Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Méréstechnika és Információs Rendszerek Tanszék

Hangszórók nemlineáris viselkedésének modellezése

Diplomaterv

Készítette
Simon Tibor

Konzulens
dr. Bank Balázs

2015. május 24.

Tartalomjegyzék

Ábrák jegyzéke

HALLGATÓI NYILATKOZAT

Alulírott *Simon Tibor*, szigorló hallgató kijelentem, hogy ezt a diplomatervet 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, 2015. május 24.

Simon Tibor
hallgató

Kivonat

Mindennapi életünk szinte minden percében a közelében vagyunk legalább egy hangszórónak. Legyen az a mobiltelefonunkban lévő kicsiny, vagy a gitárerősítőnkben lévő nagyteljesítményű hangszóró. Tapasztalataink alapján tudjuk, hogy a hangszórók hangvisszaadása olykor nem tökéletes. Ez főleg a hangszóró méretével és a rá adott teljesítménnyel függ össze. A dolgozat célja, hogy megismertesse az Olvasót a hangszórók viselkedésével, majd az identifikáció segítségével olyan módszert adjon a kezébe, hogy virtuálisan modellezhesse azokat.

Identifikáció alatt a rendszerek valósághű modellezését értjük a dolgozatban, blokk struktúrájú modelleket felhasználva. Az ismert lineáris modellek mellett bemutatjuk az Olvasónak a nemlineáris rendszerek leírására is alkalmas modelleket. A blokkos rendszerek előnye, hogy ellenben a nemlineáris rendszereket leíró többi modellel, a lineáris és a nemlineáris részek egymástól teljesen elkülönítve kezelhetőek. A dolgozatban három módszert is bemutatok blokkok, a blokkos modellek identifikációjára, amiből kettőt felhasználunk egy bővíthető mérési rendszer kialakítására.

A dolgozat a digitális jelfeldolgozás területén túl az objektumorientált módon megvalósított mérési rendszerekkel történő programozás területét is érinti. Jelfeldolgozás terén ez nem megszokott eljárás, azonban megfigyelhető, hogy a beépített MATLAB függvények is kezdenek áttérni az objektumorientáltságra a 2014-es verzió óta. Objektumorientáltsággal nem feltétlenül lehet jobb alkalmazásokat készíteni, azonban jelentősen leegyszerűsíti a fejlesztést és a karbantartást, valamint - ami a legfontosabb - a későbbi bővítést.

Abstract

We are surrounded by loudspeakers. Speaking of a small speaker in our mobile phone, or a powerful loudspeaker in the guitar cabinet, they are all around us. The sound quality of most speakers are not the best in all the time. During heavy load, especially in the case of a small speakers, distortion appears in all kinds. The main goal of this thesis is to familiarize the Reader with the nature of the electrodynamic loudspeakers, and the identification methods, that can be used for modeling the loudspeakers.

In this thesis the goal of identification is to copy the identifiable system's behavior. For that purpose we will use block structured models. These block structured models are capable of separating the linear and nonlinear parts. We will discuss three identification methods to construct block oriented models, and we will choose two of them to implement a measurement and identification system.

We will combine the field of digital signal processing with computer science, by implementing the measurement and identification systems using object oriented paradigms. This is not the conventional method to use in MATLAB, but after the release of the 2014a version, built in functionalities appeared in object oriented style. It is always a good idea to keep up with the flow of technologies. It will increase significantly the quality (and not the functionality, that depends on the skills of the programmer) of the programs. Leading to easier development, maintainability and future expandability.

1. fejezet

Bevezető

1.1. A dolgozat célja és felépítése

A hangszórók a mindennapi életünkben mindenhol körülvesznek bennünket. Minden pillanatban hanghullámok sokaságai érnek minket, folyamatosan hanggal "szennyezett" környezetben élünk, aminek jelentős részét hangszórók keltik. Elektronikus eszközeink maguktól nem képesek az igényeinknek megfelelő hangot kiadni[1]. Ehhez szükségük van egy átalakítóra, ami képes az elektromos jeleket hanghullámokká átalakítani. A dolgozatban bemutatott eszköz, a dinamikus hangszóró képes ellátni ezt a feladatot. Felépítésének köszönhetően a ráadott elektromos jeleket hanghullámokká képes átalakítani. Teszi mindezt kis hatásfokkal, torzítást adva a bemeneti jelhez. A dolgozat egyik célja, hogy ezeknek a torzításoknak felfedje okait és forrásait.

Az identifikáció a rendszerek megismerésének egyik módja. A rendszer gerjesztőjele és a rá adott válasza alapján szerzünk ismereteket, hogy létrehozzuk a rendszer modelljét, ami a szükségeinknek megfelelő mértékben képes utánozni az eredeti rendszer viselkedését. A lineáris rendszerek identifikációja megoldott probléma, az irodalma kidolgozott. A világ azonban nem minden esetben modellezhető lineáris modellekkel.

Nemlineáris rendszerek identifikációja még ma is aktív kutatási terület. A dolgozatban a nemlineáris rendszereket blokk alapú modellekkel írjuk le. A blokkos modellek az alapjai a hagyományos Wiener és Hammerstein modelleknek. A dolgozat másik célja, hogy módszereket adjon a blokk alapú modellek identifikációjához. A közölt módszereket a dolgozat végén elektrodinamikus hangszóróra alkalmazzuk.

A dolgozat felépítése alapján két nagyobb részre osztható. Az egyik az elméleti bevezető, a másik pedig az implementációra koncentráló rész, amiben a konkrét alkalmazott mérések kivitelezésére koncentrálunk. A dolgozat végén a függelékekben kiegészítő információk kaptak helyet, amelyek nélkül a dolgozat ugyan teljes lenne, de fontosnak tartottam megemlíteni őket.

A második fejezetben áttekintjük a lineáris és nemlineáris rendszereket, azok tulajdon-

[1] Működésükből adódó neszeket és zörejeket sugároznak a környezetükbe, azonban ez a legtöbb esetben nem az, amit el szeretnénk érni

ságait, valamint a leírásuk lehetőségeit.

A harmadik fejezet az elektrodinamikus hangszórókról szól. Itt megismerjük a hangszóró általános felépítését, és a használata során felmerülő problémákat, melyek lineáris és nemlineáris torzításban, valamint kis hatásfokban jelennek meg. Minden problémának fel fogjuk térképezni az okát.

A negyedik fejezet az identifikáció elméletéről szól. Megismerjük az identifikáció öt fő lépését, majd három identifikációs módszer elméletében merülünk el egy rövid időre: lineáris regresszió, iteratív adaptív identifikáció, és swept sine módszer.

Az ötödik fejezetben megismerkedünk az implementált lineáris regresszióra épülő identifikációs rendszerrel.

A hatodik fejezetben az iteratív identifikációs eljárást tárgyaljuk. Az implementált komolyabb bonyolultságú rendszer megismerése által ismereteket szerzünk a blokkos rendszerek identifikációjára használható ezen módszer működéséről.

A hetedik fejezetben az implementált blokkos modelleket identifikáló rendszerünket egy hangszóró modellezésére alkalmazzuk. Az eredményeket ki is értékeljük.

A nyolcadik és egyben utolsó fejezet a dolgozat záró gondolatait közvetíti az Olvasó felé.

2. fejezet

Lineáris és nemlineáris rendszerek

2.1. Lineáris rendszerek

Az emberiség megismerésre való törekvése során mindig is a környezetét próbálta leírni, egységes szabályokat keresve. Ehhez feltételezte, hogy a körülöttünk lévő világ tökéletes, mindent le lehet írni gyönyörűen kompakt egyenletek segítségével. A lineáris rendszerek világában mindezt megtehetjük, hiszen érvényes két törvény: a szuperpozíció és a homogenitás.

A szuperpozíció lehetővé teszi számunkra, hogy a világunkra gyakorolt hatásokat egymástól elkülönítve vizsgáljuk és tárgyaljuk. Lineáris rendszereknél igaz, hogy arra a gerjesztőjelre, ami két különböző jel összege, a lineáris rendszer válasza nem lesz más, mint a gerjesztőjelet alkotó jelekre adott egyenkénti válaszok összege.

$$f(x(t) + y(t)) = f(x(t)) + f(y(t)) \tag{2.1}$$

A homogenitás biztosítja, hogy ha egy rendszer gerjesztése konstansszorosára nő, akkor a rendszer kimenete is konstansszorosára fog nőni.

$$f(kx(t)) = kf(x(t)), k \in \mathbb{R} \tag{2.2}$$

A két tulajdonság kompakt formában:

$$f(k_1x_1(t) + k_2x_2(t)) = k1f(x_1(t)) + k2f(x_2(t)), k_1, k_2 \in \mathbb{R} \tag{2.3}$$

Minden olyan rendszer rendszer, ami teljesíti a szuperpozíció és a homogenitás feltételét, lineáris rendszer.

A lineáris rendszereket szeretjük, hiszen könnyű őket megérteni. Szeretjük azt gondolni, hogy ha egy mérlegre kétszer annyi súlyt teszünk, akkor a mutatója kétszer annyit fog mutatni. Ha külön mérünk meg két nehezéket, akkor a mért összegek a két nehezék összegét teszik ki egyben. Ha kétszer olyan gyorsan haladunk járművünkkel, kétszer olyan gyorsan érjük el az úticélunkat.

A lineáris rendszereket a tudományban is szeretjük. Egy lineáris rendszer teljesen leírható átviteli függvényével. Képzeljük csak el, mekkora jelentőséggel bír ez. Minden olyan rendszer, amely eleget tesz a linearitás feltételeinek, leírható elegánsan véges számú paraméter segítségével.

Ez azonban nem mindig van így. Sok esetben a lineáris feltételezésünk egyszerűen nem működik. Az öröm, amit egy kedvenc zeneszám meghallgatása okoz nem lesz nagyobb akkor, ha két kedvelt zeneszámot hallgatunk egyszerre. Az örömünk a kilencedik sütemény elfogyasztása után nem lesz kilencszer nagyobb. Az ilyen, és hasonló esetekben a lineáris szemléletünk felmondja a szolgálatot. A tudományban is, amint a linearitás feltételei sérülnek, abban a pillanatban már nem írható le a rendszer az átviteli függvényével. Áttérünk a nemlineáris rendszerek világába.

2.2. Nemlineáris rendszerek

Az előző szakaszban tárgyalt lineáris rendszerek az összes rendszert tartalmazó halmaznak csupán egy kis részhalmazát teszik ki. A természetben azonban csak egy bizonyos közelítéssel léteznek lineáris rendszerek. Ha szétnézünk, nagyon ritkán találkozunk egyenes vonalakkal. Azonban a legtöbb ember alkotta tárgy egyenes vonalakból épül fel. Ennek a magyarázata visszavezethető a természet matematikai úton való megismerésére.

Elődeink a lehető legegyszerűbben, legáltalánosabban igyekeztek leírni a természetben látott jelenségeket. A tudományban sokáig tökéletes geometriai alakzatokkal próbáltak közelíteni a természetet. A tökéletes alakzatok előnye, hogy rendkívül elegáns és kompakt matematikával írhatóak le. Megszületett a matematika analitikus ágazata. Minden levezethető, minden tökéletes.

Azonban amint feladjuk a megkötést, hogy minden tökéletes, problémákkal találjuk magunkat szembe. A képletek adta eredmények helyett a valóságban eltérő számértékeket kapunk. Szerencsétlenebb esetekben az eltérés jelentős.

Az ok a szemléletmódban keresendő. Megtehetjük ugyan, hogy a nemlineáris tulajdonságokkal rendelkező jelenséget leírjuk lineáris közelítésekkel. A közelítések azonban gyakran csődöt mondanak.

A mérnökök között ismert a mondás, hogy közelről minden lineáris, és messziről minden pontszerű. Ez azonban nem érvényes akkor, ha nem tudunk elég közel vagy távol menni az adott technológiai korlátozások, vagy a jelenség/rendszer természete miatt.

2.3. Lineáris rendszerek leírása

Lineáris rendszerek leírására rengeteg lehetőségünk van. A teljes rendszer leírható az impulzusválaszával, az átviteli függvényével, az állapotteres leírásával valamint a rendszeregyenletével.

2.3.1. Nem paraméteres leírás

Nem paraméteres leírásoknál a rendszer valamilyen jelre adott válaszát szoktuk felhasználni. Ez tipikusan az impulzusválasz, vagy az egységugrás gerjesztésre adott válasza. A legelterjedtebb az impulzusválasz, hiszen így a teljes rendszert le is írjuk vele, és nem kell további transzformációkat végezni, hogy a teljes rendszert leíró alakot kapjuk, mint az egységugrásra adott válasz esetében[1]. Az ilyen leírásokat a szakirodalomban *kernel*nek szokás nevezni.

Folytonos időben az impulzuzválasz segítségével a rendszer bármely jelre adott válasza számolható - ezért is írható le vele teljes egészében a rendszerünk. Ha $h(\tau)$ a rendszerünk impulzusválasza, akkor tetszőleges bemenetre a válasz megadható a konvolúciós integrállal:

$$y(t) = \int_0^T h(\tau)u(t-\tau)d\tau \tag{2.4}$$

ahol T a vizsgálódás időtartama. A konvolóciót a 0-ból indítjuk, mert feltételezzük, hogy a vizsgált rendszerünk kauzális[2].

Mivel további vizsgálódásainkat diszkrét időben fogjuk végezni, ezért a konvolúciós integrált át kell konvertálni diszkrét időbe, aminek az eredménye a következő lesz.

A jelölésmód is változni fog. Ezentúl a $x(t)$ egy diszkrét időbeli jelet fog leírni, ahol t egész értékeket vehet fel. Az áttérés haszna, hogy a további képletekben mind az n és a k jelöléseket futóindexként tudjuk használni.

$$y(t) = \Delta t \sum_{k=0}^{N-1} h(t)u(t-k) \tag{2.5}$$

A fenti képletben a Δt jelenti a mintavételezési időt. Mi a továbbiakban ezt egységnyinek fogjuk venni, ezért a képletekből elhagyjuk.

Mivel a kernel egy jelre adott válasz, ezért a hossza rendszertől függően változó lehet, de általánosan elmondható, hogy ugyanaz a rendszer kompaktabban leírható paraméteresen, mint nem paraméteresen.

2.3.2. Paraméteres leírás

A lineáris rendszerek paraméteres leírása többek között differenciál és differencia egyenleteket jelent. Jelen esetünkben mi csak a differenciaegyenleteket vesszük számításba, hiszen kizárólag diszkrét időbeli rendszereket vizsgálunk.

Egy differenciaegyenlet a rendszer bemenetei és kimenetei között írja le az összefüggést,

[1] Az áttérés a két alak között egyszerű differenciálásban merül ki. Ez bevett szokás olyan esetekben, ahol az impulzus-gerjesztés előállítása nehezen vagy egyáltalán nem kivitelezhető.

[2] A feltételezés minden valós rendszerre jogos, hiszen a természetben nincs olyan rendszer, ami behatás nélkül cselekedne. Akauzális rendszerek általában ott találunk, ahol az adatsor előre megléte biztosított, ezért az offline feldolgozás során megengedhetjük az akauzalitást. Ilyenek a hangzásokat kiegyenlítő digitális megoldások, és a hangvisszadó rendszerek [52].

amit más néven rendszeregyenletnek is nevezünk. A dolgozatban csak a legegyszerűbb esetet vizsgáljuk, mikor egy bemenete és egy kimenete van a vizsgált rendszerünknek. Egy ilyen rendszeregyenletet láthatunk alább:

$$y(t) + a_1 y(t-1) + \cdots + a_{n_a} y(t-n_a) = b_0 u(t) + b_1 u(t-1) + \cdots + b_{n_b} u(t-n_b) \quad (2.6)$$

$$A(z) = 1 + a_1 * z^{-1} + \cdots + a_{n_a} * z^{-n_a} \quad (2.7)$$

$$B(z) = b_0 + b_1 * z^{-1} + \cdots + b_{n_b} * z^{-n_b}, \quad (2.8)$$

ahol a z^{-1} az egységnyi késleltetés operátora. Ha élünk ezzel a behelyettesítéssel, akkor a rendszeregyenlet a következő alakot ölti:

$$y(t) = B(z)U(z) + [1 - A(z)Y(z)] \quad (2.9)$$

Ez a modelleírás a lineáris rendszereket leíró modellek egyik speciális esete. Ez a modell az úgynevezett ARX (*auto-regressive exogenous*) modell, ahol a zajt, mint bemenetet figyelmen kívül hagyjuk. Ezt a modellt a szakirodalom IIR (*infinite impulse response*) rendszernek is nevezi, mivel a modell kimenetére hatással van annak késleltetett értéke/értékei is, ezért a modell válasza végtelen hosszú.

A modellt tovább egyszerűsítve eljutunk a szakirodalom által FIR (*finite impulse response*) modell alakjához, aminek a kimenete csak és kizárólag a rendszer bemenettől és annak késleltetett mintáitól függ:

$$y(t) = B(z)U(z). \quad (2.10)$$

Egyetemi tanulmányaink során ezzel a két rendszerrel ismerkedtünk meg, azonban ezeknél léteznek jóval bonyolultabb leírások is. Ha számításba vesszük a rendszerben keletkező zajt is, ami rászuperponálódik a kimeneti jelre, akkor megkapjuk az általánosabb ARMAX modellt (*auto-regressive moving average exogenous*)

$$y(t) = B(z)u(t) + [1 - A(z)y(t)] + C(z)e(t), \quad (2.11)$$

ahol $e(t)$ a rendszerben lévő zajkeltő folyamat, és $C(z)$ a zajfolyamatot szűrő (színesítő) rendszer paramétereit tartalmazó polinom.

Számításba vehetjük még a bemenet érvényre jutásának késleltetését is. Ekkor minden bemeneti mintát meghatározott mértékben késleltetünk.

$$y(t) = z^{-d}B(z)U(z) + [1 - A(z)Y(z)], \quad (2.12)$$

ahol d mintányi idő múlva lép csak érvénybe a rendszer bemenete.

2.4. Nemlineáris rendszerek leírása

A nemlineáris rendszerek leírására is számos lehetőségünk van. A következőkben a nem paraméteres és a blokkos leírást vesszük szemügyre.

2.4.1. Nemparaméteres leírás

Ahogy azt (2.4)-es egyenlet is mutatta, a lineáris rendszerek leírhatóak az impulzusválaszuk segítségével. Volterra [65, 54] ezt továbbfejlesztette, és egy általánosított leírást alkotott, amiben az impulzusválaszt nem paraméteres kernelek integráljának a sorozatára cserélte le.

$$y(t) = \sum_{n=0}^{\infty} \int_0^T \dots \int h_n(\tau_1, \dots, \tau_n) u(t-\tau_1) \dots u(t-\tau_n) d\tau_1 \dots d\tau_n \tag{2.13}$$

A nulladik fokú tag konstan 1 érékű, és semmilyen bemeneti jeltől nem függ.

Az elsőfokú tag hasonló a lineáris rendszerekre jellemző konvolúciós képletre, hogy lineáris rendszerek esetén a Volterra soros kifejtés egyenlő a hagyományos konvolúciós képletben megismert alakkal.

Másodfokú esetben már bonyolódik a helyzet, és a következő alakot ölti:

$$\int_0^T \int_0^T h_2(\tau_1, \tau_2) u(t-\tau_1) u(t-\tau_2) d\tau_1 d\tau_2 \tag{2.14}$$

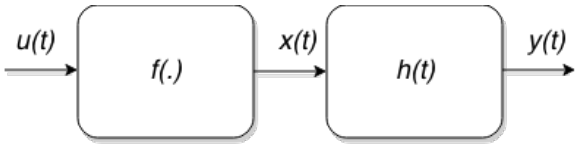
Látható, hogy a leírás a fokszám növekedésével kezd egyre több paramétert és kernelt tartalmazni, így gyakorlati szempontból a használata túl számításigényes, az identifikáció nehézkes, így leginkább elméleti jelentőséggel rendelkezik.

2.4.2. Blokkos leírás

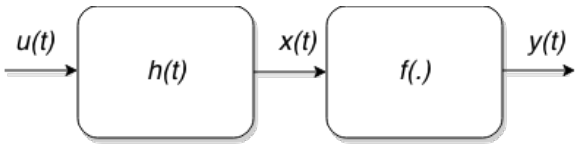
Gyakorlati szempontból sokkal jelentősebb nemlineáris rendszereket modellező forma a blokkos rendszerleírás [46, 16, 55, 14, 33]. Ebben az esetben az egyes blokkok reprezentálják külön-külön a lineáris és a nemlineáris részegységeket. Egymással nincsenek paraméterszinten kapcsolatban, csupán az egyik kimenete a másik bemenete lesz.

A blokkok lehetnek statikusak és dinamikusak. Mivel nemlineáris rendszereket szeretnénk leírni a modellekkel, ezért a lineáris és a nemlineáris részt érdemes szétválasztani. Az egyszerűség kedvéért a nemlineáris blokkokról feltételezzük, hogy nem tartalmaznak dinamikát, azaz csupán egy statikus hozzárendelésről van szó. A lineáris blokkok a rendszer dinamikájáért lesznek felelősek. Lehetne bonyolítani a leírást dinamikus nemlineáris tagokkal, de ez túlmutat a dolgozat keretein.

A legegyszerűbb nemlineáris rendszereket is leíró blokk alapú modellek a Hammerstein és Wiener-rendszerek. A 2.1. ábrán látható a Hammerstein-rendszer, a 2.2. ábrán pedig



2.1. ábra. *Hammerstein-rendszer*



2.2. ábra. *Wiener rendszer*

a Wiener-rendszer blokkvázlata. Látható, hogy Hammerstein esetben a statikus nemlinearitás blokkja megelőzi a lineáris tag blokkját, míg a Wiener-rendszer esetében ez pont fordítva van.

A blokkos modellek gyakorlati jelentősége óriási. Míg a nemparaméteres leírások nagy számításigénnyel járnak, addig a blokkos leírásoknál ez hatékonyan megoldható, kis számításigényű feladatot jelent. Identifikációjuk lesz a dolgozat egyik fő témája.

3. fejezet

Hangszórók

Ahogy a diplomamunka címe is előrevetíti, a dolgozatban hangszórók viselkedését fogjuk modellezni.

A hangrögzítés és -visszaadás kezdeti két fázisa után (F.3. fejezet), a modern hangvisszaadó rendszerek népszerűségének növekedésével fejlődtek az elektrodinamikus hangszórók is. A kezdeti gyenge próbálkozások után fokozatosan egyre kiforrottabb konstrukciókat fejlesztettek ki, míg végül ki nem alakult a mai hangszórókra jellemző konstrukció.

Ebben a fejezetben megismerkedünk a legelterjedtebb hangszórótípussal, az elektrodinamikus hangszóróval. Felépítésének áttekintése után tüzetesebben megvizsgáljuk a tökéletlenségeit, torzításainak forrásait.

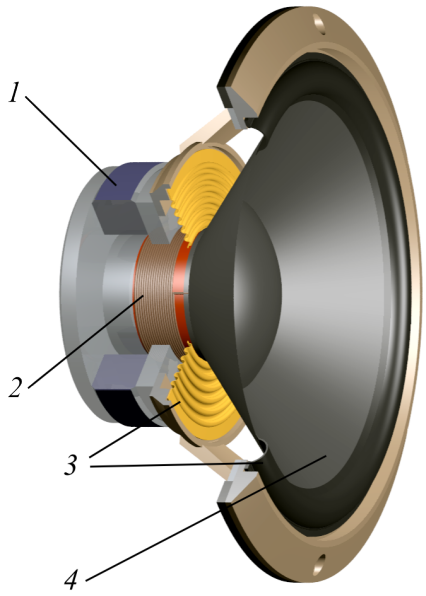
A fejezet végére az Olvasó tisztában lesz minden olyan hatással, ami a hangminőség romlását okozza. A fejezetben közölt információkat Wolfgang Klippel áttekintő cikke [40] alapján írtam.

3.1. Az elektrodinamikus hangszóró felépítése

A hangszórók és más hangnyomást keltő eszközök lényegében olyan fizikai rendszerek, amelyek a bemeneti jelként rájuk adott feszültségjelet hangnyomássá alakítják. Ezt több lépésben teszik meg.

Az elektrodinamikus hangszórók a bemeneti feszültséget először árammá, majd az elektromágneses indukció által kitéréssé és erőhatássá, végül egy membrán segítségével légnyomássá alakítják.

A hangszórók másként viselkednek kis és nagy amplitúdók esetén. Az amplitúdófüggő viselkedés okai magában a hangszóró kialakításában keresendőek. Ezen viselkedés mellett a hangszóróknak más nemlineáris torzításai is hozzájárulnak a kimeneti hanghullámok és a gerjesztés spektrális tartalmának eltéréséért, ami jellemzően harmonikus torzításban is megjelennek. Színuszos jel esetén a frekvencia egész számú többszörösei jelennek meg a keltett hang spektrumában. A hangszóró felépítése után megnézzük, hogy a felépítésből adódóan milyen jelenségeket figyelhetünk meg a hangszóró működése közben.



3.1. ábra. *Elektrodinamikus hangszóró felépítése [7]*

A 3.1. ábrán látható egy tipikus elektrodinamikus hangszóró szerkezeti képe. Nézzük meg, hogy mi az egyes elemek feladata és funkciója.

1. Mágnes - állandó mágneses teret hoz létre ahhoz, hogy a tekercsre vonzó vagy taszító erőhatást fejtsen ki. A mágnes a hangszóróban fix pozíciót foglal el, a tekercs hozzá képest tud elmozdulni. Az ábrán is látható, hogy a mágneshez további fémalkatrészek csatlakoznak: a vasmag és a póluslap. Ezek az alkatrészek alakítják ki a lengőtekercs mozgásterét és annak környezetében a megfelelő mágneses teret, valamint segítenek a tekercs hőelvezetésében is.[1]

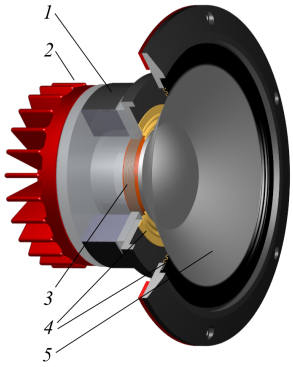
2. Lengőtekercs - korlátozott irányban szabad elmozdulásra képes tekercs. Szerepe a hangszóróra adott feszültség kitéréssé alakítása. Ehhez először impedanciája révén a feszültség hatására áram folyik rajta, ami mágneses teret kelt.

3. Felfüggesztés - korlátozza a tekercs és a membrán mozgását egy fix irány mentén, és visszatérítő erőt biztosít, hogy nyugalmi állapotban a lengőtekercs a kiindulási helyzetébe kerüljön vissza. A hangszóróban a lengőtekercs és a membrán külön felfüggesztéssel rendelkezik.

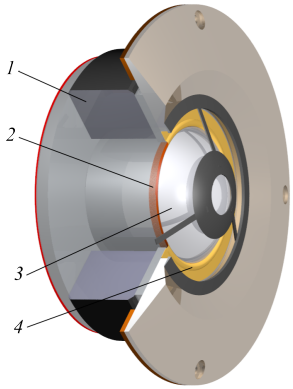
4. Membrán - felülete által a kitérést hangnyomássá alakítja azáltal, hogy pozitív vagy negatív nyomást hoz létre mozgásával.

Ahogy az a 3.2. és 3.3. ábrán is látható, a hangszóróknak több fajtája van. Ennek az az oka, hogy egy hangszóróval nem lehet olyan jól lefedni a teljes frekvenciatartományt.

[1] Néhány magassugárzónál a további hőelvezetés érdekében *ferrofluidot* is alkalmaznak, hogy a termikus ellenállás még kisebb legyen a tekercs és a hőelvezető fémalkatrészek között.



3.2. ábra. *Középsugárzó felépítése [4]*



3.3. ábra. *Magassugárzó felépítése [8]*

Alacsony frekvenciákon a hangszórók membránjának sokkal nagyobb levegőtérfogatot kell megmozgatnia azonos hangnyomás keltéséért, mint a magasabb frekvenciák esetében, így az alacsonyabb hangtartományt csak a nagyobb átmérőjű membránok képesek megfelelő módon átadni. A nagyobb membránok azonban, ahogy haladunk felfelé a frekvenciában, úgy egyre inkább irányítottan kezdenek sugározni (elveszítik pontforrás-jellegüket). Ez a jelenség az oka annak, hogy egyhangszórós rendszereknél csak egy bizonyos szögben jó a magas és a mély hangok aránya. Ez fejhallgatóknál és fülhallgatóknál nem probléma, hiszen ott a kialakításuknak köszönhetően a legjobb irányban adják le a hangokat a fülbe, azonban egy teremben, ahol a hallgató nem feltétlenül tartózkodik mindig a hangszóróval szemben, ez a viselkedés már megengedhetetlen a jó hangzás érdekében. A probléma orvosolható, ha szétválasztjuk a különböző frekvenciasávokat, és más konstrukciójú hangszórókat használunk alacsony és magas hangokra.

A rezonancia a másik hatás, ami szintén közrejátszik az eltérő hangszórótípusok jogosultságában. Ahogy minden valós fizikai rendszernek, úgy egy adott hangszóró-konstrukciónak is léteznek saját frekvenciái, ahol az átvitel erősen kiemel. Ebből az okból kifolyólag érdemes a hangszórók által általában lefedett frekvenciatartományt felbontani több részre, és mindegyik résztartományban egy külön konstrukciójú hangszórót alkalmazni, aminek a rezonanciafrekvenciája kívül esik ezen a tartományon.

A hangszórókat ezért több kategóriába szokás sorolni: mélysugárzó, középsugárzó és magassugárzó hangszórók. Nevükből is adódik, hogy milyen frekvenciasávra optimalizált a felépítésük.

3.1.1. Nemlineáris viselkedés

A hangszórók felépítéséből adódóan torzítják a gerjesztésként rájuk adott jeleket. Ez esetenként egy kívánt hatás, ahogy azt a fejezet végén is tárgyaljuk, de az esetek túlnyomó többségében igyekszünk elkerülni. Az emberi fül sajnos nagyon érzékeny a torzításokra, könnyen észrevesszük a legkisebb nemkívánt eltéréseket is.

A leginkább hallható torzításokért a következő hatások a felelősek [40]:

- Felfüggesztés kitérésfüggő merevsége
- Lengőtekercs kitérésfüggő induktivitása
- Kitérítő erő kitérésfüggése
- Membrán és keret Young-modulusának nemlinearitása
- Geometriai transzfer mátrix változása vibráció hatására
- Doppler effektus

3.1.1.1. Felfüggesztés kitérésfüggő merevsége

A hangszórókban található felfüggesztések dupla célt szolgálnak. Egyrészről lekorlátozzák a lengőtekercs és a membrán mozgástartományát, hogy csak a hangszóró hossztengelyére merőleges irányba tudjanak mozogni. Ezzel meggátolják az esetleges oldalirányú és csavaró erőhatások érvényrejutását, másrészről visszatérítő erőt biztosítanak a lengőtekercsnek és a membránnak, hogy nyugalmi állapotban a kiindulási helyzetbe térjenek vissza.

A felfüggesztések kis kitérések esetében közel lineáris visszatérítő erőt biztosítanak, hasonlóan egy lineáris rugóhoz, azonban nagyobb kitéréseknél a felfüggesztés nagyobb erővel válaszol, mint azt a lineáris rugó-modellünk feltételezi. A felfüggesztések által alkotott rendszer nagy kitérések esetében úgymond mechanikusan telítődik.

A legtöbb hangszórónál a lengőtekercs és a membrán is rendelkezik külön felfüggesztéssel, ezért a két mechanikus telítődésnek össze kellene adódnia. Azonban ez nem így van, hiszen a membrán a rugalmasságából adódóan - ahogy azt egy későbbi pontban tárgyalni is fogjuk - a lengőtekercs kitérésére nézve aluláteresztő szűrőként viselkedik, így a két nemlineáris rugalmassági hatás a hangszóróra adott frekvenciától függően fog összeadódni.

Összegezve: ez a hatás az egyik legjelentősebb kitérésfüggő nemlinearitás, ami a hangminőség romlását okozza.

3.1.1.2. Lengőtekercs kitérésfüggő induktivitása

A hangszórók bemenőimpedanciája függ a lengőtekercs kitérésétől. A jelenség könnyen belátható. Maga a hangszóró nem más, mint egy vasmagos tekercs, és egy tekercs induktivitása függ a benne lévő vasmag helyzetétől, hiszen a tekercset érő fluxust a vasmag helyzete nagymértékben befolyásolja[2]. A rádiótechnikában ezt kihasználva, hangolható tekercsekkel állítják be a rezgőkörök rezonancia frekvenciáját.[3]

Ha a bemeneti impedancia változik a kitérés függvényében, akkor változik a hangszóróra adott feszültség hatására a tekercsen folyó áram nagysága is, ami pedig hatással van a kitérés mértékére, azaz az impedancia változását okozó hatásra. Belátható, hogy a hangszóró egy rendkívül összetett, visszacsatolásokat tartalmazó rendszer, nemkívánt tulajdonsággal.

3.1.1.3. Kitérítő erő kitérésfüggése

A hangszóró lengőtekercse az állandómágnes és az azt körülvevő fém alkatrészek által kialakított mágneses térben mozog. Ez a tér nem homogén, mivel a rés, amiben a tekercs mozog, valamint a tekercs is véges hosszúságú. Ez a két korlátozott hosszúság miatt előfordulhat olyan kitérítési helyzet, amikor a tekercs kimegy a homogénnek tekinthető mágneses térből, a mágneses tér kisebb lesz, és ezáltal a homogén térbe való visszatérésig más erők fognak hatni a tekercsre azonos gerjesztés hatására.

A hatás legrelevánsabb esete, mikor egyszerre szólal meg kis- és nagyfrekvenciás hang. A kisfrekvenciás hanghoz nagy kitéréseket kell a hangszórónak generálnia, így a kitérés szélsőértékeinél előfordulhat, hogy a tekercs már nem a homogén mágneses térben fog tartózkodni. A nagyfrekvenciás hang így más kitérítési erőhatásokat fog produkálni a tekercsre a kisfrekvenciás hang fázishelyzetétől függő módon, mintegy amplitúdó-modulálva a nagyfrekvenciás hangot.

A hatás csökkentése érdekében bevett szokás, hogy hosszabb lengőtekercseket alkalmaznak, mint a vasmag által alkotott légrés hossza, így a tekercs közel hasonló mágneses térben mozoghat a kitérés szélsőértékeinél is.

3.1.1.4. Membrán és keret Young-modulusa

Az eddigiektől gyengébb nemlineáris hatás is megfigyelhető, ami az anyag mechanikai tulajdonságaival hozható összefüggésbe. Kisfrekvenciákon, mechanikai szempontból csak a dugattyúszerű mozgás következtében alakul ki nemlineáris hatás[4], azonban nagyobb frekvenciákon, mikor a hangszóró anyagában jelentős mechanikai feszültség lép fel, az anyag

[2] A vasmagot a fluxus vezetésére, irányítására használjuk.

[3] Változtatható induktivitást mechanikailag sokkal könnyebb készíteni, mint változtatható kapacitást, ezért is használnak inkább minden lehetséges helyen változtatható induktivitást. Felépítését tekintve ez nem más, mint egy menetes csévetestre feltekert kábel, amibe egy csavarszerű ferritet csavarnak. A behajtás mértékétől függ a konstrukció induktivitása. Az állításához csak egy csavarhúzó szükséges.

[4] Felfüggesztés kitérésfüggő merevsége, lengőtekercs kitérésfüggő induktivitása, kitérítő erő kitérésfüggése.

mechanikus tulajdonságai képesek megváltozni. Ilyen változás figyelhető meg az anyagok Young-modulusában[5], ami a mechanikai feszültségtől függővé válik, és ezáltal a rezgést nemlineárisan befolyásolja.

3.1.1.5. Geometriai nemlinearitás vibráció hatására

Az előzőtől lényegesebb torzítást eredményez a hangszóró anyagának viselkedése rezgés hatására. Szilárd anyagoknál a rezgés még konstans Young modulus esetén is nemlineárissá válik akkor, ha az anyag rezgésből adódó kitérésének mértéke nem lesz elhanyagolható az anyag kiterjedésbeli méretéhez képest.

Ilyen hatás éri a membránt, mikor nagyfrekvenciás hangok kiadása közben a membrán anyagának hullámzási módusainak méretei kezdenek összemérhetőek lenni a membrán vastagságával.

3.1.1.6. Doppler-effektus

A Doppler-effektus is szerepet játszhat a hangszórók torzításában. Képzeljük csak el, hogy egy hangszórón egyszerre szólaltatunk meg egy nagyon mély és egy nagyon magas hangot. Tudjuk, hogy a mély hanghoz sokkal nagyobb kitérést kell a hangszórónak produkálnia, mint a magas hanghoz. Azaz a kisfrekvenciás, nagy amplitúdós hullámzásra rászuperponálódik a nagyfrekvenciás, kis amplitúdójú hullámzás. Ez a hallgató számára úgy nyilvánul meg, hogy a mély hang frekvenciájától függően minimális mértékben hol magasabbnak, hol alacsonyabbnak hallja a magas frekvenciás hangot. A jelenség könnyen beláthatóan a Doppler-effektus.

