**UNIVERSITATEA „SAPIENTIA” DIN CLUJ-NAPOCA**

**FACULTATEA DE ȘTIINȚE TEHNICE ȘI UMANISTE,**

**TÎRGU MUREȘ**

**SPECIALIZAREA AUTOMATICĂ ȘI INFORMATICĂ APLICATĂ**

**CONTROLUL TRAFICULUI URBAN**

**Proiect DE DIplomă**

**Coordonator științific: Absolvent:**

**Prof.dr.ing. Dávid László, Mikló József-Péter**

**Prof.dr.ing. Farkas Csaba**

**2024**

|  |  |  |
| --- | --- | --- |
| UNIVERSITATEA „SAPIENTIA” din CLUJ-NAPOCA **Viza facultății:**  Facultatea de Științe Tehnice și Umaniste din Târgu Mureș  Specializarea: **Automatică și informatică aplicată** | | |
| **LUCRARE DE DIPLOMĂ** | | |
| Coordonator științific:  **ș.l. dr. ing. Nume cadru didactic** | | Candidat: **Mikló József-Péter**  Anul absolvirii: **2024** |
| **a) Tema lucrării de licență:**  MODELAREA SI CONTROLAREA TRAFICULUI URBAN  **b) Problemele principale tratate:**  - Studiu bibliografic privind sistemele de reglare.  - Realizarea unei aplicații pentru simularea procesului studiat.  - Clasificarea metodelor de reglare  **c) Desene obligatorii:**  - Schema bloc al aplicației  - Diagrame UML privind software-ul realizat.  **d) Softuri obligatorii:**  -Aplicație  **e) Bibliografia recomandată:**  **-** Márton Lőrinc, Irányítsátechnika, Scientia, 2009  - Dávid László, Tehnici de optimizare: metode numerice de calcul în tehnica reglării optimale, Editura Universității Petru Maior, Tg. Mureș, 2000, | | |
| **f) Termene obligatorii de consultații: săptămânal**  **g) Locul și durata practicii:** Universitatea „Sapientia” din Cluj-Napoca,  Facultatea de Științe Tehnice și Umaniste din Târgu Mureș  Primit tema la data de: 10.05.2021  Termen de predare: 27.06.2022 | | |
| Semnătura Director Departament  Semnătura responsabilului  programului de studiu | Semnătura coordonatorului  Semnătura candidatului | |

**Declarație**

Subsemnata/ul ............................................................., absolvent(ă) al/a specializării …………………………………………………………., promoția………… cunoscând prevederile Legii Educației Naționale 1/2011 și a Codului de etică și deontologie profesională a Universității Sapientia cu privire la furt intelectual declar pe propria răspundere că prezenta lucrare de licență/proiect de diplomă/disertație se bazează pe activitatea personală, cercetarea/proiectarea este efectuată de mine, informațiile și datele preluate din literatura de specialitate sunt citate în mod corespunzător.

Localitatea,

Data: Absolvent

Semnătura………………………

Ide kerül a Turnitin similarity report

Controlul traficului urban

Extras

A dolgozat 1 oldalas kivonata román nyelven (Times New Roman betűtípus 1,5 sorköz) .

**SAPIENTIA ERDÉLYI MAGYAR TUDOMÁNYEGYETEM**

**MAROSVÁSÁRHELYI KAR**

**AUTOMATIKA ÉS ALKALMAZOTT INFORMATIKA SZAK**

**VÁROSI FORGALOMIRÁNYÍTÁS**

**DIPLOMADOLGOZAT**

**Témavezető: Végzős hallgató:**

**Dr. Dávid László, egyetemi tanár Mikló József-Péter**

**Dr. Farkas Csaba, egyetemi tanár**

**2024**

Kivonat

A dolgozat magyar kivonata 150-200 szó között (Times New Roman betűtípus 1,5 sorköz).

***Kulcsszavak***: amelyek meghatározzák a dolgozat témáját, maximum 5 kulcsszó

Abstract

A dolgozat angol kivonata 150-200 szó között (Times New Roman betűtípus 1,5 sorköz).

***Keywords***: motion detection, motion tracking, biometry

**Tartalomjegyzék**

[1. Bevezető 11](#_Toc168332077)

[2. Elméleti megalapozás és szakirodalmi tanulmány 11](#_Toc168332078)

[2.1. Közlekedési áramlás matematikai modellezése 11](#_Toc168332079)

[2.2. A dinamikus modellek stabilitásvizsgálata 15](#_Toc168332080)

[3. Célkitűzések 19](#_Toc168332081)

[4. Szabályozás szimulációban 20](#_Toc168332082)

[4.1. Szimulációs szoftver 20](#_Toc168332083)

[4.2. TraCI 22](#_Toc168332084)

[4.3. Fuzzy szabályozás 24](#_Toc168332085)

[5. Üzembe helyezés és kísérleti eredmények 32](#_Toc168332086)

[5.1. Üzembe helyezési lépések 32](#_Toc168332087)

[5.2. Felmerült problémák és megoldásaik 32](#_Toc168332088)

[5.3. Kísérleti eredmények, mérések 33](#_Toc168332089)

[6. A rendszer felhasználása 36](#_Toc168332090)

[7. Következtetések 36](#_Toc168332091)

[7.1. Megvalósítások 36](#_Toc168332092)

[7.2. Hasonló rendszerekkel való összehasonlítás 36](#_Toc168332093)

[7.3. Továbbfejlesztési lehetőségek 36](#_Toc168332094)

[8. Irodalomjegyzék 36](#_Toc168332095)

[9. Függelék 37](#_Toc168332096)

**Ábrák jegyzéke**

[2.1.1. ábra – Sebességfüggvény ábrázolása 13](#_Toc168332162)

[2.1.2. ábra – Optimális sebességfüggvény ábrázolása 14](#_Toc168332163)

[4.1.1. ábra – Szcenárió generálás OSM térképen 21](#_Toc168332164)

[4.1.2. ábra – SUMO konfigurációs fájl 22](#_Toc168332165)

[4.2.1. ábra – SUMO szimuláció indítása Python környezetben 23](#_Toc168332166)

[4.2.2. ábra – Sávterület detektor létrehozása 23](#_Toc168332167)

[*4.3.1. ábra – Fuzzy szabályozás lépései* 25](#_Toc168332168)

[4.3.2. ábra – Kereszteződés felosztása szekvenciákra 26](#_Toc168332169)

[4.3.3. ábra – Sávterület detektorok a szimulációban 27](#_Toc168332170)

[4.3.4. ábra – Fuzzy szabályozó osztálydiagramja 28](#_Toc168332171)

[4.3.5. ábra – Bemeneti tagsági függvények ábrázolása 29](#_Toc168332172)

[4.3.6. ábra – Kimeneti tagsági függvények ábrázolása 30](#_Toc168332173)

**Táblázatok jegyzéke**

[1. táblázat: Fuzzy szabályok 31](#_Toc168327364)

# Bevezető

Régen az utcák és az utak passzív infrastrukturák voltak. Jelentős céljuk az volt, hogy gyors és kényelmes vezetést biztosítsanak. Azonban manapság az utak már sok helyen nem felelnek meg ennek a célnak. Az utóbbi évtizedekben jelentősen megnőtt az járművek száma, ami forgalmi dugókhoz és balesetekhez vezetett. Így kezdetben fix idősítésű vezérlőlámpákat, majd később számítógépes programok által vezérelt lámpákat vezettek be.[1]

A forgalmi dugókkal kapcsolatos problémák enyhítésére különböző megoldásokkal is

próbálkoznak: új utak építése, út díjak kivetése, tömegközlekedés előmozdítása, vagy a

meglévő infrastruktúra hatékonyabb kihasználása.

