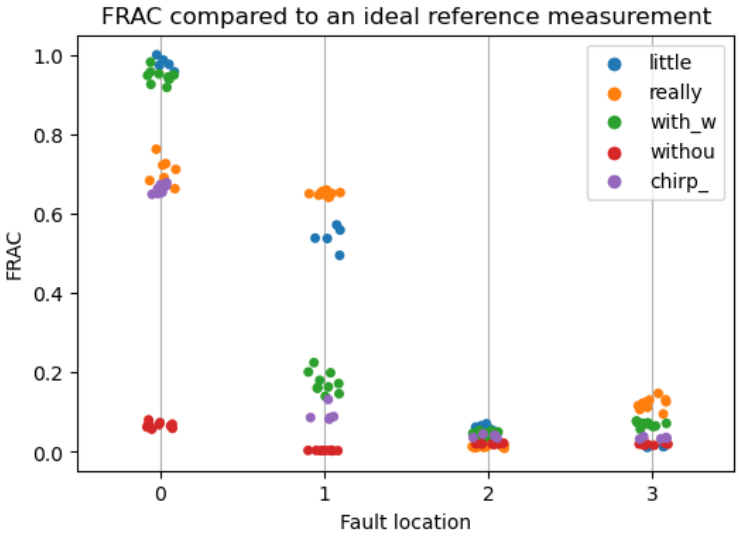


6.4. ábra. Fehérzaj gerjesztés eredményei, a sztinger kapcsolatoknál alátétet használva és különböző metódusok limitjei.

átviteli függvény használatára nem alkalmas a FRAC kiértékelés, mivel sokszor más ideális elrendezésekkel sem talált egyezést, annak ellenére, hogy a mért tárgy sosem változott.



6.5. ábra. Fehérzaj gerjesztés eredményei, a sztinger kapcsolatoknál alátét nélkül de plusz súly ráhelyezve és különböző metódusok limitjei.



6.6. ábra. Az összes mérés FRAC eredménye az egyik "kicsit" meglazított ideális méréshez képest.

7. fejezet

A megvalósított program felépítése

A következő fejezetben a megvalósított szoftvert mutatom be. A fő megoldandó funkciók a valósidejűség, adatbeolvasás, automatizálás és grafikus kezelőfelület. Fontosnak tartottam, hogy kétféle futási állapota legyen a szoftvernek, a fő, valós idejű monitorozás mellett lehetséges legyen egy korábban elvégzett mérés tesztelése is, ha esetleg a mérést végző személy elfelejtette volna elindítani a szoftvert, akkor is lehetősége legyen utólagos ellenőrzésre.

A szoftvert *Python* programnyelvben írtam, mert a szükséges jelfeldolgozási függvényeket implementálták (*scipy* könyvtár [24]) hasonlóan a Matlabhoz, és a mérések kezeléséhez és adatfeldolgozásához szükséges API (Application Programming Interface) többek között ezt a nyelvet támogatja.

7.1. Valós idejű adatfeldolgozás

A legelső megoldandó probléma a mérési adatok kinyerése valós idejűleg. Az elrendezés mérőjeleit Head Acoustics frontenddel monitorozzuk, amihez tartozik egy felvevő szoftver is (Head Recorder), amin keresztül kommunikál a számítógéppel, és az adatok eltárolását végzi. Emiatt ahhoz, hogy nyers valós idejű adatokhoz hozzá tudjunk férni a frontendből, mindenképpen használnunk kell a Head Recorder szoftvert valamilyen szinten.

7.1.1. Head Acoustics API

A Head Acoustics csapata négy API-t valósított meg [11], amik mind különböző célt szolgálnak:

- Data Access API, a felvételeket és elemzési eredményeket tartalmazó HEAD Data Files (HDF) fájlok olvasására és írására használható.
- Data Processing and Representation API, egy másik általuk fejlesztett szoftver amely a mérési adatok kiértékelésére szolgál, az ArtemiS SUITE. Ennek a vezérlése ezen az API-n keresztül lehetséges. Ezt a szoftvert vezérelve egyszerűbb kiértékelési metódusok lesznek elérhetőek.
- Data Acquisition API, a korábban említett Head Recorder felvevő szoftver funkcióinak a vezérlését teszi lehetővé. Ez az API lehetővé teszi, hogy a különböző funkciókat dedikált szoftveralkalmazásokkal kezeljék, és ezáltal egyetlen testreszabott, specializált munkafolyamatba integrálják.
- Documentation and Metadata API különböző a korábbi API-k segítségével megvalósított bővítmének létrehozásának dokumnetációjában segít. Ez egyfajta templateként szolgál.

Látható, hogy a lehetőségek közül a Data Processing and Representation API tűnik a legalkalmasabbnak a szoftver működési szempontjából. Fontos megemlíteni, hogy ezek az API-k licenszekhez vannak kötve, és ha nincs meg a kellő licensz, akkor nem használható, még saját felhasználásra sem a csomag. Sajnos, a rendelkezésre álló licensz egyedül az első, azaz a Data Access API, ami csak a mérési fájlok beolvasására szolgál. Így a megvalósított szoftver a mérés teljes vezérléséért nem tud felelős lenni.

Ennek tudatában folytattam a szoftver megvalósítását, amihez először így a Head Recordert kellett előre beállítanom. Az első lehetőség, ami a legrosszabb is egyben, hogy amíg a mérés zajlik a monitorozó rendszer nem fut – mivel nincs hozzáférésünk a mérési fájlokhoz – és csak a Head Recorderen keresztül veszünk fel egy igen hosszú mérést, amit a végén Pythonban beolvasva kiértékelünk visszamenőleg. Ezt szerettem volna elkerülni, mert ezzel a valós idejű kiértékelés nem megvalósítható.