A nagyfrekvenciás, kis amplitúdójú hang szempontjából a hang forrása mindig az aktuális mebránállás. Mély frekvenciás hangok hatására a membrán kitérése nagy értékeket képes felvenni. Szélső értékeknél, amikor a membrán kitérése minimális és maximális - sebessége nulla -, a nagyfrekvenciás hang ténylegesen hallott frekvenciája a hangszóróra adott frekvencia lesz. Azonban abban a pillanatban, ahogy a membrán a kisfrekvenciás hang fázisa szerint elkezd gyorsulni, sebességre tesz szert, és a nagyfrekvenciás hang ténylegesen hallott frekvenciája kisebb vagy nagyobb lesz, attól függően, hogy a hallgatóhoz képest távolodik vagy közeledik. A fázis és ezáltal a frekvencia modulálódik a kisfrekvenciás hang frekvenciája és amplitúdója függvényében.

A hatás csökkenthető, ha többutas rendszereket alkalmazunk, amelyekben a magas és a mély tartományok egymástól szét vannak választva.

3.1.1.7. További nemlineáris hatások

A fent említett hatások mellett hangvisszaadó rendszerekben még szerepet kap, a hangszóró dobozolásának a viselkedése, és a hangszóróhoz közvetlenül nem köthető külső hatások is.

[5] A Young-modulos megmondja a szilárd anyagokról, hogy mennyire merevek. Ez szerkezeti anyagoknál a legtöbb esetben terheléstől, azaz mechanikai feszültségtől független, azonban műanyagoknál és gumiknál a rugalmassági modulus nemlineáris módon függhet a mechanikai feszültségtől.

Ilyen hatás a bassreflexes hangszórókban a szellőző lyuk levegőellenállásának nemlineáris változása a frekvencia függvényében, vagy a hanghullámok terjedése nagy amplitúdóknál. Ekkor a pozitív maximumnál gyorsabban terjed a hang mint a negatív maximális kitérésnél, így a hanghullám torzul.

Több hangszórót tartalmazó rendszerekben további említésre méltó torzítást okoz maga a keresztváltó, ami a hangszórórendszerre adott gerjesztést hivatott frekvenciasávok szerint szétosztani. A keresztváltó felépítését tekintve egy passzív szűrőrendszer, ami a bemeneti jelét kettő vagy több frekvenciatartományra osztja. Két tartomány esetén egy felül- és egy aluláteresztő passzív szűrőről van szó. Kettőnél több tartomány esetén sávszűrőkről beszélünk. A passzív szűrők tekercsekből és kondenzátorokból állnak. Tekercsek esetén a ferritmag okozza a nemlineáris viselkedés legjavát. A probléma a ferritmag hiszterézises viselkedése adja, ami képes telítődésre is. Kondenzátoroknál az elektrolit kondenzátor okozza a szűrő nemlineáris viselkedését. Ezért többek között a kondenzátor polaritása, permittivitásának térerősség-függése és piezoelektromos hatása a felelős.

Természetesen említeni sem kell, hogy a hangszórók fizikai sérülés hatására nem lesznek képesek a hangot torzítás nélkül visszaadni, így ezt a lehetőséget nem is vizsgáljuk meg tüzetesebben.

3.1.2. A hangszóró hatásfoka

Ismeretes, hogy a hangszórók hatásfoka az összes gyakorlatban alkalmazott átalakítóhoz (*transducer*-hez) képest az egyik legrosszabban teljesítő. A hangszóróba betáplált energia csupán 3-5 százalékát alakítja hangenergiává, a többi hő formájában a hangszóró konstrukciója és tökéletlensége miatt kárba vész. A rendkívül kis hatásfok oka a hangszóró és a levegő impedanciaillesztésében keresendő. Mindenki elvégezheti maga is a következő kísérletet: próbáljunk meg minél erősebben megütni egy szabadon lógó papírzsebkendőt. Nem tudjuk erősen megütni, hiszen kis ellenállást fejt ki rá a levegő. Így van ez a hangszórók esetében is. Nem tudnak elég nagy erőt átadni a levegőnek. Eltérő konstrukcióval jobb hatásfokot is el lehet érni. Az F.3. fejezetben tárgyalt hangtölcsérekkel 25-50 százalékos hatásfok is elérhető.

3.1.3. Tipikus nemlineáris alkalmazások

Vannak a gyakorlatban olyan esetek, amikor egy hangszórót tudatosan a nemlineáris tartományában használunk, vagy azért, mert másképpen nem tudjuk működtetni, vagy mert direkt a nemlineáris torzítását kívánjuk felhasználni, hogy színesebbé tegyük a visszaadott hangot.

Az az eset, amikor nem tudjuk máshogy felhasználni a hangszórót, mint a nemlineáris tartományában, tipikusan a mobiltelefonok és hordozható kisméretű eszközök esetében van. Ekkor a lehető legnagyobb hangerőt akarjuk kipréselni a lehető legkisebb hardverből,

aminek az eredménye torzított hang lesz. Ez a legtöbb esetben nem is zavar minket, hiszen az elvárásaink alacsonyak az ilyen esetekben, de olykor igen zavaróak tudnak lenni, amikor abban a frekvenciatartományban, ahol a számunkra fontos információ vagy zenei részlet van, megjelenik egy zavaró frekvencia.

Az elektromos gitárok világa egy érdekes terület, ugyanis a teljesen tiszta és lineáris hangvisszaadást kevésbé kedvelik, mint a torzítottat. Ennek oka abban rejlik, hogy az elektromos gitár hangja teljesen tisztán és torzítatlanul cincogó és gyenge. Ha próbáltunk már elektromos gitárral játszani mondjuk egy szintetizátor erősítőn, majd az erősítőt lecseréltük egy direkt gitárhoz valóra, akkor a különbség azonnal szembetűnő. A szintetizátor esetében a lényeg, hogy a hangszíneket a lehető legtisztábban hangosítsuk. Az erősítő lényege csupán a hangerő megtöbbszörözésében van, mást nem várunk el tőle. Minden hangzásbeli beállítást az erősítő előtt a hang szintetizálásánál végzünk el.

Elektromos gitároknál ez teljesen máshogy van. Itt a cél, hogy az erősítő ne csak felhangosítsa a gitár tiszta hangját, hanem hogy színt is vigyen bele, testet adjon neki. Ezt az erősítő nemlineáris torzítással éri el, ami által felharmonikusokat generál a bemenő hanghoz, és így a hangzás jóval gazdagabb, érdekesebb lesz.

A kezdeti időkben a gyártók nem készítettek 15-30 wattnál nagyobb teljesítményű hangszórókat, hiszen a létező erősítők sem adtak le nagyobb teljesítményt. Azonban az 1950-es években a Fender kiadta az első nagyobb teljesítményű erősítőjét a 80 wattos Fender Twint, ami jóval nagyobb teljesítményt tudott belepréselni a hangszórókba, mint amire tervezték azokat. Ez kis jelszinteken nem volt probléma, azonban teljesen kihajtva, a gitár hangjára jelentős hatást gyakorolt a hangszóró túlvezérelt, nemlineáris viselkedése. A hang torzított és kompresszált lett, ami az alapja volt a kor rock és blues irányzatainak.

4. fejezet

Identifikáció

Az identifikáció röviden egy megismerési folyamat, ahol egy ismeretlen rendszer belső működését próbáljuk megérteni, és visszaadni egy modell segítségével.

Rendszeren egy kívánt hatást elérő folyamatot értünk. Ez lehet egy gyár vagy egy vegyimű, ahol a nyersanyagokból különböző lépések által késztermékeket kapunk. Lehet egy akusztikai rendszer, ahol a bemenet tetszőleges hang, a kimenet pedig a bemenet visszhangos változata, de lehet akár egy gazdasági folyamat is, aminek a bemenetei különböző politikai döntések, gazdasági lépések, kulturális események, a kimenete pedig az árfolyamok aktuális értéke. A fenti példák mindegyikére felvehető a rendszer viselkedését utánozni hivatott modell, amit a megfigyeléseink alapján alkotunk. Egy modell feladata, hogy leegyszerűsítse a valóságot. Csak a számunkra releváns mértékben szükséges - és kell is - bonyolítani a rendszert leíró modellt. A modell nem feltétlenül fedi le a valóság minden részletét, sőt, lehet, hogy csak a számunkra érdekes tartományon ad értékelhető eredményt.

Az identifikáció egy iteratív folyamat, ami öt jól elkülöníthető lépésre bontható [51, 57].

1. **Kísérlet tervezése** - Ebben a lépésben fontoljuk meg, hogy mi legyen az a kísérlet, amit ha elvégzünk az ismeretlen rendszeren, a lehető legtöbb releváns információt kapjuk a modellünk felállításához. Ilyenkor tudatosan olyan gerjesztést készítünk elő, ami a rendszert abban az állapotában tartja meg, amit később modellezni akarunk. Ellenkező esetben a moddellezés számára irreleváns információkat is gyűjtenénk. Felhasználhatunk előzetes információkat (a priori ismereteket), amelyekkel célirányosabban tudunk tervezni.

2. **Kísérlet elvégzése, adatgyűjtés** - Fontos, hogy a megtervezett kísérletet el is tudjuk végezni. Legyen megfigyelhető a rendszernek azon pontja, amiből az információt kívánjuk gyűjteni. Az információgyűjtés módja is fontos. Legyen megbízható, a kísérletet pedig célszerű többször elvégezhetőre tervezni, hiszen az identifikáció legtöbbször egy iteratív folyamat.

3. **Modellstruktúra megválasztása** - Ebben a lépésben megválasztjuk, hogy a gyűjtött adatokból milyen modellt kívánunk felállítani. Ahogy az első lépésben is, itt

is segítségünkre lehet a priori ismeret a struktúra kiválasztásában. Ha nem jól választunk, az az iteráció végén úgyis kiderül, tehát nem kell elsőre a legbonyolultabb modellt kiválasztani. Bevett szokás, hogy kezdésnek a lehető legegyszerűbb modellt választják, majd egyre jobban bonyolítják azt (tipikusan fokszámot növelnek). A modell bonyolultsága jelentősen függ attól, hogy mire akarjuk felhasználni. Elmondható, hogy két út között lehet dönteni. Ha az a cél, hogy vezéreljünk egy rendszert a modellünk felhasználásával, és a modell csak megfigyelésre kell, akkor célszerű a lehető legegyszerűbbet választani, ami már teljesíti az összes elvárásunkat. Ha az identifikációval a célunk egy ismeretlen rendszer minél jobb megismerése, akkor modellnek a lehető legbonyolultabbat válasszuk, amit még igazol a kísérlet során gyűjtött adatsorunk.

4. **Módszerválasztás, paraméterbecslés** - Minden szükséges előkészületet megtettünk ahhoz, hogy elindíthassuk a tényleges identifikációs folyamatot, módszert, amivel a mért adatok alapján feltöltjük a modellünket a rendszerre jellemző paraméterekkel. Ez lehet online vagy offline feldolgozás a rendszertől és a modelltől függően. A paraméterbecslés nem más, mint egy hibafüggvény minimalizálása. Ekkor a rendszer és az elvárt kimenet között felállítunk egy költségfüggvényt, ami a minimalizálás tárgya lesz. A minimalizáló módszer nagyban befolyásolja az identifikáció sikerességét.

5. **Modell ellenőrzése** - Ha az identifikáció lezajlott, akkor rendelkezünk egy modellel, ami valószínűsíthetően olyan paramétereket tartalmaz, amelyekkel képesek vagyunk hűen leírni a rendszerünket. Ezt a legegyszerűbben egy összehasonlítással tudjuk eldönteni, ami lehet időtartománybeli, vagy frekvenciatartománybeli is. Egy új gerjesztőjel használata szükséges ahhoz, hogy biztosak lehessünk az eredményben (ugyanarra a jelre az identifikációs eljárás miatt helyes eredményt adna az összehasonlítás. Csak akkor szabad elfogadni a modellt, ha az egy teljesen független, új bemenő jelre is az elvárt tulajdonságokat produkálja).

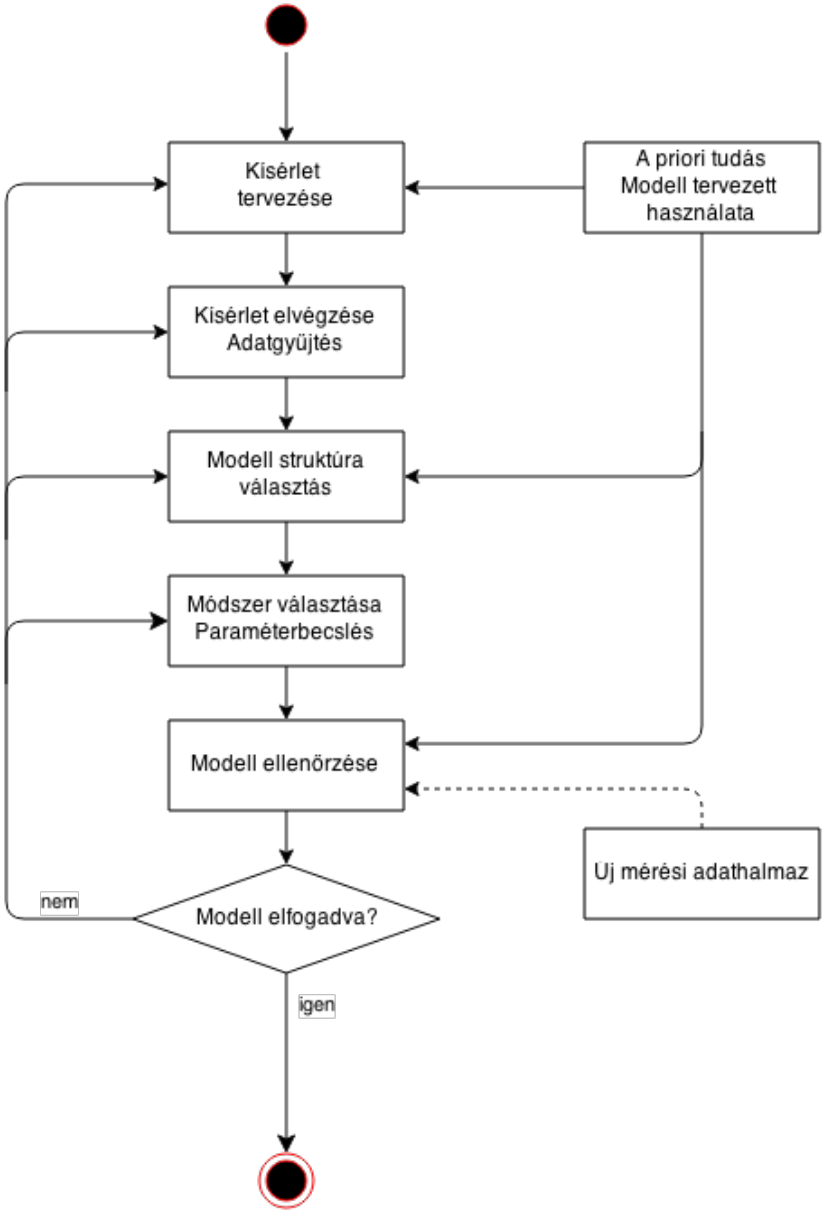
Ahogy az a 4.1 ábrán is látható, az identifikáció egy iteratív folyamat, a számolt modell tesztelése után derül ki, hogy szükséges-e még módosítani rajta.

4.1. Online és offline megközelítés

Az identifikáció elvégzésére alkalmazási területtől függően kétféle lehetőségünk van:

- Offline eljárásnak nevezzük azt az identifikációs módszert, amikor minden mérési adatunk egyszerre rendelkezésre áll, és az alkalmazott modell engedi az egy lépésben történő számolást. Ekkor minden ismeretünket a megfigyelt rendszerről egyszerre használjuk fel az eredmény elérése érdekében.

- Online identifikációról abban az esetben beszélünk, ha a rendszer megfigyelési adataihoz nem férünk hozzá egyszerre. Ilyen tipikusan akkor fordul elő, amikor működése közben identifikálunk. Ekkor természetesen nem lehet arról szó, hogy megvárjuk, mire az identifikáló algoritmus elegendő adatot gyűjt össze, majd egyszer lefuttatjuk



4.1. ábra. *Az identifikáció lépései*

az identifikációt, hanem minden új adat beérkezésekor el kell végezni a szükséges számításokat. Ezt a módszert tipikusan adaptív vezérléseknél alkalmazzák, ahol a szabályzókör adaptívan képes reagálni a megfigyelt rendszer változásaira.

4.2. Lineáris regresszió

Lineáris regresszió az egyik legegyszerűbb offline identifikációs módszer. Az eljárás központi elemében a pszeudóinverz [28] játszik szerepet.

A pszeudóinverz, a hagyományos mátrix inverzének általánosítása. Segítségével megoldhatóvá válnak szinguláris vagy nem négyzetes mátrixos invertálását megkövetelő feladatok is. Ilyen módon közelítően invertálni tudunk amúgy nem invertálható mátrixokat, least

sqaures megoldást tudunk mondani túldefiniált egyenletrendszerekre. Mi az utóbbi alkalmazást használjuk a továbbiakban.

4.2.1. Identifikációs feladat

Tegyük fel, hogy van egy folytonos statikus nemlinearitásunk, amit közelíteni szeretnénk hatványfüggvények összegeként. A modellünk legyen a következő alakú:

$$y(t) = \sum_{k=0}^{n_f} f_k u^k(t), \tag{4.1}$$

ahol $y(t)$, $u(t)$ és f_k rendre a modell kimenete, bemenete és paraméterei. A képletből látható, hogy a modellünk konstans tagtól kezdve minden felsőbb fokú hatványt tartalmaz. Az identifikálandó paramétereink száma a maximális fokszámnál eggyel nagyobb.

Adjunk az ismeretlen rendszerünk bemenetére gerjesztőjelet, és mérjük a bemenetét. Azonban mielőtt ezt megtennénk, írjuk fel képletesen, hogy mi is fog történni:

$$y(t) = f_0 + f_1 x(t) + \cdots + f_{n_f} x^{n_f}(t). \tag{4.2}$$

A fenti egyenlet nem más, mint az előző, tömörebb egyenlet kifejtett alakja. Az egyenlet felírható mátrixos alakban is, mivel a rendszer paramétereiben lineáris.

$$y(t) = \begin{bmatrix} 1 & x(1) & \dots & x^{n_f}(1) \\ 1 & x(2) & \dots & x^{n_f}(2) \\ 1 & x(3) & \dots & x^{n_f}(3) \\ \vdots & \vdots & \dots & \vdots \\ 1 & x(N) & \dots & x^{n_f}(N) \end{bmatrix} \begin{bmatrix} f_0 \\ f_1 \\ \vdots \\ f_{n_f} \end{bmatrix}. \tag{4.3}$$

A két leírás egymásnak tökéletesen megfelel. Nevezzük el az egyes elemeket az egyszerűség kedvéért a következőképpen:

$$y = \mathbf{\Phi}\ \theta, \tag{4.4}$$

ahol az egyes elemek elnevezései a következő összefoglalóban láthatóak:

- $\mathbf{\Phi}$ - regresszormátrix, amiben a bemeneti jel szerepel olyan alakba, ahogy azt a modellünk felhasználná. Annyi oszlopa van, ahány paraméteres a modellünk, és annyi sora van, ahány mintája van a gerjesztőjelünknek.
- θ - paramtervektor, ami az identifikálandó paramétereket tartalmazza.

Fontos, hogy a paramétervektorban és a regresszormátrixban az elemek sorrendje nem kötött.[1] Csupán annyi a megkötés, hogy páronként összetartozó értékek legyenek azonos indexen a paramétervektorban és a regresszormátrix oszlopai között[2].

[1] A sorrend nem kötött, hiszen az összeadás művelete kommutatív tulajdonságú.
[2] Az összetartozás látható a modellünk képletében. A sorrend ezért mindegy.

Megfigyelhető, hogy ez az elrendezés egy lineáris egyenletrendszer, ami több egyenletből áll, mint ismeretlen paraméterből. A túlhatározott egyenletrendszer hagyományos módszerekkel nem oldható meg. Az egyenletrendszernek a megoldása egy olyan közelítés lesz, ami minden gerjesztőjel mintára minimálisan tér el a kívánt kimenettől. A pszeudóinverz[3] felhasználásával megkapható a paramétervektor értéke[4]:

$$\theta = \mathbf{\Phi}^{+} y(t). \tag{4.5}$$

A lineáris regresszió nem csak statikus rendszerekre használható. Minden további nélkül képes dinamikus rendszereket is identifikálni. A fenti számoláshoz képest teljesen analóg módon kell eljárni:

1. Alkalmas modell kiválasztása
2. $\mathbf{\Phi}$ regresszormátrix összeállítása
3. $\mathbf{\Phi}^{+}$ pszeudóinverz számolása
4. θ - paramétervektor számolása a regresszormátrix és az ismeretlen rendszer kimenete alapján

Igazolható, hogy a túlhatározott egyenletrendszer pszeudóinverzzel való megoldása az egyenletek hibáinak négyzetösszegé minimumát adja [28].

4.3. Blokkos modellek lineáris regressziója

A lineáris regresszió jól használható identifikációs módszerként egyszerű rendszereknél, ahol az identifikálandó modellnek csak bemenetei és kimenetei vannak. Azonban blokkos modelleknél, amint két blokkot összekapcsolunk, és megjelennek megfigyelhetetlen belső változók, úgy már nem lehet alkalmazni. A belső változókat számításaink során fel kell használni, tehát értékükre szükség van, azonban megfigyelhetetlenek számunkra, ezért értéküket becsülni kell.

A becslést az úgynevezett *key term separation* [70, 74, 71] elv szerint tehetjük meg. A pontos megértéshez nézzünk egy példát egy Hammerstein-rendszer identifikációjának előkészítéséről [15].

Tegyük fel, hogy az identifikálni kívánt rendszerünk modellezhető Hammerstein elrendezésű blokkos rendszerrel [74, 71, 70]. Az elrendezés blokkvázlata a 2.1. ábrán látható. A példa során tartjuk magunkat az ábra elnevezéseihez, azaz a rendszer bemenete $u(t)$, kimenete $y(t)$, és a belső megfigyelhetetlen jel $x(t)$.

[3] A pszeudoinverz számítása, amennyiben az A mátrix teljes rangú: $A^{+} = (A^{T}A)^{-1}A^{T}$. Ez az úgynevezett Moore-Penrose Pszeudoinverz. Számítási szempontból sokkal hatékonyabb kiszámítási mód az szinguláris érték szerinti felbontás felhasználásával: $A^{+} = V(S^{T}S)^{-1}S^{T}U^{T}$

[4] A pszeudóinverz jelölése hagyományosan a $\mathbf{\Phi}^{+}$

A nemlineáris blokk legyen egy polinomiális statikus nemlinearitás, aminek fokszáma tetszőlegesen állítható a modellezendő rendszer és az identifikáció pontosságának függvényében. A statikus nemlinearitás az alábbi módon írható le matematekailag:

$$x(t) = \sum_{k=1}^{n_f} f_k u^k(t), \tag{4.6}$$

ahol $x(t)$ a statikus nemlinearitás kimeneti jele, ami egy belső megfigyelhetetlen jel. Ez a jel lesz a lineáris tag bemeneti jele, ami ebben az esetben egy FIR rendszer. A lineáris blokk leírható a következő egyenlettel:

$$y(t) = \sum_{i=0}^{n_b} b_i x(t-i). \tag{4.7}$$

Ha behelyettesítjük a nemlineáris blokk egyenletét a lineáris blokk egyenletébe, akkor megkapjuk a blokkos rendszer kimenetét a bemenete függvényében.

$$y(t) = \sum_{i=0}^{n_b} b_i \sum_{k=1}^{n_f} f_k u^k(t-i). \tag{4.8}$$

Látható, hogy a behelyettesítés után az identifikálandó paraméterek száma $(n_b+1)n_f$. Összetettebb modell esetében jelentősen meg tud nőni. Paramétereiben és adataiban ez a modell nemlineáris, hiszen a kapott paraméterpárokat szét kell tudni választani a tényleges blokk-paraméterekre. Ez nem triviális feladat.

A megoldást a problémára a *key term separation* elve adja [70, 68], ami szerint ha szét tudjuk választani a paramétercsoportokat úgy, hogy egy teljes blokk összes paramétere magában szerepelhessen, akkor a továbbiakban ezt fel lehet használni az adott blokk kimenetének a becslésére.

Nézzük meg ezt a gyakorlatban. Bontsuk fel a lineáris rész szummáját.

$$y(t) = b_0 \sum_{k=1}^{n_f} f_k u^k(n) + b_1 \sum_{k=1}^{n_f} f_k u^k(n-1) + \cdots + b_{b_n} \sum_{k=1}^{n_f} f_k u^k(n-b_n). \tag{4.9}$$

Látható, hogy ha az első tagban a b_0 paraméter 1 lenne, akkor tisztán a nemlineáris részt kapnánk meg a kimenet képletének első tagjában. És valóban, tegyük fel, hogy $b_0 = 1$. Ezt a megkötést mindig megtehetjük, hiszen egy adott értékű szorzótényező mindig felbontható egy szabad és egy kötött értékű szorzótényezőre [70, 68, 69, 66, 67].

Egy differenciaegyenlet átalakítható lineáris algebrai algoritmussá. Ez itt sincs másképp. Paramétereink adottak, és elő tudjuk állítani a regresszort is a nemlineáris blokk tulajdonságai alapján, tehát becsülni tudjuk a blokk kimenetét, ami történetesen a következő,

lineáris blokk kimenete, és nem utolsósorban a megfigyelhetetlen belső jelünk.

De hogyan becsüljük a megfigyelhetetlen belső jelet, ha az aktuális paraméterek számításához már szükségünk van minden elemére a regresszornak? Tipikus tyúk és a tojás dilemma.

A megoldás egyszerű. Nem az aktuális még számolatlan paramétereket használjuk fel, hanem az előző lépés paramétereit! Ha az eljárásunk konvergál, akkor ezzel a módszerrel a belső megfigyelhetetlen jel jó közelítéssel becsülhető.[5]

A *key term separation* tehát lehetővé tette az eredetileg megfigyelhetetlen belső jelek rekurzív becslését. A következő lépés, hogy ezt a belső jelet felhasználjuk a további számolásokhoz. A továbbiakban minden olyan helyen, ahol a nemlineáris blokk kimenetét kellene használni, ezt a becsült jelet használjuk fel. Ezt képletek szintjén a következő módon számoljuk:

$$y(t) = \sum_{k=1}^{n_f} f_k u^k(n) + b_1 x(n-1) + \cdots + b_{b_n} x(n-b_n). \tag{4.10}$$

Ezzel a képlettel már megírható az identifikációs algoritmus, hiszen létezik olyan tagja az egyenletnek, amiben csak az egyik blokk paraméterei szerepelnek.

A belső változó becslései miatt nem tudjuk közvetetten számolni a végeredményt, hiszen minden lépésben először becsüljük a rejtett változókat a kiemelt tagok előző paramétereivel, majd azután számoljuk a rendszer kimenetét a becsült belső változókkal, sorban haladva a blokkokat egymás után véve.

A lineáris regresszióhoz hasonlóan, ugyanúgy felírhatjuk a paramétervektort, és a regresszort:

$$\theta = \begin{bmatrix} \theta_{nonlinear} \\ \theta_{linear} \end{bmatrix} = \begin{bmatrix} [f_1, f_2, \ldots, f_{n_f}]^T \\ [b_1, b_2, \ldots, b_{n_b}]^T \end{bmatrix} \tag{4.11}$$

$$\Phi = \begin{bmatrix} \Phi_{nonlinear} \\ \Phi_{linear} \end{bmatrix} = \begin{bmatrix} [u(n), u^2(n), \ldots, u^{n_f}(n)]^T \\ [\hat{x}(n-1), \hat{x}(n-2), \ldots, \hat{x}(n-n_b)]^T \end{bmatrix} \tag{4.12}$$

ahol az $\hat{x}(n)$ értékeit az előző paraméter és regresszorvektor skaláris szorzatából kapjuk:

$$\hat{x}(n) = \Phi^T_{nonlinear} \cdot \theta_{nonlinear} = \begin{bmatrix} f_1, f_2, \ldots, f_{n_f} \end{bmatrix} \begin{bmatrix} u(n) \\ u^2(n) \\ \cdots \\ u^{n_f}(n) \end{bmatrix} \tag{4.13}$$

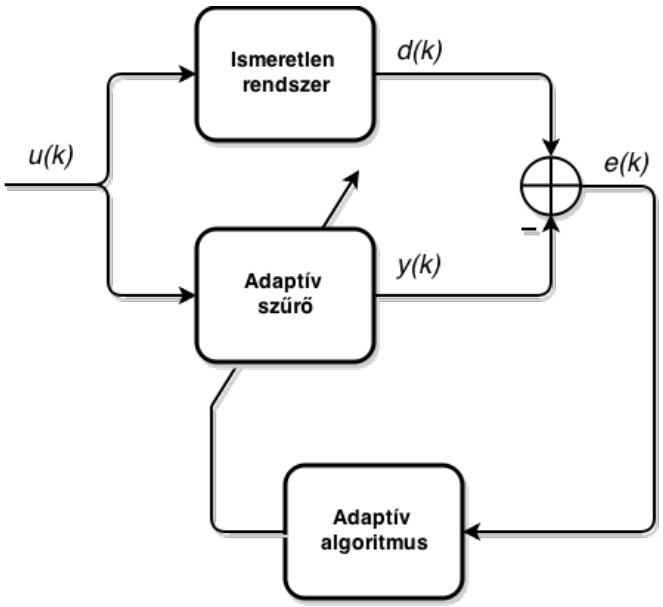
A tényleges regresszorvektorba már a késleltetett $\hat{x}(n)$ értékek kerülnek.

[5] A konvergencia **RLS** algoritmusnál nem bizonyított. A módszer inicializálásán múlik, hogy végül stabil eredményt kapunk-e, vagy a paraméterek oszcillációba kezdenek az identifikáció során.

Nincs más hátra, mint egy megfelelő adaptív iterációs módszerrel becsülni kell a paramétereket az összeállított regresszor alapján. Ha nem lenne probléma a belső változók megfigyelése, akkor a meglévő paramétervektor és a regresszor segítségével, pszeudóinverzzel számolható lenne a modell paramétervektora egyetlen lépésben [18, 47, 69].

4.3.1. Adaptív szűrési módszerek iteratív identifikációhoz

Adaptív szűrés alatt egy olyan szűrési eljárást értünk, ami alkalmazkodni képes a környezete változásaira [72, 23]. Adaptív eljárást alkalmazunk akkor, ha a követelmények egy rendszerrel szemben nem ismertek, vagy ismertek, de időben változó mivoltukból nem teljesíthetőek időinvariáns modellekkel.



4.2. ábra. *Adaptív identifikáció blokkvázlata*[6]

Az adaptív eljárások hajtóereje az ismeretlen rendszer és a modellezett rendszer kimenete közti hiba minimalizálása. A hibát egy adaptív algoritmus dolgozza fel, és állítja a modell paramétereit olyan módon, hogy minimalizálja a hibát.

Az 4.2. ábrán látható jelöléseknek magyarázata a következő:

- $u(k)$ - rendszer bemeneti jele, amit a modell is megkap
- $d(k)$ - ismeretlen rendszer kimenete, a kívánt (*desired*) jel
- $y(k)$ - modell kimenete
- $e(k)$ - ismeretlen rendszer kimenetet és a modell kimenete közötti eltérés, hibajel

[6]Ez természetesen csak egy az adaptív szűrők felhasználási lehetőségei közül. A dolgozat nem tér ki a további két felhasználási módra, a csatornakiegyenlítést és a jelerősítést megvalósító struktúrára [23].

A hiba minimalizására vezessük be a költségfüggvényt, ami a hibától függ:

$$F = F[e(k)]. \tag{4.14}$$

A költségfüggvény ismeretében az adaptív algoritmus úgy próbálja változtatni a modell paramétereit ($\Delta\theta$), hogy a költségfüggvényt minimalizálja. Egy ilyen változtatás mindig az előző paramétervektorra épül.

$$\theta(n+1) = \theta(n) + \Delta\theta. \tag{4.15}$$

A költségfüggvény megválasztása nagy hatással van az identifikáció tulajdonságaira. Mi a dolgozatban a Leas Squares költségfüggvény-minimalizálást fogjuk választani. Általánosan elmondható, hogy az LS módszer kényelmesen használható stacionárius környezetben, azaz identifikációhoz ideális választás.[7]

A Least Squares költségfüggvény alakja a következő:

$$F = F[e(k)] = \sum_{i=0}^{k} |e(k-i)|^2. \tag{4.16}$$

A minimalizálásra számos algoritmus született [13]. A következő részekben ismerjünk meg ebből két eltérő elven működő típust. A dolgozatnak nem célja, hogy a minimalizáló algoritmusokat elemezze, csupán betekintést kíván adni a közölt algoritmusok tulajdonságaiba.

4.3.2. LMS

Az **LMS** (Least Mean Square) algoritmus az egyik legismertebb és legkedveltebb minimalizálási algoritmus [35, 36]. Sikere a rendkívül kis számításigényében keresendő, azonban az egyik leggyengébben teljesítő algoritmus is az adaptív minimalizáló algoritmusok között.

Az **LMS** algoritmus az adaptálni kívánt rendszert egy adaptív szűrővel közelíti, ami általában egy FIR szűrőben nyilvánul meg. Az algoritmus egyetlen paramétere a bátorsági tényező (változtatható még az N is, azaz hogy hány tapes FIR szűrővel közelítse a rendszert, de ez nem tekinthető a szó szoros értelmében paraméternek).

A módszer hátránya, hogy csak a pillanatnyi hibát veszi figyelembe, és ennek alapján lép a bátorsági tényező nagyságával megegyezőt, az éppen legjobbnak vélt irányba, amit a szűrő paramétereinek gradiensvektora sztochasztikus tulajdonságai alapján határoz meg. Ez lehet, hogy nem a legjobb irány (csak pillanatnyilag tűnik annak), de az idő múlásával ezek a hibák kiátlagolódnak, és végső soron jó irányba tart majd az algoritmus. Mivel a bátorsági tényező szabja meg a lépés nagyságát, ezért az algoritmus nem fogja pontosan megtalálni a hibaminimumot, hanem egy kicsiny környezetében fog folyamatosan oszcillál-

[7] Hasonlóan az adaptív szűrőstruktúrákhoz, a minimalizálás kritériumáról is csak felületesen értekezünk. Az Olvasó érdeklődésére bízzuk, hogy utánanézzen a Mean Square Error (MSE), Weighted Least Sqares (WLS) avagy a felejtő LS, vagy az Instantenious Squared Value (ISV) módszereknek.

ni. Ez a gyakorlatban általában nem probléma.

A működést biztosító rekurzív egyenletek a következők:

$$e(k) = d(k) - \tilde{\theta}^T(k)\mathbf{x}(k), \tag{4.17}$$

$$\theta(k+1) = \theta(k) + 2\mu e(k)\mathbf{x}(k), \tag{4.18}$$

ahol μ a bátorsági tényező.

A paramétervektort 0-ból szokás indítani. Az algoritmus lépésenként közelíti a valós értékek felé őket.

A dolgozatban nem ezt az algoritmust fogjuk használni, hanem a következő részben bemutatott **RLS** algoritmust, ami egy jóval robosztusabb eljárás ismeretlen rendszerek identifikációjára.

4.3.3. RLS

Hasonlóan az **LMS** algoritmushoz, az **RLS** (Recursive Least Squares) algoritmus is egy online algoritmus, ami mintáról mintára próbálja közelíteni az adaptív modellt az ismeretlen rendszerhez [34, 35]. Azonban ezt az identifikálandó rendszer és a modell kimenetének négyzetes minimalizálásával éri el. Míg az **LMS** módszer sztochasztikus módszerekre épít, addig az **RLS** algoritmus determinisztikus, lineáris algebrára alapozó számítási elvet követ.

Ez az eljárás is az ismertebb algoritmusok közé tartozik. Előnye, hogy sokkal robosztusabb és gyorsabban konvergáló algoritmus, mint az **LMS**, még folyamatosan változó zajos környezetben is, azonban ennek ára is van: a módszer számításigénye lényegesen nagyobb.

