



DIPLOMATERVEZÉSI FELADAT

Klaffl Attila András

Mechatronikai mérnök hallgató részére

Korszerű rajintelligencia módszerek vizsgálata villamos hajtások szabályozásában

Az elmúlt évtizedben modell prediktív szabályozási módszereket egyre szélesebb körben kezdtek el alkalmazni teljesítményelektronikai átalakítókról táplált villamos hajtásokban. Prediktív szabályozási technikák esetén egy matematikai modell segítségével előre megjósolható a hajtás jövőbeni viselkedése, ami alapján kiszámítható, hogy a működés adott kritériumai és korlátai függvényében melyik a legjobb következő kapcsolás. A számítási intelligencia az utóbbi 15-20 évben rengeteg fejlődésen ment keresztül, egyre szerteágazóbb a bio-inspirált számítási technikák fejlődése és egyre több igény is jelentkezik hatékony algoritmusokra a mérnöki tervezés különböző, egyre speciálisabb területeiről. A raj intelligencia egy modern mesterséges intelligencia diszciplína, melynek működése analóg olyan biológiai mechanizmusokkal, ahol sok egyedből álló populációk egyedei külön-külön öncélú viselkedést tanúsítva a populáció egészében megjelenik egy kollektív intelligens viselkedés.

A hallgató feladatának a következőkre kell kiterjednie:

- Végezzen irodalomkutatást a villamos hajtásokban alkalmazott modell prediktív szabályozások területén!
- Végezzen irodalomkutatást a rajintelligencia módszerek területén!
- Valósítsa meg Matlab/Simulink környezetbe az állandómágneses gép modell prediktív szabályozó algoritmusát és végezzen szimulációs vizsgálatokat!
- Alkalmazzon különböző rajintelligencia módszereket az optimális feszültségektor meghatározásához!
- Implementálja a kiválasztott algoritmust a rendelkezésre álló mikrovezérlőn vagy DSP-n. Tesztelje a szabályozás valós idejű működését.
- Végezzen méréseket, a kapott eredményeket hasonlítsa össze a szimulációs eredményekkel.

Tanszéki konzulens: Dr. Stumpf Péter Pál, egyetemi docens

Budapest, 2025. március 10.

Dr. Charaf Hassan
egyetemi tanár
tanszékvezető





Budapesti Műszaki és Gazdaságtudományi Egyetem

Villamosmérnöki és Informatikai Kar
Automatizálási és Alkalmazott Informatikai
Tanszék

Klaffl Attila András

**KORSZERŰ RAJINTELLIGENCIA
MÓDSZEREK VIZSGÁLATA
VILLAMOS HAJTÁSOK
SZABÁLYOZÁSBAN**

KONZULENS

Dr. Stumpf Péter Pál

BUDAPEST, 2025

Tartalomjegyzék

| | |
|---|-----------|
| Összefoglaló | I |
| Abstract..... | II |
| 1 Bevezetés | 1 |
| 2 Elméleti áttekintés..... | 3 |
| 2.1 Állandómágneses szinkronmotor (PMSM) | 3 |
| 2.1.1 PMSM matematikai modellje | 4 |
| 2.2 Háromfázisú kétszintű feszültséginverter | 10 |
| 2.3 Térvektor-moduláció | 11 |
| 2.4 Kálmán-szűrő | 12 |
| 3 Modell prediktív irányítások | 15 |
| 3.1 Elméleti háttér..... | 16 |
| 3.2 Költségfüggvény | 18 |
| 3.3 MPC villamos hajtásokban | 20 |
| 3.3.1 Predikciós modell felállítása | 22 |
| 3.3.2 MPC szabályozás hatásvázlata | 24 |
| 3.3.3 MPC szimulációja MATLAB környezetben | 25 |
| 3.3.4 Szimulációs eredmények | 27 |
| 4 Korszerű rajintelligencia módszerek | 31 |
| 4.1 Áttekintés | 31 |
| 4.2 Rajintelligencia alapú optimalizálási algoritmusok | 32 |
| 4.2.1 Részecske-rajintelligencia optimalizálás (PSO) | 32 |
| 4.2.2 Mesterséges méhraj optimalizálás (ABC) | 36 |
| 4.2.3 Szürkefarkas optimalizálás (GWO) | 40 |
| 4.3 Optimalizálási módszerek összehasonlítása | 43 |
| 5 Modell prediktív irányítás rajintelligencia módszer alapú optimalizálással | 47 |
| 5.1 Kezdeti állapot és célfüggvény | 47 |
| 5.2 Egyszerűsített Kálmán-szűrő zajos mérésekre..... | 49 |
| 5.3 Véletlenszám generálás..... | 52 |
| 5.4 Szimuláció MATLAB környezetben | 53 |
| 5.5 Valós hardveres implementáció | 58 |
| 5.5.1 Prediktív irányítás C nyelvű megvalósítása..... | 60 |

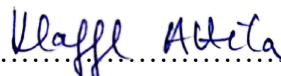
| | |
|--|-----------|
| 5.5.2 Mérési eredmények és következtetések | 63 |
| 6 Konklúzió..... | 74 |
| 7 Irodalomjegyzék..... | 76 |
| 8 Ábrajegyzék..... | 79 |
| Függelék..... | 81 |

HALLGATÓI NYILATKOZAT

Alulírott **Klaffl Attila András**, 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 átvettettem, egyértelműen, a forrás megadásával megjelöltetem.

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 hitelesített 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.

Kelt: Budapest, 2025. 12. 11.



Klaffl Attila András

Összefoglaló

A diplomamunka célja, hogy megvizsgálja a korszerű optimalizálási módszerek alkalmazhatóságát modern villamos hajtásokban. Ezen technikák egyike az úgynevezett a rajintelligencia-alapú algoritmusok. A dolgozatban állandómágneses szinkrongép esetében kutattam, miképpen lehetséges már ismert, klasszikus szabályozási módszereket a természet ihlette önszerveződő algoritmusokkal kombinálni. A cél, hogy elemezzem, milyen hatékonysággal lehet sztochasztikus keresési módszerek segítségével a rendszer dinamikáját optimalizálni, biztosítva a stabil és költséghatékony működést.

A rajintelligencia-alapú algoritmusok, úgymint a részecske rajintelligencia, a mesterséges méhraj algoritmus és a szürkefarkas optimalizáció, lehetőséget kínálnak olyan problémák megoldására, ahol a gyors beavatkozás iránti igény és az optimális irányítás nemlineáris feltételei megnehezítik a hagyományos szabályozási módszerek alkalmazását. Ezek az algoritmusok az önszervező és kollektív viselkedés elveit építik be mérnöki alkalmazásokba, így hozzájárulva egy intelligens és megbízható szabályozási rendszer kialakításához.

A dolgozatomban áttekintem a korszerű optimalizálási módszerek irodalmát, majd elemzem ezen módszerek hatékonyságát és a rendszer stabilitására gyakorolt hatását mind szimulációs környezetben, mind valós hardveren végzett vizsgálatok keretében. A kutatási célkitűzés az, hogy a dolgozat rámutasson a modern irányítástechnika és a természet ihlette algoritmusok ötvözésének lehetőségeire a mechatronikai rendszerek fejlesztésében, és bemutassa, hogy ezek a módszerek miként segíthetnek komplex hajtásrendszerek tervezésében és optimalizálásában.

Abstract

The aim of the master's thesis is to examine the applicability of modern optimization methods in contemporary electric drives. One such technique is the family of swarm-intelligence-based algorithms. In this work, I investigated how classical, well-known control methods can be combined with nature-inspired self-organizing algorithms in the case of a permanent magnet synchronous machine. The goal is to analyze how effectively stochastic search methods can be used to optimize the system dynamics while ensuring stable and cost-efficient operation.

Swarm-intelligence-based algorithms, such as Particle Swarm Optimization, the Artificial Bee Colony algorithm, and Grey Wolf Optimization, offer solutions for problems where the demand for rapid intervention and the nonlinear conditions of optimal control make the application of conventional control methods difficult. These algorithms incorporate the principles of self-organization and collective behavior into engineering applications, thereby contributing to the development of an intelligent and reliable control system.

In the thesis, I review the literature on modern optimization methods, then analyze the efficiency of these methods and their impact on system stability both in a simulation environment and through experiments conducted on real hardware. The research objective is to highlight the potential of combining modern control engineering with nature-inspired algorithms in the development of mechatronic systems, and to demonstrate how these methods can support the design and optimization of complex drive systems.

1 Bevezetés

A téma választásom célja, hogy villamos hajtások szabályozását új és modern módszerekkel vizsgáljam. Különösen érdekesnek találtam azon keresési módszerek adaptálását ezen hajtásokba, amelyek egyfajta intelligencia inspirálta elvet alkalmaznak. Úgy gondolom, ezen irányok új lehetőséget teremthetnek ezen a területen. Emellett jól illeszkednek korszerű mechatronikai rendszerekhez, hiszen a villamos hajtástechnikai fejlesztések ma kiemelt hangsúlyt kapnak. A modern rendszerekben, a háztartási eszközöktől az ipari berendezéseken át a biztonságkritikus energetikai alkalmazásokig, egyre nagyobb szerepet kap az energiahatékonyság, a megbízhatóság és az automatizált működés kérdése. Ennek megfelelően a villamos motorok mára az intelligens mechatronikai rendszerek aktív beavatkozóivá váltak.

A villamos hajtások vezérléstechnikájának fejlődése szoros kapcsolatban állt a félvezetőipar és technológiák rohamos növekedésével. A korszerű átalakítóknak és mikrovezérlőknek köszönhetően a motorokkal képesek voltak nagyobb dinamikával és pontossággal bíró szabályzásokat megvalósítani. A számítási kapacitás és memória növekedésével új lehetőségek nyíltak meg. A fejlett hardveres háttérrel már nem okozott gondot bonyolultabb és összetettebb algoritmusok implementálása, lehetővé téve, hogy a mai kor elvárásai már sokkal inkább az energiahatékonyság és a fenntarthatóság irányába mutatnak. Ez mérnöki szemmel egy optimális működés tervezését, míg matematikailag optimalizációs eljárások kidolgozását jelenti. Ahogyan a diplomamunka későbbi részében látni fogjuk, már az 1970-es években voltak törekvések egy működés szempontjából optimális irányítás megvalósítására, azonban az akkoriban rendelkezésre álló hardveres közök még nem tették lehetővé összetettebb módszerek használatát és gyorsan változó rendszerekben történő felhasználását. Ezzel párhuzamosan a számítási kapacitás növekedése, valamint kommunikációs hálózatok fejlődése lehetővé tette a valós idejű adatfeldolgozást és a decentralizált vezérlési architektúrák kialakítását.

Az utóbbi években az adatközpontú technológiák térnyerésével megjelentek a mesterséges intelligencia alapú vezérlési megközelítések. Ezek az új módszerek lehetőséget kínálnak a hagyományos modellalapú szabályozások kiegészítésére vagy részbeni kiváltására, különösen olyan esetekben, amikor a rendszer pontos matematikai leírása nehezen megadható. Ezekkel egyidejűleg megjelentek olyan megközelítések,

amelyek a természetben megfigyelhető kollektív viselkedési mintázatokon alapulnak. A kutatók ezért a különböző fajok – például madarak, méhkolóniák, hangyák – önszervező és csoportos viselkedésének vizsgálata felé fordultak. A legkorábbi kutatások és matematikai leírások még a 20. század második felére nyúlnak vissza, szélesebb körben ismertté és alkalmazottá csak a 2000-es években vált, elsősorban a már említett számítási kapacitásnövekedés végett.

A rajintelligencia módszerek a villamos hajtások területén is új lehetőségeket kínálnak. Az állatok kollektív, döntéshozatalon és önszervező viselkedésen alapuló elvek hozzájárulnak a jövő intelligens és megbízható hajtásrendszereinek kialakításához, amelyek képesek a környezetükkel együttműködve hatékonyan, stabilan és energiahatékonyan működni.

Az első fejezet betekintést nyújt a villamos hajtások fejlődésébe és ismerteti a modern szabályozási módszerek kialakulását és jelentőségét. A második fejezet az állandómágneses szinkrongépre vonatkozó elméleti alapokat tárgyalja, valamint bemutatja a dolgozat további részeiben és a szabályozókör megtervezésében felhasznált egyéb technikák matematikai leírását, a koordinátranszformációtól egészen a Kálmán-szűrő alapú becslésig.

Ahogy említettük, a tanulmány a modell prediktív irányítások témakörére épít, így a harmadik fejezetben először ennek elméleti háttere kerül bemutatásra. Továbbá ismerteti annak előnyeit a klasszikus szabályozásokkal szemben. A fejezet végén szimulációban mutattam be a hatékonyságát.

A negyedik fejezet a korszerű rajintelligencia-módszereket mutatja be. Először általánosságban vizsgáljuk ezen technikák felépítését, működési elveit és alapvető jellemzőit. Majd ezt követően konkrét algoritmusok kerülnek részletes bemutatásra, mint például a részecske raj, a mesterséges méhraj és a szürkefarkas optimalizációs algoritmusok. A fejezet kifejti a fent említett módszerek előnyeit és korlátait, valamint elemzi alkalmazhatóságukat mérnöki problémák megoldásában.

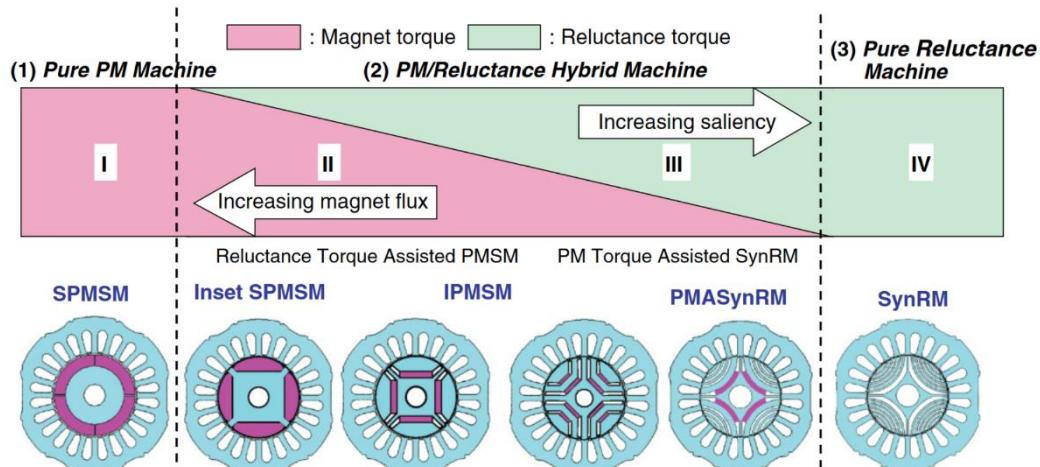
Az ötödik fejezet ezeket az algoritmusokat integrálja a harmadik fejezetben megépített szabályozás optimalizálásába. A vizsgálatok célja rendszer dinamikus viselkedésének megfigyelése. A fejezet továbbá bemutatja a valós hardveres megvalósítást, és ismerteti a szabályozás eredményeit a tényleges rendszer dinamikus viselkedésén.

2 Elméleti áttekintés

2.1 Állandómágneses szinkronmotor (PMSM)

A permanens mágneses szinkronmotorokat (PMSM) széles körben alkalmazzák magas hatásfokuk és kedvező nyomatéki tulajdonságai miatt. Megtalálhatóak különböző iparágakban, háztartási gépekben, ipari alkalmazásokban és elsősorban elektromos járművek hajtásrendszereiben. A motor gerjesztését a rotorban elhelyezett állandómágnesek szolgáltatják, így nincsen szükség külön gerjesztőáramra, amely hozzájárul az alacsonyabb energiafogyasztáshoz és nagyobb hatékonyságú működéshez. A PMSM szinuszos feszültséget (back EMF) állít elő, amely megfelelő vezérléssel egyenletesebb nyomatékot és alacsonyabb nyomatékingadozást eredményezhet. A PMSM rotor és állórész felépítése sok tekintetben hasonlít a kefe nélküli egyenáramú motoréhoz (BLDC), azonban az állórész háromfázisú tekercselése az aszinkron motorokhoz hasonlóan szinuszos mágneses mezőt hoz létre a légrésben. A rotor pólusai követik ezt a forgó mezőt, így nyomaték keletkezik. [6][23]

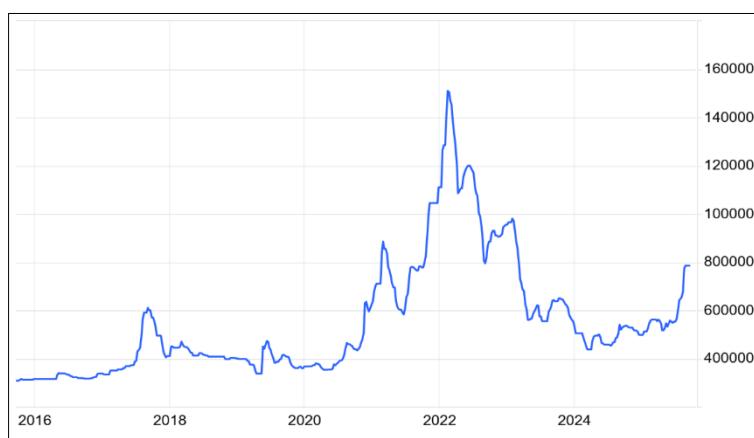
Az állandómágneses szinkrongépeket a rotor kialakítása szerint alapvetően kettő csoportba sorolhatjuk: vannak felületre szerelt és süllyesztett/belső mágneses kivitelű motorok. A felületre szerelt konstrukció egyszerűbb gyártást és alacsonyabb költséget igényel. Hátránya, hogy a reluktancia nyomaték kihasználására nem, vagy csak nagyon kis mértékben alkalmas, így a teljes nyomaték kizárolag a mágnesekből származik. A belső mágneses elrendezésű esetben a mágnesek a rotor belejében kerülnek elhelyezésre. Ez az elrendezés nagyobb mechanikai szilárdságot biztosít magasabb fordulatszámon is.



2.1. ábra PMSM motorok típusai kialakításuk szerint [23]

Továbbá lehetővé teszi a reluktancia nyomaték kihasználását, amely tovább növeli a motor hatékonyságát. [6][23]

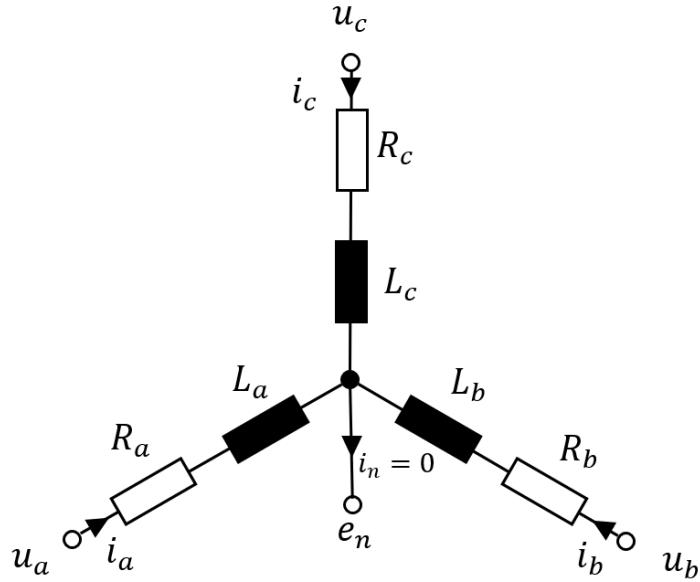
Az 2.1 .ábra szemlélteti a különböző szinkrongépek csoportosítását és az azok közötti átmenetet. A tisztán permanens mágneses gépektől, a reluktancia nyomatéket vegyesen kihasználó hibrid gépeken át a tisztán reluktancia elven működő motorokig terjed a skála. Jól látható, hogy a mágneses nyomaték folyamatosan csökken, míg a reluktancia nyomaték aránya folyamatosan nő, ahogyan egyre nő az induktivitáskülönbség. A komplex rotorkialakítások és ezzel együtt növekvő reluktancia nyomaték aránya lehetőséget nyújt a mágnesanyag felhasználás csökkentésére a hatékonyság lényeges romlása nélkül. A rotorban használt permanens mágnesesek, jellemzően valamilyen ritkaföldfém ből (pl. neodímium) készülnek, ezeket azonban globális szinten jelentős áringadozás jellemzi (2.2. ábra). Mivel a ritkaföldfémek túlnyomó részét, kb. 85 %-át Kína termeli, ezért az árakat a világpiaci és geopolitikai tényezők erősen befolyásolják. Az ellátási lánc zavarai, például geopolitikai feszültségek és pandémiák miatti megszorítások komoly hatással vannak az árak alakulására. Mindezen tényezők miatt a fejlesztések egyre inkább irányulnak alternatív mágneses anyagok és ötvözletek létrehozására, amelyek csökkenthetik vagy akár teljesen ki is válthatják a neodímium használatát. [23]



2.2. ábra Neodímium világpiaci árának alakulása az elmúlt 10 évben

2.1.1 PMSM matematikai modellje

Ebben az alfejezetben a háromfázisú inverterrel megtáplált állandómágneses szinkrongép matematikai modellje kerül bemutatásra, amely a későbbi szabályozási rendszer alapját képezik.



2.3. ábra Háromfázisú, csillagkapcsolású PMSM elektromos modellje

A 2.3. ábra a háromfázisú, csillagkapcsolású szinkrongép helyettesítőképét szemlélteti. A motor viselkedése a tekercsek villamos paramétereinek segítségével leírható. Az állórész tekercselései egyenként egy ohmos ellenállás és induktivitás sorba kapcsolásával helyettesíthetőek. A csillagponti feszültséget, ahogy a fenti ábra is mutatja, külön jelöltük. Az egyes fázisokra vonatkozó hurokegyenleteket a (2.1) – (2.3) egyenletek adják meg.

$$u_a = R_a i_a + \frac{d\psi_a}{dt} + e_n \quad (2.1)$$

$$u_b = R_b i_b + \frac{d\psi_b}{dt} + e_n \quad (2.2)$$

$$u_c = R_c i_c + \frac{d\psi_c}{dt} + e_n \quad (2.3)$$

ahol u_a, u_b, u_c az állórész fázisfeszültségeit, i_a, i_b, i_c a fázisáramokat, míg ψ_a, ψ_b, ψ_c a megfelelő fázisok csatolt fluxusát jelentik. Továbbá e_n a csillagponti feszültség. A tekercselés ellenállását és induktivitását a későbbiekben R_a, R_b, R_c , illetve L_a, L_b, L_c jelölik.

A szimmetrikus háromfázisú rendszerekben a fázismennyiségeket egyetlen egy vektorral, az úgynevezett térvvektorral (2.4) és egy zérus sorrendű összetevővel (2.5) is

leírhatjuk. Másként *Clarke transzformációnak* is nevezik. Definiáljuk $\mathbf{a} = e^{j\frac{2\pi}{3}}$ vektort, ekkor a fázismennyiségek térvvektorok reprezentációja:

$$\mathbf{x} = \frac{2}{3} [x_a + \mathbf{a}x_b + \mathbf{a}^2x_c] \quad (2.4)$$

$$x_0 = \frac{x_a + x_b + x_c}{3} \quad (2.5)$$

ahol x_a, x_b, x_c a fázismennyiségek pillanatértékei.

Így tehát a térvvektoros transzformáció a folyamatosan változó fázismennyiségeket egyetlen álló koordinátarendszerben (ÁKR) értelmezett vektorrá alakítja. Amennyiben az állórész ellenállása, illetve induktivitása fázisonként eltérő, akkor konstansok helyett mátrix szerepel az egyenletben. A (2.1) - (1.2) egyenletekre alkalmazva a Clarke transzformációt:

$$\mathbf{u} = R\mathbf{i} + \frac{d\boldsymbol{\Psi}_s}{dt} \quad (2.6)$$

$$(\text{mivel } \mathbf{e} = \frac{2}{3} e_n [1 + \mathbf{a} + \mathbf{a}^2] = \mathbf{0})$$

A fluxus két fő részből áll, egyrészt az állórész árama által létrehozott forgó mágneses mezőből, másrészt a forgórész mágneses fluxusából ($\boldsymbol{\Psi}_m$). Ezt írja le a (2.7.) egyenlet.

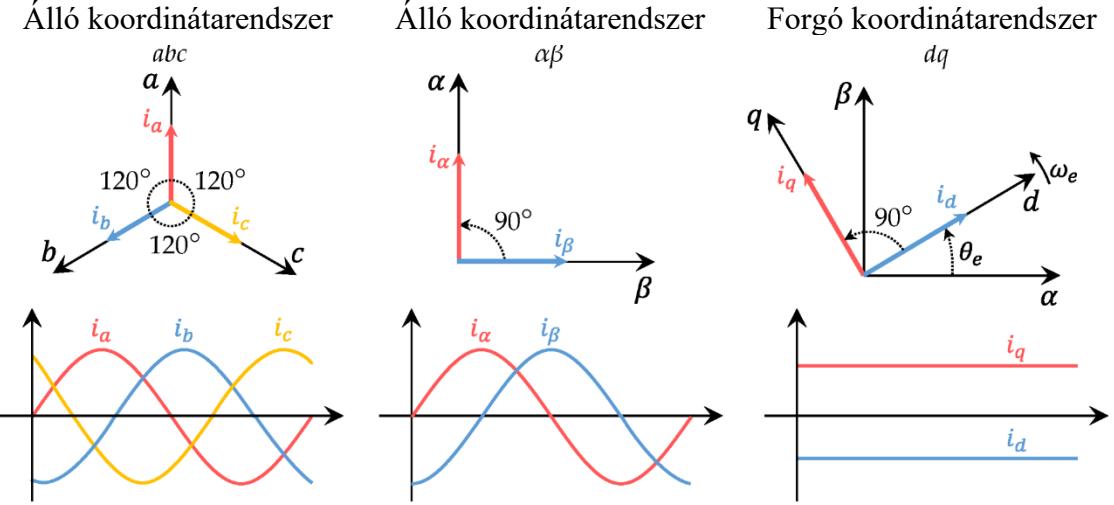
$$\boldsymbol{\Psi}_s = L\mathbf{i} + \boldsymbol{\Psi}_m \quad (2.7)$$

A három váltakozó mennyiségből kettő szinuszos mennyiséget kaptunk. Ez azonban megnehezítené a szabályozás megtervezését, ezért az álló koordinátarendszerből áttérünk egy forgó koordinátarendszerben (FKR), amelyben állandósult állapotban konstans mennyiségeket kapunk. A koordinátarendserek tengelyeit rendszerint $d-q$ betűkkel szokás jelölni. Ezt a vonatkoztatási rendszert úgy választjuk meg, hogy együtt forogjon a rotor által létrehozott mágneses mezővel, és a d tengely egybeessen a mágneses mezőt leíró térvvektorral. A FKR a villamos körfrekvenciával (ω_e) forog, amely szinkrongép esetén a póluspárszám és a mechanikai szögsebesség szorzatával egyezik meg ($p\Omega$). Az egyes koordinátarendserek egymáshoz viszonyított elhelyezkedését az 2.4. ábra szemlélteti.

Írjuk át a (2.6.) egyenletet a közös koordinátarendszerbe:

$$\mathbf{u}^{(k)} e^{j\omega_e t} = R \mathbf{i}^{(k)} e^{j\omega_e t} + \frac{d\psi_s^{(k)} e^{j\omega_e t}}{dt} \quad (2.8)$$

$$\mathbf{u}^{(k)} e^{j\omega_e t} = R \mathbf{i}^{(k)} e^{j\omega_e t} + \frac{d\psi_s^{(k)}}{dt} e^{j\omega_e t} + j\omega_e \psi_s^{(k)} e^{j\omega_e t} \quad (2.9)$$

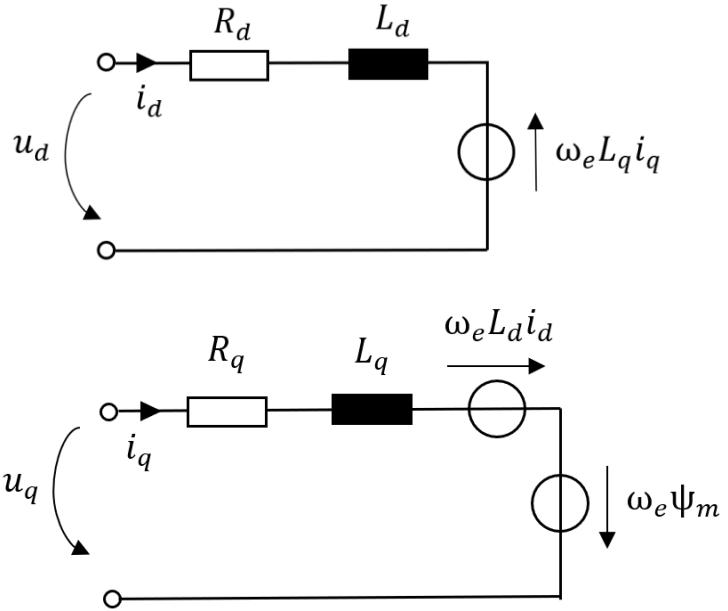


2.4. ábra Különböző koordinátarendszerek orientációja, és áramjelek [24]

A (2.7) egyenletet behelyettesítve és a felső indexet elhagyva:

$$\mathbf{u} = R \mathbf{i} + L \frac{di}{dt} + j\omega_e L \mathbf{i} + j\omega_e \boldsymbol{\Psi}_m \quad (2.10)$$

A motor szabályozása során az áram összetevőit külön-külön vezéreljük, ezért az (2.10) egyenletet célszerű szétbontani d és q irányú komponensekre. Az egyes irányokra vonatkozó helyettesítőképet a 2.5. ábra mutatja.



2.5. ábra A PMSM motor helyettesítő képe [20]

$$u_d = R_i d + L_d \frac{di_d}{dt} - \omega_e L_q i_q \quad (2.11)$$

$$u_q = R_i q + L_q \frac{di_q}{dt} + \omega_e (\psi_m + L_d i_d) \quad (2.12)$$

A motor nyomatékára általánosan az (2.13.) egyenlet adódik.

$$T = \frac{3}{2} p \boldsymbol{\psi}_s \times \mathbf{i} \quad (2.13)$$

A (2.7.) egyenletet a fluxus helyére beírva és kifejtve:

$$T = \frac{3}{2} p (L \mathbf{i} + \boldsymbol{\psi}_m) \times \mathbf{i} = \frac{3}{2} p [(L_d - L_q) i_d + \psi_m] i_q = \frac{3}{2} p \psi_m i_q \quad (2.14)$$

ha $L_d = L_q$.