Láttam, hogy a Head Recorder teljes eliminálása a kiértékelés szempontjából lehetetlen, emiatt végül olyan megoldást választottam, ahol egy egyszerűbb előkonfigurálás után részadatokhoz is hozzáférünk a mérés során. Ezt úgy oldottam meg, hogy a Head Recorderen belül lehetőségünk van egyszerűbb automatizáció megvalósítására és ennek segítségével létrehozható egy mérési szekvencia, ami minden egyes részmérés után elkészít egy HDF fájlt, amit utána azonnal beolvashatunk Pythonban. A szekvenciák időtartamát érdemes úgy választani – ha például chirp gerjesztést használunk – hogy ez megegyezzen a jel periódus-idejével, tehát ez 5-10 másodperc jelen esetben. Ezzel a megvalósítással sikeresen elérhetőek a mérési adatok néhány másodperces késéssel.

7.2. Szoftverstruktúra

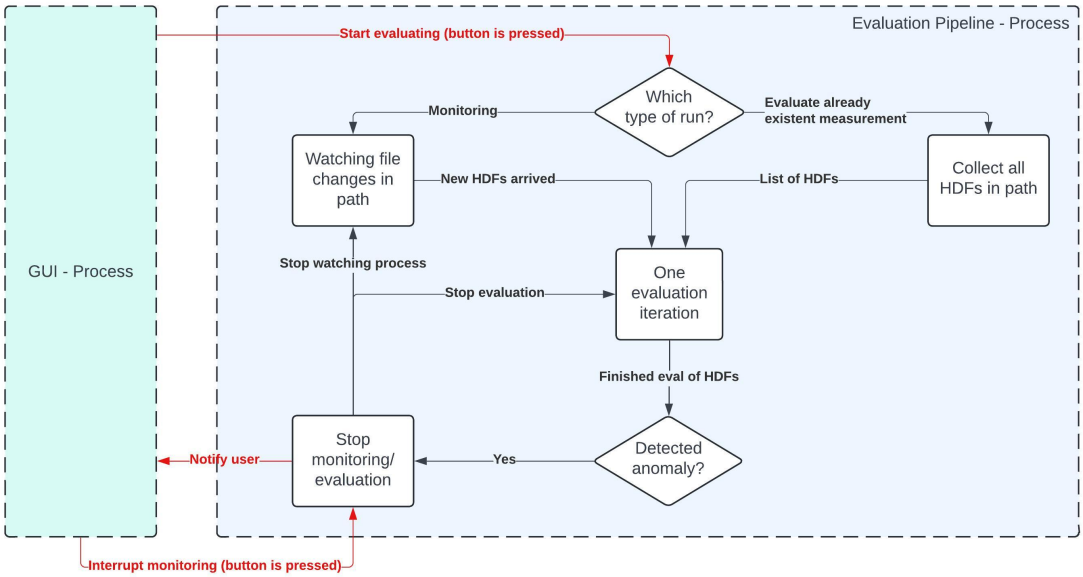
A szoftver két párhuzamos folyamatból épül fel, ahogy azt a 7.1. ábra is mutatja. Az egyik a grafikus felületért felelős (zöld) a másik pedig a monitorozásért (kék). Az objektumorientáltságra törekedve a kettejük közötti kommunikáció és függőség minimális. A kétféle futási mód implementálása érdekében törekedtem arra, hogy a megírt függvények és osztályok univerzálisan felhasználhatóak legyenek mindkettő esetben. A fő eltérés a monitorozó és teljes kiértékelő mód között, hogy az egyik esetén folyamatosan vizsgálni kell egy mappa tartalmának változását, míg a másik esetén csak össze kell gyűjteni a mappában található fájlokat és azokat kiértékelni.

A *One evaluation iteration* folyamat végrehajtja a fájlbeolvasást és kiértékelést, függetlenül a beolvasandó fájlok számától. A kezelőfelületről bármikor meg lehet állítani a szoftver futását, ennek a megvalósítását a 7.4.1. alszakasz részletezi. A *Stop monitoring/evaluation* az egyedüli blokk az ábrán, amihez nem kapcsolható általam megvalósított osztály vagy függvény. Ennek a kezelését a *multiprocessing* [20] könyvtárra bíztam, és elég volt eseményekhez kötni (például, ha anomáliát érzékel, vagy rákattintanak a *Stop Evaluation* gombra) a kiértékelés futásának megállítását, amit a 7.4.1. alszakasz részletez.

7.3. Evaluation pipeline folyamat

`evaluation_pipeline_process` a feldolgozásért és kiértékelésért felelős osztály, amely a *multiprocessing* könyvtár `Process` osztályának a leszármazottja. Ez amiatt szükséges, hogy ha grafikus felületen keresztül szeretnénk a kiértékelést futási állapottól függetlenül leállítani, ahhoz szükséges, hogy ez a két folyamat párhuzamosan működhessen.

