Tudományos Diákköri Konferencia

Gépi tanulási algoritmusok és modell alapú prediktív szabályozás kombinációs lehetőségeinek vizsgálata a járműirányításban

Szerző:
Bokor Ákos

Budapesti Műszaki és Gazdaságtudományi Egyetem
Közlekedés- és Járműirányítási Tanszék

Konzulens:
Dr. Szabó Ádám

Tartalomjegyzék

1. Bevezető **2**

2. Szakirodalmi áttekintés **4**
2.1. Szabályozási megoldások áttekintése 4
2.2. Gépi tanulási algoritmusok áttekintése 6
2.3. ML-MPC megoldások áttekintése 8

3. Járműdinamikai modellek **10**
3.1. Dinamikai bicikli modell 10
3.2. Dinamikai hiba-alapú bicikli modell 13

4. Algoritmusok **15**
4.1. Modell Prediktív Szabályozás 15
4.1.1. Állapottér-modell 15
4.2. MPC költségfüggvény és minimalizálása 17
4.2.1. Globális megoldás 19
4.2.2. Kényszerek figyelembe vétele 20
4.2.3. Egyenlőtlenségi kényszerek megoldása Kuhn-Tucker feltételekkel . . 21
4.2.4. Hildreth kvadratikus programozás 22
4.2.5. Zárt alakú megoldás λ^*-ra 23
4.3. Felügyelt tanulás 23

5. Javasolt Supervised Learning - Model Predictive Control struktúrák **27**
5.1. A QP teljes kiváltását célzó struktúra 27
5.2. A QP gyorsítását célzó struktúrák 29

6. Eredmények **31**

7. Konklúzió **40**

Kivonat

A magasan automatizált és önvezető járművek egyre nagyobb szerepet játszanak a közlekedésben és a mindennapi életben, ahogy egyre gyorsabban haladunk a Society of Automotive Engineers (SAE) által javasolt 5-ös járműautomatizálási szint elérése felé. Az intelligens járművek és eszközök elterjedése lehetővé teszi azok magas szintű és kooperatív irányítását, ezáltal hozzájárulva a balesetek számának csökkenéséhez, az energiahatékonyság növekedéséhez és a szén-dioxid-kibocsátás mérsékléséhez. Ennek eléréséhez azonban az egyre bonyolultabb rendszerek megnövekedett hardveres és adatfeldolgozási igényei magas költségeket és új kihívásokat jelentenek. Mindemellett az önvezető járműveknek egyre szigorúbb követelményeknek kell megfelelnie, mint például minél pontosabb referenciajel követés és az utaskényelem biztosítása, amelyek sokszor ellentmondásos feltételek lehetnek. Ezen követelmények egyidejű teljesítéséhez számos különböző közlekedési szituációban "emberi szintű" teljesítmény szükséges, ezért kezdtek elterjedni a gépi tanuláson alapuló megoldások. Ugyan a gépi tanulás alapú megközelítések képesek kizárólag adatok alapján erősen nemlineáris leképezések közelítésére, alkalmazásuk sok területen nehézségekbe ütközik. Kiemelendő ezek közül a mélytanulási algoritmusokra jellemző bonyolultság ami megnehezíti ezek valós idejű kiértékelését. Ezen kívül az optimalizált (megtanult) paraméterek gyakran nem rendelkeznek fizikai jelentéssel, ami nehezíti az ilyen jellegű megoldások alkalmazását biztonságkritikus területeken. Irányítási feladatok esetén például, a klasszikus módszerekkel ellentétben nem garantált a stabilitás megléte.

Ennek fényében, a dolgozat célja a kényszeres MPC (Model Predictive Control) és felügyelt tanulás kombinációs lehetőségeinek vizsgálata az időigényes optimalizáció gyorsítására, a költséghatékonyság növelése érdekében. A dolgozatban három különböző NN-MPC struktúra kerül bemutatásra, amelyek a hagyományos MPC és a "warm start" MPC megoldásokkal kerülnek összehasonlításra. A felügyelt tanuláshoz egy többrétegű perceptron neurális hálózatot alkalmaztam. A rendszer sebességfüggő dinamikájának figyelembevétele érdekében a szabályozó a rendszer lineáris paraméter-változó modelljén alapul. Mindhárom NN-MPC algortimus alkalmazásával jelentősen csökkent az irányítási algoritmus futási ideje, valamint a kvadratikus programozás iterációinak száma a kényszerek kezelésében. Ezek az eredmények hozzájárulnak a gyorsabb online működéshez és a számítási kapacitás igényének csökkentéséhez, valamint a klasszikus módszerekkel való ötvözés miatt olyan teljesítmény garanciák adhatóak, melyek tisztán ML alapú módszerek esetében nem.

Köszönetnyilvánítás

Szeretném megköszönni elsősorban Dr. Szabó Ádámnak a közvetlen konzulensemnek, a TDK dolgozatomhoz nyújtott fáradhatatlan támogatását és konzultációkat. Köszönöm Dr. Aradi Szilárdnak, a projektvezetőmnek a szakmai támogatást. Köszönöm Knáb Istvánnak, a feladat során felmerűlö Gépi Tanulással kapcsolatos szakmai problémák megvitatását, és megoldását. Végezetül köszönöm Dózsa Tamásnak a HUN-REN SZTAKI kutatójának a hasznos észrevételeket.

Budapest, 2024. november 6.

Bokor Ákos

1. fejezet

Bevezető

Az autóipar az elmúlt évtizedekben jelentős fejlődésen ment keresztül, különösen az autonóm járműtechnológiák terén. A járművek autonómiájának skáláján előre haladva egyre magasabb szintű önvezető képességek valósulnak meg, kezdve az alapvető vezetéstámogató rendszerektől egészen az SAE Level 5 szintű, teljesen autonóm járművekig, amelyek képesek önállóan navigálni a városi és országúti közlekedésben.

A közlekedési rendszerek hatékonyságának, biztonságának és kényelmének növelése érdekében a gyártók egyre gyakrabban alkalmaznak adatvezérelt megközelítéseket. Az összekapcsolt járművek és infrastruktúrák által generált hatalmas mennyiségű adat lehetővé teszi a gépi tanulás (ML) alapú módszerek integrálását a járműirányítási rendszerekbe. Ugyanakkor a járművek nemlineáris dinamikája és a valós idejű irányítási kihívások szükségessé teszik a többcélú, feltételes optimalizálást.

A Model Prediktív Szabályozás (MPC) egy olyan irányítási stratégia, amely a rendszer aktuális állapotából kiindulva, képes előre jelezni a jövőbeli állapotokat. Az MPC alkalmazása az autóiparban már az 1990-es évek közepe óta elterjedt, és az utóbbi években egyre inkább kombinálják a gépi tanulás alapú megközelítésekkel, létrehozva az ML-MPC módszereket [1][2]. Ennek a kombinált megközelítésnek a célja az MPC hátrányainak leküzdése, mint például a magas számítási igény és az pontos rendszermodell szükségessége.

Az ML-MPC technológiák alkalmazása a járműirányítási rendszerekben elősegíti a járművek magasabb szintű döntéshozatali képességeit és a környezeti változásokhoz való gyors adaptációt. Ez nemcsak az autonóm járművek navigációs hatékonyságát és a közlekedésbiztonságot javítja, hanem növeli a felhasználói kényelmet is. Az ilyen rendszerek lehetővé teszik, hogy a járművek intelligensen reagáljanak a változó környezeti feltételekre, optimalizálják útvonalaikat és hatékonyabban kezeljék az előre nem látható eseményeket.

Az SAE Level 5 szintű automatizáltság elérésének érdekében az ML-MPC megoldások kulcsszerepet játszanak. A jövő autonóm járműrendszereinek fejlesztése során ezen technológiák továbbfejlesztése és integrálása alapvető fontosságú annak érdekében, hogy a járművek emberi beavatkozás nélkül is biztonságosan és hatékonyan működhessenek.

Ebben a TDK dolgozatban olyan új ML-MPC módszereket vezettem be, melyek alkal-

masak a kényszeres MPC sokszor időigényes optimalizációjának a gyorsítására, az előírt kényszerek betartása mellett. Az optimalizáció gyorsítására több megoldás is létezik. Amíg a [3] cikkben bemutatott algoritmus az egész szabályozási sorozatot prediktálja visszacsatolva egy megerősítést a háló becslésére, addig az általam létrehozott struktúra A Lagrange szorzók egzakt becslésére összpontosít, egy kezdeti értéket adva az optimalizációnak. Egy másik megközelítés egy olyan valószínűségi (HMM) modell alkalmazása, ami az adott időpillanatban a legnagyobb valószínűséggel aktív kényszerkonfigurációk előrejelzését becsli, míg az általam bemutatott módszer bináris predikcióval közvetlenül előre jelzi, hogy az adott kényszer aktív vagy passzív. A dolgozat legfontosabb eredményei közé tartozik, a kényszeres kompenzáció neurális hálóval való becslése által a teljes kvadratikus programozás kihagyása, miközben betartja az előírt kényszereket, és megfelel a stabilitási előírásoknak. Az általam kidolgozott másik két struktúrával, ahol a Lagrange szorzók becslésével adok egy kezdő értéket a kvadratikus programozásnak, valamint előre becsülöm az adott időpillanatban aktív kényszereket ezáltal zárt alakban megoldva a problémát, nagy mértékben csökkentettem a kvadratikus programozás iterációs számán. A futási idő csökkentése mellett, a struktúra megfelelt a stabilitási kritériumoknak. A javasolt megoldások hatékonyságát, MATLAB szimulációs környezetben kísérleti úton igazoltam.

2. fejezet

Szakirodalmi áttekintés

2.1. Szabályozási megoldások áttekintése

A szabályozási stratégiák fejlődése jelentős mértékben átalakította a járművek irányítási rendszerit, a PID irányítástól kezdve egészen a modern adaptív, és prediktív szabályozókig. A PID szabályzók, amelyek az arányos, integráló és deriváló jeleket használják a jármű stabilitásának biztosítására és az adott szabályozási feladat megoldására, kezdetben domináltak a járműiparban. A modell alapú és a gépi tanuláson alapuló algoritmusok csak később, a számítási kapacitás növekedésével kezdtek elterjedni.

A modell prediktív szabályozás kérdésköre már a hetvenes évektől kutatási terület azonban a hardveres erőforrások hiánya miatt túl számításigényes volt valós idejű alkalmazáshoz. Az idő előrehaladtával az iparban is népszerűvé vált, és már több mint 5800 sikeres ipari alkalmazása van különböző területeken [4]. A valós alkalmazásokhoz azonban egyre szigorúbb teljesítménykövetelmények és számos környezeti, valamint biztonsági kritériumok kerültek meghatározásra.

Ezek a kritériumok szükségessé tették a rendszerek pontosabb modellezését, és ezzel együtt a lineáris MPC algoritmusok kibővítését. Ezek a követelmények gyakran nemlineáris folyamatokkal írják le a környezetet, ami elvezetett a nemlineáris MPC (NMPC) kialakulásához és alkalmazásához a kilencvenes évek elején [5]. Az NMPC az MPC-hez hasonlóan olyan visszacsatolási szabályozási algoritmus, amely az optimális szabályozási problémák mozgó horizonton történő on-line megoldásán alapul, azonban képes kezelni az adott rendszer nemlineáris dinamikáját.

Az NMPC területén végzett későbbi kutatások több továbbfejlesztett változatot is vizsgáltak, mint például az előre lépő NMPC-t (asNMPC). [6] Ez a szabályozási módszer a pillanatnyi rendszerállapotok alapján becsüli meg a jövőbeli állapotokat, ezáltal csökkentve a számítási eljárások során keletkező késleltetést. Az asNMPC szabályzó a hagyományos NMPC szabályzó nominális stabilitási jellemzőit hordozza, miközben a zavarokra

és a modellbeli eltérésekre reagálva az optimális szabályozási probléma (OCP) parametrikus sajátosságait kihasználva, a valóságos optimális megoldást Nonlinear Programming (NLP) érzékenységi elvek alkalmazásával közelíti.

Az NMPC kialakulásával egyidőben kezdődtek el a robusztus modell prediktív szabályozáshoz kapcsolódó kutatások. A robusztus MPC diszkrét időben bizonytalan rendszerekre alkalmazható, és képes olyan rendszerek hatékony szabályozására, ahol a mintavételi idő többszöröse a számítási időnek. Egy ilyen robusztus modell prediktív algoritmus lehetővé teszi a járműrendszer stabilitásának megőrzését még akkor is, ha a következő irányítási művelet kiszámításához többszörös mintavételi idő szükséges. Ezen algoritmusok fejlesztése hozzájárul a járművek gyors manőverezési képességének és magas sebességű pályakövetésének javításához, miközben fenntartják a rendszer optimális és megbízható működését.[7]

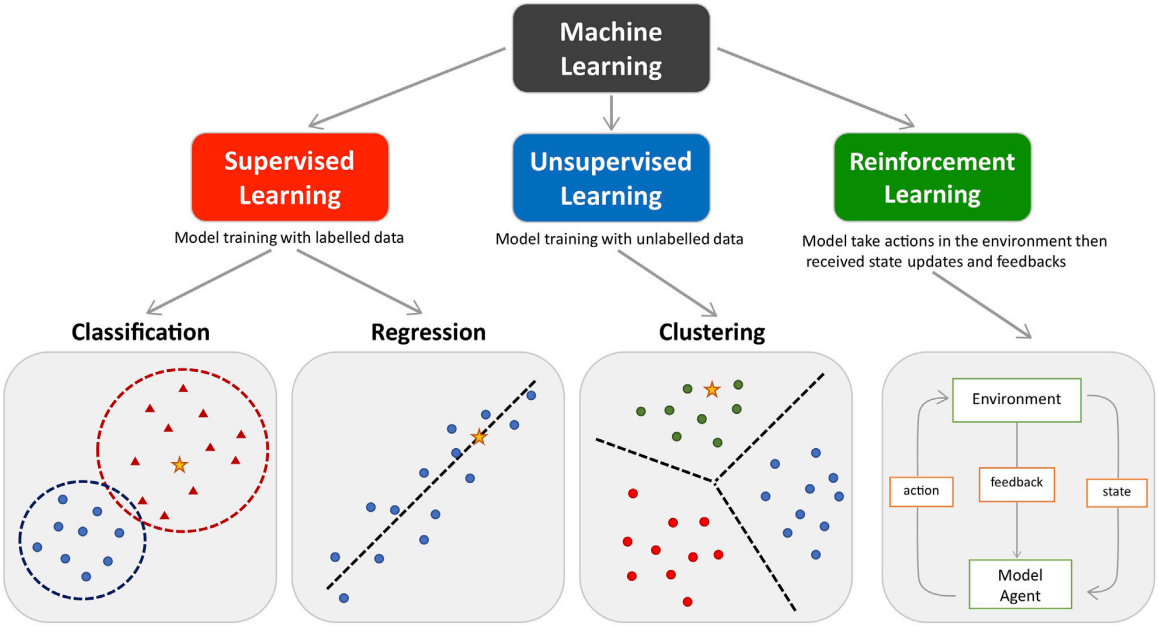
A 2000-es években felgyorsultak az MPC-hez kapcsolódó kutatások, amelyek eredményeként megjelentek a hibrid és explicit MPC-k. Az MPC esetében a beavatkozó jel meghatározásához kvadratikus programozás használata szükséges, ami gyakran idő- és számításigényes. Az explicit MPC ezt a problémát oldja meg, mivel teljesen kihagyja az online megoldókat. Az egyik ilyen megoldás az explicit MPC szabályozó erősítésének kifejezése "lookup-table" segítségével, amivel jelentősen felgyorsítható a számítási folyamat. [8]

A valós idejű alkalmazás mellett a pontosabb referencia követés valamint a stabilitás biztosítása is egyre nagyobb hangsúlyt kapott. Ennek elérésére a [9] cikkben egy Linear time-varying MPC-t (LTV-MPC) javasolnak. Az egyik előnye ennek a megközelítésnek a csökkentett számítási igény, ami a a nemlineáris rendszer aktuális működési pont körüli lineáris időfüggő modellezésével érhető el. Az LTV-MPC alkalmazása során a referencia jelkövetési pontosság is jelentősen javul, ami [10] alapján alacsonyabb RMSE 4.57 értékekben mutatkozik meg különböző tesztkörnyezetekben. A stabilitás biztosítása érdekében a módszer konvex kényszereket vezet be, amelyek a predikált állapotok és a beavatkozó jel közötti kvadratikus kapcsolatot korlátozzák. Amennyiben a rendszer dinamikája és a rendszermodell paraméterei is időben változnak, a Linear Parameter-Varying Model Predictive Control (LPV-MPC) alkalmazása szükséges [11]. Ebben a kutatásban [12] a módszer integrálja a változó jármű- és aktuátor-dinamikát a szabályozási stratégiába, így biztosítva a szabályozás robosztusságát sebességkövetési feladatokban és kezelve a környezeti és operatív bizonytalanságokat, mint például a tömeg- és az útviszonyok változásait. A cikkben bemutatott LPV-MPC technika előnye, hogy dinamikusan kompenzálja a mért zavaró hatásokat és a nem modellezett dinamikai bizonytalanságokat.

2.2. Gépi tanulási algoritmusok áttekintése

A gépi tanulási algoritmusok már az 1950-es évektől kezdve megjelentek, bár kezdetben még nem olyan fejlett formában, mint ahogyan ma ismerjük őket. Az egyik legelső sikeres alkalmazás Samuel [13] tanuló algoritmusa volt, amely képes volt a Checkers játékban saját magával játszva fejleszteni a teljesítményét, önállóan tanulva a játék során. Ez az alapötlet fontos mérföldkővé vált a mesterséges intelligencia területén, és ezzel megnyílt az út a gépi tanulási algoritmusok további fejlődése előtt.

