Lantos Béla

BME Irányítástechnika és Informatika Tanszék

MATLAB PROGRAMFEJLESZTÉS AUTÓPÁLYA HÁLÓZAT IRÁNYÍTÁSÁRA. ALGORITMUS, SZOFTVER ÉS DOKUMENTÁCIÓ

Tanulmány

Készült a RET 1.1 Járműforgalmi rendszerek modellezése és irányítása projekt keretében

Elektronikus Jármű és Járműirányítási Tudásközpont

Budapest, 2006. szeptember

Tartalmi összefoglaló

Autópálya hálózatok makromodelljén alapuló dinamikus modelleket, valamint klasszikus elven és nemlineáris prediktív irányításon alapuló irányítási algoritmusokat dolgoztunk ki irodalmi források felhasználásával. A kifejlesztett módszerek egy részhalmazát megvalósító MATLAB alapú programrendszert fejlesztettünk ki, amely alkalmas általános autópálya hálózat forgalmi viszonyainak szimulációs vizsgálatára és klasszikus szabályozások tervezésére, és lehetőséget ad az irodalomból ismert, de költséges METANET autópálya szimulációs rendszer szolgáltatásainak kiváltására.

Az autópálya hálózat több szakaszból állhat, megengedvén az elágazásokat (bifurcations) és összekapcsolódásokat (junctions) is. Kétféle dinamikus modellt fejlesztettünk ki, az első modell (nem célorientált üzemmód) csak a felhajtók forgalmának jelzőlámpákkal történő irányítását teszi lehetővé (ramp metering control), a második modell (célorientált üzemmód) ennek általánosítása arra az esetre, amikor az irányítás kezeli az OD (origin-destination) információt is, és javaslatot tesz az elágazási helyeken a kedvező útvonalra az alternatív lehetőségek közül a különféle végcélok esetén (VMS=variable message sign, DRIP=dynamic route guidance panel). Mivel a vezetők a VMS jelzéseket nem szükségképpen akceptálják, ezért virtuális járművek rendszerbe injektálásával és a Logit modell elvére épülő útvonalkövetéssel lehetséges a javasolt útvonaltól való eltérés hatásának becslése is.

A kifejlesztett MATLAB alapú szoftver a végső programrendszer 1. verziójának tekinthető. Megvalósítja a kifejlesztett dinamikus modellt a nem-célorientált esetben, lehetővé teszi általános autópálya hálózatok forgalmi viszonyainak meghatározását irányítás nélkül szimuláció keretében, továbbá az autópálya hálózat irányítását egyszerű PID-jellegű irányítási stratégia esetén.

A szimulációs üzemmód biztosítja a forgalomtorlódás okainak és helyeinek felderítését csúcsforgalmi időszakban az autópályán. A szimuláció speciális esetként, konstans bejövő forgalmakat feltételezve a felhajtókon, alkalmas az egyensúlyi állapot (steady state) meghatározására is. Ezáltal biztosítható, hogy az irányítások egyensúlyi állapotból is indíthatók legyenek, és a kezdeti feltétel miatti tranziensek és az irányítási tranziensek ne keveredjenek össze a nemlineáris rendszerben. A szimuláció eredményei alapján megválasztható, hogy mely felhajtó vagy felhajtók esetén célszerű irányítást alkalmazni a forgalmi viszonyok javítása érdekében. Feltételeztük, hogy (az Európában és USA-ban gyakori módon) a felhajtók jelzőlámpákkal vannak ellátva és a forgalmat a felhajtók jelzőlámpáinak átbocsátási ideje révén lehet szabályozni (ramp metering). Az irányítás hatása a forgalom alakulására a tárolt forgalmi tranziensek kiértékelésével elemezhető. A tranziensek dokumentálása grafikusan történik, az eredmények a MATLAB szolgáltatásaival dokumentumokba menthetők.

A forgalmi viszonyok magas szintű és tömör numerikus jellemzésére költségfüggvény szolgál, amely tartalmazza a teljes hálózatban töltött időt (TTS=TTT+TWT, [veh.h]) , a teljes utazási időt (TTT, [veh.h]), a felhajtók várakozási soraiban töltött időt (TWT, [veh.h]) és a beavatkozó jel változásának négyzetösszegét (QDC).

Különös figyelmet fordítottunk az autópálya struktúra definiálásának megkönnyítésére a felhasználó szemszögéből nézve, amely egy mintafájl átírásával végezhető el, amelyben a felhasználót kommentek vezetik. A hálózat struktúráját a program automatikusan olyan adatstruktúrákká konvertálja, amelyek lecsökkentik a valós időben szükséges számításokat, növelve ezáltal a valósidejűség elérésének lehetőségét. Az autópálya szekciók és szegmenseik forgalmi jellemzői (sűrűség, átlagsebesség, folyam), továbbá a felhajtók (várakozási sorok, folyamok, kiszolgálási ráták) és lehajtók (lehajtónkénti folyamok, teljes folyam) forgalmi jellemzői automatikusan kigyűjtésre kerülnek, és felhasználásra kerülnek mind a grafikus megjelenítés során, mind pedig a költségfüggvény számításkor.

A programfejlesztés során kerültük a MATLAB toolboxainak intenzív használatát, amely hosszú távon jó esélyt adhat a MATLAB C-Compiler alkalmazására a későbbi időszakra tervezett végső C-nyelvű implementációhoz.

A tapasztalatok alapján a jövőben folytatható a programrendszer bővítése nemlineáris prediktív irányítással és célorientált üzemmódú irányítással. A program első verziójának kifejlesztésekor a bővítések későbbi beillesztésének igényét figyelembe vettük.

Jelen tanulmány struktúrája úgy van kialakítva, hogy lehetővé teszi felhasználását jegyzet részeként is.

Tartalomjegyzék

1. Célkitűzés

A tanulmány a forgalom makromodelljén alapuló autópálya irányítási algoritmusokkal, valamint és az irányítások tervezéséhez és vizsgálatához szükséges MATLAB alapú programrendszer fejlesztésével és megvalósításával foglalkozik. Az algoritmusok építenek a Lantos: "Autópálya forgalom és jármű irányítások" c. tanulmány eredményeire (Lantos, 2005), és új publikációkra.

Az autópálya irányítások alapja a forgalom átlagos alakulását leíró diszkrétidejű nemlineáris makromodell, amelynek alapesete az elágazások nélküli autópálya szakasz leírására kifejlesztett Payne-féle klasszikus modell. Az autópálya hálózat több szakaszból állhat, megengedvén az elágazásokat (bifurcations) és összekapcsolódásokat (junctions) is. Kétféle modellre készülünk fel autópálya hálózat esetén. Az első modell (nem célorientált üzemmód) csak a felhajtók forgalmának jelzőlámpákkal történő irányítását teszi lehetővé (ramp metering control), és az autópálya hálózat (speciális esetben elágazások nélküli autópálya szakasz) klasszikus elvű vagy prediktív irányításához alkalmazható. A második modell (célorientált üzemmód) ennek általánosítása arra az esetre, amikor az irányítás kezeli az OD (origin-destination) információt is, és javaslatot tesz az elágazási helyeken a kedvező útvonalra az alternatív lehetőségek közül a különféle végcélok esetén (VMS=variable message sign, vagy más néven DRIP=dynamic route guidance panel).

Jelen kutatási szakaszban a cél az algoritmusok megválasztása mindkét esetre, továbbá a MATLAB programrendszer első verziójának kifejlesztése a nem célorientált esetre. A programrendszer első verziója már legyen képes általános autópálya hálózatok forgalmi viszonyainak meghatározására irányítás nélkül szimuláció keretében, továbbá az autópálya hálózat irányítására egyszerű irányítási stratégiák szerint. Az első verzió fejlesztési tapasztalatai megalapozzák a későbbi bővítéseket.

A szimulációs üzemmód biztosítja a forgalomtorlódás okainak és helyeinek felderítését csúcsforgalmi időszakban az autópályán. A szimuláció speciális esetként, konstans bejövő forgalmakat feltételezve a felhajtókon, alkalmas az egyensúlyi állapot (steady state) meghatározására is. Ezáltal biztosítható, hogy az irányítások egyensúlyi helyzetből is indíthatók legyenek. A szimuláció eredményei alapján megválasztható, hogy mely felhajtó vagy felhajtók esetén célszerű irányítást alkalmazni a forgalmi viszonyok javítása érdekében. Az irányítás során feltételezzük, hogy a beavatkozás lehetőségei adottak, vagyis a felhajtók jelzőlámpákkal vannak ellátva, amelyek átbocsátási ideje szabályozható (ramp metering). Az irányítás hatása a forgalom alakulására a tárolt forgalmi tranziensek kiértékelésével elemezhető. A tranziensek dokumentálása grafikusan történik, az eredmények a MATLAB szolgáltatásaival dokumentumokba menthetők.

A forgalmi viszonyok magas szintű és tömör numerikus jellemzésére költségfüggvény szolgál, amely tartalmazza a teljes hálózatban töltött időt (TTS=TTT+TWT, total time spent, [veh.h]) , a teljes utazási időt (TTT, total travel time, [veh.h]), a felhajtók várakozási soraiban töltött időt (TWT, total weighting time at origins, [veh.h]) és a beavatkozó jel változásának négyzetösszegét (QDC, total fuel consumed).

Különös figyelmet fordítunk az autópálya struktúra definiálásának megkönnyítésére a felhasználó szemszögéből nézve, amely egy mintafájl átírásával végezhető el. A hálózat struktúráját a program automatikusan olyan adatstruktúrákká konvertálja, amelyek megőrzik az általánosságot, de a későbbi valósidejű alkalmazás lehetősége érdekében lecsökkentik a valós időben szükséges számításokat, növelve ezáltal a valósidejűség elérésének lehetőségét. Az autópálya szekciók és szegmenseik forgalmi jellemzői (sűrűség, átlagsebesség, folyam), továbbá a felhajtók (várakozási sorok, folyamok, kiszolgálási ráták) és lehajtók (lehajtónkénti folyamok, teljes folyam) forgalmi jellemzői automatikusan kigyűjtésre kerülnek, és felhasználásra kerülnek mind a grafikus megjelenítés során, mind pedig a költségfüggvény számításkor.

A tapasztalatok alapján a jövőben folytatható a programrendszer bővítése nemlineáris prediktív irányítással és célorientált üzemmódú irányítással, továbbá a forgalmi állapotok, torlódási gócok és sebességviszonyok jóslásával mérési eredményekből. A program első verziójának kifejlesztésekor a bővítések későbbi beillesztésének igényét figyelembe vesszük.

2. Autópálya hálózatok dinamikus modellje

Az autópálya irányítások forgalommodellje az átlagos viselkedést leíró diszkrétidejű nemlineáris makromodell. A bemutatandó makromodellek konkrét alakja erősen függ az alkalmazott irányítási elvtől. A nem célorientált irányítás esetén alkalmazott autopálya modell a klasszikus Payne-féle modell (Payne, 1971) egy általánosítása autópálya hálózat esetére. A felhajtó helyek (on-ramps) forgalma jelzőlámpákkal befolyásolható. A modell megengedi elágazások (bifurcations) és összekapcsolódások (junctions) jelenlétét a hálózatban, szemben a klasszikus Payne-féle modellel. A második, célorientált irányítás megvalósítását is megengedő modell a felhajtó helyek forgalmának jelzőlámpával történő befolyásolása (ramp metering) mellett úgy bővíti a beavatkozásokat, hogy üzenetet küld a járművezetőnek, melyik autópálya szakaszt (fővonalat vagy mellékvonalat) válassza a lehetséges alternatívák közül célállomásának függvényében (VMS, variable message sign, vagy más néven DRIP, dynamic route guidance information systems panel). A lehajtók és felhajtók az elágazások és összekapcsolódások speciális esetei. A modellek lehetőséget kínálnak a beavatkozások klasszikus (PID típusú, pl. az ALINEA estén I-típusú), illetve nemlineáris prediktív vagy más optimális elvű megválasztására. Mindkét modellnél lehetőség van a beavatkozások mellett a dinamikus modellek online kiértékeléséből nyerhető forgalom adatokból további javaslatokat levezetni, például az átlagsebesség alakulása alapján javaslatot tenni a megválasztandó sebességre az egyes szakaszokon. Megfelelő számú és típusú mérőhely kialakítása esetén a modellben szereplő forgalomjellemzők és/vagy az aktuális állapot is becsülhető pl. kiterjesztett Kalman-szűrővel. Jelen kutatási szakaszban csak a nem célorientált modell kerül implementálásra.

2.1 Autópálya szakasz Payne-féle klasszikus dinamikus modellje

Payne 1971-ben egy diszkrétidejű másodrendű nemlineáris modellt javasolt a forgalomirányítás dinamikus modelljeként. A modell a hosszirányú változót (x) is diszkretizálja. Az autópálya szakasz N darab egymáshoz csatlakozó $[x_l, x_{l+1}]$ szakaszra (links) van bontva ($x_l < x_{l+1}$). A forgalmat a forgalom sűrűség (traffic density) ρ [veh/km/lane], az átlagos sebesség (mean speed) v [km/h] és a forgalom folyam (traffic flow) q [veh/h] jellemzi, ahol veh a járművek (vehicle) száma, lane a sávok száma. A sávok számát λ fogja jelölni a dinamikus modellben. A szakasz indexe l, a szakaszhoz tartozó forgalmi adatokat a forgalom változók alsó indexe jelöli. A szakasz hossza $L_l = x_{l+1} - x_l$. Az x_l helyen a forgalmi adatok ρ_l, v_l, q_l. A szakaszon lehet felhajtás és lehajtás, amelynek jellemzésére felhajtás esetén a $q_{in,l}$ bejövő folyam, lehajtás esetén a $q_{out,l}$ kimenő folyam szolgál. A $t = kT$ időt a T mintavételi idő ütemében értékeljük ki és zárójelben adjuk meg k értékét.

A dinamikus modellt a forgalom sűrűség, átlagos sebesség és folyam egyenletek írják le:

$$\rho_l(k+1) = \rho_l(k) + \frac{T}{L_l \lambda_l}(q_{in,l}(k) - q_{out,l}(k)) \tag{2.1}$$

$$v_l(k+1) = v_l(k) + \frac{T}{\tau}[V(\rho_l(k)) - v_l(k)] + \frac{T}{L_l} v_l(k)[v_{l-1}(k) - v_l)k)] - \frac{\nu T[\rho_{l+1}(k) - \rho_l(k)]}{\tau L_l[\rho_l(k) + \kappa]} \tag{2.2}$$

$$q_l(k) = \rho_l(k) v_l(k) \lambda_l \tag{2.3}$$

$$V(\rho_l(k)) = v_{f,l} \exp\left[-\frac{1}{a_m} \left(\frac{\rho_l(k)}{\rho_{cr,l}} \right)^{a_m} \right]. \tag{2.4}$$

A (2.2) egyenletben szereplő τ, ν, κ paramétereket a forgalom megfigyeléséből szerzett adatokból kell meghatározni valamilyen identifikációs technikával (például nemlineáris paraméterbecsléssel). A $v_l(k)$ után álló három additív tag rendre az ú.n. relaxációt, konvekciót és anticipációt modellezi. A $V(\cdot)$ függvény a relaxációs tagban azt fejezi ki, hogy az átlagos sebesség minden szekcióban egy sűrűségtől függő $V(\rho_l(k))$ egyensúlyi érték felé tart, amelyben $v_{f,l}$ a szabad haladás sebessége és $\rho_{cr,l}$ a sávonkénti kritikus sűrűség a szekcióban, a_m pedig modell

paraméter, melyeket a forgalom megfigyeléséből szerzett adatokból és identifikációs technika bevonásával lehet meghatározni. A (2.2) és (2.4) egyenletben szereplő paraméterek a forgalmon kívül függenek az autópálya geometriájától, a járművek jellemzőitől és a vezetők viselkedésétől is.

Ha az l-edik szekcióhoz tartozik jelzőlámpával irányítható felhajtó, akkor a jelzőlámpa bevonható a forgalom irányításába a forgalmi viszonyoktól függően. Ha a pillanatnyi forgalmi igény (demand) $D_l(k)$ a jelzőlámpával ellátott felhajtón meghaladja a pillanatnyi kiszolgálható $q_{in,l}(k)$ rátát, akkor egy várakozási sor alakul ki a felhajtón, amelynek $w_l(k)$ hossza és $q_{in,l}(k)$ kiszolgálási rátája a következő stratégia szerint képezhető:

$$w_l(k+1) = w_l(k) + T[D_l(k) - q_{in,l}(k)] \tag{2.5}$$

$$q_{in,l}(k) = r_l(k)\hat{q}_{in,l}(k) = r_l(k)\min\left\{D_l(k) + \frac{w_l(k)}{T}, Q_l \min\left\{1, \frac{\rho_{\max,l} - \rho_l(k)}{\rho_{\max,l} - \rho_{cr,l}}\right\}\right\}. \tag{2.6}$$

A (2.6) egyenletben Q_l [veh/h] a felhajtó maximális kapicitása, $\rho_{\max,l}$ a maximálisan lehetséges forgalom sűrűség a szekcióban és $r_l(k) \in [r_{\min,l}, 1]$ a kiszolgálási arány, amely az irányítási algoritmusban megválasztható beavatkozó jel (metering control). Jól látható, hogy $\hat{q}_{in,l}(k)$ a forgalmi viszonyoktól függően lehet $D_l(k)$, ha nincs várakozási sor és $D_l(k) < Q_l$, vagy Q_l vagy annál kisebb érték.

Vegyük észre, hogy $q_{in,l}(k)$ a (2.1) egyenletben is felhasználásra kerül. Ha van lehajtó az l-edik szekcióban, akkor $q_{out,l}(k)$ valamilyen leosztott hányada lehet $q_l(k)$-nak, $q_{out,l}(k) = \beta_{out,l} q_l(k)$, ahol $\beta_{out,l} \in (0,1)$ a forgalmi tapasztalatok alapján előre megválasztandó konstans érték (nem beavatkozó jel).

A forgalmi modell, figyelembe véve, hogy (2.2) szerint q_l kifejezhető ρ_l, v_l-lel, alkalmas állapot és bemenő jel választással a standard diszkrétidejű nemlineáris rendszeralakra hozható:

$$x = (\rho_1, \ldots, \rho_N, v_1, \ldots, v_N, w_1, \ldots w_N)^T \tag{2.7a}$$

$$u = (r_1, \ldots, r_N)^T \tag{2.7b}$$

$$x(k+1) = f(x(k), u(k)). \tag{2.7c}$$

2.2 Autópálya hálózat nem célorientált dinamikus modellje

Az elágazásokkal (bifurcations) és összekapcsolódásokkal (junctions) is rendelkező általános autópálya hálózat irányításához az egyetlen autópálya forgalmának Payne-féle klasszikus dinamikus modelljét általánosítani kell, megengedvén elágazásokat és összekapcsolódásokat is, így lehetőséget adva fővonal és mellékvonalak együttes vizsgálatára is. A lehajtók (out-ramps) és felhajtók (on-ramps) az elágazások és az összekapcsolódások speciális eseteinek tekinthetők. A dinamikus modell felállításánál felhasználjuk (Kotsialos, Papageorgiu, Mangeas & Haj-Salem, 2002) eredményeit.

A nem célorientált üzemmódban (non-destination oriented mode) az autópálya hálózat csomópontokból (nodes) és autópálya szekciókból (links) áll, ahol minden egyes szekció tovább van osztva a szekción belül egyforma hosszú és sávszámú szegmesekre. Ha m egy ilyen szekció, akkor szegmenseinek száma N_m, a szegmensek kettős indexelése m,l, ahol $l = 1, \ldots, N_m$, a szegmens hossza L_m és a szegmens sávjainak száma λ_m. Az autópálya hálózat $a, b, c, \ldots$ csomópontjait ott kell elhelyezni, ahol nagyobb változás lép fel az útvonal geometriájában, valamint az elágazási, lehajtási, felhajtási és összekapcsolódási helyeken (bifurcations, out-ramps, on-rumps, junctions). A szekciók szegmensekre osztása javítja a modell pontosságának esélyeit a hosszparaméter diszkretizálásakor.

Az autópálya hálózat forgalmát makroszkópikusan a $\rho_{m,l}(k)$ sűrűség [veh/km/lane], a $v_{m,l}(k)$ átlagos sebesség [km/h] és a $q_{m,l}(k)$ folyam [veh/h] jellemzik a $[kT,(k+1)T]$ időintervallumban és az m szekció l szegmensében.

A Payne-féle klasszikus modell a következőképp általánosítható nem célorientált üzemmódú autópálya hálózatra:

$$\rho_{m,l}(k+1) = \rho_{m,l}(k) + \frac{T}{L_m \lambda_m}(q_{m,l-1}(k) - q_{m,l}(k)) \tag{2.8}$$

$$v_{m,l}(k+1) = v_{m,l}(k) + \frac{T}{\tau}[V(\rho_{m,l}(k)) - v_{m,l}(k)] + \frac{T}{L_m} v_{m,l}(k)[v_{m,l-1}(k) - v_{m,l})k)] - \frac{\nu T[\rho_{m,l+1}(k) - \rho_{m,l}(k)]}{\tau L_m[\rho_{m,l}(k) + \kappa]} \tag{2.9}$$

$$q_{m,l}(k) = \rho_{m,l}(k) v_{m,l}(k) \lambda_m \tag{2.10}$$

$$V(\rho_{m,l}(k)) = v_{f,m} \exp\left[-\frac{1}{a_m} \left(\frac{\rho_{m,l}(k)}{\rho_{cr,m}} \right)^{a_m} \right]. \tag{2.11}$$

A (2.9) egyenletben szereplő τ, ν, κ paramétereket a forgalom megfigyeléséből szerzett adatokból kell meghatározni valamilyen identifikációs technikával (például nemlineáris paraméterbecsléssel). A $v_{m,l}(k)$ után álló három additív tag rendre az ú.n. relaxációt, konvekciót és anticipációt modellezi. A $V(\cdot)$ függvény a relaxációs tagban azt fejezi ki, hogy az átlagos sebesség minden szekcióban egy sűrűségtől függő $V(\rho_{m,l}(k))$ egyensúlyi érték felé tart, amelyben $v_{f,m}$ a szabad haladás sebessége és $\rho_{cr,m}$ a sávonkénti kritikus sűrűség a szekcióban, a_m pedig modell paraméter, melyeket a forgalom megfigyeléséből szerzett adatokból és identifikációs technika bevonásával lehet meghatározni. A (2.9) és (2.11) egyenletben szereplő patraméterek a forgalmon kívül függenek az út geometriájától, a járművek jellemzőitől és a vezetők viselkedésétől is.

