**Lantos Béla**

**BME Irányítástechnika és Informatika Tanszék**

# STAND-ALONE C PROGRAMFEJLESZTÉS GÉPJÁRMŰ ÜTKÖZÉSMENTES PÁLYATERVEZÉSÉRE ÉS PREDIKÍV IRÁNYÍTÁSÁRA MÉRHETŐ ÁLLAPOTOK ESETÉN. SZOFTVER ÉS DOKUMENTÁCIÓ

## Tanulmány

Készült a RET 1.1 Járműforgalmi rendszerek modellezése és irányítása projekt keretében

**Elektronikus Jármű és Járműirányítási Tudásközpont**

**Budapest, 2007. június**

# Tartalmi összefoglaló

A Matlab licenszhez kötött fejlesztési környezet rendelkezik olyan fordítóval, amely jelentős megszorításokkal ugyan, de lehetővé teszi a költséges Matlab fejlesztési környezet elhagyását és Matlab licenszet nem igénylő stand-alone futtatható programok létrehozását. Ennek során figyelembe kellett venni, hogy a Matlab interpretált nyelv, és a stand-alone programnak az interpreterek hiányában is korrektül kell működnie. A kutatás során először meghatároztuk a rendelkezésre álló két legfejletteb termék, a Matlab 6.5 (R13) Compiler 3 és a Matlab R2006a (R14) Compiler 4 korlátait stand-alone programok létrehozása során.

Megállapítottuk, hogy a Matlab 6.5 Compiler Version 3 C/C++ nyelvre tudja a Matlab függvényeket lefordítani, amely azután stand-alone programmá összeszerkeszthető, de a Matlab toolboxok nem, vagy csak lényeges megszorításokkal használhatók. Mivel a perspektívikus célok előnyben részesítik a C/C++ nyelvet a továbbfejlesztések során, ezért részletesen analizáltuk a fordítót a korlátok tekintetében. A vizsgálatok eredményeképpen szükség volt a feladat szempontjából kulcsfontosságú Optimization Toolbox lecserélésére egy saját fejlesztésű változatra a korlátok felszámolása érdekében.

A szoftver technológiailag fejlettebb Matlab R2006a Compiler 4 nem C/C++ nyelvre, hanem a felhasználó számára nem definiált és üzleti okokból kriptografikus kulcsokkal védett közbenső CTF formátumra fordít, amely függ a futtató rendszertől, és amelyet azután a stand-alone program az indításkor automatikusan értelmez és végrehajt. A Matlab Optimization Toolbox szolgáltatásai lefordírhatók, a korlátok az alkalmazás szempontjából nem kritikusak, a probléma a C/C++ kimenet hiánya.

Az így szerzett tapasztalatokra alapozva lehetőség nyílt stand-alone program kifejlesztésére az automatikus akadályelkerülő pálya meghatározására és irányítással történő megvalósítására. Ennek során figyelembe vettük a Matlab Compiler megismert korlátait, és a szükséges mértékben átdolgozzuk a korábbi programverziót a stand-alone korlátok figyelembevételével. 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) alkotják, valamint a mozgó akadály és a saját jármű sebessége. További bemeneti adatok azonosítják az útszakaszt (bal oldali és jobb oldali sávszélesség). Az akadályelkerülő pályát az elasztikus szalag elvére épülve határozzuk meg, amely nagyméretű nemlineáris egyenletrendszerre vezet, amelyet az Oprimization Toolbox *fsolve* függvényével oldunk meg.

Az akadályelkerülő pályát differenciálgeometriai elvű és prediktív irányítási (mozgó horizontú, RHC) módszerekkel valósítjuk meg. Minden horizont kezdetén a rendszer 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 kifejlesztendő stand-alone program első verziója mérhető állapotokat (sebesség, oldalcsúszási szög, orientáció és deriváltja, X és Y pozíció) feltételez. Ugyan az összes állapot csak ritkán mérhető közvetlenül, de így lehetőség nyílt az irányítás alaposabb önálló tesztelésére.

A fejlesztés erdményeként a Matlab Compiler 3 felhasználásával 2 stand-alone verzió került kifejlesztésre, egy a saját, egy másik pedig az eredeti Optimization Tollbox felhasználásával (az utóbbi fordításakor számos warning utalt az eredeti toolbox hiányosságaira, amely komoly kockázatot jelent valós idejű körülmények között). Egy harmadik verzió a Matlab Compiler 4 felhasználásával készült. Mindhárom verzió telepítve lett a Matlabot nem tartalmazó futtató környezetben, és helyes müködésük bemutatásra került a BME Közlekedésautomatika Tanszéken a RET 1.1 projekt keretében.

A fejlesztés további iránya kétantennás GPS, valamint 3D gyorsulásérzékelők és giroszkópok adataira épülő állapotbecslés beillesztése lehet a stand-alone programmokba.

# Tartalomjegyzék

# 1. Célkitűzés

A Matlab licenszhez kötött fejlesztési környezet rendelkezik C/C++ nyelvre fordítóval, amely jelentős megszorításokkal ugyan, de lehetővé teszi a Matlab fejlesztési környezet elhagyását és Windows vagy más (Unix, Linux, target processzor) rendszer és C fejlesztési környezet (Visual C stb.) számára C nyelvű programkomponensek előállítását. A C nyelvű programkomponensek a Matlab C Compiler szolgáltatásaival bizonyos korlátozások mellett Matlab licenszet nem igénylő stand-alone futtatható programmá szerkeszthetők.

A kutatási cél kettős: egyrészt fel kívánjuk deríteni a rendelkezésünkre álló két legfejlettebb termék, a Matlab 6.5 (R13) Compiler 3 és a Matlab R2006a (R14) Compiler 4 korlátait stand-alone programok létrehozása során. Másrészt tovább kívánjuk fejleszteni az automatikus akadályelkerülő pálya (CAS) megvalósításához a felső szintű irányítási módszereket, és a korábban kifejlesztett differenciálgeometriai elvű (DGA) irányítás mellé ki kívánjuk fejleszteni a mozgó horizontú (RHC) nemlineáris prediktív irányítást, valamint az irányítási módszerek stand-alone megvalósítását.

Az első cél megoldásakor figyelembe kell venni, hogy a Matlab interpretált nyelv, amely 3 interpretert (Matlab, Simulink, Java) tartalmaz. A stand-alone programok nem igényelik a Matlab licenszet, de nem is tartalmazzák a fejlesztési környezetet és interpreterjeit. A stand-alone programnak az interpreterek hiányában is korrektül kell működnie.

A Matlab 6.5 és a Matlab R2006a között lényeges koncepcióváltást hajtott végre a MathWorks cég: A Matlab 6.5 Compiler Version 3 C/C++ nyelvre tudja a Matlab függvényeket lefordítani, amelyek azután stand-alone programmá összeszerkeszthetők. Matlab toolboxok azonban nem használhatók. Ezzel szemben a Matlab R2006a Compiler 4 nem C/C++ nyelvre, hanem egy (a felhasználó számára nem definiált és üzleti okokból kriptografikus kulcsokkal védett) közbenső CTF formátumra (Component Technology File) fordít, amely függ a futtató rendszertől, és amelyet azután a stand-alone program az indításkor automatikusan értelmez és végrehajt. A Matlab R2006a és toolboxai objektum orientáltak, a Matlab toolboxok néhány szolgáltatása lefordírható, ezek korlátairól a Matlab dokumentációk adnak felvilágosítást. Mivel a perspektívikus célok előnyben részesírhetik a C/C++ nyelvet, célszerű mindkét Matlab fordítót vizsgálni. Ehhez azonban szükség lehet az Optimization Toolbox lecserélésére a korlátok felszámolása érdekében.

A második cél a korábban kifejlesztett Matlab programcsomag stand-alone továbbfejlesztése gépjármű ütközésmentes pályatervezésére és prediktív irányítására mérhető állapotok esetén. Ennek során figyelembe vesszük a Matlab C Compiler megismert korlátait, és a szükséges mértékben átdolgozzuk a korábbi programverziót a stand-alone korlátok figyelembevételével. 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) alkotják, valamint a mozgó akadály és a saját jármű sebessége. További bemeneti adatok azonosítják az útszakaszt (bal oldali és jobb oldali sávszélesség). Az akadályelkerülő pályát az elasztikus szalag elvére épülve határozzuk meg, amely nagyméretű nemlineáris egyenletrendszerre vezet, amelyet az Oprimization Toolbox *fsolve* függvényével oldunk meg.

Az akadályelkerülő pályát differenciálgeometriai elvű és prediktív irányítási (mozgó horizontú, RHC) módszerekkel valósítjuk meg. Minden horizont kezdetén a rendszer 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 kifejlesztendő stand-alone program első verziója mérhető állapotokat (sebesség, oldalcsúszási szög, orientáció és deriváltja, X és Y pozíció) feltételez. Ugyan a feltételezés rendszertechnikailag kérdéses, de lehetőséget ad az irányítás és a későbbi állapotbecslés tesztelésének szétválasztására.

# 2. A MATLAB C Compiler korlátainak felderítése

A Matlab licenszhez kötött fejlesztési környezet rendelkezik fordítóval (Matlab Compiler), amely jelentős megszorításokkal ugyan, de lehetővé teszi a Matlab fejlesztési környezet elhagyását és Windows vagy más (Unix, Linux) rendszer számára Matlab licenszet nem igénylő ú.n. stand-alone futtatható program generálását.

## 2.1 A Matlab Compiler általános megszorításai

1) A Matlab interpretált nyelv, amely 3 interpretert (Matlab, Simulink, Java) tartalmaz. A stand-alone programok nem igényelik a Matlab licenszet, de nem is tartalmazzák a fejlesztési környezetet és interpreterjeit. A stand-alone programnak az interpreterek hiányában is korrektül kell működnie. A fejlesztési környezet elhagyása nem függetleníthető a Matlab Compilert tartalmazó (a fordítást végző) rendszer C/C++ fejlesztő környezetétől és annak könyvtáraitól, valamint a stand-alone programot futtató (Matlab-ot nem tartalmazó) rendszer és környezete könyvtáraitól. A Matlab Compiler kéri a C/C++ fejlesztő környezet megadását, amely számára egységesen Microsoft Visual C++ 6.0 lett megadva (és a fordítást végző rendszeren installálva).

2) A Matlab toolboxok szolgáltatásainak lefordíthatóságát a Matlab Compiler általában nem garantálja, ezért csak a saját fejlesztésű `m`-függvények lefordíthatóságára lehet számítani. Csak a program modulok egyenkénti fordításakor és az összeszerkesztés során kapott hibaüzenetekből azonosítható, hogy a saját fejlesztésű Matlab forrásprogram kielégíti-e a Matlab Compiler elvárásait.

3) A saját fejlesztésű `m`-programok automatikusan bevonhatnak Matlab függvényeket is (lineáris algebra, grafika stb.) a megvalósításukkor, ezért a saját `m`-függvény használata sem abszolút garancia a lefordíthatóságra.

## 2.2 Matlab Compiler verziók eltérései

Stand-alone programok generálására a két legfejlettebb rendelkezésünkra álló verzió, a Matlab 6.5 (R13) és a Matlab R2006a (R14) jöhetett számításba. A két verzió között lényeges koncepcióváltást hajtott végre a MathWorks Inc. cég:

1) A Matlab 6.5 (R13) Compiler Version 3 C/C++ nyelvre tudja a Matlab függvényeket lefordítani, amelyek azután stand-alone programmá összeszerkeszthetők. Matlab toolboxok nem használhatók. A szerkesztés lehetővé teszi megosztott könyvtárak bevonását (DLL) is.

2) A Matlab R2006a (R14) Compiler 4 (két automatikusan generált modul kivételével) nem C/C++ nyelvre, hanem egy (a felhasználó számára nem definiált és üzleti okokból kriptografikus kulcsokkal védett) közbenső CTF formátumra (Component Technology File) fordít, amely függ a futtató rendszertől, és amelyet azután a stand-alone program az indításkor automatikusan értelmez és végrehajt. A Matlab R2006a és toolboxai objektum orientáltak. A Matlab toolboxok néhány szolgáltatása lefordírható, ezek korlátairól a Matlab dokumentációk adnak felvilágosítást. Az Optimization Toolbox szolgáltatásai általában lefordíthatók. Általános korlát, hogy a GUI-t tartalmazó toolbox szolgáltatások nem fordíthatók le. A szerkesztés lehetővé teszi megosztott könyvtárak bevonását (DLL) is.

Kisérleteket végeztünk mindkét Compiler bevonásával az automatikus akadályelkerülő (CAS) és a mozgó horizontú (RHC) prediktív irányítási programrendszer fokozatos konvertálására stand-alone programmokká. Ennek során, mivel szükség volt a Optimization Toolbox bizonyos szolgáltatásaira (`fsolve` stb.), ezért a Matlab 6.5 Compiler esetén módosítottuk és lecseréltük az optimalizálási toolboxot egy saját fejlesztésűre, hogy elkerüljük a fordító által jelzett worning üzeneteket.

Megállapítást nyert a kisérletek során, hogy egyik Compiler sem kezeli a `pause` utasításokat. Ez azért fontos, mert a `pause` teszi lehetővé, hogy a keletkező ábrákat (figures) megszemléljük, mielőtt a számítás továbblépne. Kínálkozott ezért a törekvés a `pause` emulálására egyetlen karakter beolvasására váró `input` utasítással, amelynek keretében a karakter bevitelre várakozás

alatt megfigyeljük az ábrákat, a karakter leütése után pedig továbblépünk a számítás folytatására. Megállapítottuk, hogy a kétféle Compiler az `input` utasítás alatt is eltérően viselkedik:

1) A Matlab 6.5 (R13) Compiler Version 3 felhasználásával keletkezett ábrák az `input` utasítás keretében a karakter leütésre várakozás alatt nem tekinthetők meg, mert a command pront ablaka nem hagyható el. Ezért az `input` utasítás keretében a karakter bevitelre várakozásnak nincs értelme. A számítás nem függeszthető fel, az ábrák futás közben nem tekinthetők meg. Az egyedüli lehetőség a program logikai vége elérésének kivárása, amikor is a keletkezett és megmaradt ábrák (figures) a command promt ablak letörléséig megtekinthetők. Az ábrák nem menthetők world dokumentumba, mert hiányzik a figures menüben a Matlab alatt szokásos felső parancs sor, amelyben a vágólapra mentéshez szerepelne a `copy figure` és `copy option` parancs. Az egyedüli lehetőség a figure fájlba mentése, amely a stand-alone futtatáskor is elérhető a figure parancs sorában (megfelel a Matlab alatt szokásos második menü sornak). A kimentett ábra a fájlból később standard eszközökkel áttehető a word dokumentumba. Vegyük észre, hogy az ábrák megtekintésének befejezése együtt jár a command prompt ablakának lezárásával, ezért újabb stand-alone futtatáshoz új command pont ablak és új path beállítás kell.
2) A Matlab R2006a (R14) Compiler 4 felhasználásával keletkezett stand-alone program futásakor a keletkezett ábrák a `pause` utasítást karakter leütésre váró `input('text','s')` utasítással emuláló megoldással futás közben az `input` elérésekor megtekinthetők, a figures parancs sor szolgáltatásaival pedig igény esetén fájlba menthetők, de a továbblépéshez be kell lépni a command promt ablakába a karakter leütése előtt. A program logikai végének elérésekor az 1) alatti lehetőségek is megmaradnak. Most azonban a program logikai végére kerülhet egy `input`-ra alapozott wait/stop ciklus, amelynek alkalmazásával stop (s) hatására lezárhatók az ábrák (`close all`), és az újabb indításhoz nem kell a path-t ismét beállítani.

## 2.3 Stand-alone programok telepítése

A stand-alone programok mindkét Compiler esetén igénylik egy futtató környezet telepítését a Matlab-ot nem tartalmazó futtató célrendszerre, amelyet szintén előállítottunk. A kétféle Compiler lehetőséget ad az alkalmazási cél függvényében a választásra a könnyebben bővíthető C/C++ filozófiájú (Matlab 6.5) és a kevesebb megszorítást tartalmazó, de CTF formátumú (Matlab R2006a) és ezért nehezebben bővíthető stand-alone program között.

A Matlab környezetben előállított stand-alone C applikációt egy valós alkalmazás esetén tipikusan olyan környezetben kell futtatni, ahol nincs Matlab installálva. A továbbiakban megadjuk azokat a lépéseket, amelyek szükségesek ahhoz, hogy a stand-alone applikáció Matlab hiányában is futtatható legyen. Az átültetési mechanizmus függ attól, hogy az algoritmusok fejlesztéseés a fordítás eredetileg Matlab R13 vagy Matlab 2006a környezetben történt.

### 2.3.1. Átültetési mechanizmus Matlab 2006a esetén

A Matlab környezetben lefordított automatikus akadályelkerülő (CAS) és a mozgó horizontú (RHC) prediktív irányítási program stand-alone applikációja a `Standalone\CAS0R2006a` könyvtárban található. Az applikáció Matlab mentes környezetbe való átültetéséhez 4 fájlra van szükség [6]:

| | |
|---|---|
| `funframeCAS.exe` | A Matlab compiler segítségével előállított fájl, amelynek futtatásával az applikáció majd indítható lesz. |
| `funframeCAS.ctf` | Component Technology File. Ez a fájl tartalmazza kódolt formában a matlab függvényeket. Platform függő fájl, amelynek meg kell egyeznie a végfelhasználó platformjával. |
| `Syspar_File.txt` | Kötött formátumú szövegfájl, amelyben a felhasználónak az = jelek után meg kell adnia az akadályok és a futtatás paramétereit. |
| `MCRInstaller.exe` | Önkitömörítő Matlab Component Runtime library. Platform függő fájl, amelynek meg kell egyeznie a végfelhasználó platformjával. Windows esetén ez a fájl tartalmazza azokat a dll fájlokat, amelyek a végső felhasználó platformján való futtatáshoz szükségesek. A fájl a *`matlabroot`*`\toolbox\compiler\deploy\win32` |

könyvtárban található meg, de a dokumentumhoz csatolt CD-n is megtalálható a `d:\MCR` könytárban

A stand-alone applikáció átültetéséhez szükséges lépések:

1. A végső felhasználó (Windows) platformján adminisztrátori jogkörrel be kell jelentkezni.
2. Az `MCRInstaller.exe` fájlt át kell másolni egy (ideiglenes) munkakönyvtárba.
3. Az `MCRInstaller.exe` indításával egy varázsló végigvezet az installációs épéseken. Itt megadható a Runtime library végső helye a célgépen.
4. Kilépés adminisztrátori jogkörböl és felhasználói jelszóval bejelentkezés.
5. `funframeCAS.exe, funframeCAS.ctf` és `Syspar_File.txt` fájlok átmásolása.
6. `funframeCAS.exe` program futtatása. (Nem szükséges Command Prompt-ból indítani)

Bármely további futtatás során csak a 6. lépést kell végrehajtani.
A program első futtatásakor keletkezik egy `funframeCAS.mcr` könyvtár, ezért az első futás lassúbb, mint a többi. Ez a könyvtár 3 további alkönyvtárat és bennük ismét alkönyvtárakat és fájlokat tartalmaz. Figyelem: az ezekben található *.m fájlok nem Matlab függvények, hanem speciálisan kódolt információ a futtatáshoz.

### 2.3.2. Átültetési mechanizmus Matlab 6.5 esetén

A Matlab környezetben lefordított automatikus akadályelkerülő (CAS) és a mozgó horizontú (RHC) prediktív irányítási program stand-alone applikációi a `Standalone\CAS06p5` és a `Standalone\OrigCAS06p5` könyvtárakban találhatók. A saját fejlesztésű Optimization Toolbox alkalmazása esetén `CAS06p5` (fordítás során nem volt warning), az eredeti Matlab Optimization Toolbox alkalmazása esetén `OrigCAS06p5` (fordítás során volt warning) tartalmazza a fordítás eredményeit. A továbbiakban csak a `CAS06p5` esetet mutatjuk be, a másik esetben hasonlóan kell eljárni, de értelemszerűen a másik nevet kell alkalmazni. Az applikáció Matlab mentes környezetbe való átültetéséhez 4 fájlra van szükség [5]:

| | |
|---|---|
| `funframeCAS.exe` | A Matlab compiler segítségével előállított fájl, amelynek futtatásával az applikáció majd indítható lesz. |
| `Syspar_File.txt` | Kötött formátumú szövegfájl, amelyben a felhasználónak az = jelek után meg kell adnia az akadályok és a futtatás paramétereit. |
| `bin` | A `FigureMenuBar.fig` és a `FigureToolBar.fig` fájlokat tartalmazza. |
| `dataread.dll` | Az `mglinstaller.exe` hibája folytán pótolandó fájl, amely a Matlab *`matlabroot`*`\bin\win32` része |
| mglinstaller.exe | Önkitömörítő Matlab Compiler Run-Time Library installációs fájl. Windows esetén ez a fájl tartalmazza azokat a dll fájlokat, amelyek a végső felhasználó (Windows alapú) platformján való futtatáshoz szükségesek. A fájl a *`matlabroot`*`\extern\lib\win32` könyvtárban található meg, de a dokumentumhoz csatolt CD-n is megtalálható a `d:\mglinstaller` könytárban |

A stand-alone applikáció átültetéséhez szükséges lépések:

1. A végső felhasználó (Windows) platformján adminisztrátori jogkörrel be kell jelentkezni.
2. Az `mglinstaller.exe` fájlt át kell másolni abba a könyvtárba, ahová installálni szeretnénka programot. Legyen ez pl.: `C:\Program Files\MGLI`
3. Az `mglinstaller.exe` indításával a Matlab Compiler Run-Time Library fájljai kicsomagolódnak a megadott könyvtárban.
4. Kilépés adminisztrátori jogkörböl és felhasználói jelszóval bejelentkezés.
5. `funframeCAS.exe, Syspar_File.txt, bin, dataread.dll` fájlok átmásolása a végfelhasználó számítógépére.
6. Command Prompt inditása (Az operációs rendszer Start\Kellékek ill. angol nyelvű Start)

7. A Command Promptban a cd paranccsal `funframeCAS.exe` fájl könyvtárának aktuálissá tétele.
8. A Command Promptban a path paranccsal a Run-Time Library elérési útvonalát hozzá kell adni a korábbi útvonalakhoz. Ez az `mglinstaller.exe` könyvtárán belül a `\bin\win32` alkönyvtárban található. A 2. lépésben megadott könyvtár esetén az a következő parancs kiadását jelenti:
   ```
   path C:\Program Files\MGLI\bin\win32; %path%
   ```

   *Megjegyzés:* A Command Prompthoz tartozó ablak bezárása és újranyitása után ezt a parancsot meg kell ismételni. Gyakori futtatásnál célszerű egy `.bat` kieterjesztésű fájlba másolni a fenti parancssort és ezt a batch fájlt inditani.