Az 1950-es évek végén a Perceptron megalkotása [14] fektette le a mesterséges neurális hálózatok (Artificial Neural Networks, ANN) alapkövét. Ez volt az első sikeres neurális háló modell, amely képes volt különböző osztályozási feladatokat megoldani. Ebben az időszakban terjedt el a felügyelt tanulás, ahol a modell címkézett adatokból tanul. A felügyelt tanulási algoritmusok célja ahogy a 2.1 ábra is mutatja, hogy a bemenetek és a kimenetek közötti kapcsolatot megismerve képesek legyenek új adatokat is előre jelezni. Két fő alkalmazási területe a klasszifikáció, ahol a modell kategóriákba sorolja az új adatokat, valamint a regresszió, ahol folytonos értékek előrejelzése a cél.



2.1. ábra. A gépi tanulás három fő típusa. Forrás: [15]

A 1990-es évek elején új korszak kezdődött a gépi tanulásban: a korábbi ismeret alapú megközelítést felváltotta az adatvezérelt tanulás [16]. A gépi tanulás ezen formája lehetővé tette, hogy nagy mennyiségű adat elemzésével a modellek mélyebb mintázatokat fedezzenek fel, amelyek új lehetőségeket nyitottak meg az adatbányászat és a prediktív modellezés terén. Ekkor terjedt el a felügyelet nélküli tanulás, ahol a modellek már címkézetlen adatokból tanulnak, és rejtett mintázatokat, klasztereket ismernek fel az adatokban.

Az elmúlt évtizedekben a mélytanulás fejlődése is egyre nagyobb figyelmet kapott. Ez a neurális hálózati architektúra több rétegű neuronhálókat használ, amelyek képesek összetett mintázatok felismerésére és bonyolult problémák megoldására [17]. A mélytanulási modellek különösen jól alkalmazhatóak olyan feladatokra, mint a képfelismerés vagy a beszédfelismerés. A megerősítéses tanulás szintén egyre nagyobb teret nyer, különösen olyan komplex rendszerekben, ahol a dinamikus környezetben való adaptív döntéshozatal kulcsfontosságú.

A gépi tanulás különböző formái, mint a felügyelt, felügyelet nélküli és megerősítéses tanulás, alapvető szerepet töltenek be az autonóm járművek irányításában, mivel lehetővé teszik a valós idejű adatfeldolgozást és a gyors döntéshozatalt. A közlekedés és járművezetés általános megoldásának komplexitása miatt a gépi tanulás ígéretes irányt jelent, hiszen képes kezelni a bonyolult és változó környezetből származó adatokat. A mesterséges intelligencia és az intelligens ágensmodellek ezen a területen egyre fontosabbá válnak, mivel az autonóm rendszereknek képesnek kell lenniük a környezet pontos modellezésére, az adatok feldolgozására és a logikai alapú következtetések levonására. Ezek a képességek biztosítják a járművek számára, hogy a dinamikusan változó környezetben biztonságosan és hatékonyan navigáljanak, miközben valós időben megbízható döntéseket hoznak. Ebből adódóan ennek a TDK dolgozatnak az egyik fő erénye, hogy a klasszikus módszerek és ML ötvözésével, mindkét megközelítés előnyeit kihasználom.

Alkalmazásukat tekintve az autonóm járművek és robotok számára kritikus funkciókat biztosítanak, mint például a navigáció, az önálló útvonaltervezés, az útvonal követése és a kommunikáció [18] [19]. Ezek a rendszerek képesek valós időben döntéseket hozni, adaptálódni a dinamikus környezethez, és hatékonyan együttműködni más rendszerekkel és emberekkel.

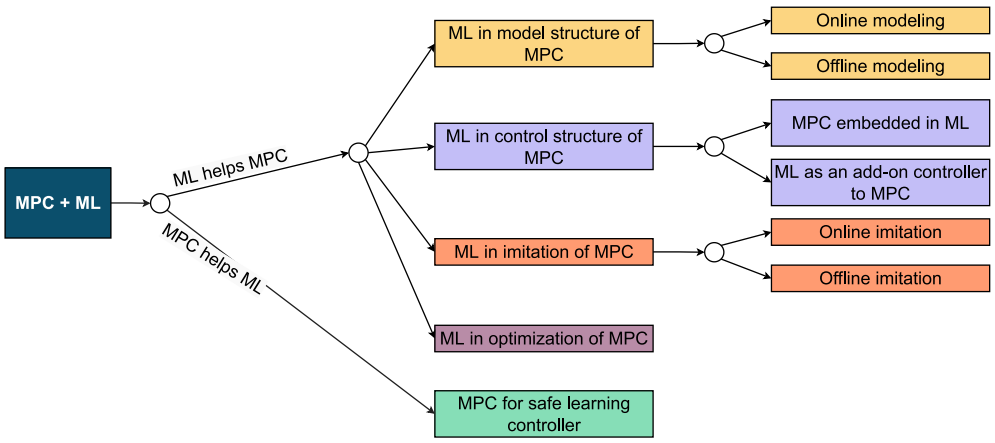
Az irányítástechnika területén végzett kutatások a járművek alacsonyabb szintű automatizáltságára korlátozódtak, elsősorban a navigációs illetve pályakövetés problémákra és előre programozott reaktív viselkedésekre összpontosítva. Az intelligens ágens alapú mesterséges intelligencia modellekkel kapcsolatos kutatások viszont lehetőséget kínálnak a járművek magasabb szintű döntéshozatali képességeinek fejlesztésére. Ezek az ágensarchitektúrák összetett döntéshozatali folyamatokat tesznek lehetővé, amelyek jelentősége túlmutat a technikai problémák egyszerű megoldásán. Az ilyen típusú kutatások célja, hogy elősegítsék az intelligens döntéshozatali módszerek gyakorlati alkalmazását az autonóm járművek terén.[20]

Az autonóm járművek irányításában az eddig tárgyalt lehetőségeken felül a mélytanulás és megerősítéses tanulás módszerei jelentős fejlődést mutatnak. A [21] cikkben bemutatott a módszer eltér a hagyományos végponttól végpontig terjedő tanulási módszertől, mivel a látásalapú oldalirányú irányítási algoritmust két modulra bontja: egy érzékelési modulra és egy irányítási modulra. Az érzékelési modul egy többfeladatos tanulási (Multi-Task Learning) neurális hálózaton alapul, amely először a sofőr látószögéből készült képet

veszi bemenetként, majd előrejelzi a pálya jellemzőit. Az irányítási modul, amely megerősítéses tanuláson alapul, ezután ezek alapján hoz döntést az irányításról. A megerősítéses tanulási módszer lehetővé teszi, hogy az irányítási modul az optimalizált vagy majdnem optimális irányítási parancsot generálja, amely az előre tervezett pályát követi. Ez a megközelítés kiemelkedően teljesít a hagyományos lineáris kvadratikus szabályozókkal (LQR) és a modell prediktív kontrollokkal (MPC) kombinálva különböző pályákon.

2.3. ML-MPC megoldások áttekintése

A gépi tanulás (ML) és MPC kombinációja már a kétezres évek elején elkezdődött, azóta azonban szignifikánsan nőtt. Ez az integráció, amelyet általában tanuló MPC-nek (learning MPC) [22] vagy adatvezérelt (data-driven) MPC-nek [23] neveznek, három fő kategóriába sorolható amit a 2.2 ábra is jól szemléltet : ML a modellezésben, ML a számítási igény csökkentésében, és MPC a biztonságos tanulásban (safe-ML).



2.2. ábra. MPC és machine learning kombinációinak a lehetőségei. Forrás: [24]

Amikor a gépi tanulást az MPC modellstruktúrájába integráljuk, a modellezés offline vagy online valósulhat meg. Offline esetben az ML modell implementációja és tanítása mért adatok alapján történik, ami széles körben elterjedt a kutatások és ipari alkalmazások terén. Az egyik ilyen alkalmazás amit a [25] mutat be, egy autonóm járművek kormányzás szabályozására kifejlesztett neurális hálózati modell tervezése. Ez a modell képes kezelni a nemlineáris kormányzási dinamikát és az ezzel kapcsolatos bizonytalanságokat. Vissza térő neurális hálózatot (RNN) alkalmazva pontosabban előrejelzi a kormányzási dinamikát, és integrálható egy NMPC szabályozóba. Egy másik megközelítés a vision-based MPC. A [26] alapján kifejlesztett rendszer konvolúciós neurális hálózatokat (CNN) alkalmaz a költségfüggvények predikciójára bemeneti videók alapján, amelyeket közvetlenül integrálnak az MPC algoritmusba online pályatervezés céljából. Ezt a módszert egy 1:5 méretarányú autonóm járművel validálták, amely képes nagy sebességű, agresszív

vezetési manőverek végrehajtására. Az eljárás egyetlen monokuláris kamerát és egy mély konvolúciós neurális hálózatot használ a jármű előtti pálya költségtérképének előrejelzéséhez, amelyet az MPC valós időben tud felhasználni az opimalizációhoz. Az integráció Gauss folyamatok (lásd pl. [27] segítségével is megvalósítható. A [26] kutatás alapján kifejlesztett rendszer a Gauss-folyamat modelljeit használja a szabályozáshoz szükséges modell felállítására. Ebben az esetben a Gauss-folyamatok előnye, hogy a modell minőségére vonatkozó bizonytalanságot is figyelembe veszik amelyeket az MPC algoritmus optimalizációs folyamatában közvetlenül használnak fel.

Az online tanítás ugyan kevésbé elterjedt az ipari alkalmazásokban, de bizonyos területeken, például az autóipari irányítástechnikai rendszerekben, már találhatók példák az alkalmazására. A [28] kutatás egy online finomhangolási keretrendszert javasol lineáris paraméter-változó állapottér modellekhez, amely mesterséges neurális hálózatokat használ zárt hurkú adatok alapján. Az eredmények igazolják, hogy az online transzfer tanulás és az adaptív MPC kombinációja javítja a teljesítményt. Egy másik megközelítés, amelyet a [29] kutatás mutat be, a gépi tanuláson alapuló adatalapú abroncsmodellezés. Ez a módszer hatékonyan alkalmazható az autonóm járművek modell prediktív szabályozásában, csökkentve a számítási költségeket és növelve a rendszer pontosságát.

A szerkezetét tekintve a gépi tanulás hozzájárulhat az optimalizáció felgyorsításához is. A [3] kutatásban egy nemlineáris MPC-t gyorsítanak neurális háló segítségével. Az NMPC inicializásánál egy Warm Start típusú becslést adnak az NMPC számára, ezzel gyorsítva az optimalizációt. Ehhez hasonlóan a [30] kutatás, egy Hidden Markov Model alapú megközelítést alkalmaz az optimalizáció gyorsítására. Ebben az esetben a HMM az aktuális aktív kényszereket becsüli meg, és az Active Set Method segítségével oldák meg az optimalizációs problémát.

Ezen túl a gépi tanulás szimulálhatja az MPC működését. Ez a kutatás [31] egy hierarchikus megközelítést alkalmaz, ahol az úgynevezett Imitation learning segítségével egy neurális hálózat tanulja meg az MPC által generált hosszútávú optimális vezetési stratégiákat, míg egy rövidtávú optimalizációs réteg garantálja a biztonságot és megvalósíthatóságot valós időben.

A másik oldalról megközelítve az MPC alkalmazásával növelhető a gépi tanulási modellek robusztussága amivel elérhető az úgynevezett biztonságos gépi tanulás. [32] Ez viszont egyelőre csak szimulációban valósult meg. Ezeknek az algoritmusoknak a lényege ahogy a [33] kutatás is bemutatja, hogy egy olyan probabilisztikus model-predictív biztonsági tanúsítási keretet (PMPSC) hoznak létre, amely lehetővé teszi, hogy gépi tanulás alapú kontrollerek biztonságosak maradjanak.

3. fejezet

Járműdinamikai modellek

A járműdinamikai modellezés olyan matematikai keretrendszert kínál, amely lehetővé teszi a mérnökök számára, hogy kvantitatív módon közelítsék meg a járművek mozgásállapot-változásait, változatos vezetési körülmények között. Ezen modellek alapvető célja a járműfizika precíz reprezentációja, beleértve a tömegeloszlást, a hajtási erőket és a torziós dinamikát, melyeket matematikai egyenletekkel írnak le. A nemlineáris, high-fidelity modellek mélyebb megértést nyújtanak a jármű mozgásának alapkérdéseiről, a járművek tömegdinamikájáról, míg a redukált modellrendszerek, mint amilyen az egynyomvonalú járműmodell (bicikli modell), az oldalirányú stabilitás és a kormányzás alapvető összefüggéseit hivatottak szemléltetni.

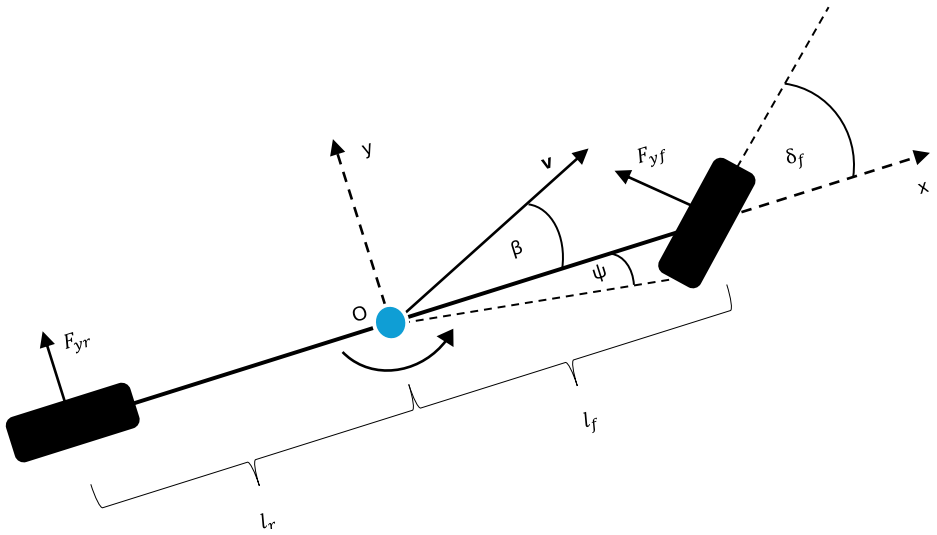
Ezen túlmenően, speciális modellek, mint a Pacejka formula a jármű és az út közötti egyedi interakciókra fókuszálnak, mint például a gumiabroncsok viselkedésére,különböző oldalkúszási szögek és terhelések mellett. Ezek a modellek kulcsfontosságúak az aktív vezetéstámogató rendszerek és az automatizált vezetési technológiák fejlesztésében, mivel képesek a jármű pontos pályakövetési képességét modellezni és előre jelezni. A fejezet további részeiben bemutatott modelleket [34] alapján lettek implementálva.

3.1. Dinamikai bicikli modell

Magasabb járműsebességek esetén a kinematikai modell helyett dinamikai modellt szükséges alkalmazni, amely az oldalkúszási szögeket és az ezekből eredő oldalirányú erőket is figyelembe veszi, így pontosabban írja le a jármű oldalirányú mozgását. Ilyen sebességeknél nem feltételezhetjük, hogy a kerekek sebessége mindig megegyezik a kerék orientációjának irányával, mivel a kerekek és az útfelület között kialakuló oldalirányú erők jelentős oldalkúszási szögeket eredményeznek. Az oldalkúszási szög—vagyis a jármű tényleges mozgásának és a kerekek orientációjának irányeltérése—jelentős hatással van a jármű stabilitására és irányíthatóságára.

A bicikli modell két szabadságfokkal írható le. Ahogy a 3.1 ábrán látható, a két szabadságfok a jármű y-al jelölt oldalirányú pozícióját, és ψ-vel jelölt legyezési szögét

reprezentálja. Az oldalirányú pozíció a jármű oldalirányú tengelye mentén mérhető az O referenciaponttól, ami maga a jármű tömegközéppontja. A legyezési szög a globális X tengelyhez képest mérhető. A jármű hosszirányú sebessége a tömegközéppontnál V_x.



3.1. ábra. A dinamikai bicikli modell geometriája és a laterális hiba.

Az út dőlésszögének befolyását figyelmen kívül hagyva és felhasználva Newton II. törvényét a mozgásegyenletek felírásához az y tengely mentén, a következő egyenlet kapható:

$$ma_y = F_{yf} + F_{yr}, \tag{3.1}$$

ahol a_y a laterális gyorsulás a jármű tömegközéppontjánál az y tengely irányában, F_{yf} és F_{yr} pedig az első és hátsó kerekek oldalirányú erői. A jármű oldalirányú gyorsulása a következőképpen fejezhető ki:

$$a_y = \ddot{y} + V_x\dot{\psi}, \tag{3.2}$$

ahol $\dot{y}$ az y irányú sebességváltozás, V_x a jármű x tengely menti sebessége, és $\dot{\psi}$ a jármű legyezési szögsebessége. Ezt követően a jármű oldalirányú mozgásegyenlete felírható:

$$m(\ddot{y} + V_x\dot{\psi}) = F_{yf} + F_{yr}, \tag{3.3}$$

amely az oldalirányú erők és a gyorsulás kapcsolatát írja le. Végül a jármű legyezési szögsebesség dinamikájához tartozó nyomatékegyensúly a következő:

$$I_z\ddot{\psi} = \ell_f F_{yf} - \ell_r F_{yr}, \tag{3.4}$$

ahol I_z a jármű z tengely körüli tehetetlenségi nyomatéka, ℓ_f és ℓ_r a jármű első, illetve hátsó kerekeinek a tömegközépponttól mért távolsága.