Az intelligens városi közlekedés területén számos elméleti és technológiai innováció és

alkalmazás tanúja voltunk, köztük a forgalmi irányítás, amit az intelligens közlekedés

koronájának tartanak. Ez kulcsfontosságú intézkedésként szolgált a forgalmi dugók

enyhítésére és a forgalmi problémák megoldására. Ennek eredményeként olyan fejlett

forgalmi jel irányítási rendszerek jelentek meg, mint például a SCOOT, SCATS és a

modellezés-alapú algoritmusok, adatvezérelt algoritmusok és mesterséges intelligencia

alapú kiváló forgalmi irányítási algoritmusok, amelyek szinte egy évszázad fejlesztésének

során jöttek létre, és támogatták az urbanizáció gyors fejlődését.[2]

A városi hálózatok forgalomirányító felépítésük szerint a következő kategóriákba sorolhatók: centrális, elosztott (decentralizált) és vegyes. A centrális forgalomirányító esetében minden döntést egy központi gép hoz, amit később továbbít a terepi berendezéseknek. Az elosztott architektúra eseténél központi gép nélkül van megvalósítva az irányítás, a terepi gépek elosztják egymás között a számításokat. Az elosztott és a vegyes irányítási architektúrák kevésbé elterjedtek, a centrális architektúrához viszonyítva. Viszont az utóbbi két architektúra előnye, hogy nem áll fent a központi gépről való leszakadás veszélye és nagyobb biztonsági üzemelés valósítható meg velük. Alkalmazási példaként az elosztott irányításási rendszerekre megemlíthető az ausztráliai SCATS és az Európában működö Utopia [3].

# Elméleti megalapozás és szakirodalmi tanulmány

## Közlekedési áramlás matematikai modellezése

A közlekedési dinamika egyik legjelentősebb problémája a forgalmi torlódások kialakulása. A torlódásokat balesetek, közlekedési lámpák vagy az utak túlterhelése okozhatja, amik instabilitást hoznak a rendszerbe. Ezek tanulmányozására egy dinamikus rendszermodellt vezetünk be, ami a közlekedési áramlást dinamikusan modellezi.

Vegyünk egy egyszerű modellt, amit M. Bandoés társai (1995) cikkében találunk OVM (Optimal Velocity Model) néven [4]. Ebben a modellben nem vesszük figyelembe a járművek hosszúságát és a vezetők jellemét sem. Tehát minden autóvezető ugyanolyan érzékenységgel vezet. Feltételezzük, hogy minden jármű a legális V sebességgel halad és reagál az előtte lévő jármű távolságára. Így fékezéssel vagy gázolással szabályozhatjuk a jármű gyorsulását a következő képlet alapján:

|  |  |  |
| --- | --- | --- |
|  |  | (1) |

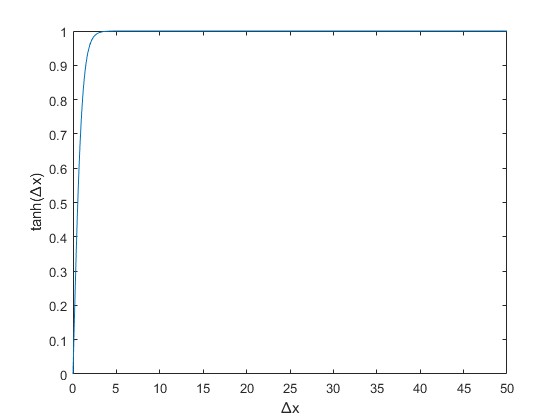
ahol jelöli az n-dik autó gyorsulását, -val jelöltük a vezető érzékenységét , , jelöli az (n+1)-dik és az n-dik jármű közötti távolságot, jelöli a sebességfüggvényt és pedig az n-dik jármű sebességét.

A sebességfüggvény a következő egszerü képlettel adhatjuk meg:

|  |  |  |
| --- | --- | --- |
|  |  | (2) |

ahol a járművek közötti távolságot jelöli.

A következő ábrán láthatjuk a sebességfüggvény ábrázolását, ahogyan a változik 0-tól 50-ig. Megfigyelhetjük, hogy a függvény monoton növekvő és 1-ben van a felső korlátja.



.. ábra – Sebességfüggvény ábrázolása

Egy következő modellt is vegyünk szemügyre, amit Rui Jiang és munkatársai (2001) foglaltak össze és nevezték FVDM(Full Velocity Difference Model) modellnek [5]. Ebben a modellben figyelembe veszik mind a pozitív, mind a negatív sebességkülönbségeket és határozzák meg az autó gyorsulását. Ezt a modellt a következő alakban írhatjuk le:

|  |  |  |
| --- | --- | --- |
|  |  | (3) |

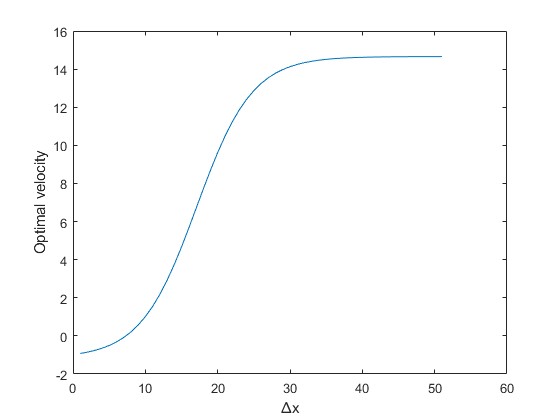
ahol paraméterek megfelelnek az (1) képletben lévő paramétereknek, és még kiegészülnek a következőkkel: jelöli az n-dik jarmű sebességét, jelöli az (n+1)-dik és az n-dik jármű sebességkülönbségét, a vezető érzékenységi együtthatója a sebességkülönbséghez , V(.) az optimális sebességfüggvény.

Az optimális sebességfüggvényt a Dirk Helbing és Benno Tilch által javasolt alakban írhatjuk fel [7]:

|  |  |  |
| --- | --- | --- |
|  |  | (4) |

ahol jelöli a már fentebb említett két autó közötti távolságot, jelöli a vezető autó hosszát, egy eltolást, míg egy skálázást jelöl, a távolság együtthatója és -vel állítjuk be a megfelelő paramétert a tangh() függvénynek.

Tekintsük meg a 2.1.2. ábrát, ami szemlélteti az optimális sebességfüggvényt függvényében. A használt paraméterek Stuttgart forgalmára voltak jellemzőek, amiket a következőképpen választottak meg:



.. ábra – Optimális sebességfüggvény ábrázolása

Megfigyelhetjük, hogy ezen az ábrán már egy teljes tangens hiperbolikus függvény látható eltolva és skálázva, ami már valóságosabban ábrázolja a sebességet, mint a Bando és társai cikkében [4] leírt sebesség, ahol csak a két jármű közötti távolságot vették figyelembe.

A FVDM modellnek egy továbbfejlesztett változatát készítették el Shaowei Yu és társai(2012), amit FVDAM (Full Velocity Difference and Acceleration Model) modellnek neveztek [6]. A FVDM és a FVDAM közötti lényeges különbség az, hogy a FVDAM modell figyelembe veszi a vezető (előtte levő) autó gyorsulását,tehát ha az aktuális autó sebessége nagyobb az előtte lévőnél, viszont az előtte lévő gyorsabb, akkor nem fog rögtön fékezni a követő autó, még ha biztonságos követési távolságot meg is haladja.

|  |  |  |
| --- | --- | --- |
|  |  | (5) |

ahol a jelölések megegyeznek az (1) és (2) egyenletekben használt paraméterekkel, kiegészítve -vel, ami jelöli az (n+1)-dik jármű (vezető) gyorsulásá, k pedig követő autó érzékenységi együtthatóját jelöli, . Ha k=0 akkor a modell FVDM modell, ha k>0 akkor FVDAM modellel dolgozunk.

Ugyamcsak a FVDM modellből kiindulva Jing Zhang és munkatársai (2019) létrehoztak egy modellt, ami előrejelzi az vezető jármű viselkedését [8]. A modellt az (6) egyenlet mutatja be:

|  |  |  |
| --- | --- | --- |
|  |  | (6) |

Az (6) egyenletet Taylor sorbafejtéssel leegyszerüsíthetjük a következőképpen: , amiből kapjuk a (7) egyenletet:

|  |  |  |
| --- | --- | --- |
|  |  | (7) |

ahol a távolságok, sebességek és sebességkülönbségek jelölése megegyezik a fent említett modellek jelölésével. Bővitésként megjelenik a ami egy erősségi együttható és egy prediktív időtartam.

A bemutatott négy modell közül a FVDAM modellt választottuk a közlekedési áramlás modellezésére, mert ez a modell nemcsak a sebességkülönbséget, hanem még az vezető jármű gyorsaságát is figyelembe veszi a modellezésben.

## A dinamikus modellek stabilitásvizsgálata

A dinamikus modell meghatározása után kulcsfontosságú a modell stabilitásának vizsgálata. A stabilitás az egyik alapvető tényező a modell megbízhatósága és hatékonysága szempontjából. Ha a modell instabil, akkor a rendszerben váratlan és kiszámíthatatlan jelenségek is történhetnek, amik veszélyeztethetik a közlekedést és növelhetik a balesetek kockázatát.