9. A `funframeCAS.exe` program futtatása Command Promptból, ami egyszerűen a `funframeCAS` begépelésével és az Enter billentyű megnyomásával megtehető.

Bármely további futtatás esetén az előzőekben megadott lépéssorozatot a 6. lépéstől kell végrehajtani.

## 2.4 Az automatikus akadályelkerülő és prediktív irányítási program stand-alone applikációjának felhasználói leírása

A program a korábban kifejlesztett Matlab programcsomag továbbfejlesztése az automatikus akadályelkerülő és a mozgó horizontú prediktív irányítási algoritmus stand-alone megvalósításával. A továbbfejlesztett program integrálva tartalmazza az akadályelkerülő pálya tervezését, a differenciálgeometriai elvű (DGA) és a mozgó horizontú (RHC) prediktív irányítást, a jelen fázisban mérhető állapotokat feltételezve.

Ennek során tekintetbe vettük a Matlab Compiler megismert korlátait, és a szükséges mértékben átdolgoztuk a korábbi programverziót a stand-alone korlátok figyelembevételével. 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) alkotják, valamint a mozgó akadály és a saját jármű sebessége. További bemeneti adatok azonosítják az útszakaszt (bal oldali és jobb oldali sávszélesség). Az akadályelkerülő pályát az elasztikus szalag elvére épülve határozzuk meg, amely nagyméretű nemlineáris egyenletrendszerre vezet, amelyet az Oprimization Toolbox *fsolve* függvényével oldunk meg. Az akadályelkerülő pályát differenciálgeometriai elvű és prediktív irányítási (mozgó horizontú, RHC) módszerekkel valósítjuk meg. Minden horizont kezdetén a rendszer meghatározza a saját jármű (időinvariáns vagy időben változó) linearizált modelljét az aktuális állapot körül, és a prediktív irányítást a mozgó horizonton belül erre alapozza. A program első verziója feltételezi az összes állapot (sebesség, oldalcsúszási szög, orientáció és deriváltja, $X$ és $Y$ pozíció) mérhetőségét. Ugyan a mérhető állapotok feltételezése rendszertechnikailag kérdéses, mindazonáltal lehetőséget ad az irányítás és a későbbi állapotbecslés tesztelésének szétválasztására.

Mind a Matlab 6.5 Compiler 3, mind pedig a Matlab R2006a Compiler 4 felhasználásával kifejlesztettük a stand-alone program egy-egy verzióját gépjármű ütközésmentes pályatervezésére és prediktív irányítására mérhető állapotok esetén. A kétféle Compiler lehetőséget ad az alkalmazási környezet és a potenciális hosszú távú bővítések elvárásai függvényében a választásra a könnyebben bővíthető C/C++ filozófiájú (Matlab 6.5), valamint a kevesebb megszorítást tartalmazó, de a CTF közbenső formátum (Matlab R2006a) miatt nehezebben bővíthető stand-alone programok között.

A program vezérlése egységesen a `SysparFile.txt` alapján történik. A szövegfájl kötött formátumú, az egyes paramétereket a nevük azonosítja, amely `=` jelre végződik, és amely után kell megadni a paraméter numerikus vagy string értékét. A `%` jeltől a sor végéig a karakterek szűrve lesznek (`comment`). A paraméterek értelmezéséhez kellő mértékben el kell sajátítani a soronkövetkező fejezeteket (akadályok megadása, rendszermodellek, szabályozási módszerek stb.).

**Syspar_File.txt fájl preparált tartalma az átadott anyagban:**

```
%************************************************
%Input file for CAS system parameters
%************************************************
%initband parameters
fv_own=20;                  %Own car average velocity
fstat_obs1=[40 0 2.5];      %[rx ry d] static_obstacle_1
fmov_obs=[120 3.5 4 15];    %[rx ry d v] moving_obstacle
froad_wide=[7 0.75 0.25];   %[total_wide left_portion right_portion]
%funframeCAS parameters
fsys_appr=0;                %1->approximated_model_in_use
fsys_estim=0;               %1->state_estimator_is_running
fsys_contr='nonlinpred';    %1->predictive_control_in_use
fdeltaw_horizon=0;          %1->u(1)_is_deltaw, 0->u(1)_is_Sv_transversal
fdgfresh_horizon=1;         %1->uN_by_diffgeom, 0->xNp1_to_0, 2->uN_to_uNm1
flambda_horizon=10;         %lambda_weights_u_or_deltau_in_cost_function
fint_horizon=1;             %1->integrator_in_RHC_controller
fLTV_horizon=1;             %1->LTV_linearization_in_the_horizons
```

**A paraméterek jelentése:**

| | |
|---|---|
| `fv_own` | saját kocsi átlagsebessége |
| `fstat_obs1` | statikus akadály *rx, ry* koordinátái és *d* átmérője (a saját kocsi koordinátái a kiinduláskor (0,0) a jobb oldali sáv közepén, minden akadály koordinátája ehhez értendő) |
| `fmov_obs` | mozgó akadály *rx, ry* koordinátái, *d* átmérője és *v* sebessége |
| `froad_wide` | kétsávú út teljes szélessége, a bal oldali sávhatár aránya és a jobb oldali sávhatár aránya a saját kocsi kiindulási helyzetében |
| `fsys_appr` | tranziensek szimulálásakor használt nemlineáris modell, 1=>közelítő, 0=>precíz |
| `fsys_estim` | 1=>van állapotbecslés, 0=>nincs állapotbecslés, |
| `fsys_contr` | ‘nonlinpred’ vagy ‘diffgeom’ a választott szabályozási algoritmus szerint |
| `fdeltaw_horizon` | 1=>az első beavatkozó jel az első kerék $\delta w$ szögelfordulása, 0=>az első beavatkozó jel az első kerék *Sv* transzverzális ereje, (második beavatkozó jel mindig a hátsó kerék *FlR* longitudinális ereje) |
| `flambda_horizon` | a beavatkozó jel nominálistól való eltéréséből álló $U$ sorozat normájának $\lambda$ súlyozó tényezője a költségfüggvényben a nagy beavatkozások büntetésére |
| `fint_horizon` | 1=>van integrátor a szabályozóban, 0=>nincs integrátor a szabályozóban |
| `fLTV_horizon` | 1=>LTV modell a horizontban, 0=>LTI modell a horizontban, amely a nemlineáris modell linearizálása révén áll elő prediktív irányítás esetén |

**Nem változtatható paraméterek:**

A jármű modelljének paramétereit az `initvehicle.m`, az elasztikus szalag inicializálási adatait az `initband.m` függvény tatalmazza (lásd Matlab `*.m` függvények listája). A mintavételi idő $T = 0.01\,\mathrm{s}$, a horizont mérete a mintavételek számában mérve $N = 10$.

**A keletkező ábrák jelentése:**

| | |
|---|---|
| Figure 1 | Az elasztikus szalag kiindulási helyzete |
| Figure 2 | Az elasztikus szalag egyensúlyi helyzete |
| Figure 3 | Az *x(t)* jel és deriváltjai |
| Figure 4 | Az *y(t)* jel és deriváltjai |
| Figure 5 | A pályamenti sebesség és deriváltja |
| Figure 6 | A pályamenti görbület és a pálya |
| Figure 7 | Az orientáció és deriváltjai |
| Figure 8 | Az *x* jel második deriváltja |
| Figure 9 | Az *x* jel harmadik deriváltja |
| Figure 10 | Az *y* jel második deriváltja |
| Figure 11 | Az *y* jel harmadik deriváltja |
| Figure 12 | Az irányítások *x(t)* és *y(t)* alapjelei |
| Figure 13 | Állapotváltozók |
| Figure 14 | Beavatkozó jelek |
| Figure 15 | A hibajelek alakulása zárt szabályozási körben |

## 2.5 CD-n átadott fájlok térképe

**Lantos_2007_06_26**
- CAS0_Doc
  - CAS0_Doc_Lantos.doc, CAS0_Doc_Lantos.pdf, M1.doc, M2.doc
- CAS0_6p5
  - OptimModif, PrivatexModif, SourceAndCompiled, Standalone
- OrigCAS0_6p5
  - SourceAndCompiled, Standalone
- CAS0_R2006a
  - SourceAndCompiled, Standalone

**Standalone**
- CAS06p5
  - funframeCAS.exe, Syspar_File.txt, bin, dataread.dll
- OrigCAS06p5
  - funframeCAS.exe, Syspar_File.txt, bin, dataread.dll
- CAS0R2006a
  - funframeCAS.exe, funframeCAS.ctf, Syspar_File.txt, funframeCAS_mcr
- mginstaller6p5
- mceinstallerR2006a

# 3. Automatikus ütközésmentes pályatervezési algoritmus

Az automatikus pályatervezés központi problémája a gépjármű ütközést elkerülő rendszereknek (Collision Avoidance Systems, CAS). A megvalósított pályatervezési algoritmus az elasztikus szalag elvén alapul (Brandt & Sattel, 2005). A módszer lehetővé teszi a pályatervezés során statikus akadályok (például elől haladó járműről leeső teher) és dinamikus akadályok (például másik sávban szembejövő jármű) kikerülését. A pályatervezés automatikus és valós időben zajlik. A statikus és dinamikus akadályok paraméterei a program bemeneti adatai, amelyek a `SysparFile.txt` kötött szitaxisú szövegfájlban adhatók meg. Ezek az információk rendszerint valós idejű képfeldolgozás eredményei, amellyel a rendszer a jövőben bővíthető. A pályatervezési algoritmus a következő lépésekre bontható:

1. Induló elasztikus szalag generálása és iterálása az erőegyensúly eléréséig.
2. Pályatervezés az egyensúlyi akadályelkerülő elasztikus szalaghoz.
3. A pályához tartozó állapotváltozók kiszámítása.

## 3.1 Induló elasztikus szalag generálása

A kifejlesztett program a gyakorlat számára fontos szituációra, egy statikus akadály és egy szembejövő jármű esetére koncentrál. Ez a lépés a bemeneti adatként szolgáló specifikáció (statikus akadály elhelyezkedése és mérete az útpálya sávjaiban, szembejövő dinamikus akadály kiindulási helyzete, mérete és sebessége) ismeretében felvesz egy induló elasztikus szalagot. A bemeneti specifikációban a pályatervezés kezdeti $t_0$ pillanatában az akadályelkerülést végző járműhöz relatívan kell megadni a statikus és dinamikus akadályok geometriai adatait és a szembejövő jármű sebességét. Az akadályelkerülést végző jármű $(0,0)^T$ kiindulási helyzetéhez (tipikusan a saját sáv közepéhez) relatívan kell meghatározni az akadályelkerülési pályát. Meghatározásra kerül a pályatervezésre rendelkezésre álló becsült maximális idő és az akadályelkerülést végző jármű becsült célhelyzete. A becsült célhelyzet és az akadályok helyzetének ismeretében tisztán geometriai szemlélet alapján történik az akadályelkerülés egy megengedett kiindulási pályájának kiválasztása és elasztikus szalaggal történő közelítése. Ennek során az akadályoktól és az úttest határaitól való távolságtartásra, valamint egyenesekből és körívekből felépülő pálya kiválasztására kell törekedni. Az elasztikus szalag szekcióinak darabszáma és a rugók kiindulási hossza programkonstans, nagyságrendjük kb. 40 csomópont illetve 1m.

## 3.2 Az akadályelkerülő elasztikus szalag generálása

Az elasztikus szalag (elastic band) egy rugókból láncszerűen összeillesztett rendszer. A rugók potenciál mezeje, az akadályok virtuális potenciál mezeje és az útszegélyek virtuális potenciál mezeje eredőjében a jármű pályája úgy lesz kiválasztva, hogy a potenciál mezőkből keletkező erőhatások egyensúlyban legyenek. A pályatervezés egy iteraratív folyamat, mivel véletlenszerűen statikus akadály keletkezhet és jármű bukkanhat fel. A környezetben bekövetkező változások beindítják a pályatervezési folyamatot.

### 3.2.1 Az elasztikus szalag struktúrája

A jármű (vehicle) pillanatnyi helyzetében az $(x_v, y_v)$ koordináta-rendszer origójából $(r_0)$ indul az elasztikus szalag. A szalag $N$ darab rugóból áll, a rugók csatlakozási helyei a csomópontok (nodes), melyeket a jármű koordináta-rendszerében $r_i$ azonosít az $i$-dik csomópont esetén. A rugószakasz rugóállandója $k_i$, a rugószakasz kezdeti hossza $l_{0,i}$, amelyek a pályatervezés számára megválaszthatók. Az erőegyensúlyi helyzet keresésekor minden iteratív lépésben meghatározásra kerül a csomópont $t_i$ elérési ideje is, amely a csomópontok közötti lineáris interpoláción és a jármű megválasztható átlagos longitudinális sebességén alapul. Ha az erőegyensúlyi helyzethez tartozó pálya az iteráció során meghatározásra került, akkor a jármű

végighalad az elasztikus szalag csomópontjain az átlagos longitudinális sebesség alapján számított időpontok figyelembevételével, a csomópontok között spline-technikával interpolálva.

Az elasztikus szalagot a következő információ jellemzi:

$$(r_0, t_0) \xrightarrow{k_0, l_{0,i}} (r_1, t_1) \xrightarrow{\cdots} (r_i, t_i) \xrightarrow{k_i, l_{0,i}} (r_{i+1}, t_{i+1}) \xrightarrow{\cdots} (r_N, t_N) \tag{3.1}$$

Az akadályokat biztonsági körök modellezik, az $O_j$ akadály biztonsági körének középpontja $r^{O_j}$, átmérője pedig $d_j$, amelybe beleértendő az akadályt teljesen lefedő kör átmérője és a jármű szélessége is.

### 3.2.2 Elasztikus szalag belső potenciálja

Az elasztikus szalag minden $r_i$ csomópontjához belső potenciálként választható a rugószakasz potenciális energiája (ennek negatív gradiense lesz a csomópontra ható erő):

$$V_i^{\text{int}} = \frac{1}{2} k_i (|r_{i+1} - r_i| - l_{0i})^2, \tag{3.2}$$

ahol $|r_{i+1} - r_i| = \sqrt{(r_{i+1,x} - r_{i,x})^2 + (r_{i+1,y} - r_{i,y})^2}$ az euklideszi távolság. A teljes elasztikus szalag poteniális energiája ezért

$$V^{\text{int}} = \sum_{i=0}^{N-1} V_i^{\text{int}} = \sum_{i=0}^{N-1} \frac{1}{2} k_i (|r_{i+1} - r_i| - l_{0i})^2 . \tag{3.3}$$

### 3.2.3 Az útszegélyek külső potenciáljai

Az útszegélyek külső taszító potenciálja csökkenjen logaritmikusan az út középvonala felé haladva. Az útszegélyek legyenek folytonosak. Az útszegélyek mentén referencia pontokat választunk, amelyekbe mutassanak az $r_j^{B_q}$, $q \in (l, r)$ vektorok, ahol az $l$ index a bal oldali (left), az $r$ index pedig a jobb oldali (right) útszegély referencia pontjait különbözteti meg. Az elasztikus szalag minden $r_i$ csomópontjához meghatározásra kerül egy-egy $r_i^{B_l}$ és $r_i^{B_r}$ referencia pont az útszegélyeken (tipikusan az $r_i$ csomóponthoz legközelebbi), és az útszegély külső hatását az $r_i$ csomópontra egyedül csak ezek fogják meghatározni. Az $r_i$ csomópontra ható és az útszegélyektől származó erő legyen $F_i^{B_q}$.

Párhuzamos útszegély estén akadálymentes esetben elvárható, hogy az elasztikus szalag a jobb oldali sáv közepére legyen pozícionálva, ezért $b$ széles utat és 2 sávot (egyet-egyet mindkét irányban) feltételezve az $F_i^{B_l} = F_i^{B_r}$ erőegyensúly miatt

$$\frac{k^{B_l}}{k^{B_r}} = \frac{0.75b}{0.25b} = 3 \tag{3.4}$$

aránynak kell teljesülnie az útszegélyek rugóállandói között. A rogramban az útszegély potenciálhoz tartozó erőt úgy választottuk meg, hogy akadálymentes esetben az induló elasztikus szalagot a saját sáv közepére visszapozicionálja, továbbá hasonlóan hasson mind a bal oldali, mind pedig a jobb oldali sávhatáron:

$$F_i^{B_q} = M^B \exp[-\frac{1}{2}(|r_i - r_i^{B_q}| / \sigma^{B_q})^2] \cdot \frac{r_i - r_i^{B_q}}{|r_i - r_i^{B_q}|}. \tag{3.5}$$
$$\sigma^{B_q} = k^{B^q} / \sqrt{2 \ln(M^B / m^B)}$$

A programban alkalmazott választás $M^B = 2,\ m^B = 0.05$.

### 3.2.4 Statikus akadályok külső potenciálja

A statikus akadályok (static obstacles) külső potenciálja alapvetően befolyásolja az erőegyensúlyhoz tartozó elasztikus szalag alakját, és így indirekt módon a pályának vezetői szempontból esztétikus voltát, például a pálya görbületét, a centripetális gyorsulást stb. Ennek biztosításához a statikus akadálytól távolodva csak lassan megszünő taszító hatást kell kifejtenie az akadálynak, amely az eredeti exponenciális lecsengéssel nem biztosítható, ezért a programban a statikus akadályhoz tartozó erőt a következőnek választottuk:

$$F_i^{O_j} = k^{O_j} \frac{d^{O_j}/2}{|r_i - r_i^{O_j}|} \cdot \frac{r_i - r_i^{O_j}}{|r_i - r_i^{O_j}|} \tag{3.6}$$

A programban alkalmazott választás $k^{O_j} = 3$, a statikus akadály átmérőjét $d^{O_j}$ jelöli.

### 3.2.5 Mozgó akadályok külső potenciálja

A mozgó $O_j$ akadály (moving obstacle) potenciálfüggvénye hasson minden $i$ csomópontra (nem csak a hozzá közel lévőre).

A potenciálfüggvény kiértékelésekor figyelembe kell venni, hogy az elasztikus szalag $i$ csomópontja az $r_i$ pontban a $t_i$ időpontban lesz, ezért az ehhez az időponthoz tartozó $r^{O_j}(t_i)$ akadályhelyzetet meg kell határozni. Ehhez szükség van az akadály mozgását jellemző kezdeti $r^{O_j}(t_0)$ akadályhelyzetre és $v^{O_j}(t_0)$ feltételezett akadály sebességre, amelyekből lineáris extrapolációval számítható ki $r^{O_j}(t_i)$. Az $O_j$ akadály által az elasztikus szalag $i$ csomópontjára ható erő a programban a mozgó akadály hatását az aktuális környezetére korlátozza:

$$F_i^{O_j} = k^{O_j} \exp\left(|r_i - r^{O_j}(t_i)| - \frac{d_j}{2}\right)^2 \cdot \frac{r_i - r^{O_j}(t_i)}{|r_i - r^{O_j}(t_i)|}. \tag{3.7}$$

### 3.2.6 Az egyensúlyi helyzet meghatározása

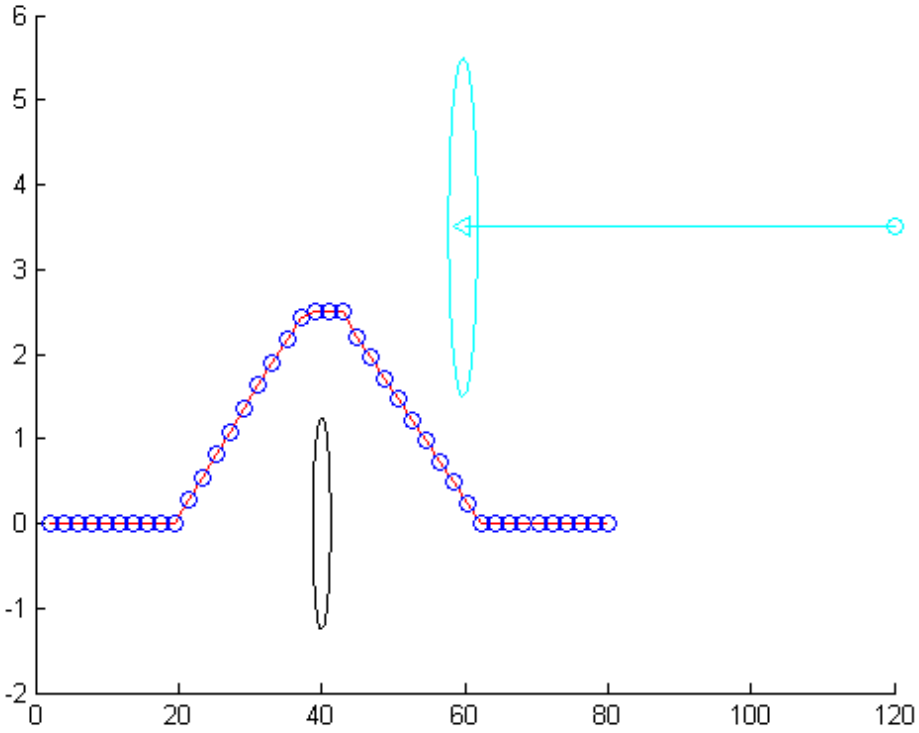
Legyen a mozgó és álló akadályok száma $M$. Az elasztikus szalag minden egyes $i$ csomópontjára ható erőnek egyensúlyban kell lenni:

$$F_i^{sum} = F_i^{\text{int}} + F_i^{B_l} + F_i^{B_r} + \sum_{j=1}^{M} F_i^{O_j} = 0. \tag{3.8}$$

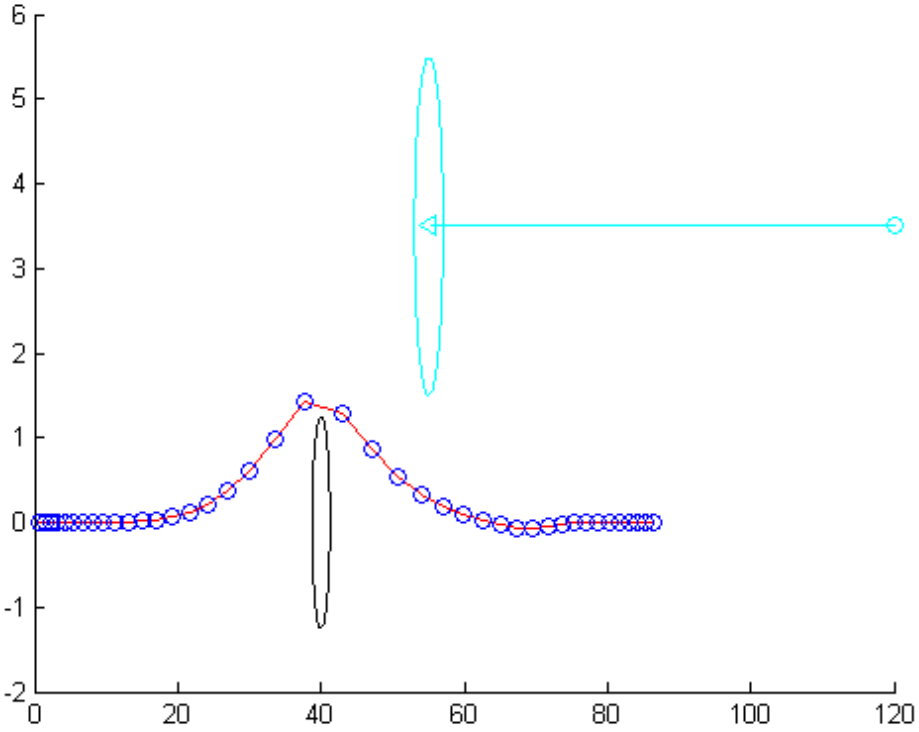
Az egyensúlynak minden $i$ csomópontra szimultán kell fennállnia. Legyen ezért

$$x = (r_1^T, r_2^T, \ldots, r_N^T)^T \text{ és } f(x) = \left((F_1^{sum})^T, (F_2^{sum})^T, \ldots, (F_N^{sum})^T\right)^T, \tag{3.9}$$

akkor a feladat az $f(x) = 0$ nemlineáris egyenletrendszer megoldása.




