Budapesti Műszaki és Gazdaságtudományi Egyetem
Gépészmérnöki Kar

Műszaki Mechanikai Tanszék

# Kvadkopter Paraméterfüggő Irányítástervezése
# SZAKDOLGOZAT

Készítette: Bezsilla János

Konzulens:
Luspay Tamás Gábor Ph.D.
Tudományos munkatárs

Témavezető:
Dr. Antali Máté
Posztdoktori kutató

Budapest, 2018

**(Az eredeti, lepecsételt feladatkiírási lap helye)**

# Nyilatkozatok

## Beadhatósági nyilatkozat

A jelen szakdolgozat az üzem/intézmény által elvárt szakmai színvonalnak mind tartalmilag, mind formailag megfelel, beadható.

Budapest, 2018. 12. 07.

Az üzem részéről:

*üzemi konzulens*

## Elfogadási nyilatkozat

Ezen szakdolgozat a Budapesti Műszaki és Gazdaságtudományi Egyetem Gépészmérnöki Kara által a Diplomatervezési és Szakdolgozat feladatokra előírt valamennyi tartalmi és formai követelménynek, továbbá a feladatkiírásban előírtaknak maradéktalanul eleget tesz. E szakdolgozatot a nyilvános bírálatra és nyilvános előadásra alkalmasnak tartom.

A beadás időpontja: 2018. 12. 07.

*témavezető*

## Nyilatkozat az önálló munkáról

Alulírott Bezsilla János (LQES96) a Budapesti Műszaki és Gazdaságtudományi Egyetem hallgatója, büntetőjogi és fegyelmi felelősségem tudatában kijelentem és sajátkezű aláírásommal igazolom, hogy ezt a szakdolgozatot meg nem engedett segítség nélkül, saját magam készítettem, és dolgozatomban csak a megadott forrásokat használtam fel. Minden olyan részt, melyet szó szerint vagy azonos értelemben, de átfogalmazva más forrásból átvettem, egyértelműen, a hatályos előírásoknak megfelelően, a forrás megadásával megjelöltem.

Budapest, 2018. 12. 07.

*szigorló hallgató*

# Summary

The main goal of this thesis is to find a control solution to the problems that are posed by the nonlinear dynamics of quadrotors. Based on literary research, the most promising candidates for this are gain scheduled controllers, and LPV (linear parameter-varying) controllers.

First, a system of nonlinear differential equations is developed that models the dynamics of the quadrotor. From this, a linearized state-space representation of the vehicle is built, and PID controllers are tuned for each of the four underlying dynamics (lift, roll, pitch, yaw).

Then, two scheduling variables are chosen ($\phi$ and $\theta$), and the design process begins for both of the advanced controllers that the thesis presents. The main difference between these two lies in their approach to the nonlinear system.

During the gain scheduling design process, the system is linearised at a wide array of trim points, and the controller parameters need to be represented as polynomial surfaces (with regards to the chosen scheduling variables) in order to tune them. The end result is a tuned system of 16 polynomial equations for each controller parameter, and for each of the four dynamical modes.

In comparison, the LPV decoupling method that is presented in this thesis solves this problem with a decoupling filter that reduces the nonlinear system to four double integrators. The parameters of the decoupling filter are dependent on the scheduling variables. The main advantage of this approach is the simplicity of controller tuning, and the rapid calculation time.

Finally, Simulink models are built for both approaches, and for three sets of reference signals, the response of four variables ( $p_z, \phi, \theta, \psi$ ) is analysed. It can then be concluded that while gain scheduling performs slightly better when it comes to keeping the quadrotor at a constant z position, the LPV decoupling method is a much better choice when looking at the graphs of the three angles. The maximum overshoot is usually three to five times bigger with gain scheduling than it is with LPV control.

The direction of future research on the topic points to the development of a MIMO LPV controller with better performance, based on the $H_\infty$ method that is briefly discussed at the end of the thesis.



Keywords: quadrotor, pid control, gain scheduling, lpv decoupling

# Összefoglalás

A szakdolgozat fő célja, hogy megoldást találjon azokra a problémákra, amiket a kvadrotorok dinamikai modelljében megjelenő nemlineáris viselkedés okoz. Az elvégzett irodalomkutatás alapján két lehetséges megközelítés alkalmazható: az egyik a gain scheduling, a másik az LPV szabályozás.

Elsőként levezetésre kerül a kvadrotor dinamikáját leíró nemlineáris differenciálegyenlet rendszer. Ebből egy linearizált állapottér-modell alkotható, aminek a négy alrendszerére (emelés, billentés, bólintás, legyezés) tervezett PID szabályozók működését vizsgáljuk.

Ezután két állapotváltozó ($\phi$ és $\theta$) kiválasztása után megkezdődhet az összetettebb kontrollerek tervezési folyamata. A fő különbség köztük abban rejlik, ahogyan a nemlineáris rendszert kezelik.

A gain scheduling esetében a rendszerről egy széles munkatartományban, több trim pontban is linearizált modell készül, a PID szabályozó paramétereit pedig polinomiális, a korábban megjelölt változóktól függő felületek írják le. Végeredményként 16 egyenlet képezhető, a kontrollerparaméterek és részdinamikák mindegyikére.

Ezzel ellentétben, a szétcsatolt LPV módszer a nemlinearitásból származó problémákat egy olyan szűrő segítségével oldja meg, ami a rendszert szétbontja négy darab kettős integrátorra. A szétcsatoló szűrő együtthatói a korábban választott állapotváltozóktól függnek. A technika fő előnye a szétcsatolás utáni szabályozótervezés egyszerűsége, illetve a gyors számítási idő.

Végül elkészítjük mindkét megközelítés Simulink implementációját, és három trajektóriára vizsgáljuk négy állapotváltozó ( $p_z, \phi, \theta, \psi$ ) válaszait. Ebből arra a következtetésre juthatunk, hogy amíg a gain scheduling jobban teljesít z pozíciótartás szempontjából, addig a szétcsatolt LPV szabályozó sokkal jobb választás a szöghelyzetek kitérése alapján. A túllövés gain scheduling esetében háromszor, vagy akár ötször akkora is lehetett, mint LPV szabályozás esetén.

A lehetséges továbbfejlesztési irány így az LPV szabályozás felé található. A dolgozat végén röviden bemutatásra kerül a $H_\infty$ módszer, amit a későbbiekben kiegészítve jobb eredmények várhatók egy MIMO (több bemenetű, több kimenetű) szabályozó behangolásával.

Kulcsszavak: kvadrotor, pid szabályozás, gain scheduling, lpv szétcsatolás

# Tartalomjegyzék

## Köszönetnyilvánítás

Ezúton szeretném megköszönni konzulensemnek, Luspay Tamás Gábornak, és témavezetőmnek, Dr. Antali Máténak a félév közben nyújtott útmutatásukat és támogatásukat.

A kutatást az "Ipar 4.0 kutatási és innovációs kiválósági központ" című GINOP-2.3.2-15-2016-00002 támogatás tette lehetővé.

*Budapest, 2018. december*

*B.J.*

# Jelölések jegyzéke

A táblázatban a többször előforduló jelölések magyar és angol nyelvű elnevezése, valamint a fizikai mennyiségek esetén annak mértékegysége található. Az egyes mennyiségek jelölése - ahol lehetséges - megegyezik hazai és a nemzetközi szakirodalomban elfogadott jelölésekkel. A ritkán alkalmazott jelölések magyarázata első előfordulási helyüknél található.

## Latin betűk

| Jelölés | Megnevezés, megjegyzés, érték | Mértékegység |
|---|---|---|
| $\mathbf{A}$ | állapottér-modell A mátrixa | 1 |
| $b$ | légellenállási tényező $(1,14 \cdot 10^{-6})$ | N m $s^2$ |
| $\mathbf{B}$ | állapottér-modell B mátrixa | 1 |
| $\mathbf{C}$ | állapottér-modell C mátrixa | 1 |
| $\mathbf{D}$ | állapottér-modell D mátrixa | 1 |
| $\delta\mathbf{u}$ | trim-be tolt bemeneti jelvektor | 1 |
| $\delta\mathbf{x}$ | trim-be tolt állapot jelvektor | 1 |
| $\delta\mathbf{y}$ | trim-be tolt kimeneti jelvektor | 1 |
| $f$ | erő | N |
| $f_{max}$ | maximális rotoronkénti felhajtóerő (30) | N |
| $g$ | nehézségi gyorsulás $(9,81)$ | m $s^{-}2$ |
| $G(s)$ | átviteli függvény | 1 |
| $\mathbf{h}$ | perdületvektor | N m s |
| $\mathbf{I}_{nxn}$ | nxn méretű egységmátrix | 1 |
| $\mathbf{J}$ | tehetetlenségi nyomatéki mátrix | kg $m^2$ |
| $J_x$ | 1. fő tehetetlenségi nyomaték $(4,856 \cdot 10^{-3})$ | kg $m^2$ |
| $J_y$ | 2. fő tehetetlenségi nyomaték $(4,856 \cdot 10^{-3})$ | kg $m^2$ |
| $J_z$ | 3. fő tehetetlenségi nyomaték $(8,801 \cdot 10^{-3})$ | kg $m^2$ |
| $k$ | motorparaméter $(2,98 \cdot 10^{-5})$ | N $s^2$ |
| $K_d$ | D tag együtthatója | 1 |
| $K_i$ | I tag együtthatója | 1 |
| $K_p$ | P tag együtthatója | 1 |
| $l$ | rotorkar hossza (0,275) | m |
| $m$ | kvadrotor tömege (0,468) | kg |
| $\mathbf{0}_{nxn}$ | nxn méretű zérus mátrix | 1 |
| $p$ | 1. főiránybeli szögsebesség | rad $s^{-1}$ |
| $p_x$ | x pozíció | m |
| $p_y$ | y pozíció | m |
| $p_z$ | z pozíció | m |
| $q$ | 2. főiránybeli szögsebesség | rad $s^{-1}$ |

## Latin betűk

| Jelölés | Megnevezés, megjegyzés, érték | Mértékegység |
|---|---|---|
| $r$ | 3. főiránybeli szögsebesség | rad $s^{-1}$ |
| $\mathbf{R}$ | Euler-féle forgatómátrix | 1 |
| $s$ | Laplace-operátor | 1 |
| $T$ | "thrust" - felfelé ható erő | N |
| $t$ | idő | s |
| $\mathbf{T}(s)$ | szétcsatoló szűrő átv. fv. mátrixa | 1 |
| $T_0$ | trim állapotú thrust | N |
| $T_f$ | deriváló szűrő időállandója | s |
| $\mathbf{u}$ | bemenő jelvektor | 1 |
| $\mathbf{u}_0$ | trim bemenő jelvektor | 1 |
| $v_x$ | x irányú sebesség | m $s^{-1}$ |
| $v_y$ | y irányú sebesség | m $s^{-1}$ |
| $v_z$ | z irányú sebesség | m $s^{-1}$ |
| $\mathbf{x}$ | állapot jelvektor | 1 |
| $\mathbf{x}_0$ | trim állapot jelvektor | 1 |
| $\mathbf{y}$ | kimeneti jelvektor | 1 |
| $\mathbf{y}_0$ | trim kimeneti jelvektor | 1 |

## Görög betűk

| Jelölés | Megnevezés, megjegyzés, érték | Mértékegység |
|---|---|---|
| $\theta$ | bólintási szög | rad |
| $\tau$ | járműre ható nyomatékok vektora | N m |
| $\tau_\theta$ | bólintó nyomaték | N m |
| $\tau_\phi$ | billentő nyomaték | N m |
| $\tau_\psi$ | legyező nyomaték | N m |
| $\phi$ | billentési szög | rad |
| $\psi$ | legyezési szög | rad |
| $\omega$ | jármű szögsebességvektora | rad s$^-$1 |
| $\omega$ | rotor szögsebessége | rad s$^-$1 |
| $\omega_c$ | vágási körfrekvencia | rad s$^-$1 |

## Indexek, kitevők

| Jelölés | Megnevezés, megjegyzés, érték |
|---|---|
| $i$ | általános futóindex |
| max | maximum |
| ref, r | referencia |

# 1. fejezet

# Bevezetés

A forgószárnyas légi járművek szabályozása több évtizedes múltra tekint vissza. A dinamika nemlinearitása ugyanis nem teszi lehetővé, hogy az LTI rendszereknél megszokott egyszerű PID kontrollert alkalmazzunk. A probléma megoldására több javaslat is született, az átfogó irodalomkutatás alapján két technika ismertetése mellett döntöttem, ezek a gain scheduled, illetve az LPV szabályozás.

A második fejezet a differenciálegyenlet rendszer levezetését mutatja be, egy szakterületen általánosan használt eljárást alapul véve.

A harmadik fejezet ennek a modellnek a Matlab, illetve Simulink környezetben történő implementálásával foglalkozik, és bemutatja a szimulációhoz szükséges blokkokat. Ezt követően az alapképzés során megismert PID szabályozó tervezését végzem el a negyedik fejezetben, a Matlab beépített függvényeit segítségül hívva. A zárt kör válaszának vizsgálata során bemutatásra kerülnek a korlátai és stabilitási problémái a lineáris modellnek.

Az ötödik fejezet a gain scheduled szabályozás alapjainak ismertetésével, illetve a szabályozó megtervezésével és behangolásával foglalkozik. Az eljárás lényege a munkaterület kibővítése, bizonyos állapotváltozók kijelölésével és azokból egy rács felépítésével. Ezt követi a szabályozó együtthatóinak állapotváltozóktól való analitikus függésének meghatározása. A szimulációs eredmények bemutatásra kerülnek három trajektóriára a fejezet végén.

A hatodik fejezet egy Thomas T.R. van de Wiel, Tóth Roland, és Vsevolod I. Kiriouchine által kifejlesztett szétcsatoláson alapuló LPV (lineáris, változó paraméterű) szabályozási módszert ismertet, illetve annak Simulink implementációjával és az eredmények bemutatásával foglalkozik. A módszer lényege egy előre kijelölt állapotváltozóktól függő szétcsatoló szűrő megtervezése, ami kettős integrátorokra egyszerűsíti a modell egyes dinamikáit. Ezek már PID szabályozóval kezelhető, lineáris rendszerek.

A hetedik fejezet bemutatja a lehetséges továbbfejlesztési utat, egy másik szabályozási eljárás rövid ismertetésével. A $H_\infty$ módszer előnye, hogy MIMO szabályozóként képes a rendszer egészére egyszerre hatást gyakorolni. Ezen túl kisebb kiegészítésekkel LPV modell képezhető belőle, amivel a már LTI rendszerekre ismert irányítástechnikai eszköztár kiterjeszthetővé válik nemlineáris dinamikákra is.

A dolgozat a gain scheduled, illetve a szétcsatoláson alapuló LPV szabályozók válaszainak összevetésével zárul, a nyolcadik fejezetben.

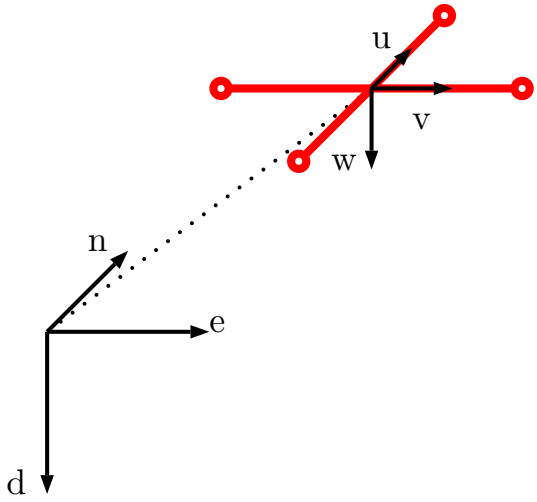
# 2. fejezet

# A dinamikai modell

A kvadrotor modellezéséhez az első lépés a megfelelő egyenletrendszer felírása. Az alkalmazott koordináta-rendszertől, illetve egyszerűsítésektől függően ez a feladat több irányból is megközelíthető. Az elvégzett irodalomkutatás alapján két módszert érdemes megemlíteni, amiket egymástól a sebességvektorok bázisai különböztetnek meg.

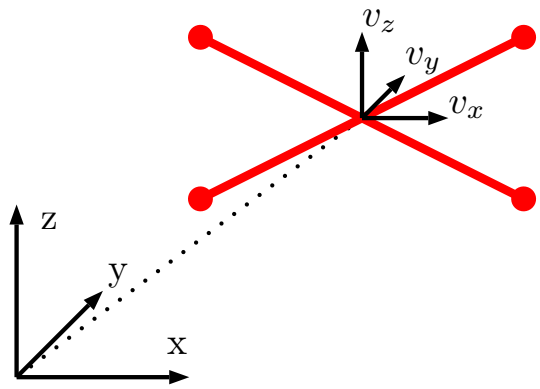
## 2.1. Modellezési eljárások a szakirodalomban

Az egyik, repüléstechnikában széles körben használt eljárás [1] a sebességeket egy testhez rögzített koordináta-rendszerben értelmezi (*2.1. ábra*). A jármű pozíciója NED (North East Down) inerciarendszerben van értelmezve, a pozíciók deriváltjai pedig forgatómátrixon keresztül vannak kapcsolatban a sebességvektorral. A leírási mód előnye, hogy a koordináta-tengelyek mentén a geometria változatlansága miatt könnyedén értelmezhetők a járműre ható aerodinamikai erők és nyomatékok. Hátránya viszont a forgó koordinátarendszer miatt a sebességvektor deriváltjában megjelenő Coriolis-gyorsulás. Ez jelentősen befolyásolja a rendszer szétcsatolhatóságát LPV szabályozásnál, és kényelmetlenné teszi a modell használatát.




2.1. ábra. NED pozíciók, testrendszerben értelmezett sebességek

A másik leírási módot [2] a repüléstechnikai szakirodalom jellemzően csak kvadrotorokkal foglalkozó írásokban használja. A pozíciók egy jobbsodrású xyz koordinátarendszerben vannak definiálva (*2.2. ábra*). A sebességek ugyanebben a bázisban adottak, így nincs szükség a deriválás során a jármű forgását figyelembe venni. A módszer nagy előnyeként ez már lehetővé teszi a modell szétcsatolását. A leírás hátránya, hogy nem értelmezhetők benne könnyedén az aerodinamikai hatások. Mivel ezek a kvadkopterek geometriai és teljesítménybeli jellemzői alapján gyengék, ezért a modellezés során elhanyagolhatóak.




2.2. ábra. xyz pozíciók, inerciálisan értelmezett sebességek

## 2.2. A modellezendő rendszer

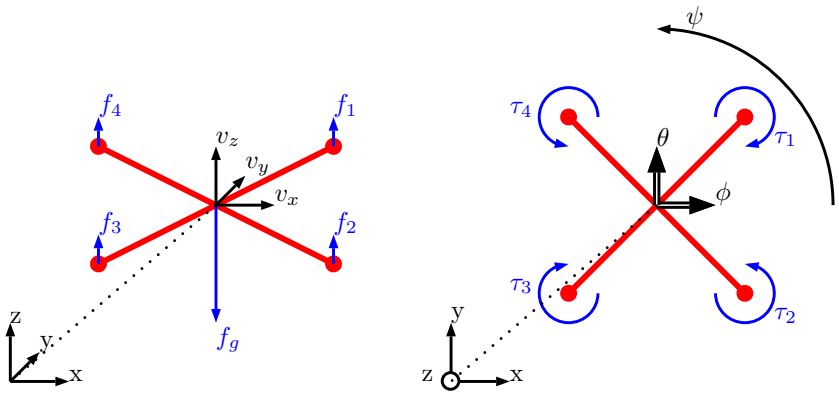
A leírás előnyei miatt a dinamikai modell megalkotásakor 2.2. ábrán látható rendszert célszerű alapul venni.

A kvadrotor állapotát négy bemenettel tudjuk befolyásolni, ezek négy rotor különkülön vezérelhető szögsebességei. A szögsebességekkel ($\omega$) arányosan (k motorparaméter az arányossági tényező) a propellerek forgása felhajtóerőt (f) kelt, a forgást gátló légellenállás (tényezője b) pedig forgással ellentétes nyomatékot ($\tau$).

$$f = k \cdot \omega^2 \tag{2.1}$$

$$\tau = b \cdot \omega^2 + J_M \cdot \dot{\omega} \tag{2.2}$$

Emiatt szükségesen párban ellentétes irányú forgást végeznek a propellerek (*2.3. ábra*), hogy a nyomatékok kiejtsék egymást. $J_M$ a motor tehetetlenségi nyomatéka, de mivel a nyomatékra való hatása $\dot{\omega}$-nek csak felszálláskor számottevő (egyébként az egyes rotorok fordulatszáma csak csekély mértékben változik repülés közben), így az egyenletekből elhanyagolható [3]. A 2.3. ábrán feltüntetett jelölés alapján látható, hogy a rotorok forgásából származó erők és nyomatékok összegezhetők, így tengelyirányú eredő erő, illetve nyomatékok kaphatók.