Kiindulási szekcióknak fogjuk nevezni azokat a szekciókat, amelyek forgalmi igényt fogadnak és továbbítják azt az autópálya hálózat belsejébe. Kiindulási szekciók a külső tápláló forgalom belépési helyei és a felhajtók. Ezekre nem a (2.8)-(2.11) egyenleteket, hanem várakozási sor modellt alkalmazunk. Az o kiindulási (origin) szekció csatlakozzon az autópálya hálózatba a μ főáram szekción keresztül. Feltesszük, hogy egy o-hoz csak egy μ tartozik. Az o kiindulási szekció kimenő folyamát meghatározza az aktuális főáram (mainstream) és az alkalmazott becsatlakozás irányítás aktuális mértéke (ramp metering control measure). Ha irányítjuk a becsatlakozást, akkor a kimenő folyam, amely elhagyhatja az o szekciót és becsatlakozhat a főáramba a k ütemben, a $\hat{q}_o(k)$ kimenő folyamnak csak egy $r_o(k)$ része lehet, ahol $\hat{q}_o(k)$ az a kimenő folyam, amely akkor állna fenn, ha nem lenne becsatlakozás irányítás. Becsatlakozás irányítás esetén $r_o(k) \in [r_{\min,o},1]$ az irányítás beavatkozó jele. Ha nem alkalmazunk becsatlakozás irányítást az o kiindulási szekció esetén, akkor $r_o(k) = 1$, különben $r_o(k) < 1$. A várakozási sor modellje a következő egyenletekkel írható le:

$$w_o(k+1) = w_o(k) + T[D_o(k) - q_o(k)] \tag{2.12}$$

$$q_o(k) = r_o(k)\hat{q}_o(k) = r_o(k) \min\left\{ D_o(k) + \frac{w_o(k)}{T}, Q_o \min\left\{ 1, \frac{\rho_{\max} - \rho_{\mu,1}(k)}{\rho_{\max} - \rho_{cr,\mu}} \right\} \right\}. \tag{2.13}$$

Legyen n az autópálya hálózat egy csomópontja (tehát elágazás, lehajtó, felhajtó vagy összekapcsolódási hely). A forgalom az n csomópontba bemeneti szekciókon keresztül lép be és szétosztódik a kimeneti szekciók felé a következő szabály szerint:

$$Q_n(k) = \sum_{\mu \in I_n} q_{\mu,N_\mu}(k) \tag{2.14}$$

$$q_{m,0}(k) = \beta_n^m(k) Q_n(k), \quad \forall m \in O_n, \tag{2.15}$$

ahol I_n azon szekciók halmaza, amelyek belépnek az n csomópontba, O_n azon szekciók halmaza, amelyek elhagyják az n csomópontot, a $\beta_n^m(k)$ elágazási ráta (turning rate) pedig azon része $Q_n(k)$ -nak, amely az m szekción keresztül hagyja el az n csomópontot. Feltesszük, hogy a $\beta_n^m(k)$ elágazási ráta ismert a teljes időhorizont számára. Vegyük észre, hogy (2.14)-(2.15) definiálja $q_{m,0}(k)$ értékét, amely szükséges (2.8)-ban, ha $l = 1$.

Ha az n csomóponthoz egynél több elhagyó szekció tartozik, akkor az elmenő áram (upstream) befolyását a sűrűségre (2.9) szerint figyelembe kell venni az utolsó szekcióban $l = N_m$ esetén. Erre a következő szabály választható:

$$\rho_{m,N_m+1}(k) = \sum_{\mu \in O_n} \rho_{\mu,1}^2(k) \Big/ \sum_{\mu \in O_n} \rho_{\mu,1}(k), \tag{2.16}$$

ahol $\rho_{m,N_m+1}(k)$ az m belépő szekció virtuális lefelé haladó áramsűrűsége (dowstream), amelyet $l = N_m$ esetén kell alkalmazni, $\rho_{\mu,1}(k)$ pedig a μ elhagyó szekció első szegmensének sűrűsége, amelyet $l = 1$ esetén kell alkalmazni (2.9)-ben. A (2.16)-ban alkalmazott kvadratikus kifejezés azt az elvet tükrözi vissza, hogy egy elhagyó szekció telítődött forgalom folyama visszahat a belépő szekcióra akkor is, ha a többi elhagyó szekcióban a forgalom folyam szabad.

Ha az n csomópont egynél több belépő szekcióval rendelkezik, akkor a lefelé haladó áram (downstream) befolyását az átlagos sebességre $l = 1$ esetén figyelembe kell venni (2.9)-ben. Ez az átlagos sebesség számítható

$$v_{m,0}(k) = \sum_{\mu \in I_n} v_{\mu,N_\mu}(k) q_{\mu,N_\mu}(k) \Big/ \sum_{\mu \in I_n} q_{\mu,N_\mu}(k) \tag{2.17}$$

alapján, ahol $v_{m,0}$ az m elhagyó szekció felfelé haladó áramának virtuális sebessége, amelyet $l = 1$ esetén kell alkalmazni (2.9)-ben.

Nem célorientált módban az x állapotvektor komponensei a $\rho_{m,l}$ sűrűségek és a $v_{m,l}$ átlagos sebességek minden m szekció l szegmense esetén, valamint a w_o várakozási sorok minden o kiindulási szekció esetén. A bavatkozó jel vektor komponensei az $r_o \in [0,1]$ kiszolgálási ráták minden o kiindulási szekció esetén. Az $x(k+1) = f(x(k), u(k), d(k))$ állapotegyenlet úgy keletkezik, hogy behelyettesítjük a (2.10), (2.14), (2.15) egyenleteket (2.8)-ba, a (2.11), (2.16), (2.17) egyenleteket (2.9)-be, továbbá a (2.13) egyenletet (2.12)-be. A $d(k)$ zavaró jel vektor komponensei a D_o igények az o kiindulási szekciók esetén, továbbá a β_n^m elágazási ráták, ha az n csomópont az m szekción keresztül kerül elhagyásra.

2.3 Autópálya hálózat célorientált üzemmódú dinamikus modellje

Célorientált üzemű módban (destination oriented mode) további forgalom jellemzők és OD-információk szükségesek. Először is ismerni kell a bejövő helyeket (origin links) és a célhelyeket (destination links), továbbá a bejövő helyeken a bejövő forgalom megoszlási arányát minden onnan elérhető célhely felé. Hasonlóan ismerni kell az elágazási helyeken a forgalom megoszlási arányát a különböző elérhető célhelyek felé. A hálózat struktúrájából és az OD-információból meg kell határozni a lehetséges utakat minden egyes belépő hely és onnan elérhető célhely között, figyelembe véve az alternatív utakat is a hálózatban. Minden út jellemezhető az út mentén érintett csomópontok és szegmensek rendezett halmazával. Ezáltal minden bemenet-cél párhoz egy vagy több ilyen rendezett halmaz keletkezik, ilymódon az összes OD-pár az útvonalhalmazok készletét generálja. A dinamikus modell felállításánál felhasználjuk (Karami, Hegyi, De Schutter, Hallendorn, & Middelham, 2004) és (Bellemans, 2003) eredményeit.

A felhajtókon modellezni kell a célhelyek felé a belépő forgalom megosztását az onnan elérhető célhelyek felé, továbbá külön várakozási sort kell képezni minden, a felhajtóról elérhető célhely felé. Ezen túlmenően ha az m szekció (link) szerepel valamelyik generált útvonalhalmazban, akkor a szekció forgalmában modellezni kell az onnan elérhető célhelyek felé áramló forgalmat is, azaz be kell vezetni a $\rho_{m,l,j}(k)$ parciális forgalmat is, ahol j annyi értéket vesz fel, amennyi az m szekcióból elérhető célhelyek száma. Ha tehát az m szekción keresztül elérhető célhelyek halmazát J_m jelöli, akkor a $\rho_{m,l,j}(k)$ parciális sűrűség azon járművek sűrűsége az m szekció l szegmensében a kT pillanatban, amelyeknek az m szekción keresztül elérhető $j \in J_m$ cél felé kell haladniuk az OD információ szerint. A parciális forgalom sűrűség egyenlete ennek megfelelően a következőképp alakul:

$$\rho_{m,l,j}(k+1) = \rho_{m,l,j}(k) + \frac{T}{L_m \lambda_m}[\gamma_{m,l-1,j}(k) q_{m,l-1}(k) - \gamma_{m,l,j}(k) q_{m,l}(k)] \quad \forall j \in J_m, \text{(2.18a)}$$

$$\rho_{m,l}(k) = \sum_{j \in J_m} \rho_{m,l,j}(k), \text{(2.18b)}$$

$$\gamma_{m,l,j}(k) := \rho_{m,l,j}(k) / \rho_{m,l}(k), \text{(2.18c)}$$

ahol $\gamma_{m,l,j}(k)$ azon hányada a $q_{m,l}(k)$ forgalom folyamnak, amelynek a $j \in J_m$ cél felé kell haladnia az OD információ szerint. A $v_{m,l}(k)$ átlagsebesség és a $q_{m,l}(k)$ eredő folyam számítása továbbra is (2.9)-(2.10) alapján történik. Részletesebb vizsgálatokhoz értelmezhető a parciális folyam is, amely $q_{m,l,j}(k) = \rho_{m,l,j}(k) v_{m,l}(k) \lambda_m$, a teljes folyam ezek összege: $q_{m,l}(k) = \sum_{j \in J_m} q_{m,l,j}(k) = \sum_{j \in J_m} \rho_{m,l,j}(k) v_{m,l}(k) \lambda_m = \rho_{m,l}(k) v_{m,l}(k) \lambda_m$, ami a (2.10) egyenlet.

Célorientált üzemmód esetén általánosítani kell a β_n^m elágazási rátát (turning rate) is. Ehhez bevezetjük a parciális elágazási ráta (splitting rate) fogalmát. Tegyük fel, hogy a j célhely az n csomópontból egynél több kimeneti szekción keresztül érhető el. Legyen $Q_{n,j}(k)$ a teljes forgalom folyam, amely belép az autópálya n csomópontjába a k ütemben a j célhely felé. Akkor a $\beta_{n,j}^m(k)$ parciális elágazási ráta azon hányada a $Q_{n,j}(k)$ folyamnak, amely az m szekción keresztül hagyja el az n csomópontot a k ütemben, feltéve, hogy a j célhely elérhető az m szekción keresztül. Világos, hogy $0 \le \beta_{n,j}^m(k) \le 1$. Ezért minden csomópont esetén definiálhatók a következő forgalom jellemzők:

$$Q_{n,j}(k) = \sum_{\mu \in I_n} q_{\mu,N_\mu}(k) \gamma_{\mu,N_\mu,j}(k) \quad \forall (n,j) \text{(2.19)}$$

$$q_{m,0}(k) = \sum_{j \in J_m} Q_{n,j}(k) \beta_{n,j}^m(k) \quad \forall m \in O_n \text{(2.20)}$$

$$\gamma_{m,0,j}(k) = \beta_{n,j}^m(k) Q_{n,j}(k) / q_{m,0}(k) \quad \forall m \in O_n \quad \forall j \in J_m. \text{(2.21)}$$

Itt továbbra is J_m jelöli az m szekción keresztül elérhető célhelyek halmazát, míg O_n az n csomópontot közvetlenül elhagyó szekciók (links) halmaza. A (2.31)-(2.33) egyenletek meghatározzák $q_{m,0}(k)$ és $\gamma_{m,0,j}(k)$ értékét, amely szükséges lesz (2.34)-ben $l = 1$ esetén. Másrészt teljesülnie kell a $\sum_{m \in O_n} \beta_{n,j}^m(k) = 1$ feltételnek, amely minden elágazási csomópontnál (bifurcation) eggyel redukálja a független parciális elágazási ráták (splitting rates) számát.

Hasonló okból kiindulási szekciók esetén $w_{o,j}(k)$ parciális várakozási sort alkalmazunk a következő szabály szerint:

$$w_{o,j}(k+1) = w_{0,j}(k) + T[\theta_{o,j}(k)D_o(k) - \theta_{o,j}(k)q_o(k)], \tag{2.22}$$

$$D_{o,j}(k) = \theta_{o,j}(k)D_o(k) \tag{2.23}$$

$$q_o(k) = r_o(k)\hat{q}_o(k) = r_o(k)\min\left\{\sum_{j\in J_o}\left(D_{o,j}(k) + \frac{w_{o,j}(k)}{T}\right),\ Q_o\min\left\{1, \frac{\rho_{\max} - \rho_{\mu,1}(k)}{\rho_{\max} - \rho_{cr,\mu}}\right\}\right\} \tag{2.24}$$

ahol a $\theta_{o,j}(k)$ kompozíciós ráta azon (ismert vagy becsült) hányada $D_o(k)$-nak, amely a j célhely felé irányul a k ütemben, továbbá $w_{o,j}(k)$ a járművek száma az o kindulási csomóponthoz tartozó és j célhelyhez tartó járművek parciális várakozási sorában, J_o pedig azon célhelyek halmaza, amelyek elérhetők az o felhajtótól kiindulva valamelyik útvonalon. Szabályozott felhajtó esetén teljesülni kell az $r_o(k) \in [r_{\min}, 1]$ korlátozásnak, szabályozás nélkül pedig $r_o(k) = 1$.

A VMS üzenet célja az elágazási (bifurcation) csomópontoknál javaslatot tenni a vezetők számára, akik egy bizonyos célhely felé tartanak, hogy melyik irányt (szekciót, outlink) válasszák az alternatív irányok közül a cél felé haladva. A javaslat hat a vezetők viselkedésére azok egyetértésétől (compliance) függően. Mivel a VMS hivatkozik a célhelyre, az útvonal választást le kell vetíteni a csomóponthoz tartozó parciális elágazási rátára. Az n elágazási csomópontnál a j célhely számára megkülönböztetjük a vezetők $\beta^m_{N,n,j}$ névleges parciális elágazási rátáját útmutatás nélkül (N, no guide), és az irányító rendszer által javasolt (G, guided) $\beta^m_{G,n,j}$ irányított parciális elágazási rátát. A $\beta^m_{n,j}$ valódi parciális elágazási ráta a vezető alkalmazkodásától függ, amely a következőképp modellezhető:

$$\beta^m_{n,j} = (1-\varepsilon)\beta^m_{N,n,j} + \varepsilon\beta^m_{G,n,j}, \tag{2.25}$$

ahol ε az alkalmazkodási ráta (compliance rate), $0 \le \varepsilon \le 1$. Ha $\varepsilon = 0$, akkor egyáltalán nincs alkalmazkodás (zero compliance) és $\beta^m_{n,j} = \beta^m_{N,n,j}$. Teljes alkalmazkodás esetén (full compliance) $\varepsilon = 1$ és $\beta^m_{n,j} = \beta^m_{G,n,j}$. Az ε alkalmazkodási ráta modellezésére be lehet vonni mérési eredményeket és azok kiértékelését, sztochasztikus hipotéziseket, mint például a Logit model, lásd Theil (1969), Cremer (2001), és virtuális járműveket, amelyeket a felhajtókon lehet bejuttatni a rendszerbe és felhasználni haladásukat a rendszerben, lásd Cremer (1995). A probléma egyszerűsödik, ha feltesszük, hogy $\varepsilon = 1$, ekkor $\beta^m_{n,j} = \beta^m_{G,n,j}$ beavatkozó jelnek tekinthető, amelynek ki kell elégítenie a $\beta^m_{n,j} \in [0,1]$ korlátozást.

Célorientált módban az x állapotvektor komponensei a $\rho_{m,l,j}$ parciális sűrűségek minden m szekció l szegmense és az m-ből elérhető j célhely esetén, a $v_{m,l}$ átlagos sebességek minden m szekció l szegmense esetén, valamint a $w_{o,j}$ parciális várakozási sorok minden o kiindulási szekció és azzal összhangban lévő j célhely esetén. Az $u(k)$ bavatkozó jel vektor a $\beta^m_{n,j}$ független parciális elágazási rátákból és a bevezető szakaszok (motorway-to-motorway) és a felhajtók (on-ramps) r_o kiszolgálási rátáiból áll, ahol a beavatkozó jeleknek a $[0,1]$ tartományba kell esniük. A $d(k)$ zavaró jel vektor komponensei a D_o igények és a $\theta_{o,j}$ kompozíciós ráták az o kiindulási szekciók és felhajtók esetén, továbbá a $\beta^m_{N,n,j}$ alkalmazkodás nélküli parciális elágazási ráták, ha az n csomópont az m szekción keresztül kerül elhagyásra a j célhely felé, valamint a vezető ε engedékenysége.

Az állapotegyenlet úgy keletkezik, hogy behelyettesítjük a (2.10), (2.19)-(2.21) egyenleteket (2.18)-ba, a (2.11), (2.16), (2.17) egyenleteket (2.9)-be, továbbá a (2.13) egyenletet (2.12)-be, és ezekhez hozzávesszük a parciális várakozó sorokat leíró (2.22)-(2.24) egyenleteket. Az állapotegyenlet zavaró jellel bővített alakja (mind nem célorientált, mind célorientált esetben) az

$$x(k+1) = f(x(k), u(k), d(k)). \tag{2.24}$$

általános alakra hozható.

Vegyük észre, hogy az f függvényben a 0 és N_{m+1} indexű forgalomjellemzők nem állapotváltozók, számításuk a felhajtók, elágazások, lehajtók és összekapcsolódások helyén speciális képletekkel történik. Ezért minden olyan esetben, ahol az f függvény deriváltjára van szükség, például a nemlineáris prediktív irányítás vagy az állapotbecslésre alkalmazott kiterjesztett Kalman-szűrő esetén, komoly számítási problémát jelent az f függvény deriváltjának analitikus számítása, különös tekintettel az autópálya hálózat komplexitására és a nagy változószámra. Az utóbbiak miatt a véges differenciákon alapuló numerikus deriválás a valósidejű elvárások miatt nem jöhet érdemben szóba.

3. Útvonalválasztás modellezés célorientált üzemmódban

Egy elterjedt visekedés modell, amely jól alkalmazható a résztvevők viselkedésének modellezésére alternatív költségek esetén a Logit model (Theil, 1969), (Cramer, 2001). A modell a következő módon alkalmazható útvonalválasztásra.

3.1 Elágazási ráta számítása a Logit modellben

Legyen az n csomópontnál m_1 és m_2 két lehetséges útvonalválasztás a j célhely felé. A Logit model az elágazási ráta számítására a következő szabályt javasolja:

$$\beta_{n,j}^{m}(k) = \frac{\exp(\sigma \cdot \vartheta_{n,j}^{m}(k))}{\exp(\sigma \cdot \vartheta_{n,j}^{m_1}(k)) + \exp(\sigma \cdot \vartheta_{n,j}^{m_2}(k))} \tag{3.1}$$

ahol $m = m_1$ vagy $m = m_2$ esetén $\exp(\sigma \cdot \vartheta_{n,j}^{m}(k))$ jelöli a DRGIS táblán az n csomópontban jelzett utazási időt a j célhely (destination) felé az m szekción (link) keresztül, és a σ paraméter írja le, hogyan reagálnak a résztvevők az utazási idők közötti differenciára két alternatíva esetén.

3.2 Egyéni utazási idő modellezése

Az egyéni utazási idők számítására szükség van, ha meg kell határozni az eltérést a javasolt és a realizált utazási idő között, és az eltérést be akarjuk vonni a forgalmi stratégia kialakításába. El kell tehát érni, hogy ha egy jármű áthalad egy elágazáson (bifurcation), akkor tárolódjék a DRGIS (Dynamic Route Guidance Information Systems) panel által mutatott információ, és ha a jármű a célhelyen elhagyja a hálózatot, akkor meghatározható legyen az eltérés a realizált és a bemutatott utazási idő között. Az eltérés ebben az esetben bevonható a költségfüggvénybe. Az utazási idő becslésére a következő módszer alkalmazható (Cremer, 1995), (Karami, Hegyi, De Schutter, Hallendorn, & Middelham, 2004).

Minden, mondjuk N-edik szimulációs lépésben néhány virtuális jármű injektálódjon a hálózatba a szimulációs rendszer részeként, amelynek haladását a hálózatban minden szimulációs lépésben követni kell a rendszerben. Legyen ζ egy virtuális jármű, akkor az alábbi információt kell követni a rendszerben:

1. Az útvonalat, amelyen a virtuális jármű halad.
2. A szekciót (link) és szegmenst a szekcióban, ahol a virtuális jármű aktuálisan tartózkodik, és s pozícióját a szegmensben.
3. Az utazási időt, amelyet a virtuális jármű látott mindazon DRGIS panelen, amely mellett már elhaladt.
4. A virtuális jármű τ utazási idejét a már érintett DRGIS panelektől az aktuális pozíciójáig.
5. A tényt, hogy a virtuális jármű elhagyta-e már a hálózatot, és ha igen, azt a pillanatot, amikor áthaladt a célhelyén.

A virtuális jármű aktuális pozícióját a következő módon lehet meghatátozni. Jelölje $s_{\zeta,m,i}(k)$ a ζ virtuális jármű pozícióját, ha a k ütemben az m szekció i szegmensében van, és frissítsük pozícióját az

$$s_{\zeta,m,i}(k+1) = s_{\zeta,m,i}(k) + v_{m,i}(k)T \tag{3.2}$$

szabály szerint, ahol $v_{m,i}(k)$ a modellben szereplő átlagsebesség a szegmensben a k ütemben. Ha az új pozíció meghaladná a szekció L_m hosszát, akkor áttesszük a virtuális járművet a következő szekcióba és annak szegmensébe a tárolt útvonala mentén, mondjuk az m' szekció i' szegmensébe, és a már megtett L_m út levonásával beállítjuk új $s_{\zeta,m',i'}(k+1)$ pozícióját. A virtuális jármű $\tau_{\zeta,\eta}(k)$ utazási ideje az η DRGIS paneltől az aktuális pozícióig

$$\tau_{\zeta,\eta}(k+1) = \tau_{\zeta,\eta}(k) + T \tag{3.3}$$

alapján frissíthető.

A virtuális járművek számára a DRGIS panelen jósolt és a valóban realizált utazási idő közötti eltérés bevonható a célorientált üzemmódú esetben a nemlineáris prediktív irányítás költségfüggvényébe. A virtuális járművek számát a rendszerben úgy kell megválasztani, hogy statisztikailag értelmes információt kapjunk a költségek alakulásáról. Világos, hogy a virtuális járművek száma terheli a rendelkezésre álló számítási időt, ezért számuk megválasztása egy kompromisszum része.

4. Autópálya forgalomirányítási algoritmusok

A gyakorlatban elterjedt autópálya hálózat forgalomirányítási algoritmusok a hálózat dinamikus modelljén alapulnak. Az elágazásokkal (bifurcations) és összekapcsolódásokkal (junctions) rendelkező autópálya hálózat irányításakor a beavatkozás a felhajtó helyek forgalmának jelzőlámpával történő befolyásolása (ramp metering) és/vagy a járművezetőknek küldött üzenetek (VMS, variable message sign) révén történik. A VMS különböző célhelyek esetén javaslatot tesz az útvonalra és/vagy az utazási sebességre az alternatív útvonalak forgalmi viszonyainak figyelembevételével. A továbbiakban két jellegzetes autópálya forgalomirányítási algoritmust, egy klasszikusat és egy nemlineáris prediktív irányításon alapulót mutatunk be. Jelen kutatási fázisban nem foglalkozunk a forgalmi paraméterek becslésével az autópálya hálózat telepített mérőhelyeiből nyerhető adatok kiértékelésével, hanem a paramétereket ismertnek véve, a dinamikus modellből szimulációval nyerhető adatokra alapozva képzeljük el a szabályozást. A beavatkozó jelek számításakor párhuzamosan fut a dinamikus modell szimulációja. Ezért szigorúan megkülönböztetjük a dinamikus modell szimulációjakor használt finomabb $T := T_{sim}$ lépésközt és beavatkozások számításakor használt T_{contr} mintavételi időt. Feltesszük, hogy T_{contr} egész számú többszöröse T -nek, ezért a ritkábban számított beavatkozás hatása több szimulációs lépés alatt változatlan marad.