A járművek oldalirányú dinamikájának modellezése során elengedhetetlen megérteni

a kerékerők és a kormányzási dinamika közötti kapcsolatot. A járműre ható oldalirányú kerékerők F_{yf} és F_{yr} modellezésének következő lépéseként megállapítható az oldalkúszási szög és az oldalirányú erő közötti összefüggés. Az első és hátsó kerék oldalkúszási szöge:

$$\alpha_f = \delta - \theta_{yf}, \tag{3.5}$$

$$\alpha_r = -\theta_{yr}, \tag{3.6}$$

ahol δ az első kerék kormányszöge és θ_{yf} az első kerék θ_{yr} pedig a hátsó kerék sebességvektorának a szöge. Kis oldalkúszási szögek esetén a gumiabroncsra ható oldalirányú erő egyenesen arányos az oldalkúszási szöggel, ezért közelítőleg alkalmazható lineáris összefüggés az első és hátsó kerekekre ható oldalirányú erő felírására:

$$F_{yf} = 2C_f(\delta - \theta_{yf}), \tag{3.7}$$

$$F_{yr} = 2C_r(-\theta_{yr}), \tag{3.8}$$

ahol C_f és C_r az első, illetve hátsó kerekek kanyarmerevségének arányossági állandói. A sebességvektor szöge a következő kicsi szögek közelítése alapján számolható:

$$\theta_{yf} = \frac{\dot{y} + \ell_f\dot{\psi}}{V_x}, \tag{3.9}$$

$$\theta_{yr} = \frac{\dot{y} - \ell_r\dot{\psi}}{V_x}. \tag{3.10}$$

A dolgozatban is használt LPV (Lineáris Paraméter-Változó) modell olyan állapottérmodell, amelyben a rendszer paraméterei valamilyen mérhető változó (jelen esetben a hosszirányú sebesség V_x, függvényében módosulnak. Ez a megközelítés lehetővé teszi, hogy a nemlineáris rendszer viselkedését egy lineáris modell segítségével írjuk le, amely a paraméterek változásával alkalmazkodik a dinamikához.

A linearizált rendszer így az alábbi formában, LPV modellként írható fel, ahol a mátrixok a sebesség függvényében változnak:

$$\dot{x} = A(V_x)x + B\delta. \tag{3.11}$$

A rendszer linearizálást követően, a 3.3 3.4 3.9 3.10 egyenleteket felhasználva az állapottér modell a következőképpen írható fel:

$$\frac{d}{dt}\begin{bmatrix} y \\ \dot{y} \\ \psi \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -\frac{2C_f+2C_r}{mV_x} & 0 & \frac{2C_f\ell_f-2C_r\ell_r}{mV_x} \\ 0 & 0 & 0 & 1 \\ 0 & \frac{2C_f\ell_f-2C_r\ell_r}{I_zV_x} & 0 & -\frac{2C_f\ell_f^2+2C_r\ell_r^2}{I_zV_x} \end{bmatrix}\begin{bmatrix} y \\ \dot{y} \\ \psi \\ \dot{\psi} \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{2C_f}{m} \\ 0 \\ \frac{2\ell_fC_f}{I_z} \end{bmatrix}\delta. \tag{3.12}$$

3.2. Dinamikai hiba-alapú bicikli modell

A dolgozatban a szabályozók tervezéséhez azonban a hagyományos dinamikai bicikli modell egyik változatát, a dinamikai hibaalapú biciklimodellt használtam. Ebben az esetben a 3.12 képletben látható állapotváltozók helyett ezeknek a hibái szerepelnek, amelyek a kívánt és a tényleges állapotok közötti eltéréseket mutatják. Ennek a megközelítésnek az előnye, hogy a szimmuláció során a hibaszámítási lépés kihagyható, ezzel kiküszöbölve a numerikus deriválásból eredő pontatlanságot, és egyszerűsíti a szabályozó felírását.

Ennek elérése érdekében az előző fejezetben tárgyalt hagyományos bicikli modellt újra definiáljuk, és bevezetjük a jármű tömegközéppontjának az út középvonalától való laterális elmozdulását jelző e_1 állapotot (változót), valamint a jármű longitudinális tengelyének az út irányához képesti orientációs eltérését reprezentáló e_2 állapotot. A modell felírása során azt feltételeztük, hogy a hosszirányú sebesség állandó és a kanyarodási sugár nagy, így a kis szögek közelítése érvényes. A jármű által követni kívánt irányváltozás sebességét a következő módon határozzuk meg:

$$\dot{\psi}_{des} = \frac{V_x}{R} y, \tag{3.13}$$

amiből következik a jármű kívánt centripetális gyorsulása:

$$\frac{V_x^2}{R} = V_x \dot{\psi}_{des}. \tag{3.14}$$

Az $\dot{e}_1$, $\ddot{e}_1$ és e_2 hibaváltozók a következő módon definiálhatók:

$$\dot{e}_1 = \left(\dot{y} + V_x \dot{\psi}\right) - \frac{V_x^2}{R} = \dot{y} + V_x \left(\dot{\psi} - \dot{\psi}_{\text{des}}\right), \tag{3.15}$$

$$e_2 = \psi - \psi_{\text{des}}, \tag{3.16}$$

$$\dot{e}_2 = \dot{y} + V_x \left(\psi - \psi_{\text{des}}\right). \tag{3.17}$$

Amennyiben a V_x sebesség nem állandó, akkor a kapott modell nemlineáris és időfüggő tulajdonságokkal rendelkezik. Közelítés képpen, a longitudinális sebességet konstansként kezeljük, így kapva egy Lineáris Időinvariáns (LTI) modellt. Ha a longitudinális sebesség változik, akkor az LTI modellt Lineáris Paraméter Változós (LPV) modell váltja fel, ahol a sebesség mint időfüggő paraméter szerepel. A (3.15-3.17) egyenleteket behelyettesítve a (3.3) és (3.4) egyenletekbe, felírható állapottér alakban.

$$\frac{d}{dt}\begin{bmatrix} e_1 \\ \dot{e}_1 \\ e_2 \\ \dot{e}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -\frac{2C_f+2C_r}{mV_x} & \frac{2C_f+2C_r}{mV_x} & --\frac{2C_f l_f+2C_r l_r}{mV_x} \\ 0 & 0 & 0 & 1 \\ 0 & -\frac{2C_f l_f-2C_r l_r}{I_z V_x} & \frac{2C_f l_f-2C_r l_r}{I_z} & -\frac{2C_f l_f^2+2C_r l_r^2}{I_z V_x} \end{bmatrix} \begin{bmatrix} e_1 \\ \dot{e}_1 \\ e_2 \\ \dot{e}_2 \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{2C_f}{m} \\ 0 \\ \frac{2C_f l_f}{I_z} \end{bmatrix} \delta + \begin{bmatrix} 0 \\ -\frac{2C_f l_f+2C_r l_r}{mV_x} - V_x \\ 0 \\ -\frac{2C_f l_f^2+2C_r l_r^2}{I_z V_x} \end{bmatrix} \dot{\psi}_{des} \tag{3.18}$$

4. fejezet

Algoritmusok

4.1. Modell Prediktív Szabályozás

A Modell Prediktív Szabályozás egy fejlett szabályozási módszer, amely a jövőbeli viselkedés előrejelzésén alapulva irányítja a dinamikus rendszereket.

Az MPC lényege, hogy egy adott időpontban, a rendszer jelenlegi állapotától kiindulva az irányítási algoritmus előrejelzi a rendszer jövőbeli viselkedését egy adott időhorizonton belül. Ez a predikció egy belső modell segítségével történik, amely leírja, hogy a rendszer hogyan reagál különböző beavatkozó jelek hatására. A modell által generált előrejelzés alapján az MPC egy optimalizációs problémát old meg annak érdekében, hogy megtalálja az optimális beavatkozó jel-sorozatot, amely minimalizálja a költségfüggvényt. A költségfüggvény általában magában foglalja a kívánt trajektória és a prediktált kimenet közötti eltérést, valamint a beavatkozó jelek használatával járó költségeket.

4.1.1. Állapottér-modell

Az MPC tervezéshez a dolgozatban egy single-input és single-output rendszert alkalmaztam. Egy általános rendszer leírása a következő:

$$x_m(k+1) = A_m x_m(k) + B_m u(k), \tag{4.1}$$

$$y(k) = C_m x_m(k). \tag{4.2}$$

Az integráló tulajdonság biztosítása érdekében az állapottér modellt az állapot változásra írjuk fel:

$$\Delta x_m(k+1) = x_m(k+1) - x_m(k); \quad \Delta x_m(k) = x_m(k) - x_m(k-1), \tag{4.3}$$

$$\Delta u(k) = u(k) - u(k-1). \tag{4.4}$$

A változásokat bevezetve az állapot-egyenletbe az alábbi alakra jutunk:

$$\Delta x_m(k+1) = A_m \Delta x_m(k) + B_m \Delta u(k). \tag{4.5}$$

ahol az új állapotvektor az állapotok megváltozásának és a kimenet összekapcsolásából adódik:

$$x(k) = \begin{bmatrix} \Delta x_m(k)^T \\ y(k)^T \end{bmatrix}^T, \tag{4.6}$$

$$y(k+1) - y(k) = C_m(x_m(k+1) - x_m(k)) = C_m A_m \Delta x_m(k) + C_m B_m \Delta u(k). \tag{4.7}$$

A 4.1.1 és 4.1.1 egyenletek egyesítésével, a következő állapottér modellt kapjuk:

$$\begin{bmatrix} \Delta x_m(k+1) \\ y(k+1) \end{bmatrix} = \begin{bmatrix} A_m & 0 \\ C_m A_m & 1 \end{bmatrix} \begin{bmatrix} \Delta x_m(k) \\ y(k) \end{bmatrix} + \begin{bmatrix} B_m \\ C_m B_m \end{bmatrix} \Delta u(k), \tag{4.8}$$

$$y(k) = \begin{bmatrix} o_m & 1 \end{bmatrix} \begin{bmatrix} \Delta x_m(k) \\ y(k), \end{bmatrix} \tag{4.9}$$

ahol o_m egy olyan nulla vektor, melynek mérete megegyezik az állapotvektor x_m méretével. Az (A, B, C) mátrixokat kiterjesztett (augmented) modellnek nevezzük, amit a prediktív szabályozás tervezéséhez használunk.

Ezek alapján a kibővített állapotegyenlet az alábbi egyenlettel írható fel:

$$x(k+1) = Ax(k) + B\Delta u(k), \tag{4.10}$$

$$y(k) = Cx(k). \tag{4.11}$$

Az állapotegyenlet alapján, a k-adik állapotból a k+1; k+2; ... ; k + N_p predikciókat, az alábbi iterációval kapjuk, ahol N_p a predikciós horizont hossza:

$$\mathbf{x}(k+i|k) = A^i x(k) + \sum_{j=0}^{i-1} A^{i-1-j} B \Delta u(k+j). \tag{4.12}$$

Ez felírható mátrixegyenlet formában is:

$$\begin{bmatrix} x(k+1|k) \\ x(k+2|k) \\ \vdots \\ x(k+N_p|k) \end{bmatrix} = \begin{bmatrix} A \\ A^2 \\ \vdots \\ A^{N_p} \end{bmatrix} x(k) + \begin{bmatrix} B & 0 & \cdots & 0 \\ AB & B & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ A^{N_p-1}B & A^{N_p-2}B & \cdots & B \end{bmatrix} \begin{bmatrix} \Delta u(k) \\ \Delta u(k+1) \\ \vdots \\ \Delta u(k+N_c-1) \end{bmatrix}. \tag{4.13}$$

A predikciós horizontban a kimenetek a következő mátrixegyenlettel számíthatók ki ahol N_c a kontroll horizont hossza:

$$\begin{bmatrix} y(k+1|k) \\ y(k+2|k) \\ \vdots \\ y(k+N_p|k) \end{bmatrix} = \begin{bmatrix} CA \\ CA^2 \\ \vdots \\ CA^{N_p} \end{bmatrix} x(k) + \begin{bmatrix} CB & 0 & \cdots & 0 \\ CAB & CB & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ CA^{N_p-1}B & CA^{N_p-2}B & \cdots & CB \end{bmatrix} \begin{bmatrix} \Delta u(k) \\ \Delta u(k+1) \\ \vdots \\ \Delta u(k+N_c-1) \end{bmatrix}. \tag{4.14}$$

A fenti összefüggés felírható kompaktabb formában:

$$Y = F(C,A)x(k) + \Phi(C,B)\Delta u, \tag{4.15}$$

ahol $\Phi(C,B)$ egy Töeplitz-mátrix, és F(C,A) = O(C,A)A ahol $O(C,A)$ a megfigyelhetőségi mátrix soraiból képzett mátrix. Egyszerűsített jelölésekkel:

$$Y = Fx(k) + \Phi\Delta u. \tag{4.16}$$

4.2. MPC költségfüggvény és minimalizálása

Az MPC célja, hogy minimalizálja a referencia jel és a prediktált kimenet közötti eltérést, valamint az irányítójel nagyságát és változását. Ennek a Δu irányítójelnek a meghatározása az alábbi kvadratikus költségfüggvény minimalizálása alapján történik:

$$\begin{aligned} J(\Delta U) &= (R_s - Y)^T(R_s - Y) + \Delta U^T R \Delta U \\ \text{feltéve, hogy} \quad & M\Delta U \leq \gamma, \end{aligned} \tag{4.17}$$

ahol R_s a referencia jel, Y a kimenet, ΔU a beavatkozó jel változása, és R pdig a súlyozó mátrixa. Azonban a hiba-alapú modell esetén, mivel a referencia és a kimenet közötti eltérés mint állapot szerepel a modellben, $R_s - Y = 0$ lesz.

A költségfüggvény minimumának a feltétele, hogy az első deriváltja nulla a második deriváltja pedig pozitív definit legyen:

$$\frac{\partial J}{\partial \Delta U} = 0, \tag{4.18}$$

$$\det\left(\frac{\partial^2 J}{\partial \Delta U^2}\right) > 0. \tag{4.19}$$

Az MPC tervezése során többféle kényszert lehet figyelembe venni, például a beavatkozó jelre, annak változására, valamint a rendszer kimenetére és állapotváltozóira vonatkozó kényszereket. A kimeneti és állapotkényszereket gyakran „puha" kényszerekként kell kezelni, mivel szigorú betartásuk esetén ellentmondásosak lehetnek a beavatkozó jel kényszereivel, ami a rendszer stabilitását veszélyeztetheti. Ebből adódóan, ebben a dolgozatban a beavatkozó jelre és annak gradiensére vonatkozó kényszereket alkalmaztam, mivel ezek kritikusak a rendszer működése szempontjából, viszont elkerülik a lehetséges ellentmondásokat a szabályozási feladatban. A beavatkozó jelre és annak gradiensére vonatkozó kényszerek a teljes kontroll horizontra nézve a következők:

$$u_{min} \leq u(k+i) \leq u_{max}, \quad \forall k \in N_c, \tag{4.20}$$

$$\Delta u_{min} \leq \Delta u(k+i) \leq \Delta u_{max}, \quad \forall k \in N_c, \tag{4.21}$$

ahol u_{min} u_{max} a beavatkozó jel, Δu_{min} Δu_{max} a beavatkozó jel változásának az alsó és felső korlátai. Ezeket a kényszereket két egyenlőtlenséggel lehet kifejezni ami mátrix alakban a következő:

$$\begin{bmatrix} -I \\ I \end{bmatrix} \Delta U \leq \begin{bmatrix} -\Delta U_{\min} \\ \Delta U_{\max} \end{bmatrix}, \tag{4.22}$$

A kényszerek definiálása után az MPC költségfüggvényének minimum értékének megtalálásához a következő egyenlet deriválása szükséges:

$$\begin{aligned} J(\Delta U) = {} & (R_s - Fx(k_i))^T (R_s - Fx(k_i)) \\ & - 2\Delta U^T \Phi^T (R_s - Fx(k_i)) \\ & + \Delta U^T (\Phi^T \Phi + \bar{R}) \Delta U, \end{aligned} \tag{4.23}$$

$$\text{feltéve, hogy} \quad \begin{bmatrix} M_1 \\ M_2 \end{bmatrix} \Delta U \leq \begin{bmatrix} N_1 \\ N_2 \end{bmatrix}, \tag{4.24}$$

ahol

$$M_1 = \begin{bmatrix} -C_2 \\ C_2 \end{bmatrix}, \quad N_1 = \begin{bmatrix} -U_{\min} + C_1 u(k_i - 1) \\ U_{\max} - C_1 u(k_i - 1) \end{bmatrix}, \tag{4.25}$$

$$M_2 = \begin{bmatrix} -I \\ I \end{bmatrix}, \quad N_2 = \begin{bmatrix} -\Delta U_{\min} \\ \Delta U_{\max} \end{bmatrix}. \tag{4.26}$$

A $\Phi^T\Phi + \bar{R}$ a Hessian-mátrix, amely pozitív definitnek van feltételezve. Mivel a költségfüggvény J kvadratikus, és a kényszerek lineáris egyenlőtlenségek, a probléma megoldása egy standard kvadratikus programozási feladattá válik. Az egyenlőtlenségi kényszerek felírhatók kompaktabb alakban is:

$$M\Delta U \leq \gamma, \tag{4.27}$$

ahol M egy mátrix amely a kényszereket, γ pedig egy vektor amely a megengedett extremális értékeket tartalmazza. A szabályozási feladat megoldása két részből tevődik össze. A globális megoldás, valamint a kényszerek hatását figyelembe vevő tag.