*3.1. ábra.* Az elasztikus szalag kiindulási helyzete




*3.2 ábra.* Az elasztikus szalag egyensúlyi helyzete.

Az egyensúlyi helyzet meghatározására az `fsolve` függvényt használtuk, amely az Optimization Toolbox része.. Jelentős gyorsítást sikerült elérni a futási idő tekintetében azáltal, hogy a függvény mellett a függvény deriváltját (Jacobi-mátrixát) is átadtuk `fsolve`-nak, és a függvény és deriváltja számítását vektorizáltuk. Az akadálymentes pálya tervezésében résztvevő saját fejlesztésű fontosabb MATLAB függvények (függvény prototípus és 1 soros comment a help-hez):

```
function initband
```
%Initiate elastic band

```
function [fout,Jout]=banditer(rin)
```
%Compute forces, Jacobian and new nodes for elastic band

Az `fsolve`-nak átadott függvény `banditer`, amelynek bemenete az aktuális elasztikus szalag az iteráció során, kimenete pedig a függvény értéke és Jacobi-mátrixa.

Az akadálymentes pályatervezés eredményét egy tipikus esetben mutatjuk be (41 csomópont, egy 2.5m átmérőjű statikus és egy 3.5m átmérőjű mozgó akadály, az utóbbi sebessége 15m/s, a saját jármű átlagos sebessége 20m/s). A kiindulási helyzetet a 3.1. ábra, a megtalált egyensúlyi helyzetet a 3.2 ábra mutatja.

## 3.3 Referencia jel tervezés az irányításokhoz

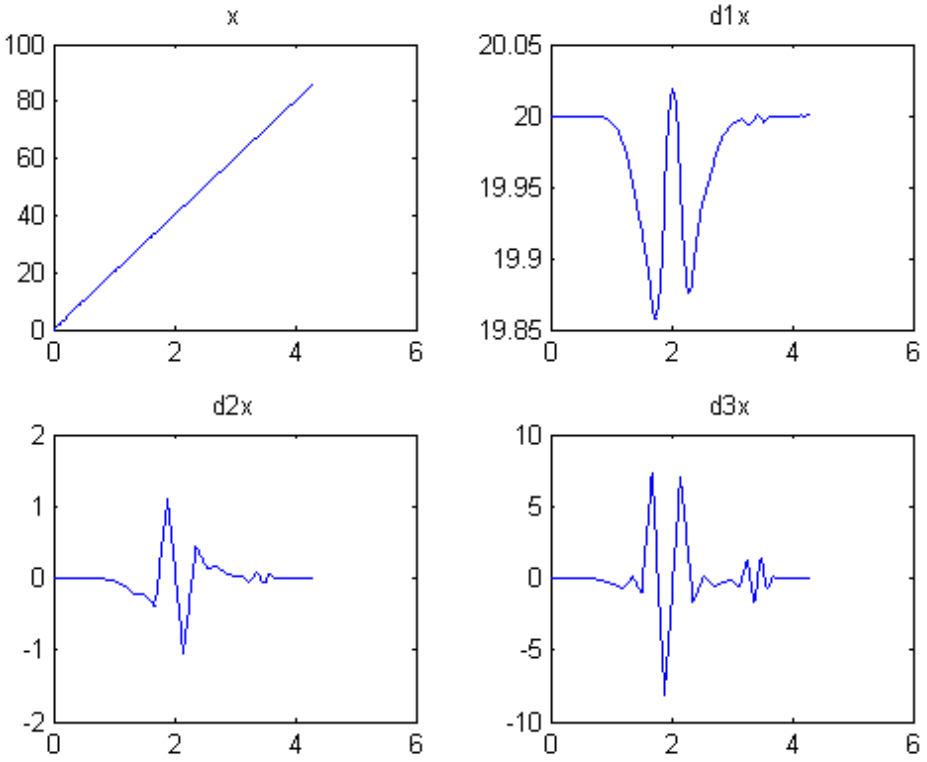
Az elasztikus szalag egyensúlyi helyzete az akadályelkerülő pálya absztrakt formája, amelyet azonban az irányításokhoz referencia jel időfüggvényekké kell konvertálni. Célszerű ezen kívül még a mozgás további kinematikai jellemzőit (a pálya idő szerinti magasabb deriváltjait) is meghatározni, mivel ezek jelentősen támogatni tudják a rendszertechnikai vizsgálatokat.

A kifejlesztett programban az elasztikus szalag által generált pályához a saját jármű névleges sebességének figyelembevételével a MATLAB spline-szolgáltatásaival (`spline, ppval, unmkpp, mkpp`) meghatározzuk a pályához tartozó kinematikai jellemzőket és az időparaméter elosztását, melyek az irányítások számára referencia jelként fognak szolgálni.

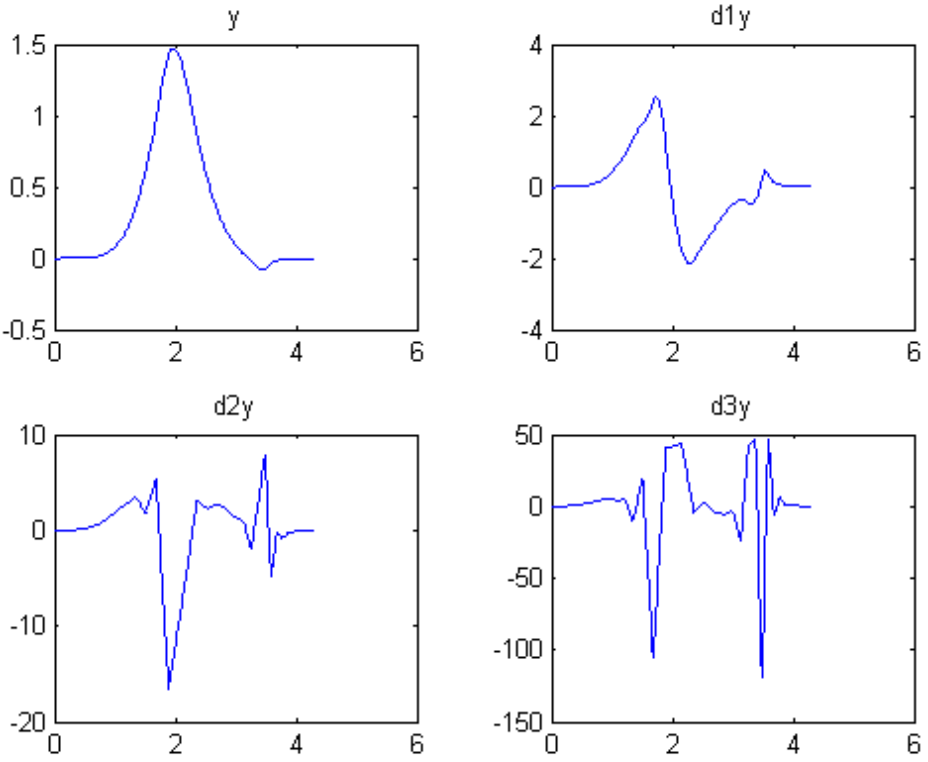
Az elasztikus szalag csomópontjait először poligonnal (törött vonallal) approximáljuk, amely egy $(x, y)$ adatsorozat, majd a saját jármű átlagos sebességének felhasználásával meghatározzuk a csomópontokhoz tartozó $(t, x)$, $(t, y)$ adatsorozatokat. Ezután a MATLAB spline technikájának bevonásával a csomópontok között harmadfokú polinomokkal approximálunk, ami lehetővé teszi a deriváltak számítását a harmadik deriválttal bezárólag. Mivel azonban ez utóbbi már szakaszonként konstans, ami a dinamikus tulajdonságok miatt kedvezőtlen, ezért ennek simítására az első deriváltat ismét approximáljuk harmadfokú polinommal, ami sima harmadik deriváltat eredményez.

Az eredményeket a spline-technika alkalmazása után a 3.3-3.4. ábrák mutatják. Ezekből a jelekből nulla oldalcsúszási szöget feltételezve számítjuk a pályamenti sebesség *v* abszolút értékét és *d1v* első deriváltját, valamint a *kappa* pályamenti görbületet. Az eredményeket a 3.5-3.6.ábrák mutatják, bemutatván a második approximáció simító hatását is. A spline-techniával kapott sima pályát szintén a 3.6. ábra tartalmazza. Az orientációt és magasabbrendű deriváltjait, az $x$ és $y$ jelek magasabbrendű deriváltjait, valamint az $x(t)$ és $y(t)$ referencia jeleket (alapjel időfüggvényeket) rendre a 3.7-3.12. ábrák mutatják be.

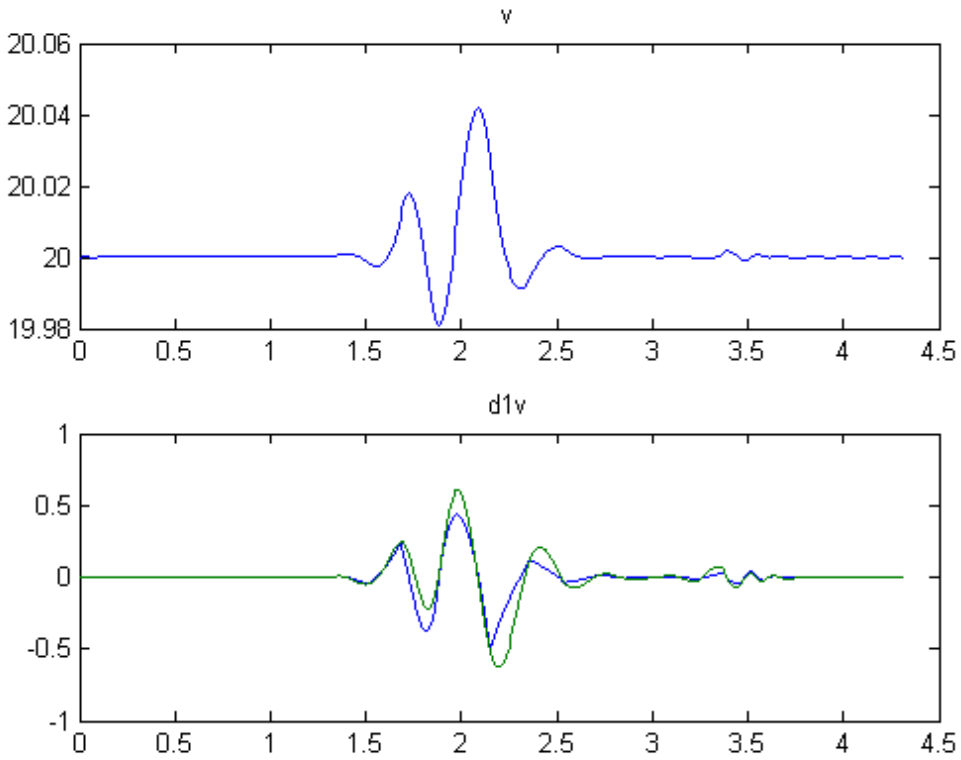
Itt és minden további esetben a jeleket SI egységben kell érteni (pl. s, m, m/s, rad, rad/s stb.).



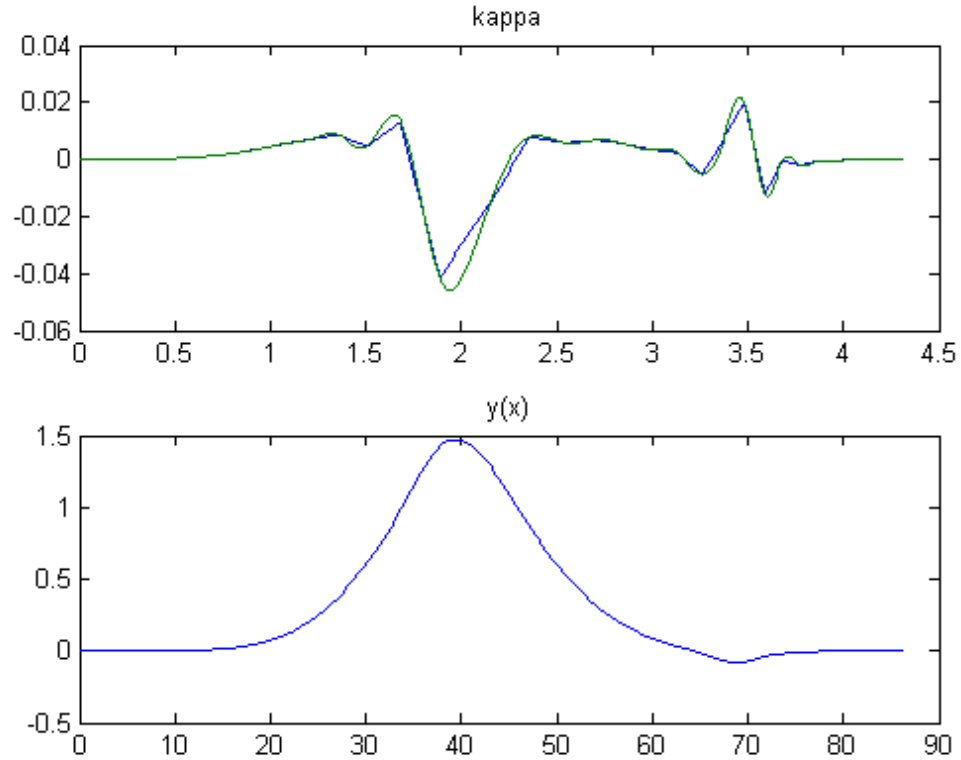

*3.3. ábra.* Az *x(t)* jel és deriváltjai



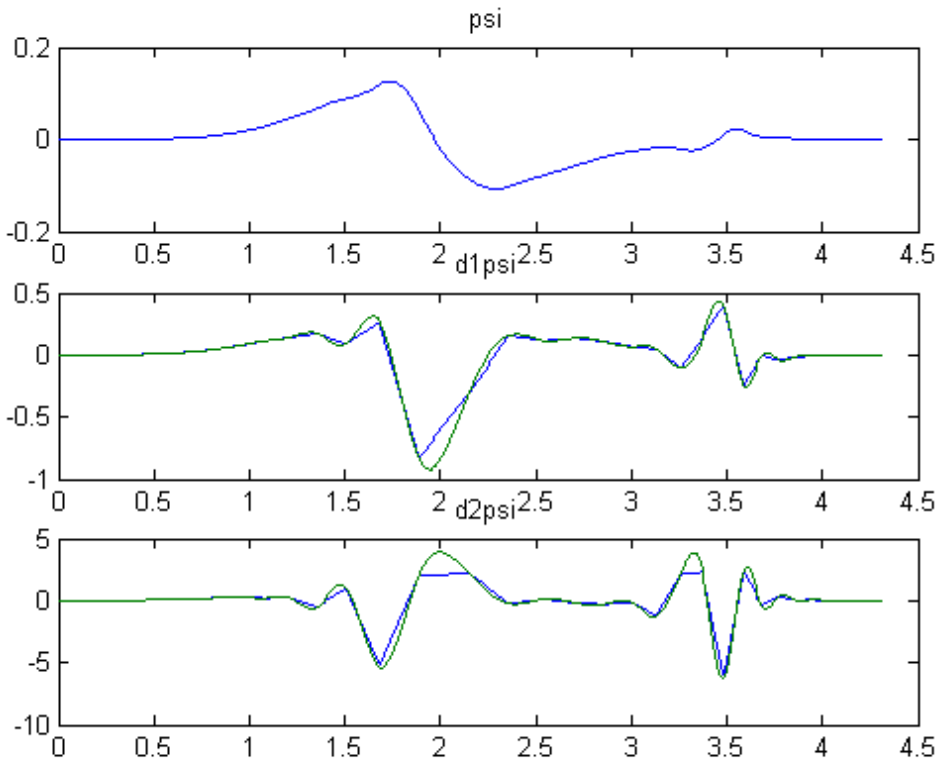

*3.4. ábra.* Az *y(t)* jel és deriváltjai



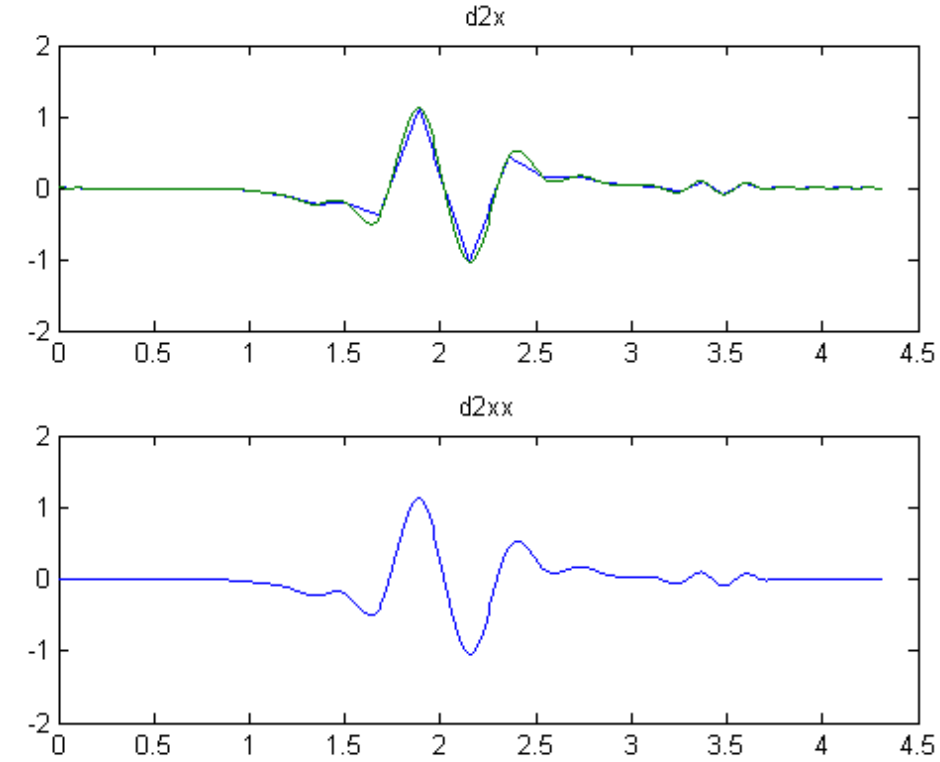

*3.5. ábra.* A pályamenti sebesség és deriváltja.



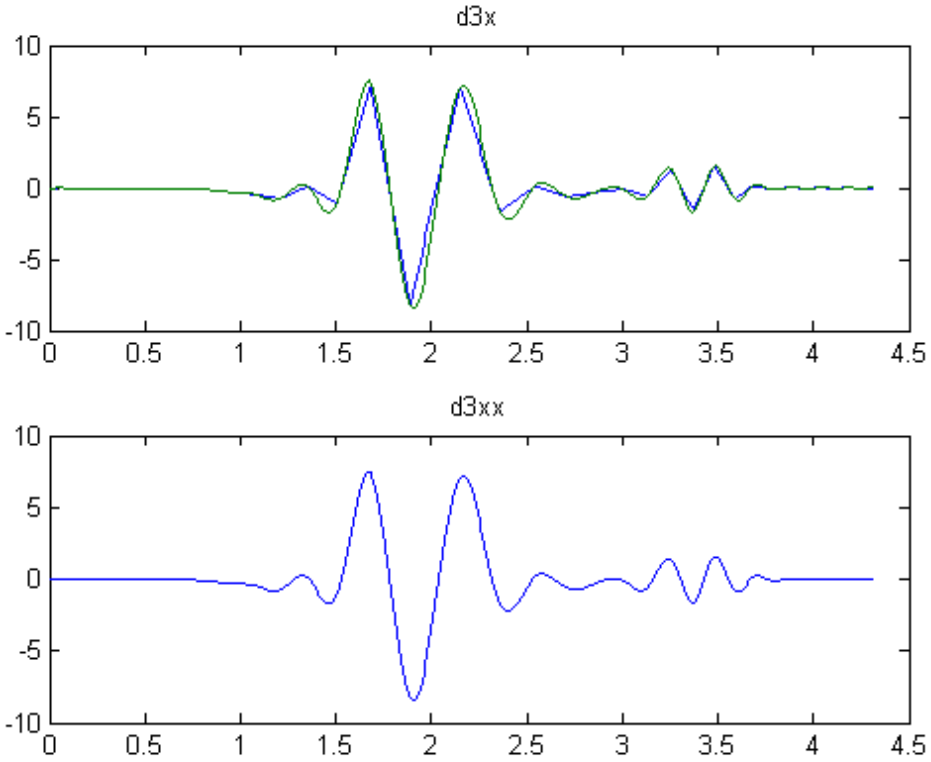

*3.6. ábra.* A pályamenti görbület és a pálya.