A fenti egyenletből jól látható, hogy az állandómágneses szinkrongép villamos nyomatéka kettő részből áll. Az első tag a d és q irányú induktivitások különbségéből származó, úgynevezett reluktancia nyomaték. Amennyiben a kettő induktivitás megegyezik, ahogy a dolgozatban felhasznált motor esetén, a reluktancia nyomaték eltűnik és kizárolag a permanensmágnes fluxusából eredő nyomaték marad.

A szabályozó megtervezéséhez, valamint szimulációk elvégzéséhez felhasználhatjuk az egyes szoftverekbe beépített, kész motormodelleket, de a motormodellt mi magunk is megépíthetjük, ami különösen hasznos, ha egyedi paraméterekkel vagy feltételekkel szeretnénk vizsgálni a gép viselkedését. Ehhez célszerű a modellt operátortartományba transzformálni, a hatásvázlat pedig segít az egyes komponensek közötti kapcsolatok szemléltetésére. Az operátortartományba történő áttéréshez a (2.11) és (2.12) egyenleteket, valamint a Laplace-transzformáció jól ismert tulajdonságait használjuk fel. Az s operátort bevezetve és elvégezve a transzformációt az (2.15) és (2.16) egyenleteket kapjuk:

$$U_d = RI_d + sL_d I_d - \Omega_e L_q I_q \quad (2.15)$$

$$U_q = RI_q + sL_q I_q + \Omega_e (\Psi_m + L_d I_d) \quad (2.16)$$

Az egyenleteket célszerű átrendezni az egyes áramkomponensekre. Ezáltal leegyszerűsödik a hatásvázlat felrajzolása, valamint a szabályozónk bemenete az az egyes irányokra vonatkozó áramreferenciát várja.

$$I_d = \frac{U_d + \Omega_e L_q I_q}{R + sL_d} \quad (2.17)$$

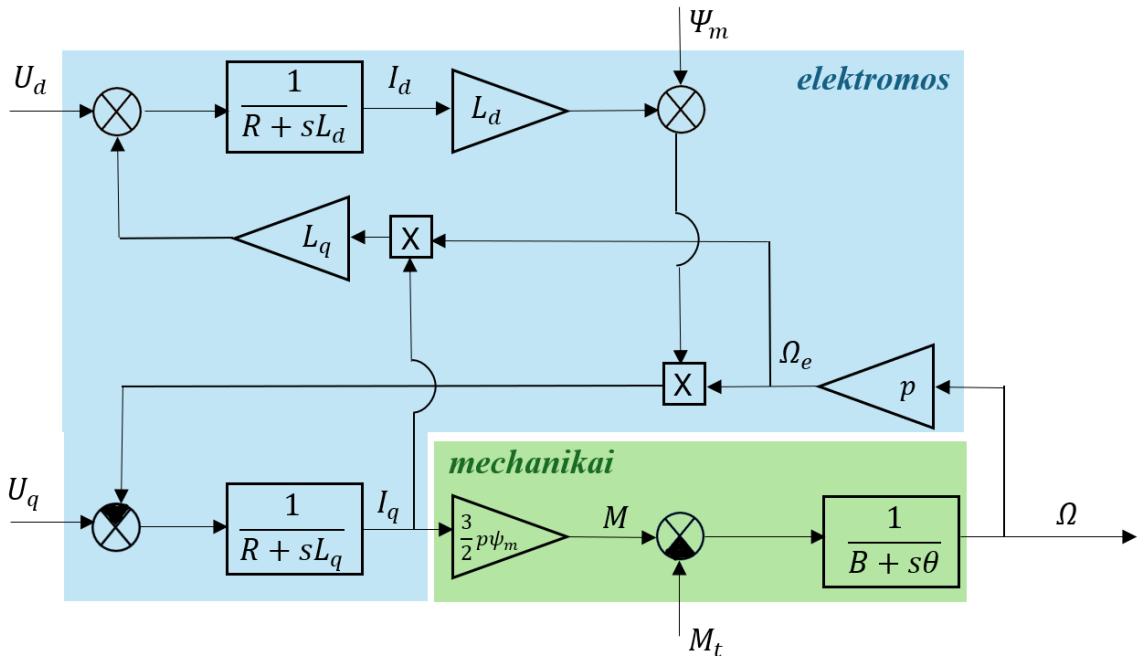
$$I_q = \frac{U_q - \Omega_e (\Psi_m + L_d I_d)}{R + sL_q} \quad (2.18)$$

A motor nyomatékának ismeretében felírhatjuk a motorra vonatkozó mechanikai egyenletet (2.19) viszkózus csillapítást feltételezve (B) és annak Laplace-transzformáltját (2.20).

$$\theta \frac{d\omega_{mech}}{dt} + B \cdot \omega_{mech} = M - M_t = \frac{3}{2} p \psi_m i_q - M_t \quad (2.19)$$

$$s\theta\Omega + B\Omega = \frac{3}{2}p\psi_m I_q - M_t \rightarrow \Omega = \frac{1}{s\theta+B} \left(\frac{3}{2}p\psi_m I_q - M_t \right) \quad (2.20)$$

Az (2.17) és (2.18), valamint a (2.20) egyenletek alapján felrajzolható a PMSM motor hatásvázlata (2.6. ábra).



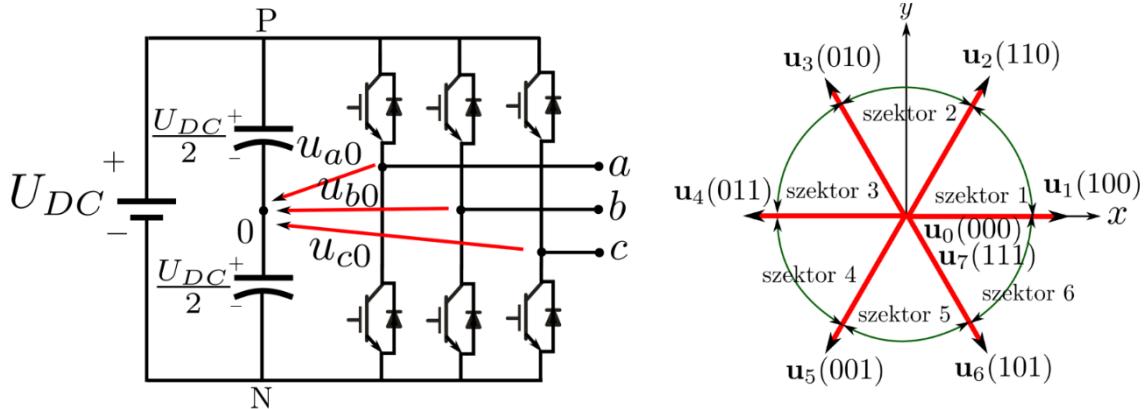
2.6. ábra A PMSM motor hatásvázlata [20]

2.2 Háromfázisú kétszintű feszültséginverter

Az inverter egy olyan vezérelhető átalakító, amely egyenáramú (DC) jelet képes átalakítani váltakozó (AC) jelére. Az inverter bemeneti tápellátása általában egyenirányítóból származik, de egyes alkalmazásokban ez lehet akkumulátor is. Motorhajtások esetében az inverter feladata a váltakozó kimeneti feszültség biztosítása a motor számára. A hatékony működéshez azonban elengedhetetlen a megfelelő vezérlési technika. [12]

A 2.7. ábra egy háromfázisú és kétszintű feszültséginverter sematikus raját mutatja. A nullával jelölt pont fiktív, valóságban nem tudjuk terhelni. A háromfázisú, kétszintű inverter alapesetben egy 8 állásos beavatkozó egység. A kimeneti feszültségek vektorok formában egyszerűen felírható, ha az egyes kimeneti értékekre alkalmazzuk a

(2.4) egyenletben felírt Clarke-transzformációt. Az egyes állapotokat és a velük előállítható kimeneti feszültségvektorokat a 2.8. ábra szemlélteti. Az állapotokhoz tartozó kapcsolótáblát a 2.8. ábra mutatja. A sorok az adott ág felső tranzisztorának állapotát jelentik, 0 esetén a tranzisztor ki van kapcsolva, míg 1-nél a tranzisztor bekapcsolt állapotban van. [12][21]



2.7. ábra Kétszintű, háromfázisú inverter sematikus ábrája és kiadható feszültségvektorok [21]

| Kapcsolójelek | | | Feszültség térvvektor |
|---------------|----------|----------|---|
| S_{a+} | S_{b+} | S_{c+} | \mathbf{u}_{ki} |
| 0 | 0 | 0 | $\mathbf{u}_0 = 0$ |
| 1 | 0 | 0 | $\mathbf{u}_1 = \frac{2}{3}U_{DC}$ |
| 1 | 1 | 0 | $\mathbf{u}_2 = \frac{2}{3}U_{DC}e^{j\frac{\pi}{3}}$ |
| 0 | 1 | 0 | $\mathbf{u}_3 = \frac{2}{3}U_{DC}e^{j\frac{2\pi}{3}}$ |
| 0 | 1 | 1 | $\mathbf{u}_4 = -\frac{2}{3}U_{DC}$ |
| 0 | 0 | 1 | $\mathbf{u}_5 = \frac{2}{3}U_{DC}e^{j\frac{4\pi}{3}}$ |
| 1 | 0 | 1 | $\mathbf{u}_6 = \frac{2}{3}U_{DC}e^{j\frac{5\pi}{3}}$ |
| 1 | 1 | 1 | $\mathbf{u}_7 = 0$ |

2.8. ábra Kétszintű, háromfázisú inverter kapcsolótáblája [21]

2.3 Térvektor-moduláció

A térvektor-moduláció (Space Vector Modulation – SVM) egy gyakran alkalmazott technika szinkron vagy aszinkron gépek meghajtásában. A moduláció feladata, hogy PWM jeleket generáljon az inverter kapcsolóinak vezérléséhez, így előállítva a kívánt sebességhez, illetve nyomatékhoz szükséges feszültségeket az átalakító kimenetén. Az előző részben összefoglaltuk, hogy milyen kimeneti feszültségállapotokat

hozhatunk létre az inverter egyszerű vezérlésével. A valóságban azonban szeretnénk a diszkrét vektorok közötti feszültségeket is kapcsolni. Ezt a problémát oldja meg az SVM. A módszer lényege, hogy a kívánt feszültségvektort a kettő szomszédos és kettő nullvektorral állítjuk elő. Az SVM lényege, hogy meghatározzuk, hogy az egyes kapcsolók a megadott kapcsolási periódusidő mekkora részében legyenek bekapcsolva. Ezt a (2.21) egyenlet alapján írhatjuk fel, ahol \mathbf{u}_k és \mathbf{u}_{k+1} a szomszédos feszültségvektorokat, T_s pedig a kapcsolási periódusidőt jelöli. [12][21]

$$\mathbf{u}^* = \frac{\mathbf{u}_k t_k}{T_s} + \frac{\mathbf{u}_{k+1} t_{k+1}}{T_s} + \frac{\mathbf{u}_0 t_0}{T_s} + \frac{\mathbf{u}_7 t_7}{T_s} \quad (2.21)$$

Az egyes időtartamok összege természetesen meg kell egyezzen a periódusidővel:

$$T_s = t_0 + t_k + t_{k+1} + t_7 \quad (2.22)$$

Ez összesen 3 egyenletet és 4 ismeretlenetet jelent, mivel (2.21) vektoregyenlet. A gyakorlati megvalósításban azonban feltesszük, hogy a kettő nullvektort azonos ideig kapcsoljuk be, hogy a tranzisztorok terhelése egyenletesebb legyen. A $t_0 + t_7$ feltétellel már megoldható az egyenletrendszer. [12][21]

Az SVM működési koncepciójának lényegét a (2.21) és (2.22) egyenletek szemléltetik. A gyakorlatban azonban a modulációt vivőfrekvenciás módszerrel valósítottam meg. Ez a megközelítés ekvivalens az SVM leírásával, ugyanakkor egyszerűbben implementálható.

2.4 Kálmán-szűrő

A valós megvalósítás során az áramjelek jelentős zajt tartalmaztak, ami a tisztán modell alapú szabályozás esetén nagymértékű pontatlanságot eredményezett, és ezáltal nem optimális viselkedéshez vezetett. A mérési zaj hatásának csökkentése érdekében Kálmán-szűrőt alkalmaztam, amely becslés segítségével képes javítani a rendszer predikciós pontosságát. Ezért ebben a fejezetben a Kálmán-szűrő elméleti háttere kerül bemutatásra. [22]

Kálmán Rudolf 1960-ban publikálta a róla elnevezett Kálmán-szűrőt, amely a diszkrét idejű lineáris rendszerek állapotainak rekurzív becslésére szolgál. Megjelenése

óta az egyik legelterjedtebb állapotbecslési technikává vált, különösen az autonóm rendszerekben. Az algoritmus széles körű alkalmazhatóságát nagyban elősegítette a digitális számítástechnika fejlődése, amely lehetővé tette a szűrő valós idejű megvalósítását is. [22]

A Kálmán-szűrő célja egy olyan diszkrét idejű folyamat állapotának becslése, amelyet a következő lineáris differenciaegyenlet ír le:

$$\mathbf{x}_k = \mathbf{A}\mathbf{x}_{k-1} + \mathbf{B}\mathbf{u}_{k-1} + \mathbf{w}_{k-1} \quad (2.23)$$

míg a hozzá tartozó mérési érték az alábbi formában adható meg:

$$\mathbf{z}_k = \mathbf{C}\mathbf{x}_k + \mathbf{v}_k. \quad (2.24)$$

\mathbf{x}_k a rendszer állapotvektora, \mathbf{u}_k a bemeneti vezérlőjel, \mathbf{z}_k pedig a mérés. A \mathbf{w}_k és \mathbf{v}_k tagok a modell- és mérési zajokat jelentik, melyek normál eloszlású, egymástól független véletlen változók, zérus várható értékkel és \mathbf{Q} , illetve \mathbf{R} kovarianciamátrixszal. A mátrixok (\mathbf{A} , \mathbf{B} és \mathbf{C}) rendre az állapotátmenetet, a bemeneti hatást, illetve a mérési kapcsolatot határozzák meg. [22]

A Kálmán-szűrő működése két fő lépésre osztható: predikcióra és korrekcióra. Az predikció során az algoritmus előrejelzi a következő időlépésben várható állapotot és a hiba kovarianciáját. A korrekció során a beérkező mérési adat alapján a szűrő korrigálja ezt az előrejelzést, hogy közelebb kerüljön a valós állapothoz. [22]

Az állapotbecslés alapképlete a következő formában írható fel:

$$\widehat{\mathbf{x}}_k = \widehat{\mathbf{x}}_k^- + \mathbf{K}_k(\mathbf{z}_k - \mathbf{C}\widehat{\mathbf{x}}_k^-), \quad (2.25)$$

ahol $\widehat{\mathbf{x}}_k^-$ az előzetes, modell alapján számított becslés, $\widehat{\mathbf{x}}_k$ a módosított becslés, a $\mathbf{z}_k - \mathbf{C}\widehat{\mathbf{x}}_k^-$ különbség pedig a reziduum, amely a mért és a becsült érték közötti eltérést. A \mathbf{K} mátrix a Kálmán-erősítés, amely meghatározza, milyen mértékben „bízik” a szűrő az új mérésben az előrejelzéshez képest. [22]

A Kálmán-erősítés optimális értéke a hiba kovarianciamátrixának (\mathbf{P}_k^-) minimalizálásából vezethető le:

$$\mathbf{K}_k = \mathbf{P}_k^- \mathbf{C}^T (\mathbf{C} \mathbf{P}_k^- \mathbf{C}^T + \mathbf{R})^{-1}. \quad (2.26)$$

Amennyiben a mérési zaj kovarianciája \mathbf{R} kicsi, azaz a mérés megbízható, a Kálmán-erősítés nagy lesz, így a szűrő erősen támaszkodik mérésre. Ideális esetben az állapot megegyezik a szenzor felől érkező jellet. Ezzel szemben, ha a mérési zaj jelentős, a szűrő inkább a modellt veszi alapul. [22]

Ahogy korábban láthattuk, a Kálmán-szűrő algoritmusának lépései a következők:

Predikció:

$$\hat{\mathbf{x}}_k^- = \mathbf{A} \hat{\mathbf{x}}_{k-1} + \mathbf{B} \mathbf{u}_{k-1}, \quad (2.27)$$

$$\mathbf{P}_k^- = \mathbf{A} \mathbf{P}_{k-1} \mathbf{A}^T + \mathbf{Q} \quad (2.28)$$

Korrekción:

$$\mathbf{K}_k = \mathbf{P}_k^- \mathbf{C}^T (\mathbf{C} \mathbf{P}_k^- \mathbf{C}^T + \mathbf{R})^{-1}, \quad (2.29)$$

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}_k (\mathbf{z}_k - \mathbf{C} \hat{\mathbf{x}}_k^-), \quad (2.30)$$

$$\mathbf{P}_k = (\mathbf{I} - \mathbf{K}_k \mathbf{C}) \mathbf{P}_k^- \quad (2.31)$$

Az algoritmus a folyamat során minden időlépésben a korábbi becsléseket használja. Ez teszi a Kálmán-szűrőt hatékonnyá és alkalmassá valós idejű alkalmazásokra. A szűrő tehát összefoglalva egyfajta visszacsatolt rendszerként működik, amely a modell alapján előrejelzi a rendszer állapotát, majd a mérések segítségével korrigálja azt, folyamatosan közelítve a valós fizikai folyamat viselkedéséhez. [22]

3 Modell prediktív irányítások

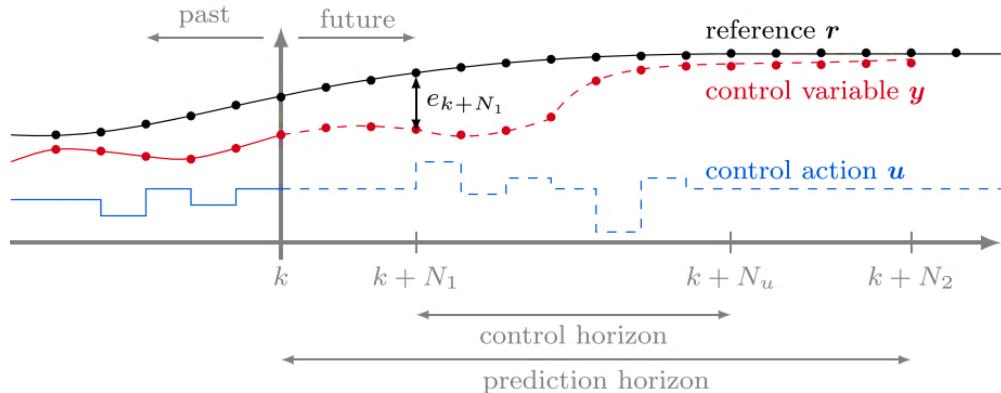
A zárt hurkú szabályozási rendszerekben a referencia jel és a szabályozott jellemző különbségből származó hibajel alapján a szabályozó előírja a szükséges beavatkozó jelet. A szabályozó működése alapján megkülönböztethetünk klasszikus, valamint prediktív irányítási módszereket. A klasszikus szabályozások általában egybemenetű és egy kimenetű (SISO) rendszerekre optimalizáltak. A szabályozók tervezése rendszerint operátortartományban történik, ehhez viszont szükséges a szakasz lineáris modellje. A paraméterek meghatározása történhet hangolással is, az irodalomban számos módszer megtalálható, az egyik legismertebb például a Ziegler-Nichols metódus. Ipari területeken szintén kiemelkedő jelentőséggel bír és széles körben alkalmazott a PID szabályozó. Fontos megjegyezni, hogy a megfelelő beállítás nem mindenkor magától értetődő, különösképpen nemlineáris, valamint idővariáns rendszerek esetén. Ennek ellenére a klasszikus lineáris szabályozók továbbra is jó és megbízható megoldást jelentenek a felmerülő mérnöki feladatok többségében. [2][3][6]

A prediktív szabályozások széles területet foglalnak magukba és új alkalmazások is kitaláltak teljesítményelektronikai alkalmazásokban is. A prediktív szabályozások legfontosabb tulajdonsága, ahogy a neve is utal rá, hogy a szabályozó az ismert rendszermodell alapján előre megjósolja a kimenet jövőbeli viselkedését, majd ezt az információt felhasználva igyekszik meghatározni a beavatkozó jel optimális mértékét a megadott kritériumok alapján. Ezutóbbi az esetek többségében egy minimalizálálandó költségfüggvényt jelent. A prediktív irányítások előnye, hogy ügyes tervezéssel egyszerű koncepciót hozhatunk létre, de a robusztusság növelése a feladatot már összetettebb problémává teheti. Modell alapú irányítás használatával elkerülhetjük a klasszikus lineáris szabályozások kaszkád struktúráját, a nemlineáritások könnyen beépíthetőek a modellbe, pedig elkerülhetjük a linearizálás miatt felmerült munkaponti függőségeket. További előnyt jelent, hogy korlátozások közvetlenül beépíthetők a költségfüggvénybe, valamint a tervezés többváltozós rendszerek (MIMO) esetén sem jelent nehézséget. Kiemelendő, hogy a szakasz matematikai modelljére továbbra is szükség van, annak komplexitása pedig erősen befolyásolja a predikció jóságát. A bonyolult leírás viszont nagyobb számítási teljesítményt követel meg. [2][3][13][14]

3.1 Elméleti háttér

A prediktív irányítások elmélete, valamint a rendszertechnika és optimalizációelmélet összekapcsolása egészen az 1960-as évekig nyúlik vissza, tényleges ipari alkalmazásokban azonban csak az 1970-es évektől kezdték el felhasználni. A korabeli számítógépek természetesen jóval kisebb számítási kapacitással rendelkeztek mai társaikhoz képest, így elsősorban vegyipari területen kísérleteztek vele, lassú, nagy időállandójú folyamatok esetében. A mikroprocesszorok fejlődésével viszont ezen területen végzett kutatások is erőteljes lendületet vettek. [2][3][13]

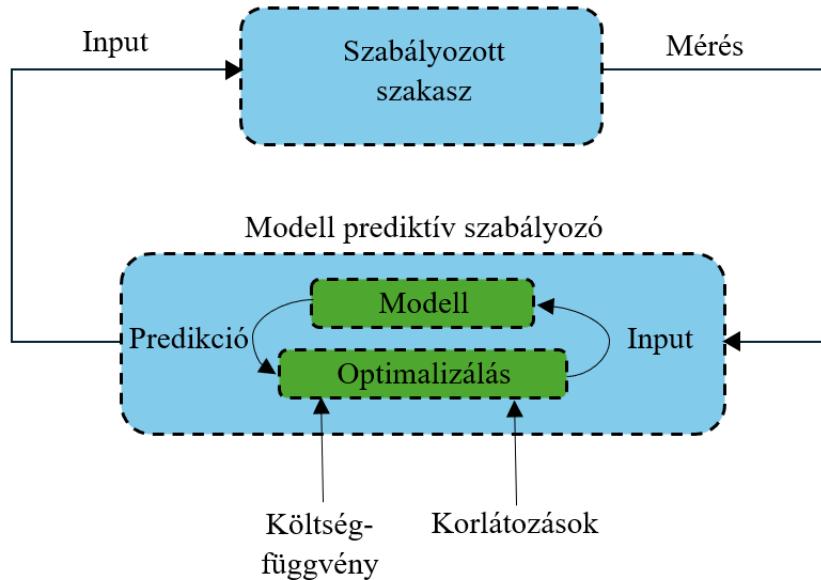
Az MPC célja, hogy a szabályozott jellemző a lehető leg pontosabban kövesse a referencia jelet a predikciós horizonton. Ehhez először definiáljuk a predikciós horizont fogalmát. Legyen a beavatkozó jel, a referenciajel és a szabályozott jellemző rendszerint $k+N_2$ -ig bezárólag adott, azaz értékeik számunkra ismertek. Vegyük fel ezeket a jeleket a 3.1. ábrán ismertetett módon az időtengely mentén. Célunk, hogy megtaláljuk a beavatkozó jel értékeit $k, k+N_1, \dots$ időlépésekre, amelyek mellett a probléma megoldása optimális lesz. Azaz keressük a már korábban definiált költségszámlálási minimumhelyét, amely garantálja, hogy a kimenő jel a legjobban követi a referencia jelet a horizont mentén. [2][3]



3.1. ábra Predikciós horizont az időtengely mentén [3]

Hosszabb predikciós horizont esetén a beavatkozó jelek sorozata kerül meghatározásra, azonban a szakaszra akkor is csak az optimális trajektória első eleme kerül kiadásra. Ez a trajektória minden mintavételi időláncban újra kiszámításra kerül. A módszer lényege pontosan ebben rejlik, kis időegységekre történő optimalizálás képes optimális szabályozott kimenetet adni a teljes időtartamra. A predikciós és optimalizációs

lépések kombinációja, ami az MPC-t más irányítási technikától megkülönbözteti. Ezt szemlélteti a 3.2. ábra is.



3.2. ábra MPC felépítése

A mérés során rögzített állapotváltozók alapján a rendszer matematikai modellje meghatározza a szabályozandó mennyiségek jövőbeli alakulását a predikciós horizonton. Az optimalizálás célja egy olyan vezérlőjel megtalálása, amely segítségével a predikció leginkább képes a referencia jel követésére. Ez a lépés a költségfüggvény minimalizálásával történik, miközben figyelembe kell venni a rendszerre vagy beavatkozó szervre vonatkozó korlátozásokat is. Ez a komponens írja le a lágy (*soft*) és kemény (*hard*) korlátokat, amelyeknek a rendszernek meg kell felelnie. Legtöbbször fizikai határokat jelent, de idetartoznak egyéb működési feltételek is. Végül a kapott optimális megoldás kerül a következő időlépésben a szakasz bemenetére.

A horizontnak azonban megfelelő hosszúságúnak kell lennie ahhoz, hogy a rendelkezőjel hatását megfelelően reprezentálja. A holtidő kompenzáció általában egy időlépéssel előretartó predikcióval valósítják meg ($N_I = 1$), majd ettől az időlépéstől számítható a szabályozási horizont. [2][3][6]

Egy tetszőleges rendszer állapotgyenletének általános alakja (3.1) és (3.2). egyenletek alapján írható fel.

$$x(k+1) = f(x(k), u(k)) \quad (3.1)$$

$$y(k) = h(x(k)) \quad (3.2)$$

Ahol $f(\cdot)$ és $h(\cdot)$ akár nemlineáris függvények is lehetnek, éppen ezért lehet ideális megoldás a prediktív irányítás, ahogy azt a fejezet elején is láttuk. Definiáljuk a minimalizálandó költségfüggvényt (3.3). egyenlet alapján.

$$J(N_1, N_2) = E \left\{ \sum_{k=N_1}^{N_2} [r(t+k) - y(t+k)]^2 \right\} \rightarrow \min \quad (3.3)$$

A cél a referencia jel és a kimeneti jel közötti négyzetes eltérés várható értékének minimalizálása a teljes predikciós horizonton. Egy jó költségfüggvény felírása kritikus pontja lehet a feladat megoldásának. A véges hosszúságú predikciós horizont extrém korlátozást jelent stabilitási szempontból. Ahhoz, hogy az aszimptotikus stabilitás továbbra is garantálható legyen, az MPC-hez időben monoton csökkenő függvényt kell választani. Az egyes tagok megválasztása feladatspecifikus, villamos hajtásokban leggyakrabban áram-, szögsebesség- vagy fluxusreferencia követését célozza, a motor típusától függően. [2][3]

3.2 Költségfüggvény

A költségfüggvény, vagy célfüggvény, a szabályozó teljesítményének egyik lehetséges mérőszáma, amely kvantitatív módon jellemzi, mennyire képes a kiválasztott optimalizálási algoritmus a referenciaérték követésére. Az optimalizálás során ennek minimalizálása a cél, amely azt jelenti, hogy a rendszer teljesítménye közelít az ideális viselkedéshez. Ezért az egyes tagok helyes megválasztása és a súlyok precíz meghatározása fontos a kívánt dinamikai viselkedés eléréséhez. Hiszen ez határozza meg, hogy az algoritmus mely szempontokat részesíti előnyben és milyen kompromisszumokra képes a referenciajel követése és a beavatkozó jel megválasztása között. [1][2][3]

A gyakorlatban a függvény megfelelően több, egymással kombinált komponensből tevődik össze, amelyek mindegyike a szabályozó teljesítményének egy-

egy aspektusát értékeli. Az egyes tagoknak a súlyozott összegét célszerű venni. Az úgynevezett hibaköltséget kifejező tag a legfontosabb. Ez adja meg a szabályozott jellemző és a referenciajel közötti eltérést. Általánosságban a két érték távolságának abszolútértékét jelenti, de implementáció, illetve későbbi matematikai megfontolásokból sokkal gyakrabban számítják a hiba négyzetét, emiatt pedig egy folytonosan differenciálható függvényt kapunk eredményül. [1][3]

Ezen a ponton ki kell emelni a modell prediktív irányítás egy újabb előnyét. Több állapotváltozó esetén egyszerűen beépíthetjük az egyes tagokat a költségfüggvénybe, itt viszont fontos figyelembe venni, hogy a különböző mennyiségeket szükség estén skálázzuk, különben ennek hiánya jelentős torzulást okozhat a működésben. [1][3]

Motorok vezérlésénél értelmezhető lépés a bemeneti változásokra vonatkozó tag bevezetése is. Ez lehetővé teszi a vezérlés kíméletesebb működését, mivel a hirtelen feszültséglépéseket súlyozottan bünteti. Így mérsékelhető a nyomaték ingadozása és a kapcsolóelemek terhelése is. [1][3]

3.3 MPC villamos hajtásokban

Ahogy az elméleti összefoglalóban is említettük az MPC alapú irányítás elsősorban ipari alkalmazásokban terjedt el, de a processzorok rohamos fejlődésével egyre nagyobb sikereket értek el teljesítményelektronika területén. Mára már az egyik legvonzóbb szabályozási stratégiává vált a motorhajtások területén. Ebben a fejezetben kifejezetten a hajtásokra vonatkozó MPC alkalmazásokat és azok jellemzőit tekintjük át.