A konstruktorban megkapja a grafikus felülettől a szükséges paramétereket (7.1. kódrészlet). Ezek két csoportra bonthatóak: A `UserInput` (7.2. kódrészlet) egy általam létrehozott *dataclass* típusú osztály, melynek fő feladata a grafikus felületen, a felhasználó által megadott adatok tárolása és átadása az `evaluation_pipeline_process(Process)`



7.1. ábra. Megvalósított szoftver futási blokkdiagramja.

osztálynak. Például itt van eltárolva a megfigyelendő mappa elérési útvonalja, vagy a kiértékelés típusa. A másik átadott paraméterek az egyes események, *Eventek*. Ez szintén a *multiprocessing* könyvtár egyik osztálya és fő funkciója a párhuzamos folyamatok közötti függőség létrehozása. Az *Event* egyfajta boole-i állapotváltozóként működik, és két műveletet lehet rajta végezni: állítani az állapotát és azt kiolvasni. Ha ugyanazt az *Eventet* több különböző párhuzamos process is megkapja, akkor lehetséges az, hogy az egyik addig várjon, amíg az állapota nem kerül "set" állapotba, amit másik szálon futó kód is állíthat. Ebben az esetben, ha például anomáliát észlel a kiértékelő process, akkor az `anomaly_event`-et és az `end_event`-et is állítom, így amikor a grafikus felület teszteli ezeket, tudni fogja, hogy azért lett vége a kiértékelésnek, mert anomáliát érzékelt. A másik eset, ha a grafikus felületen állítjuk meg a kiértékelést, akkor csak az `end_event`-et állítom.

```
def __init__(self, UserInput, end_event, anomaly_event):
    Process.__init__(self)
    self.UserInput = UserInput
    self.eval_end_event = end_event
    self.anomaly_event = anomaly_event
    self.reference_data = None
    self.units = []
    self.eval_methods = eval_methods(
            fs = 48000,
            window_size = 8192,
            overlap = 0.66,
            valid_freqs = UserInput.freq_limits
            )
```

7.1. lista. Az `evaluation_pipeline_process` osztály konstruktora, Python kód

Látható még a 7.1. kódrészletben, hogy létrehoz a konstruktor egy `eval_methods` objektumot is. Ezt az osztályt még a különböző kiértékelési metódusok vizsgálata során valósítottam meg, aminek köszönhetően nem kellett mindig megadnom az alapvető paramétereket az összehasonlítás közben. A végső szoftverben csak a `calc_FRAC` függvényét használom, és esetlegesen a mintavételi frekvenciát frissítem a beolvasott HDF fájlokhoz igazítva. Pythonban az osztályok változói csak public elérésűek, így ehhez nem kellett külön függvényt írnom.

```
@dataclass
```

```
class UserInput:
    name = ""
    HDF_folder = None
    channels = []
    freq_limits = None
    limit = None
    watch = True
```

7.2. lista. `UserInput dataclass` típusú osztály, Python kód

Az `evaluation_pipeline_process(Process)` osztálynak két függvénye van:

- `run`: A két kiértékelési metódus aggregálásáért felelős függvény, a `Process` osztályból felülírt függvény, és akkor hívódik meg, ha futtatjuk a process-t a beépített `start` függvénnyel. Először a `self.UserInput.watch` boolean értékét vizsgálja meg, ami, ha igaz állapotban van, akkor a monitorozást indítja el(7.3.1.2. szakasz), ha pedig hamis állapotban, akkor az adott mappában a HDF-ek összegyűjtését és kiértékelését (7.3.1.1. szakasz).
- `calc_one_eval_process`: Feladata, hogy a kapott HDF fájlokat kiértékelje (7.3.2. alszakasz).

7.3.1. Fájlkezelés

Az automatizációhoz a Head Recorder megfelelő konfigurációja után, a monitorozó szoftvernek a mérési fájlok mappáját kell vizsgálnia futási módtól függően.

7.3.1.1. Mérési fájlok összegyűjtése

Ha korábban elkészült fájlokat szeretnénk kiértékelni, akkor az adott mappából kell összegyűjteni az összes HDF kiterjesztésű fájlt. Ahogy azt a 7.3. kódrészlet mutatja, ez egy egyszerű mélységi bejárással megoldható, ami lehetővé teszi az esetleg almappákba került mérési fájlok megtalálását is. A függvény megvalósításához az *os* könyvtár [22] beépített függvényeit használtam.

```
def get_HDFs(HDF_path: Path) -> list:
    HDFs = []
    for path, _, files in os.walk(HDF_path):
        for file in files:
            fullpath = os.path.join(path, file)
            _, file_extension = os.path.splitext(fullpath)
            if os.path.isfile(fullpath) and (file_extension == ".HDF"):
                HDFs.append(fullpath)
    return HDFs
```

7.3. lista. HDF fájlkereső függvénye, adott elérési utvonalon, Python kód

7.3.1.2. Új fájlok monitorozása

A másik futási mód, a valós idejű monitorozás. Ehhez folyamatosan figyelni kell az adott mappa változásait. Egy egyszerű és triviális megoldás, ha a `get_HDFs` függvényt `while`-ciklusban folytonosan futtatjuk, viszont ez a metódus nagyon számításigényes, így ennek a használata nem ajánlott. A *watchdog* [26] könytár erre a problémára nyújt megoldást. Először létre kell hozni egy leszármazottját a `FileSystemEventHandler` osztálynak (7.4. kódrészlet), aminek az `on_created` függvényét módosíthatjuk a célunknak megfelelően. Ez a függvény akkor fut le, ha egy új fájl érkezik az adott mappába. A konstruktort is kibővítettem úgy, hogy a `calc_one_eval_process` tagfüggvényt argumentumként át kelljen adni neki, ezzel lehetővé téve, hogy egy másik osztály függvényét (jelen esetben a kiértékelő függvényt) futtathassam az `on_created` függvény lefutásakor.

Az `evaluation_pipeline_process` osztály `run` függvényében a 7.5. kódrészlet felel a monitorozásért, ahol a `new_file_monitoring` osztály egy objektumát hozom létre, majd a *watchdog* könyvtárnak az `Observer` osztály segítségével időzítem és indítom a mappa tartalmának monitorozását.