4.2.1. Globális megoldás

Első lépésként a globális megoldás kiszámítása szükséges. A 4.23 egyenletből kiindulva, elvégezve a szorzásokat és az összevonásokat követően a következő egyenletet kapjuk:

$$\frac{\partial J}{\partial \Delta u} = -2\Phi_k^T(R_s - F_k u(k)) + 2(\Phi_k^T\Phi_k + R)\Delta u = 0. \tag{4.28}$$

Megoldva az egyenletet, megkapjuk a keresett optimális irányítójelet a teljes kontroll horizontra:

$$\Delta u_{opt} = (\Phi^T\Phi + R)^{-1}\Phi^T(R_s - Fx(k)). \tag{4.29}$$

Ezt kifejtve, a kényszerek nélküli (globális) eset megoldása az alábbiak szerint alakul:

$$\begin{aligned} \Delta u_{opt} &= -(\Phi^T\Phi + R)^{-1}\Phi^T F x(k) + (\Phi^T\Phi + R)^{-1}\Phi^T R_s \\ &= -K_{MPC}x(k) + K_r R_s, \end{aligned} \tag{4.30}$$

ahol K_{MPC} a visszacsatolási erősítési mátrix, melynek utolsó oszlopa megegyezik a K_r vekorral.

Az MPC során felhasznált "receeding-horizon" megközelítésnek köszönhetően feltétel nélküli esetben elegendő a K_{MPC} az állapot-visszacsatolási erősítési mátrix első sorának megtartása:

$$K_{MPC} = \begin{bmatrix} 1 & 0 & .... & 0 \end{bmatrix} (\Phi^T\Phi + R)^{-1}\Phi^T F x(k). \tag{4.31}$$

4.2.2. Kényszerek figyelembe vétele

A kvadratikus programozás lehetővé teszi, hogy a rendszerek optimális irányítása kényszerek mellett is megoldható legyen. Ezek az optimalizálási problémák a költségfüggvény minimalizálására irányulnak, azonban ez sokszor nem elegendő, mivel valós alkalmazáskor gyakran figyelembe kell venni az irányításra vonatkozó fizikai vagy matematikai kényszereket is. Az döntési változó x, és a költségfüggvény a következőképpen írható fel:

$$\begin{aligned} J &= \frac{1}{2}x^T Ex + x^T F, \\ \text{feltéve, hogy} \quad & Mx \leq \gamma. \end{aligned} \tag{4.32}$$

ahol E a kvadratikus, F pedig a lineáris tagot súlyozza. Feltételezhető, hogy E egy szimmetrikus és pozitív definit mátrix. A cél annak az x értéknek a megtalálása, amely a költségfüggvényt minimalizálja úgy, hogy az megfeleljen az egyenlőségi kényszereknek. Az egyenlőségi kényszerek esetében $Mx = \gamma$ feltétel teljesül. Az optimalizálás eléréséhez a Lagrange-féle kifejezés használható:

$$J = \frac{1}{2}x^T Ex + x^T F + \lambda^T (Mx - \gamma), \tag{4.33}$$

ahol λ a Lagrange-multiplikátorokat jelöli. Amikor az egyenlőség-kényszer teljesül, a fenti kifejezés értéke megegyezik az eredeti költségfüggvénnyel (4.32). A minimalizálási probléma megoldása során x és λ parciális deriváltjait nullává téve a következő egyenletrendszerre vezet:

$$\frac{\partial J}{\partial x} = Ex + F + M^T \lambda = 0, \tag{4.34}$$

$$\frac{\partial J}{\partial \lambda} = Mx - \gamma = 0. \tag{4.35}$$

Ez az egyenletrendszer biztosítja, hogy az optimalizált x értékek és a λ Lagrange-szorzók megfeleljenek a kényszereknek, miközben a költségfüggvényt minimalizáljuk.

Az optimális λ és x értékek a következő lineáris egyenletek segítségével határozhatók meg:

$$\lambda = -(ME^{-1}M^T)^{-1}(\gamma + ME^{-1}F), \tag{4.36}$$

$$x = -E^{-1}(M^T \lambda + F). \tag{4.37}$$

A 4.37 egyenletbe behelyettesítve 4.36-t, az összefüggés két tagra bontható:

$$x = -E^{-1}F - E^{-1}M^T \lambda = x^0 - E^{-1}M^T \lambda, \tag{4.38}$$

ahol az első tag, x^0 a globális optimális megoldás amely az eredeti költségfüggvény minimális értékét adja kényszer nélkül, míg a második tag tartalmazza az optimális egyenlőség-kényszerekhez tartozó megoldást.

Az optimális kényszereket teljesítő megoldáshoz az egyenlőségi kényszereknek lineárisan függetlennek kell lenniük az optimalizálandó változóktól. Amennyiben a rendszer fizikai korlátaiból adódóan az optimalizálási feladat egyenlőtlenségi kényszereket is tartalmaz, ezeket figyelembe kell venni a megoldás során. Ezeknek a kezelése a Kuhn-Tucker feltételek alkalmazásával érhető el.

4.2.3. Egyenlőtlenségi kényszerek megoldása Kuhn-Tucker feltételekkel

Ha egyenlőtlenségi kényszerekkel minimalizálunk, a kényszerek száma nagyobb lehet az optimalizálandó változókénál. Az egyenlőtlenségi kényszer lehetnek aktívak vagy inaktívak. Egy egyenlőtlenségi kényszer akkor aktív, ha $Mx = \gamma$, és inaktív, ha $Mx < \gamma$. Az aktív és inaktív kényszerek a Kuhn-Tucker feltételek alapján, a Lagrange-multiplikátorok segítségével határozhatók meg.

A Kuhn-Tucker feltételek, amelyek biztosítják az optimalizálási feladat megoldását, a következők:

$$Ex + F + M^T\lambda = 0, \tag{4.39}$$

$$Mx - \gamma \leq 0, \tag{4.40}$$

$$\lambda^T(Mx - \gamma) = 0, \quad \lambda \geq 0. \tag{4.41}$$

Ezek a kifejezések felírhatók az S_{act} indexhalmazzal, amley az aktív kényszereket jelöli:

$$Ex + F + \sum_{i \in S_{\text{act}}} \lambda_i M_i^T = 0, \tag{4.42}$$

$$M_i x - \gamma_i = 0 \quad i \in S_{\text{act}}, \tag{4.43}$$

$$M_i x - \gamma_i < 0 \quad i \notin S_{\text{act}}, \tag{4.44}$$

$$\lambda_i \geq 0 \quad i \in S_{\text{act}}, \quad \lambda_i = 0 \quad i \notin S_{\text{act}}. \tag{4.45}$$

Amennyiben egy kényszer aktív, a hozzá tartozó Lagrange-multiplikátor pozitív, míg ha egy kényszer inaktív, a Lagrange-multiplikátor értéke nulla. Ha M_{act} és λ_{act} adottak, az optimális zárt megoldás a következő:

$$\lambda_{\text{act}} = -(M_{\text{act}}E^{-1}M_{\text{act}}^T)^{-1}(\gamma_{\text{act}} + M_{\text{act}}E^{-1}F), \tag{4.46}$$

$$x = -E^{-1}(F + M_{\text{act}}^T \lambda_{\text{act}}). \tag{4.47}$$

Amennyiben az aktív kényszerek ismertek, az eredeti probléma helyettesíthető egyenlőségi feltételekkel. Az adott időpillanatban aktív kényszerek az Active Set módszer segítségével határozhatók meg, melynek során minden lépésében egy úgynevezett "munkahalmaz" (working set) definiálható. Ez a halmaz az aktuális ponthoz kapcsolódó ténylegesen aktív kényszereket tartalmazza, és a megoldás során csak ezek kerülnek figyelembe vételre. Ezt követően, az algoritmus minden lépésében egy egyenlőségi kényszer-problémát old meg. Ha az összes Lagrange-multiplikátor $\lambda_i \geq 0$, akkor az adott pont lokális megoldás az eredeti problémára. Amennyiben létezik olyan $\lambda_i < 0$, akkor a kényszer módosítható, és a probléma újraértékelhető. Ehhez az optimalizációs probléma megoldásához, egy iteratív algoritmus szükséges. A dolgozat során erre a Hildreth kvadratikus prgramozást használtam.

4.2.4. Hildreth kvadratikus programozás

Amennyiben ismertek az aktív kényszerek a Hildreth-féle kvadratikus programozási eljárás segítségével iteratív módon optimalizálhatók az egyes Lagrange-multiplikátorok értékei. Az algoritmus során a döntési változók egy-egy komponensét kezeljük egy időben. Az aktuális komponens λ_i értékét optimalizáljuk, hogy csökkentsük a célfüggvényt. Az iterációs folyamat a következőképpen írható fel:

$$\lambda_i^{m+1} = \max(0, w_i^{m+1}), \tag{4.48}$$

ahol

$$w_i^{m+1} = -\frac{1}{h_{ii}} \left[k_i + \sum_{j=1}^{i-1} h_{ij} \lambda_j^{m+1} + \sum_{j=i+1}^{n} h_{ij} \lambda_j^m \right]. \tag{4.49}$$

Ebben a képletben h_{ij} a $H = ME^{-1}M^T$ mátrix i, j-edik eleme, míg $k_i = \gamma_i + ME^{-1}F$ az i-edik elem a K vektorban. A λ Lagrange-multiplikátorok alapján a megoldás az alábbi formában adódik:

$$x = -E^{-1}(F + M^T \lambda^*). \tag{4.50}$$

A Hildreth-algoritmus nagy előnye, hogy nem szükséges minden iterációban mátrix-inverziót végezni, ezáltal hatékonyan alkalmazható valós idejű rendszerekben, ahol gyors megoldásokra van szükség.

4.2.5. Zárt alakú megoldás λ^*-ra

A Hildreth-féle kvadratikus programozási eljárás során a megoldás konvergál a Lagrange-multiplikátorok egy olyan értékkészletéhez, amely nulla értéket vesz fel az inaktív kényszerekhez, és pozitív értéket rendel az aktív kényszerekhez. A zárt alakú megoldás λ^*_{act} segítségével csak az aktív kényszerekhez tartozó komponensek maradnak meg:

$$\lambda^*_{\text{act}} = -(M_{\text{act}} E^{-1} M^T_{\text{act}})^{-1}(\gamma_{\text{act}} + M_{\text{act}} E^{-1} F). \tag{4.51}$$

Az aktív kényszerekhez tartozó komponensek megoldásával a feladat minimalizálható, és az összes többi kényszer automatikusan teljesül. Ezzel a zárt alakú megoldással az optimalizálás hatékonyan végrehajtható anélkül, hogy az inaktív kényszereket külön kellene kezelni.

Az adott időpillanatban az optimális Lagrange-szorzók meghatározásának egyik jól ismert megközelítése a Warm Start módszer. Ebben az esetben a korábbi iteráció optimális Lagrange-szorzóit adjuk meg kezdőértékként az aktuális kvadratikus programozási feladatnak, ezáltal kiküszöbölve azt, hogy minden időlépésben nulláról kelljen kezdeni az optimalizálást.

4.3. Felügyelt tanulás

A felügyelt tanulás (supervised learning) a gépi tanulás egyik legismertebb ága, ahol a modell címkézett adatokból tanul. Az adatok egy bemeneti változóból (x) és egy kimeneti változóból (y) állíthatók elő. A cél az, hogy a modell megtanulja, hogyan lehet a bemenetek alapján előre jelezni a megfelelő kimeneteket. A felügyelt tanulási két legismertebb feladata a regresszió és a klasszifikáció. A regresszió a folytonos értékek, míg a klasszifikáció diszkrét kategóriák előrejelzésére szolgál.

Egy általános tanulási probléma célja egy olyan függvény meghatározása, amely a bemeneteket a megfelelő kimenetekhez társítja. Ennek általános alakja a következő:

$$\hat{y} = f(x; \theta), \tag{4.52}$$

ahol $\hat{y}$ a modell által becsült kimenet, x a bemenet, és θ tartalmazza a modell paramétereit. A cél olyan paramétereket találni, amelyek minimalizálják a predikciós hibát.

A tanulási folyamat első lépése a forward propagation, melynek során a neurális háló rétegein keresztülvezetve a bemeneteket, előáll a becsült kimenet. Az így előállított kimenetet a rejtett rétegek dolgozzák fel a modell belső struktúrájában, amelyek a tanult súlyok és eltolások alapján meghatározzák a végső becsült értéket.

A regressziós feladatoknál gyakran használt modell a Multilayer Perceptron (MLP), egy olyan feedforward neurális háló, amely lineáris rétegeket és azokhoz tartozó aktiváci-

ós függvényeket tartalmaz. A rétegek a modell által tanult belső reprezentációkat írják le. Az MLP rétegei különböző aktivációs függvényekkel rendelkezhetnek, amelyek nemlinearitást vezetnek be a modellbe, és lehetővé teszik a komplex minták tanulását. A gyakran használt aktivációs függvények lehetnek például a ReLU (Rectified Linear Unit), a tangens hiperbolikus (tanh) vagy a sigmoid (sgm) függvény:

$$\sigma(z) = \max(0, z), \tag{4.53}$$

$$\sigma(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}, \tag{4.54}$$

$$\sigma(z) = \frac{1}{1 + e^{-z}}. \tag{4.55}$$

Az előreterjesztés során minden egyes réteg kimenetét kiszámítjuk a bemenetekből, amely a következő képlet alapján történik:

$$h^{(l)} = \sigma(W^{(l)} h^{(l-1)} + b^{(l)}), \tag{4.56}$$

ahol $W^{(l)}$ a súlymátrix, $b^{(l)}$ az eltolás (bias), és σ az adott aktivációs függvény.

A modell teljesítményének mérésére szolgáló költségfüggvény határozza meg, mennyire pontos a modell predikciója. A regressziós problémáknál gyakran használt költségfüggvény a négyzetes hiba (mean squared error, MSE):

$$MSE = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2, \tag{4.57}$$

ahol y_i a valós kimenet, és $\hat{y}_i$ a modell által becsült érték az i-edik adatpont esetén. Klasszifikációs feladatoknál, különösen bináris osztályozás esetén, a Binary Cross-Entropy (BCELoss) a gyakran használt költségfüggvény. A BCELoss a valószínűségeket kezeli, és a predikciók és a valós címkék közötti különbséget méri. Az alábbi képlet írja le a Binary Cross-Entropy-t:

$$BCELoss = -\frac{1}{n} \sum_{i=1}^{n} \left(y_i \cdot \log(\hat{y}_i) + (1 - y_i) \cdot \log(1 - \hat{y}_i) \right), \tag{4.58}$$

ahol y_i a valós címke (0 vagy 1), $\hat{y}_i$ pedig a modell által becsült valószínűség az i-edik adatpont esetén. A cél az adott költségfüggvény minimalizálása, tehát a predikciós hiba csökkentése.

A következő lépés az algoritmusban, hogy a háló rétegein visszafelé haladva kiszámítjuk, hogy a háló különböző rétegeinek súlyai milyen mértékben járulnak hozzá a hibához, ami lényegében az összetett függvényre vonatkozó deriválási szabály ismételt alkalmazása. Ezt a folyamatot nevezik backpropagationnek. A backpropagation során a költségfügg-

vény gradiensét használjuk, amely megmutatja, hogyan kell változtatni a súlyokat a hiba csökkentése érdekében:

$$\theta \leftarrow \theta - \alpha \nabla_\theta J(\theta), \tag{4.59}$$

ahol α a tanulási ráta, $\nabla_\theta J(\theta)$ pedig a költségfüggvény gradiensvektora, amely megmutatja, hogy a súlyokat milyen irányban kell változtatni a hiba csökkentése érdekében.

A hálózat súlyainak és eltolásainak a finomhangolását optimalizációs algoritmusok segítségével végezhetjük el. Az egyik legszélesebb körben használt algoritmus az Adam (Adaptive Moment Estimation) [35].

Az Adam algoritmus előnye, hogy két mozgó átlagot vezet be: az első momentum az átlagos gradiens (m_t), a második momentum pedig a gradiens négyzetének átlagát (v_t) reprezentálja. Ezeket a momentumokat használva az Adam adaptív tanulási rátákat számít minden paraméterhez:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t, \tag{4.60}$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2, \tag{4.61}$$

ahol β_1 és β_2 a momentum becslésének csökkenési rátái, g_t pedig a célfüggvény t-edik időpillanatban vett gradiense.

A korai lépésekben ezek az értékek torzítottak lehetnek, mivel a momentumok kezdeti értékei nulláról indulnak, ezért a becslés a kezdeti néhány lépésben alulbecsüli a valós gradiens nagyságát. Ezért az Adam algoritmus torzítás-korrigált becsléseket használ, hogy ellensúlyozza ezt a kezdeti eltérést, és pontosabb gradiensértékeket adjon a tanulás elején is:

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}, \quad \hat{v}_t = \frac{v_t}{1 - \beta_2^t}. \tag{4.62}$$

A modell paraméterei ezek után a következőképpen frissíthetők:

$$\theta_{t+1} = \theta_t - \frac{\alpha}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t, \tag{4.63}$$

ahol α a tanulási ráta, és ϵ egy kis konstans a nullával való osztás elkerülése érdekében.