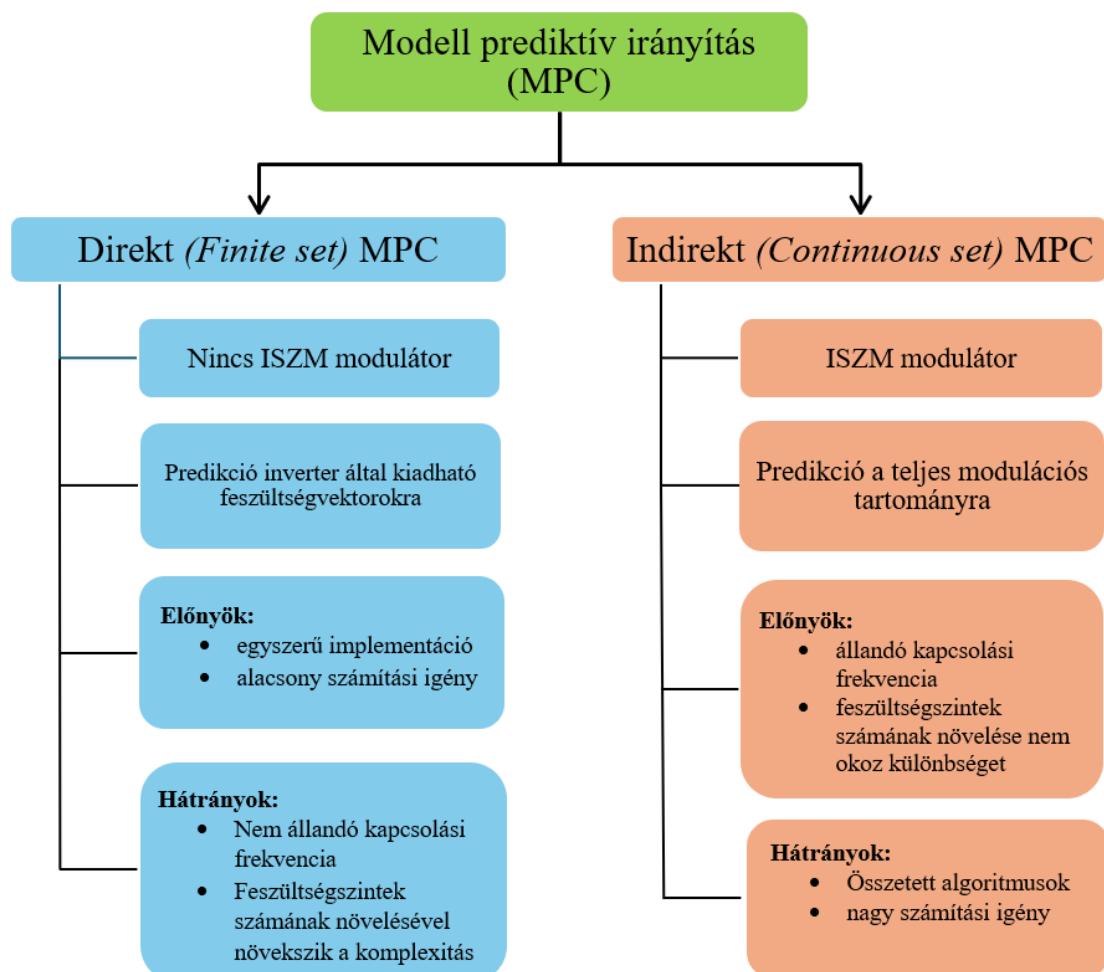
A dolgozatban használt villamos gépet egy kétszintű, háromfázisú inverter táplálja. Habár az ipari gyakorlatban elterjedtek lehetnek többszintű invertertopológiák is, ezek az irányítás alapvető működését nem befolyásolják, inkább spektrumot, illetve a teljesítményt javítják.

Az MPC stratégiák a fent említett vezérlési elemeket figyelembe véve két fő csoportba bonthatók: direkt (FCS-MPC) és indirekt (CCS-MPC) módszerekre. Ezek fontosabb jellemzőit, valamint előnyeit és hátrányait a 3.3. ábra tartalmazza. [13]

A direkt módszer esetén nincsen szükség modulátorra, mivel ez kizárolag az inverter által kiadható véges számú feszültségvektort használja. Az optimális feszültségvektor meghatározása után egy táblázatból könnyen kiolvashatóak az ahhoz tartozó kapcsolójelek. Ennél a módszernél egy egyszerűsített optimalizálás történik. A felállított predikciós modellbe sorban behelyettesítjük az egyes feszültségvektorokat, ezekhez meghatározzuk a költségfüggvény értékét az adott pontokban, majd a legkisebb értékű vektort kiválasztjuk. Előnye az egyszerű implementáció, valamint a lehetséges megoldások véges számából adódó alacsony számítási igény. Hátránya a változó kapcsolási frekvencia, illetve a komplexitás növekedésének lehetősége. Ezt könnyen beláthatjuk, ha a feszültségszintek számára gondolunk. Kétszintű inverter esetén ez hétféle lehetséges megoldást jelent (hiszen nullvektorból kettő van, elég csak egyszer kiszámítani), míg három szint esetén ez már huszonhét vektort jelent és ezek folyamatos újraszámítását. [13]

A folytonos tartományú vagy indirekt szabályozási módszer valós matematikai optimalizálási módszerek segítségével határozza meg a költségfüggvény minimumának helyét, ezáltal az optimális vezérlőjelet. A direkt MPC-vel szemben szükség van impulzusszélesség modulátorra, mivel a predikció a teljes modulációs tartományban történik. Ennek egyik legfontosabb előnye az állandó kapcsolási frekvencia, amellyel jól kontrollálható a kimeneti feszültség és áram harmonikus tartalma, valamint csökkenhető

az elektromágneses interferencia is. Fontos tulajdonsága, hogy az algoritmus komplexitása független az alkalmazott inverter feszültségszintjeinek számától. Míg az előbbi esetben exponenciálisan nőtt a számítási lépések száma, itt a folytonos keresési térben paraméterként megadható elemszámmal történik az optimalizálás, ezzel biztosítva egy determinisztikus számítási időt. A különböző matematikai optimalizálási módszerek általában bonyolult algoritmusok implementálását teszik szükségessé, sok esetben sztochasztikus elemek hozzáadásával. Direkt módszer alkalmazásánál a predikciós modellbe történő behelyetteséssel könnyen megtalálható volt az optimális megoldás, itt viszont az algoritmusok futtatása nagy számítási kapacitást igényel. [13]



3.3. ábra Villamos hajtásban alkalmazott modell prediktív irányítási módszerek

3.3.1 Predikciós modell felállítása

A dolgozat első fejezetében levezettük az állandómágneses szinkrongép elektromos dinamikáját leíró matematikai egyenleteket. A megismert koordinátatranszformációk segítségével a háromfázisú rendszer leírása jelentősen leegyszerűsödött, végül két differenciálegyenletet kaptunk, amelyek a predikció modell kiindulási egyenleteit jelentik.

$$u_d = Ri_d + L_d \frac{di_d}{dt} - \omega_e L_q i_q \quad (3.4)$$

$$u_q = Ri_q + L_q \frac{di_q}{dt} + \omega_e (\psi_m + L_d i_d) \quad (3.5)$$

Diszkretizáció

A diszkretizáció célja, hogy a folytonos idejű rendszereket olyan formába alakítsuk, amely számítógépes szimulációban vagy digitális vezérlésben kezelhető, azaz diszkrét idejű rendszerré váljanak. A valóságban a rendszerek viselkedését számítógépen vizsgáljuk, ahol a számításokat kizárolag diszkrét időlépésekben lehet elvégezni. Ez a korlát a mintavételi időből adódik, amely meghatározza a rendszer állapotainak és bemeneteinek értékeit az egyes időpontokban. Emellett a valós mérési jelek is diszkrét formában állnak rendelkezésre, mivel a szenzorok mintavételi frekvenciája szabja meg az adatpontok időbeli sűrűségét. Ezért a folytonos idejű differenciálegyenleteket először diszkretizálni kell, hogy a vezérlés implementációja pontos és stabil legyen. A diszkretizáció során alkalmazott módszerek nemcsak a számítások pontosságát, hanem a diszkrét rendszer stabilitását is meghatározzák. Emiatt a mérnöki gyakorlatban többféle diszkretizációs technikát alkalmaznak, attól függően, hogy a cél a számítási egyszerűség, a stabilitás.

Egy egyszerű és széles körben alkalmazott módszer az előrelépéses (Forward) Euler-féle diszkretizáció, amely a rendszer állapotváltozóinak következő időlépésbeli értékét a jelenlegi állapot és a rendszer dinamikai egyenletei alapján határozza meg. Tekintsünk egy általános elsőrendű differenciálegyenletet:

$$\dot{x}(t) = f(x(t), u(t)) \quad (3.6)$$

ahol $x(t)$ az állapotváltozó, $u(t)$ a bemeneti jel, és f a rendszer dinamikáját leíró függvény. A forward Euler-módszer diszkretizációs formája:

$$x[k+1] = x[k] + T_s \cdot f(x[k], u[k]) \quad (3.7)$$

$$\dot{x}(t=k) = f(x[k], u[k]) \approx \frac{x[k+1] - x[k]}{T_s} \quad (3.8)$$

, ahol T_s a mintavételi frekvencia reciproka.

Diszkrét predikciós modell

A motor vezérlésénél referenciaként a motor d és q irányú áramkomponenseit írjuk elő, ezért először rendezzük ennek megfelelően a (3.7) – (3.8). egyenleteket.

$$\frac{di_d}{dt} = \frac{1}{L_d} (u_d - Ri_d + \omega_e L_q i_q) \quad (3.9)$$

$$\frac{di_q}{dt} = \frac{1}{L_q} (u_q - Ri_q - \omega_e (\Psi_m + L_d i_d)) \quad (3.10)$$

Ezután alkalmazzuk a (3.8). kifejezést.

$$\frac{i_d[k+1] - i_d[k]}{T_s} = \frac{1}{L_d} (u_d[k] - Ri_d[k] + L_q \omega_e [k] i_q[k]) \quad (3.11)$$

$$\frac{i_q[k+1] - i_q[k]}{T_s} = \frac{1}{L_q} (u_q[k] - Ri_q[k] - \omega_e [k] (\Psi_m + L_d i_d[k])) \quad (3.12)$$

, ahol $u_d = u_d[k]$, $\omega_e = \omega_e[k]$, $L_d = L_q = L_s$, illetve $R = R_s$.

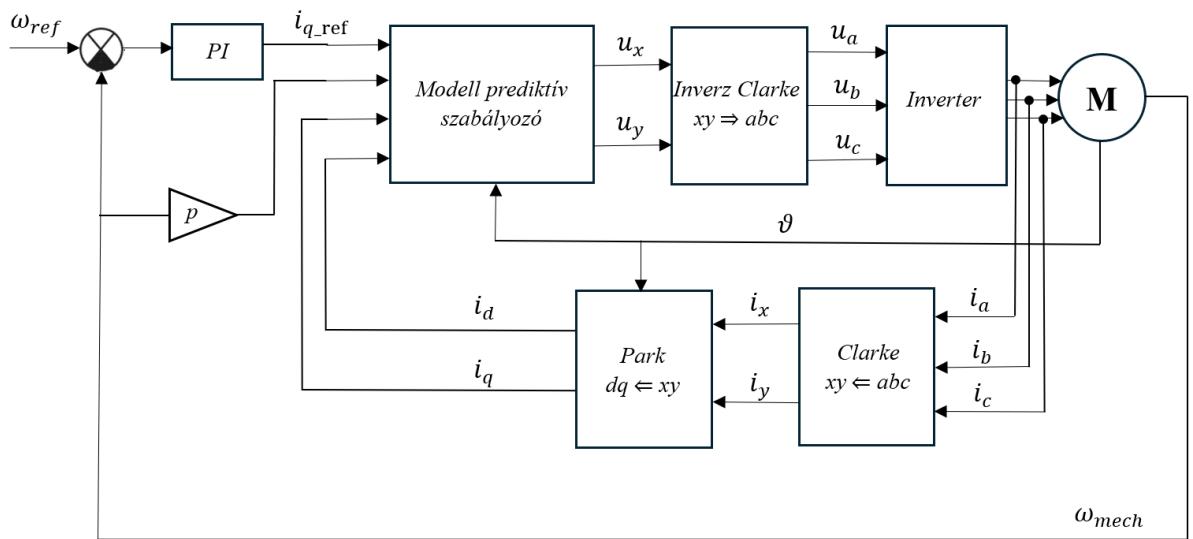
Szemléletesen rendezzük mátrixos alakba az összefüggéseket, ennek megléte az algoritmus implementálása miatt is hasznos.

$$\begin{bmatrix} i_d[k+1] \\ i_q[k+1] \end{bmatrix} = \begin{bmatrix} 1 - T_s \frac{R_s}{L_s} & \omega_e T_s \\ -\omega_e T_s & 1 - T_s \frac{R_s}{L_s} \end{bmatrix} \begin{bmatrix} i_d[k] \\ i_q[k] \end{bmatrix} + \begin{bmatrix} \frac{T_s}{L_s} & 0 \\ 0 & \frac{T_s}{L_s} \end{bmatrix} \begin{bmatrix} u_d[k] \\ u_q[k] \end{bmatrix} + \begin{bmatrix} 0 \\ -\omega_e T_s \frac{\psi_m}{L_s} \end{bmatrix} \quad (3.13)$$

$$\begin{aligned}
\mathbf{A} &= \begin{bmatrix} \left(1 - T_s \frac{R_s}{L_s}\right) & \omega_e T_s \\ -\omega_e T_s & \left(1 - T_s \frac{R_s}{L_s}\right) \end{bmatrix} \\
\mathbf{B} &= \begin{bmatrix} \frac{T_s}{L_s} & 0 \\ 0 & \frac{T_s}{L_s} \end{bmatrix} \\
\mathbf{d} &= \begin{bmatrix} 0 \\ -\omega_e T_s \frac{\psi_m}{L_s} \end{bmatrix}
\end{aligned} \tag{3.14}$$

3.3.2 MPC szabályozás hatásvázlata

A felállított predikciós modell és a korábbi fejezetekben leírtak ismeretében felállítható a PMSM motor modell prediktív irányítását megvalósító szabályozási kör vázlata. Ezt a sematikus rajzot mutatja a 3.4. ábra.



3.4. ábra PMSM motor MPC alapú szabályozása [20]

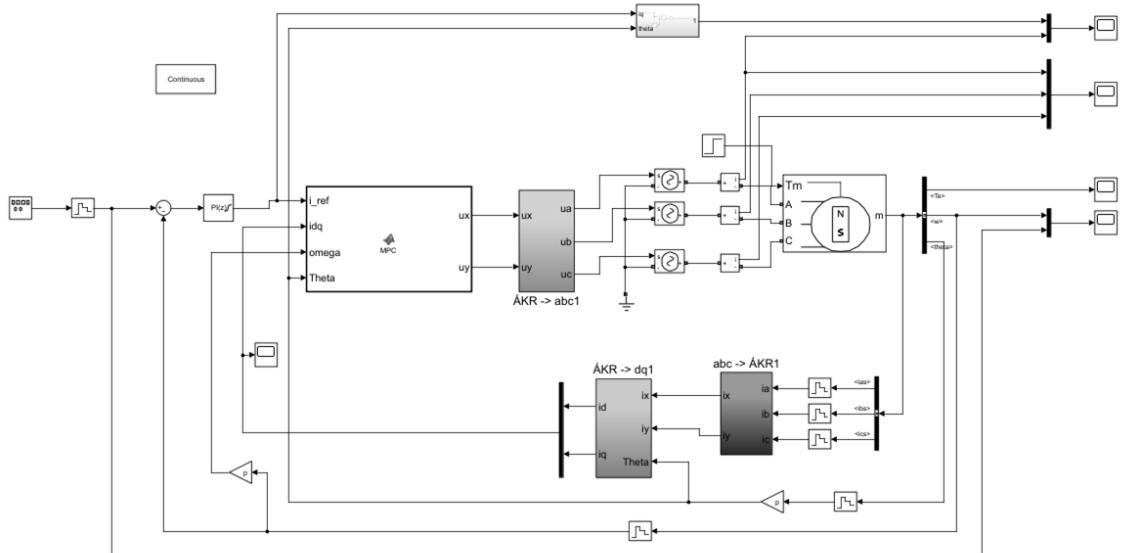
A rendszer bemenetén a gyakorlathoz hasonlóan a motor szögsebeségét írjuk elő referenciajelként. A referencia és a tényleges jel közötti különbségből egy PI szabályozó írja elő a nyomatékképző áramkomponenst az MPC számára. Ez a q irányú áramot jelöli. A d irányú áram szabályozásával lehetőség van az állandómágnes által létrehozott mágneses fluxus gyengítésére, normál üzemben általában erre nincsen szükség, ezért ezt

konstans zéró értékre szabályozzuk. A prediktív szabályozó bemenetén az áramreferenciaértékek és a tényleges, mért áramok jelennek meg, amelyek alapján az optimalizálási algoritmus meghatározza a következő mintavételi időre vonatkozó vezérlési jelet. Ez a folyamat magába foglalja az áramkövetési hibát és a feszültségváltozások mértékét is. Az elektromos szögre az esetleges koordinátatranszformációk miatt van szükség. Az optimalizálási algoritmus keresési terét a d és q tengelyek által kifeszített sík jelenti, tehát a megtalált legjobb megoldás is ebben a koordinátarendszerben van értelmezve. Ahogy az ábra is mutatja, a következő blokk előtt ezt a jelet még vissza kell transzformálni x-y koordinátarendszerbe. Az inverz Clarke transzformációval visszakapjuk a háromfázisú feszültségkomponenseket, melyeket aztán az inverter alakít át PWM jelekké. A motor egyes tekercseiben folyó áramokat a jól megválasztott mintavételi frekvenciával mintavételezzük, majd a szinuszosan váltakozó jeleket az első fejezetben ismertetett módon két, konstans jellé alakítjuk. A belső hurok a visszacsatolt áramjelekkel jön létre. A rotorpozíció meghatározása szintén szükséges, ez történhet közvetlenül szenzorok segítségével, de elterjedtek szenzor nélküli megoldások, úgynévezett állapotbecslők alkalmazása. A felépített rendszer lehetővé teszi a gyors áram-, és nyomatékszabályozást, továbbá a hagyományos lineáris szabályozókkal szemben képes közvetlenül figyelembe venni a rendszer fizikai korlátait.

3.3.3 MPC szimulációja MATLAB környezetben

A korábbi fejezetekben bemutatásra kerültek a modell prediktív irányítás elméleti háttere, matematikai leírása, valamint a megvalósítható szabályozás sematikus vázlata. Jelen alfejezet célja, hogy az elméleti ismereteket szimulációs környezetben is implementáljuk és megvizsgáljuk annak működését.

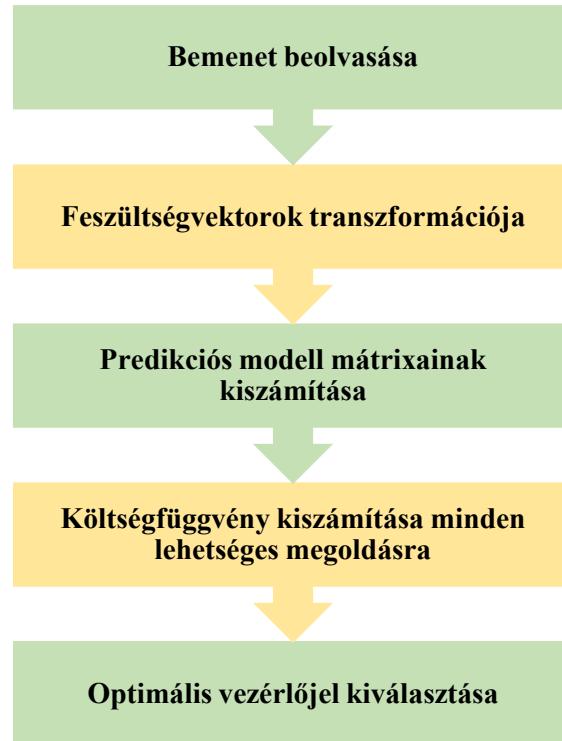
A szimuláció során a MATLAB által rendelkezésre bocsátott PMSM motormodellt használtam, a szabályozási algoritmus alapját pedig az előző alfejezetben felállított matematikai modell jelenti. A 3.5. ábra a Simulink környezetben megépített szabályozási kört mutatja be.



3.5. ábra Szabályozás Simulink környezetben megvalósított szimulációja

A modell főbb komponensei közé tartozik az MPC blokk, amely a rendelkezésre álló paraméterek és matematikai modell alapján meghatározza az optimális vezérlőjelet az inverter számára, az álló és forgó koordinátarendszerek közötti transzformációkat megvalósító blokkok, az inverter és motor modellek. Ezentúl a mérési pontok és szkópok lehetővé teszik a motor kimeneti jeleinek és a vezérlő bemeneti jeleinek folyamatos rögzítését és vizualizációját. A szabályozó algoritmus MATLAB Function blokkban került megvalósításra. Most ezt vizsgáljuk meg kicsit részletesebben. A függvény a két kimeneti feszültségkomponenst határozza meg az álló koordinátarendszerben, amely aztán az inverteren keresztül a motorra kerül. A függvény bemenetei a sebességszabályzó által meghatározott q irányú áramkomponens (hiszen a d irányú áramot automatikusan nullára szabályozzuk), az elektromos szög a szükséges transzformációhoz, az elektromos szögsebesség, valamint a motor tekercseinek árama a megfelelő rendszerbe transzformálva. A függvényben mentésre kerülnek az előző időlépéshoz tartozó állapotváltozók, valamint a különböző konstanst motor és rendszerparaméterek. A vezérlés a korábban ismertetett direkt módszert követi. A lehetséges megoldások a kétszintű inverter által kiadható hét feszültségvektor. Az elektromos szögre azért van szükség, mert ezek a vektorok az x-y álló koordinátarendszerben vannak értelmezve, de a modellünk forgó referenciarendszerben van értelmezve. Az optimális feszültségjel meghatározása nem igényel bonyolult algoritmust, a paraméterek behelyettesítésre kerülnek a predikciós modellbe, majd minden vektorhoz meghatározzuk a

költségfüggvény értékét. A legalacsonyabb értékkel rendelkező vezérlőjel kerül kiválasztásra. A 3.6. folyamatábra az imént leírt működést foglalja össze.



3.6. ábra MPC folyamatábra

3.3.4 Szimulációs eredmények

A továbbiakban megvizsgáljuk a PMSM motor modell prediktív irányításon (MPC) alapuló sebességszabályozásának szimulációs eredményeit. A cél, hogy bemutassuk a motor nyomatékának és sebességének alakulását, valamint a háromfázisú áramok viselkedését. A megvalósítás során direkt (*finite set*) MPC szabályozás biztosította a rendszer dinamikáját, ahol a referenciajel követése a (3.15). egyenletben megadott költségfüggvénnyel történt.

$$g = \left(i_{\text{ref}} - i_{k+2,q}(i) \right)^2 + \left(i_{k+2,d}(i) \right)^2 + f \left(i_{k+2,d}(i), i_{k+2,q}(i) \right). \quad (3.15)$$

AZ $f(\cdot)$ büntetőtag a megengedett maximális áramértékek túllépésenek elkerülésére szolgál, i pedig a megadott, diszkrét feszültségvektorok indexeit jelöli. Az alkalmazott büntetőfüggvény általános alakját a (3.16) egyenlet adja meg.

$$f(x, y) = 10^3(\text{abs}(x) > i_{d,max} \text{ || } \text{abs}(y) > i_{q,max}) \quad (3.16)$$

A négyzetes költségfüggvény mellé egy harmadik, nemlineáris tagot is használtunk, amely szükségessé teszi olyan minimalizáló eljárások keresését, ahol a nemlinearitás az optimalizálást nem befolyásolja.

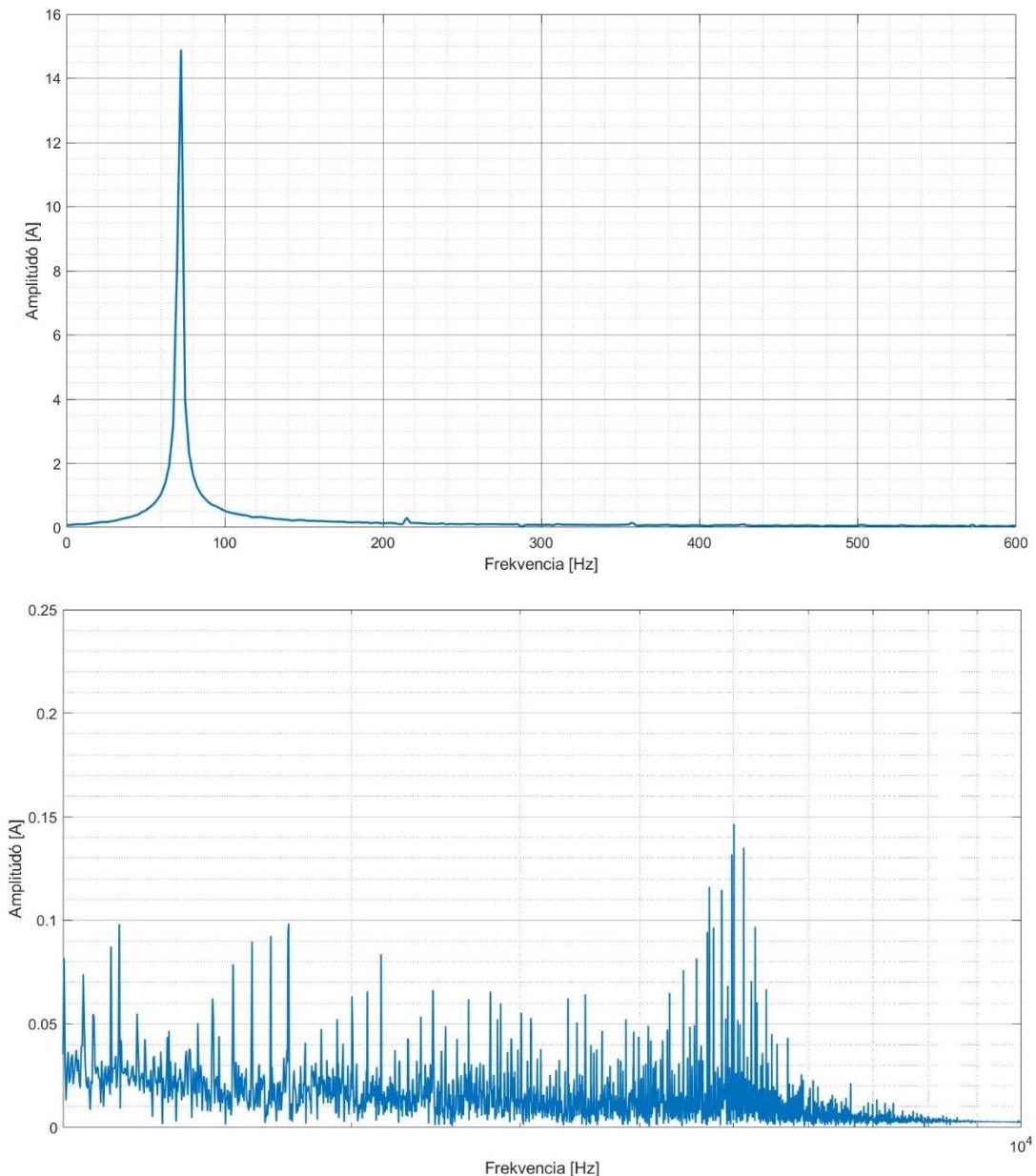
A szimuláció során impulzus referenciajelet alkalmaztam, illetve ugrásszerű terhelőnyomatékot, amellyel lekövethetjük a motor dinamikus válaszát. A szimulációt a következő gépparaméterek mellett végeztem: az állórész induktivitása 82 mH, ellenállása 0.3 ohm, a rotorfluxus 0.125 Vs, a póluspárok száma 3, a tehetetlenségi nyomaték 0.008 kg·m², a viszkózus csillapítást feltételezve a hozzáartozó érték 0.001 Nms, valamint a mintavételi idő 100 µs volt.

A 3.8a. ábra a motor áramait szemlélteti. A fenti ábrán a fázisáram, míg az alsón a d és q irányú áramok szerepelnek. A d irányúáramot nullára szabályozzuk, a q irányú áram pedig a lineáris szabályozó által kiadott referenciajelet követi a megfelelő nyomaték előállításához.

A 3.8b. ábráról a mechanikai jellemzőket olvashatjuk le. A fenti görbék a motor nyomatékát és a ráadott terhelést mutatja. Alul a szögsebességreferenciajel követésének karakterisztikáját figyelhetjük meg, mind állandósult állapotban mind pedig gyorsítási fázisban.

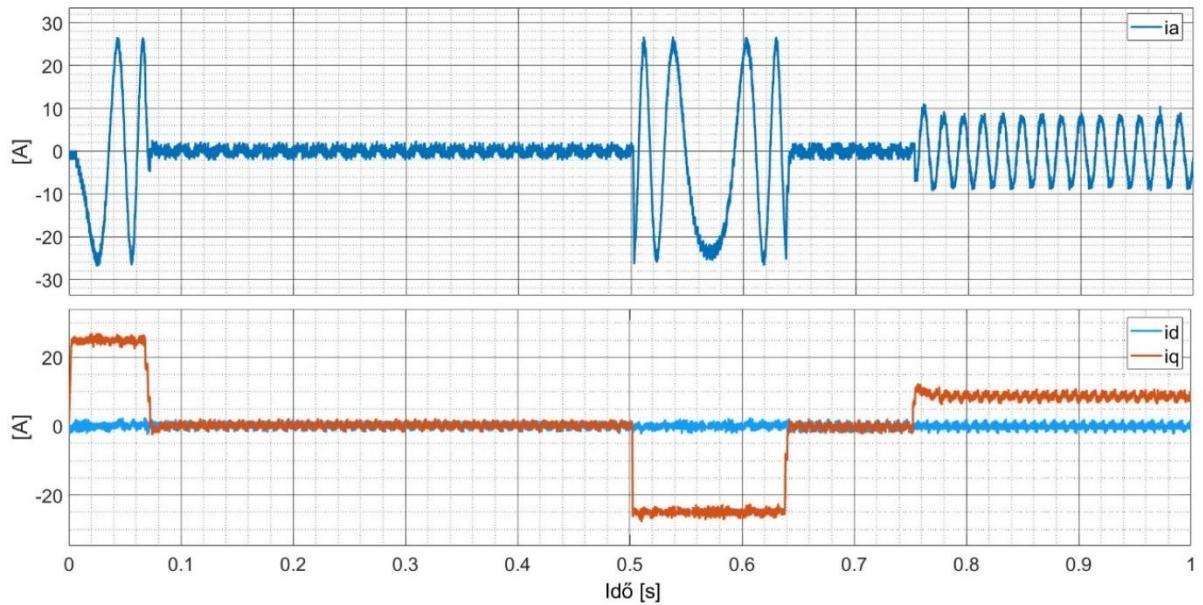
A szimuláció során a motor áramjeleinek spektrumát vizsgáltam moduláció alkalmazása nélkül. A későbbi mérések során vivőfrekvenciás mérést alkalmaztam, így láthatjuk mi a jelentősége a moudlációnak a spektrum szempontjából.