A *try-finally* blokk és benne a `time.sleep(1)` függvény azért felelősek, hogy mindaddig, amíg a monitorozás fut, ne lépjen ki a függvényből, ha pedig befejeződik, akkor pedig a *finally* blokkon belüli rész mindenképp fusson le, ezzel befejezve a folyamatot.

```
class new_file_monitoring(FileSystemEventHandler):
    def __init__(self, calc_one_eval_process):
        self.calc_one_eval_process = calc_one_eval_process
        super().__init__()

    def on_created(self, event):
        print("got new file")
        new_file_path = event.src_path
        _, file_extension = os.path.splitext(new_file_path)
        if os.path.isfile(new_file_path) and (file_extension == ".HDF"):
            self.calc_one_eval_process([new_file_path])
```

7.4. lista. Mappa monitorozáshoz szükséges osztály, Python kód

```
event_handler = new_file_monitoring(self.calc_one_process)
observer = Observer()
observer.schedule(event_handler, path=self.UserInput.HDF_folder, recursive=False)
observer.start()

try:
    while True:
        time.sleep(1)
finally:
    observer.stop()
    observer.join()
```

7.5. lista. A `new_file_monitoring` osztály egy objektumának létrehozása és futtatása, Python kód

7.3.2. Kiértékelés

A szoftver fő kiértékelésért felelős egysége a `calc_one_eval_process` függvény. A függvény argumentumaként kap egy vagy több HDF fájl elérési útvonalát, amit utána kiértékel egymás után automatikusan. Ehhez a *Prefect* könyvtárat [23] találtam alkalmasnak. Maga a *Prefect* nagyobb - akár online - rendszerek kezelésére lett tervezve, ahol nagyméretű adatok feldolgozását kell kezelnie. Ennek a struktúrának a központi eleme az úgynevezett *Flow*. Ez tekinthető egyfajta függvénynek, amit különböző *taskok* építenek fel. Emellett azt is definiálhatjuk benne, hogy ezeket a taskokat milyen módon szeretnénk futtatni. Például, ha az a cél, hogy az összes beérkező új adatot egymástól függetlenül, a lehető leggyorsabban szeretnénk kiértékelni, akkor érdemes párhuzamosítani ezek futását, amit itt lehet definiálni. Egy Flow futását illetőleg kétféle típus között választhatunk:

- LocalExecutort abban az esetben érdemes használni, ha a *Flow* „elég gyorsan” fut, vagy nincs lehetőség párhuzamosításra. Ez a legegyszerűbb lehetőség, és a legkönnyebben kezelhető.
- LocalDaskExecutort pedig akkor érdemes használni, ha nagyobb adathalmazt szeretnénk feldolgozni és a feladatok párhuzamosítása is lehetséges. A LocalDaskExecutor lehetővé teszi a feladatok párhuzamos futtatását helyi szálakként vagy folyamatokként.

A megvalósítandó szoftver esetében, ha már elkészült mérésfájlok kiértékelését szeretnénk elvégzeni, a párhuzamosítás hasznos lenne, viszont az első néhány mérésből szükséges

referenciát vennünk ebben a futási módban is, emiatt nem használható semmilyen párhuzamosítás, mert akkor nem garantált, hogy az első mérés fog lefutni először.

A 7.6. kódrészletben látható a megvalósított *Flow* a beérkező adatok kiértékeléséhez. Látható, hogy a *Flow* definiálása után meghívom az egyes *taskokat*, végül pedig futtatom a *Flowt* a `run` beépített függvényével. Látható még, hogy négy *taskot* valósítottam meg, ami alapvetően három funkcionális részre osztható szét:

1. Adatok belovasása - az első két *task*
2. Transzformálás - FRAC értékek kiszámolása
3. Döntéshozatal és visszajelzés a grafikus felületnek.

```
try:
    license_handler = LicenseHandler({5091})

    with Flow(
        name="Local Exe",
        executor=LocalExecutor(),
    ) as flow:
        print("entered flow")
        meta = extract_metadata(new_file_path_list)
        data = extract_samples.map(meta)
        results = transform.map(data)
        decision = evaluate.map(results)

    flow.run()

finally:
    license_handler.release()
```

7.6. lista. Flow definiálása és futtatása, Python kód

Látható még a 7.6. kódrészletben, hogy bizonyos *taskok* `map()` függvényen keresztül hívódnak meg. Ez a technikája, ha egy adathalmazon iterálva szeretnénk futtatni az adott taskot. Esetünkben az `extract_metadata` *task* még a HDF-ek elérési útvonallistáját kapja és egy lista `HDFMetadata` objektumot (7.3.2.1. zsakasz) ad vissza, viszont az `extract_samples(meta_data: HDFMetadata)` *taskot* már a `map` függvényen keresztül hívom meg, így lehetősége lesz több mérési fájl esetén iterálni az adott és további *taskokon.*

Végül, hogy a Head Acoustics API-t használni lehessen, a megfelelő licenszt le kell foglalni a szükséges időre, amit a `license_handler = LicenseHandler({5091})` és `license_handler.release()` kombinációjával lehet megvalósítani. A *try-finally* blokk pedig azért felelős, hogy ha nem áll rendelkezésünkre a megfelelő licensz, akkor ne kerüljön futási hibába a szoftver.