A tanulási ráta finomhangolásához, illetve a hatékonyabb tanulási folyamat érdekében gyakran alkalmaznak learning rate scheduler-t, amely dinamikusan csökkenti a tanulási rátát a tanulás során, különösen akkor, ha a modell teljesítménye stagnál. Azonban a tanulási rátán kívül a modell általánosítási képességeinek javítása érdekében szükség lehet egy további finomításra, amit a súlyok csökkentésével (weight decay) érhetünk el. Ez a megközelítés az AdamW optimalizáló algoritmus egyik előnye, amely külön alkalmazza a súlycsökkentést a gradiens információktól függetlenül, ezzel segítve a túltanulás

(overfitting) elkerülését. Ez a súlycsökkentés a következő képpen írható fel:

$$\theta_t = \theta_t - \lambda\theta_t, \tag{4.64}$$

ahol λ a weight decay tényező. Az AdamW az Adam algoritmus előnyeit kombinálja a paraméterek csökkentésének hatékonyságával, így különösen előnyös lehet nagy méretű és komplex neurális hálók esetében.

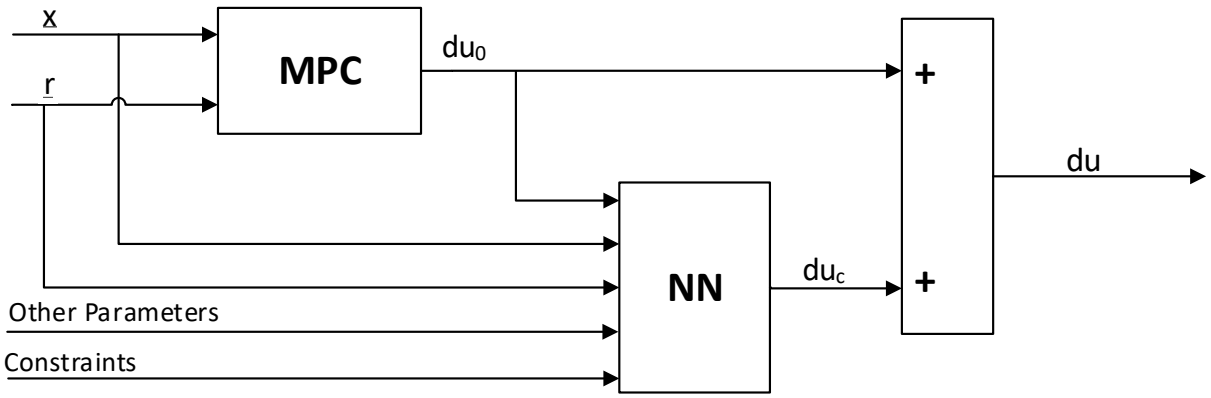
5. fejezet

Javasolt Supervised Learning - Model Predictive Control struktúrák

5.1. A QP teljes kiváltását célzó struktúra

Ebben a fejezetben bemutatom az általam létrehozott struktúrákat, valamint a neurális hálózat tanításához szükséges paramétereket és adatszett generálását.

A kényszeres MPC megoldása során az egyik legszámításigényesebb folyamat a kvadratikus programozás, amely a kényszerek figyelembevételével határozza meg a beavatkozó jelet. Ennek következtében az elsődleges ötlet az volt, hogy a neurális háló segítségével megbecsüljük a kényszeres feltételek hatását, ami a 4.2.2 egyenletben a második tag, teljesen kihagyva a kvadratikus programozást. Ehhez az 5.1 ábrán látható struktúrát hoztam létre.



5.1. ábra. A korlátos beavatkozó jel becslésére létrehozott NN-MPC1 struktúra

Ebben a struktúrában az MPC és a neurális háló párhuzamosan működik, hogy meghatározza a megfelelő beavatkozó jelet. Az MPC itt a globális megoldást (du_0) biztosítja, amelyet az állapotvektor (x) és a referencia jel alapján számít ki, míg a neurális hálózat feladata, megbecsülni a kényszerek hatását. A 5.1 táblázat bemutatja az ehhez a

struktúrához használt MLP modell tanításához szükséges paramétereket.

5.1. táblázat. Az NN-MPC1 struktúrához használt MLP modell tanítási paraméterei.

Paraméter	**Érték**
Tanulási ráta (α)	0.0001
Súlycsökkentés	0.01
Epoch-ok száma	1200
Batch méret	256
Rejtett rétegek száma	3
Neuronok száma	256, 128, 64
Rejtett rétegek aktivációja	Tanh (4-es skálázás)
Kimeneti réteg aktivációja	Lineáris
Optimalizáló	AdamW
Költségfüggvény	Mean Squared Error (MSE)
Tanulási ráta ütemező	ReduceLROnPlateau

A tanításhoz szükséges adatokat MATLAB környezetben generáltam, a későbbiekben bemutatott véletlenszerű referencia pályák alkalmazásával. A kényszeres MPC-t ezeken a random pályákon futtattam végig, hogy változatos és sokféle tanulási adathalmazt állíthassak elő. Az MLP modell tanításához felhasznált bemeneti és kimeneti adatok a 5.2 táblázatban találhatók. Az input adatok megválasztásakor különös figyelmet fordítottam arra, hogy reprezentatívak legyenek a neurális háló számára. Annak érdekében, hogy a hálózat pontos előrejelzést tudjon nyújtani és összefüggést találjon a bemeneti és kimeneti adatok között, a rendszer aktuális, előző állapotait, valamint a referencia jelet határoztam meg bemenetnek. Ezen felül, az aktuális és előző kényszerek nélküli beavatkozó jelek hozzáadása lehetővé tette, hogy a hálózat megtanulja a kényszerek betartásához szükséges beavatkozási mintázatokat.

5.2. táblázat. Az NN-MPC1 struktúrához használt neurális háló tanítási adatai

Bemeneti változók
A rendszer aktuális állapota
Állapot visszacsatolás az MPC-ből
Az aktuális referencia értéke
Előző beavatkozó jel
Kényszerek nélküli beavatkozó jel változása
Kényszerekkel figyelembe vett beavatkozó jel változása
Kimeneti változó
Kompenzáló beavatkozó jel (NN kimenet)

Az algoritmus célja, hogy ezen input adatok alapján a háló megbecsülje a referenciajel pontos követéséhez szükséges kényszeres beavatkozó jelet, miközben betartja a megadott kényszereket. Az MPC globális megoldást biztosít, amelyet a neurális háló kiegészít, így

áll elő a végső beavatkozó jel. Ezzel a struktúrával a kvadratikus programozás teljes mértékben kihagyható.

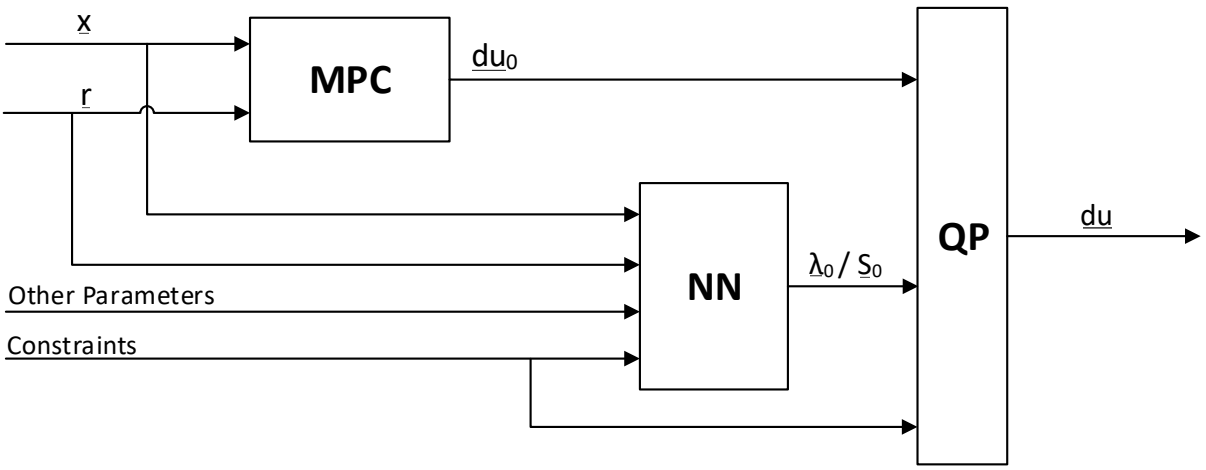
5.2. A QP gyorsítását célzó struktúrák

A QP iteratív jellege miatt, alapesetben a rendszer minden időlépésben nulláról kezdi a Lagrange-szorzók optimalizálását, ami megnöveli a számítási időt. Ennek megoldására két megközelítést alkalmaztam.

Az első esetben a neurális háló célja, hogy a QP iterációs folyamatának pontos kezdőértéket adjon a Lagrange-szorzók (λ) számára. Így a QP gyorsabban konvergál, csökkentve az iterációs számot és a számítási időt. A Lagrange-szorzók értékei nem negatívak, ezért a neurális háló kimeneti rétegének aktivációs függvényeként, valamint a köztes rétegekhez is ReLU-t választottam, amely ennek a tartománynak megfelelő kimeneteket biztosít. Mivel a cél egy nagyobb méretű, egzakt vektor becslése, nagyobb neuronszám használata volt szükséges a rejtett rétegekben.

A második megközelítés az adott időlépésben aktív korlátok predikciója volt. Ezt egy bináris klasszifikációval lehet elérni, ahol a zárt alakú megoldás segít csökkenteni a számítási igényt. Amennyiben az adott időpillanatban ismerjük az aktív kényszereket, a QP során ezek zárt alakú megoldásából kiindulva szükséges elvégezni az optimalizálást. A tanítás során szigomoid aktivációs függvényt használtam, amelyhez egy kisebb struktúrájú háló is elegendő volt.

A kétféle predikciós struktúrát az alábbi 5.2 ábra szemlélteti:



5.2. ábra. Lagrange-szorzók kezdő értékének NN-MPC2, és az aktív kényszerek becslésének NN-MPC3 a struktúrája

Az 5.3 táblázat a pontos lambda becsléséhez, valamint az aktív korlátok előrejelzéséhez használt neurális háló tanítási paramétereit mutatja be:

5.3. táblázat. Az NN-MPC1 és NN-MPC2 struktúrához használt MLP modellek tanítási paraméterei a Lagrange-szorzók becsléséhez és az aktív korlátok előrejelzéséhez

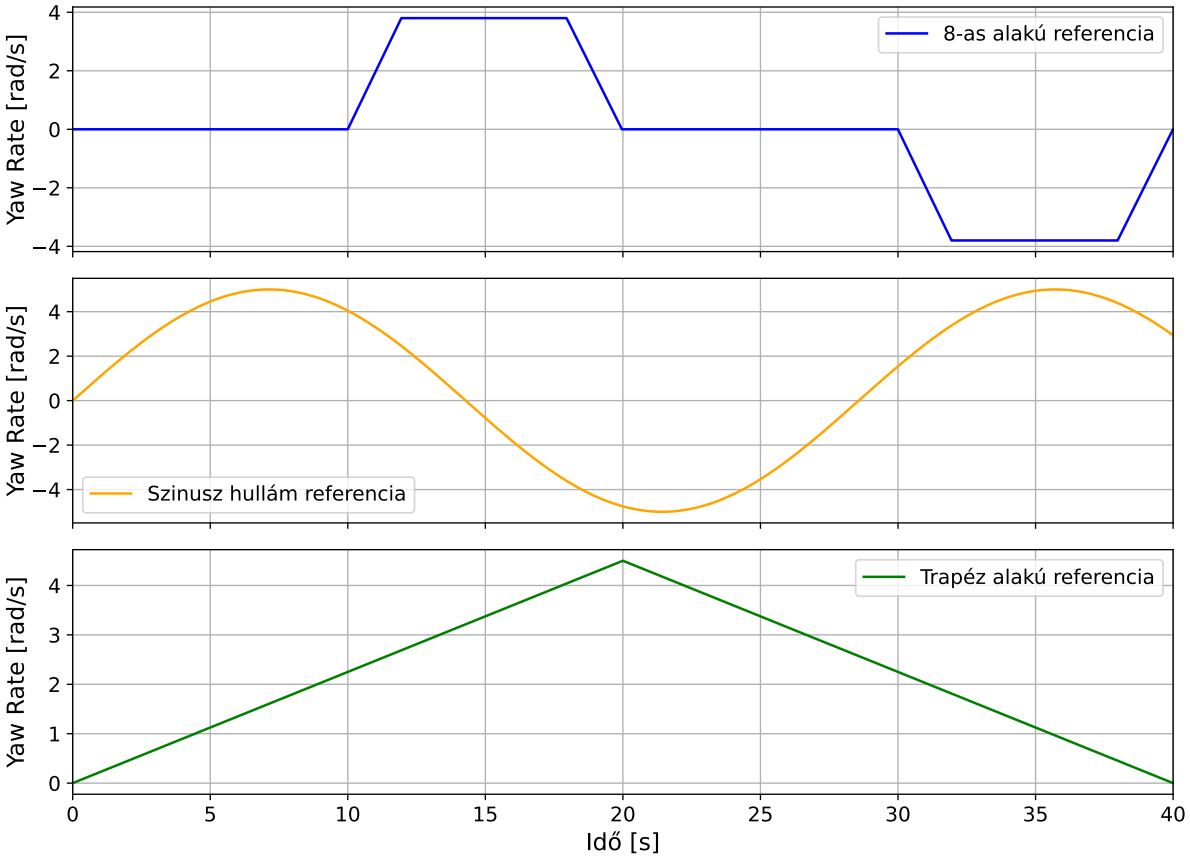
Paraméter	**Lagrange-szorzók becslése**	**Aktív korlátok előrejelzése**
Tanulási ráta (α)	0.0001	0.0001
Súlycsökkentés (λ)	0.01	-
Ütemező	ReduceLROnPlateau	ExponentialLR
Gamma (ütemező csökkentési tényező)	-	0.9
Batch méret	64	64
Rejtett rétegek száma	3	2
Rejtett rétegek mérete	2048, 1024, 512	128, 64
Kimeneti réteg mérete	16	16
Dropout ráta	-	0.4
Rejtett rétegek aktivációs függvénye	ReLU	ReLU
Kimeneti aktivációs függvény	-	Sigmoid
Optimalizáló	AdamW	Adam
Költségfüggvény	Mean Squared Error (MSE)	BCELoss (Binary Cross-Entropy)
Epochok száma	820	500

6. fejezet

Eredmények

Az algoritmusok teszteléséhez három különböző referencia pályát hoztam létre. Az első pálya egy állandó referencia, amely klotoid átmenetet (lineárisan csökkenő vagy növekvő szakasz) követően újabb állandó referenciát tartalmaz. A pálya átmenetei úgy lettek megválasztva, hogy az X-Y síkon egy nyolcas alakú referencia pályához vezet.

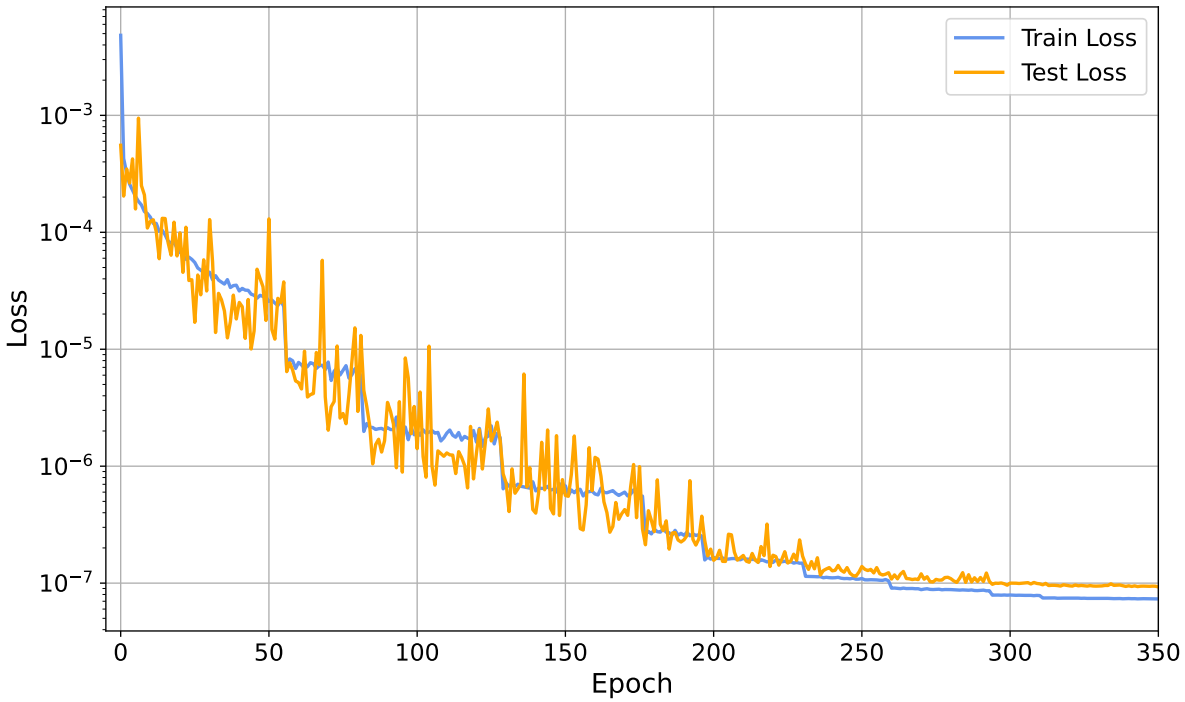
A második pálya egy lineárisan növekvő és csökkenő referenciát tartalmaz, míg a



6.1. ábra. Az algoritmusok teszteléséhez használt referencia pályák.

harmadik egy szinuszhullám-alapú referencia. Ezek a 6.1 ábrán láthatók.