Az adaptív FIR szűrőre épülő, N együtthatót tartalmazó szűrő bemeneti vektora egy adott k pillanatban az alábbi alakban írható fel:

$$\mathbf{x}(k) = [x(k) \quad x(k-1) \quad ... \quad x(k-N)]^T, \tag{4.19}$$

ahol $\mathbf{x}(k)$ egy oszlopvektor. A szűrő együtthatói $\theta_j(k)$, ahol $j = 0, 1, ..., N$. Az algoritmus a futása alatt ezeket az együtthatókat állítja úgy, hogy az identifikált rendszer kimenete $y(k)$ és a kívánt kimenet $d(k)$ egymáshoz képest a legkisebb négyzetek elve alapján a legközelebb legyen. A kimenetek közti hiba felírható a következő alakban:

$$e(k) = d(k) - y(k). \tag{4.20}$$

Az **RLS** algoritmus esetében a minimalizálandó költségfüggvény az alábbi alakot veszi fel:

$$F(k) = \sum_{i=0}^{k} \lambda^{k-i} e^2(i) = \sum_{i=0}^{k} \lambda^{k-i} [d(i) - \mathbf{x}^T(i)\mathbf{\Phi}(k)]^2, \tag{4.21}$$

ahol $\mathbf{\Phi}(k) = [\theta_0(k) \quad \theta_1(k) \quad ... \quad \theta_N(k)]^T$ a teljes paramétervektor egy adott k lépésben. Célunk, hogy a költségfüggvényt minimalizáljuk úgy, hogy a paramétervektor értékeit állíthatjuk. Ez a matematika nyelvén a költségfüggvény paramétervektor szerinti parciális deriválásával érhetjük el:

$$\frac{\partial F(k)}{\partial \mathbf{\Phi}(k)} = -2 \sum_{i=0}^{k} \lambda^{k-i} \mathbf{x}(i) [d(i) - \mathbf{x}^T(i)\mathbf{\Phi}(k)]^2. \tag{4.22}$$

Az egyenletet nullvektorral egyenlővé téve és a paramétervektorra rendezve az alábbi alakot kapjuk:

$$\mathbf{\Phi}(k) = \Big[\sum_{i=0}^{k} \lambda^{k-i} \mathbf{x}(i)\mathbf{x}^T(i) \Big]^{-1} \sum_{i=0}^{k} \lambda^{k-i} \mathbf{x}(i) d(i) \tag{4.23}$$

Az fenti egyenletben elnevezve az egyes elemeket, a következő egyszerűsítéssel élhetünk:

$$\mathbf{\Phi}(k) = {\mathbf{R_d}}^{-1}(k)\mathbf{p_d}(k), \tag{4.24}$$

ahol ${\mathbf{R_d}}^{-1}(k)$ és $\mathbf{p_d}(k)$ rendre a bemeneti jel korreláció mátrixa és a bemenet valamint a kívánt kimenet keresztkorreláció mátrixa. Ha élünk az $\mathbf{S_d}(k) = {\mathbf{R_d}}^{-1}(k)$ behelyettesítéssel[8], valamint előre kiszámoljuk és helyettesítjük a $\mathbf{\Psi}(k) = \mathbf{S_d}(k-1)\mathbf{x}(k)$ vektort[9], akkor az alábbi **RLS** algoritmus megvalósítást kapjuk eredményül:

$$\mathbf{\Psi}(k) = \mathbf{S_d}(k-1)\mathbf{x}(k) \tag{4.25}$$

$$\mathbf{S_d(k)} = \frac{1}{\lambda}\Big[\mathbf{S_d}(k-1) - \frac{\mathbf{\Psi}(k)\mathbf{\Psi}^T(k)}{\lambda + \mathbf{\Psi}^T(k)\mathbf{x}(k)}\Big] \tag{4.26}$$

$$\mathbf{p_d}(k) = \lambda \mathbf{p_d}(k-1) + d(k)\mathbf{x}(k) \tag{4.27}$$

$$\mathbf{\Phi}(k) = \mathbf{S_d}(k)\mathbf{p_d}(k). \tag{4.28}$$

Az így megvalósított **RLS** algoritmus rekurzívan képes a költségfüggvény minimalizálására, viszont sokkal számításigényesebb, mint az előzőleg ismertetett **LMS** algoritmus, azonban ennek fejében robosztusabb, és sokkal gyorsabban konvergál. A dolgozat további részében ezt az algoritmust használom a blokkos identifikációs rendszerhez.

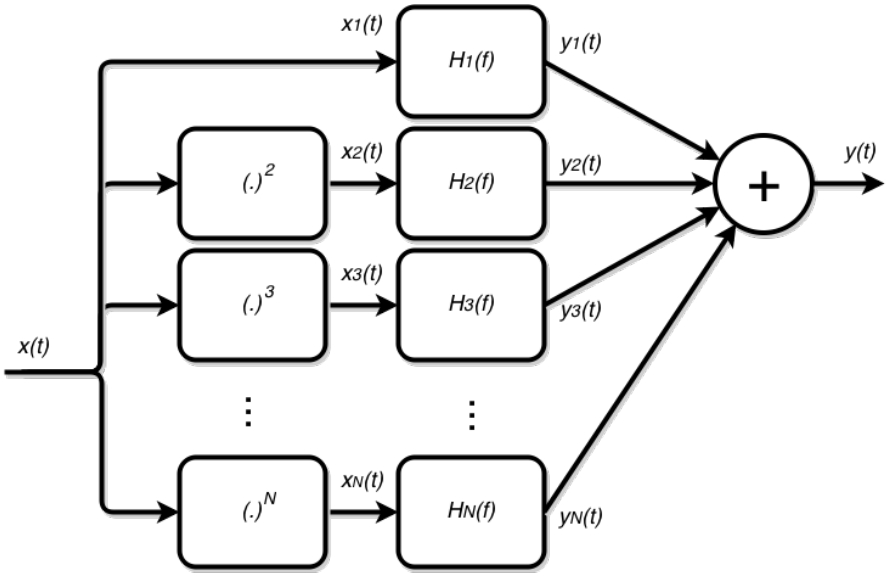
[8] Az invertálás elvégzésére az invrtálási lemmát használjuk [28].

[9] Ezzel a helyettesítéssel jelentősen csökkenthető az algoritmus számolásigénye.

4.4. Swept sine alapú identifikáció

Az előző szakaszban az identifikációs eljárásunk tisztán lineáris algebrára épült. A következőkben megismerünk egy viszonylag újnak számító identifikációs eljárást, ami a lineáris algebra helyett a jelfeldolgozási ismereteinkre hagyatkozik.

A *swept sine* eljárást a közelmúltban kezdték el használni akusztikai mérésekre [48]. Előnye, hogy míg a korábban alkalmazott eljárásokkal a mért rendszerek nemlineáris viselkedését nem lehetett jól becsülni, addig ezzel az újnak számító módszerrel a nemlineáris hatások is jól mérhetőek lettek. A módszer alapja az exponenciálisan növekvő frekvenciájú szinuszos gerjesztőjel és a nemlineáris konvolúció. Tipikusan párhuzamos Hammerstein modelleket identifikálnak az eljárással, hiszen a kapott eredmény egy az egyben átültethető a modellbe minimális számításigénnyel.



4.3. ábra. *Polinomiális Hammerstein-modell*

Ahogy a fejezet elején tárgyaltuk, az identifikáció egyik első lépése a kísérlet megtervezése, amibe beletartozik a gerjesztőjel megtervezése is [11, 49]. Ha a gerjesztőjellel úgy tudjuk gerjeszteni a rendszert, hogy az a lehető legtöbb információt tartalmazza a válaszában, akkor az identifikációs módszerünk jó úton halad.

A kezdeti akusztikai méréseknél impulzus gerjesztést[10] használtak [27], majd áttértek az

[10]A módszer a jelfeldolgozás szakirodalmából vezethető le. Egy rendszer impulzusválasza teljesen leír egy lineáris rendszert. Az impulzusválasz méréséhez a legegyszerűbb impulzus gerjesztéssel gerjeszteni a rendszert, ez azonban nem mindig kivitelezhető. Elterjedt módszer volt az impulzív hangot kiadni képes eszközök használata, mint például a felfújt léggömb kidurrantása vagy vaktöltény használata. A módszer természetesen nem tartalmazta a nemlinearitásra vonatkozó információkat, és nem vette figyelembe a háttérzaj jelenlétét sem.

MLS[11] és TDS[12] módszerekre. Jelenleg a legmodernebbnek számító exponenciális swept sine alapú mérés (továbbiakban ESS) [50, 49, 11, 26, 22, 9].

Mikor a PC-technológia nem volt elég fejlett ahhoz, hogy valós időben képes legyen komolyabb számítások elvégzésére, dedikált DSP eszközöket használtak a mérések elvégzésére. Ez esetenként drága és nehezen hozzáférhető volt. A modern PC-k használatával már nem jelent problémát egy hosszabb mérés kivitelezése sem. Elég hozzá egy általános számítógép és egy hanginterfész, valamint a mérést lebonyolító szoftveres csomag.

Az ESS módszerhez használt gerjesztés a következő tulajdonságokkal rendelkezik:

- **Sávkorlátozott** - a jelnek van egy kézdő és végfrekvenciája, amit a felhasználó képes állítani annak függvényében, hogy a mérendő rendszeren milyen frekvenciatartományt szeretne gerjeszteni.
- **Exponenciális frekvencia növekedés** - Az alacsonyabb frekvenciákon több időt tölt a gerjesztőjel, aminek köszönhetően a lehető legtöbb energiát adja át az alacsonyabb frekvenciákon, így javítva az előző módszerekhez képest a jel-zaj viszonyon kisfrekvenciákon. Az exponenciális frekvencianövekedésnek köszönhetően a gerjesztőjel jobban illeszkedik a logaritmikus frekvencia felbontáshoz.
- **Nullátmenet szinkronizálás** - A gerjesztőjel nullátmenetei szinkronizálva vannak a kiindulási frekvencia egész számú többszöröseivel, azaz mikor a pillanatnyi frekvencia pont eléri a a kiindulási frekvencia egész számú többszörösét, a gerjesztőjel tervezése biztosítja, hogy ekkor a jelnek nullátmenete legyen. Ez előfeltétele a módszer alapját képező felharmonikus szétválasztó tulajdonságnak.

A gerjesztőjel előállítása az alábbi módon lehetséges [50, 27]:

$$x(t) = \sin\left(2\pi f_1 L\left(e^{\frac{t}{L}} - 1\right)\right), \tag{4.29}$$

$$L = \frac{1}{f_1} Round\left(\frac{\hat{T} f_1}{log\left(\frac{f_2}{f_1}\right)}\right), \tag{4.30}$$

ahol f_1 és f_2 rendre a sweep kezdő és befejező frekvenciája, $\hat{T}$ a közelített jelhossz, a *Round* pedig a közelebbi egész számra való kerekítés operátora.

A módszer az úgynevezett nemlineáris konvolúciót használja a válaszjelből való információ kinyerésére [11]. Ez abban nyilvánul meg, hogy a válaszjelet konvolváljuk a gerjesztőjel

[11] Maximum Length Sequence, egy pszeudo random fehérzaj gerjesztés, ami a mintasort tekintve zajszerű, de reprodukálható

[12] Time Delay Spectrometry. Lényegében egy lineáris szinuszos sweep-jellel való mérést jelenti. Time Delayed, hiszen az egyes frekvenciákat időben elcsúsztatva gerjesztik. Szokás "stretched pulse" vagy "chirp" jelnek is nevezni az ilyen gerjesztést.

időtartománybeli tükörképének amplitúdomodulált változatával. Ezt a jelet inverz szűrőnek nevezzük. A módszer sikerességéhez követelmény, hogy a gerjesztőjel és az inverz szűrő konvolúciója egy tiszta *dirac deltát* adjon. A konvolúció után a kapott jel tartalmazni fogja magát a lineáris impulzusválaszt, valamint az egyes nemlineáris felharmonikusokra vonatkozó átvitelt is jól elkülöníthető helyeken (4.5. ábra).

Az elkülönítésnek előfeltétele a szinkronizált nullátmenetek az alapharmonikus egész számú többszöröseinél. Matematikailag ez a következő módon fogalmazható meg:

$$s_m(t) = s(t + \Delta t_m), \tag{4.31}$$

ahol $s(t)$ az exponenciális jel, $s_m(t)$ az exponenciális jel olyan szakasza, aminek a kezdeti frekvenciája az eredeti jel kiindulási frekvenciájának pont m-szeres. Δt_m az az időbeli távolság, ami a gerjesztőjelnek kell, hogy a kiindulási frekvenciától elérje annak m-szeresét. Belátható, hogy ez a feltétel biztosítja a nullátmenet szinkronizálást.

Ha a gerjesztőjel teljesíti a 4.31 egyenletet, akkor az alábbi példán keresztül könnyen belátható, hogy a nemlineáris konvolúció eredményeképp felharmonikusonként elkülönített impulzusválaszokat kapunk: tegyük fel a gerjesztőjelet egy tisztán statikus nemlinearitást tartalmazó rendszerbe vezetjük, ahol a statikus nemlinearitás az alábbi egyenlettel írható le, ahol $x(t)$ és $y(t)$ rendre a rendszer bemenete és kimenete:

$$y(t) = x^3(t). \tag{4.32}$$

Mivel szinuszos jellel gerjesztjük a rendszert, ezért érvényes lesz a következő trigonometrikus azonosság:

$$\sin^3(\alpha) = \frac{3}{4}\sin(\alpha) - \frac{1}{4}\sin(3\alpha). \tag{4.33}$$

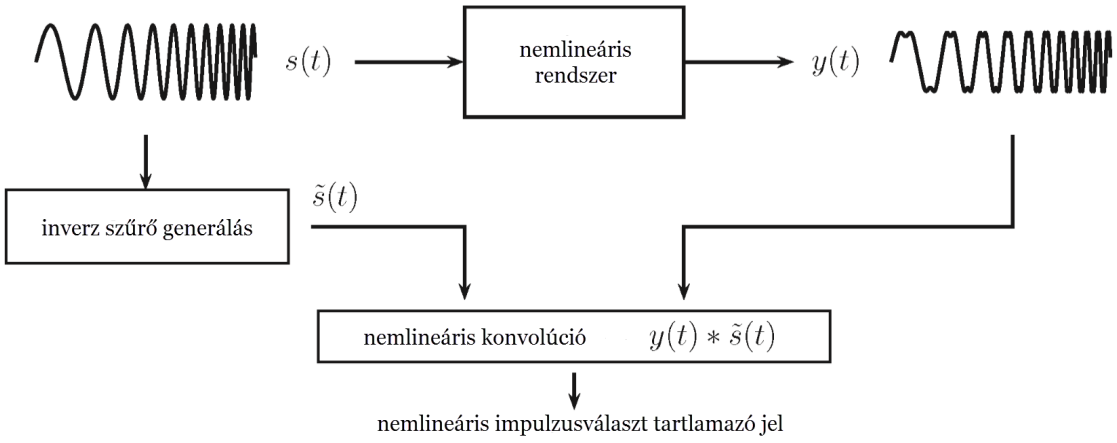
Alkalmazva erre a 4.31 egyenletet és a rendszer kimenete is egyben, a következő alakot ölti:

$$s^3(t) = \frac{3}{4}s(t) - \frac{1}{4}s(t + \Delta t_3). \tag{4.34}$$

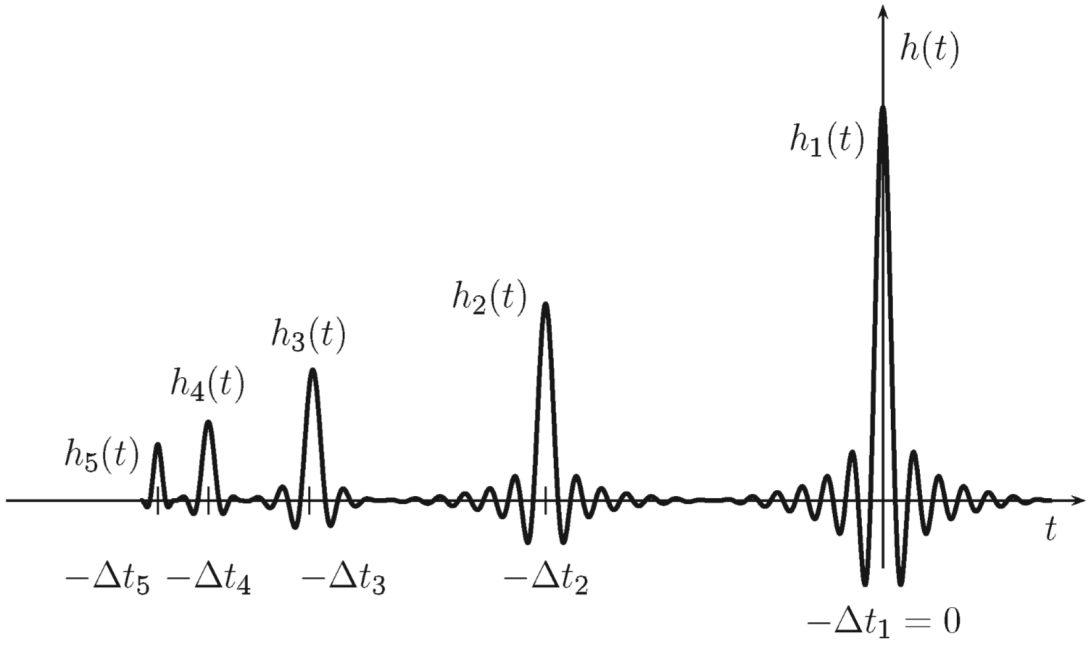
Ha ezt a jelet konvolváljuk az inverz szűrővel[13], akkor eredményként két *dirac deltát* kapunk, ahol az egyik Δt_3 idővel el van csúsztatva, ami pont az az idő, amíg a gerjesztőjel eléri a kiindulási frekvencia háromszorosát. Ezzel belátható, hogy ha gerjesztőjellel vizsgált rendszer tartalmazna dinamikus hatásokat is, akkor azoknak az impulzusválasza a generálódott dirac delták helyén lenne. Ilyen módon belátható a harmonikusokra vonatkozó

[13]A gerjesztőjel és az inverz szűrő konvolúciója *dirac deltát* kell adjon. Ezt az inverz szűrő tervezése biztosítja.

elkülönítő tulajdonság.



4.4. ábra. *Nemlineáris konvolúció [49]*



4.5. ábra. *Nemlineáris konvolúció eredménye [49]*

A felharmonikusokhoz tartozó impulzusválaszokat a legegyszerűbben FIR szűrökkel valósíthatjuk meg, hiszen ekkor maga az impulzusválasz lesz a szűrő együtthatóit tartalmazó vektor. Léteznek hatékonyabb megvalósítások is, de jelen dolgozatnak nem feladata magának a megvalósításnak az optimalizálása.

A következő megoldandó probléma az, hogy a kapott szűrők hatása csak az adott fokszámú felharmonikusra szabad érvényesüljön. Ezt kétféle módunk van elérni: vagy a modell végén hajtunk végre transzformációt a kivágott impulzusválaszokon [47, 50], vagy rögtön

a modell elején, a bemeneti jelet alakítjuk a követelményeknek megfelelően [12, 48].

Az impulzusválaszok utólagos transzfomálása nem elegáns. Bonyolult transzformációval próbáljuk kompenzálni a bemeneti jelek előállításánál elkövetett hibánkat: a jel hatványra emelése nem állít elő tiszta felharmonikust, így nem is használható fel arra, hogy a felharmonikushoz tartozó szűrővel szűrjük.

A másik módszer sokkal hatékonyabb ebből a szempontból. Az előző megoldásban elkövetett hiba kiküszöbölésére a bemeneti jel olyan transzformációját hajtjuk végre, aminek eredménye tisztán a bemeneti jel felharmonikusainak sorozata. Ezt a jelsorozatot a megfelelő impulzusválaszokból alkotott szűrőkkel szűrve megkapjuk a kívánt eredményt. A transzformáció a Chebishev polinomok tulajdonságait használja ki [41]: ha a bemeneti jel szinuszos jel, akkor az adott fokszamú Chebishev polinomot hattatva rá, az eredményezett jel spektrálisan tisztán az eredeti jel megfelelő fokszámú felharmonikusa lesz.

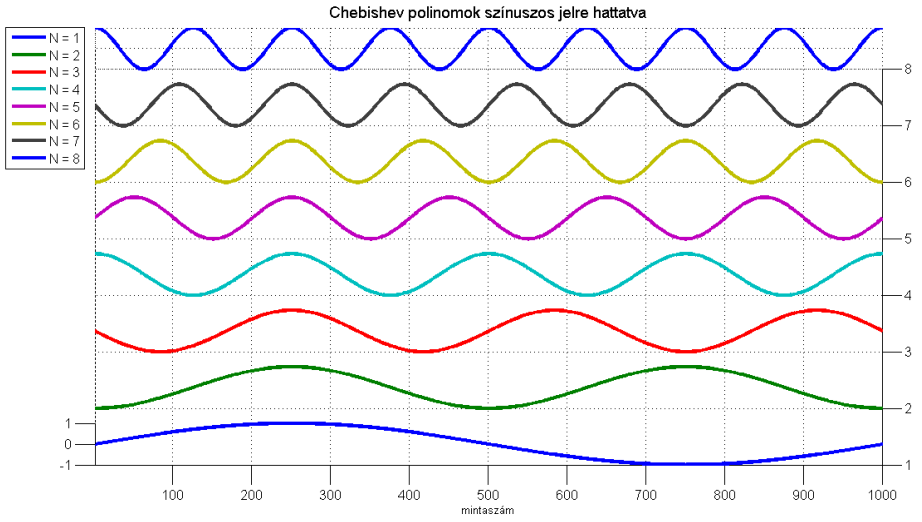
$$T_n(x) = \cos(n\theta), \qquad ahol \quad x = \cos(\theta). \tag{4.35}$$

A Chebishev polinomok további előnye, hogy előállításuk rekurzív módon történik az alábbi szabályosságok alapján.

$$T_n(x) = 2xT_{n-1}(x) - T_{n-2}(x), \qquad ahol \quad n = 2, 3..., \tag{4.36}$$

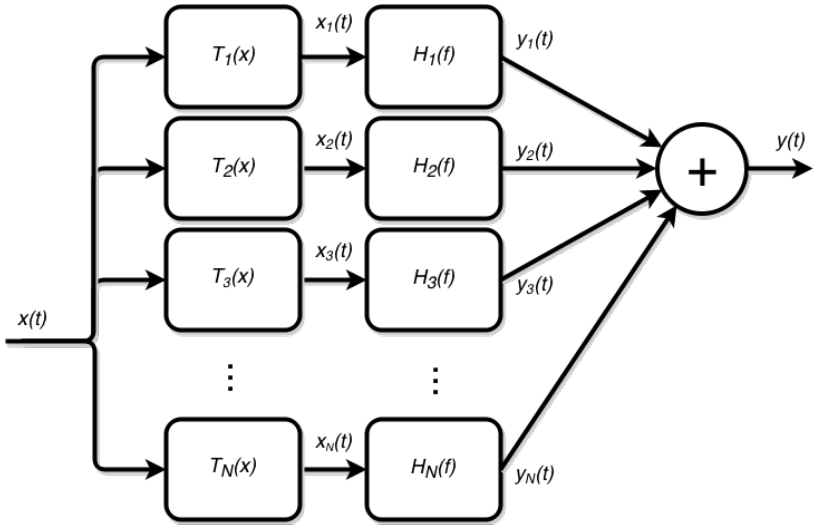
$$T_0(x) = 0, \qquad T_1(x) = x. \tag{4.37}$$

A transzformáció hatását remekül szemlélteti a 4.6. ábra, ahol egy adott frekvenciájú szinuszos jelet transzformálunk felharmonikusaik sorozatává 8. fokszámig. Jól látható, hogy a fokszám növekedésével arányosan növekszik a frekvencia is.



4.6. ábra. *Chebyshev polinomok hatása szinuszos jelre*

Minden szűrőág bemenete a bemeneti jel, amit a szűrőág számának megfelelő indexű Chebishev polinom transzformál a megfelelő felharmonikussá. A felharmonikus jelet ezután szűrjük az ágnak megfelelő szűrővel. Az egyes ágak kimeneteinek összege lesz a modell a kimenete. A modell felépítése a 4.7. ábrán látható.



4.7. ábra. *Chebyshev polinomiális párhuzamos Hammerstein-modell*

A Chebishev polinomokkal transzformált modell elegáns megoldás. Nem tartalmaz felesleges, később korrigálandó műveleteket.

Az identifikációs módszer implementációs szempontból tovább javítható, ha a modellben szereplő hagyományos lineáris szűrőket lecseréljük közös, logaritmikus skálán elhelyezett pólusú másodfokú szűrőkre [12]. Az eredeti modell átalakítása releváns, hiszen az emberi fül frekvenciafelbontása is közel logaritmikus.

5. fejezet

Lineáris regresszió implementálása

Az előző fejezetben megismerkedtünk az identifikáció 5 lépésével és a dolgozatban használatos identifikációs algoritmusokkal. A következő fejezetekben megnézzük, hogy az elmélet hogyan ültethető át a gyakorlatba.

Előzőleg egy folytonos statikus nemlinearitás identifikálási elméletét néztük át. Most alkalmazzuk az identifikáció öt lépését a lineáris regresszió példáján keresztül!

5.1. Kísérlet tervezése

Kétféle statikus nemlineáris rendszert fogunk vizsgálni a módszer bemutatása során. Mindkettő folytonos telítődéses nemlinearitás. Az egyik egy szigmoid függvénnyel leírható, sima, S alakú $f(x)$ nemlinearitás (6.1), a másik egy erős levágást tartalmazó, átmenet nélküli - ezzel deriváltjában ugrást tartalmazó - telítődéses $g(x)$ nemlinearitás (6.2).

$$f(x) = c_f \frac{1}{1 + e^{-x}}, \tag{5.1}$$

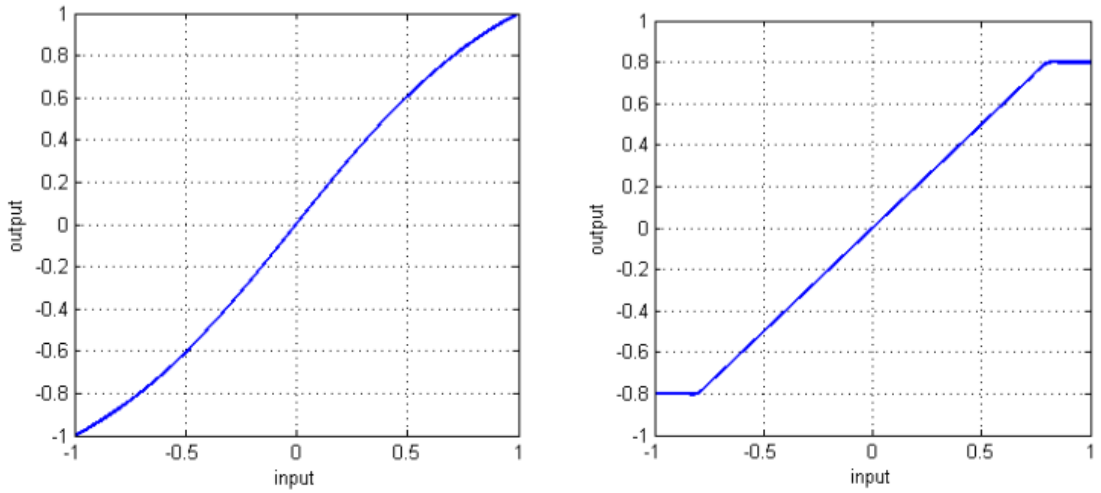
ahol c_f egy skálázó konstans.

$$g(x) = \begin{cases} c_g & \text{ha } x > c_g \\ -c_g & \text{ha } x < -c_g \\ x & \text{egyébként,} \end{cases} \tag{5.2}$$

ahol c_g határozza meg a levágás mértékét.[1]

A rendszerek felépítését ismerjük, paramétereiket azonban nem. Az identifikációhoz kísérletek keretében információt kell gyűjtenünk az identifikálandó rendszerekről. Mivel statikus nemlinearitásokat vizsgálunk, ezért az identifikáció célja, hogy az identifikált modellünk egy meghatározott tartományon belül hasonlóan viselkedjen az eredeti rendszerhez képest. Ennek következtében bemeneti jelnek olyan jelet kell válasszunk, ami értékkészletben lefedi az egész vizsgálni kívánt tartományt. Ennek a követelmények megfelel egy szinuszos jel

[1] E két paraméter pontos értéke az identifikáció szempontjából irreleváns, a modellünk ellenőrzésekor ki fog derülni, hogy megfelelő lett-e a paraméterek becslése, ezért értékük közlésével nem foglalkozom.



(a) *Szigmoid telítődés* **(b)** *Levágásos telítődés*

5.1. ábra. *Nemlineáris telítődéses rendszerek*

is, de akár háromszögjelet is alkalmazhatnánk. Az identifikációhoz egységnyi amplitudós kisfrekvenciás szinusz jelet fogunk használni.

Az identifikáció utáni ellenőrzéshez érdemes szinusz jelet választani, hiszen ezzel frekvenciatartományban vizuálisan is könnyen ellenőrizhetjük, hogy a rendszer és az identifikált modellünk hasonlóan adják-e vissza a felharmonikus komponenseket. Mivel a vizsgált rendszereink statikusak, nem rendelkeznek memóriával, ezért viszonylag rövid jelekkel is identifikálhatóak. Egyedüli feltétel, hogy a gerjesztőjel értékkészlete kellően kis felbontással lefedje az identifikálni kívánt intervallumot, különben felléphetnek nem várt jelenségek.[2]

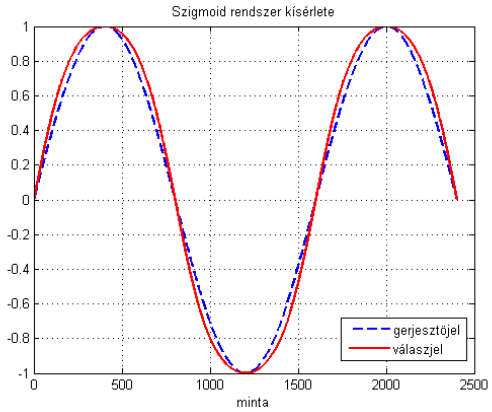
A méréseket két zajszinten végezzük el. Legyen SNR_a=40 dB és SNR_b=10 dB. Ezen két zajszinttel jól megfigyelhető, hogyan viselkedik az eljárás kevésbé zajos, és igen zajos környezetben.

5.2. Kísérlet elvégzése, adatgyűjtés

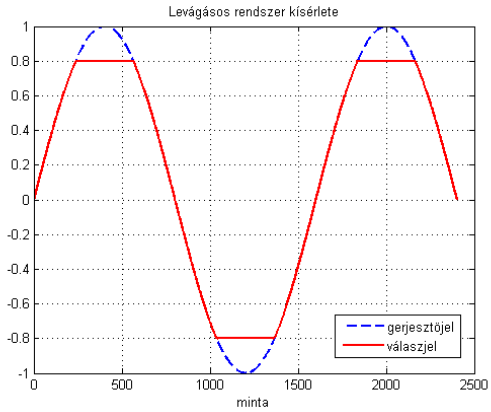
Gerjesztőjelek generálása után mindkét rendszert gerjesztjük, aminek eredményeképpen két identifikációs adatsor párunk lesz, amelyekkel elvégezhető az identifikáció.

Az gerjesztőjelek és a válaszok a 5.2. ábrán láthatóak.

[2]Ilyen nem várt jelenség lehet, amikor két meghatározott helyű pont között a függvény viselkedése nem definiált. Ilyenkor érhetik meglepetések az embert, mikor egy ilyen nem definiált tartományon belülre esik a bemeneti jel. A jelenség elkerülhető, ha kellően sűrűek az identifikáló gerjesztés mintái, és az alkalmazott modell fokszáma nem végtelen.



(a) *Szigmoid rendszer kísérleti eredménye*



(b) *Levágásos rendszer kísérleti eredménye*

5.2. ábra. *Nemlineáris telítődéses rendszerek*

5.3. Modellstruktúra megválasztása

Modellnek egy elterjedten használatos folytonos nemlinearitási modellt fogunk használni. A felépítéséből adódóan polinomiális modellnek nevezzük. Általános felépítése a következő:

$$y(t) = \sum_{k=1}^{n_f} f_k N_k[x(t)] \tag{5.3}$$

Az általános polinomiális modell bázisfüggvényekből, és az azok hatását állító paraméterekből áll. Bázisfüggvényként tetszőleges nemlineáris függvényeket lehet választani.

Általában bázisfüggvények valamilyen sorozatot szoktak választani, hogy a rendszer fokszáma könnyen állítható legyen. A leggyakrabban használt ilyen bázisfüggvények a hatványsorok, ahol a egyes elemek fokszáma megfelelő hatványra emelik a gerjesztőjelet.[3]

Szélsőséges esetben egy-egy teljes nemlineáris függvény is állhat egy bázisfüggvény helyén, és az identifikáció egy kiválasztó algoritmusként üzemel. Megmondja egy kimeneti mintasorról, hogy a bemenet ismeretében milyen rendszeren ment keresztül a jel.

Az identifikáció során kétféle bázisfüggvényt fogunk alkalmazni. Az egyik a hagyományos hatványfüggvény-modell, a másik pedig Chebyshev-polinomokra épülő modell lesz.

Hatványfüggvényes bázisfüggvény

$$N_k^f[x(t)] = x^k(t) \tag{5.4}$$

Chebyshev bázisfüggvény [41]

[3] A módszer nagy hátránya, hogy nagy hatványoknál a rendszer könnyen kezelhetetlenné válik nagy bemenőjelek esetén.

$$N_k^g(x) = 2xN_{k-1}^g(x) - N_{k-2}^g(x), \qquad n = 3, 4, \ldots \tag{5.5}$$

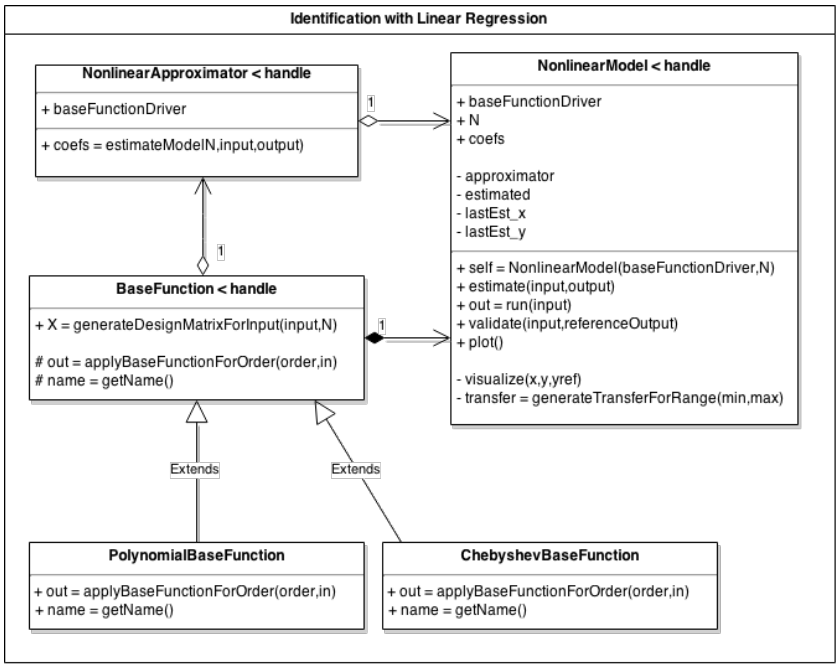
$$N_0^g(x) = 1, \qquad N_1^g(x) = x \tag{5.6}$$

A bázisfüggvények definiálásával a modellek is teljesen definiálva lettek. Az identifikációhoz szükség van még a modellek fokszámára. Ez legyen esetünkben 40-edfokú. Később megnézzük kisebb fokszámú modellre is az identifikáció eredményét.

5.4. Módszerválasztás, paraméterbecslés

A módszer, ahogy azt tisztáztuk, a lineáris regresszió lesz, amihez a szükséges adatok fele már rendelkezésre áll (gerjesztőjel és a rá adott válasz). Szükségünk van még a regresszormátrixra, ami számolásához a modelljeink struktúráját kell figyelembe venni.

Az identifikáció elvégzésére MATLAB-ban létrehoztam egy objektumorientált architektúrát, kellőképpen rugalmas ahhoz, hogy könnyen hozzá lehessen adni új funkcionalitást. Az architektúra vázlata a 5.4. ábrán látható.



5.3. ábra. *Lineáris regresszión alapuló mérési rendszer architektúrája*

Az architektúra lehetővé teszi, hogy a *BaseFunction* osztályból leszármaztatva tetszőleges bázisfüggvényt alkalmazó folytonos nemlinearitást leíró modellt készítsünk. Ehhez nem kell mást tenni, mint implementálni a *BaseFunction* osztály absztrakt metódusait. A metódusok nevei magukért beszélnek.

A rendszer az `applyBaseFunctionForOrder(self,order,in)` metóduson keresztül kéri el az implementált adott fokszámú bázisfüggvénytől a regressziós-mátrix egy sorát, amihez megadja a feldolgozandó bemeneti jelet és a fokszámot is.

5.1. kódrészlet. *PolynomialBaseFunction osztály implementációja*

```
classdef PolynomialBaseFunction < BaseFunction
    methods
        function out = applyBaseFunctionForOrder(self,order,in)
            out = power(in,order);
        end

        function name = getName(self)
            name = 'Polynomial Base Function';
        end
    end
end
```

5.2. kódrészlet. *ChebyshevBaseFunction osztály implementációja*

```
classdef ChebyshevBaseFunction < BaseFunction
    methods
        function out = applyBaseFunctionForOrder(self,order,in)
            in = in(:);
            cheb = zeros(length(in),max(order,2));
            cheb(:,1) = ones(size(in));
            cheb(:,2) = in;
            if order == 1
                out = cheb(:,1);
            elseif order == 2
                out = cheb(:,2);
            else
                for k = 3:order
                    cheb(:,k) = 2*in.*cheb(:,k-1) - cheb(:,k-2);
                end
                out = cheb(:,order);
            end
        end

        function name = getName(self)
            name = 'Polynomial Base Function';
        end
    end
end
```

A bázisfüggvények implementálása után nincs más dolgunk, mint létrehozni egy *NonlinearModel* objektumot, aminek átadjuk az implementált bázisfüggvényt és a kívánt fokszámát, majd a gerjesztőjel és a válaszjel segítségével elvégezhetjük az implementációt a következő metódushívásokkal:

5.3. kódrészlet. *Nemlineáris modell létrehozása és identifikációja*

```
% x - gerjesztojel
% y - valaszjel
% N - modell fokszama

model = NonlinearModel(ChebyshevBaseFunction(),N);
model.estimate(x,y)
```

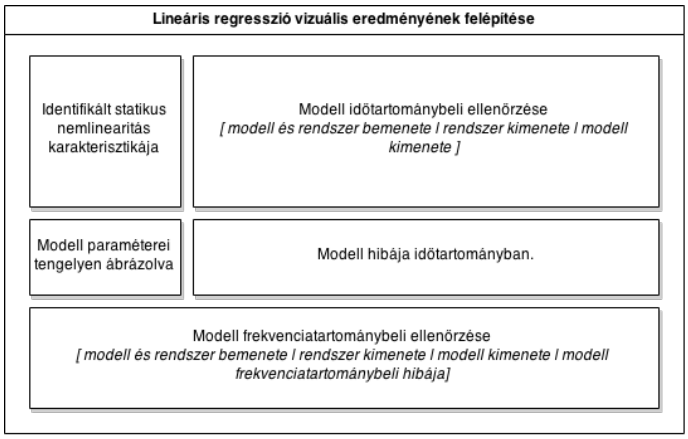
Az identifikációt elvégezzük mindkét bázisfüggvénnyel felszerelt modellre, kétféle fokszámmal identifikálva (40-edfokú, 10-edfokú), minden esetben két, romló *SNR-értékű* vá-

laszjelet alkalmazva. Minden mérést elvégzünk a kétféle ismeretlen rendszerre, így 16 darab elemezhető identifikációt kapunk eredményül, és ebből 8 darabot közlünk a később látható táblázatban.