4.1 Autópálya forgalom klasszikus irányítása: ALINEA

Az autópálya forgalom irányítás klasszikus formája a felhajtó helyek várakozási sorainak szabályozása egyszerű PID-típusú algoritmusokkal. Ennek az irodalomból ismert egyik jellegzetes formája az ALINEA irányítás, ahol az elnevezés a francia “Asservissement linéaire d’entrée autoroutière” akronimból származik (Papageorgiou, Hadj-Salem & Blosseville, 1991).

Csatlakozzon a szabályozott o felhajtó (origin) az n csomóponton (node) kersztül a μ szekcióhoz (link). Az ALINEA a felhajtó helyek forgalmát a dinamikus modellben szereplő $r_o(k)$ kiszolgálási arány (metering control) révén befolyásolja, amely részt vesz az μ-edik szekció jelzőlámpával ellátott felhajtó helyén a $w_o(k)$ várakozási sor (dimenziója [veh]) változtatásában a $q_o(k) = r_o(k)\hat{q}_o(k)$ pillanatnyi kiszolgálható rátára (dimenziója [veh/h]) gyakorolt hatása révén. Egy egyszerű I-jellegű (integráló) szabály a felhajtóhely forgalmának irányítására lehet például

$$r_o(k) = r_o(k-1) + K_{R,o}(\hat{\rho}_o - \rho_{\mu 1}(k)), \tag{4.1}$$

ahol $K_{R,o} > 0$ a szabályozó megválasztható paramétere, $\rho_{\mu 1}(k)$ a forgalomsűrűség a felhajtó helyet közvetlenül követő μ szekció (link) első szegmensében és $\hat{\rho}_o$ az alapjel, amelyet célszerű a ρ_{cr} értékre választani. A kiértékelést MATLAB jelöléssel a $rem(k / T_{contr}) = 0$ feltétel teljesülésekor, vagyis T_{contr} egész számú többszöröseinek megfelelő időpontokban kell elvégezni. A kiszámított $r_o(k)$ értékét az $[r_{\min}, 1]$ tartomány határán fel kell ütköztetni, és az így kapott értéket kell alkalmazni. Értelemszerűen mérési lehetőségek esetén a szimulált $\rho_{\mu 1}(k)$ helyébe a $\rho_{meas}(k)$ mért érték lép.

4.2 Nemlineáris modellprediktív irányítás (NMPC)

A továbbiakban előbb megfogalmazzuk a nemlineáris modellprediktív irányítás általános algoritmusát és bemutatunk néhány gyakran használt numerikus optimalizálási módszert az optimális beavatkozás számítására, majd megadjuk az NMPC irányítás megvalósításának sémáját autópálya hálózat irányításakor.

4.2.1 Modellalapú nemlineáris prediktív irányítás általános algoritmusa

Nemlineáris rendszerek esetén a jelenlegi modellalapú prediktív irányítási módszerek rendszerint a prediktív irányításhoz testreszabott új optimumkereső eljárásokon vagy tradícionális analitikus

optimum feltételeken és gradiens-alapú optimalizálási módszereken alapulnak (Allgöver & Zheng, 2000), (Kim & Shin, 2003). Az utóbbiak alapja a Lagrange-multiplikátor szabály egy általános alakja, lásd például (Lantos, 2003).

Tipikus véges horizontú nemlineáris prediktív irányítási problémák diszkrét időben végesdimenziós térben megoldandó optimalizálási feladatra vezetnek, ahol a változók az $x=\{x_i\}_{i=0}^{N}$ állapotsorozat és az $u=\{u_i\}_{i=0}^{N-1}$ beavatkozó jel sorozat, az optimalizálási kritérium például kvadratikus alak a végállapot külön büntetésével,

$$\begin{aligned} F_0(x,u) &= \sum_{i=0}^{N-1}[<Q_i x_i, x_i> + <R_i u_i, u_i>]/2 + <Q_N x_N, x_N>/2 \\ &=: \sum_{i=0}^{N-1} L_i(x_i,u_i) + \Phi(x_N), \end{aligned} \tag{4.1}$$

a korlátozások pedig az állapotegyenlet

$$\varphi(x_i,u_i) - x_{i+1} = 0, \tag{4.2}$$

az irányítási halmaz $u_i \in M$ és a kezdeti feltétel $a - x_0 = 0$. Ha (x^*,u^*) az optimális megoldás, akkor az általános Lagrange-multiplikátor szabály szerint

$$f(x,u) = J'_x(x^*,u^*)x + J'_u(x^*,u^*)u \tag{4.3}$$

a deriváltja $J(x,u)$ -nak, ahol

$$J(x,u) = F_0(x,u) + <\lambda_0, a - x_0> + <\lambda_1, \varphi(x_0,u_0) - x_1> + \cdots + <\lambda_N, \varphi(x_{N-1},u_{N-1}) - x_N> \tag{4.4}$$

Bevezetve a Hamilton-függvényt, amelynek alakja

$$H_i = <\lambda_{i+1}, \varphi(x_i,u_i)> + L_i(x_i,u_i), \tag{4.5}$$

az optimum szükséges feltétele tetszőleges sima $L_i(x_i,u_i)$ és $\Phi(x_N)$ esetén, és speciálisan kvadratikus optimalizálási kritérium esetén, a következő lesz:

$$\begin{aligned} \lambda_N &= \partial\Phi/\partial x_N = Q_N x_N, \\ \lambda_i &= \partial H_i/\partial x_i = \partial L_i/\partial x_i + (\partial\varphi/\partial x_i)^T \lambda_{i+1} = Q_i x_i + (\partial\varphi/\partial x_i)^T \lambda_{i+1}, \\ \partial H_i/\partial u_i &= \partial L_i/\partial u_i + (\partial\varphi/\partial u_i)^T \lambda_{i+1} = R_i u_i + (\partial\varphi/\partial u_i)^T \lambda_{i+1}, \\ dJ &= \sum_{i=0}^{N-1} <\partial H_i/\partial u_i, u_i^* - u_i> \le 0. \end{aligned} \tag{4.6}$$

A szabályozó tervezéshez az aktuális horizonton belül először szükség van az x_0 kezdeti feltételre és az u irányítás (sorozat) kezdeti approximációjára (az utóbbi lehet az előző horizonton belül kapott megoldás egy lépéssel jobbra eltolva és a hiányzó értéket valamilyen alkalmas technikával pótolva).

Az optimalizálás a következő lépéseket ismétli ciklikusan (Lantos, & Kiss, 2005), (Lantos, 2006):

1. Az állapotegyenlet megoldása, azaz az $x=\{x_i\}_{i=0}^{N}$ sorozat meghatározása (előretartó rekurzió) az $u=\{u_i\}_{i=0}^{N-1}$ sorozat kezdeti, majd az iteráció során kialakult aktuális értékét alkalmazva.
2. A λ_i Lagrange-multiplikátorok meghatározása (hátratartó rekurzió).
3. A $\partial H_i/\partial u_i$ deriváltak kiszámítása.
4. Numerikus optimalizálás gradiens alapú (gradiens, konjugált gradiens, Davidon-Fletcher-Powell stb.) módszerrel az $u=\{u_i\}_{i=0}^{N-1}$ beavatkozó jel (sorozat) meghatározásához, vagy egy korlátozásokat is megengedő általános NP (Nonlinear Programming) feladat megoldásával.
5. Az 1.–4. lépések ismétlése a pontossági korlátok teljesülésének eléréséig.
6. Az optimális $u=\{u_i\}_{i=0}^{N-1}$ sorozat első u_0 beavatkozó jelének kiadása zárt körben, majd a horizont jobbra tolása egy ütemmel és az új horizont számára új $u=\{u_i\}_{i=0}^{N-1}$ közelítő megoldás számítása.

Megjegyezzük, hogy a gradiens alapú optimumkeresés helyett a korlátozásokat is megengedő, nemlineáris programozáson (NP) alapuló optimalizálási módszerek bonyolultabbak és jelentősen nagyobb számítási igényűek, ezért valós időben csak megfelelő kritikával alkalmazhatók, mivel az optimalizálást az aktuális horizonton belül T_{contr} mintavételi idő alatt el kell végezni. Az első horizont esetén nem prediktív tervezési módszer szükséges az u irányítás sorozat kezdeti approximációjának meghatározására.

Ha az eredeti rendszer folytonosidejű, akkor először approximálni kell diszkrétidejű rendszerrel, például

$$\dot{x} = f_c(x,u) \Rightarrow x_{i+1} = x_i + Tf_c(x_i,u_i) =: \varphi(x_i,u_i)\,, \tag{4.7}$$

ahol T a mintavételi idő. Ha a teljes állapot nem mérhető, akkor x_0 becsülhető egy kiterjesztett Kalman-szűrővel.

Ha $y = Cx$ a rendszer kimenő jele és $\tilde{y} = y_d - y$ a hiba, továbbá elvárás, hogy a célfüggvénynek a kimenő jel hibájára kell érzékenynek lennie, akkor a költségfüggvény módosítható a következő alakúra:

$$\begin{gathered} \tilde{y} = y_d - Cx, \\ 2L_i(x_i,u_i) = <\tilde{Q}_i\tilde{y}_i,\tilde{y}_i> + <S_ix_i,x_i> + <R_iu_i,u_i>, \\ 2\Phi(x_N) = <\tilde{Q}_N\tilde{y}_N,\tilde{y}_N>, \end{gathered} \tag{4.8}$$

ahol a deriváltak a következő szabály szerint számíthatók:

$$\begin{aligned} \partial L_i / \partial x_i &= -C^T\tilde{Q}_i\tilde{y}_i + S_ix_i\,, \\ \partial\Phi / \partial x_N &= -C^T\tilde{Q}_N\tilde{y}_{N.} \end{aligned} \tag{4.9}$$

Ha korlátozás nélküli optimumkereső eljárást alkalmazunk, akkor u_i projektálandó a korlátozási halmazra. Az állapotváltozóra vonatkozó korlátozások büntetőfüggvény (penalty function) alakjában vehetők figyelembe, amelyeket hozzá kell adni $L_i(x_i,u_i)$-hoz a költségfüggvényben. Ismeretes, hogy a $\Phi(x_N)$ tag súlyozása befolyással van a rendszer stabilitására és dinamikus viselkedésére prediktív irányítás esetén (Allgöver & Zheng, 2000).

4.2.2 Optimumkereső módszerek

A numerikus optimumkeresés széles tárházából optimumkeresésre a korlátozás nélküli esetben elsősorban a gradiens alapú konjugált gradiens, Davidon/Fletcher/Powell (DFP), Fletcher/Reeves és Polak/Ribierre módszerek javasolhatók, korlátozás esetén büntetőfüggvény bevonásával, vagy a megfelelő számítási idő rendelkezésre állásakor a Schittkowski-féle SQP módszer az általános nemlineáris programozási feladat megoldására korlátozások esetén. A részletek tekintetében lásd (Lantos, 2003). A továbbiakban röviden összefoglaljuk ezeket a módszereket arra az esetre, ha a feladat egy $f(x)$ függvény minimumának meghatározására vezethető vissza.

Konjugált gradiens algoritmus tetszőleges függvény esetén:

1. Inicializálás: Legyen x_0 az induló érték, $g_0 := f'(x_0)$, $d_0 = -g_0$, $k = 0$.
2. Tegyük fel, hogy már meg lett határozva x_k, $g_k = f'(x_k)$ és a d_k ú.n. konjugált irány. Keressük meg a minimumát a $\varphi(\lambda) = f(x_k + \lambda d_k)$ függvénynek optimumkereséssel egyetlen skalár változóban (pl. Fibonacci-kereséssel, harmadfokú polinommal való közelítéssel), és legyen λ_k a minimum helye. Határozzuk meg az új keresési irányt a következő szabály szerint:

$$x_{k+1} := x_k + \lambda_k d_k\,, \quad g_{k+1} := f'(x_{k+1})\,, \tag{4.10a}$$

$$\beta_k := \frac{< g_{k+1}, g_{k+1} - g_k >}{< d_k, g_{k+1} - g_k >}, \tag{4.10b}$$

$$d_{k+1} := -g_{k+1} + \beta_k d_k . \tag{4.10c}$$

Ismételjük a 2. lépést $k = n$ eléréséig, ha $\|g_k\| > 0$. Ha elértük $k = n$-et, akkor legyen $x_0 := x_n$ és folytassuk az 1. lépéstől (újra inicializálás). Ha $\|g_k\| \approx 0$, akkor fogadjuk el a minimum helyének x_k-t és stop.

Megjegyzés: Ha $f(x)$ konvex és kvadratikus, akkor az algoritmus maximum n lépésben konvergál. Ha azonban nem kvadratikus, akkor a módszer csak közelítő, és általában nem érhető el a minimum $n = \dim x$ lépés alatt. A tapasztalat szerint célszerű az algoritmust újra inicializálni minden n-edik lépés után, ahogy azt a fentiekben javasoltuk.

Davidon/Fletcher/Powell eljárás

A *Davidon/Fletcher/Powell eljárás* inicializáláskor igényel egy $H_0 > 0$ szimmetrikus és pozitív definit mátrixot is x_0 mellett. A keresési irány $d_k := -H_k f'(x_k)$, a λ_k értékét iránymenti kereséssel kell meghatározni, $x_{k+1} := x_k + \lambda_k d_k$, majd korrigálni kell a mátrixot a következő szabály szerint:

$$\begin{aligned} q_k &:= f'(x_{k+1}) - f'(x_k), \quad r_k := x_{k+1} - x_k, \\ H_{k+1} &:= H_k + \frac{r_k r_k^T}{< r_k, q_k >} - \frac{(H_k q_k)(H_k q_k)^T}{< q_k, H_k q_k >}. \end{aligned} \tag{4.11}$$

Az algoritmus garantálja, hogy $H_k > 0$ marad. Az $f''(x)$-re vonatkozó bizonyos simasági és korlátossági feltételek teljesülésekor a Davidon/Fletcher/Powell eljárás konvergenciája erősebb a lineárisnál.

Fletcher/Reeves és Polak/Ribierre eljárás

A *Fletcher/Reeves* és *Polak/Ribierre* módszernél inicializáláskor x_0 mellett beállítandó még $\beta_0 := 0$ is. A keresési irány $d_k := -f'(x_k) + \beta_{k-1} d_{k-1}$, a λ_k értékét iránymenti kereséssel kell meghatározni, $x_{k+1} := x_k + \lambda_k d_k$, majd korrigálni kell β_k értékét a következő szabály szerint:

$$\begin{aligned} \beta_k &:= \frac{\|f'(x_{k+1})\|^2}{\|f'(x_k)\|^2} && \text{(Fletcher/Reeves)} \\ \beta_k &:= \frac{< f'(x_{k+1}) - f'(x_k), f'(x_{k+1}) >}{\|f'(x_k)\|^2} && \text{(Polak/Ribierre)} \end{aligned} \tag{4.12}$$

Konvex és kvadratikus függvény esetén a d_k keresési irányok konjugált irányok, továbbá a két eljárás azonos. Ha a függvény nem kvadratikus, akkor célszerű minden $n = \dim x$ lépés után újra inicializálni az eljárást: $\beta_k := 0$, ha $k = 0 \pmod{n}$.

Schittkowski-féle SQP módszer

Az optimalizálás általános feladata (GP, general problem):

$$\begin{aligned} &\min f(x) \\ &g_i(x) = 0, \; i = 1, \ldots, m_e \\ &g_i(x) \le 0, \; i = m_e + 1, \ldots, m \\ &x_l \le x \le x_u \end{aligned}$$

ahol $x \in R^n$ a manipulálható változók vektora, m_e az egyenlőség alakjában, $m - m_e$ az egyenlőtlenség alakjában adott korlátozások száma, és az x_l, x_u vektorok közé kell esnie x-nek (vektor értelemben, komponensenként).

Az optimalizálást végző függvények igénylik az $f'(x) = \nabla f(x)$ gradienst és numerikus iterációval közelítik az $f''(x) = H(x)$ Hess-mátrixot (BFGS módszer). Ha a gradiens számítási szabálya megadható, akkor a keresés gyorsítható. Ellenkező esetben az elsőrendű parciális deriváltak numerikus becslésére van szükség a véges differenciák módszerével az $f(x + \Delta x)$ értékekből, ami növekvő bemeneti változószám és bonyolult költségfüggvény esetén nagyban megnöveli a számítási időt.

A korlátozások melletti általános minimalizálási feladatat megoldását a MATLAB Optimization Toolbox

$$x = \texttt{fmincon}(\texttt{fun}, x0, A, b, Aeq, beq, lb, ub, \texttt{nonlcon,options}, P1, P2, \ldots)$$

függvénye a Schittkowski-féle szekvenciális kvadratikus programozás módszerével (SQP, sequential quadratic programming) határozza meg. A módszer minden iterációs lépésben az x_i becslés környezetében kvadratikus függvénnyel közelíti a költségfüggvényt, lineárissal a korlátozásokat, megoldja az így keletkező QP kvadratikus programozási feladatot az ú.n. aktív halmaz módszerrel, és mindezt ciklikusan ismétli a pontossági követelmények eléréséig. Jelölje $L(x, \lambda) = f(x) + \sum_i \lambda_i g_i(x)$ a Lagrange függvényt, akkor az SQP módszer fő iterációs lépései a következők:

1. Az $L(x, \lambda)$ Lagrange függvény Hess-mátrixának frissítése (BFGS módszer).
2. QP kvadratikus programozási részfeladat megoldása a keresési irány meghatározására (aktív halmaz módszer), a QP feladat optimális d_i^* megoldása lesz a keresési irány. Az optimum új $x_{i+1} = x_i + \alpha d_i^*$ becslését iránymenti optimalizálással kell meghatározni.
3. Értékelő függvény (merit function) számítás és iránymenti keresés. Az értékelő függvény induláskor jobban bünteti a kis gradiensű korlátozásokat, mert ezek a megoldási pont környékén fontosabbak, ha szerepelnek az aktív korlátozások között.

Az optimumkereső eljárások csak a lokális optimum meghatározására alkalmasak, és hogy melyiket (a globálisat vagy annál rosszabbat) talják meg, az attól függ, hogy milyen kezdeti értékről indul a keresés. Ha a globális optimum közeléből, akkor van esély a globális optimum megtalálására, különben nem. Ezért a jó kezdeti érték megválasztása kulcsfontosságú, mert növekvő dimenziók mellett esélytelen a kezdeti becslések hatásának feltérképezése. Ezért különösen jelentős, ha technológiai oldalról kedvező x_0 kezdeti becslés és x_l, x_u tartomány adható meg. A globális optimum megtalálását jobban garantáló és sztochasztikus keresésen alauló GA (Genetic Algorithm) módszer nagy futási ideje miatt valósidejű feladatoknál nem jöhet számításba.

4.2.3 NMPC implementálási szempontok autópálya hálózat irányításakor

A nemlineáris modellprediktív irányítás implementálásakor célszerű bevezetni egy szimulációs lépésköz számlálót és egy szabályozási számlálót: $t = kT = zT_{contr} \Rightarrow k = zT_{contr} / T$. Az optimalizálandó változók számának csökkentése alapvető fontosságú mind a valósidejűség, mind pedig a beavatkozó jel simaságának biztosítása érdekében.

Az NMPC a $t = zT_{contr}$ kezdőpontú horizontban a jövőbeli folyamatjeleket a $[t, t + N_p T_{contr})$ intervallumban a modellre alapozva becsli, ahol N_p a predikciós horizont. Az NMPC numerikus optimalizálással meghatározza az optimális $u(z), u(z+1), \ldots, u(z + N_p - 1)$ beavatkozó jel (irányítás) sorozatot a horizonton belül, lásd altalános modellpredikciós algoritmus a 4.2.1 pontban. Azonban autópálya forgalomirányításnál nincs előírt referencia pálya, ezért az optimalizálási cél a TTS teljes utazási idő, a predikciós hiba és az irányítási variancia súlyozott

összegének minimalizálása. A feladat komplexitásának csökkentése és a beavatkozó jelek simaságának biztosítása érdekében egy N_c ($\leq N_p$) irányítási horizontot is definiálunk, és az irányítást konstansnak választjuk az irányítási horizont elérése után: $u(z+l) = u(z+N_c-1)$, ha $l = N_c, \ldots, N_p - 1$. A zavaró jelek és a rendszer paramétereinek hatását is figyelembe véve az autópálya NMPC irányítása a következő mozgó horizontú (RHC, Reeceding Horizon Control) megközelítést alkalmazza:

RHC algoritmus:

1. Az aktuális $t = zT_{contr}$ pillanatban mérni vagy becsülni kell (pl. kiterjesztett Kalman-szűrővel) az aktuális forgalmi állapotot. Ilyen lehetőségek hiányában korábban keletkezett forgalmi statisztikák bevonásával a dinamikus modell szimulációjával számítandó az aktuális forgalmi állapot.
2. Meg kell oldani az NMPC irányítási feladatot az autópálya dinamikus modelljének bevonásával a becsült optimális $u(z), u(z+1), \ldots, u(z+N_c-1)$ irányítás meghatározására. Ennek során alkalmazandó a korábban tárgyalt általános NMPC algoritmus, és azon belül gradiens alapú optimumkeresési technika (konjugált gradiens, DFP stb. módszer), vagy SQP illetve az MATLAB Optimization Toolbox `fmincon` függvénye.
3. Alkalmazzuk az optimális sorozat első $u(z)$ mintáját a rendszer számára aktuális bevatkozó jelként. A beavatkozó jel a felhajtók $r_o(z)$ kiszolgálási rátája és az elágazások $\beta_{n,j}^m$ javasolt elágazási rátája, amely utóbbit konvertálni lehet célhelyfüggő utazási időkké (VMS).
4. A következő szabályozási ütem számára legyen $z := z+1$, és ugrás az 1. lépésre.

Költségfüggvény

Az autópálya forgalomirányítás hatékonysága minőségének jellemzésére kölönféle kritériumok terjedtek el. Elterjedt szempont, hogy a költségfüggvény (cost function, objective function) vegye figyelembe az összes jármű által az autópálya hálózatban és a felhajtó helyek várakozási soraiban eltöltött idő (TTS, total time spent by all vehicles on the motorway and in the queues), a beavatkozó jelek varianciáját (a beavatkozó jelek megválozása négyzetének súlyozott összegét, control variance), és opcionálisan a jósolt és valódi utazási idő közötti eltérés négyzetének súlyozott összegét.

A célfüggvényt a $[zT_{cntr}, (z+N_p)T_{cntr}]$ periódusban kell kiértékelni. Az N_p és N_c paraméterek megválasztására a következő ökölszabály alkalmazható. Az N_p predikciós horizontot úgy kell megválasztani, hogy bármelyik jármű végig tudjon haladni a teljes autópálya hálózaton a predikciós horizont alatt. Ez azt jelenti, hogy a worst case szituációt, tehát a torlódást (congestion) kell kiindulási alapnak tekinteni. Az N_c irányítási horizontot úgy kell behangolni, hogy kis számítási költség mellet biztosítani tudja az optimális viselkedést (optimal performance). Értéke mindig egy kompromisszum az optimum pontossága és a számítás komplexitása között, nem megfeledkezve arról, hogy az N_c irányítási horizont után az N_p horizont végéig az optimális szekvencia hosszának csökkentése érdekében az irányítás konstans marad.

