Lantos Béla

BME Irányítástechnika és Informatika Tanszék

GÉPJÁRMŰ AUTOMATIKUS AKADÁLYELKERÜLŐ RENDSZER VALÓS IDEJŰ KÖRNYEZETBE INTEGRÁLÁSÁHOZ SZÜKSÉGES FELTÉTELEK ÉS A GYORS PROTOTÍPUS RENDSZEREKBEN VALÓ TESZTELHETŐSÉG VIZSGÁLATA MATLAB/SIMULINK, REAL TIME WORKSHOP ÉS dSPACE TARGET COMPILER KÖRNYEZETBEN. SZOFTVER ÉS DOKUMENTÁCIÓ

Tanulmány

Készült a RET 1.1 Járműforgalmi rendszerek modellezése és irányítása projekt keretében

Elektronikus Jármű és Járműirányítási Tudásközpont

Budapest, 2008. szeptember

Tartalmi összefoglaló

Műszaki alkalmazások az irányítástechnikában intenzíven építenek a Matlab és toolboxainak szolgáltatásaira. A Matlab licensz azonban költséges és a Matlab környezet nem biztosítja a valós idejű elvárások teljesíthetőségét. Két út kínálkozik ennek megkerülésére, az egyik Matlab licenszet nem igénylő stand-alone alkalmazások kifejlesztése host PC-re Windows vagy Linux alá, a másik pedig a Matlab alatt kifejlesztett szolgáltatás kifordítása beágyazott processzorokra vagy más gyors prototípus rendszerekre. Az előbbi több lehetőséget biztosít a toolboxok felhasználására és könnyebben bővíthető, de gyors rendszerek esetén nem képes a valós idejű elvárások kielégítésére, ezzel szemben az utóbbi gyors rendszerek valós idejű irányítására is képes, de csak a Matlab szolgáltatások egy erősen korlátozott szeletét használhatja és csakis a szabályozó Simulink modelljéből vezethető le.

A pályázat korábbi fázisában stand-alone programcsomagot hoztunk létre gépjármű ütközésmentes pályatervezésére és prediktív irányítására, amely a toolbox szolgáltatások bevonása érdekében nem épített a szabályozó Simulink modelljére, ellenben kihasználta Matlab Compiler szolgáltatásait és host PC-n futtatható, Matlab licenszet már nem igénylő stand-alone programot eredményezett.

A gyors valós idejű működéshez el kellett hagyni a PC-környezetet és speciális target rendszerekre kellett alapozni a megoldást. Ez az út azonban kizárta a Matlab Compiler használatát, és csakis a Simulink –> Real Time Workshop –> Target Compiler szekvenciában tette lehetővé valós idejű megoldás létrehozását. A célrendszerül (target) a Tudásközpont adottságainak megfelelően a dSPACE AutoBox rendszert választottuk. Súlyos korlátozó tényező volt, hogy a Simulink kizárja a toolboxok használatát, és bár a Simulink ú.n. embedded function (beágyazott függvény) blokkja bizonyos lehetőségeket megenged saját fejlesztésű függvények bevonására, a Real Time Workshop ezek körét tovább korlátozza A korlátozások szükségessé tették a korábbi algoritmusok átértékelését és továbbfejlesztését, a pályatervezés és irányítás Simulink alakra hozatalát, lényegében a teljes akadályelkerülési rendszer újratervezését Simulink alapon.

A kutatás első fázisában mintafeladatok keretében felderítettük a Simulink –> Real Time Workshop –> Target Compiler korlátozásait, majd a kutatás második fázisában a tapasztalatokra alapozva létrehoztuk az automatikus akadályelkerülés nemlineáris prediktív irányító rendszerének Simulinkre és beágyazott függvényekre alapozott megvalósítását, és elvégeztük részletes tesztelését először csak Simulink környezetben, majd a dSPACE AutoBox rendszeren. Eközben fokozatosan elimináltuk a szoftver környezet hiányosságait a hibaüzenetek analizálása és a szoftver kolátok megkerülése révén.

Az akadályelkerülő pályát differenciálgeometriai elvű (DGA) és prediktív irányítási (mozgó horizontú, RHC) módszerekkel valósítottuk meg. Prediktív irányítás esetén a szabályozó minden horizont kezdetén meghatározza a mozgó jármű (időinvariáns vagy időben változó) linearizált modelljét az aktuális állapot vagy a teljes állapot-trajektória körül, és a prediktív irányítást a mozgó horizonton belül a keletkező LTI vagy LTV rendszerre alapozza. Az irányításhoz a nem mérhető állapotokat (sebesség, oldalcsúszási szög, orientáció és deriváltja, X és Y pozíció) GPS/INS szenzorok jeleiből állapotbecsléssel határozza meg.

A megvalósítás az akadályelkerülő pálya adatait előfeldolgozott tömör kódolt formában kéri a bemenetek között. A szabályozó az állapotbecslést GPS és IMS érzékelők adataira alapozza, amelyekből a gépkocsi állapotvektorát kétszintű kiterjesztett Kalman-szűrővel határozza meg. A dSPACE AutoBox rendszeren a ControlDesk felügyelete alatt futó valós idejű szabályozó megvalósítás biztosítja a mozgó horizontú nemlineáris prediktív irányítás (RHC) pontossági elvárásaihoz szükséges 10ms mintavételi időt. Jelen fázisban a valósidejű rendszer szimulálja a gépjármű dinamikus modelljét és az érzékelők mérési folyamatát. Ezek az információk később rádiókapcsolattal és CAN buszon keresztül juttathatók el a szabályozóhoz a szabályozó végső technologizálásakor.

Simulink környezetben az implementált 3 irányítási módszer (differenciálgeometriai elvű, integrátort nem tartalmazó prediktív, integrátort tartalmazó prediktív) kiválóan működött és hatékony szabályozást biztosított. A Simulink modell AutoBox környezetre is hibátlanul lefordítható volt, az első két irányítási módszer jól működött, de a harmadik irányítási módszer rejtett rendszerhibákra visszavezethetően indítás után elabortálódott.

A fejlesztés során a MATLAB R2006a és a hozzátartozó Simulink, Real Time Workshop és dSPACE AutoBox szoftver és hardver környezetet használtuk. Ennek dSPACE AutoBox része az EJJT Tudásközpont tulajdona és használata hardver kulcshoz kötött, melyet a fejlesztés során a BME IIT Tanszék számára kölcsönzött. A szoftver környezet többi része a BME IIT Tanszék tulajdona.

Tartalomjegyzék

1. Célkitűzés

Műszaki alkalmazások az irányítástechnikában intenzíven építenek a Matlab és toolboxainak szolgáltatásaira. A Matlab licensz azonban költséges és a Matlab környezet nem biztosítja a valós idejű elvárások teljesíthetőségét. Két út kínálkozik ennek megkerülésére, az egyik Matlab licenszet nem igénylő stand-alone alkalmazások kifejlesztése host PC-re Windows vagy Linux alá, a másik pedig a Matlab alatt kifejlesztett szolgáltatás kifordítása beágyazott processzorokra vagy más gyors prototípus rendszerekre. Az előbbi több lehetőséget biztosít a toolboxok felhasználására és könnyebben bővíthető, de gyors rendszerek esetén nem képes a valós idejű elvárások kielégítésére, ezzel szemben az utóbbi gyors rendszerek valós idejű irányítására is képes, de csak a Matlab szolgáltatások egy erősen korlátozott szeletét használhatja és csakis a szabályozó Simulink modelljéből vezethető le.

A pályázat korábbi fázisában stand-alone programcsomagot hoztunk létre gépjármű ütközésmentes pályatervezésére és prediktív irányítására, amely a toolbox szolgáltatások bevonása érdekében nem épített a szabályozó Simulink modelljére, ellenben kihasználta Matlab Compiler szolgáltatásait és host PC-n futtatható, Matlab licenszet már nem igénylő stand-alone programot eredményezett.

A gyors valós idejű működéshez el kell hagyni a PC-környezetet és speciális target rendszerekre kell alapozni a megoldást. Ez az út azonban kizárja a Matlab Compiler használatát, és csakis a Simulink –> Real Time Workshop –> Target Compiler szekvenciában teszi lehetővé valós idejű megoldás létrehozását. A célrendszerül (target) a Tudásközpont adottságainak megfelelően a dSPACE AutoBox rendszert választjuk. Súlyos korlátozó tényező, hogy a Simulink kizárja a toolboxok használatát, és bár a Simulink ú.n. embedded function blokkja bizonyos lehetőségeket megenged saját fejlesztésű függvények bevonására, a Real Time Workshop ezek körét tovább korlátozza A korlátozások szükségessé teszik a korábbi algoritmusok átértékelését és továbbfejlesztését, a pályatervezés és irányítás Simulink alakra hozatalát, lényegében a teljes akadályelkerülési rendszer újratervezését Simulink alapon.

Célunk a kutatás első fázisában mintafeladatok keretében felderíteni a Simulink –> Real Time Workshop –> Target Compiler korlátozásait, majd a kutatás második fázisában a tapasztalatokra alapozva létrehozni az automatikus akadályelkerülés nemlineáris prediktív irányító rendszerének Simulinkre és beágyazott függvényekre alapozott megvalósítását, és a részletes tesztelését elvégzése először csak Simulink környezetben, majd a dSPACE AutoBox rendszeren. Eközben fokozatosan eliminálni kell a szoftver környezet hiányosságait a hibaüzenetek analizálása és a szoftver kolátok megkerülése révén a lehetőségek határain belül.

Az akadályelkerülő pályát differenciálgeometriai elvű (DGA) és prediktív irányítási (mozgó horizontú, RHC) módszerekkel valósítjuk meg. Prediktív irányítás esetén a szabályozó minden horizont kezdetén meghatározza a mozgó jármű (időinvariáns vagy időben változó) linearizált modelljét az aktuális állapot vagy a teljes állapot-trajektória körül, és a prediktív irányítást a mozgó horizonton belül a keletkező LTI vagy LTV rendszerre alapozza. Az irányításhoz a nem mérhető állapotokat (sebesség, oldalcsúszási szög, orientáció és deriváltja, X és Y pozíció) GPS/INS szenzorok jeleiből állapotbecsléssel határozza meg.

A kifejlesztendő megvalósítás az akadályelkerülő pálya adatait előfeldolgozott tömör kódolt formában kéri a bemenetek között. A szabályozó az állapotbecslést GPS és IMS érzékelők adataira alapozza, amelyekből a gépkocsi állapotvektorát kétszintű kiterjesztett Kalman-szűrővel határozza meg. A dSPACE AutoBox rendszeren a ControlDesk felügyelete alatt futó valós idejű szabályozó megvalósításnak biztosítania kell a mozgó horizontú nemlineáris prediktív irányítás (RHC) pontossági elvárásaihoz szükséges 10ms mintavételi időt. A fejlesztés során a valósidejű rendszer szimulálja a gépjármű dinamikus modelljét és az érzékelők mérési folyamatát is. Ezek az információk a jövőbeni továbbfejlesztések során rádiókapcsolattal és CAN buszon keresztül juttathatók el a szabályozóhoz a szabályozó végső technologizálásakor.

A fejlesztés során a MATLAB R2006a és a hozzátartozó Simulink, Real Time Workshop és dSPACE AutoBox szoftver és hardver környezetet használjuk. A dSPACE AutoBox az EJJT Tudásközpont, a szoftver többi része a BME IIT Tanszék tulajdona.

2. Gépjármű akadályelkerülő rendszer valós idejű környezetbe integrálásához szükséges feltételek vizsgálata

Műszaki alkalmazások az irányítástechnikában intenzíven építenek a Matlab és toolboxainak szolgáltatásaira. A Matlab licensz azonban költséges és a Matlab környezet nem biztosítja a valós idejű elvárások teljesíthetőségét. Két út kínálkozik ennek megkerülésére, az egyik Matlab licenszet nem igénylő stand-alone alkalmazások kifejlesztése host PC-re Windows vagy Linux alá a Matlab Compiler szolgáltatásainak kiaknázásával, a másik pedig a Matlab alatt kifejlesztett szabályozási rendszer kifordítása beágyazott processzorokra vagy más gyors prototípus rendszerekre. Az előbbi több lehetőséget biztosít a toolboxok felhasználására és könnyebben bővíthető. Korábbi kutatásainkban ezt a fejlesztési módszert már bemutattuk [?]. Mivel azonban ez a módszer gyors szabályozási rendszerek esetén nem képes a valós idejű elvárások kielégítésére, ezért megvizsgáljuk a Matlab alatt kifejlesztett szabályozási rendszer kifordítását beágyazott processzorokra vagy más gyors prototípus rendszerekre.

2.1 A Matlab/Simulink környezet általános megszorításai

A gyors valós idejű működéshez el kell hagyni a PC-környezetet és speciális target rendszerekre kell alapozni a megoldást. Ez az út azonban kizárja a Matlab Compiler használatát, és csakis a Simulink –> Real Time Workshop –> Target Compiler szekvenciában teszi lehetővé valós idejű megoldás létrehozását. A célrendszer (target) sokféle lehet, számunkra ezek közül a BME IIT Tanszék hardver/szoftver adottságait is figyelembe véve a beágyazott rendszerek létrehozásához bevált MPC555 (Motorola) processzor és a gyors prototípus tervezést támogató további célrendszerek, köztük elsősorban a dSPACE DS1103 és dSPACE AutoBox célrendszerek jöhetnek számításba.

Súlyos korlátozó tényező azonban, hogy a Simulink kizárja a toolboxok használatát, és bár a Simulink ú.n. embedded function (beágyazott függvény) blokkja bizonyos lehetőségeket megenged saját fejlesztésű függvények bevonására, a Real Time Workshop ezek körét tovább korlátozza (csak max. kétidexű tömbök lehetségesek, struktúra használata nem megengedett, a Matlab részét képező standard függvényeknek csak egy szűk köre megengedett, fájl-műveletek nem megengedettek, toolboxok használata kizárt). A beágyazott függvények írásakor tekintettel kell lenni arra, hogy beágyazott függvényekből a Matlab függvényeknek csak egy szűk listája hívható. Pozitívum, hogy ezen a listán szerepelnek az `inv, svd, qr` függvények, negatívum viszont, hogy nem hívhatók a `rand` és `randn` véletlen szám generárorok.

A korlátozások szükségessé teszik a korábbi algoritmusok átértékelését és továbbfejlesztését, a pályatervezés és irányítás Simulink alakra hozatalát, lényegében a teljes akadályelkerülési rendszer újratervezését Simulink alapon. További problémát okozhat, hogy a Real Time Workshop és a Target Compiler további korlátozásokat definiálhat, ezért nem garantált, hogy a Matlab/Simulink környezetben futó és tesztelt Simulink program valóban kifordítható a target rendszerre.

2.2 A Matlab/Simulink környezet speciális megszorításai az automatikus akadályelkerüléskor

Korábban az akadályelkerülő pálya megtervezésekor a program elvárt bemeneti adatait a statikus akadály (pl. elől haladó járműről leesett teher) és a mozgó akadály (szembejövő jármű) geometriai paraméterei (befoglaló kör középpontja és sugara) alkották, valamint a mozgó akadály és a saját jármű sebessége. További bemeneti adatok azonosították az útszakaszt (bal oldali és jobb oldali sávszélesség). Az akadályelkerülő pályát az elasztikus szalag elvére épülve határoztuk meg, amely nagyméretű nemlineáris egyenletrendszerre vezet, most azonban nem használható megoldására az *fsolve* függvény, mivel az az Optimization Toolbox része és toolboxok nem használhatók a Simulink-ben. Ezért a valós idejű megoldás a pályatervezésre nem fog kiterjedni, azt külső eszközön kell megvalósítani, amely esetleg rádiókapcsolat bevonásával CAN buszon juttatja el az

akadályelkerülő pályainformációt a mintavételi idő ütemében a szabályozóhoz. A CAN busz szolgáltatást a Simulink, Real Time Toolbox es Target Compiler támogatja.

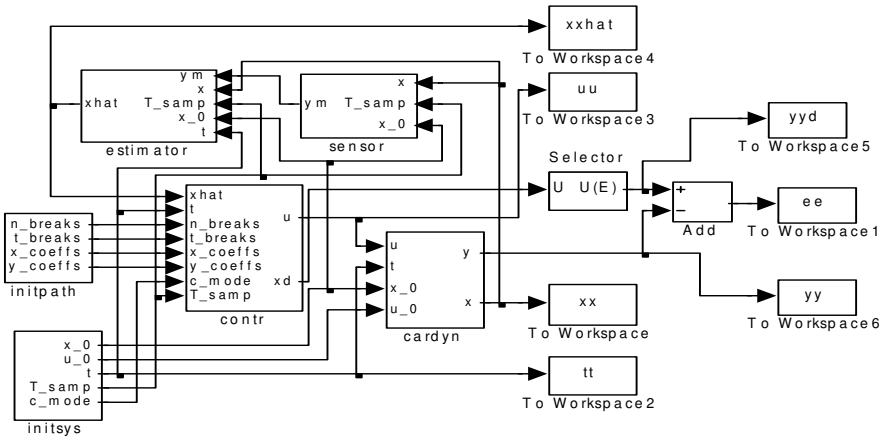
Hasonlóan egy külső szenzor processzorra telepítve képzelendő el a szenzorjelek mérése, amelyek kétantennás differenciális GPS, valamint 3D gyorsulás és szögsebesség érzékelő (INS, Inertial Navigation System) mérési adatait foglalják magukba. A szenzorok mintavételi frekvenciája eltérő lehet, adataikat a szenzor processzor perspektivikusan CAN buszon keresztül továbbítja az irányító rendszerhez.

A fejlesztés első fázisa az irányító rendszer Simulink modelljének kifejlesztése és tesztelése Matlab licenszet tartalmazó fejlesztési környezetben. A kutatás második fázisában erre lehet alapozni a beágyazott irányítórendszer kifejlesztését a Simulink –> Real Time Workshop –> Target Compiler szolgáltatások kiaknázásával. A cél továbbra is az automatikus akadályelkerülő pálya megvalósítása prediktív irányítás és állapotbecslés bevonásával.
A szabályozó számára az akadályelkerülő pálya és a szenzorjelek bemenő jelek, a beavatkozó jelek magas szintű vezérlő jelek a longitudinális gyorsító erő és a kormány szögelfordulás alakjában. Feltételezzük, hogy megvalósításukat az alacsony szinten megvalósított szabályozások elvégzik. A kapcsolatot ebben az irányban is perspektivikusan CAN buszon keresztül tervezzük megvalósítani. Az irányításhoz a nem mérhető állapotokat (sebesség, oldalcsúszási szög, orientáció és deriváltja, X és Y pozíció) GPS/INS szenzorok jeleiből kétszintű Kalman-szűrő határozza meg.

Az akadályelkerülő pályát differenciálgeometriai elvű (DGA) és prediktív irányítási (mozgó horizontú, RHC) módszerekkel valósíthatjuk meg. Prediktív irányítás esetén a rendszer minden horizont kezdetén meghatározza a mozgó jármű (időinvariáns vagy időben változó) linearizált modelljét az aktuális állapot vagy a teljes állapot-trajektória körül, és a prediktív irányítást a mozgó horizonton belül a keletkező LTI vagy LTV rendszerre alapozza.
A prediktív szabályozó Simulink modelljének kifejlesztésekor ki kell dolgozni tesztelési célra a pályatervezés és a szenzorjelek szimulációját is. Ezeket egy-egy Simulink blokk valósíthatja meg, melynek eredményeit az állapotbecslő és szabályozó felhasználja. A szenzorjelek szimulációja célszerűen szintén egy Simulink program lehet.

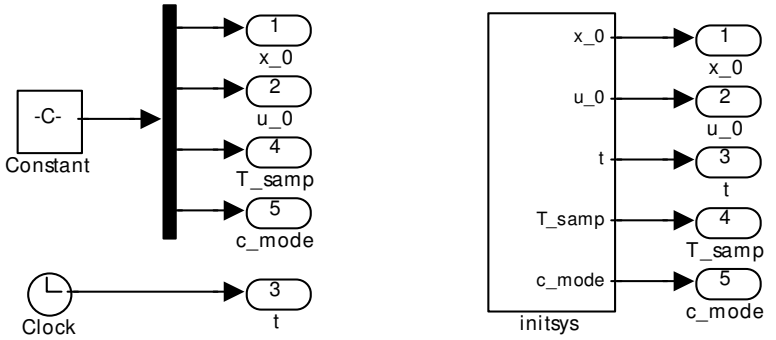
2.3 Az akadályelkerülő szabályozás Simulink modelljének koncepciója

Az akadályelkerülő szabályozási rendszer Simulink modelljét a *2.1. ábra* mutatja be. A főbb blokkokat a Simulink embedded function (beágyazott függvény) szolgáltatására alapozzuk.

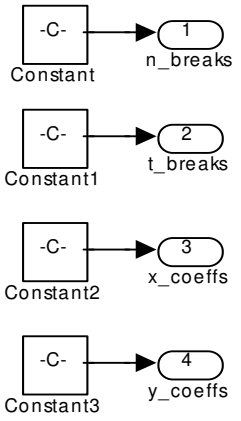


2.1. ábra. Az akadályelkerülő rendszer (CAS) irányításának closedsys.mdl Simulink modellje

A főbb egységek közül `cardyn` a gépjárművet emulálja annak nemlineáris dinamikus modellje alajában, `sensor` pedig a GPS/IMS érzékelőt. Mindketten a szabályozó kifejlesztéséhez és megbízható teszteléséhez szükséges kiegészítő egységek. A szabályozó későbbi technologizálásakor a `contr` szabályozó `cardyn` helyett az alacsonyszintű longitudinális gyorsító erőt és a kormányszöget megvalósító alrendszerek felé küldi el kimenő jelét. A későbbi technologizáláskor `sensor` helyébe a valódi GPS/IMS érzékelő kerül, amely a gépjárműre van telepítve és CAN buszon keresztül csatlakozik az `estimator` kétszintű kiterjesztett Kalman-szűrőhöz.
Az akadályelkerülő szabályozó az `initpath, initsys, estimator, contr` egységekből áll. Az `initpath` és `initsys` egységek nem tartalmaznak beágyazott függvényt, felépítésüket a 2.2-2.3. ábrák mutatják.