7.3.2.1. Head Data Files (HDF) beolvasás

A HDF fájlok beolvasását a Head Acoustics API [11] segítségével két nagyobb lépésben valósítottam meg: A metaadatok beolvasása és utána a csatornajelek beolvasása. Ahhoz, hogy megérthessük a metaadatok beolvasásának szükségességét, először a csatorna jelek kiolvasásának a működési elvét ismertetem. A `extract_samples` megvalósított függvény felel ezért a feladatért, amit a 7.7. kódrészlet mutat be. Látható, hogy a függvény argumentuma a `HDFMetadata` osztály egy objektuma. Ez az, amit a meta adatok kinyerése során a függvény előállít.

```
@task
def extract_samples(meta_data: HDFMetadata) -> np.array:
    sampled_channels = []
    with open_time_data_HDF(meta_data.path) as time_data_HDF:
        for sampled_channel in list(time_data_HDF.GetSampledChannels()):
            # only read the needed channel
```

```
            for channel_name in meta_data.channels:
                if sampled_channel.ChannelInfo.Name == channel_name:
                    sampled_channels.append(
                        np.array(
                            sampled_channel.GetData(
                                int(meta_data.start_index), int(meta_data.duration)
                            )
                        )
                    )
    return np.array(sampled_channels)
```

7.7. lista. HDF adatok beolvasása, Python kód

Egy `HDFMetadata` objektum, az adott fájl kiolvasáshoz szükséges összes adatot tárolja (7.8. kódrészlet). Ennek a dataclass-nak a motivációja abból fakad, hogyha esetleg össze szeretnénk fűzni több mérési fájlt, és egy HDF fájl részletét szeretnénk csak beolvasni, akkor erre is lehetőségünk legyen. Látható a 7.7. kódrészletben, hogy a HDF mérési adatokat a `GetData(start_index: int, duration: int)` belső függvénnyel lehet kiolvasni, emiatt szükségesnek találtam a kezdeti és hossz paraméterek előbbi kiolvasását.

```
@dataclass
class HDFMetadata:
    path: Path = None
    channels: list[str] = field(default_factory=list)
    start_index: int = None
    duration: int = None
```

7.8. lista. HDFMetadata dataclass típusú osztály, Python kód

7.3.2.2. Transzformálás

Az adatok feldolgozásáért felelős függvény a `transform`, ami az `eval_methods.calc_FRAC` függvényt hívja meg. Itt merül fel a referenciajel kérdése. A végleges megoldáshoz létrehoztam egy új belső változót a `evaluation_pipeline_process` osztályban, amit None-ként elődefiniáltam a konstruktorban (`self.reference_data = None`). A transform *taskban* ennek az értékét figyelem, és ha az értéke még None, akkor módosítom az adott mérés jeleire, és csak ezt követően végzem el a FRAC transzformálást. Igy az első mérés jelét fogom referenciának venni, és így az első kiértékelés során a FRAC értéke egy lesz, mivel önmagával hasonlítja össze, viszont a többi kiértékelendő mérés már használható eredményeket fog mutatni.

7.3.2.3. Döntéshozatal

Az utolsó feladat a döntéshozatal. Ezt a 6. fejezetben megállapított limit szerint valósítottam meg. Elegendő egy egyszerű összehasonlítást elvégezni, és ha a számolt érték kisebb, mint a megadott limit, akkor a grafikus felületnek az *eventeken* (7.3. szakasz) keresztül jeleznie kell (7.9. kódrészlet).

```
@task
def evaluate(result_frac: float):
    if result_frac < self.UserInput.limit:
        self.anomaly_event.set()
        self.eval_end_event.set()
```

7.9. lista. A döntéshozó task megvalósítása, Python kód

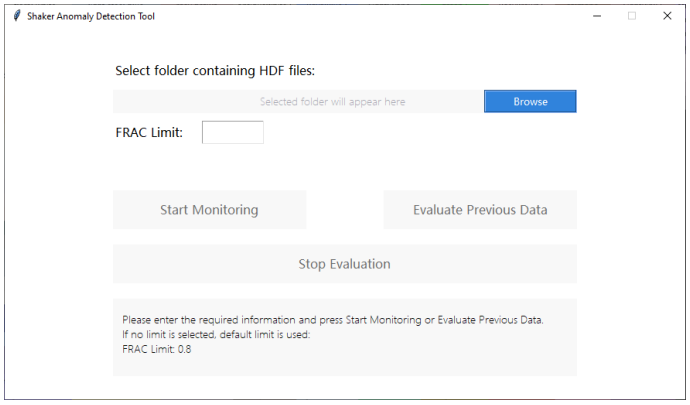
7.4. Kezelőfelület, grafikus interfész

A grafikus felületet a *Tkinter* könyvtár [25] segítségével valósítottam meg. Használata nagyon egyszerű: Különböző *widget* objektumokat lehet elhelyezni a grafikus felületen

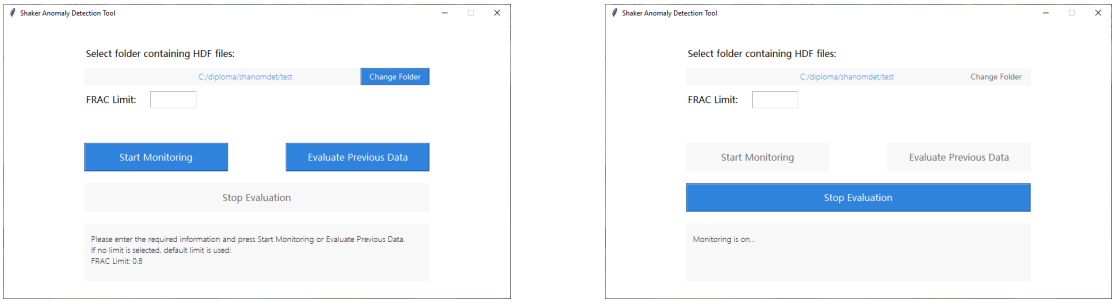
az osztály konstruktorában, amiknek a viselkedését függvényekhez lehet kötni. A függvényekben egyszerűen implementálható a megfelelő funkcionalitás, legyen ez grafikus felület állapotváltozás, belső paraméterek bevitele vagy folyamat indítása/megállítása.