Vezessük be a következő jelöléseket. Legyen $\Omega(z) = \{k_0, k_0+1, \ldots, k_0 + N_p T_{cntr}/T - 1\}$, ahol $k_0 = zT_{cntr}/T$, jelölje $\mathcal{L}$ a szekciók (links) halmazát, S_m az m szekció szegmenseinek halmazát, O a felhajtók (origins) halmazát, $V(z)$ mindazon járművek indexének halmazát, amelyek elhagyják a hálózatot a $[zT_{cntr}, (z+N_p)T_{cntr}]$ intervallumban, továbbá legyen $D(\zeta)$ a DRGIS (Dynamic Route Guidance Information System) panelek azon halmaza, amellyel a ζ jármű találkozott, $\vartheta_{pred}(\zeta, \eta)$ az utazási idő, melyet a DRGIS panel mutatott a ζ jármű számára és $\vartheta_{real}(\zeta, \eta)$ a ζ jármű által valóban realizált utazási idő DRGIS η-tól a végcéljáig. Ezekkel a jelölésekkel a költségfüggvény a következő alakban írható fel:

$$J(z) = \xi_1 \alpha_1 \sum_{k \in \Omega(z)} [T \sum_{m \in \mathcal{L}} \sum_{i \in S_m} \rho_{m,i}(k) L_m \lambda_m + \gamma T \sum_{o \in O} w_o(k)] + \xi_2 \alpha_2 \sum_{l=z}^{z+N_c-1} \left\| u(l) - u(l-1) \right\|^2 + \xi_3 \alpha_3 \sum_{\zeta \in V(z)} \sum_{\eta \in D(\zeta)} [\vartheta_{pred}(\zeta, \eta) - \vartheta_{real}(\zeta, \eta)]^2 \tag{4.13}$$

ahol α_i normalizálási, γ és ξ_i súlyozó tényezők. Nem célorientált üzemmódú rányítás esetén a harmadik tag hiányzik.

5. MATLAB alapú szoftver autópálya forgalomirányítására

Autópálya hálózatok szimulációjára a költséges METANET program jól ismert a gyakorlatban. Célunk ennek a szoftvernek fokozatos kiváltása kutatási célra MATLAB bázisú saját fejlesztésű programmal. A jelen kutatási fázisban kifejlesztett szoftver 1. verziója ú.n. nem célorientált üzemmódban képes az autópálya hálózat forgalmi jellemzőinek vizsgálatára szimulációval irányítás nélkül és ALINEA elvű I-jellegű klasszikus algoritmussal irányítás keretében. A szoftver MATLAB bázison került kifejlesztésre, futásához nem szzükségesek toolboxok azért, hogy jó esély legyen a szoftver MATLAB licenszet nem igénylő későbbi stand alone változatánaak kialakítására. A kutatás jövőbeli fázisaiban tervezzük a szoftver szolgáltatásainak bővítését előbb a nem célorientált üzemmódban nemlineáris prediktív irányítással, majd egy további fázisban célorientált üzemmódú irányítással.

A továbbiakban bemutatjuk a szoftver felépítését és felhasználói leírását az alkalmazáshoz szükséges mélységben, valamint egy-egy alkalmazási példát a szimuláció és az irányítás hatékonyságának bemutatására. A szoftver tetszőleges autópálya hálózat definiálását teszi lehetővé a nem célorientált üzemmódban, de felhasználói leírásának ismertetésekor, valamint a szimuláció és a klasszikus irányítás bemutatásakor az eredményeket egy konkrét autópálya hálózat példáján mutatjuk be.

5.1 A vizsgálatok során feltételezett autópálya hálózat struktúra

Az autópálya hálózat struktúrája mindkét esetben a következő lesz:

$$\text{Bevezető szakasz: } \xrightarrow{O_1} a \xrightarrow{L_0} b \tag{5.1a}$$

$$\text{Fővonal: } b \xrightarrow{L_1} c \xrightarrow{L_4} d \xrightarrow{L_3} e \quad \begin{matrix} \downarrow D_{2r} & \uparrow O_{2r} \end{matrix} \tag{5.1b}$$

$$\text{Mellékvonal: } b \xrightarrow{L_2} f \xrightarrow{L_5} g \xrightarrow{L_6} e \quad \begin{matrix} \uparrow D_{3r} & \downarrow O_{3r} \end{matrix} \tag{5.1c}$$

$$\text{Kivezető szakasz: } e \xrightarrow{D_1} \tag{5.1d}$$

A szekciók hossza, a pályák száma és a szegmensek hossza az 5.1 táblázatban található. A két felhajtó mindegyike egysávos.

5.1. táblázat. A feltételezett autópálya struktúrája

Szekció	Hossz [km]	Pályaszám	Szegmens hossz [km]	Szegmensszám
L_0	2	4	1	2
L_1	3	2	1	3
L_2	4	2	1	4
L_3	3	2	1	3
L_4	0.5	2	0.5	1
L_5	0.5	2	0.5	1
L_6	4	2	1	4

A fővonal hossza $L_1 + L_4 + L_3 = 6.5$ [km], a mellékvonalé $L_2 + L_5 + L_6 = 8.5$ [km]. A dinamikus modell paraméterei a 2. fejezetben használt jelölésekkel az 5.2 táblázatban találhatók.

5.2. táblázat. Az autópálya hálózat dinamikus modelljében szereplő paraméterek

Paraméter	Érték	Dimenzió
τ	18	s
κ	40	veh/km/lane
ν	60	km^2 / h
$\rho_{\max}$	180	veh/km/lane
$\rho_{cr,m}$	33.5	veh/km/lane
$v_{f,m}$	110	km/h
a_m	1.636	
δ	0.0122	
ϕ	2.98	

A $v_{f,m}, \rho_{cr,m}, a_m$ paraméterek alapján a szekciók (links) kapacitása 2000 [veh/h/lane]. A kiindulási helyek (felhajtók) kapacitása $Q_0 = 1500$ [veh/h/lane]. A szimulációs lépésköz $T = 10$ [s].

Az igények (demands) a kiindulási helyeken (origins) trapézalakúak, a bevezetésnél kb. 7000 [veh/h], a felhajtóknál kb. 1600 [veh/h] a maximum. Az elágazási helyen az a feltevés, hogy a vezetők 60 %-a, amennyiben céljuk D_1, a fővonalat, a maradék 40 % pedig az alternatív mellékvonalat választja. Szintén feltesszük, hogy az O_1 kiindulási helyen az igény (demand) 4 %-ának a célja a D_2 lehajtó, egy másik 4 %-ának a célja a D_3 lehajtó, és a maradék 92 %-nak pedig a D_1 kivezetés. Ezek az értékek az egész vizsgált időintervallumban érvényesek voltak. Ezekből az adatokból a statikus elágazási és lehajtási ráták kiszámíthatók. Mint már hangsúlyoztuk, a szoftver 1. verziója csak a nem célorientált üzemmódú vizsgálatokat teszi lehetővé, ezért VMS, vagy más néven DRGIS információk kezelésére még nincs lehetőség.

5.2 A szoftver struktúrája

A szoftver futását a `frameMotorway.m` script fájl vezérli, amely a következő lépéseket hajtja végre:

1. Autópálya hálózat inicializálás az `initnetwork()` függvénnyel, amelynek tartalma a mintahálózatra van beállítva. Kimenete a hálózatstruktúrát szimbolikus formában megadó `Snet` MATLAB stuktúra. Az Snet struktúra tartalmazza a geometriai és forgalmi paramétereket (`Snet.Pars`), az autópálya szakaszok azonosítóit és geometriai jellemzőit (`Snet.MotorwayLinks`), a felhajtók azonosítóit (`Snet.OriginLinks`), a lehajtók azonosítóit (`Snet.DestinationLinks`), a hálózati struktúra leírását: a csomópontok azonosítóit, bemeneteit, kimeneteit és elágazási rátáit (`Snet.NodeModels`), az irányított felhajtók azonosítóit (`Snet.ControlledOrigins`) és további komponenseket a későbbi bővítésekhez.
2. A valósidejű igények miatt az `Snet` szimbólikus hálózatleírást a netw2tabs() függvény feldolgozza és célorientált globális változókba és táblázatokba konvertálja, amelyekből az adatok valós időben minimális számítással kiemelhetők.
3. A `CONTR_MODE` globális változó inicializálása (1=szimuláció, 2=ALINEA szerinti I-típusú irányítás).
4. A magasabb szintű paraméterek inicializálása az `inithighpar()` függvénnyel. Itt állítódik be többek között a szimulációs lépésköz (`T_Sim`) és az irányítás mintavételi ideje (`T_Cont`). Itt kell megadni töréspontjaival a felhajtókon érkező forgalmi igényeket (demands), melyekből

`sec2fun()` idősorokat állít elő a vizsgálatokhoz a szimulációs lépésköz ütemében, melyeket fel is rajzol. Tekintettel az állapotváltozók nagy memóriaigényre, amely megnehezítené a futásközbeni dinamikus memóriafoglalást valós időben, itt történik a statikus helyfoglalás az `X_act` és `X_extact` táblákban az állapotváltozók és a bővített állapotváltozók számára.

5. Az állapotváltozó kezdeti értékének beállítása az `initstate()` függvénnyel. Javasolt tipikus konstans forgalmi igények előzetes beállítása a felhajtókon, szimulációval a hozzá tartozó állandósult állapot megkeresése és `.mat` adatfájlba mentése, ami után a továbbiakban már a felhajtókra érkező időben változó forgalmi igények esetén ez az információ visszaállítható az adatfájlból, és ebből a kezdeti feltételből indítható a szimuláció vagy az irányítás. Ezáltal biztosítható, hogy a kezdeti feltételhez és az irányításhoz tartozó tranziensek ne keveredjenek. Az `frameMotorway` script az ehhez szükséges módosításokat kommentekben leírja, és elvégzése esetén az adatfájlba mentés automatikusan megtörténik.
6. Memóriafoglalás történik az irányítás előző értékének tárolásához a globális változók között és beállításra kerül a felhajtók kiszolgálási rátájának kezdeti értéke (`RVEC_OLD`). Az ALINEA I-típusú irányítás esetén beállításra kerül a szabályozók erősítése (`Krvec`). A későbbi vizsgálatoknál a szabályozó erősítése egységesen $Kr_o = 0.005$ értékre van választva.
7. A nagy változószám miatt a szimuláció vagy irányítás során keletkező tranziensek tárolásához rendkívül nagy memóriaterület kell, ezért ezek számára statikus memórialefoglalás történik a globális változók között (`RVQL_tran, WQL_tran, QDEL_tran`). Az ezekben tárolt információ felhasználásra kerül a költségfüggvény számításakor is. Ez a megoldás különösen fontos lesz a nemlineáris prediktív irányítással történő jövőbeli bővítéskor.
8. A forgalmi tranziensek számítása ciklusban szimuláció vagy irányítás esetén a `funmw()` függvénnyel az állapotegyenlet megoldása révén, és a tranziensek tárolása a lefoglalt globális változókban.
9. A szimuláció vagy irányítás tranzienseinek automatikus felrajzolása a `plotrvq()`, `plotwq()` és `plotqdest()` függvényekkel több ablakban, amelyek később analizálhatók, és a MATLAB standard szolgáltatásaival dokumentumokba menthetők.
10. A költségfüggvény értékének és komponenseinek kiszámítása (`TTS, TTT, TWT, QDC`) és kiírása.

A program a fentiek közül csak minimális tevékenységet vár a felhasználótól, ha nem a mintahálózat van érvényben: `initnetwork()` lecserélését a saját feladatának megfelelően, `CONTR_MODE, Krvec` beállítását a frameMotorway scriptben, valamint a felhajtók forgalmi igényének megfogalmazását a sarokpontok megadásával az `inithighpar()` függvényben, lásd a mintahálózatnak megfelelő `DO1, DO2r, DO3r` értékeket ugyanott.

Itt adjuk meg a globális változók teljes készletét illusztrációul. Ezek egységesen minden globális változót használó függvényben bemásolásra kerültek, akkor is, ha közülük csak bizonyosakra, nem pedig a teljes készletre van szükség. Ennek oka a szoftverfejlesztés biztonságosságában keresendő. A globális változók többségét a `frameMotorway` script, vagy a `netw2tabs()` függvény inicializálja.

```
global CONTR_MODE
global N_MWL N_ORL N_DEL N_SAFL N_DUL N_LFULL
global N_NODE N_CORIG N_VMSD N_WQUEUE N_CONTR N_VMSWAYS
global N_STATE N_STATEext
global MWL_Names ORL_Names DEL_Names SAFL_Names DUL_Names
global NODE_Names CORIG_Names VMSD_Names
global NODE_Table LINK_Table WQUEUE_Table CONTR_Table VMSWAYS_Table
global STATE_Table DEMAND_Table
global T_Sim T_Cont T_Full N_Cont N_Hor X_act X_extact TX_tran
global ORLtoNODE_Table DELfromNODE_Table

global RVQL_tran WQL_tran WDEL_tran
global RVEC_OLD
```

5.3 Az autópálya hálózat megadása az initnetwork függvényben

A választott mintahálózat esetén bemutatjuk az autópálya hálózat megadási módját az Snet struktúrában. Más hálózat esetén a felhasználónak ezt a függvényt kell lecserélnie a saját feladatának mefelelően. A kommentek alapján a cseréhez csak elemi rendszertechnikai és MATLAB ismeretek kellenek. Bizonyos kommenteket, amelyek a cseréhez nem szükségesek, de a `funmw()` függvény esetleges továbbfejlesztésekor fontosak lehetnek, most elhagytunk, részletes listájuk később megtalálható lesz.

```
function Snet=initnetwork
%Initialize motorway network

Snet=struct('Pars',[],...
    'MotorwayLinks',[],...
    'OriginLinks',[],...
    'DestinationLinks',[],...
    'StoreAndForwardLinks',[],...
    'DummyLinks',[],...
    'NodeModels',[],...
    'ControlledOrigins',[],...
    'VMSforDestinations',[]);

%Q0 [veh/h/lane] is the capacity of origin links in case of free flow
Snet.Pars=struct('tau',18,'kappa',40,'nu',60,...
    'vfm',110,'rhocr',33.5,'am',1.636,...
    'rhomax',180,...
    'delta',0.0122,'phi',2.98,...
    'Q0',1500);

%Snet.MotorwayLinks={Name1,[Pars1]; Name2, [Pars2];...}
%[Parsi]: [SectionLength Lane SegmentLength] %Length in km
%Number of segments in the section: SectionLength/SegmentLength;
Snet.MotorwayLinks={...
    'L0',[2 4 1];...
    'L1',[3 2 1];...
    'L2',[4 2 1];...
    'L3',[3 2 1];...
    'L4',[0.5 2 0.5];...
    'L5',[0.5 2 0.5];...
    'L6',[4 2 1]};

Snet.OriginLinks={'O1','O2r','O3r'};

Snet.DestinationLinks={'D1','D2r','D3r'};

%Snet.NodeModels={Node1,{Iset1},{Oset1},{Tset1};
Node2,{Iset2},{Oset2},{Tset2};...};
%Example by Kotsialos et al. 2002
%x*y=0.408*0.098=0.04, (1-x)*z=0.592*0.0676=0.04
%(x*(1-y))/((1-x)*(1-z))=0.368/0.552=0.4/0.6=0.6667
%x*(1-y)+(1-x)&(1-z)=0.368+0.552=0.92
Snet.NodeModels={...
    'a',{'O1'},{'L0'},{1};...
    'b',{'L0'},{'L1','L2'},{0.592, 0.408};...
    'c',{'L1'},{'L4','D2r'},{0.9324, 0.0676};...
    'd',{'L4','O2r'},{'L3'},{1};...
    'e',{'L3','L6'},{'D1'},{1};...
    'f',{'L2'},{'L5','D3r'},{0.902, 0.098};...
    'g',{'L5','O3r'},{'L6'},{1}};

Snet.ControlledOrigins={'O1','O2r','O3r'};
```

```
Snet.VMSforDestinations={'D1','D2r','D3r'};
```

Jól látható,hogy a hálózati struktúra leírása MATLAB eszközökkel a felhasználóra csak minimális feladatot ró, lásd a formális hasonlóságot (5.1a-d) egyenletekkel.

5.4 A felhasználói paraméterek megadása inithighpar függvényben

Az inithighpar() függvény szolgál a magasabbszintű felhasználói paraméterek, mint a szimulációs lépésköz, a szabályozó mintavételi ideje és a felhajtók forgalmi igényének (demand) megadására. Ennek egy részletét közöljük itt, a részletes lista később megtalálható lesz.

```
%Sampling
T_Sim=10;           %sec
T_Cont=6*T_Sim;     %sec
T_Full=1200*T_Sim;  %sec

%----------------------------------------
%Demand description for origins in pair of (n in step,dn in veh/h)
%nmin=0 needed
%nmax should be the same for every demands
%For real investigation
DO1=[0 1000; 180 7000; 360 7000; 540 5000; 1400 5000];
DO2r=[0 600; 90 1800; 270 1800; 360 600; 1400 600];
DO3r=DO2r;
%For steady state computation and store
% DO1=[0 1000;  1400 1000];
% DO2r=[0 600;  1400 600];
% DO3r=DO2r; %DO3r(2,2)=1000;
%----------------------------------------

[yi,xi]=sec2fun(DO1); %Uses interp1
DEMAND_Table=NaN*ones(length(xi),N_ORL+1);
DEMAND_Table(:,1)=xi;
DEMAND_Table(:,2)=yi;
DEMAND_Table(:,3)=sec2fun(DO2r);
DEMAND_Table(:,4)=sec2fun(DO3r);
```

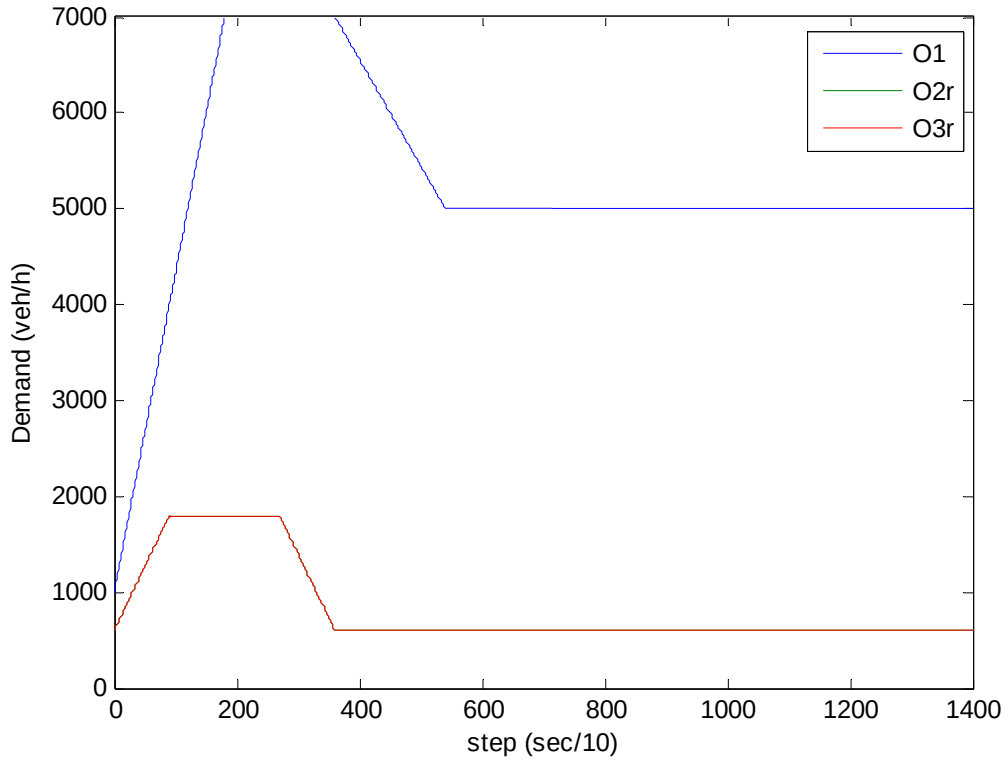
A felhasználónak a `T_Sim, T_Cont, T_Full` globális változók értékét, valamint a felhajtók szakaszonként lineáris forgalmi igényinek paramétereit kell megadnia (utóbbi itt a mintahálózatnak felel meg). A forgalmi leírást `sec2fun()` függvénnyel (amely meghívja a MATLAB interp1 függvényét) adatsorrá kell konvertálni és elhelyezni a `DEMAND_Table` globális táblázatban. Az `inithighpar()` függvény a felhajtókon érkező forgalmi igények függvényeit fel is rajzolja.

6. Autópálya hálózat forgalmi viszonyainak vizsgálata szimulációval irányítás nélkül

A kifejlesztett MATLAB alapú szoftver 1. verziójának alkalmazási lehetőségeit autópálya hálózat forgalmi viszonyainak analízisére a mintahálózaton mutatjuk be. A szimuláció egyensúlyi állapotból indul, amelyet a korábban leírt technikával szintén szimulációval határoztunk meg és mentettünk a state0.mat adatfájlba. A szimulációnál használt autópálya mintahálózatot, a mintahálózat paramétereit, a magasabb szintű felhasználói paramétereket és a felhajtók forgalmi adatait az 5. fejezetben már részletesen ismertettük.

6.1 Forgalmi igények és paraméter beállítások

A felhajtókon érkező forgalmi igényekett a 6.1. ábra mutatja.



6.1. ábra. A felhajtókon érkező forgalmi igények

A szimulációs lépésköz a T_Sim=10 sec volt, azaz a dinamikus modell állapotegyenletét (irányítás nélkül) ilyen gyakorisággal kell megoldani a bemeneti adatként szolgáló elágazási ráták és 6.1. ábrabeli forgalmi igények mellett. A teljes szimulációs időintervallum $1400 \cdot \text{T\_Sim}=14000\text{sec}=3.8889\text{h}$.

6.2 Szimuláció eredményei és értékelésük

A 6.2-6.8 ábrák az L0-L6 szekciókban a forgalom sűrűséget, átlagsebességet és folyamot mutatják be (szekciónként külön-külön). A 6.9-6.11 ábrák mutatják az O1 becsatlakozásnál és az O2r, O3r felhajtóknál a várakozási sorok hosszát és a felhajtókon érkező forgalmi folyamot (demands). Irányítás hiányában $r_o(k)=1$ a kiszolgálási ráta minden felhajtónál. A lehajtóknál kimenő forgalmi folyamot minden egyes lehajtónál külön-külön, valamint az összesre összegezve a 6.12. ábra mutatja be.