2.2. ábra. Az `initsys` blokk részletes (bal oldalt) és tömör (jobb oldalt) alakja



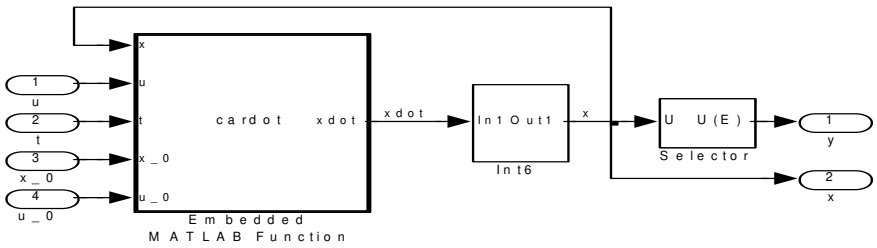
2.3. ábra. Az `initpath` blok felépítése

Az `initsys` blokk fő feladata az x_0 kezdeti állapot, az u_0 kezdeti beavatkozó jel, a T_samp mintavételi idő és a c_mode szabályozási mód (DG, RHC) beállítása, továbbá folyamatosan a *t* valós idő előállítása a szabályozó számára, amely *t*-re és az `initpath` által adott `n_breaks, t_breaks, x_coeffs, y_coeffs` adatokra alapozva a `contr` részét képező `embed_path()` beágyazott függvénnyel rendre meghatározza a pálya adatokat a *t* pillanatban. Itt `n_breaks, t_breaks, x_coeffs, y_coeffs` megfelelnek a statikus és dinamikus akadály adatainak ismeretében korábban az elasztikus szalag elvén alapulva, az `fsolve()` és a

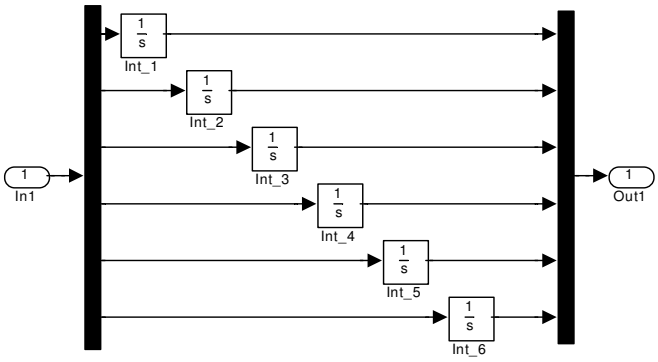
`spline()` toolbox függvények felhasználásával számított és simított pálya adtoknak. Mivel a toolbox szolgáltatások a Simulink-ben és Real Time Workshop-ban nem elérhetők, ezért ezeket előre ki kell számítani és tárolni az `embed_breaks_coeffs.mat` fájlban, és a szimuláció indítása előtt a MATLAB munkaterületére be kell olvasni (`load`).

Az `embed_breaks_coeffs.mat` fájlnak tartalmaznia kell a `T_samp` mintavételi időt, az `n_breaks, t_breaks, x_coeffs, y_coeffs` akadályelkerülő pálya adatokat, az `x_0` kezdeti állapotot és az `u_0` kezdeti beavatkozó jelet, a gépjármű `v_own` átlagsebességét és a differenciálgeometriai elven alapuló irányítás `lambda` szabályozási paraméterét. A `c_mode` paraméter a Matlab parancsmezejében állítandó be a választott szabályozási módnak megfelelően (1, ha differenciálgeometriai elvű, 2, ha prediktív irányítás szükséges).

A gépjármű nemlineáris dinamikus modelljét a *2.4. ábra* mutatja be. A dinamikus modellben a `cardot` beágyazott függvény képezi a nemlineáris állapotegyenlet jobb oldalát, amelyből az állapotvektort az `Int6` integrátorok határozzák meg. A `selector` ezek közül kiválasztja az X,Y pozíció és ψ orientáció kimeneteket. A 2.5. ábra szerinti `Int6` integrátorok beállítják a kezdeti állapotot is `x_0` alapján. A helyes működéshez az integrálást Euler-módszerrel kell elvégezni.



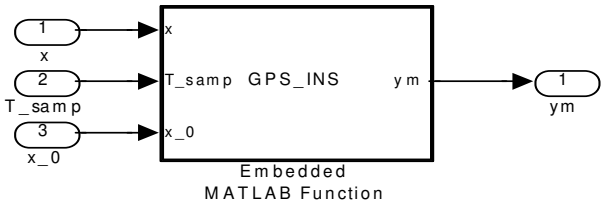
2.4. ábra. A gépjármű nemlineáris dinamikus modellje.



2.5. ábra. Az `Int6` integrátorok Simulink modellje

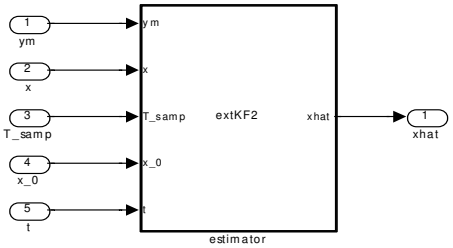
A `sensor` egység emulálja a GPS/IMS érzékelőket. Működéséhez szükség van a `rand` és `randn` véletlenszám generátorokra, amelyek azonban nem szerepelnek a beágyazott függvények által meghívható függvények listáján. Ezért ezeket beágyazott függvényként meg kellett írni,

amelyhez a Matlab 5.x-ben is használt algoritmus került megvalósításra. A `rand_own1()` saját függvény a Park-Miller algoritmust, a `randn_own1()` saját függvény pedig a polar algoritmust valósítja meg. Ezek a függvények részét képezik a `sensor` egységet megvalósító `GPS_INS()` függvénynek a *2.6. ábrán*.



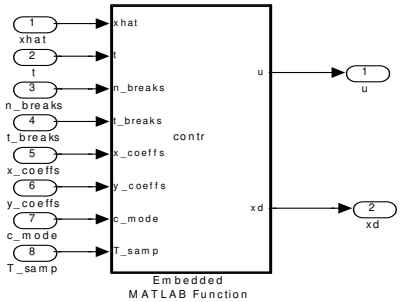
2.6. ábra. A `sensor` egységet megvalósító `GPS_INS` beágyazott függvény.

A beágyazott függvényként megírt állapotbecslőt az `estimator` Simulink blokk valósítja meg, lásd *2.7. ábra*.



2.7. ábra. Az állapotbecslőt beágyazott függvényként megvalósító `estimator` kétszintű Kalman-szűrő

A beágyazott függvényként megírt szabályozót a `controller` Simulink blokk valósítja meg, lásd *2.8. ábra*. Két lehetséges üzemmódja a DG (differenciálgeometriai elvű) és RHC (mozgó horizontú prediktív) irányítás.



2.8. ábra. A szabályozót beágyazott függvényként megvalósító Simulink blokk

A továbbiakban összefoglaljuk a `contr` és `estimator` blokkokban megvalósított algoritmusokat, majd megadjuk az algoritmusokat implementáló beágyazott függvények prototípusait és a Simulink környezetben megvalósításukkal kapott szabályozási tranzienseket.

3. Az irányítási és állapotbecslési algoritmusok

Az algoritmusok főbb egységeit a referencia jel tervezés, a járműmodell, a differenciálgometriai elvű irányítás, a nemlineáris prediktív irányítás és az állapotbecslés algoritmusai alkotják.

3.1 Referencia jel tervezés az irányításokhoz

Az `n-breaks, x_coeffs, y_coeffs` bemeneti adatok tömören kódolt formában tartalmazzák az akadályt elkerülő pálya adatait. Ezeket a szabályozó a későbbi technologizálás után CAN buszon kapja meg, ezért a tömörség alapvető szempont az adatátviteli idő csökkentése érdekében. Ezért `n_breaks` tartalmaza a korábbi spline szakaszok számát, `x_coeffs` és `y_coeffs` pedig a harmadfokú approximációs polinomok adatait kétindexes tömb formájában, mivel a struktúra adat típus beágyazott függvényekben nem megengedett. Ezekből az `embed_path()` és az abból hívott `embed_deriv3()` függvények harározzák meg a referencia jeleket (alapjel időfüggvényeket). A függvények kiváltják a `ppval, unmkpp, mkpp` spline-technikájára épülő függvényeket. A deriváltak segítségével a kinematikai mennyiségeket (a referencia mozgásban jogosan nulla oldalcsúszási szöget feltételezve) az alábbi összefüggések adják:

$$
\begin{aligned}
v &= (\dot{x}^2 + \dot{y}^2)^{1/2} \\
\dot{v} &= \frac{\dot{x}\,\ddot{x} + \dot{y}\,\ddot{y}}{(\dot{x}^2 + \dot{y}^2)^{1/2}} \\
\psi &= \arctan(\dot{y}/\dot{x}) \\
\dot{\psi} &= \frac{\ddot{y}\,\dot{x} - \dot{y}\,\ddot{x}}{\dot{x}^2 + \dot{y}^2} \\
\ddot{\psi} &= -2\frac{\dot{x}\,\ddot{x} + \dot{y}\,\ddot{y}}{(\dot{x}^2 + \dot{y}^2)^2} + \frac{\dddot{y}\,\dot{x} - \dot{y}\,\dddot{x}}{\dot{x}^2 + \dot{y}^2} \\
\kappa &= \frac{\dot{x}\,\ddot{y} - \dot{y}\,\ddot{x}}{\left(\dot{x}^2 + \dot{y}^2\right)^{3/2}}
\end{aligned}
\tag{3.1}
$$

A deriváltakat számító függvény prototípus alakja és egysoros commentje a helphez a következő:

```
function [yy,d1yy,d2yy,d3yy]=embed_deriv3(xx,breaks,coeffs)
%Embedded deriv3()
```

A referencia jeleket számító függvény prototípus alakja és egysoros commentje a helphez a következő:

```
function [xt,d1xt,d2xt,d3xt,yt,d1yt,d2yt,d3yt]=...
    embed_path(t,t_breaks,x_coeffs,y_coeffs)
%Compute reference signals and derivatives
```

3.2 Gépjármű dinamikus modellje és Lie-algebrai tulajdonságai

A jármű dinamikus modelljében az első kerekekre (front wheels) F betűvel, a hátsó kerekre (rear wheels) R betűvel hivatkozunk. Kétféle dinamikus modellt különböztetünk meg (Lantos, 2006a). Az approximált nemlineáris modell bemenet-affin approximációt ad, amely szükséges a differenciálgeometriai elvű irányításhoz. A jármű dinamikus modelljében az oldalcsúszási szög és a kormányzási szög trigonometrikus függvényeit elsőfokú Taylor-polinomokkal approximáljuk, de a többi változóban megőrizzük a nemlineáris összefüggéseket. Ezen kívül bemenő jelnek nem a longitudinális gyorsító erőt és a kormányzási szöget tekintjük, hanem a longitudinális gyorsító erőt és az első kerékre ható oldalirányú származtatott erőt, amely utóbbiból a kormányzási szög számítható.

Vezessük be a következő további jelöléseket az első S_v és hátsó S_h oldalirányú erőkre:

$$S_h = c_R(-\beta - \frac{l_R\dot{\psi}}{v_G}), \quad S_v = c_F(\delta_w - \beta - \frac{l_F\dot{\psi}}{v_G}) \tag{3.2}$$

Nullának vesszük az első kereket meghajtó F_{lF} és nemnullának a hátsó kereket meghajtó F_{lR} erőt. A rendszer bemenetének tekintjük az $u = (u_1, u_2)^T = (S_v, F_{lR})^T$ bemenő jel vektort, és állapotvektornak az $x = (\beta, \psi, \dot{\psi}, v_G, X, Y)^T$ vektort.

3.2.1 A pontos nemlineáris járműmodell

$$\begin{aligned}
\dot{\beta} &= -\dot{\psi} + \frac{1}{m_v v_G}\{F_{lF}\sin(\delta_W - \beta) - (F_{lR} - T)\sin(\beta) \\
&\quad + c_F(\delta_W - \beta - \frac{l_F\dot{\psi}}{v_G})\cos(\delta_W - \beta) + c_R(-\beta + \frac{l_R\dot{\psi}}{v_G})\cos(\beta)\} \\
\dot{\psi} &= \dot{\psi} \\
\ddot{\psi} &= \frac{1}{I_{zz}}\{l_F F_{lF}\sin(\delta_W) + l_F c_F(\delta_W - \beta - \frac{l_F\dot{\psi}}{v_G})\cos(\delta_W) - l_R c_R(-\beta + \frac{l_R\dot{\psi}}{v_G})\} \\
\dot{v}_G &= \frac{1}{m_v}\{F_{lF}\cos(\delta_W - \beta) + (F_{lR} - T)\cos(\beta) - c_F(\delta_W - \beta - \frac{l_F\dot{\psi}}{v_G})\sin(\delta_W - \beta) \\
&\quad + c_R(-\beta + \frac{l_R\dot{\psi}}{v_G})\sin(\beta)\} \\
\dot{X} &= v_G\cos(\psi + \beta) \\
\dot{Y} &= v_G\sin(\psi + \beta)
\end{aligned} \tag{3.3}$$

A vizsgálatok során az $F_{lF} = 0$ és $T = 0$ választással éltünk. Az állapotegyenlet jobb oldalát számító függvény prototípus alakja és egysoros commentje a helphez a következő:

```
function xdot = cardot(x,u,t,x_0,u_0)
% Embedded function for computing the right side of the car's state
equation
```

3.2.2 Approximált nemlineáris járműmodell

Az X, Y jelek kivételével Taylor-sorfejtéssel a következő approximált lineáris modellhez jutunk:

$$\dot{x} = \begin{pmatrix} -x_3 + (Tx_1 + S_h)/(m_v x_4) \\ x_3 \\ -S_h l_R / I_{zz} \\ -T/m_v \\ x_4\cos(x_1 + x_2) \\ x_4\sin(x_1 + x_2) \end{pmatrix} + \begin{bmatrix} 1/(m_v x_4) & -x_1/(m_v x_4) \\ 0 & 0 \\ l_F / I_{zz} & 0 \\ 0 & 1/m_v \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{pmatrix} S_v \\ F_{lR} \end{pmatrix} = A(x) + B(x)u \tag{3.4}$$

$$y = \begin{pmatrix} x_5 \\ x_6 \end{pmatrix} = C(x)$$

A vizsgálatok során az $F_{lF} = 0$ és $T = 0$ választással éltünk. Az állapotegyenlet jobb oldalát számító függvény prototípus alakja és egysoros commentje a következő:

```
function [xdot,F_lR,deltaw,Sv,f,G]=apprfunxdot(t,x,F_lR,deltaw,...
    c_F,l_F,c_R,l_R,m_v,I_zz)
%Compute right side of approximated vehicle dynamic model
```

3.2.3 Az approximált modell Lie-algebrai tulajdonságai

Az approximált nemlineáris modellre Freund, E., & Mayr, R. (1997) alapján elvégezve a

$$N_A^k C_i(x) = \left[\frac{\partial}{\partial x} N_A^{k-1} C_i(x)\right] A(x), \quad N_A^0 C_i(x) = C_i(x)$$
$$d_i = \min\left\{ j : \left[\frac{\partial}{\partial x} N_A^{k-1} C_i(x)\right] B(x) \neq 0^T,\ j = 1,2,\ldots,n \right\} \tag{3.5}$$

számításokat és bevezetve a $C_{12}\cos(x_1 + x_2)$, $S_{12} = \sin(x_1 + x_2)$ jelölést kapjuk, hogy

$$N_A^0 C_1(x) = x_5,\ N_A^1 C_1(x) = x_4 C_{12},\ N_A^2 C_1(x) = -\frac{1}{m_v}[T(S_{12}x_1 + C_{12}) + S_{12}S_h]$$

$$N_A^0 C_2(x) = x_6, N_A^1 C_2(x) = x_4 S_{12},\ N_A^2 C_2(x) = \frac{1}{m_v}[T(C_{12}x_1 - S_{12}) + C_{12}S_h]$$

Az approximált nemlineáris rendszer differenciális rendje $d_1 = d_2 = 2$, továbbá teljesül

$$C^*(x) = \begin{bmatrix} N_A^{d_1} C_1(x) \\ N_A^{d_2} C_2(x) \end{bmatrix} = \begin{bmatrix} -\frac{1}{m_v}[T(S_{12}x_1 + C_{12}) + S_{12}S_h] \\ \frac{1}{m_v}[T(C_{12}x_1 - S_{12}) + C_{12}S_h] \end{bmatrix}_{2\times 1}$$

$$S(x) = \begin{bmatrix} \frac{-S_{12}}{m_v} & \frac{S_{12}x_1 + C_{12}}{m_v} \\ \frac{C_{12}}{m_v} & \frac{-C_{12}x_1 + S_{12}}{m_v} \end{bmatrix}_{2\times 2} \quad \text{és} \quad \det(S(x) = -\frac{1}{m_v^2} \tag{3.6}$$

$$S^{-1}(x) = m_v \begin{bmatrix} C_{12}x_1 - S_{12} & S_{12}x_1 + C_{12} \\ C_{12} & S_{12} \end{bmatrix}_{2\times 2}$$

$$M^*(x) = \begin{bmatrix} \alpha_{01} N_A^0 C_1(x) + \alpha_{11} N_A^1 C_1(x) \\ \alpha_{02} N_A^0 C_2(x) + \alpha_{12} N_A^1 C_2(x) \end{bmatrix} = \begin{bmatrix} \alpha_{01}x_5 + \alpha_{11}x_4 C_{12} \\ \alpha_{02}x_6 + \alpha_{12}x_4 S_{12} \end{bmatrix}_{2\times 1}$$

Ezért a nemlineáris rendszerek Lie-algebrán alapuló eredményei szerint választható

$$u = S^{-1}(x)\left\{ -C^*(x) + \Lambda w - M^*(x) \right\} \tag{3.7}$$

ahol $\lambda = diag(\lambda_1, \lambda_2)$. Vezessük be a következő jelöléseket:

$$\bar{y}_1 = \lambda_1 w_1 - \alpha_{01}x_5 - \alpha_{11}x_4 C_{12}$$
$$\bar{y}_2 = \lambda_2 w_2 - \alpha_{02}x_6 - \alpha_{12}x_4 S_{12} \tag{3.8}$$

akkor átalakítások után kapjuk, hogy a zárt rendszerre teljesül:

$$
\begin{aligned}
u_1 &= -S_h + [(C_{12}x_1 - S_{12})\bar{y}_1 + (S_{12}x_1 + C_{12})\bar{y}_2]m_v \\
u_2 &= T + [C_{12}\bar{y}_1 + S_{12}\bar{y}_2]m_v \\
\dot{y}_1 &= x_4 C_{12} \\
\dot{y}_2 &= x_4 S_{12} \\
\ddot{y}_1 &= \dot{x}_4 C_{12} - x_4 S_{12}(\dot{x}_1 + \dot{x}_2) \\
\dot{x}_1 + \dot{x}_2 &= (-S_{12}\bar{y}_1 + C_{12}\bar{y}_2)/x_4 \\
\dot{x}_4 &= C_{12}\bar{y}_1 + S_{12}\bar{y}_2 \\
\ddot{y}_1 = \bar{y}_1 &= \lambda_1 w_1 - \alpha_{01}x_5 - \alpha_{11}x_4 C_{12} = \lambda_1 w_1 - \alpha_{01}y_1 - \alpha_{11}\dot{y}_1 \\
\ddot{y}_2 = \bar{y}_2 &= \lambda_2 w_2 - \alpha_{02}x_6 - \alpha_{12}x_4 S_{12} = \lambda_2 w_2 - \alpha_{02}y_2 - \alpha_{12}\dot{y}_2
\end{aligned}
\tag{3.9}
$$

3.3 Nemlineáris kimeneti visszacsatoláson alapuló irányítás

Választható $\lambda_1 = \lambda_2 := \lambda$, $\alpha_{01} = \alpha_{02} := \lambda$ és $\alpha_{11} = \alpha_{12} = 2\sqrt{\lambda}$, amellyel két szétcsatolt aperiodikus határesetű másodrendű rendszerhez jutunk, amelynek karakterisztikus egyenlete $s^2 + 2\sqrt{\lambda}s + \lambda = 0.$

$$
\ddot{y}_i + \alpha_{1i}\dot{y}_i + \alpha_{0i}y_i = \lambda_i w_i \tag{3.10}
$$

Választható ezért $w_i := w_{ia} + \frac{1}{\lambda}(\alpha_{1i}\dot{w}_{ia} + \ddot{w}_{ia})$, amely után a két szétcsatolt rendszer

$$
\begin{aligned}
\ddot{y}_i + \alpha_{1i}\dot{y}_i + \lambda y_i = \lambda[w_{ia} + \frac{1}{\lambda}(\alpha_{1i}\dot{w}_{ia} + \ddot{w}_{ia})] \Rightarrow \\
(\ddot{w}_{ia} - \ddot{y}_i) + \alpha_{1i}(\dot{w}_{ia} - \dot{y}_i) + \lambda(w_{ia} - y_i) = 0
\end{aligned}
\tag{3.11}
$$

alakú stabil rendszer lesz, ahol a referencia jelek szerepét betöltő $w_{1a} = X_a(t)$ és $w_{2a} = Y_a(t)$ a megtervezett ütközést elkerülő pályák.

A differenciálgeometriai elven alapuló irányításhoz szükség van a pálya első és második deriváltjára is, amelyet spline-technikával korábban már megterveztünk.

Vegyük észre, hogy az S_v irányítás és az állapotváltozók (vagy a becsült állapotváltozók) ismeretében a valódi irányítás, vagyis a δ_w kormányzási szög meghatározható:

$$
\delta_w = \frac{S_v}{c_F} + \beta + \frac{l_F\dot{\psi}}{v_G} \tag{3.12}
$$

A differenciálgeometriai elvekre épülő szabályozó c_mode=1 esetén közvetlenül kerül megvalósításra a `contr` szabályozó törzsében. A meghívott saját fejlesztésű függvények prototípus alakja és a hatásukat körvonalazó egysoros comment a következő:

```
function xdvect=funxdvec(t,t_breaks,x_coeffs,y_coeffs)
%Compute desired state from CAS path database

function [xt,d1xt,d2xt,d3xt,yt,d1yt,d2yt,d3yt]=...
      embed_path(t,t_breaks,x_coeffs,y_coeffs)
%Compute reference signals and derivatives

function [yy,d1yy,d2yy,d3yy]=embed_deriv3(xx,breaks,coeffs)
%Embedded deriv3()

function ix=embed_maxfind(x,x0)
%Embedded simple find
```

3.4. A nemlineáris prediktív irányítási algoritmus

A nemlineáris modellalapú prediktív irányítás egy lehetséges megvalósítása a mozgó horizontú irányítás (RHC, Receding Horizon Control), amely minden horizont kezdetén linearizálja a nemlineáris rendszert, és az így keletkező lineáris időinvariáns (LTI), vagy lineáris időben változó (LTV) rendszert optimalizálja. Az optimalizálási feladat egy költségfüggvény minimalizálása felnyitott körben a rendszer jövőbeli viselkedésének jóslására alapozva (predikció). Meghatározásra kerül az optimális beavatkozó jel (control) sorozat a horizonton belül, és kiadásra kerül a sorozat első eleme aktuális beavatkozó jelként zárt körben. Ez a lépés ciklikusan ismétlődik az új horizontra, amely az előzőnek T mintavételi idővel való eltolásával keletkezik. Nyitott kérdés a zárt nemlineáris rendszer stabilitása, amelyre jó hatással van a horizont N szélességének növelése (a horizont szélessége időben mérve NT).