Az FFT analízis lehetővé teszik ezek vizsgálatát. A 3.7. ábra eredményei azt mutatják, hogy az alacsony frekvenciatartományban (2000 Hz alatt) a spektrum megegyezik a elvártakkal. A különbség a felharmonikusok estén adódik. Moduláció alkalmazása nélkül a felharmonikusok nem különülnek el jól egymástól, a harmonikusok elmosódottabbak, kevésbé definiáltak.

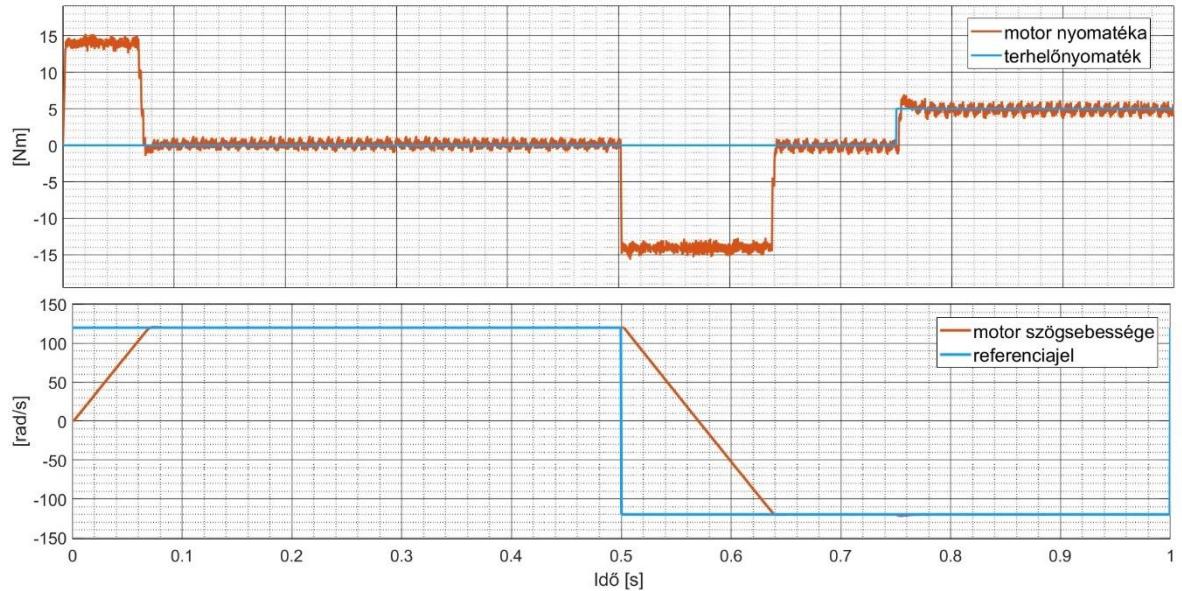


3.7. ábra Fázisáram spektruma moduláció nélküli esetben

Összességében a szimulációs eredmények alátámasztják, hogy az MPC alapú szabályozás jó dinamikát és pontosságot biztosít a rendszer számára, különösen a korlátozások kezelésének területén. A szabályozó képes hatékonyan kezelni a rendszer fizikai korlátait. Emellett az MPC előnyei különösen a tranziszorok során jelentkeznek, ahol gyors reakció szükséges. Ezek az eredmények megerősítik, hogy az MPC szabályozás alkalmat a rendszer megbízható és pontos működtetésére a gyakorlatban is.



3.8a. ábra MPC szimulációja: Fázisáram és d és q irányú áramok



3.8b. ábra MPC szimulációja: motor nyomatékgörbái és szögsebességkarakterisztikája

4 Korszerű rajintelligencia módszerek

4.1 Áttekintés

A modern mérnöki alkalmazások során egyre gyakrabban találkozhatunk optimalizálási feladattal. Így van ez a teljesítményelektronikai rendszerek és hajtások esetén is. A korszerű hajtásrendszerekkel szemben támasztott követelmények folyamatosan nőnek, elsősorban a nagyobb hatékonysági és gazdaságossági szempontok miatt. Gondolhatunk az energiafogyasztás minimalizálására, vagy a rendszer dinamikájának növelésére. Mindez egy összetett matematikai optimalizálási probléma megoldását jelenti. Az utóbbi néhány évtizedben számos megoldási módszert megismertünk, napjainkban azonban fokozódó érdeklődés mutatkozik a rajintelligencia, vagy ehhez hasonló, más módszerek irányában. [4][5]

A metaheurisztikus algoritmusok népszerűsége számottevő ezen alkalmazásokban. Közös jellemzőjük, hogy nem igénylik a gradiensek kiszámítását, amely képes a számítás bonyolultságát jelentősen növelni. A véletlenszerűen történő keresés lehetővé teszi a lokális optimumban történő megakadás elkerülését. A szakirodalomban különféle szempontok szerint kategorizálják az algoritmusokat. A legtöbb publikációban alapvetően három fő csoportot különböztethetünk meg. Az evolúcióalapú, fizikaalapú és rajalapú módszerek. [4] [5]

A fizikaalapú módszereket a környezetben megfigyelhető fizikai folyamatok inspirálták, bizonyos fizikai törvényszerűségek utánzásával hozták létre. A modellek a fizika több területét is felölelik, például klasszikus mechanika, optika és termodinamika. Néhány algoritmust említve ide sorolhatóak a központi erőtér vagy gravitációs optimalizálás. [4] [5][10][16]

Az evolúcióalapú módszerek igen elterjedtek más, mesterséges intelligenciát alkalmazó területen. A biológiai evolúción és természetes kiválasztódáson alapuló technikák hatékonynak bizonyultak optimalizálási feladatokban. Egyik legismertebb stratégia a genetikus algoritmus, amely a genetikai öröklődés folyamatát imitálja, keresztezési és mutációs lépések alkalmazásával. [4] [5] [10][16]

Dolgozatomban a harmadik kategóriába tartozó rajalapú módszerek hatékonyságát vizsgáltam, azok fejlődését és módszertanát bemutatva. Továbbá négy

különböző algoritmus ismertetésével bemutatom ezek hatékonyságát és felhasználhatóságát.

A rajintelligencia módszerek (*Swarm Optimization Algorithms*) a természetben előforduló biológiai rajok (pl. hangyák, méhek, részecskék, halak) kollektív feltérképezési eszközeit utánozzák optimalizálási problémák megoldására. Az optimum megtalálása egy önszerveződési folyamat eredménye. Az egyedek közötti közvetlen, illetve közvetett lokális interakciók együtt segítik a kolónia globális szerveződését. A globális, illetve lokális keresés ötvözésével, valamint sztochasztikus elemek használatával képes nagyméretű, neplineáris problémák megoldására. A folyamat során decentralizált struktúra jön létre, nincsen központi irányítás, az egyedeket ösztönök vezérlők. Ezzel egy sokkal dinamikusabb és hatékonyabb keresési módszert implementálva. James Kennedy, a PSO egyik kitalálója úgy fogalmazott, hogy „*a raj fogalma sokféleséget, sztochasztikusságot, véletlenszerűséget és rendezetlenséget sugall, míg az intelligencia fogalma arra utal, hogy a problémamegoldó módszer valamilyen módon sikeres*”. Azonban fontos megjegyezni, hogy mindezek mellett sem garantálható a globális optimum megtalálása. Mindazonáltal ezen algoritmusoknak általánosságban vannak hátrányaik is. A problémától függően megnőhet a paraméterek száma, amely jelentősen megemeli a szükséges számítási időt, mivel minden egyes egyedet ki kell értékelni a továbblépéshez. Ez pedig akár megakadályozhatja a valós időben történő felhasználását. Ezek mellett a lokális szélsőérték helyeken ragadhat, amiből nem minden esetben képes kiszabadulni. [4] [5] [10][16]

4.2 Rajintelligencia alapú optimalizálási algoritmusok

4.2.1 Részecske-rajintelligencia optimalizálás (PSO)

A PSO-ról (*Particle Swarm Optimization*), - másnéven részecske-rajintelligencia - először James Kennedy, szociálpszichológus és Russel Eberhart, amerikai villamosmérnök publikált 1995-ben. Az ötletet a madárrajok élelemkeresés során megfigyelt mozgása adta, melyet aztán algoritmusként implementáltak. minden részecske a keresési tér egy pontját, egy megoldást jelöl. A PSO két legfontosabb fázisa a keresési tér minél alaposabb felfedezése, majd a várhatóan legjobb eredményt adó területre történő összpontosítás. Ezutóbbi a részecskék egyidejű kollektív mozgását

jelenti az optimális pozíció irányába. Tegyük fel, hogy van egy D dimenziós keresési terünk, ebben az esetben minden részecskehez három darab D dimenziós vektort rendelünk. Ezeket rendre x_i , $pbest_i$, és $gbest$ nevekkel szokás jelölni. Az első az i -dik részecske aktuális pozíóját, a második az adott részecske által korábban megtalált legjobb megoldást jelenti. A $gbest$ a globális legjobb állapotot jelöli. Szomszédossági topológiától függően ezt kiválaszthatjuk az összes részecske közül, vagy a részecskék előre definiált szomszédjai közül kerül ki. [4] [5][7][8][9][10]

Első lépésként véletlenszerűen inicializáljuk a meghatározott darabszámú részecskét, illetve a kezdeti sebességvektorokat, melyek aztán minden iterációban újra kiszámításra kerülnek a 3.1. és 3.2. egyenletek szerint:

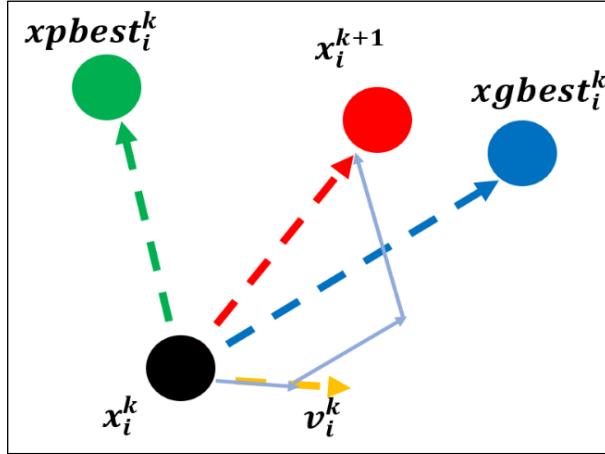
$$\mathbf{v}_i(t+1) \leftarrow w_i \mathbf{v}_i(t) + c_1 r_1 (\mathbf{pbest}_i(t) - \mathbf{x}_i(t)) + c_2 r_2 (\mathbf{gbest}_i(t) - \mathbf{x}_i(t)) \quad (4.1)$$

$$\mathbf{x}_i(t+1) \leftarrow \mathbf{x}_i(t) + \mathbf{v}_i(t+1) \quad (4.2)$$

Ahol r_1 és r_2 kettő, egyenletes eloszlást követő véletlenszám a $[0, 1]$ intervallumból, w_i az ún. „*inerciális*” vagy tehetetlenségi súly, c_1 és c_2 pedig a „*kognitív*” és „*szociális*” együtthatók. Ezen vektorok részecskére ható vonzóerőként (hasonlóan a rugó viselkedéséhez) is felfoghatók, amelyek eredője határozza meg a részecske mozgását.

Más optimalizálási megoldókhöz hasonlóan általában kétféle leállási feltételt határozhatunk meg. A maximális iterációs szám megadásával, illetve a globálisan legjobb értéket adó megoldás stagnálásának vizsgálatával korlátozhatjuk a ciklus hosszát. Valós idejű rendszerekben mindenkorban az előbbi, iterációs számra vonatkozó feltételt adjuk meg, hiszen csak így garantálható a determinisztikus viselkedés, amely a számítási idő ismeretét feltételezi. [4] [5][9][10]

Ahogy korábban említésre került, a PSO algoritmusok metaheurisztikus eljárások, ezért az eljárás sikereségét a kezdeti inicializálás, a részecske-, és iterációs szám, a paraméterek megválasztása erősen befolyásolja.



4.1 . ábra Részecske-re ható „erők” vektorábrája

Alapvetően kijelenthető, hogy a kezdeti részecske-szám növelésével nagyobb eséllyel találja meg az optimumot, de ez sok esetben feleslegesen növeli meg a számítási igényt. Költségfüggvénytől függően, de kevesebb egyeddel is képes jó megoldást adni. Az inerciális súly az algoritmus kezdeti fázisában segíthet a tér felfedezésében, illetve a lokális szélsőértékekkel is képes kimozdítani az elemet, azonban a későbbi szakaszban a nagy lépésköz megnehezítheti a feltételezhető optimum körül, szűkebb téren való mozgását. Ezek mellett fontos a helyes szomszédossági topológia megválasztása is. Ugyanis ez határozza meg, hogy mely részecskék vannak hatással egymás mozgására. A legegyszerűbb a *csillag* topológia, amely minden részecskét szomszédosnak tekint. Ekkor a *gbest* a globális legjobb eredményt jelenti, minden elemnél azonos. A *gyűrű* topológia minden egyedhez a tőle balra és jobbra lévő szomszédot rendeli. Ebben az esetben inkább *lbest*, azaz lokálisan legjobb megoldásról beszélhetünk. A *von Neumann* elv mátrixba rendezzi az elemeket, és a felette, alatta, illetve a tőle balra és jobbra lévőket tekinti szomszédnak. Az eddig felsorolt három esetben a szomszédossági viszonyok változatlanok maradtak az iteráció folyamán. Van lehetőség az úgynevezett *dinamikus topológia* választására, amely kezdetben kevés szomszéddal indul, majd folyamatosan növeli a számát, végül pedig csillagtopológiát eredményez, amely a részecskék együttes haladását eredményezi. Ez szintén kedvező hatással lehet az optimum megtalálására. [4] [5] [7][8]

A részecske raj intelligencia algoritmusának pszeudódija a 4.2. ábrán látható, amely részletesen bemutatja az optimalizáló eljárás működését – a kezdeti populáció létrehozásától a részecskék sebességének és pozíciójának frissítésén át egészen a globális megoldás megtalálásáig.

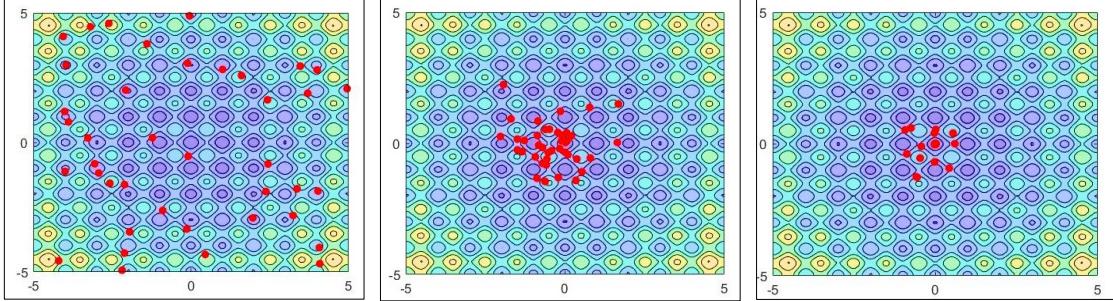
```

1: Paraméterek inicializálása
2:       $N, w, c1, c2, MaxIter, g$ 
3: for  $i=1$  to  $N$ 
4:      Részecske pozíciójának inicializálása egyenletes eloszlású véletlenvektorral  $\mathbf{x}_i$ 
5:      Részecske legjobb pozíciójának inicializálása a kezdeti pozícióval  $\mathbf{p}_i \leftarrow \mathbf{x}_i$ 
6:      Költségfüggvény kiszámítása  $f(\mathbf{x}_i)$ 
7:      if  $f(\mathbf{x}_i) < f(\mathbf{g})$  then
8:          Legjobb megoldás frissítése  $\mathbf{g} \leftarrow \mathbf{x}_i$ 
9:      end if
10:     Kezdeti sebesség inicializálása véletlenszerű vektorral  $\mathbf{v}_i$ 
11: end for
12: while  $iter < MaxIter$  do
13:     for  $i=1$  to  $N$ 
14:          $r1, r2$  – véletlenszerű számok  $\sim U(0,1)$ 
15:         Részecske sebességének frissítése az ismert összefüggés alapján
16:          $\mathbf{v}_i \leftarrow w_i \mathbf{v}_i(t) + c_1 r_1 (\mathbf{p}_i - \mathbf{x}_i) + c_2 r_2 (\mathbf{g} - \mathbf{x}_i)$ 
17:         Pozíció frissítése:  $\mathbf{x}_i \leftarrow \mathbf{x}_i + \mathbf{v}_i$ 
18:         Költségfüggvény kiszámítása  $f(\mathbf{x}_i)$ 
19:         if  $f(\mathbf{x}_i) < f(\mathbf{p}_i)$  then
20:             Részecske lgjobb pozíciójának frissítése  $\mathbf{p}_i \leftarrow \mathbf{x}_i$ 
21:         end if
22:         if  $f(\mathbf{x}_i) < f(\mathbf{g})$  then
23:             Legjobb megoldás frissítése  $\mathbf{g} \leftarrow \mathbf{x}_i$ 
24:         end if
25:     end for
26: end while

```

4.2 . ábra PSO algoritmus pszeudókódja

A 4.3. ábrán a részecskék mozgásának alakulását követhetjük figyelemmel egy minimumkeresési eljárás folyamán. A keresési algoritmust harminc elem csillag topológiába való kapcsolásával teszteltem. A minimalizálandó függvény a számtalan lokális minimumhellyel rendelkező Rastrigin-függvény volt, melynek globális minimuma az origó. Ahogy az ábra is szemlélteti, a kezdeti inicializálás után a részecskék rendre a globális legjobb értéket adó egyed felé haladnak.



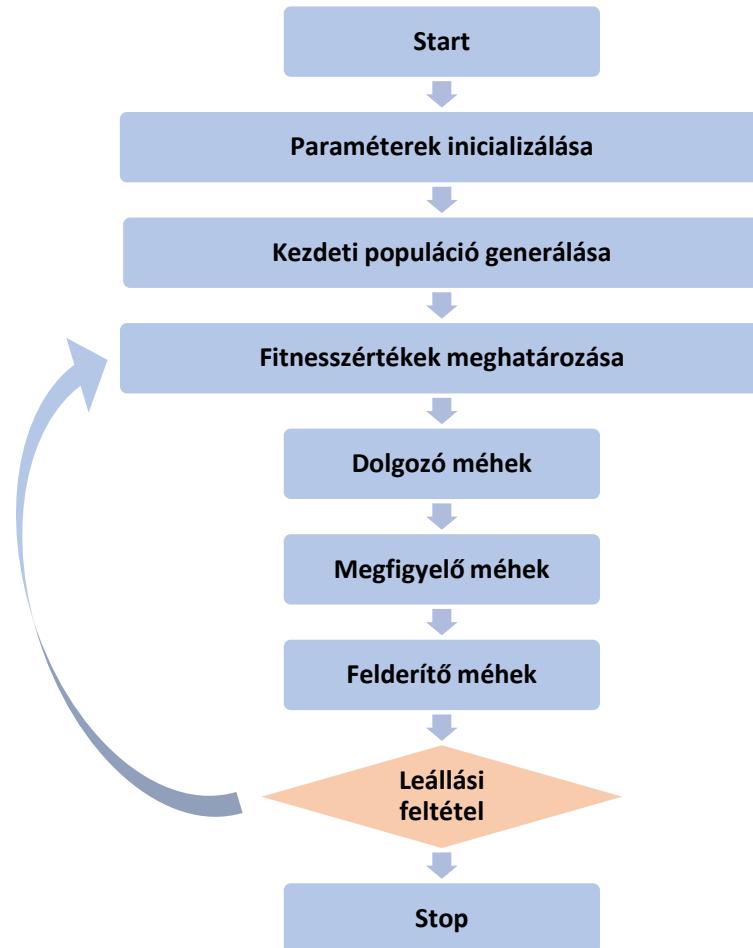
4.3. ábra Részecskék mozgásának folyamata
inicializálás (*bal*), 8. iteráció (*középen*), leállás után (*jobbra*)

4.2.2 Mesterséges méhraj optimalizálás (ABC)

A mesterséges méhraj optimalizálási módszert (*Artificial Bee Colony*) a méhcsaládok élelemszerzési tevékenysége ihlette, melyet David Karaboga mutatott be 2005-ben. Az algoritmus kidolgozásában jelentős szerepet játszott Tereshko és Loengarov méhek kollektív viselkedését leíró matematikai modellje. Az elképzelés lényege, hogy a kolónia egyes egyedei, az úgynevezett dolgozó méhek folyamatosan térképezik fel a területet élelem után, majd a fellelt forrásokról információt cserélnektársaikkal. A decentralizált keresés és információmegosztás lehetővé teszi, hogy a méhcsalád a legkedvezőbb forrást azonosítsa. A folyamat során a keresési tér minden pontja egy lehetséges élelemforrást vagy megoldást jelöl, melynek minőségét a költségfüggvény értéke határozza meg. Az algoritmus szintén követi az elvet, amely a raj intelligencia módszereket jellemzi: egyensúlyt kíván teremteni a feltárás és a kiaknázási lépések között. Tehát egyszerre igyekszik a keresési tér minél nagyobb részét lefedni, de amennyiben egy optimum pontot talál, úgy annak környezetét részletesen megvizsgálja. Ebben a fejezetben részletesen megismerhetjük az élelemkeresés folyamatát, valamint az algoritmikus leírását. A leírás során először a természeti folyamatot vizsgáljuk meg, majd ezután annak algoritmusban betöltött szerepét. [10][18][19]

Az élelemszerzési folyamat egy komplex, több lépésből álló folyamat. A természetben először a felderítő méhek (*scout bee*) indulnak útnak véletlenszerűen kutatva források után. A gyakorlatban ez a kezdeti egyedek sztochasztikus inicializálását jelenti. A keresési téren véletlenszerűen felveszünk megadott számú pontot. Ezután a méh visszatér a kaptárba és úgynevezett méhtánc segítségével kommunikálja társai felé

az általa fellelt élelemforrás minőségét, irányát és a kaptártól való távolságát. Az algoritmusban minden felvett pontban kiszámítjuk a költségfüggvény értékét. Ezután a méh néhány társával, másnéven toborzott méhekkel visszatér erre a helyre, hogy ennek környezetében kutatva jobb nektárforrást találjanak. A toborzott méhek közül is visszatérnek a kaptárba és a toborzási és felfedezési mechanizmus kezdődik előlről. [10][18][19]



4.4. ábra ABC algoritmus folyamatábrája

Az algoritmus pontos folyamatát a 4.4. ábra mutatja. Az optimalizálás során három féle „méh” viselkedését modellezük. A dolgozó méhek (*employed bees*), a figyelő méhek (*onlooker bees*) és felderítő méhek (*scout bees*). A kezdeti inicializálás során rögzítjük az algoritmus során használt paramétereket, mint a populációs szám, megengedhető maximális kísérletek száma, iterációs szám. Ezután véletlenszerűen

felvesszük a kezdeti populációt, majd elkezdődik az optimalizációs ciklus. Először a dolgozó méheket hozzárendeljük az egyes egyedekhez, és minden ponthoz véletlenszerűen keres, illetve felvesz egy szomszédot annak környezetében. [10][18][19]

Amennyiben a szomszéd fitnesszértéke jobb, mint az adott ponté, úgy az egyedet kicseréljük a szomszédjára. Amennyiben nem találtunk jobb megoldást a közelében, az adott ponthoz tartozó kísérletek számát megnöveljük eggyel. A maximális kísérletek száma azt jelenti, hogy ha adott ciklusszámig nem találunk jobb szomszédot a környezetében, azaz az optimalizálás során nem találunk jobb megoldást, elvetjük és helyette egy új pontot veszünk fel. A dolgozó méhek után a megfigyelő méhek következnek. minden egyedhez hozzárendelhetünk egy kiválasztási valószínűséget a (4.3) - (4.5). egyenletek szerint. [10][18][19]

Fitneszértékek vektora:

$$f = [f_1, f_2, \dots, f_n] \quad (4.3)$$

Minden elemhez kiszámítunk egy inverz értéket, majd egy kiválasztási valószínűséget.

$$f_{\text{inv},i} = \begin{cases} \frac{1}{1+f_i}, & f_i \geq 0 \\ 1 + |f_i|, & f_i < 0 \end{cases} \quad (4.4)$$

$$p_i = \frac{f_{\text{inv},i}}{\sum_{j=1}^n f_{\text{inv},j}} \quad (4.5)$$

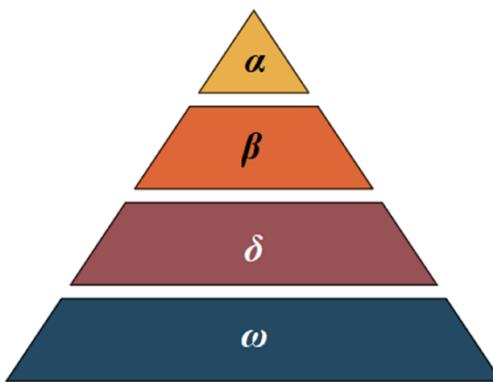
Ez egy nulla és egy közötti érték. Minél jobb egy megoldás annál nagyobb lesz ez az érték. Ezután generálunk egy véletlen számot. Ha ez a véletlen szám kisebb, mint a ponthoz rendelt érték, akkor annak a pontnak környezetében keresünk egy szomszédot, és az előző lépéshoz hasonlóan, ha költsége kisebb, akkor kicseréljük. A kísérletek száma azonban itt is módosul, amennyiben sikertelen volt a keresés. Ez a lépés jól mutatja, hogy nem minden pont kerül kiválasztásra, hanem legnagyobb valószínűsséggel a legjobb megoldások környezetét térképezzük fel részletesebben. A felderítő méhek feladata az utolsó lépés, figyelik, hogy melyik forrás az, amelyik elérte a maximálisan megengedett ciklusszámot, anélkül, hogy javított volna az értékén és ezek helyére új egyedeket keres és vesz fel. Az algoritmus pszeudókódját a 4.5 ábra szemlélteti. [10][18][19]

1: Paraméterek inicializálása
 2: $N, MaxIter, limit$
 3: **for** $i=1$ **to** N
 4: Egyedek pozíciójának inicializálása egyenletes eloszlású véletlenvektorral \mathbf{x}_i
 5: Költségfüggvény kiszámítása $f(\mathbf{x}_i)$
 6: Egyes limitértékek nullával inicializálása
 7: **end for**
 8: **while** $iter < MaxIter$ **do**
 9: **for** $i=1$ **to** N
 10: Szomszéd generálása
 11: Szomszéd fitnesszértékének meghatározása $f(\mathbf{x}_{i_neighbour})$
 12: **if** $f(\mathbf{x}_{i_neighbour}) < f(\mathbf{x}_i)$ **then**
 13: $\mathbf{x}_i \leftarrow \mathbf{x}_{i_neighbour}$
 14: egyed fitnesz = szomszéd fitnesz
 15: $limit_i = 0$
 16: **else**
 17: $limit_i = limit_i + 1$
 18: **end if**
 19: **end for**
 20: minden egyed kap egy valószínűsségi értéket $p(\mathbf{x}_i)$ 4.4-4.5 egyenletek alapján
 21: **for** $i=1$ **to** N
 22: rI véletlenszerű szám $\sim U(0,1)$
 23: **if** $p(\mathbf{x}_i) > rI$ **then**
 24: Szomszéd generálása
 25: Szomszéd fitnesszértékének meghatározása $f(\mathbf{x}_{i_neighbour})$
 26: **if** $f(\mathbf{x}_{i_neighbour}) < f(\mathbf{x}_i)$ **then**
 27: $\mathbf{x}_i \leftarrow \mathbf{x}_{i_neighbour}$
 28: egyed fitnesz = szomszéd fitnesz
 29: $limit_i = 0$
 30: **else**
 31: $limit_i = limit_i + 1$
 32: **end if**
 33: **end if**
 34: **end for**
 35: **for** $i=1$ **to** N
 36: **if** $limit_i < limit$ **then**
 37: Egyedek pozíciójának véletlenszerű kiválasztása
 38: Költségfüggvény kiszámítása $f(\mathbf{x}_i)$
 39: $limit_i = 0$
 40: **end if**
 41: **end for**
 42: **end while**

4.5. ábra ABC algoritmus pszeudókódja

4.2.3 Szürkefarkas optimalizálás (GWO)

A szürkefarkas optimalizálási algoritmus (*Grey Wolf Optimization*) a rajalapú, heurisztikus optimalizáló módszerek egyike. Viszonylag új módszert jelent, mivel legkorábban Seyedali Mirjalili publikálta 2014-ben. A többi algoritmushoz hasonlóan ezt is szintén a természet ihlette: a farkasfalkák vadászstartégiáját modellezi matematikai leírásokkal. A falka együttműködő, hierarchikus szerveződése ideális mintát ad a keresési tér feltárásához és a legjobb megoldás megtalálásához. [15][17]



4.6. ábra GWO algoritmus egyedeinek hierarchikus felépítése [17]

A falka tagjainak hierarchikus elrendeződését a 4.6. ábra szemlélteti. A falka élén az alfa állat áll, amely egyben a falkavezér szerepét is betölti. Döntéseik határozzák meg a vadászat időpontját és egyéb tevékenységeiket. Ugyanakkor megfigyelhető a részükről is egy követő viselkedés a többiek felé. A hierarchia második szintjén az úgynevezett béta farkasok állnak, akik a döntéshozatalban segítik az alfákat. A béta egyidejűleg respektálja a felette állót, azonban önállóan is irányíthatja a hierarchiában alatta elhelyezkedő egyedeket is. A piramis legalsó szintén az omegák helyezkednek el, ők mindenkinél alá vannak rendelve, látszólag nem fontos tagjai a falkának, azonban megfigyelték, hogy esetleges elvesztésük komoly feszültségeket okozott a többiek között. A be nem sorolt farkasokat deltával szokták jelölni, ide sorolhatjuk például a felderítő vagy őrszem egyedeket. [15][17]