A szimulációs eredmények kiértékeléséhez megismételjük a hálózat korábbi szimbólikus leírását:

Bevezető szakasz: $\xrightarrow{O_1} a \xrightarrow{L_0} b$

Fővonal: $b \xrightarrow{L_1} c \xrightarrow{L_4} d \xrightarrow{L_3} e$
$\downarrow D_{2r} \qquad \uparrow O_{2r}$

Mellékvonal: $b \xrightarrow{L_2} f \xrightarrow{L_5} g \xrightarrow{L_6} e$
$\uparrow D_{3r} \qquad \downarrow O_{3r}$

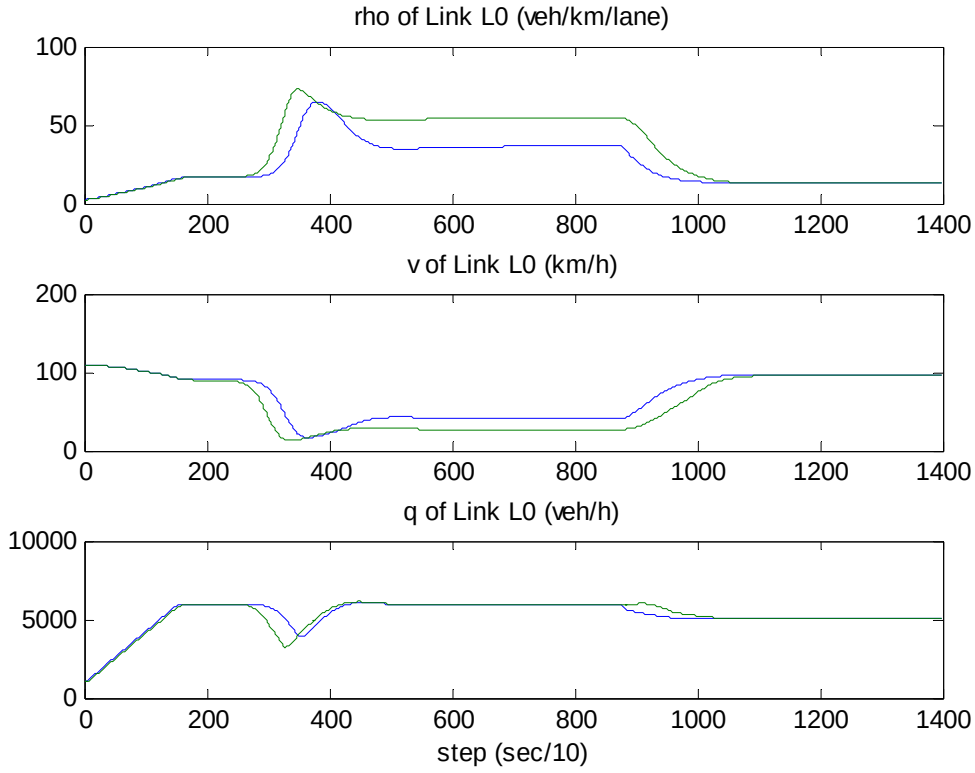
Kivezető szakasz: $e \xrightarrow{D_1}$

A szimulációs eredmények azt mutatják, hogy a fővonalon az L3 szekcióban torlódás alakul ki, az átlagsebesség nagymértékben lecsökken, lásd 6.5. ábra, és a torlódás visszafelé terjed az L4, L1, L0 szekciókra, ezekben az átlagsebesség szintén nagymértékben lecsökken, lásd 6.6, 6.3, 6.2. ábra, és az O1 becsatlakozásnál (origin) jelentős várakozási sor alakul, lásd 6.9. ábra. Eközben a mellékvonalon az L6, L5 szekcióban az átlagsebesség az elvárt normális tartományban marad, lásd 6.8, 6.7. ábra, a torlódás L0-ból csak kisebb mértékben terjed át a mellékvonalra, és az elágazást a mellékvonalon közvetlenül követő L2 szekcióban az átlagsebesség csak kisebb mértékben csökken le a sűrűségnövekedés miatt, lásd 6.4. ábra, és várakozási sor sem alakul ki a mellékvonalon, lásd 6.11. ábra.

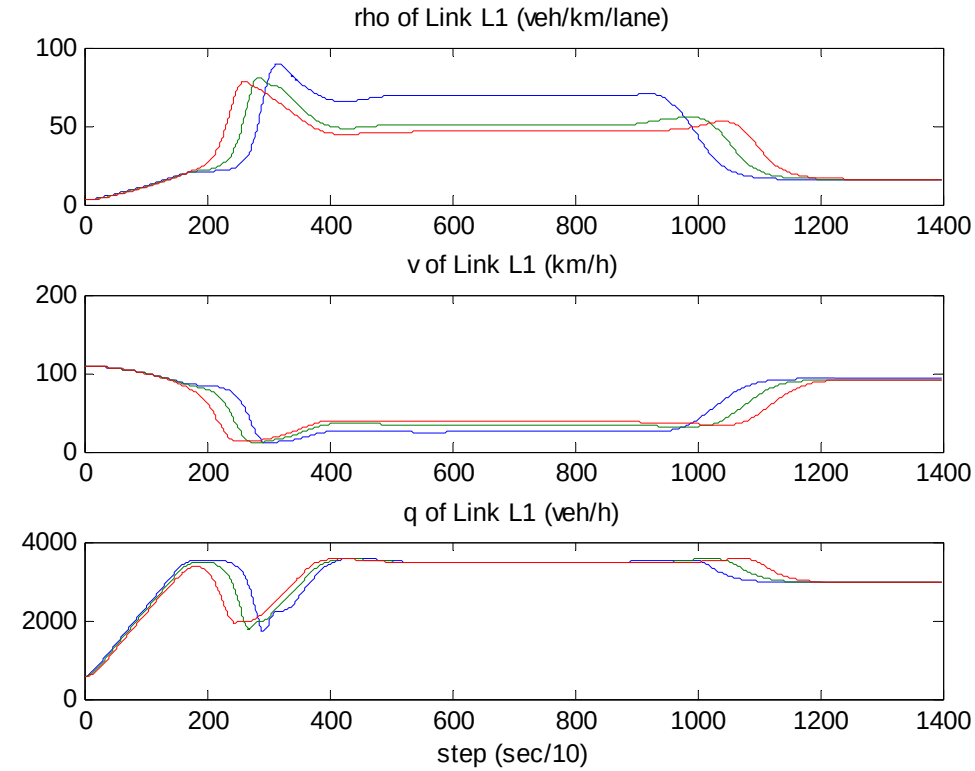
A program kiszámítja és kiadja a költségfüggvényt és annak komponenseit:

```
%*****************************************
%* Simulation without control
%* TTS=3228.21 (veh.h) total time spent
%* TTT=2262.01 (veh.h) total travel time
%* TWT=966.198 (veh.h) total waiting time at origins
%* QDC=0 total fuel consumed (weighted by af=1)
%*****************************************
```

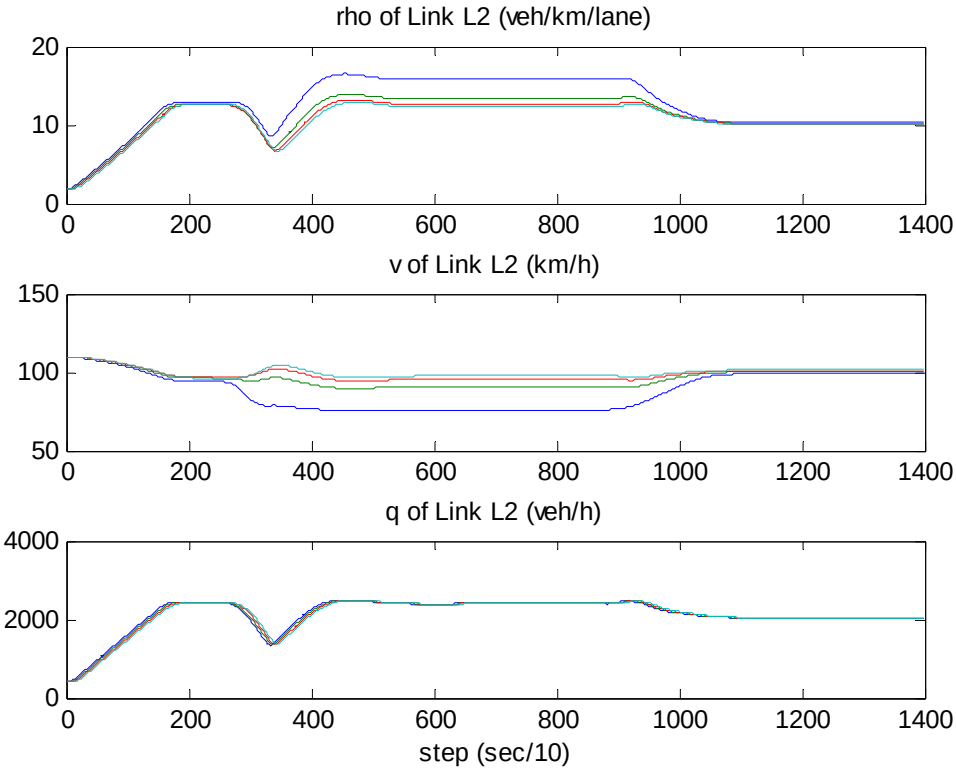
Ezek az értékek jó összehasonlítási alapul szolgálnak a későbbi vizsgálatok során az irányítás javító hatásának értékelésekor.



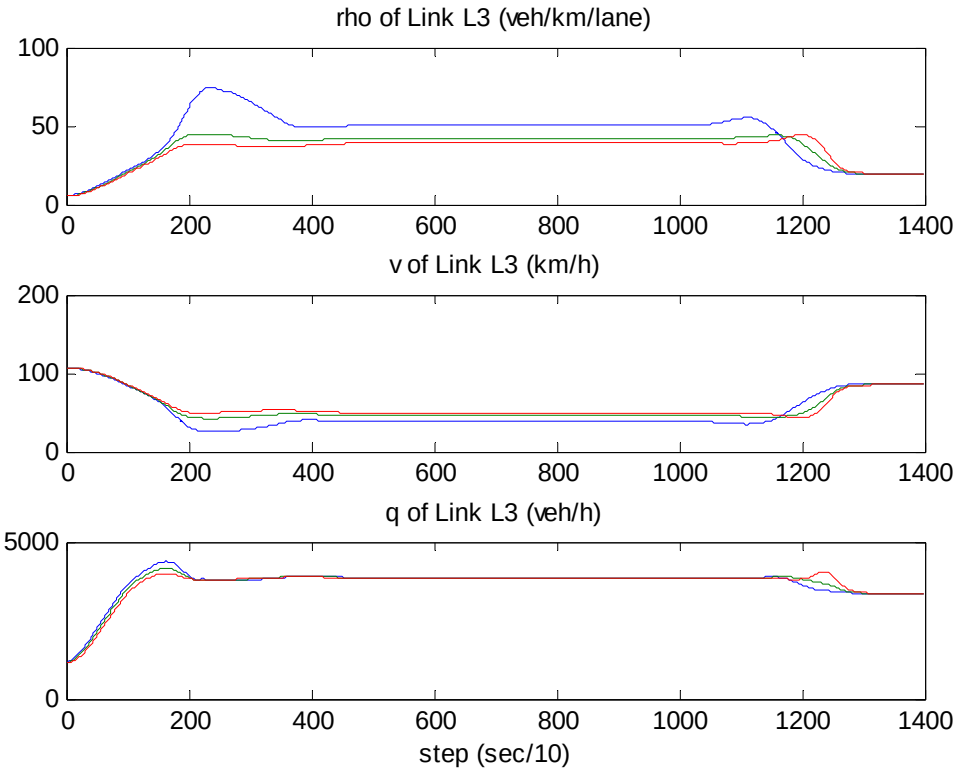
6.2. ábra. L0 szekció forgalomsűrűség, átlagsebesség és folyam



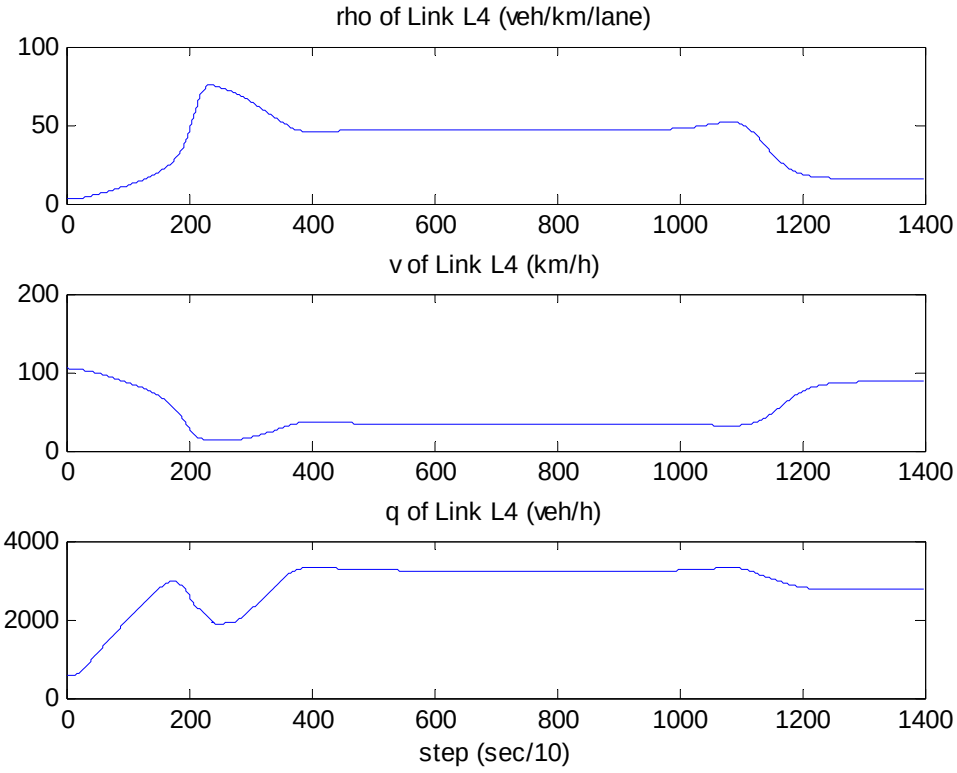
6.3. ábra. L1 szekció forgalomsűrűség, átlagsebesség és folyam



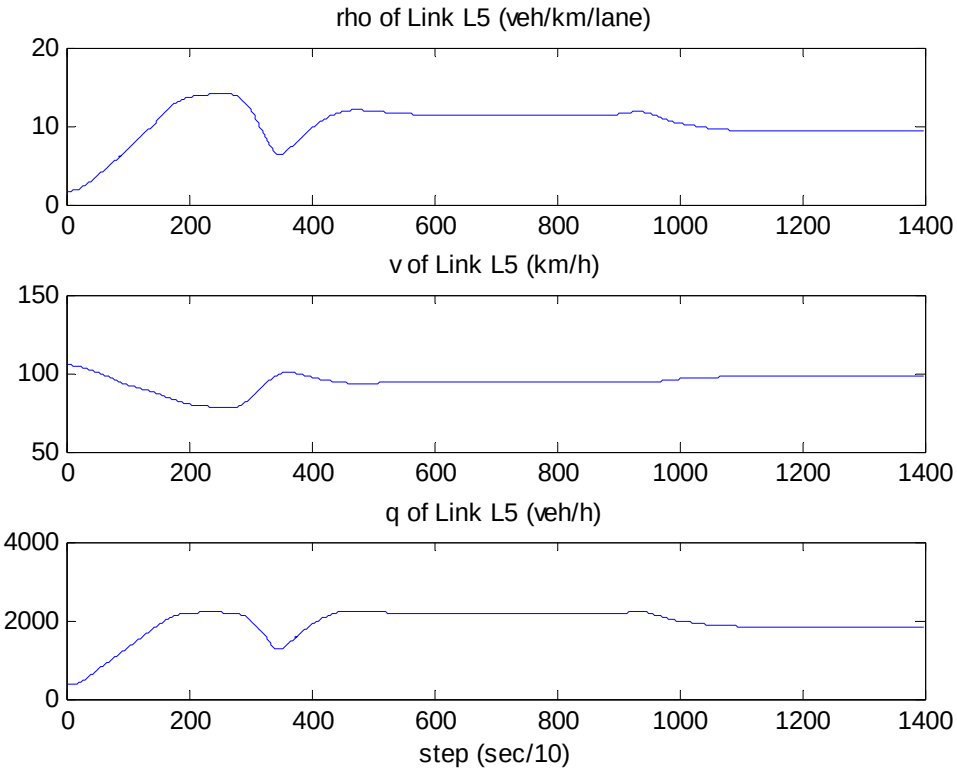
6.4. ábra. L2 szekció forgalomsűrűség, átlagsebesség és folyam



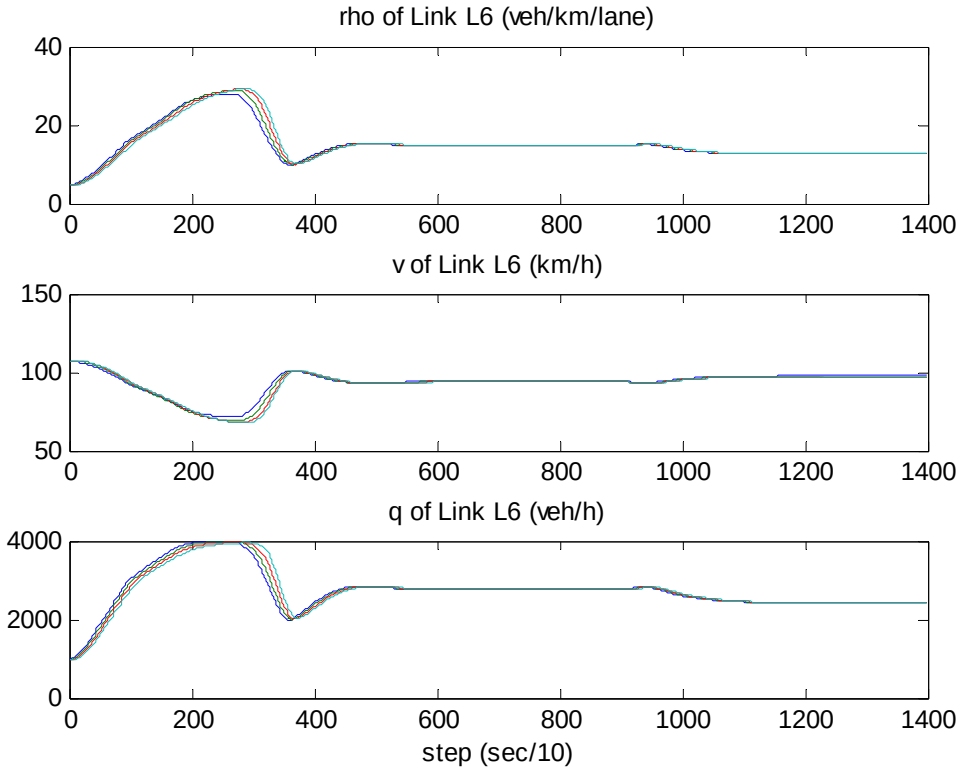
6.5. ábra. L3 szekció forgalomsűrűség, átlagsebesség és folyam



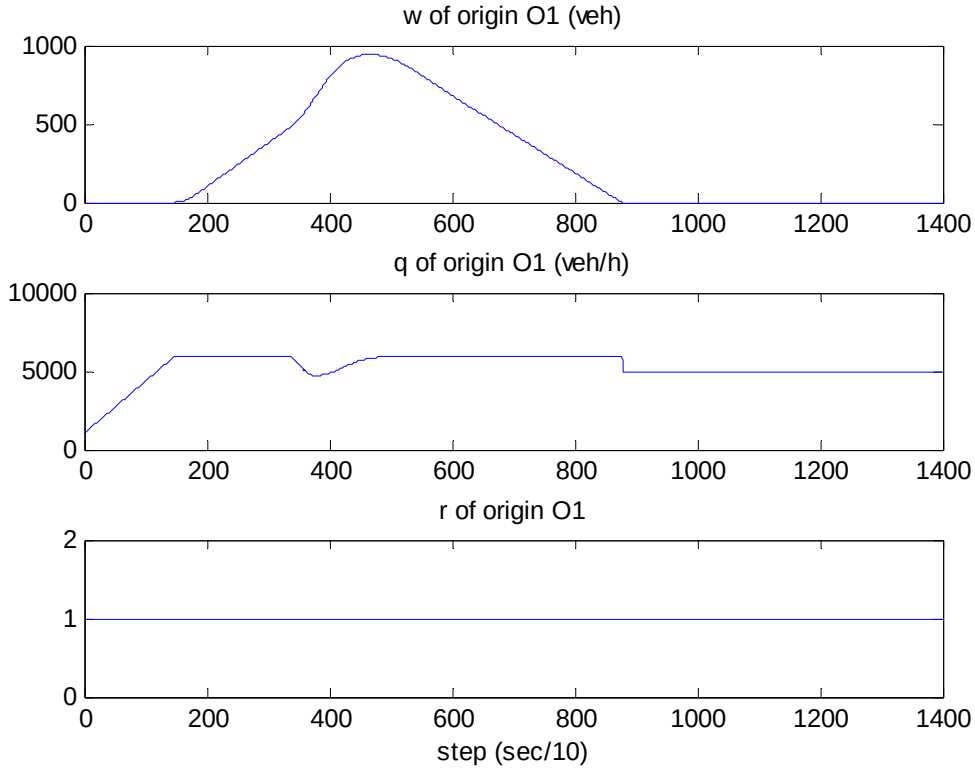
6.6. ábra. L4 szekció forgalomsűrűség, átlagsebesség és folyam



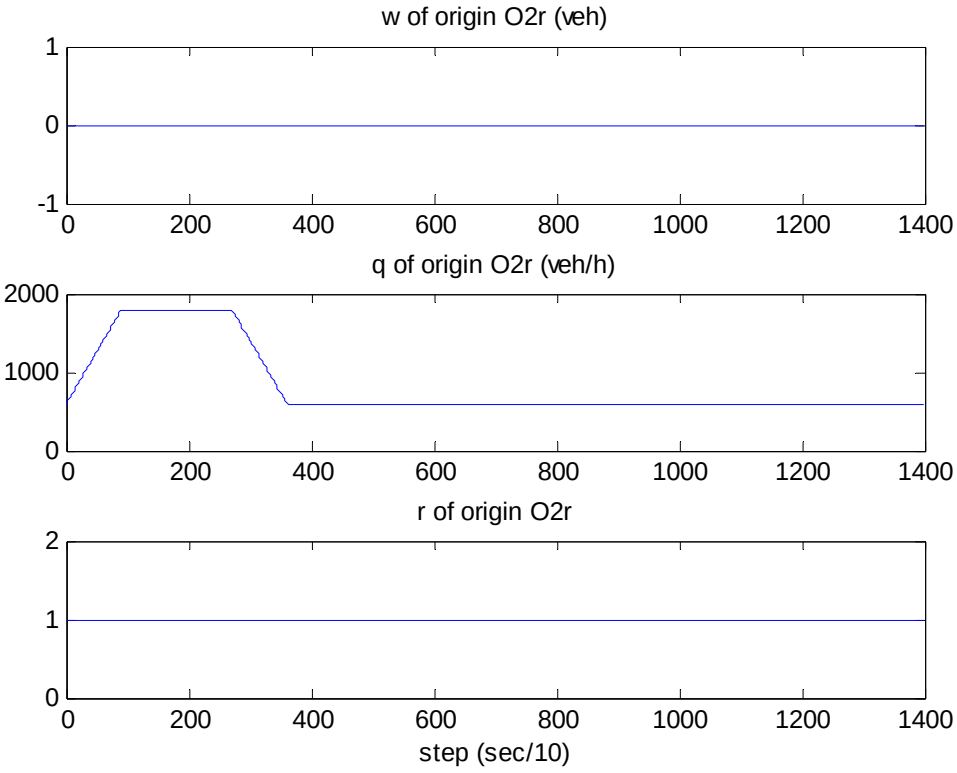
6.7. ábra. L5 szekció forgalomsűrűség, átlagsebesség és folyam