A nemlineáris rendszerek stabilitás vizsgálatára a Popov kritérium mellett a direkt és az indirekt Lyapunov módszerek a legelterjedtebbek. A direkt Lyapunov módszer a dinamikus rendszerhez rendel egy Lyapunov függvénynek nevezett energiafüggvényt, majd az energiafüggvény változásából von le következtetéseket a rendszer stabilitásáról. Legyen egy nemlineáris dinamikus modell és =0 minden –ra. Ha létezik egy olyan Lyapunov függvény, amely deriválható egyensúlyi pont körül és az alábbi feltételeket teljesíti, akkor beszélhetünk stabil rendszerről:

1. pozitív definit: és
2. negatív szemidefinit:
3. negatív definit:

Ha az 1. és a 2. feltétel teljesül, akkor egyensúlyi pontja a rendszernek stabil és ha az 1. és 3. Feltétel teljesül akkor egyensúlyi pont aszimptotikusan stabil.

Az indirekt Lyapunov módszer esetén a nemlineáris rendszert linearizájuk, hogy meghatározzuk a nemlineáris rendszer helyi stabilitását. Vegyük az előbbi példaként vett nemlineáris dinamikus modellt: , =0 minden –ra. Legyen az Jacobi mátrixa, melyet x=0 kezdőpontban értékelünk ki. Minden t-re kapunk egy maradékot, amit a következőképpen írhatunk le: . Mivel a maradék nem biztos, hogy egyenletesen közelít a 0-hoz, ezért egy erősebb feltételre van szükség: . Ha ez az egyenlet teljesül, akkor rendszer az egyenletes linearizálása az eredeti rendszernek az origó körül. Amikor a linearizáció létezik akkor annak a stabilitása meghatározza az eredeti nemlineáris rendszer helyi stabilitását.

Azért hogy összehasonlítsuk a szakirodalomban lévő modellek stabilitásávizsgálatával mi az indirekt Lyapunov elvet fogjuk alkalmazni a stabilitásvizsgálatra. Ha a linearizált modell stabil, akkor a nemlineáris is, csak nem tudjuk meghatározni, hogy milyen környezetben fog stabil maradni.

Először is vizsgáljuk meg az első, (1) egyenletben meghatározott modell lineáris stabilitását. Legyen az egyenletes állandósúlt állapotbeli áramlás a következő:

|  |  |  |
| --- | --- | --- |
|  | , | (8) |

ahol b jelöli a két jármű közötti állandó távolságot, n=0,1,...N, ahol N jelöli az összes járművek számát, L jelöli az út hosszát és V(b) az optimális sebességet jelöli. Ahhoz hogy a (8) egyenletet linearizáljuk szükséges alkalmaznunk a perturbáció módszerét, ahol egy kis zavar amit hozzáaudunk a stabil egyenlethez.

|  |  |  |
| --- | --- | --- |
|  |  | (9) |

A (8) és (9) egyenletet behelyettesítve az (1) egyenletbe, majd leegyszerüsítve és megkapjuk a következő összefüggést:

|  |  |  |
| --- | --- | --- |
|  |  | (10) |

Majd ezek után Taylor sorba fejtve az (10) egyenlet leegyszerüsödik a következőképpen:

|  |  |  |
| --- | --- | --- |
|  |  | (11) |

Fourier transzformáció után a köetkezővel helyettesítve: , a deriváltak pedig: , , amit behelyettesítve a (11) egyenletbe majd az egész egyenletet elosztva -el a következő összefüggést kapjuk:

|  |  |  |
| --- | --- | --- |
|  |  | (12) |

ahol z egy komplex számot jelöl,

!!!??? és

|  |  |  |
| --- | --- | --- |
|  |  | (13) |

Azért kell Fourier transzformáltal számoljunk, mert egy periódikus jelről beszélünk. A Laplace transzformált esetében egy kezdőpontból indulunk ki (integral 0->végtelen), míg a Fourier transzformáltat periódikus jelek vizsgálatára alkalmazzuk és itt nincs kezdőpont, tehát az integrál –végtelentől -> +végtelenig tart.

Hasonlatosképpen vizsgáljuk meg a FVDM modell lineáris stabilitását és hasonlítsuk össze miben tér el az előzőtől. A (8)-(9) egyenletek nem változnak, viszont a (10) egyenlet a következőképpen alakul: , majd Taylor sorbafejtés után a (11) egyenlet helyett a következőt kapjuk:

|  |  |  |
| --- | --- | --- |
|  |  | (14) |

Előzőhöz képest még bejött tag az egyenletbe. A (12) egyenlet pedig a következőképpen változik meg:

|  |  |  |
| --- | --- | --- |
|  |  | (15) |

Az előbbi modellhez képest itt még bejött egy tag az egyenletbe. Behelyettesítve és a (15) egyenletbe kapjuk a következő egyenletet:

|  |  |  |
| --- | --- | --- |
|  |  | (16) |

Innen pedig megkapjuk a stabilitási feltételt:

|  |  |  |
| --- | --- | --- |
|  |  | (17) |

Itt a stabilitáshoz már hozzájárul a is, ami az előző modellben még nem jelentm meg.

A FVDAM modell stabilitásvizsgálata is eltér néhány helyen a FVDM-hez képest. A (8)-(9) egyenletek ebben az esetben is megmaradnak, a (10) egyenlet pedig a következőképpen alakul: , Taylor sorbafejtés után pedig:

|  |  |  |
| --- | --- | --- |
|  |  | (18) |

Látható, hogy itt már bejött két plusz tag az első modellhez képest és egy plusz tag a FVDM modellhez képest (). behelyettesítve a modellbe a következőképpen alakul az egyenlet:

|  |  |  |
| --- | --- | --- |
|  |  | (19) |

A FVDM modellhez viszonyítva a meg lett szorozva még -val.

Behelyettesítve és a (19) egyenletbe kapjuk a következő egyenletet:

|  |  |  |
| --- | --- | --- |
|  |  | (20) |

Innen pedig megkapjuk a semleges stabilitási feltételt:

|  |  |  |
| --- | --- | --- |
|  |  | (21) |

Itt már teljesen megváltozik a stabilitási feltétel, de k=0-ra a FVDAM leegyszerüsödik FVDM modellre.

Végül pedig vizsgáljuk meg a Jing Zhang és munkatársai által bemutatott modellt, amit a (6) egyenlet mutatott be. Az állandósult állapotbeli áramlás (8) és a perturbációs módszer (9) egyenletek ebben az esetben sem változnak. A (10) egyenlet itt a következőképpen alakul: , majd Taylor sorba fejtés után a következőképpen alakul az egyenlet:

|  |  |  |
| --- | --- | --- |
|  |  | (22) |

Jing Zhang és munkatársai cikkét tanulmányozva egy hibára bukkantunk. A (22) egyenlet náluk helytelenül szerepel: helyett szerepel, amivel a későbbi eredményeket sem tudjuk megkapni.

behelyettesítve a (22) egyenletbe következőt összefüggést kapjuk:

|  |  |  |
| --- | --- | --- |
|  |  | (23) |

FVDM modelnél a (15) egyenlettel összehasonlítva ebben az egyenletben az még meg van szorozva -vel.

és behelyettesítve a (23) egyenletbe megkapjuk a stabilitási feltételt, ami ebben az esetben a követk

ező:

|  |  |  |
| --- | --- | --- |
|  |  | (24) |

# Célkitűzések

A városok forgalmasak helyein napközben (munkába, iskolába menetkor vagy jövetkor) torlódások alakulnak ki a sok járművek miatt. Ilyenkor az utak kapacitása meghaladja a járművek számát. Ezekre a torlódásokra a jelzőlámpák nincsenek megfelelően szabályozva. A jelzőlámpák fő célja az lenne, hogy a torlódásokat enyhítse. megfelelően irányítsa a forgalmat és minél több járművet engedjen át egy időegység alatt.