Az első struktúrában (NN-MPC1), amelyben a neurális háló becsüli meg a kényszerek okozta kompenzációs tagot, a kvadratikus programozás teljes elhagyásával jelentős futási időcsökkenést értem el. A fő kérdés az volt, hogy a hálózat mennyire képes biztosítani a rendszer stabilitását és a korlátok megfelelő betartását. Ahogy a 6.2 ábrán is látható, a háló a 5 fejezetben bemutatott paraméterekkel 320 epoch után konvergált. A betanítás során 10^{-7} nagyságrendű hibát értünk el, ami azt mutatja, hogy a háló alacsony hibával megfelelően betanult.



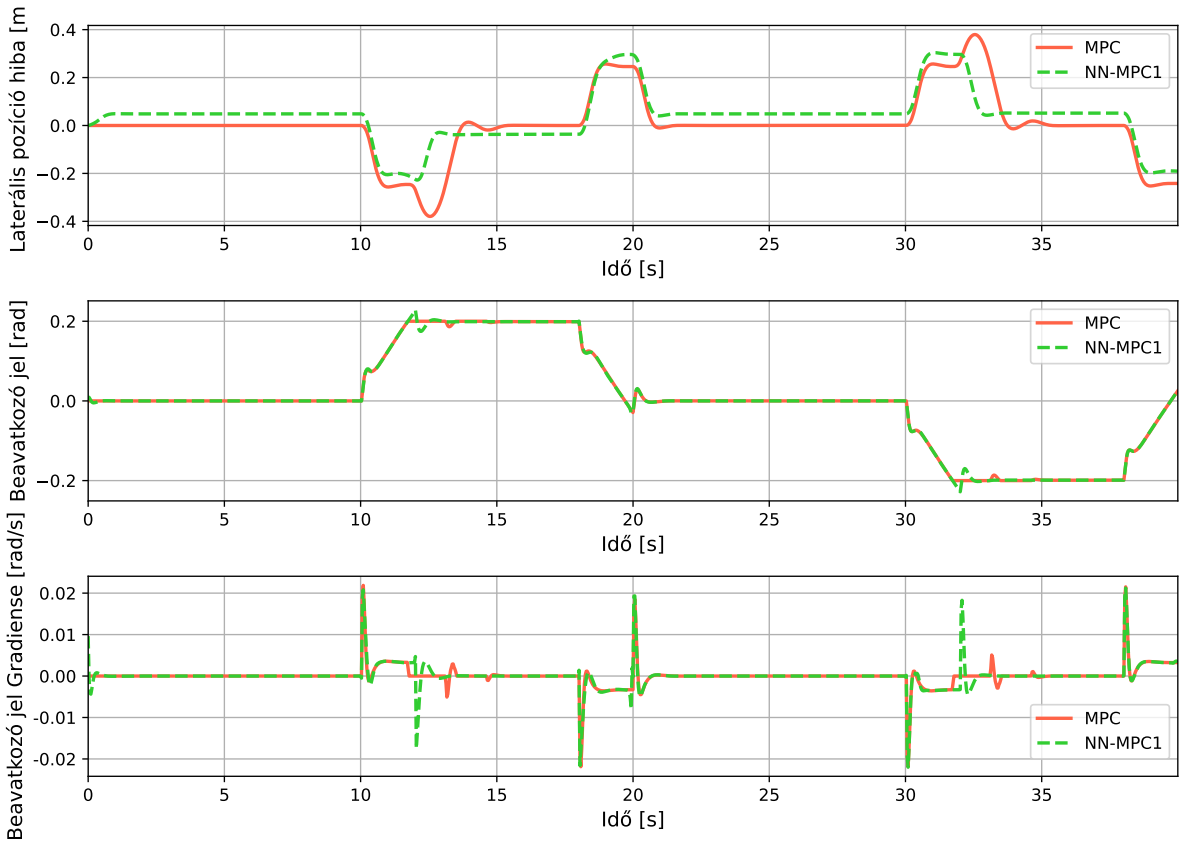
6.2. ábra. Train/ test loss a korlátos beavatkozó jel prediktálása esetén

Az eredmények szemléltetéséhez az első struktúrát viszonylag nagy, a valóságban ritkán előforduló referenciaértékeken teszteltem, ezzel demonstrálva, hogy a megoldás extrém esetekben is működőképes. A futási idő és a teljesítmény kiértékelése során ugyanakkor több véletlenszerűen generált referenciaértékre vett átlagot vizsgáltam, így reálisabb képet kapva a struktúra általános működéséről.

A jármű fizikai korlátainak betartása érdekében a beavatkozó jelre $0.2rad$, annak gradiensére pedig $0.05rad/s$ értékű korlátokat írtam elő.

Az első struktúra performanciái a 6.3 ábrán láthatók, ahol az y tengely a laterális pozíció hibát, beavatkozó jelet és a beavatkozó jel változását, míg az x tengely az időt ábrázolja. Megfigyelhető, hogy a hálóval kiegészített struktúra mind a beavatkozó jelre, mind annak gradiensére vonatkozó korlátokat kis mértékű túllépéssel de betartotta. Megfigyelhető továbbá, hogy a hálós struktúra csökkentette a laterális pozíció hibát is. Ahogy az ábrán látható a kezdetben kialakuló állandósult állapotbeli hibának köszönhe-

tően, abszolút értékben kisebb lett a laterális pozíció hiba, azonban a hirtelen referencia változásoknál lévő hibacsökkenést a kényszerek kis mértékben túllépése magyarázza. Kisebb referencia értékek esetén viszont, a hálóval kiegészített struktúra pontosan betartja az előírt kényszereket.

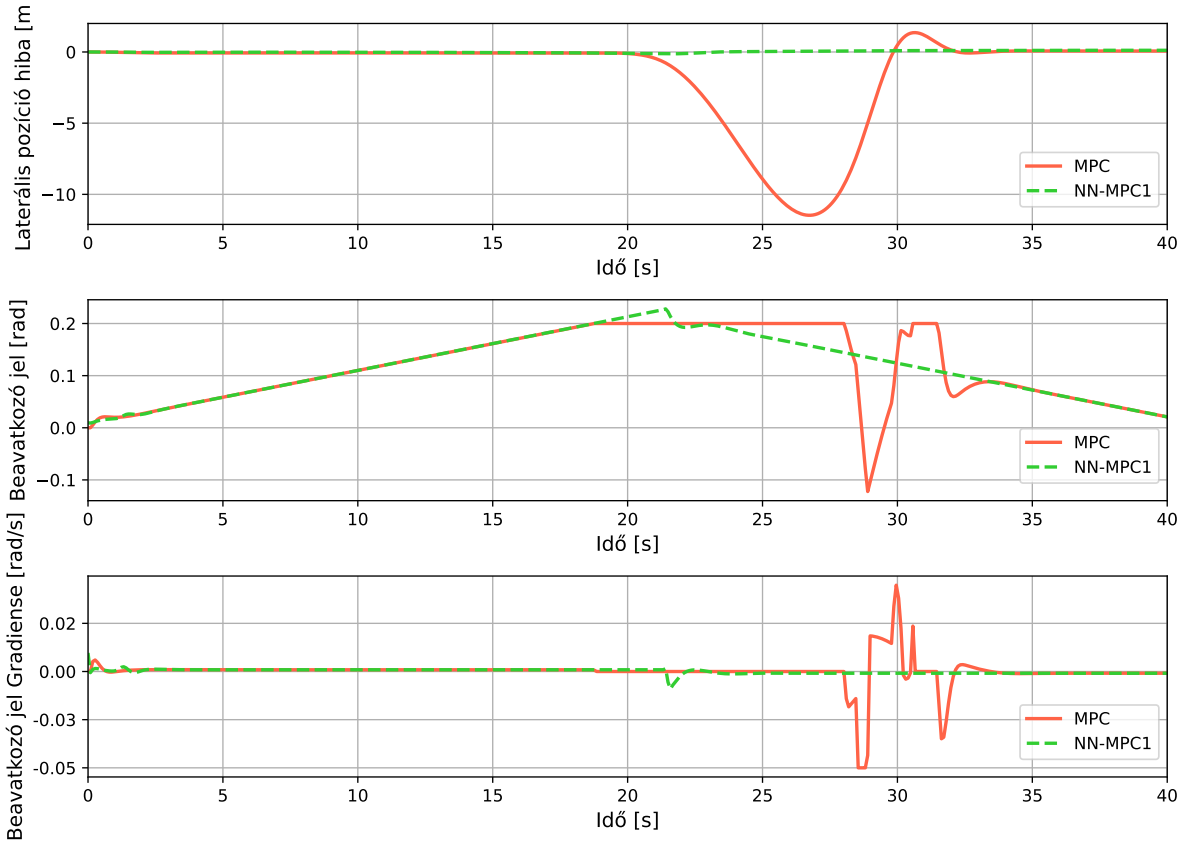


6.3. ábra. Az MPC és az NN-MPC1 performanciája 8-as referencia esetén

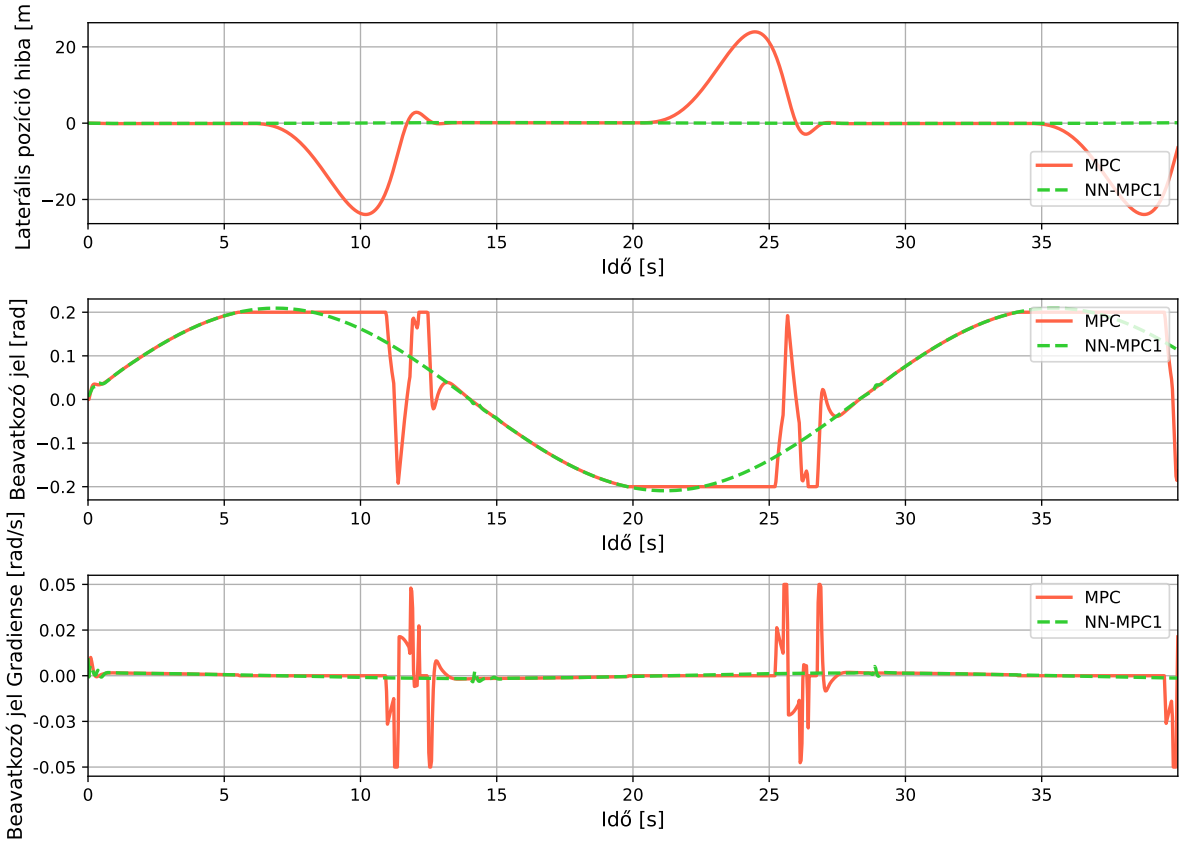
Hasonlóan eredményes volt a rendszer a lineárisan növekvő és csökkenő referenciák esetében is. Az 6.4 ábra alapján, mivel a referenciajel viszonylag hirtelen vált növekvőből csökkenőbe, az MPC maximális beavatkozó jellel próbál kompenzálni, ami nem bizonyul elegendőnek, így 10 m-es oldalirányú pozícióhibát eredményez, melyet lassan tud korrigálni. A neurális háló alapú struktúra viszont, minimálisan megszegve a beavatkozó jelre vonatkozó korlátokat, 10 cm-en belüli hibát ért el, ezáltal sokkal pontosabb referenciajelkövetést biztosítva.

A 6.5 ábra szinusz hullám referenciajel esetén mutatja be a legjobban, hogy ugyan kismértékben megszegi a kényszereket a neurális háló, de jelentősen javítja a jelkövetést, emellett pedig simítja a beavatkozó jelet és annak változását is, ezáltal simább és komfortosabb szabályozást valósítva meg.

A szakirodalomban két megközelítés terjedt el a párhuzamosan működő szabályozók esetén. Az egyik módszer, hogy a két párhuzamosan alkalmazzott szabályozó különböző súlyozásával biztosítjuk a stabilitást. Azonban ebben az esetben a hálózat black-box

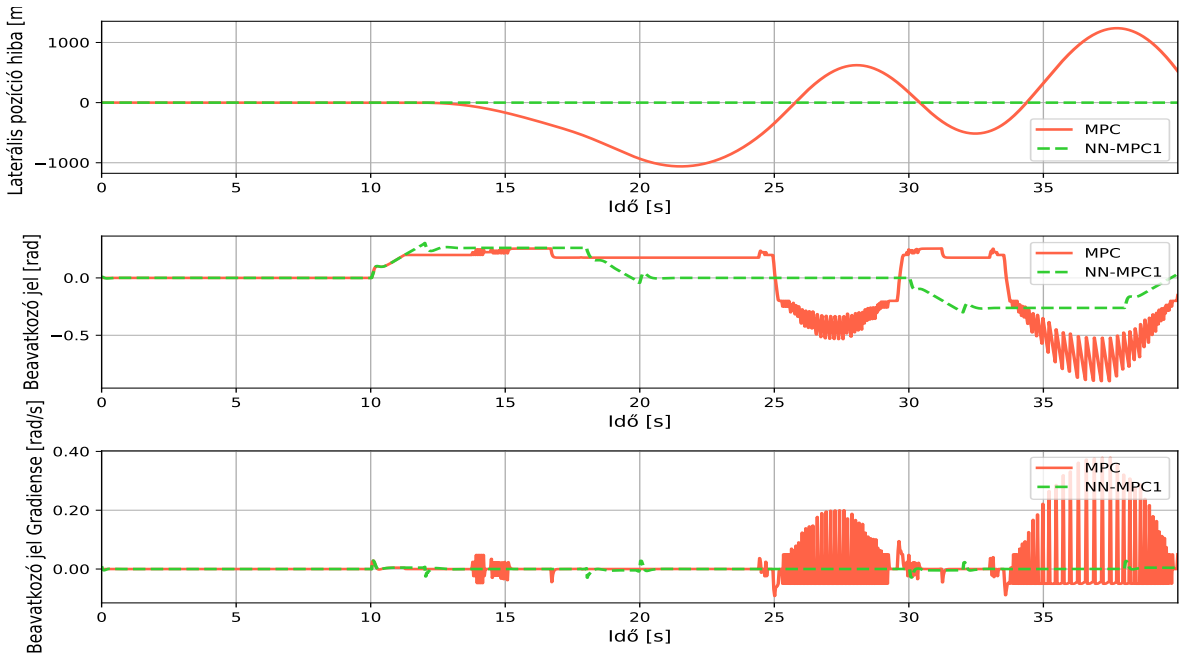


6.4. ábra. Az MPC és az NN-MPC1 performanciája lineárisan változó referencia esetén