A 7.2. ábra mutatja a felület kezdeti állapotát. Először a célmappát kell megadni, csak ezek után fog aktiválódni a monitorozás vagy kiértékelő gomb. A limitet lehet még módosítani, de azt nem szükséges. Ezután 3 gomb látható, amik állapota dinamikusan változik attól függően, hogy melyik állapotban van a szoftver. Például monitorozás közben a megállítás aktiválódni fog, de a start gomb deaktiválva lesz (7.3. ábra). Legalul található egy szöveg ablak, ami folyamatosan visszajelez a felhasználónak a szoftver állapotáról és kiértékelés szakaszáról.

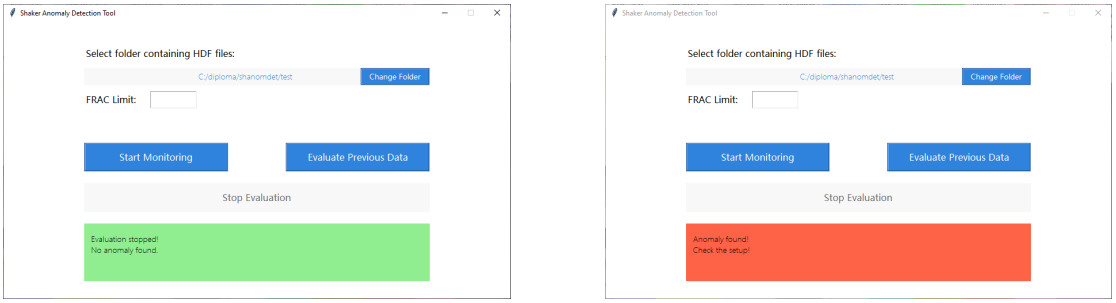
Ha az *Evaluate Previous Data* gombra kattintunk és befejeződik a kiértékelés, akkor két lehetséges állapotba fog kerülni a szoftver, attól függően, hogy talált-e anomáliát (7.4. ábrák).



7.2. ábra. Grafikus interfész kezdeti állapota.



7.3. ábra. Grafikus interfész célmappa meghatározása utáni és futás alatti állapota.



7.4. ábra. Grafikus interfész kiértékelés utáni lehetséges állapotai.

7.4.1. Kommunikáció a grafikus felület és a kiértékelés között

A grafikus felület felel a monitorozás elindítására és esetleges visszajelzésre, ha anomáliát talál. Ezt 3 függvény segítségével valósítottam meg, amint azt a 7.10. kódrészlet is mutatja. A `startEvaluation` akkor fut le ha lenyomják a *Start Monitoring* vagy az *Evaluate Prewious Data* gombot. Ekkor létrehoz a függvény 2 *Event* objektumot, amiket átadunk az `evaluation_pipeline_process`-nek és ezt követően el is indít. A `wait_for_evaluationThread` függvény egy párhuzamos szálon elindul és célja, a kiértékeléstől függően állítani a grafikus felület állapotát. Ez a függvény viszont az első soránál megáll a `self.eval_end_event.wait()` függvény miatt, ami az eventek egy beépített függvénye és futtatása azt eredményezi, hogy nem hagyja a kódot tovább futni mindaddig, amíg ez az event "set" állapotba nem kerül. Esetünkben addig nem fut tovább, amíg a `eval_end_event` nem kerül *set* állapotba. Erre 3 lehetőség van:

- A *Stop Evaluation* gombra kattintva lefut a `stopEvaluation` függvény ezzel megállítva a kiértékelést.
- Anomáliát talált az algoritmus és a döntéshozatal során (7.3.2.3. alalszakasz) az `anomaly_event` és az `eval_end_event` is *set* állapotba kerül.
- A nem valós idejű kiértékelés végére ért és nem talált anomáliát.

```
class App(Frame):
...
    def startEvaluation(self):
        ...
        self.eval_end_event = Event()
        self.anomaly_event = Event()
        self.p1 = evaluation_pipeline_process(self.UIData, self.eval_end_event, self.anomaly_event)
        self.p1.start()
        self.wait_for_evaluation_end = Thread(
            target=self.wait_for_evaluationThread,
            name="wait_for_evaluation_end")
        self.wait_for_evaluation_end.start()

    def wait_for_evaluationThread(self):
        self.eval_end_event.wait()
        if self.anomaly_event.is_set():
            self.MessageTab.config(bg="tomato")
            self.message_text.set(f"Anomaly found! \nCheck the setup!")
        else:
            self.MessageTab.config(bg="light green")
            self.message_text.set(f"Evaluation done! \nNo anomaly found.")
        ...

    def stopEvaluation(self):
        ...
        self.p1.terminate()
        self.eval_end_event.set()
```

7.10. lista. Thread és Event-ek megvalósítása a grafikus felület osztályában, Python kód

Miután az `eval_end_event` "set" állapotba került, a függvény tovább fog futni és egy vizsgálat indul el, amihez elegendő csak az `anomaly_event` állapotát vizsgálni, és ettől függően visszajelezni a felhasználónak. A 7.10. kódrészletben a "..." részek a grafikus felülettel kapcsolatos beállításokat takarják.