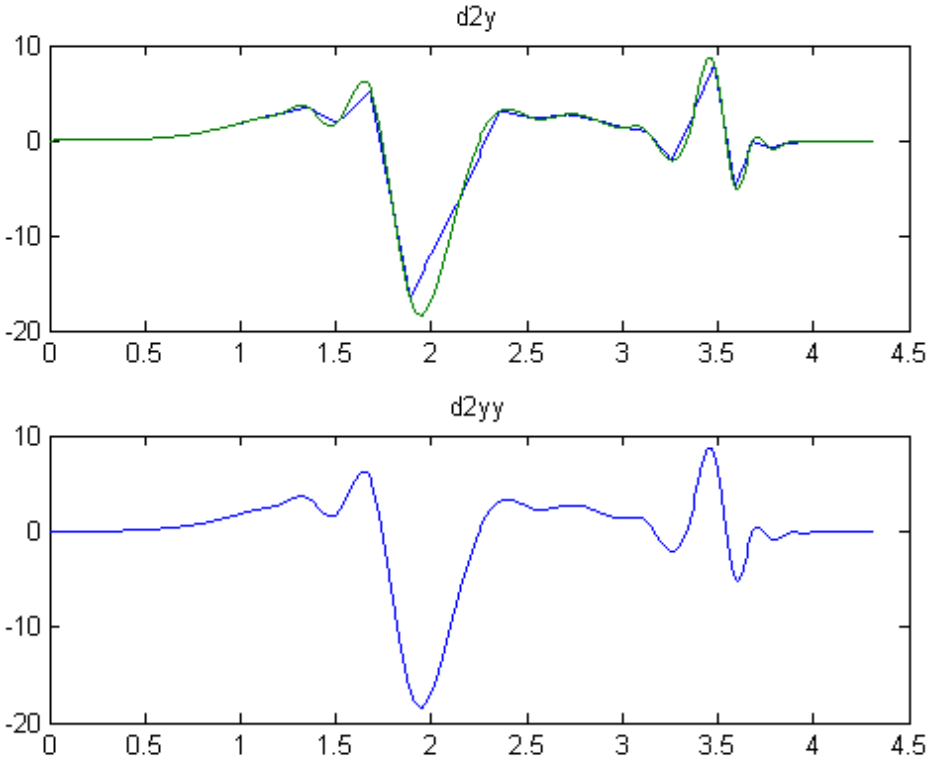

*3.7. ábra.* Az orientáció és deriváltjai.



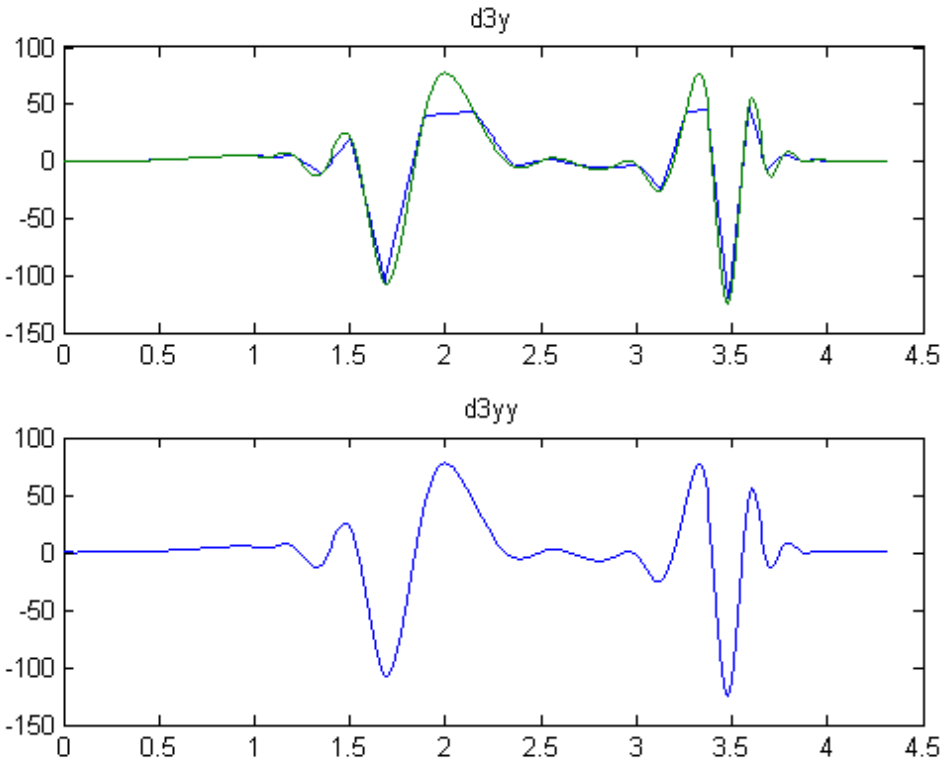

*3.8. ábra.* Az *x* jel második deriváltja.



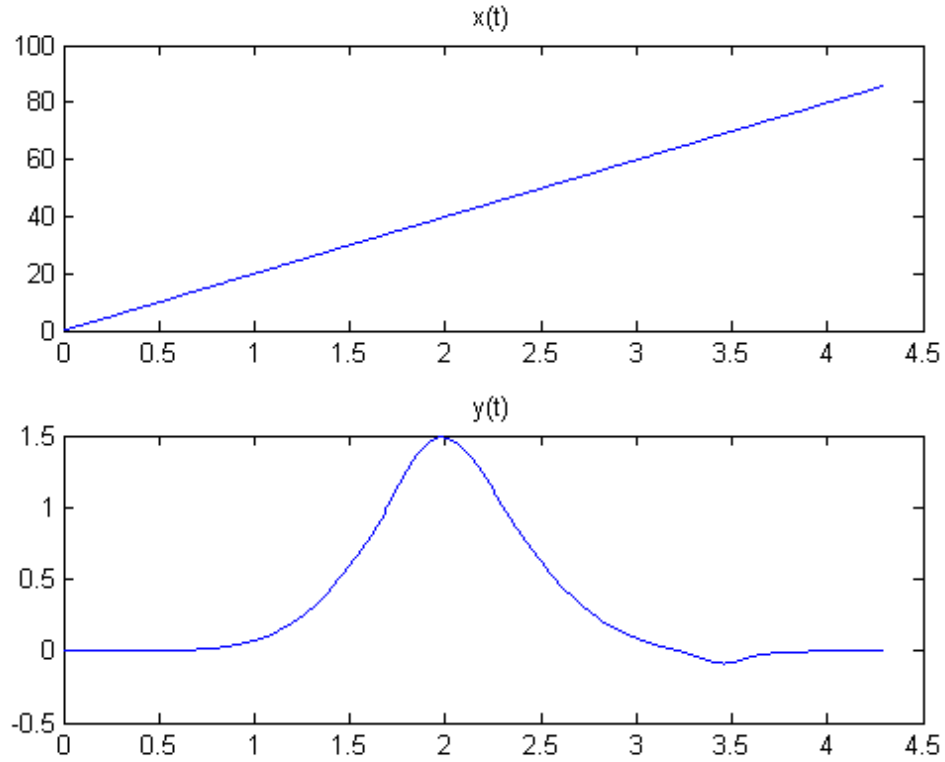

*3.9. ábra.* Az *x* jel harmadik deriváltja.




*3.10. ábra.* Az *y* jel második deriváltja.




*3.11. ábra.* Az *y* jel harmadik deriváltja.




*3.12. ábra.* Az irányítások *x(t)* és *y(t)* alapjelei.

A MATLAB standard `spline, ppval, unmkpp, mkpp` spline-technikájára épülő saját fejlesztésű fontosabb MATLAB függvények (függvény prototípus és 1 soros comment a help-hez) a következők:

```
function [vy,vd1y,vd2y,vd3y,spp]=deriv3(x,y,xx)
%3rd order derivative
```

```
function [s_kin,sppx,sppy]=kinematics(T)
%Find reference states and controls for CAS
```

A deriváltak segítségével a kinematikai mennyiségeket (a referencia mozgásban jogosan nulla oldalcsúszási szöget feltételezve) az alábbi összefüggések alapján számítjuk a `kinematics()` függvényben:

$$
\begin{aligned}
v &= (\dot{x}^2 + \dot{y}^2)^{1/2} \\
\dot{v} &= \frac{\dot{x}\ddot{x} + \dot{y}\ddot{y}}{(\dot{x}^2 + \dot{y}^2)^{1/2}} \\
\psi &= \arctan(\dot{y}/\dot{x}) \\
\dot{\psi} &= \frac{\ddot{y}\dot{x} - \dot{y}\ddot{x}}{\dot{x}^2 + \dot{y}^2} \\
\ddot{\psi} &= -2\frac{\dot{x}\ddot{x} + \dot{y}\ddot{y}}{(\dot{x}^2 + \dot{y}^2)^2} + \frac{\dddot{y}\dot{x} - \dot{y}\dddot{x}}{\dot{x}^2 + \dot{y}^2} \\
\kappa &= \frac{\dot{x}\ddot{y} - \dot{y}\ddot{x}}{\left(\dot{x}^2 + \dot{y}^2\right)^{3/2}}
\end{aligned}
\tag{3.10}
$$

A 3.8-3.11. ábrákon az *xx*-re illetve *yy*-ra végződő jelek már a későbbi szabályozásokkal kapott válaszjelek, amelyek alapján ellenőrizhető, mennyire tudta megközelíteni a differenciálgeometriai elvekre épülő szabályozás a referencia kinematikai mennyiségeket.

# 4. Jármű 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. A dinamikus modell alapjait a Lantos: "Algoritmusok gépjármű ütközésmentes pályatervezésére és pályakövetésére" c. tanulmányban már összefoglaltuk 2006. márciusában, itt csak a programfejlesztés során alkalmazott módosításokat foglaljuk össze.

A jármű dinamikus modelljében, felhasználva (Freund and Mayr, 1997) ötletét, 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:

$$S_h = c_R(-\beta - \frac{l_R\dot{\psi}}{v_G}), \quad S_v = c_F(\delta_w - \beta - \frac{l_F\dot{\psi}}{v_G}) \tag{4.1}$$

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.

Két nemlineáris modellt különböztetünk meg, az elsőben nem alkalmazzuk a Taylor-sorfejtést (precíz nemlineáris modell), a másodikban alkalmazzuk a Taylor-sorfejtést (approximált nemlineáris modell). Mindkét modell nemlineáris, de az approximált nemlineáris modellnek az az előnye, hogy differenciálalgebrai (Lie-algebrai) értelemben létezik a MIMO nemlineáris relatív fokszáma, ezért kimeneti visszacsatolással linearizálható. A $T(x)$ légellenálást az irányítások során nem vesszük figyelembe.

## 4.1 A pontos nemlineáris járműmodell

$$\begin{aligned}
\dot{\beta} &= -\dot{\psi} + \frac{1}{m_v v_G}\{F_{lF}\sin(\delta_W - \beta) - (F_{lR} - T)\sin(\beta) \\
&\quad + c_F(\delta_W - \beta - \frac{l_F\dot{\psi}}{v_G})\cos(\delta_W - \beta) + c_R(-\beta + \frac{l_R\dot{\psi}}{v_G})\cos(\beta)\} \\
\dot{\psi} &= \dot{\psi} \\
\ddot{\psi} &= \frac{1}{I_{zz}}\{l_F F_{lF}\sin(\delta_W) + l_F c_F(\delta_W - \beta - \frac{l_F\dot{\psi}}{v_G})\cos(\delta_W) - l_R c_R(-\beta + \frac{l_R\dot{\psi}}{v_G})\} \\
\dot{v}_G &= \frac{1}{m_v}\{F_{lF}\cos(\delta_W - \beta) + (F_{lR} - T)\cos(\beta) - c_F(\delta_W - \beta - \frac{l_F\dot{\psi}}{v_G})\sin(\delta_W - \beta) \\
&\quad + c_R(-\beta + \frac{l_R\dot{\psi}}{v_G})\sin(\beta)\} \\
\dot{X} &= v_G\cos(\psi + \beta) \\
\dot{Y} &= v_G\sin(\psi + \beta)
\end{aligned} \tag{4.2}$$

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=precfunxdot(t,x,F_lR,deltaw)
%Compute right side of vehicle dynamic model
```

## 4.2 Approximált nemlineáris járműmodell

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

$$\dot{x} = \begin{pmatrix} -x_3 + (Tx_1 + S_h)/(m_v x_4) \\ x_3 \\ -S_h l_R / I_{zz} \\ -T/m_v \\ x_4 \cos(x_1 + x_2) \\ x_4 \sin(x_1 + x_2) \end{pmatrix} + \begin{bmatrix} 1/(m_v x_4) & -x_1/(m_v x_4) \\ 0 & 0 \\ l_F / I_{zz} & 0 \\ 0 & 1/m_v \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{pmatrix} S_v \\ F_{lR} \end{pmatrix} = A(x) + B(x)u \tag{4.3}$$

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

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

```
function xdot=apprfunxdot(t,x,F_lR,deltaw)
%Compute right side of approximated vehicle dynamic model
```

## 4.3 Az approximált modell Lie-algebrai tulajdonságai

Az approximált nemlineáris dinamikus modellre elvégezve az

$$\begin{aligned} N_A^k C_i(x) &= \left[\frac{\partial}{\partial x} N_A^{k-1} C_i(x)\right] A(x), \quad N_A^0 C_i(x) = C_i(x) \\ d_i &= \min\left\{ j : \left[\frac{\partial}{\partial x} N_A^{k-1} C_i(x)\right] B(x) \neq 0^T,\ j = 1,2,\ldots,n \right\} \end{aligned} \tag{4.4}$$

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

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

$$S(x)=\begin{bmatrix} \dfrac{-S_{12}}{m_v} & \dfrac{S_{12}x_1+C_{12}}{m_v} \\ \dfrac{C_{12}}{m_v} & \dfrac{-C_{12}x_1+S_{12}}{m_v} \end{bmatrix}_{2\times 2} \quad \text{és} \quad \det(S(x)=-\frac{1}{m_v^2} \tag{4.5}$$

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

$$M^*(x)=\begin{bmatrix} \alpha_{01}N_A^0C_1(x)+\alpha_{11}N_A^1C_1(x) \\ \alpha_{02}N_A^0C_2(x)+\alpha_{12}N_A^1C_2(x) \end{bmatrix}=\begin{bmatrix} \alpha_{01}x_5+\alpha_{11}x_4C_{12} \\ \alpha_{02}x_6+\alpha_{12}x_4S_{12} \end{bmatrix}_{2\times 1}$$

Ezért a nemlineáris rendszerek Lie-algebrán alapuló eredményei szerint választható

$$u=S^{-1}(x)\left\{-C^*(x)+\Lambda w-M^*(x)\right\} \tag{4.6}$$

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

$$\begin{aligned} \bar{y}_1&=\lambda_1w_1-\alpha_{01}x_5-\alpha_{11}x_4C_{12} \\ \bar{y}_2&=\lambda_2w_2-\alpha_{02}x_6-\alpha_{12}x_4S_{12} \end{aligned} \tag{4.7}$$

akkor átalakítások után kapjuk, hogy a zárt rendszerre teljesül:

$$\begin{aligned} u_1&=-S_h+[(C_{12}x_1-S_{12})\bar{y}_1+(S_{12}x_1+C_{12})\bar{y}_2]m_v \\ u_2&=T+[C_{12}\bar{y}_1+S_{12}\bar{y}_2]m_v \\ \dot{y}_1&=x_4C_{12} \\ \dot{y}_2&=x_4S_{12} \\ \ddot{y}_1&=\dot{x}_4C_{12}-x_4S_{12}(\dot{x}_1+\dot{x}_2) \\ \dot{x}_1+\dot{x}_2&=(-S_{12}\bar{y}_1+C_{12}\bar{y}_2)/x_4 \\ \dot{x}_4&=C_{12}\bar{y}_1+S_{12}\bar{y}_2 \\ \ddot{y}_1&=\bar{y}_1=\lambda_1w_1-\alpha_{01}x_5-\alpha_{11}x_4C_{12}=\lambda_1w_1-\alpha_{01}y_1-\alpha_{11}\dot{y}_1 \\ \ddot{y}_2&=\bar{y}_2=\lambda_2w_2-\alpha_{02}x_6-\alpha_{12}x_4S_{12}=\lambda_2w_2-\alpha_{02}y_2-\alpha_{12}\dot{y}_2 \end{aligned} \tag{4.8}$$

## 4.4 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 $s^2+2\sqrt{\lambda}s+\lambda=0.$

$$\ddot{y}_i+\alpha_{1i}\dot{y}_i+\alpha_{0i}y_i=\lambda_iw_i \tag{4.9}$$

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

$$\begin{aligned} &\ddot{y}_i+\alpha_{1i}\dot{y}_i+\lambda y_i=\lambda[w_{ia}+\frac{1}{\lambda}(\alpha_{1i}\dot{w}_{ia}+\ddot{w}_{ia})]\Rightarrow \\ &(\ddot{w}_{ia}-\ddot{y}_i)+\alpha_{1i}(\dot{w}_{ia}-\dot{y}_i)+\lambda(w_{ia}-y_i)=0 \end{aligned} \tag{4.10}$$

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 $\delta_w$ kormányzási szög meghatározható:

$$\delta_w = \frac{S_v}{c_F} + \beta + \frac{l_F \dot{\psi}}{v_G} \tag{4.11}$$

A differenciálgeometriai elvekre épülő szabályozó kimenetét számító függvény prototípus alakja és egysoros commentje a helphez a következő:

```
function [xdot,F_lR,deltaw,Sv]=dg_controller(t,x)
%Compute control for 'diffgeom'
```

# 5. 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.

## 5. 1 Választható alternatívák

Két lehetőség kínálkozik: 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.

## 5.2 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:

$$J = \frac{1}{2}\sum_{i=1}^{N-1} \left\| e_i - \delta y_i \right\|^2 + \frac{1}{2}\lambda \sum_{i=0}^{N-1} \left\| \delta u_i \right\|^2 \tag{5.1}$$

Az állapot és kimenő jel perturbációk a következőképp számíthatók:

$$\begin{pmatrix} \delta x_1 \\ \delta x_2 \\ \vdots \\ \delta x_{N-1} \\ \hline \delta x_N \end{pmatrix} = \left[\begin{array}{c} A_0 \\ A_1 A_0 \\ \vdots \\ A_{N-2}\cdots A_1 A_0 \\ \hline A_{N-1}\cdots A_1 A_0 \end{array}\right] \delta x_0 + \left[\begin{array}{ccccc} B_0 & 0 & \cdots & 0 & 0 \\ A_1 B_0 & B_1 & \cdots & 0 & 0 \\ \vdots & \cdots & \ddots & 0 & 0 \\ A_{N-2}\cdots A_1 B_0 & A_{N-2}\cdots A_2 B_1 & \cdots & B_{N-2} & 0 \\ \hline A_{N-1}\cdots A_1 B_0 & A_{N-1}\cdots A_2 B_1 & \cdots & A_{N-1}B_{N-2} & B_{N-1} \end{array}\right] \begin{pmatrix} \delta u_0 \\ \delta u_1 \\ \vdots \\ \delta u_{N-2} \\ \hline \delta u_{N-1} \end{pmatrix} \tag{5.2}$$

$$\begin{pmatrix} \delta y_1 \\ \delta y_2 \\ \vdots \\ \delta y_{N-1} \\ \hline \delta y_N \end{pmatrix} = \left[\begin{array}{c} CA_0 \\ CA_1 A_0 \\ \vdots \\ CA_{N-2}\cdots A_1 A_0 \\ \hline CA_{N-1}\cdots A_1 A_0 \end{array}\right] \delta x_0 + \left[\begin{array}{ccccc} CB_0 & 0 & \cdots & 0 & 0 \\ CA_1 B_0 & CB_1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & 0 & 0 \\ CA_{N-2}\cdots A_1 B_0 & \cdots & \cdots & CB_{N-2} & 0 \\ \hline CA_{N-1}\cdots A_1 B_0 & \cdots & \cdots & CA_{N-1}B_{N-2} & CB_{N-1} \end{array}\right] \begin{pmatrix} \delta u_0 \\ \delta u_1 \\ \vdots \\ \delta u_{N-2} \\ \hline \delta u_{N-1} \end{pmatrix} \tag{5.3}$$

vagy tömör alakban

$$\begin{pmatrix} \delta y_1 \\ \delta y_2 \\ \vdots \\ \delta y_{N-1} \end{pmatrix} = P_1 \delta x_0 + H_1 \delta U$$
$$\delta y_N = P_2 \delta x_0 + H_2 \delta U \tag{5.4}$$

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:

$$J = \frac{1}{2}\sum_{i=1}^{N-1} \left\| e_i - \delta y_i \right\|^2 + \frac{1}{2}\lambda \sum_{i=0}^{N-1} \left\| \delta u_i \right\|^2 \to \min$$
$$e_N - \delta y_N = e_N - (P_2 \delta x_0 + H_2 \delta U) = 0 \tag{5.5}$$

A (5.4) jelölésekkel a költségfüggvény részletes alakja:

$$
\begin{aligned}
J = \frac{1}{2}\sum_{i-1}^{N-1} < e_i, e_i > - < (\sum_{i=1}^{N-1} p_i e_i), \delta x_0 > \\
- < (\sum_{i=1}^{N-1} h_i e_i), \delta U > + \frac{1}{2} < (\sum_{i=1}^{N-1} p_i p_i^T)\delta x_0, \delta x_0 > \\
+ < (\sum_{i=1}^{N-1} h_i p_i^T)\delta x_0, \delta U > \\
+ \frac{1}{2} < (\sum_{i=1}^{N-1} h_i h_i^T)\delta U, \delta U > + \frac{1}{2}\lambda < \delta U, \delta U > .
\end{aligned}
\tag{5.6}
$$

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

$$
\begin{aligned}
L &= J + < \mu, e_N > - < P_2^T \mu, \delta x_0 > - < H_2^T \mu, \delta U > \\
0 &= \frac{dL}{d\delta U} = -\sum_{i=1}^{N-1} h_i e_i + (\sum_{i=1}^{N-1} h_i p_i^T)\delta x_0 \\
&+ (\sum_{i=1}^{N-1} h_i h_i^T)\delta U + \lambda \delta U - H_2^T \mu \\
&= -\overline{m} + H_1^T P_1 \delta x_0 + (H_1^T H_1 + \lambda I)\delta U - H_2^T \mu
\end{aligned}
\tag{5.7}
$$

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

$$
L_1 := H_1^T H_1 + \lambda I, \;\; L_\mu := H_2 L_1^{-1} H_2^T \tag{5.8}
$$

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

$$
\begin{aligned}
\delta U = L_1^{-1}\{ & H_2^T L_\mu^{-1} e_N + (I - H_2^T L_\mu^{-1} H_2 L_1^{-1})\overline{m} \\
& - [H_1^T P_1 + H_2^T L_\mu^{-1}(P_2 - H_2 L_1^{-1} H_1^T P_1]\delta x_0 \}
\end{aligned}
\tag{5.9}
$$

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 $\delta U$ sorozat első eleme.

## 5.3 A megvalósított prediktív irányítási algoritmus:

Minden horizontban a következő lépések ismétlődnek:

*1. lépés.* 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 $\beta$ 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 $\delta U$ 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 $\delta r_i$ változását kell optimalizálni. Bevezetve a

$$A_i := \begin{bmatrix} A_i & B_i \\ 0 & I \end{bmatrix} \text{ és } B_i := \begin{bmatrix} B_i \\ I \end{bmatrix} \tag{5.10}$$

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

# 6. A szabályozásokat megvalósító keretprogram

A szabályozásokat megvalósító keretprogram különböző szabályozások szimulációs vizsgálatára alkalmas, amelyek közül a differenciálgeometriai elvű irányítás és a nemlineáris prediktív irányítás került kifejlesztésre. A keretprogram azonban lehetővé teszi más irányítások későbbi beillesztését is. Jelen tanulmányban a prediktív irányítási elvű szabályozások eredményeit részletezzük. A szoftver megvalósításakor kerüljük a Simulink és a MATLAB toolboxok használatát annak érdekében, hogy a későbbi C-nyelvű megvalósításkor alkalmazhatók legyenek a MATLAB Compiler szolgáltatásai. Ehhez tudni kell, hogy a MATLAB egy interpretált nyelv, amelynek használatakor a MATLAB, Simulink, Java interpreterek futnak, és ezek más futtató környezetben már nem állnak rendelkezésre, ennek következtében a MATLAB Compiler szolgáltatásai erősen korlátozottak.

## 6.1 Modell konverzió folytonos időről diszkrét időre

A szabályozók mintavételes (DDC) szabályozók. Minden mintavételi pillanatban kiszámításra kerül a szabályozó új kimenete, amely rákerül a szakasz (a pontos vagy approximált nemlineáris modell) bemenetére, és meghatározásra kerül az állapotváltozók új értéke az állapotegyenlet megoldásával. A folytonosidejű állapotegyenletet Euler-módszerrel diszkrét idejűre konvertáljuk:

$$\dot{x} = f_c(x,u) \Rightarrow x_{i+1} = x_i + Tf_c(x_i,u_i) =: f_d(x_i,u_i) \tag{4.1}$$

ahol $T$ a mintavételi idő, és ennek alapján számítjuk az új $x_{i+1}$ állapotot. Egy teljes szabályozási tranziens megvalósítását végző saját fejlesztésű függvény `ddc_euler()`. Ezt a függvényt a MATLAB `ode45()` függvényére cserélve irányítás közben ellenőrizhető az Euler-módszer pontossága a választott $T$ mintavételi idő esetén, ha a kétféle megoldás eredményeit grafikusan összehasonlítjuk. Az összehasonlítást elvégeztük, és megállapítottuk, hogy a $T = 0.01\,\text{sec}$ választás megfelelő pontosságot biztosít.

## 6.2 A szabályozásokat megvalósító frameCAS keretprogram

A szabályozásokat megvalósító saját fejlesztésű `frameCAS` keretprogram (MATLAB script) az eddigiek alapján a következő lépésekből áll:

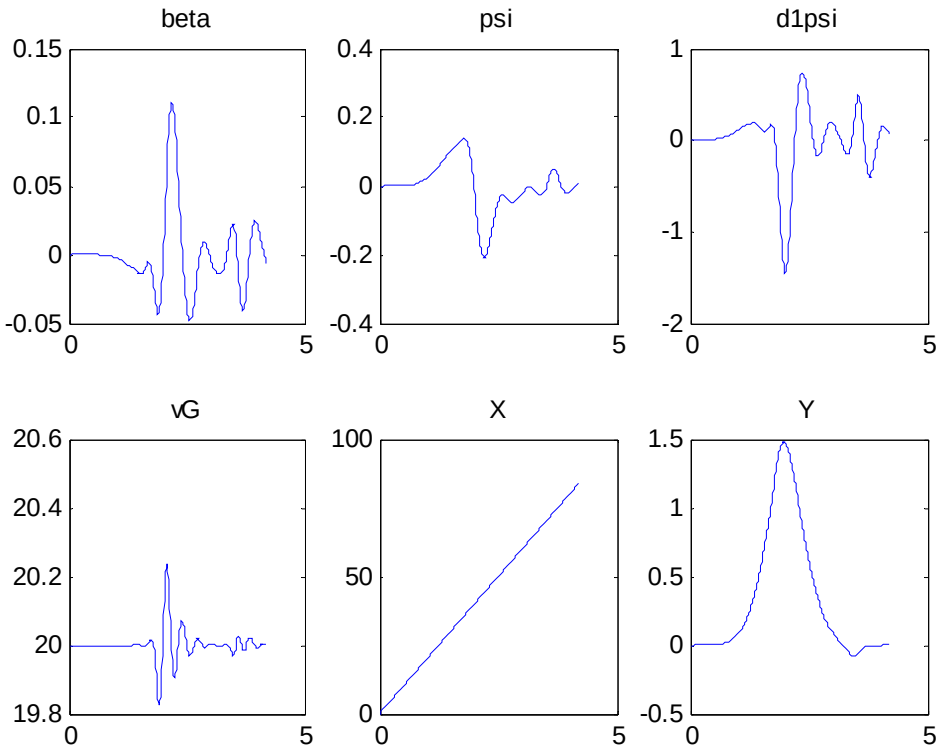
1. `initband()`, amely beágyazva tartalmazza a banditer() függvényt is
2. `kinematics()`, amely előállítja a szabályozások referencia jeleit
3. `ddc_euler()`, amely meghatározza a szabályozási rendszer tranzienseit
4. A szabályozási tranziensek grafikus megjelenítése

A keretprogramot és a meghívott függvényeket globális program switch-ek vezérlik:

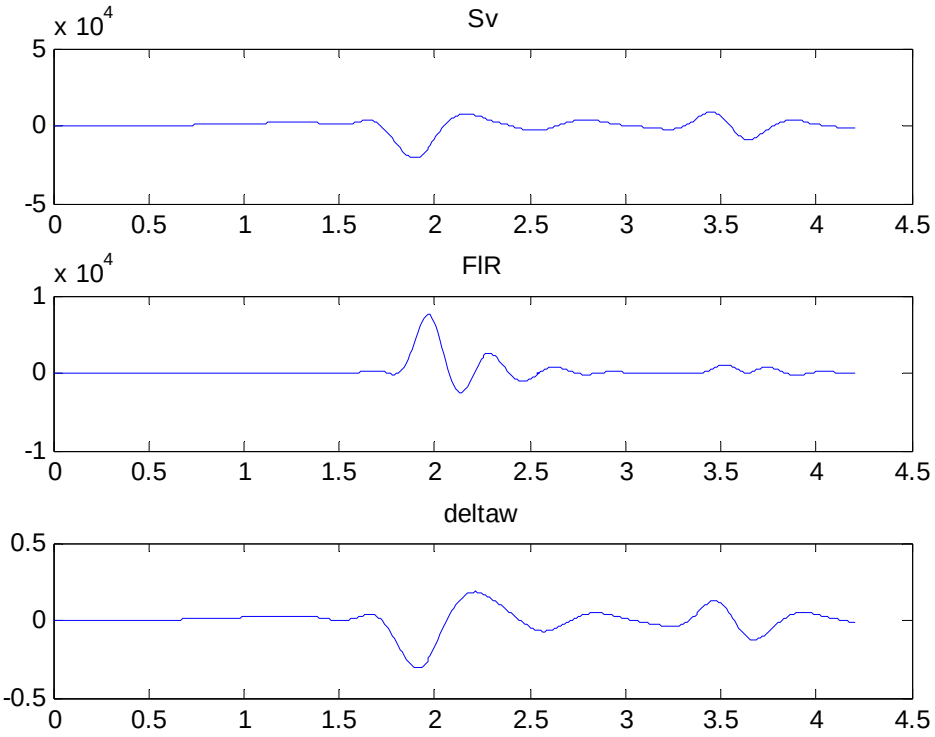
a) `sys_appr`: arról dönt, hogy a tranziensek számítása az approximált nemlineáris vagy a pontos nemlineáris modell alapján történik-e (a szabályozó kimenetének számításakor mindig az approximált modellt használjuk)
b) `sys_estim`: arról dönt, hogy a szabályozó kimenetének számításakor az állapotváltozókat, vagy azok állapotbecsléssel számított értékét használjuk-e
c) `sys_contr`: arról dönt, mi az érvényes szabályozó, például a differenciálgeometriai (Lie-algebrai) elvekre épülő szabályozó esetén értéke a `'diffgeom'` string, nemlineáris prediktív irányítás esetén `'nonlinpred'` string.
d) `int_horizon`: arról dönt, van-e integrátor a prediktív irányítás szerinti szabályozóban.

Ezek a programkapcsolók különféle kombinációkra adnak lehetőséget.

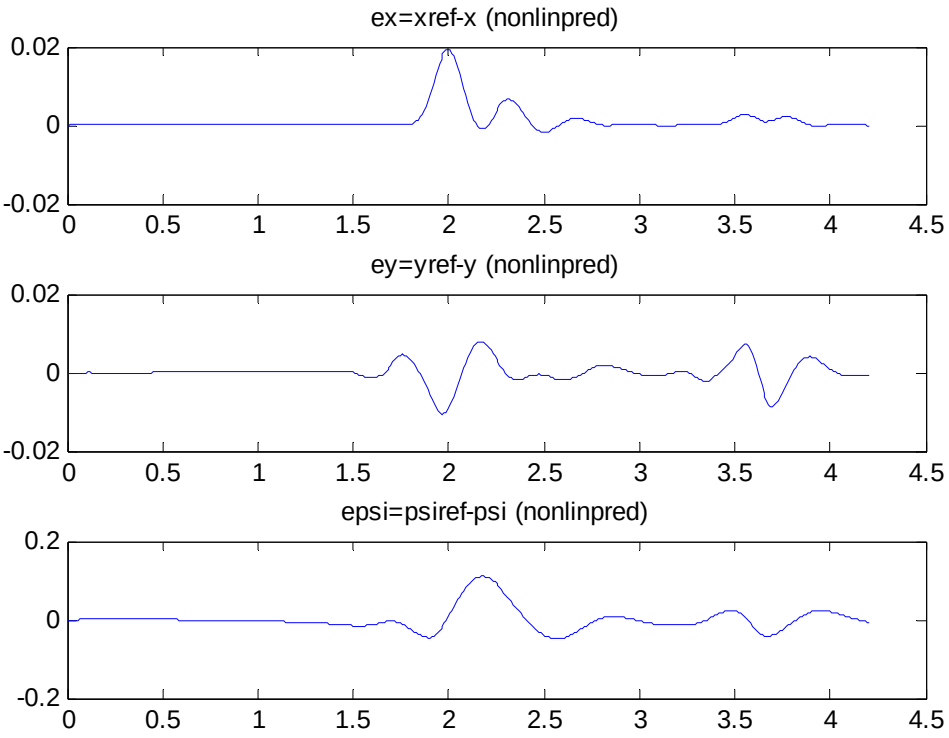
A programkapcsolók értékeit például `frameCAS`-ben `sys_appr=0, sys_estim=0, sys_contr='nonlinpred', sys_int=1` értékűre választva `ddc_euler()`-rel a 6.1-6.3. ábrákon látható tranzienseket kaptuk.




*6.1. ábra.* Állapotváltozók sys_appr=0, sys_estim=0, sys_contr=’nonlinpred’ esetén




*6.2. ábra.* Beavatkozó jelek sys_appr=0, sys_estim=0, sys_contr=’nonlinpred’ esetén




*6.3. ábra.* Hibajelek sys_appr=0, sys_estim=0, sys_contr='nonlinpred' esetén

Vizsgálható a predikcióba bevont nemlineáris modell hatása is a beavatkozó jel alakjára és nagyságára. Ugyan a szabályozó kimenete mindig az approximált nemlineáris modell alapján kerül meghatározásra, de `sys_appr=0` esetén a pontos nemlineáris modell, `sys_appr=1` esetén viszont az approximált nemlineáris modell alapján alakul a szabályozási rendszer állapotváltozóinak tranziense, ezért mások lesznek a beavatkozó jelek is a két estben, mivel a szabályozó kimenete az állapotváltozók függvénye. Ez figyelemreméltó hatást fejt ki az $F_{lR}$ beavatkozó jel nagyságrendjére (pontos rendszer esetén $F_{lR}$ jelentősen nagyobb).

# 7. Összefoglalás

A Matlab licenszhez kötött fejlesztési környezet rendelkezik olyan fordítóval, amely jelentős megszorításokkal ugyan, de lehetővé teszi a költséges Matlab fejlesztési környezet elhagyását és Matlab licenszet nem igénylő stand-alone futtatható programok létrehozását. Ennek során figyelembe kellett venni, hogy a Matlab interpretált nyelv, és a stand-alone programnak az interpreterek hiányában is korrektül kell működnie. A kutatás során először meghatároztuk a rendelkezésre álló két legfejletteb termék, a Matlab 6.5 (R13) Compiler 3 és a Matlab R2006a (R14) Compiler 4 korlátait stand-alone programok létrehozása során.

Megállapítottuk, hogy a Matlab 6.5 Compiler Version 3 C/C++ nyelvre tudja a Matlab függvényeket lefordítani, amely azután stand-alone programmá összeszerkeszthető, de a Matlab toolboxok nem, vagy csak lényeges megszorításokkal használhatók. Mivel a perspektívikus célok előnyben részesítik a C/C++ nyelvet a továbbfejlesztések során, ezért részletesen analizáltuk a fordítót a korlátok tekintetében. A vizsgálatok eredményeképpen szükség volt a feladat szempontjából kulcsfontosságú Optimization Toolbox lecserélésére egy saját fejlesztésű változatra a korlátok felszámolása érdekében.

A szoftver technológiailag fejlettebb Matlab R2006a Compiler 4 nem C/C++ nyelvre, hanem a felhasználó számára nem definiált és üzleti okokból kriptografikus kulcsokkal védett közbenső CTF formátumra fordít, amely függ a futtató rendszertől, és amelyet azután a stand-alone program az indításkor automatikusan értelmez és végrehajt. A Matlab Optimization Toolbox szolgáltatásai lefordírhatók, a korlátok az alkalmazás szempontjából nem kritikusak, a probléma a C/C++ kimenet hiánya.

Az így szerzett tapasztalatokra alapozva lehetőség nyílt stand-alone program kifejlesztésére az automatikus akadályelkerülő pálya meghatározására és irányítással történő megvalósítására. Ennek során figyelembe vettük a Matlab Compiler megismert korlátait, és a szükséges mértékben átdolgozzuk a korábbi programverziót a stand-alone korlátok figyelembevételével. 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) alkotják, valamint a mozgó akadály és a saját jármű sebessége. További bemeneti adatok azonosítják az útszakaszt (bal oldali és jobb oldali sávszélesség). Az akadályelkerülő pályát az elasztikus szalag elvére épülve határozzuk meg, amely nagyméretű nemlineáris egyenletrendszerre vezet, amelyet az Oprimization Toolbox *fsolve* függvényével oldunk meg.

Az akadályelkerülő pályát differenciálgeometriai elvű és prediktív irányítási (mozgó horizontú, RHC) módszerekkel valósítjuk meg. Minden horizont kezdetén a rendszer 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 kifejlesztendő stand-alone program első verziója mérhető állapotokat (sebesség, oldalcsúszási szög, orientáció és deriváltja, X és Y pozíció) feltételez. Ugyan az összes állapot csak ritkán mérhető közvetlenül, de így lehetőség nyílt az irányítás alaposabb önálló tesztelésére.

A fejlesztés erdményeként a Matlab Compiler 3 felhasználásával 2 stand-alone verzió került kifejlesztésre, egy a saját, egy másik pedig az eredeti Optimization Tollbox felhasználásával (az utóbbi fordításakor számos warning utalt az eredeti toolbox hiányosságaira, amely komoly kockázatot jelent valós idejű körülmények között). Egy harmadik verzió a Matlab Compiler 4 felhasználásával készült. Mindhárom verzió telepítve lett a Matlabot nem tartalmazó futtató környezetben, és helyes müködésük bemutatásra került a BME Közlekedésautomatika Tanszéken a RET 1.1 projekt keretében.

A fejlesztés további iránya kétantennás GPS, valamint 3D gyorsulásérzékelők és giroszkópok adataira épülő állapotbecslés beillesztése lehet a stand-alone programmokba.

# 8. Felhasznált irodalom


1. Brandt, T., & Sattel, T. (2005). Path planning for automotive collision avoidance based on elastic bands. *Proceedings of the IFAC World Congress, Prague,* Paper TU-M16-TO/4 02746.pdf.
2. Freund, E., & Mayr, R. (1997) *Nonlinear Path Control in Automated Vehicle Guidance*. IEEE Transactions on Robotics and Automation, Vol. 13, pp. 49-60.
3. Lantos, B. (2006). *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.

# 9. MATLAB programlisták

## 9.1 ab2ph.m

```
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
% AA=zeros(N*n,n);
% BB=zeros(N*n,r);
% ee=zeros(N,m);

% %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);

if inthor~=0
    AAA=zeros(N*(n+r),n+r);
    BBB=zeros(N*(n+r),r);
    CCC=[C zeros(m,r)];
    for i=0:N-1
        rai=i*n;
        rbi=i*n;
        raai=i*(n+r);
        rbbi=i*(n+r);
        Ai=AA(rai+1:rai+n,:);
        Bi=BB(rbi+1:rbi+n,:);
        AAi=[Ai Bi; zeros(r,n) eye(r)];
        BBi=[Bi; eye(r)];
        AAA(raai+1:raai+n+r,:)=AAi;
        BBB(rbbi+1:rbbi+n+r,:)=BBi;
    end;
    AA=AAA;
    BB=BBB;
    C=CCC;
    n=size(AA,2);
    r=size(BB,2);
    N=size(AA,1)/n;
    m=size(C,1);
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;

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;

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;

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;

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
```

## 9.2 apprfunxdot.m

```
function [xdot,F_lR,deltaw,Sv,f,G]=apprfunxdot(t,x,F_lR,deltaw)
%Compute right side of approximated vehicle dynamic model

global N_band N_rigobs N_moveobs R_0 R_band F_band J_band
global K_band L_band k_Bl k_Br s_Bl s_Br B_wide
global  K_rigobs  S_rigobs  R_rigobs  D_rigobs  k_moveobs  s_moveobs
r_moveobs d_moveobs v_moveobs
global v_own
global Vers_rigobs Vers_moveobs

%frame global
global s_kin T_samp lambda spp_x spp_y
global c_F l_F c_R l_R m_v I_zz
global sys_appr
global sys_estim
global sys_contr

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;
```

## 9.3 avoidrigobs.m

```
function avoidrigobs
%Initiate elastic band avoiding obstacles
global N_band N_rigobs N_moveobs R_0 R_band F_band J_band
global K_band L_band k_Bl k_Br s_Bl s_Br B_wide
global  K_rigobs  S_rigobs  R_rigobs  D_rigobs  k_moveobs  s_moveobs
r_moveobs d_moveobs v_moveobs
global v_own
global Vers_rigobs Vers_moveobs

M=size(R_rigobs,1);
R_band=[R_band(:,1) zeros(N_band,1)];
if M==0, return; end;

%M=1
scale=2;
xbmax=max(R_band(:,1));
xo=R_rigobs(1,1);
yo=R_rigobs(1,2);
do=D_rigobs(1);
x1=xo-scale*do/2;
y1=yo+scale*do/2;
x2=xo+scale*do/2;
y2=yo+scale*do/2;
if 2*xo>xbmax
    R_band=2*xo/xbmax*R_band;
end;
ix1=find(R_band(:,1)<=x1);
ileft=max(ix1);
imidleft=floor((1+ileft)/2);
xmidleft=R_band(imidleft,1);
ix2=find(R_band(:,1)>=x2);
iright=min(ix2);
imidright=ceil((iright+N_band)/2);
xmidright=R_band(imidright,1);

for i=1:N_band
    if i<=imidleft
        ;
    elseif (i>imidleft & i<=ileft)
        xi=R_band(i,1);
        yi=y1/(x1-xmidleft)*(xi-xmidleft);
        R_band(i,:)=[xi yi];
    elseif (i>ileft & i<=iright)
        xi=R_band(i,1);
        yi=y1;
        R_band(i,:)=[xi yi];
    elseif (i>iright & i<=imidright)
        xi=R_band(i,1);
        yi=y1/(x2-xmidright)*(xi-xmidright);
        R_band(i,:)=[xi yi];
    else
        ;
    end;
end;
rinx=R_band';
rin=rinx(:);
plotband(rin,[],R_rigobs,D_rigobs,r_moveobs,d_moveobs,v_moveobs,R_0,v
_own);
```

```
pause
```

## 9.4 banditer.m

```
function
[fout,Jout,F_int,F_Bl,F_Br,F_rigobs,F_moveobs,J_int,J_Bl,J_Br,...
    J_rigobs]=banditer(rin)
%Compute forces, Jacobian and new nodes for elastic band
global N_band N_rigobs N_moveobs R_0 R_band F_band J_band
global K_band L_band k_Bl k_Br s_Bl s_Br B_wide
global  K_rigobs  S_rigobs  R_rigobs  D_rigobs  k_moveobs  s_moveobs
r_moveobs d_moveobs v_moveobs
global v_own
global Vers_rigobs Vers_moveobs

eps=0;

oldR_band=R_band;
R_bandx=NaN*ones(2,N_band);
R_bandx(:)=rin;
R_band=R_bandx';

%R_band=[r_1; r_2; ...; r_N] where ri=[xi yi]
%K_band=[k_0 k_1 ... k_N-1]'
%L_band=[l0_0 l0_1 ...; l0_N-1]'
%b_wide=[b 0.75 0.25]
%F_band=[f_1; f_2; ...; f_N] where f_i=[f_ix f_iy]
%

F_int=zeros(N_band,2);
F_Bl=zeros(N_band,2);
F_Br=zeros(N_band,2);
F_rigobs=zeros(N_band,2);
F_moveobs=zeros(N_band,2);

Rext=[R_0; R_band];
Next=size(Rext,1);