Ha a nemlineáris dinamikus modellt használnánk a predikcióra, akkor egy nemlineáris optimizálási feladat keletkezne, amelynek valós idejű megoldása gyors rendszerek esetén idő-kritikus. Ezért a nemlineáris modell linearizálását választottuk az előírt nominális trajektória mentén minden horizont kezdetén, amelyet követ a perturbációs hatás optimalizálása a horizonton belül kvadratikus kritérium szerint és analitikusan kezelhető végfeltétel mellett.

Komoly problémát jelent azonban, hogy a jármű alulaktuált, azaz nagyobb a szabadságfoka, mint a beavatkozó jelek száma, következésképp nem minden nominális pályához létezik azt pontosan megvalósító nominális irányítás. Ezért a megtervezett CAS akadályelkerülő pályához sem határozható meg egyszerű numerikus technikával a megfelelő approximációt biztosító nominális beavatkozó jel sorozat. Legfeljebb egy approximáló irányítást tételezhetünk fel, amelynek hatására a mozgás a nominális pálya közelében halad.

3.4.1 Választható alternatívák

Két lehetőség kínálkozik (Lantos, 2006b): i) LTI rendszer választása, amelyhez csak (x_0, u_0) szükséges, nem pedig a teljes $u(t)$ időfüggvény a horizonton belül. ii) LTV rendszer generálása a később ismertetendő algoritmus szerint. Ehhez abból indulhatunk ki, hogy ha a hiba a horizonton belül kicsi, akkor a horizonton belüli optimális beavatkozó jel sorozat a megtervezett nominális CAS pályához szükséges beavatkozó jel sorozat egy jó approximációjának tekinthető. Ezért linearizáljuk a rendszert az előző horizonton belül kapott optimális $x(t), u(t)$ mentén az új horizont kezdetén, és optimalizáljuk az ekörüli perturbációt az új horizonton belül. Vegyük azonban észre, hogy az előző horizonton belüli optimális beavatkozó jel sorozat balra tolása miatt az új u_{N-1} a horizont végén hiányzik, ezért ennek számítását is ki kell dolgozni. A programban mindkét eset kiválasztható.

A továbbiakban az egyszerűség kedvéért *nominálisnak* nevezzük az előző horizontban keletkezett optimális, majd eltolt és kiegészített beavatkozó jel sorozatot, és ezzel szemben a *kívánt* (desired) beavatkozó jel sorozat az lesz, amely az LTV linearizált rendszer körüli perturbációk hatását minimalizálja. A módszer természetesen LTI rendszer esetére is alkalmazható, mivel az egyszerűbben képezhető LTI approximáció prediktív irányítási szempontból az LTV rendszer irányítása speciális esetének tekinthető.

3.4.2 A megvalósított prediktív irányítás elméleti alapjai

Jelölje $\{x_0, x_1, \ldots, x_N\}$ és $\{u_0, u_1, \ldots, u_{N-1}\}$ a nominális állapot és bemenő (beavatkozó) jel sorozatot a horizonton belül, legyen továbbá $\hat{x}_0$ a becsült állapot a horizont kezdetén és $\{y_0 = Cx_0, y_1 = Cx_1, \ldots, y_N = Cx_N\}$ az állapot sorozathoz tartozó kimenő jel sorozat. Legyen $\{y_{d0}, y_{d1}, \ldots, y_{dN}\}$ a megkívánt (desired) kimenő jel sorozat és jelölje az ehhez képesti hiba jel sorozatot $\{e_0 = y_{d0} - y_0, e_1 = y_{d1} - y_1, \ldots, e_N = y_{dN} - y_N\}$. A nominális bemenő jel sorozat lehet a differenciálgeometriai elvű algoritmus (DGA) szerint számított irányítás a legelső horizont esetén, vagy a továbbiakban az előző optimális bemenő jel sorozat eggyel eltolva és egy új elemmel kiegészítve, amely pl. egy végfeltételből számítható vagy az utolsó bemenő jel egyszerű megismétlése (lásd később az 1. lépést az algoritmusban).

A nemlineáris dinamikus modell linearizálható a nominális állapot és bemenő jel sorozatok mentén, és tekinthetők a keletkező lineáris időben változó (LTV) rendszer körüli $\delta x_0 = \hat{x}_0 - x_0$, $\delta x_1, \ldots, \delta x_N$, $\delta u_0, \ldots, \delta u_{N-1}$ perturbációk. A linearizálás történhet a pontos és az appriximált (input affin) nemlineáris modell körül. A perturbációkat a $\delta x_{i+1} = A_i \delta x_i + B_i \delta u_i$ LTV rendszer írja le. A kimeneti hibasorozat $y_{di} - C(x_i + \delta x_i) = e_i - \delta y_i$, a J költségfüggvény pedig választható egy kvadratikus függvénynek, amely bünteti a kimeneti hibákat és anévleges irányítástól való nagy eltéréseket:

$$J = \frac{1}{2}\sum_{i=1}^{N-1} \left\| e_i - \delta y_i \right\|^2 + \frac{1}{2}\lambda \sum_{i=0}^{N-1} \left\| \delta u_i \right\|^2 \tag{3.13}$$

Az állapot és kimenő jel perturbációk a következőképp számíthatók:

$$\begin{pmatrix} \delta x_1 \\ \delta x_2 \\ \vdots \\ \delta x_{N-1} \\ \delta x_N \end{pmatrix} = \begin{bmatrix} A_0 \\ A_1 A_0 \\ \vdots \\ A_{N-2} \cdots A_1 A_0 \\ \hline A_{N-1} \cdots A_1 A_0 \end{bmatrix} \delta x_0 + \begin{bmatrix} B_0 & 0 & \cdots & 0 & 0 \\ A_1 B_0 & B_1 & \cdots & 0 & 0 \\ \vdots & \cdots & \ddots & 0 & 0 \\ A_{N-2} \cdots A_1 B_0 & A_{N-2} \cdots A_2 B_1 & \cdots & B_{N-2} & 0 \\ \hline A_{N-1} \cdots A_1 B_0 & A_{N-1} \cdots A_2 B_1 & \cdots & A_{N-1} B_{N-2} & B_{N-1} \end{bmatrix} \begin{pmatrix} \delta u_0 \\ \delta u_1 \\ \vdots \\ \delta u_{N-2} \\ \delta u_{N-1} \end{pmatrix} \tag{3.14}$$

$$\begin{pmatrix} \delta y_1 \\ \delta y_2 \\ \vdots \\ \delta y_{N-1} \\ \delta y_N \end{pmatrix} = \begin{bmatrix} CA_0 \\ CA_1 A_0 \\ \vdots \\ CA_{N-2} \cdots A_1 A_0 \\ \hline CA_{N-1} \cdots A_1 A_0 \end{bmatrix} \delta x_0 + \begin{bmatrix} CB_0 & 0 & \cdots & 0 & 0 \\ CA_1 B_0 & CB_1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & 0 & 0 \\ CA_{N-2} \cdots A_1 B_0 & \cdots & \cdots & CB_{N-2} & 0 \\ \hline CA_{N-1} \cdots A_1 B_0 & \cdots & \cdots & CA_{N-1} B_{N-2} & CB_{N-1} \end{bmatrix} \begin{pmatrix} \delta u_0 \\ \delta u_1 \\ \vdots \\ \delta u_{N-2} \\ \delta u_{N-1} \end{pmatrix} \tag{3.15}$$

vagy tömör alakban

$$\begin{pmatrix} \delta y_1 \\ \delta y_2 \\ \vdots \\ \delta y_{N-1} \end{pmatrix} = P_1 \delta x_0 + H_1 \delta U$$
$$\delta y_N = P_2 \delta x_0 + H_2 \delta U \tag{3.16}$$

ahol

$$P_1 = \begin{bmatrix} p_1^T \\ \vdots \\ p_{N-1}^T \end{bmatrix}_{m(N-1)\times n}, \qquad H_1 = \begin{bmatrix} h_1^T \\ \vdots \\ h_{N-1}^T \end{bmatrix}_{m(N-1)\times Nr}, \qquad P_2 = \left[p_N^T \right]_{m\times n}, \qquad H_2 = \left[h_N^T \right]_{m\times Nr}$$

és $n = \dim x,\ r = \dim u,\ m = \dim y$.

Az optimalizálási probléma végfeltétellel a következő alakú:

$$J = \frac{1}{2}\sum_{i=1}^{N-1} \left\| e_i - \delta y_i \right\|^2 + \frac{1}{2}\lambda \sum_{i=0}^{N-1} \left\| \delta u_i \right\|^2 \to \min$$
$$e_N - \delta y_N = e_N - (P_2 \delta x_0 + H_2 \delta U) = 0 \tag{3.17}$$

A (5.4) jelölésekkel a költségfüggvény részletes alakja:

$$
\begin{aligned}
J = & \frac{1}{2}\sum_{i-1}^{N-1} < e_i, e_i > - < (\sum_{i=1}^{N-1} p_i e_i), \delta x_0 > \\
& - < (\sum_{i=1}^{N-1} h_i e_i), \delta U > + \frac{1}{2} < (\sum_{i=1}^{N-1} p_i p_i^T)\delta x_0, \delta x_0 > \\
& + < (\sum_{i=1}^{N-1} h_i p_i^T)\delta x_0, \delta U > \\
& + \frac{1}{2} < (\sum_{i=1}^{N-1} h_i h_i^T)\delta U, \delta U > + \frac{1}{2}\lambda < \delta U, \delta U > .
\end{aligned}
\tag{3.18}
$$

Mivel a költségfüggvény konvex és a korlátozás lineáris, ezért az optimum szükséges és elégséges feltétele a Lagrange multiplikátor szabály. Jelölje a μ vektor a Lagrange multiplikátorokat, akkor

$$
\begin{aligned}
L &= J + < \mu, e_N > - < P_2^T \mu, \delta x_0 > - < H_2^T \mu, \delta U > \\
0 &= \frac{dL}{d\delta U} = -\sum_{i=1}^{N-1} h_i e_i + (\sum_{i=1}^{N-1} h_i p_i^T)\delta x_0 \\
&\quad + (\sum_{i=1}^{N-1} h_i h_i^T)\delta U + \lambda \delta U - H_2^T \mu \\
&= -\overline{m} + H_1^T P_1 \delta x_0 + (H_1^T H_1 + \lambda I)\delta U - H_2^T \mu
\end{aligned}
\tag{3.19}
$$

ahol $\overline{m} = \sum_{i=1}^{N-1} h_i e_i$. A következő jelölésekkel az eredmények egyszerűbb alakban írhatók fel:

$$
L_1 := H_1^T H_1 + \lambda I, \;\; L_\mu := H_2 L_1^{-1} H_2^T \tag{3.20}
$$

Ekkor (5.7) és a végfeltétel korlátozás figyelembevételével:

$$
\begin{aligned}
\delta U &= L_1^{-1}(\overline{m} + H_2^T \mu - H_1^T P_1 \delta x_0) \\
e_N &- P_2 \delta x_0 - H_2[L_1^{-1}(\overline{m} + H_2^T \mu - H_1^T P_1 \delta x_0)] = 0 \\
\mu &= L_\mu^{-1}[e_N - H_2 L_1^{-1}\overline{m} - (P_2 - H_2 L_1^{-1} H_1^T P_1)\delta x_0]
\end{aligned}
$$

és ezért

$$
\begin{aligned}
\delta U = L_1^{-1}\{ & H_2^T L_\mu^{-1} e_N + (I - H_2^T L_\mu^{-1} H_2 L_1^{-1})\overline{m} \\
& - [H_1^T P_1 + H_2^T L_\mu^{-1}(P_2 - H_2 L_1^{-1} H_1^T P_1]\delta x_0 \}
\end{aligned}
\tag{3.21}
$$

A beavatkozó jel zárt körben $u_0 + \delta u_0$ ahol u_0 a nominális irányítás a horizont kezdetén és $\delta u_0 \in R^r$ a felnyitott körben optimális δU sorozat első eleme.

3.4.3 A megvalósított prediktív irányítási algoritmus:

Minden horizontban a következő lépések ismétlődnek:

<u>*1. lépés.*</u> Az x_0 kezdeti állapotból és az $\{u_0, u_1, \ldots, u_{N-1}\}$ nominális bemenő jel sorozatból meghatározásra kerül az $\{x_0, x_1, \ldots, x_N\}$ nominális állapot sorozat a jármű approximált nemlineáris dinamikus modellje alapján. Itt x_0 az eltolt előző horizontból jön, és eltérhet a becsült $\hat{x}_0$ kezdeti állapottól (az állapotbecslés beillesztése után).

Az előírt (desired) állapot sorozat a megtervezett CAS akadályelkerülő pályából számítható nulla β oldalirányú elcsúszási szög (slide slip angle) esetén. A kimenő jel $y = (X, Y)^T$, ezért a kívánt (desired) kimenő jel sorozat kiszámítható a kívánt (desired) állapot sorozatból a horizontban a $C = [e_5 \; e_6]^T$ mátrix segítségével, ahol kivételesen e_i az i-edik standard egységvektort jelöli. A

hibajel sorozat ezek különbsége. A legelső horizont esetén a nominális bemenő jel sorozat a DGA módszerrel kerül meghatározásra és x_0 inicializált értéke a megtervezett CAS akadályelkerülő pálya állapotvektora nulla oldalcsúszási szög (slide slip angle) esetén.

2. lépés. A diszkrétidejű $\delta x_{i+1} = A_i \delta x_i + B_i \delta u_i$ LTV rendszer meghatározása a $\dot{x} = f_c(x,u)$ folytonosidejű approximált nemlineáris dinamikus modellből Euler-formulával:
$A_i := I + T\, df_c / dx\big|_{(x_i,u_i)}$, $B_i := T\, df_c / du\big|_{(x_i,u_i)}$.

3. lépés. Az optimális δU bemenő jel változás sorozat meghatározása (5.9) alapján és $\delta x_0 = \hat{x}_0 - x_0$ felhasználásával, ahol $\hat{x}_0$ a becsült állapot (az állapotbecslés beiktatása után). Az optimális bemenő jel sorozat $U := U + \delta U$. A sorozat első u_0 eleme kerül kiadásra beavatkozó jelként zárt körben.

4. lépés. Azért, hogy inicializálható legyen a bemenő jel sorozat a következő horizont számára, az x_0 kezdeti állapothoz és az új $\{u_0, u_1, \ldots, u_{N-1}\}$ optimális bemenő jel sorozathoz az $\dot{x} = f_c(x,u)$ folytonosidejű approximált nemlineáris dinamikus modell felhasználásával meghatározásra kerül a nemlineáris rendszer állapot sorozata, az eredményt a tranziens végén x_N jelöli. Az ismeretlen u_N háromféleképpen határozható meg: i) u_N a DGA módszerrel kerül meghatározásra x_N felhasználásával. ii) u_N úgy lesz megválasztva, hogy az Euler-formulával képzett diszkrétidejű nemlineáris rendszer x_{N+1} válasza és a megtervezett CAS akadályelkerülő pálya $x_{d,N+1}$ állapota közötti differencia, amely $f(x_N) + G(x_N)u_N - x_{d,N+1}$, legyen minimális LS (least squares) értelemben. iii) Az utolsó bemenő jel az optimális bemenő jel sorozatban egyszerűen ismétlésre kerül: $u_N := u_{N-1}$.

5. lépés. A nominális bemenő jel sorozat a következő horizont számára $\{u_1, u_2, \ldots, u_N\}$, amely a kiegészített optimális bemenő jel sorozat $\{u_0, u_1, \ldots, u_N\}$ eggyel balra eltolva.

Lehetséges integrátor beillesztése a prediktív szabályozóba a $\delta x_i := (\delta x_i^T, \delta u_{i-1}^T)^T$ bővített (augmented) állapot bevezetésével, ahol $\delta u_i = \delta u_{i-1} + \delta r_i$, ekkor azonban a bemenő jel δr_i változását kell optimalizálni. Bevezetve a

$$A_i := \begin{bmatrix} A_i & B_i \\ 0 & I \end{bmatrix} \text{ és } B_i := \begin{bmatrix} B_i \\ I \end{bmatrix} \tag{3.22}$$

helyettesítéseket, a korábbi eredmények az új változókkal érvényben maradnak. Mindazonáltal ebben az esetben δR lesz az optimális bemenő jel változás, amely meghatározásra kerül, és az optimális δU bemenő jel sorozat a kumulatív összege δR -nek.

A nemlineáris prediktív irányítási algoritmus c_mode=2 esetén a `contr` törzsében kerül megvalósításra, de hívja még a következő függvényeket is, melyeket függvény prototípus alakjukkal és a funkciót körvonalazó egysoros comment-tel adunk meg:

```
function [xdot,F_lR,deltaw,Sv]=dgi_controller(ti,xi,T_samp,...
      t_breaks,x_coeffs,y_coeffs,...
      c_F,l_F,c_R,l_R,m_v,I_zz,...
      lambda,alpha0i,alpha1i)
%Implement dg_controller for control horizon initialization

function [AA,BB]=LTV2vehicle(xx,uu,Ts,LTVhor,...
      c_F,l_F,c_R,l_R,m_v,I_zz,deltawinput)
%Linearize vehicle dynamic model along given trajectory and control
%Approximated model, u=(Sv,F_lR)' if deltawinput=0,
%otherwise u=(deltaw,F_lR)'

function [AAA,BBB,CCC]=preint_ab2ph(AA,BB,C)
%Compute AAA,BBB,CCC for integral control
```

```
function [P1,H1,P2,H2,mvec,eN]=ab2ph(AA,BB,C,ee,inthor)
%Compute PP and HH from AA and BB for use in LTV RHC control

function [dfdx,dfdu]=dfapprdx(x,u,...
      c_F,l_F,c_R,l_R,m_v,I_zz,deltawinput)
%First derivative of f(x,u) (approximated)

function xdvect=funxdvec(t,t_breaks,x_coeffs,y_coeffs)
%Compute desired state from CAS path database

function Sv=deltaw2Sv(deltaw,x,c_F,l_F);
%Convert deltaw to Sv using approximated modell

function deltaw=Sv2deltaw(Sv,x,c_F,l_F);
%Convert Sv to deltaw using approximated modell

function xx=uux02xx(uu,x0,t0,deltawinput,Tsamp,...
      c_F,l_F,c_R,l_R,m_v,I_zz)
%Compute xx tansient belonging to uu and x0

function [xdot,F_lR,deltaw,Sv,f,G]=apprfunxdot(t,x,F_lR,deltaw,...
      c_F,l_F,c_R,l_R,m_v,I_zz)
%Compute right side of approximated vehicle dynamic model

function [xt,d1xt,d2xt,d3xt,yt,d1yt,d2yt,d3yt]=...
      embed_path(t,t_breaks,x_coeffs,y_coeffs)
%Compute reference signals and derivatives

function [yy,d1yy,d2yy,d3yy]=embed_deriv3(xx,breaks,coeffs)
%Embedded deriv3()

function ix=embed_maxfind(x,x0)
%Embedded simple find
```

3.5 GPS/INS érzékelők jelein és kétszintű Kalman-szűrőn alapuló állapotbecslés

A reális rendszerekben, így járművekben is, az állapotváltozók többsége nem mérhető, ezért becslésüket az érzékelők mérhető jeleire kell alapozni. Feltételeztük, hogy az érzékelők jeleit 2-antennás differenciális GPS rendszer jelei és egy INS rendszer (Inertial Navigation System) gyorsulás érzékelő/szögsebesség jelei szolgáltatják.

A GPS/INS jelek kiértékelését (Ryu and Gerdes, 2004) eredményeire alapoztuk. A berendezések a drivereik segítségével magas szintű, állapotbecslésre alkalmas információt szolgáltatnak, amelyek a következők:

V_m^{GPS} : a jármű (mért, erre utal az m-index) sebességvektora a GPS földi inerciarendszerében

ψ_m^{GPS} : a jármű orientációja a GPS földi inerciarendszerében

$a_{x,m}$: a jármű x -irányú (longitudinális) gyorsulása a járműhöz rögzített koordináta-rendszerben

$a_{y,m}$: a jármű y -irányú (transzverzális) gyorsulása a járműhöz rögzített koordináta-rendszerben

r_m : a jármű z -irányú (yaw-irányú) szögsebessége a járműhöz rögzített koordináta-rendszerben

Feltesszük, hogy az első antenna közvetlenül az INS rendszer felett helyezkedik el, és az INS rendszer a tömegközéppontban (COG) van. (Az általános esetben változtatások szükségesek.)

Az érzékelők drift-tel (bias) is rendelkeznek, amelyeket a Kalman-szűrőknek szintén becsülni kell. Az érzékelők pontosságát a $\sigma 1$ érték és a bias jellemzi. Az additív Gauss-zaj (noise) lényegében a mért érték körüli $[-3\sigma, 3\sigma]$ sávba esik statisztikailiag, amelyhez még hozzáadódik a bias értéke.