Az optimum megkeresése a farkasok vadászati stratégiáján alapszik, melynek a főbb lépései a zsákmány felderítése, körbefogása és végül a letámadás. Ebben a részben a matematikai modell felállítása a cél és annak algoritmikus megvalósítása. Az algoritmus elején a korábbiakhoz hasonlóan inicializáljuk a kezdeti paramétereiket, a falka mérete, az iterációs zárt, valamint az alfa, béta és delta egyedeket. minden iterációs ciklus elején

frissítjük a megtalált megoldásokat. A legkisebb költséget adó az alfa, ezután béta, delta és a többi az omega pontok lesznek. Ezután következik a körbezárási ciklus. [15][17] Ezt a következő matematikai egyenletekkel modellezhetjük:

$$\vec{D} = |\vec{C} \cdot \vec{X}_p - \vec{X}| \quad (4.6)$$

$$\vec{X}(t+1) = \vec{X}_p - \vec{A} \cdot \vec{D} \quad (4.7)$$

, ahol \vec{X}_p a zsákmány pozícióvektora (a legjobb megoldás, tehát alfa pozíciójával egyenlő), \vec{X} az egyed pozícióvektora, és \vec{A} és \vec{C} pedig irány és véletlenszerűségi együtthatók. [15][17]

Utóbbi együtthatókat a következőképpen számíthatjuk ki.

$$a = 2 - t \cdot \frac{2}{\text{maximális iterációs szám}} \quad (4.8)$$

$$\vec{A} = 2 \cdot a \cdot \vec{r}_1 - a \quad (4.9)$$

$$\vec{C} = 2 \cdot \vec{r}_2, \quad (4.10)$$

ahol \vec{r}_1 és \vec{r}_2 nulla és egy közötti véletlenszámokat tartalmazó vektorok. „ a ” egy lineárisan csökkenő paraméter, amely a körbezárás és támadás mértékét szabályozza, „ t ” pedig az aktuális iteráció száma. [15][17]

A vadászat fázisa modellezi a legoptimálisabb megoldás felderítését a keresési térben. Ezt általában az alfa vezeti, de a béta és delta tagok is esetenként részt vehetnek benne. A valós megoldás helyét azonban nem ismerjük, ezért feltételezzük, hogy a hierarchia első három egyede rendelkezik a leg pontosabb információval a zsákmányt illetően. A 4.11-4.13. egyenletek meghatározzák az egyes távolságokat a vezető egyedektől, majd

$$\vec{D}_\alpha = |\vec{C}_1 \cdot \vec{X}_\alpha - \vec{X}_J| \quad (4.11)$$

$$\vec{D}_\beta = |\vec{C}_2 \cdot \vec{X}_\beta - \vec{X}_J| \quad (4.12)$$

$$\vec{D}_\delta = |\vec{C}_3 \cdot \vec{X}_\delta - \vec{X}_J| \quad (4.13)$$

A felderítő egyedek a 4.14-4.16. egyenletekhez hasonlóan frissítik a pozícióikat a három vezető farkashoz viszonyítva. [15][17]

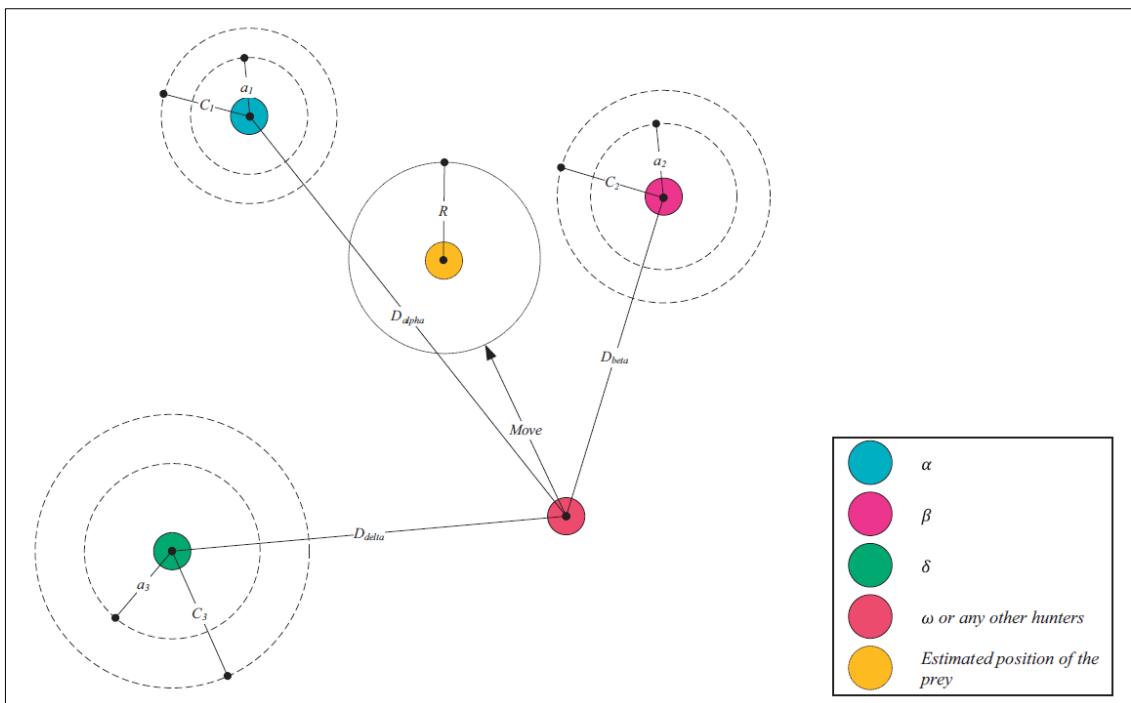
$$\vec{X}_1 = \vec{X}_\alpha - \vec{A}_1 \cdot \vec{D}_\alpha \quad (4.14)$$

$$\vec{X}_2 = \vec{X}_\beta - \vec{A}_2 \cdot \vec{D}_\beta \quad (4.15)$$

$$\vec{X}_3 = \vec{X}_\delta - \vec{A}_3 \cdot \vec{D}_\gamma \quad (4.16)$$

Az új végső pozíciót azonban az alfa, béta és delta által meghatározott pozíciók átlagaként számítjuk ki. Ez biztosítja, hogy a legjobb három vezető körül mozogjanak és folyamatosan konvergáljanak az optimum felé. Ezt mutatja a 4.7. ábra is. [15][17]

$$\vec{X}(t+1) = \frac{\vec{X}_1 + \vec{X}_2 + \vec{X}_3}{3} \quad (4.17)$$



4.7. ábra Adott pozíció frissítése a három legjobb megoldást jelentő egyed alapján [17]

Az algoritmust leíró pszeudódokódot a 4.8.ábra szemlélteti.

-
- 1: Paraméterek inicializálása
 - 2: $N, MaxIter$
 - 3: **for** $i=1$ **to** N
 - 4: Egyedek pozíciójának inicializálása egyenletes eloszlású véletlenvektorral x_i
 - 5: Költségfüggvény kiszámítása $f(x_i)$
 - 6 **end for**
 - 7: $X_\alpha = \text{legjobb egyed}$
 - 8: $X_\beta = \text{második legjobb egyed}$
 - 9: $X_\delta = \text{harmadik legjobb egyed}$

```

10: while  $iter < MaxIter$  do
11:   for  $i=1$  to  $N$ 
12:     Költségfüggvény kiszámítása  $f(\mathbf{x}_i)$ 
13:      $X_\alpha, X_\beta, X_\delta$  pozíciók frissítése a következő szabály szerint
14:     if  $f(\mathbf{x}_i) < f(X_\alpha)$  then
15:        $X_\alpha \leftarrow \mathbf{x}_i$ 
16:       béta fitnesz = előző alfa fitnesz
17:       delta fitnesz = előző béta fitnesz
18:       alfa fitnesz frissítése az aktuális pozíció szerint
19:     else
20:     if  $f(\mathbf{x}_i) < f(X_\beta)$  then
21:        $X_\beta \leftarrow \mathbf{x}_i$ 
22:       delta fitnesz = előző béta fitnesz
23:       béta fitnesz frissítése az aktuális pozíció szerint
24:     else
25:     if  $f(\mathbf{x}_i) < f(X_\delta)$  then
26:        $X_\delta \leftarrow \mathbf{x}_i$ 
27:       béta fitnesz frissítése az aktuális pozíció szerint
28:     end if
29:      $r1, r2$  – véletlenszerű számok  $\sim U(0,1)$ 
30:      $a$  – lineárisan csökkenő paraméter
31:      $A, C$  frissítése
32:     távolságok kiszámítása 4.11-4.13 egyenletek alapján
33:     köztes pozíciók kiszámítása 4.14-4.16 egyenletek alapján
34:     aktuális pozíció frissítése a 4.17 egyenlet alapján, az előző állapotok átlagából
35:   end for
36: end while

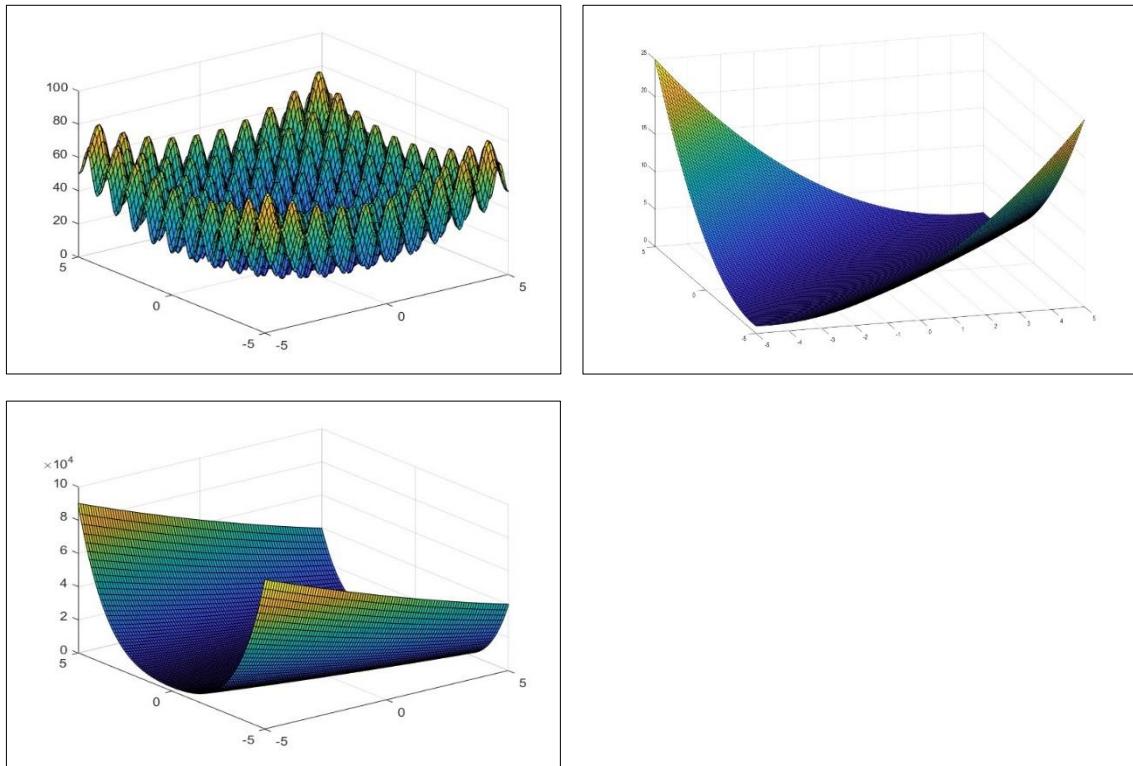
```

4.8. ábra GWO algoritmus pszeudódaja

4.3 Optimalizálási módszerek összehasonlítása

Az optimalizálási feladatok megoldására kifejlesztett algoritmusok hatékonyságának vizsgálatára gyakran alkalmaznak különböző tesztfüggvényeket. Ezek lehetővé teszik a módszerek összehasonlítását, valamint az eredményekkel segítséget nyújtanak a paraméterek hangolásában. A függvények speciális tulajdonságokkal rendelkezhetnek. Ide sorolhatók a lokális szélsőértékek nagy mennyisége, az optimum közelében ellaposodó forma, amely erősen befolyásolja a konvergencia sebességét. Amennyiben ismerjük, vagy becsülhetjük a költségfüggvény alakját (például kvadratikus függvény, vagy trigonometrikus tagok akár több szélsőértéket eredményezhetnek), úgy a módszerek gyengeségeinek ismerete segítséget nyújthat, hogy a maximális hatékonyság

érdekében a leginkább megfelelőt választhassuk ki. Az előbbi fejezetben részletesebben is ismertetett algoritmusokat a 4.1. táblázatban megadott tesztfüggvényeken futtattam le. A 4.9. ábrák grafikusan is szemléltetik a különböző tesztfüggvényeket.



4.9. ábra Tesztfüggvények. Balról jobbra, soronként: Rastrigin, Matyas és Rosenbrock függvények

| No | Függvény neve | Intervallum | Formula | Globális optimum |
|----|---------------|-------------|--|----------------------|
| 1 | Rastrigin | $[-5,5]$ | $20 + [x^2 - 10 \cos(2\pi x)] + [y^2 - 10 \cos(2\pi y)]$ | $\mathbf{x} = (0,0)$ |
| 2 | Rosenbrock | $[-5,5]$ | $(1-x)^2 + 100(y-x^2)^2$ | $\mathbf{x} = (1,1)$ |
| 3 | Matyas | $[-10,10]$ | $0.26(x^2 + y^2) - 0.48xy$ | $\mathbf{x} = (0,0)$ |

4.1. táblázat

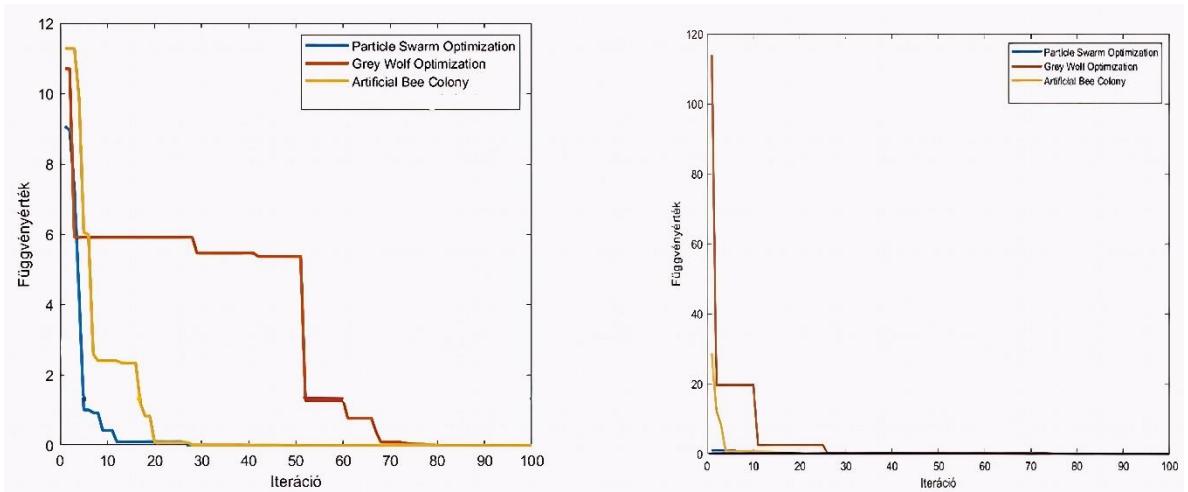
Az algoritmusok összehasonlíthatósága végett az egyes módszereket azonos kezdeti értékekkel, kezdeti populáció nagysága, valamint maximális iterációk száma, kerültek inicializálásra. A populáció mérete minden esetben 30, míg a számítások maximum 100

iterációig futnak. A különböző algoritmusokkal megtalált optimumokat a 4.2. táblázat foglalja össze. Az többszöri futtatás eredményei ezredes pontossággal lettek ábrázolva. Jól látható, hogy alapvetően minden a három módszer alkalmas akár összetettebb költségfüggvények optimumának megtalálására.

| No | Függvény neve | Megtalált globális optimum | | |
|----|---------------|----------------------------|----------------|------------------|
| | | PSO | GWO | ABC |
| 1 | Rastrigin | (-0.003, -0.0005) | (0,0) | (0.009, 0.004) |
| 2 | Rosenbrock | (1, 1) | (0.965, 0.925) | (0.921, 0.877) |
| 3 | Matyas | (0, 0.002) | (0, 0) | (-0.019, -0.019) |

4.2. táblázat

A 4.10. ábrán összehasonlíthatjuk a három módszer konvergenciájának sebességét. A bal ábrán a Rastrigin függvény optimumkeresési folyamata látható. A részecske rajintelligencia bizonyult a leghatékonyabbnak, de a méhraj optimalizálás is gyorsan, húsz iteráció után az optimum kis környezetébe kerül. A lokális minimumok a szürke farkas algoritmus esetén azonban egy hosszú platót eredményezett, amelyből kilépve a megoldás gyorsan az zérus hely felé tartott. Ahogyan láthatjuk a GWO algoritmus könnyen ragad lokális szélsőérték-helyen, amelyből nem garantálható, hogy kiszabadul. Általánosságban a költségfüggvény azonban nem ennyire komplex, mint az iménti szituációban, egyszerűbb függvények esetén mindegyik módszer jó megoldást eredményez alacsony iterációs zárt mellett. Gyakorlati alkalmazásokban a leggyakrabban a négyzetes hibát szeretnénk minimalizálni, amely biztosan rendelkezik globális minimummal és esetlegesen néhány lokális minimumot tartalmazhat, amennyiben plusz tagokat is figyelembe veszünk. Ebben az esetben a jobb oldali ábrán látható konvergenciát mutat az algoritmus.



4.10. ábra PSO, ABC és GWO algoritmusok teljesítményének összehasonlítása azonos iterációszám mellett

5 Modell prediktív irányítás rajintelligencia módszer alapú optimalizálással

Az ismertetett optimumkeresési eljárások – úgymint a részecske-rajintelligencia (PSO), szürkefarkas optimalizáció (GWO) és méhraj intelligencia (ABC) – új perspektívát kínálhatnak a hajtásszabályozási problémák kezelésére. E megközelítés lehetővé teszi, hogy a szabályozási paraméterek keresési terében globális optimumközeli megoldásokat találjunk, miközben a rendszer dinamikai tulajdonságait javítja, a válaszidőt csökkenti és a szabályozási hibát minimalizálja. A módszerek újszerű alkalmazása különösen érdekes a prediktív szabályozási keretbe integrálva, ahol adaptív módon segítik a vezérlőjelek optimalizálását.

Ebben a fejezetben a három említett intelligens algoritmust integráltuk a már meglévő, modell-alapú prediktív szabályozási keretbe. A beágyazás során módosítottuk a működési stratégiát, hogy a hagyományos, előre definiált vezérlési döntések helyett az algoritmusok adaptív módon választhassák ki az optimális vezérlőjelet minden időpillanatban. A legfontosabb különbség az egyes megoldások között az optimum megtalálásának algoritmusában rejlik: míg a PSO az egész keresési teret egy populáció részecskéinek mozgásával járja be, a GWO a farkasok falkán belüli hierarchikus mechanizmusra támaszkodik, az ABC pedig a méhek táplálékkeresési viselkedését modellezi.

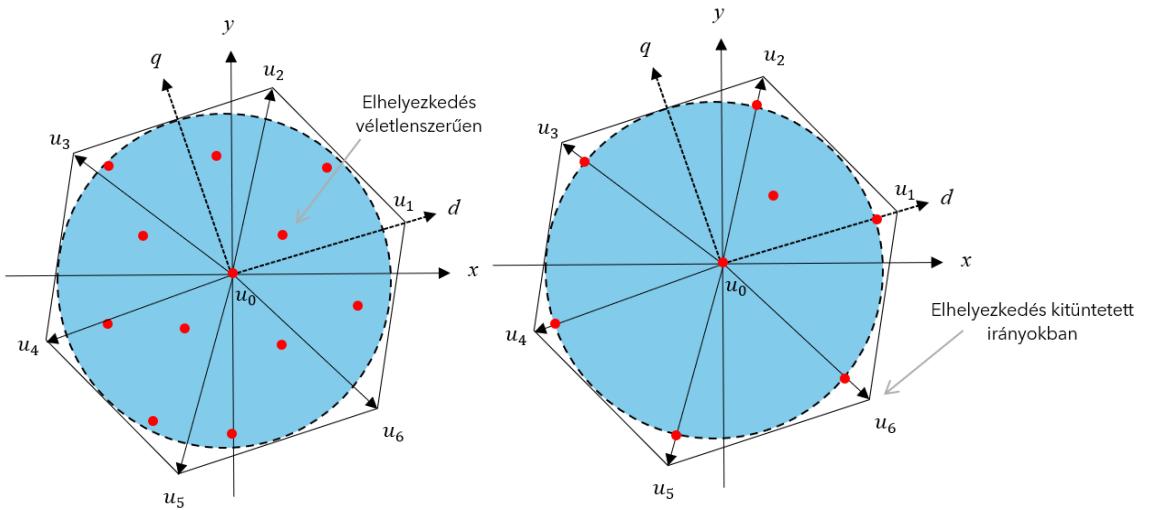
5.1 Kezdeti állapot és célfüggvény

A kezdeti populáció kiválasztása az optimumkeresési algoritmusok teljesítménye szempontjából kiemelkedően fontos tényező, mivel meghatározza az algoritmus keresési hatékonyságát és a megtalált megoldások minőségét. A populáció inicializálására többféle stratégia alkalmazható, amelyek eltérő előnyöket biztosíthatnak a rendszer számára. Ezeket mutatja be a 5.1. ábra is.

Az egyik lehetőség a populáció véletlenszerű generálása a teljes keresési térben, meghatározott populációs szám mellett. Ebben az esetben a keresési pontok a lehetséges vezérlőjelek teljes tartományát lefedik, így az algoritmus széleskörűen vizsgálhatja a lehetséges megoldásokat. Ez a módszer különösen hasznos akkor, ha nincs előzetes

információ a rendszer optimális működéséről, vagy ha a cél egy globális optimum megtalálása.

Alternatív megközelítésként a populáció kialakítása a már korábban alkalmazott diszkrét feszültségvektorokból történhet, amelyeket a kétszintű, háromfázisú inverter képes generálni. Ez a stratégia biztosítja, hogy minden kezdeti pont fizikailag érvényes és a rendszer által megvalósítható vezérlőjelet képvisel, így csökkentve a hirtelen, nem kívánt dinamikai változások kockázatát.



5.1 . ábra Kezdeti populáció inicializálásának lehetőségei [1]

A populáció minden esetben tartalmazza a nullvektort, valamint az előző időpillanatban kapott optimális megoldást. Ez a gyakorlatban azt jelenti, hogy az algoritmus minden rendelkezik egy kiindulási ponttal, amely biztosítja a vezérlőjelek folytonosságát, és megakadályozza a hirtelen, nagymértékű ugrásokat, amelyek instabilitáshoz vagy túlzott mechanikai terheléshez vezethetnek. Az előző megoldás felhasználása különösen fontos a prediktív szabályozásban, ahol a rendszer folyamatos működése és a vezérlőjelek sima változása alapfeltétel a stabil és megbízható működéshez. [1]

Ahogy korábban láthattuk, a prediktív szabályozás célja a rendszer jövőbeli viselkedésének előrejelzése és optimalizálása az aktuális állapotok alapján a predikciós horizonton. A szabályozó a lehetséges beavatkozások közül azt választja, amely minimalizálja a definiált költségfüggvényt. A jelen esetben alkalmazott költségfüggvény az alábbi formában írható fel: [1]

$$J = w_{iq}(i_q^*[k+2] - i_q)^2 + w_{id}(i_d^*[k+2] - i_d)^2 + \\ + \lambda_u \sum_{x \in \{d,q\}} (u_x^*[k+2] - u_x)^2 \quad (5.1)$$

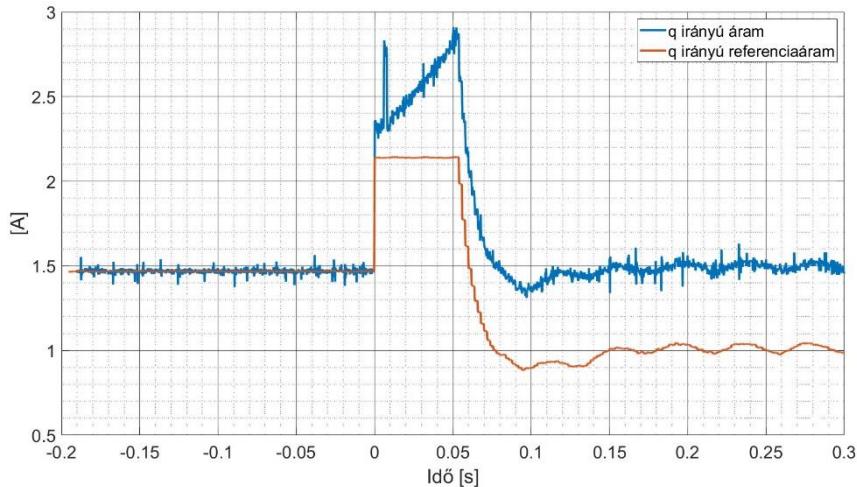
A fenti összefüggés három fő tagból áll, amelyek az áram- és feszültségkomponensek közötti eltérések súlyozott. Az első két tag a d és q tengelyek menti áramkomponensek referenciaértékeit és mért értékeit hasonlítja össze. Az i_d áram a mágneses fluxus szabályozásáért, míg az i_q áram a motor által előállított elektromágneses nyomatékért felelős. A két taghoz tartozó w_{id} és w_{iq} súlyokkal szabályozható, hogy a költségfüggvény milyen mértékben büntesse az egyes áramkomponensek eltérését. Általában a nyomatékpontosság az elsődleges cél, ezért előfordulhat, hogy a q irányú áram eltérését nagyobb súlyal ($w_{iq} > w_{id}$) veszik figyelembe. [1]

A harmadik tag a d és q irányú feszültségkomponensek hibáját tartalmazza, amelyek az aktuális feszültségek és az algoritmus által előrejelzett értékek különbségeit mérik közös λ_u súlytényező alatt. Ez a kifejezés több szempontból is fontos: egyrészt korlátozza a feszültségvektor nagyságát, másrészt simítja a szabályozási beavatkozásokat, ezáltal csökkenti a kapcsolási veszteségeket. A λ_u érték növelésével a szabályozó kisebb feszültségváltozásokat enged meg, ami simább, de lassabb dinamikát eredményez. Ezzel szemben kisebb λ_u esetén a szabályozó gyorsabb válaszidőt biztosít, viszont növelheti a feszültségingadozásokat és a kapcsolási igénybevételt. [1]

5.2 Egyszerűsített Kálmán-szűrő zajos mérésekre

A motor áramainak lehető leg pontosabb mérése elengedhetetlen a vezérlés és az állapotbecslés szempontjából. A gyakorlatban azonban az áramérzékelők kimenete mindenkor tartalmaz bizonyos mértékű mérési zajt. Ezek a hibák közvetlenül befolyásolják a vezérlés pontosságát, különösen nagy sebességnél jelentős eltérést okozva a modellünkben.

A zajos mérés közvetlen felhasználása torzítja a motor állapotára vonatkozó becsléseket, emiatt a szabályozó nem a valós fizikai állapotokra reagál, a predikciós modell nem követi megfelelően a valós áramalakokat. Ezt mutatja az 5.2 .ábra is. A q irányú áramokban egy állandó offset jelenik meg.



5.2. ábra Áramkövetés Kálmán-szűrő nélkül

A probléma kiküszöbölésére [25] alapján bevezettem egy egyszerűsített, Kálmán-szűrő alapú állapotbecslő algoritmust, amely a mért adatokból és a motor matematikai modelljéből származó információkat egyaránt felhasználja, így képes a mérési zajt kiszűrni.

A Kálmán-szűrő működésének elméleti alapjait a 2.5. fejezet ismerteti részletesen, ebben a fejezetben pedig a gyakorlati megvalósítás és az alkalmazott egyszerűsítés kerül bemutatásra.

A tényleges megvalósítás során a teljes Kálmán-szűrő implementálása túlzott számítási igényt jelentene a beágyazott rendszeren, ezért egy egyszerűsített formát alkalmaztam, amelyben a d és q tengelyű komponensek becslése külön-külön, egydimenziós Kálmán-szűrőkkel történik. Ez a megközelítés jelentősen csökkenti a számítási igényt, miközben a módszer megőrzi a Kálmán-szűrő legfontosabb tulajdonságait: a zajcsökkentést, a mérési pontok eltolódásának kompenzációját és a pontosság javítását. [25]

A predikciós lépésben a d- és q-tengelyű komponensek előrejelzése az alábbi összefüggések alapján történik:

$$\hat{i}_{k,p}^d = A_{11}i_d + A_{12}i_q + Bv_d, \quad (5.2)$$

$$\hat{i}_{k,p}^q = A_{21}i_d + A_{22}i_q + d + Bv_q, \quad (5.3)$$

A Kálmán-erősítések meghatározása a mérési és becslési bizonytalanság arányán alapul, és az alábbi formában adható meg:

$$K_d = \frac{P_k^d}{P_k^d + R}, \quad (5.4)$$

$$K_q = \frac{P_k^q}{P_k^q + R}, \quad (5.5)$$

A korrekciós lépés során a becsült állapot a mért értékek figyelembenével módosul:

$$\hat{i}_k^d = \hat{i}_{k,p}^d + K_d(i_d - \hat{i}_{k,p}^d - b_d), \quad (5.6)$$

$$\hat{i}_k^q = \hat{i}_{k,p}^q + K_q(i_q - \hat{i}_{k,p}^q - b_q), \quad (5.7)$$

ahol b_d és b_q a d- és q-tengelyekhez tartozó offszet értékeket jelölik, melyek szintén rekurzívan frissülnek [25]:

$$b_d \leftarrow b_d + K_d(i_d - \hat{i}_{k,p}^d - b_d), \quad (5.8)$$

$$b_q \leftarrow b_q + K_q(i_q - \hat{i}_{k,p}^q - b_q). \quad (5.9)$$

A kovarianciák frissítésére a következő összefüggések érvényesek:

$$P_{k+1}^d = (1 - K_d)P_k^d + Q_d, \quad (5.10)$$

$$P_{k+1}^q = (1 - K_q)P_k^q + Q_q, \quad (5.11)$$

A korrekció után a becsült és eltolás-kompenzált áramértékek a következőképpen adódnak:

$$i_k^{d,corr} = \hat{i}_k^d + b_d, \quad (5.12)$$

$$i_k^{q,corr} = \hat{i}_k^q + b_q, \quad (5.13)$$

A következő időpillanatra vonatkozó előrejelzés ismét a modell alapján számítható:

$$\hat{i}_{k+1,p}^d = A_{11}i_k^{d,corr} + A_{12}i_k^{q,corr}, \quad (5.14)$$

$$\hat{i}_{k+1,p}^q = A_{21}i_k^{d,corr} + A_{22}i_k^{q,corr} + d. \quad (5.15)$$

Az alkalmazott eljárás előnye, hogy a becslés rekurzív formában történik, így nincs szükség nagyméretű mátrixműveletekre. A szűrő megfelelő paraméterezése mellett hatékonyan csökkenti a mérési zajt, kompenzálja az esetleges eltolásokat, és javítja az áramkomponensek becslésének pontosságát.