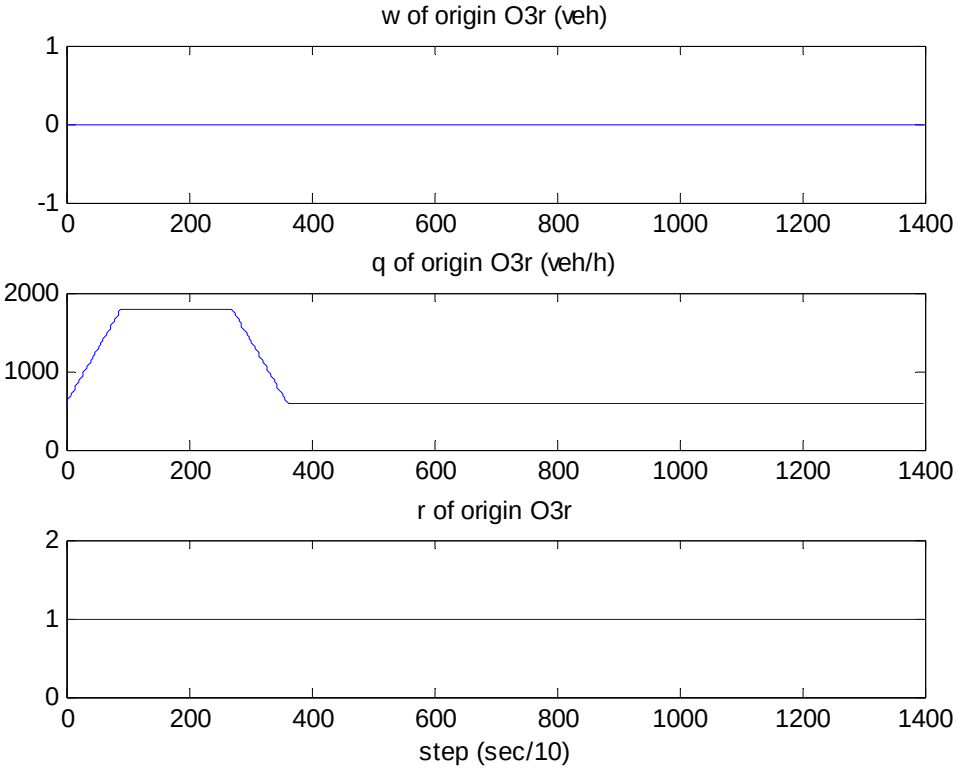
6.8 ábra. L6 szekció forgalomsűrűség, átlagsebesség és folyam



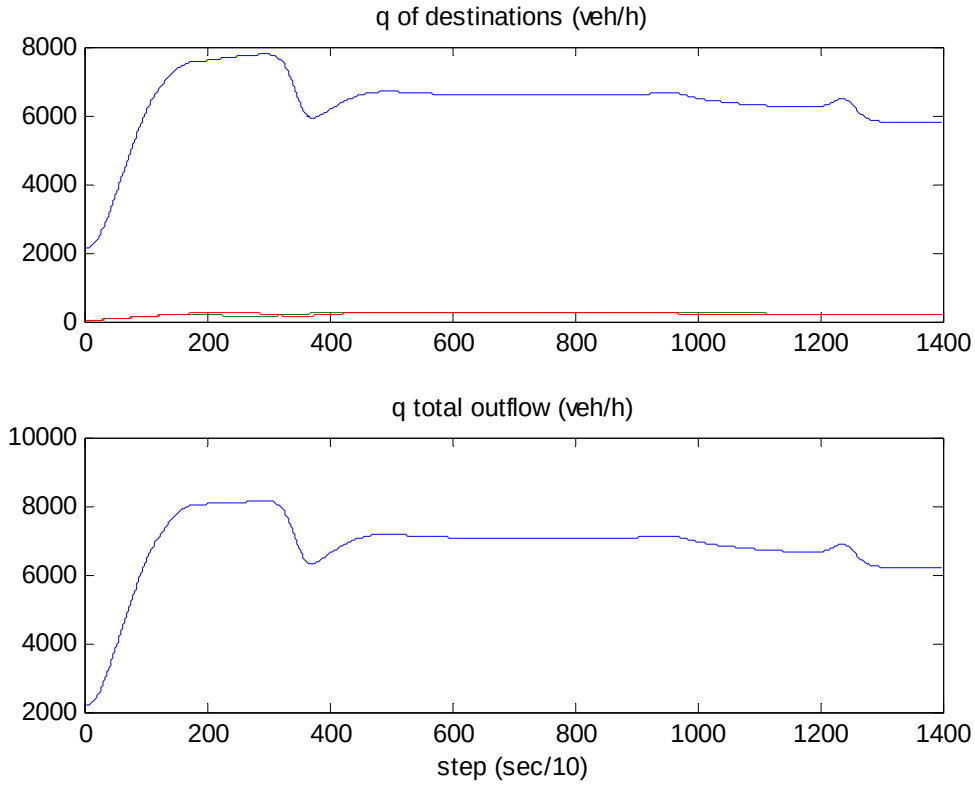
6.9 ábra. O1 becsatlakozásnál várakozási sor, folyam és kiszolgálási ráta



6.10 ábra. O2r felhajtónál várakozási sor, folyam és kiszolgálási ráta



6.11 ábra. O3r felhajtónál várakozási sor, folyam és kiszolgálási ráta



6.12 ábra. D1 kivezetés és D2r, D3r lehajtók egyenkénti és összegzett forgalom folyama

7. Autópálya hálózat forgalmi viszonyainak vizsgálata szimulációval ALINEA I-típusú irányítás esetén

A kifejlesztett MATLAB alapú szoftver 1. verziójának alkalmazási lehetőségeit autópálya hálózat forgalmi viszonyainak irányítására ALINEA I-típusú klasszikus irányítással szintén a mintahálózaton mutatjuk be. Az irányítás most is egyensúlyi állapotból indul, amelyet a korábban leírt technikával előzetes szimulációval határoztunk meg és mentettünk a state0.mat adatfájlba. Az irányítás keretében használt autópálya mintahálózatot, a mintahálózat paramétereit, a magasabb szintű felhasználói paramétereket és a felhajtók forgalmi adatait az 5. fejezetben már részletesen ismertettük.

7.1 Forgalmi igények és paraméter beállítások

A felhajtókon érkező forgalmi igény ugyanaz, mint a 6.1. ábrán. A szimulációs lépésköz T_Sim=10 sec volt, azaz a dinamikus modell állapotegyenletét (irányítással) ilyen gyakorisággal kell megoldani a bemeneti adatként szolgáló elágazási ráták és 6.1. ábra forgalmi igényei mellett. A teljes irányítási időintervallum $1400 \cdot \text{T\_Sim}=14000\text{sec}=3.8889\text{h}$. A felhajtók szabályozóinak mintavételi ideje $\text{T\_Cont}=6 \cdot \text{T\_Sim}=60$ sec=1 min volt. A felhajtók szabályozóinak Krvec erősítéseit egységesen 0.005 értékre választottuk.

7.2 Irányítási eredmények és értékelésük

A 7.1-7.7 ábrák az L0-L6 szekciókban a forgalom sűrűséget, átlagsebességet és folyamot mutatják be (szekciónként külön-külön). A 7.8-7.10 ábrák mutatják az O1 becsatlakozásnál és az O2r, O3r felhajtóknál a várakozási sorok hosszát és a felhajtókon érkező forgalmi folyamot (demands). Az irányítás az $r_o(k)$ kiszolgálási rátákat szabályozza (4.1) szerint. Az ábrák arra az esetre vonatkoznak, amikor mindhárom felhajtó (O1, O2r, O3r) irányításra van kijelölve (ezek közül csak O2r irányítása aktiválódik a forgalmi viszonyok mellett, a többi el is hagyható lenne). A lehajtóknál kimenő forgalmi folyamot minden egyes lehajtónál külön-külön, valamint az összesre összegezve a 7.11. ábra mutatja be.

A szimulációs eredmények kiértékeléséhez megismételjük a hálózat korábbi szimbólikus leírását:

Bevezető szakasz: $\xrightarrow{O_1} a \xrightarrow{L_0} b$

Fővonal: $b \xrightarrow{L_1} c \underset{\downarrow D_{2r}}{\xrightarrow{L_4}} d \underset{\uparrow O_{2r}}{\xrightarrow{L_3}} e$

Mellékvonal: $b \xrightarrow{L_2} \underset{\uparrow D_{3r}}{f} \xrightarrow{L_5} \underset{\downarrow O_{3r}}{g} \xrightarrow{L_6} e$

Kivezető szakasz: $e \xrightarrow{D_1}$

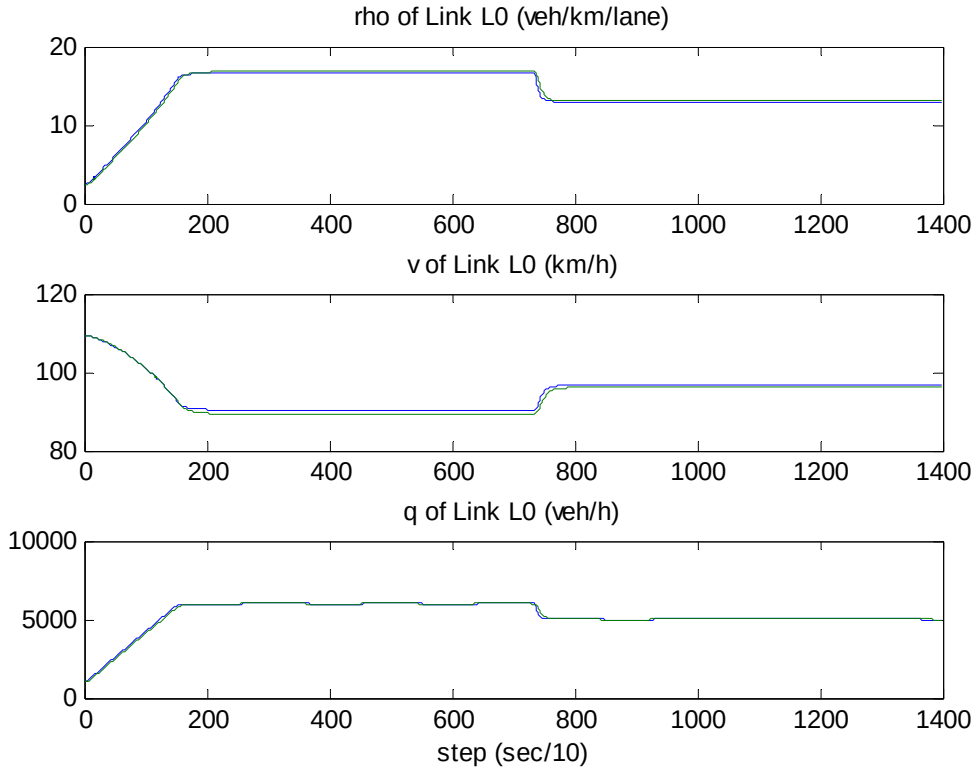
A szimulációs eredmények azt mutatják, hogy az irányítás hatására nem alakul ki torlódás a fővonalon sem, az átlagsebesség az L3 szekcióban csak kis mértékben csökken le a megnövekedett forgalom miatt az O2r felhajtón, lásd 7.4. ábra, és alig hat ki visszafelé az átlagsebességre az L4, L1, L0 szekciókban, lásd 7.5, 7.2, 7.1. ábra. Az O1 becsatlakozásnál (origin) kialakul ugyan egy várakozási sor, de ezt nem kell irányítani, lásd 7.8. ábra. Mint az várható a korábbi szimuláció eredményéből (irányítás nélkül), az O2r felhajtónál a várakozási sor és a felhajtó irányítása megoldja a fővonal tehermentesítését, lásd 7.9. ábra. Eközben a mellékvonalon az L6, L5 szekcióban az átlagsebesség az elvárt normális tartományban marad,

lásd 7.7, 7.6. ábra, a sűrűség növekedés L0-ból csak kisebb mértékben terjed át a mellékvonalra, és az elágazást a mellékvonalon közvetlenül követő L2 szekcióban az átlagsebesség csak kisebb mértékben csökken le a sűrűségnövekedés miatt, lásd 7.3. ábra, és várakozási sor sem alakul ki a mellékvonalon, lásd 7.10. ábra.

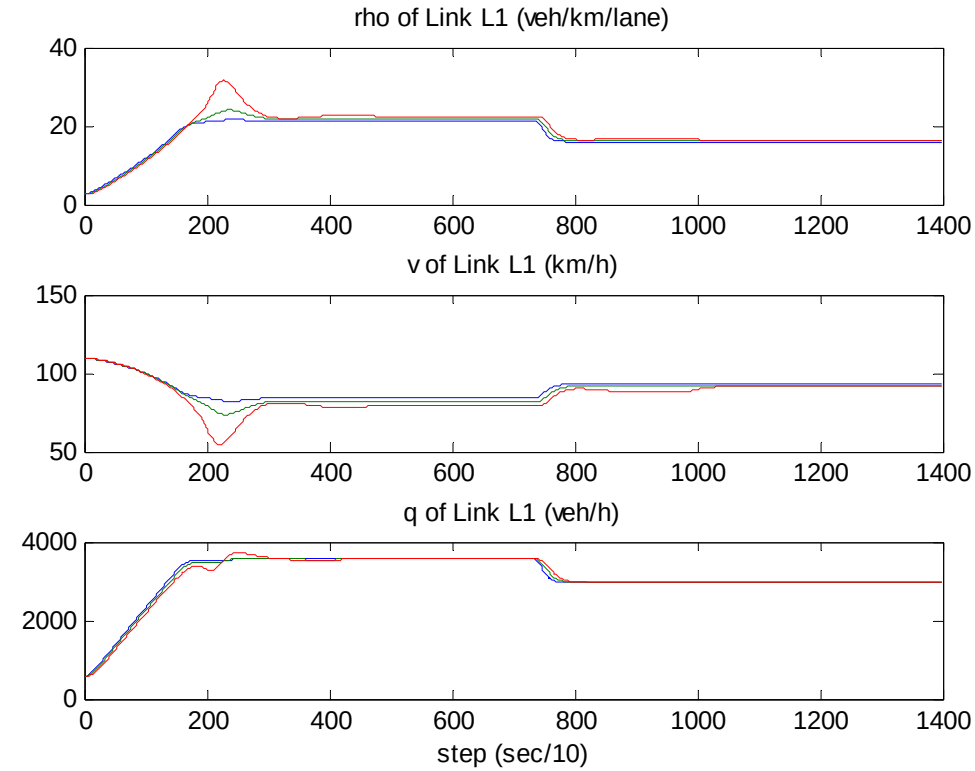
A program kiszámítja és kiadja a költségfüggvényt és annak komponenseit:

%***
%* Control: ALINEA (I)
%* Controlled origin(s): O1, O2r, O3r
%* Gains: Kr1=0.005, Kr2=0.005, Kr3=0.005
%* TTS=2914.15 (veh.h) total time spent
%* TTT=1551.77 (veh.h) total travel time
%* TWT=1362.38 (veh.h) total waiting time at origins
%* QDC=0.00253178 total fuel consumed (weighted by af=1)
%***

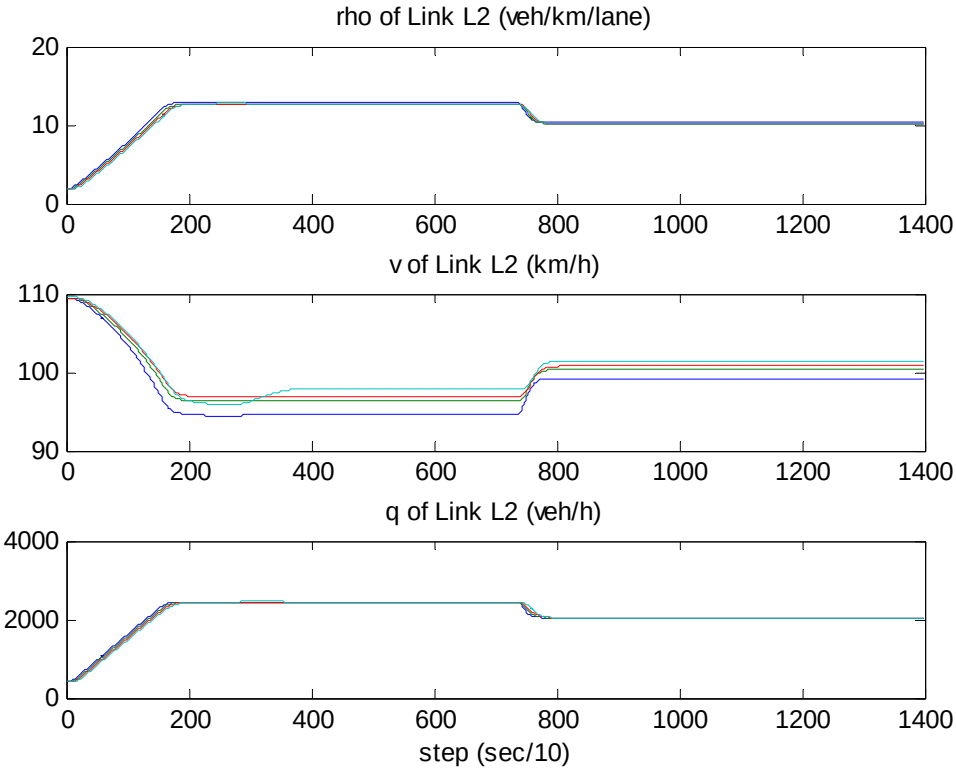
Ezek az értékek jól mutatják, hogy az ALINEA I-típusú irányítással a választott beállítások mellett a TTS érték a szimulációnál tapasztalt 3228.21 (veh.h) értékről 2914.15 (veh.h) értékre csökkent, ami $314.06 / 3228.21 \cdot 100 = 9.7\%$ javulásnak felel meg.



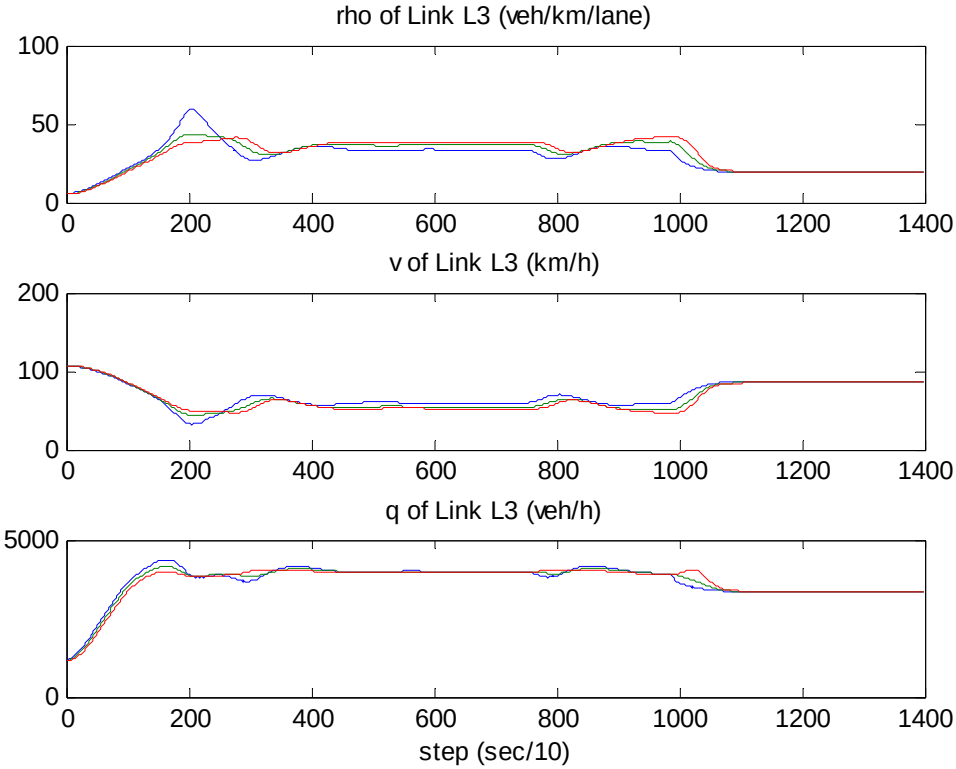
7.1. ábra. L0 szekció forgalomsűrűség, átlagsebesség és folyam (ALINEA irányítás)



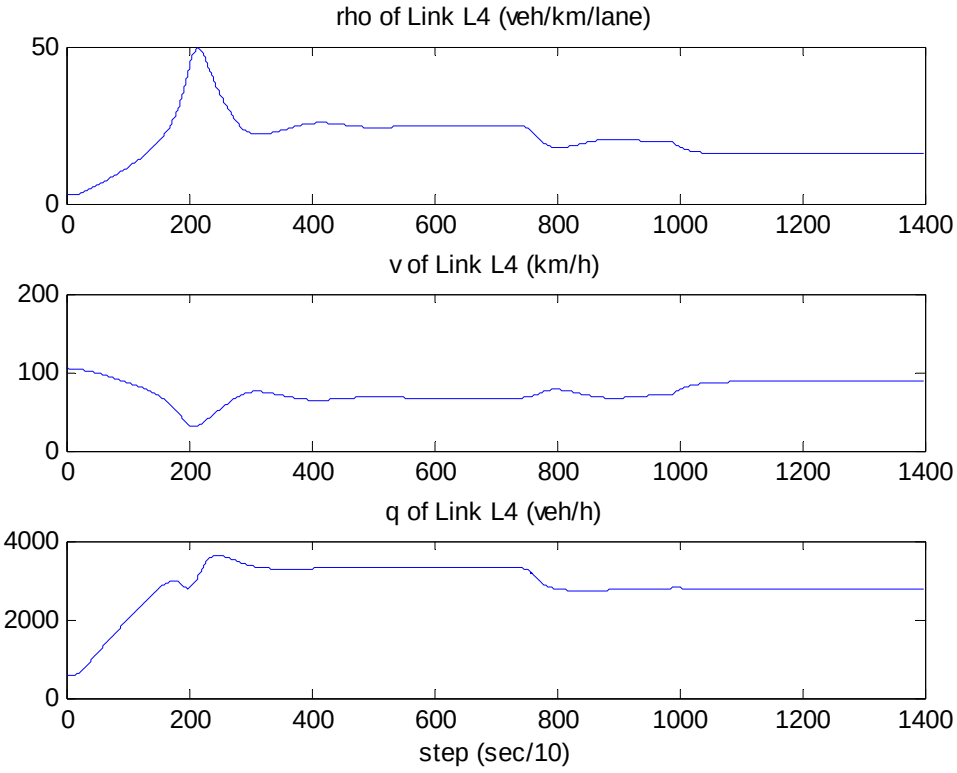
7.2. ábra. L1 szekció forgalomsűrűség, átlagsebesség és folyam (ALINEA irányítás)