Ezen problémák enyhítésére egy olyan rendszert szeretnénk létrehozni, ami megfelelően irányítja a forgalmat nagy forgalom esetén is és nyugodtabb környezetben is. Egy kereszteződésben ha torlódások alakulnak ki, akkor növeljük annak a jelzőlámpának a zöld idejét, ahol sok autó áll sorban és ezzel egyszerre csökkentjük a többi jelzőlámpa zöld idejét. Éjszakai forgalom esetén, amikor viszonylag kevés jármű közlekedik akkor az lenne a cél, hogy az autók ne álljanak feleslegesen a jelzőlámpák előtt, ha a kereszteződésben egyetlen jármű sem közlekedik. Ilyenkor a rendszer vegye észre a közeledő járművet és úgy állítsa be a jelzőlámpákat, hogy a közeledő jármű zöld jelzést kapjon.

Egy másik fontos célja a dolgozatnak, hogy olyan útvonalakat valósítsunk meg, ahol megfelelő sebességgel haladva az utasok végig zöld jelzéseket kapnak és nem kell várakozniuk a jelzőlámpák előtt. Ezeket az útvonalakat zöldhullámnak nevezzük és nagyon segít a torlódások elkerülésében, illetve a környezetet is kevésbé szennyezik így az autók. Ennek a megvalósítására szükséges az egymást követő jelzőlámpák összehangolása.

A következő cél, amit a dolgozatunkba szeretnénk megvalósítani az a forgalom elkerülése. Ez a cél azt jelenti, hogy amikor egy hosszabb útszakaszon torlódások alakulnak ki és ezt előre látjuk, akkor a torlódások előtt lévő jelzőlámpák úgy tereljék a járműveket, hogy azok egy hosszabb útvonalon hamarabb érjenek a céljukhoz, mint ha a nagy forgalomba vették volna az irányt.

Összefoglalva a célkitűzéseink a következőek:

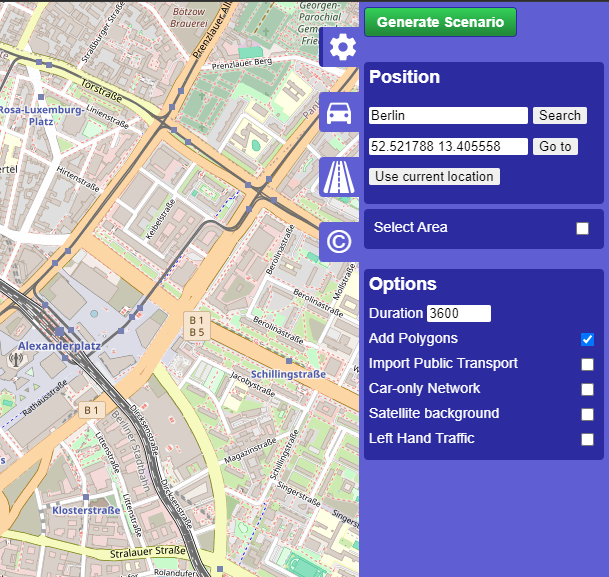
* torlódások elkerülése és enyhítése
* megfelelő szabályozás nagy és kis forgalom esetén is
* zöldhullám kialakítása
* elkerülő utak ütemezése

# Szabályozás szimulációban

## Szimulációs szoftver

Közlekedési hálózatok modellezésére és elemzésére a SUMO, MATSim, VISSIM vagy AIMSUN szoftverek állhatnak rendelkezésre. Mi ezek közül a szimulációt a SUMO (Simulation of Urban MObility) szoftverrel végeztük. Ez egy nyílt forráskódú szimulációs szoftver, mely lehetővé teszi, hogy a felhasználók a saját igényeik,céljaik szerint módosíthatssák a szimulációt. Nagyon sok függvényel és hosszú dokumentációval rendelkezik. Ez a szoftver megengedi a felhasználók számára, hogy különböző környezetet: városi, vidéki területeket vagy autopályákat szimuláljanak. Még egy nagy előnye ennek a szoftvernek, hogy több kiegészítő eszköz és interfész is kapcsolodik a szimulációhoz.

A SUMO telepítése után a “tools” nevezetű mappában az “osmWebWizard.py” scriptet futtatva megjelenik a böngészőben az OpenStreetMap. Itt a felhasználó bármilyen helyre rá tud keresni és könnyedén egy szcenáriót tud generálni, amit a script importál SUMO szimulációban. Még generálás előtt külőnböző paramétereket be lehet állítani, például: szimuláció időtartamát, buszok, teherautók, gyalogosok, vonatok, hajók szimulálását, autók sűrűségét, jobb vagy bal oldali közlekedésmódot stb. A “Generate Scenario” gombra kattintva kigenerálja számunkra a szimulációhoz szükséges fájlokat és elindítja a szimulációs környezetet, ahol már szimulálni tudjuk az alapértelmezett beállításokkal a kiválasztott területet.



4.1.1. ábra – Szcenárió generálás OSM térképen

Az egyik legfontosabbb generált fájl a “.sumocfg”-vel kiterjesztett fájl, amiben a szimuláció konfigurációja, beállítása van leírva. Ebbe a konfigurációs fájlba kell megadni a bemeneti “.xml” fájlokat, amik tartalmaznak információkat az élekről, csomópontokról és ezek kapcsolatairól, utvónalakat, utazási információkat minden autóról. Kiegészítő fájlként alapjáraton egy fájl adott, ami az épületek méretét és dimenzióját tárolja. Ezek után különböző feldolgozási, újratervezési beállításokat lehet megadni, naplózási beállításokat és a grafikus felhasználói felülettel kapcsolatos dolgokat lehet beállítani. Egy generált konfigurációs fájl a következőképpen néz ki:

<?xml version="1.0" encoding="UTF-8"?>

<configuration xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="http://sumo.dlr.de/xsd/sumoConfiguration.xsd">

    <input>

        <net-file value="osm.net.xml.gz"/>

        <route-files value="osm.passenger.trips.xml"/>

        <additional-files value="osm.poly.xml.gz"/>

    </input>

    <processing>

        <ignore-route-errors value="true"/>

    </processing>

    <routing>

        <device.rerouting.adaptation-steps value="18"/>

        <device.rerouting.adaptation-interval value="10"/>

    </routing>

    <report>

        <verbose value="true"/>

        <duration-log.statistics value="true"/>

        <no-step-log value="true"/>

    </report>

    <gui\_only>

        <gui-settings-file value="osm.view.xml"/>

    </gui\_only>

</configuration>

.. ábra – SUMO konfigurációs fájl

A rendezettebb munka érdekében az xml fájlokat egy mappába helyeztük, majd az elérési útvonalakat ennek megfelelően megváltoztattuk a konfigurációs fájlban is.

## TraCI

A TraCI (Traffic Control Interface) egy olyan protokollt biztosít, amely lehetővé teszi, hogy a külső alkalmazások kommunikáljanak a SUMO-val. A TraCI segítségével egy kapcsolatot tudunk létrehozni a SUMO és a Python scriptek között, ahol adatokat tudunk lekérdezni, beállítani, de akár különböző szabályozásokat vagy gépi tanulásokat is lehet implementálni.

Python-ban a TRACI interfész használatához szükségünk van a “traci” könyvtárra. Ennek a könyvtárnak a segítségével tudunk parancsokat kiadni a szimuláció futtatására, a járművek vezérlésére: sebesség, gyorsulás lekérdezése, beállítása, jelzőlámpák adatainak lekérdezése és beállítása, sávterület detektorok beállítása és lekérdezése stb. A szimuláció inditását első lépésben a következőképpen tudjuk a legegyszerübben lekódolni:

import traci

sumoCmd = ["sumo-gui", "-c", "osm.sumocfg"]

traci.start(sumoCmd)

step=0

while traci.simulation.getMinExpectedNumber()>0:

        traci.simulationStep()

        step += 1

traci.close()

.. ábra – SUMO szimuláció indítása Python környezetben

Ebben a kódrészletben importoljuk a “traci” könyvtárat, majd a sumoCmd listában megadjuk a szimuláció indításához szükséges parancsokat. A “sumo-gui” parancsal megadjuk, hogy egy grafikus felhasználói felületet indítson (GUI), a “-c” argumentum arra utal, hogy egy konfigurációs fájlt adunk meg a SUMO-nak, majd végül megadjuk a SUMO konfigurációs fájlt, ami a mi esetünkben az “osm.sumocfg”. Ezt a konfigurációs fájlt a 4.1.2-es ábra szemlélteti.