3.5.1 GPS/INS jelek primer feldolgozása

Vízszintes síkban haladó jármű esetén $V_m^{GPS} = (V_1^{GPS}, V_2^{GPS}, 0)^T$ alakú, ahonnan meghatározható az oldalcsúszási szög mért értéke, abból pedig a járműhöz rögzített koordináta-rendszerben az u sebességvektor komponensei:

$$\begin{aligned} \gamma &= \text{atan2}(\text{V}_2^{\text{GPS}}, V_1^{GPS}) \Rightarrow \beta^{GPS} = \gamma - \psi_m^{GPS} \\ u_{x,m}^{GPS} &= \left\| V^{GPS} \right\| \cos(\beta^{GPS}) + noise \\ u_{y,m}^{GPS} &= \left\| V^{GPS} \right\| \sin(\beta^{GPS}) + noise \end{aligned} \tag{3.23}$$

Az állapotbecslést 2 fokozatból álló Kalman-szűrőre alapozzuk.

3.5.2 Első Kalman-szűrő fokozat: a szögsebesség becslése

Az első fokozat becsli a $\dot{\psi} = r$ szögsebességet. Erre a következő két módszer valamelyike javasolható:

$$\begin{aligned} \begin{pmatrix} \dot{\psi} \\ \dot{r}_{bias} \end{pmatrix} &= \begin{bmatrix} 0 & -1 \\ 0 & 0 \end{bmatrix} \begin{pmatrix} \psi \\ r_{bias} \end{pmatrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix} r_m + noise \\ \psi_m^{GPS} &= \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{pmatrix} \psi \\ r_{bias} \end{pmatrix} + noise \end{aligned} \tag{3.24}$$

$$\begin{aligned} \frac{d}{dt} \begin{pmatrix} \psi \\ 1/s_r \\ r_{bias}/s_r \end{pmatrix} &= \begin{bmatrix} 0 & r_m & -1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{pmatrix} \psi \\ 1/s_r \\ r_{bias}/s_r \end{pmatrix} + noise \\ \psi_m^{GPS} &= \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} (\psi \quad 1/s_r \quad r_{bias}/s_r)^T + noise \end{aligned} \tag{3.25}$$

ahol s_r a giroszkóp érzékenysége. Az állapotbecslést diszkrétidejű Kalman-szűrő végzi, ehhez a választott folytonosidejű modellről diszkrétidejűre kell áttérni. Jelölje T a mintavételi időt, akkor a diszkrétidejű modellek mátrixai a következők:

$$A_{d1} = \begin{bmatrix} 0 & -T \\ 0 & 1 \end{bmatrix},\ B_{d1} = \begin{bmatrix} T \\ 0 \end{bmatrix},\ C_{d1} = \begin{bmatrix} 1 & 0 \end{bmatrix} \tag{3.26a}$$

$$A_{d1} = I_3 + A_{c1}T,\ B_{d1} = 0_{3\times 1},\ C_{d1} = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \tag{3.26b}$$

Az állapotbecslés eredményéből a szögsebesség becsült értéke kifejezhető a választott modell alapján:

$$r = \hat{\dot{\psi}} := -\hat{r}_{bias} + r_m \tag{3.27a}$$

$$r = \hat{\dot{\psi}} := r_m(1/\hat{s}_r) - (\hat{r}_{bias}/\hat{s}_r) \tag{3.27b}$$

3.5.3 Második Kalman-szűrő fokozat: a sebesség becslése

Az első szinten meghatározott szögsebesség becslés értéke felhasználásra kerül a második Kalman-szűrőben a jármű u_x, u_y sebességkomponenseinek becslésekor a járműhöz rögzített koordináta-rendszerben. A becslés alapja az $a = \dot{u} + \omega \times u$ összefüggés, amely a következő folytonosidejű modellt eredményezi:

$$\frac{d}{dt}\begin{pmatrix} u_x \\ a_{x,bias} \\ u_y \\ a_{y,bias} \end{pmatrix} = \begin{bmatrix} 0 & -1 & r & 0 \\ 0 & 0 & 0 & 0 \\ -r & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 \end{bmatrix}\begin{pmatrix} u_x \\ a_{x,bias} \\ u_y \\ a_{y,bias} \end{pmatrix} + \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}\begin{pmatrix} a_{x,m} \\ a_{y,m} \end{pmatrix} + noise$$

$$\begin{pmatrix} u_{x,m}^{GPS} \\ u_{y,m}^{GPS} \end{pmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}\begin{pmatrix} u_x \\ a_{x,bias} \\ u_y \\ a_{y,bias} \end{pmatrix} + noise \tag{3.28}$$

A modell a valós időben változó r értékét tartalmazza, ezért előnyös, hogy a folytonos időről diszkrét időre áttérés eredménye analitikusan megadható:

$$A_{d2} = \begin{bmatrix} \cos(rT) & -\sin(rT)/r & \sin(rT) & -[1-\cos(rT)]/r \\ 0 & 1 & 0 & 0 \\ -\sin(rT) & [1-\cos(rT)]/r & \cos(rT) & \sin(rT)/r \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$B_{d2} = \begin{bmatrix} \sin(rT)/r & [1-\cos(rT)]/r \\ 0 & 0 \\ -[1-\cos(rT)]/r & \sin(rT)/r \\ 0 & 0 \end{bmatrix} \tag{3.29}$$

$$C_{d2} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Mivel a gyakorlati esetben $r \approx 0$ nem kizárható, ezért a kritikus határértékeket a L'Hospital-szabállyal analitikusan határoztuk meg a szoftver implementáció számára. Az állapotbecslést a második Kalman-szűrő végzi. A becslésből kapott $\hat{u}_x, \hat{u}_y$ és az első Kalman-szűrő eredményeként kapott $\hat{\psi}$ értékéből további állapotváltozók becslése határozható meg:

$$\hat{v}_G = \sqrt{\hat{u}_x^2 + \hat{u}_y^2}; \quad \hat{\beta} = \text{atan2}(\hat{u}_y, \hat{u}_x) \tag{3.30}$$

A hiányzó X, Y pozíció jellegű állapotváltozókat a Kalman-szűrőkkel kapott becsült értékekből téglalap szabályon alapuló numerikus integrálással határozzuk meg:

$$\hat{X} := \hat{X} + T\hat{v}_G \cos(\hat{\psi} + \hat{\beta}); \quad \hat{Y} := \hat{Y} + T\hat{v}_G \sin(\hat{\psi} + \hat{\beta}) \tag{3.31}$$

A Kalman-szűrőket az alábbi alakban implementáltuk:

$$\begin{aligned} x_-(t+1) &= A_d x_+(t) + B_d u(t) \\ P_-(t+1) &= A_d P_+(t) A_d^T + Q \end{aligned} \quad \text{(time update)} \qquad (3.32a)$$

$$\begin{aligned} x_+(t) &= x_-(t) + K[y(t) - Cx_-(t)] \\ K &= P_-(t)C^T[CP_-(t)C^T + R]^{-1} \quad \text{(measurement update)} \\ P_+(t) &= [I - KC]P_-(t) \end{aligned} \qquad (3.32b)$$

A szenzormérések tipikus mintavételi ideje $T_{INS} = T = 0.01s$ $(100Hz)$, $T_{GPS,vel} = 0.1s$ $(10Hz)$, $T_{GPS,att} = 0.2s$ $(5Hz)$, ahol T a szabályozás mintavételi ideje. Az eltérő mintavételi időket az állapotbecsléskor figyelembe kell venni.

Az állapotbecslés algoritmusa `estim` törzsében kerül megvalósításra, de hívja még a következő függvényeket is, melyeket függvény prototípus alakjukkal és a funkciót körvonalazó egysoros comment-tel adunk meg:

```
function [xp,xm,Pp,Pm,R,Q]=KalmanFilter(xp,xm,Pp,Pm,R,Q,...
      Ad,Bd,Cd,um,ym)
%Compute xhat by Kalman Filter

function x=x12tox(Ts,x,x1,x2,rm,gyrosens_corr)
%Convert x1 and x2 to x
```

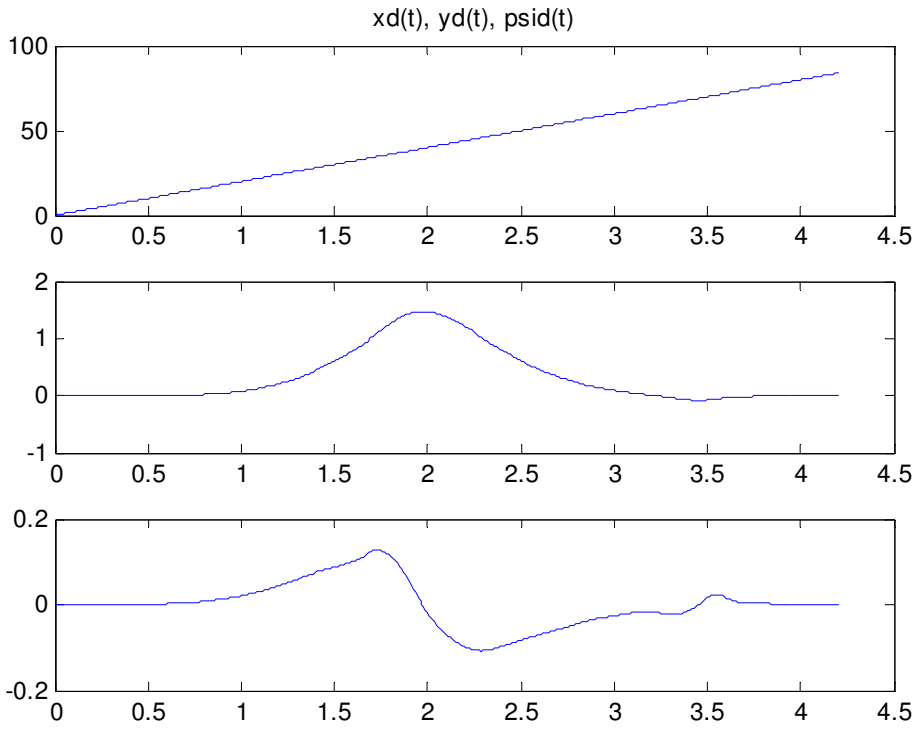
4. A szabályozások tesztelése Matlab/Simulink környezetben

A szabályozásokat megvalósító Simulink modell a kifejlesztett beágyazott függvényekre épül, hogy jó esély legyen kifordításukra target processzorra. A Matlab/Simulink alatti tesztelés alkalmas az irányítási algoritmus tesztelésére, de még nem valós idejű. Mindazonáltal erre a tesztelésre szükség van a target processzorra való kifordítás megkisérlése előtt, mert ez a technika képes a beágyazott függvények specialitásainak előzetes és futás közbeni tesztelésére, és sikeressége esetén van csak esély a kifordításra. A továbbiakban bemutatjuk a Matlab/Simulink környezetben kapott futtatási eredményeket, amelyek meggyőzően bizonyítják a szabályozástechnikus számára kifejlesztett algoritmusok és a beágyazott függvényekre épülő Simulink modell helyességét.

4.1 A differenciálgeometriai módszeren alapuló szabályozás tesztje

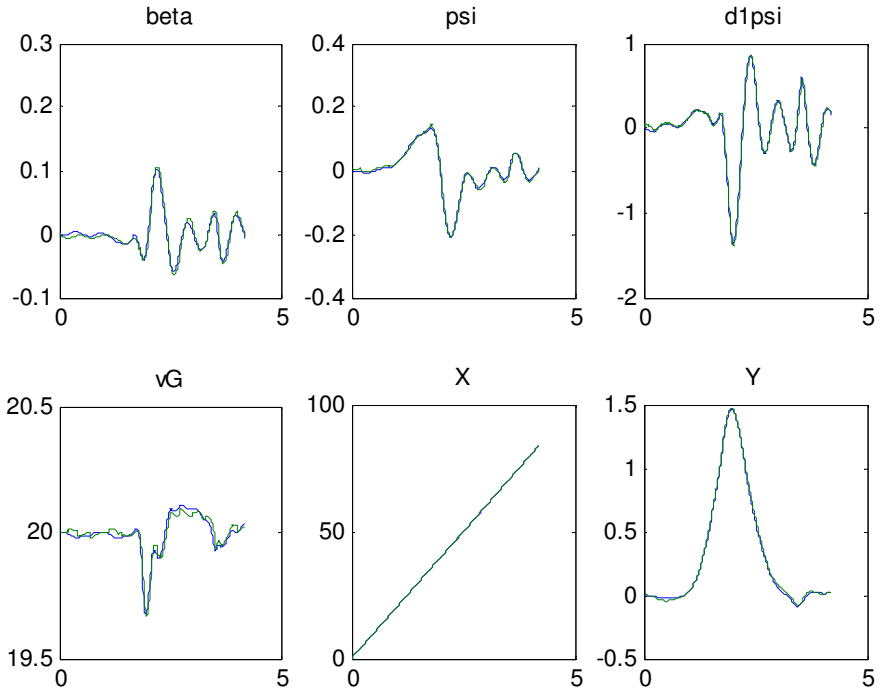
A differenciálgeometriai módszeren (DG) alpuló irányítás szimulációs eredményei 10ms fix lépésköz és ode1 (Euler) integrálási módszer esetén készültek, mely a target processzorra kifordításkor előírás lesz. A szimulációs eredmények munkaterületen (Workspace) tárolódtak, ahonnan az erre a célra kifejlesztett `plot_ttyyd`, `plot_ttyy`, `plot_ttxxh`, `plot_ttuu`, `plot_ttee` függvényekkel lettek felrajzolva. Minden jel SI mértékegységben (m, rad, m/s, N stb.) értendő.

A referencia jeleket (desired values) a *4.1. ábra* mutatja be.

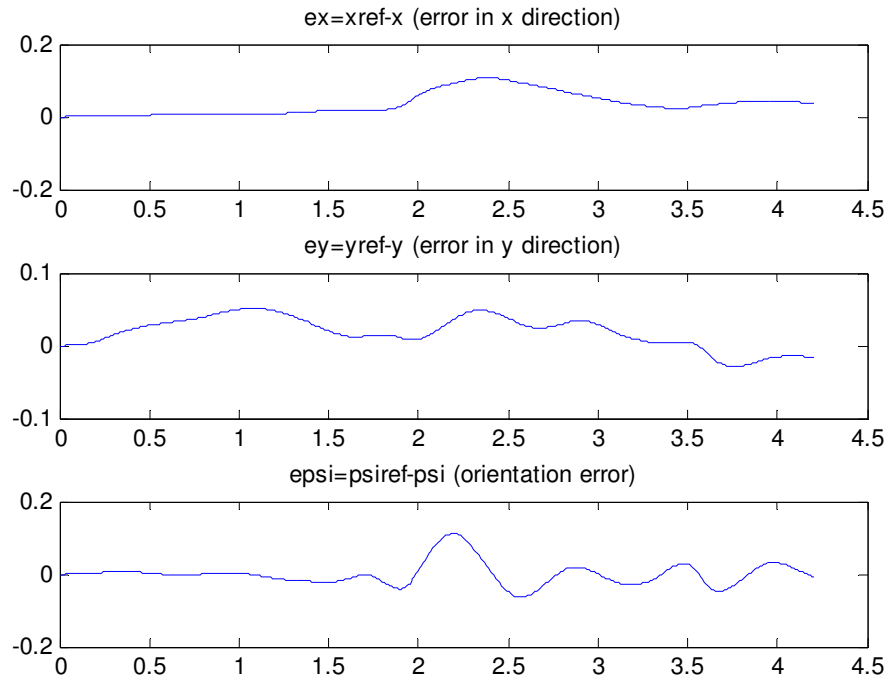


4.1. ábra. Referencia jelek Matlab/Simulink alatt DG irányítás esetén

Az állapotváltozókat és becsült értéküket a *4.2. ábra* mutatja be. Jól látható, hogy a kétszintű kiterjesztett Kalman-szűrő jó állapotbecslést biztosít. Lényeges kiemelni, hogy a `cardyn`-ben szereplő `Int6` integrátorok szintén ode1 (Euler) mellett adnak csak jó eredményt, minden finomabb integrálás elrontja az $x(t)$ függvényt és vele a többi jelet is.

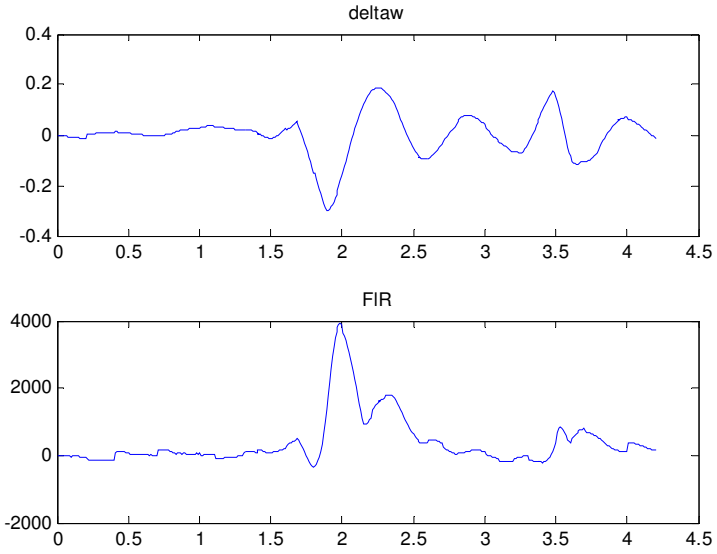


4.2. ábra. Állapotváltozók és becsült értékeik Matlab/Simulink alatt DG irányítás esetén



4.3. ábra. A pozíció és orientáció hibajelek Matlab/Simulink alatt DG irányítás esetén

A kormányszög és longitudinális gyorsító erő beavatkozó jeleket a *4.4. ábra* mutatja be. Jól látható, hogy a beavatkozó jelek értékei SI egységben mérve reálisak.



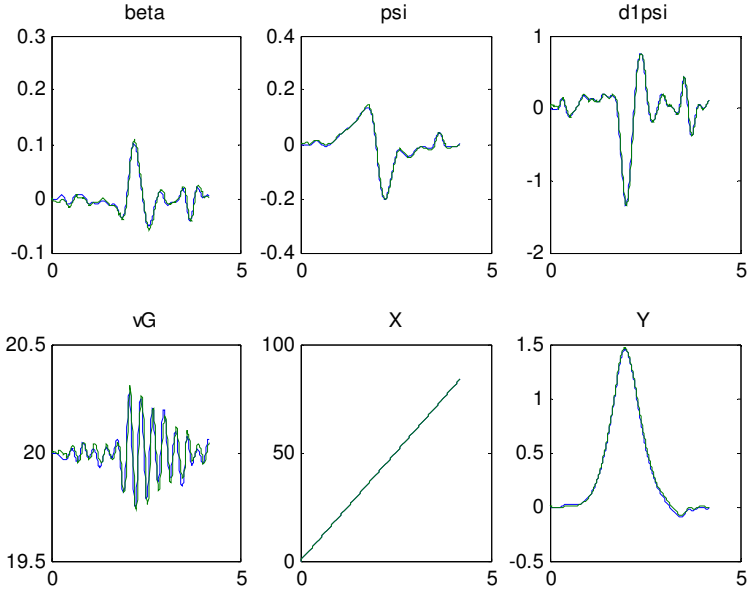
4.4. ábra. Kormányszög és longitudinális gyorsító erő beavatkozó jelek Matlab/Simulink alatt DG irányítás esetén

4.2 Nemlineáris prediktív irányítás tesztje

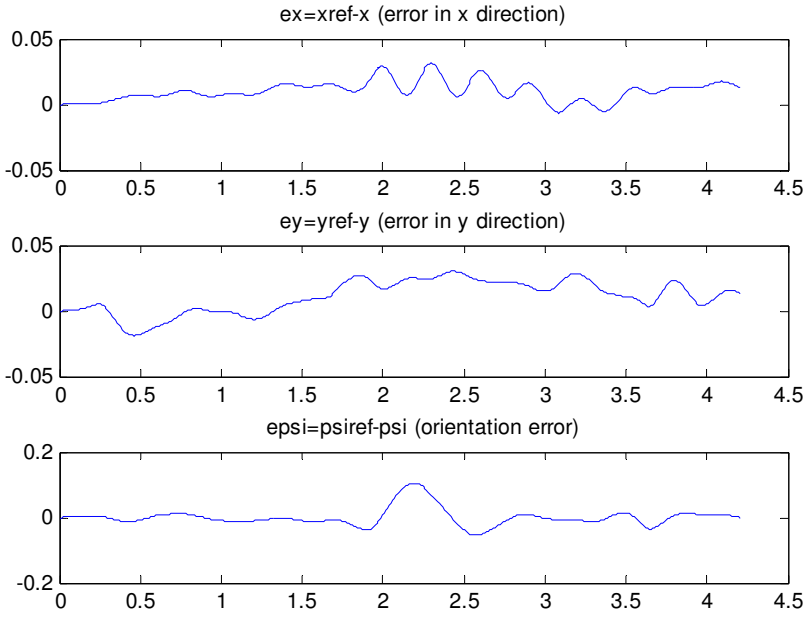
A nemlineáris prediktív irányítás (RHC, receding horizon control) szimulációs eredményei 10ms fix lépésköz és ode1 (Euler) integrálási módszer esetén készültek, mely a target processzorra kifordításkor előírás lesz. A horizont méret N=10 (100ms). Minden horizont kezdetén egy új LTI modell kerül meghatározásra, amely a nemlineáris modellt approximálja a horizont alatt.

4.2.1 Integrátort tartalmazó prediktív irányítás tesztje

Elsőként az integrátort is tartalmazó RHC irányítást vizsgáljuk. A referencia jel megegyezik a *4.1. ábrán* adottal. Az állapotokat és becsült értéküket a *4.5. ábra*, a hibajeleket a *4.6. ábra* mutatja be.