5.5. Modell ellenőrzése

Az identifikáció elvégzése után tesztjelekkel gerjesztjük a modelljeinket és az ismeretlen rendszereket. A kapott válaszjelek alapján ellenőrizzük az eredményeket.

A rendszer rendelkezik egy elemző metódussal, ami képes összehasonlítani az identifikált modellt a valós, ismeretlen rendszerrel, és vizuálisan meg is jeleníteni az eredményeket:



5.4. ábra. *Vizuális eredmény értelmezése*

Az eredményeket az alábbi ábrákban láthatjuk a táblázatban megjelölt logika szerint[4]:

1	Sigmoid	Chebyshev modell	N=40	SNR=40dB	5.5. ábra
2	Sigmoid	Polynomial modell	N=40	SNR=40dB	5.6. ábra
3	Sigmoid	Chebyshev modell	N=10	SNR=10dB	F.7.1. ábra
4	Sigmoid	Polynomial modell	N=10	SNR=10dB	F.7.2. ábra
5	Hard Limit	Chebyshev modell	N=40	SNR=40dB	F.7.3. ábra
6	Hard Limit	Polynomial modell	N=40	SNR=40dB	F.7.4. ábra
7	Hard Limit	Chebyshev modell	N=10	SNR=10dB	F.7.5. ábra
8	Hard Limit	Polynomial modell	N=10	SNR=10dB	F.7.6. ábra

[4]A frekvenciatartománybeli ellenőrzéshez tarozó ábráknál az eredeti rendszer frekvenciabinjei pontokkal, míg a modell frekvenciabinjei körökkel lettek ábrázolva. Ez egy remek vizuális módszer ahhoz, hogy pusztán ránézéssel ellenőrizhessük, a modellünk úgymond "elkapta-e" a rendszer frekvenciatartománybeli viselkedését. Ha köröknek pont a közepén van a pont, az jelenti a közel tökéletes modellezést, míg ha kicsit eltér a pont a középponttól, vagy üres a kör, akkor ott a modell nem teljesíti a követelményeket.

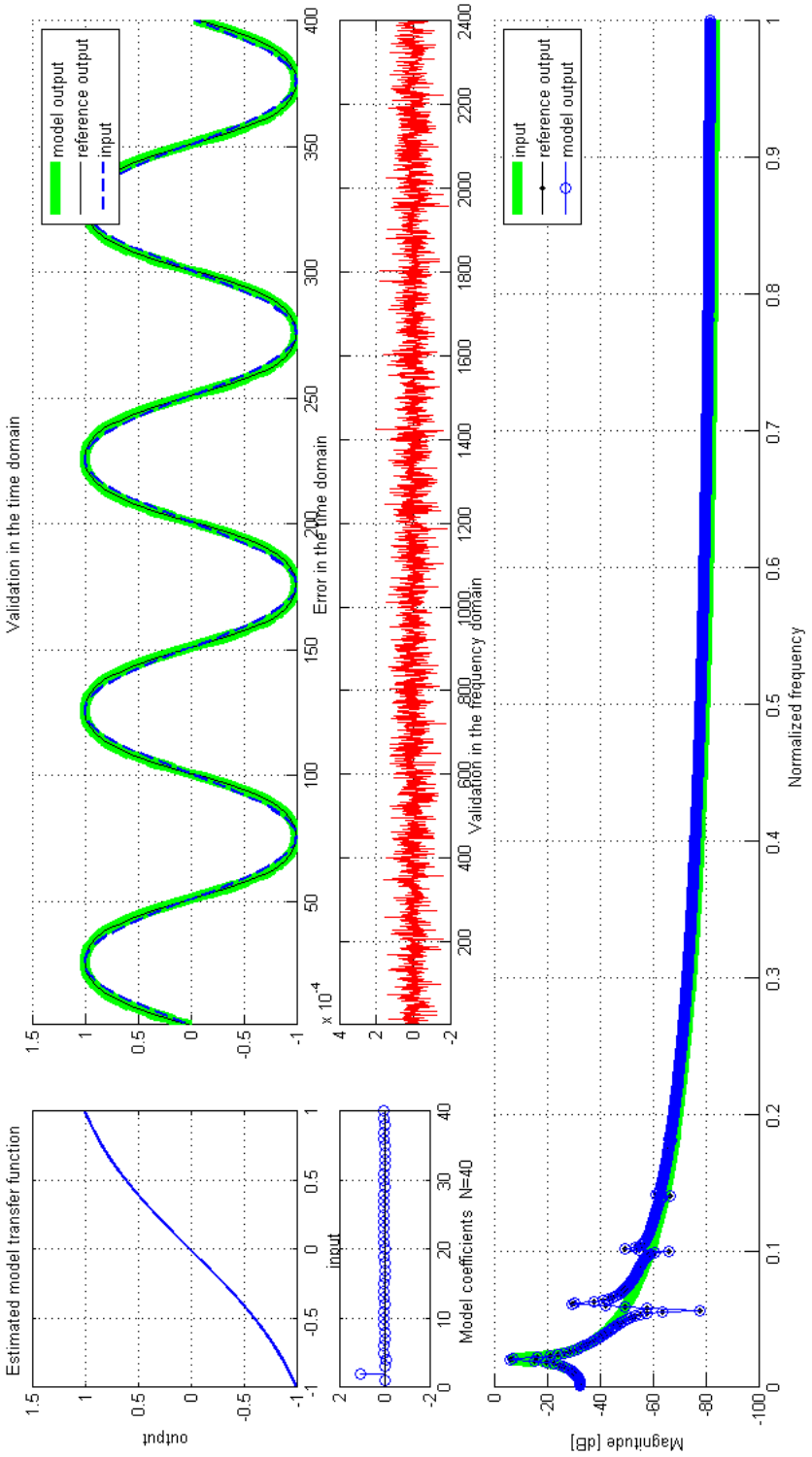
Az eredményekből egyértelműen látszik, hogy a Chebyshev és a hatványfüggvényes modell mindkét identifikált rendszer esetén hasonlóan teljesített.

A hasonló eredmény ellenére szembeötlő a különbség a két bázisfüggvényre épülő modell között. A polinomiális modell együtthatói 10^6 nagyságrendben helyezkednek el. Ez azt jelenti, hogy míg a modellre 1-nél kisebb gerjesztést adunk, addig jól fog működni numerikusan, azonban amint átlépjük az egységnyi határt, a rendszerünk numerikusan kezelhetetlenné fog válni, és össze fog omlani. Ez a Chebyshev bázisfüggvényes esetben nincs így. Ott a legnagyobb együttható értéke 1 közelében van, a többi jóval kisebb, tehát numerikusan stabilnak nevezhető.

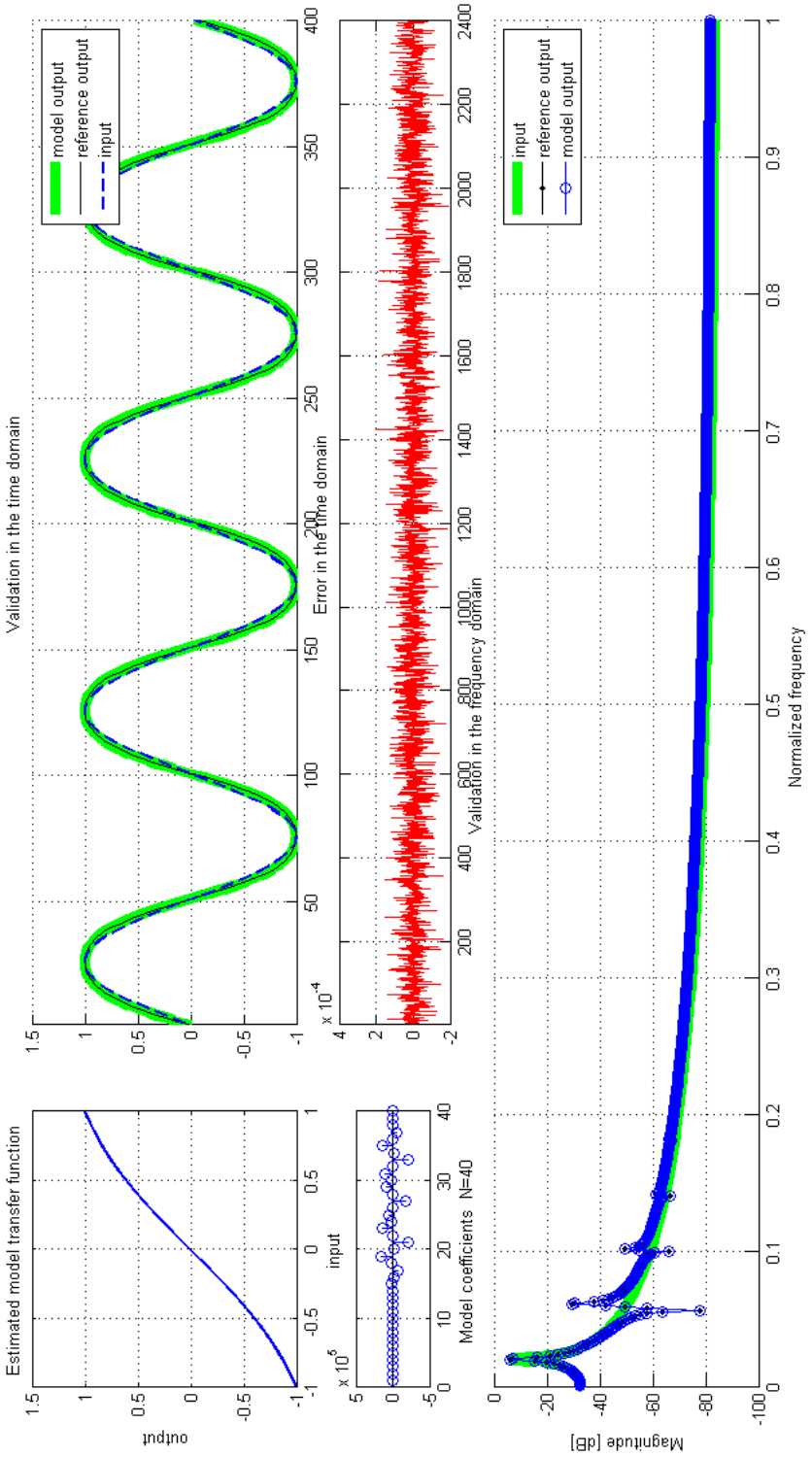
Mindkét modellről elmondható, hogy zajos környezetben nem adták vissza hűen a statikus nemlinearitás alakját, viszont spektrálisan hozzávetőlegesen jól közelítették a rendszerek viselkedését az alapharmonikusig. Használatuk korlátozások mellett akár még el is képzelhető, annak ellenére, hogy a statikus átviteli függvényük az eredetihez képest jelentős torzítással bír.

A lineáris regressziót csak statikus nemlinearításokra próbáltuk ki, de a módszer ugyanígy használható lenne dinamikus rendszerekre is. A feladatat semmiben sem változna. A bemeneti jelből a modell alapján elő kell állítani a regressziós-mátrixot, majd meg kell oldani a túlhatározott egyenletrendszert.

A módszer mindaddig használható eredményt ad, amíg a modelljeink nem tartalmaznak belső megfigyelhetetlen változókat. Ekkor egy lépésben a probléma megoldhatatlan, iteratív módszerekhez kell folyamodni.



5.5. ábra. *1: Szigmoid rendszer Chebyshev modell N=40 SNR=40dB*



5.6. ábra. *2: Szigmoid rendszer Polynomial modell N=40 SNR=40dB*

6. fejezet

Blokkos iteratív mérési módszer

A fejezet célja, hogy bevezesse az olvasót az előző fejezetekben megismert blokkos, iteratív identifikációs módszer használatába. A fejezet első részében betekintést nyerünk az identifikációs rendszer architektúrájába, majd elemezzük a működését egy egyszerű példán keresztül. A rendszernek nem közlöm a teljes forráskódját a függelékekben és a mellékletek között közlöm.

A rendszer tervezésénél az elsődleges szempont egy olyan belső architektúra létrehozása volt. Ehhez az architektúra elsőre indokolatlanul összetettnek tűnhet, de ahogy fokozatosan megismerjük, úgy válik világossá az Olvasónak, hogy minden absztrakció mögött valós indokok vannak.

A rendszer blokkvázlata a F.2.1. ábrán látható. Piros kerettel kiemelem azokat az osztályokat, amelyeket a használathoz feltétlenül ismerni kell. A zölddel kiemelt osztályok olyan osztályok, amelyek az architektúrát használva konkrét funkcionalitást valósítanak meg. Ezek az osztályok jó kiindulási alapot nyújtanak a rendszer megismeréséhez, és alapvető identifikációs alkalmazásokhoz.

Vegyük sorba a rendszert alkotó osztályokat, hogy melyiknek milyen szerep jut az identifikációban.

- **Interrogatable**

  Ez az interfész osztály felelős azért, hogy egy blokkról el lehessen dönteni, hogy valós blokk-e, vagy pedig egy helykitöltő blokk. Minden olyan osztálynak implementálnia kell ezt az interfészt, amelyik blokként akar viselkedni a rendszer szempontjából.

  Az Olvasónak nem kell ezzel az interfésszel foglalkoznia. Számára már implementálva lett az interfész a `Block` osztályban, amiből leszármaznia ajánlott.

- **Interconnectable**

  Az identifikációs rendszerben az egyes blokkokat tetszőleges módon össze lehet kötni. Ez az absztrakt osztály végzi el a szükséges összekötési és átmozgatási feladatokat,

és egy interfészt nyújt, hogy a blokkokat tetszőleges rendszerekbe lehessen kötni.

Az implementáció alapja a kettős láncolt lista (F.6), amiben minden elemet bármelyik másik elemhez hozzá lehet kötni, elvenni, kicserélni. A definiált interfész nagy szabadságot ad:

  - `attachBefore(blockBefore)` - a blokkot, amire meghívjuk beköti a paraméterként adott blokk elé
  - `attachAfter(blockAfter)` - a blokkot, amire meghívják beköti a paraméterként megadott blokk mögé
  - `clearInput` - törli a blokk bemenetét, amire meghívták
  - `clearOutput` - törli a blokk kimenetét, amire meghívták
  - `swapWith(other)` - felcsréli a blokkot a paraméterben megadott blokkra, minden összekötésükkel együtt[1]

- **Unconnected**

  Ez az osztály az egyedüli a rendszerben, ami az `Interrogatable` interfészre hamis értékkel tér vissza. Az egyes blokk-láncok lezárására használatos.

- **Identifiable**

  Ez az interfész felelős alacsony szinten, azért hogy az egyes blokkok indentifikálhatóak legyenek. Megvalósítja az iteratív identifikáció paraméterszétosztását, regresszor továbbítást normális és key term separation módban. További felelőssége, hogy eldöntse, hogy az adott blokk, a környezete által, milyen módban van. Az identifikátor osztály ezt az általa biztosított interfészt használja az identifikáció lebonyolítására.

  Ezen az interfészen keresztül adhatjuk a blokk-láncunkat hozzá az identifikátorhoz, és szintén ezen keresztül adhatjuk meg az identifikációhoz szükséges input és output adatsorokat is. A hozzáadás és az adatfeltöltés tetszőleges, láncon belüli blokkra hívható, az `Identifiable` osztály gondoskodik az adatok szétosztásáról minden egyes blokkhoz.

[1] Egy kétszeresen láncolt listában nem is olyan egyértelmű két elemet kicserélni. A módszer, amit kitaláltam a cseréhez a F.6. fejezetben látható. Elolvasása után látható, hogy a módszerrel le lehet egyszerűsíteni a gondolatmenetet jelentős mértékben.

- **Runnable**

  A tisztán absztrakt interfész-osztály biztosítja, hogy a blokkok identifikáció után akár közvetlenül is használhatóak legyenek külön futtató osztály nélkül. Hasonlóan a `Identifiable` interfészhez, itt is tetszőleges blokkra hívhatjuk a futtatási adatsort, az a teljes láncra érvényesülni fog, és az eredményt hívásunkkal meg is kapjuk.

- **Block**

  Az identifikációs rendszer egyik legfontosabb eleme. Egyrészt implementálja az összes szükséges, a felhasználók által irreleváns, de a rendszer számára nélkülözhetetlen funkcionalitást, másrészt egy absztrakt felelősséghalmazt hagy a felhasználóra, amin keresztül tetszőleges blokk alapú rendszer létrehozható. A `Block` osztály összes eddig tárgyalt interfésszel rendelkezik, és implementálja is őket a szükséges absztrakciós szintig. A funkcionalitások leszármaztatással érhetőek el, és a felhasználónak implementálási kötelezettsége van az absztrakt metódusokkal szemben. Az implementáció teljességéről könnyen meg lehet győződni, hiszen máskülönben nem is lenne futtatható a rendszer MATLAB környezetben.

  Megvalósítja még az egyes blokkok kirajzolási rendszerét, szétosztja az identifikációhoz használt, az `Identificator`-tól kapott paramétervektort az egyes blokkoknak, összegyűjti a generált regresszorvektort a blokkoktól.

- **Identificator**

  A `Block` rendszer után a másik központi eleme a rendszernek. Szintén egy absztrakt ősosztályról van szó, ami egy interfészt biztosít a felhasználónak, hogy tetszőleges identifikációs algoritmust valósítson meg. A dolgozatban az RLS algoritmus lett implementálva.

  Az osztály feladata még, hogy átadja a paramétereket a blokkoknak és elvegye tőlük a generált regresszorvektorokat.

- **Validator**

  Egyedüli feladata, hogy az identifikált blokkokat egy tesztjellel ellenőrizze, és az eredményeket vizualizálja.

- **SignalGenerator**

  Jelgeneráló osztály, ami a `Validator` osztályhoz használható, mint jelforrás. Hasonlóan az `Identificator` és a `Block` osztályhoz, ezt is az Olvasónak kell kiegészítenie, hogy a kívánt gerjesztőjelet képes legyen legenerálni. Ehhez rendelkezésre áll a szükséges interfész.

Az osztályok megismerése után nézzük meg, hogy miként használható a rendszer egy Hammerstein-modell identifikációjára.

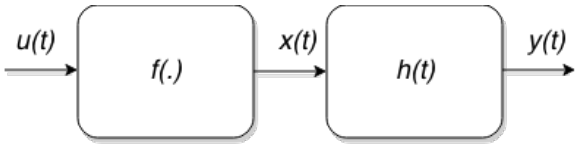
6.1. A rendszer használata

Ahogy azt a lineáris regressziónál is tettük, a blokkos identifikációs rendszer esetében is megnézzük, hogy hogyan alkalmazható az identifikáció öt lépése. A leírás során feltérképezzük az összes lehetőséget, amit a rendszer nyújt, és ki is próbáljuk egy egyszerű modellidentifikáción keresztül.

6.1.1. Kísérlet tervezése

A módszer bemutatásához vegyünk egy egyszerű Hammerstein-modellel identifikálható rendszert, ami álljon egy polinomiális bázisfüggvényekből álló folytonos statikus nemlinearitásból - ahogy azt az előző fejezetben is láttuk - , és egy viszonylag alacsony fokszámú IIR szűrőből, aminek az együtthatóit kézzel állítjuk be az átláthatóság érdekében.

Ha visszaidézzük az első fejezetekben lévő Hammerstein-modellt, emlékezhetünk, hogy a következő jelölésrendszert használtuk:



6.1. ábra. *Hammerstein-rendszer*

Az identifikálandó rendszerünk legyen a következő alakú:

$$x(t) = f_1 u(t) + f_2 u^2(t) \tag{6.1}$$

$$y(t) = b_0 x(t) + b_1 x(t-1) + b_2 x(t-2) + b_3 x(t-3) + b_4 x(t-4) - \\ - a_1 y(t-1) - a_2 y(t-2) - a_3 y(t-3) \tag{6.2}$$

f1 = 1
f2 = 0.5
b0 = 1
b1 = 0.2
b2 = 0.3
b3 = 0.4
b4 = 0.5
a1 = 0.6
a2 = 0.7
a3 = 0.8

A generált gerjesztőjellel információt gyűjtünk az ismeretlen rendszerről, majd a kimenet és a bemenet ismeretében megkezdhetjük az identifikációs eljárást a felépített modellünkön.

6.1.2. Kísérlet elvégzése, adatgyűjtés

Gerjesztőjelnek válasszunk ebben az esetben egy 2000 minta hosszúságú, normális eloszlású fehérzajt. Jelen esetben a könnyű ismételt felhasználás érdekében egy burkoló osztályt készítünk az ismeretlen rendszer számára, ami implementálja a `Runnable` interfészt, így a rendszer osztályai által is felhasználható lesz:

6.1. kódrészlet. *Adatgyűjtés*

```
u = (randn(2000,1));
un = UnknownSystem('hammer'); % burkolo osztaly, ami a fent definialt rendfszert
    valositja meg
d = un.run(u);
```

6.1.3. Modellstruktúra megválasztása

Az implementált rendszer másik nagy előnye a rugalmas bővíthetőségen kívül az, hogy rendkívül kevés kódot kell írni ahhoz, hogy egy egyébként bonyolult identifikációs algoritmust lefuttassunk, ráadásul az eredmények is rögtön elemezhetőek lesznek az osztályok nyújtotta funkcionalitásokon keresztül.

Az identifikációban csalunk egy keveset, ugyanis feltételezzük, hogy a priori tudásként ismerjük az identifikálandó rendszer struktúráját. Ez jelen esetben megengedhető, hiszen csak az implementált rendszer működését mutatjuk be így. A valóságban természetesen nem ismerjük az identifikálandó rendszer belső felépítését.[2]

Adjuk meg a Hammerstein-rendszerünk modelljét az identifikációs rendszernek!

6.2. kódrészlet. *Modell definiálása*

```
b_poly = PolynomialBlock('Poly',5); % a blokk neve es fokszama a parameter
b_iir = IIR_Block('IIR',4,4);

b_iir.attachAfter(b_poly);
```

A kódrészletből jól látható, hogy a kód mennyisége minimális, mégis egy teljesértékű modellt definiáltunk vele.

[2] *Black box* típusú identifikáció esetén nincs semmilyen információnk az identifikálandó rendszerről, csakis a kísérletezésünkben bízhatunk. *Grey box* identifikációnál ismerünk részleteket a rendszerről. Ilyen, amikor a rendszer fizikájából következtetni lehet, hogy ott milyen modellel lehetne jól leírni az adott viselkedést. A legegyszerűbb dolgunk identifikáció szempontjából a *white box* rendszereknél van. Ekkor ismeretes a teljes rendszer fizikája, ami remekül modellezhető.

Igaz a háromféle esetre, hogy az egyre több információval használt modelljeink egyre specifikusabbak lesznek, a lehetőségeink beszűkülnek a máshol való használatukhoz. Ez könnyen belátható, hiszen egy konrét fizikai rendszer struktúráját csak arra a rendszerre lehet használni, más rendszert, más struktúrával nem ír le. Azonban egy olyan rendszernél, ahol csak a bemenetet és a kimenetet tudjuk megfigyelni, a választott modell általánosabb is lehet, amit más rendszer identifikálására is tudjuk haználni.

6.1.4. Módszerválasztás, paraméterbecslés

Ahogy azt a rendszer felületes leírásánál írtuk, az identifikálási módszer adott. Az RLS algoritmust használja az identifikációs rendszer. Identifikációink ebből következően gyorsan képesek konvergálni.

Az RLS algoritmust megvalósító osztályból példányosítani kell egy objektumot, aminek meg kell adni az algoritmus kezdeti paramétereit és az előzőleg elkészített modellt.

6.3. kódrészlet. *Identifikátor létrehozása*

```
delta = .01;  % batorsagi tenyezo
lambda = .96; % felejtesi faktor
id = RLS(delta,lambda);

b_iir.addIdentificator(id);
```

A modell hozzáadásával az identifikációt végző objektum beregisztrálja a blokkokat, elkéri a paramétereik számát és várja, hogy átadjuk neki az identifikációhoz szükséges bemeneti és kimeneti adatokat.

A regisztráció sikerességéről értesítést is ad, valamint kirajzolja a kapott modellstruktúrát is:

6.4. kódrészlet. *Sikeres regisztráció kimenete*

```
Blocks are registered to the identificator with the structure of:

   [ Poly ]==>[ IIR ]
```

A bemeneti és a kimeneti adatokat az identifikátornak adjuk át, ami minden blokknak továbbítja majd azokat.

6.5. kódrészlet. *Bemeneti és kimeneti adatok átadása*

```
id.addIOData(u,d);
```

Az adatok átadása után elindíthatjuk az identifikációt:

6.6. kódrészlet. *Identifikáció elindítása*

```
id.start(10e-7);
```

A `start` metódusnak egy opcionális paraméteren keresztül meg lehet adni, hogy az identifikáció mekkora hiba esetén álljon le. A hiba a modell kimenete és a rendszer kimenete közötti eltérésre értendő. Ha nincs megadva paraméter, akkor az alapértelmezett futás lép érvénybe, amikor a rendszer a teljes megadott adatsoron végigmegy identifikáció alatt.

Az identifikáció során az identifikátor kimenete a következő:

6.7. kódrészlet. *Identifikáció kimenete*

```
Identification started..
   Number of blocks: 2
   Number of parameters: 13
   Length of input signal: 2000
   Error limit: 1e-06

   Error limit reached at the 582th iteration!

Identification has finished..
```

6.1.5. Modell ellenőrzése

Az identifikátor kimenete szerint az identifikáció rendben zajlott le úgy, hogy a modellünk teljesítette a kitűzött követelményt. 2000 mintából elég volt számára 582, hogy a minimalizáló RLS algoritmus kellően beállítsa a modellparamétereket.

A modell ellenőrzésére két lehetőség van. Egyrészről kirajzoltathatjuk az identifikált paraméterek alakulását az iterációs algoritmus lépései függvényében (6.2. ábra), másrészt a `Validator` osztály segítségével tesztelésképpen egy új gerjesztőjelet is ráadhatunk a modellre és az ismeretlen rendszerre, hogy összehasonlíthassuk a kimeneteiket (6.3. ábra[3]).

6.8. kódrészlet. *Identifikáció ellenőrzése*

```
id.plot(); % paraméterek alakulásának kirajzolása blokkonként

% Ellenorzeshez hasznalt multiszinuszos gerjesztojel generalasa
T = 0.3;
fs = 48e3;
f = [440,1000,4030];
mu = MultisineGenerator(T,fs,f); % generator objektum peldanyositasa

% Ellenorzest vegzo osztaly peldanyositasa es hasznalata
val = Validator(un,b_iir);
val.validateWithGenerator(mu);
```

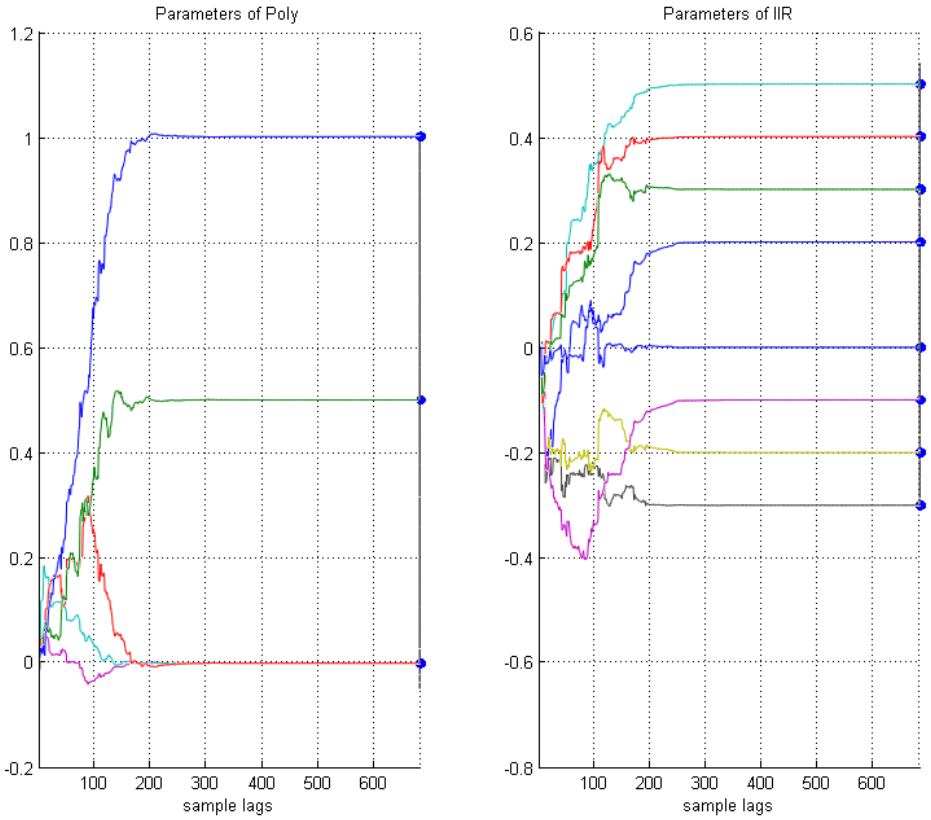
6.1.6. Key term separation

Ahogy azt az előző részben láttuk, minden egyes blokk rendelkezik olyan metódusokkal, amelyekkel a rendszer képes tőlük elkérni a paramétereik számát inicializáláskor, majd az identifikáció alatt a regresszorvektorok számolását kérhetik adott paramétervektorra.

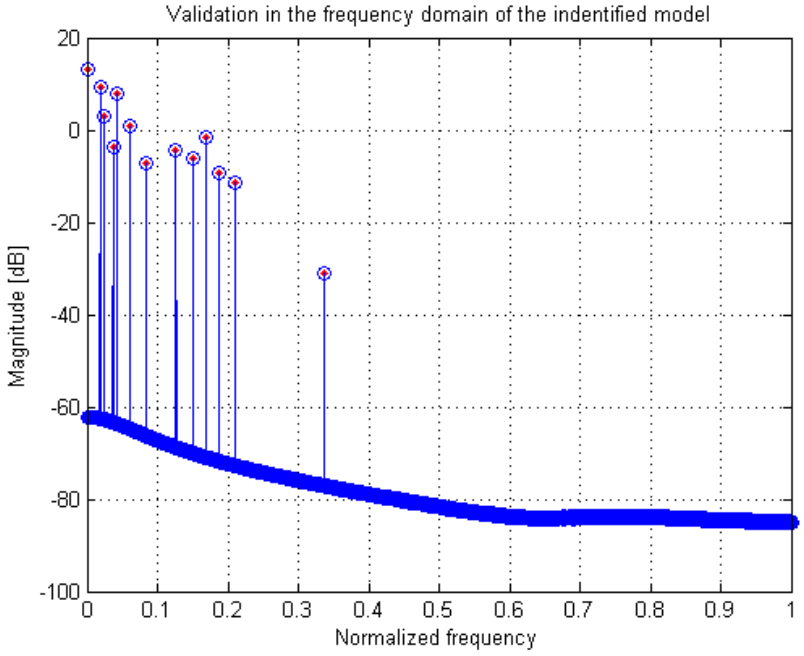
A rendszer belsejének szemszögéből ez a következőképpen játszódik le:

1. Az identifikátor példányosítása után a már elkészült blokkos modell hozzáadásakor, a blokkok maguktól beregisztrálnak az identifikátorhoz, átadva neki a szükséges paraméterszámokat. Az identifikátor a paraméterszámokat összegzi és az identifikáció

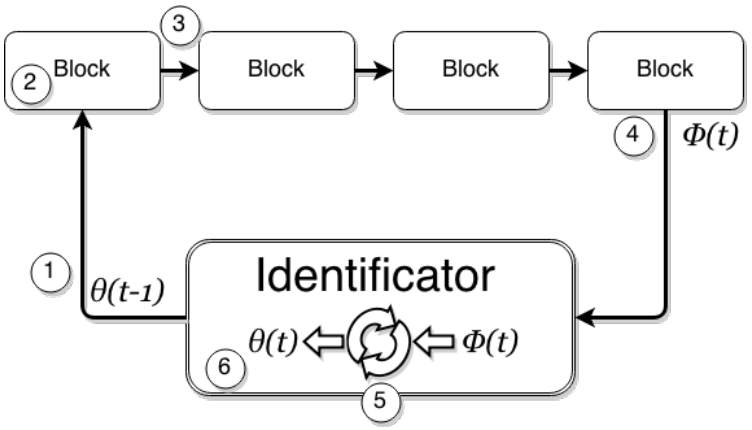
[3] A frekvenciatartománybeli összehasonlításhoz ismét a jól bevált módszert alkalmazzuk, miszerint a valós rendszer spektrumát pontokkal, az identifikált modell spektrumát pedig körökkel jelöljük. Ha a körök "elkapják" a pontokat, akkor az identifikáció sikeres volt.



6.2. ábra. *Paraméterek alakulása az identifikáció folyamán Hammerstein-rendszer esetén*



6.3. ábra. *Tesztgerjesztéssel való ellenőrzés Hammerstein-rendszer esetén*



6.4. ábra. *Az identifikációs rendszer működése*

megkezdésekor beküldi az első blokkba az összes blokk által igényelt hosszú nulla értékű[4] paramétervektort, hogy a hozzá tartozó regresszorvektornak megkezdjék a kiszámolását.

2. A blokkokban a felhasználó elől elrejtett funkcionalitás levágja a blokkhoz tartozó kis paramétervektort, és a blokk helyzete alapján eldönti, hogy key term separation módban vagy normál módban van-e a blokk. Az ennek megfelelő felhasználó által implementált metódust hívja meg a levágott paramétervektorral, amire a metódus összerakja a blokknak megfelelő regresszorvektort, amit visszatérési értékként kap meg a blokk belső funkcionalitása a blokk aktuális bemenetével együtt.

3. A belső funkcionalitás továbbadja a csonkított paramétervektort, a blokk kimenetét, és az eddig elkészült regresszorvektort a következő blokk rejtett funkcionalításának ami szintén leveszi a paramétervektorból a neki szükséges paramétereket, és meghívja a felhasználó metódusát. A metódus összerakja a regresszorvektort és a blokk kimenetét. A kapott regresszorvektort összefűzi az előző blokk regresszorvektorával, majd szintén továbbítja a csonkított paramétervektort, az egyre gyarapodó regresszorvektort és a blokk kimenetét.

4. Az utolsó blokkban a paramétervektorból pont annyi marad, mint amennyire a blokknak szüksége van. A regresszorvektor elkészítése után hozzáfűzi a kapott regresszorvektorhoz, és az így teljes hosszúságúra fokozatosan kiegészített regresszorvektort átadja az identifikátornak.

5. Az identifikátor a kapott regresszorvektorral képes lesz lefuttatni egy iterációt, a felhasználó által definiált algoritmus segítségével, aminek eredménye egy új paramétervektor lesz.

6. Az új paramétervektort ezután ismét átadja az első blokknak, hogy az megkezdje a következő regresszorvektor összeállítását az aktuális paraméterekre.

[4]Inicializáláskor a paramétervektor nulla értékű.

Látható, hogy a rendszer a key term separation elvet követi. Minden blokk kiadja az éppen aktuális kimeneti értékét, amit a következő blokk el tud menteni, és felhasználni a saját regressziós vektorának előállításában.

A dolgozatban a rendszer csak két blokkig lett kihasználva, azaz csak sima Wiener és Hammerstein-rendszereket identifikáltam vele, de a módszer kiterjeszthető jóval összetettebb konstrukciók identifikálására is.

A következőkben nézzünk meg egy valós, nagy paraméterszámú alkalmazást, ahol egy Wiener-modellel közelíthető rendszert identifikálunk, szintén MATLAB-on belül. Az eredmény kiértékelésénél látható lesz egy eddig nem említett funkció, ami lehetővé teszi, hogy a paraméterek alakulását mutató ábrát térben elforgassuk, ezzel intuitívabbá téve az eredmények kiértékelését.

A rendszer álljon egy FIR blokkból, amit egy hatványfüggvényes nemlinearitás követ. Az ismeretlen rendszer FIR szűrője egy 50 tap-es szűrő legyen. FIR szűrők esetében a paraméterek kirajzolják magának a dinamikus rendszernek az impulzusválaszát, ami az elforgatható ábrán jól látható lesz (6.5. ábra). Az 6.6. ábrán látható, hogy az identifikáció sikeres volt.

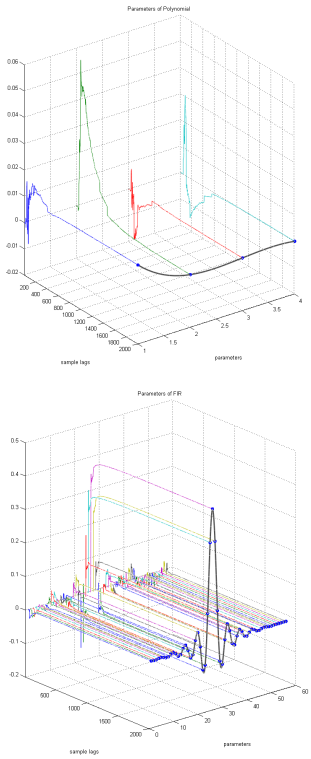
A Wiener-rendszer identifikációját elvégző kód az alábbi részben látható. A felépítés teljesen azonos a részletesen tárgyalt Hammerstein-rendszer identifikációjával, azzal a különbséggel, hogy a blokkok összekapcsolása és az ismeretlen rendszer hívása különbözik.