A “traci.start(sumoCmd)” parancs elindítja a szimulációt a megadott konfigurációs fájlal és létrehozza a TraCI kapcsolatot. Ezek után egy step változót 0 kezdeti értékkel létrehozunk, majd egy while ciklust indítunk, ami addig fut, míg a várhatóan jelen lévő járművek száma 0-nál nagyobb. A “traci.simulationStep()” parancs minden iterációban egy szimulációs lépéssel lépteti a szimulációt és közben mi is növeljük a step változót, amivel nyomon követjük a szimulációs lépést. Végül pedig a “traci.close()” parancsal megszakítjuk a Traci kapcsolatot és lezárjuk a szimulációt.

A jelzőlámpák megfelelő szabályozásához szükségünk van a jelzőlámpáknál kialakult sorok hosszára. Ezekre szolgálnak a sávterület detektorok vagy angol megnevezéssel a lane area detector-ok. Minden egyes sávra külön detektorokat kell megadjunk. Ezeket a detectorokat egy xml fájlban adjuk meg, amit “additional-file”-ként hozzáadunk a konfigurációs fájlhoz is.

<?xml version="1.0" encoding="UTF-8"?>

<additional>

   <laneAreaDetector id="laneAreaDetector1" lanes="-948993200#1\_0" pos="0" endPos="39.7" file="output.xlsx" tl="cluster\_1936414352\_1936414379\_26003429\_7516041220\_#2more"/>

</additional>

.. ábra – Sávterület detektor létrehozása

A 4.2.2-es ábrán egy detektor létrehozását láthajuk, ami egy megadott hosszúságú sávon figyeli a járművek számát. A detektor létrehozásához szükséges megadnunk egy id-t, a megfigyelt sávok id-jait (egymás után lévő sávokat össze lehet kötni egy detektorba), a sáv kiinduló és végpontját, ameddig szeretnénk, hogy tartson a detektálás egy kimeneti fájlt és a jelzőlámpa id-ját.

## Fuzzy szabályozás

A Fuzzy logikát 1965-ben Lotfi Zadeh matematikus mutatta be. A Fuzzy logika egy matematikai eszköz a bizonytalanságok kezelésére. A lágy számítástechnika területén fontos szerepet játszik a szavakkal való számítás fogalmának bevezetésével. Lehetőséget biztosít a nyelvi konstrukciók reprezentálására. Nyelvi konstrukciók lehetnek a következőek: nagyon magas, sok, közepes, alacsony, kevés, gyakran, ritkán stb. Általánosan a Fuzzy logika olyan következtetéseket biztosít, ami az emberi gondolkodásmód képességeit alkalmazza. A hagyományos bináris halmazelmélet éles eseményeket ír le, amelyek vagy bekövetkeznek, vagy nem. Ezzel szemben a Fuzzy halmazok képesek modellezni a bizonytalan vagy kétértelmű adatokat. [9]

A Fuzzy szabályozás egy irányítástechnikai módszer, a Fuzzy logika egy gyakorlati alkalmazása, amelyet különösen olyan rendszerek esetében alkalmaznak, ahol a hagyományos matematikai modellezés nehézkes vagy lehetetlen. Ez a módszer az emberi gondolkodás és döntéshozatal egyes aspektusait modellezi. A Fuzzy szabályozás lehetővé teszi a pontos vezérlést olyan rendszerekben is, ahol bizonytalanságok vagy pontatlan információk vannak. A szabályozás lépéseit a 4.3.1-es ábrán láthatjuk.



*4.3.1. ábra – Fuzzy szabályozás lépései*

A Fuzzy szabályozónak egy vagy akár több bemeneti adatot is megadhatunk, amiből a lépések során egy vagy több kimeneti értéket térít vissza. Első lépésként a bemeneti változókat fuzzy halmazokká alakítja. Ez azt jelenti, hogy minden bemeneti adatot hozzárendel egy vagy több tagsági függvényhez és ezek a függvények meghatározzák a bemeneti értékekhez tartozó tagsági fokokat. A tagsági függvények sokfélék lehetnek, de a leggyakrabban használtak a háromszög és a trapéz alakú tagsági függvények.

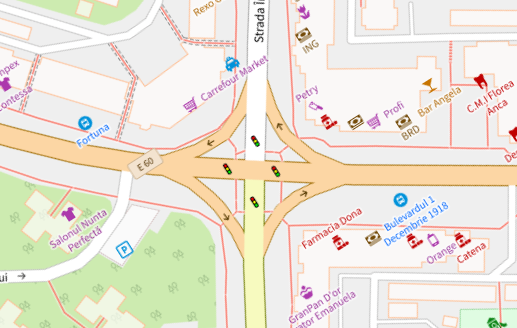
A szabálybázisban megadjuk a szabályokat ami alapján szeretnénk, hogy döntsön a rendszer. Ezek a szabályok emberi gondolkodásmódon alapulnak. Például ha jelzőlámpáknál sorban álló autók hossza az egyik bemenet: “hosszú” és a másik bemenet a változása: “növekszik”, akkor a fuzzifikált kimenetünk a zöld idő “növelése” lehet. Ez a szabályrendszer ha-akkor (if-then) alakban van felépítve.

A következtetés során a szabályrendszer segítéségével megkapjuk a fuzzifikált kimenetet és annak a mértékét. A fuzzifikált kimenetek ugyanugy tagsági függvények, de egyszerübb megvalósításban szingletonok is lehetnek.

A defuzzifikáció az a lépés amikor fuzzifikált kimeneteket defuzzifikáljuk és megkapjuk a pontos kimeneti értéket. A defuzzifikálásra többféle módszer is létezik, ezek közül a legelterjedtebb a súlypont módszer, a területközéppont módszer és a területfelezéses módszer.

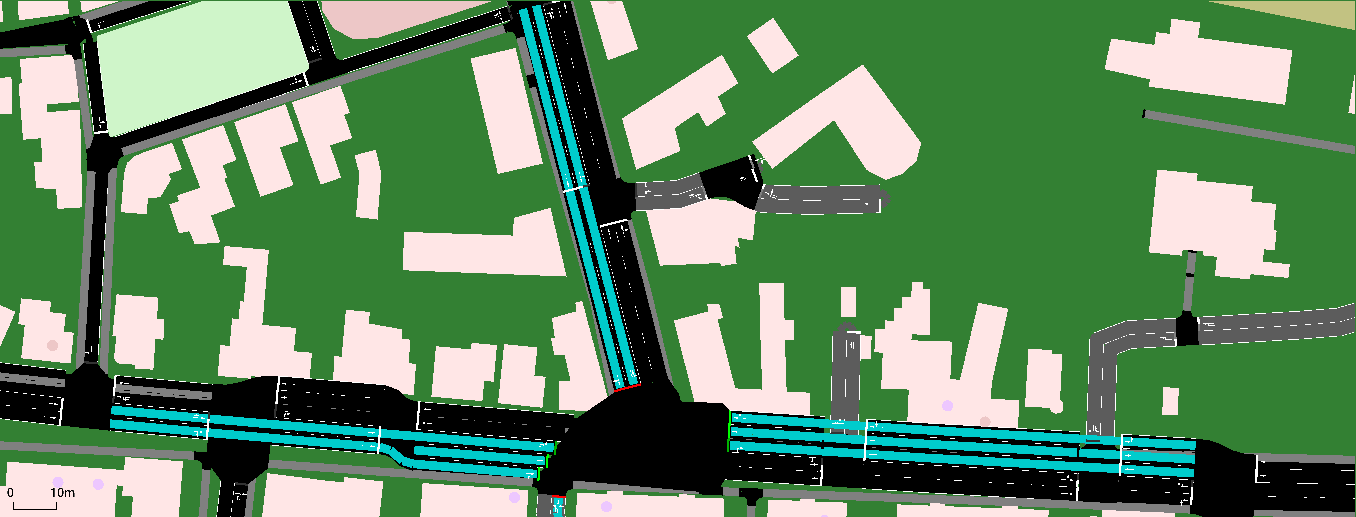
Első lépésben egy kereszteződés szabályozását oldottuk meg fuzzy szabályozóval. Ezt úgy valósítottuk meg, hogy felosztottuk a kereszteződést három vagy négy szekvenciára a kereszteződés bonyolultságának megfelelően. Ha a 4.3.2-es ábrát tekintjük akkor ezt a kereszteződést négy szekvenciára bonthatjuk fel:

1. a vízszintes úton közlekedő járművek kapnak előre és jobbra zöld jelzést
2. a vízszintes úton közlekedő járművek kapnak balra zöld jelzést és a függőleges útszakaszon közlekedő járművek is kapnak jobbra zöld jelzést
3. a függőleges úton közlekedő járművek kapnak előre és jobbra zöld jelzést
4. a függőleges úton közlekedő járművek kapnak balra zöld jelzést és a vízszintes útszakaszon közlekedő járművek is kapnak jobbra zöld jelzést



.. ábra – Kereszteződés felosztása szekvenciákra

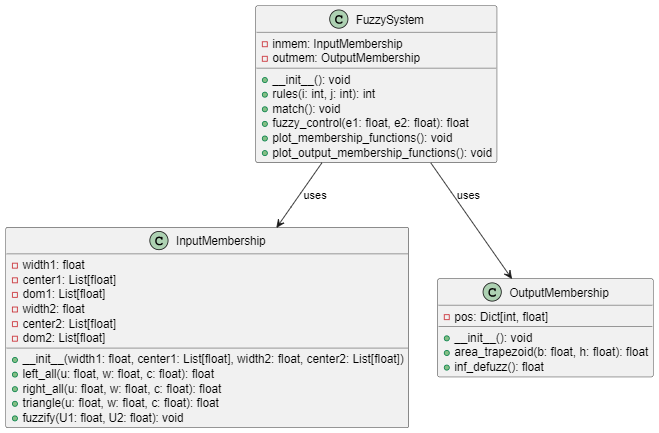
Ezek után minden szekvenciának kezdetben megadtunk egy időtartamot és ezt a szabályozással csökkentjük vagy növeljük a forgalomnak megfelelően. A fuzzy szabályozást úgy írtuk meg, hogy két bemeneti értéket kell megadjunk: a sorra következő szekvenciában lévő várakozó járművek számát és ezeknek a változását. Kimenetként pedig vissatérít egy értéket, ami megmondja, hogy hány másodpercel növeljük vagy csökkentsük az adott szekvencia idejét. Ha növelni kell az adott szekvencia idejét akkor a többi szekvenciától egyenlően elosztva levonja azt az időt, ha pedig csökkenteni kell a szekvenciát akkor megfelelően elosztva hozzáadja azt az időt a többi szekvenciához.



.. ábra – Sávterület detektorok a szimulációban

Ahhoz hogy lekérdezzük a jelzőlámpáknál sorban álló autókat szükséges minden jelzőlámpa előtti sávra detektorokat beépítsünk. Ezeket a 4.2.2-es ábrának megfelelő sablonnal hozzuk létre és a szimulációban egy kék réteg tevődik a sávokra, ahogyan ezt a 4.3.3-as ábrán is láthatjuk. Ezekről a detektorral ellátott sávokról pedig Pythonban a traci könyvtáron keresztül le tudjuk kérdezni a detektált járműveket, amik elengedhetetlenül fontosak a szabályozás céljából.

A fuzzy szabályozást mi egy Python scriptbe objektum orientált programozással valósítottuk meg. Ez a script három osztályt tartalmaz: “FuzzySystem”, “InputMembership” és “OutputMembership”. A kódrészlet osztálydiagramját a következő ábrán szemléltetjük.

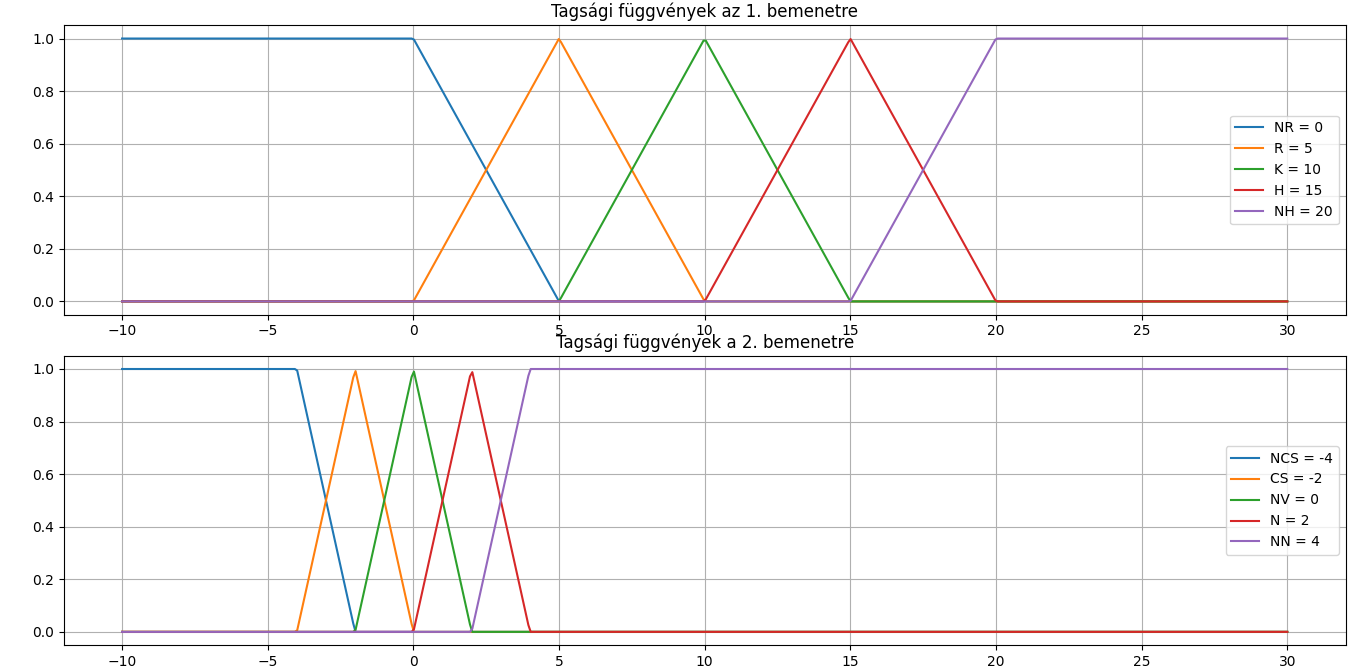


4.3.4. ábra – Fuzzy szabályozó osztálydiagramja

Az “InputMembership” osztály a bemeneti értékek fuzzifikációjával foglalkozik. Hat attribútuma van, ami a következőket jelöli:

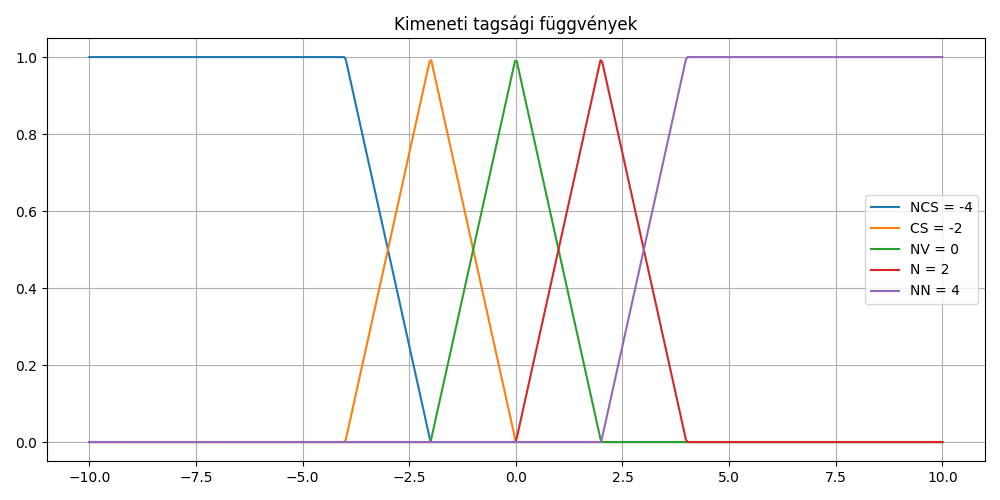
1. “width1”: egy érték, ami az első bemenet tagsági függvényeinek szélessége
2. “center1”:egy vektor, ami az első bemenet tagsági függvényeinek középpontjait tartalmazza
3. “dom1”: egy vektor, ami az első bemenet tagsági függvényekhez tartozás mértékeit adja meg
4. “width2”: egy érték, ami a második bemenet tagsági függvényeinek szélessége
5. “center2”:egy vektor, ami a második bemenet tagsági függvényeinek középpontjait tartalmazza
6. “dom2”: egy vektor, ami az második bemenet tagsági függvényekhez tartozás mértékeit adja meg.