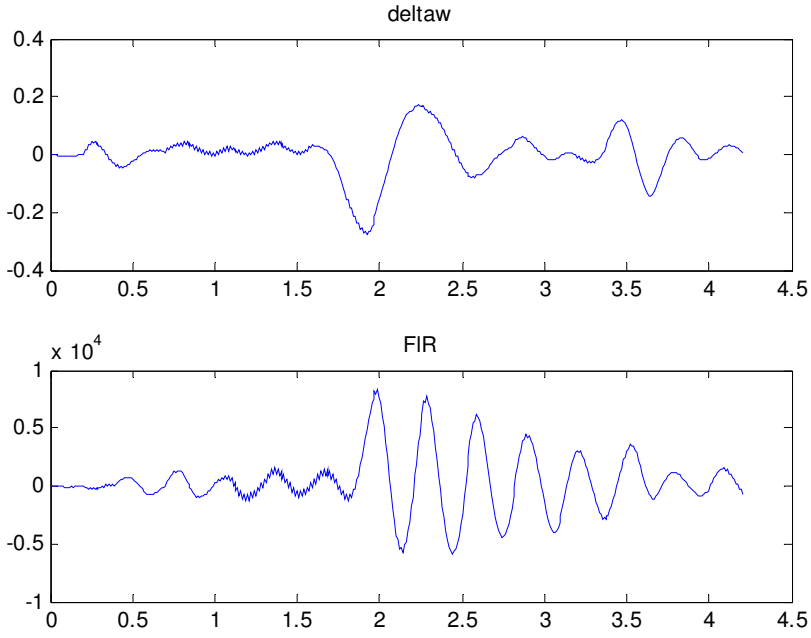


4.5. ábra. Állapotváltozók és becsült értékeik Matlab/Simulink alatt integrátort tartalmazó RHC irányítás esetén



4.6. ábra. A pozíció és orientáció hibajelek Matlab/Simulink alatt integrátort tartalmazó RHC irányítás esetén

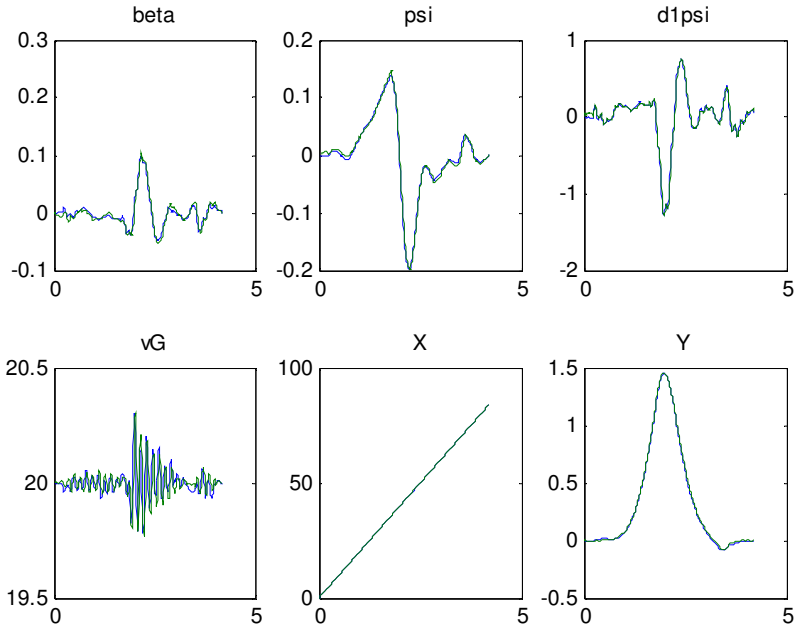
A kormányszög és longitudinális gyorsító erő beavatkozó jeleket a *4.7. ábra* mutatja be. Jól látható, hogy a beavatkozó jelek értékei SI egységben mérve reálisak.



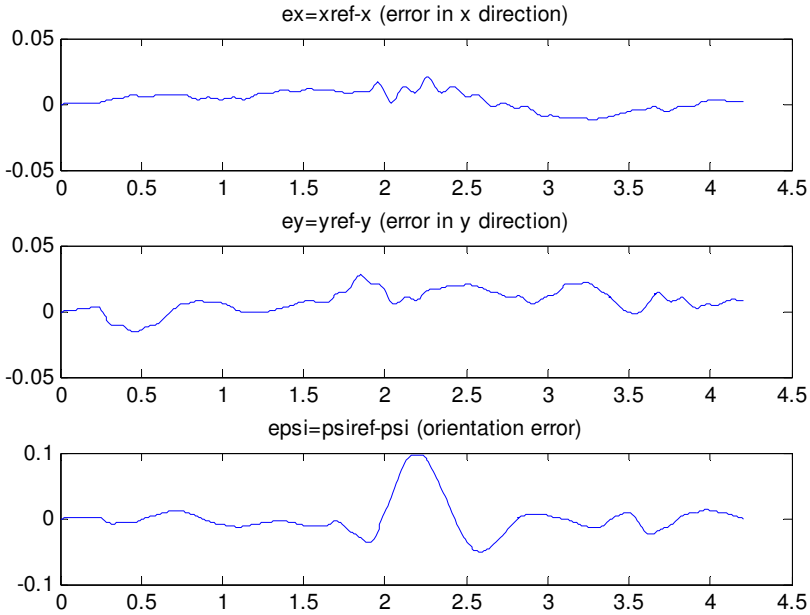
4.7. ábra. Kormányszög és longitudinális erő beavatkozó jelek Matlab/Simulink alatt integrátort tartalmazó RHC irányítás esetén

4.2.2 Integrátort nem tartalmazó prediktív irányítás tesztje

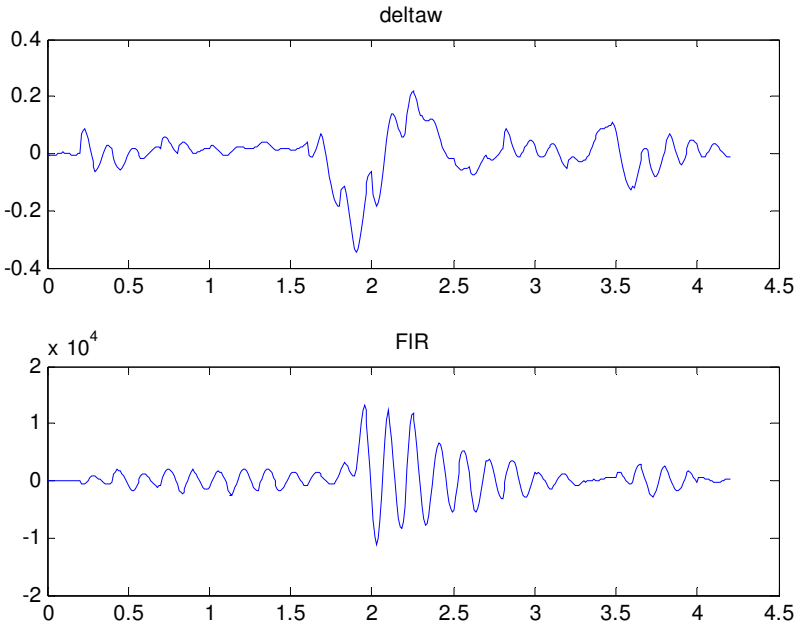
Másodikként az integrátort nem tartalmazó RHC irányítást teszteljük. A referencia jel megegyezik a *4.1. ábrán* adottal. Az állapotokat és becsült értéküket a *4.8. ábra*, a hibajeleket a *4.9. ábra,* a beavatkozó jeleket pedig a *4.10. ábra* mutatja be.



4.8. ábra. Állapotváltozók és becsült értékeik Matlab/Simulink alatt integrátort nem tartalmazó RHC irányítás esetén



4.9. ábra. A pozíció és orientáció hibajelek Matlab/Simulink alatt integrátort nem tartalmazó RHC irányítás esetén



4.10. ábra. Kormányszög és longitudinális erő beavatkozó jelek Matlab/Simulink alatt integrátort nem tartalmazó RHC irányítás esetén

5. A szabályozások tesztelése dSPACE AutoBox környezetben

A teszteléshez rendelkezésre kell állni a dSPACE AutoBox hardver rendszernek és a ControlDesk szoftver felügyelő rendszernek, továbbá eme rendszerek hardver kulcsának. Ezen kívül szükséges a Matlab R2006a rendszer a Real Time Workshop kiegészítéssel és a dSPACE 1005 board target compiler.

Az AutoBox a BME IIT Tanszéken külön egységet alkot, amely hálózaton keresztül érhető el az IB/313 Intelligens Robotok labor számítógépjeiről. A hardver kulcsot abba a PC-be kell betenni, amely az AutoBox-ra fordítást és a valós idejű tesztelést végzi. Értelemszerűen az AutoBox egységnek be kell kapcsolva lennie.

A Matlab/Simulink R2006a és a Real Time Workshop licensz a BME IIT Tanszék tulajdona. Az AutoBox, ControlDesk és ezek hardver kulcsa az EJJT Tudásközpont tulajdona, melyet a fejlesztés idejére a BME IIT Tanszék kölcsön kapott az EJJT Tudásközponttól.

5.1 Általános korlátok dSPACE AutoBox környezetben

A dSPACE target compiler elvárja, hogy csak olyan tömb specifikációk forduljanak elő a beágyazott függvényekben, amelyet a fordító a fordítási fázisban statikus méretű helyfoglalássá tud konvertálni. Ezért alternatívák, melyek eltérő méretű tömböket eredményeznének nem fordulhatnak elő. Ez azt jelenti, hogy ha a beágyazott függvény funkcionálisan különböző alternatívákra lehetne képes, akkor az ezt vezérlő paraméterek értékét a beágyazott függvény fejlécében fixen be kell állítani, és ehhez kell elvégezni a fordítást a target processzorra. Más alternatívához át kell írni a paramétereket, és újra el kell végezni az ezekhez tartozó fordítást a target processzorra. A `contr` beágyzott függvény fejlécében szereplő paraméterek erre a célra szolgálnak.

Fordítás előtt a következő lépéseket kell elvégezni:

1. Meg kell győződni a teljes rendszert modellező `closedsys.mdl` helyes működéséről Matlab/Simulink alatt. Ennek során fix lépésköz, ode1 (Euler) integrálási módszer és T_samp=0.01 szimulációs lépésköz beállításnak kell érvényesnek lennie.
2. A Simulink modell `Optimization` felparaméterezésében törölni kell a "pipákat" `Block reduction, Conditional input, Signal storage` előtt.
3. A Real Time Workshop számára módosítani kell a `System target file` értékét `rti1005.tlc`-re.
4. RTI simulation options számára be kell állítani az `Initial simulation state` értékét `STOP`-ra.
5. RTI variable description számára ki kell jelölni "pipával" az `Include mask and workspace` igényt.
6. Ezután el kell indítani a fordítást DS1005 target processzorra az `Incremental build` ikonnal (lefelé mutató nyilak). A fordításhoz hardver kulcs kell, az AutoBox-nak be kell kapcsolva lennie. Helyes fordítás végén `### Successful compilation ...`, `Finished RTI build ...` üzeneteket kapunk, és a betöltés megtörténik az AutoBox-ra.

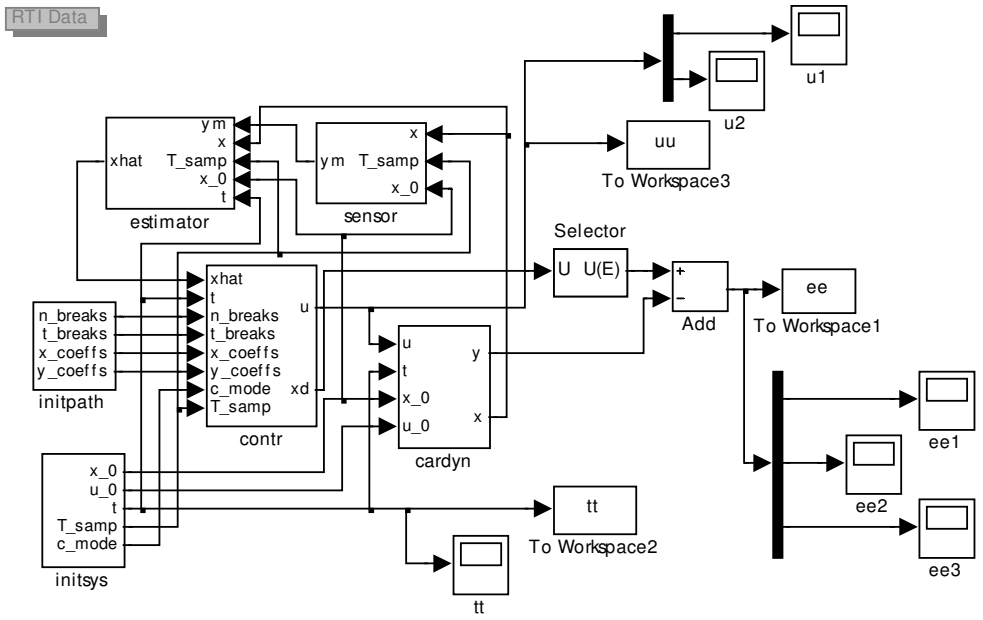
5.2 Futtatás előkészítése AutoBox rendszeren ControlDesk alatt

Futtatás előtt a következő lépéseket kell elvégezni:

1. El kell indítani a `ControlDesk` programot, ráklikkelve a PC-n lévő ikonjára.
2. Be kell állítani a hálózati kapcsolatot Change connection–>Network connection–>Network client: 192.168.240.149 OK szekvenciával.

3. A felső ikonsoron esetleg takarítást is végezve ki kell választani a “Load Application/Model” fület és be kell állítani a modell könyvtárat *.sdf megjelenő fül alatt: closedsys +Model Root+STOP, ami után látszanak a modell alrendszerei.
4. ControlDesk Selection Box szolgáltatásával előállítandók a szimuláció START|STOP|PAUSE állapotai és kiválaszthatók a gyűjtendő jelek a layout felületen. Erre a célra a closedsyslayout2 program lett létrehozva.
5. Ezután a szimuláció elindítható.
6. A szimuláció végén a closedsyslayout2 `save` gombjára ráklikkelve megadandó a *.mat fájl neve, ahol a jelek elmenthetők, majd a szimuláció végeztével Matlab alá beolvashatók és felrajzolhatók.

A ControlDesk adottságaihoz igazodva, amely csak oszcilloszkóp terminátorokat tud *.mat fájlba menteni és a Matlabnak analízisre átadni, a *2.1. ábrához* képest új terminátorokat vezettünk be, megőrizve a korábbi beágyazott függvényeket. Az új Simulink modell az *5.1. ábrán* látható. A szimuláció során a tranzienseket a `dspace_rt_sim.mat` fájlban kell elmenteni, amely hatására az adatok egy struktúrában kelekeznek, ahonnan később a Matlab alá beolvashatók (`load`) és a `convert` saját függvénnyel felrajzolhatók. A *.m fájl nevét a closedsyslayout2 program layout elrendezése esetén a `save` paramétereként lehet megválasztani, `save` nélkül nincs tárolás fájlban.



5.1. ábra. A ControlDesk alatt használt Simulink modell oszcilloszkóp terminátorokkal

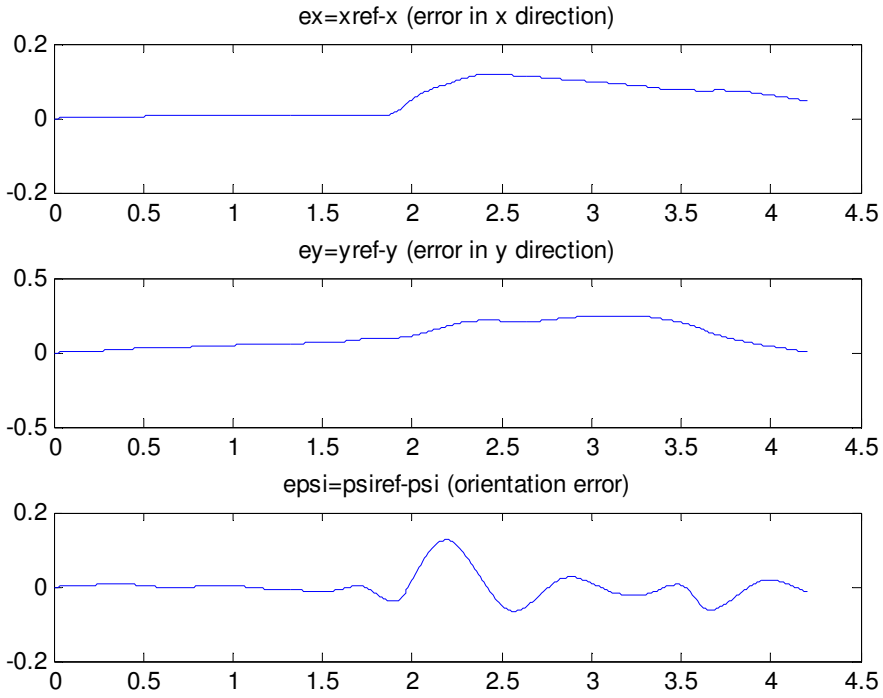
Az új terminátorok mellett elvégeztük a Simulink modell fordítását a target processzorra.

A fordítás eredményei:
A szabályozás Simulink modellje mindhárom üzemmódban hibátlanul lefordítható volt a DS1005 target processzorra, amit a 6.5 pontban szereplő lista demonstrál:

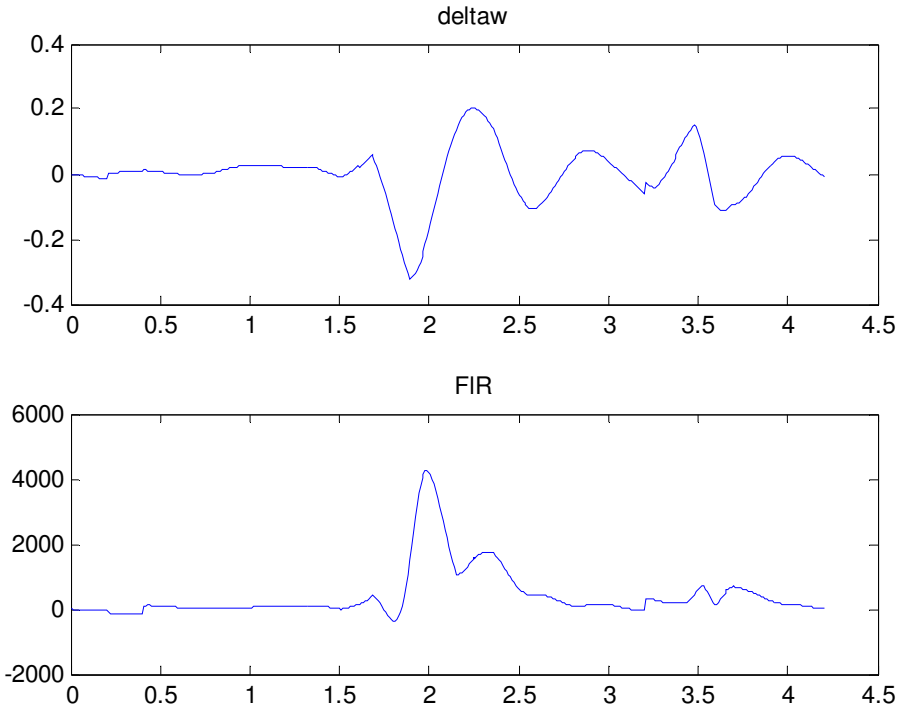
5.3 Futtatási eredmények és anomáliák AutoBox rendszeren ControlDesk alatt

5.3.1 Futtatási eredmények differenciálgeometriai elvű irányítás esetén

Ez az irányítás c_mode=1 esetén jut érvényre. A Simulink alatt kapott eredményeket a 4.1-4.4. ábrák tartalmazták. Az AutoBox rendszerrel kapott tranzienseket most az 5.1-5.2. ábrák mutatják be, az eredmények kissé eltérnek a 4.3-4.4. korábbi eredményektől.



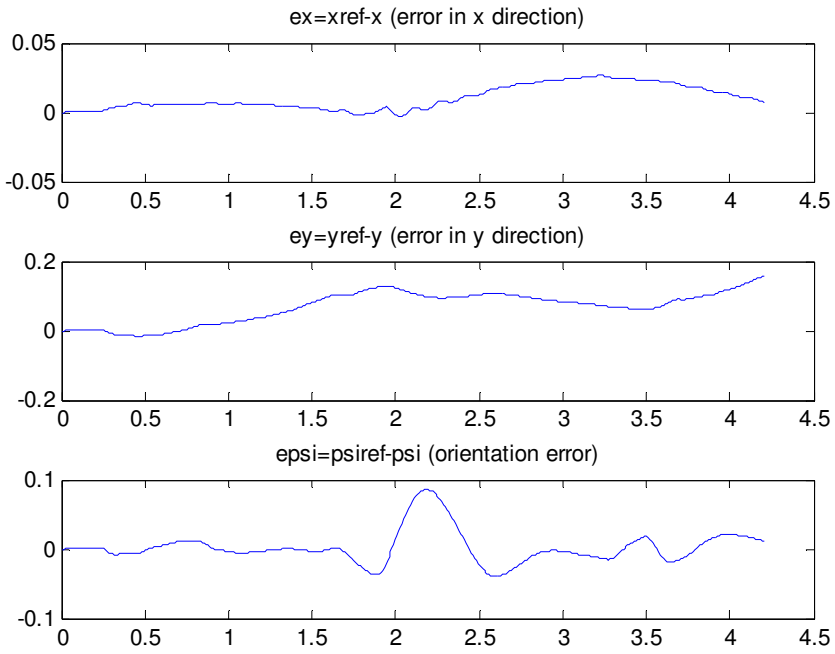
5.1. ábra. A pozíció és orientáció hibajelek AutoBox alatt DG irányítás esetén



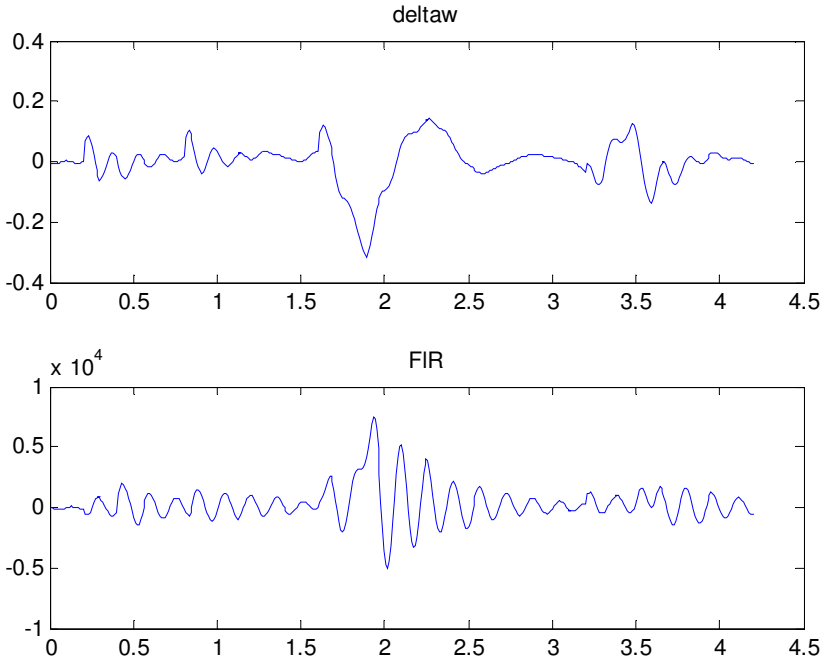
5.2. ábra. Kormányszög és longitudinális gyorsító erő beavatkozó jelek AutoBox alatt DG irányítás esetén

5.3.2 Futtatási eredmények integrátor nélküli prediktív irányítás esetén

A Simulink alatt a korábbi eredményeket a 4.8-4.10. ábrák tartalmazták. Az AutoBox alatt kapott eredményeket az 5.3-5.4. ábrák tartalmazzák. Ezeket összevetve a 4.9-4.10. ábrák eredményeivel, csak az y-irányú hibák tekintetében van jelentősebb eltérés, a Simulink alatt kisebb, az AutoBox alatt nagyobb a hiba.