6.9. kódrészlet. *Wiener-rendszer identifikációja*

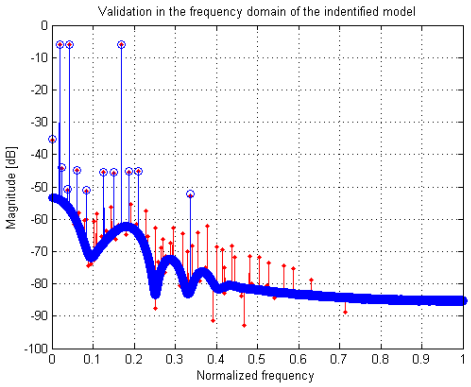
```
%% Wiener system identification
clear all; close all; clc;

b_fir = FIR_Block('FIR',55);
b_poly = PolynomialBlock('Polynomial',5);

b_fir.attachBefore(b_poly);

%% adding the identificator

delta = 1;
lambda = .996;
id = RLS(delta,lambda);

b_fir.addIdentificator(id);

%% initializing with the input and output
u = (randn(2000,1));

un = UnknownSystem('wiener');
d = un.run(u);
id.addIOData(u,d);

%% start identification

id.start();
```

```
27  id.plot();
28
29  val = Validator(un,b_fir);
30  T = 0.3;
31  fs = 48e3;
32  f = [440,1000,4030];
33  mu = MultisineGenerator(T,fs,f);
34  val.validateWithGenerator(mu);
```

A fejezetben bemutatott identifikációs rendszerrel az Olvasó képes lesz kibővíteni a meglévő rendszert, és identifikációs feladatokat teljesíteni. A méréshez implementált blokkokat és a továbbfejlesztéshez használható blokk API-t a függelékek F.1 fejezetében lehet megtekinteni. A teljes rendszer blokkvázlatát az F.2.1. ábra mutatja.



6.5. ábra. *Wiener rendszer paramétereinek alakulása 3D nézetben*



6.6. ábra. *Az identifikált Wiener rendszer ellenőrzése*

7. fejezet

Mérés a gyakorlatban

Az előző fejezetekben megismertük az implementált mérési eljárások elméletét, felépítését és használatát. Ebben a fejezetben használatba vesszük a módszereket, és identifikálunk velük egy valós hangszórót.

7.1. Adatgyűjtő rendszer

A mérés során alkalmazott adatgyűjtő rendszer szinte bárki számára elérhető. Egy PC-ből és egy külső hangkártyából áll. A mérést az Adobe Audition 3.0 programmal végeztem. Mint ahogy a legtöbb PC-n elérhető hangrögzítő szoftvernek, ennek programnak is hátránya, hogy nem tudja biztosítani a szinkronizált lejátszást és felvételt, ezért a rendszer eredendő késleltetését[1] nem tudja magától eliminálni.

Egy megoldás a problémára a manuális késleltetés visszaállítása. Ehhez a gerjesztőjelbe bele kell integrálni egy jól felismerhető mintát, ami a felvétel elején az identifikáló jel megszólalása előtt biztosít egy szinkronizálási pontot. A módszer lényege, hogy mind a gerjesztésben, mind a válaszban meg található lesz ez a minta, így a két sávot össze lehet kézzel szinkronizálni.

Mikrofonnak egy Beyerdynamic Opus 29 S dinamikus mikrofont használtam, ami egy Focusrite Scarlett 2i4 hangkártya szimmetrikus bemenetéhez volt kötve.

7.2. A mérendő hangszóró

A dolgozatban bemutatott módszerek kipróbálására találni kellett egy olyan kisméretű hangszórót, amin remekül szemléltethetőek a dinamikus hangszórók torzításai. A választás egy kis hordozható Marshall MS-2 gitárerősítő hangszórójára esett. A benne található 6 cm átmérőjű 8 ohmos, 0.8 wattos kis hangszóró kellőképpen torzít már kis amplitúdókon is, így méréseink szempontjából tökéletes szemléltetőeszköz.

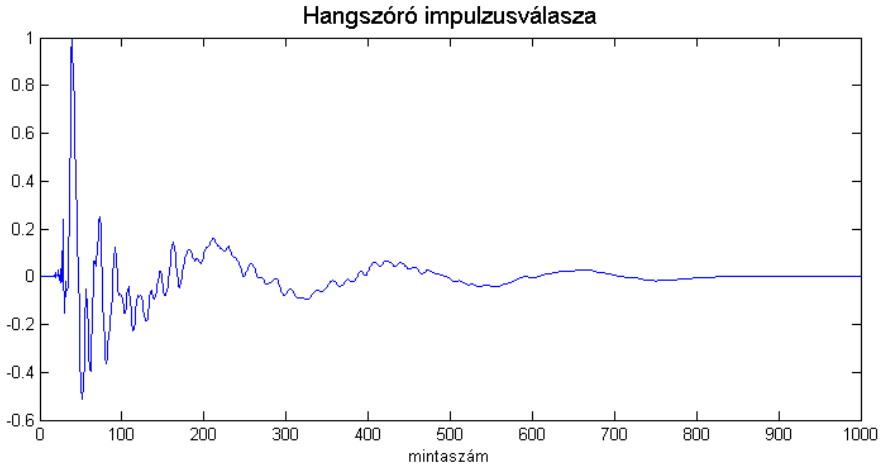
[1] A késleltetés abból áll, hogy a rendszerre adott bemenet csak egy bizonyos késleltetéssel jut érvényre. Ennek oka a digitális feldolgozás során használt megoldások: AD/DA konverzió, blokkos feldolgozás, operációs rendszer miatti késleltetések.

7.1. ábra. *Marshall MS-2 hangszórója*

7.2.1. Lineáris viselkedés

A hangszóró lineáris viselkedésének vizsgálatára a legcélszerűbb egy olyan módszert választani, ami képes tolerálni a nemlineáris viselkedést is. Sajnos a hagyományos mérési módszerek, nem rendelkeznek ilyen tulajdonságokkal [59], és a nemlineáris viselkedés miatt a mérésbe hiba csúszik. Általában valamelyest eliminálható a nemlineáris hatás, ha a mérési jel jelszintjét csökkentjük olyan mértékben, hogy a nemlineáris hatások még ne vagy csak kevésbé lépjenek érvénybe. Azonban ez nem minden esetben kifizetődő. Kis jelszinteken megnő a jel-zaj viszony, ami szintén korlátozza a mérés pontosságát.

A megoldás az előző fejezetekben ismertetett swept sine alapú mérés lesz. A rendszer válaszának a mérése és az inverz szűrővel való dekonvolúció után rendelkezésünkre áll az egyes felharmonikusokhoz tartozó jól elkülönített impulzusválasz sorozata. Ebből a legutolsó tag maga a lineáris impulzusválasz, amit a 7.2. ábrán láthatunk is.

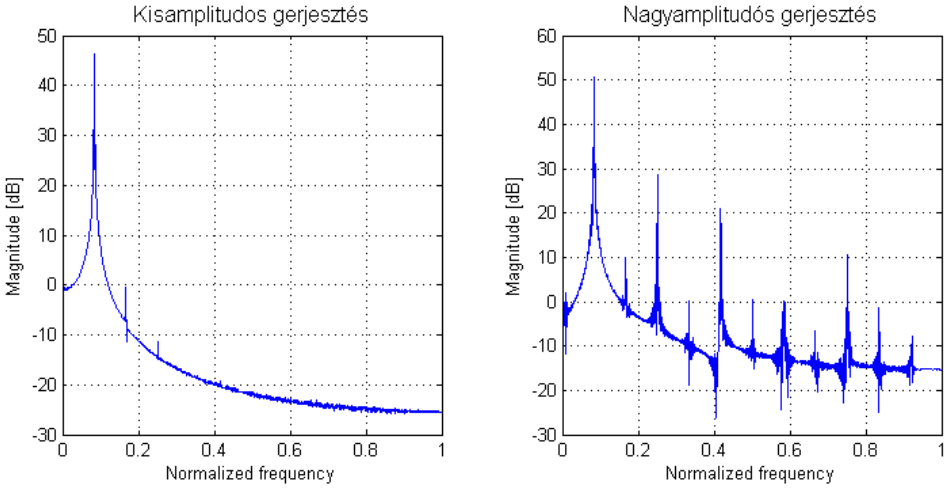


7.2. ábra. *Hangszóró lineáris impulzusválasza*

Az identifikálási mérések előtt a hangszórót megvizsgáltam, hogy milyen mértékű harmonikus és intermodulációs torzításokkal rendelkezik.

7.2.2. Harmonikus torzítás

Harmonikus torzítások mérésére szinuszos jelet használtam, aminek az amplitúdóját változtattam a külső hangkártyán. Két mérést végeztem: az egyiket kis amplitúdón (100 mVPP), a másikat nagyobb amplitúdóval (350 mVPP). A relatív kis jelamplitúdók oka, hogy már ezeken a szinteken is az átlagos hangszórókhoz képes nagy torzítást produkált a mért hangszóró. Az úgynevezett one-tone mérések eredményei a 7.3. ábrán láthatóak spektrogramban ábrázolva.



7.3. ábra. *Hangszóró harmonikus torzításai kis és nagy jelszinten*

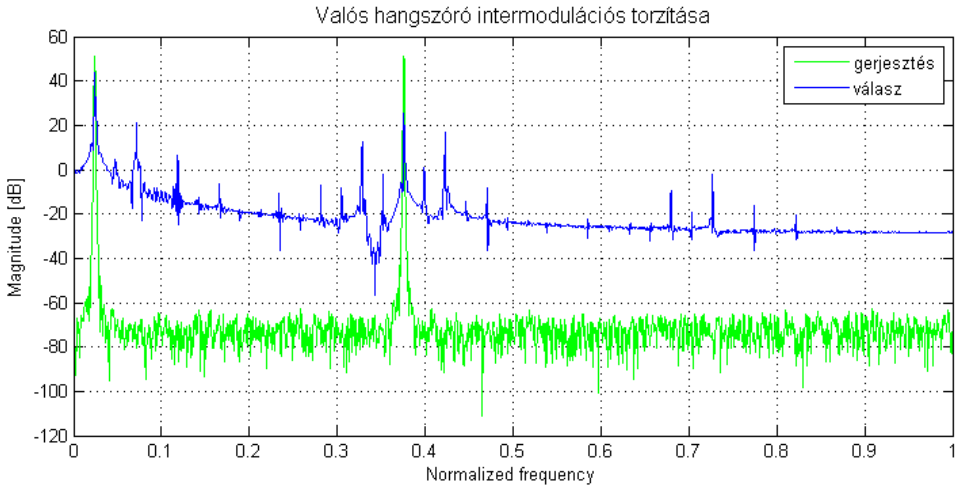
Ahogy az az ábrán is jól látható, már kis amplitúdón is jelentős harmonikus torzítással bír a kis hangszóró.

7.2.3. Intermodulációs torzítás

Az intermodulációs viselkedést az úgynevezett dual-tone méréssel tudjuk megvizsgálni. Ekkor egy két szinuszból álló jellel vizsgáljuk a mérendő rendszert. Ha a rendszer intermodulációs torzítással bír, akkor kimenetének spektrumában megjelennek a gerjesztőjel két szinuszos frekvenciáinak lineáris kombinációi. Ebből a célból érdemes olyan gerjesztőjelet alkalmazni, amelyeknek frekvenciája nem egész számú többszörösei egymásnak, és így jól elkülöníthetőek lesznek a generált spektrális komponensek.

Az intermodulációs mérés eredménye a 7.4. ábrán látható. Megfigyelhető, hogy a hangszóró erős nemlineáris viselkedéssel rendelkezik, ami gitárerősítő révén az előnyére válik, azonban mint általános hangszóró, nagyon gyengén szerepel.

A választott hangszóró vizsgálataival bebizonyosodott, hogy a további méréseinkhez megfelelő alany lesz.



7.4. ábra. *Intermodulációs torzítás vizsgálata dual-tone méréssel*

7.3. Swept sine modell identifikációja

A hangszóró alapvető nemlineáris tulajdonságainak megismerése után most alkalmazzuk a már bemutatott swept sine identifikációs módszert.

A mérés megkezdéséhez generálni kell egy gerjesztőjelet. Ezt a korábban ismertetett módszerekkel az alábbi paraméterek felhasználásával tettem meg:

f1 = 22 Hz
f2 = 24 kHz
fs = 48 kHz
L = 15 s

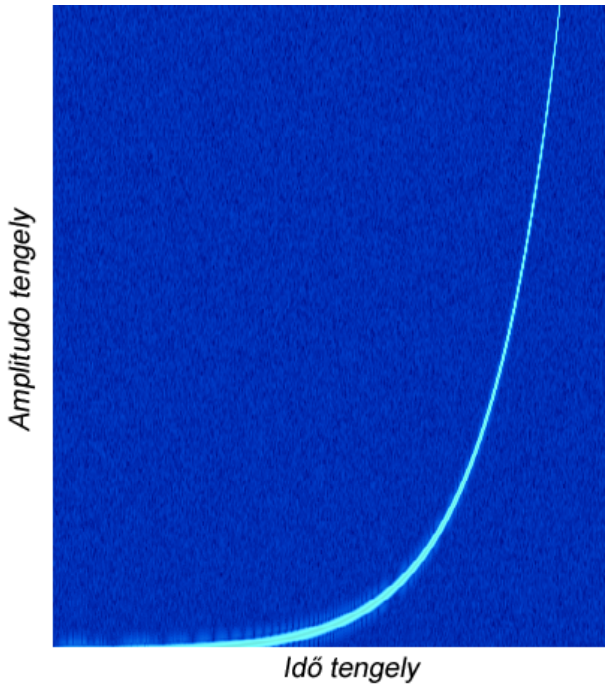
A gerjesztőjel spektrogramja a 7.5. ábrán látható. Megfigyelhető, hogy a frekvencia valóban exponenciálisan emelkedik az idő függvényében.

A mérés elvégzéséhez szükség van még a dekonvolválójelre, amivel a mért választ dekonvolválva megkapjuk a swept sine mérés alapját képező mintasort. Ez jól elkülönítve tartalmazza az egyes felharmonikusokhoz tartozó impulzusválaszokat. A gerjesztőjelhez generált dekonvolválójel a 7.6. ábrán látható. Megfigyelhető, hogy pont a gerjesztőjel inverzét képezi.

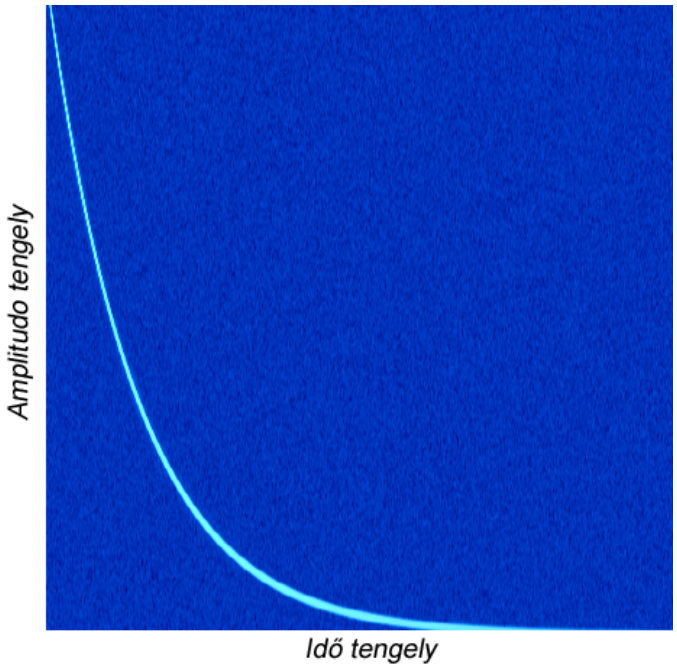
Erről meg is tudunk győződni, ha vesszük a gerjesztőjel és a dekenvolválójel konvolúcióját. Az eredmény az 7.7. ábrán látható majdnem tökéletes Dirac delta lett.

A mérés elvégzése után a hangszóró válasza a 7.8. ábrán látható. Jól megfigyelhetőek az ábrán a harmonikus torzitás miatt keletkező komponensek. A spektrogramon ezek egymástól egyre távolodó frekvenciakomponensként jelennek meg.

A mérés a hangszóró válaszának mérésével lezárult, az identifikáció további lépései offline történnek. Ehhez a kapott választ először dekonvolválni kell a dekonvolválójellel, majd a kapott, jól elkülönített impulzusválaszokat kivágva, és a modellstruktúra megfelelő ágába berakva megkapjuk a hangszórót megvalósító digitális hangszórómodellünket.



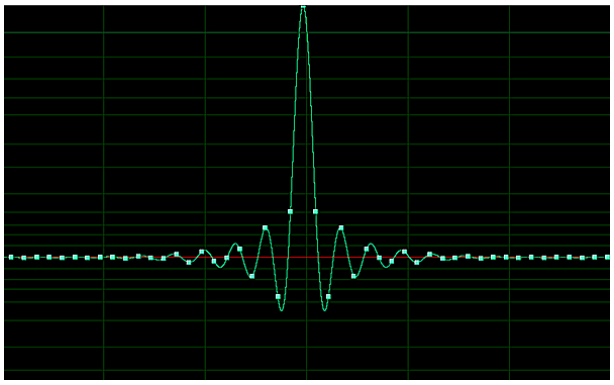
7.5. ábra. *Swept sine mérés gerjesztőjele spectrogramon ábrázolva*



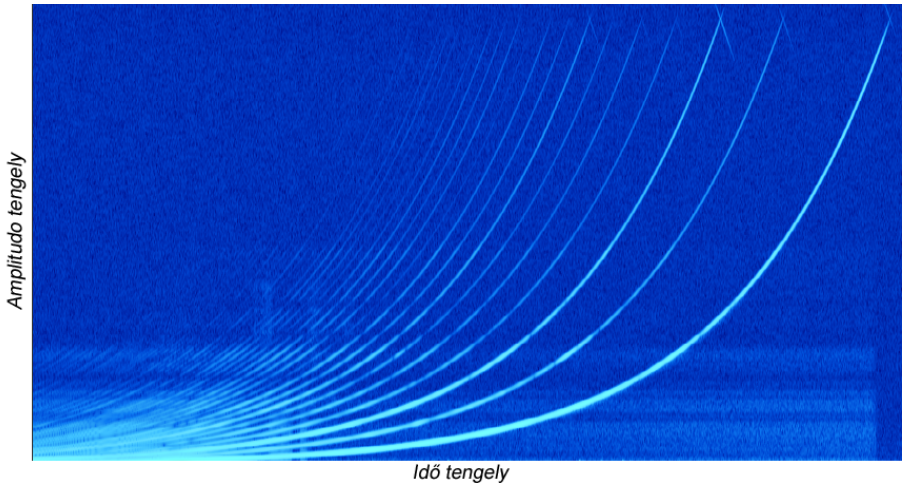
7.6. ábra. *Swept sine mérés dekonvolválójele spectrogramon ábrázolva*

Az identifikációt tizedfokú modellel végeztem, azaz 10 párhuzamos ágból álló polinomiális Hammerstein modellt alkalmaztam, ahol az egyes ágakban egy Chebishev-polinom és egy FIR szűrő található. A kapott impulzusválaszokat a 7.9. ábra szemlélteti.

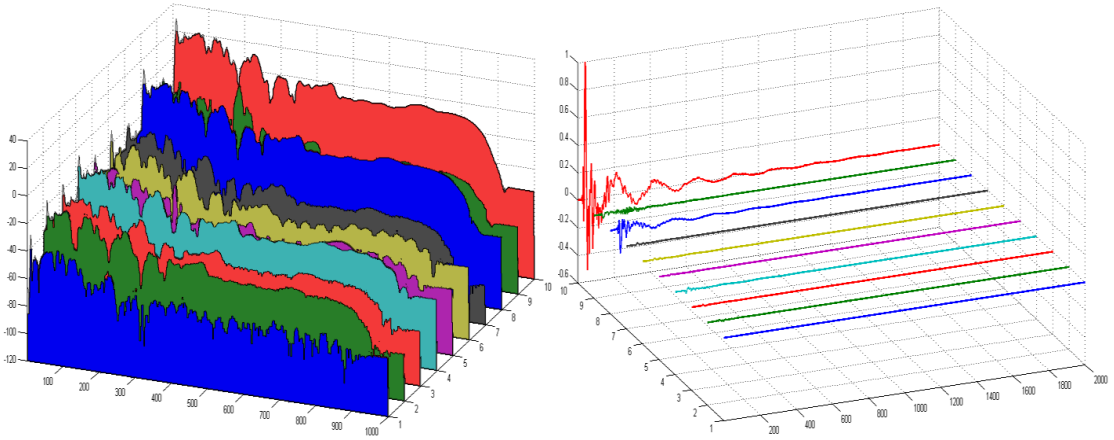
Az egyes felharmonikusokhoz tartozó impulzusválaszokkal kiegészített modellel is elvégeztem a hangszórón végzett eredeti torzításvizsgáló dual-tone-os mérést, és a modell a még viszonylag alacsony fokszáma ellenére is hűen utánozta az eredeti hangszóró viselkedését. A kapott mérési eredmény a 7.10. ábrán látható.



7.7. ábra. *Gerjesztőjel és dekonvolválójel konvolúciója*

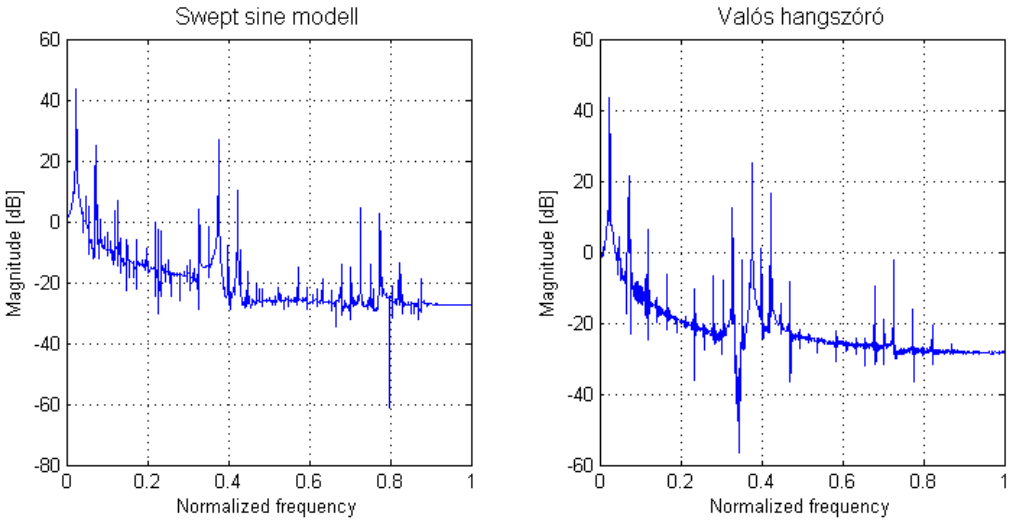


7.8. ábra. *Hangszóró válasza a swept sine mérés során spectrogramon ábrázolva*



7.9. ábra. *Dekonvolvált és kivágott felharmonikusokhoz tartozó impulzusválaszok és azok spektruma*

A modell még viszonylag kis fokszám mellett is aránylag hűen visszaadta az identifikált hangszóró spektrális jellegzetességeit. Nem csak speciális gerjesztőjelek dolgozhatóak fel általa, hanem hangfájlok is. A modellel processzálva egy hangfelvételt, az eredmény megdöbbentően hasonlított az identifikált hangszóróval lejátszott hangfelvételre.



7.10. ábra. *Eredeti hangszóró és modell viselkedése dual-tone tesztelésre*

7.4. Blokk alapú identifikáció

A swept sine alapú identifikáció sikereit sajnos nem sikerült reprodukálni a blokk alapú identifikációs rendszerrel.

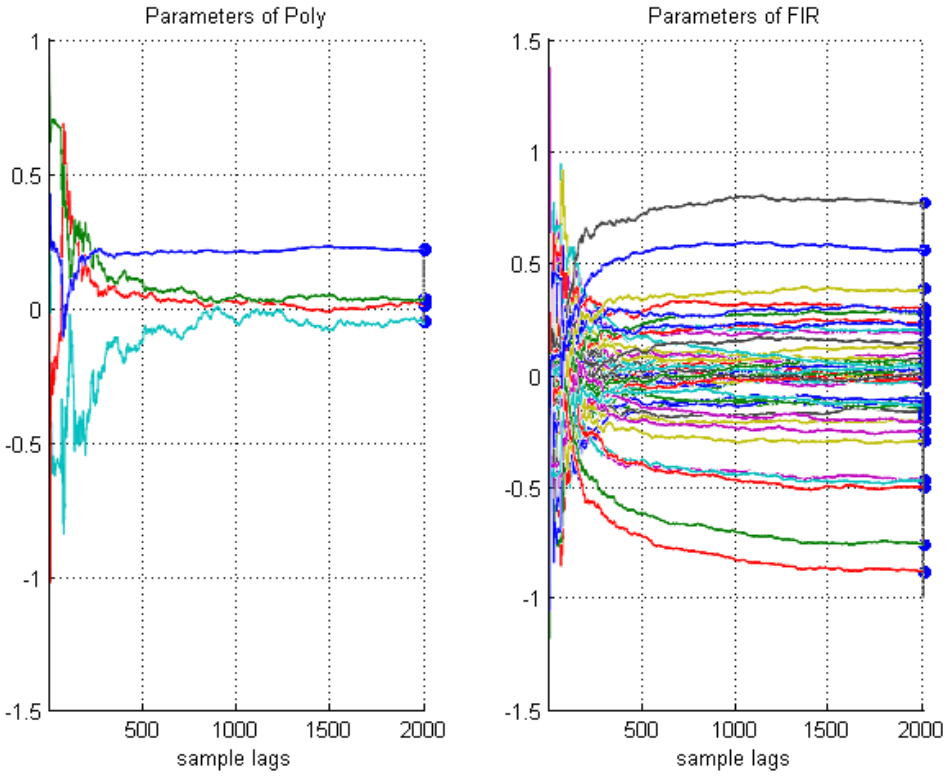
A mérést hasonlóan végeztem, mint a swept sine identifikáció esetén. Azonban az identifikációs algoritmus lefutása után a módszer nem hozta az ideálisabb tisztán MATLAB-os környezetbeli teljesítményét. Akárhogyan is módosítottam az identifikáló algoritmus paramétereit, az nem volt képes konvergenciára. A közölt ábrákon a látszólagos konvergencia csak megtévesztés. Ha az identifikáció hosszát megnöveljük, akkor a paraméterek megállíthatatlanul növekednek vagy csökkennek tovább.

Ennek ellenére kipróbáltam a modell működését egy olyan paramétervektorral, amelyek abban az állapotban voltak éppen aktuálisak, amikor a konvergencia-diagram változóinak deriváltja egységesen közel zérus értéket vett fel egy hosszabb szakaszon át. A modell a vártnál valamennyivel jobban teljesített, de a swept sine módszer eredményét meg sem közelítette.

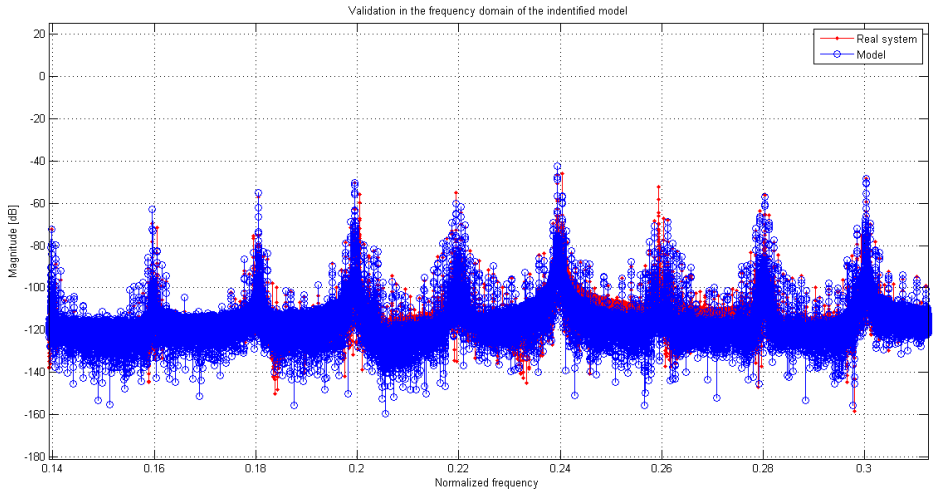
Az eredmény a 7.12. ábrán látható. A modell képes produkálni majdnem minden felharmonikust, amit a valós rendszer is, azonban nem megfelelő amplitúdókban.

Sajnos ha nem ideális az identifikálandó rendszer az algoritmus számára (nem a modellben szereplő modell alapján szűrjük a jelet), akkor az identifikáció nem lesz sikeres.

Az identifikáció elvégzése után ellenőrizni is kell az elkészült modellt. Ehhez egy külön gerjesztést használtam, amit először lejátszottam és lemértem a hangszórón keresztül, az identifikált modellen is futtattam, majd a kapott két kimeneti jelet összehasonlítottam. Az eredmény a 7.12. ábrán látható. Megfigyelhető, hogy a modell produkálja az szinte az összes felharmonikust, amit a hangszóró is produkál, csupán a visszaadás amplitúdójában van eltérés. A modell a nem konvergáló identifikáció ellenére is képes volt modellezni a mért hangszórót egy bizonyos szintig.



7.11. ábra. *Modell paramétereinek beállása*



7.12. ábra. *A modell ellenőrzése*

8. fejezet

Eredmények összegzése

8.1. Összegzés

A dolgozat elején felelevenítettem a lineáris és nemlineáris rendszerekről meglévő ismereteket, leírásuk formáit. Az elméleti áttekintés után megvizsgáltuk a dinamikus hangszórók felépítését és jellemző nemlineáris torzításainak forrásait.

Az identifikáció fogalma és öt lépése után bemutattam három különböző identifikációs eljárást (lineáris regresszió, iteratív identifikációs eljárás, swept sine mérési módszer), amiből az utóbbi kettőt tovább is vittem az implementáció szintjére.

A blokkos identifikációs rendszerhez egy objektumorientált mérési rendszert készítettem, ami egyszerűsíti a mérési elrendezések összeállítását. A rendszer leveszi a terhet a felhasználóról úgy, hogy a megoldandó implementációs feladatot jól elkülönített objektumokra bontja. A felhasználó építkezhet a meglévő blokkokból, vagy új blokkot is implementálhat a meglévő ősosztályokból leszármaztatva. Az objektumorientált megoldás lényegében a key term separation elvére építve, blokkosítva fűzi össze a regresszorvektort [1], majd iterációként adja át a használt rekurzív identifikáló algoritmusnak. A rendszer további előnye, hogy moduláris. Az objektumok absztrakt interfészeken keresztül kommunikálnak, így bármikor le lehet cserélni őket egy velük hasonló interfészt megvalósító másik objektummal. Ha például a felhasználó az RLS algoritmus helyett az LMS algoritmust szeretné használni, akkor egyszerűen az identifikáló burkolóosztálynak az LMS algoritmust megvalósító objektumot adja át.

Szimulált környezetben a módszer remekül működött, azonban a gyakorlatban kipróbálva nem bizonyult használhatónak. A hiba oka a nem ideális feltételek megszűnésében keresendő.

A swept sine eljárás tisztán a jelfeldolgozás elméletére építve identifikálja a mérendő rendszereket egy viszonylag újnak számító módszerrel. Az újítás a gerjesztőjel tervezésében jelentkezik. A dolgozatban bemutatásra kerül a módszer elmélete és az implementációhoz szükséges anyagok is. A módszer szimuláltan és a gyakorlatban is remek eredményeket

[1] A regresszorvektor összeállítása olykor nem egyértelmű feladat, de a rendszerrel mindez automatizáltan történik meg.

adott. A dolgozat fő eredménye a swept sine eljárásra épülő identifikációs környezet. A mellékletben megtalálhatóak hangfájlok is, amik jól mutatják a módszer sikerességét.

A gyakorlatban is kipróbált két identifikációs módszer összehasonlítását az alábbi táblázat tartalmazza:

	Swept sine modell	**Blokk alapú modell**
Gerjesztőjel	Logaritmikus sweep	Fehérzaj
Identifikáció alapja	Jelfeldolgozás	Lineáris algebra
Pontosság	Már kis fokszámok esetén is meglepően jól visszaadta az identifikálandó rendszer spektrális viselkedését.	Ideális rendszert[2]identifikálva a módszerrel nagy pontosságot lehet elérni, azonban nem ideális rendszer esetén a pontosság nem elégséges.
Számításigény	Az identifikált modell fokszámát növelve a számításigény egy újabb szűrő beiktatásával változik. Identifikációnál előnyös a minél hosszabb gerjesztőjel használata, ezért az identifikáció számításigényes feladat.	A modell felépítése erősen befolyásolja az identifikáció számításigényét, hasonlóan a gerjesztés hosszától. Minden mintánként egy iterációval számolva a módszer számításigényes.
Identifikáció bonyolultsága	Jelfeldolgozás elméletére építve a számítások elvégzése nem bonyolult: konvolúció, FFT.	A lineáris algebra eszköztárát használva mátrixműveletekre épít. Az implementált objektumorientált rendszer csupán a regresszorvektor összefűzését segíti moduláris formában.

Összegezve elmondható, hogy a dolgozatban tárgyalt identifikációs módszerek közül a swept sine módszer mind stabilitásban, mind teljesítményben kiemelkedően jól teljesített. Ha az identifikációs feladat szempontjából megfelelő a párhuzamos Hammerstein-rendszer alapú modell, akkor bátran használhatjuk.

A blokk alapú modell csak szimulált környezetben teljesített jól, a valóságban nem szerepelt fényesen. Ugyan az identifikált modell tudta valamilyen szinten produkálni az eredeti rendszer spektrális tulajdonságait, de az nem volt kielégítő.

8.2. További lehetőségek

Ugyan a dolgozat érdemi része véget ért, de a következő oldalakon még további érdekességek várják az Olvasót. Ezek közül kiemelném a MATLAB objektumorientált programozásáról szóló F.5. fejezetet. Ez az a terület, ami a MATLAB programozásával aktívan foglalkozók számára egy fehér folt szokott lenni, pedig használatával rendkívüli módon leegyszerűsíthető és strukturálttá tehető a munka. Bátorítok mindenkit az objektumorientáltság elsajátítására.

[2]Ideális identifikált rendszer alatt olyan rendszert értek, amire az alkalmazott identifikációs modell illeszkedik. Fokszembeli eltéréseket még el képes tűrni a módszer, azonban struktúrabeli változásokra már nagyon érzékeny, és nem megfelelő identifikációs modell alkalmazása esetén a konvergencia nem következik be.

Teszem mindezt azért is, mert a dolgozatban közölt rendszer *bővíthető*. Az architektúra megengedi, hogy mind a modellek, mind az identifikációhoz használható algoritmusok lecserélhetőek legyenek plug-and-play[3] módon. Ez hatalmas előny, és az új funkcionalitás implementációját rendkívüli módon leegyszerűsíti. Az implementált rendszer jelenlegi verziója támogatja a blokkok és az identifikációhoz használt minimalizáló algoritmus tetszőleges cseréjét.

A blokkos rendszer identifikációja hagy kívánnivalókat maga után. A módszer szimulált környezetben remekül működött, azonban valós mérési elrendezésben nem szerepelt jól. További, ezen dolgozaton túlmutató feladat lehet egy mélyreható kutatás, ami felfedi a problémák hátterében meghúzódó jelenségeket.

[3] A módszer lényege, hogy csupán az új modult tesszük a régi modul helyére - a példányosító részben az új modul konstruktorát hívjuk - és a rendszer működőképes lesz az új funkcionalításokkal

Irodalomjegyzék

[1] The cone loudspeaker. http://upload.wikimedia.org/wikipedia/commons/e/e7/Collapsible_phonograph_horn.jpg.

[2] Diaphrag miniature speaker. http://3.imimg.com/data3/JT/LI/MY-136449/diaphragms-250x250.jpg.

[3] The horn loudspeaker. http://upload.wikimedia.org/wikipedia/en/d/d0/Altec1978multicell.jpg.

[4] Midrange construction. http://upload.wikimedia.org/wikipedia/commons/4/41/Midrange-speaker.png.

[5] Midrange construction. http://www.matlabtips.com/structure-your-cells/.

[6] Picture of edison's phonograph. http://media.liveauctiongroup.net/i/8339/10334277_1.jpg?v=8CDA86901F0D1E0.

[7] Speaker construction. http://upload.wikimedia.org/wikipedia/commons/7/79/Loudspeaker-bass.png.

[8] Tweeter construction. http://upload.wikimedia.org/wikipedia/commons/6/60/Tweeter.png.

[9] C. Antweiler, A. Telle, P. Vary, and G. Enzner. Perfect-sweep nlms for time-variant acoustic system identification. In *Proc. of the IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP)*, pages 517–520, Kyoto, Japan, March 2012.

[10] W. Apel. *The notation of polyphonic music, 900-1600.* Publication (Mediaeval Academy of America). The Mediaeval academy of America, 1942.