2.3. ábra. Erők és nyomatékok a kvadkopteren

$$\mathbf{f} = \sum_{i=1}^{4} \begin{bmatrix} 0 \\ 0 \\ f_i \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ T \end{bmatrix} \tag{2.3}$$

$$\tau_\phi = (f_1 + f_4 - f_2 - f_3) \cdot \frac{l}{\sqrt{2}} \tag{2.4}$$

$$\tau_\theta = (f_3 + f_4 - f_1 - f_2) \cdot \frac{l}{\sqrt{2}} \tag{2.5}$$

$$\tau_\psi = (f_2 + f_4 - f_1 - f_3) \cdot \frac{b}{k} \tag{2.6}$$

Az egyenletekben szereplő paraméterek közül l a kvadrotor karjainak hossza, b ellenállástényező, k pedig motorparaméter. A nyomatékok indexeiben $\phi$, $\theta$ és $\psi$ jelölik azokat a szögeket, amikre hatással vannak. Ezek Euler-szögek, később kerülnek kifejtésre.

## 2.3. Az egyenletek levezetése

A kvadrotor dinamika modelljének megalkotásához fel kell írni a mozgását leíró differenciálegyenleteket. A 6 szabadsági fokból hogy elsőrendű differenciálegyenlet rendszer jöjjön létre, 12 egyenletre van szükség. Ebből az első három az x-, y-, és z pozíciók $(p_x, p_y, p_z)$ deriváltjaira vonatkozik. A választott koordináta-rendszer (2.2. ábra) miatt a jármű pozíciójára vonatkozók egyszerűen felírhatók.

$$\begin{bmatrix} \dot{p}_x \\ \dot{p}_y \\ \dot{p}_z \end{bmatrix} = \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} \tag{2.7}$$

A sebességek deriváltjainak kiszámítása Newton 2. törvényének alkalmazásával tehető meg.

$$\mathbf{f} = m\dot{\mathbf{v}} \tag{2.8}$$

Az egyenlet átrendezhető, a vektorok elemei azonban további figyelmet igényelnek. Az erővektor két erőnek az összegéből származtatható: a gravitációs erőből, illetve a propellerek forgásából adódó felhajtóerők eredőjéből. Míg az előbbi az általunk választott koordináta-rendszerben -z irányba mutat, addig a T felhajtóerő függ a kvadrotor szöghelyzetétől.

$$\begin{bmatrix} \dot{v_x} \\ \dot{v_y} \\ \dot{v_z} \end{bmatrix} = \frac{1}{m} \begin{bmatrix} f_x \\ f_y \\ f_z \end{bmatrix} = \frac{1}{m} \left( \mathbf{R}^T \begin{bmatrix} 0 \\ 0 \\ T \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix} \right) \tag{2.9}$$

R egy olyan mátrix, ami egy inerciális koordináta-rendszerben értelmezett vektort testrendszerbe forgat. Elemei függnek a $\phi$, $\theta$, illetve $\psi$ Euler-szögektől. A számítása a következőképpen tehető meg:

$$\boldsymbol{R} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{bmatrix} \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix} \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.10}$$

Összeszorozva a három mátrixot, illetve az érhetőség céljából $\sin\phi = s_\phi$ és $\cos\phi = c_\phi$ egyszerűsítéseket megtéve a szögekre, a következő forgatómátrix áll elő:

$$\boldsymbol{R} = \begin{bmatrix} c_\theta c_\psi & c_\theta c_\psi & -s_\theta \\ s_\phi s_\theta c_\psi - c_\phi s_\psi & s_\phi s_\theta s_\psi + c_\phi c_\psi & s_\phi c_\theta \\ c_\phi s_\theta c_\psi + s_\phi s_\psi & c_\phi s_\theta s_\psi - s_\phi c_\psi & c_\phi c_\theta \end{bmatrix} \tag{2.11}$$

A mátrixot a (2.9) egyenletbe helyettesítve, illetve a szükséges szorzásokat elvégezve adódik a sebességek deriváltjaira vonatkozó differenciálegyenlet.

$$\begin{bmatrix} \dot{v_x} \\ \dot{v_y} \\ \dot{v_z} \end{bmatrix} = \frac{1}{m} \begin{bmatrix} c_\theta c_\psi & s_\phi s_\theta c_\psi - c_\phi s_\psi & c_\phi s_\theta c_\psi + s_\phi s_\psi \\ c_\theta s_\psi & s_\phi s_\theta s_\psi + c_\phi c_\psi & c_\phi s_\theta s_\psi - s_\phi c_\psi \\ -s_\theta & s_\phi c_\theta & c_\phi c_\theta \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ T \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} \tag{2.12}$$

$$\begin{bmatrix} \dot{v}_x \\ \dot{v}_y \\ \dot{v}_z \end{bmatrix} = \begin{bmatrix} (\sin\phi \cdot \sin\psi + \cos\phi \cdot \cos\psi \cdot \sin\theta) \\ (\cos\phi \cdot \sin\psi \cdot \sin\theta - \cos\psi \cdot \sin\phi) \\ \cos\phi \cdot \cos\theta \end{bmatrix} \cdot \frac{T}{m} - \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} \tag{2.13}$$

A jármű szögsebességei (p, q, r) kiszámíthatók az Euler-szögek deriváltjaival, a megfelelő forgatómátrixok alkalmazása mellett [1]:

$$\begin{aligned} \begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} \dot{\phi} \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{bmatrix} \begin{bmatrix} 0 \\ \dot{\theta} \\ 0 \end{bmatrix} \\ + \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{bmatrix} \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix} \end{aligned} \tag{2.14}$$

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} 1 & 0 & -\sin\theta \\ 0 & \cos\phi & \sin\phi\cos\theta \\ 0 & -\sin\phi & \cos\phi\cos\theta \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \tag{2.15}$$

A szögsebességekre vonatkozó egyenletrendszerből látható, hogy a mátrix invertálásával megkaphatóak közvetlenül a szöghelyzetek deriváltjai. A művelet Matlabban paraméteresen is elvégezhető. A mátrix elemeiből látható, hogy ha $\cos\theta$ értéke megközelíti a nullát, a modell viselkedése kiszámíthatatlanná válik. Kvadkopterek működés közben ritkán repülnek olyan szöghelyzetben, ami a veszélyes tartományba esne, így a továbbiakban nem foglalkozunk a modell ezen korlátjával.

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & \sin\phi\,\mathrm{tg}\,\theta & \cos\phi\,\mathrm{tg}\,\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \frac{\sin\phi}{\cos\theta} & \frac{\cos\phi}{\cos\theta} \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{2.16}$$

Szögsebességek deriváltjainak meghatározásához felírható Newton II. törvénye [1]. Ez alapján kifejtve az egyenletet:

$$\boldsymbol{\tau} = \frac{d\mathbf{h}}{dt} + \boldsymbol{\omega} \times \mathbf{h} = \mathbf{J}\frac{d\boldsymbol{\omega}}{dt} + \boldsymbol{\omega} \times (\mathbf{J}\boldsymbol{\omega}) \tag{2.17}$$

Az egyenletben szereplő $\mathbf{h}$ az impulzusvektor, $\mathbf{J}$ a tehetetlenségi nyomatéki mátrix, amelynek elemei a kvadrotor szimmetriája miatt kizárólag a főátlóban helyezkednek el:

$$\mathbf{J} = \begin{bmatrix} J_x & 0 & 0 \\ 0 & J_y & 0 \\ 0 & 0 & J_z \end{bmatrix} \tag{2.18}$$

A szögsebesség-deriváltakat megadó egyenletrendszer átrendezéssel, és a tagok kifejtésével kapható meg. A diagonális tehetetlenségi nyomatéki mátrix inverze kézzel is kiszámítható, így meggyorsítva a numerikus számítást:

$$\dot{\boldsymbol{\omega}} = \mathbf{J}^{-1} \left( -\boldsymbol{\omega} \times (\mathbf{J}\boldsymbol{\omega}) + \boldsymbol{\tau} \right) \tag{2.19}$$

$$\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} \frac{1}{J_x} & 0 & 0 \\ 0 & \frac{1}{J_y} & 0 \\ 0 & 0 & \frac{1}{J_z} \end{bmatrix} \left( \begin{bmatrix} p \\ q \\ r \end{bmatrix} \times \begin{bmatrix} \frac{1}{J_x} & 0 & 0 \\ 0 & \frac{1}{J_y} & 0 \\ 0 & 0 & \frac{1}{J_z} \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} + \begin{bmatrix} \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} \right) \tag{2.20}$$

$$\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} \frac{J_y - J_z}{J_x} \cdot q \cdot r \\ \frac{J_z - J_x}{J_y} \cdot p \cdot r \\ \frac{J_x - J_y}{J_z} \cdot p \cdot q \end{bmatrix} + \begin{bmatrix} \frac{\tau_\phi}{J_x} \\ \frac{\tau_\theta}{J_y} \\ \frac{\tau_\psi}{J_z} \end{bmatrix} \tag{2.21}$$

## 2.4. A dinamikai egyenletrendszer

Összegzésképpen, az előző alfejezetben az állapotváltozók deriváltjaira levezetett 12 egyenlet a következő egyenletrendszert alkotja:

$$\begin{bmatrix} \dot{p}_x \\ \dot{p}_y \\ \dot{p}_z \end{bmatrix} = \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} \tag{2.22}$$

$$\begin{bmatrix} \dot{v_x} \\ \dot{v_y} \\ \dot{v_z} \end{bmatrix} = \begin{bmatrix} (\sin\phi \cdot \sin\psi + \cos\phi \cdot \cos\psi \cdot \sin\theta) \\ (\cos\phi \cdot \sin\psi \cdot \sin\theta - \cos\psi \cdot \sin\phi) \\ \cos\phi \cdot \cos\theta \end{bmatrix} \cdot \frac{T}{m} - \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} \tag{2.23}$$

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & \sin\phi\,\mathrm{tg}\,\theta & \cos\phi\,\mathrm{tg}\,\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \frac{\sin\phi}{\cos\theta} & \frac{\cos\phi}{\cos\theta} \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{2.24}$$

$$\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} \frac{J_y - J_z}{J_x} \cdot q \cdot r \\ \frac{J_z - J_x}{J_y} \cdot p \cdot r \\ \frac{J_x - J_y}{J_z} \cdot p \cdot q \end{bmatrix} + \begin{bmatrix} \frac{\tau_\phi}{J_x} \\ \frac{\tau_\theta}{J_y} \\ \frac{\tau_\psi}{J_z} \end{bmatrix} \tag{2.25}$$

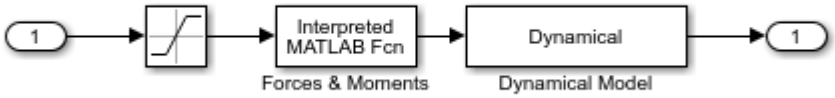
Ezek segítségével Matlabban már szimulálható a modell. Ha bemenő jelnek nem az erőket és nyomatékokat, hanem a rotorok által keltett felhajtóerőket tekintjük, akkor a következő mátrixra van szükség a bemeneti konverzióhoz:

$$\begin{bmatrix} T \\ \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ \frac{l}{\sqrt{2}} & -\frac{l}{\sqrt{2}} & -\frac{l}{\sqrt{2}} & \frac{l}{\sqrt{2}} \\ -\frac{l}{\sqrt{2}} & -\frac{l}{\sqrt{2}} & \frac{l}{\sqrt{2}} & \frac{l}{\sqrt{2}} \\ -\frac{b}{k} & \frac{b}{k} & -\frac{b}{k} & \frac{b}{k} \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \end{bmatrix} \tag{2.26}$$

# 3. fejezet

# Matlab implementáció

A kvadrotor szimulációja a Matlab R2018a verziójával készült.




3.1. ábra. A szimulációra használt Simulink-modell

Három blokk használatával modellezhető Simulink környezetben a jármű (*3.1. ábra*). A szimuláció alapját a korábban levezetett egyenletrendszert tartalmazó dinamikai modell képzi. A Matlab lehetőséget biztosít arra, hogy a differenciálegyenleteket diszkrét időben kezelve követhessük az állapotváltozók értékeit tetszőleges bemenő jelre, az S-Function blokk segítségével. Ezen kívül szükség van egy `Interpreted Matlab Function` blokkra a bemenetek átalakítására, illetve az aktuátordinamika pontosabb modellezése érdekében egy telítődésre. A kvadrotor paraméterei és állapotváltozóinak kezdeti értékei a szimuláció futtatása előtt, kívülről tölthetők be a programba.

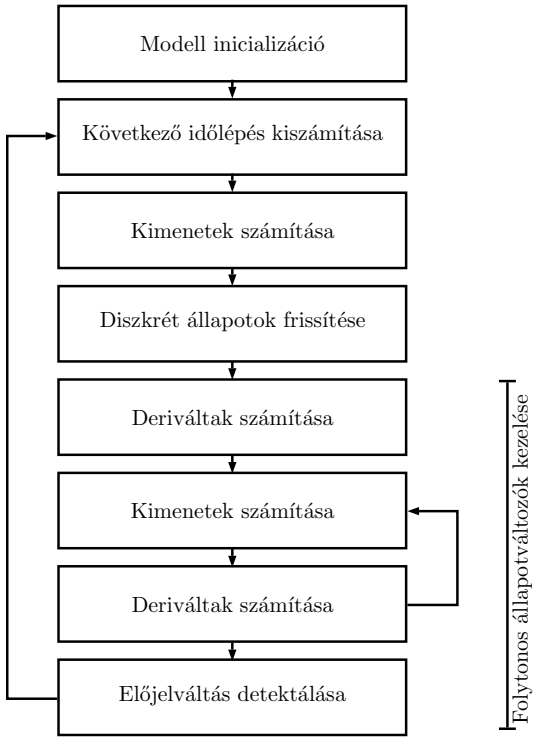
## 3.1. Az erőmodell

A dinamikai modell bemenő jeleként z irányú eredő erőt, illetve három nyomatékot igényel. Ennek előállítására egy `Interpreted Matlab Function` blokkot kell használnunk, ami a (2.26) egyenletben megadott mátrix segítségével kiszámítja a megfelelő bemeneteket a blokkba befutó négy felhajtóerőből. Ezek az erők az egyes rotorokon képződnek, viszont ezen a ponton fizikai korlátokba ütközik a modell.

A rotorok nem tudnak visszafelé forogni, és egy maximális fordulatszámon túl nem tudnak tovább gyorsulni. Emiatt szaturációt kell alkalmazni, aminek minimális értéke zérus (mivel a rotor nem foroghat visszafele), a maximális értéke pedig $f_{max}$, rotorparaméterektől függő, mérhető adat. Célszerű lehet az aktuátordinamika pontosabb modellezése érdekében a rotorok szöggyorsulását is korlátok közé vonni, azonban ez jelentősen elbonyolítaná a szabályozás tervezését, így eltekintünk tőle.

## 3.2. Az S-function

A dinamikai modell differenciálegyenlet-rendszerének kezelését a Matlab `S-Function` segítségével oldja meg. A blokk egy szubrutin (*3.2. ábra*), ami futásidőben egymás után több feladatot is végez [4]. Az egyes blokkok funkcióit a felhasználó szabadon megadhatja, az iterációt pedig a Matlab végzi el.




3.2. ábra. Az S-Function blokk működése

Az első lépés a modell incializálása, ami során a program előkészíti a szubrutin számára szükséges szimulációs struktúrát. A felhasználó megadhatja a szükséges állapotváltozók, kimenetek, illetve bemenetek számát, az állapotváltozók kezdeti értékeit, és az időlépést. A pontosság növelése érdekében az időlépések nagyságát a Simulink szabályozza automatikusan.

A második lépésben a blokk kiszámolja a következő időlépés nagyságát. Ezt követi a kimenetek kiszámítása, ami során a szubrutin átadja az állapotváltozókat a kimenő portra, majd a diszkrét állapotok frissítése. A modellezett rendszer nem igényli az ilyen típusú állapotváltozók használatát, így a további részletezésük nem szükséges.

Az utolsó szakasza a szubrutinnak a folytonos állapotváltozók kezelése. Ehhez először kiszámítja a megadott differenciálegyenletek segítségével az állapotváltozók adott időpillanatbeli deriváltjait, majd a kellő pontosság elérésig iterálja az állapotok, illetve a deriváltjaik számítását.

Az állapotváltozók új értékeinek meghatározása után a szubrutin visszatér a következő időlépés nagyságának kiszámításához. Előjelváltás esetén csökkenti az időlépést. Amennyiben azt észleli, hogy a szimuláció véget ért, az utolsó időlépésben végrehajt egy takarító parancsot.

# 4. fejezet

# PID szabályozótervezés

A létrehozott Simulink modell a Matlab saját eljárásaival linearizálható, majd PID szabályozó tervezhető rá. Az egyes tagok behangolását szintén Matlab parancsokkal lehet végezni, egyszerűbb rendszereknél a kívánt vágási körfrekvencia megadásával, bonyolultabb esetekben a beállás dinamikájára (pl. beállási idő, maradó hiba, túllövés, stb.) való feltételekkel.

## 4.1. A munkaponti linearizálás alapjai

A linearizálás előtt szükséges megtalálni a rendszer trim pontját. Ez egy olyan bemeneti kombináció, amire a rendszer viselkedése egy bizonyos állandósult állapotban marad. A Matlab `trim` algoritmusa a keresést úgy végzi, hogy a felhasználó által megadott bemenetek, állapotváltozók, és azok deriváltjainak értékeit összeveti az általa megtalált bemeneti kombinációra kapott értékekkel, és a hibát iteratív módszerrel minimalizálja.

Repüléstechnikában egy szokványos trim pont a "straight and level flight", amikor a repülőgép vízszintesen repül, állandó sebességgel. A kvadkoptereknél célszerű trim pont a "hover", azaz lebegés, ami hasonló elven, a vízszintes mozgás (x, y) megengedése mellett a függőleges mozgást (z) zérusnak veszi. Az x- és y-koordináták mentén való mozgást amiatt nem lehet korlátozni, mivel bedöntés esetén ezek semmilyen módon nem tarthatók állandó értéken (hiszen csak "felfelé" képesek erőt kifejteni a rotorok), a z-koordináta viszont igen.