Ennek az osztálynak egy konstruktora van, ami inicializálja az említett attribútumokat. Ezen kívül pedig még négy metódusa van. A “left\_all()”, “right\_all()” olyan metódusok, amik a fuzzifikáció során a bemenetekhez hozzárendelik a szélső tagsági függvényekhez való tartozás mértékét. Ezek a tagsági függvények egy-egy fél trapéznak felelnek meg és a bal, illetve a jobb szélen helyezkednek el. A “triangle()” metódus a háromszög tagsági függvényekhez való tartozás mértékét téríti vissza. Ezeknek a metódusoknak a visszatérített értékeit tároljuk a “dom1” és “dom2” vektorokban. Mindkét bemenetre öt tagsági függvényt határoztunk meg és ezt a 4.3.5-ös ábrán tüntettük fel. Az első bemenet tagsági függvényei tehát a szekvencia hosszára utalnak és a következőképpen jelöltük: “NR” = nagyon rövid, “R” = rövid, “K” = közepes, “H” = hosszú, “NH” = nagyon hosszú. A tagsági függvények középpontjai pedig a 0, 5, 10, 15 és 20-ban helyezkednek el. A második bemenet a szekvencia hosszának a változását jelöli és a tagsági függvényeit a következőképpen definiáltuk: “NCS”=nagyon csökken, “CS”=csökken, “NV”=nem változik, “N”=növekszik, “NN”=nagyon növekszik. A tagsági függvények középpontjai pedig a -4, -2, 0, 2 és 4 – ben helyezkednek el. A “fuzzify()” az “InputMembership” osztály negyedik metódusa az előbbi három metódus segítségével a bemeneti változókat fuzzifikálja vagyis fuzzy halmazokká alakítja.



4.3.5. ábra – Bemeneti tagsági függvények ábrázolása

Az “OutputMembership” osztály a kimeneti tagsági függvények defuzzifikációjával foglalkozó osztály, aminek egy “pos” azonósítójú attribútuma van. Ez az attribútum egy szótár (dictionary) típusú adat, amiben kulcs-érték párokat tárolunk. Ez esetben a kulcsok a kimeneti tagsági függvényeket jelölik: “NCS”=nagyon csökkentjük, “CS”=csökkentjük, “NV”=nem változtatjuk, “N”=növeljük, “NN”=nagyon növeljük. Az értékek pedig a tagsági függvényekhez való tartozást jelölik. A kódrészletben kulcsoknak mi már konkrét értékeket adtunk meg a “NCS”, “CS”, “NV”, “N”, “NN” jelölések helyett, amik a tagsági függvények középpontjai: {-4, -2, 0, 2, 4}. Tehát ha nagyon csökkentjük akkor -4 másodperc és a nagyon növelés esetén 4 másodperc a tagsági függvény. A 4.3.6-os ábrán láthatjuk a kimeneti tagsági függvények ábrázolását.



4.3.6. ábra – Kimeneti tagsági függvények ábrázolása

Az osztálynak egy konstruktora van, ami inicializálja az osztály attribútumát kezdetben 0 értékekkel. Ezen kívül még két metódussal rendelkezik: az “area\_trapezoid()” metódus egy “@staticmethod” dekorátorral ellátott függvény, ami nem függ az osztály példányaitól vagy az osztály belső állapotától, a szerepe az, hogy a levágott háromszögből lett trapéz területét kiszámítsa, úgy hogy adott a trapéz nagy alapja és a magassága; az “inf\_defuzz()” metódus pedig a “pos” attribútumot végigiterálva a területközéppont módszerrel meghatározza a defuzzifikált kimenetet.

A harmadik osztály: “FuzzySystem” a szabálybázis és következtetéssekkel foglalkozó osztály. Két attribútummal rendelkezik:

1. “inmem”: egy “InputMembership” objektumot tárol, tehát a bemeneti osztály példányát
2. “outmem”: egy “OutputMembership” objektumot tárol.

Az osztálynak hat metódusa van, amit a következőkben röviden bemutatunk. A \_\_init\_\_() konstruktor inicializálja az attribútumokat, vagyis létrehoz egy “InputMembership” és egy OutputMembership” objektumot. A “rules()” metódus ugyszintén független az osztály példányától tehát ez is egy “@staticmethod” kulcsszóval van ellátva a megfelelő müködés érdekében. Ez a metódus a szabályokat definiálja. Két paramétere van: egyik a szekvencia hossza (sorban álló járművek száma) és a másik a szekvencia hosszának a változása. A szabályok alapján visszatérít egy értéket, ami megfelel az egyik fent említett kimeneti tagsági függvénynek. A szabályokat az 1. táblázatban adtuk meg, ahol a táblázat első sora az első bemenet tagsági függvényeinek rövidítéseit tartalmazza, az első oszlop pedig a második bemenet tagsági függvényeinek rövidítését tartalmazza. A táblázat többi cellájában pedig a kimeneti tagsági függvények rövidítései láthatóak. Ha ki szeretnénk olvasni néhány értéket a táblázatból akkor a következőképpen tehetjük meg:

* 2. sor 2. oszlop: Ha a szekvencia hossza nagyon rövid (NR) és a változása nagyon csökken (NCS) akkor a nagyon csökkentjük (NCS) a zöld időt
* 5. sor 4. oszlop: Ha a szekvencia hossza közepes (K) és a változása növekszik (N) akkor növeljük (N) a zöld jelzést .

. táblázat: Fuzzy szabályok

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | NR | R | K | H | NH |
| NCS | NCS | NCS | CS | NV | N |
| CS | NCS | CS | NV | N | N |
| NV | CS | NV | N | N | NN |
| N | NV | NV | N | NN | NN |
| NN | N | N | NN | NN | NN |

A “match()” metódus az “InputMembership” osztály “dom1” és “dom2” vektorain két for ciklussal végigiterál. Ha mindkét vektorban az aktuális elem nem nulla akkor azokra a tagsági függvényekre meghívjuk a “rules()” metódust, ami visszatéríti számunkra a kimeneti tagsági függvényt. A kimeneti tagsági függvényhez való tartózás mértékét a két bemeneti tagsági függvényhez tartozás mértékének a minimuma adja meg, vagyis az aktuális “dom1” és “dom2” vektor értékeinek a minimuma. Ezeket az értékeket eltároljuk az “OutputMemory” osztály “pos” attribútumában. Ha az adott szabály többször is létezik akkor mindig a legnagyobb (max) mértékét tároljuk el a “pos” attribútumban. Ezt a technikát nevezik MiniMax technikának a fuzzy következtetések esetében. A következő metódus a “fuzzy\_control()”, ami a paraméterként kapott két bemeneti értékre meghívja a “fuzzify()” metódust. A fuzzifikáció után meghívja az előbb említett “match()” metódust és ezek után meghívja az “inf\_defuzz()” metódust, amit vissza is térít. Ezen kívül még a könnyebb átláthatóság tekintetében beépítettünk egy “plot\_input\_membership\_functions()” metódust, ami ábrázolja a bemeneti tagsági függvényeket és egy “plot\_output\_membership\_functions()” metódust, ami a kimeneti tagsági függvényeket ábrázolja.

# Üzembe helyezés és kísérleti eredmények

## Üzembe helyezési lépések

A szimulációt Matlab környezetben végeztük, hogy bizonyosak legyünk a modell helyes müködéséről.

A közlekedési jelzőlámpa piros és 10 autó várakozik a piros lámpánál 7.4 m követési távolsággal egymástól. A lámpa –ban zöldre vált és az autók elindulnak.

A szimuláció során vizsgáljuk az autók pozicióját (1-2 ábra), sebességét(3-4 ábra) és gyorsulását(5-6 ábra) idő függvényében k=0 (FVDM) és k=0.5 (FVDAM) esetekben.