[11] Enrico Armelloni, Alberto Bellini, and Angelo Farina. Not-linear convolution: A new approach for the auralization of distorting systems. In *Proc. of the Audio Engineering Society Convention 110, Preprint No. 5359*, Amsterdam, The Netherlands, May 2001.

[12] Balázs Bank. Computationally efficient nonlinear chebyshev models using common-pole parallel filters with the application to loudspeaker modeling. In *Proc. of the*

Audio Engineering Society Convention 130, Preprint No. 8426, West London, UK, May 2011.

[13] Albert Benveniste, Michel Metivier, and Pierre Priouret. *Adaptive Algorithms and Stochastic Approximations*. Springer Publishing Company, Incorporated, 1st edition, 2012.

[14] S.A. Billings. Identification of nonlinear systems:a survey. *Control Theory and Applications, IEE Proceedings D*, 127(6):272–285, November 1980.

[15] Stephen A Billings. Model structure detection and parameter estimation. In *J. Wiley Online Library*, pages 61–104, Aug 2013.

[16] Stephen A Billings. *Models for Linear and Nonlinear Systems*. John Wiley and Sons, Ltd, 2013.

[17] J. Borwick. *Loudspeaker and Headphone Handbook*. Focal Press, 2001.

[18] Bengt Carlsson. Recursive identification. In *Lecture notes of Uppsala Universitet Institutionen for Informationsteknologi*, pages 1–11, Feb 2011.

[19] E.H.I.I. Charles. Loudspeaker waveguide design, May 9 2000. US Patent 6,059,069.

[20] Thomas S. Christensen, editor. *The Cambridge History of Western Music Theory*. Cambridge University Press, Cambridge, 2002.

[21] M. Coleman. *Playback: From the Victrola to MP3, 100 Years of Music, Machines, and Money*. Da Capo Press, Incorporated, 2009.

[22] T.M. Culda, V. Popa, D. Stanomir, and C. Negrescu. Reducing time in acoustic impulse response measurements using exponential sine sweeps. In *Proc. of the International Symposium on Signals Circuits and Systems (ISSCS)*, pages 1–4, Iasi, Romania, July 2013.

[23] P.S.R. Diniz. *Adaptive Filtering: Algorithms and Practical Implementation*. The Kluwer international series in engineering and computer science. Kluwer Academic Publishers, 2002.

[24] John M. Eargle. *Horn Systems*. Loudspeaker Handbook Springer, 2003.

[25] Engadget. *Kyocera piezoelectric film speaker delivers 180-degree sound to thin TVs and tablets*. engadget.com, 2013.

[26] Angelo Farina. Advancements in impulse response measurements by sine sweeps. In *Proc. of the Audio Engineering Society Convention 122, Preprint No. 7121*, Vienna, May 2007.

[27] Angelo Farina. Contrasting log sine sweep method and mls for room acoustics measurements. In *Proc. of the Presentation slides of the Industrial Engineering Dept., University of Parma*, Via delle Scienze 181/A Parma, 43100 Italy, Sep 2008.

[28] Wettl Ferenc. *Felsőbb matematika - Lineáris algebra jegyzet.* BME, 2013.

[29] F.W. Gaisberg. *The music goes round.* Opera biographies. Arno Press, 1977.

[30] R. Gelatt. *The Fabulous Phonograph: The Story of the Gramophone from Tin Foil to High Fidelity.* Cassell, 1956.

[31] P. Gronow and I. Saunio. *International History of the Recording Industry.* Literature & the Arts. Bloomsbury Academic, 1999.

[32] Pekka Gronow. The record industry: the growth of a mass medium. *Popular Music*, 3:53–75, 1 1983.

[33] R. Haber and L. Keviczky. *Nonlinear System Identification - Input-Output Modeling Approach, Volume 1: Nonlinear System Parameter Identification.* Mathematical Modelling: Theory and Applications. Springer Netherlands, 1999.

[34] Monson H. Hayes. *Recursive Least Squares.* Wiley. p. 541, 1996.

[35] S. Haykin. *Adaptive Filter Theory.* Pearson Education, Limited, 2013.

[36] S. Haykin and B. Widrow. *Least-Mean-Square Adaptive Filters.* Adaptive and Cognitive Dynamic Systems: Signal Processing, Learning, Communications and Control. Wiley, 2003.

[37] J.P. Joule. *On the Effects of Magnetism upon the Dimensions of Iron and Steel Bars.* The London Edinburgh and Dublin philosophical magazine and journal of science 30. Third Series 76-87. 225-241, 1847.

[38] A.D. Kilmer and M. Duchesne-Guillemin. *The Strings of Musical Instruments: Their Names, Numbers, and Significance.* 1975.

[39] Anne Draffkorn Kilmer. *Old Babylonian Musical Instructions Relating to Hymnody.* Journal of Cuneiform Studies, 1986.

[40] Wolfgang Klippel. Tutorial: Loudspeaker nonlinearities-causes, parameters, symptoms. *Journal of the Audio Engineering Society*, 54(10):907–939, 2006.

[41] J.C. Mason. *Chebyshev polynomials.* CRC Pr I Llc, 2003.

[42] A. Millard. *America on Record: A History of Recorded Sound.* Cambridge University Press, 2005.

[43] G. Milner. *Perfecting Sound Forever: An Aural History of Recorded Music.* Faber & Faber, 2009.

[44] F.W. Mostert and L.E. Apolzon. *From Edison to IPod: Protect Your Ideas and Make Money.* DK, 2007.

[45] H.H. Mueller. Automatic interlocking dual phonograph record player, January 3 1956. US Patent 2,729,455.

[46] O. Nelles. *Nonlinear System Identification: From Classical Approaches to Neural Networks and Fuzzy Models*. Engineering online library. Springer, 2001.

[47] A. Novak. *Identification of Nonlinear Systems in Acoustics*. PhD thesis, Université du Maine, Le Mans, France and Czech Technical University, Prague, Czech Republic, 2009.

[48] A. Novak, L. Simon, , P Lotton, and J. Gilbert. Chebyshev model and synchronized swept sine method in nonlinear audio effect modeling. In *Proc. of the 13th Int. Conference on Digital Audio Effects (DAFx-10)*, pages 423–426, Graz, Austria, Sep 2010.

[49] Antonin Novak, Laurent Simon, Frantisek Kadlec, and Pierrick Lotton. Nonlinear system identification using exponential swept-sine signal. *Instrumentation and Measurement, IEEE Transactions on*, 59(8):2220–2229, 2010.

[50] Antonin Novak, Laurent Simon, Pierrick Lotton, and Frantisek Kadlec. Modeling of nonlinear audio systems using swept-sine signals: Application to audio effects. In *Proc. of the 12th Int. Conference on Digital Audio Effects (DAFx-09)*, pages 1–4, Politecnico di Milano, Como, Italy, 2009.

[51] Kristiaan Pelckmans. *Lecture Notes for a course on System Identification*. Division of Systems and Control Department of Information Technology Uppsala University, 2012.

[52] O. Read. *The recording and reproduction of sound*. H.W. Sams, 1952.

[53] O. Read and W.L. Welch. *From tin foil to stereo: evolution of the phonograph*. H.W. Sams, 1959.

[54] W.J. Rugh. *Nonlinear system theory: the Volterra/Wiener approach*. Johns Hopkins series in information sciences and systems. Johns Hopkins University Press, 1981.

[55] M. Schetzen. *The Volterra and Wiener theories of nonlinear systems*. A Wiley - Interscience publication. Wiley, 1980.

[56] A. Schneider. *Musik, Sound, Sprache, Schrift: Transkription und Notation in der vergleichenden Musikwissenschaft und Musikethnologie*. Zeitschrift für Semiotik. 1987.

[57] Jonas Sjöberg, Qinghua Zhang, Lennart Ljung, Albert Benveniste, Bernard Delyon, Pierre-Yves Glorennec, Hakan Hjalmarsson, and Anatoli Juditsky. Nonlinear black-box modeling in system identification: a unified overview. *Automatica*, 31(12):1691 – 1724, 1995.

[58] J.A. Sotorrío. *Bilinear Music Notation: A New Notation System for the Modern Musician*. Spectral Music, 2007.

[59] Guy-Bart Stan, Jean-Jacques Embrechts, and Dominique Archambeau. Comparison of different impulse response measurement techniques. *Journal of the Audio Engineering Society*, 50(4):249–262, 2002.

[60] B.M. Starobin. Bass-reflex loudspeaker, December 9 1997. US Patent 5,696,357.

[61] Nicolas-Alexander Tatlas, Fotios Kontomichos, and John Mourjopoulos. Design and performance of a sigma-delta digital loudspeaker array prototype. *Journal of the Audio Engineering Society*, 57(1/2):38–45, 2009.

[62] Nicolas-Alexander Tatlas and John N. Mourjopoulos. Digital loudspeaker arrays driven by 1-bit signals. In *Proc. of the Audio Engineering Society Convention 116, Preprint No. 6036*, Berlin, Germany, May 2004.

[63] I. Todoruk. *Century of Sound: 100 Years of Recorded Sound 1877-1977.* Studio 123, 1977.

[64] P. Tortiglione. *Semiography and Semiology of Contemporary Music: Aspects and Analysis of Musical Notation : a Practical Guide to Realisation and Interpretation.* Rugginenti, 2012.

[65] V. Volterra. *Theory of Functionals and of Integral and Integro-differential Equations.* Dover Books on Mathematics Series. Dover Publications, 2005.

[66] J. Voros. Modeling and parameter identification of systems with multisegment piecewise-linear characteristics. *IEEE Transactions on Automatic Control*, 47(1):184–188, Jan 2002.

[67] J. Voros. Modeling and identification of wiener systems with two-segment nonlinearities. *IEEE Transactions on Control Systems Technology*, 11(2):253–257, Mar 2003.

[68] J. Voros. Recursive identification of hammerstein systems with discontinuous nonlinearities containing dead-zones. *IEEE Trans. Automat. Contr.*, 48(12):2203–2206, 2003.

[69] J. Voros. On identification of nonlinear dynamic systems using three-block cascade models. In *Proc. of the 19th International Conference on Process Control (PC)*, pages 247–251, Bratislava, Slovakia, June 2013.

[70] Jozef Voros. Recursive identification of hammerstein systems with polynomial nonlinearities. *Journal of Electrical Engineering, Bratislava*, 57(1):42, 2006.

[71] Feng Wang, Keyi Xing, Xiaoping Xu, Huixia Liu, and Xiaojing Sun. Research on identification algorithm of hammerstein model. In *Proc. of the IEEE Fifth International Conference on Bio-Inspired Computing: Theories and Applications (BIC-TA)*, pages 80–85, Changsha, Sept 2010.

[72] B. Widrow and S.D. Stearns. *Adaptive Signal Processing.* Edited by Alan V. Oppenheim. Prentice-Hall, 1985.

[73] C.F.A. Williams. *The Story of Notation*. Music story series, ed. by F. J. Crowest. Walter Scott publishing Company, 1903.

[74] Xiaoping Xu, Fucai Qian, Ding Liu, and Guangjun Liu. Study on identification algorithm of a class of nonlinear model. In *Proc. of the 3rd International Congress on Image and Signal Processing (CISP)*, volume 8, pages 3664–3668, Yantai, China, Oct 2010.

[75] Susana. Zapke, Anscario M. Mundó, and BBVA (Firm). *Hispania Vetus : musical-liturgical manuscripts from Visigothic origins to the Franco-Roman transition (9th-12th centuries) / Susana Zapke, ed. ; foreword, Anscari M. Mundó*. Fundación BBVA Bilbao, 2007.

Függelék

F.1. Implementált blokkok

A rendszer rugalmasságát adó architektúra lehetőséget biztosít arra, hogy az Olvasó tetszőleges blokkokat implementáljon a méréseihez. Ehhez nem kell mást tegyen, mint létrehozni egy új objektumot, ami a `Block` osztálból származik le, majd implementálnia kell a leszármaztatás által ráhagyott absztrakt implementálási felelősségeket. Nem kell attól tartani, hogy kifelejtünk egy metódust megvalósítani, hiszen a MATLAB nem engedi futni a rendszert a teljesen definiált állapotig.

Egy új blokk létrehozása öt fő részből áll. Az első lépés kivételével minden további lépést két esetre kell implementálni: normál módra és key term separation módra.

- **Inicializálás** Itt definiáljuk a blokkunk belső változóit és a blokk tényleges létrehozását. Felvehetünk tetszőleges számú belső változót, és fel is inicializálhatjuk őket a konstruktoron keresztül.
- **Szükséges paraméterszám megadása** A rendszer indulásakor elkéri minden blokktól az általa felhasznált paraméterek számát. Mint az elkövetkező minden lépést, ezt is két külön esetben kell megtenni: normál módban és key term separation módban.[4]
- **Adatok inicializálása** Ebben a metódusban a blokk megkapja a rendszer bementét és a kívánt kimeneti adatsort, amit a blokk felépítésétől és az adott üzemmódtól függően vagy elment, vagy a méretének megfelelően tárhelyet allokál későbbi adatok elmentésére.
- **Identifikációs iteráció** Identifikáció alatt minden iterációs lépésnél meghívódik sorrendben minden blokkban ez a metódus. Ezen keresztül az identifikáló rendszer elkéri a blokkhoz tartozó regressziós vektort és a blokk aktuális kimenetét. Key term separation módban értelemszerűen megkapja az előző blokk kimenetét is paraméterként.
- **Futtatás** Identifikáció után a blokkos rendszer futtatásra kész. Azaz adható neki egy gerjesztőjel, amit az identifikációval meghatározott paraméterekkel feldolgoz.

[4]Erre azért van szükség, mert a paraméterszám eltérő lesz a két üzemmódban, hiszen a key term separation mód feltételezi, hogy egy blokk egy paraméterét fixre választottuk. Hasonlóan lesz ez a inicializáló, iterációs és futtató metódussal is. A két üzemmódban máshogy kell viselkednie a blokknak, ezért külön metóduson keresztül kéri el a rendszer a viselkedésnek megfelelő adatokat.

A `Block` osztálból leszármazva, a következő implementálási mintát tudjuk felhasználni és kitölteni:

F.1.1. kódrészlet. *Saját blokk létrehozási útmutatója*

```
classdef MyBlock < Block

    % Elerheto orokolt parameterek az ososztalytol
    %  name          - blokk neve, az ososztaly tarolja
    %  identificator - az identifikalo rendszer identifikatora. nem feltetlenul
    %                  erheto el az objektum eletenek az elejen

    properties
        % sajat paramaterek definialasak a helye
    end

    methods
        % Blokkot inicializáló konstruktor, az ososztalyan valo nevatadas
        % nem hagyhato el
        function self = IIR_Block(
            name % kotelezo parameter
            % tovabbi parameterek felvehetoek amennyiben szukseg van rajuk.
            % Altalanosan plusz parameter lehet a blokk parameterszamanak
            % az atadasa, de barmilyen mas adat is atadhato itt.
        )
            self = self@Block(name);
            % sajat parameterek inicializalasa a kapott parameterek alapjan
        end

        %=========================================================================
        %% Identifikacio : Normal Mode

        function n = tellNumberOfParametersForNormalMode(
            self % kotelezo parameter, mivel a metodus nem statikus
        )
            % ezen a metoduson keresztul keri al az identifikalo rendszer a
            % normal modhoz tartozo paramaterszamot a visszateresi erteken
            % keresztul
        end

        function initRegressorVectorForNormalMode(
            self, % kotelezo parameter, mivel a metodus nem statikus
            inputVector, % ismeretlen rendszerhez hasznalt gerjesztojel
            desiredOutputVector % ismeretlen rendszer valasza a gerjesztojelre
        )
            % Ez a metodus az bemeneti es kimeneti adatok hozzadasakor kerul
            % meghivasra normal modban. Itt lementhetjuk a blokk szamara
            % szukseges adatsorokat, vagy allokalhatunk tarolo vektorokat
        end

        function [regressor, blockOutput] = calculateRegressorForNormalMode(
            self, % kotelezo parameter, mivel a metodus nem statikus
            parameters, % blokk parametervektora
            currentSystemInput, % az eppen aktualis rendszer bemeneti mintaja
            k % identifikacios lepesszam
        )
            % Ez a metodus minden identifikacios iteracioban meghivodik normal
            % modban. Parameterkent megkapja a parametervektorat, aminek a
            % hossza az elore elkert parameterszamnak megfelel. A parameterek
            % pozicioja tetszoleges mindaddig, amig a generalt regresszorvektorban
```

```
56              % a parameterhez tartozo ertek a megfelelo helyen van.
57              % A parameterlistan kivul megkapjuk meg a rendszer aktualas
58              % bemeneti mintajat, arra az esetre, ha nem lenne lementve
59              % az adatok inicializasakor, valamint az aktualis iteracios
60              % lepesszamot is, hogy indexelni tudjuk a lementett adatvektorokat.
61              %
62              % A metodus celja, hogy visszaadjuk a blokkunkhoz tartozo
63              % regresszorvektort es a blokk kimentetet noemal modban.
64          end
65
66          %========================================================================
67          %% Identifikacio : Key Term Separation Mode
68          function n = tellNumberOfParametersForKeyTermSeparationMode(
69              self % kotelezo parameter, mivel a metodus nem statikus
70          )
71              % Hasonloan, mint normal modban, key term separation modban is
72              % elkerjuk a a blokk parameterszamat
73          end
74
75          function initRegressorVectorForKeyTermSeparationMode(
76              self, % kotelezo parameter, mivel a metodus nem statikus
77              inputVector, % ismeretlen rendszerhez hasznalt gerjesztojel
78              desiredOutputVector % ismeretlen rendszer valasza a gerjesztojelre
79          )
80              % Key term separation modhoz tartozo inicializalo metodus.
81              % Hasonloan a normal modban, ez is az adatok atadasakor hivodik meg
82              % a blokk bekotesenek fuggvenyeben. Letarolhatjuk a bemeneti es
83              % kimeneti adatokat, vagy tarhelyeket allokalhatunk kesobbi
84              % hasznalatra
85          end
86
87          function [regressor, blockOutput] =
        calculateRegressorForKeyTermSeparationMode(
88              self, % kotelezo parameter, mivel a metodus nem statikus
89              parameters, % blokk parametervektora
90              currentSystemInput, % az eppen aktualis rendszer bemeneti mintaja
91              lastBlockOutput, % az elozo blokk kimeneti mintaja
92              k % identifikacios lepesszam
93          )
94              % Identifikacio soran hivodik meg minden iteracioban. A normal
95              % modhoz kepest megkapja meg az elozo blokk kimeneti mintajat is
96              % A metodus calja itt is az, hogy elkeszitsuk a regresszorvektort es
97              % a blokk kimenetet.
98
99          end
100
101         %========================================================================
102         %% Running : Normal Mode
103         function output = processSignalForNormalMode(
104             self, % kotelezo parameter, mivel a metodus nem statikus
105             input, % blokk bemenete
106             parameters % blokk parameterei
107         )
108             % Rendszer futtatasa normal modban. Cel, hogy a bemenet
109             % es a kapott parameterlista alapjan generaljuk a kimeneti vektort.
110         end
111
112         %========================================================================
113         %% Running : Key Term Separation Mode
114         function output = processSignalForKeyTermSeparationMode(
```

```
115             self, % kotelezo parameter, mivel a metodus nem statikus
116             input, % blokk bemenete
117             parameters % blokk parameterei
118         )
119             % Rendszer futtatasa key term separation modban. Cel, hogy a bemenet
120             % es a kapott parameterlista alapjan generaljuk a kimeneti vektort.
121         end
122
123     end
124
125 end
```

A fenti minta egyertelmű támpontot biztosít tetszőleges blokkok implementálására. A továbbiakban bemutatom a FIR, IIR és a hatványsoros statikus nemlinearitást megvalósító blokkok forráskódját, hogy az esetleges kérdéseket feloldja az új blokkok készítésével kapcsolatban.

F.1.1. FIR

F.1.2. kódrészlet. *FIR blokk implementációja*

```
1  classdef FIR_Block < Block
2      properties
3          nb;
4          u_pre;
5          x_pre;
6      end
7
8      methods
9          function self = FIR_Block(name,nb)
10             self = self@Block(name);
11             self.nb = nb;
12         end
13
14         function n = tellNumberOfParametersForNormalMode(self)
15             n = self.nb+1;
16         end
17
18         function n = tellNumberOfParametersForKeyTermSeparationMode(self)
19             n = self.nb;
20         end
21
22         function initRegressorVectorForNormalMode(self,inputVector,
       desiredOutputVector)
23             self.u_pre = [zeros(self.nb+1,1);inputVector];
24         end
25
26         function initRegressorVectorForKeyTermSeparationMode(self,inputVector,
       desiredOutputVector)
27             self.x_pre = zeros(self.nb + length(inputVector),1);
28         end
29
30
31
32         function [regressor, blockOutput] = calculateRegressorForNormalMode(self,
       parameters,currentSystemInput,step)
33             regressor = self.u_pre(step+self.nb+1:-1:step+1);
```

```
34              blockOutput = regressor'*parameters;
35          end
36
37          function [regressor, blockOutput] =
        calculateRegressorForKeyTermSeparationMode(self,parameters,currentSystemInput,
        lastBlockOutput,step)
38              self.x_pre(step+self.nb) = lastBlockOutput;
39              regressor = self.x_pre(step+self.nb-1:-1:step);
40              blockOutput = regressor'*parameters;
41          end
42
43          function output = processSignalForNormalMode(self,input,parameters)
44              output = filter(parameters,1,input);
45          end
46
47          function output = processSignalForKeyTermSeparationMode(self,input,parameters
        )
48              output = filter([1; parameters],1,input);
49          end
50      end
51  end
```

F.1.2. IIR

F.1.3. kódrészlet. *IIR blokk implementációja*

```
1   classdef IIR_Block < Block
2       properties
3           nb;
4           na;
5           y_pre;
6           u_pre;
7           x_pre;
8       end
9
10
11      % the used model
12      %
13      % y(t) + a_1*y(t-1) +...+ a_na*y(t-n_a) = b_0*u(t) +...+ b_nb*u(t-nb)
14      %
15      % where
16      %   A(z) = 1   + a_1*z^(-1) + ... + a_na*z^(-na)
17      %   B(z) = b_0 + b_1*z^(-1) + ... + b_nb*z^(-nb)
18      %
19      % The given parameters describes the maximum order of denominator (nb)
20      % and nominator (na).
21      %
22      % In normal mode the parametervector contains the following:
23      %
24      %   [
25
26      methods
27          % Initialization
28          function self = IIR_Block(name,nb,na)
29              self = self@Block(name);
30              self.nb = nb;
31              self.na = na;
32          end
```

```
33
34          %% Identification : Normal Mode
35
36          function n = tellNumberOfParametersForNormalMode(self)
37              n = self.nb+1 + self.na;
38          end
39
40          function initRegressorVectorForNormalMode(self,inputVector,
        desiredOutputVector)
41              self.u_pre = [zeros(self.nb+1,1);inputVector];
42              self.y_pre = [zeros(self.na,1);desiredOutputVector];
43          end
44
45          function [regressor, blockOutput] = calculateRegressorForNormalMode(self,
        parameters,currentSystemInput,k)
46              regressor = [self.u_pre(k+self.nb+1:-1:k+1); -self.y_pre(k+self.na-1:-1:k
        )];
47
48
49              b = parameters(1:self.nb+1);
50              b = [b];
51              a = parameters(self.nb+2:end);
52
53              pp = [b; a];
54              r = [self.u_pre(k+self.nb+1:-1:k+1); -self.y_pre(k+self.na-1:-1:k)];
55
56              blockOutput = r'*pp;
57          end
58
59          %% Identification : Key Term Separation Mode
60          function n = tellNumberOfParametersForKeyTermSeparationMode(self)
61              n = self.nb + self.na;
62          end
63
64          function initRegressorVectorForKeyTermSeparationMode(self,inputVector,
        desiredOutputVector)
65              self.u_pre = zeros(self.nb + length(inputVector),1);
66              if self.isLast()
67                  self.y_pre = [zeros(self.na,1);desiredOutputVector];
68              else
69                  self.y_pre = [zeros(self.na,1);zeros(size(desiredOutputVector))];
70              end
71          end
72
73          function [regressor, blockOutput] =
        calculateRegressorForKeyTermSeparationMode(self,parameters,currentSystemInput,
        lastBlockOutput,k)
74              self.u_pre(k+self.nb) = lastBlockOutput;
75              regressor = [self.u_pre(k+self.nb-1:-1:k); -self.y_pre(k+self.na-1:-1:k)
        ];
76
77
78              b = parameters(1:self.nb);
79              b = [1; b];
80              a = parameters(self.nb+1:end);
81
82              pp = [b; a];
83              r = [self.u_pre(k+self.nb:-1:k); -self.y_pre(k+self.na-1:-1:k)];
84
85
```

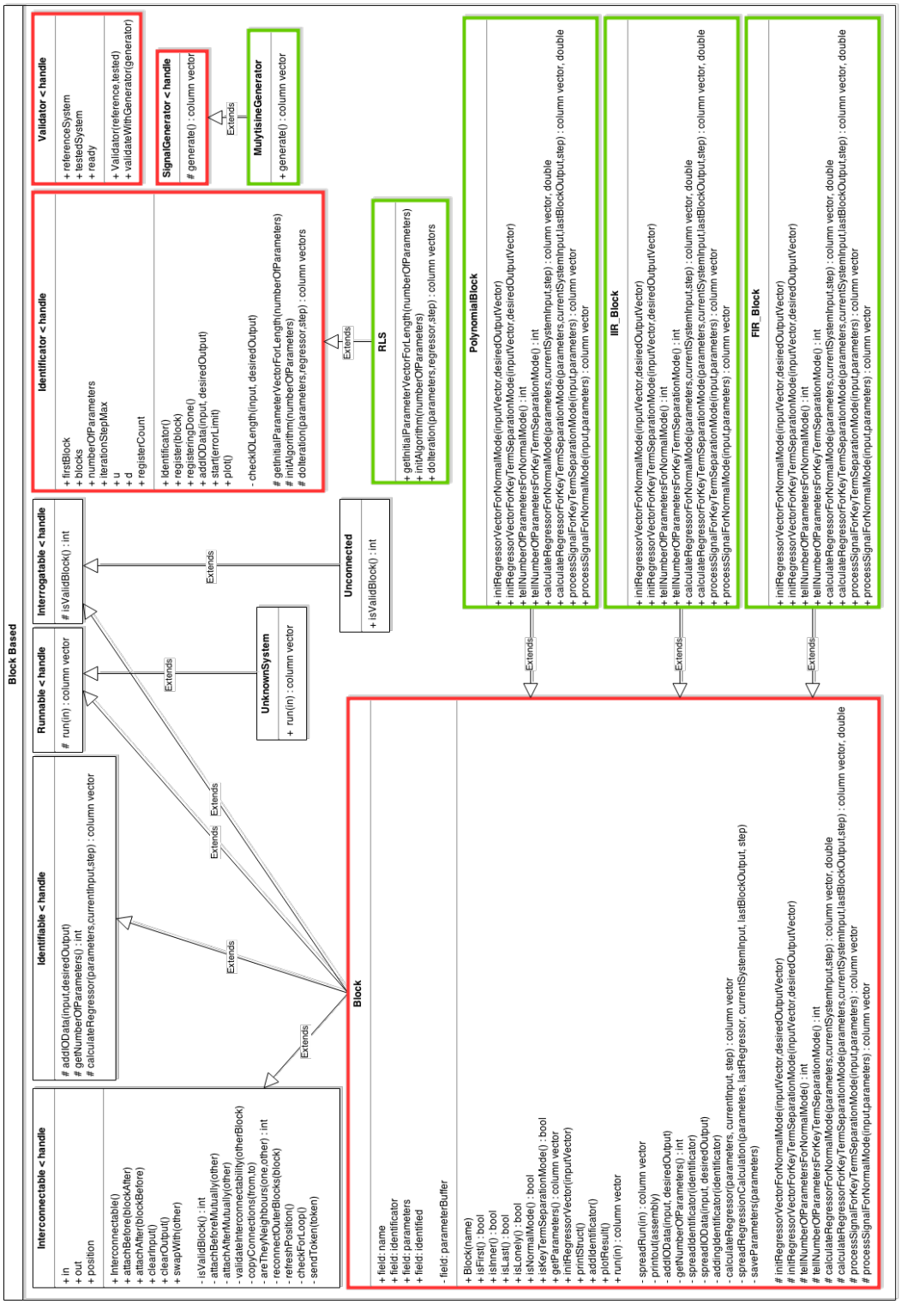
```
86
87              blockOutput = r'*pp;
88              if ~self.isLast()
89                  self.y_pre(k+self.na-1) = blockOutput;
90              end
91          end
92
93          %% Running : Normal Mode
94
95          function output = processSignalForNormalMode(self,input,parameters)
96              output = filter(parameters,1,input);
97          end
98
99          %% Running : Key Term Separation Mode
100
101         function output = processSignalForKeyTermSeparationMode(self,input,parameters
        )
102             b = parameters(1:self.nb);
103             b = [1; b];
104             a = parameters(self.nb+1:end);
105             a = [1; a];
106             output = filter(b,a,input);
107         end
108
109     end
110
111 end
```

F.1.3. Polynomial Block

F.1.4. kódrészlet. *Hatványsoros polinomiális blokk implementációja*

```
1  classdef PolynomialBlock < Block
2      % available read only properties from the superclass
3      %  name          - the name of the block
4      %  identificator - the identificator object for the block
5      %  parameters    - the parameters of the block
6
7      % editable parameters
8      properties
9          nf;
10     end
11
12     methods
13         function self = PolynomialBlock(name,nf)
14             self = self@Block(name);
15             self.nf = nf;
16         end
17
18         function n = tellNumberOfParametersForNormalMode(self)
19             n = self.nf;
20         end
21
22         function n = tellNumberOfParametersForKeyTermSeparationMode(self)
23             n = self.nf-1;
24         end
25
```

```
26          function initRegressorVectorForNormalMode(self,inputVector,
        desiredOutputVector)
27              % nothing to do here
28          end
29
30          function initRegressorVectorForKeyTermSeparationMode(self,inputVector,
        desiredOutputVector)
31              % nothing to do here
32          end
33
34
35
36          function [regressor, blockOutput] = calculateRegressorForNormalMode(self,
        parameters,currentSystemInput,step)
37              regressor = zeros(self.nf,1);
38              for kk = 1:self.nf
39                  regressor(kk) = currentSystemInput^kk;
40              end
41              blockOutput = regressor'*parameters;
42          end
43
44          function [regressor, blockOutput] =
        calculateRegressorForKeyTermSeparationMode(self,parameters,currentSystemInput,
        lastBlockOutput,step)
45              regressor = zeros(self.nf-1,1);
46              for kk = 2:self.nf
47                  regressor(kk-1) = lastBlockOutput^kk;
48              end
49              blockOutput = regressor'*parameters;
50          end
51
52          function output = processSignalForNormalMode(self,input,parameters)
53              output= zeros(size(input));
54              for kk = 1:self.nf
55                  output = output + (input.^(kk)).*parameters(kk);
56              end
57          end
58
59          function output = processSignalForKeyTermSeparationMode(self,input,parameters
        )
60              output = input;
61              for kk = 1:self.nf-1
62                  output = output + (input.^(kk+1)).*parameters(kk);
63              end
64          end
65
66      end
67
68  end
```



F.2.1. ábra. *Iteratív blokkos identifikáló rendszer blokkvázlata*

F.2. Iteratív blokkos identifikáló rendszer blokkvázlata

F.3. Hangok reprodukciója

F.3.1. A kezdetek

A hangok reprodukálásának az igénye az emberiség kezdetére datálható [39, 56]. Ekkor még nem beszélhetünk rögzítésről, inkább csak zörejek, kezdetleges hangok, majd később

dallamok tapasztalat utáni elsajátításról. Az első ténylegesen rögzített dallamok i.e. 2000 körülre tehetőek. Ezek kezdetleges kották voltak [20], amik egyedi rögzítési rendszerben írták le a művet.

Az ókori görögöktől kezdve nevezhetjük kottának a megörökített és fennmaradt írásokat, azonban az egész világon kezdett megjelenni ez az egységesítési törekvés.

Ekkor még nem volt szó hangrögzítésről és reprodukcióról a ma ismert formában. A legelső rögzített hangok zeneművek voltak, amelyeket lekottáztak az éppen ismert módon [73, 75]. Később kialakult az egységes kottázási rendszer, és a világon mindenki számára elérhetővé váltak a darabok [10].

Ezzel a módszerrel elérhetővé vált, hogy a szerzők műveit világszerte megismerhesse bárki, aki ismeri a reprodukció módját, a kottaolvaását [58, 64]. Hasonlóan az íráshoz, ez a módszer a mai napig közismert, az egyik alapvető módja a zene rögzítésének. A módszer előnye, hogy szinte bárki számára elérhető, a visszaadás minősége kiváló.

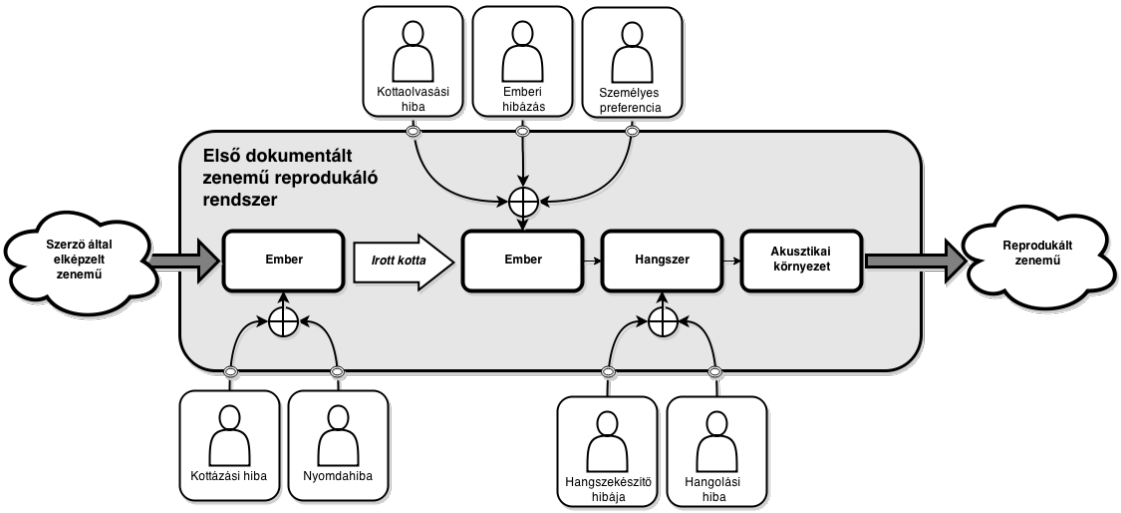
Egyedüli hátránya, hogy az eredeti zene visszaadása gyakorlatilag lehetetlen volt. A kottát emberek írták és emberek olvasták. A mérnöki gyakorlatban nem hiába próbálják kizárni az emberi beavatkozást amennyire csak lehet, hiszen az ember nem egy gépezet, amit tökéletes és ismétlődő feladat végrahajtásra terveztek. Az emberi visszajátszás eredményeként soha nem játszottak el zenedarabot teljesen azonos módon, ahogy azt a szerzője megálmodta.

A műkedvelő emberek a zeneművek címe mellett elkezdték a kedvenc előadójukat is számon tartani, azt, aki számukra a legtetszetősebben adta elő a darabot. Sokszor a beavatatlan fülek számára teljesen egyforma előadások között is jelentős eltérések lehetnek. Ha az előadó egy hangot is kicsit hosszabban tart zengésben, mint ahogy a kottában le van írva, a művet jól ismerő műkedvelő számára egyértelműen kitűnik a minimális eltérés. Ne is beszéljünk a tipikus emberi hibának, a tévesztésnek a jelenlétéről. Az előadó elüt egy hangot, a kottázó félreírt egy hangot, elírt egy ütemet. A lehetséges hibázások a tárháza nagyon széles, és nem áll meg a zenemű feljegyzésénél.

Az eredeti elképzelés egy bizonyos hangszerre, hangszercsoportra lett komponálva, amikhez a zenemű írója hozzáfért [38]. A hangszereket emberek készítik, hangolják, és ugyan a gyártási technológia kézműves keretek között is tökélyre fejleszthető, még sincs a világon két egyforma kézzel készített hangszer. Egyszerűen annyi paramétertől függ egy hangszer karakterisztikája, hogy képtelenség mindenben egyező modellt gyártani. Ha csak az alapanyagokból indulunk ki, egy szükséges, de nem elégséges feltétel lenne az, hogy az alapanyag, amiből a hangszer készül, molekuláris szinten megegyezzen, ami - lássuk be - képtelenség. A faanyag rostjai, a fémalkatrészek doménei mind egyedi hangzást kölcsönöznek a hangszernek.

Az adott hangszer megléte után egy újabb emberi változóba ütközünk, ami a hangszer hangolása. Nem feltétlenül lesz ugyanaz a hangmagasság frekvenciára pontosan akkor, ha ember hangolja be a hangszereket. Bár léteznek etalon hangforrások, ezek nem tudják behangolni a hangszert, A hangolást ember végzi, tehát hibázhat.

Az emberi tényezőkön kívül szerepet játszik még környezet, az akusztikai tér, amiben a művet előadják. E felett a mű szerzőjének nincs behatása. A művet előadhatják hosszú zengésű teremben (templom, színház, operaház), de zengés nélküli helyszíneken is. Minden esetben más és más lesz a mű hatása a hallgatóra. Képzeljük csak el, hogy ha a zene szerzője a műben kihasználja a hosszú lecsengéseket, akkor a művet egy kis szobában előadó művész által játszott dal a szerzőnek, és a tapasztalt hallgatóságnak is furán fog hatni.