A rendszer egyszerűsége miatt analitikusan is kiszámolható a szükséges bemenet. A lebegés megtartásához az szükséges, hogy a rotorokon képződő eredő erő függőleges komponense leküzdje a gravitációt. Ez szögfüggvények megfelelő használatával a következőképpen adható meg [2]:

$$T_0 = \frac{m \cdot g}{\cos\phi \cdot \cos\theta} \tag{4.1}$$

A linearizálást a Matlab `linmod` parancsával lehet elvégezni. Ez a megadott rendszerből (3.1. ábra) a munkapontbeli állapotváltozók és bemenetek értéke körül egy lineáris

állapottér-modellt épít fel. Mivel nem feltétlenül csak zérus értékekkel kapható linearizált modell, így a bemenetek, kimenetek, és állapotok trimtől való eltéréseit kell vizsgálni:

$$\delta\mathbf{x}(t) = \mathbf{x}(t) - \mathbf{x}_0 \tag{4.2}$$

$$\delta\mathbf{u}(t) = \mathbf{u}(t) - \mathbf{u}_0 \tag{4.3}$$

$$\delta\mathbf{y}(t) = \mathbf{y}(t) - \mathbf{y}_0 \tag{4.4}$$

A linearizált állapottér-modell az irányítástechnikában szokványos formában előállítható, a jelek említett eltéréseit figyelembe véve:

$$\delta\dot{\mathbf{x}}(t) = \mathbf{A} \cdot \delta\mathbf{x}(t) + \mathbf{B} \cdot \delta\mathbf{u}(t) \tag{4.5}$$

$$\delta\mathbf{y}(t) = \mathbf{C} \cdot \delta\mathbf{x}(t) + \mathbf{D} \cdot \delta\mathbf{u}(t) \tag{4.6}$$

## 4.2. A modell linearizálása

A kvadrotor lineáris állapottér-modelljének felépítése az előző fejezetben tárgyalt eljárással végrehajtható. Az egyes jelek a következőképpen definiálhatók, és az alábbi kikötések fogalmazhatók meg rájuk:

$$\mathbf{u}(t) = \begin{bmatrix} f_1 & f_2 & f_3 & f_4 \end{bmatrix}^T \tag{4.7}$$

$$\mathbf{u}_0 = \begin{bmatrix} \frac{T_0}{4} & \frac{T_0}{4} & \frac{T_0}{4} & \frac{T_0}{4} \end{bmatrix}^T \tag{4.8}$$

$$\mathbf{x}(t) = \begin{bmatrix} p_x & p_y & p_z & v_x & v_y & v_z & \phi & \theta & \psi & p & q & r \end{bmatrix}^T \tag{4.9}$$

$$\mathbf{x}_0 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T \tag{4.10}$$

$$\mathbf{y}(t) = \begin{bmatrix} p_x & p_y & p_z & v_x & v_y & v_z & \phi & \theta & \psi & p & q & r \end{bmatrix}^T \tag{4.11}$$

$$\mathbf{y}_0 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T \tag{4.12}$$

Az állapotér-modell mátrixai egy MIMO (több bemenetű, több kimenetű) rendszert írnak le. Ezek Matlabbal kiszámolva a következők:

$$\mathbf{A} = \begin{bmatrix} \mathbf{0}_{3x3} & \mathbf{I}_{3x3} & \mathbf{0}_{3x3} & \mathbf{0}_{3x3} \\ \mathbf{0}_{3x3} & \mathbf{0}_{3x3} & \mathbf{A}_{23} & \mathbf{0}_{3x3} \\ \mathbf{0}_{3x3} & \mathbf{0}_{3x3} & \mathbf{0}_{3x3} & \mathbf{I}_{3x3} \\ \mathbf{0}_{3x3} & \mathbf{0}_{3x3} & \mathbf{0}_{3x3} & \mathbf{0}_{3x3} \end{bmatrix} \tag{4.13}$$

$$\mathbf{0}_{3x3} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad \mathbf{I}_{3x3} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \mathbf{A}_{23} = \begin{bmatrix} 0 & 9,81 & 0 \\ -9,81 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{4.14}$$

$$\mathbf{B} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 2,14 & 2,14 & 2,14 & 2,14 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 28,32 & -28,32 & -28,32 & 28,32 \\ -28,32 & -28,32 & 28,32 & 28,32 \\ -4,35 & 4,35 & -4,35 & 4,35 \end{bmatrix} \tag{4.15}$$

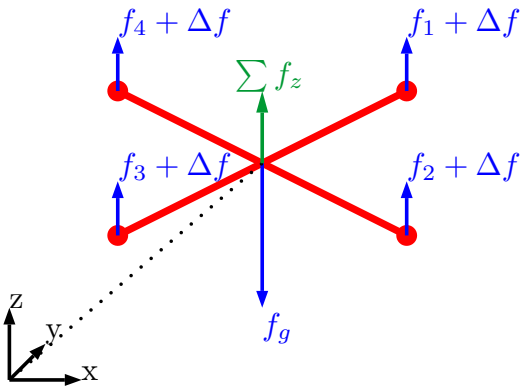
$$\mathbf{C} = \mathbf{I}_{12x12} \quad \mathbf{D} = \mathbf{0}_{12x4} \tag{4.16}$$

## 4.3. A modell szétbontása

Mivel a PID szabályozást SISO rendszerre lehetséges megalkotni, ezért a **B** és **C** mátrixokat célszerű úgy átalakítani, hogy a linearizált rendszer egyes, egymástól elkülönülő alrendszereit külön-külön is le lehessen írni. A rendszer állapotai nem változnak, így az **A** mátrix módosítására nincs szükség, a **D** mátrix pedig zérusnak tekinthető. Négy ilyen "részdinamika" definiálható, hasonló módon, mint az erők és nyomatékok meghatározásánál (2.26).

$$\delta\dot{\mathbf{x}}(t) = \mathbf{A} \cdot \delta\mathbf{x}(t) + \mathbf{b} \cdot \delta\mathbf{u}(t) \tag{4.17}$$

$$\delta y(t) = \mathbf{c} \cdot \delta\mathbf{x}(t) + 0 \cdot \delta\mathbf{u}(t) \tag{4.18}$$



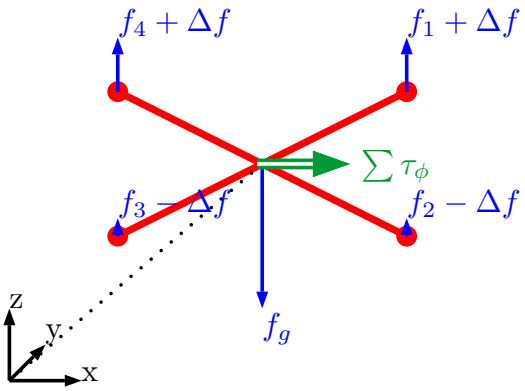

4.1. ábra. Lift dinamika ("Emelés")

Az első a Lift (*4.1. ábra*) dinamika, ami kizárólag a kvadrotor z pozíciójának a szabályozásáért felel. Belátható, hogy ez a négy rotor által kiadott erőnek az együttes megemelésével, illetve csökkentésével érhető el. A kimenet ezáltal $p_z$, a $\mathbf{b}$ vektor pedig $\mathbf{B}$ oszlopainak összegzésével kapható meg.

$$y(t) = p_z \tag{4.19}$$

$$\mathbf{b} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 8,547 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T \tag{4.20}$$

$$\mathbf{c} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T \tag{4.21}$$



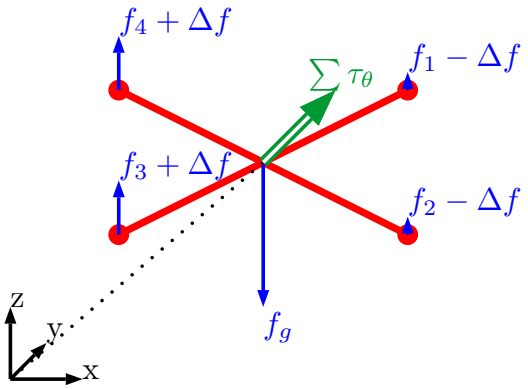

4.2. ábra. Roll dinamika ("Billentés")

A második a Roll (*4.2. ábra*) dinamika, ami kvadkopterre ható $\tau_\phi$ nyomatékon keresztül hat a $\phi$ szögre, illetve járulékosan a $p_y$ pozícióra is. A szög pozitív változásához az 1. és 4. rotorok erejét kell egyenlően megnövelni, a 2. és 3. rotorokét pedig azonos mértékben csökkenteni. Így a kimenet $\phi$, a $\mathbf{b}$ vektor pedig $\mathbf{B}$ oszlopainak (1) + (4) − (2) − (3) összegzésével kapható meg.

$$y(t) = \phi \tag{4.22}$$

$$\mathbf{b} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 113,262 & 0 & 0 \end{bmatrix}^T \tag{4.23}$$

$$\mathbf{c} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T \tag{4.24}$$




4.3. ábra. Pitch dinamika ("Bólintás")

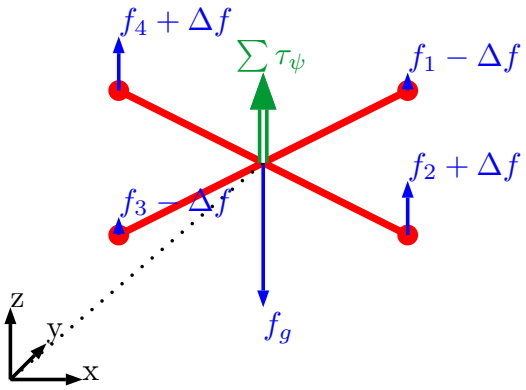
A harmadik a Pitch (*4.3. ábra*) dinamika, ami kvadkopterre ható $\tau_\theta$ nyomatékon keresztül hat a *theta* szögre, illetve járulékosan a $p_x$ pozícióra is. A szög pozitív változásához az 3. és 4. rotorok erejét kell egyenlően megnövelni, az 1. és 2. rotorokét pedig azonos mértékben csökkenteni. Így a kimenet $\theta$, a $\mathbf{b}$ vektor pedig $\mathbf{B}$ oszlopainak (3)+(4)−(1)−(2) összegzésével kapható meg.

$$y(t) = \theta \tag{4.25}$$

$$\mathbf{b} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 113,262 & 0 \end{bmatrix}^T \tag{4.26}$$

$$\mathbf{c} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}^T \tag{4.27}$$

A negyedik a Yaw (*4.4. ábra*) dinamika, ami kvadkopterre ható $\tau_\psi$ nyomatékon keresztül kizárólag a $\psi$ szögre hat. Idáig különös tekintettel kellett lenni arra, az előző három dinamika felírása során, hogy a propellerekre ható nyomatékok kiejtsék egymást,




4.4. ábra. Yaw dinamika ("Legyezés")

így mindig párban kellett működtetni őket. Most az ellenkező a cél, a szög pozitív változásához pedig a 2. és 4. rotorok erejét kell egyenlően megnövelni, az 1. és 3. rotorokét pedig azonos mértékben csökkenteni. Így a kimenet $\psi$, a $\mathbf{b}$ vektor pedig $\mathbf{B}$ oszlopainak $(2) + (4) - (1) - (3)$ összegzésével kapható meg.

$$y(t) = \psi \tag{4.28}$$

$$\mathbf{b} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 17,387 \end{bmatrix}^T \tag{4.29}$$

$$\mathbf{c} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}^T \tag{4.30}$$

A négy SISO modellt a Matlab saját változótípussal (state space model) képes eltárolni. Ezeket külön fájlba kimentve, később is felhasználhatók a tervezési folyamatban.

## 4.4. A szabályozó behangolása

A rendszerek átviteli függvényeit megvizsgálva arra a következtetésre jutunk, hogy kizárólag kettős integrátorokat tartalmaznak. Ezek Laplace operátortartományban a következő általános alakban állnak elő:

$$G(s) = \frac{b_0}{s^2} \tag{4.31}$$

A rendszer egyszerűsége miatt a szabályozó behangolása a Matlab `pidtune` algoritmusával történik. A függvény három bemenetű: az első a szabályozott rendszer, a második a szabályozó típusa, a harmadik pedig a kívánt vágási körfrekvencia ($\omega_c$).

A 6. fejezetben (LPV modellezés) a felhasznált irodalom [2] alapján választott vágási körfrekvenciákat használjuk fel a szabályozótervezés során. Ez amiatt szükséges, hogy a

későbbi összehasonlítások során tisztán a technikák hatékonyságát lehessen vizsgálni, és az esetleges paraméterbeli eltérések ne játsszanak nagy szerepet. Idézve a későbbi indoklásból: "A szerzők által alkalmazott kvadrotor paraméterek nagyságrendileg azonosak az általunk használtakkal, így a szabályozótervezéskor célszerű először a szerzők által javasolt feltételekre kipróbálni a modellt. A választott szabályozó PDF (proporcionális tag, és szűrővel ellátott deriváló tag) a három szöghelyzetre, a magasságra pedig PIDF (integráló tagot is tartalmaz)." A vágási körfrekvenciák értékei a következők:

$$\omega_{c,z} = 7[Hz] \quad \omega_{c,\phi} = 50[Hz] \quad \omega_{c,\theta} = 40[Hz] \quad \omega_{c,\psi} = 10[Hz] \tag{4.32}$$

A PIDF, illetve a PDF szabályozók felépítése a Matlab egységes jelölése alapján a következő:

$$C_{PIDF} = K_p + K_i \cdot \frac{1}{s} + K_d \cdot \frac{s}{T_f \cdot s + 1} \tag{4.33}$$

$$C_{PDF} = K_p + K_d \cdot \frac{s}{T_f \cdot s + 1} \tag{4.34}$$

A behangolt szabályozók együtthatói az egyes dinamikákra az alábbiakban láthatók. A Simulink implementációban ezek használandók a PID blokkok megadható értékeiként.

$$p_z : K_p = 1,6 \quad K_i = 0,349 \quad K_d = 0,798 \quad T_f = 0,0262 \tag{4.35}$$

$$\phi : K_p = 4,63 \quad K_d = 0,431 \quad T_f = 0,000175 \tag{4.36}$$

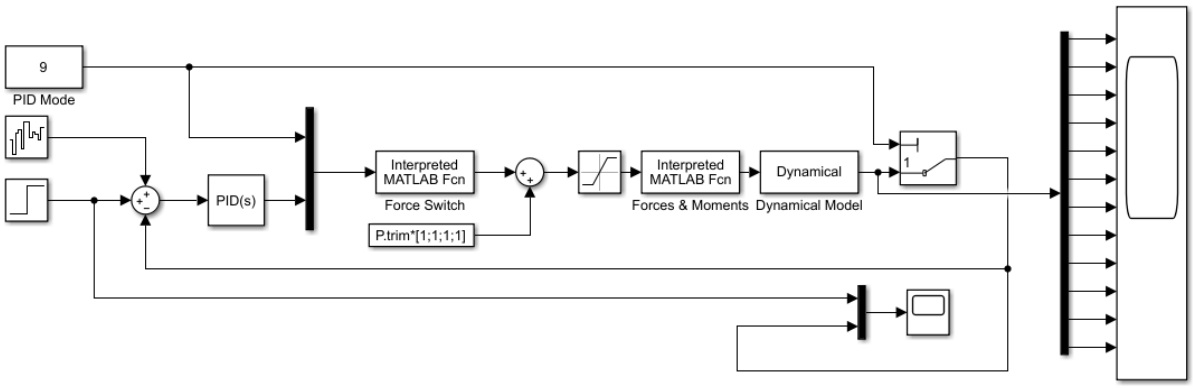
$$\theta : K_p = 2,96 \quad K_d = 0,345 \quad T_f = 0,000219 \tag{4.37}$$

$$\psi : K_p = 1,21 \quad K_d = 0,561 \quad T_f = 0,000875 \tag{4.38}$$

## 4.5. A szabályozó ellenőrzése

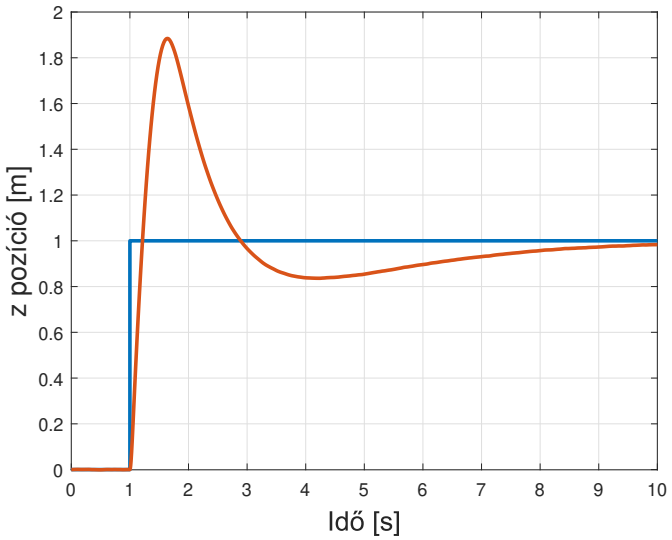
A Simulink implementációban (*4.5. ábra*) egységugrás jellel gerjesztjük meg a szabályozott rendszert. A PID szabályozó együtthatói megegyeznek az előző fejezetben megadottakkal, szöghelyzetek szabályozása esetén $K_i = 0$.

A "PID Mode" állandóval jelezhető a "Force Switch" blokk számára, hogy a bemenő jelek milyen kombinációját állítsa elő a szabályozó kimenetén kapott $\delta f$ erőből, a 4.3 fejezetben leírt módon. Az ábrán jelenleg a kilencedik állapotváltozót szabályozzuk ($\psi$).



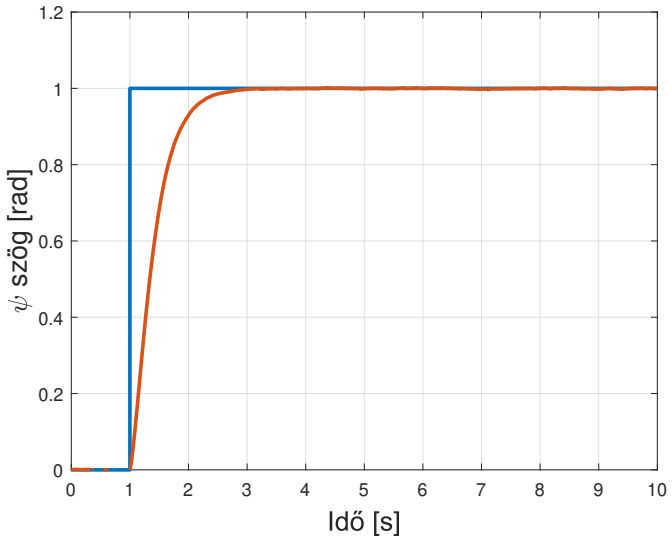

4.5. ábra. A szabályozókör Simulink modellje

A visszacsatoláshoz hozzáadunk egy gyenge fehér zajt (a jelnél 5 nagyságrenddel kisebb), ezzel szimulálva a valóságban jelenlévő zavaró hatást. A szabályozott rendszer a korábban már látott (3.1. ábra) modell, aminek a bemenő jeléhez hozzá kell adni a linearizáláskor használt trim bemenetet. A kimenetet is korrigálni kellene, azonban a linearizálást zérus állapotokkal végeztük, így erre most nincs szükség. A rendszer munkapont körüli helyes működését egységugrás referenciajellel lehet bemutatni.



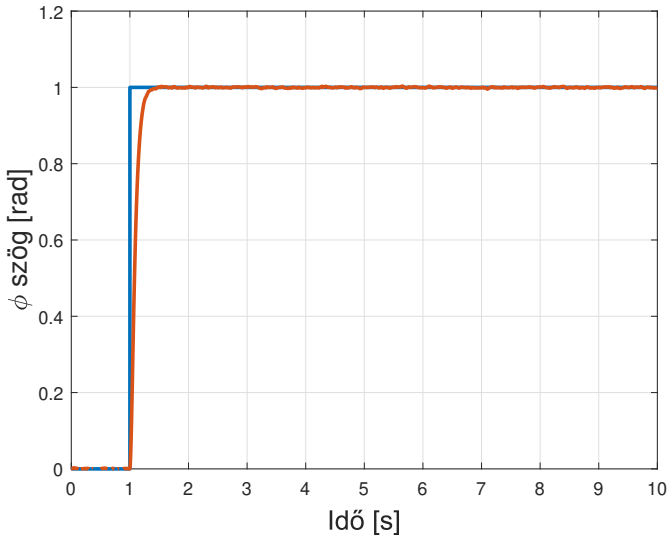

4.6. ábra. A Lift dinamika válasza egységugrásra

A z pozícióra, illetve a $\psi$ szögre vonatkozó referenciajel, és a rendszer válasza kirajzolva látható a *4.6. és 4.7 ábrákon.* Lehetőség van a többi állapotváltozó ábrázolására is, azonban lift szabályozásnál csak a z pozíció és sebesség, yaw szabályozásnál pedig a $\psi$ szög és szögsebesség változik. A többi állapotváltozó zérus.

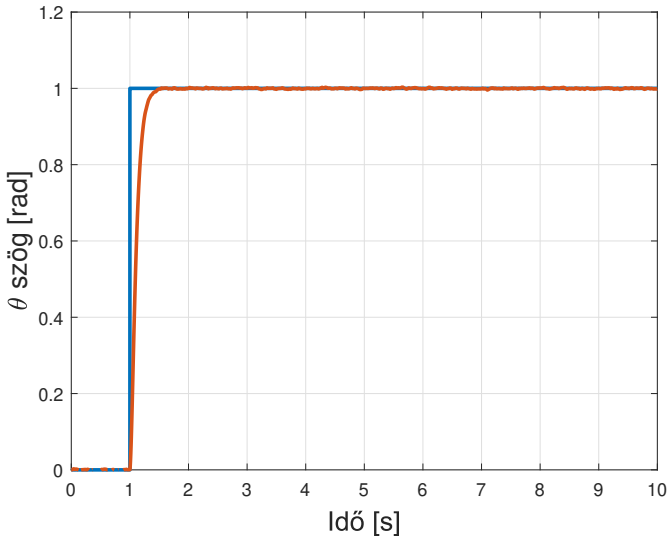



4.7. ábra. A Yaw dinamika válasza egységugrásra

A $\phi$, illetve $\theta$ szögek szabályozásakor (*4.8. és 4.9 ábrák*) látható, viszonylag gyors beállási idővel képes működni a kontroller. Az egyszerű PID szabályozó alkalmazásának hátrányai, és a komplexebb módszerek szükségessége akkor válik egyértelművé, amikor a többi állapotváltozó értékét is megvizsgáljuk.