5.3 Véletlenszám generálás

A véletlenszám-generálás kulcsfontosságú szerepet tölt be számos sztochasztikus optimalizálási módszerek esetén, mint például a részecskeraj vagy a szürke farkas algoritmus. Ezek az eljárások a keresési tér feltérképezése során véletlenszerű változókat alkalmaznak annak érdekében, biztosításak a megoldások megfelelő diverzitását és a globális optimum megtalálásának valószínűségét. [11]

A véletlen vagy pszeudo-véletlen számok előállítására számos algoritmus létezik. Ezen algoritmusok fejlesztésekor a fő tervezési szempontok a gyorsaság, az egyszerűség, a jó statisztikai tulajdonságok és a kód könnyű integrálhatósága voltak. Ezen kritériumok alapján kerül gyakran alkalmazásra egy, a lineáris kongruens módszerre (Linear Congruential Method, LCM) épülő megoldás, amelyet eredetileg D. Lehmer javasolt 1951-ben. [11]

A módszer fő előnye a nagy futási sebesség, az egyszerű megvalósíthatóság, valamint a könnyű használat. Ugyanakkor fontos megjegyezni, hogy a szorzó (MULT) és a növekmény (INC) értékeinek nem megfelelő megválasztása az eredmények gyors degenerálódásához vezethet. A jelen megvalósítás 65 536 egyedi számot képes előállítani és csak a legkisebb helyiértékű bit (LSB) mutat ismétlődő mintázatot minden 16. hívás után. A lineáris kongruens módszer matematikai alakja a következőképpen írható fel: [11]

$$Rndnum(n) = (Rndnum(n - 1) \times MULT) + INC(modM)$$

Ahol $Rndnum(n)$ az aktuális véletlenszám, $Rndnum(n - 1)$ az előző véletlenszám, , $Rndnum(1)$ a kezdeti (SEED) érték, $MULT$ a szorzó (állandó), INC a növekmény (állandó), a modulus érték, amely a processzor szószélessége-től függ (esetünkben 65 536). [11]

5.4 Szimuláció MATLAB környezetben

Az előző fejezetben bemutatott rajintelligencia módszereket először MATLAB/Simulink környezetben implementáltam. A rendszermodell elemei megegyeznek a direkt MPC hajtás szimulációjában megépített blokkokkal. Ide tartozik a PMSM motor és a háromfázisú, kétszintű inverter modellje. Az optimális irányítás a belső áramszabályozási körben valósul meg, a külső, sebességszabályozó körben egy egyszerű PI szabályozót alkalmaztam. A mezőorientált vezérlés (FOC) megvalósítása érdekében a modell tartalmazza az ehhez szükséges koordináta-transzformációkat, úgymint a Clarke és Park transzformáció. A koordinátatranszformációk segítségével a háromfázisú, időben szinuszosan váltakozó mennyiségek egy forgó d-q koordinátarendszerbe képezhetőek le. Ez lehetővé teszi, hogy a fluxus-, és nyomatékképző áramkomponenseket egymástól függetlenül kezeljük. Az eredményeket a részecskeraj optimalizáció (PSO), a szürke farkas optimalizáció (GWO) és a mesterséges méhraj algoritmusok (ABC) esetében a 5.3, 5.4. és 5.5. ábrák mutatják be. Az ábrákon a motor áram-, nyomaték- és szögsebesség-időfüggvényei láthatók, amelyeket ugrásszerűen változó terhelőnyomaték mellett vizsgáltam.

A szimuláció során a motor paraméterei a valós, laboratóriumban alkalmazott PMSM motor paramétereivel egyeznek meg, melyeket az 5.1. táblázat tartalmaz. A szimuláció 0,25 s-ig tartott ugrásszerűen változó szögsebesség-referencia mellett. Az idő körülbelül 2/3-ánál, azaz 0,16 s-nél, 0,18 Nm nagyságú terhelőnyomatékot kapcsoltam a motorra.

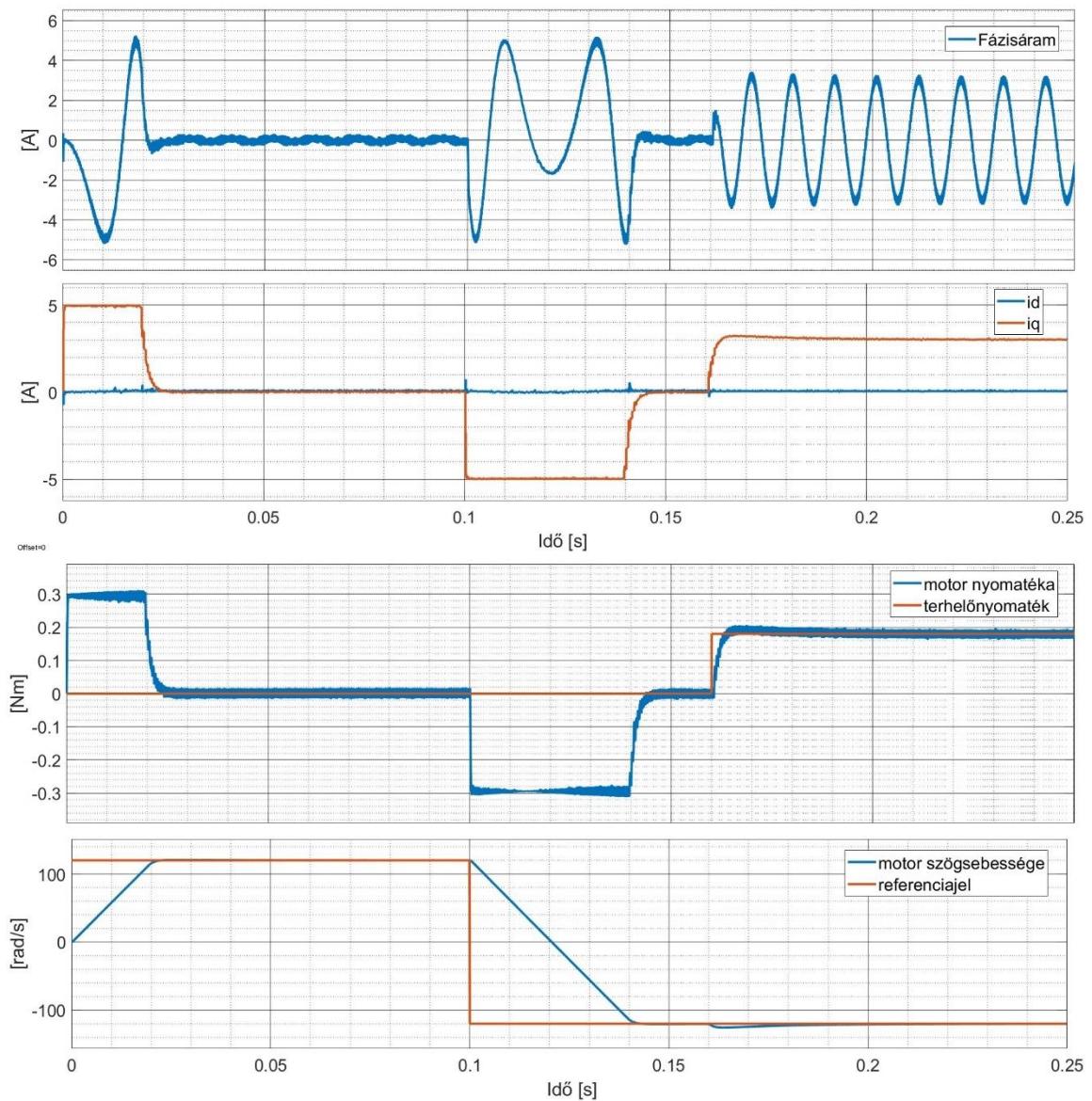
| Gépparaméterek | Jelölés | Érték |
|----------------------------|----------|-----------|
| Állórész ellenállása | R_s | 235 mΩ |
| Forgórész mágneses fluxusa | Ψ | 0.0079 Vs |
| Állórész induktivitása | L_s | 320 μH |
| Póluspárok száma | p | 5 |
| DC-busz feszültsége | U_{DC} | 24 V |

5.1. táblázat PMSM motor paraméterei

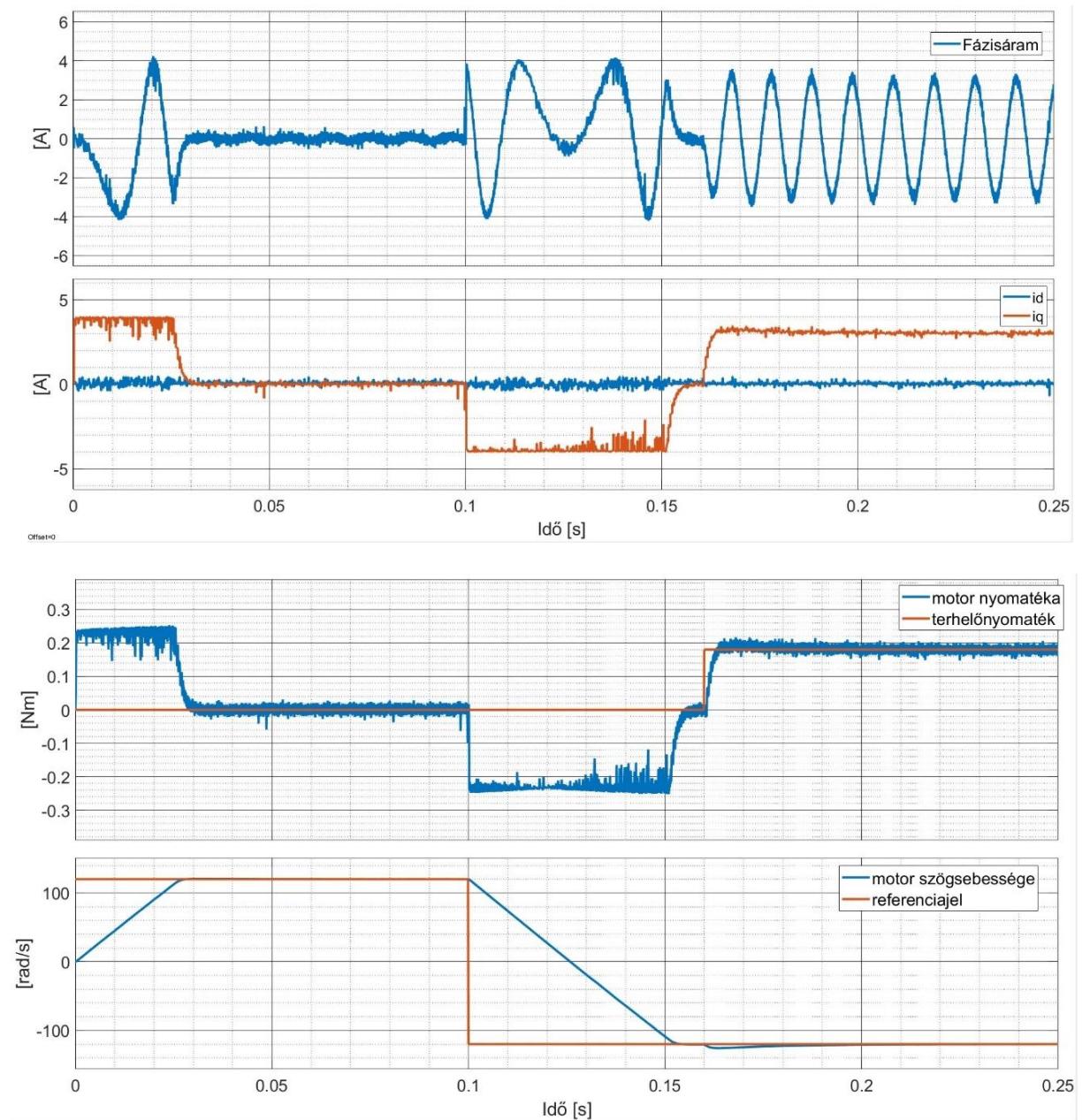
A mintavételi frekvenciát 10 kHz-nek választottam, továbbá a mechanikai szögsebesség szabályozásához szükség volt a tehetetlenségi nyomaték és a csillapítás megadására is. Ezek rendre 0.5 kgcm^2 és 0.01 mNm .

A külső sebességszabályozóban egy PI szabályozó került implementálásra, 0.5-ös proporcionális és 50-es értékű integráló együtthatóval. Mintavételi ideje pedig ötszöröse a belső szabályozási körnek, tehát 0.5 ms. Az elintegrálódás elkerülése végett egy 5 A-es szaturációs érték is beállításra került. Az (5.1) egyenletben leírt költségfüggvény áramokra előírt súlyait 1-nek vettetem, míg az aktuálisan kapcsolt feszültségvektoruktól vett távolság súlyai 0.1-es értékűek, mind d és q irány esetén.

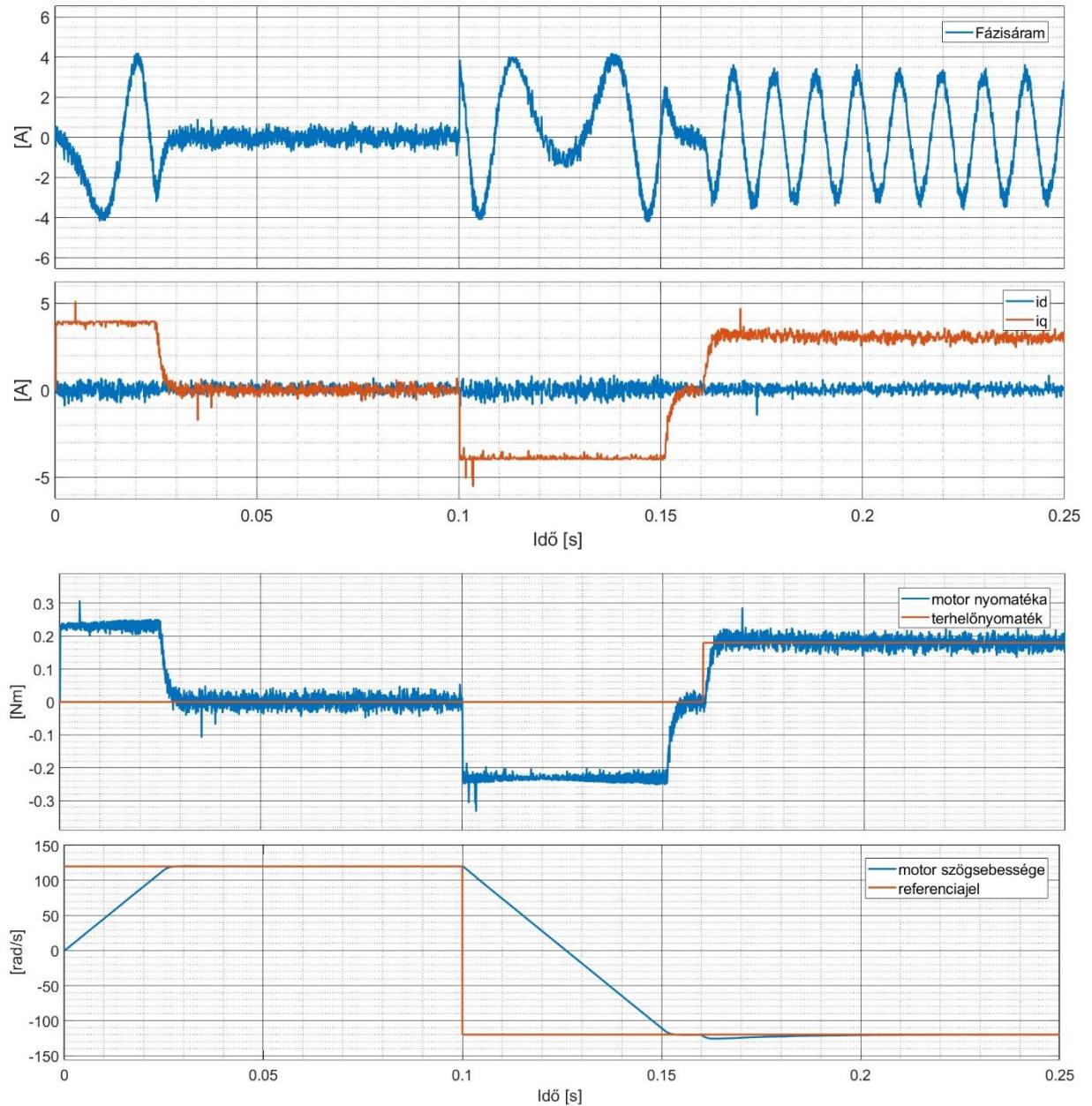
A három optimalizációs eljárás mindegyike megfelelően követte a rendszer dinamikáját, azonban a kimeneti jel zajossága eltérő mértékben jelentkezett. A legkisebb zaj a PSO algoritmus esetében figyelhető meg, ezt követte a GWO, míg a legnagyobb zaj az ABC algoritmus eredményeinél tapasztalható.



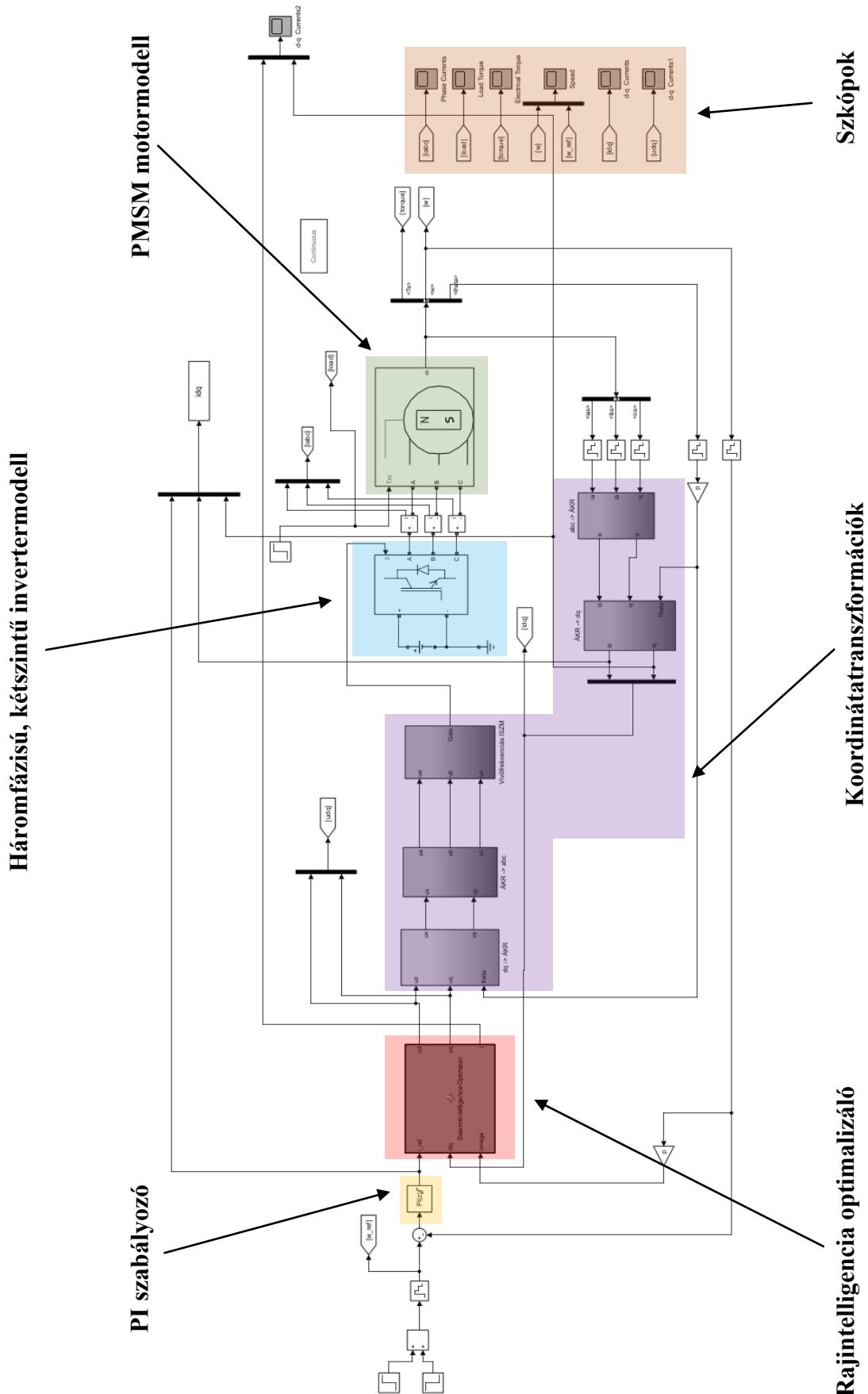
5.3. ábra MPC irányítás szimulációja PSO optimalizálással



5.4. ábra MPC irányítás szimulációja GWO optimalizálással



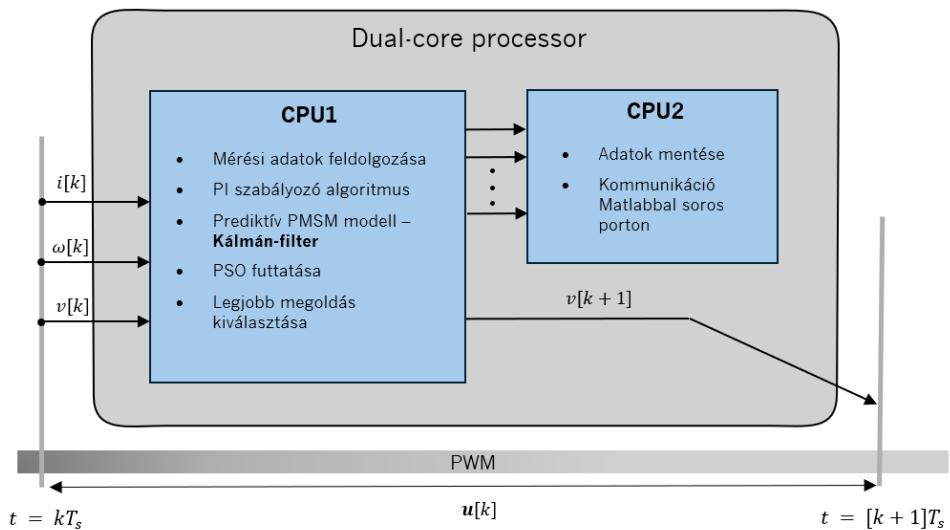
5.5. ábra MPC irányítás szimulációja ABC optimalizálással



5.6. ábra Rajaintelligencia-alapú MPC irányítás Simulink modellje

5.5 Valós hardveres implementáció

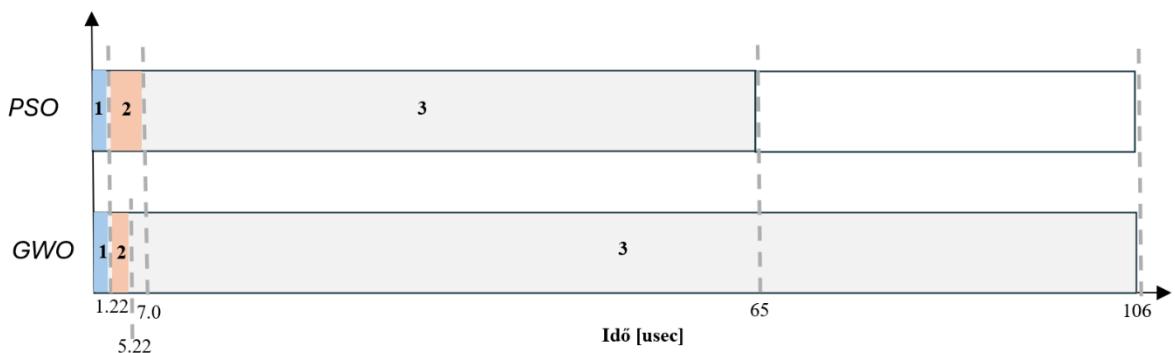
A szimulációs környezetben futtatott teszt után a rajintelligencia alapú modell prediktív irányítást egy Texas Instruments gyártmányú TMS320F2837D mikrovezérlőn implementáltam. Az eszközben található kétmagos processzor lehetőséget biztosít bizonyos feladatok párhuzamos feldolgozására. Az egyes processzorok közötti feladatmegosztást a 5.7. ábra mutatja. Az analóg-digitális átalakító felől érkező méréseket a CPU1 fogadja, amelyen egyidejűleg került sor a predikciós és vezérlési lépés megvalósítására. A számítógép soros porton keresztül kommunikál a mikrovezérlővel. Itt történik a referenciajelek előírása, esetenként bizonyos paraméterek módosítása, valamint a kívánt adatok mentése is.



5.7. ábra CPU1 és CPU2 processzorok közötti feladatmegosztás

A szenzorjelek beolvasása és a vezérlő algoritmus futtatása megszakításban történik. A megfelelő mintavételi frekvencia megválasztása elengedhetetlen a helyes működés szempontjából. A túl nagy mintavételi frekvencia kritikus lehet az optimalizációra nézve, biztosítani kell, hogy a teljes algoritmus beleférjen a megadott időkeretbe. A ritka mintavételezés a szabályozás instabilitását okozhatja. A feladat során 5 kHz-es mintavételi frekvenciát választottam, amely mindegyik kritériumot kielégíti. Ehhez először meg kellett vizsgálni az algoritmusok futási idejét, ezenkívül ez alkalmat ad arra, hogy a módszereket összehasonlíthassuk teljesítményük alapján. A 5.8. ábra a PSO és GWO optimalizáló módszerek teljes ciklusidejét szemlélteti. A pontos adatokat az ábra alatt található táblázatban foglaltam össze. Először a szenzorjelek beolvasása

történik. Ide tartozik a mérési adatok feldolgozása, az implementált Kálmán-szűrő futtatása és az MPC-hez szükséges két lépéssel előretartó predikciós lépés. Ez természetesen módszertől függetlenül azonos mindegyik esetben, kb. 1.2 μ s-ot vesz igénybe. A következő lépésben az egyedek inicializálása történik. PSO algoritmus esetén, ahogy láthattuk a pozíciók mellett kezdeti sebességeket is meg kell határoznunk, ami kissé megnöveli a szükséges időt, GWO algoritmus ebben 1.78 μ s-al gyorsabb. A ciklusidő legnagyobb részét a célfüggvény minimumának megkeresése teszi ki. Itt jelentős eltérés van a kettő módszer között. GWO esetén a távolságszámítások okozzák az idő megnövekedését. Az első három legjobb pozíciótól kell kiszámítani a távolságokat, minden esetben kétdimenziós vektorokkal dolgozunk. Egy egyed esetén ez már 6 távolságot jelent és még szorzódik az egyedek és iterációk számával. Ezután ezzel még további műveleteket is végzünk. PSO esetén csak a lokális és globális ponttól vett távolságra van szükségünk, és a pozíció frissítése is egyszerűbb logikát követ. Míg előbbi kicsivel több mint 100 μ s-ig tart, addig utóbbi esetben rövidebb idő is elég. A teljes futási idő tekintetében 1.63-as szorzó adódik PSO javára.