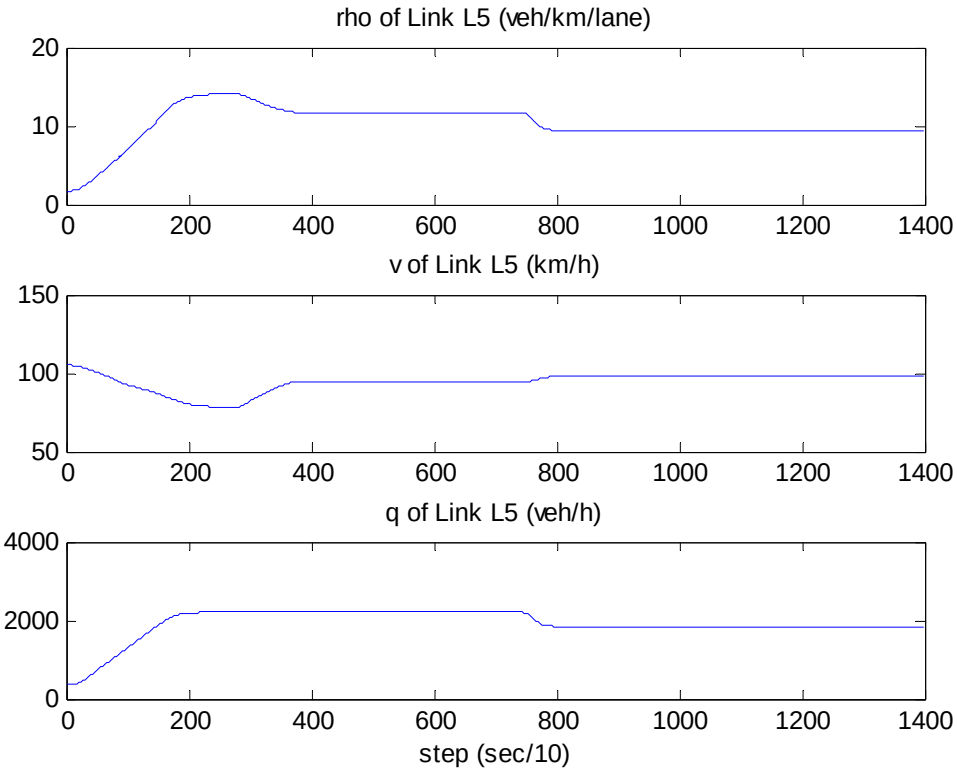
7.3. ábra. L2 szekció forgalomsűrűség, átlagsebesség és folyam (ALINEA irányítás)



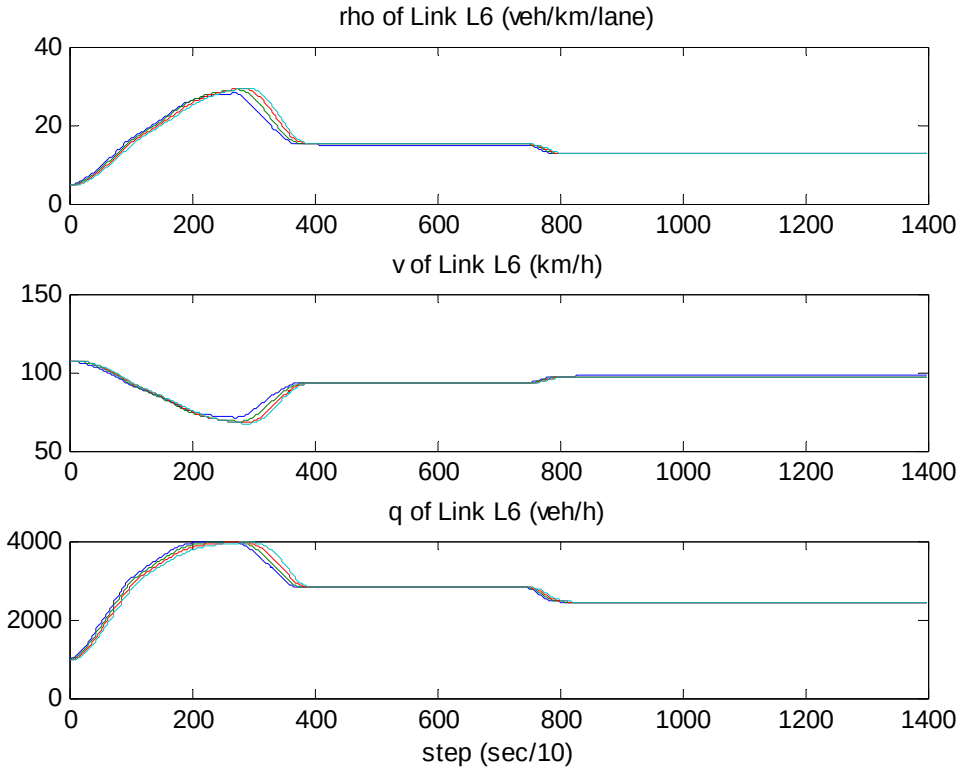
7.4. ábra. L3 szekció forgalomsűrűség, átlagsebesség és folyam (ALINEA irányítás)



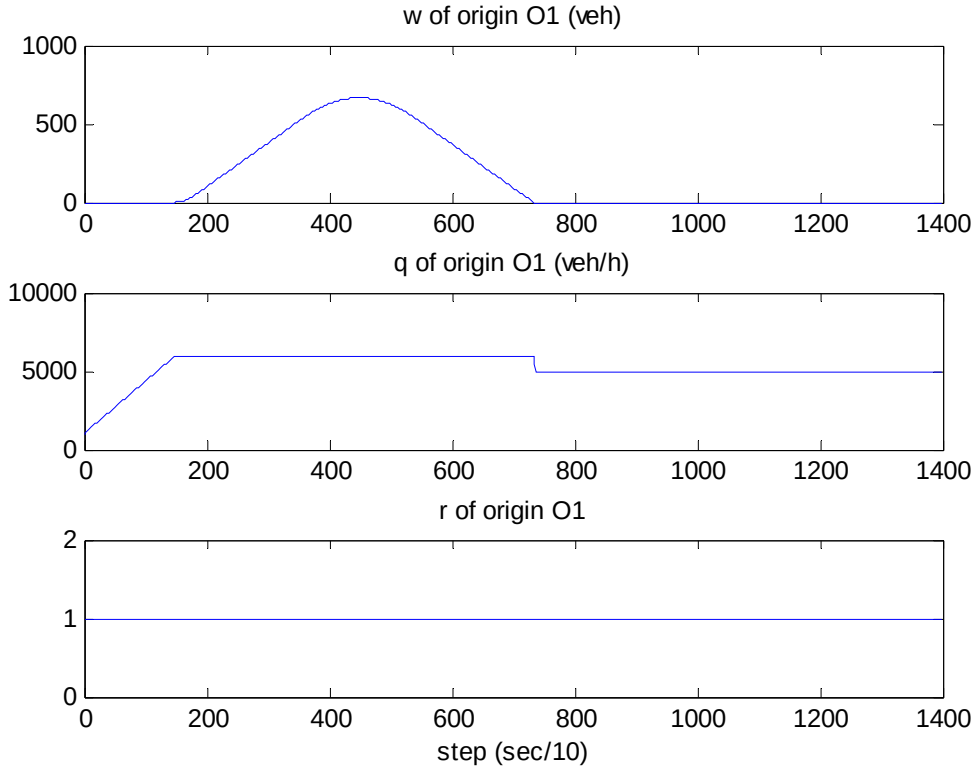
7.5. ábra. L4 szekció forgalomsűrűség, átlagsebesség és folyam (ALINEA irányítás)



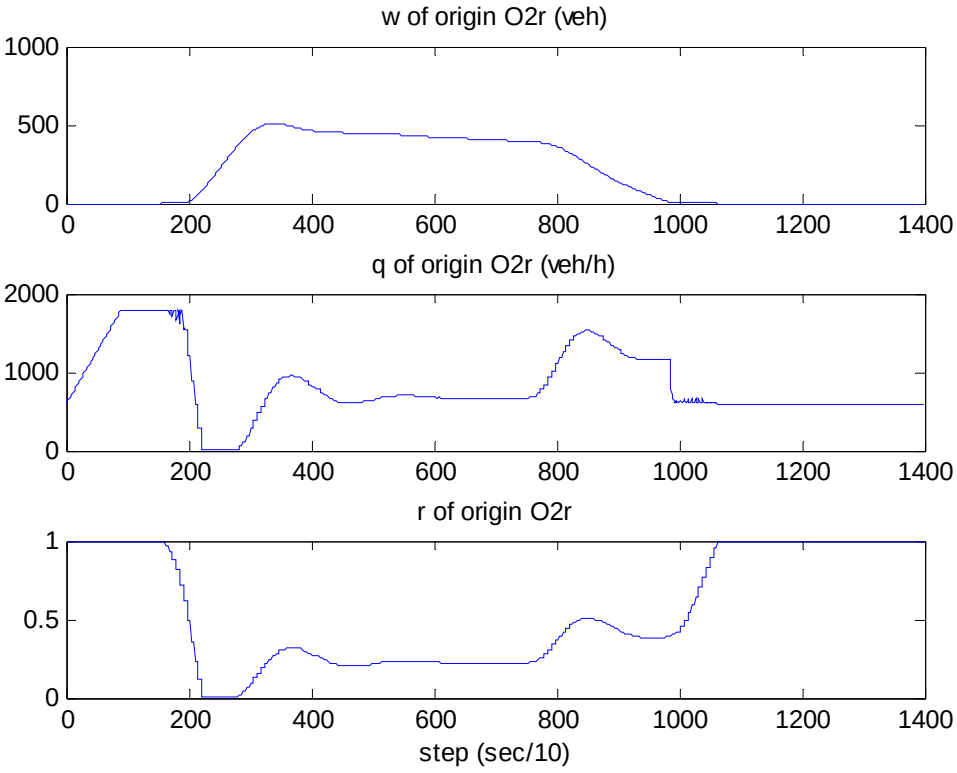
7.6. ábra. L5 szekció forgalomsűrűség, átlagsebesség és folyam (ALINEA irányítás)



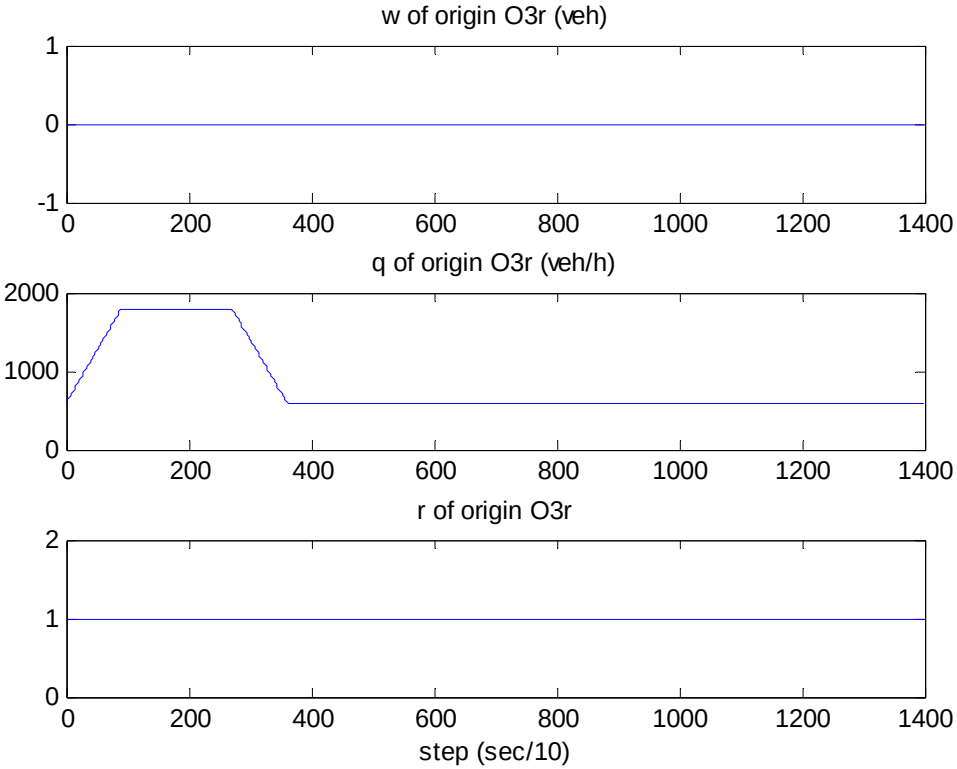
7.7. ábra. L6 szekció forgalomsűrűség, átlagsebesség és folyam (ALINEA irányítás)



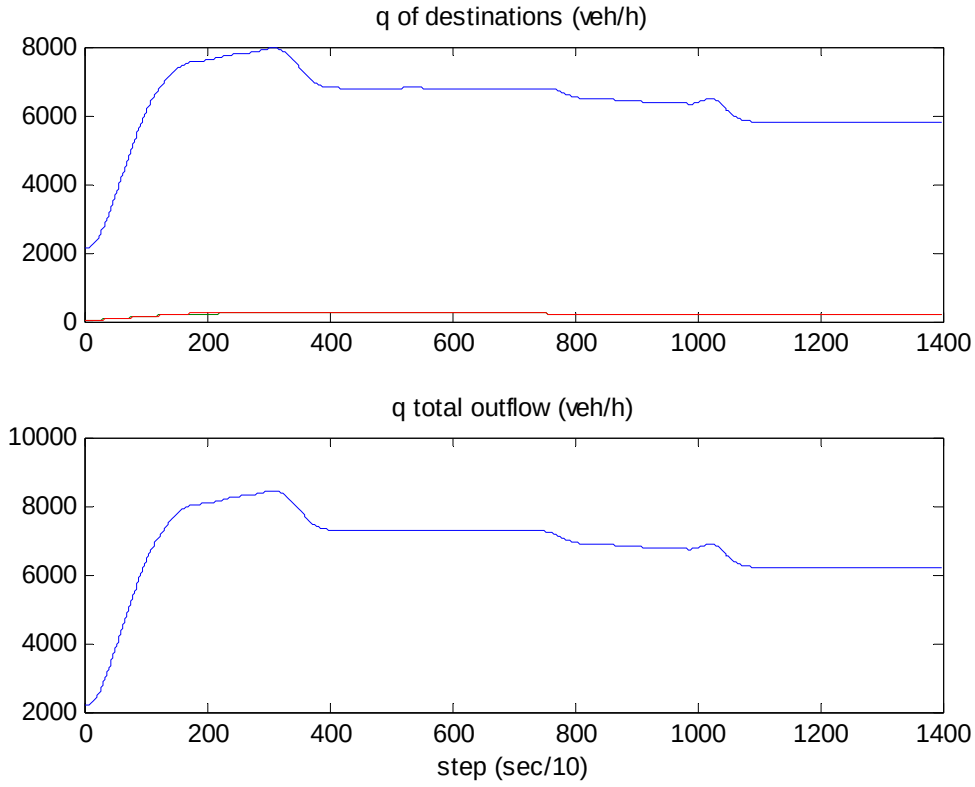
7.8. ábra. O1 becsatlakozás várakozási sor, folyam és kiszolgálási ráta (ALINEA irányítás)



7.9. ábra. O2r felhajtó várakozási sor, folyam és kiszolgálási ráta (ALINEA irányítás)



7.10. ábra. O3r felhajtó várakozási sor, folyam és kiszolgálási ráta (ALINEA irányítás)



7.11. ábra. A D1 kivezetés és a D2r, D3r lehajtók egyenkénti és összegzett forgalom folyama (ALINEA irányítás)

8. Összefoglalás

Autópálya hálózatok makromodelljén alapuló dinamikus modelleket, valamint klasszikus elven és nemlineáris prediktív irányításon alapuló irányítási algoritmusokat dolgoztunk ki irodalmi források felhasználásával. A kifejlesztett módszerek egy részhalmazát megvalósító MATLAB alapú programrendszert fejlesztettünk ki, amely alkalmas általános autópálya hálózat forgalmi viszonyainak szimulációs vizsgálatára és klasszikus szabályozások tervezésére, és ki tudja váltani az autópálya hálózat szimulációjára alkalmas, de költséges METANET szoftver lényeges szolgáltatásait.

Az autópálya hálózat több szakaszból állhat, megengedvén az elágazásokat (bifurcations) és összekapcsolódásokat (junctions) is. Kétféle dinamikus modellt fejlesztettünk ki, az első modell (nem célorientált üzemmód) csak a felhajtók forgalmának jelzőlámpákkal történő irányítását teszi lehetővé (ramp metering control), a második modell (célorientált üzemmód) ennek általánosítása arra az esetre, amikor az irányítás kezeli az OD (origin-destination) információt is, és javaslatot tesz az elágazási helyeken a kedvező útvonalra az alternatív lehetőségek közül a különféle végcélok esetén (VMS=variable message sign, DRIP=dynamic route guidance information systems panel). Mivel a vezetők a VMS jelzéseket nem szükségképpen akceptálják, ezért virtuális járművek rendszerbe injektálásával és a Logit modell elvére épülő útvonalkövetéssel lehetséges a javasolt útvonaltól való eltérés hatásának becslése is.

A kifejlesztett szoftver a végső programrendszer 1. verziójának tekinthető. A programrendszer MATLAB-alapú, megvalósítja a kifejlesztett dinamikus modellt a nem-célorientált esetben, lehetővé teszi általános autópálya hálózatok forgalmi viszonyainak meghatározását irányítás nélkül szimuláció keretében, továbbá az autópálya hálózat irányítását egyszerű irányítási stratégia, az ALINEA I-típusú irányítás esetén.

A szimulációs üzemmód biztosítja a forgalomtorlódás okainak és helyeinek felderítését csúcsforgalmi időszakban az autópályán. A szimuláció speciális esetként, konstans bejövő forgalmakat feltételezve a felhajtókon, alkalmas az egyensúlyi állapot (steady state) meghatározására is. Ezáltal biztosítható, hogy az irányítások egyensúlyi helyzetből is indíthatók legyenek, és a kezdeti feltétel miatti tranziensek és az irányítási tranziensek ne keveredjenek össze a nemlineáris rendszerben. A szimuláció eredményei alapján megválasztható, hogy mely felhajtó vagy felhajtók esetén célszerű irányítást alkalmazni a forgalmi viszonyok javítása érdekében. Az irányítás során feltételeztük, hogy a felhajtók jelzőlámpákkal vannak ellátva és a forgalomat a felhajtók jelzőlámpáinak átbocsátási ideje révén lehet szabályozni (ramp metering). Az irányítás hatása a forgalom alakulására a tárolt forgalmi tranziensek kiértékelésével elemezhető. A tranziensek dokumentálása grafikusan történik, az eredmények a MATLAB szolgáltatásaival dokumentumokba menthetők.

A forgalmi viszonyok magas szintű és tömör numerikus jellemzésére költségfüggvény szolgál, amely tartalmazza a teljes hálózatban töltött időt (TTS=TTT+TWT, [veh.h]) , a teljes utazási időt (TTT, [veh.h]), a felhajtók várakozási soraiban töltött időt (TWT, [veh.h]) és a beavatkozó jel változásának négyzetösszegét (QDC).

Különös figyelmet fordítottunk az autópálya struktúra definiálásának megkönnyítésére a felhasználó szemszögéből nézve, amely egy mintafájl átírásával végezhető el. A hálózat struktúráját a program automatikusan olyan adatstruktúrákká konvertálja, amelyek lecsökkentik a valós időben szükséges számításokat, növelve ezáltal a valósidejűség elérésének lehetőségét. Az autópálya szekciók és szegmenseik forgalmi jellemzői (sűrűség, átlagsebesség, folyam), továbbá a felhajtók (várakozási sorok, folyamok, kiszolgálási ráták) és lehajtók (lehajtónkénti folyamok, teljes folyam) forgalmi jellemzői automatikusan kigyűjtésre kerülnek, és felhasználásra kerülnek mind a grafikus megjelenítés során, mind pedig a költségfüggvény számításkor.

A tapasztalatra alapozva a jövőben folytatható a programrendszer bővítése nemlineáris prediktív irányítással és célorientált üzemmódú irányítással. A program első verziójának kifejlesztésekor a bővítések későbbi beillesztésének igényét figyelembe vettük.

9. Felhasznált irodalom

1. Allgöver, F., & Zheng, A. ed. (2000). *Nonlinear predictive control.* Basel: Birkhäuser.
2. Bellemans, T., De Schutter, B., & De Moor, B. (2005). Model predictive control of ramp metering of motorway traffic: A case study. *Control Engineering Practice*, 14, pp. 757-768.
3. Bellemans, T. (2003). *Traffic control on motorways.* PhD Thesis. Katholieke Universiteit, Leuven, p. 194.
4. Cramer, J. (2001). *An introduction of the Logit model for Economists*. Timberlake Consultants Press, UK, 2001.
5. Cremer, M. (1995). On the calculation of individual travel times by macroscopic models. *Proc. 1995 Vehicle Navigation and Information Systems Conference*, Washington, pp. 187-193.
6. Karami, A., Hegyi, A., De Schutter, B., Hallendorn, H., and Middelham, F. (2004). Integration of dynamic route guidance and freeway ramp metering using model predictive control. *Proc. 2004 American Control Conference*, Boston, pp. 5533-5538.
7. Kim, H., & Shim, D. (2003). A flight control system for aerial robots. *Control Engineering Practice,* 11, pp. 1389-1400.
8. Kotsialos, A., Papageorgiu, M., Mangeas, M., & Haj-Salem, H. (2002). Coordinated and integrated control of motorway networks via non-linear optimal control. *Transportation Research Part C,* 10 , pp. 65-84.
9. Lantos, B. (2003). *Irányítási rendszerek elmélete és tervezése II. Korszerű szabályozási rendszerek.* Budapest: Akadémiai Kiadó.
10. Lantos, B., & Kiss, B. (2005). Nonlinear model predictive control of robots, cranes and vehicles. *Proceedings of the IFAC World Congress, Prague,* Paper WE-A04-TP/6 02495.pdf
11. Lantos, B. (2005). *Autópálya forgalom és jármű irányítások.* Tanulmány és előzetes rendszerterv. RET 1.1 Járműforgalmi rendszerek modellezése és irányítása projekt, EJJT-BME IIT, p. 81.
12. Lantos, B. (2006). Nonlinear model predictive control of robots, cranes and ground vehicles. *Proc. of the Workshop on System Identification and Control Systems, Budapest.* BUTE Advanced Vehicles and Vehicle Control Knowledge Center, pp. 201-215.
13. Papageorgiou, M., Haj-Salem,, H., & Blosseville, J. (1991). ALINEA: a local feedback control law for ramp metering. *Transportation Research Record*, 1320, pp. 58-64.
14. Payne, H. (1971) Models of freeway trafic and control. *In G. A. Bekey (ed.) Mathematical models of public systems, simulation council proceedings series,* vol. 1 (pp. 51-61).
15. Theil, H. (1969). A multinominal extension of the linear logit model. *International Economic Review*, 10, pp. 251-259.

10. MATLAB programlisták

A függvények funkciói a függvény prototípus alakja után álló egysoros kommentből azonosíthatók. A magyarázat követi az 2.-5. fejezetek algoritmusaiban használt jelöléseket.