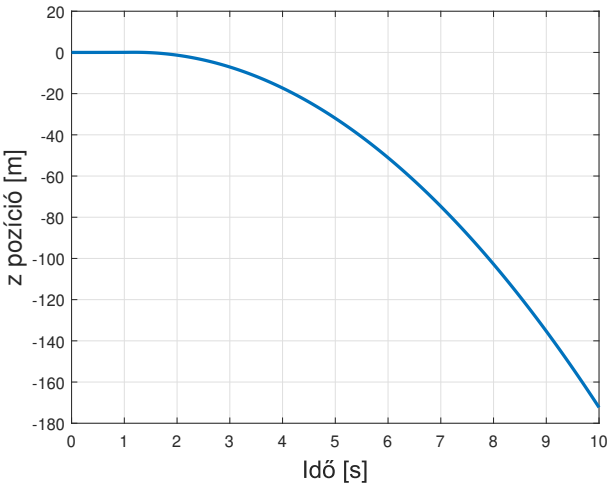



4.8. ábra. A Roll dinamika válasza egységugrásra

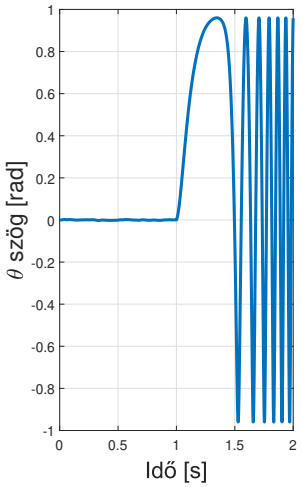



4.9. ábra. A Pitch dinamika válasza egységugrásra

Ilyen például a z pozíció. A kvadrotor bedöntésekor ugyanis a bemenő trim érték változatlan marad, így a jármű elkezd zuhanni (*4.10. ábra*).




4.10. ábra. Zuhanás Pitch szabályozóval



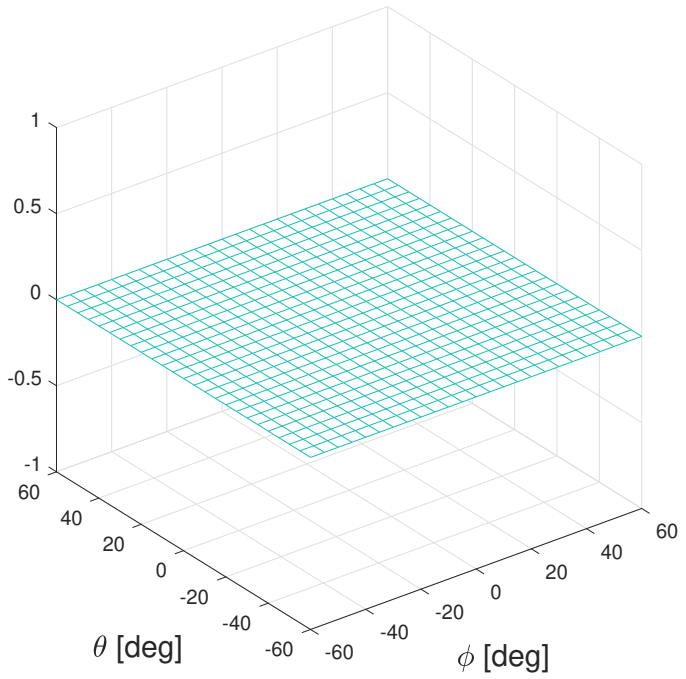

4.11. ábra. Az átcsatolás hatása ($\phi = 35°$)

A linearizálási munkaponttól távoli szabályozás esetén pedig további problémát jelent az alrendszerek csatoltsága miatt megjelenő instabilitás. $\phi = 35°$ szöghelyzet esetén a Pitch szabályozó működése jelentős akadályokba ütközik (*4.11. ábra*). Az átcsatolás miatt pörögni kezd a kvadrotor (a $\psi$ szög fokozatosan növekszik), és ennek során többször is átbillen (a $\phi$ és $\theta$ szögek egymással ellentétesen lengnek).

# 5. fejezet

# Gain scheduled szabályozás

Az előző fejezet során láthatóvá váltak a működési határai a hover állapotban linearizált modellnek. Célszerű megoldás lehet a munkatartomány kiterjesztése, gain scheduling alkalmazásával. Az eljárás lényege az, hogy a linearizált modell mátrixai, és a kontrollerek együtthatói előre megválasztott paraméterektől függnek. Az egyértelmű választás ezen változókra a kvadrotor a roll ($\phi$) és a pitch ($\theta$) szögei, hiszen ezek változására romlik el leghamarabb a linearitás.




5.1. ábra. A szögek által kifeszített 25x25 csomópontú háló

Ezekkel a szögekkel kijelölünk egy munkatartományt, amelyet "behálózunk" (*5.1. ábra*), azaz feltöltjük az állapotváltozók lehetséges értékeivel egy `ndgrid` (n dimenziós rács) struktúrát. A szélsőértékei ennek a tartománynak -60°, és 60°, mivel a kvadrotor z pozíciója ezen túl már nehezen tartható nulla közelében. Ez amiatt van, mert a gravitációból származó erőt képtelenek lesznek kompenzálni túl nagy bedöntés esetén a rotorok.

A gain scheduled kontroller tervezés négy lépésből álló folyamat [5]. A soron következő fejezetek ezeket a lépéseket fejtik ki bővebben.

## 5.1. Az állapottér modell-mátrix létrehozása

Az első lépésben a háló minden csomópontjában linearizáljuk a modellt. Ezt a 4. fejezetben részletezett módon, a Matlab `linmod` parancsával tehetjük meg. Fontos, hogy már a linearizálás közben változóként kell kezelni a trim bemenetet, hogy biztosan hover állapotban maradjon a jármű! Ezen kívül az állapotváltozók és kimenetek esetében is figyelni kell a szögek számításba vételére.

$$T_0 = \frac{m \cdot g}{\cos\phi \cdot \cos\theta} \tag{5.1}$$

$$\mathbf{u}(t) = \begin{bmatrix} f_1 & f_2 & f_3 & f_4 \end{bmatrix}^T \tag{5.2}$$

$$\mathbf{u}_0 = \begin{bmatrix} \frac{T_0}{4} & \frac{T_0}{4} & \frac{T_0}{4} & \frac{T_0}{4} \end{bmatrix}^T \tag{5.3}$$

$$\mathbf{x}(t) = \begin{bmatrix} p_x & p_y & p_z & v_x & v_y & v_z & \phi & \theta & \psi & p & q & r \end{bmatrix}^T \tag{5.4}$$

$$\mathbf{x}_0 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & \phi_0 & \theta_0 & 0 & 0 & 0 & 0 \end{bmatrix}^T \tag{5.5}$$

$$\mathbf{y}(t) = \begin{bmatrix} p_x & p_y & p_z & v_x & v_y & v_z & \phi & \theta & \psi & p & q & r \end{bmatrix}^T \tag{5.6}$$

$$\mathbf{y}_0 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & \phi_0 & \theta_0 & 0 & 0 & 0 & 0 \end{bmatrix}^T \tag{5.7}$$

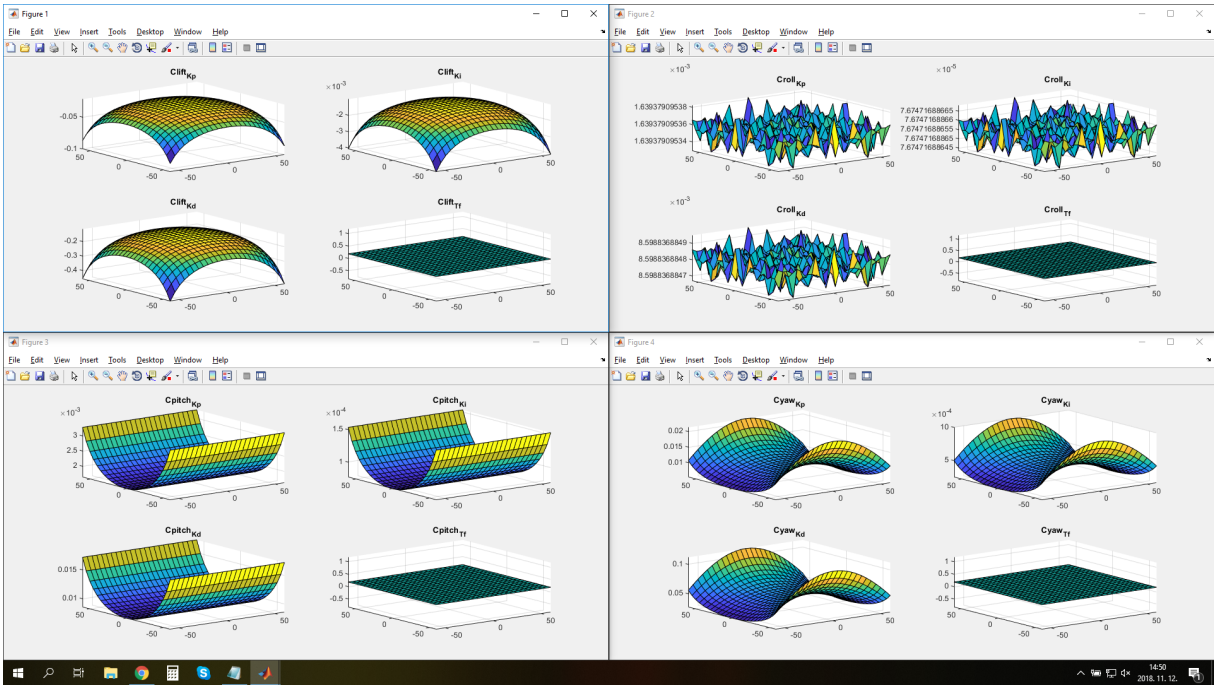
Ezt követően a dinamikák szétbontása következik: külön-külön létrehozzuk a Lift, Roll, Pitch, és Yaw alrendszerekre az állapottér-modelleket, szintén a 4. fejezetben részletezett módon. Az alrendszerek modelljeivel futásidőben négy darab 25x25 elemű mátrixot töltünk fel, amiket külön fájlokban tárolunk el, későbbi felhasználásra.

## 5.2. A szabályozók behangolása

A második lépés a tervezési folyamatban a szabályozók kiválasztása és behangolása. A korábban használt PID kontrollertől a fő eltérés az, hogy az együtthatók paraméterfüggőek, a következő módon:

$$C_{PIDF} = K_p(\phi_0, \theta_0) + K_i(\phi_0, \theta_0) \cdot \frac{1}{s} + K_d(\phi_0, \theta_0) \cdot \frac{s}{T_f \cdot s + 1} \tag{5.8}$$

Az együtthatók paraméterfüggését célszerű úgy meghatározni, hogy egy analitikus kifejezéssel azok leírhatók legyenek. Ehhez először is a 4. fejezetben használt `pidtune` eljárás alkalmazásával behangolunk az négy alrendszer minden munkapontjára, összesen 2500 PID szabályozót. Ezután ezek együtthatóit kirajzoljuk a 5.1. ábrán látható hálót alapul véve. Az eredményről készített képernyőkép a *5.2. ábrán* látható.



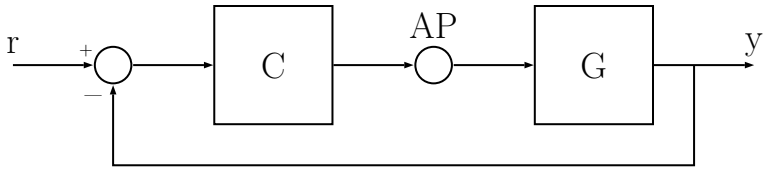
5.2. ábra. Képernyőkép a hangolás korai eredményéről

A szabályozóseregre ezután a Matlab Curve Fitting (Görbeillesztő) applikációja segítségével függvényeket illesztünk addig, amíg meg nem találjuk a lehető legkevesebb tagból álló, megfelelő pontosságú kombinációt. Az eredmény egy negyedfokú polinom, ami a következő formában áll elő:

$$K = C_{00} + C_{20} \cdot \phi^2 + C_{02} \cdot \theta^2 + C_{22} \cdot \phi^2 \cdot \theta^2 + C_{40} \cdot \phi^4 + C_{04} \cdot \theta^4 \tag{5.9}$$

A lineáris modellsereghez a szabályozó tervezésére célszerű eszköz a Matlab `systune` parancsa. Egy tetszőleges modellre a *5.3. ábra* alapján áll össze a szabályozási kör, amiben

a rendszer (G), és az előtte elhelyezett szabályozó (C) között egy `AnalysisPoint` (AP) elem tűnik fel, ami a későbbiekben a felnyitott kör vizsgálatára használható.




5.3. ábra. A szabályozási kör modellje

A `systune`-nak a szabályozási kör modelljén kívül szüksége van megadott hangolási célokra, erre `TuningGoal`-okat kell definiálni. A kvadkopter szabályozásához két ilyen előírásra van szükség, amiket szabadon kombinálva lehet felhasználni a megfelelő dinamikájú válasz eléréséhez.

Az egyik a `TuningGoal.StepTracking`, ami két jelet (egy bemenet, egy kimenet), és egy stabil referenciarendszert kíván bemenetéül. A cél az, hogy a választott bemenetre adott gerjesztésre a referenciarendszer válaszát kövesse minél pontosabban a választott kimeneten mérhető válasz. Célszerűen a bemenő jel r (referenciajel), a kimenő jel pedig y (a zárt kör kimenete).

A másik a `TuningGoal.Tracking`, amit a `StepTracking` helyett lehet használni, ha pontosan meg kívánjuk adni a rendszer válaszának a dinamikáját. A `StepTracking`-hez hasonlóan, egy megadott bemenő jelre (célszerűen r) adott gerjesztésre adott választ figyelünk egy megadott kimenő jelen (célszerűen y). Előírható a beállási idő, a maximális állandósult állapotbeli maradó hiba, illetve a maximális túllövés.

A szabályozósereg hangolásához a korábban létrehozott `ndgrid` struktúrára, illetve az állapottér modell-mátrix betöltésére is szükség van, ugyanis ezek szolgálnak scheduling változóként, illetve scheduled modellként. A szabályozókat a (5.8) egyenletben megadott formájukban hangoljuk. Fontos előzetesen megadni a deriváló tag szűrőjének időállandóját ($T_f$).

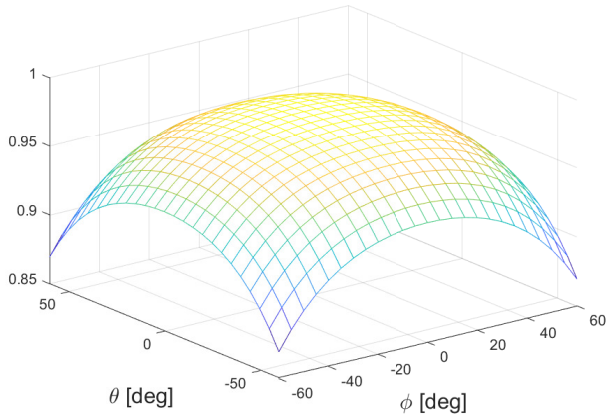
Az együtthatók előfordulási formája a korábban meghatározott negyedfokú polinom (5.9), amit a Matlab `Shape Function`-ként kezel. Ez egy olyan szimbolikus függvény, amelynek létrehozása előfeltétele a hangolásnak. Maguk az együtthatók a `tunableSurface` függvény segítségével definiálhatók, ami a $C_{00}$ kezdeti értékének megválasztása (célszerűen 1) után a Shape Function alapján, az `ndgrid` rács terjedelmében kifeszít egy felületet. Ez a hangolható gain felület, amivel a `systune` dolgozni tud.

A Lift alrendszer szabályozóinak behangolására `TuningGoal.Tracking` feltétel használható. A beállási idő 3 s, a állandósult állapotbeli maximális hiba 0.0001, a maximális túllövés pedig 1.2. Ezen kívül $T_{f,lift} = 0,0262$. A kapott felületek:

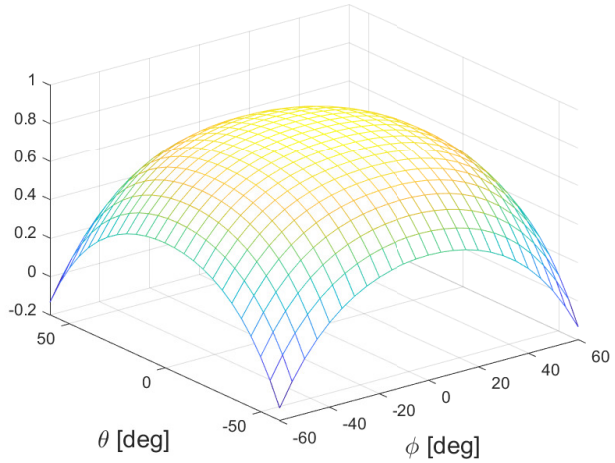
$$C_p = \begin{bmatrix} 0,9808 & -0,0192 & -0,0192 & -0,0192 & -0,0192 & -0,0192 \end{bmatrix} \tag{5.10}$$

$$C_i = \begin{bmatrix} 0,8333 & -0,1667 & -0,1667 & -0,1667 & -0,1667 & -0,1667 \end{bmatrix} \tag{5.11}$$

$$C_d = \begin{bmatrix} 1,0000 & 0,0000 & 0,0000 & 0,0000 & 0,0000 & 0,0000 \end{bmatrix} \tag{5.12}$$




5.4. ábra. A Lift dinamika P felülete




5.5. ábra. A Lift dinamika I felülete

A Roll, Pitch és Yaw alrendszerek szabályozóinak behangolására `TuningGoal.StepTracking` feltétel használható, a következő referenciarendszerrel:

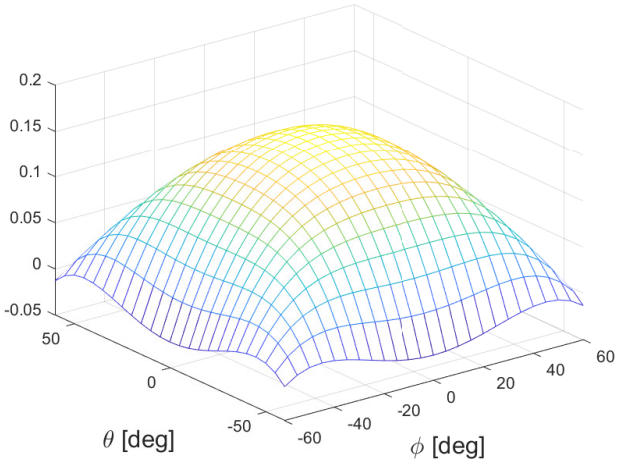
$$G_{ref} = \frac{1}{T_r \cdot s + 1} \tag{5.13}$$

A felhasznált időállandók, és a kapott felületek, a három dinamika szabályozására, a Roll-al kezdve $T_{f,roll} = 0,000175$, illetve $T_{r,roll} = 0,08$.