8. fejezet

Összegzés

Dolgozatomban egy mérés monitorozó rendszert mutattam be és valósítottam meg. Az adott mérési környezet bemutatását követően részleteztem a különböző előforduló hibatípusokat és lehetséges okait. Ezek a hibatípusok két csoportra oszthatóak és láthattuk, hogy rezgésakusztikai mérések során a nemlineáris hibák jelentkeznek gyakrabban a modális viselkedés változásához képest.

Ezt követően különböző algoritmusokat vizsgáltam meg, melyek két adathalmaz között frekvenciatartományban és időtartományban is vizsgálják a relációjukat. Először a jelfeldolgozás területén közkedvelten használt koherenciát vizsgáltam, mely két jel közötti frekvenciafüggő linearitást vizsgálja. Majd a spektrális lapossággal folytattam, amihez elegendő egy jel spektrumát vizsgálni. Az RDC időtartományban vizsgál két jelet és nemlineáris függőségeket is képes kimutatni különböző nemlineáris projekciók segítségével. Az algoritmus egy hasonlósági számot ad meg. Végül pedig a FRAC arányszámot vizsgáltam, mely a többi kiértékelési metódussal szemben teljes átvitelifüggvényeket hasonlít össze, emiatt ehhez szükségünk volt egy referencia átvitelifüggvényre.

A dolgozatomban ezeket a metódusokat teszteltem valós mérési eredményeken és összehasonlítottam az eredményeiket miközben a megvalósítandó szoftver elvárásait végig figyelemben tartottam. A mérések során tudatosan vittem hibát a rendszerbe és 3 különböző hibapontot vizsgáltam. Végül az algoritmusok közül a FRAC metódus ígérkezett a legalkalmasabbnak, mert mindkét hibatípusra érzékeny és a gerjesztőjel típusától függetlenül alkalmazható.

Küszöbérték meghatározással folytattam a munkám, ami a szoftver döntéshozatali szempontjából volt fontos. Ehhez több mérést is elvégeztem, hogy nagyobb adathalmaz álljon rendelkezésemre. Ez alatt a folyamat alatt próbáltam valamilyen szintű diverzitást belevinni a mérésekbe. Minden elrendezési esetben ugyanazokat a bevitt hibatípusokat mértem meg az összehasonlíthatóság érdekében. Az eredményeket különböző metodikákkal vizsgálva határoztam meg a végleges döntési határt, ami 0.8-ra adódott.

Végül a szoftver implementációját végeztem el. Két nagyobb egységre bontottam a szoftvert, a grafikus felületért felelős és a kiértékelésért felelős folyamatokra. Ezek kommunikációját próbáltam minimálisra csökkenteni, ezzel is növelve függetlenségüket, ami a szoftver egyszerűségét és stabilitását segítette elő.

A megvalósított monitorozó szoftver jelenleg alkalmas valós idejű anomáliadetekcióra rázóasztalos mérések során. A mérést végző mérnöknek ehhez egy kisebb frontend, egy uniaxiális gyorsulásszenzor és egy laptop szükséges, így viszonylag hordozható az elrendezés. A szenzort először fel kell helyezni a mérendő testre, majd az automatikus monitorozást a két szoftveren el kell indítani. Érdemes külön tárolni egy szenzort kifejezetten erre a célra, így nem kell minden alkalommal a paramétereit külön bevinni a szoftverbe. Ezt követően a fő mérési elrendezés felparaméterezése és a mérés elindítása megkezdődhet. A laptopon

futó szoftver azonnal jelez, ha eltérést észlel, ekkor érdemes megvizsgálni az elrendezést, hogy például minden megfelelően kapcsolódik-e. Ha minden rendben folytatódhat a mérés és ennek monitorozása.

Továbbfejlesztésre a szoftver és az algoritmus oldaláról is van lehetőség. Ergonómiai szempontból a monitorozás során a köztes szoftver (Head Recorder) elhagyása jelentős kényelmi javulást eredményezne. A kiértékelés párhuzamosítását is valamilyen szinten meg lehet valósítani ami a szofter kiértékelésének gyorsulását eredményezné. A grafikus felületen több beállítási lehetőség megadása a finomhangolhatóságon segít és ezzel jobban kezelhető lesz a rendszer. Például a vizsgált frekvenciatartomány még nem állítható kívülről. Az algoritmus szempontjából érdemes lehet a referenciát több mérés átlagából venni, nem csak az első mérésből, ezzel esetleges zajokat szűrhetünk ki az első mérés esetleges hibáiból.

Irodalomjegyzék

[1] S. Cohen: The basics of machine learning: Strategies and techniques. In *Artificial Intelligence and Deep Learning in Pathology*. 2 fejezet. 2021, Elsevier Inc., 13–40. p. DOI: 10.1016/B978-0-323-67538-3.00002-6.

[2] M. Colledani – T. Tolio – A. Fischer – B. Iung – G. Lanza – R. Schmitt – J. Váncza: Design and management of manufacturing systems for production quality. *CIRP Annals*, 63. évf. (2014) 2. sz., 773–796. p. DOI: 10.1016/j.cirp.2014.05.002.

[3] L. Dooho – A. Taesu – K Hyun-Soo: Development of a new frequency response function similarity metric for finite element model updating in mid-frequency range. *Journal of Sound and Vibration*, 436. évf. (2018), 32–45. p. DOI: 10.1016/j.jsv.2018.08.051.

[4] G. Doran: RDC algorithm in Python. https://github.com/garydoranjr/rdc. (hozzáférés dátuma: 2024. május 22.).