%Compute F_int
dR=Rext(2:Next,:)-Rext(1:Next-1,:);
dRabs=sqrt(sum((dR.*dR)'))';
dRexp=K_band.*(dRabs-L_band);
dRexpnorm=dRexp./dRabs;
dRexpnorm2=[dRexpnorm dRexpnorm];
F_int=[dRexpnorm2(2:Next-1,:).*dR(2:Next-1,:); 0 0]-dRexpnorm2.*dR;

%-------------------------------------------------------------------
------
%-------------------------------------------------------------------
%Compute F_Bq
b=B_wide(1);
bleft=B_wide(2);
bright=B_wide(3);
%
rBl=b*bleft;
dRBl=R_band(:,2)-rBl;
dRBlabs=abs(dRBl);
dRBlabs2=dRBlabs.^2;
kexpBl=k_Bl*exp(-0.5*(dRBlabs/s_Bl).^2);
dRBlnorm=kexpBl./dRBlabs;
dRBlnorm2=[dRBlnorm dRBlnorm];
F_Bl=dRBlnorm2.*[zeros(N_band,1) dRBl];
```

```
%
rBr=-b*bright;
dRBr=R_band(:,2)-rBr;
dRBrabs=abs(dRBr);
dRBrabs2=dRBrabs.^2;
kexpBr=k_Br*exp(-0.5*(dRBrabs/s_Br).^2);
dRBrnorm=kexpBr./dRBrabs;
dRBrnorm2=[dRBrnorm dRBrnorm];
F_Br=dRBrnorm2.*[zeros(N_band,1) dRBr];

%Compute F_rigobs
M=size(R_rigobs,1);
F_rigobs=zeros(N_band,2);
for j=1:M
    switch Vers_rigobs
        case 1
            %Version 1
            robsj=R_rigobs(j,:);
            Robsj=repmat(robsj,N_band,1);
            dRobsj=R_band-Robsj;
            dRobsjabs=sqrt(sum((dRobsj.*dRobsj)'))';
            dRobsjexp=K_rigobs(j)*exp(-
0.5*((dRobsjabs./S_rigobs(j)).^2));
            dRobsjnorm=dRobsjexp./dRobsjabs;
            dRobsjnorm2=[dRobsjnorm dRobsjnorm];
            F_rigobs=F_rigobs+dRobsjnorm2.*dRobsj;
        case 2
            %Version 2
            robsj=R_rigobs(j,:);
            Robsj=repmat(robsj,N_band,1);
            dRobsj=R_band-Robsj;
            dRobsjabs=sqrt(sum((dRobsj.*dRobsj)'))'+eps;
            dRobsjhyp=(K_rigobs(j)*D_rigobs(j)/2)./dRobsjabs;
            dRobsjnorm=dRobsjhyp./dRobsjabs;
            dRobsjnorm2=[dRobsjnorm dRobsjnorm];
            F_rigobs=F_rigobs+dRobsjnorm2.*dRobsj;
        otherwise
            ;
    end;
end;

%Compute F_moveobs
%----------------
if N_moveobs>0
    % Rext=[R_0; R_band];
    % Next=size(Rext,1);
    % dR=Rext(2:Next,:)-Rext(1:Next-1,:);
    % dRabs=sqrt(sum((dR.*dR)'))';
    switch Vers_moveobs
        case 1
            %Version 1
            t_own=dRabs./v_own;
            t_own=cumsum(t_own);
            rt_moveobs=repmat(r_moveobs,N_band,1)+[-v_moveobs*t_own
zeros(N_band,1)];
            dRm=R_band-rt_moveobs;
            dRmabs=sqrt(sum((dRm.*dRm)'))';
            dRmabsexp=k_moveobs*exp(-0.5*((dRmabs/s_moveobs).^2));
            dRmabsnorm2=[dRmabsexp dRmabsexp];
            F_moveobs=dRmabsnorm2.*dRm;
        case 2
```

```
            %Version 2
            t_own=dRabs./v_own;
            t_own=cumsum(t_own);
            rt_moveobs=repmat(r_moveobs,N_band,1)+[-v_moveobs*t_own
zeros(N_band,1)];
            dRm=R_band-rt_moveobs;
            dRmabs=sqrt(sum((dRm.*dRm)'))';
            dRmabshyp=(k_moveobs*d_moveobs/2)./(dRmabs.^2);
            dRmabsnorm2=[dRmabshyp dRmabshyp];
            F_moveobs=dRmabsnorm2.*dRm;
        otherwise
            ;
    end;
end;
%---------------

%-----------------------------------------------------------------------

%-----------------------------------------------------------------------
%Compute F_total
F_total=F_int+F_Bl+F_Br+F_rigobs+F_moveobs;
%-----------------------------------------------------------------------

% F_int
% F_Bl
% F_Br
% F_total
% pause

%-----------------------------------------------------------------------
%Cumpute Jacobian
Jmat=zeros(2*N_band,2*N_band);
J_int=zeros(2*N_band,2*N_band);
J_Bl=zeros(2*N_band,2*N_band);
J_Br=zeros(2*N_band,2*N_band);
J_rigobs=zeros(2*N_band,2*N_band);
J_moveobs=zeros(2*N_band,2*N_band);
%
%Preparation F_int
KL=K_band.*L_band;
dRabs2=dRabs.^2;
dRabs3=dRabs.^3;
dX=dR(:,1);
dY=dR(:,2);
dX2=dX.*dX;
dXY=dX.*dY;
dY2=dY.*dY;
WX2=K_band-KL.*(1./dRabs-dX2./dRabs3);
WXY=KL.*dXY./dRabs3;
WY2=K_band-KL.*(1./dRabs-dY2./dRabs3);
%
%Preparation F_Bq
% xi-xi_Bq=0
% rBq=b*bq;
% dRBq=R_band(:,2)-rBq;
% dRBqabs=abs(dRBq);
% dRBqabs2=dRBqabs.^2;
% kexpBq=k_Bq*exp(-0.5*(dRBqabs/s_Bq).^2);
%
% WEBlYm1=kexpBl.*(1./dRBlabs);
% WEBlY2=kexpBl.*dRBlabs2*(1+1/(s_Bl^2));
```

```
% WEBrYm1=kexpBr.*(1./dRBrabs);
% WEBrY2=kexpBr.*dRBrabs2*(1+1/(s_Br^2));
WEBlY1=kexpBl.*dRBlabs/(s_Bl^2);
WEBrY1=kexpBr.*dRBrabs/(s_Br^2);
%
%-----------------------------------------------------------------
%Compute effects of F_int and F_Bq
for i=1:N_band
    i0=2*(i-1)+1;
    j0=2*(i-1)+1;
    %Fiint
    if i==1
        dFiintdxi=[-WX2(i+1)-WX2(i) -WXY(i+1)-WXY(i)]';
        dFiintdyi=[-WXY(i+1)-WXY(i) -WY2(i+1)-WY2(i)]';
        dFiintdxip1=[WX2(i+1) WXY(i+1)]';
        dFiintdyip1=[WXY(i+1) WY2(i+1)]';
        J_int(i0:i0+1,j0:j0+3)=[dFiintdxi    dFiintdyi    dFiintdxip1
dFiintdyip1];
        %i
        %J_int
    elseif (i>1 & i<N_band)
%         dFiintdxim1=[WX2(i-1) WXY(i-1)]';
%         dFiintdyim1=[WXY(i-1) WY2(i-1)]';
%         dFiintdxi=[-WX2(i+1)-WX2(i) -WXY(i+1)-WXY(i)]';
%         dFiintdyi=[-WXY(i+1)-WXY(i) -WY2(i+1)-WY2(i)]';
%         dFiintdxip1=[WX2(i+1) WXY(i+1)]';
%         dFiintdyip1=[WXY(i+1) WY2(i+1)]';
        dFiintdxim1=[WX2(i) WXY(i)]';
        dFiintdyim1=[WXY(i) WY2(i)]';
        dFiintdxi=[-WX2(i+1)-WX2(i) -WXY(i+1)-WXY(i)]';
        dFiintdyi=[-WXY(i+1)-WXY(i) -WY2(i+1)-WY2(i)]';
        dFiintdxip1=[WX2(i+1) WXY(i+1)]';
        dFiintdyip1=[WXY(i+1) WY2(i+1)]';

        J_int(i0:i0+1,j0-2:j0+3)=[dFiintdxim1  dFiintdyim1  dFiintdxi
dFiintdyi dFiintdxip1 dFiintdyip1];
        %i
        %J_int
    else
%         dFiintdxim1=[WX2(i-1) WXY(i-1)]';
%         dFiintdyim1=[WXY(i-1) WY2(i-1)]';
%         dFiintdxi=[-WX2(i) -WXY(i)]';
%         dFiintdyi=[-WXY(i) -WY2(i)]';
        dFiintdxim1=[WX2(i) WXY(i)]';
        dFiintdyim1=[WXY(i) WY2(i)]';
        dFiintdxi=[-WX2(i) -WXY(i)]';
        dFiintdyi=[-WXY(i) -WY2(i)]';
        J_int(i0:i0+1,j0-2:j0+1)=[dFiintdxim1  dFiintdyim1  dFiintdxi
dFiintdyi];
        %i
        %J_int
    end;

    %----------------------
    %Compute effect of Fibq
    dFiBldxi=[0 0]';
    dFiBldyi=[0 -WEBlY1(i)]';
    dFiBrdxi=[0 0]';
    dFiBrdyi=[0 -WEBrY1(i)]';
%     dFBqdri=[dFiBldxi+dFiBrdxi dFiBldyi+dFiBrdyi];
%     J_mat(i0:i0+1,j0:j0+1)=J_mat(i0:i0+1,j0:j0+1)+dFBqdri;
```

```
    J_Bl(i0:i0+1,j0:j0+1)=[dFiBldxi dFiBldyi];
    J_Br(i0:i0+1,j0:j0+1)=[dFiBrdxi dFiBrdyi];
    %-------------------------------------------------------------------
-
end;

%-----------------------------------------------------------------------
%Compute effect of F_rigobs
%M=size(R_rigobs,1);
for j=1:M
    switch Vers_rigobs
        case 1
            %Version 1
            robsj=R_rigobs(j,:);
            Robsj=repmat(robsj,N_band,1);
            dRobsj=R_band-Robsj;
            dRobsjabs=sqrt(sum((dRobsj.*dRobsj)'))';
            dRobsjexp=K_rigobs(j)*exp(-
0.5*((dRobsjabs./S_rigobs(j)).^2));
            dRobsjabs3=dRobsjabs.^3;
            dRobsjabs13=(S_rigobs(j)^(-2))./dRobsjabs+1./dRobsjabs3;
            dXobsj=dRobsj(:,1);
            dYobsj=dRobsj(:,2);
            dXobsj2=dXobsj.^2;
            dXYobsj=dXobsj.*dYobsj;
            dYobsj2=dYobsj.^2;
            WOX=dRobsjexp.*(1./dRobsjabs-dXobsj2.*dRobsjabs13);
            WOXY=dRobsjexp.*(-dXYobsj.*dRobsjabs13);
            WOY=dRobsjexp.*(1./dRobsjabs-dYobsj2.*dRobsjabs13);
        case 2
            %Version 2
            robsj=R_rigobs(j,:);
            Robsj=repmat(robsj,N_band,1);
            dRobsj=R_band-Robsj;
            dRobsjabs=sqrt(sum((dRobsj.*dRobsj)'))'+eps;
            dRobsjhyp=(K_rigobs(j)*D_rigobs(j)/2)./dRobsjabs;
            dRobsjabs3=dRobsjabs.^3;
            dXobsj=dRobsj(:,1);
            dYobsj=dRobsj(:,2);
            dXobsj2=dXobsj.^2;
            dXYobsj=dXobsj.*dYobsj;
            dYobsj2=dYobsj.^2;
            WOX=dRobsjhyp.*(1./dRobsjabs-2*dXobsj2./dRobsjabs3);
            WOXY=dRobsjhyp.*(-2*dXYobsj./dRobsjabs3);
            WOY=dRobsjhyp.*(1./dRobsjabs-2*dYobsj2./dRobsjabs3);
        otherwise
            ;
    end;
    %
    for i=1:N_band
        i0=2*(i-1)+1;
        j0=2*(i-1)+1;
        dFiobsjdxi=[WOX(i) WOXY(i)]';
        dFiobsjdyi=[WOXY(i) WOY(i)]';

J_rigobs(i0:i0+1,j0:j0+1)=J_rigobs(i0:i0+1,j0:j0+1)+[dFiobsjdxi
dFiobsjdyi];
    end;
end;

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

```
% %Compute effect of F_moveobs
% %----------------
if N_moveobs>0
%   % Rext=[R_0; R_band];
%   % Next=size(Rext,1);
%   % dR=Rext(2:Next,:)-Rext(1:Next-1,:);
%   % dRabs=sqrt(sum((dR.*dR)'))';
%   t_own=dRabs./v_own;
%   t_own=cumsum(t_own);
%             rt_moveobs=repmat(r_moveobs,N_band,1)+[-v_moveobs*t_own
zeros(N_band,1)];
%   dRm=R_band-rt_moveobs;
%   dRmabs=sqrt(sum((dRm.*dRm)'))';
    switch Vers_moveobs
        case 1
            %Version 1
            dRmabsexp=k_moveobs*exp(-0.5*((dRmabs/s_moveobs).^2));
            dRmabs3=dRmabs.^3;
            dRmabs13=(s_moveobs^(-2))./dRmabs+1./dRmabs3;
            dXm=dRm(:,1);
            dYm=dRm(:,2);
            dXm2=dXm.^2;
            dXYm=dXm.*dYm;
            dYm2=dYm.^2;
            WmX=dRmabsexp.*(1./dRmabs-dXm2.*dRmabs13);
            WmXY=dRmabsexp.*(-dXYm.*dRmabs13);
            WmY=dRmabsexp.*(1./dRmabs-dYm2.*dRmabs13);
        case 2
            %Version 2
            dRmabshyp=(k_moveobs*d_moveobs/2)./dRmabs;
            dRmabs3=dRmabs.^3;
            dXm=dRm(:,1);
            dYm=dRm(:,2);
            dXm2=dXm.^2;
            dXYm=dXm.*dYm;
            dYm2=dYm.^2;
            WmX=dRmabshyp.*(1./dRmabs-2*dXm2./dRmabs3);
            WmXY=dRmabshyp.*(-2*dXYm./dRmabs3);
            WmY=dRmabshyp.*(1./dRmabs-2*dYm2./dRmabs3);
        otherwise
            ;
    end;
    %
    for i=1:N_band
        i0=2*(i-1)+1;
        j0=2*(i-1)+1;
        dFimobsdxi=[WmX(i) WmXY(i)]';
        dFimobsdyi=[WmXY(i) WmY(i)]';

J_moveobs(i0:i0+1,j0:j0+1)=J_moveobs(i0:i0+1,j0:j0+1)+[dFimobsdxi
dFimobsdyi];
    end;
end;
% %----------------
%
%-------------------------------------------------------------------
Jmat=J_int+J_Bl+J_Br+J_rigobs+J_moveobs;
%-------------------------------------------------------------------

% %Compute new nodes
% Rxy=R_band';
```

```
% xold=Rxy(:);
% Fxy=F_total';
% fold=Fxy(:);
% %
% xnew=xold-pinv(Jmat)*fold;
% Rxynew=NaN*ones(2,N_band);
% Rxynew(:)=xnew;
% %
% R_band=Rxynew';
% F_band=F_total;
% J_band=Jmat;

%Conversion to long input vector
Fxy=F_total';
fold=Fxy(:);
fout=fold;
Jout=Jmat;
fint=F_int';
F_int=fint(:);
fbl=F_Bl';
F_Bl=fbl(:);
fbr=F_Br';
F_Br=fbr(:);
frigobs=F_rigobs';
F_rigobs=frigobs(:);
R_band=oldR_band;
```

## 9.5 ddc_euler.m

```
function
[tt,xx,xxestim,F_lRR,deltaww,Svv]=ddc_euler(funxdot,tspan,x0);
%Compute DDC vehicle control transients

global N_band N_rigobs N_moveobs R_0 R_band F_band J_band
global K_band L_band k_Bl k_Br s_Bl s_Br B_wide
global  K_rigobs  S_rigobs  R_rigobs  D_rigobs  k_moveobs  s_moveobs
r_moveobs d_moveobs v_moveobs
global v_own
global Vers_rigobs Vers_moveobs

%frame global
global s_kin T_samp lambda spp_x spp_y
global c_F l_F c_R l_R m_v I_zz
global sys_appr
global sys_estim
global sys_contr

%GPS/INS
global gyrosens_corr
global TsINS TsGPSvelocity TsGPSattitude
global      sigma1_acc      sigma1_rategyro      sigma1_GPSvelocity
sigma1_GPSattitude sigma1_X sigma1_Y
global    bias_rate    bias_acc    sigma1_ratebias    sigma1_accbias
sigma1_rateinvsens sigma1_ratebiasdivsens
global cov1p cov1m cov2p cov2m dimx1p irbias
global xhat_KF xp_KF xm_KF x1p_KF x1m_KF x2p_KF x2m_KF P1p_KF P1m_KF
P2p_KF P2m_KF
global xsys_prev
global xhat_KF_prev
global N_horizon t0_horizon xx_horizon uu_horizon

%Initiate stochastic parameters of GPS/INS for Kalman Filters
init_GPS_INS;

t0=tspan(1);
tf=tspan(2);
nf=floor((tf-t0)/T_samp);
t0=0;
tf=nf*T_samp;

tt=[0:T_samp:tf]';
nf=length(tt);
xx=NaN*ones(nf,6);
xxestim=NaN*ones(nf,6);
F_lRR=NaN*ones(nf,1);
deltaww=NaN*ones(nf,1);;
Svv=NaN*ones(nf,1);

xx(1,:)=x0';
xi=x0;

xhat_KF=x0;
xxestim(1,:)=xhat_KF';
xp_KF=xhat_KF;
xm_KF=xp_KF;
x1p_KF=[x0(2) ones(1,dimx1p-2) 0]';
x1m_KF=x1p_KF;
```

```
x2p_KF=[x0(4) 0 0 0]';
x2m_KF=x2p_KF;
xsys_prev=xi;
xhat_KF_prev=xhat_KF;

for i=1:nf
    ti=(i-1)*T_samp;
    [xdot,FlR,deltaw,Sv]=feval(funxdot,ti,xi);
    xx(i,:)=xi';
    xxestim(i,:)=xhat_KF_prev';
    xip1=xi+T_samp*xdot;
    xsys_prev=xi;
    xi=xip1;
    F_lRR(i,:)=FlR;
    deltaww(i,:)=deltaw;
    Svv(i,:)=Sv;
end;
```

## 9.6 deltaw2Sv.m

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

%frame global
global s_kin T_samp lambda spp_x spp_y
global c_F l_F c_R l_R m_v I_zz
global sys_appr
global sys_estim
global sys_contr

col=size(x,2);
if col==1, x=x'; 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);

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

## 9.7 deriv3.m

```
function [vy,vd1y,vd2y,vd3y,spp]=deriv3(x,y,xx)
%2nd order derivative
if nargin<3, xx=x; end;
pp=spline(x,y);
vy=ppval(pp,xx);
[breaks,coeffs,l,k,d]=unmkpp(pp);
[nc,mc]=size(coeffs);
d1=[3 2 1 0];
d1r=repmat(d1,nc,1);
xcoeffs=coeffs.*d1r;
d1coeffs=xcoeffs(:,1:3);
d2=[6 2 0 0];
d2r=repmat(d2,nc,1);
xcoeffs=coeffs.*d2r;
d2coeffs=xcoeffs(:,1:2);
d3=[6 0 0 0];
d3r=repmat(d3,nc,1);
xcoeffs=coeffs.*d3r;
d3coeffs=xcoeffs(:,1);
ppd1=mkpp(breaks,d1coeffs,d);
vd1y=ppval(ppd1,xx);
ppd2=mkpp(breaks,d2coeffs,d);
vd2y=ppval(ppd2,xx);
ppd3=mkpp(breaks,d3coeffs,d);
vd3y=ppval(ppd3,x);
vd3y=interp1(x,vd3y,xx);
spp.pp=pp;
spp.ppd1=ppd1;
spp.ppd2=ppd2;
return;
figure(1);
subplot(111);
if length(x)~=length(y), y0=y(2:length(y)-1); else y0=y; end;
plot(x,y0,xx,vy);
title('vy & y');
figure(2);
subplot(111);
plot(xx,vd1y);
title('vd1y');
figure(3);
subplot(111);
plot(xx,vd2y);
title('vd2y');
figure(4);
subplot(111);
plot(xx,vd3y);
title('vd3y');
```

## 9.8 dfapprdx.m

```
function [dfdx,dfdu]=dfapprdx(x,u,deltawinput)
%First derivative of f(x,u) (approximated)

%-----------------------------------------------------
%Modified: 2007.06.20. (corrections if u(1)=deltaw)
%-----------------------------------------------------

%frame global
global s_kin T_samp lambda spp_x spp_y
global c_F l_F c_R l_R m_v I_zz

if nargin<3, 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);
else
    deltaw=u(1);
    F_lR=u(2);
    Sv=deltaw2Sv(deltaw,x);
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;
```

## 9.9 dg_controller.m

```
function [xdot,F_lR,deltaw,Sv]=dg_controller(t,x)
%Compute control for 'diffgeom'

global N_band N_rigobs N_moveobs R_0 R_band F_band J_band
global K_band L_band k_Bl k_Br s_Bl s_Br B_wide
global  K_rigobs  S_rigobs  R_rigobs  D_rigobs  k_moveobs  s_moveobs
r_moveobs d_moveobs v_moveobs
global v_own
global Vers_rigobs Vers_moveobs

%frame global
global s_kin T_samp lambda spp_x spp_y
global c_F l_F c_R l_R m_v I_zz
global sys_appr
global sys_estim
global sys_contr

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);

alpha0i=lambda;
alpha1i=2*sqrt(lambda);

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

[xt,yt,d1xt,d1yt,d2xt,d2yt]=ybar(t);
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;

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;
```