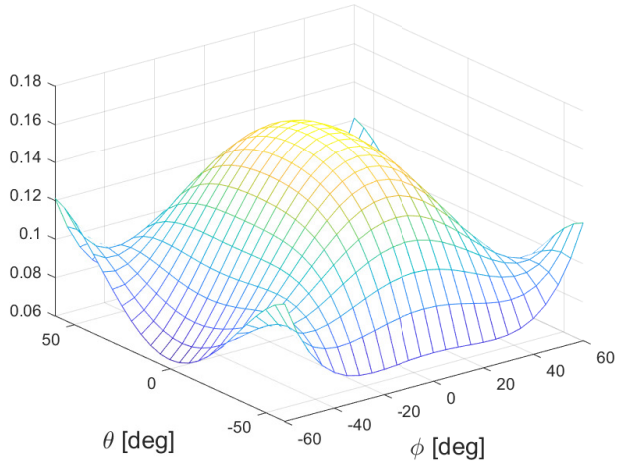
$$C_p = \begin{bmatrix} 0,1571 & -0,0928 & -0,0840 & 0,1571 & -0,0643 & -0,0731 \end{bmatrix} \tag{5.14}$$

$$C_i = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{5.15}$$

$$C_d = \begin{bmatrix} 0,1641 & -0,1532 & -0,0391 & 0,1212 & 0,0571 & -0,0386 \end{bmatrix} \tag{5.16}$$




5.6. ábra. A Roll dinamika P felülete



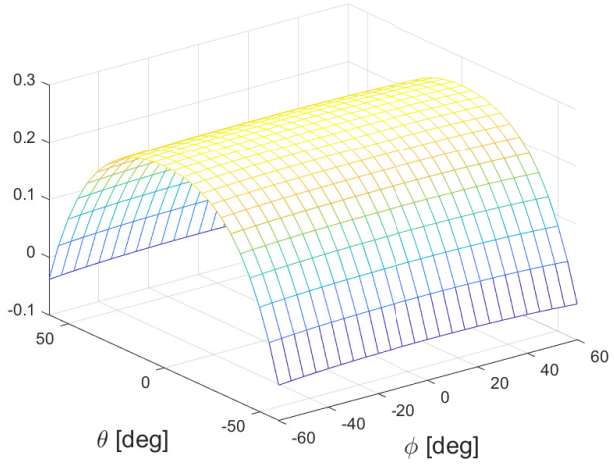

5.7. ábra. A Roll dinamika D felülete

$$T_{f,pitch} = 0,000219 \quad T_{r,pitch} = 0,1 \tag{5.17}$$

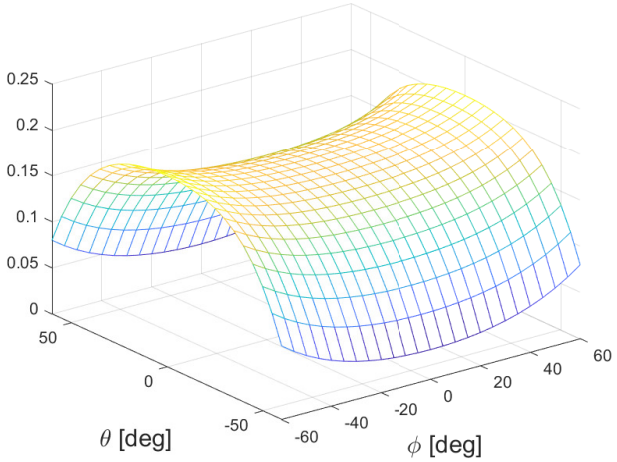
$$C_p = \begin{bmatrix} 0,2556 & -0,0030 & -0,1240 & -0,0030 & -0,0016 & -0,1240 \end{bmatrix} \tag{5.18}$$

$$C_i = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{5.19}$$

$$C_d = \begin{bmatrix} 0,1766 & 0,0145 & -0,0629 & 0,0145 & 0,0121 & -0,0629 \end{bmatrix} \tag{5.20}$$




5.8. ábra. A Pitch dinamika P felülete



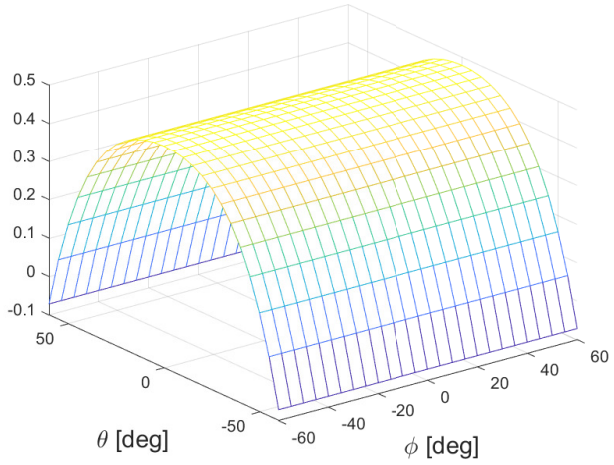

5.9. ábra. A Pitch dinamika D felülete

$$T_{f,roll} = 0,000875 \quad T_{r,roll} = 0,4 \tag{5.21}$$

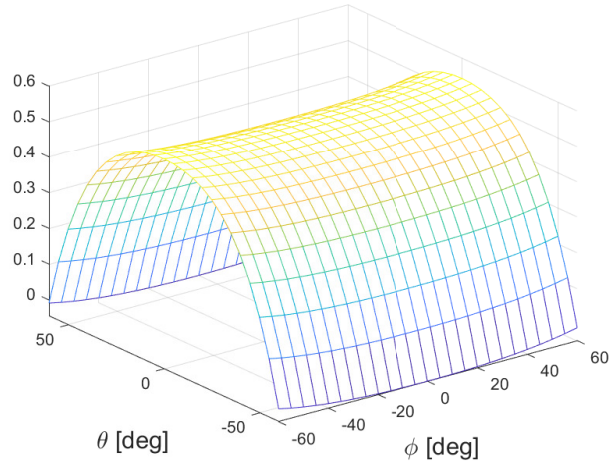
$$C_p = \begin{bmatrix} 0,4779 & -0,0000 & -0,2389 & -0,0000 & 0,0001 & -0,2389 \end{bmatrix} \tag{5.22}$$

$$C_i = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{5.23}$$

$$C_d = \begin{bmatrix} 0,5190 & 0,0104 & -0,2452 & 0,0104 & 0,0101 & -0,2452 \end{bmatrix} \tag{5.24}$$



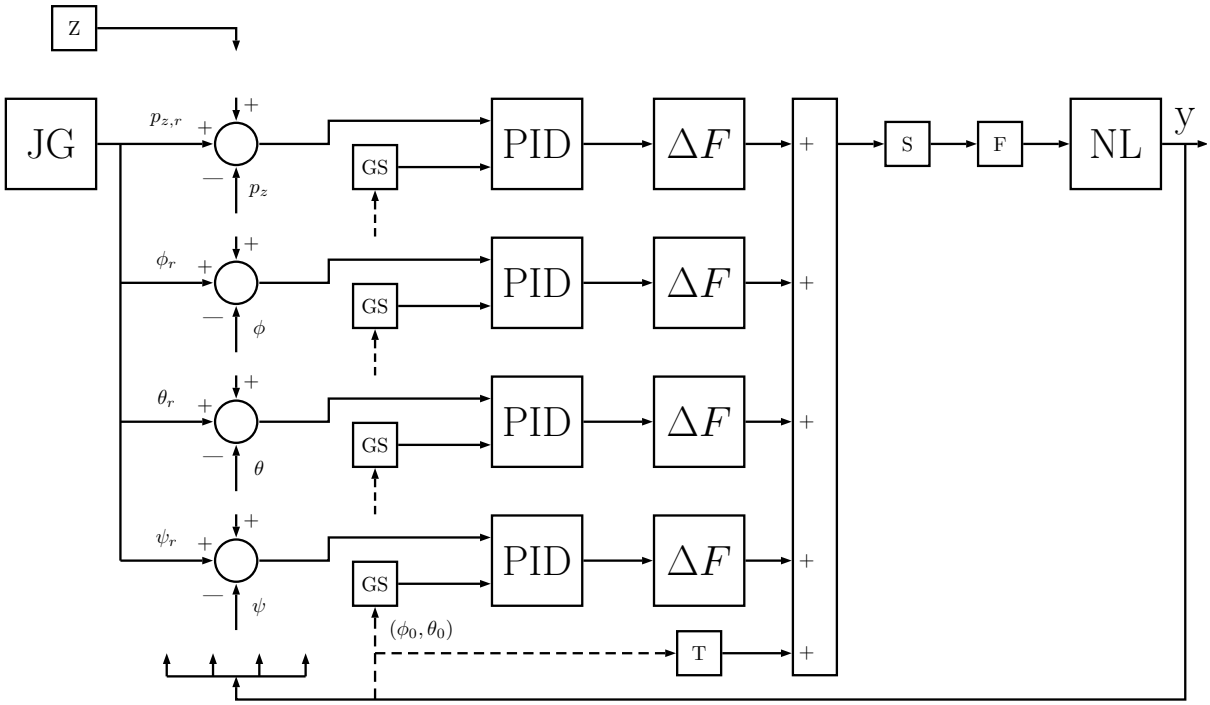

5.10. ábra. A Yaw dinamika P felülete




5.11. ábra. A Yaw dinamika D felülete

## 5.3. Simulink implementáció

A kábelek nagy száma miatt a Simulink-modell bemutatása nem tehető meg egyszerűen képernyőképek segítségével. Ehelyett, az érthetőség javítása miatt egy letisztultabb ábrán (*5.12. ábra*) célszerű megmutatni az egyes felhasznált blokkokat, illetve azok működését.




5.12. ábra. Gain scheduled PID Simulink modell

A jelgenerátoron (`Signal Generator` blokk, ábrán JG) három, nagy szögkitérésű referenciajelet hozunk létre. Mindhárom jelben $p_{z,ref} = 0$, illetve két szögre megkötés vonatkozik. A feldolgozott cikk [2] alapján ezek:

1. Nagy $\phi$ trajektória: $\theta_{ref} = 40° \quad \psi_{ref} = 0°$
2. Nagy $\theta$ trajektória: $\phi_{ref} = 40° \quad \psi_{ref} = 0°$
3. Nagy $\psi$ trajektória: $\phi_{ref} = 40° \quad \theta_{ref} = 40°$

A visszacsatolás során fehér zajt (Z) adunk a hibajelhez, hogy szimuláljuk a valóságban megjelenő zavaró hatásokat. A hibajel a PID szabályozóba fut be, aminek az együtthatóit kívülről adjuk meg. Ezért a Gain Scheduling blokk (GS) felel, ami a $\phi$ és $\theta$ szögekből kiszámolja az aktuálisan szükséges együtthatókat. Ez egy Matlab függvény, ami bemeneteként a két szöget igényli, és tartalmazza az előző fejezetben meghatározott polinomokat. A blokk implementálása lehetséges `Lookup Table` segítségével is, ami az egyes csomópontokban fellelhető értékek között interpolációval határozza meg az együtthatókat.

A PID szabályozó által kiadott $\Delta f$ alapján a 4. fejezetben ismertetett Force Switch (ábrán $\Delta F$) blokkon megjelenik az egyes rotorokra vett erő. Ezeket az erőket mind a négy rendszerre összegezzük, hiszen a szétbontás jellege miatt ezt megtehetjük (egy adott

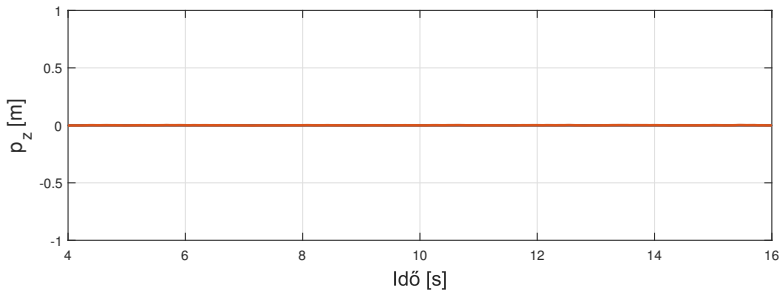
részdinamikán csak egy adott nyomaték, vagy erő ébred). Az így kapott összeghez hozzáadjuk továbbá az (immáron paraméterfüggő) trim értéket (ábrán T blokk által számolt), ami megpróbálja hover állapotban tartani a járműt.

Az összegzett rotorerőket szaturálás (ábrán S blokk) után az erőmodellnek (ábrán F blokk) átadva megkapjuk az erőket és nyomatékokat, amik már bemenő jeléül szolgálnak a nemlineáris modellnek (ábrán NL blokk). A tizenkét kimenő állapotváltozót ezután ábrázoljuk, és a szükséges kimeneteket visszacsatoljuk.

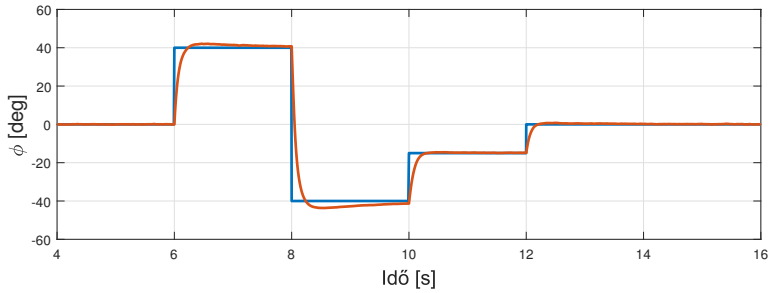
## 5.4. Futtatási eredmények

A szabályozás működésének ellenőrzésére lefuttatjuk a Simulink szimulációt. Négy, számunkra érdekes jelet vizsgálunk: a $p_z$ pozíciót, illetve a $\phi$, $\theta$, és $\psi$ szögeket.

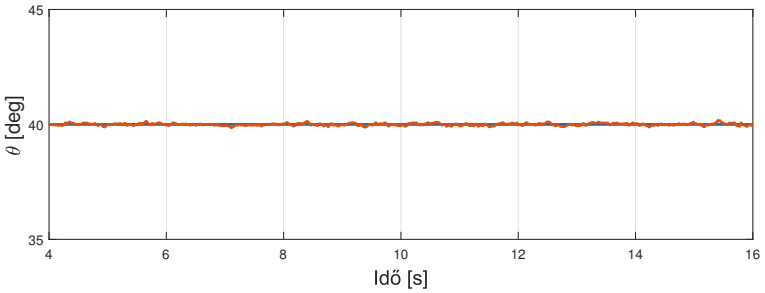
Az elsőként vizsgált referenciajel a nagy $\phi$ trajektória, majd a nagy $\theta$ trajektória, végül a nagy $\psi$ trajektória. Az eredmények pontos elemzése a 8. fejezetben olvasható. A kapott válaszok a következő három oldalon láthatók.



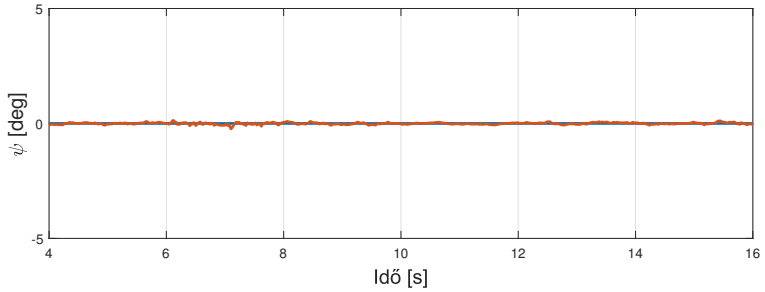

5.13. ábra. A z pozíció roll trajektóriára



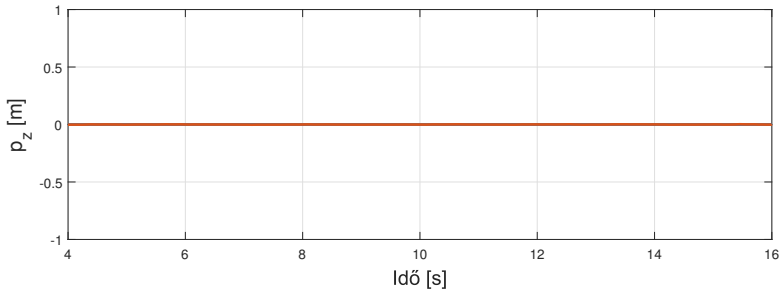

5.14. ábra. A $\phi$ szög roll trajektóriára



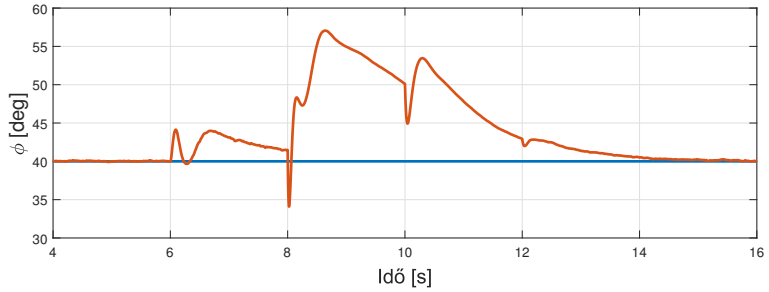

5.15. ábra. A $\theta$ szög roll trajektóriára



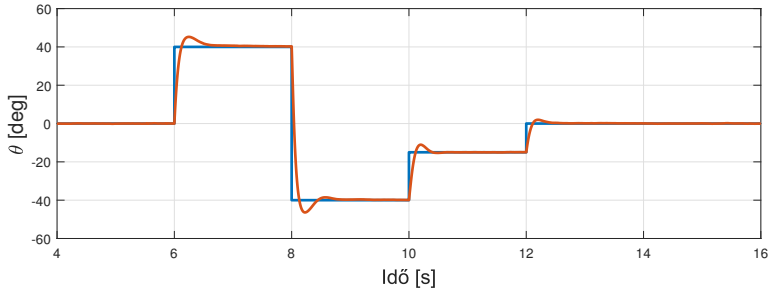

5.16. ábra. A $\psi$ szög roll trajektóriára



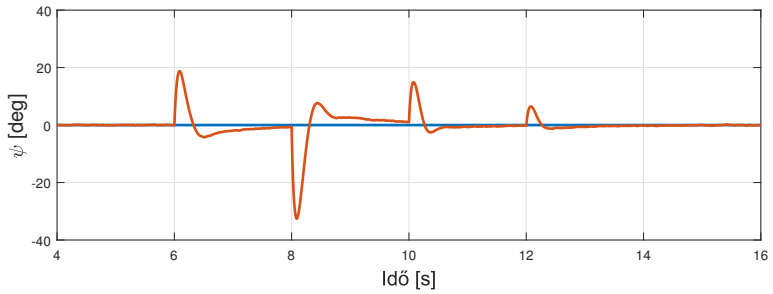

5.17. ábra. A z pozíció pitch trajektóriára



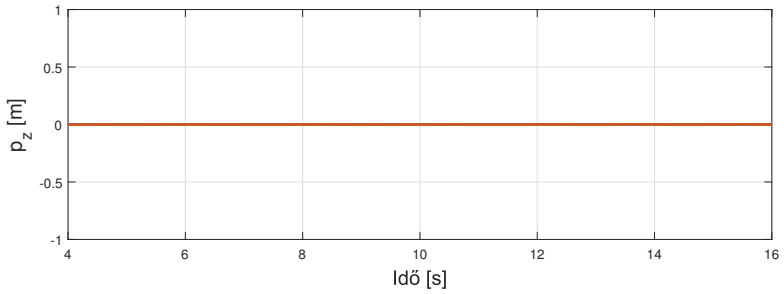

5.18. ábra. A $\phi$ szög pitch trajektóriára



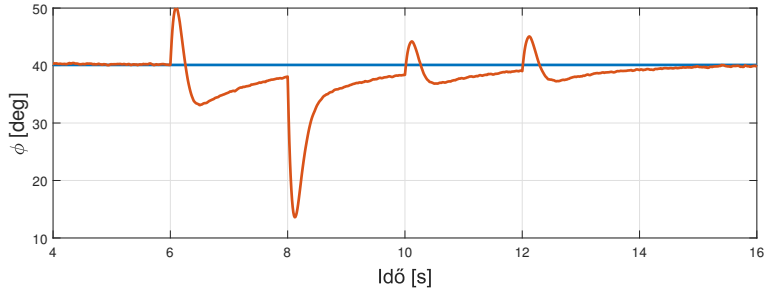

5.19. ábra. A $\theta$ szög pitch trajektóriára



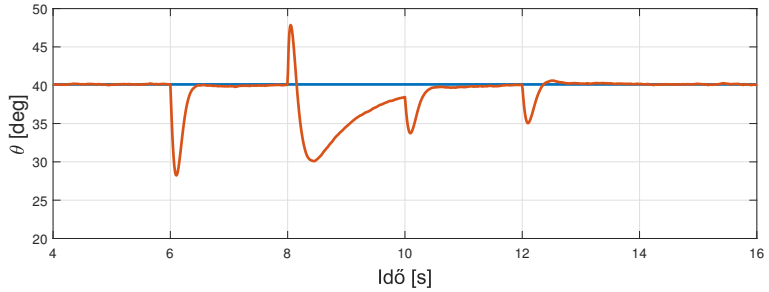

5.20. ábra. A $\psi$ szög pitch trajektóriára



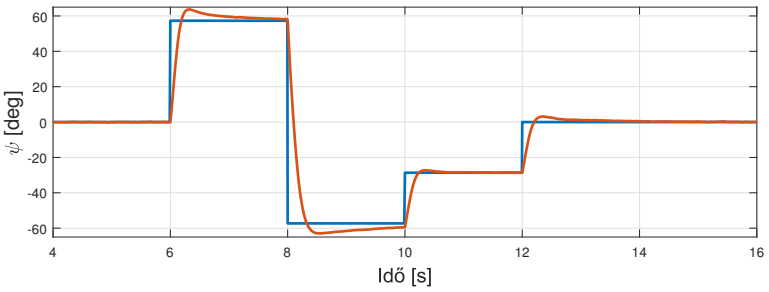

5.21. ábra. A z pozíció yaw trajektóriára




5.22. ábra. A $\phi$ szög yaw trajektóriára




5.23. ábra. A $\theta$ szög yaw trajektóriára




5.24. ábra. A $\psi$ szög yaw trajektóriára

# 6. fejezet

# LPV modellezés

Mivel a gain scheduled PID esetében a választott állapotváltozók értékétől függnek a szabályozó együtthatói, ezért a háló felbontásának növelésével jelentősen megnőhet a számítási idő. A szabályozás javításához a következő lépést egy LPV (*Linear Parameter-Varying*, lineáris változó paraméterű) technika alkalmazása jelenti, ami jelentősen leegyszerűsíti a hangolási folyamatot, így gyorsítja a tervezést.