F.3.1. ábra. *Az első dokumentált zenemű reprodukciós rendszer*

Ha a zenemű reprodukálását egy rendszernek vesszük, aminek a bemente a szerző elképzelése a zeneműről, a kimenete pedig a hallható zenemű, akkor a rendszer felvázolható olyan módon, ahogy az a F.3.1. ábrán is látható. A külső, emberi torzítások a következőek:

- Kottázási hiba - a szerző nem azt írja, mint amit gondol
- Nyomdahiba - sokszorosításnál hiba csúszik a mesterpéldányba
- Kottaolvasási hiba - az előadó elnéz egy hangot a kottán
- Emberi hibázás - az előadó ugyan a helyes hangot olvassa, azonban a hangot elvéti
- Személyes preferencia - az előadó felülbírálja a szerző elképzelését, és mást játszik, mint ami a kottában szerepel
- Hangszerkészítő hibája - alapanyagválasztásbeli és megmunkálásbeli eltérések
- Hangolási hiba - az előadó hibásan hangolja, már referenciához hangolja

Az előzőekben felsorolt emberi és nem emberi tényezőket vehetjük egyfajta torzításnak, amelyek torzítják a mű szerzője által kialakított képet és a hallgatóság által elvárt igényeket a zeneművel kapcsolatban.

A következtetés az, hogy kottából, más által játszott mű soha nem lesz oyan, mint amilyennek azt a mű szerzője megálmodta, viszont a hang minősége kifogástalan, hiszen művészek játsszák eredeti hangszereken. A zeneművek ilyen módon való rögzítése a legősibb, mégis a mai napig gyakorolt formája a művek reprodukálásának.

F.3.2. Az emberi tényező kivétele

Később, a IX. században a Banu Musa fivérek által kifejlesztett mechanikus viziorgona jelentette a következő lépést, amely az emberi tényező visszaszorítását volt hivatott megoldani. Elsődleges cél az volt, hogy a zene visszajátszható legyen többször is egymás után ugyanolyan minőségben. Ezt a fivérek egy cserélhető hangdobos szerkezettel oldották meg egy viziorgonában. A hengeren mélyedések és kiemelkedések voltak, amelyek szelepeket működtettek, és ezáltal megszólaltatták a hangszert. A fivérek ezen kívül még egy automata fuvolát is alkottak. Ennek a lényege is a cserélhető, zenei programot tartalmazó dob volt.

A technológia elterjedésével, az óraművesek egyre bonyolultabb szerkezeteket készítettek, amelyek egyre összetettebb darabokat adtak elő. Teljes szimfónikus zenekart utánzó szekrényeket építettek, óraműveket, amelyek az új órák kezdetekor előadtak egy darabot, automatikus orgonákat, amelyek maguktól kísértek istentiszteleteket, hordozható zenedobozokat, amelyek segítségével bárki, zenei tudás nélkül elmehetett utcai zenésznek

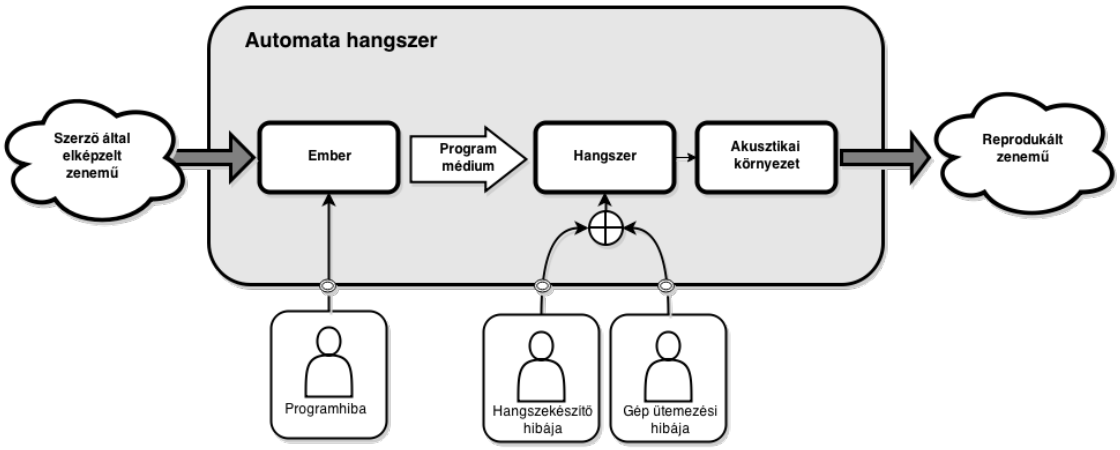
A hengeren kívül más vezérlőegységet tartalmazó szerkezeteket is alkottak. Elterjedt volt a lyukpapíros megoldás, mikor egy hosszú papírlapon perforált lyukak hosszirányú helyzete adta meg a hang hosszát, míg szélességbeli helyzete kódolta a hang magasságát.

Az ilyen elven működő szerkezetek voltak az emberiség első programozható gépei. Ugyan megoldották az emberi előadó problémáját, és bárki számára elérhetővé tették a zenét, azonban még mindig megmaradtak az elkészítésüknél fellépő hibák. Ha egy kiemelkedést nem a megfelelő helyre faragtak, egy lyukat nem a megfelelő helyre lyukasztottak, akkor a mű hangzását módosították vele.

Tipikusan ezek a szerkezetek gépiesen szóltak, azaz az emberre nem jellemző néha pontos, néha nem pontos hangindításokkal egyből felismerhető volt, hogy a zenét gép játssza, néhány kivételesen precíz gép kivételével.

Az így felmerülő hibákat ismét torzításnak felfogva, a következő megállapításokat tehetjük:

- Program hiba - a mű "programozója" elvétette a médium készítését
- Hangszerkészítő hibája - mechanikai rendellenesség, ami befolyásolja a hangszer megszólalását
- Hangolási hiba - a gép ütemezési sebessége nem konstans, tipikusan tekerős szerkezetek esetén



F.3.2. ábra. *Automata hangszer torzításai*

Az ilyen módon rögzített zeneművek hangzásban nem voltak tökéletesek, hiszen kikerült a képletből az emberi játék, ami egyrészről pozitívum, hiszen nem kell az emberi hibázással és pontatlansággal számolni. Ennek hatására a visszajátszás a gép pontosságával korlátozva ugyan de a programozó médiumnak megfelelően hibátlanul játszotta minden egyes alkalommal a zeneművet. Ennek az ára, ami ezen gépek nagy negatívuma, hogy a hangzás a legtöbb esetben gépies, és monoton. Mind a szerkezet készítője és a programozója tökéletes munkát kell végezzen, hogy a darab megszólalása emberi művészt megszégyenítő legyen. Itt jön be újra az emberi hibázás lehetősége.

A hangzást befolyásoló akusztikai tér hatása ugyanúgy megmaradt, mint a kottával rögzített zene esetében.

A megismert korai két rögzítési módszernek volt egy nagy hiányossága: csak és kizárólag kottázható zeneművek rögzítésére és visszajátszására voltak alkalmasak, ami az automata esetében további korlátozásokkal is járt: kizárólag hangszeres darabokat lehetett megszólaltatni velük. Tetszőleges hangok, zörejek és az emberi beszéd rögzítésre teljességgel alkalmatlanok voltak. Ugyan készültek az emberi anatómiát modellező gépezetek, amelyek képesek voltak emberi hangokat kiadni, és az automata gépekbe építhettek nem zenei hangokat megszólaltató egységeket, de ezek csak a saját zörejeiket tudták "eljátszani".

Amire szükség volt, az egy tetszőleges hangot rögzítő megoldás kifejlesztése, amivel hangszeres műveket, emberi hangot, éneket és tetszőleges hangot rögzíteni lehetett.

Erre nem is kellett sokat várni, ugyanis egy párizsi feltaláló, Édouard-Léon Scott de Martinville 1857-ben levédette az első tetszőleges hangot rögzítő szerkezetet az úgynevezett *phonautograph*-ot.

F.3.3. Hangszerek leváltása

Az új találmány érdeme abban rejlett, hogy leváltotta a valós hangszereket egy szerkezetre, ami képes volt tetszőleges hangot rögzíteni és kiadni.

A szerkezetet Charles Cros, francia író és humorista nevéhez köthető. 1877-ben ő adta be az első ilyen képességű szerkezet terveit a párizsi Tudományos Akadémiára [45], azonban ez első gyakorlati implementációt Thomas Edison feltaláló készítette, szintén 1877-ben, és 1878-ban le is szabadalommal le is védette [21, 29, 30].

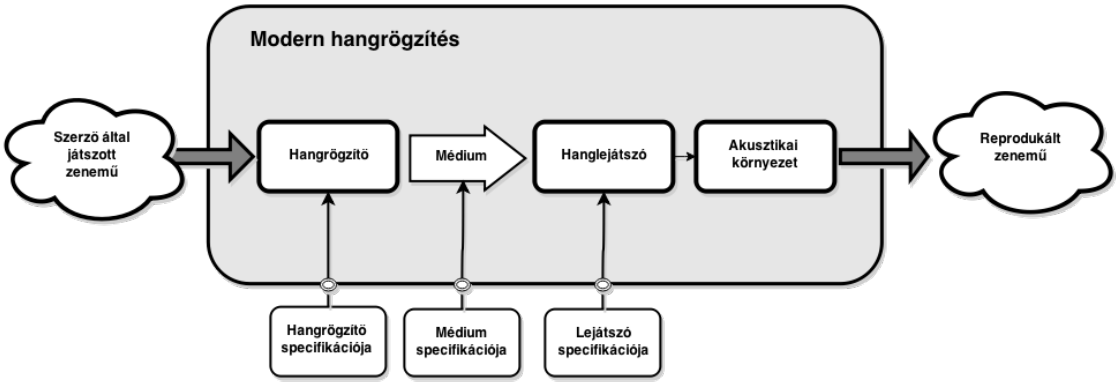
F.3.3. ábra. *Edison kereskedelmi forgalomban kapható fonográfja [6]*

Edison gépe egy membránhoz erősített tű segítségével viaszhengerre rögzítette (karcolta) a gép működtetése közben keltett hangokat. Az eljárással megszületett a ma is használt eljárás struktúrája, miszerint egy hangrögzítő eszköz a hang tárolására szolgála médiumra rögzíti a hangot, amit egy hangvisszaadó szerkezet képes visszajátsszani. A séma ma is azonos, csak az eljárás minősége javult értékrendekkel.

A találmány olyan jól sikerült, hogy tömeggyártásba került [32, 31]. Rendkívül keresett termék volt a modern országokban. Azonban terjedése a 1900-as évek elejéig tartott csak, ugyanis a lemez megjelenése teljesen kiszorított a piacról.

Az új architektúra lehetővé tette, hogy az emberi zavaró tényezőket teljes mértékben kizárjuk a láncból. A felvett hangot játszhatja maga a szerző pontosan olyan módon, ahogy megálmodta a művét. A rendszer többi részét a technológia aktuális állása szerinti torzítások befolyásolják csak.

- Hangrögzítő specifikációja
- Médium hibája
- Lejátszó specifikációja



F.3.4. ábra. *Automata hangszer torzításai*

Mindhárom tényező egy az egyben befolyásolja a felvett és visszajátszott hang minőségét. Torzításaikat a gyártási technológia és minőség befolyásolja egyedül. Amíg ezeknél biztosítható az emberi tényező kiküszöbölése, addig a hangrögzítő rendszer minőségbeli téren csak minimálisan fog függeni tőle (kezelésbeli és beállításbeli kérdések).

A modern hangrögzítő rendszer új iparágak megjelenését, és versenyét hozta el [42]. Feltalálók sorra alkották meg az aktuális technikai színvonalnak megfelelő hangrögzítő, médium és hanglejátszó eszközöket [44, 43]. A kezdeti tisztán mechanikai megoldásokat hamar leváltották az elektromos rendszerek [53, 63]. Kezdetben analóg, majd a digitális technika fejlődésével digitális módon működő hangrögzítő eljárások születtek meg.

Összegzésként elmondható, hogy a modern hangrögzítési eljárás kivette az emberi tényezőt a folyamatból annyira, amennyire csak lehetett, ezáltal elérhetővé tette a mind minőségbeli és struktúrális tökéletességet ismételt hallgatás igénye esetén is.

F.4. További hangszórótípusok

Kezdetben az emberiség a saját hangját használta hangok, dalok megszólaltatására. A hangszerek megjelenésével az ember saját képességeit meghazudtoló hangokat is képes volt produkálni, ezzel kibővült a lehetőségeinek a száma. Eleinte ütős hangszerekkel kísérték az énekeket, majd megjelentek a zenei hangot kiadni képes hangszerek, és megindult a megállíthatatlan fejlődés. Újabb és újabb típusú hangszerek láttak napvilágot. Fúvós, húros, pengetős, vonós, a lehetőségek száma szinte végtelen a a generált hanghatás tekintetében. Azonban minden hangszernek volt egy nagy hiányossága. Egy trombita nem tudott úgy szólni, mint egy hegedű, egy klarinét nem tudott gitár hangon megszólalni.

A kezdeti hangrögzítés, majd az automatizált hangszerek is csak a rendelkezésre álló hangszerek hangján voltak képesek megszólalni. Ha valaki egy szimfónikus zeneművet kívánt meghallgatni, akkor két lehetőség közül választhatott. Vagy elment egy előadásra, vagy beszerzett magának egy kisebb ház méretű és bonyolultságában szinte felfoghatatlan szerkezetet, ami bármikor lejátszott neki egy darabot. Mindkét esetben csak azok a

hangszerek hangjai szerepelhetnek a hallott műben, amilyen hangszerek megtalálhatóak a zenekarban, vagy a kusza mechanika erdejében.

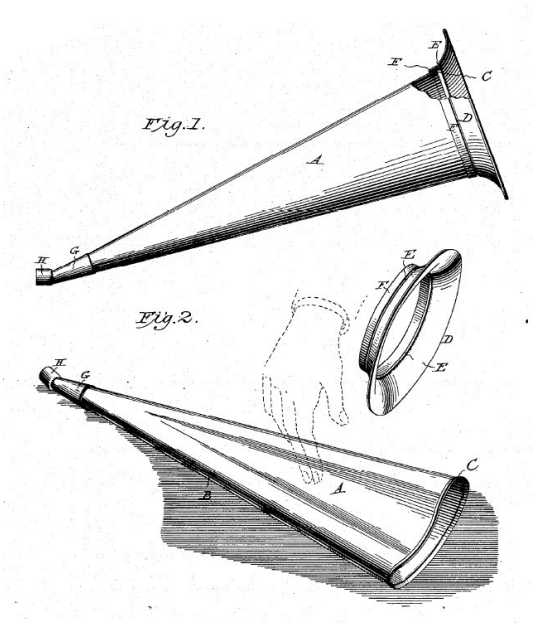
Belátható, hogy a költségekkel jelentősek. Egy mű többszöri meghallgatásáért minden egyes alkalommal fizetni kell egy előadáson. Az automatizált rendszert ugyan csak egyszer kell megvenni, utána bármikor lejátszatható vele a kívánt darab, viszont a rendszer ellenértéke szinte mindenki számára megfizethetetlen.

A modern hangrögzítés megteremtette a lehetőséget mindenki számára, hogy ésszerű költségvetés keretében bármikor élvezhesse kedvenc művét az otthonában, sőt, ennél még többet adott. Mostantól bármilyen hangot vissza lehetett játszani: beszédet, a természet hangjait, tetszőleges zörejeket, és természetesen a hangszeres zenét kiegészítette az énekes zene.

F.4.1. Hangtölcsér

A legelső eszköz a hang visszadására a fúvós hangszerek legvégső elemére hasonlító hangtölcsér volt. Ez a megoldás tisztán mechanikus volt, és a médium olvasása közben keletkezett rezgéseket erősítette fel [17]. Nagy hiányossága volt, hogy mivel tisztán mechanikus "hangszóróról" volt szó, az erősítése csekély volt. Nem volt elég, hogy betöltsön egy színháztermet.

Edison, MagnaVox és Victrola gyár mind kifejlesztett egy-egy hangtölcsértípust, amelyeknek jól teljesítettek a hangok visszaadásában. Egy ilyen hangtölcséres megoldás látható a F.3.3. ábrán.



F.4.1. ábra. *Fonográf hangtölcsér [1]*

A működése nagyon egyszerű. Jellemzően viaszhengeres vagy modernebb korongos/lemezes fonográfokban használták őket. A hangfelvevő eszköz a médium anyagában barázdákat karcolt, amelyek a felvétel közben hallható hangokat rögzítették. A hanglejátszó eszköz egy fémtű segítségével "leolvasta", a rögzített hanghullámokat egy membrán rezgéssé alakította, és ezeket a rezgéseket a hangtölcsér felerősítette.

Különféle méretű hangtölcsérek léteztek, a kisebbek rendszerint kisebb szobákba voltak használatosak. Rézből, ónból és fából és készítettek hangtölcséreket.

Az elektrodinamikus hangszórók megjelenéséig uralták a zenei piacot, azonban hamar eltűntek az új technológia megjelenésekor.

F.4.2. ábra. *Modern hangtölcsér hangosításhoz [3]*

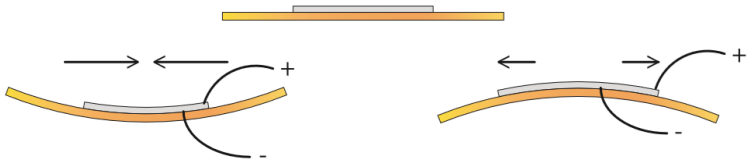
A hagyományos, tisztán mechanikus hangtölcsérek eltűnése után felütötték a fejüket elektromos gerjesztővel hajtott tölcséres kialakítású hangszórók. A hangtölcsér lényege, hogy nagy hatékonysággal erősíti tovább a gerjesztést. Elérhető vele akár 10dB-es erősítés is [60, 24, 19], a hagyományos tölcsér alakhoz képest.

Remek hatásfokuk miatt előszeretettel használják őket nyílt terek hangosítására és megafonokban. Léteznek egytölcséres és többtölcséres változatai is. Vájtfülű alkalmazásokban is előszeretettel használják őket.

F.4.2. Piezoelektromos hangszóró

A piezoelektromos hangszórók teljesen más elven működnek, mint az elektrodinamikus társaik. Az ilyen hangszórók a piezoelektromos hatást hasznosítják, ami szerint piezoelektromos tulajdonsággal rendelkező anyagokban mechanikai feszültség hatására villamos feszültség keletkezik. A jelenség visszafelé is igaz, tehát a piezoelektromos anyagokra fe-

szültséget adva, mechanikai feszültség keletkezik bennük. Mechanikai feszültség kialakulhat összenyomás hatására és hajlítás hatására. A piezoelektromos hangszórók az utóbbit használják ki. Váltakozó feszültség ráadásakor a piezoelektromos anyag rezgésbe kezd, aminek a frekvenciája megegyezik a rá adott feszültség frekvenciájával. Így váltakozó feszültséget hanghullámokká átalakító szerkezetet kapunk.



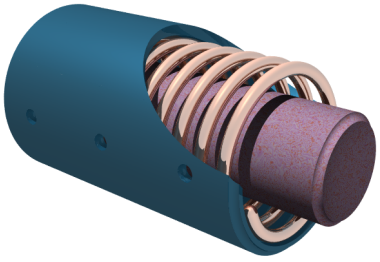
F.4.3. ábra. *Piezoelektromos hangszóró működési elve*

Ezen az elven működő hangszórók jellemzően keskeny sávtartományban használhatóak, ezért leginkább magassugárzóként használják őket [25].

F.4.3. Magnetostrikciós hangszórók

A magnetostrikció a ferromágnese anyagok egy tulajdonsága, ami abban nyilvánul meg, hogy mágnesezés hatására megváltoztatják az alakjukat, kiterjedésüket [37]. Az elektromos tér hatására a magnetostrikciós feszültség telítődésig nő bennük, miközben ez kihatással van a fizikai kiterjedésükre. A jelenség könnyebb megérthetősége érdekében képzeljük el, hogy az anyagot ovális (egyik kétdimenziós kiterjedése jelentősen nagyobb, mint a másik) dipólusok alkotják, amelyek képesek elfordulni mágneses tér hatására. Amíg a dipólusok függőlegesen állnak, addig az anyag hosszirányú kiterjedése rövidebb, mint amikor a dipólusok vízszintesen állnak.

A jelenség felhasználható hangszórók készítésére is. Azonban az ilyen elven működő hangszórók nem tartalmaznak erőt hangnyomássá átalakító elemeket, hanem a felhasználóra bízzák, hogy valamilyen felületet felhasználjon erre. Így ezt a típusú hangszórót nem is lehet igazából hangszórónak nevezni, inkább átalakítónak, ami az elektromos jeleket fizikai rezgéssé, erővé transzformálja. Megvalósítása a F.4.4. ábrán látható.



F.4.4. ábra. *Magnetostrikciós átalakító általános felépítése*

Tipikus alkalmazási forma, hogy egy lapos felületre, például asztallapra, falra, ablakra szerelik fel az átalakítókat, de léteznek olyan megoldások, ahol tetőszerkezetre vagy padlóra

erősítik fel.

Az ilyen típusú hangszórók előnye, hogy omnidirekciójúak, azaz iránykarakterisztikájuk jó közelítéssel homogén. Ez előny nagy helységek hangosításában, ahol hagyományos hangszórókkal ügyesen kell lefedni a teret.

Nagy hátrányuk azonban, hogy a hangminőséget nem tudja a gyártó specifikálni, ugyanis az a felhasználótól függ, hogy hova szereli fel, és annak a helynek milyenek a fizikai tulajdonságai

F.4.4. Digitális hangszórók

Ahogy a technika fejlődött, úgy váltotta le az analóg képi megjelenítést a digitális megjelenítés. A digitalizálással a kép pixelekből áll össze, a pixeleket pedig egy adatfolyam alapján vezérli egy logika. A felépítés viszonylag összetett, de kivitelezhető, és jóval hatékonyabb átvitelt biztosít, mint az analóg megjelenítés tette.

A hangtechnikában a ma ismert hangszórók mind analóg technológiára épülnek. Kellőképpen komplex szerkezetek, ezért megfelelő minőségű változatok kifejlesztése igen összetett folyamat, általában több iterációs cikluson kell átesnie, hogy az elvárt specifikációt elérje, ráadásul mindezt nagy súlyú kompromisszumok árán lehet elérni.

A hangszórók nagyon rossz hatásfokú átalakítók, hiszen a beadott energia egy vagy két százalékát alakítják át hangnyomássá, a többi hőenergiaként elvész a tekercselésben. Képzeljük el, hogy ez egy hordozható eszközben mekkora pazarlást jelent.

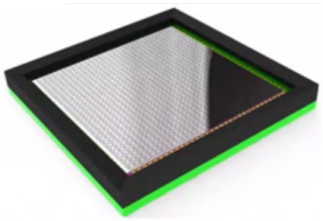
A jó képességű hangszórók jelentős fizikai méretekkel rendelkeznek. Már csak ezért is képtelenség hordozható eszközöket felszerelni igényes hangszórókkal, mert rengeteg helyet elfoglal csak a hangszóró, ráadásul a nagyobb méretű hangszórók jóval többet is fogyasztanak, hiszen nagyobb tömeget kell megmozgatniuk.

A képi megjelenítés mintájára jött az az ötlet, hogy mi lenne, ha a hangszintetizálást is digitalizálnák [62, 61]. Ezt digitális átalakítókkal kívánták elérni, amelyek olyan hangszórók voltak, amiknek két állapotunk volt összesen: vagy teljesen kitérített, vagy az alapállapot. A hangnyomást a két állapot közötti tranziensekkel érték el. Ekkor vagy pozitív vagy negatív egységnyi nyomásváltozást tudtak elérni.

A rendszer lényege, hogy ezáltal nagyon lapos átalakítókat lehet készíteni, nem kell DAC az átalakításhoz, kis fogyasztás az analóg hangszórókhoz képest, hiszen csak két állapotot kell tudni fenntartani, és csupán a tranziensekben történik tényleges energiafelvétel. Meghajtására a D osztályú erősítők tökéletesen alkalmasak. Nincsen meg bennük az analóg hangszórókra jellemző harmonikus torzítás, sőt a torzítást, amit generálnak, feltolják az

ultraszonikus tartományba.

Hátrányuk, hogy komplex vezérlés kell hozzájuk, ami a pontos hangvisszaadáshoz szükséges nagy elemszámú digitális átalakítók pontos vezérlését kell hogy megoldja. Az alacsony frekvenciás hangokat kis hangnyomással tudják csak visszaadni, ami a konstrukciójukból adódik (ferde frekvencia átvitel, tipikusan 6dB/octáv emelkedést produkálva). Megjelenik a kvantálási zaj, ami abból adódik, hogy az átalakító nem képes az egész számú egységnyi hangnyomásegységekből "kirakni" a szükséges hangnyomásértéket. Pontos hangszóróknál nagy átalakítószám szükséges, ebből adódhatnak gyártási hibák. Az egyes átalakítók tömege, felülete, sebessége nem feltétlenül tud megegyezni, ezáltal nem lesz homogén az általuk generált egységnyi hangnyomásváltozás, ami torzítást okoz a hang szintetizálásában. A hangvisszadás nem tökéletes közeli hallgatáskor, valamint a zaj erősen függ a hallgató elhelyezkedési szögétől is.



F.4.5. ábra. *Egyenletes elosztású digitális hangszóró*

Három fő változtuk ismert:

- PWM - Egy elemű, nagy hangnyomáshoz egyre sűrűbb tranzienseket produkál, az emberi fület használja "visszaállító aluláteresztő szűrőnek". A legrosszabb tulajdonságokkal bír a három típus közül.
- Binárisan súlyozott - Az átalakítók csoportokba vannak szervezve, amelyeknek a kettő hatványai szerint arányul egymáshoz a felületük és a tömegük. Belátható, hogy nagyon bonyolult konstrukciók, hiszen meg kell oldani, hogy a legkisebb egységnyi, és a nála 1024-szer nagyobb felületű és tömegű legnagyobb csoport (10 bites átalakító esetén) szinte tökéletes pontosan tartsa az arányokat, ugyanolyan sebességű tranziens mellett. Az energiafogyasztása is jóval nagyobb, mint az egyenletes elosztású átalakítónál, hiszen nullátmenetnél a legnagyobb és az összes többi kisebb csoport egyszerre vált magasból alacsonyba vagy fordítva.
- Egyenletes eloszlású - Sok azonos méretű átalakítóból áll. A legjobb tulajdonságú a három közül. Rendkívül kis energiafogyasztású. Belátható, hogy ebből a konstrukcióból a másik kettő konstrukció realizálható. (F.4.5.ábra)

A digitális hangszórók még egy aktív kutatási terület, az eredmények még messze alulmúlják az elvárásokat, de ha sikerül az áttörés, akkor képesek lesznek megváltoztatni a ma ismert hangsugárzó rendszereinket.

F.4.5. Flat panel hangszórók

Ahogy a nevük is mutatja, ezek a hangszórók szakítanak a hagyományosnak mondható elektrodinamikus hangszóró felépítésével, és egy új megközelítést alkalmaznak. Tervezésüknél a cél, hogy a hangszóró minél laposabb legyen, minél kevesebb teret töltsön ki, ezért magát a tekercset, ami a kitérést generálja egy lapos membránhoz rögzítik, ami lehet vinilből vagy más polimerből.

Ezek a hangszórók konstrukciójukból adódóan nehezen kivitelezhetőek, mert nehéz elérni, hogy a teljes membrán egyformán rezegjen.

Az elektrodinamikuson kívül van elektrosztatikus elven működő változata is, ahol két membrán van egymáshoz közel, mindkettő vezető, az egyiken állandó töltés található, a másikat pedig a meghajtóáramkör tölti periodikusan, ami által elektrosztatikus taszítás vagy vonzás lesz a két membrán között, ami a külső membrán kitérését okozza, így hanghullámok keletkeznek.

Ha belegondolunk, ez tökéletesen az ellentéte a kondenzátormikrofonok működésének azonos konstrukció mellett. A kondenzátormikrofonoknál a hanghullámok mozgatnak egy membránt egy másikhoz képest, ami elő van feszítve egy bizonyos feszültségszintre. A feszültség megváltozását a két egymáshoz közel lévő membrán alkotta kondenzátor kapacitásának a megváltozása okozza, miszerint a két membrán távolsága nő vagy csökken az őket érő hanghullámok szerint.

A hangszóró hátránya, hogy alsóbb frekvenciák átvitelére nem alkalmas, de kiegészítve egy mély hangtartományú hangszóróval, egy komplett hangrendszer része lehet.

Egy másik különleges fajtájuk a flat panel hangszóróknak a több tekercset alkalmazó hangszórók. Itt a homogén membránmozgás eléréséhez több hosszú tekercset alkalmaznak közös gerjesztéssel. Különlegessége, hogy a tekercsek közvetlenül a membránhoz vannak erősítve, és ezek mozgatják a membránt. Felépítésüket tekintve ezek a legrobosztusabb flat panel típusú hangszórók.

Nem csak flat panel konstrukcióban létezik membránba ágyazott tekercses hangszóró. A legtöbb miniatűr hangszóró ilyen konstrukcióban készül egy tekerccsel. Ilyenek a laptopokban, a fülhallgatókban, és néhány komolyabb mobiltelefonban megtalálható hangszórók. Egy ilyen miniatűr hangszóró látható a F.4.6.ábrán.

F.4.6. Plazma hangszórók

A plazma a szilárd, folyékony és légnemű halmazállapotok mellett az anyagok negyedik lehetséges halmazállapota. Ha két elektróda között gázban átívelés keletkezik, akkor az áram az ionizált gázon - plazmán - keresztül folyik. A plazmának tömege van, és hasonlóan

F.4.6. ábra. *Membránba ágyazott tekercses miniatűr hangszóró [2]*

a membránokhoz, rezgésbe hozható, és hangot képes kiadni. Azonban a generált hang minőségét illetően nem a legjobb, és a plazma folyamatos fenntartása is nehéz feladat.

További érdekessége, hogy a szabad levegőn működtetett plazma hangszóró ózont és dinitrogén-oxidot fejleszt, ezért a kísérleti hangszórókat levegő helyett hélium gázban működtették.

F.5. Objektumorientált MATLAB

F.5.1. Adatok tárolása MATLAB-ban

Adatokat általában változókban tárolunk. Minden változónak van típusa, ami megmondja, hogy mekkora helyet foglal a változó a memóriában. Ha több értéket akarunk letárolni egy bizonyos változóból, akkor erre a legjobb megoldás a tömbök használata. Egy tömb egy bizonyos típusú adathalmaz szekvenciális tárolására használható. Egy tömb használatához elég tudni, hogy mekkora helyet foglal a típus, amiből a tömb áll, hol van az első elem helye, és hány elemű a tömb. Ezen három adatból tetszőleges elem kivehető az tömbből. Akár többdimenziós tömböket is készíthetünk, ahol az adatok ugyanúgy szekvenciálisan vannak letárolva, csak a hozzáférés lesz egy átláthatóbb logika szerint megoldva.

A MATLAB-ban a tömböket vektoroknak nevezzük. Mivel egy interpretált dinamikus nyelvről van szó, azért nem kell megadni típust egy változó létrehozásánál, az interpreter kitalálja a megfelelő típust az értékül adott értékből. Ezzel a változók olyanok lesznek, mint az általános rekeszek, amikbe bármit belerakhatunk. Ennek a hátránya, hogy ellenben a statikus típusos nyelveknél, itt nincs típusellenőrzés fordítási időben (szinte fordítás sem szükséges), így a vétett hibák csak futási időben derülnek ki.

MATLAB-ban is létrehozhatunk többdimenziós tömböket. A kettő vagy többdimenziós tömböket mátrixoknak nevezzük. A nyelv ezek manipulálására lett kifejlesztve eredetileg lineáris algebrai alapon, majd később nőtte ki magát a mostani egyik legelterjedtebb tudományos eszközzé.

A tömböknek egy nagy hátránya van: csakis azonos típusú értékeket tárolhatunk bennük,

különben felborul a szekvenciális indexelés szabálya, ha két különböző típusnak más a mérete a memóriában.

Mi van akkor, ha mi mégis különböző méretű elemeket szeretnénk letárolni, és elérni szekvenciálisan? A legjobb példa erre a stringek listája.

A MATLAB erre nyújtott megoldása a cellák [5] használata. Egy cellában több tetszőleges méretű adat tárolható le tetszőleges dimenzióban.

F.5.1. kódrészlet. *Példa cell használatára*

```
numToStr{1} = 'one';
numToStr{2} = 'two';
numToStr{3} = 'three';
```

Ahogy az a példakódban is látható, a cellák használata megegyezik a vektorokéval (1-től indexelt elérés akár többdimenzióban is), azzal a különbséggel, hogy itt tetszőleges változót és objektumot is el lehet bennük tárolni.

Ha nem szeretnénk indexekkel hivatkozni egy-egy elemre egy adatláncon belül, akkor rendelkezésünkre állnak a struktúrák, amelyekben kulcsok szerint lehet eltárolni tetszőleges méretű adatokat.

F.5.2. kódrészlet. *Példa struct használatára*

```
point.x = 203,4;
point.y = 512.9987;
point.color = 'green';
```

Ezek a tárolási módszerek megfelelnek a procedurális programozás elvárásainak, azonban az adathoz nem társítanak eljárásokat. Obejktumorientált esetben adatok tárolására osztályokat hozunk létre, amelyekben egyrészről megadjuk, hogy mit tárolunk, másrészről megmondjuk, hogy milyen műveleteket tudunk végrehajtani az adott adaton.

Például ha egy pontokat tároló adatstruktúrát szeretnénk készíteni, ahogy azt az előbbi példában is megtettük, akkor a struktúra létrehozása után a szerkezet "lógni fog a levegőben", nem lesz semmilyen dokumentáció róla, hogy a kapott adathalmaz pontosan mi, milyen elemei vannak, mi hívható rá (az utóbbi állítások a MATLAB nyelvének sajátosságára igazak, itt ugyanis nem kell korrekt típusdefiníciót írni struktúrák esetére).

Ahhoz például, hogy ellenőrizhessük, hogy a létrehozott pont bizonyos korlátozásokon belül van-e, egy külön függvényt kell írni, amiben célszerű lekezelni minden olyan esetet, mikor nem található egy bizonyos mező a kapott adathalmazban. És minden egyes további, a struktúrán dolgozó függvényben ezt meg kell tenni. Ezek természetesen külön fájlokba kerülnek a MATLAB megkötései miatt, így egy több műveletet támogató adathalmaznál hamar átléphetjük a több 10 fájlos méretet. Nem mellékesen ezzel a módszerrel az adat és

a művelet egymástól külön van definiálva, azaz nem az adaton hajtunk végre műveletet, hanem a műveletnek adunk egy adatot, amit vagy fel tud dolgozni, vagy nem.

Belátható, hogy ha ezen az úton kezdünk el egy komolyabb rendszert készíteni, nagyon hamar elveszíthetjük az irányítást a rendszer struktúrája felett.

Összegzésképpen a MATLAB környezetében az adatok magasabb szintű tárolására három lehetőségünk van.

- Cellák
- Struktúrák
- Objektumok

Mindhárom módszer alkalmazásának megvan a helye, azonban vigyázni kell, hogy tényleg a számukra legalkalmasabb helyeken használjuk csak őket, különben könnyen bajba kerülhetünk kódkarbantartás szempontjából.

F.5.2. Objektumorientált programozás MATLAB-ban

Komolyabb projekteknél érdemes fontolóra venni, hogy objektumorientált megközelítést alkalmazzunk. Ennek az oka, hogy sokkal átláthatóbb lesz a kód, rengeteget spórolhatunk a polimorfizmussal és az öröklléssel, valamint rendkívül flexibilis struktúrát tudunk kialakítani interfészek használatával.

Az objektumoriantált programozás alapja az osztályok definiálása. Ezt a következő szintaxissal tehetjük meg egy külön fájlban:

F.5.3. kódrészlet. *Osztály definíciója*

```
classdef MyClass < Superclass
    properties
        % publikus tagváltozók
    end

    methods
        % publikus metódusok
    end
end
```

A szintaxis egyszerű, mindennek megvan a maga helye, és egy fájl elég egy osztály definiáláshoz, hasonlóan a Java nyelvhez.

Egy osztály több osztálytól is örökölhet. Ilyenkor 'and' karakterrel választjuk el az egyes ősosztályokat egymástól.

F.5.4. kódrészlet. *Több ősosztály megadása*

```
classdef MyClass < Superclass1 & Superclass2
  ...
end
```

A MATLAB érdekesen kezeli az éppen aktuális objektumra mutató referenciát[5]. Ellentétben a többi nyelvvel (legelterjedtebb objektumorienált nyelvek), itt nem egy rejtett paraméteren keresztül fér hozzá a *this* mutatón keresztül, hanem implicit nekünk kell megadni egy első paramétert, amit a futtatási környezet automatikusan átad minden nem statikus metódushívásnál. Ez alól kivétel a konstruktor hívása. Ez a két eset (konstruktor, statikus metódusok) az, amikor a definiált, és a ténylegesen hívott paraméterlista egyforma hosszú, ha az opcionális paramétereket nem vesszük figyelembe.

F.5.5. kódrészlet. *Konstruktor szintaxisa*

```
classdef MyClass < Superclass
    properties
        name;
    end
    methods
        function self = Myclass(name)
            self.name = name;
        end
    end
end
```

A konstruktorban az önmagunkra mutató pointert adjuk vissza, amit a konstruktor törzsében teljesen felkonfigurálunk. Jelen esetben egy paraméternek kapott nevet mentünk le egy tagfüggvénybe. A saját magunkra mutató referencia elnevezése saját preferencia kérdése. Én szeretem *self*-nek nevezni, de bármi más is használható lenne.