## 9.10 funframeCAS.m

```
function funframeCAS
%frameCAS.m
%Compute path, trajectory and control transients for Collision
Avoidance System

%-----------------------------------------------------
%Modified: 2007.06.20. (corrections if u(1)=deltaw)
%-----------------------------------------------------

clear all
close all

%-------------------------------------
%Input parameters in Syspar_file.txt
%-------------------------------------
global   fv_own   fN_statobs   fpar_statobs   fN_movobs   fpar_movobs
froad_wide
global     fsys_appr     fsys_estim     fsys_contr     fdeltaw_horizon
fdgfresh_horizon
global flambda_horizon fint_horizon fLTV_horizon

retcode=read_syspar;
%-------------------------------------

global N_band N_rigobs N_moveobs R_0 R_band F_band J_band
global K_band L_band k_Bl k_Br s_Bl s_Br B_wide
global  K_rigobs  S_rigobs  R_rigobs  D_rigobs  k_moveobs  s_moveobs
r_moveobs d_moveobs v_moveobs
global v_own
global Vers_rigobs Vers_moveobs

%frame global
global s_kin T_samp lambda spp_x spp_y spp_psi spp_d1psi spp_v
global c_F l_F c_R l_R m_v I_zz
global sys_appr
global sys_estim
global sys_contr

%GPS/INS
global gyrosens_corr
global TsINS TsGPSvelocity TsGPSattitude
global       sigma1_acc       sigma1_rategyro       sigma1_GPSvelocity
sigma1_GPSattitude sigma1_X sigma1_Y
global    bias_rate    bias_acc    sigma1_ratebias    sigma1_accbias
sigma1_rateinvsens sigma1_ratebiasdivsens
global cov1p cov1m cov2p cov2m dimx1p irbias
global xhat_KF xp_KF xm_KF x1p_KF x1m_KF x2p_KF x2m_KF P1p_KF P1m_KF
P2p_KF P2m_KF
global prev_x2p_KF

%Predictive control
global  N_horizon  t0_horizon  xx_horizon  uu_horizon  deltaw_horizon
dgfresh_horizon ulast_horizon
global lambda_horizon int_horizon LTV_horizon

%Initiate elastic band
initband;
T_samp=0.010;   %Sampling time
```

```
%Initiate kinematics
[s_kin,spp_x,spp_y,spp_psi,spp_d1psi,spp_v]=kinematics(T_samp);

fig0=2; %save initband, optimband
plots_kin(s_kin,fig0);

%Initiate vehicle parameters c_F, l_F, c_R, l_R, m_v, I_zz
initvehiclepar;
% c_F=100000;
% l_F=1.203;
% c_R=100000;
% l_R=1.217;
% m_v=1280;
% I_zz=2500;

%----------------------------
%Initiate eigenvalue lambda for yibar in the Controller
lambda=10;  %for controller 'diffgeom'
%----------------------------

%Initiate approximation
sys_appr=fsys_appr;
%sys_appr=1;

%Initiate estimation
sys_estim=fsys_estim;
%sys_estim=1;

%Initiate controller type
%sys_contr='diffgeom';
sys_contr=fsys_contr;
%sys_contr='nonlinpred';

%----------------------------
%Initiate horizon length (only for nonlinpred
%sys_contr='nonlinpred'
N_horizon=10;
deltaw_horizon=fdeltaw_horizon;
%deltaw_horizon=0;   %If 0 then u(1)=Sv, otherwise u(1)=deltaw
dgfresh_horizon=fdgfresh_horizon;
%dgfresh_horizon=1;   %1=>u(N) by diffgeom, 0=>uN makes xNp1 to 0
(affin appr.), otherwise uN=uNm1
lambda_horizon=flambda_horizon;
%lambda_horizon=10;  %lambda weights u or deltau in cost function for
RHC controller
int_horizon=fint_horizon;
%int_horizon=1;      %If 1 then integrator in RHC controller
LTV_horizon=fLTV_horizon;
%LTV_horizon=1;         %If 0 then LTI linearized system is built,
otherwise LTV
%
if deltaw_horizon, LTV_horizon=0; end;
%u(1)=deltaw => only LTV linearization is allowed
%
t0_horizon=[];
xx_horizon=[];
uu_horizon=[];
%----------------------------

%Init Receding horizon
```

```
if strcmp(sys_contr,'nonlinpred')
    inithorizon;
end;

%Compute initial state
x0=zeros(6,1);
x0(4)=v_own;

%Compute state transients
t0=0;
ttkin=s_kin.tt;
tf=ttkin(length(ttkin));
if strcmp(sys_contr,'nonlinpred')
    tf=tf-(N_horizon+2)*T_samp; %tf-(N+1)*T_samp is also OK
else
    tf=tf-2*T_samp; %tf-0 is also OK
end;
%
tspan=[t0 tf];
fprintf('%%************************************************************
\n');
fprintf('''%s'' control is running, t0=%g, tf=%g, Tsamp=%g\n',...
    sys_contr,t0,tf,T_samp);
%options=odeset('RelTol',1e-6,'AbsTol',1e-10);
[tt,xx,xxestim,F_lRR,deltaww,Svv]=ddc_euler(@funxdot,tspan,x0);
fprintf('''%s'' control has been finished\n',sys_contr);
fprintf('%%************************************************************
\n');

%Compute path x(t), y(t) in inertia system
figure
subplot(211);
plot(tt,xx(:,5)); title('x(t)');
subplot(212);
plot(tt,xx(:,6)); title('y(t)');

%Draw state variables
if ~sys_estim
    figure
    subplot(231);
    plot(tt,xx(:,1)); title('beta');
    subplot(232);
    plot(tt,xx(:,2)); title('psi');
    subplot(233);
    plot(tt,xx(:,3)); title('d1psi');
    subplot(234);
    plot(tt,xx(:,4)); title('vG');
    subplot(235);
    plot(tt,xx(:,5)); title('X');
    subplot(236);
    plot(tt,xx(:,6)); title('Y');
else
    figure
    subplot(231);
    plot(tt,[xx(:,1) xxestim(:,1)]); title('beta');
    subplot(232);
    plot(tt,[xx(:,2) xxestim(:,2)]); title('psi');
    subplot(233);
    plot(tt,[xx(:,3) xxestim(:,3)]); title('d1psi');
    subplot(234);
    plot(tt,[xx(:,4) xxestim(:,4)]); title('vG');
```

```
    subplot(235);
    plot(tt,[xx(:,5) xxestim(:,5)]); title('X');
    subplot(236);
    plot(tt,[xx(:,6) xxestim(:,6)]); title('Y');
end;

%Compute F_lR and deltaw
if strcmp(sys_contr,'diffgeom')
    figure
    subplot(311);
    plot(tt,Svv); title('Sv');
    subplot(312);
    plot(tt,F_lRR); title('FlR');
    subplot(313);
    plot(tt,deltaww); title('deltaw');
elseif strcmp(sys_contr,'nonlinpred')
    figure
    subplot(311);
    plot(tt,Svv); title('Sv');
    subplot(312);
    plot(tt,F_lRR); title('FlR');
    subplot(313);
    plot(tt,deltaww); title('deltaw');
else
    ;
end;

[xxref,yyref]=ybar(tt);
exx=xxref-xx(:,5);
eyy=yyref-xx(:,6);
psii=ppval(spp_psi.pp,tt);
epsii=psii-xx(:,2);
figure
subplot(311);
plot(tt,exx); title(['ex=xref-x (' sys_contr ')']);
subplot(312);
plot(tt,eyy); title(['ey=yref-y (' sys_contr ')']);
subplot(313);
plot(tt,epsii); title(['epsi=psiref-psi (' sys_contr ')']);

fprintf('%%************************************\n');
fprintf('sys_contr=''%s''\n',sys_contr);
fprintf('sys_appr=%g\n',sys_appr);
fprintf('sys_estim=%g\n',sys_estim);
fprintf('T_samp=%g\n',T_samp);
if strcmp(sys_contr,'nonlinpred')
    fprintf('N_horizon=%g\n',N_horizon);
    fprintf('deltaw_horizon=%g\n',deltaw_horizon);
    fprintf('dgfresh_horizon=%g\n',dgfresh_horizon);
    fprintf('lambda_horizon=%g\n',lambda_horizon);
    fprintf('int_horizon=%g\n',int_horizon);
    fprintf('LTV_horizon=%g\n',LTV_horizon);
else
    fprintf('lambda=%g\n',lambda);
end;
fprintf('%%************************************\n');

%Because of standalone running after compilation
s_stop='w';
while ~strcmp(s_stop,'s')
    s_stop=input('wait/stop? => (w/s)','s');
```

```
end;
close all
return
```

## 9.11 funxdot.m

```
function [xdot,F_lR,deltaw,Sv]=funxdot(t,x)
%Compute right side of vehicle dynamic model

%Modified: 2007.03.16.

global N_band N_rigobs N_moveobs R_0 R_band F_band J_band
global K_band L_band k_Bl k_Br s_Bl s_Br B_wide
global  K_rigobs  S_rigobs  R_rigobs  D_rigobs  k_moveobs  s_moveobs
r_moveobs d_moveobs v_moveobs
global v_own
global Vers_rigobs Vers_moveobs

%frame global
global s_kin T_samp lambda spp_x spp_y
global c_F l_F c_R l_R m_v I_zz
global sys_appr
global sys_estim
global sys_contr

%GPS/INS
global gyrosens_corr
global TsINS TsGPSvelocity TsGPSattitude
global      sigma1_acc       sigma1_rategyro       sigma1_GPSvelocity
sigma1_GPSattitude sigma1_X sigma1_Y
global    bias_rate    bias_acc    sigma1_ratebias    sigma1_accbias
sigma1_rateinvsens sigma1_ratebiasdivsens
global cov1p cov1m cov2p cov2m dimx1p irbias
global xhat_KF xp_KF xm_KF x1p_KF x1m_KF x2p_KF x2m_KF P1p_KF P1m_KF
P2p_KF P2m_KF
global xsys_prev
global xhat_KF_prev
global N_horizon t0_horizon xx_horizon uu_horizon

if strcmp(sys_contr,'diffgeom')
    if ~sys_estim
        %No estimation
        %Compute controller output based on nonestimated state
        [apprxdot,F_lR,deltaw,Sv]=dg_controller(t,x);
        %Simulation
        if sys_appr~=0
            %Simulate  system  answer  using  approximated  system  and
correct system state
            xdot=apprfunxdot(t,x,F_lR,deltaw);
        else
            %Simulate system answer using precise system and correct
system state
            xdot=precfunxdot(t,x,F_lR,deltaw);
        end;
    else
        %Estimation
        fprintf('Estimation will be developed later\n');
        ;
        pause;
        return
    end;
elseif strcmp(sys_contr,'nonlinpred')
    if ~sys_estim
        %No estimation
```

```
        %Compute controller output based on nonestimated state
        x0=x;
        [xdot,F_lR,deltaw,Sv]=rhc2_controller(x0);
        %Simulation
        if sys_appr~=0
            %Simulate system answer using approximated system and
correct system state
            xdot=apprfunxdot(t,x,F_lR,deltaw);
        else
            %Simulate system answer using precise system and correct
system state
            xdot=precfunxdot(t,x,F_lR,deltaw);
        end;
    else
        %Estimation
        fprintf('Estimation will be developed later\n');
        ;
        pause;
        return
    end;
else
    %For future extension
    return;
end;
```

## 9.12 funxdvec.m

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

%frame global
global s_kin T_samp lambda spp_x spp_y spp_psi spp_d1psi spp_v
global c_F l_F c_R l_R m_v I_zz
global sys_appr
global sys_estim
global sys_contr

nt=length(t);
[xt,yt]=ybar(t);
psit=ppval(spp_psi.pp,t);
d1psit=ppval(spp_d1psi.pp,t);
vGt=ppval(spp_v.pp,t);
if nt==1
    xdvect=[zeros(nt,1) psit d1psit vGt xt yt]';
else
    xdvect=[zeros(nt,1) psit d1psit vGt xt yt];
end;
```

## 9.13 init_GPS_INS.m

```
function init_GPS_INS
%Initiate stochastic paprameters of GPS/INS for Kalman Filtering

%GPS/INS
global gyrosens_corr
global TsINS TsGPSvelocity TsGPSattitude
global       sigma1_acc       sigma1_rategyro       sigma1_GPSvelocity
sigma1_GPSattitude sigma1_X sigma1_Y
global    bias_rate    bias_acc    sigma1_ratebias    sigma1_accbias
sigma1_rateinvsens sigma1_ratebiasdivsens
global cov1p cov1m cov2p cov2m dimx1p irbias
global xhat_KF xp_KF xm_KF x1p_KF x1m_KF x2p_KF x2m_KF P1p_KF P1m_KF
P2p_KF P2m_KF

randn('state',0);

%---------------------------
%Gyro sensitivity switch
gyrosens_corr=0;
%---------------------------
%
TsINS=0.01;             %100 Hz (assumed to be equal Tsamp)
TsGPSvelocity=0.1;      %10 Hz
TsGPSattitude=0.2;      %5 Hz
%
sigma1_acc=0.05;                %m/s^2
sigma1_rategyro=0.2*pi/180;     %rad/s
sigma1_GPSvelocity=3*0.01;      %m/s
sigma1_GPSattitude=0.2*pi/180;  %rad
sigma1_X=sigma1_GPSvelocity*TsINS; %m
sigma1_Y=sigma1_GPSvelocity*TsINS*0.1; %m
%
bias_rate=0.05;                 %rad/sec
bias_acc=0.05;                  %m/s^2
sigma1_ratebias=0.05;           %rad/sec
sigma1_accbias=0.05;            %m/s^2
sigma1_rateinvsens=0.05;        %-
sigma1_ratebiasdivsens=0.05;    %rad/sec
%Covariance matrices for process and measurement noises
if ~gyrosens_corr
    cov1p=0.01*diag([sigma1_rategyro^2 sigma1_ratebias^2]);
    dimx1p=2;
    irbias=2;
else
    cov1p=0.01*diag([sigma1_rategyro^2           sigma1_rateinvsens^2
sigma1_ratebiasdivsens^2]);
    dimx1p=3;
    irbias=3;
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
P1p_KF=1e-6*eye(dimx1p);
P1m_KF=P1p_KF;
P2p_KF=1e-6*eye(4);
P2m_KF=P2p_KF;
```

```
%Reserve memory for states
xp_KF=[0 0 0 0 0 0]';
xm_KF=xp_KF;
x1p_KF=[0 ones(1,dimx1p-2) 0]';
x1m_KF=x1p_KF;
x2p_KF=[0 0 0 0]';
x2m_KF=x2p_KF;
```

## 9.14 initband.m

```
function initband
%Initiate elastic band

global N_band N_rigobs N_moveobs R_0 R_band F_band J_band
global K_band L_band k_Bl k_Br s_Bl s_Br B_wide
global  K_rigobs  S_rigobs  R_rigobs  D_rigobs  k_moveobs  s_moveobs
r_moveobs d_moveobs v_moveobs
global v_own
global Vers_rigobs Vers_moveobs

close all

%-------------------------------------------------------
%Input parameters by read_syspar from Syspar_File.txt
%-------------------------------------------------------
global   fv_own   fN_statobs   fpar_statobs   fN_movobs   fpar_movobs
froad_wide
global     fsys_appr     fsys_estim     fsys_contr     fdeltaw_horizon
fdgfresh_horizon
global flambda_horizon fint_horizon fLTV_horizon

%retcode=read_syspar;
N_rigobs=fN_statobs;
N_moveobs=fN_movobs;
v_own=fv_own;
B_wide=froad_wide;
R_rigobs=fpar_statobs(:,1:2);
D_rigobs=fpar_statobs(:,3);
d_moveobs=fpar_movobs(3);
r_moveobs=fpar_movobs(1:2);
v_moveobs=fpar_movobs(4);
%-------------------------------------------------------

%--------------------
%Example 1
N_band=41;
%N_rigobs=1;
%N_moveobs=1;
Vers_rigobs=2;
Vers_moveobs=1;
%
R_0=[0 0];
%v_own=20;  %m/s
K_band=1*ones(N_band,1);
L_band=1*ones(N_band,1);
% R_band=[cumsum(L_band) 5*ones(N_band, 1)];
% dx=1*ones(N_band,1);
% L_band=[1:N_band]';
R_band=[cumsum(L_band) 1*ones(N_band, 1)];
dx=0*ones(N_band,1);
dy=0*dx;
xy=[dx dy];
R_band=R_band+xy;
% R_band=[-1.1772   -0.0759; ...
%    -0.7139    0.7627; ...
%     0.3780    1.3017; ...
%     1.8844    1.8065; ...
%     3.7689    1.8686; ...
```

```
%     5.3657    1.3988; ...
%     6.5042    0.8133];
%
%-------------------------
%Road Baundary
%B_wide=[7 0.75 0.25];
b=B_wide(1);
bleft=b*B_wide(2);
bright=b*B_wide(3);
M_B=2; %1   %8;
m_B=0.05;   %0.1
k_Bl=M_B;
% expright=log(exp(-bleft^2)/exp(-bright^2));
% k_Br=8*exp(expright);
k_Br=M_B;
s_Bl=bleft/sqrt(2*log(M_B/m_B));
s_Br=bright/sqrt(2*log(M_B/m_B));
%-------------------------
%Rigid obstacles
switch Vers_rigobs
    case 1
        %Version 1
        M_rigobs=4*ones(N_rigobs,1);
        m_rigobs=1*ones(N_rigobs,1);
        K_rigobs=M_rigobs;
        %R_rigobs=40*[[1:N_rigobs]' 0*ones(N_rigobs,1)];
        %D_rigobs=2.5*ones(N_rigobs,1);
        S_rigobs=(D_rigobs/2)./sqrt(2*log(M_rigobs./m_rigobs));
    case 2
        %Version 2
        M_rigobs=3*ones(N_rigobs,1);
        K_rigobs=M_rigobs;
        %R_rigobs=40*[[1:N_rigobs]' 0*ones(N_rigobs,1)];
        %D_rigobs=2.5*ones(N_rigobs,1);
    otherwise
        ;
end;
%-------------------------
%Moving obstacle
switch Vers_moveobs
    case 1
        %Version 1
        M_moveobs=4;
        m_moveobs=1;
        %d_moveobs=4;
        k_moveobs=M_moveobs;
        s_moveobs=(d_moveobs/2)/sqrt(2*log(M_moveobs/m_moveobs));
        %r_moveobs=[120 3.5];
        %v_moveobs=15;   %m/s
    case 2
        %Version 2
        M_moveobs=0.5;
        %d_moveobs=4;
        k_moveobs=M_moveobs;
        %r_moveobs=[120 3.5];
        %v_moveobs=15;   %m/s
    otherwise
        ;
end;
%-------------------------
figure
```

```
avoidrigobs;
%-------------------------
F_band=[];
J_band=[];
%----------
rinx=R_band';
rin=rinx(:);
options=optimset('Jacobian','on','LargeScale','off');
[xout,fval,exitflag,output,jacobian]=fsolve(@banditer,rin,options);
%xout
rbandx=NaN*ones(2,N_band);
rbandx(:)=xout;
R_band=rbandx';

%close all
% fout=banditer(xout);
figure
plotband(xout,[],R_rigobs,D_rigobs,r_moveobs,d_moveobs,v_moveobs,R_0,
v_own);
pause
%close all
% s_kin=kinematics(0.020);
% plots_kin(s_kin);
%keyboard;
return;

cycle=1
R_band
F_band
pause
while cycle
    rinx=R_band';
    rin=rinx(:);

[fout,Jout,F_int,F_Bl,F_Br,F_rigobs,F_moveobs,J_int,J_Bl,J_Br,J_rigob
s]=banditer(rin);
    R_band
    %F_band
    %J_band
    %keyboard
    pause
end;
%----------
keyboard
```

## 9.15 inithorizon.m

```
function inithorizon
%Initialize state and control for the first horizon of 'nonlinpred'

%band global
global N_band N_rigobs N_moveobs R_0 R_band F_band J_band
global K_band L_band k_Bl k_Br s_Bl s_Br B_wide
global  K_rigobs  S_rigobs  R_rigobs  D_rigobs  k_moveobs  s_moveobs
r_moveobs d_moveobs v_moveobs
global v_own
global Vers_rigobs Vers_moveobs

%frame global
global s_kin T_samp lambda spp_x spp_y spp_psi spp_d1psi spp_v
global c_F l_F c_R l_R m_v I_zz
global sys_appr
global sys_estim
global sys_contr

%Predictive control
global  N_horizon  t0_horizon  xx_horizon  uu_horizon  deltaw_horizon
dgfresh_horizon ulast_horizon
global lambda_horizon int_horizon LTV_horizon

%
N=N_horizon;
t0=0;
ts=T_samp*N;
tt=[0:T_samp:ts]';
xx=NaN*ones(N+1,6);     %x0, x1, ..., xN
%uu(i,:) order:         %Sv, F_lR
uu=NaN*ones(N,2);       %u0, u1, ..., uNm1

%
x0=zeros(6,1);
x0(4)=v_own;
xx(1,:)=x0';
nf=length(tt);

for i=1:nf-1
    ti=(i-1)*T_samp;
    xi=xx(i,:)';
    [xdot,F_lR,deltaw,Sv]=dg_controller(ti,xi);
    xip1=xi+T_samp*xdot;
    if ~deltaw_horizon
        ui=[Sv F_lR]';
    else
        ui=[deltaw F_lR]';
    end
    uu(i,:)=ui';
    xx(i+1,:)=xip1';
end;

t0_horizon=t0;
xx_horizon=xx;
uu_horizon=uu;
ulast_horizon=uu_horizon(1,:)';
```

## 9.16 initvehiclepar.m

```
function initvehiclepar
%Initiate vehicle parameters c_F, l_F, c_R, l_R, m_v, I_zz

%frame global
global c_F l_F c_R l_R m_v I_zz

c_F=100000;
l_F=1.203;
c_R=100000;
l_R=1.217;
m_v=1280;
I_zz=2500;
```

## 9.17 kinematics.m

```
function [s_kin,sppx,sppy,spppsi,sppd1psi,sppv]=kinematics(T)
%Find reference states and controls for CAS

global N_band N_rigobs N_moveobs R_0 R_band F_band J_band
global K_band L_band k_Bl k_Br s_Bl s_Br B_wide
global  K_rigobs  S_rigobs  R_rigobs  D_rigobs  k_moveobs  s_moveobs
r_moveobs d_moveobs v_moveobs
global v_own
global Vers_rigobs Vers_moveobs

Rext=[R_0; R_band];
Next=size(Rext,1);
dR=Rext(2:Next,:)-Rext(1:Next-1,:);
dRabs=sqrt(sum((dR.*dR)'))';
t_cum=dRabs./v_own;
t_cum=cumsum(t_cum);
t_final=t_cum(length(t_cum));

t0=[0; t_cum];
tt=[0:T:t_final]';
x0=Rext(:,1);
y0=Rext(:,2);

[x,d1x,d2x,d3x,sppx]=deriv3(t0,x0,tt);
[y,d1y,d2y,d3y,sppy]=deriv3(t0,y0,tt);