## Felmerült problémák és megoldásaik

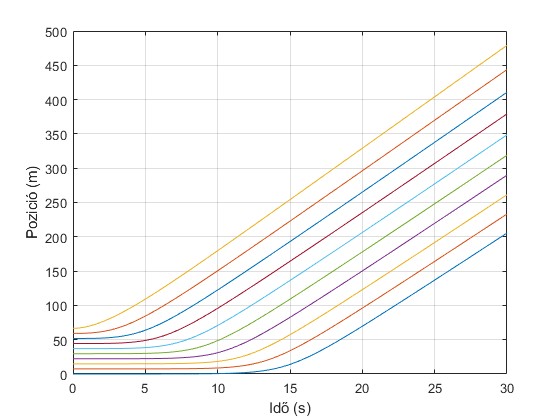
Tavolsag az autok kozt novekszik (headway)

Sebesseg nem all be minden autonal egyformara

## Kísérleti eredmények, mérések

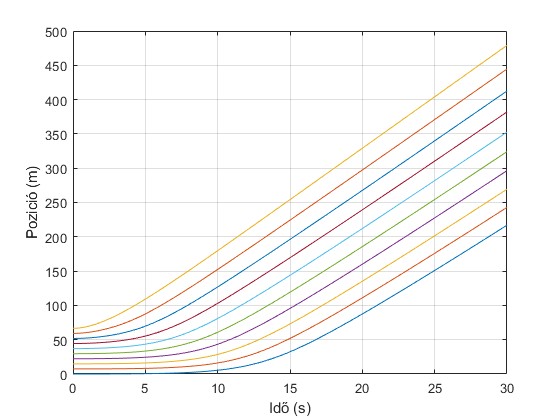
A következő ábrán a szimulációban végzett eredményeket jelenítjük meg. Az 5.3.1. ábra a tíz autó pozícióját szemlélteti idő függvényében a FVDM modell esetében, míg a 5.3.2. ábra a FVDAM modell pozíciókat idő függvényében.

Távolság egyre növekszik az autók között, ahogy telik az idő!!!



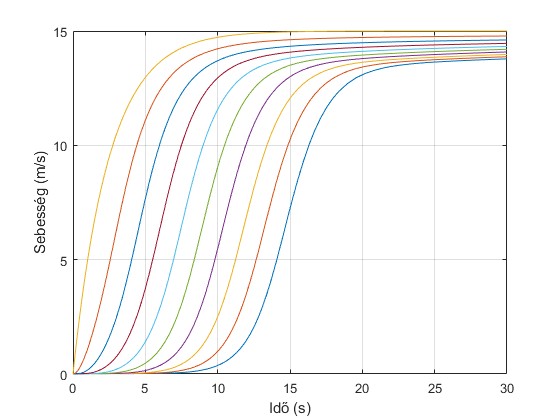
5.3.1. ábra Autók poziciója (FVDM)

+tavolsagok megjelenitese



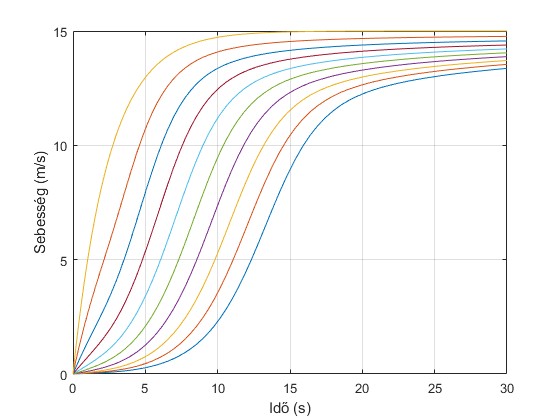
5.3.2. ábra Autók poziciója (FVDAM)

Az 5.3.3-as és az 5.3.4-es ábra a járművek sebességét szemlélteti idő függvényében. Minden járműnek be kellene állnia egy konstans sebességre.



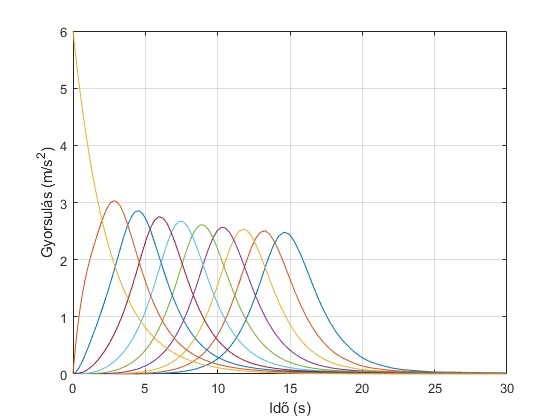
5.3.3. ábra Autók sebessége (FVDM)

Sebesseg egyseges kellene legyen!!! Ezert lesz nagyobb a tavolsag kulonbseg

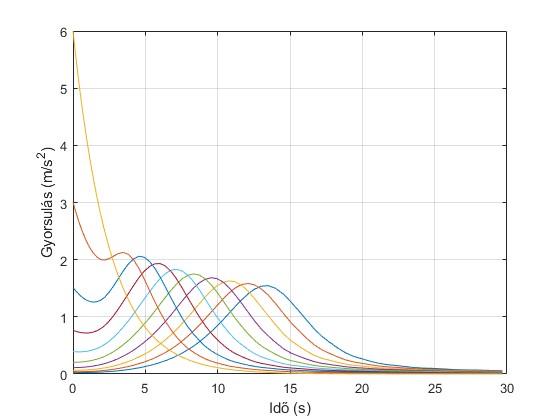


5.3.4. ábra Autók sebessége (FVDAM)

Az 5.3.5. illetve az 5.3.6. ábrák pedig a gyorsulást mutatják be úgszintén idő függvényében. Megfigyelhetjük, hogy minde autó gyorsulása 0-ba tart. Nem fog gyorsulni tovább, tehát beáll egy konstans sebesség.



5.3.5. ábra Autók gyorsulása (FVDM)



5.3.6. ábra Autók gyorsulása (FVDAM)

# A rendszer felhasználása

Amennyiben a rendszer Terjedelem: 2-5 oldal.

# Következtetések

Ebben a fejezetben össze kell foglalni az elvégzett munka végén levont következtetéseket és tapasztalatokat az alábbi al-pontok szerint, de lehet más szempontokat is választani.

## Megvalósítások

## Hasonló rendszerekkel való összehasonlítás

## Továbbfejlesztési lehetőségek

Terjedelem: 1-2 oldal.

# Irodalomjegyzék

1. B. De Schutter, H. Hellendoorn, A. Hegyi, M. van den Berg, and S.K. Zegeye, “Modelbased control of intelligent traffic networks,” Chapter 11 in Intelligent Infrastructures (R.R. Negenborn, Z. Lukszo, and H. Hellendoorn, eds.), vol. 42 of Intelligent Systems, Control and Automation: Science and Engineering, Dordrecht, The Netherlands: Springer, ISBN 978-90- 481-3598-1, pp. 277–310, 2010.
2. Lili Zhang , Qi Zhao , PeiYu , Jing Li , DiYao , XinzheWang , LiWang & Lingyu Zhang, “Research on integrated simulation platform for urban trafc control connecting simulation and practice”, 2022.
3. Tettamanti T., Varga I., “MPC alapú, elosztott városi forgalomirányító rendszer”, 2009.
4. M. Bando and K. Hasebe, A. Nakayama, A. Shibata, Y. Sugiyarna, “Dynamical model of traffic congestion and numerical simulation”, 1995
5. Rui Jiang, Qingsong Wu, Zuojin Zhu, “Full velocity difference model for a car-following theory”, 2001.
6. Shaowei Yu , Qingling Liu, Xiuhai Li, “Full velocity difference and acceleration model for a car-following theory”, 2012.
7. Dirk Helbing and Benno Tilch, “Generalized force model of traffic dynamics”, 1998.
8. Jing Zhang, Bo Wang, Shubin Li, Tao Sun, Tao Wang, “Modeling and application analysis of car-following model with predictive headway variation”, 2019.
9. Javed Alam, Dr. M. K. Pandey, “Development of Traffic Light Control System for Emergency Vehicle Using Fuzzy Logic”, 2012.

# Függelék

**A függelék tartalmazza a forráskódot és dokumentációt tartalmazó adathordozót (pl. CD), ezen kívül pedig bármilyen más kiegészítő anyagot ami nem fért be a dolgozatba. Ezekre hivatkozni kell a dolgozat szövegéből.**

UNIVERSITATEA SAPIENTIA DIN CLUJ-NAPOCA

FACULTATEA DE ȘTIINȚE TEHNICE ȘI UMANISTE, TÎRGU-MUREȘ

SPECIALIZAREA AUTOMATICĂ ȘI INFORMATICĂ APLICATĂ

Vizat decan Vizat director departament

Conf. dr. ing. Domokos József Ș.l. dr. ing Szabó László Zsolt