A konstruktor hívásában nincs semmi egyedi.

F.5.6. kódrészlet. *Az előző példa osztályának példányosítása*

```
myObject = MyClass('Botond')
```

F.5.7. kódrészlet. *Az előző példa kimenete*

```
myObject =

  MyClass with properties:

    name: 'Botond'
```

Legyen egy metódusunk, ami két paramétert vár. Mint a példakódban is látható, a metódus definiálásánál három paramétert kell megadni, hiszen az első paraméter a saját magunkra mutató referencia:

F.5.8. kódrészlet. *Konstruktor szintaxisa*

```
classdef MyClass < Superclass
    properties
        name;
    end
    methods
        function ret = myMethod(self,par1,par2)
            ret = par1 + par2;
```

[5]Ezen a referencián keresztül érhető el az adott metódushoz tartozó objektum, és így férünk hozzá az objektomunkhoz tartozó tagváltozókhoz is.

```
        end
    end
end
```

Ahogy azt a fenti példakódok felett tárgyaltuk, a metódusokban a saját magunkra mutató referencia kezelése érdekes a MATLAB nyelvében, és a metódusok hívása is. A következő példában a két hívás teljesen egyenértékű:

F.5.9. kódrészlet. *Metódusok hívása*

```
myObject = MyClass();

sum = myObject.myMethod(4,2);
sum = myMethod(myObject,4,2);
```

Lehetőség van metódusok és tagváltozók elrejtésére is, hiszen a MATLAB, mikor egy létező objektumot lekérdezünk tőle, kilistázza az összes látható tagváltozót, a *methods* kulcsszóra pedig az összes látható tagfüggvényét listázza ki.

F.5.10. kódrészlet. *Rejtett tagváltozók és metódusok*

```
classdef MyClass < Superclass

    properties (Hidden)
        ...
    end

    methods (Hidden)
        ...
    end

end
```

Absztrakt metódusok definiálásához a következő szintaxis használható:

F.5.11. kódrészlet. *Rejtett tagváltozók és metódusok*

```
classdef MyClass < Superclass

    methods (Abstract)
        ...
    end

end
```

Ha egy osztálynak vannak absztrakt metódusai, akkor már nem lehet példányosítani, de ez nem zárja ki, hogy ne legyenek sima metódusai, amelyeknek létezik implementációja. Ez teljesen bevett technika arra, ha egy absztrakt ősosztály nem csak implementációs felelősséget ad át a leszármazottainak, hanem kész képességeket is. Ezt ki is használjuk a következő fejezetben bemutatott blokkos mérési rendszer implementáció tárgyalásánál.

F.5.3. Value és handle típusú ősosztályok

A MATLAB világában, ha egy osztály nem örököl semmitől, akkor a *value* típusú osztályok kategóriájába fog esni[6]. Ez azt jelenti, hogy a következő kódrészlet nem fog működni:

F.5.12. kódrészlet. *Érték típusú osztály definiálása*

```
classdef MyClass

    properties
        a;
    end

    methods
        function self = MyClass(a)
            self.a = a;
        end

        function setATo(self,a)
           self.a = a;
        end
    end

end
```

F.5.13. kódrészlet. *Érték típusú osztály használata*

```
clear all;

o = MyClass(3);

o.setATo(4)

if o.a == 4
    disp('ok')
end
```

A fenti kódrészlet nem fog az elvártak szerint működni! Igaz, hogy az objektum metódusán keresztül beállítjuk a tagváltozóját az objektumnak 4-re, de ne felejtsük el, hogy *érték* típusú osztályunk van, azaz ha metódust hívunk meg egy belőle példányosított objektumra, akkor az nem az eredeti példányosított objektum adódik át referenciaként, hanem az objektum egy új másolata. Erre egyébként a MATLAB figyelmeztet is, mikor *érték* típusú osztály metódusában a tagváltozókat módosítunk.

Érték típusú objektumnál csak és kizárólag a konstruktorban tudunk belső tagváltozókat beállítani.

Mi a megoldás erre a problémára? Minden osztályunkat, amiből a származtatott objektumokat referenciaként akarunk átadni metódusoknak - én személy szerint a még nem használtam olyan esetet, ahol nem referenciaként adtam volna át az objektumomat - le

[6] Az elnevezés önkényes a MATLAB szempontjából. Vehetjük úgy, hogy az ilyen osztályokat érték szerint adjuk át, tehát egy konkrét másolat fog készülni, és nem egy kezelésére alkalmas - *handle* - referencián át érjük el.

kell származtatni a MATLAB *handle* beépített ősosztályából, ami tartalmazza a szükséges metódusokat a referenciaként való kezeléshez.

F.5.14. kódrészlet. *Handle típusú osztály definiálása*

```
classdef MyClass < handle

    properties
        a;
    end

    methods
        function self = MyClass(a)
            self.a = a;
        end

        function setATo(self,a)
           self.a = a;
        end
    end

end
```

Ha újra lefuttatjuk a F.5.13. kódrészletet, akkor már egy 'OK' üzenetet fog kiírni nekünk a MATLAB.

F.5.4. Néhány jótanács

F.5.4.1. Objektumok frissítése

Objektumorientált fejlesztés közben, ha módosult az osztályunk, akkor ki kell törölni a MATLAB minden olyan objektumát a *Workspace*-ből, ami a frissült osztályból van példányosítva, mert a MATLAB a nem frissített objektumokat a régi osztály szerint értelmezi, és látszólag megfejthetetlen, értelmetlen hibaüzeneteket képes adni rá az interpreter, hogy olyan metódusokra hivatkozunk, amik nem is léteznek, holott éppen az előbb mentettük el a módosításokat.

Számomra a jól bevált megoldás, hogy minden scriptem elejét a következő mindent törlő, mindent bezáró, konzolt törlő varázsutasításokkal kezdem:

F.5.15. kódrészlet. *Megfontolandó script-kezdő utasítássorozat*

```
clear all; close all; clc;
```

F.5.4.2. Debuggolás

Debuggolásnál figyeljünk oda, hogy ha látszólag nem fut bele a kód egy osztály belsejében egy metódusra, pedig meghívjuk azt egy scriptből, és még breakpoint-ot is állítottunk rá az osztályban, akkor az nem a mi hibánt, hanem a MATLAB-é. Az interpreter úgy veszi, hogy ha a hívási fában a legmagasabban elhelyezkedő elemben (ami általában az osztályainkat használó scriptek) nincs breakpoint, akkor nem is kell debuggolni semmit, és minden további beállított breakpointot töröl minden megnyitott fájlból.

Ez nagyon idegesítő viselkedés.. Nem lehet mást tenni, mint a scriptben oda tenni egy breakpoint-ot, amelyik utasítás hatására majd egyszer meghívódik a számunkra érdekelt objektum metódusa, és a *Step* debugger parancs helyett a *Step in* parancsot kell használjuk a megfelelő helyeken. Ez felesleges gondolkodást, és sokszor félrenyomott debugger-parancsokat jelent. Az esetemben a legkiemelkedőbb eset, mikor egy metódushívást 8-9 másik metódushívás előzőtt meg, és ha egy ciklusban van ez, ahol az ellenőrzendő metódus működését a ciklusban iterált értékekekkel akarjuk ellenőrizni, akkor kellemes perceket tud okozni a MATLAB debuggere.

Egy lehetségees megoldás a problémára, ha scriptekben tesztelés helyett, élőben a konzolban próbáljuk ki az osztályainkat, hiszen így meghívva egy utasításláncot, a debugger természetesen hajlandó megállni mélyebb hívási fáknál is..

F.6. Duplán láncolt lista: elemek cseréje

A mérés lényege, hogy a hangszórót egy swept sine méréssel és az azt követő lineáris regressziós eljárással modelleztük Wiener-Hammerstein modellel.

Feladat: duplán láncolt listában meg akarunk cserélni két elemet. Ez elsőre egyszerűnek hangzik, de a valóságban nem is olyan triviális feladat. A lehetséges esetek leírásával azonban könnyen beláthatóak az implementálandó szabályok. A rész végén egy működő C kódot közlök, amivel akár az interneten, egy online fordítóval ellenőrizhető a módszer helyessége.

Legyen a két kicserélendő elemünk A és B. Két eset lehetséges. Vagy szomszédosak, vagy nem.

F.6.1. Szomszédos eset

Tegyük fel, hogy a következő eset áll fent:

```
[X] - [A] - [B] - [Y]

A->prev = X;
A->next = B;
B->prev = A;
B->next = Y;
```

Ha felcseréljük A és B elemet, akkor a következőre változik a lánc:

```
[X] - [B] - [A] - [Y]

A->prev = B;
A->next = Y;
B->prev = X;
B->next = A;
```

Mátrixos alakban is felírhatjuk az átalakítást:

```
      A B              A B
prev X A    =>    prev B X
next B Y          next Y A
```

Csak a mátrixokat felírva:

```
X A   --\   B X
B Y   --/   Y A
```

Figyeljük meg, hogy a csere elforgatja az eredeti mátrixot jobbra 90 fokkal. Ha indexeljük az elemeket, akkor könnyen felírhatunk egy asszociációs táblázatot:

```
0 1  --\  2 0
2 3  --/  3 1
```

Ha a mutatókat egy tömbben tároljuk, akkor a csere egyszerű tömbindexeléssel könnyen megoldható:

```
0,1,2,3 -> 2,0,3,1
```

F.6.2. Nem szomszédos eset

A szomszédos esethez hasonló szabály vezethető le ugyanolyan következtetésekkel a nem szomszédos esetben is. Csak a levezetést közlöm:

```
[W] - [A] - [X] - ... - [Y] - [B] - [Z]

A->prev = W;
A->next = X;
B->prev = Y;
B->next = Z;
```

Swap A with B:

```
[W] - [B] - [X] - ... - [Y] - [A] - [Z]

A->prev = Y;
A->next = Z;
B->prev = W;
B->next = X;

     A B               A B
prev W Y    =>    prev Y W
next X Z          next Z X

W Y   --\   Y W
X Z   --/   Z X

0 1  --\  1 0
2 3  --/  3 2

0,1,2,3 -> 1,0,3,2
```

F.6.3. Mutatók a cserélendő elemek felé

Mivel kétszeresen láncolt listával dolgozunk, ezért nem elég, ha csak az elemek két mutatóját kicseréljük. Be kell még állítani az eredetileg a cserélendő elemekre mutató mutatókat a

másik cserélendő elemre. Szerencsére a pointerek, amiket módosítani kell, pont a cserélendő elemek szomszédságában vannak, így a cserélendő elemeken keresztül elérjük őket.

Az eljárás a következő: miután elvégeztük a megfelelő mutató-transzformációt a cserélendő elemekre, a bennük frissült új mutatókra meghívjuk a következő kódot, ami saját magukra állítja a szomszédos elemek mutatóit, ahogy az az elvártak szerint van.

```
A->prev->next = A
A->next->prev = A
```

Ezekkel a lépésekkel már megírhatjuk a csrét elvégző függvényeket:

```
int areTheyNeighbours(Node A,Node B) {
    return ( A->next == B && B->prev == A ) ||
           ( A->prev == B && B->next == A );
}

void refreshOuterPointers(Node A) {
    if (A->prev != NULL)
        A->prev->next = A;

    if (A->next != NULL)
        A->next->prev = A;
}

void swap(Node A,Node B) {
    Node swapperVector[4];

    swapperVector[0] = A->prev;
    swapperVector[1] = B->prev;
    swapperVector[2] = A->next;
    swapperVector[3] = B->next;

    if (areTheyNeighbours(A,B)) {
        A->prev = swapperVector[2];
        B->prev = swapperVector[0];
        A->next = swapperVector[3];
        B->next = swapperVector[1];
    } else {
        A->prev = swapperVector[1];
        B->prev = swapperVector[0];
        A->next = swapperVector[3];
        B->next = swapperVector[2];
    }

    refreshOuterPointers(A);
    refreshOuterPointers(B);
}
```

A következő kód egy működő programot tartalmaz, aminek a kimenete a következő:

```
Initial state:   [1]-[2]-[3]-[4]
[1] <-> [2]:     [2]-[1]-[3]-[4]
[2] <-> [4]:     [4]-[1]-[3]-[2]
```

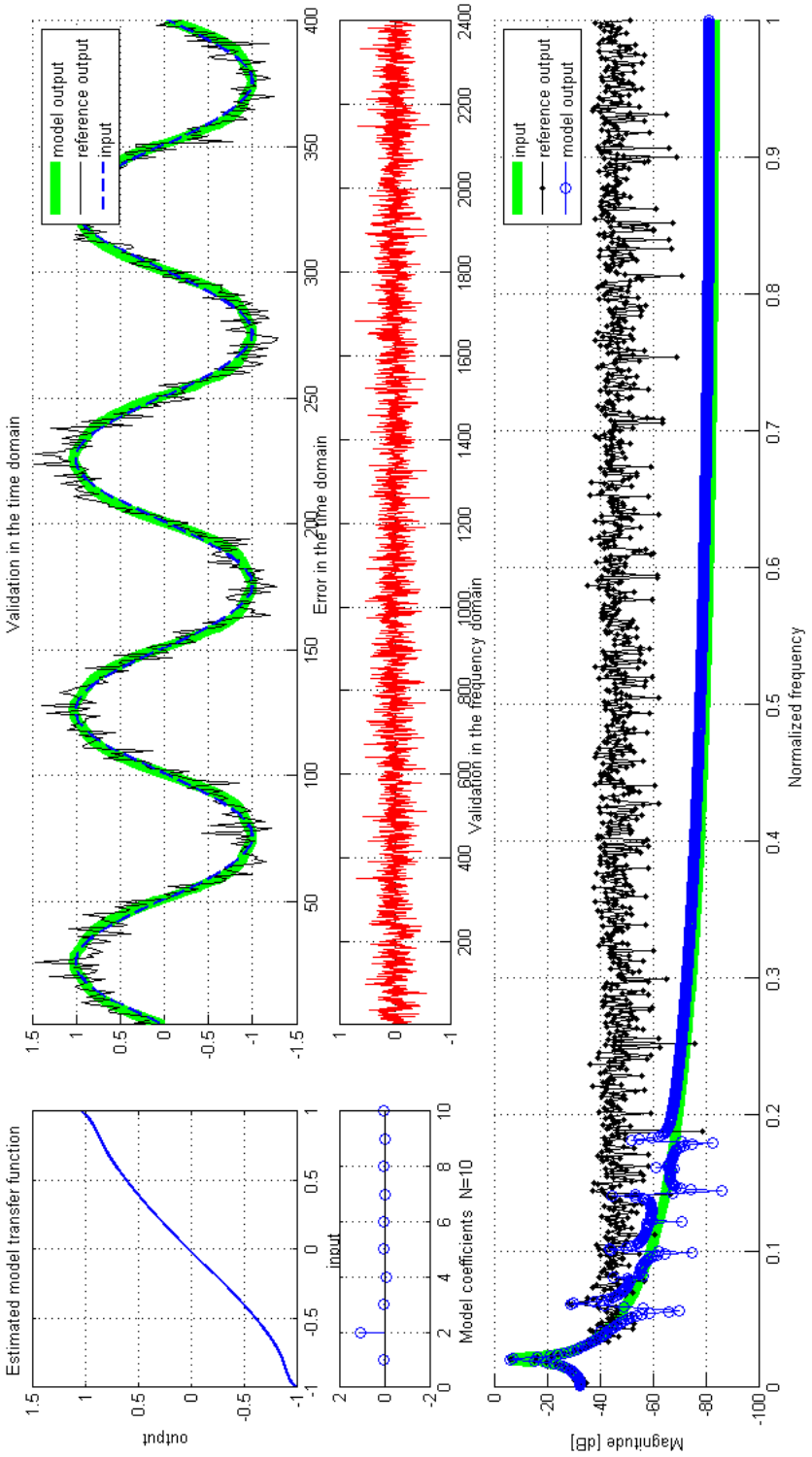
A kód futtatható egy online fordítóban az alábbi linken: `http://codepad.org/UHcmmag1`.

```
#include <stdio.h>
```

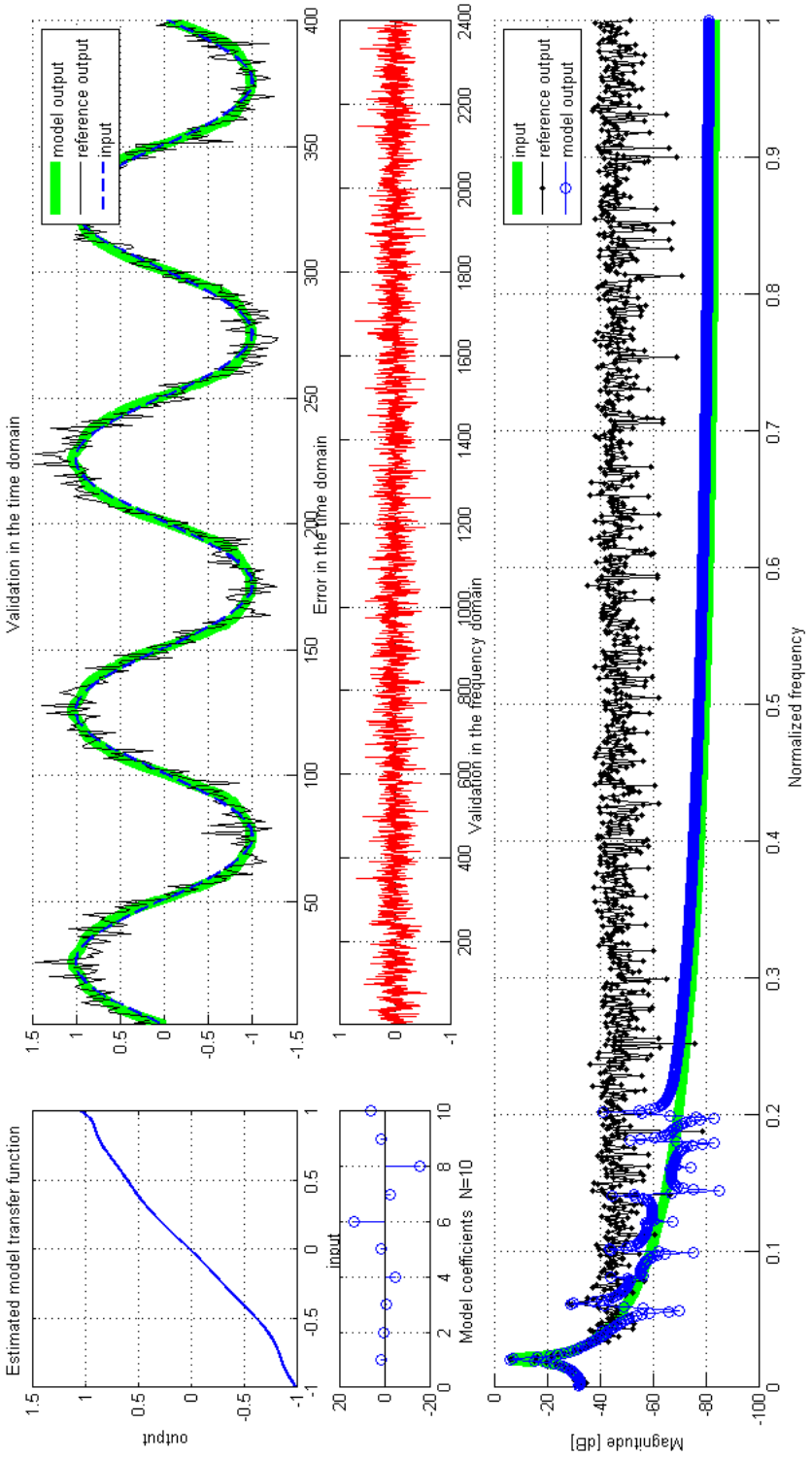
```c
#include <stdlib.h>

typedef struct node_v {
    int id;
    struct node_v* prev;
    struct node_v* next;
} Node_v;

typedef Node_v* Node;

void print(Node node) {
    while (node->prev != NULL)
        node = node->prev;

    printf("   [%d]",node->id);

    while (node->next != NULL) {
        node = node->next;
        printf("-[%d]",node->id);
    }

    printf("\n");
}

void connect(Node first,Node second) {
    first->next = second;
    second->prev = first;
}

Node createNode(int id) {
    Node node = (Node)malloc(sizeof(Node_v));
    node->id = id;
    node->prev = NULL;
    node->next = NULL;
    return node;
}

int areTheyNeighbours(Node A,Node B) {
    return ( A->next == B && B->prev == A ) ||
           ( A->prev == B && B->next == A );
}

void refreshOuterPointers(Node A) {
    if (A->prev != NULL)
        A->prev->next = A;

    if (A->next != NULL)
        A->next->prev = A;
}

void swap(Node A,Node B) {
    Node swapperVector[4];

    swapperVector[0] = A->prev;
    swapperVector[1] = B->prev;
    swapperVector[2] = A->next;
    swapperVector[3] = B->next;

    if (areTheyNeighbours(A,B)) {
        A->prev = swapperVector[2];
```

```
62          B->prev = swapperVector[0];
63          A->next = swapperVector[3];
64          B->next = swapperVector[1];
65      } else {
66          A->prev = swapperVector[1];
67          B->prev = swapperVector[0];
68          A->next = swapperVector[3];
69          B->next = swapperVector[2];
70      }
71
72      refreshOuterPointers(A);
73      refreshOuterPointers(B);
74  }
75
76  int main() {
77      Node n1 = createNode(1);
78      Node n2 = createNode(2);
79      Node n3 = createNode(3);
80      Node n4 = createNode(4);
81
82      connect(n1,n2);
83      connect(n2,n3);
84      connect(n3,n4);
85
86      printf("\nInitial state:");
87      print(n2);
88
89      printf("[1] <-> [2]:  ");
90      swap(n1,n2);
91      print(n1);
92
93      printf("[2] <-> [4]:  ");
94      swap(n2,n4);
95      print(n4);
96
97      return 0;
98  }
```

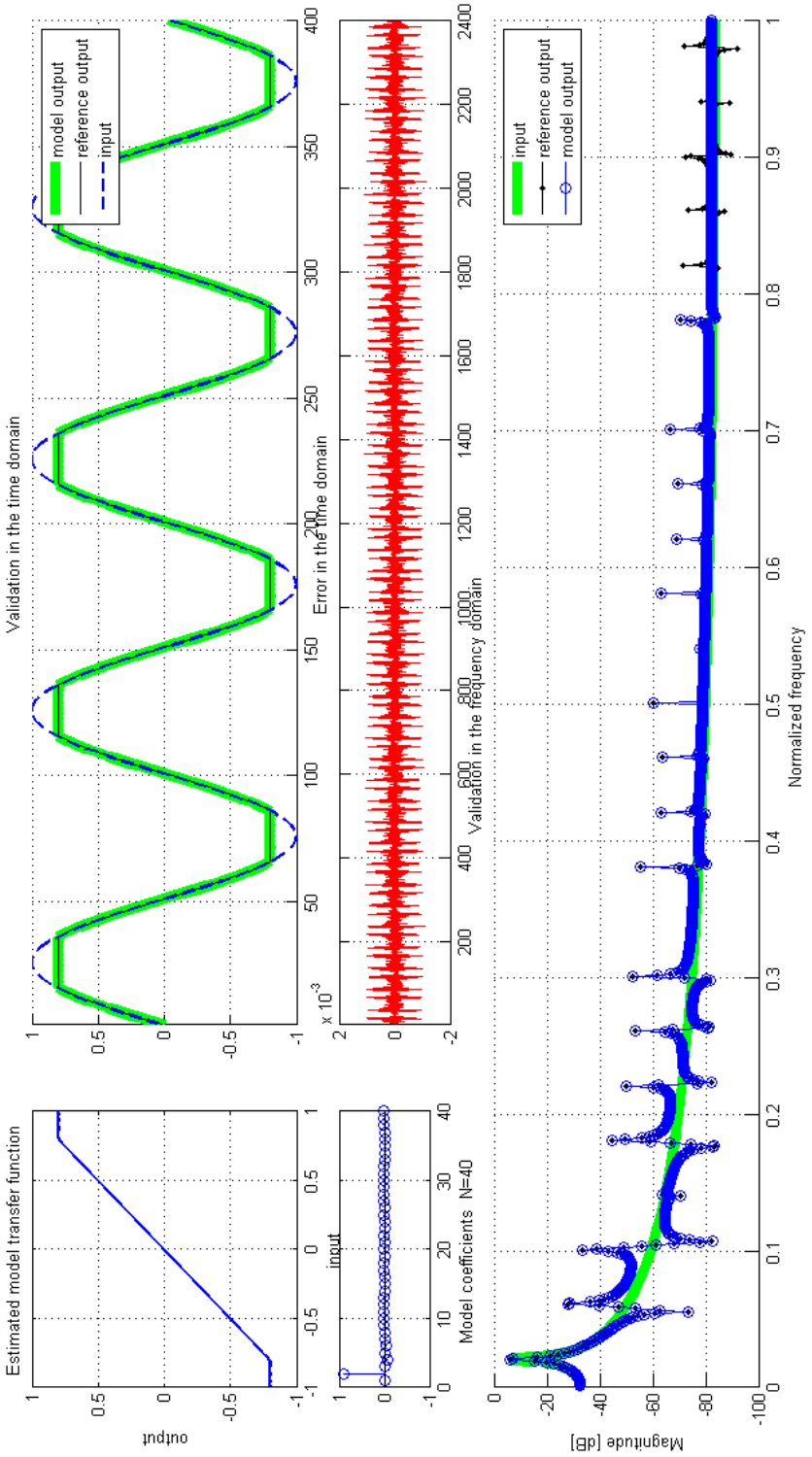
F.7. Lineáris regresszió további szemléltető példái



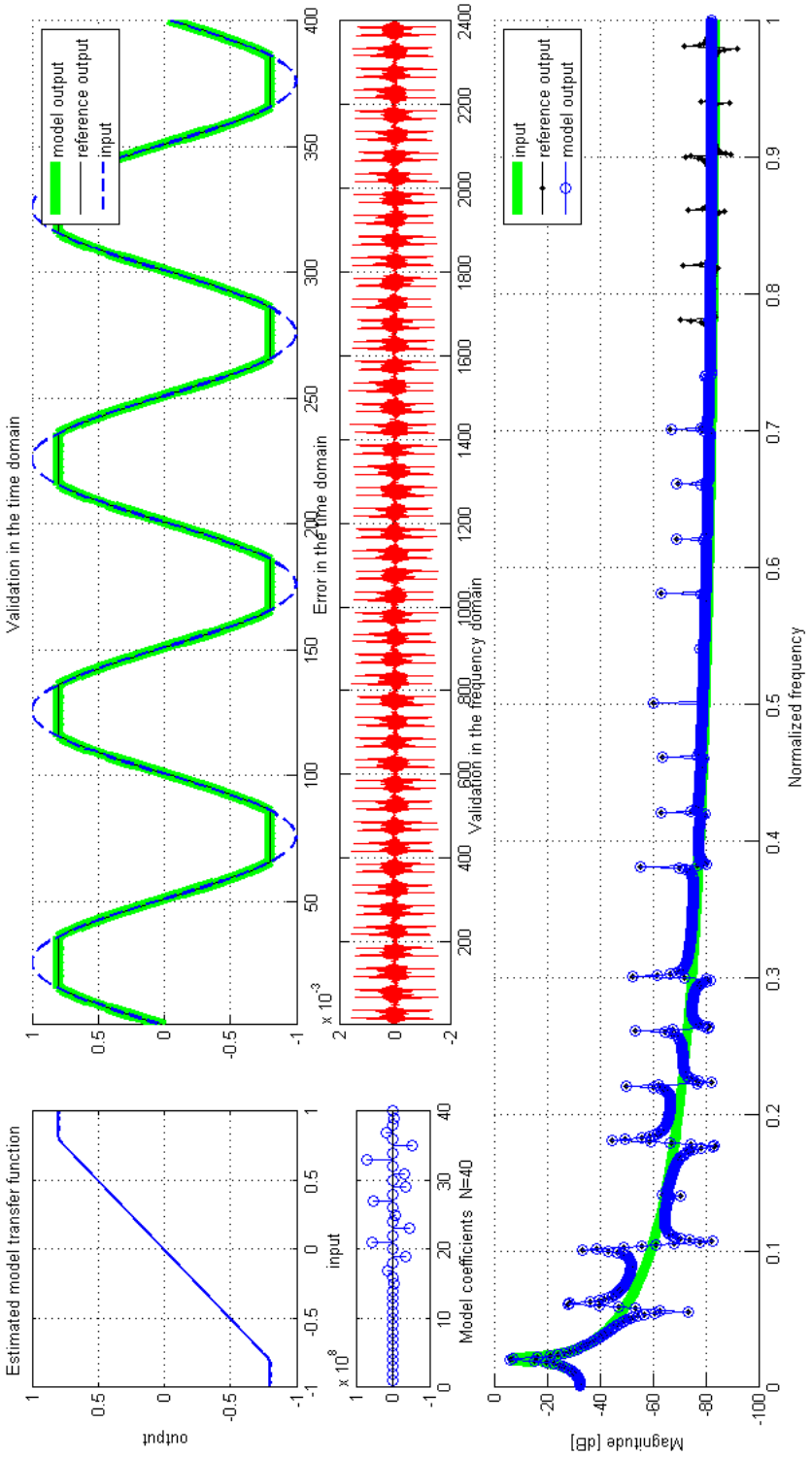
F.7.1. ábra. *3: Szigmoid rendszer Chebyshev modell N=10 SNR=10dB*



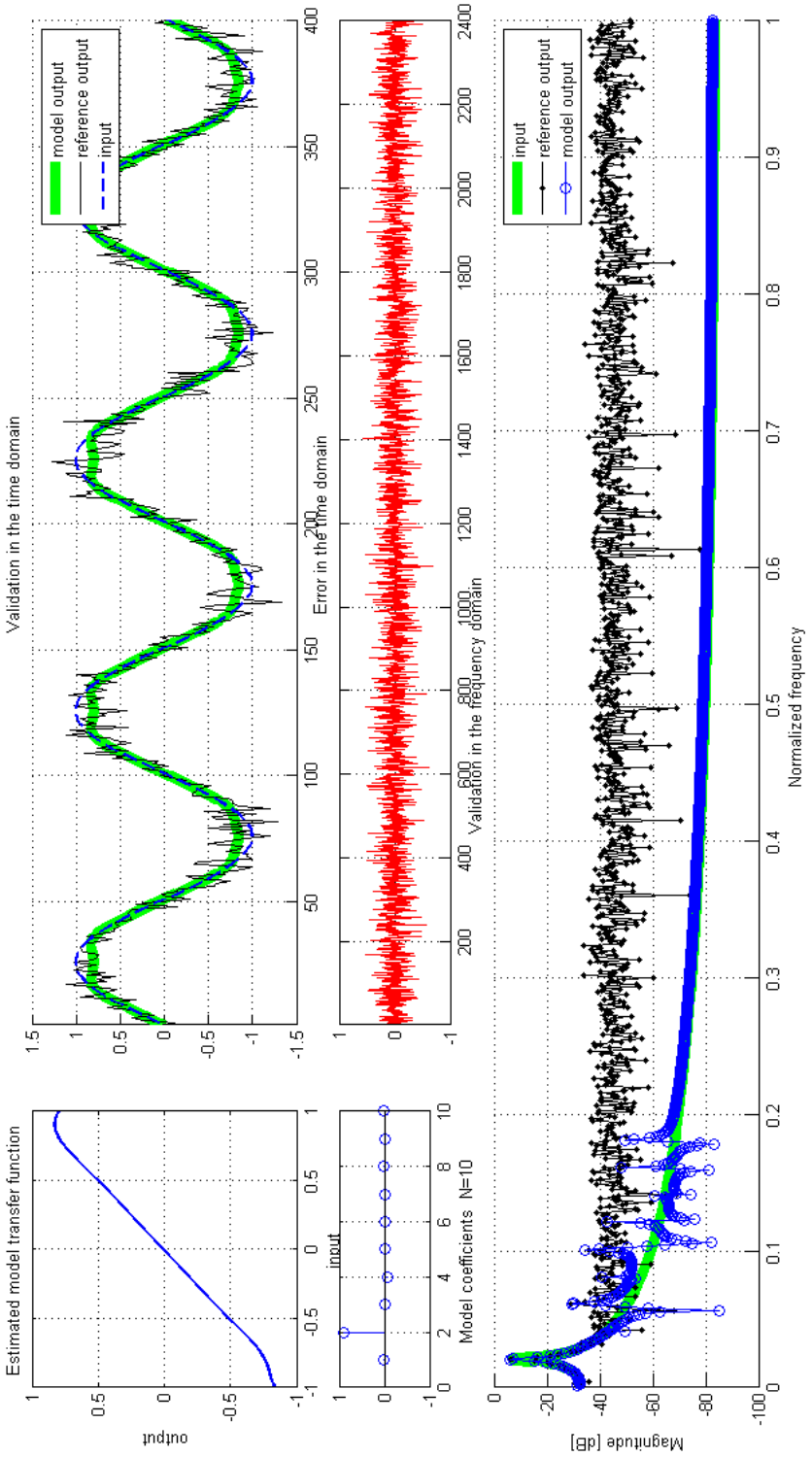
F.7.2. ábra. *4: Szigmoid rendszer Polynomial modell N=10 SNR=10dB*



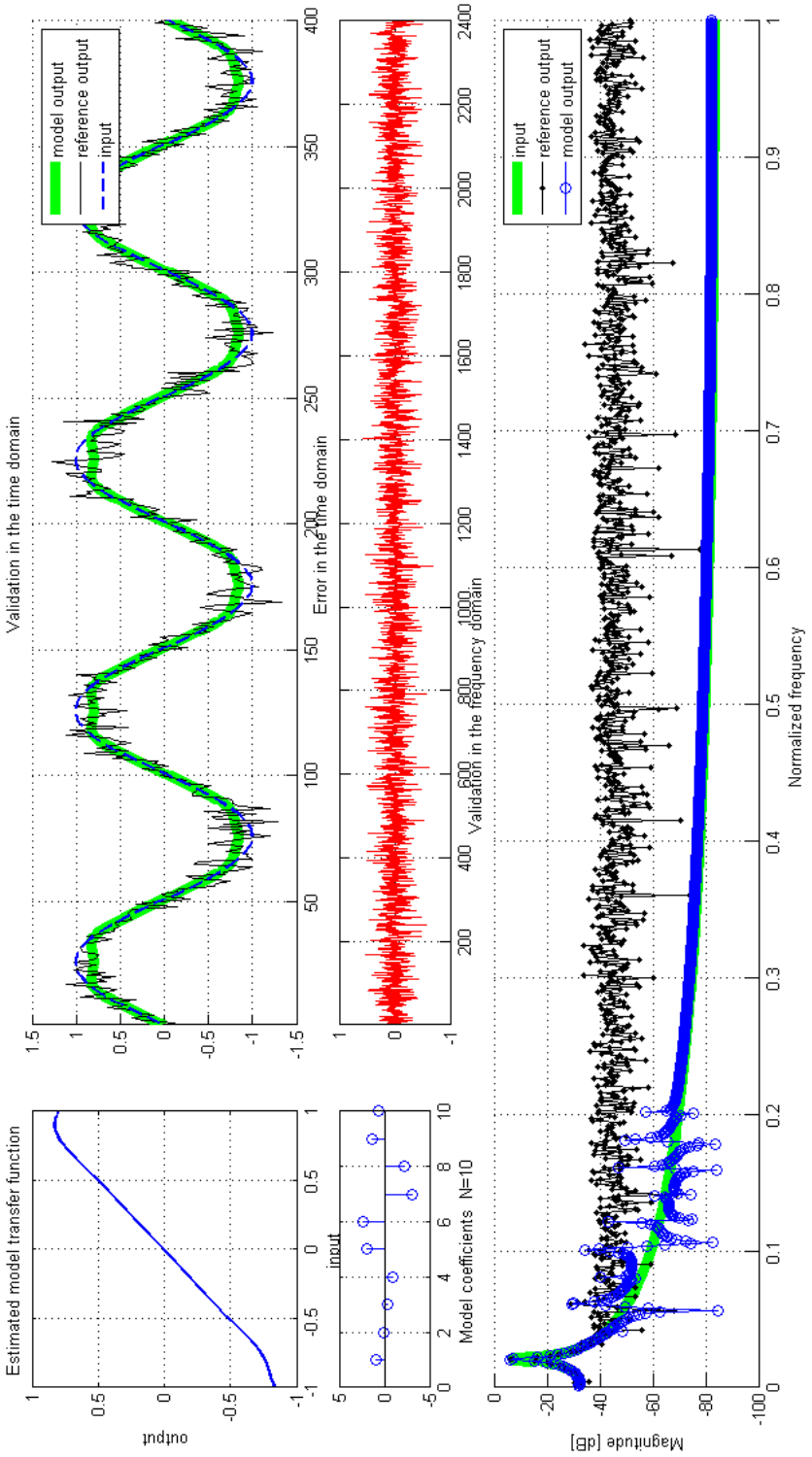
F.7.3. ábra. *5: Telítődéses rendszer Chebyshev modell N=40 SNR=40dB*



F.7.4. ábra. *6: Telítődéses rendszer Polynomial modell N=40 SNR=40dB*



F.7.5. ábra. *7: Telítődéses rendszer Chebyshev modell N=10 SNR=10dB*



F.7.6. ábra. *8: Telítődéses rendszer Polynomial modell N=10 SNR=10dB*