5.3. ábra. A pozíció és orientáció hibajelek AutoBox alatt integrátor nélküli prediktív irányítás esetén



5.4. ábra. Kormányszög és longitudinális gyorsító erő beavatkozó jelek AutoBox alatt integrátor nélküli prediktív irányítás esetén

5.3.3 Futási anomália integrátort tartalmazó prediktív irányítás esetén

Bár a fordítás a DS1005 target compiler-rel c_mode=2 és integráló prediktív irányítás esetén is szabályosan befejeződött, a végrehajtáskor a ControlDesk és AutoBox alatt t=0 szimulációs időnél a következő hibaüzenet után a végrehajtás abortálódott:

ds1005: Excetion at address 0xC25C: alignement error (49756)
[#7] ds1005 – RTLIB: Applocation terminated. (14)

A hibaüzenet oka és indokoltsága a rendelkezésre álló Real Time Workshop és dSPACE anyagokból nem állapítható meg, feltehetőleg software bag-re vezethető vissza. Software bag-gel találkoztunk már más helyen is, például dim=2 esetén az svd lefordított C-kódjába belenézve elemi tömb indexelési hiba (explicit -1 index) volt detektálható.

5.3.4 Értékelés

Az implementált 3 szabályozási módszer küzül az AutoBox alatt 2 sikeres volt, a harmadik azonban a Real Time Workshop és/vagy a dSPACE szoftver rejtett hibái miatt indítás után elabortálódott, annak ellenére, hogy a Simulink alatt helyesen működött a szabályozási rendszer és a hibátlan fordítást protokoll dokumentálta. Jelenlegi tapasztalataink szerint a DS1005 target compiler rejtett hibákkal rendelkezhet a belső függvények (target functions) fordításakor, amelyek a rendelkezésre álló eszközökkel nem határolhatók be.

Gyanus lehet a nem elemenkénti (blokk-) mátrixműveletek fordítása C-kóddá (a Matlab oszlopfolytonosan tárol, a C ettől eltérhet), valamint a stack és más memóriaterületek méretezése a végrehajtható kódban a ControlDesk számára. Ezek analízise és okainak behatárolása jelenleg nem volt lehetséges.

6. A CAS rendszer Simulink modelljében használt beágyazott függvények listái

Az akadályelkerülő rendszer (CAS) Simulink modelljét a *2.1. ábra* mutatta be. A 3. fejezetben megadtuk az alkalmazott algoritmusok és a beágyazott függvények kapcsolatát a függvény prototípusok és tömör comment-jeik alakjában. A beágyazott függvények struktúra tekintetében követik a *2.1. ábra* Simulink modelljét. Minden egység a struktúrában egy-egy beágyzott függvény, amely belsejében további függvényeket tartalmaz. A struktúra egyes egységei önmagukban zártak, közöttük adatkapcsolat csak a Simulink blokkok interfészén keresztül lehetséges. A `contr` és `estimator` blokkok a szabályozó részei, a többi egység (`sensor`, `cardyn`) beágyazott függvényei a tesztelést támogatják.

6.1 A szabályozót megvalósító beágyazott függvény

```
function [u,xd]=contr(xhat,t,n_breaks,t_breaks,x_coeffs,y_coeffs,...
    c_mode,T_samp)
% Compute controller output
% This block supports an embeddable subset of the MATLAB language.
% See the help menu for details.

%
%-------------------------------------
% Parameters for predictive control
N_horizon=10;       %horizon length: 10 for T_samp<=0.025, otherwise 5
deltaw_horizon=1;   %1=>u(1)=deltaw, 0=>u(1)=Sv
dgfresh_horizon=1;  %1=>u(N) by dg, 0=>uN makes xNp1=0, otherwise uN=uNm1
lambda_horizon=[1000 0.1];  %weight for u or du
int_horizon=0;      %1=>integrator in RHC
LTV_horizon=0;      %1=>LTV linearization in the horizon
if deltaw_horizon
    LTV_horizon=0;  %u(1)=deltaw => only LTI linearization is allowed ???
end;
tf=t_breaks(end);
tf=floor(tf/T_samp)*T_samp;
if c_mode==2    %rc_controller
    tf=tf-(N_horizon+2)*T_samp;
else
    tf=tf-2*T_samp;
end;
%-------------------------------------
%
persistent c_F l_F c_R l_R m_v I_zz v_own xh
persistent lambda alpha0i alpha1i dataset
%
persistent t0_horizon xx_horizon uu_horizon ulast_horizon
%

if isempty(dataset)
    %
    c_F=100000;
    l_F=1.203;
    c_R=100000;
    l_R=1.217;
    m_v=1280;
    I_zz=2500;
    v_own=20; %m/s
    %
    lambda=10;
    alpha0i=lambda;
    alpha1i=2*sqrt(lambda);
    %
    xh=[0 0 0 v_own 0 0]';
    y=xh;
```

```
    %
    t0_horizon=NaN*ones(N_horizon+1,1);
    xx_horizon=NaN*ones(N_horizon+1,6);
    uu_horizon=NaN*ones(N_horizon,2);
    ulast_horizon=NaN*ones(2,1);
    %
    dataset=1;
    %
end;

u=NaN*ones(2,1);
xd=NaN*ones(6,1);

%--------------------------
% c_mode=1: dg_controller
%--------------------------
if c_mode==1 %dg_controller
    xh=xhat;
    x1=xh(1);    %beta
    x2=xh(2);    %psi
    x3=xh(3);    %d1psi
    x4=xh(4);    %vG
    x5=xh(5);    %X
    x6=xh(6);    %Y

    C12=cos(x1+x2);
    S12=sin(x1+x2);

    Sh=c_R*(-x1+l_R*x3/x4);
    T=0;

    [xt,d1xt,d2xt,d3xt,yt,d1yt,d2yt,d3yt]=...
        embed_path(t,t_breaks,x_coeffs,y_coeffs);

    xt=xt+1/lambda*(alpha1i*d1xt+d2xt);
    yt=yt+1/lambda*(alpha1i*d1yt+d2yt);

    y1bar=lambda*xt-alpha0i*x5-alpha1i*x4*C12;
    y2bar=lambda*yt-alpha0i*x6-alpha1i*x4*S12;
    u1=-Sh+m_v*((C12*x1-S12)*y1bar+(S12*x1+C12)*y2bar); %Sv
    u2=T+m_v*(C12*y1bar+S12*y2bar); %F_lR

    Sv=u1;
    F_lR=u2;
    deltaw=Sv/c_F+x1+l_F*x3/x4;
    %
    u=[deltaw; F_lR];
    xd=funxdvec(t,t_breaks,x_coeffs,y_coeffs);
    return
end

%--------------------------
%c_mode=2: rh_controller
%--------------------------
if c_mode==2 %rc_controller
    if t==0  %init horizon
        t0=0;
        ts=T_samp*N_horizon;
        x_0=xhat;
        xx_horizon(1,:)=x_0';
        for i=1:N_horizon
            ti=(i-1)*T_samp;
            xi=xx_horizon(i,:)';
            [xdot,F_lR,deltaw,Sv]=...
                dgi_controller(ti,xi,T_samp,t_breaks,x_coeffs,y_coeffs,...
                c_F,l_F,c_R,l_R,m_v,I_zz,...
                lambda,alpha0i,alpha1i);
            xip1=xi+T_samp*xdot;
```

```
        if ~deltaw_horizon
            ui=[Sv F_lR]';
        else
            ui=[deltaw F_lR]';
        end;
        uu_horizon(i,:)=ui';
        xx_horizon(i+1,:)=xip1';
        t0_horizon(i)=ti;
    end;
    ulast_horizon=uu_horizon(1,:)';
end;
t0=t;
for i=1:N_horizon+1, t0_horizon(i)=t0+(i-1)*T_samp; end;
tN=t0+N_horizon*T_samp;
xxd=funxdvec(t0_horizon,t_breaks,x_coeffs,y_coeffs);
C=[0 0 0 0 1 0; 0 0 0 0 0 1];
yyd=(C*xxd')';
yy=(C*xx_horizon')';
ee=yyd-yy;
EE=ee(2:N_horizon+1,:);
%
[AA,BB]=LTV2vehicle(xx_horizon,uu_horizon,T_samp,LTV_horizon,...
    c_F,l_F,c_R,l_R,m_v,I_zz,deltaw_horizon);
if int_horizon~=0
    [AAA,BBB,CCC]=preint_ab2ph(AA,BB,C);
else
    AAA=AA; BBB=BB; CCC=C;
end;
[P1,H1,P2,H2,mvec,eN]=ab2ph(AAA,BBB,CCC,EE,int_horizon);
delta_x00=xhat-xx_horizon(1,:)';
%
my=size(C,1);
ru=size(uu_horizon,2);
nx=size(xx_horizon,2);
nI=N_horizon*ru;
%
if int_horizon~=0
    delta_x0=[delta_x00; zeros(ru,1)];
else
    delta_x0=delta_x00;
end;
%
L1inv=pinv(H1'*H1+diag(repmat(lambda_horizon,1,N_horizon))*eye(nI));
Lmuinv=inv(H2*L1inv*H2'); %DS1005 C code error for pinv if dim=2
delta_U=L1inv*(H2'*Lmuinv*eN+(eye(nI)-H2'*Lmuinv*H2*L1inv)*mvec...
    -(H1'*P1+H2'*Lmuinv*(P2-H2*L1inv*H1'*P1))*delta_x0);
delta_uut=NaN*ones(ru,N_horizon);
delta_uut(:)=delta_U;
delta_uu=delta_uut';
%
if int_horizon~=0
    DELTAuu=[ulast_horizon'; delta_uu];
    DELTAuusum=cumsum(DELTAuu);
    delta_uu=DELTAuusum(2:N_horizon+1,:);
    uupred=delta_uu;
else
    uupred=uu_horizon+delta_uu;
end;
%
u0=uupred(1,:)';
ulastnew=u0;
%
if ~deltaw_horizon
    Sv0=u0(1);
    F_lR0=u0(2);
    deltaw0=Sv2deltaw(Sv0,xhat,c_F,l_F);
else
    deltaw0=u0(1);
```

```
        F_lR0=u0(2);
        Sv0=deltaw2Sv(deltaw0,xhat,c_F,l_F);
    end;
    %
    %Pepare next horizon

xxnext=uux02xx(uupred,xhat,t0,deltaw_horizon,T_samp,c_F,l_F,c_R,l_R,m_v,I_zz
);
    uunext=[uupred(2:N_horizon,:); NaN*ones(1,2)];
    xN=xxnext(N_horizon+1,:)';
    %
    if dgfresh_horizon==1   %uN by diffgeom
        [xdotN,F_lRN,deltawN,SvN]=dgi_controller(tN,xN,...
            T_samp,t_breaks,x_coeffs,y_coeffs,...
            c_F,l_F,c_R,l_R,m_v,I_zz,...
            lambda,alpha0i,alpha1i);
    elseif dgfresh_horizon==0   %uN makes xNp1 to zero
        Pmat=eye(6);
        tNp1=t0+(N_horizon+1)*T_samp;
        xdNp1=funxdvec(tNp1,t_breaks,x_coeffs,y_coeffs);
        [xdotNaN,F_lRNaN,deltawNaN,SvNaN,fN,GN]=apprfunxdot(tN,xN,0,0,...
            c_F,l_F,c_R,l_R,m_v,I_zz);
        fN=xN+T_samp*fN;
        GN=T_samp*GN;
        uN=-pinv(GN'*Pmat*GN)*GN'*Pmat*(fN-xdNp1);
        if ~deltaw_horizon
            SvN=uN(1);
            F_lRN=uN(2);
            deltawN=Sv2deltaw(SvN,xN,c_F,l_F);
        else
            deltawN=uN(1);
            F_lRN=uN(2);
            SvN=deltaw2Sv(deltawN,xN,c_F,l_F);
        end;
    else    %for example -1, in which case uN repeats uNm1
        uN=uupred(N_horizon,:)';
        if ~deltaw_horizon
            SvN=uN(1);
            F_lRN=uN(2);
            deltawN=Sv2deltaw(SvN,xN,c_F,l_F);
        else
            deltawN=uN(1);
            F_lRN=uN(2);
            SvN=deltaw2Sv(deltawN,xN,c_F,l_F);
        end;
    end;
    %
    if ~deltaw_horizon
        uN=[SvN F_lRN]';
    else
        uN=[deltawN F_lRN]';
    end;
    uunext(N_horizon,:)=uN';
    %
    tNp1=tN+T_samp;
    xdotNp1=apprfunxdot(tNp1,xN,F_lRN,deltawN,...
        c_F,l_F,c_R,l_R,m_v,I_zz);
    xNp1=xN+T_samp*xdotNp1;
    xxnext(N_horizon+1,:)=xNp1';
    %
    xx_horizon=[xxnext(2:N_horizon+1,:); xNp1'];
    uu_horizon=uunext;
    t0=t0+T_samp;
    ulast_horizon=ulastnew;
    %
    F_lR=F_lR0;
    deltaw=deltaw0;
    Sv=Sv0;
```

```
%     if ~deltaw_horizon
%         u=[Sv; F_lR];
%     else
%         u=[deltaw; F_lR];
%     end;
    %
    u=[deltaw; F_lR];
    xd=funxdvec(t,t_breaks,x_coeffs,y_coeffs);
    return
end

%-------------------------------------------------------
function [yy,d1yy,d2yy,d3yy]=embed_deriv3(xx,breaks,coeffs)
%Embedded deriv3()

nb=length(breaks)-1;

nx=length(xx);
yy=NaN*ones(nx,1);
d1yy=NaN*ones(nx,1);
d2yy=NaN*ones(nx,1);
d3yy=NaN*ones(nx,1);

for i=1:nx
    x=xx(i);
    %ix=max(find(breaks<=x));
    ix=embed_maxfind(breaks,x);
    if ix>nb, ix=nb; end;
    x0=breaks(ix);
    cc=coeffs(ix,:);
    dx=x-x0;
    y=cc(1)*dx^3+cc(2)*dx^2+cc(3)*dx+cc(4);
    d1y=3*cc(1)*dx^2+2*cc(2)*dx+cc(3);
    d2y=6*cc(1)*dx+2*cc(2);
    d3y=6*cc(1);
    yy(i)=y;
    d1yy(i)=d1y;
    d2yy(i)=d2y;
    d3yy(i)=d3y;
end;

%-------------------------------------------------------
function [xt,d1xt,d2xt,d3xt,yt,d1yt,d2yt,d3yt]=...
    embed_path(t,t_breaks,x_coeffs,y_coeffs)
%Compute reference signals and derivatives

[xt,d1xt,d2xt,d3xt]=embed_deriv3(t,t_breaks,x_coeffs);
[yt,d1yt,d2yt,d3yt]=embed_deriv3(t,t_breaks,y_coeffs);

%-------------------------------------------------------
function ix=embed_maxfind(x,x0)
%Embedded simple find

ix=max((x<=x0).*[1:length(x)]');

%-------------------------------------------------------
function
[xdot,F_lR,deltaw,Sv]=dgi_controller(ti,xi,T_samp,t_breaks,x_coeffs,y_coeffs
,...
    c_F,l_F,c_R,l_R,m_v,I_zz,...
    lambda,alpha0i,alpha1i)
%Implement dg_controller for control horizon initialization

x1=xi(1);    %beta
x2=xi(2);    %psi
x3=xi(3);    %d1psi
x4=xi(4);    %vG
x5=xi(5);    %X
```

```
x6=xi(6);    %Y

C12=cos(x1+x2);
S12=sin(x1+x2);

Sh=c_R*(-x1+l_R*x3/x4);
T=0;

[xt,d1xt,d2xt,d3xt,yt,d1yt,d2yt,d3yt]=...
    embed_path(ti,t_breaks,x_coeffs,y_coeffs);

%xd=[xt d1xt d2xt yt d1yt d2yt]';

xt=xt+1/lambda*(alpha1i*d1xt+d2xt);
yt=yt+1/lambda*(alpha1i*d1yt+d2yt);

y1bar=lambda*xt-alpha0i*x5-alpha1i*x4*C12;
y2bar=lambda*yt-alpha0i*x6-alpha1i*x4*S12;
u1=-Sh+m_v*((C12*x1-S12)*y1bar+(S12*x1+C12)*y2bar); %Sv
u2=T+m_v*(C12*y1bar+S12*y2bar); %F_lR

Sv=u1;
F_lR=u2;
deltaw=Sv/c_F+x1+l_F*x3/x4;

u=[deltaw; F_lR];

f=[-x3+1/(m_v*x4)*Sh+x1/(m_v*x4)*T; x3; -l_R/I_zz*Sh; -T/m_v; x4*C12;
x4*S12];
g1=[1/(m_v*x4); 0; l_F/I_zz; 0; 0; 0];
g2=[-x1/(m_v*x4); 0; 0; 1/m_v; 0; 0];

xdot=f+g1*Sv+g2*F_lR;

%-------------------------------------------------------
function
[AA,BB]=LTV2vehicle(xx,uu,Ts,LTVhor,c_F,l_F,c_R,l_R,m_v,I_zz,deltawinput)
%Linearize vehicle dynamic model along given trajectory and control
%Approximated model, u=(Sv,F_lR)' if deltawinput=0, otherwise
u=(deltaw,F_lR)'

%-----------------------
%Modified: 2007.06.20. (corrections if u(1)=deltaw)
%-----------------------
if nargin<11, deltawinput=0; end;

N=size(xx,1)-1;
n=size(xx,2);   %6
r=size(uu,2);   %2

% Ai=zeros(n,n);
% Bi=zeros(n,r);
% dfcdxi=zeros(n,n);
% dfcdui=zeros(n,r);
%
% AA=zeros(N*n,n);
% BB=zeros(N*n,r);
% for i=0:N-1
%     if LTVhor~=0
%         xi=xx(i+1,:)';
%         ui=uu(i+1,:)';
%
[dfcdxi,dfcdui]=dfapprdx(xi,ui,c_F,l_F,c_R,l_R,m_v,I_zz,deltawinput);
%     else
%         if i==0
%             xi=xx(1,:)';
%             ui=uu(1,:)';
```

```
%
[dfcdxi,dfcdui]=dfapprdx(xi,ui,c_F,l_F,c_R,l_R,m_v,I_zz,deltawinput);
%         end;
%     end;
%     Ai=eye(n)+Ts*dfcdxi;
%     Bi=Ts*dfcdui;
% %
[dfcdxi,dfcdui]=dfapprdx(xi,ui,c_F,l_F,c_R,l_R,m_v,I_zz,deltawinput);
% %     Ai=eye(n)+Ts*dfcdxi;
% %     Bi=Ts*dfcdui;
%     ra=i*n;
%     rb=i*n;
%     for ii=1:n
%         AA(ra+ii,:)=Ai(ii,:);
%     end;
%     for ii=1:n
%         BB(rb+ii,:)=Bi(ii,:);
%     end;
%     %AA(ra+1:ra+n,:)=Ai;
%     %BB(rb+1:rb+n,:)=Bi;
% end;
AA=zeros(N*n,n);
BB=zeros(N*n,r);
if LTVhor~=0
    for i=0:N-1
        xi=xx(i+1,:)';
        ui=uu(i+1,:)';

[dfcdxi,dfcdui]=dfapprdx(xi,ui,c_F,l_F,c_R,l_R,m_v,I_zz,deltawinput);
        Ai=eye(n)+Ts*dfcdxi;
        Bi=Ts*dfcdui;
        ra=i*n;
        rb=i*n;
        for ii=1:n
            AA(ra+ii,:)=Ai(ii,:);
            BB(rb+ii,:)=Bi(ii,:);
        end;
        %AA(ra+1:ra+n,:)=Ai;
        %BB(rb+1:rb+n,:)=Bi;
    end;
else
    x0=xx(1,:)';
    u0=uu(1,:)';
    [dfcdx0,dfcdu0]=dfapprdx(x0,u0,c_F,l_F,c_R,l_R,m_v,I_zz,deltawinput);
    A0=eye(n)+Ts*dfcdx0;
    B0=Ts*dfcdu0;
    for i=0:N-1
        ra=i*n;
        rb=i*n;
        for ii=1:n
            AA(ra+ii,:)=A0(ii,:);
            BB(rb+ii,:)=B0(ii,:);
        end;
        %AA(ra+1:ra+n,:)=Ai;
        %BB(rb+1:rb+n,:)=Bi;
    end;
end

%-------------------------------------------------------
function [AAA,BBB,CCC]=preint_ab2ph(AA,BB,C)
%Compute AAA,BBB,CCC for integral control

n=size(AA,2);
r=size(BB,2);
N=size(AA,1)/n;
m=size(C,1);

AAA=zeros(N*(n+r),n+r);
```

```
BBB=zeros(N*(n+r),r);
CCC=[C zeros(m,r)];

Ai=NaN*ones(n,n);
Bi=NaN*ones(n,r);

for i=0:N-1
    rai=i*n;
    rbi=i*n;
    raai=i*(n+r);
    rbbi=i*(n+r);
    for ii=1:n
        Ai(ii,:)=AA(rai+ii,:);
        Bi(ii,:)=BB(rbi+ii,:);
    end;
    %Ai=AA(rai+1:rai+n,:);
    %Bi=BB(rbi+1:rbi+n,:);
    AAi=[Ai Bi; zeros(r,n) eye(r)];
    BBi=[Bi; eye(r)];
    for ii=1:n+r
        AAA(raai+ii,:)=AAi(ii,:);
        BBB(rbbi+ii,:)=BBi(ii,:);
    end;
    %AAA(raai+1:raai+n+r,:)=AAi;
    %BBB(rbbi+1:rbbi+n+r,:)=BBi;
end

%-------------------------------------------------------
function [P1,H1,P2,H2,mvec,eN]=ab2ph(AA,BB,C,ee,inthor)
%Compute PP and HH from AA and BB for use in LTV RHC control

% %TEST1
% C=[0 1 0; 0 0 1]
% pause
% e1=[41 42]'
% e2=[43 44]'
% e3=[45 46]'
% e4=[47 48]'
% ee=[e1'; e2'; e3'; e4']
% pause
% A0=[1 2 3; 4 5 6; 7 8 9]
% A1=[11 12 13; 14 15 16; 17 18 19]
% A2=[21 22 23; 24 25 26; 27 28 29]
% A3=[31 32 33; 34 35 36; 37 38 39]
% pause
% B0=[-1 -2; -3 -4; -5 -6]
% B1=[-11 -12; -13 -14; -15 -16]
% B2=[-21 -22; -23 -24; -25 -26]
% B3=[-31 -32; -33 -34; -35 -36]
% pause
% AA=[A0; A1; A2; A3]
% pause
% BB=[B0; B1; B2; B3]
% pause
% PPx=[A0; A1*A0; A2*A1*A0; A3*A2*A1*A0]
% pause
% I=eye(3); Z=zeros(3,3);
% HH0x=[I Z Z Z; A1 I Z Z; A2*A1 A2 I Z; A3*A2*A1 A3*A2 A3 I]
% pause
% ZZ=zeros(3,2);
% HHx=[B0 ZZ ZZ ZZ; A1*B0 B1 ZZ ZZ; A2*A1*B0 A2*B1 B2 ZZ; A3*A2*A1*B0
A3*A2*B1 A3*B2 B3]
% pause
% CPPx=[C*A0; C*A1*A0; C*A2*A1*A0; C*A3*A2*A1*A0]
% pause
% ZZZ=zeros(2,2);
% CHHx=[C*B0 ZZZ ZZZ ZZZ; C*A1*B0 C*B1 ZZZ ZZZ; C*A2*A1*B0 C*A2*B1 C*B2
ZZZ;...
```

```
%         C*A3*A2*A1*B0 C*A3*A2*B1 C*A3*B2 C*B3]
% pause
% h1t=[C*B0 ZZZ ZZZ ZZZ]
% h2t=[C*A1*B0 C*B1 ZZZ ZZZ]
% h3t=[C*A2*A1*B0 C*A2*B1 C*B2 ZZZ]
% pause
% mvecx=h1t'*e1+h2t'*e2+h3t'*e3
% pause
% eNx=e4
% pause
% %TEST1

n=size(AA,2);
r=size(BB,2);
N=size(AA,1)/n;
m=size(C,1);

PP=zeros(N*n,n);
PP(1:n,:)=AA(1:n,:);

Ai=zeros(n,n);
PPj=zeros(n,n);

for i=1:N-1
    rai=i*n;
    rp0i=(i-1)*n;
    for ii=1:n
        Ai(ii,:)=AA(rai+ii,:);
        for jj=1:n
            PPj(jj,:)=PP(rp0i+jj,:);
        end;
        PP(rp0i+n+ii,:)=Ai(ii,:)*PPj;
    end;
end;
%------------------------
% PP=zeros(N*n,n);
% PP(1:n,:)=AA(1:n,:);
% for i=1:N-1
%     rai=i*n;
%     rp0i=(i-1)*n;
%     Ai=AA(rai+1:rai+n,:);
%     PP(rp0i+n+1:rp0i+n+n,:)=Ai*PP(rp0i+1:rp0i+n,:);
% end;
%------------------------

Ai=zeros(n,n);

HH0=zeros(N*n,N*n);
HH0(1:n,1:n)=eye(n);
for i=1:N-1
    rai=i*n;
    rh0i=(i-1)*n;
    for ii=1:n
        Ai(ii,:)=AA(rai+ii,:);
    end;
    %Ai=AA(rai+1:rai+n,:);
    Hi=zeros(n,(N-1)*n);
    for jj=1:n
        for kk=1:i*n
            Hi(jj,kk)=HH0(rh0i+jj,kk);
        end;
    end;
    %Hi=HH0(rh0i+1:rh0i+n,1:i*n);
    for jj=1:n
        for kk=1:i*n
            ss=0;
            for ll=1:n
                ss=ss+Ai(jj,ll)*Hi(ll,kk);
```

```
            end;
            HH0(rh0i+n+jj,kk)=ss;
        end;
    end;
    for ii=1:n
        HH0(rh0i+n+ii,i*n+ii)=1;
    end;
end;
%------------------------
% HH0=zeros(N*n,N*n);
% HH0(1:n,1:n)=eye(n);
% for i=1:N-1
%     rai=i*n;
%     rh0i=(i-1)*n;
%     Ai=AA(rai+1:rai+n,:);
%     Hi=HH0(rh0i+1:rh0i+n,1:i*n);
%     HH0(rh0i+n+1:rh0i+n+n,1:(i+1)*n)=[Ai*Hi eye(n)];
% end;
%------------------------

Bi=zeros(n,r);
HHi=zeros(N*n,n);

HH=zeros(N*n,N*r);
for i=0:N-1
    rbi=i*n;
    ch0i=i*n;
    chi=i*r;
    for ii=1:n
        Bi(ii,:)=BB(rbi+ii,:);
        HHi(:,ii)=HH0(:,ch0i+ii);
    end;
    HHBi=HHi*Bi;
    for ii=1:r
        HH(:,chi+ii)=HHBi(:,ii);
    end;
end;
%------------------------
% HH=zeros(N*n,N*r);
% for i=0:N-1
%     rbi=i*n;
%     ch0i=i*n;
%     chi=i*r;
%     Bi=BB(rbi+1:rbi+n,:);
%     Hi=HH0(:,ch0i+1:ch0i+n);
%     HH(:,chi+1:chi+r)=Hi*Bi;
% end;
%------------------------

Pi=zeros(n,n);
HHHi=zeros(n,N*r);

CPP=zeros(N*m,n);
CHH=zeros(N*m,N*r);
for i=0:N-1
    rpi=i*n;
    rhi=i*n;
    rcpi=i*m;
    rchi=i*m;
    for ii=1:n
        Pi(ii,:)=PP(rpi+ii,:);
    end;
    CPi=C*Pi;
    for ii=1:m
        CPP(rcpi+ii,:)=CPi(ii,:);
    end;
    for ii=1:n
        HHHi(ii,:)=HH(rhi+ii,:);
```

```
    end;
    CHHHi=C*HHHi;
    for ii=1:m
        CHH(rchi+ii,:)=CHHHi(ii,:);
    end;
end;
%------------------------
% CPP=zeros(N*m,n);
% CHH=zeros(N*m,N*r);
% for i=0:N-1
%     rpi=i*n;
%     rhi=i*n;
%     rcpi=i*m;
%     rchi=i*m;
%     Pi=PP(rpi+1:rpi+n,:);
%     CPP(rcpi+1:rcpi+m,:)=C*Pi;
%     Hi=HH(rhi+1:rhi+n,:);
%     CHH(rchi+1:rchi+m,:)=C*Hi;
% end;
%------------------------

hit=zeros(m,N*r);

P1=CPP(1:(N-1)*m,:);
H1=CHH(1:(N-1)*m,:);
P2=CPP((N-1)*m+1:(N-1)*m+m,:);
H2=CHH((N-1)*m+1:(N-1)*m+m,:);

mvec=zeros(N*r,1);
for i=0:N-2
    rhi=i*r;
    for ii=1:m
        hit(ii,:)=CHH(rhi+ii,:);
    end;
    ei=ee(i+1,:)';
    mvec=mvec+hit'*ei;
end;
eN=ee(N,:)';
%------------------------
% P1=CPP(1:(N-1)*m,:);
% H1=CHH(1:(N-1)*m,:);
% P2=CPP((N-1)*m+1:(N-1)*m+m,:);
% H2=CHH((N-1)*m+1:(N-1)*m+m,:);
%
% mvec=zeros(N*r,1);
% for i=0:N-2
%     rhi=i*r;
%     hit=CHH(rhi+1:rhi+m,:);
%     ei=ee(i+1,:)';
%     mvec=mvec+hit'*ei;
% end;
% eN=ee(N,:)';
%------------------------

% %TEST2
% dPP=PP-PPx
% pause
% dHH0=HH0-HH0x
% pause
% dHH=HH-HHx
% pause
% dCPP=CPP-CPPx
% pause
% dCHH=CHH-CHHx
% pause
% dmvec=mvec-mvecx
% pause
% deN=eN-eNx
```

```
% pause
% %END TEST2

% %TEST3
% PP
% pause
% HH0
% pause
% HH
% pause
% CPP
% pause
% CHH
% pause
% mvec
% pause
% eN
% pause
% %END TEST3

%-------------------------------------------------------
function [dfdx,dfdu]=dfapprdx(x,u,c_F,l_F,c_R,l_R,m_v,I_zz,deltawinput)
%First derivative of f(x,u) (approximated)
%-----------------------
%Modified: 2007.06.20. (corrections if u(1)=deltaw)
%-----------------------

if nargin<9, deltawinput=0; end;

x1=x(1);    %beta
x2=x(2);    %psi
x3=x(3);    %d1psi
x4=x(4);    %vG
x5=x(5);    %X
x6=x(6);    %Y
C12=cos(x1+x2);
S12=sin(x1+x2);

if ~deltawinput
    Sv=u(1);
    F_lR=u(2);
    deltaw=Sv2deltaw(Sv,x,c_F,l_F);
else
    deltaw=u(1);
    F_lR=u(2);
    Sv=deltaw2Sv(deltaw,x,c_F,l_F);
end;

df1dx1=1/(m_v*x4)*(-F_lR-c_R);
df1dx3=-1+1/(m_v*x4)*(c_R*l_R/x4);
df1dx4=-1/(m_v*x4^2)*(-F_lR*x1+Sv+c_R*(-x1+l_R*x3/x4))+1/(m_v*x4)*(-
c_R*l_R*x3/(x4^2));
df1dx=[df1dx1 0 df1dx3 df1dx4 0 0];

df2dx=[0 0 1 0 0 0];

df3dx1=1/I_zz*(l_R*c_R);
df3dx3=1/I_zz*(-l_R^2*c_R/x4);
df3dx4=1/I_zz*(l_R^2*c_R*x3/(x4^2));
df3dx=[df3dx1 0 df3dx3 df3dx4 0 0];

df4dx=[0 0 0 0 0 0];

df5dx1=-x4*S12;
df5dx2=-x4*S12;
df5dx4=C12;
df5dx=[df5dx1 df5dx2 0 df5dx4 0 0];
```

```
df6dx1=x4*C12;
df6dx2=x4*C12;
df6dx4=S12;
df6dx=[df6dx1 df6dx2 0 df6dx4 0 0];

df1du1=1/(m_v*x4);
df1du2=1/(m_v*x4)*(-x1);
df1du=[df1du1 df1du2];

df2du=[0 0];

df3du1=1/I_zz*(l_F);
df3du=[df3du1 0];

df4du2=1/m_v;
df4du=[0 df4du2];

df5du=[0 0];

df6du=[0 0];

dfdx=[df1dx; df2dx; df3dx; df4dx; df5dx; df6dx];
dfdu=[df1du; df2du; df3du; df4du; df5du; df6du];

if deltawinput
    b1=dfdu(:,1);
    dv1dx=[-c_F 0 -c_F*l_F/x4 c_F*l_F*x3/(x4^2) 0 0];
    dfdu=[b1*c_F dfdu(:,2)];
    dfdx=dfdx+b1*dv1dx;
end;

%-------------------------------------------------------
function xdvect=funxdvec(t,t_breaks,x_coeffs,y_coeffs)
%Compute desired state from CAS path database

nt=length(t);

[xt,d1xt,d2xt,d3xt,yt,d1yt,d2yt,d3yt]=...
    embed_path(t,t_breaks,x_coeffs,y_coeffs);

psit=atan2(d1yt,d1xt);
vGt=sqrt(d1xt.^2+d1yt.^2);
d1psit=(d2yt.*d1xt-d1yt.*d2xt)./(vGt.^2);

if nt==1
    xdvect=[zeros(nt,1) psit d1psit vGt xt yt]';
else
    xdvect=[zeros(nt,1) psit d1psit vGt xt yt];
end;

%-------------------------------------------------------
function Sv=deltaw2Sv(deltaw,x,c_F,l_F);
%Convert deltaw to Sv using approximated modell

col=size(x,2);
if col==1, xx=x'; else xx=x; end;
x1=xx(:,1);    %beta
x2=xx(:,2);    %psi
x3=xx(:,3);    %d1psi
x4=xx(:,4);    %vG
x5=xx(:,5);    %X
x6=xx(:,6);    %Y
C12=cos(x1+x2);
S12=sin(x1+x2);

Sv=c_F*(deltaw-x1-l_F*x3./x4);

%-------------------------------------------------------
```

```
function deltaw=Sv2deltaw(Sv,x,c_F,l_F);
%Convert Sv to deltaw using approximated modell

col=size(x,2);
if col==1, xx=x'; else xx=x; end;
x1=xx(:,1);    %beta
x2=xx(:,2);    %psi
x3=xx(:,3);    %d1psi
x4=xx(:,4);    %vG
x5=xx(:,5);    %X
x6=xx(:,6);    %Y
C12=cos(x1+x2);
S12=sin(x1+x2);

deltaw=Sv/c_F+x1+l_F*x3./x4;

%-------------------------------------------------------
function xx=uux02xx(uu,x0,t0,deltawinput,Tsamp,c_F,l_F,c_R,l_R,m_v,I_zz)
%Compute xx tansient belonging to uu and x0
%-------------------------------------------------------
%Modified: 2007.06.20. (corrections if u(1)=deltaw)
%-------------------------------------------------------

nx=size(x0,1);
xip1=zeros(nx,1);

N=size(uu,1);
xi=x0;
xx=NaN*ones(N+1,6);
for i=0:1:N-1
    ti=t0+i*Tsamp;
    ui=uu(i+1,:)';
    if ~deltawinput
        Svi=ui(1);
        F_lRi=ui(2);
        deltawi=Sv2deltaw(Svi,xi,c_F,l_F);
    else
        deltawi=ui(1);
        F_lRi=ui(2);
        Svi=deltaw2Sv(deltawi,xi,c_F,l_F);
    end;
    xdoti=apprfunxdot(ti,xi,F_lRi,deltawi,c_F,l_F,c_R,l_R,m_v,I_zz);
    xip1=xi+Tsamp*xdoti;
    xx(i+1,:)=xi';
    xi=xip1;
end;
xx(N+1,:)=xip1';

%-------------------------------------------------------
function [xdot,F_lR,deltaw,Sv,f,G]=apprfunxdot(t,x,F_lR,deltaw,...
    c_F,l_F,c_R,l_R,m_v,I_zz)
%Compute right side of approximated vehicle dynamic model

x1=x(1);    %beta
x2=x(2);    %psi
x3=x(3);    %d1psi
x4=x(4);    %vG
x5=x(5);    %X
x6=x(6);    %Y
C12=cos(x1+x2);
S12=sin(x1+x2);

T=0;
Sh=c_R*(-x1+l_R*x3/x4);
Sv=c_F*(deltaw-x1-l_F*x3/x4);

f=[-x3+1/(m_v*x4)*Sh+x1/(m_v*x4)*T; x3; -l_R/I_zz*Sh; -T/m_v; x4*C12;
x4*S12];
```

```
g1=[1/(m_v*x4); 0; l_F/I_zz; 0; 0; 0];
g2=[-x1/(m_v*x4); 0; 0; 1/m_v; 0; 0];
G=[g1 g2];

xdot=f+g1*Sv+g2*F_lR;

%-------------------------------------------------------
```