Célszerű választás a Thomas T.R. van de Wiel, Tóth Roland, és Vsevolod I. Kiriouchine által kifejlesztett változó paraméterű szétcsatoláson alapuló módszer [2]. A technika alapja, hogy a rendszeregyenletekből analitikusan képezhető az átviteli függvény mátrix. Ehhez olyan szétcsatoló szűrő választható, ami a rendszert kettős integrátorokra redukálja. Ezekhez már egyszerűen behangolhatók a szükséges PID szabályozók. A 6.1 és 6.2 fejezetek ennek a módszernek a bemutatásával foglalkoznak, a továbbiak pedig az elkészült modell Simulink-implementációjával.

## 6.1. A rendszeregyenletek szétcsatolása

A szétcsatolás előtt, a jobb érthetőség érdekében célszerű újra felírni a rendszeregyenleteket. Az állapotváltozók a korábbiakhoz hasonlóan írhatók fel, a bemeneteknek azonban célszerű nem az egyes rotorok felhajtóerejét, hanem az eredő erőt, illetve nyomatékot megválasztani. Az eltérés oka, hogy a tervezési folyamatot jelentősen elbonyolítaná a négy dinamika külön-külön kezelése. A számítás végén kapott átviteli függvényekből pedig már (amennyiben szükséges) egyszerűen átszámíthatóak a kívánt felhajtóerők, így a szabályozás nem veszít a pontosságából.

$$\mathbf{x} = \begin{bmatrix} p_x & p_y & p_z & v_x & v_y & v_z & \phi & \theta & \psi & p & q & r \end{bmatrix}^T \tag{6.1}$$

$$\mathbf{u} = \begin{bmatrix} T & \tau_\phi & \tau_\theta & \tau_\psi \end{bmatrix}^T \tag{6.2}$$

A 2. fejezetben levezetett differenciálegyenlet rendszer felírható vektoros alakban is, ez a kiindulópont a levezetés további lépéseihez:

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{p}_x \\ \dot{p}_y \\ \dot{p}_z \\ \dot{v}_x \\ \dot{v}_y \\ \dot{v}_z \\ \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \\ \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} v_x \\ v_y \\ v_z \\ (\sin\phi \cdot \sin\psi + \cos\phi \cdot \cos\psi \cdot \sin\theta) \cdot \frac{T}{m} \\ (\cos\phi \cdot \sin\psi \cdot \sin\theta - \cos\psi \cdot \sin\phi) \cdot \frac{T}{m} \\ \cos\phi \cdot \cos\theta \cdot \frac{T}{m} - g \\ p + \sin\phi \cdot \operatorname{tg}\theta \cdot q + \cos\phi \cdot \operatorname{tg}\theta \cdot r \\ \cos\phi \cdot q - \sin\phi \cdot r \\ \frac{\sin\phi}{\cos\theta} \cdot q + \frac{\cos\phi}{\cos\theta} \cdot r \\ \frac{J_y - J_z}{J_x} \cdot q \cdot r + \frac{\tau_\phi}{J_x} \\ \frac{J_z - J_x}{J_y} \cdot p \cdot r + \frac{\tau_\theta}{J_y} \\ \frac{J_x - J_y}{J_z} \cdot p \cdot q + \frac{\tau_\psi}{J_z} \end{bmatrix} \tag{6.3}$$

A linearizálás előtt szükséges megválasztani a vizsgált általános munkapontban az állapotváltozók, bemenetek, és kimenetek értékét. A numerikus linearizáláshoz hasonlóan itt is a lebegés fenntartása a cél.

$$\mathbf{x}_0 = \begin{bmatrix} p_{x,0} & p_{y,0} & p_{z,0} & 0 & 0 & 0 & \phi_0 & \theta_0 & \psi_0 & 0 & 0 & 0 \end{bmatrix}^T \tag{6.4}$$

$$\mathbf{u}_0 = \begin{bmatrix} \frac{m \cdot g}{\cos\phi \cdot \cos\theta} & 0 & 0 & 0 \end{bmatrix}^T \tag{6.5}$$

$$\mathbf{y} = \begin{bmatrix} p_z & \phi & \theta & \psi \end{bmatrix}^T \tag{6.6}$$

A nemlineáris differenciálegyenletek analitikusan a munkapont körüli Taylor kifejtéssel linearizálhatók. A megjelenő $f(\mathbf{x}_0, \mathbf{u}_0)$ tag zérusnak választható, mivel a fő cél az állapotváltozók deriváltjainak értékeinek zérus körül tartása.

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}) \approx f(\mathbf{x}_0, \mathbf{u}_0) + \mathbf{A} \cdot (\mathbf{x} - \mathbf{x}_0) + \mathbf{B} \cdot (\mathbf{u} - \mathbf{u}_0) \tag{6.7}$$

$$\mathbf{A} = \left.\frac{\partial f(\mathbf{x}, \mathbf{u})}{\partial \mathbf{x}}\right|_{\mathbf{x}=\mathbf{x}_0, \mathbf{u}=\mathbf{u}_0} \qquad \mathbf{B} = \left.\frac{\partial f(\mathbf{x}, \mathbf{u})}{\partial \mathbf{u}}\right|_{\mathbf{x}=\mathbf{x}_0, \mathbf{u}=\mathbf{u}_0} \tag{6.8}$$

A Matlab szimbolikus változók deklarálásával képes a szükséges mátrixok meghatározására.

$$\mathbf{A} = \begin{bmatrix} \mathbf{0}_{3x3} & \mathbf{I}_{3x3} & \mathbf{0}_{3x3} & \mathbf{0}_{3x3} \\ \mathbf{0}_{3x3} & \mathbf{0}_{3x3} & \mathbf{A}_{23} & \mathbf{0}_{3x3} \\ \mathbf{0}_{3x3} & \mathbf{0}_{3x3} & \mathbf{0}_{3x3} & \mathbf{A}_{34} \\ \mathbf{0}_{3x3} & \mathbf{0}_{3x3} & \mathbf{0}_{3x3} & \mathbf{0}_{3x3} \end{bmatrix} \tag{6.9}$$

$$\mathbf{A}_{34} = \begin{bmatrix} 1 & \sin\phi_0 \operatorname{tg}\theta_0 & \cos\phi_0 \operatorname{tg}\theta_0 \\ 0 & \cos\phi_0 & -\sin\phi_0 \\ 0 & \frac{\sin\phi_0}{\cos\theta_0} & \frac{\cos\phi_0}{\cos\theta_0} \end{bmatrix} \tag{6.10}$$

$$\mathbf{A}_{23} = \begin{bmatrix} g\frac{c_\phi \cdot s_\psi - c_\psi \cdot s_\phi \cdot s_\theta}{c_\phi \cdot c_\theta} & g\frac{c_\phi \cdot c_\psi \cdot c_\theta}{c_\phi \cdot c_\theta} & g\frac{c_\psi \cdot s_\phi - c_\phi \cdot s_\psi \cdot s_\theta}{c_\phi \cdot c_\theta} \\ -g\frac{c_\phi \cdot c_\psi + s_\phi \cdot s_\psi \cdot s_\theta}{c_\phi \cdot c_\theta} & g\frac{c_\phi \cdot c_\theta \cdot s_\psi}{c_\phi \cdot c_\theta} & g\frac{s_\phi \cdot s_\psi + c_\phi \cdot c_\psi \cdot s_\theta}{c_\phi \cdot c_\theta} \\ -g\frac{c_\theta \cdot s_\phi}{c_\phi \cdot c_\theta} & -g\frac{c_\phi \cdot s_\theta}{c_\phi \cdot c_\theta} & 0 \end{bmatrix} \tag{6.11}$$

$$\mathbf{B} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \frac{\sin\phi_0 \cdot \sin\psi_0 + \cos\phi_0 \cdot \cos\psi_0 \cdot \sin\theta_0}{m} & 0 & 0 & 0 \\ -\frac{\sin\phi_0 \cdot \cos\psi_0 - \cos\phi_0 \cdot \sin\psi_0 \cdot \sin\theta_0}{m} & 0 & 0 & 0 \\ \frac{\cos\phi_0 \cdot \cos\theta_0}{m} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & \frac{1}{J_x} & 0 & 0 \\ 0 & 0 & \frac{1}{J_y} & 0 \\ 0 & 0 & 0 & \frac{1}{J_z} \end{bmatrix} \tag{6.12}$$

Az állapottér-modell felírható a szokásos alakban. A $\mathbf{C}$ mátrix úgy írandó fel, hogy kiválassza a szükséges kimeneteket az állapotváltozók közül. A $\mathbf{D}$ mátrix értéke zérus, így az egyenletben nem jelenik meg.

$$\dot{\mathbf{x}} = \mathbf{A} \cdot \mathbf{x} + \mathbf{B} \cdot \mathbf{u} \tag{6.13}$$

$$\mathbf{y} = \mathbf{C} \cdot \mathbf{x} \tag{6.14}$$

$$\mathbf{C} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \tag{6.15}$$

Az átviteli függvény mátrix az állapottér-modell mátrixaiból, Laplace-transzformáció után felírható a következőképp:

$$\mathbf{G}(s) = \frac{\mathbf{X}(s)}{\mathbf{U}(s)} = \mathbf{C} \cdot (s \cdot \mathbf{I}_{12x12} - \mathbf{A})^{-1} \cdot \mathbf{B} \tag{6.16}$$

$$\mathbf{G}(s) = \begin{bmatrix} \frac{\cos\phi_0 \cdot \cos\theta_0}{m \cdot s^2} & -\frac{g \cdot \mathrm{tg}\,\phi_0}{J_x \cdot s^4} & -\frac{g \cdot \mathrm{tg}\,\theta_0}{\cos\phi_0 \cdot J_y \cdot s^4} & 0 \\ 0 & \frac{1}{J_x \cdot s^2} & \frac{\sin\phi_0 \cdot \mathrm{tg}\,\theta_0}{J_y \cdot s^2} & \frac{\cos\phi_0 \cdot \mathrm{tg}\,\theta_0}{J_z \cdot s^2} \\ 0 & 0 & \frac{\cos\phi_0}{J_y \cdot s^2} & -\frac{\sin\phi_0}{J_z \cdot s^2} \\ 0 & 0 & \frac{\sin\phi_0}{\cos\theta_0 \cdot J_y \cdot s^2} & \frac{\cos\phi_0}{\cos\theta_0 \cdot J_z \cdot s^2} \end{bmatrix} \tag{6.17}$$

A mátrixon két fő jellemző figyelhető meg, az egyik az átcsatolás, a másik a szétbonthatóság. Figyelembe véve, hogy az oszlopok jelölik a bemeneteket, a sorok pedig a kimeneteket, látható az átcsatolás az alrendszerek között:

| **G** | T | $\tau_\phi$ | $\tau_\theta$ | $\tau_\psi$ |
|---|---|---|---|---|
| z | × | × | × | |
| $\phi$ | | × | × | × |
| $\theta$ | | | × | × |
| $\psi$ | | | × | × |

Az átviteli függvény mátrix szétbontása megtehető úgy, hogy a kettős integrátorok szorzóit egy $\mathbf{M_1}$ mátrix, a négyes integrátorokét pedig egy $\mathbf{M_2}$ mátrix jelöli.

$$\mathbf{G}_p(s) = \mathbf{M}_1(\phi_0, \theta_0) \cdot \frac{1}{s^2} + \mathbf{M}_2(\phi_0, \theta_0) \cdot \frac{1}{s^4} \tag{6.18}$$

Ezután egy olyan $\mathbf{T_p}$ mátrixot kell keresni, amit a rendszer elé kötve az eredő átviteli függvényben kizárólag kettős integrátorok szerepelnek. Ez a szétcsatoló szűrő, ami előáll arányos tagok, és kettős integrátorok összegeként:

$$\mathbf{T}_p(s) = \mathbf{T}_1(\phi_0, \theta_0) + \mathbf{T}_2(\phi_0, \theta_0) \cdot \frac{1}{s^2} \tag{6.19}$$

$$\mathbf{G}_p(s) \cdot \mathbf{T}_p(s) = \mathbf{I}_{4x4} \cdot \frac{1}{s^2} \tag{6.20}$$

Az utolsó lépés a mátrixegyenletek felírása, és a szűrő tagjainak kiszámítása. Ehhez a (6.20) egyenletbe behelyettesítve egyenletrendszert írunk fel, majd a mátrixokra vonatkozó ismert azonosságok segítségével megoldjuk azt.

$$\begin{bmatrix} \mathbf{M}_1 & \mathbf{0}_{4x4} \\ \mathbf{M}_2 & \mathbf{M}_1 \\ \mathbf{0}_{4x4} & \mathbf{M}_2 \end{bmatrix} \begin{bmatrix} \mathbf{T}_1 \\ \mathbf{T}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{I}_{4x4} \\ \mathbf{0}_{4x4} \\ \mathbf{0}_{4x4} \end{bmatrix} \rightarrow \widehat{\mathbf{M}}_p \cdot \widehat{\mathbf{T}}_p = \widehat{\mathbf{S}} \tag{6.21}$$

$$\widehat{\mathbf{T}}_p = \left( \widehat{\mathbf{M}}_p^T \cdot \widehat{\mathbf{M}}_p \right)^{-1} \cdot \widehat{\mathbf{M}}_p^T \cdot \widehat{\mathbf{S}} \tag{6.22}$$

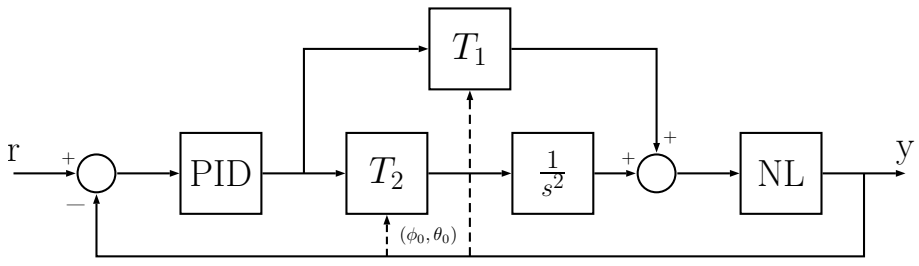
A $\widehat{\mathbf{T}}_p$ mátrixot "félbevágva" a 4. sor alatt, megkaphatók a $\mathbf{T_1}$ és $\mathbf{T_2}$ mátrixok, amik felhasználhatók Simulinkben a szűrő modellbe kötésére.

$$\mathbf{T}_1(s) = \begin{bmatrix} \frac{m}{\cos\phi_0 \cdot \cos\theta_0} & 0 & 0 & 0 \\ 0 & J_x & 0 & -J_x \cdot \sin\theta_0 \\ 0 & 0 & J_y \cdot \cos\phi_0 & J_y \cdot \cos\theta_0 \cdot \sin\phi_0 \\ 0 & 0 & -J_z \cdot \sin\phi_0 & J_z \cdot \cos\phi_0 \cdot \cos\theta_0 \end{bmatrix} \tag{6.23}$$

$$\mathbf{T}_2(s) = \begin{bmatrix} 0 & \frac{g \cdot m \cdot \mathrm{tg}\,\phi_0}{\cos\phi_0 \cdot \cos\theta_0} & \frac{g \cdot m \cdot \mathrm{tg}\,\theta_0}{\cos\phi_0 \cdot \cos\theta_0} & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{6.24}$$

## 6.2. Szabályozótervezés

A PID szabályozók behangolása a 4. fejezetben látotthoz hasonlóan, könnyedén elvégezhető, hiszen a szabályozni kívánt rendszerek itt is egyszerű kettős integrátorok. A gain scheduling-al ellentétben a tervezés ezen szakasza paraméterektől független. Fontos, hogy a szabályozókörben a szétcsatolás közvetlenül a nemlineáris modell előtt helyezkedik el (*6.1. ábra*), hiszen így valósul meg a dinamikai alrendszerek szétbontása.




6.1. ábra. A szétcsatolt LPV modell

A szabályozó behangolása most is a Matlab `pidtune` algoritmusával történik. A függvény három bemenetű: az első a szabályozott rendszer, a második a szabályozó típusa, a harmadik pedig a kívánt vágási körfrekvencia ($\omega_c$).

$$G(s) = \frac{1}{s^2} \tag{6.25}$$

A felhasznált irodalom [2] által használt kvadrotor paraméterek nagyságrendileg azonosak az általunk használtakkal, így a szabályozótervezéskor célszerű először a szerzők által javasolt feltételekre kipróbálni a modellt. A választott szabályozó PDF (proporcionális tag, és szűrővel ellátott deriváló tag) a három szöghelyzetre, a magasságra pedig PIDF (integráló tagot is tartalmaz). A vágási körfrekvenciák értékei a következők:

$$\omega_{c,z} = 7[Hz] \quad \omega_{c,\phi} = 50[Hz] \quad \omega_{c,\theta} = 40[Hz] \quad \omega_{c,\psi} = 10[Hz] \tag{6.26}$$

A PIDF, illetve a PDF szabályozók felépítése a korábban felhasználtakkal megegyezik, a Matlab egységes jelölése alapján:

$$C_{PIDF} = K_p + K_i \cdot \frac{1}{s} + K_d \cdot \frac{s}{T_f \cdot s + 1} \tag{6.27}$$

$$C_{PDF} = K_p + K_d \cdot \frac{s}{T_f \cdot s + 1} \tag{6.28}$$

A behangolt szabályozók együtthatói az egyes dinamikákra az alábbiakban láthatók. A Simulink implementációban ezek használandók a PID blokkok megadható értékeiként.

$$p_z : K_p = 9,1 \quad K_i = 2,98 \quad K_d = 6,82 \quad T_f = 0,0262 \tag{6.29}$$

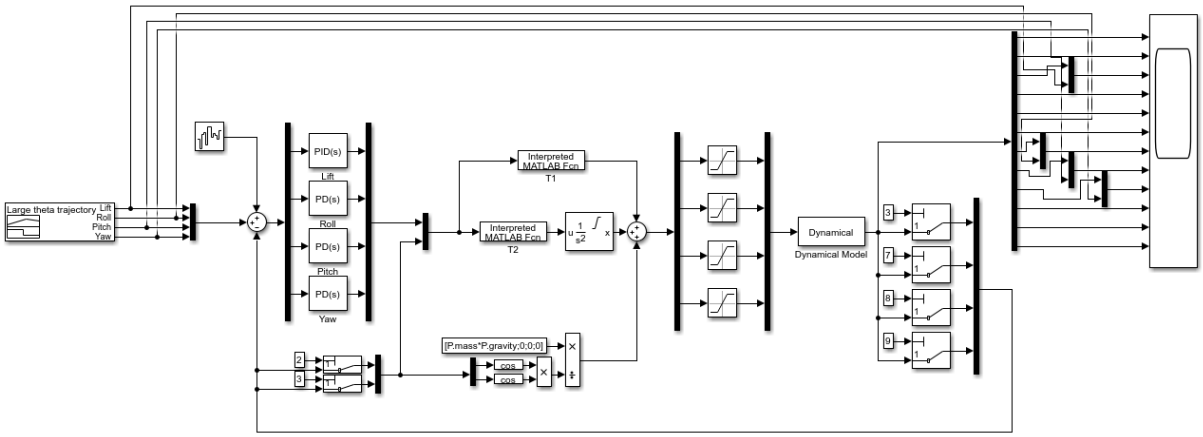
$$\phi : K_p = 524 \quad K_d = 48,8 \quad T_f = 0,000175 \tag{6.30}$$

$$\theta : K_p = 335 \quad K_d = 39 \quad T_f = 0,000219 \tag{6.31}$$

$$\psi : K_p = 21 \quad K_d = 9,76 \quad T_f = 0,000875 \tag{6.32}$$

## 6.3. Simulink implementáció

A Simulink-modell felépítése az előző két fejezetben látottakhoz hasonlóan történik (*6.2. ábra*). Az első elem a jelgenerátor, ami az 5.3. fejezetben leírtaknak megfelelően működik. Ezt fehér zajforrás figyelembevételével a hibajelképzés követi. A behangolt szabályozók ebből párhuzamosan működve létrehozzák a szétcsatoló szűrő bemenetét. A szűrő mátrixai Matlab függvényekben vannak elhelyezve, a kettős integrátor kimenete pedig ±2-re van szaturálva. A szűrő kimenetéhez adódik hozzá a trim bemenet, ami biztosítja a hover állapotot.