6.5. ábra. Az MPC és az NN-MPC1 performanciája szinusz hullám referencia esetén

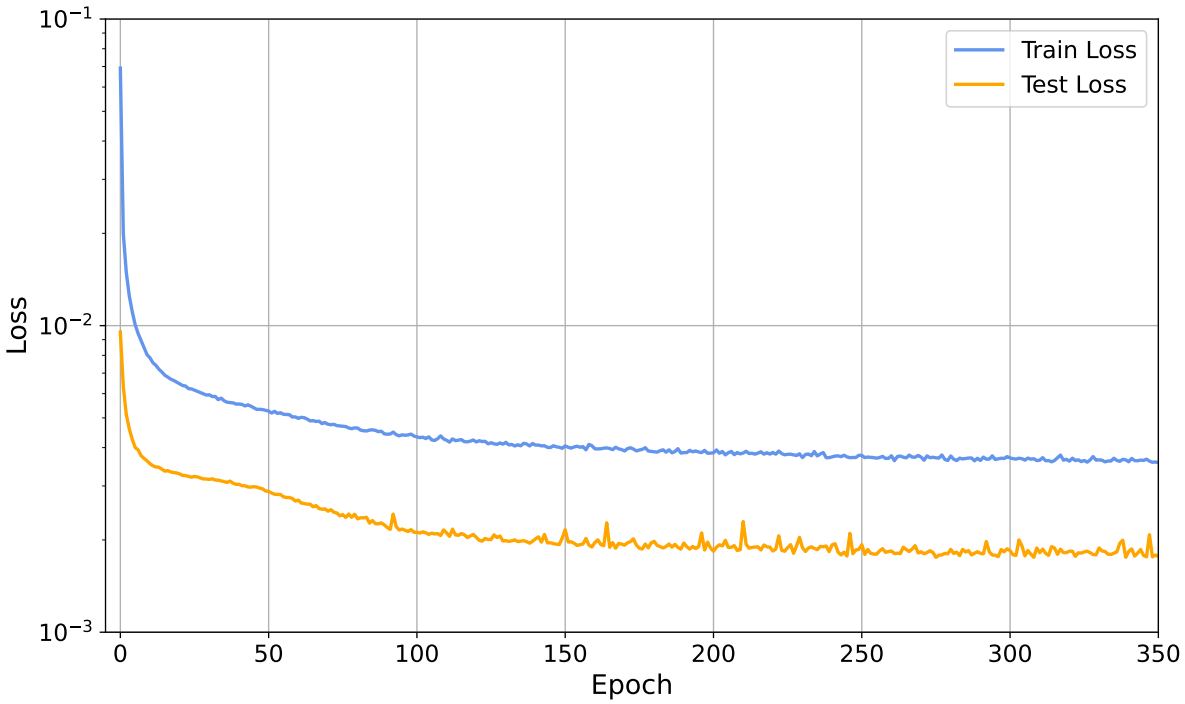
jellege miatt nem lehet egyértelműen megválasztani a megfelelő súlyokat. A másik megközelítés egy olyan struktúra amely biztosítaná, hogy az egyik szabályozó szükség esetén felülírja a másik szabályozó hatását a stabilitás érdekében. Azonban a mi esetünkben ez teljesen elnyomná a háló által becsült részt, ezért explicit módon, empirikus módszerekkel vontam le következtetéseket a rendszer viselkedéséről. Megfigyelhető, hogy a neurális hálózat ellentétes hatást fejt ki az MPC szabályozóval szemben. Ebből adódóan csökkenti az integráló tulajdonságot az Anti-Windup logikával szemben, ezáltal nem képes mindig zéró követési hibát garantálni. Ebből viszont lehet következtetni a rendszer stabilitásával kapcsolatos kérdésekre, mivel a hálózat nem engedi, hogy az MPC szabályozó elintegrálódjon vagy instabillá váljon. Ezt jól szemlélteti a 6.6 ábra is. Ebben az esetben egy olyan nagy referenciát adtam, ahol az MPC önmagában nem képes megoldani a problémát és instabillá válik. A neurális hálózat azonban, miközben csak kis mértékben sérti meg a korlátokat, stabilizálja az egyébként instabillá váló MPC szabályozót.



6.6. ábra. Az MPC és az NN-MPC1 performanciája magas referencia esetén

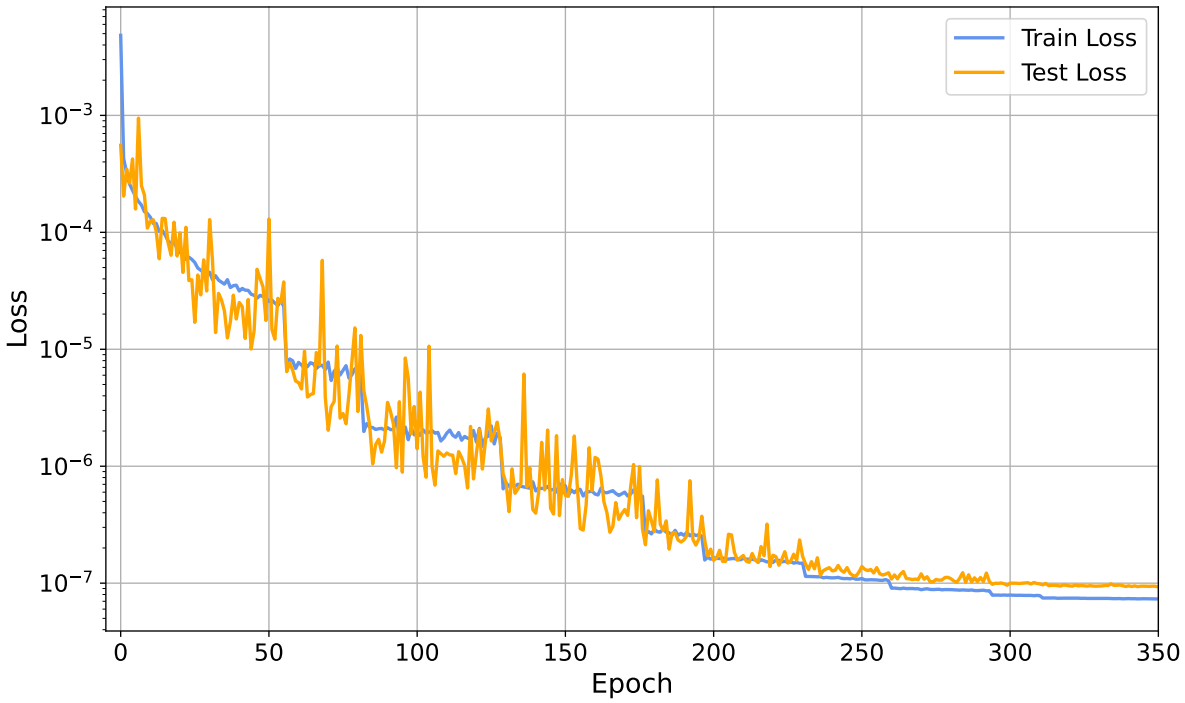
A futási idők összehasonlítása érdekében különböző referenciajeleket alkalmaztam a szimulációk során. A lineárisan növekvő és csökkenő pályák esetében véletlenszerűen generált maximális értékeket használtam, míg a szinusz hullámnál randomizált amplitúdót és frekvenciát. Ahogy az várható volt a kvadratikus programozás elhagyásával átlagosan egy nagyságrenddel sikerült csökkentenünk a futási időt. A neurális hálós megoldás esetében a szimulációs idő minden referencia esetében hasonló, míg a hagyományos MPC-nél a futási idő a probléma bonyolultságától függ.

A másik struktúra esetében a train/test loss görbék a 6.7 és 6.8 ábrán láthatók.



6.8. ábra. Train/ test loss az aktív korlátok prediktálása esetén

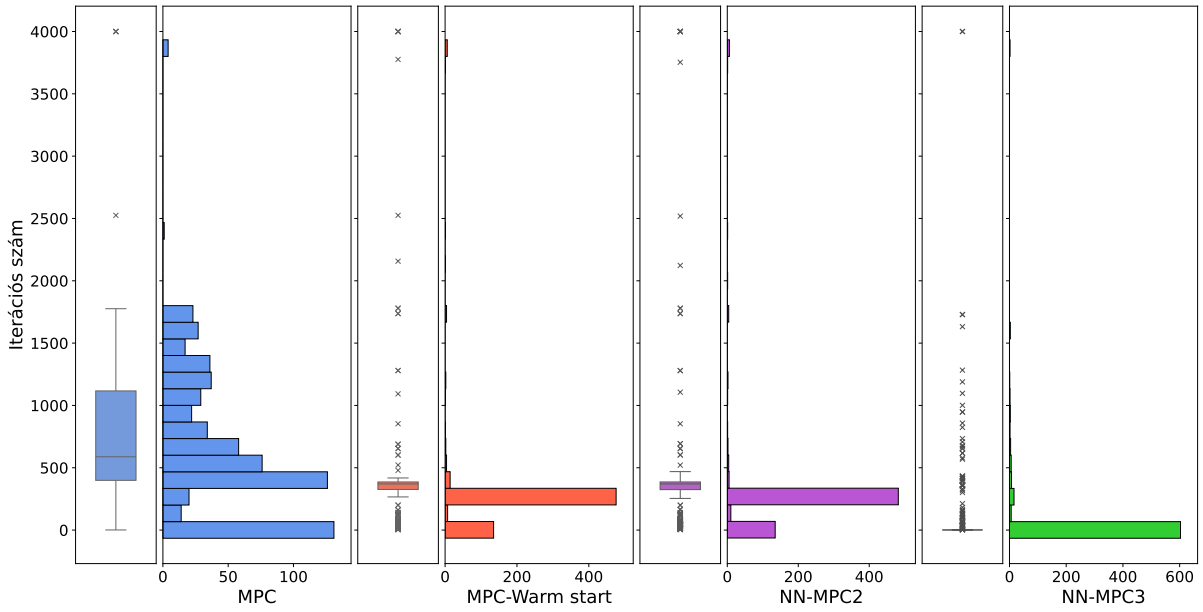
A Lagrange-szorzók pontos értékeinek becslésére a hálózat hasonló módon mint az előző struktúrában $10e-7$-nél konvergált, 320 epoch után. Az aktív és passzív korlátok prediktálása esetében egy klasszifikációs problémát kellett megoldani, amelyhez a BCELoss költségfüggvényt használtam, ahogy a 5.3 táblázatban látható. Az ábrán megfigyelhető,



6.7. ábra. Train/ test loss a pontos lambda prediktálása esetén

hogy ebben az esetben a train és test loss értékei $10e-2-10e-3$ között vannak. Klasszifikációs feladatoknál ez már kellően alacsony értéknek számít, így elmondható, hogy ez a hálózati struktúra is megfelelően betanult.

Az adott struktúra kiértékeléséhez a kvadratikus programozás iterációszámának összehasonlítását választottam box diagrammon ábrázolva, mivel a performanciában nem volt különbség. A box diagramokon az y tengelyen az időlépésekhez tartozó iterációszámok szerepelnek, míg az x tengelyen az egyes algoritmusok és az iterációszámok előfordulásainak száma látható. A Warm Start mellett az NN-MPC2 a Lagrange-szorzók pontos becslését végző, míg az NN-MPC3 az aktív és passzív korlátokat előrejelző struktúrát jelöli,

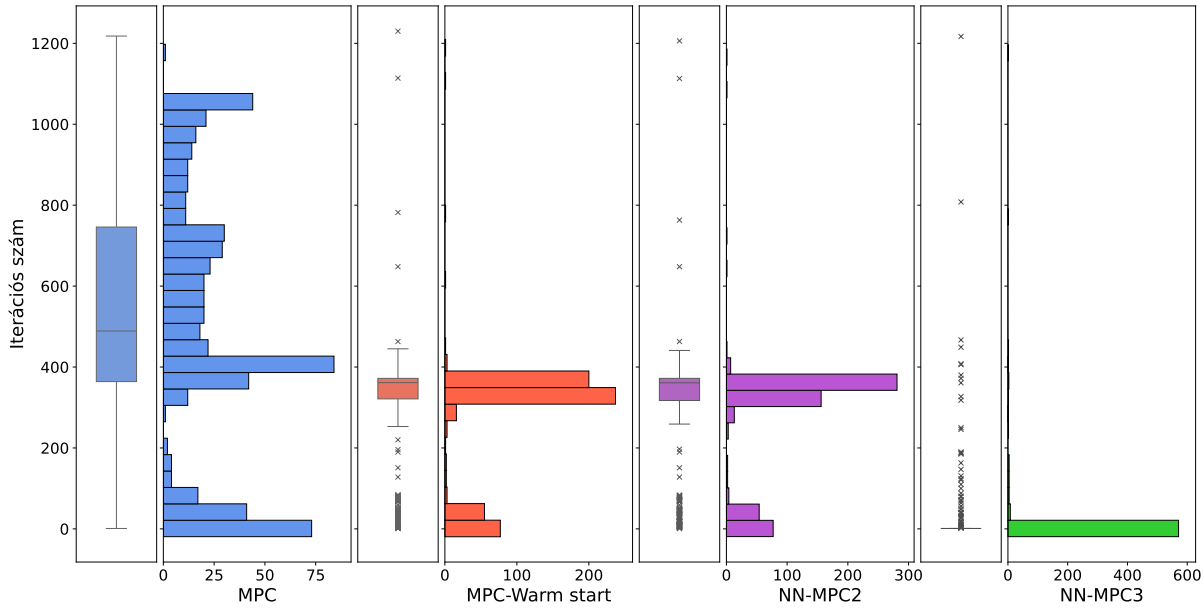


6.9. ábra. Kvadratikus programozás iterációs száma, a 8-as referencia esetén

A 6.9 ábra szemlélteti az algoritmusok teljesítményét a nyolcas alakú referenciapályán. Jól látható, hogy a kényszeres MPC átlagosan 1 és 1800 közötti iterációt igényel a feladat megoldásához. Ezzel szemben a Warm Startot alkalmazó módszer és az NN-MPC2 jelentősen csökkentette az iterációk számát. Megfigyelhető, hogy a két megoldás hasonló eredményeket produkál, bár az NN-MPC2 néhány esetben felülmúlja a Warm Startos megközelítést. A hasonlóság oka, hogy bár a neurális háló pontosabban prediktálja a Lagrange-szorzók értékeit, az egymást követő időlépésekben gyakran nagyságrendileg kicsi az eltérés a két optimális megoldás között. Így a Warm Start is hasonlóan jó megoldást nyújt az NN-MPC2 pontosabb becslésének ellenére. A különbség akkor válik jelentőssé, amikor a referencia hirtelen változik, és az előző optimális érték jelentősen eltér a következőtől, ilyenkor viszont a neurális háló továbbra is pontosan becsüli az optimális értéket.

A két megoldás közül az aktív és passzív kényszerek előrejelzése (NN-MPC3) csökkentette leginkább a kvadratikus programozás iterációszámát. Látható, hogy a pontos

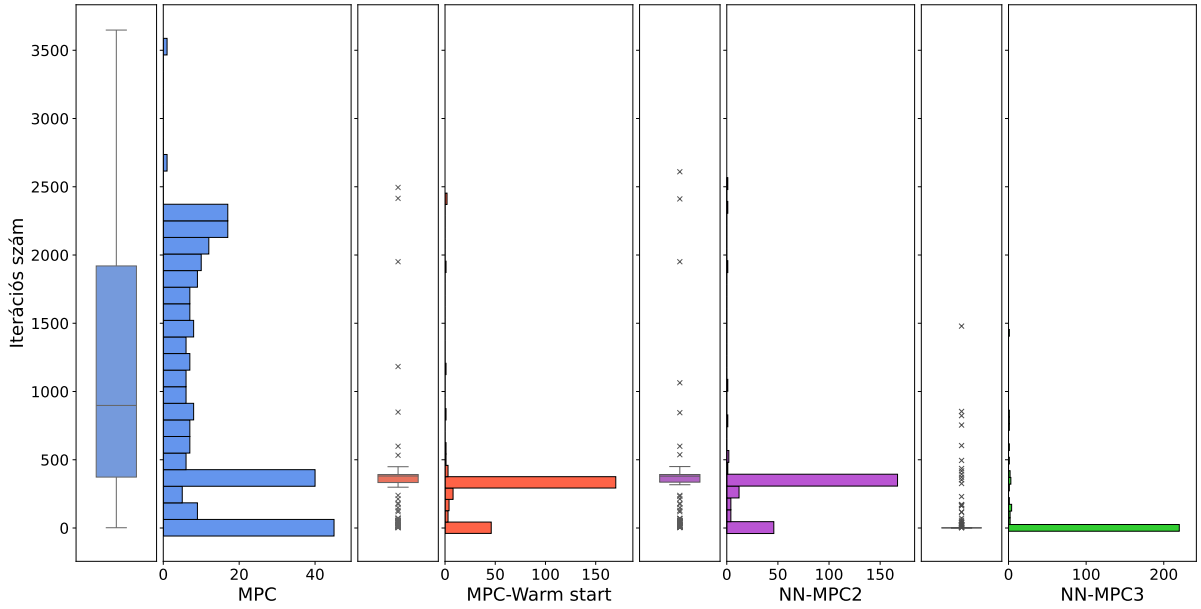
becsléseknek köszönhetően a QP az esetek többségében egyetlen iterációval meg tudta oldani a problémát. Mind a négy struktúrában megfigyelhető a 4000-es iterációs érték, amely a kvadratikus programozás iterációszámának felső korlátja. Ebben az esetben az algoritmus nem volt képes megoldani a problémát az adott időlépésben. Ez a jelenség a neurális hálós struktúrában is előfordul, mivel csak egy kezdeti értéket adunk a QP-nek, ezáltal ha a probléma fizikai korlátok miatt nem megoldható, akkor az MPC se egy közel optimális kezdeti értékkel, se nulláról kezdve nem fogja tudni megoldani az optimalizációs problémát. E struktúra esetében a stabilitási kérdést tekintve, mivel a hálózat valójában csak egy kezdeti értéket szolgáltat a QP-nek, és az aktív korlátok segítségével zárt alakban megoldható az eredetileg stabil MPC probléma, a neurális hálós struktúrával is stabil marad a rendszer, ha az alaphelyzetben stabil volt. Ebből kifolyólag a hálós struktúra ugyanúgy viselkedik, mint a hagyományos MPC, csupán az iterációk számát—tehát a futási időt csökkentettem.