6.2 Az állapotbecslőt megvalósító beágyazott függvény

```
function xhat = extKF2(ym,x,T_samp,x_0,t)
% Two level extended Kalman filter based on GPS/INS
% This block supports an embeddable subset of the MATLAB language.
% See the help menu for details.

%----------------------------
%Gyro sensitivity switch
gyrosens_corr=0;
%System estimator switch
sys_estim=1;
%----------------------------

persistent dataset
%GPS/INS
persistent sigma1_GPSvelocity sigma1_GPSattitude
persistent sigma1_acc bias_acc sigma1_rategyro bias_rate
%persistent xsys_prev
%KF
persistent TsINS TsGPSvelocity TsGPSattitude
persistent cov1p cov1m cov2p cov2m dimx1p irbias
persistent xhat_KF xp_KF xm_KF x1p_KF x1m_KF x2p_KF x2m_KF P1p_KF P1m_KF
P2p_KF P2m_KF
persistent xhat_KF_prev

if isempty(dataset)
    %----------------------------
    %init random number generator
    %randn_own('state',0);
    %----------------------------
    %init GPS/INS
    %
    TsINS=0.01;             %100 Hz (assumed to be equal Tsamp)
    TsGPSvelocity=0.1;      %10 Hz
    TsGPSattitude=0.2;      %5 Hz
    %
    sigma1_GPSvelocity=3*0.01;      %m/s
    sigma1_GPSattitude=0.2*pi/180;  %rad
    sigma1_acc=0.05;                %m/s^2
    sigma1_rategyro=0.2*pi/180;     %rad/s
    %
    bias_acc=0.05;                  %m/s^2
    bias_rate=0.05;                 %rad/sec
    %
    sigma1_ratebias=0.05;           %rad/sec
    sigma1_accbias=0.05;            %m/s^2
    sigma1_rateinvsens=0.05;        %-
    sigma1_ratebiasdivsens=0.05;    %rad/sec
    %
    %xsys_prev=x_0;
    %----------------------------
    %init KF2
    %
    %Covariance matrices for process and measurement noises
    if ~gyrosens_corr
        cov1p=0.01*diag([sigma1_rategyro^2 sigma1_ratebias^2]);
        dimx1p=2;
        irbias=2;
        P1p_KF=1e-6*eye(2); %P1p_KF=1e-6*eye(dimx1p);
        x1p_KF=[x_0(2) ones(1,0) 0]';
    else
        cov1p=0.01*diag([sigma1_rategyro^2 sigma1_rateinvsens^2
sigma1_ratebiasdivsens^2]);
        dimx1p=3;
        irbias=3;
        P1p_KF=1e-6*eye(3); %P1p_KF=1e-6*eye(dimx1p);
```

```
        x1p_KF=[x_0(2) ones(1,1) 0]';
    end;
    cov1m=sigma1_GPSattitude^2;
    cov2p=0.1*diag([sigma1_acc^2 100*sigma1_accbias^2 sigma1_acc^2
100*sigma1_accbias^2]);
    cov2m=diag([sigma1_GPSvelocity sigma1_GPSvelocity]);
    %Initiate error covariance matrices
    P1m_KF=P1p_KF;
    P2p_KF=1e-6*eye(4);
    P2m_KF=P2p_KF;

    %Reserve memory for states
    xhat_KF=x_0;
    xp_KF=xhat_KF;
    xm_KF=xp_KF;
    x1m_KF=x1p_KF;
    x2p_KF=[x_0(4) 0 0 0]';
    x2m_KF=x2p_KF;
    %
    %xsys_prev=x_0;
    xhat_KF_prev=xhat_KF;
    %
    dataset=1;
end;

if ~sys_estim
    xhat=x;
    return;
end;

%-------------------------------------------------------------
%Simulate GPS/INS near to system state
%[rm,psimGPS,VmGPS,axm,aym,Xm,Ym]=sim_GPS_INS(x,xsys_prev,Ts);
rm=ym(1);
psimGPS=ym(2);
VmGPS=ym(3:5);
axm=ym(6);
aym=ym(7);
Xm=ym(8);
Ym=ym(9);
%-----------------------------------------------------
if rem(t,TsGPSattitude)~=0, psimGPS=psimGPS*NaN; end;
if rem(t,TsGPSvelocity)~=0, VmGPS=VmGPS*NaN; end;
%-----------------------------------------------------
%Compute estimated state (global)
%[betaGPS,uxmGPS,uymGPS]=xestim(rm,psimGPS,VmGPS,axm,aym);
%function [betaGPS,uxmGPS,uymGPS]=xestim(rm,psimGPS,VmGPS,axm,aym);
%Compute vehicle state estimation
%Modified: 2007.03.16.
%New: psimGPS and/or VmGPS are NaNs if their sampling time is not reached!
%Otherwise:
%x1=[psi r_bias]';
%x2=[ux ax_bias uy ay_bias]';
%u1m=rm;
%y1m=psimGPS;
%u2m=[axm aym]
%y2m=[uxmGPS uymGPS]';

eps_own=1e-12;
Ts=T_samp;

if ~gyrosens_corr
    %State: [psi rbias]'
    Ad1=[1 -Ts; 0 1];
    Bd1=[Ts; 0];
    Cd1=[1 0];
    R1=cov1m;
    Q1=cov1p;
```

```
    [x1p_KF,x1m_KF,P1p_KF,P1m_KF,R1,Q1]=...
        KalmanFilter(x1p_KF,x1m_KF,P1p_KF,P1m_KF,R1,Q1,Ad1,Bd1,Cd1,...
        rm,psimGPS);
    psi=x1p_KF(1);
    r=rm-x1p_KF(2);
else
    %State: [psi 1/sr rbias/sr]'
    A=[0 rm -1; 0 0 0; 0 0 0];
    Ad1=eye(3)+A*Ts;
    Bd1=zeros(3,1);
    Cd1=[1 0 0];
    Bd1p=eye(3)*Ts+A*Ts^2/2;
    R1=cov1m;
    Q1=cov1p;
    [x1p_KF,x1m_KF,P1p_KF,P1m_KF,R1,Q1]=...
        KalmanFilter(x1p_KF,x1m_KF,P1p_KF,P1m_KF,R1,Q1,Ad1,Bd1,Cd1,...
        rm,psimGPS);
    psi=x1p_KF(1);
    r=rm*x1p_KF(2)-x1p_KF(3);
end;
%betaGPS=psimGPS-psi;
%-----------------------------------------
if  ~isnan(VmGPS(1))
    betaGPS=atan2(VmGPS(2),VmGPS(1))-psi;
    uxmGPS=norm(VmGPS,2)*cos(betaGPS);
    uymGPS=norm(VmGPS,2)*sin(betaGPS);
else
    betaGPS=NaN;
    uxmGPS=NaN;
    uymGPS=NaN;
end;
%-----------------------------------------
Srt=sin(r*Ts);
Crt=cos(r*Ts);
if abs(r)<=eps_own
    Srtdivr=Ts;
    Crtm1divr=0;
else
    Srtdivr=Srt/r;
    Crtm1divr=(Crt-1)/r;
end;
Ad2=[Crt -Srtdivr Srt Crtm1divr;...
        0 1 0 0;...
        -Srt -Crtm1divr Crt Srtdivr;...
        0 0 0 1];
Bd2=[Srtdivr 0 Crtm1divr 0;...
        -Crtm1divr 0 Srtdivr 0]';
Cd2=[1 0 0 0; 0 0 1 0];
R2=cov2m;
Q2=cov2p;

[x2p_KF,x2m_KF,P2p_KF,P2m_KF,R2,Q2]=...
    KalmanFilter(x2p_KF,x2m_KF,P2p_KF,P2m_KF,R2,Q2,Ad2,Bd2,Cd2,...
    [axm aym]',[uxmGPS uymGPS]');
ux=x2p_KF(1);
uy=x2p_KF(3);
vG=sqrt(ux^2+uy^2);
%vG=sqrt(uxmGPS^2+uymGPS^2);
beta=atan2(uy,ux);
%Integration
X=xp_KF(5);
Y=xp_KF(6);
X=X+Ts*vG*cos(psi+beta);
Y=Y+Ts*vG*sin(psi+beta);
%
xp_KF=x12tox(Ts,xp_KF,x1p_KF,x2p_KF,rm,gyrosens_corr);
xm_KF=x12tox(Ts,xm_KF,x1m_KF,x2m_KF,rm,gyrosens_corr);
xhat_KF_prev=xhat_KF;
```

```
xhat_KF=xp_KF;
%
xhat=xhat_KF_prev;

%-----------------------------------------------------------------
function [xp,xm,Pp,Pm,R,Q]=KalmanFilter(xp,xm,Pp,Pm,R,Q,Ad,Bd,Cd,um,ym)
%Compute xhat by Kalman Filter

%Modified: 2007.03.16.

n=length(xm);

if ~isnan(ym(1))
    %Measurement update
    K=Pm*Cd'*inv(Cd*Pm*Cd'+R); %DS1005 C code error for pinv if dim=2
    xp=xm+K*(ym-Cd*xm);
    Pp=(eye(n)-K*Cd)*Pm;
    %Time update
    xm=Ad*xp+Bd*um;
    Pm=Ad*Pp*Ad'+Q;
else
    %Measurement update
    xp=xm;
    Pp=Pm;
    %Time update
    xm=Ad*xp+Bd*um;
    Pm=Ad*Pp*Ad'+Q;
end;

%-----------------------------------------------------------------
function x=x12tox(Ts,x,x1,x2,rm,gyrosens_corr)
%Convert x1 and x2 to x

ux=x2(1);
uy=x2(3);
beta=atan2(uy,ux);
psi=x1(1);
if ~gyrosens_corr
    d1psi=rm-x1(2);
else
    d1psi=rm*x1(2)-x1(3);
end;
vG=sqrt(ux^2+uy^2);
d1X=Ts*vG*cos(psi+beta);
d1Y=Ts*vG*sin(psi+beta);
X=x(5)+d1X;
Y=x(6)+d1Y;
x=[beta psi d1psi vG X Y]';

%-----------------------------------------------------------------
```