%
[xx,d1xx,d2xx,d3xx]=deriv3(t0,x0);
d2xx=deriv3(t0,d2xx,tt);
d3xx=deriv3(t0,d3xx,tt);
[yy,d1yy,d2yy,d3yy]=deriv3(t0,y0);
d2yy=deriv3(t0,d2yy,tt);
d3yy=deriv3(t0,d3yy,tt);

%
v=sqrt(d1x.^2+d1y.^2);
v2=v.^2;    %(d1x.^2+d1y.^2)
v3=v.^3;    %(d1x.^2+d1y.^2).^(3/2)
d1v=(d1x.*d2x+d1y.*d2y)./v;
kappa=(d2y.*d1x-d1y.*d2x)./v3;
psi=atan(d1y./d1x);
d1psi=(d2y.*d1x-d1y.*d2x)./v2;
d2psi=(d3y.*d1x-d1y.*d3x)./v2-2*d1psi.*(d1x.*d2x+d1y.*d2y)./v2;
%
d1vv=(d1x.*d2xx+d1y.*d2yy)./v;
kappaa=(d2yy.*d1x-d1y.*d2xx)./v3;
d1psii=(d2yy.*d1x-d1y.*d2xx)./v2;
d2psii=(d3yy.*d1x-d1y.*d3xx)./v2-2*d1psi.*(d1x.*d2xx+d1y.*d2yy)./v2;
%
s_kin.tt=tt;
s_kin.x=x;
s_kin.d1x=d1x;
s_kin.d2x=d2x;
s_kin.d3x=d3x;
s_kin.y=y;
s_kin.d1y=d1y;
s_kin.d2y=d2y;
s_kin.d3y=d3y;
```

```
s_kin.v=v;
s_kin.d1v=d1v;
s_kin.kappa=kappa;
s_kin.psi=psi;
s_kin.d1psi=d1psi;
s_kin.d2psi=d2psi;

%
s_kin.d1xx=d1xx;
s_kin.d2xx=d2xx;
s_kin.d3xx=d3xx;
s_kin.d1yy=d1yy;
s_kin.d2yy=d2yy;
s_kin.d3yy=d3yy;
s_kin.d1vv=d1vv;
s_kin.kappaa=kappaa;
s_kin.d1psii=d1psii;
s_kin.d2psii=d2psii;
%
[psi,d1psi,d2psi,d3psi,spppsi]=deriv3(tt,psi);
[d1psi,d2psi,d3psi,d4psi,sppd1psi]=deriv3(tt,d1psi);
[v,d1v,d2v,d3v,sppv]=deriv3(tt,v);

%
```

## 9.18 LTV2vehicle.m

```
function [AA,BB]=LTV2vehicle(xx,uu,Ts,LTVhor,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<5, deltawinput=0; end;

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

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,:)';
    else
        xi=xx(1,:)';
        ui=uu(1,:)';
    end;
    [dfcdxi,dfcdui]=dfapprdx(xi,ui,deltawinput);
    Ai=eye(n)+Ts*dfcdxi;
    Bi=Ts*dfcdui;
    ra=i*n;
    rb=i*n;
    AA(ra+1:ra+n,:)=Ai;
    BB(rb+1:rb+n,:)=Bi;
end;
```

## 9.19 pause.m

```
function pause
s=input('pause');
```

## 9.20 plotband.m

```
function
plotband(rband1D,fband1D,R_rigobs,D_rigobs,r_moveobs,d_moveobs,...
    v_moveobs,R_0,v_own)
%plot elastic band and obstacles
N_band=length(rband1D)/2;
fmax=max(abs(fband1D));
R_bandx=NaN*ones(2,N_band);
R_bandx(:)=rband1D;
R_band=R_bandx';
% F_bandx=NaN*ones(2,N_band);
% F_bandx(:)=fband1D;
% F_band=F_bandx';
% quiver(R_band(:,1),R_band(:,2),F_band(:,1),F_band(:,2));

hold off
hold on

plot(R_band(:,1),R_band(:,2),'r-',R_band(:,1),R_band(:,2),'bo');


M=size(R_rigobs,1);
for j=1:M
    rj=R_rigobs(j,:);
    dj=D_rigobs(j);
    phi=[-pi:0.01:pi]';
    plot(rj(1)+dj/2*cos(phi),rj(2)+dj/2*sin(phi),'k-');
end;

Rext=[R_0; R_band];
Next=size(Rext,1);
dR=Rext(2:Next,:)-Rext(1:Next-1,:);
dRabs=sqrt(sum((dR.*dR)'))';
t_own=dRabs./v_own;
t_own=cumsum(t_own);
tlast=t_own(length(t_own));
rm=r_moveobs+[-tlast*v_moveobs 0];
dm=d_moveobs;
phi=[-pi:0.01:pi]';
plot(rm(1)+dm/2*cos(phi),rm(2)+dm/2*sin(phi),'c-');
rm12=[r_moveobs; rm];
plot(rm12(:,1),rm12(:,2),'c-
',r_moveobs(1),r_moveobs(2),'co',rm(1),rm(2),'c<');

hold off
```

## 9.21 plots_kin.m

```
function plots_kin(s_kin,fig0)
%Plot structure s_kin of kinematics quantities

if nargin<2, fig0=0; end;

%
figure(fig0+1)
subplot(221);
plot(s_kin.tt,s_kin.x); title('x');
subplot(222);
plot(s_kin.tt,s_kin.d1x); title('d1x');
subplot(223);
plot(s_kin.tt,s_kin.d2x); title('d2x');
subplot(224);
plot(s_kin.tt,s_kin.d3x); title('d3x');
%
figure(fig0+2)
subplot(221);
plot(s_kin.tt,s_kin.y); title('y');
subplot(222);
plot(s_kin.tt,s_kin.d1y); title('d1y');
subplot(223);
plot(s_kin.tt,s_kin.d2y); title('d2y');
subplot(224);
plot(s_kin.tt,s_kin.d3y); title('d3y');
%
figure(fig0+3)
subplot(211);
plot(s_kin.tt,s_kin.v); title('v');
subplot(212);
plot(s_kin.tt,[s_kin.d1v s_kin.d1vv]); title('d1v');
%
figure(fig0+4)
subplot(211);
plot(s_kin.tt,[s_kin.kappa s_kin.kappaa]); title('kappa');
subplot(212)
plot(s_kin.x,s_kin.y); title('y(x)');
%
figure(fig0+5)
subplot(311);
plot(s_kin.tt,s_kin.psi); title('psi');
subplot(312);
plot(s_kin.tt,[s_kin.d1psi s_kin.d1psii]); title('d1psi');
subplot(313);
plot(s_kin.tt,[s_kin.d2psi s_kin.d2psii]); title('d2psi');

%
figure(fig0+6)
subplot(211)
plot(s_kin.tt,[s_kin.d2x s_kin.d2xx]); title('d2x');
subplot(212)
plot(s_kin.tt,s_kin.d2xx); title('d2xx');
%
figure(fig0+7)
subplot(211)
plot(s_kin.tt,[s_kin.d3x s_kin.d3xx]); title('d3x');
subplot(212)
plot(s_kin.tt,s_kin.d3xx); title('d3xx');
```

```
%
figure(fig0+8)
subplot(211)
plot(s_kin.tt,[s_kin.d2y s_kin.d2yy]); title('d2y');
subplot(212)
plot(s_kin.tt,s_kin.d2yy); title('d2yy');
%
figure(fig0+9)
subplot(211)
plot(s_kin.tt,[s_kin.d3y s_kin.d3yy]); title('d3y');
subplot(212)
plot(s_kin.tt,s_kin.d3yy); title('d3yy');
%
```

## 9.22 precfunxdot.m

```
function xdot=precfunxdot(t,x,F_lR,deltaw)
%Compute right side of vehicle dynamic model

global N_band N_rigobs N_moveobs R_0 R_band F_band J_band
global K_band L_band k_Bl k_Br s_Bl s_Br B_wide
global  K_rigobs  S_rigobs  R_rigobs  D_rigobs  k_moveobs  s_moveobs
r_moveobs d_moveobs v_moveobs
global v_own
global Vers_rigobs Vers_moveobs

%frame global
global s_kin T_samp lambda spp_x spp_y
global c_F l_F c_R l_R m_v I_zz
global sys_appr
global sys_estim
global sys_contr

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);

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);
Sv=c_F*(deltaw-x1-l_F*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;
```

## 9.23 read_syspar.m

```
function retcode=read_syspar
%Read Syspar_File and initiate system parameters for CAS control
sytem

global   fv_own   fN_statobs   fpar_statobs   fN_movobs   fpar_movobs
froad_wide
global     fsys_appr     fsys_estim     fsys_contr     fdeltaw_horizon
fdgfresh_horizon
global flambda_horizon fint_horizon fLTV_horizon

retcode=1;

fid=fopen('Syspar_File.txt','rt');
if fid<0
    fclose('all');
    fprintf('Syspar_File is not reachable\n');
    retcode=-1;
    return;
end;

s=filt_fgetl(fid);
[s1,fv_own,s2]=strread(s,'%s%f%s','delimiter','=; ');
if ~strcmp(s1,'fv_own') | fv_own<=0
    syspar_error('Error in fv_own definition');
    retcode=-1;
    return;
end;

fN_statobs=0;
fpar_statobs=[];
fN_movobs=0;
fpar_movobs=[];
while 1
    s=filt_fgetl(fid);
    if ~ischar(s), break, end;
    if length(s)<4
        syspar_error('Error in sequence');
        retcode=-1;
        return;
    end;
    ss=s(1:5);
    sm=s(1:4);
    sr=s(1:5);
    if strcmp(ss,'fstat')
        [s1,rx,ry,d,s2]=strread(s,'%s%f%f%f%s','delimiter','[, ]');
        fN_statobs=fN_statobs+1;
        fpar_statobs=[fpar_statobs; rx ry d];
    elseif strcmp(sm,'fmov')
        [s1,rx,ry,d,v,s2]=strread(s,'%s%f%f%f%f%s','delimiter','[,
]');
        fN_movobs=fN_movobs+1;
        fpar_movobs=[fpar_movobs; rx ry d v];
    elseif strcmp(sr,'froad')
        [s1,w,l,r,s2]=strread(s,'%s%f%f%f%s','delimiter','[, ]');
        froad_wide=[w l r];
        break
    else
        syspar_error('Error in sequence');
```

```
        retcode=-1;
        return;
    end;
end

s=filt_fgetl(fid);
[s1,p,s2]=strread(s,'%s%f%s','delimiter','=; ');
if ~strcmp(s1,'fsys_appr')
    syspar_error('Error in fsys_appr definition');
    retcode=-1;
    return;
end;
fsys_appr=p;                %1=>approximated_model_in_use

s=filt_fgetl(fid);
[s1,p,s2]=strread(s,'%s%f%s','delimiter','=; ');
if ~strcmp(s1,'fsys_estim')
    syspar_error('Error in fsys_estim definition');
    retcode=-1;
    return;
end;
fsys_estim=p;               %1=>state_estimator_is_running

s=filt_fgetl(fid);
[s1,s2,ss,s3]=strread(s,'%s%s%s%s','delimiter','='; ');
if ~strcmp(s1,'fsys_contr')
    syspar_error('Error in fsys_contr definition');
    retcode=-1;
    return;
end;
if strcmp(ss,'nonlinpred'), sss='nonlinpred';
elseif strcmp(ss,'diffgeom'), sss='diffgeom';
else
    syspar_error('Error  in  fsys_contr  definition,  wrong  controller
type');
    retcode=-1;
    return;
end;
fsys_contr=sss;              %1=>predictive_control_in_use

s=filt_fgetl(fid);
[s1,p,s2]=strread(s,'%s%f%s','delimiter','=; ');
if ~strcmp(s1,'fdeltaw_horizon')
    syspar_error('Error in fdeltaw_horizon definition');
    retcode=-1;
    return;
end;
fdeltaw_horizon=p;                                  %1=>u(1)_is_deltaw,
0=>u(1)_is_Sv_transversal

s=filt_fgetl(fid);
[s1,p,s2]=strread(s,'%s%f%s','delimiter','=; ');
if ~strcmp(s1,'fdgfresh_horizon')
    syspar_error('Error in fdgfresh_horizon definition');
    retcode=-1;
    return;
end;
fdgfresh_horizon=p;                  %1=>uN_by_diffgeom,  0=>xNp1:=0,
2=>uN:=uNm1

s=filt_fgetl(fid);
```

```
[s1,p,s2]=strread(s,'%s%f%s','delimiter','=; ');
if ~strcmp(s1,'flambda_horizon')
    syspar_error('Error in flambda_horizon definition');
    retcode=-1;
    return;
end;
flambda_horizon=p;
%lambda_weights_u_or_deltau_in_cost_function

s=filt_fgetl(fid);
[s1,p,s2]=strread(s,'%s%f%s','delimiter','=; ');
if ~strcmp(s1,'fint_horizon')
    syspar_error('Error in fint_horizon definition');
    retcode=-1;
    return;
end;
fint_horizon=p;             %1=>integrator_in_RHC_controller

s=filt_fgetl(fid);
[s1,p,s2]=strread(s,'%s%f%s','delimiter','=; ');
if ~strcmp(s1,'fLTV_horizon')
    syspar_error('Error in fLTV_horizon definition');
    retcode=-1;
    return;
end;
fLTV_horizon=p;             %1=>LTV_linearization_in_the_horizons

fclose(fid);
return;

%------------------------------
function syspar_error(ss)
%Send syspar error and return
fclose('all');
fprintf('Syspar_error=%s\n',ss);
return;

%------------------------------
function s=filt_fgetl(fid)
%Remove comments from text file
while 1
    s=fgetl(fid);
    if ~ischar(s), return, end;
    if strcmp(s(1),'%')
        ;
    else
        k=strfind(s,'%');
        if ~isempty(k)
            s=s(1:k(1)-1);
        end;
        break;
    end;
end;
return
%------------------------------
```

## 9.24 rhc2_controller.m

```
function [xdot,F_lR,deltaw,Sv]=rhc2_controller(x0)
%Compute control for 'nlinpred' (Receding Horizon Control)

%-----------------------------------------------------
%Modified: 2007.03.16. (print of t0=... eliminated)
%Modified: 2007.06.20. (corrections if u(1)=deltaw)
%-----------------------------------------------------

global N_band N_rigobs N_moveobs R_0 R_band F_band J_band
global K_band L_band k_Bl k_Br s_Bl s_Br B_wide
global  K_rigobs  S_rigobs  R_rigobs  D_rigobs  k_moveobs  s_moveobs
r_moveobs d_moveobs v_moveobs
global v_own
global Vers_rigobs Vers_moveobs

%frame global
global s_kin T_samp lambda spp_x spp_y spp_psi spp_d1psi spp_v
global c_F l_F c_R l_R m_v I_zz
global sys_appr
global sys_estim
global sys_contr

%Predictive control
global  N_horizon  t0_horizon  xx_horizon  uu_horizon  deltaw_horizon
dgfresh_horizon ulast_horizon
global lambda_horizon int_horizon LTV_horizon

xx=xx_horizon;
uu=uu_horizon;
N=N_horizon;
t0=t0_horizon;
ulast=ulast_horizon;
inthor=int_horizon;
lambda=lambda_horizon;
LTVhor=LTV_horizon;
if nargin==0, x0=xx(1,:)'; end; %No estimation

%----------------------
%fprintf('t0=%g\n',t0);
%----------------------

% %uu->xx
% xx=uux02xx(uu,x0,t0,deltaw_horizon,T_samp);
% xx_horizon=xx;

tN=t0+N*T_samp;
tt=[t0:T_samp:tN]';
xxd=funxdvec(tt);
C=[0 0 0 0 1 0; 0 0 0 0 0 1];
%C=[0 0 0 0 1 0; 0 0 0 0 0 1; 0 1 0 0 0 0];
%C=eye(6);
yyd=(C*xxd')';
yy=(C*xx')';
ee=yyd-yy;
EE=ee(2:N+1,:);
%
[AA,BB]=LTV2vehicle(xx,uu,T_samp,LTVhor,deltaw_horizon);
[P1,H1,P2,H2,mvec,eN]=ab2ph(AA,BB,C,EE,inthor);
```

```
delta_x0=x0-xx(1,:)';
%
my=size(C,1);
ru=size(uu,2);
nx=size(xx,2);
nI=N*ru;
%
if inthor~=0
    delta_x0=[delta_x0; zeros(ru,1)];
end;
%
L1inv=pinv(H1'*H1+lambda*eye(nI));
Lmuinv=pinv(H2*L1inv*H2');
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);
delta_uut(:)=delta_U;
delta_uu=delta_uut';
%
if inthor~=0
    DELTAuu=[ulast'; delta_uu];
    DELTAuusum=cumsum(DELTAuu);
    delta_uu=DELTAuusum(2:N+1,:);
    uupred=delta_uu;
else
    uupred=uu+delta_uu;
end;
%
u0=uupred(1,:)';
ulastnew=u0;
%
if ~deltaw_horizon
    Sv0=u0(1);
    F_lR0=u0(2);
    deltaw0=Sv2deltaw(Sv0,x0);
else
    deltaw0=u0(1);
    F_lR0=u0(2);
    Sv0=deltaw2Sv(deltaw0,x0);
end;

%Pepare next horizon
t0=t0_horizon;
xxnext=uux02xx(uupred,x0,t0,deltaw_horizon,T_samp);
uunext=[uupred(2:N,:); NaN*ones(1,2)];
tN=t0+N*T_samp;
xN=xxnext(N+1,:)';
%
if dgfresh_horizon==1   %uN by diffgeom
    [xdotN,F_lRN,deltawN,SvN]=dg_controller(tN,xN);
elseif dgfresh_horizon==0   %uN makes xNp1 to zero
    Pmat=eye(6);
    tNp1=t0+(N+1)*T_samp;
    xdNp1=funxdvec(tNp1);
    [xdotNaN,F_lRNaN,deltawNaN,SvNaN,fN,GN]=apprfunxdot(tN,xN,0,0);
    fN=xN+T_samp*fN;
    GN=T_samp*GN;
    uN=-pinv(GN'*Pmat*GN)*GN'*Pmat*(fN-xdNp1);
    SvN=uN(1);
    F_lRN=uN(2);
    deltawN=Sv2deltaw(SvN,xN);
```

```
else    %for example -1, in which case uN repeats uNm1
    uN=uupred(N,:)';
    SvN=uN(1);
    F_lRN=uN(2);
    deltawN=Sv2deltaw(SvN,xN);
end;
%
if ~deltaw_horizon
    uN=[SvN F_lRN]';
else
    uN=[deltawN F_lRN]';
end;
uunext(N,:)=uN';
%
tNp1=tN+T_samp;
xdotNp1=apprfunxdot(tNp1,xN,F_lRN,deltawN);
xNp1=xN+T_samp*xdotNp1;
% xxnext(N+1,:)=xNp1';
%
xx_horizon=[xxnext(2:N+1,:); xNp1'];
uu_horizon=uunext;
t0_horizon=t0+T_samp;
ulast_horizon=ulastnew;
%
F_lR=F_lR0;
deltaw=deltaw0;
Sv=Sv0;
xdot=apprfunxdot(t0,x0,F_lR,deltaw);
```

## 9.25 stepping.m

```
function stepyy=stepping(tt,yy,Ts)
%Repeating last data if tt(i) is not integer multiple of Ts

%Modified: 2007.03.16.

ntt=length(tt);
myy=size(yy,2);
stepyy=NaN*ones(ntt,myy);

lasty=yy(1,:);
for i=1:ntt
    t=tt(i);
    if rem(t,Ts)==0, lasty=yy(i,:); end;
    stepyy(i,:)=lasty;
end;
```

## 9.26 Sv2deltaw.m

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

%frame global
global s_kin T_samp lambda spp_x spp_y
global c_F l_F c_R l_R m_v I_zz
global sys_appr
global sys_estim
global sys_contr

col=size(x,2);
if col==1, x=x'; 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);

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

## 9.27 uux02xx.m

```
function xx=uux02xx(uu,x0,t0,deltawinput,Tsamp)
%Compute xx tansient belonging to uu and x0

%-----------------------------------------------------
%Modified: 2007.06.20. (corrections if u(1)=deltaw)
%-----------------------------------------------------

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);
    else
        deltawi=ui(1);
        F_lRi=ui(2);
        Svi=deltaw2Sv(deltawi,xi);
    end;
    xdoti=apprfunxdot(ti,xi,F_lRi,deltawi);
    xip1=xi+Tsamp*xdoti;
    xx(i+1,:)=xi';
    xi=xip1;
end;
xx(N+1,:)=xip1';
```

## 9.28 ybar.m

```
function [xt,yt,d1xt,d1yt,d2xt,d2yt]=ybar(t)
%Compute reference signals for nonlinear control based on diff. geom.

global N_band N_rigobs N_moveobs R_0 R_band F_band J_band
global K_band L_band k_Bl k_Br s_Bl s_Br B_wide
global  K_rigobs  S_rigobs  R_rigobs  D_rigobs  k_moveobs  s_moveobs
r_moveobs d_moveobs v_moveobs
global v_own
global Vers_rigobs Vers_moveobs

%frame global
global s_kin T_samp lambda spp_x spp_y
global c_F l_F c_R l_R m_v I_zz
global sys_appr
global sys_estim
global sys_contr

xt=ppval(spp_x.pp,t);
d1xt=ppval(spp_x.ppd1,t);
d2xt=ppval(spp_x.ppd2,t);
yt=ppval(spp_y.pp,t);
d1yt=ppval(spp_y.ppd1,t);
d2yt=ppval(spp_y.ppd2,t);
```

## 9.29 Syspar_File.txt

```
%*************************************************
%Input file for CAS system parameters
%*************************************************
%initband parameters
fv_own=20;                  %Own car average velocity
fstat_obs1=[40 0 2.5];      %[rx ry d] static_obstacle_1
fmov_obs=[120 3.5 4 15];    %[rx ry d v] moving_obstacle
froad_wide=[7 0.75 0.25];   %[total_wide left_portion right_portion]
%funframeCAS parameters
fsys_appr=0;                %1->approximated_model_in_use
fsys_estim=0;               %1->state_estimator_is_running
fsys_contr='nonlinpred';    %1->predictive_control_in_use
fdeltaw_horizon=0;                          %1->u(1)_is_deltaw,   0-
>u(1)_is_Sv_transversal
fdgfresh_horizon=1;             %1->uN_by_diffgeom,  0->xNp1_to_0,  2-
>uN_to_uNm1
flambda_horizon=10;
%lambda_weights_u_or_deltau_in_cost_function
fint_horizon=1;             %1->integrator_in_RHC_controller
fLTV_horizon=1;             %1->LTV_linearization_in_the_horizons
```