1 – Szenzorjelek feldolgozása és Kálmán szűrő

2 – Egyedek inicializálása

3 – Optimumkeresés/Algoritmus futtatása

| | PSO | GWO |
|----------------------------|-----------------------------|------------------------------|
| Mérés, szűrés és predikció | 1.22 μ s | 1.22 μ s |
| Inicializálás | 5.78 μ s | 4 μ s |
| Optimumkeresés | 58 μ s | 101 μ s |
| Összesen | 65 μs | 106 μs |

5.8. ábra PSO és GWO ciklusidők összehasonlítása megadva

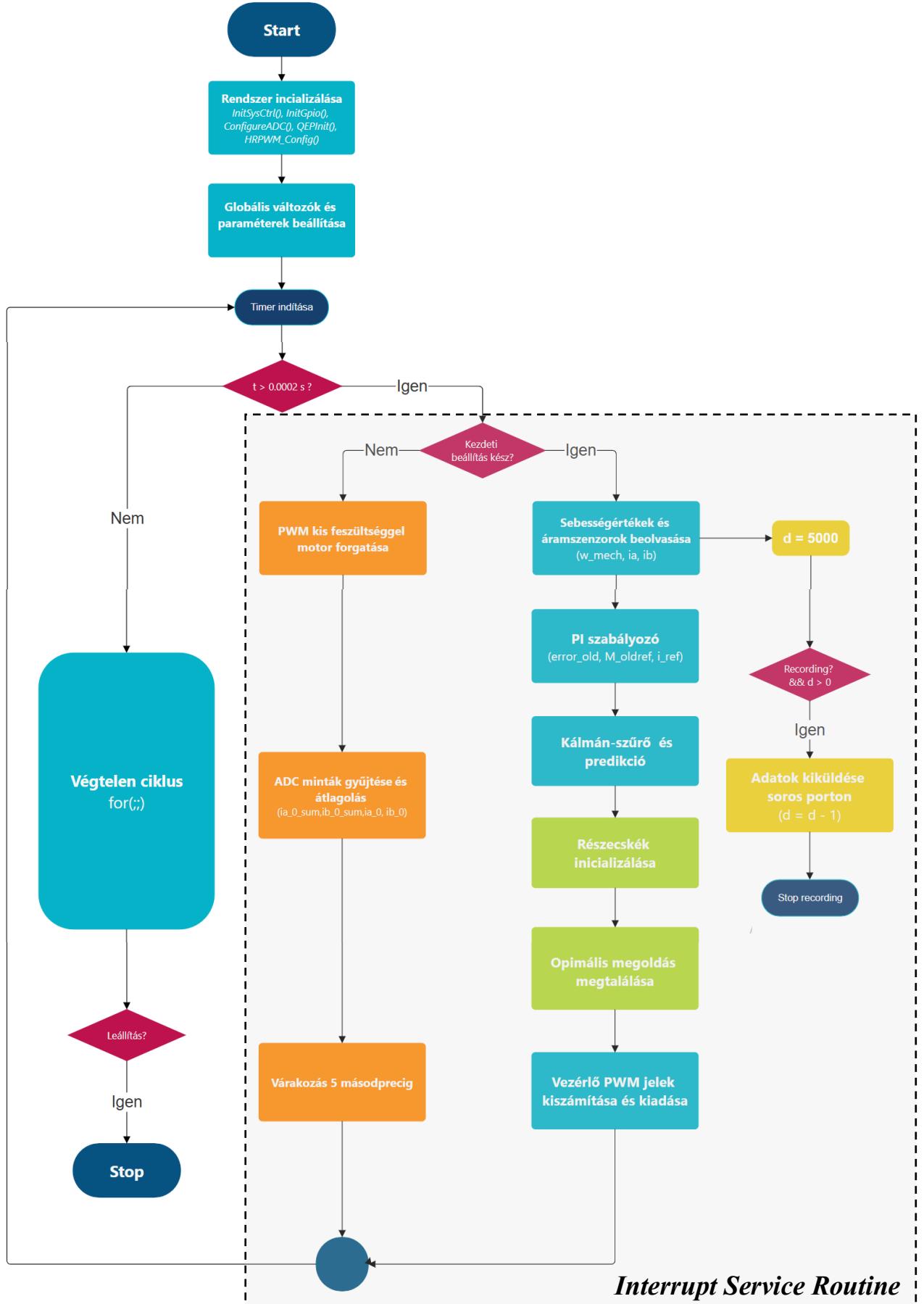
5.5.1 Prediktív irányítás C nyelvű megvalósítása

Ezen alfejezet célja, hogy bemutassa a háromfázisú PMSM motor prediktív szabályozásának valós idejű megvalósítását C nyelven, az előbbi fejezetben ismertetett beágyazott eszközön. A program a Code Composer Studio fejlesztőkörnyezetében került implementálásra, amely nem csak a program fordítását és indítását teszi lehetővé, de tesztelés során debug üzemmódban is használhatjuk. A folyamat során a motor áramait és forgórészének sebességét kívánjuk optimalizált módon szabályozni. A program működése logikailag három fő szakaszra bontható: a rendszer inicializációjára, a valós idejű vezérlési ciklusra, valamint az adatrögzítés és kommunikáció kezelésére. A program működését és legfontosabb lépései a 5.9. ábrán megrajzolt flowchart mutatja be.

Miután biztosítottuk a megfelelő tápellátást, és a lefordított kódot feltöltöttük a mikorvezérlőre, a fejlesztőkörnyezetebe egy gombnyomással elindíthatjuk a program futását. Ahogy már említettük az első lépés az inicializálás. Ennek során elvégezzük motor és a szabályozó, valamint az optimalizációs algoritmus paramétereinek betöltését amelyek a motor fizikai tulajdonságait és a vezérlési algoritmus működését definiálják. A motorparaméterek közé tartozik a d és q tengelyekre értelmezett induktivitás, valamint a motor armatúraellenállása, rotorfluxusa és a póluspárok száma is. Itt definiáltam a mintavételi időt, amely a megszakítások ütemezéséhez szükséges. Jelen esetben ez 5 kHz. Továbbá ez került felhasználásra a diszkrét szabályozó és a prediktív rendszermodell megépítésében is. A konstansokhoz tartoznak még az optimalizáló algoritmust meghatározó jellemzők, úgymint az egyedek száma, az iterációszám, illetve az egyes populációk számára kialakított adatstruktúra definiálása is.

A paraméterek betöltését követően a program a hardveres perifériák konfigurálásával folytatódik. Az EPWM modulok beállítása a megfelelő kimenetek generálásához szükségesek, amelyek a motor egyes fázisait vezérlik. Az ADC modulok konfigurációja a szenzorbemenetek kezelésére szolgálnak. Az időzítő úgy van beállítva, hogy minden mintavételkor megszakítást generáljon, ezáltal biztosítva, hogy a vezérlési ciklus determinisztikusan induljon el minden 200 μ s-ban.

A rotor pozíójának és szögsebességének meghatározására a QEP (Quadrature Encoder Pulse) egység kerül konfigurálásra. Ez lehetővé teszi a motor forgórészének nyomon követését, és biztosítja a d–q koordinátarendszerbe történő transzformációhoz szükséges szöginformációt.



5.9. ábra Prediktív irányítás C nyelvű megvalósításának folyamatábrája

A GPIO lábak és a dead-time modul konfigurációja biztosítja a PWM jelek biztonságos generálását. A dead-time engedélyezése megakadályozza a tranzisztorok együttes vezetését, összenyitását, ezáltal elkerülhető a rövidzárlat. A GPIO lábak konfigurációja kiterjed még a mikrovezérlő és a motor közötti fizikai interfészre, beleértve tesztelés céljából használt trigger jeleket, valamint a szenzorjelek kezelését is.

A második és egyben legfontosabb fázis a vezérlési ciklus, amely minden mintavételkor az ADC megszakítás révén kerül aktiválásra, amelyet egy belső időzítő modul indít és a meghatározott periódusidővel triggerel. A program elindulásakor a rendszer egy előre meghatározott indítási várakozási időt alkalmaz, amely jelen esetben 5 másodperc. Ez a szakasz biztosítja, hogy a hardver stabil állapotba kerüljön és a rendszer kezdeti állapotba álljon. Az ábrán a narancssárga rész jelöli. Ezalatt elvégezzük az úgynevezett nullpont korrekciót. Ennek során előzetesen meghatározzuk a nullponti áram értéket, mivel ezt a későbbiekben kivonjuk az aktuális ADC mintából. Ez az áramérték a rendszer a motor nyugalmi állapotában, több mintavétel átlagolásával kerül meghatározásra, így kompenzáljuk az ADC offszetjét és a mérés konstans hibáit. Ezt a folyamatot elég indításkor elvégezni, a továbbikban a motor már normál üzemmódban működik.

Normál üzemmód esetén az első lépés a motor állapotának beolvasása. Az ADC modul folyamatosan mintavételezi a fázisáramokat. Ezt követően az áramjeleket a már ismertetett koordinátranszformációk segítségével alakítja át a program a rotor referenciarendszerébe.

A rotor helyzetének és sebességének meghatározása a QEP impulzusok feldolgozásán alapul. A program az impulzusokból kiszámítja a rotor mechanikai szögét és szögsebességét. Ezeket már közvetlenül csazolhatjuk a szabályozók bemeneteire. A beolvasás, transzformációk és állapotszámítás minden egyes 200 μ s-os mintavételnél végrehajtódik, a teljes vezérlési ciklus garantáltan a mintavételi periódus alatt zajlik le, ezzel biztosítva a stabil működést. Az 5.9. ábrán látható külső sebességszabályozó egy hagyományos, lineáris PI szabályozó, amelynek valamivel lassabban kell működnie, mint a belső áramszabályozónak. Ezt egy *SpeedLoopCounter* nevű változóval érhetjük el. A számláló értékét minden megszakításban megnöveljük eggyel, és ha elérte az általunk megadott értéket, akkor frissítjük a PI algoritmusunkat és új vezérlőbemenetet adunk a belső kör számára. Továbbá a küszöbérték segítségével szabadon változtathatjük a sebességszabályozó ciklusidejét.

A prediktív irányítás célja, hogy az áramjelek függvényében a szabályozó olyan feszültségeket generáljon, amelyek a legjobban kielégítik a költségfüggvényben megfogalmazott elvárásokat. Ehhez különféle optimalizáló algoritmusokat alkalmazhatunk. minden algoritmus két fő részből áll, amelyek mindegyike ismételten végrhajtódik minden ciklusban. Először a keresési folyamathoz szükséges egyedek generálása történik, majd ezután kezdődik el a tényleges optimalizáció. Az optimalizálás során az 5.1. fejezetben ismertetett költségfüggvény mentén kerestük a legjobb megoldást. Az optimálisnak vélt, megtalált feszültségeket ezek után visszatranszformáljuk a három fázis terébe és a megfelelő módon kiadjuk a PWM vezérlőjeleket. Ez a ciklus ismétlődik mintavételi időnként, amíg le nem állítjuk a programunkat.

Az adatok kimentésének lehetősége elengedhetetlen a megfelelő működés monitorozásához. A szabályozó futtatása és egy másik ezsközzel való kommunikáció azonban valós időben túlságosan letehelné a processzort, ezért célszerű kihasználni a multiprocesszoros működést a párhuzamosan futtatható feladatok szétosztására. A mikrovezérlő soros porton keresztül képes kommunikálni a számítógéppel, amelyen egy Matlab szkript fut, és amely folyamatosan dolgozza fel a beérkező adatokat. A kódban ablakos rögzítést valósítottam meg. Egy változó "igaz" állapotba billentésével lehet elindítani az adatküldést, amely összesen egy másodpercig tart. Egyszerre két float típusú változót küld tovább a második CPU felé, amely aztán továbbküldi onnan. Egy switch-case architektúrába történő megvalósítással lehetővé válik, hogy kettőnél több adatot küldhessünk ki. Azonban fontos megjegyzeni, hogy az egyes minták gyorsan követik egymást, de nem pontosan egy időben kerültek rögzítésre, hasonlóan a szekvenciális folyamatokhoz.

5.5.2 Mérési eredmények és következtetések

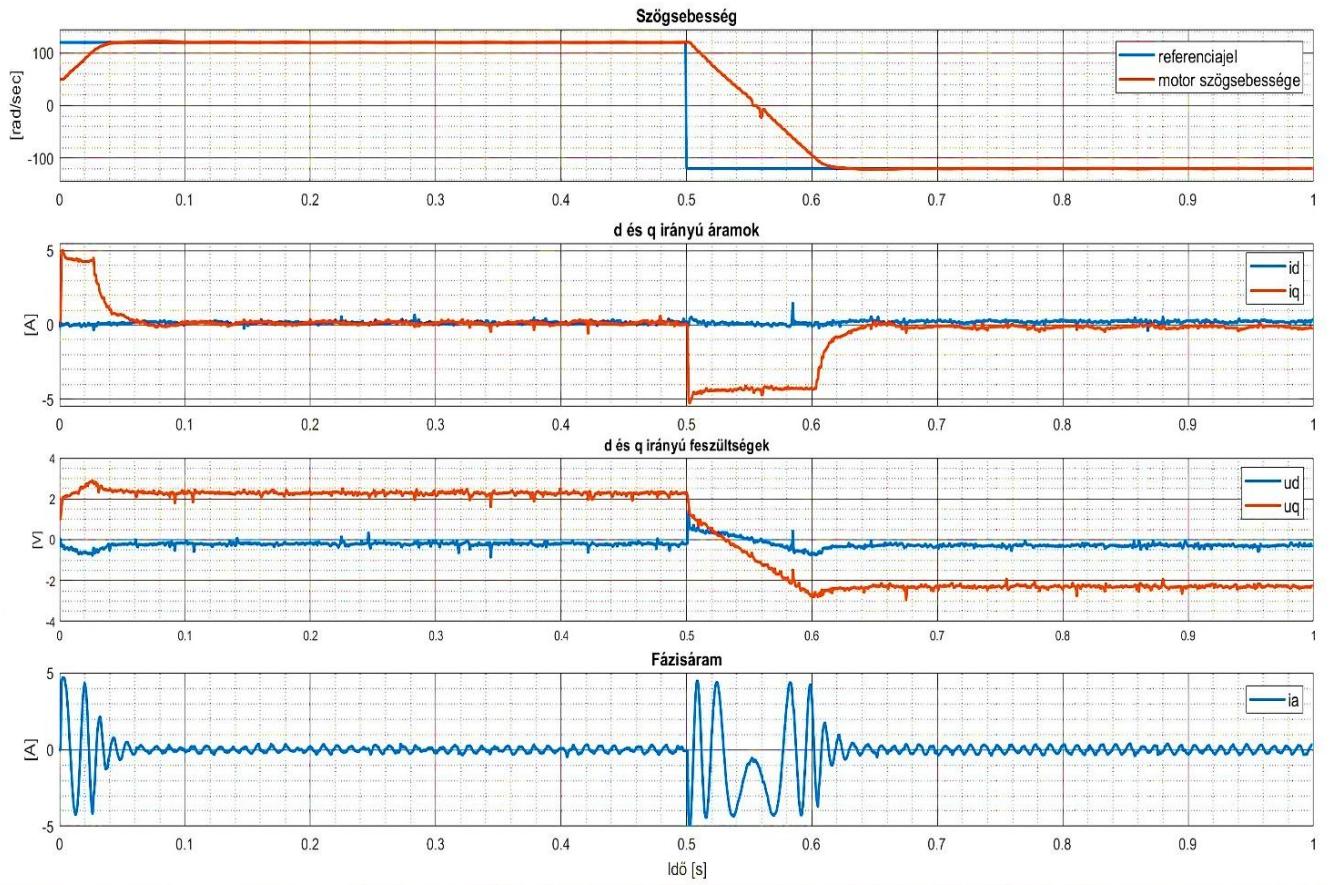
A mérésben felhasznált motor paramétereit megegyeznek az 5.1. táblázat értékeivel. Az 5.10. és 5.11. ábrák szemléltetik a PSO és a GWO algoritmusokkal hangolt modell prediktív szabályozók teljesítményét egy valós hajtásrendszerben végzett vizsgálat esetén. A vizsgálat időtartama 1 másodperc volt, amely során a motor 50 rad/sec alapjáratú szögsebességről indul. A vizsgálat első felében a motor 150 rad/sec referenciaugrás jelet kapott, majd az időablak felénél a referenciajel előjele megfordult, -

150 rad/sec sebességre. A motor a kísérlet alatt végig tehermentesen üzemelt, az esetleges zavarások Jól mutatják szabályozás pontosságát változó referenciajel esetén is. Ezzel párhuzamosan a motoráramok alakulása látható, amelyek szintén fontos visszajelzést adnak a vezérlő teljesítményéről.

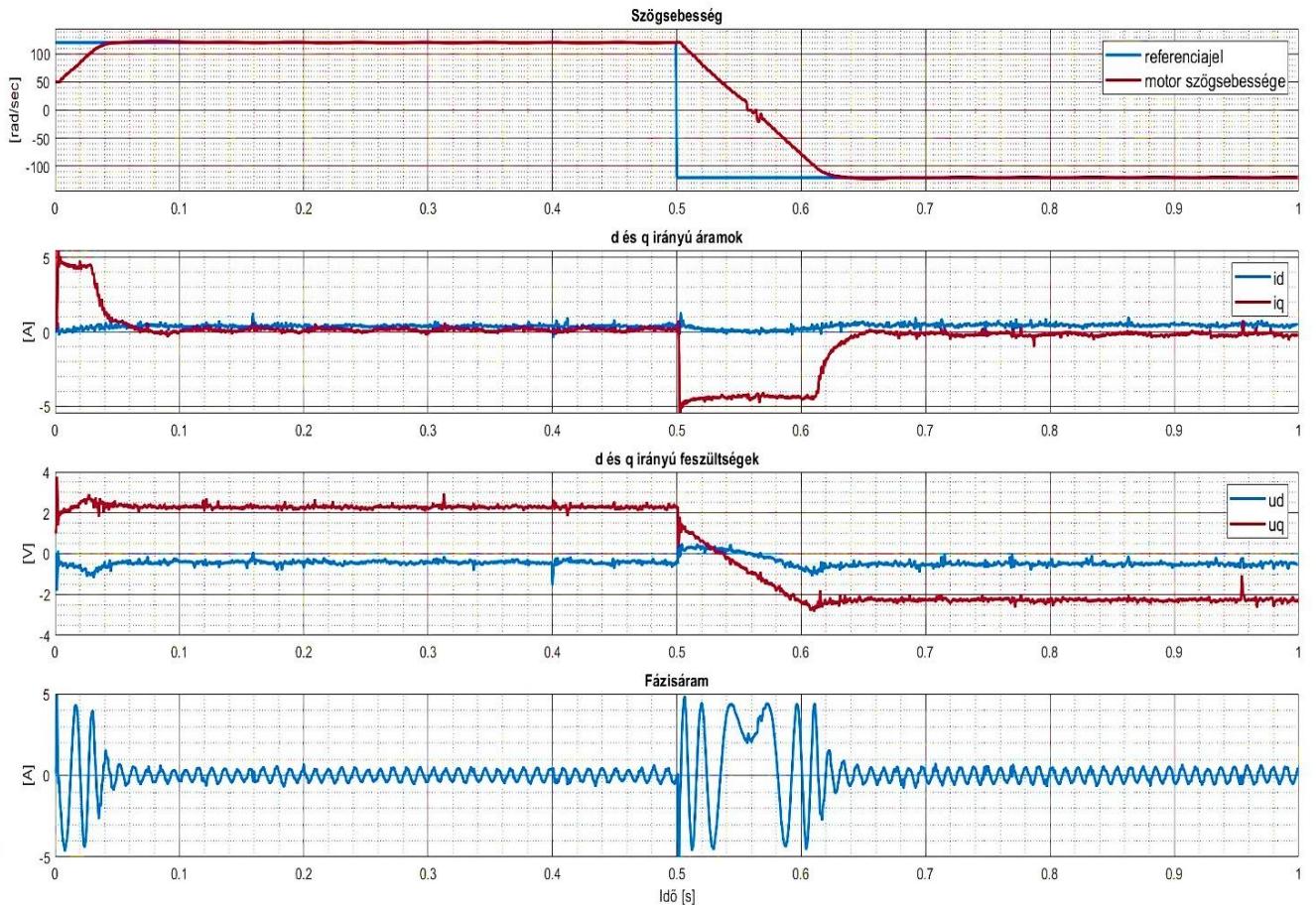
Az ábrák a mechanikai jellemzőkön túl tartalmazzák a d és q irányú áramkomponenseket is, amelyek a mezőorientált vezérlés alapját képezik. Ezek az áramkomponensek közvetlenül befolyásolják a motor nyomatékát és fluxusát. Gyorsítási és fékezési fázisban a q irányú áram ugrásszerűen eléri a szaturációs értéket, hiszen a cél a maximális dinamika kihasználása (természetesen a gyorsítás mértéke felhasználástól függően korlátozható), majd állandósult állapotban nulla csökken, mivel a motor tehermentesen üzemel és a súrlódásból származó fékezőnyomaték elhanyagolható, de a szabályozó a megfelelő referencia áramértékkel ezt is képes kompenzálni. Eközben leolvashatjuk, hogy a d irányú áram végig nulla körüli értéken marad.

A vezérlő által, optimalizációval meghatározott d és q irányú feszültségeket is nyomon követhetjük. Ezek a feszültségek jelzik, hogy az algoritmus milyen mértékben módosítja a motorra kapcsolt feszültséget a kívánt áram- és nyomatékértékek elérése érdekében. Amikor a rendszer a referenciajel változását követően „beáll” állandósult állapotba a q irányú feszültség, a motor forgásirányának megfelelően egy pozitív vagy negatív, nem nulla érték körül stabilizálódik, míg a d irányú komponens nulla. Ez a viselkedés megfelel az állandómágneses szinkrongép szabályozásának elméleti elvárásainak.

Összességében lehetővé teszi a két algoritmus összehasonlítását a motor dinamikus viselkedésének, az áram- és feszültségjelek, valamint a nyomaték szempontjából, bemutatva, hogy minden módszer teljesíti valós időben a kívánt előírásokat.



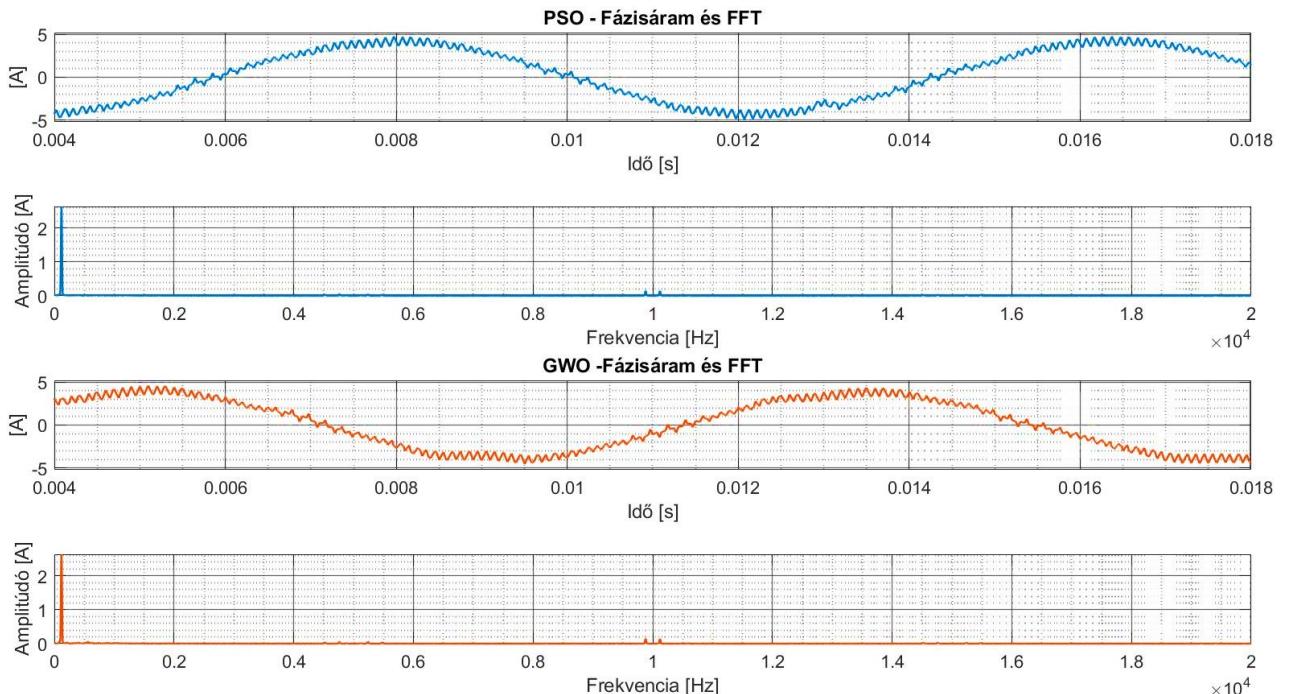
5.10. ábra Mérési eredménye PSO optimalizálással



5.11. ábra Mérési eredménye GWO optimalizálással

A mért áramjelek frekvenciatartománybeli jellemzőinek vizsgálatára az úgynevezett gyors Fourier-transzformáció (FFT) módszerét alkalmaztam. A 5.12. ábra a két módszer esetén mutatja az egyes fázisáramokat és azok spektrumát állandósult állapotban. Az áramok vízszintes tengelyén a mérés során felvett időablak van ábrázolva. A spektrumok részletes vizsgálatához tekintsük a relevánst részleteket felnagyító ábrákat.

A spektrumok két külön ábrán kerültek bemutatásra a jobb átláthatóság érdekében. A mintavételi frekvencia 320 kHz. A 5.13. ábra a 0–1000 Hz tartományt (alacsonyabb frekvenciák), míg a 5.14. ábra az 4000–16000 Hz tartományt (nagyobb frekvenciás komponensek) ábrázolja. Mindkét ábrán egyszerre ábrázoltam a kettő módszer áramjeleinek spektrumát a jobb összehasonlíthatóság végett. A spektrumok amplitúdóját amperben a függőleges tengelyen, a frekvenciát pedig Hertzben, lineáris skálán adtam meg a vízszintes tengelyen.



5.12. ábra Fázisáramok és Fourier transzformáltjuk PSO és GWO esetén

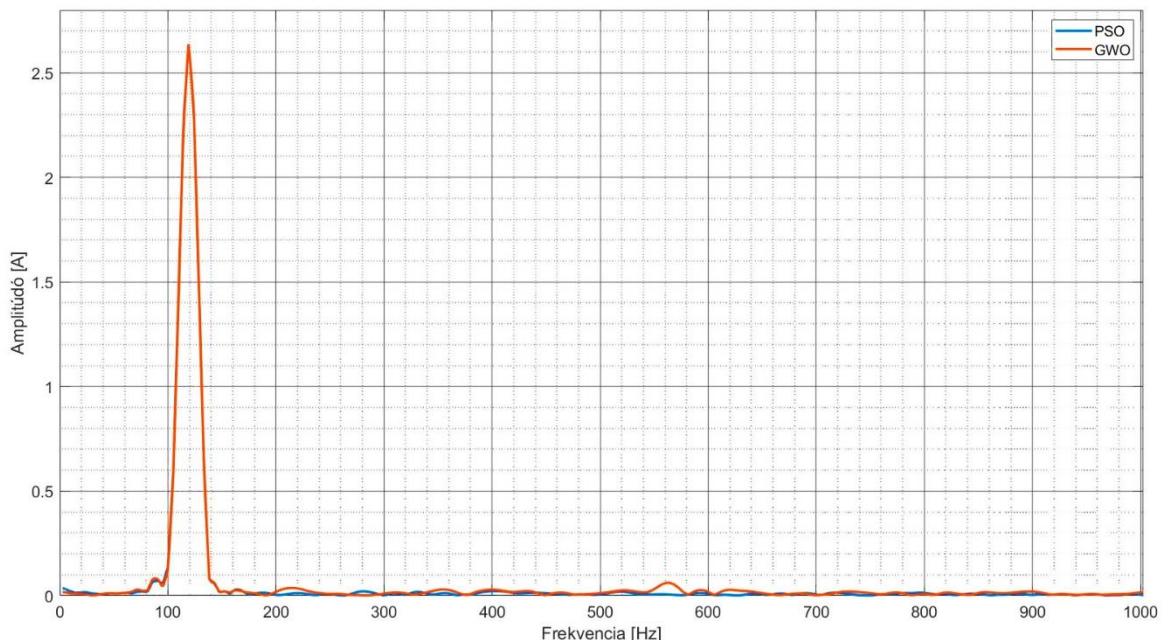
Az 5.13. ábra az 1000 Hz alatti frekvenciatartományban lévő spektrumvonalakat emeli ki. A domináns komponensek minden esetben a 120 Hz körüli sávban koncentrálódnak, amely az alapfrekvenciáknak felel meg 150 rad/s referenciasebesség mellett. A PSO algoritmus alkalmazásával kapott spektrumban az alapfrekvenciának megfelelő komponens 119 Hz körül helyezkedik el 2.6 A-es amplitúdóval, míg ugyanez GWO esetén 120 Hz, szintén 2.6 A-es amplitúdóértékkel. A frekvenciák összhangban

vannak az előző ábrán bemutatott fázisáramokkal, amely az áramgörbék alapján könnyen ellenőrizhetők.

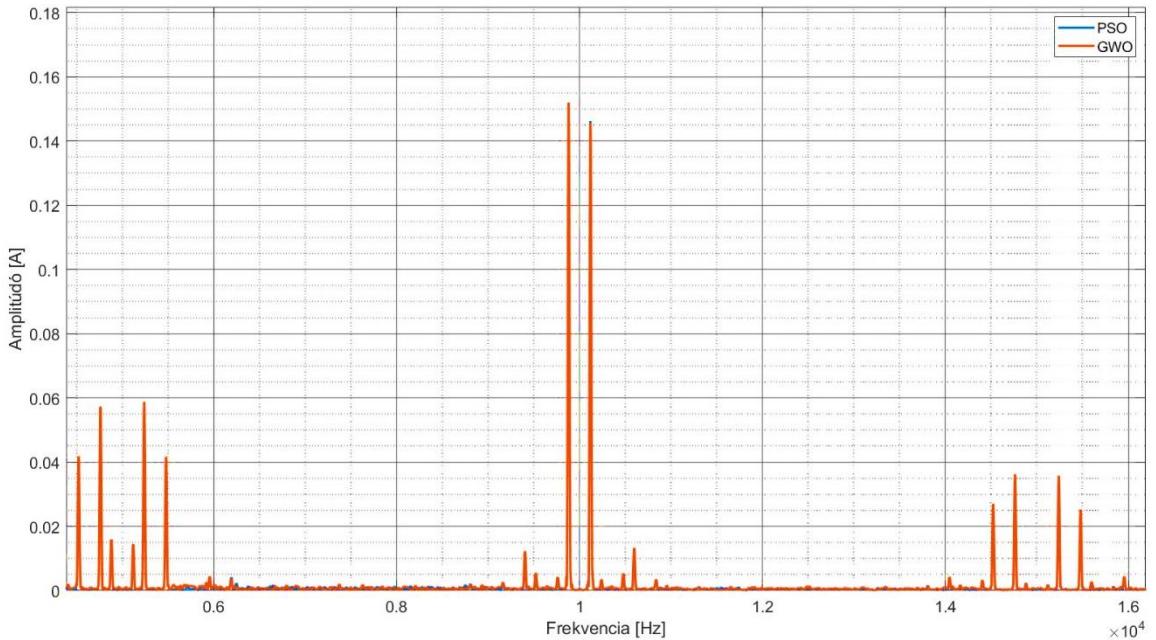
A vizsgálat alapján megfigyelhető, hogy az alacsony frekvenciákon elhelyezkedő spektrumvonalak PSO algoritmus alkalmazása esetén simább jelleget mutatnak, míg a GWO algoritmusnál enyhe hullámzás figyelhető meg a spektrumban. Ez is azt mutatja, hogy a választott optimalizációs algoritmus hatással van a motoráramok frekvenciaspektrumának minőségére.

A magasabb frekvenciatartományban, a 4000 és 16000 Hz közötti sávban szintén észlelhetők spektrumvonalak. Ezen tartományban három kiemelkedő frekvencia jelenik meg. Kisebb amplitúdójú komponensek találhatók körülbelül 5000 Hz és 15000 Hz környékén, ahol az amplitúdó értékei a 0.02–0.06 A közötti tartományban helyezkednek el. Ezzel szemben jelentősen kiemelkednek a 10 kHz körüli frekvenciák, ahol az amplitúdó megközelítőleg 0.15 A. Ezen esetekben a két módszer közel azonos amplitúdójú felharmonikusokat eredményezett.