6.10. ábra. Kvadratikus programozás iterációs száma, a szinuszhullám referencia esetén

Ugyanez a figyelhető meg a másik két referenciajel esetében is. A 6.10 és 6.11 ábrákon látható, hogy az iterációk száma jelentősen csökkent mind a Warm Start, mind az NN-MPC2 esetében, bár az előző referenciához képest kis mértékű növekedés tapasztalható. Azonban az NN-MPC3 megközelítés továbbra is lényegesen jobban teljesít mindhárom másik módszernél. Emellett megfigyelhetők kiugró értékek mindkét struktúrában. Ezek akkor jelentkeznek, amikor az aktív korlátok hirtelen passzívvá válnak. Ez a jelenség a szabályozóhoz használt hibaalapú modellből adódik, mivel a szabályozótervezéshez használt modell felírásakor elhanyagoltuk a nem irányítható tagot, amely a referencia hatását tartalmazza, így a neurális háló (és tanítása alapján szolgáló MPC) erről nem rendelkezik információval. Ennek ellenére, még a legrosszabb esetekben is elértük az eredeti MPC

maximális iterációszámát. Ez azt bizonyítja, hogy még nem teljesen pontos predikció esetén sem ront a neurális háló alkalmazása a szabályozás minőségén.



6.11. ábra. Kvadratikus programozás iterációs száma, a lineárisan növekvő és csökkenő referencia esetén

Az öt algoritmus összehasonlítása érdekében mindhárom referencia pályán véletlenszerűen generáltam a maximális értékeket és számoltam ki a futási idők átlagát, ami 6 táblázatban látható. Az eredmények alapján megfigyelhető, hogy a Warm Start esetében az átlagos futási idő kis mértékben alacsonyabb volt, mint az NN-MPC1 struktúránál. Ez azonban az algoritmusok sajátosságából adódik, mivel a Warm Start esetén egyszerűen hozzárendeljük a Lagrange-szorzókat, míg a predikciós módszernél egy külön függvény vezeti végig az inputot a neurális háló rétegein a predikció előállításához.

Mindazonáltal, ahogyan a blokkdiagramokon is látható, a kvadratikus programozás futási idejét tekintve hasonló vagy alacsonyabb értékeket értünk el az NN-MPC2 struktúrával mint a Warm Start. Az NN-MPC3 algoritmus esetében a Warm Start és az NN-MPC2 struktúrákhoz képest közel 40%-kal csökkentettük a futási időt. Az összes vizsgált algoritmus közül az NN-MPC1 struktúra bizonyult a leggyorsabbnak, ami várható volt a kvadratikus programozás teljes kihagyásából adódóan.

6.1. táblázat. Futási idők átlaga a különböző implementált algoritmusokra

Algoritmus	**Átlagos futási idő (s)**
MPC	1.4967
Warm Start	0.6004
NN-MPC1	0.0527
NN-MPC2	0.7590
NN-MPC3	0.2349

7. fejezet

Konklúzió

A dolgozat három különböző megoldást mutat be a kényszeres Model Prediktív Szabályozás feltételes optimalizációjának gyorsítására, hibaalapú biciklimodell alkalmazásával, mely figyelembe veszi a hagyományos dinamikai bicikli modell négy járműállapotának hibáit amelyek: a laterális pozíció, sebesség, legyezési szög és legyezési szögsebesség. Az MPC szabályozó egy globális és kényszeres megoldásból épül fel. Amennyiben az egyenlőségi kényszerek lineárisan nem függetlenek az optimalizálandó változóktól, az egyenlőtlenségi kényszereket is figyelembe kell venni, melyek esetén a Kuhn-Tucker feltételek alkalmazásával lehet megoldani a problémát. Ha előre ismerjük, mely kényszerek lesznek aktívak, a probléma egyenlőségi kényszerekkel is felírható, és zárt alakban megoldható.

A kényszeres megoldást kvadratikus programozással lehet megoldani A dolgozatban a Hildreth-féle kvadratikus programozási módszert választottam. Ez egy iteratív eljárás, amely nem igényel mátrixinverziót, hanem minden időpillanatban a Lagrange-szorzók értékein iterálva találja meg az optimális megoldást. A gépi tanulási algoritmusok közül a felügyelt tanulást választottam, mivel ez lehetővé teszi, hogy egy stabilan működő valós rendszer input-output adataiból tanuljon a hálózat, ezáltal pontos predikciókat érve el. A neurális hálózatok tanításához MATLAB környezetben implementált kényszeres MPC segítségével generáltam az adatokat.

Az optimalizáció gyorsítására két struktúrát hoztam létre. Az első struktúra esetében a kvadratikus programozást teljesen kihagyásával, a neurális hálózat közvetlenül a kényszer hatását becsüli meg. Ehhez egy háromrétegű MLP-t használtam, skálázott tangens hiperbolikus aktivációs függvénnyel. Ez a megközelítés jelentősen csökkentette a futási időt, miközben pontosabb referenciajel-követést ért el mint az MPC, és a kényszerek betartását is biztosította.

A második struktúra esetében nem hagytam ki a kvadratikus programozást, hanem egy kezdőértéket prediktált a háló, hogy az optimalizáció ne minden időlépésben nulláról induljon. Ezt a struktúrát két részre osztottam. Az egyik megközelítés amikor a neurális háló pontosan a közel optimális Lagrange-szorzók értékét becsüli meg. Mivel a probléma 16 kimenetű volt, egy nagyobb, háromrétegű, több neuront tartalmazó MLP-t alkalmaz-

tam ReLU aktivációs függvénnyel. A másik esetben a hálózat előrejelzi, mely kényszerek lesznek aktívak az adott időpillanatban, így a probléma zárt alakban megoldható, tovább gyorsítva az optimalizációt. Ez egy bináris osztályozási feladat volt, amelyhez egy kétrétegű MLP-t használtam szigmoid aktivációs függvénnyel. Az algoritmusok teszteléséhez három különböző referencia pályát hoztam létre: egy klotoid átmenetekkel rendelkező állandó refernciájú pályát, egy szinuszhullám alapú pályát, valamint egy lineárisan csökkenő és növekvő referencia pályát.

A tanítási és tesztelési train/test loss diagrammok alapján megállapítható, hogy mindhárom hálózat kis predikciós hibával pontosan betanult. Az első struktúra esetében a neurális hálózat nemcsak a kényszerek betartását biztosította, hanem pontosabb referenciajelkövetést értem el, és egy nagyságrenddel tudta csökkenteni a futási időt. Megfigyelhető volt továbbá, hogy a hálózat ellentétes hatást fejt ki az MPC-hez képest, nem engedve annak elintegrálódását. Nagy referencia esetén, amikor az MPC önmagában nem tudta megoldani a problémát, a hálózattal kiegészített megoldás a kényszerek kis mértékű megszegésével is stabil maradt, ezáltal empirikusan következtetve a stabilitást.

A Lagrange-szorzók egzakt értékének és az aktív kényszerek becslésének kiértékeléséhez a kvadratikus programozás iterációszámának összehasonlítását választottam, összehasonlítva a jól ismert Warm Start megoldással. Megállapítottam, hogy mindhárom algoritmus jelentősen csökkentette az iterációk számát. A konkrét lambda értékek becslésével kis mértékű javulást értem el a Warm Starthoz képest, különösen a referenciajel hirtelen változásai esetén. Az aktív kényszerek becslését végző megoldás messze felülmúlta a másik két alkalmazást. Ebben az esetben a hálózat képes volt pontosan előre jelezni, mely kényszerek lesznek aktívak, így az algoritmus legtöbbször egyetlen lépésben megoldotta a problémát. A stabilitási garanciák szempontjából, feltételezve, hogy az eredeti rendszer stabil volt, és mivel csak kezdőértéket adunk a kvadratikus programozásnak, a rendszer nem válik instabillá, az ábrákon is látható, hogy a legrosszabb esetben is az eredeti MPC maximális iterációszámát éri el.

A jövőbeni tervek között szerepel a stabilitás további vizsgálata, különösen az első struktúra esetében, ahol a neurális hálózat párhuzamos szabályozóként működik. Itt a stabilitás bizonyítása összetett feladat, gyakran a két szabályozó közötti súlyozással lehet bizonyítani, hogy ha az egyik minimális vagy maximális súly esetén stabil, akkor a különböző arányú súlyok esetén is stabil marad a rendszer. Egy másik lehetséges továbbfejlesztési irány, a járműmodellben elhanyagolt nem irányítható "előretekintett" görbület beépítése az MPC-be, ami lehetővé tenné a referenciahatás explicit figyelembevételét a modellben. Ezzel javítható a szabályozás pontossága és csökkenthethető a kiugró értékek előfordulása. Továbbá, az algoritmusok CarMaker szimulációs környezetbe való implementálása folyamatban van. Amennyiben ott is sikerül eredményeket elérni, a következő cél a valós járműves implementáció és tesztelés lesz, amellyel egy valós idejű alkalmazásra alkalmas új szabályozási struktúrát hozhatunk létre.

Irodalom

[1] Árpád Fehér, Szilárd Aradi és Tamás Bécsi. „Hierarchical evasive path planning using reinforcement learning and model predictive control". *IEEE Access* 8 (2020), 187470–187482. old.

[2] Mohammad Rokonuzzaman és tsai. „Learning-based model predictive control for path tracking control of autonomous vehicle". *2020 IEEE international conference on systems, man, and cybernetics (SMC)*. IEEE. 2020, 2913–2918. old.

[3] Yannic Vaupel és tsai. „Accelerating nonlinear model predictive control through machine learning". *Journal of process control* 92 (2020), 261–270. old.

[4] Mazen Alamir. *A pragmatic story of model predictive control: self-contained algorithms and case-studies*. CreateSpace Independent Publishing Platform, 2013.

[5] Frank Allgower, Rolf Findeisen, Zoltan K Nagy és tsai. „Nonlinear model predictive control: From theory to application". *Journal-Chinese Institute Of Chemical Engineers* 35.3 (2004), 299–316. old.

[6] Victor M Zavala és Lorenz T Biegler. „The advanced-step NMPC controller: Optimality, stability and robustness". *Automatica* 45.1 (2009), 86–93. old.

[7] T. Peni és J. Bokor. „Robust Model Predictive Control for Controlling Fast Vehicle Dynamics". *2006 14th Mediterranean Conference on Control and Automation*. 2006, 1–5. old. DOI: 10.1109/MED.2006.328864.

[8] Alberto Bemporad, Francesco Borrelli, Manfred Morari és tsai. „Model predictive control based on linear programming˜ the explicit solution". *IEEE transactions on automatic control* 47.12 (2002), 1974–1985. old.

[9] Paolo Falcone és tsai. „Linear time-varying model predictive control and its application to active steering systems: Stability analysis and experimental validation". *International Journal of Robust and Nonlinear Control: IFAC-Affiliated Journal* 18.8 (2008), 862–875. old.

[10] Hui Pang és tsai. „A practical trajectory tracking control of autonomous vehicles using linear time-varying MPC method". *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering* 236.4 (2022), 709–723. old.

[11] Olivier Sename, Peter Gaspar és József Bokor. *Robust control and linear parameter varying approaches: application to vehicle dynamics*. 437. köt. Springer, 2013.

[12] Balázs Németh és tsai. „LPV-based Control Design of an Adaptive Cruise Control System for Road Vehicles". *IFAC-PapersOnLine* 48.14 (2015). 8th IFAC Symposium on Robust Control Design ROCOND 2015, 62–67. old. ISSN: 2405-8963. DOI: https://doi.org/10.1016/j.ifacol.2015.09.434. URL: https://www.sciencedirect.com/science/article/pii/S2405896315015530.

[13] Arthur L Samuel. „Some studies in machine learning using the game of checkers". *IBM Journal of research and development* 3.3 (1959), 210–229. old.

[14] Frank Rosenblatt. „The perceptron: a probabilistic model for information storage and organization in the brain." *Psychological review* 65.6 (1958), 386. old.

[15] Junjie Peng és tsai. „Machine Learning Techniques for Personalised Medicine Approaches in Immune-Mediated Chronic Inflammatory Diseases: Applications and Challenges". *Frontiers in Pharmacology* 12 (2021. szept.). DOI: 10.3389/fphar.2021.720694.

[16] Jiawei Han, Yandong Cai és Nick Cercone. „Data-driven discovery of quantitative rules in relational databases". *IEEE transactions on Knowledge and Data Engineering* 5.1 (1993), 29–40. old.

[17] Geoffrey E Hinton, Simon Osindero és Yee-Whye Teh. „A fast learning algorithm for deep belief nets". *Neural computation* 18.7 (2006), 1527–1554. old.

[18] Sergey Sergeevich Shadrin, Oleg Olegovich Varlamov, Andrey Mikhailovich Ivanov és tsai. „Experimental autonomous road vehicle with logical artificial intelligence". *Journal of advanced transportation* 2017 (2017).

[19] Dániel Fényes, Balázs Németh és Péter Gáspár. „LPV-based autonomous vehicle control using the results of big data analysis on lateral dynamics". *2020 American Control Conference (ACC)*. 2020, 2250–2255. old. DOI: 10.23919/ACC45564.2020.9147548.

[20] Sandor M Veres és tsai. „Autonomous vehicle control systems—a review of decision making". *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering* 225.2 (2011), 155–195. old.

[21] Dong Li és tsai. „Reinforcement Learning and Deep Learning Based Lateral Control for Autonomous Driving [Application Notes]". *IEEE Computational Intelligence Magazine* 14.2 (2019), 83–98. old. DOI: 10.1109/MCI.2019.2901089.

[22] Lukas Hewing és tsai. „Learning-based model predictive control: Toward safe learning in control". *Annual Review of Control, Robotics, and Autonomous Systems* 3.1 (2020), 269–296. old.

[23] Dario Piga és tsai. „Performance-oriented model learning for data-driven MPC design”. *IEEE control systems letters* 3.3 (2019), 577–582. old.

[24] Armin Norouzi és tsai. „Integrating machine learning and model predictive control for automotive applications: A review and future directions”. *Engineering Applications of Artificial Intelligence* 120 (2023), 105878. old.

[25] Gowtham Garimella és tsai. „Neural network modeling for steering control of an autonomous vehicle”. *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2017, 2609–2615. old.

[26] Paul Drews és tsai. „Aggressive deep driving: Combining convolutional neural networks and model predictive control”. *Conference on Robot Learning*. PMLR. 2017, 133–142. old.

[27] Miao Liu és tsai. „Gaussian processes for learning and control: A tutorial with examples”. *IEEE Control Systems Magazine* 38.5 (2018), 53–86. old.

[28] Yajie Bao, Javad Mohammadpour Velni és Mahdi Shahbakhti. „An online transfer learning approach for identification and predictive control design with application to RCCI engines”. *Dynamic systems and control conference*. 84270. köt. American Society of Mechanical Engineers. 2020, V001T21A003.

[29] Lucas Castro Sousa és Helon Vicente Hultmann Ayala. „Nonlinear tire model approximation using machine learning for efficient model predictive control”. *IEEE Access* 10 (2022), 107549–107562. old.

[30] Roman Kohút és tsai. „Hidden Markov model-based warm-start of active set method in model predictive control”. *2021 23rd International Conference on Process Control (PC)*. IEEE. 2021, 60–65. old.

[31] Liting Sun és tsai. „A fast integrated planning and control framework for autonomous driving via imitation learning”. *Dynamic Systems and Control Conference*. 51913. köt. American Society of Mechanical Engineers. 2018, V003T37A012.

[32] Xiaojing Zhang, Monimoy Bujarbaruah és Francesco Borrelli. „Safe and near-optimal policy learning for model predictive control using primal-dual neural networks”. *2019 American Control Conference (ACC)*. IEEE. 2019, 354–359. old.

[33] Kim P Wabersich és tsai. „Probabilistic model predictive safety certification for learning-based control”. *IEEE Transactions on Automatic Control* 67.1 (2021), 176–188. old.

[34] Rajesh Rajamani. *Vehicle dynamics and control*. Springer Science & Business Media, 2011.

[35] Diederik P Kingma. „Adam: A method for stochastic optimization”. *arXiv preprint arXiv:1412.6980* (2014).

Ábrák jegyzéke

2.1. A gépi tanulás három fő típusa. Forrás: [15] 6
2.2. MPC és machine learning kombinációinak a lehetőségei. Forrás: [24] 8

3.1. A dinamikai bicikli modell geometriája és a laterális hiba. 11

5.1. A korlátos beavatkozó jel becslésére létrehozott NN-MPC1 struktúra 27
5.2. Lagrange-szorzók kezdő értékének NN-MPC2, és az aktív kényszerek becslésének NN-MPC3 a struktúrája 29

6.1. Az algoritmusok teszteléséhez használt referencia pályák. 31
6.2. Train/ test loss a korlátos beavatkozó jel prediktálása esetén 32
6.3. Az MPC és az NN-MPC1 performanciája 8-as referencia esetén 33
6.4. Az MPC és az NN-MPC1 performanciája lineárisan változó referencia esetén 34
6.5. Az MPC és az NN-MPC1 performanciája szinusz hullám referencia esetén . 34
6.6. Az MPC és az NN-MPC1 performanciája magas referencia esetén 35
6.8. Train/ test loss az aktív korlátok prediktálása esetén 36
6.7. Train/ test loss a pontos lambda prediktálása esetén 36
6.9. Kvadratikus programozás iterációs száma, a 8-as referencia esetén 37
6.10. Kvadratikus programozás iterációs száma, a szinuszhullám referencia esetén 38
6.11. Kvadratikus programozás iterációs száma, a lineárisan növekvő és csökkenő referencia esetén 39