6.3 A nemlineáris gépjárművet emuláló beágyazott függvény

```
function xdot = cardot(x,u,t,x_0,u_0)
% Embedded function for computing the right side of the car's state equation
% This block supports an embeddable subset of the MATLAB language.
% See the help menu for details.

persistent c_F l_F c_R l_R m_v I_zz v_own u1_is_deltaw dataset

if isempty(dataset)
    c_F=100000;
    l_F=1.203;
    c_R=100000;
    l_R=1.217;
    m_v=1280;
    I_zz=2500;
    v_own=20; %m/s
    u1_is_deltaw=1; %0=>u1=Sv; 1=>u1=deltaw
    dataset=1;
end;

if t==0, x=x_0; u=u_0; end;

x1=x(1);    %beta
x2=x(2);    %psi
x3=x(3);    %d1psi
x4=x(4);    %vG
x5=x(5);    %X
x6=x(6);    %Y

if ~u1_is_deltaw
    Sv=u(1);
    deltaw=Sv/c_F+x1+l_F*x3/x4;
else
    deltaw=u(1);
    Sv=c_F*(deltaw-x1-l_F*x3/x4);
end;
F_lR=u(2);

C12=cos(x1+x2);
S12=sin(x1+x2);

beta=x1;
Sdb=sin(deltaw-beta);
Cdb=cos(deltaw-beta);
Sb=sin(beta);
Cb=cos(beta);
Sd=sin(deltaw);
Cd=cos(deltaw);

F_lF=0;
T=0;
Sh=c_R*(-x1+l_R*x3/x4);

f=[-x3+1/(m_v*x4)*(F_lF*Sdb-(F_lR-T)*Sb+Sv*Cdb+Sh*Cb);...
        x3;...
        1/I_zz*(l_F*F_lF*Sd+l_F*Sv*Cd-l_R*Sh);...
        1/m_v*(F_lF*Cdb+(F_lR-T)*Cb-Sv*Sdb+Sh*Sb);...
        x4*C12;...
        x4*S12];

xdot=f;
```

6.4 A GPS/INS érzékelőket emuláló beágyazott függvény

```
function ym = GPS_INS(x,T_samp,x_0)
% Emulate GPS and INS sensors
% This block supports an embeddable subset of the MATLAB language.
% See the help menu for details.

persistent dataset
persistent sigma1_GPSvelocity sigma1_GPSattitude
persistent sigma1_acc bias_acc sigma1_rategyro bias_rate
persistent xsys_prev

%-------------------------------------------------------------
if isempty(dataset)
    %---------------------------
    %init random number generator
    %randn_own('state',0); %See rand_own1
    %---------------------------
    %init GPS/INS
    %
    TsINS=0.01;             %100 Hz (assumed to be equal T_samp)
    TsGPSvelocity=0.1;      %10 Hz
    TsGPSattitude=0.2;      %5 Hz
    %
    sigma1_GPSvelocity=3*0.01;      %m/s
    sigma1_GPSattitude=0.2*pi/180;  %rad
    sigma1_acc=0.05;                %m/s^2
    sigma1_rategyro=0.2*pi/180;     %rad/s
    %
    bias_acc=0.05;                  %m/s^2
    bias_rate=0.05;                 %rad/sec
    %
    sigma1_ratebias=0.05;           %rad/sec
    sigma1_accbias=0.05;            %m/s^2
    sigma1_rateinvsens=0.05;        %-
    sigma1_ratebiasdivsens=0.05;    %rad/sec
    %
    xsys_prev=x_0;
    %
    dataset=1;
end;

xprev=xsys_prev;
Ts=T_samp;

%Data without noise
beta=x(1);
psi=x(2);
r=x(3);
vG=x(4);
X=x(5);
Y=x(6);
%Generate noiseless sensor data
psiGPS=psi;
gammaGPS=psi+beta;
ux=vG*cos(beta);
uy=vG*sin(beta);
uxGPS=ux;
uyGPS=uy;
V1GPS=vG*cos(gammaGPS);
V2GPS=vG*sin(gammaGPS);
VGPS=[V1GPS V2GPS 0]';
%--------------
betaprev=xprev(1);
vGprev=xprev(4);
uxprev=vGprev*cos(betaprev);
uyprev=vGprev*sin(betaprev);
```

```
d1ux=(ux-uxprev)/Ts;
d1uy=(uy-uyprev)/Ts;
ax=d1ux-r*uy;
ay=d1uy+r*ux;
%--------------
%Generate noisy sensor data
uxmGPS=uxGPS+randn_own1*sigma1_GPSvelocity/2;
uymGPS=uyGPS+randn_own1*sigma1_GPSvelocity/2;
psimGPS=psiGPS+randn_own1*sigma1_GPSattitude;
%--------------
axm=ax+randn_own1*sigma1_acc/2+bias_acc;
aym=ay+randn_own1*sigma1_acc/2+bias_acc;
%--------------
rm=r+randn_own1*sigma1_rategyro+bias_rate;
V1mGPS=V1GPS+randn_own1*sigma1_GPSvelocity/2;
V2mGPS=V2GPS+randn_own1*sigma1_GPSvelocity/2;
VmGPS=[V1mGPS V2mGPS 0]';
%--------------
Xm=X;
Ym=Y;
%
ym=[rm,psimGPS,VmGPS',axm,aym,Xm,Ym]';
xsys_prev=x;

%-------------------------------------------
function ud1=rand_own1
% Uniform distribution by Park and Miller

persistent rand_xk rand_a rand_c rand_m
if isempty(rand_xk)
    rand_a=7^5;
    rand_c=0;
    rand_m=2^31-1;
    rand_xk=1;
end;

rand_xk=rem(rand_a*rand_xk+rand_c,rand_m);
ud1=rand_xk/rand_m;

%-------------------------------------------
function nd1=randn_own1
% Normal distribution by polar algorithm

u=zeros(2,1);
r=inf;
while r>1
    u1=2*rand_own1-1;
    u2=2*rand_own1-1;
    u=2*[u1 u2]'-1;
    r=u'*u;
end;
nd1=sqrt(-2*log(r)/r)*u(1);

%-------------------------------------------
```

6.5 Az AutoBoxra fordítás helyességét dokumentáló protokoll

```
------------------------------------------------------------------------
Starting build procedure with RTI 5.3.6 (RTI1005, 29-May-2006)
Model: "closedsys" (F:\users\LB\AutoboxCAS3\closedsys.mdl)
------------------------------------------------------------------------
*** Using configuration set : "Configuration"
*** Working directory       : "F:\users\LB\AutoboxCAS3"
Warning: The model 'closedsys' does not have continuous states, hence using the solver 'FixedStepDiscrete' instead
of solver 'ode1'. You can disable this diagnostic by explicitly specifying a discrete solver in the solver tab of the
```

Configuration Parameters dialog, or setting 'Automatic solver parameter selection' diagnostic to 'none' in the Diagnostics tab of the Configuration Parameters dialog.
*** Optional User System Description File closedsys_usr.sdf not available
*** Initializing code generation
Starting Real-Time Workshop build procedure for model: closedsys
Generating code into build directory: F:\users\LB\AutoboxCAS3\closedsys_rti1005
Embedded MATLAB parsing for model "closedsys"...Done
Embedded MATLAB code generation for model "closedsys"....Done
Invoking Target Language Compiler on closedsys.rtw

```
tlc
-r
F:\users\LB\AutoboxCAS3\closedsys_rti1005\closedsys.rtw
f:\dspace\matlab\rti1005\tlc\rti1005.tlc
-OF:\users\LB\AutoboxCAS3\closedsys_rti1005
-If:\dspace\matlab\rti1005\tlc
-IF:\users\LB\AutoboxCAS3\closedsys_rti1005\tlc
-IC:\Matlab\R2006a\rtw\c\tlc\mw
-IC:\Matlab\R2006a\rtw\c\tlc\lib
-IC:\Matlab\R2006a\rtw\c\tlc\blocks
-IC:\Matlab\R2006a\rtw\c\tlc\fixpt
-IC:\Matlab\R2006a\stateflow\c\tlc
-aEnforceIntegerDowncast=1
-aFoldNonRolledExpr=1
-aInlineInvariantSignals=0
-aInlineParameters=0
-aLocalBlockOutputs=0
-aRollThreshold=5
-aGenerateReport=0
-aGenCodeOnly=0
-aRTWVerbose=1
-aIncludeHyperlinkInReport=0
-aLaunchReport=0
-aForceParamTrailComments=0
-aGenerateComments=1
-aIgnoreCustomStorageClasses=1
-aIncHierarchyInIds=0
-aMaxRTWIdLen=31
-aShowEliminatedStatements=0
-aPrefixModelToSubsysFcnNames=1
-aIncDataTypeInIds=0
-aInsertBlockDesc=0
-aSimulinkBlockComments=1
-aInlinedPrmAccess="Literals"
-aTargetFcnLib="ansi_tfl_tmw.mat"
-aIsPILTarget=0
-aLogVarNameModifier="rt_"
-aGenerateFullHeader=1
-aExtMode=0
-aExtModeStaticAlloc=0
-aExtModeTesting=0
-aExtModeStaticAllocSize=1000000
-aExtModeTransport=0
-aRTWCAPISignals=0
-aRTWCAPIParams=0
-aGenerateASAP2=0
-aInitialSimState="STOP"
-aExecutionMode="real-time"
-aExecutionModeNonUI="RTSIM"
-aTimeScaleFactor="1.0"
-aTimeScaleFactorNonUI="OPTION_DISABLED"
-aAssertionMode="OFF"
-aTAStimulusEngineEnable=0
-aCCompilerCommonOpts=""
-aCCompilerOptimizationOptsPopup="Default (-O5 -D_INLINE)"
-aCCompilerOptimizationOpts=""
-aCCompilerOptimizationOptsNonUI="USE_DEFAULT"
-aEnableDataSetStorage=0
```

-aEnableDataSetStorageNonUI="OFF"
-aLoadAfterBuild=1
-aLoadToFlash=0
-aLoadApplNonUI="ON"
-aPlatformSelectionPopup="Auto"
-aBoardName=""
-aBoardNameNonUI="OPTION_DISABLED"
-aNetworkClient=""
-aNetworkClientNonUI="OPTION_DISABLED"
-aTRCMaskParameters=1
-aTRCGenerateLabels=1
-aTRCGenerateVirtualBlocks=1
-aTRCGenerateStates=0
-aTRCGenerateDerivatives=0
-aTRCApplySubsystemPermissions=0
-aTRCGenerateParamValues=0
-p10000

Loading TLC function libraries

.....
Initial pass through model to cache user defined code
...
*** Postprocessing RTI blocks
*** Starting I/O block checking
*** Passed I/O block checking
Caching model source code
.....
.....
Writing header file closedsys_types.h
Writing header file closedsys.h
.
Writing header file closedsys_private.h
Writing source file closedsys.c
Writing header file rtmodel.h
Writing source file closedsys_data.c
.
Writing header file rt_nonfinite.h
Writing source file rt_nonfinite.c
TLC code generation complete.
Generating TLC interface API.
.....
*** Generating file closedsys_rti.c
.
*** Generating file closedsys_rti.mk
*** Generating Variable Description File closedsys.trc
.....
*** Optional User Variable Description File closedsys_usr.trc not available
*** Found User-Code File closedsys_usr.c from 22-Jul-2008 16:08:10
*** Found User Makefile closedsys_usr.mk from 22-Jul-2008 16:08:10
.
Processing Template Makefile: f:\dspace\matlab\rti1005\m\rti1005.tmf
closedsys.mk which is generated from f:\dspace\matlab\rti1005\m\rti1005.tmf is up to date
Building closedsys: dsmake -f closedsys.mk WORKINGBOARD=ds1005
EXTMODE_STATIC_ALLOC_SIZE=1000000

BUILDING APPLICATION (Single Timer Task Mode)

WORK DIRECTORY "F:\users\LB\AutoboxCAS3"
BUILD DIRECTORY "F:\users\LB\AutoboxCAS3\closedsys_rti1005"
TARGET COMPILER "F:\PPCTools20m"

COMPILING closedsys.c
COMPILING closedsys_data.c
COMPILING rt_nonfinite.c
COMPILING F:\dSPACE\MATLAB\RTI1005\C\rti_sim_engine.c

```
USING LIBRARY "F:\dSPACE\MATLAB\RTI1005\C\Lib\rtwlib_r2006a_ds1005.lib"

LINKING APPLICATION ...
LINKING FINISHED

LOADING APPLICATION "closedsys.sdf" ...
[#1] ds1005 - RTI:                    Initializing ... (720)
[#2] ds1005 - RTI:                Initialization completed (721)
[#3] ds1005 - RTI:                Simulation state: STOP (702)
LOADING FINISHED

MAKE PROCESS SUCCEEDED


### Successful completion of Real-Time Workshop build procedure for model: closedsys
*** Finished RTI build procedure for model closedsys
```

6.6 Az átadott szoftver és dokumentáció diszktérképe

Atad2008Sept

- **SimulinkCAS3** (a kifejlesztett 3 irányítási módszer Simulink modellje és szoftvere)
- **AutoboxCAS3** (a kifejlesztett 3 irányítási módszer AutoBox modellje és szoftvere)
- **DOC_CAS1_embed**
  - **2008Sept_CAS_Doc_Lantos.doc** (dokumentáció)

7. Összefoglalás

A pályázat korábbi fázisában Matlab licenszet nem igénylő stand-alone programcsomagot hoztunk létre gépjármű ütközésmentes pályatervezésére és prediktív irányítására, amely a toolbox szolgáltatások bevonása érdekében nem épített a szabályozó Simulink modelljére, ellenben kihasználta Matlab Compiler szolgáltatásait és host PC-n futtatható, Matlab licenszet már nem igénylő stand-alone programot eredményezett.

A gyors valós idejű működéshez el kellett hagyni a PC-környezetet és speciális target rendszerekre kellett alapozni a megoldást. Ez az út azonban kizárta a Matlab Compiler használatát, és csakis a Simulink –> Real Time Workshop –> Target Compiler szekvenciában tette lehetővé valós idejű megoldás létrehozását. A célrendszerül (target) a Tudásközpont adottságainak megfelelően a dSPACE AutoBox rendszert választottuk. Súlyos korlátozó tényező volt, hogy a Simulink kizárja a toolboxok használatát, és bár a Simulink ú.n. embedded function (beágyazott függvény) blokkja bizonyos lehetőségeket megenged saját fejlesztésű függvények bevonására, a Real Time Workshop ezek körét tovább korlátozza A korlátozások szükségessé tették a korábbi algoritmusok átértékelését és továbbfejlesztését, a pályatervezés és irányítás Simulink alakra hozatalát, lényegében a teljes akadályelkerülési rendszer újratervezését Simulink alapon.

A kutatás első fázisában mintafeladatok keretében felderítettük a Simulink –> Real Time Workshop –> Target Compiler korlátozásait, majd a kutatás második fázisában a tapasztalatokra alapozva létrehoztuk az automatikus akadályelkerülés nemlineáris prediktív irányító rendszerének Simulinkre és beágyazott függvényekre alapozott megvalósítását, és elvégeztük részletes tesztelését először csak Simulink környezetben, majd a dSPACE AutoBox rendszeren. Eközben fokozatosan eliminátuk a szoftver környezet hiányosságait a hibaüzenetek analizálása és a szoftver kolátok megkerülése révén.

Az akadályelkerülő pályát differenciálgeometriai elvű (DGA) és prediktív irányítási (mozgó horizontú, RHC) módszerekkel valósítottuk meg. Prediktív irányítás esetén a szabályozó minden horizont kezdetén meghatározza a mozgó jármű (időinvariáns vagy időben változó) linearizált modelljét az aktuális állapot vagy a teljes állapot-trajektória körül, és a prediktív irányítást a mozgó horizonton belül a keletkező LTI vagy LTV rendszerre alapozza. Az irányításhoz a nem mérhető állapotokat (sebesség, oldalcsúszási szög, orientáció és deriváltja, X és Y pozíció) GPS/INS szenzorok jeleiből állapotbecsléssel határozza meg.

A megvalósítás az akadályelkerülő pálya adatait előfeldolgozott tömör kódolt formában kéri a bemenetek között. A szabályozó az állapotbecslést GPS és IMS érzékelők adataira alapozza, amelyekből a gépkocsi állapotvektorát kétszintű kiterjesztett Kalman-szűrővel határozza meg. A dSPACE AutoBox rendszeren a ControlDesk felügyelete alatt futó valós idejű szabályozó megvalósítás biztosítja a mozgó horizontú nemlineáris prediktív irányítás (RHC) pontossági elvárásaihoz szükséges 10ms mintavételi időt. Jelen fázisban a valósidejű rendszer szimulálja a gépjármű dinamikus modelljét és az érzékelők mérési folyamatát. Ezek az információk később rádiókapcsolattal és CAN buszon keresztül juttathatók el a szabályozóhoz a szabályozó végső technologizálásakor.

Simulink környezetben az implementált 3 irányítási módszer (differenciálgeometriai elvű, integrátort nem tartalmazó prediktív, integrátort tartalmazó prediktív) kiválóan működött és hatékony szabályozást biztosított. Az AutoBox alatt az implementált 3 irányítási módszer közül az első kettő sikeres volt, a harmadik azonban a Real Time Workshop és/vagy a dSPACE szoftver rejtett hibái miatt indítás után elabortálódott, annak ellenére, hogy a hibátlan fordítást protokoll dokumentálta. Jelenlegi tapasztalataink szerint a DS1005 target compiler rejtett hibákkal rendelkezhet a belső függvények (target functions) fordításakor, amelyek a rendelkezésre álló eszközökkel nem határolhatók be.

A fejlesztés során a MATLAB R2006a és a hozzátartozó Simulink, Real Time Workshop és dSPACE AutoBox szoftver és hardver környezetet használtuk. Ennek dSPACE AutoBox része az EJJT Tudásközpont tulajdona és használata hardver kulcshoz kötött, melyet a fejlesztés során a BME IIT Tanszék számára kölcsönzött. A szoftver környezet többi része a BME IIT Tanszék tulajdona.

8. Felhasznált irodalom

1. Freund, E., & Mayr, R. (1997) *Nonlinear Path Control in Automated Vehicle Guidance.* IEEE Transactions on Robotics and Automation, Vol. 13, pp. 49-60.
2. Ryu, J., & Gerdes, J.G. (2004) *Integrating Inertial sensors with Global Positioning System (GPS) for Vehicle Dynamic Control.* Journal of Dynamic Systems, Measurement, and Control. Vol. 126, pp. 242-254.
3. Lantos, B. (2006a). *Algoritmusok gépjármű ütközésmentes pályatervezésére és pályakövetésére. Tanulmány.* BME Irányítástechnika és Informatika Tanszék, p.36. Készült a RET 1.1. Járműforgalmi rendszerek modellezése és irányítása projekt keretében.
4. Lantos, B. (2006b). *Path Design and Receding Horizon Control for Collision Avoidance System of Cars.* WSEAS Transaction on Systems and Control, Issue 2, Vol. 1, pp. 105-112.