[5] Yuan Fang – Tong Zhang: Sound quality investigation and improvement of an electric powertrain for electric vehicles. *IEEE Transactions on Industrial Electronics*, 65. évf. (2017. 08) 2. sz., 1149 – 1157. p. DOI: 10.1109/TIE.2017.2736481.

[6] P. Fiala: A hangszerek fizikája jegyzet. https://last.hit.bme.hu/sites/default/files/documents/hangfiz.pdf. (hozzáférés dátuma: 2024. május 22.).

[7] P. Flach – H. Blockeel – C. Ferri – J. Hernández-Orallo – J. Struyf: *Decision Support for Data Mining*. Boston, MA, 2003, Springer US, 81–90. p. ISBN 978-1-4615-0286-9. DOI: 10.1007/978-1-4615-0286-9_7.

[8] H. Freudenthal: The 'empirical law of large numbers' or 'the stability of frequencies'. *Educational Studies in Mathematics*, 4. évf. (1972) 4. sz., 484–490. p. http://www.jstor.org/stable/3482152.

[9] M. S. Gadala – M. A. Dokainish – G. Ae Oravas: Formulation methods of geometric and material nonlinearity problems. *International Journal for Numerical Methods in Engineering*, 20. évf. (1984) 5. sz., 887–914. p. DOI: 10.1002/nme.1620200508.

[10] A. W. Gardner: A unifying view of coherence in signal processing. *Elsevier, Signal processing*, 29. évf. (2017. 08) 2. sz., 113–140. p. DOI: 10.1016/0165-1684(92)90015-O.

[11] Head Acoustics API honlapja. https://www.head-acoustics.com/products/analysis-software/artemis-suite-extensions. (hozzáférés dátuma: 2024. május 22.).

[12] J. Herre – E. Allamanche – O. Hellmuth: Robust matching of audio signals using spectral flatness features. In *Proceedings of the 2001 IEEE Workshop on the Applications of Signal Processing to Audio and Acoustics (Cat. No.01TH8575)* (konferenciaanyag). New Platz, NY, USA, 2001, 127–130. p. DOI: 10.1109/ASPAA.2001.969559.

[13] W. Heylen – S. Lammens – P. Sas: *Modal Analysis Theory and Testing*. 2013, Katholieke Universiteit Leuven. ISBN 90-73802-61-X.

[14] D. Lopez-Paz – P. Hennig – B. Schölkopf: The randomized dependence coefficient. *Advances in Neural Information Processing Systems*, 26. évf. (2013. 04). DOI: 10.48550/arXiv.1304.7717.

[15] A. L'Heureux – K. Grolinger – H. F. Elyamany – M. A. M. Capretz: Machine Learning With Big Data: Challenges and Approaches. *IEEE Access*, 5. évf. (2017), 7776–7797. p. DOI: 10.1109/ACCESS.2017.2696365.

[16] R. Malhotra – A. Jain: Fault prediction using statistical and machine learning methods for improving software quality. *Journal of Information Processing Systems*, 8. évf. (2012. 06) 2. sz., 241 – 262. p. DOI: 10.3745/JIPS.2012.8.2.241.

[17] M. A. Panza: A Review of Experimental Techniques for NVH Analysis on a Commercial Vehicle. *Energy Procedia*, 82. évf. (2015), 1017–1023. p. DOI: 10.1016/j.egypro.2015.11.861.

[18] M Pastor – M Binda – T Harčarik: Modal assurance criterion. *Procedia Engineering*, 48. évf. (2012), 543–548. p. DOI: 10.1016/j.proeng.2012.09.551.

[19] Polytec, Scanning Vibrometer dokumentációja. https://www.polytec-cn.com/uploads/ueditor/20190719/5d5088ad3e12196826257f91343d3647.pdf. (hozzáférés dátuma: 2024. május 22.).

[20] Python, multiprocessing könyvtár documentációja. https://docs.python.org/3/library/multiprocessing.html. (hozzáférés dátuma: 2024. május 22.).

[21] Python, numpy könyvtár documentációja. https://numpy.org/doc/. (hozzáférés dátuma: 2024. május 22.).

[22] Python, os könyvtár documentációja. https://docs.python.org/3/library/os.html. (hozzáférés dátuma: 2024. május 22.).

[23] Python, prefect könyvtár documentációja. https://docs.prefect.io/latest/. (hozzáférés dátuma: 2024. május 22.).

[24] Python, scipy könyvtár documentációja. https://docs.scipy.org/doc/scipy/index.html. (hozzáférés dátuma: 2024. május 22.).

[25] Python, tkinter könyvtár documentációja. https://docs.python.org/3/library/tk.html. (hozzáférés dátuma: 2024. május 22.).

[26] Python, watchdog könyvtár documentációja. https://python-watchdog.readthedocs.io/en/stable/. (hozzáférés dátuma: 2024. május 22.).

[27] R. Randall: *Fault Detection*. 4 fejezet. 2011, John Wiley Sons, Ltd, 143–165. p. ISBN 9780470977668. DOI: doi.org/10.1002/9780470977668.ch4.

[28] A. Rényi: On measures of dependence. *Acta Mathematica Hungarica*, 10. évf. (1959) 3-4. sz., 441 – 451. p. DOI: 10.1007/bf02024507.

[29] B. Sarens – B. Verstraeten – C. Glorieux – G. Kalogiannakis – D. Van Hemelrijck: Investigation of contact acoustic nonlinearity in delaminations by shearographic imaging, laser doppler vibrometric scanning and finite difference modeling. *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control*, 57. évf. (2010) 6. sz., 1383–1395. p. DOI: 10.1109/TUFFC.2010.1557.