6.2. ábra. A szétcsatolt LPV szabályozókör Simulinkben

A nemlineáris rendszer bemenetét először szaturálni kell, hogy a kvadrotor a valóságnak megfelelően működjön. Mivel a szűrő felépítése olyan, hogy az egyes rotorokra eső felhajtóerők közvetlenül nem érhetők el, így az eredő erőre, illetve nyomatékokra adhatók meg maximális és minimális értékek. Ezek rotorparaméterektől függő, mérhető adatok. A szimuláció során használt értékek:

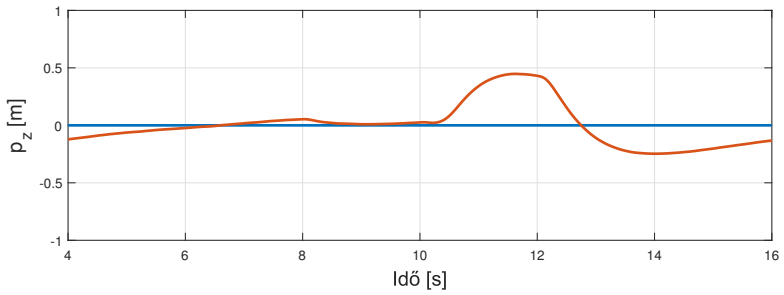
$$T_{max} = 105[N] \quad \tau_{\phi,max} = 15[Nm] \quad \tau_{\theta,max} = 15[Nm] \quad \tau_{\psi,max} = 5[Nm] \tag{6.33}$$

A nemlineáris modell kimeneteit ezután Scope-on lehet megjeleníteni, a megfelelő állapotváltozókat pedig visszacsatoljuk.

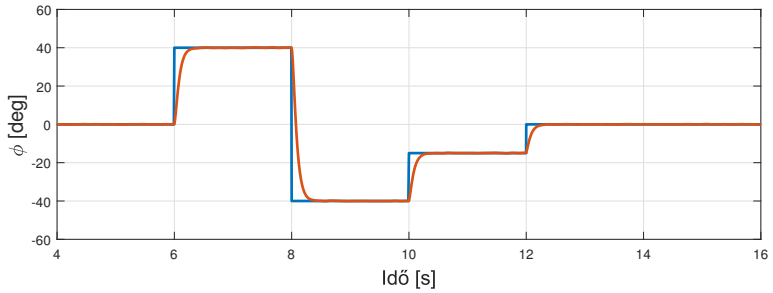
## 6.4. Futtatási eredmények

A szabályozás működésének ellenőrzésére lefuttatjuk a Simulink szimulációt. Négy, számunkra érdekes jelet vizsgálunk: a $p_z$ pozíciót, illetve a $\phi$, $\theta$, és $\psi$ szögeket.

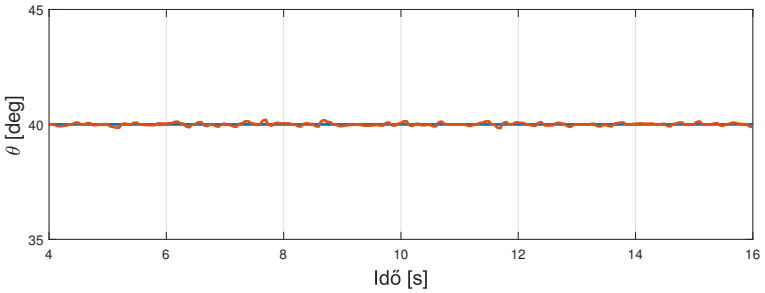
Az elsőként vizsgált referenciajel a nagy $\phi$ trajektória, majd a nagy $\theta$ trajektória, végül a nagy $\psi$ trajektória. Az eredmények pontos elemzése a 8. fejezetben olvasható. A kapott válaszok a következő három oldalon láthatók.



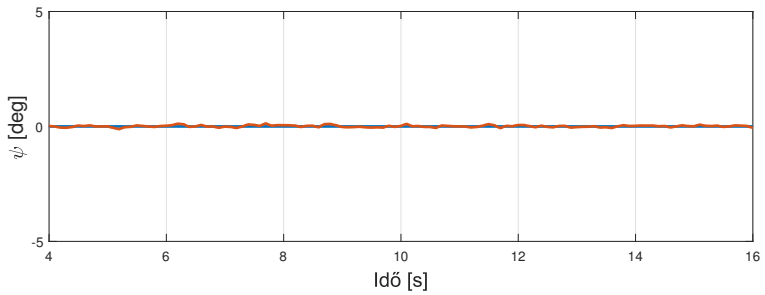

6.3. ábra. A z pozíció roll trajektóriára



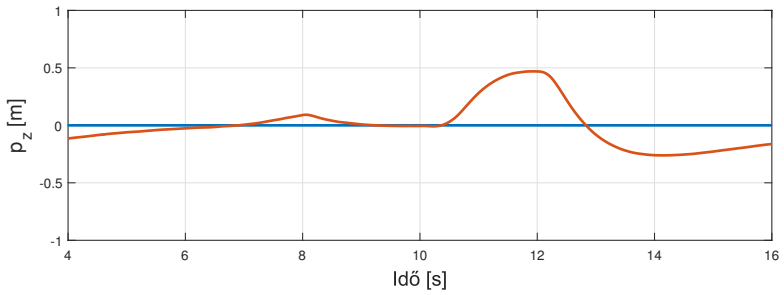

6.4. ábra. A $\phi$ szög roll trajektóriára



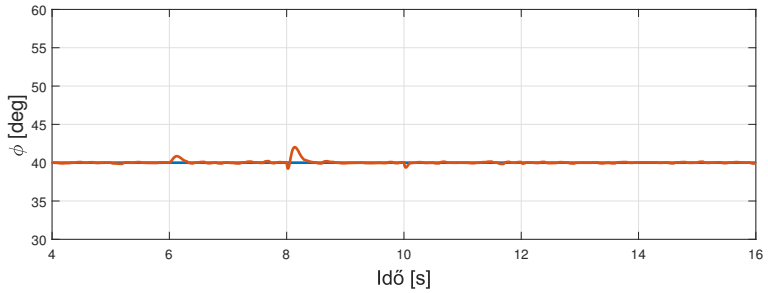

6.5. ábra. A $\theta$ szög roll trajektóriára



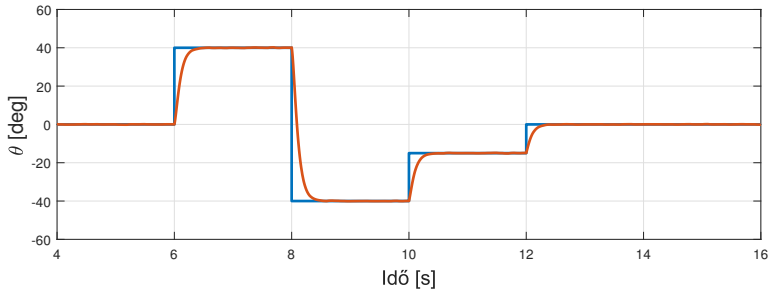

6.6. ábra. A $\psi$ szög roll trajektóriára



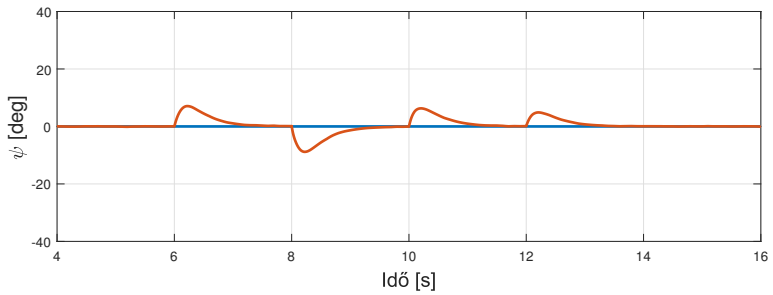

6.7. ábra. A z pozíció pitch trajektóriára



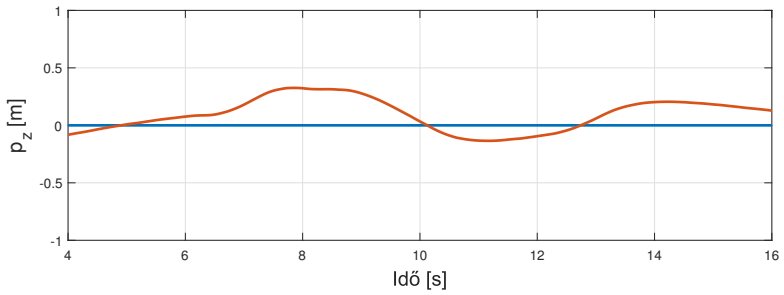

6.8. ábra. A $\phi$ szög pitch trajektóriára



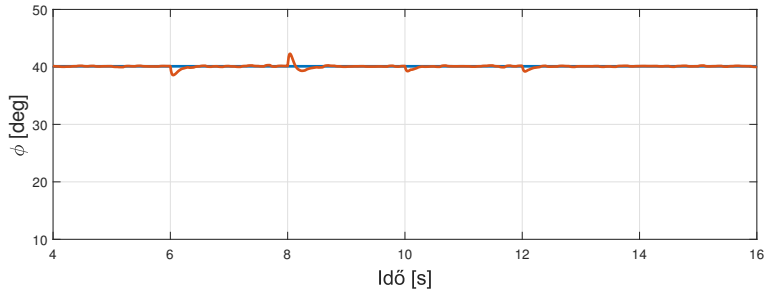

6.9. ábra. A $\theta$ szög pitch trajektóriára



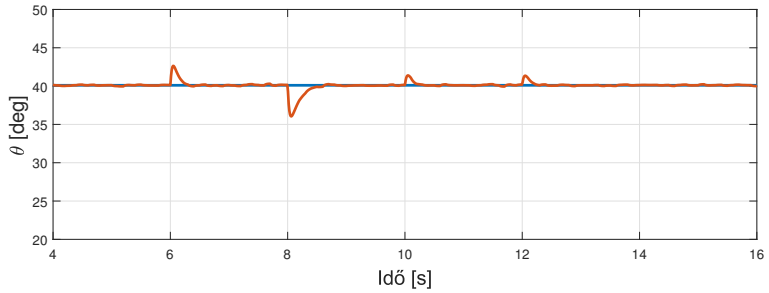

6.10. ábra. A $\psi$ szög pitch trajektóriára



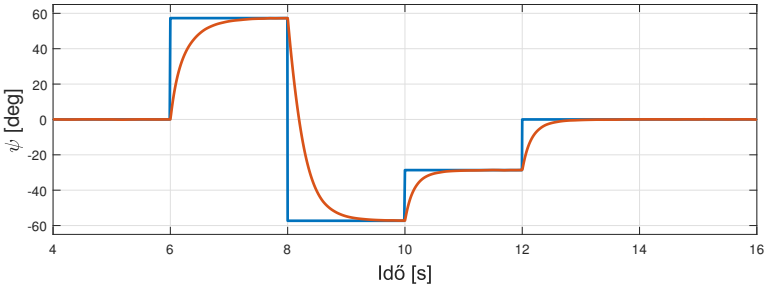

6.11. ábra. A z pozíció yaw trajektóriára




6.12. ábra. A $\phi$ szög yaw trajektóriára




6.13. ábra. A $\theta$ szög yaw trajektóriára




6.14. ábra. A $\psi$ szög yaw trajektóriára

# 7. fejezet

# Továbbfejlesztési lehetőségek

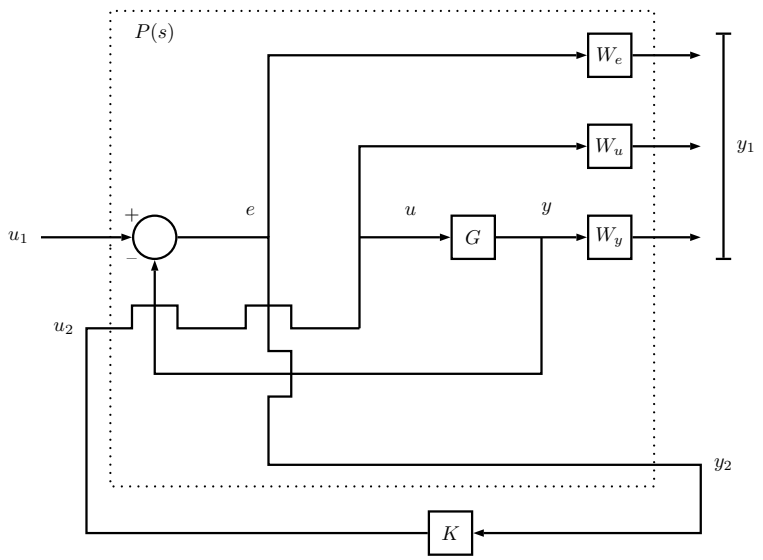
A SISO szabályozáson túllépve jobb eredmények érhetők el, ha MIMO szabályozót használunk. Erre nyújt lehetőséget a $H_\infty$ módszer. A téma ugyan nem része a szakdolgozat kiírásnak, de fontosnak tartottam röviden bemutatni, mivel a már ismertetett szétcsatoláson túl ígéretes lehetőségeket kínál a nemlineáris rendszerek szabályozásának továbbfejlesztésre.

A módszerrel elkészült egy szimuláció, aminek az eredményeit a fejezetben bemutatom. A szabályozók behangolása nem optimalizált, mivel a modell kisebb bővítésekkel közvetlenül továbbfejleszthető LPV kontrollerré, aminek a helyes behangolására meghaladja a dolgozat kereteit, de a jövőben szeretném megcsinálni.

## $H_\infty$ módszer

A $H_\infty$ módszer lényege, hogy megadott zavaró jelekről általunk definiált performancia kimenetekre nézzük a rendszer átvitelét. Ennek a $H_\infty$ normáját akarjuk minimalizálni dinamikus szabályozóval. Ez a norma SISO rendszerre a Bode amplitúdó diagram legmagasabb értéke (worst case gain), MIMO rendszerre az amplitúdó görbesereg burkológörbéjének maximális pontja.

A zárt körre vonatkozó tervezési előírásokat súlyfüggvények segítségével fogalmazzuk meg. A Matlab fő eszköze ennek a megvalósítására az augw függvény [6]. Ennek négy rendszerre van szüksége bemeneteként, ezek a hover állapotban linearizált MIMO rendszermodell, illetve három "büntető" átviteli függvény mátrix. Ezek a hibajelre ($\mathbf{W}_e(s)$), a bemenő jelre ($\mathbf{W}_u(s)$), illetve a kimenő jelre ($\mathbf{W}_y(s)$) vonatkoznak.




7.1. ábra. Az augw függvény által igényelt modell

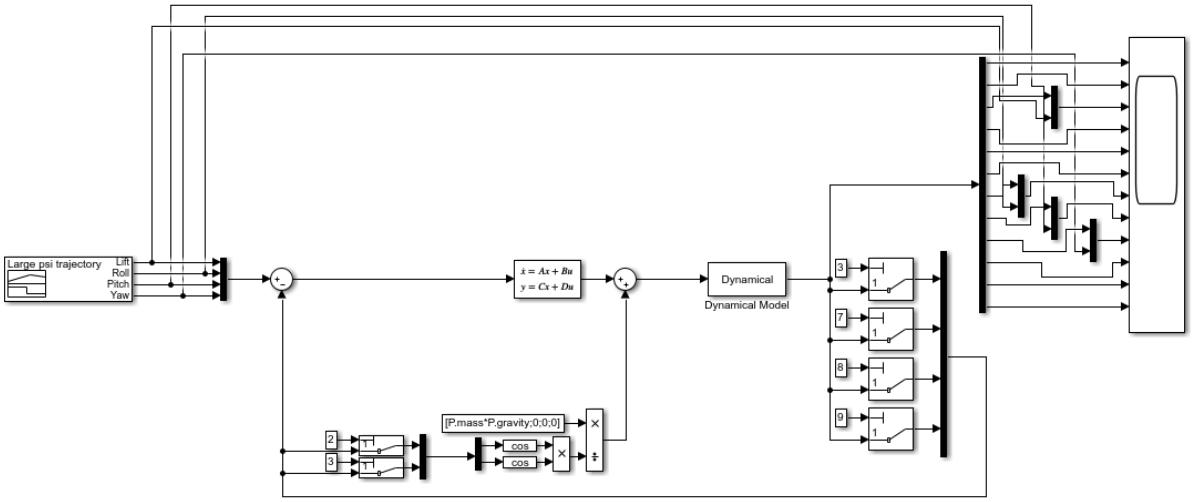
A kimenő jelet nem büntetjük (tehát $\mathbf{W}_y(s)$ üres mátrix). Figyelembe véve, hogy az első bemenet egy nagyságrenddel általában nagyobb, mint a másik három, a következőképp célszerű definiálni a mátrixokat:

$$\mathbf{W}_e(s) = \frac{s+100}{100s+1} \cdot \mathbf{I}_{4x4} \quad \mathbf{W}_u(s) = \begin{bmatrix} 0.1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{7.1}$$

Az `augw` kimenete a P(s) rendszer, amit alapul véve a szabályozó paramétereinek hangolása a Matlab `hinfsyn` parancsával végezhető el. Az így kiszámolt kontrollerrel kapott zárt kör $H_\infty$ normájának értéke 0,4843, ami kisebb, mint 1, ami azt jelenti, hogy a rendszerre kapcsolt zavaró jelek elnyomása megtörténik a performanciakimenetekre.

A $H_\infty$ módszer kibővíthető LPV szabályozássá, amennyiben a gain scheduling eljáráshoz hasonló paraméter-rács segítségével minden munkapontra linearizáljuk a modellt, és bázisfüggvények segítségével adjuk meg a szabályozó együtthatóit.

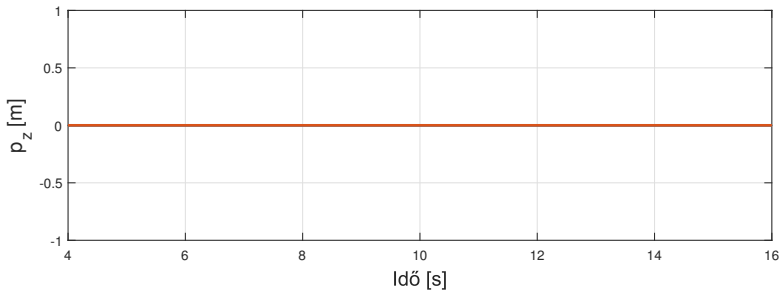
# Simulink implementáció



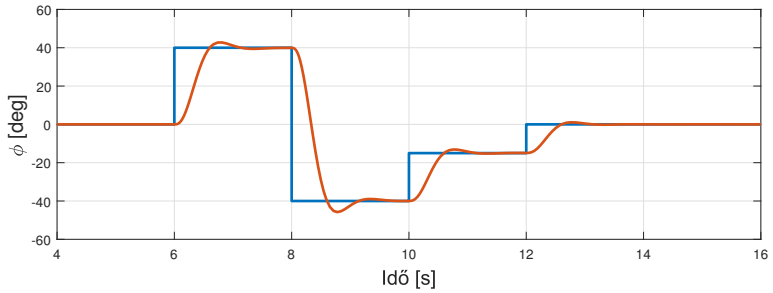

7.2. ábra. A $H_\infty$ kontroller Simulink-modellje

A paramétereket a Simulink állapottér-modell blokkjában helyezzük el, aminek a kimenetéhez visszacsatoljuk a paraméterfüggő trim bemenetet. A modell gerjesztését ugyanazokkal a trajektóriákkal végeztem, amiket az előző két fejezetben is használtam, a válaszok grafikonon kirajzolva láthatók a következő három oldalon.

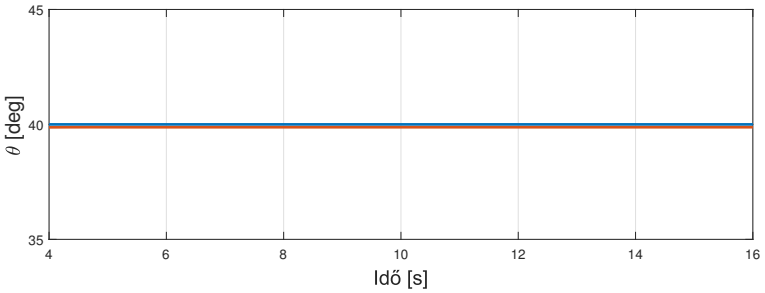
Az eredményeken látható a gain scheduled és az LPV szabályozáshoz hasonlóan a dinamika szétcsatolódása. Ugyan a válaszban van maradó hiba, de ez megszüntethető a paraméterek optimalizálásával.