Összességében az FFT-analízis eredményei azt mutatják, hogy az alkalmazott optimalizációs algoritmus megválasztása befolyásolja az áramok frekvenciaspektrumát, különösen az alacsony frekvenciájú komponensek simaságát és a magasabb frekvenciák kiemelkedését tekintve. Az eredmények alapján a spektrum elemzése alkalmas az algoritmusok hatásának összehasonlítására.



5.13. ábra Alacsony frekvenciájú komponensek



5.14. ábra Nagyfrekvenciás összetevők

A 3.7. ábrára visszatekintve láthatjuk, hogy a vivőfrekvenciás moduláció alkalmazása a felharmonikusokat jól elkülöníthető módon rendezi, jóval alacsonyabb torzítással a frekvenciák között.

A méréseken túl elvégeztem a motor fázisaiban folyó áramok harmonikus torzításának elemzését is. A vizsgálat célja, hogy értékelni tudjuk a szabályozó algoritmusok által generált áramjelek minőségét terheletlen üzemben. A torzítás mértékének kiszámításához az 5.12. ábra áramjeleit vettetem alapul.

A rögzített jel alapján kiszámítottam a teljes harmonikus torzítást (Total Harmonic Distortion, THD) mind a PSO, mind a GWO algoritmussal hangolt szabályozó esetén. A THD a motor áramának harmonikus komponenseit hasonlítja össze az alapfrekvenciával, így lehetővé teszi a zaj és neleineáris torzítások jellemzését. A THD értéke azt mutatja meg, hogy a jel energiájának mekkora része származik az alapfrekvenciától eltérő komponensekből. Minél nagyobb a THD, annál nagyobb a harmonikusok jelenléte, és annál erősebb a jel torzulása. Matlab *thd()* beépített függvényével könnyen kiszámíthatjuk az alapfrekvenciát, a felharmonikusokat, valamint a harmonikus torzítás mértékét decibelben a minták, valamint a mintavételi frekvencia megadásával, ami jelen esetben 320 kHz volt. Összesen 16000 db mintát használtam. A kettő módszerre kapott eredményeket az 5.2. táblázat foglalja magába.

| | PSO | GWO |
|----------------------------|------------|------------|
| Alapharmonikus [Hz] | 118.97 | 120.02 |
| THD [dB] | -28.29 | -23.36 |
| THD [%] | 3.85 | 6.79 |

5.2. táblázat THD értékek PSO és GWO esetén

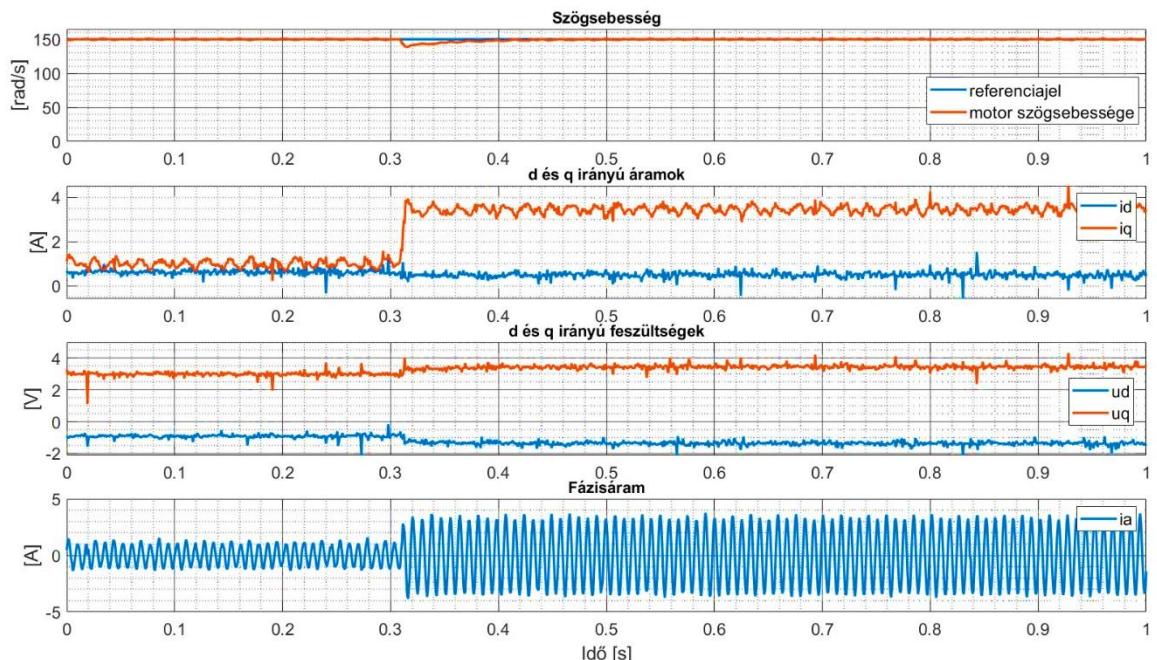
Az eredmények alapján minden két optimalizációs algoritmus esetén a torzítás mértéke relatíve alacsony volt, 3.85 % és 6.79 %-os torzítás, ami arra utal, hogy a szabályozók képesek a motor áramának szinuszos alakját jó közelítéssel fenntartani. Azonban láthatjuk, hogy PSO százalékban kifejezve fele akkora torzítást eredményezett.

Az előző vizsgálat során a motort terheletlen állapotban teszteltem, azonban az algoritmusok dinamikájának tanulmányozásához célszerű a rendszert terhelt üzemállapotban is vizsgálni. Ennek érdekében PMSM motort egy másik, hasonló paraméterekkel rendelkező motorral kapcsoltam össze tengelykapcsolón keresztül. A második motor folyamatos, állandó terhelőnyomatéket biztosított a vizsgált motor számára, így valós körülmények között tanulmányozhatóvá váltak az algoritmusok viselkedései a terhelés hatására.

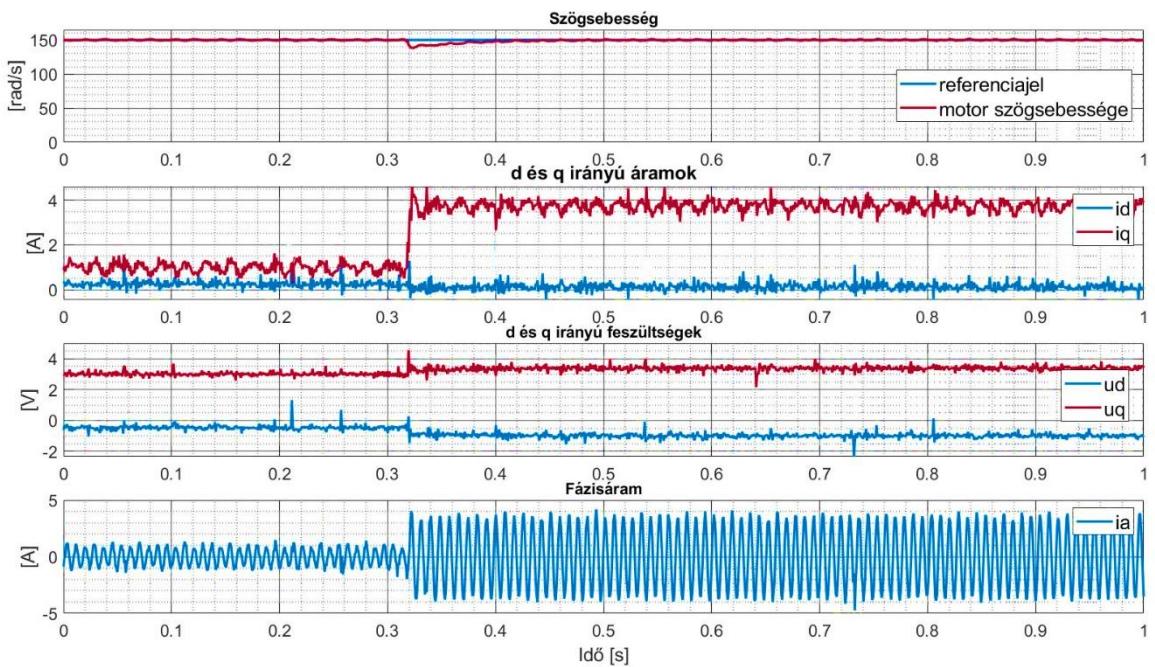
A mérés célja az volt, hogy összehasonlíthatóvá váljon a PSO és a GWO alapú szabályozók teljesítménye, a szögsebesség-, áram- és feszültségváltozások terhelésráadás esetén. Az adatrögzítés kezdetén a motor üresjárásban, terhelés nélkül üzemelt, 150 rad/s állandósult szögsebességgel. A szimulációs idő 1 s-ig tartott, amelynek első harmadában a motor stabilan követte a sebességreferenciát terhelés nélkül. Ezt követően, 0.3 s után a másik motor által terhelést kapcsoltam a rendszerre.

A motor dinamikus viselkedését minden két algoritmus esetén az 5.15. és 5.16. ábrák szemléltetik. A terhelés ráadását követően a szögsebesség rövid időre csökkent, azonban minden két optimalizációs algoritmussal hangolt szabályozó képes volt gyorsan és stabilan visszaállítani a referenciaértéket. A motor áramainak alakulása jól tükrözi a rendszer folyamatait: a terhelés megjelenésével a fázisáramok amplitúdója megnövekedett, amely a motor megnövekedett nyomatékgényének felel meg. A rotor koordinátarendszerében értelmezett áramok közül a q irányú komponens változott ugrásszerűen, amely közvetlenül a nyomaték előállításáért felelős.

A PSO és a GWO alapú szabályozók viselkedése a terhelés hatására nagyjából hasonló. Mindkét esetben a szögsebesség gyorsan és túllövés nélkül tér vissza a referenciaértékre. Az áramjelek alakulása is közel azonos. Ugyanakkor megfigyelhető, hogy a GWO algoritmus esetében a mért áramjeleken valamivel erőteljesebben megjelenik egy magasabb frekvenciájú zajkomponens. Ez a jelenség a szabályozó visszavezethető az algoritmusok paraméterérzékenységére és a GWO algoritmus más keresési dinamikájára. Összességében elmondható, hogy mind a PSO, mind a GWO alapú optimalizációs megfelelően teljesítettek a terhelt vizsgálat során. A rendszer stabilitása minden esetben biztosított volt, a referenciajelre való visszatérés gyorsan történt. A vizsgálat eredményei igazolják, hogy minden módszer alkalmas PMSM motorok optimális irányítására egyaránt.



5.15. ábra Terheléses mérés PSO optimalizálással



5.16. ábra Terheléses mérés GWO optimalizálással

A diplomatervben megvalósított prediktív szabályozás energetikai jellemzőit és veszteségeit hőmérsékletvizsgálattal elemztem. A vizsgálat során a motor állandó, 150 rad/s szögsebességreferenciajel és az imént bemutatott állandó terhelésráadás mellett üzemelt. A cél a rendszer melegedésének megfigyelése volt, különös tekintettel a kapcsolást végző tranzisztorokra és a motor egyes részeire, így a tengelykapcsolóra is. Ezzel egyidőben a lehetővé teszi a PSO és a GWO algoritmusok által optimalizált szabályozó veszteségeinek összehasonlítását.

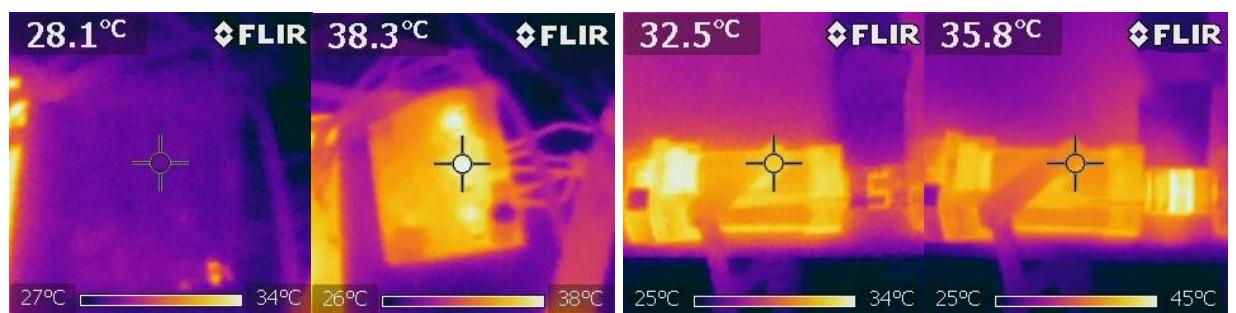
A hőmérsékletmérést a FLIR márkaú, IX szériájú sorozat egyik hőkamerájával végeztem, amely a felvételeket 240x240 pixel felbontású, .jpg formátumban rögzíti, ezzel mellőzve a felvételek esetleges konverzióját és utófeldolgozását.

Első lépésként a rendszer lehűlését követően két hőkamerás felvételt készítettem, dokumentálva a rendszer kiindulási állapotát. Ez az alapállapot szolgált referenciaként. Ezt követően tíz percen keresztül hagytam a motort állandó szögsebességen és terhelés mellett üzemelni. A méréseket egymástól függetlenül hajtottam végre. Az első körben a szabályozó PSO alapú optimalizációt alkalmazott, míg a második méréshez a rendszert hagytam visszahűlni közel azonos hőmérsékletre, mint az alapállapot. Hozzávetőlegesen 32 °C-os hőmérsékleti maximumra hagytam visszahűlni, majd GWO alapú

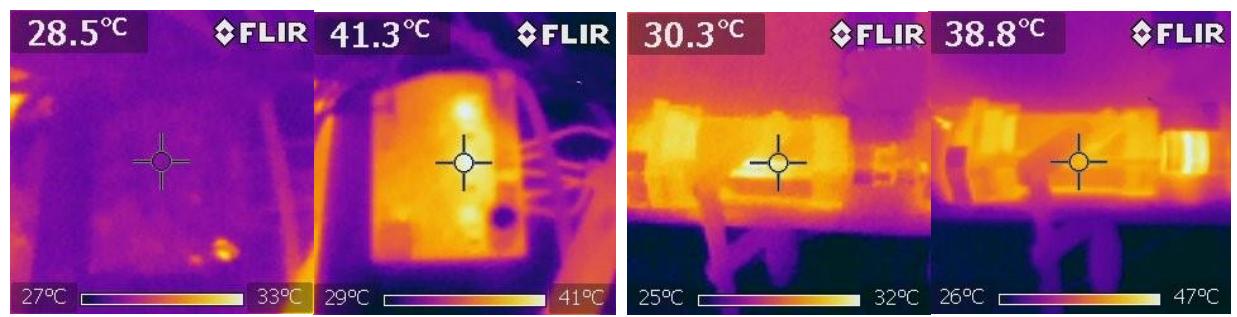
optimalizációt végeztem. Az ismételt tízperces üzemet követően ismét két felvételt készítettem a rendszer aktuális hőmérsékleti állapotáról.

A hőkamera felvételeit az 5.17-5.18. ábrák szemléltetik. A felvételekről leolvasható, hogy a rendszer kezdeti állapotában a tranzisztorok és a motor is egyenletes hőeloszlást mutattak, kiemelkedő pontok nélkül. Mind a motor, mind a tranzisztorok 28,5 °C környéki hőmérsékleten voltak. A tízperces üzem után megfigyelhető a melegedés, továbbá kirajzolódnak a kritikus pontok is. A tranzisztorok világos sárga színnel, jól körülhatárolható módon jelennek meg a felvételen, a legmelegebb pontokon 38,3 °C hőmérsékletet mérve. Ez körülbelül 3 °C-al magasabb, mint az áramkör többi része. A mérési adatok alapján a tranzisztorok hőmérsékletelemelkedése közel 10 °C volt, ami a vezetési és kapcsolási veszteségekkel magyarázható, különösen nagy kapcsolási frekvencia mellett. A mechanikai oldalon a legmelegebb pontot a tengelykapcsolón mértem, közel 45 °C-ot a motor többi részének 35 °C körüli értékéhez képest. A felvételről egyébként jól látható, hogy a motor többi része csupán néhány fokos hőmérsékletelemelést mutat. A motor és a tengelykapcsoló vesztesége részben a súrlódási veszteségből és a kisebb mechanikai rezgések ből, valamint a motor rézveszteségből ered.

GWO alapú optimalizáció esetén a hőmérsékletváltozások hasonló viselkedést mutatnak. Mind a tranzisztorok (41 °C), mind pedig a tengelykapcsoló (47 °C) magasabb hőmérsékletre melegedett, mint PSO esetén, hasonló állapotból indulva. A különbségek mindössze 2–3 °C körül mozognak, amely nem utal jelentős különbségre a veszteségek szempontjából a két algoritmus között. Ebből arra következtethetünk, hogy a PSO és GWO algoritmusok által optimalizált irányítás esetén is rendszer hőmérsékleti viselkedése stabil és elfogadható tartományban marad.



5.17. ábra Hőkamera felvételei PSO optimalizálás esetén. Balra tranzisztorok, jobbra motor hőmérsékleteloszlása



5.18. ábra Hőkamera felvételei GWO optimalizálás esetén. *Balra tranzisztorok, jobbra motor hőmérsékleteloszlása*

6 Konklúzió

A diplomamunka keretében végzett kutatás célja az állandómágneses szinkronmotorok modell prediktív irányításának továbbfejlesztése volt olyan rajaintelligencia alapú algoritmusok felhasználásával, amelyek képesek a problémára megfogalmazott költségfüggvény valós idejű optimalizálására. A téma újszerűsége, hogy nem igazán találtam magyar nyelvű munkát, amely rajaintelligenciát alkalmazott villamos hajtások valós idejű szabályozásban. A rendelkezésre álló hazai publikációk főként konkrét optimalizálási feladatokban vagy hálózati rendszerekben mutatják be ezeket az algoritmusokat, azonban motorirányítási feladatban való alkalmazásuk inkább nemzetközi területen fellelhető. A dolgozat gyakorlati és elméleti szempontból egyaránt foglalkozik ezzel a terüettel, amely hozzájárulhat későbbi fejlesztésekhez.

A vizsgálatok során három algoritmus, a részecskeraj optimalizáció (PSO), a szürkefarkas optimalizáció (GWO) és a mesterséges méhraj algoritmus (ABC) került beépítésre egy PMSM motor modell prediktív szabályozójába, amelyek közül kettőt valós hardveren is megvalósítottam. A módszerek implementációja egy Texas Instruments mikrovezérlőn történt. A megvalósítás magában foglalta a predikciós modell kiszámítását és egy nemlineáris függvény valós idejű optimalizálását.

Az elért eredmények rámutattak arra, hogy a vizsgált módszerek közül a részecskeraj optimalizáció bizonyult a legalkalmasabbnak valós idejű villamos hajtásban. A PSO alacsony harmonikus torzítással és simább áramgörbékkel jellemzhető működést mutatott, emellett pedig a leggyorsabb ciklusidőt is elérte, miközben a motor és az inverter hőterhelése normál tartományban maradt. Ez különösen figyelemre méltó annak fényében, hogy a metaheurisztikus optimalizáló algoritmusok alkalmazását gyakran kérdésessé teszi a valós idejű feldolgozás időkorlátja.

A szürkefarkas-optimalizáció alkalmazásával hasonló eredményeket értünk el néhány kivétellel. A módszer hierarchikus szerkezete és a pozíciófrissítési lépés jelentősen megnövelte a számítási igényt, valamint a kimeneti áramok zájtartalma is enyhén megnőtt.

A terhelésráadási vizsgálatok alapján megállapítható, hogy ezen szabályozók jó dinamikus tulajdonságokkal rendelkeznek. Egy hirtelen fellépő terhelőnyomaték esetén a fordulatszám gyorsan visszaáll a referenciaértékre jelentős túllövés nélkül. Ez a

viselkedés is jól mutatja, hogy a rajintelligencia alapú irányítások nemcsak állandósult állapotban képesek hatékonyan működni.

Végül a hőmérsékleti vizsgálatok következtek. A kettő módszer közel azonos veszteséget okozott az inverterben és a motorban, ami alacsony hőmérsékleti terhelést eredményezett. Ez különösen fontos az elektromos hajtások fejlesztéseiben, ahol a veszteségek minimalizálása elsődleges szempont.

Összegzésként megállapítható, hogy a kutatás bizonyította, hogy megfelelő implementáció mellett a rajintelligencia alapú optimalizációs módszerek alkalmasak valós idejű villamos hajtásirányításra. A valós hardveres implementáció és az összehasonlító mérések eredményei közvetlenül hasznosíthatók további kutatásokban és fejlesztésekben.

7 Irodalomjegyzék

- [1] O. Wallsheid, U. Ammann, J. Böcker, *Real-Time Capable Model Predictive Control Of Permanent Magnet Synchronous Motors Using Particle Swarm Optimisation*, Berlin, Vde Verlag Gmbh, 2016, P. 780.
- [2] B. Lantos, *Irányítási Rendszerek Elmélete És Tervezése II.*, Budapest: Akadémia Kiadó, 2016.
- [3] J. Rodriguez, P. Cortes, *Predictive Control Of Power Converters And Electrical Drives*, John Wiley & Sons, Ltd, 2012.
- [4] S. Chattopadhyay, A. Marik, R. Pramanik, *A Brief Overview Of Physics-Inspired Metaheuristic Optimization Technique*, Arxiv, Január 2022.
- [5] A. G. Gad, *Particle Swarm Optimization Algorithm And Its Applications: A Systematic Review*, Archives Of Computational Methods In Engineering (Springer), P. 2533, 2022.
- [6] A. Dehkordi, A. Gole, T. Maguire, *Permanent Magnet Synchronous Machine Model For Real- Time Simulation*. International Conference On Power Systems, 2005.
- [7] Mohammed Elbes, Shadi Alzubi, Tarek Kanan, Ala Al-Fuqaha, Bilal Hawashin *A Survey on Particle Swarm Optimization with Emphasis on Engineering and Network Applications*, Evolutionary Intelligence, 2019
- [8] Yudong Zhang, Shuihua Wang, Genlin Ji, *A Comprehensive Survey Particle Swarm Optimization Algorithm and Its Applications*, Mathematical Problems in Engineering, 2015
- [9] Diogo Freitas, Luiz Guerreiro Lopes, Fernando Morgado-Dias, *Particle Swarm Optimisation: A Historical Review up to the Current Developments*, Entropy, 2020
- [10] P. Karácsonyi, *Raj Intelligencia Módszerek Hatékonyságának Vizsgálata És Vizualizációja Műszaki Alkalmazásokban*, Budapest: Budapesti Műszaki És Gazdaságtudományi Egyetem, Villamosmérnöki És Informatikai Kar, Irányítástechnika És Informatika Tanszék, 2012.

- [11] E. Wilbur, *Random Number Generation on a TMS320C5x*, APPLICATION BRIEF: SPRA239, Digital Signal Processing Products, Semiconductor Group, Texas Instruments, July 1994.
- [12] D. O. Neacsu, *Space Vector Modulation – An Introduction*, Tutorial at IECON 2001: The 27th Annual Conference of the IEEE Industrial Electronics Society, Satcon Corporation, Cambridge, 2001.
- [13] P. Subramaniyane, *Continuous Control Set – Model Predictive Control of Permanent Magnet Synchronous Motor*, Degree Project in Electrical Engineering, Second Cycle, 30 Credits, KTH Royal Institute of Technology, School of Electrical Engineering and Computer Science, Stockholm, Sweden, 2022.
- [14] S. Chai, L. Wang, E. Rogers, *Model Predictive Control of a Permanent Magnet Synchronous Motor*, School of Electrical and Computer Engineering, RMIT University; School of Electronics and Computer Science, University of Southampton.
- [15] Almufti, S.M., Ahmad, H.B., Marqas, R.B. & Asaad, R.R.. *Grey wolf optimizer: Overview, modifications and applications*. International Research Journal of Science, Technology, Education, and Management, 1, 44-56, 2021.
- [16] K. A. Kana, *Function Optimization Using Swarm Intelligence Algorithms*, Master's Thesis, School of Computing, Computer Science, September 2023.
- [17] Mirjalili, S. M. Mirjalili, A. Lewis, *Grey Wolf Optimizer*, Advances in Engineering Software, vol. 69, pp. 46–61, 2014.
- [18] F. S. Abu-Mouti and M. E. El-Hawary, *Overview of Artificial Bee Colony (ABC) algorithm and its applications*, 2012 IEEE International Systems Conference SysCon 2012, Vancouver, BC, Canada, 2012, pp. 1-6
- [19] Y. Liu, L. Ma and G. Yang, *A Survey of Artificial Bee Colony Algorithm*, 2017 IEEE 7th Annual International Conference on CYBER Technology in Automation, Control, and Intelligent Systems (CYBER), Honolulu, HI, USA, 2017, pp. 1510-1515

- [20] G. Tevesz, A. Kovács, *Robotirányítás rendszertechnikája – Elektronikus jegyzet*. 2. fejezet: *Robotmanipulátorok érzékelői és beavatkozói*. Verzió 2.1, 17. Okt. 2024, BME Automatizálási és Alkalmazott Informatika Tanszék
- [21] P. Stumpf, *Teljesítményelektronika és villamos hajtások – T3 Tantermi gyakorlat*, Budapesti Műszaki és Gazdaságtudományi Egyetem, Automatizálási és Alkalmazott Informatikai Tanszék, BMEVIAUM036
- [22] G. Welch, G. Bishop, *An Introduction to the Kalman Filter*, TR 95-041, Department of Computer Science, University of North Carolina at Chapel Hill, Updated: July 24, 2006.
- [23] S. Morimoto, *Trend of Permanent Magnet Synchronous Machines*, IEEE Transactions on Electrical and Electronic Engineering, vol. 2, pp. 101–108, 2007.
- [24] Vidlak, M.; Makys, P.; Gorel, L. *A Novel Constant Power Factor Loop for Stable V/f Control of PMSM in Comparison against Sensorless FOC with Luenberger-Type Back-EMF Observer Verified by Experiments*. Appl. Sci. 2022, 12, 9179.
- [25] Y. Hou, G. Yu and J. Zou, *Current Measurement Offset Error Compensation Based On Extended State Observer For PMSM Drive Systems*, 2025 IEEE 12th International Symposium on Sensorless Control for Electrical Drives (SLED), Harbin, China, 2025, pp. 1-6

8 Ábrajegyzék

| | |
|---|----|
| 2.1. PMSM motorok típusai kialakításuk szerint..... | 3 |
| 2.2. Neodímium világpiaci árának alakulása az elmúlt 10 évben..... | 4 |
| 2.3. Háromfázisú, csillagkapcsolású PMSM elektromos modellje | 5 |
| 2.4. Különböző koordinátarendszerek orientációja, és áramjelek | 7 |
| 2.5. A PMSM motor helyettesítő képe | 8 |
| 2.6. A PMSM motor hatásvázlata | 10 |
| 2.7. Kétszintű, háromfázisú inverter sematikus ábrája és kiadható feszültségvektorok | 11 |
| 2.8. Kétszintű, háromfázisú inverter kapcsolótáblája | 11 |
| 3.1. Predikciós horizont az időtengely mentén | 16 |
| 3.2. MPC felépítése..... | 17 |
| 3.3. Villamos hajtásban alkalmazott modell prediktív irányítási módszerek | 21 |
| 3.4. PMSM motor MPC alapú szabályozása | 24 |
| 3.5. Szabályozás Simulink környezetben megvalósított szimulációja | 26 |
| 3.6. MPC folyamatábrája | 27 |
| 3.7. Fázisáram spektruma moduláció nélküli esetben | 29 |
| 3.8a. MPC szimulációja: Fázisáram és d és q irányú áramok..... | 30 |
| 3.8b. MPC szimulációja: motor nyomatékgörbái és szögsebességek karakterisztikája | 30 |
| 4.1. Részecskeire ható „erők” vektorábrája | 34 |
| 4.2. PSO algoritmus pszeudókódja | 35 |
| 4.3. Részecskék mozgásának folyamata inicializálás (<i>bal</i>), 8. iteráció (<i>középen</i>), leállás után (<i>jobbra</i>) | 36 |
| 4.4. ABC algoritmus folyamatábrája | 37 |
| 4.5. ABC algoritmus pszeudókódja | 39 |
| 4.6. GWO algoritmus egyedeinek hierarchikus felépítése | 40 |
| 4.7. Adott pozíció frissítése a három legjobb megoldást jelentő egyed alapján..... | 42 |
| 4.8. GWO algoritmus pszeudókódja..... | 43 |
| 4.9. <i>Balról jobbra, soronként:</i> Rastrigin, Matyas és Rosenbrock függvények..... | 44 |
| 4.10. PSO, ABC és GWO algoritmusok teljesítményének összehasonlítása azonos iterációszám mellett | 46 |

| | |
|---|----|
| 5.1. Kezdeti populáció inicializálásának lehetőségei..... | 48 |
| 5.2. Áramkövetés Kálmán-szűrő nélkül | 50 |
| 5.3. MPC irányítás szimulációja MPC optimalizálással..... | 54 |
| 5.4. MPC irányítás szimulációja GWO optimalizálással..... | 55 |
| 5.5. MPC irányítás szimulációja ABC optimalizálással | 56 |
| 5.6. Rajointelligencia-alapú MPC irányítás Simulink modellje | 57 |
| 5.7. CPU1 és CPU2 processzorok közötti feladatmegosztás..... | 58 |
| 5.8. PSO és GWO ciklusidők összehasonlítása | 59 |
| 5.9. Mérési eredménye GWO optimalizálással | 61 |
| 5.10. Mérési eredménye PSO optimalizálással..... | 65 |
| 5.11. Mérési eredménye GWO optimalizálással | 65 |
| 5.12. Fázisáramok és Fourier transzformáltjuk PSO és GWO esetén | 66 |
| 5.13. Alacsony frekvenciájú komponensek | 67 |
| 5.14. Nagyfrekvenciás összetevők..... | 68 |
| 5.15. Terheléses mérés PSO optimalizálással..... | 70 |
| 5.16. Terheléses mérés GWO optimalizálással | 71 |
| 5.17. Hőkamera felvételei PSO optimalizálás esetén. <i>Balra tranzisztorok, jobbra motor hőmérsékleteloszlása</i> | 72 |
| 5.18. Hőkamera felvételei GWO optimalizálás esetén. <i>Balra tranzisztorok, jobbra motor hőmérsékleteloszlása</i> | 73 |