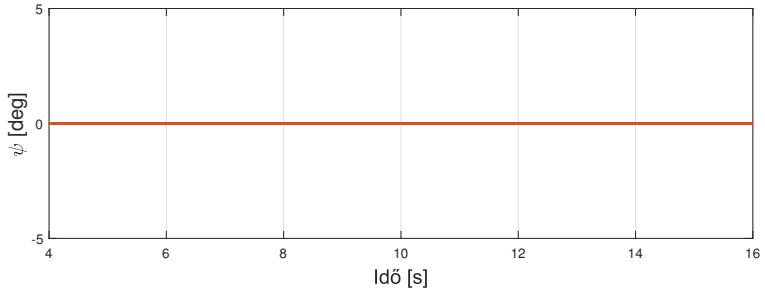

7.3. ábra. A z pozíció roll trajektóriára



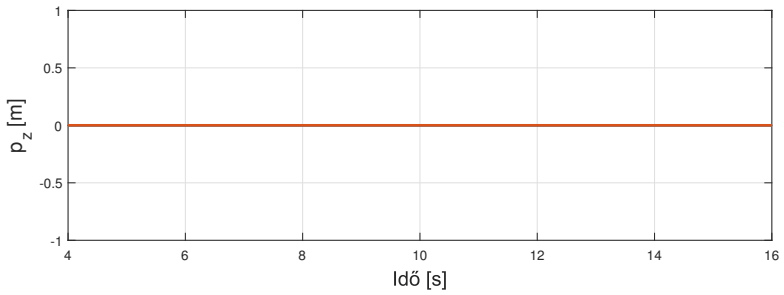

7.4. ábra. A $\phi$ szög roll trajektóriára



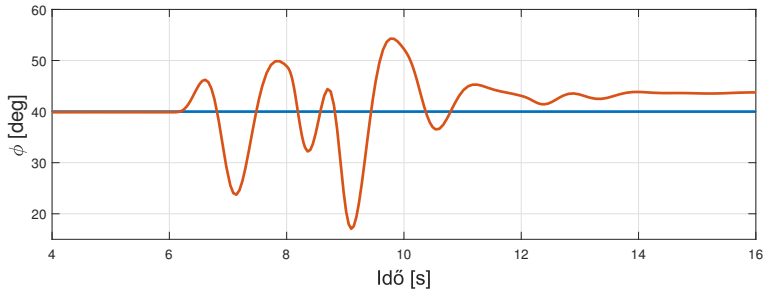

7.5. ábra. A $\theta$ szög roll trajektóriára



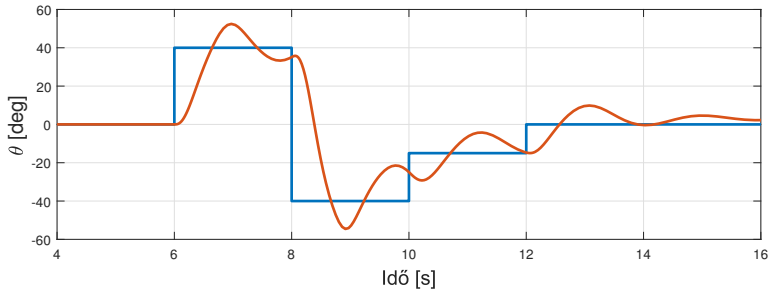

7.6. ábra. A $\psi$ szög roll trajektóriára



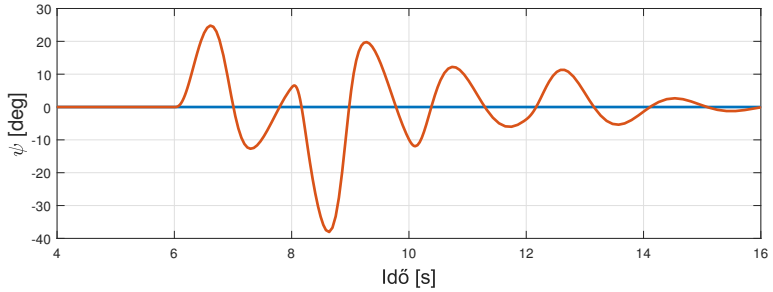

7.7. ábra. A z pozíció pitch trajektóriára



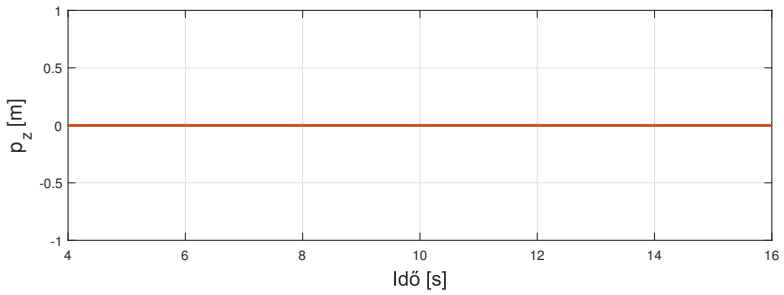

7.8. ábra. A $\phi$ szög pitch trajektóriára



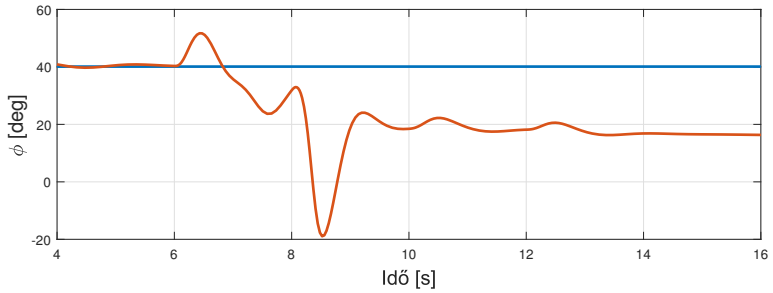

7.9. ábra. A $\theta$ szög pitch trajektóriára



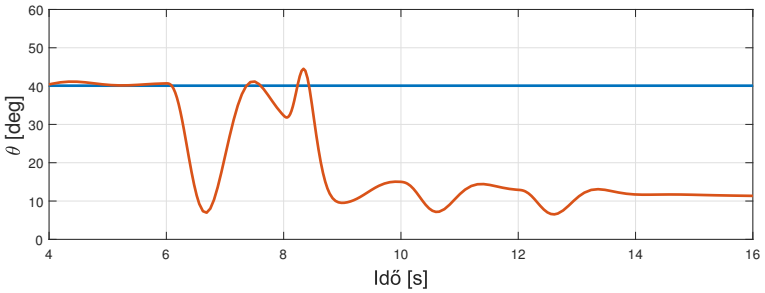

7.10. ábra. A $\psi$ szög pitch trajektóriára



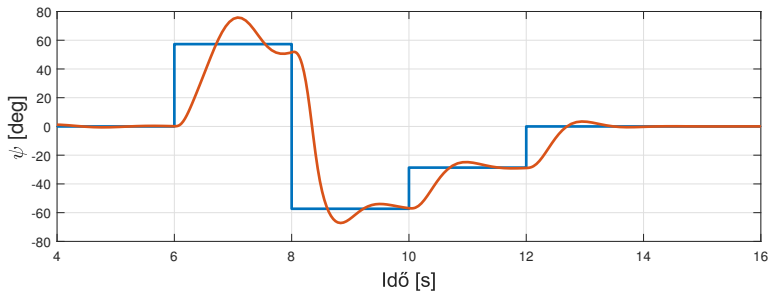

7.11. ábra. A z pozíció yaw trajektóriára




7.12. ábra. A $\phi$ szög yaw trajektóriára




7.13. ábra. A $\theta$ szög yaw trajektóriára




7.14. ábra. A $\psi$ szög yaw trajektóriára

# 8. fejezet

# Összefoglalás

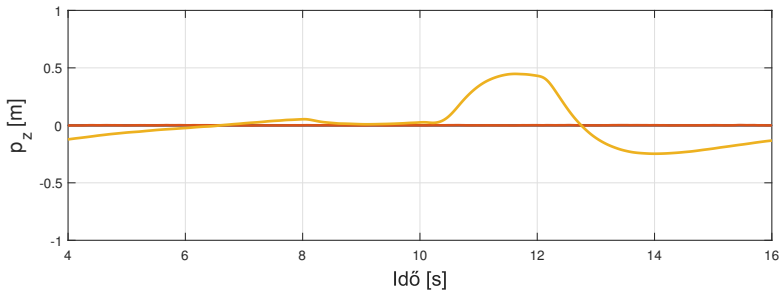
A szimulációs eredmények mindhárom előírt trajektóriára a fejezetben megtalálhatók. A grafikonokon összevetésre kerül a gain scheduled szabályozott rendszer válasza (GS, narancssárga folytonos vonal), illetve az LPV szabályozott rendszer válasza (LPV. citromsárga folytonos vonal).

A roll trajektória esetén látható, hogy mindkét szabályozó képes tartani a $\theta$ és $\psi$ szögeket. Az LPV esetében a magasság tartása csak kisebb kilengéssel lehetséges, így ilyen téren gyengébben teljesít. Azonban ha a $\phi$ szöget nézzük, jól látható, hogy a GS szabályozó túllendüléssel áll be a kívánt szöghelyzetre.

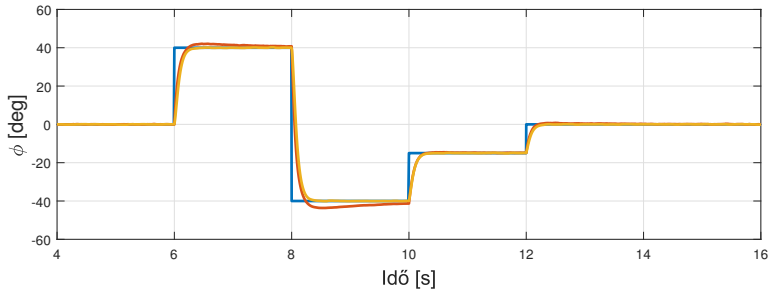
A pitch trajektóriánál már az átcsatolások számának növekedése miatt több grafikonon is eltérés látható. A magasság tartása hasonló eredményekkel történik, mint a roll trajektória esetén: a GS hiba nélkül, az LPV kis hibával veszi a kitérítést. A változtatott szög ($\theta$) esetében is hasonló képet látunk, az LPV túllendülés nélkül, a GS csak túllendüléssel áll be a kívánt értékre. A $\phi$ szögnél már jelentős különbség van a két módszer között, a GS szabályozó több, mint 15° kilendülést produkált ott, ahol az LPV csak 3-4°-ot. Ugyanez a jelenség megfigyelhető a $\psi$ szög esetén is, ahol a szétcsatolásból képződő szögkitérés a GS szabályozás esetén két-háromszoros az LPV-hez képest.

A yaw trajektóriánál a legszembetűnőbb a változás. A magasság és a szabályozott változó ($\psi$)) a másik két próbához hasonlóan működik, azonban a GS kontroller itt már néhány fokban mérhető túllendülést mutat. A $\phi$ és $\theta$ szögek esetén is látszik, hogy az LPV jelentősen képes csökkenteni a túllendüléseket, amik az egyik szabályozó esetében nem lépik át az 5°-ot, míg a másiknál akár 25° eltérés is lehet.

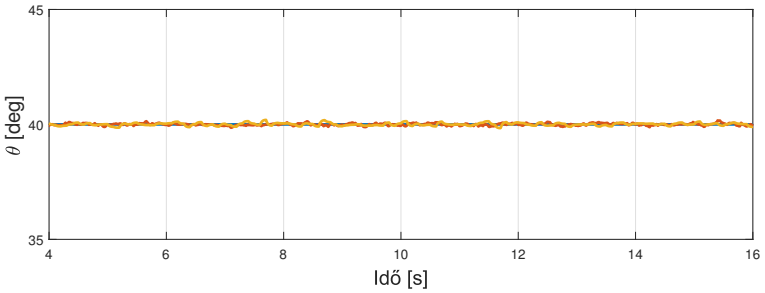
Az eltérésekre a magyarázat a szabályozás komplexitásában keresendő. Az LPV esetében a modell szétcsatolása után már a szabályozók könnyedén, gyors futásidővel behangolhatók a kettős integrátorokra, míg a gain scheduling-nál a bázisfüggvénytől, és a rácsozás felbontásától is függ a végső pontosság. Összességében mindkét megközelítés jobb eredményt ad a SISO LTI rendszerre tervezett PID szabályozóknál, állandó hiba nélkül és rövid válaszidővel képesek követni a referenciajelet. A GS és LPV összevetésében pedig a szöghelyzetekre az LPV, a magasságra a gain scheduling jelenti a jobb megoldást.



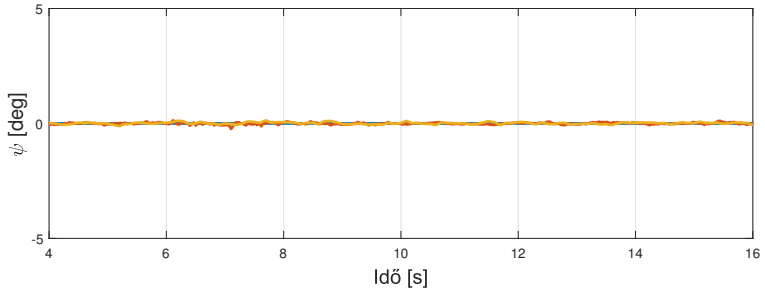

8.1. ábra. A z pozíció roll trajektóriára



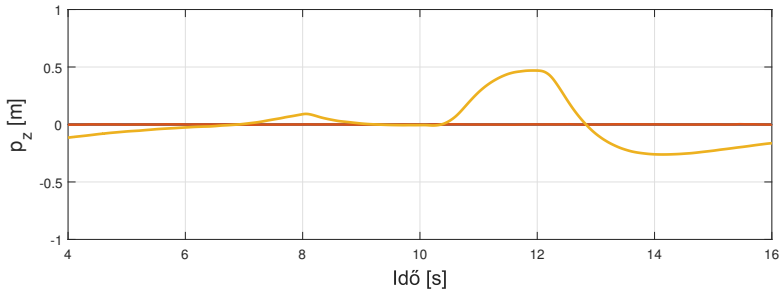

8.2. ábra. A $\phi$ szög roll trajektóriára



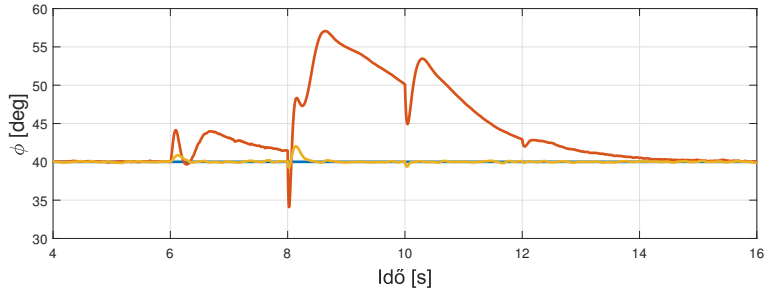

8.3. ábra. A $\theta$ szög roll trajektóriára



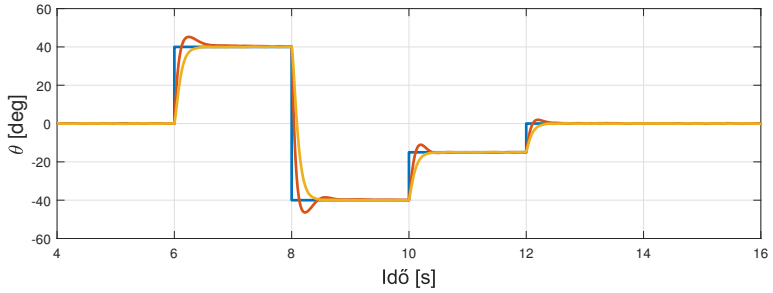

8.4. ábra. A $\psi$ szög roll trajektóriára



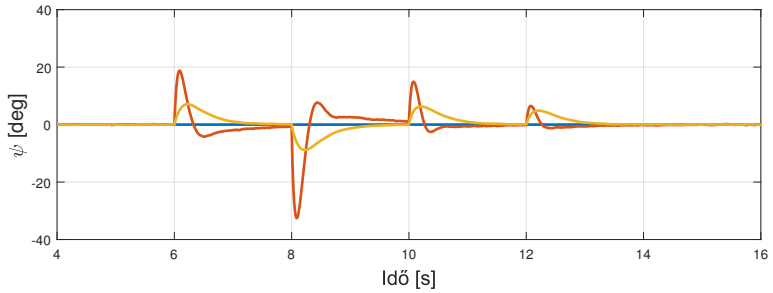

8.5. ábra. A z pozíció pitch trajektóriára



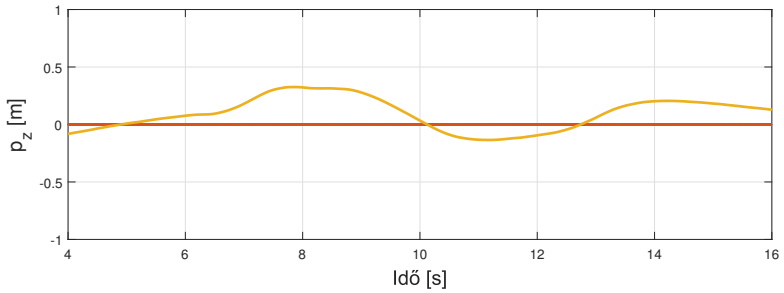

8.6. ábra. A $\phi$ szög pitch trajektóriára



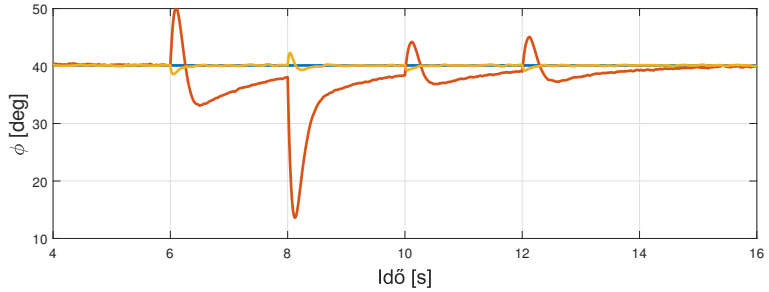

8.7. ábra. A $\theta$ szög pitch trajektóriára



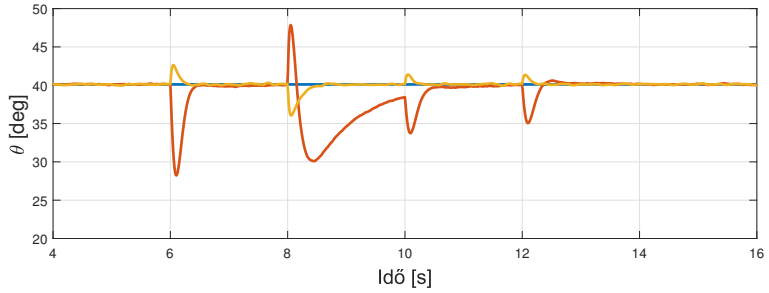

8.8. ábra. A $\psi$ szög pitch trajektóriára



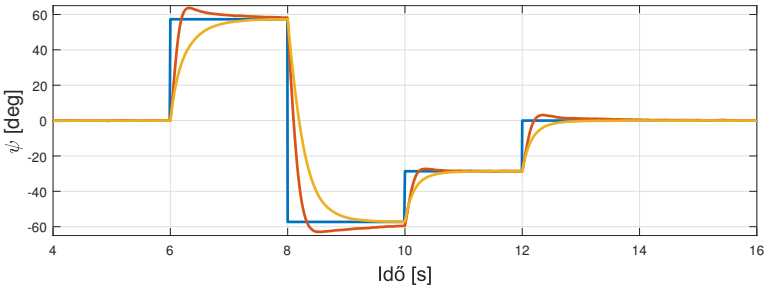

8.9. ábra. A z pozíció yaw trajektóriára




8.10. ábra. A $\phi$ szög yaw trajektóriára




8.11. ábra. A $\theta$ szög yaw trajektóriára




8.12. ábra. A $\psi$ szög yaw trajektóriára

# Irodalomjegyzék


[1] R. Beard, T. McLain: *Small Unmanned Aircraft: Theory and Practice.* Princeton University Press, 2012.

[2] van de Wiel, T., Tóth, R., Kiriouchine, V.: *Comparison of Parameter-Varying Decoupling Based Control Schemes for a Quadrotor.* Joint 9th IFAC Symposium on Robust Control Design and 2nd IFAC Workshop on Linear Parameter Varying Systems, Florianopolis, 2018.

[3] T. Luukkonen: *Modelling and control of quadcopter.* Aalto University School of Science, Espoo, 2011.

[4] MATLAB Documentation: *What Is an S-Function?* (2018. 12. 07.) https://www.mathworks.com/help/simulink/sfg/what-is-an-s-function.html

[5] Rugh, W. J., Shamma, J. S.: *Research on gain scheduling.* Automatica 36 (2000) 1401-1425

[6] MATLAB Documentation: *State-space plant augmentation for use in weighted mixed-sensitivity $H_\infty$ loopshaping design* (2018. 12. 07.) https://uk.mathworks.com/help/robust/ref/augw.html