10.1 frameMotorway

```
%frameMotorway.m
%Frame program of Motorway Control

clear all
close all


global CONTR_MODE
global N_MWL N_ORL N_DEL N_SAFL N_DUL N_LFULL
global N_NODE N_CORIG N_VMSD N_WQUEUE N_CONTR N_VMSWAYS
global N_STATE N_STATEext
global MWL_Names ORL_Names DEL_Names SAFL_Names DUL_Names
global NODE_Names CORIG_Names VMSD_Names
global NODE_Table LINK_Table WQUEUE_Table CONTR_Table VMSWAYS_Table
global STATE_Table DEMAND_Table
global T_Sim T_Cont T_Full N_Cont N_Hor X_act X_extact TX_tran
global ORLtoNODE_Table DELfromNODE_Table

global RVQL_tran WQL_tran WDEL_tran
global RVEC_OLD

Snet=initnetwork;

%---------------------------------
%Necessary before netw2tabs
%CONTR_MODE: 1=simul, 2=ALINEA
CONTR_MODE=1;
Krvec=[];
%---------------------------------

netw2tabs(Snet);

inithighpar;

initstate;

RVEC_OLD=ones(N_ORL,1);

%---------------------------------
%init ALINEA
if CONTR_MODE==2
    Krvec=[0.005 0.005 0.005];
    for i=1:N_CONTR
        corl=CONTR_Table{i}.data(1);
        CONTR_Table{i}.data(2)=Krvec(corl);
    end;
end;
%---------------------------------

Ntt=size(DEMAND_Table,1);
diml=sum(STATE_Table(1:N_LFULL,4));
%----------------------
ntt=Ntt;
```

```
%----------------------
RVQL_tran=zeros(ntt,3*diml+1);
WQL_tran=zeros(ntt,3*N_ORL+1);
QDEL_tran=zeros(ntt,N_DEL+1);

for k=1:ntt
    %Time in tact of T_Sim
    tk=k-1; %Tact instead of time
    TX_tran(k,:)=[tk X_act'];
    %New rho,v computation for full network
    [X_act,Qdest_LDEL]=funmw(tk);
    %Store new network data for transients
    WQL_tran(k,1)=tk;
    col=1;
    for orig=1:N_ORL
        rwext=STATE_Table(N_LFULL+orig,5);
        vecw=X_extact(rwext,1:3);
        WQL_tran(k,col+1:col+3)=vecw;
        %wo,qo,ro
        col=col+3;
        RVEC_OLD(orig)=vecw(3);
    end;
    %
    RVQL_tran(k,1)=tk;
    col=1;
    for m=1:N_LFULL
        dmext=STATE_Table(m,4);
        rsext=STATE_Table(m,5);
        reext=STATE_Table(m,6);
        matm=X_extact(rsext:reext,1:3);
        vecm=matm(:);
        RVQL_tran(k,col+1:col+3*dmext)=vecm';
        %rho,v,q
        col=col+3*dmext;
    end;
    %
    QDEL_tran(k,1)=tk;
    QDEL_tran(k,2:N_DEL+1)=Qdest_LDEL';
end;

%Plot transients
plotrvq(RVQL_tran);
plotwq(WQL_tran);
plotqdest(QDEL_tran,N_DEL,T_Sim);

%---------------------------------
%Find steady state
%1) Choose appropriate demands in inithighpar
%2) Delete state0.m or comment out test of existence in initstate
%3) Choose simulation, and eliminate '%' of next instruction
%save state0 X_act
%4) start frameMotorway
%---------------------------------
%Simulation or control
%5) Choose real demands for investigation in inithighpar
%6) Uncomment test of existence in initstate
%7) Set '%' of previous instruction
%8) Start frameMotorway for simulation or control
%---------------------------------

%Compute cost function
```

```
af=1;
[TTS,TTT,TWT,QDC]=costfun(RVQL_tran,WQL_tran,T_Sim,N_Cont,af);

%Document results
fprintf('\n');
fprintf('%%*****************************************\n');
switch CONTR_MODE
    case 1
        fprintf('%%* Simulation without control\n');
    case 2
        fprintf('%%* Control: ALINEA (I)\n');
        fprintf('%%* Controlled origin(s): ');
        for i=1:N_CONTR
            si=CONTR_Table{i}.n;
            fprintf('%s',si);
            if i<N_CONTR, fprintf(', ');
            else
                fprintf('\n');
            end;
        end;
        fprintf('%%* Gains: ');
        for i=1:N_CONTR
            Kri=CONTR_Table{i}.data(2);
            si=num2str(i);
            fprintf(['Kr%s=%g'],si,Kri);
            if i<N_CONTR, fprintf(', ');
            else
                fprintf('\n');
            end;
        end;
    otherwise
        ;
end;
fprintf('%%* TTS=%g (veh.h) total time spent\n',TTS);
fprintf('%%* TTT=%g (veh.h) total travel time\n',TTT);
fprintf('%%* TWT=%g (veh.h) total waiting time at origins\n',TWT);
fprintf('%%* QDC=%g total fuel consumed (weighted by
af=%g)\n',QDC,af);
fprintf('%%*****************************************\n');
fprintf('\n');
```

10.2 initnetwork()

```
function Snet=initnetwork
%Initialize motorway network

%clear all

Snet=struct('Pars',[],...
    'MotorwayLinks',[],...
    'OriginLinks',[],...
    'DestinationLinks',[],...
    'StoreAndForwardLinks',[],...
    'DummyLinks',[],...
    'NodeModels',[],...
    'ControlledOrigins',[],...
    'VMSforDestinations',[]);

Snet.Pars=struct('tau',18,'kappa',40,'nu',60,...
    'vfm',110,'rhocr',33.5,'am',1.636,...
    'rhomax',180,...
    'delta',0.0122,'phi',2.98,...
    'Q0',1500);
%Q0 [veh/h/lane] is the capacity of origin links in case of free flow
%Snet.MotorwayLinks={Name1,[Pars1]; Name2, [Pars2];...}
%[Parsi]: [SectionLength Lane SegmentLength] %Length in km
%Number of segments in the section: SectionLength/SegmentLength;
Snet.MotorwayLinks={...
    'L0',[2 4 1];...
    'L1',[3 2 1];...
    'L2',[4 2 1];...
    'L3',[3 2 1];...
    'L4',[0.5 2 0.5];...
    'L5',[0.5 2 0.5];...
    'L6',[4 2 1]};
%
%NMLinks=size(Snet.MotorwayLinks,1);
%MLinks=cell(NMLinks,2); Mlinks=Snet.MotorwayLinks;
%MLinksnamei=MLinks{i,1};
%MLinksparsi=Mlinks{i,2};

Snet.OriginLinks={'O1','O2r','O3r'};

Snet.DestinationLinks={'D1','D2r','D3r'};

%Snet.NodeModels={Node1,{Iset1},{Oset1},{Tset1};
Node2,{Iset2},{Oset2},{Tset2};...};
%Example by Kotsialos et al.
%x*y=0.408*0.098=0.04
%(1-x)*z=0.592*0.0676=0.04
%(x*(1-y))/((1-x)*(1-z))=0.368/0.552=0.4/0.6=0.6667
%x*(1-y)+(1-x)&(1-z)=0.368+0.552=0.92
Snet.NodeModels={...
    'a',{'O1'},{'L0'},{1};...
    'b',{'L0'},{'L1','L2'},{0.592, 0.408};...
    'c',{'L1'},{'L4','D2r'},{0.9324, 0.0676};...
    'd',{'L4','O2r'},{'L3'},{1};...
    'e',{'L3','L6'},{'D1'},{1};...
    'f',{'L2'},{'L5','D3r'},{0.902, 0.098};...
    'g',{'L5','O3r'},{'L6'},{1}};
%
```

```
%NModels=size(Snet.NodeModels,1);
%Models=cell(NModels,3); Models=Snet.NodeModels;
%NodeNamei=Models{i,1};
%NodeIseti=Models{i,2};
%NodeOseti=Models{i,3};
%NNodeOseti=length(NodeOseti); %Equivalent to length(Models{i,3})
%NodeOsetij=NodeOseti{j}; %Equivalent to Models{i,3}{j}

Snet.ControlledOrigins={'O1','O2r','O3r'};
%
%NControlledOrigins=length(Snet.ControlledOrigins);
%COrigini=Snet.ControlledOrigins{i};

Snet.VMSforDestinations={'D1','D2r','D3r'};
%NVMSforDestinations=length(Snet.VMSforDestinations);
%VMSDi=Snet.VMSforDestinations{i};
```

10.3 netw2tabs()

```
function netw2tabs(Snet)
%Convert network description to tables

global CONTR_MODE
global N_MWL N_ORL N_DEL N_SAFL N_DUL N_LFULL
global N_NODE N_CORIG N_VMSD N_WQUEUE N_CONTR N_VMSWAYS
global N_STATE N_STATEext
global MWL_Names ORL_Names DEL_Names SAFL_Names DUL_Names
global NODE_Names CORIG_Names VMSD_Names
global NODE_Table LINK_Table WQUEUE_Table CONTR_Table VMSWAYS_Table
global STATE_Table DEMAND_Table
global T_Sim T_Cont T_Full N_Cont N_Hor X_act X_extact TX_tran
global ORLtoNODE_Table DELfromNODE_Table

%Find table dimensions
N_MWL=length(Snet.MotorwayLinks);
N_ORL=length(Snet.OriginLinks);
N_DEL=length(Snet.DestinationLinks);
N_SAFL=length(Snet.StoreAndForwardLinks);
N_DUL=length(Snet.DummyLinks);
N_NODE=length(Snet.NodeModels);
N_CORIG=length(Snet.ControlledOrigins);
N_VMSD=length(Snet.VMSforDestinations);

%Find symbol tables
MWL_Names={};
ORL_Names={};
DEL_Names={};
SAFL_Names={};
DUL_Names={};
NODE_Names={};
CORIG_Names={};
VMSD_Names={};
if N_MWL~=0, MWL_Names=Snet.MotorwayLinks(1:N_MWL,1); end;
if N_ORL~=0, ORL_Names=Snet.OriginLinks(1,1:N_ORL)'; end;
if N_DEL~=0, DEL_Names=Snet.DestinationLinks(1,1:N_DEL)'; end;
if N_SAFL~=0, SAFL_Names=Snet.StoreAndForwardLinks(1:N_SAFL,1); end;
if N_DUL~=0, DUL_Names=Snet.DummyLinks(1:N_DUL,1); end;
if N_NODE~=0, NODE_Names=Snet.NodeModels(1:N_NODE,1); end;
if N_CORIG~=0, CORIG_Names=Snet.ControlledOrigins(1,1:N_CORIG)'; end;
if N_VMSD~=0, VMSD_Names=Snet.VMSforDestinations(1,1:N_VMSD)'; end;

%Find ISET of nodes
NODE_Table={};
for i=1:N_NODE
    Si=cell(1,4);
    Si=Snet.NodeModels(i,1:4);
    n=Si{1}; Iset=Si{2}; Oset=Si{3}; Tset=Si{4};
    %ntype: 1=start, 2=normal, 3=end, 4=bifurcation, 5=junction
    S=struct('n',[],'ntype',[],...
        'IsetMWL',[],'IsetORL',[],'IsetSAFL',[],'IsetDUL',[],...
        'OsetMWL',[],'OsetDEL',[],'OsetSAFL',[],'OsetDUL',[],...
        'TsetMWL',[],'TsetDEL',[],'TsetSAFL',[],'TsetDUL',[],...
        'IsetLF',[],'OsetLF',[],'TsetLF',[]);
    S(1).n=n;
    NIset=length(Iset);
    if NIset~=0
        for k=1:NIset
```

```
        sk=Iset{k};
        [isk,typesk]=sym2type(sk);
        switch typesk
            case 1
                S.IsetMWL=[S.IsetMWL isk];
            case 2
                S.IsetORL=[S.IsetORL isk];
            case 4
                S.IsetSAFL=[S.IsetSAFL isk];
            case 5
                S.IsetDUL=[S.IsetDUL isk];
            otherwise
                fprintf('Wrong Iset for node %s\n',sk);
                return;
        end;
    end;
end;
NOset=length(Oset);
if NOset~=0
    for k=1:NOset
        sk=Oset{k};
        [isk,typesk]=sym2type(sk);
        betask=Tset{k};
        switch typesk
            case 1
                S.OsetMWL=[S.OsetMWL isk];
                S.TsetMWL=[S.TsetMWL betask];
            case 3
                S.OsetDEL=[S.OsetDEL isk];
                S.TsetDEL=[S.TsetDEL betask];
            case 4
                S.OsetSAFL=[S.OsetSAFL isk];
                S.TsetSAFL=[S.TsetSAFL betask];
            case 5
                S.OsetDUL=[S.OsetDUL isk];
                S.TsetDUL=[S.TsetDUL betask];
            otherwise
                fprintf('Wrong Oset for node %s\n',sk);
                return;
        end;
    end;
end;
nilinks=length(S.IsetMWL)+length(S.IsetSAFL)+length(S.IsetDUL);
nolinks=length(S.OsetMWL)+length(S.OsetSAFL)+length(S.OsetDUL);
%noriglinks=length(S.IsetORL);
%ndestlinks=length(S.OsetDEL);
%--------------------------------
if nilinks==0, ntype=1;     %start
elseif nolinks>1, ntype=4;  %bifurcation
elseif nilinks>1, ntype=5;  %junction
elseif nolinks==0, ntype=3; %end
else ntype=2;               %normal
%--------------------------------
end;
S.ntype=ntype;
IsetLF=[S.IsetMWL S.IsetSAFL S.IsetDUL];
OsetLF=[S.OsetMWL S.OsetSAFL S.OsetDUL];
TsetLF=[S.TsetMWL S.TsetSAFL S.TsetDUL];
S.IsetLF=IsetLF;
S.OsetLF=OsetLF;
S.TsetLF=TsetLF;
```

```
    NODE_Table{i}=S;
end;

%Find link table
%LINK_Table components per row
%'name' ...
%data(1:12): Nm Lm Lambdam tau kappa nu vfm rhocr am rhomax delta phi
%-----------------------------
%If homogeneous for every link, otherwise correction is needed
%-----------------------------
LINK_Table={};
N_LFULL=N_MWL+N_SAFL+N_DUL;
% Snet.Pars=struct('tau',18,'kappa',40,'nu',60,...
%     'vfm',110,'rhocr',33.5,'am',1.636,...
%     'rhomax',180,...
%     'delta',0.0122,'phi',2.98,...
%     'Q0',1500);
%Q0 [veh/h/lane] is the capacity of origin links in case of free flow
tau=Snet.Pars.tau;
kappa=Snet.Pars.kappa;
nu=Snet.Pars.nu;
vfm=Snet.Pars.vfm;
rhocr=Snet.Pars.rhocr;
am=Snet.Pars.am;
rhomax=Snet.Pars.rhomax;
delta=Snet.Pars.delta;
phi=Snet.Pars.phi;
pars=[tau kappa nu vfm rhocr am rhomax delta phi];
pi=0;
for i=1:N_LFULL
    S=struct('n',[],'data',[]);
    %MWL
    if (N_MWL~=0) & (i<=N_MWL)
        k=i;
        n=MWL_Names{k};
        S(1).n=n;
        ldata=Snet.MotorwayLinks{k,2};
        Nm=round(ldata(1)/ldata(3));
        Lm=ldata(1)/Nm;
        Lambdam=ldata(2);
        S.data=[Nm Lm Lambdam pars];
        LINK_Table{i}=S;
    end;
    %SAFL
    if (N_SAFL~=0) & (i<=N_MWL+N_SAFL)
        k=i-N_MWL;
        n=SAFL_Names{k};
        S(1).n=n;
        ldata=Snet.StoreAndForwardLinks{k,2};
        Nm=round(ldata(1)/ldata(3));
        Lm=ldata(1)/Nm;
        Lambdam=ldata(2);
        S.data=[Nm Lm Lambdam pars];
        LINK_Table{i}=S;
    end;
    %DUL
    if (N_DUL~=0) & (i<=N_MWL+N_SAFL+N_DUL)
        k=i-(N_MWL+N_SAFL);
        n=DUL_Names{k};
        S(1).n=n;
        ldata=Snet.DummyLinks{k,2};
```

```
        Nm=round(ldata(1)/ldata(3));
        Lm=ldata(1)/Nm;
        Lambdam=ldata(2);
        S.data=[Nm Lm Lambdam pars];
        LINK_Table{i}=S;
    end;
end;

%Find Orig to node
ORLtoNODE_Table={};
for i=1:N_ORL

S=struct('orlname',[],'nodename',[],'node',[],'outlink',[],'Q0',[],'Q
ORLact',[]);
    orlname=ORL_Names{i};
    [nodename,node]=od2node(i,'orig');
    outlink=NODE_Table{node}.OsetMWL;
    if length(outlink)~=1
        fprintf('Number of outlinks for originlink=%s differs from
1\n',orlname);
        return;
    end;
    S(1).orlname=orlname;
    S.nodename=nodename;
    S.node=node;
    S.outlink=outlink;
    S.Q0=Snet.Pars.Q0;
    ORLtoNODE_Table{i}=S;
end;

%Find Destin from node
DELfromNODE_Table={};
for i=1:N_DEL
    S=struct('delname',[],'nodename',[],'node',[],'QDELact',[]);
    delname=DEL_Names{i};
    [nodename,node]=od2node(i,'dest');
    S(1).delname=delname;
    S.nodename=nodename;
    S.node=node;
    DELfromNODE_Table{i}=S;
end;

%Find queue table
%-----------------------------
%If homogeneous for every origin link, otherwise correction is needed
%No VMS is present, otherwise correction is needed
%-----------------------------
WQUEUE_Table={};
N_WQUEUE=N_ORL;

%Find controller table
%-----------------------------
%No VMS is present, otherwise correction is needed
%-----------------------------
CONTR_Table={};
N_CONTR=length(Snet.ControlledOrigins);
S=struct('n',[],'data',[]);
if N_CONTR~=0
    for i=1:N_CONTR
        n=Snet.ControlledOrigins{i};
        S(1).n=n;
```

```
        [ix,type]=sym2type(n,1);
        if type~=2
            fprintf('Error in controlled origins: ''%s'' is not
ORL\n',n);
        end;
        S.data=[ix NaN];
        CONTR_Table{i}=S;
    end;
end;

%Find VMS table
%-----------------------------
%No VMS is present, otherwise correction is needed
%-----------------------------
VMSWAYS_Table={};
N_VMSWAYS=4;
%D1->{{a,b,c,d,e},{a,b,f,g,e}}
%D2r={{a,b,c}}
%D3r={{a,b,f}}}
%-----------------------------
%May be computed from Snet
%-----------------------------
VMSWAYS_Table{1}={{1,2,3,4,5},{1,2,6,7,5}};
VMSWAYS_Table{2}={{1,2,3}};
VMSWAYS_Table{3}={{1,2,6}};

%Find state dimension and pointers
%-----------------------------
%No VMS is present, otherwise correction is needed
%-----------------------------
tab2xdp;
```

10.4 inithighpar()

```
function inithighpar
%Initiate higher parameters

global CONTR_MODE
global N_MWL N_ORL N_DEL N_SAFL N_DUL N_LFULL
global N_NODE N_CORIG N_VMSD N_WQUEUE N_CONTR N_VMSWAYS
global N_STATE N_STATEext
global MWL_Names ORL_Names DEL_Names SAFL_Names DUL_Names
global NODE_Names CORIG_Names VMSD_Names
global NODE_Table LINK_Table WQUEUE_Table CONTR_Table VMSWAYS_Table
global STATE_Table DEMAND_Table
global T_Sim T_Cont T_Full N_Cont N_Hor X_act X_extact TX_tran
global ORLtoNODE_Table DELfromNODE_Table

%Sampling
T_Sim=10;           %sec
T_Cont=6*T_Sim;     %sec
T_Full=1200*T_Sim;  %sec
N_Cont=T_Cont/T_Sim;
N_Hor=round(T_Full/T_Sim);
X_act=zeros(N_STATE,1); %[rho1; v1; ...]
X_extact=zeros(N_STATEext,4); %[rho1 v1 q1 Vrho1; rho2 v2 q2 Vrho2;
...]
TX_tran=zeros(N_Hor+1,1+N_STATE);

%----------------------------------------
%Demand description for origins in pars of (n in step,dn in veh/h)
%nmin=0 needed
%nmax should be the same for every demands
%For real investigation
DO1=[0 1000; 180 7000; 360 7000; 540 5000; 1400 5000];
DO2r=[0 600; 90 1800; 270 1800; 360 600; 1400 600];
DO3r=DO2r;
%For steady state computation and store
% DO1=[0 1000;  1400 1000];
% DO2r=[0 600;  1400 600];
% DO3r=DO2r; %DO3r(2,2)=1000;
%----------------------------------------

[yi,xi]=sec2fun(DO1); %Uses interp1
DEMAND_Table=NaN*ones(length(xi),N_ORL+1);
DEMAND_Table(:,1)=xi;
DEMAND_Table(:,2)=yi;
DEMAND_Table(:,3)=sec2fun(DO2r);
DEMAND_Table(:,4)=sec2fun(DO3r);
plot(DEMAND_Table(:,1), DEMAND_Table(:,2:N_ORL+1));
ylabel('Demand (veh/h)');
xlabel(['step (sec/' num2str(T_Sim) ')'])
legend('O1','O2r','O3r');
pause
```

10.5 initstate()

```
function initstate
%Initiate system state

global CONTR_MODE
global N_MWL N_ORL N_DEL N_SAFL N_DUL N_LFULL
global N_NODE N_CORIG N_VMSD N_WQUEUE N_CONTR N_VMSWAYS
global N_STATE N_STATEext
global MWL_Names ORL_Names DEL_Names SAFL_Names DUL_Names
global NODE_Names CORIG_Names VMSD_Names
global NODE_Table LINK_Table WQUEUE_Table CONTR_Table VMSWAYS_Table
global STATE_Table DEMAND_Table
global T_Sim T_Cont T_Full N_Cont N_Hor X_act X_extact TX_tran
global ORLtoNODE_Table DELfromNODE_Table

%-------------------------------------------
%Only for sintactical testing, should be changed for real cases
%-------------------------------------------
% X_act(:)=1;
X_act=[1:N_STATE]';
X_extact(:)=NaN;
TX_tran(1,:)=[0 X_act'];
rsw=N_STATE-N_WQUEUE+1;
rew=rsw+N_WQUEUE-1;
X_act(rsw:rew,1)=zeros(N_WQUEUE,1);

if exist('state0.mat')
    load state0 X_act
    return;
end;

rho0vec=30*ones(N_LFULL,1); v0vec=100*ones(N_LFULL,1);
w0vec=0*ones(N_WQUEUE,1);

for m=1:N_LFULL
    rsm=STATE_Table(m,2);
    dimm=STATE_Table(m,1)/2;
    rho0m=rho0vec(m);
    v0m=v0vec(m);
    X_act(rsm:rsm+dimm-1)=rho0m;
    X_act(rsm+dimm:rsm+2*dimm-1)=v0m;
end;

for i=1:N_WQUEUE
    rwi=STATE_Table(N_LFULL+i,2);
    w0i=w0vec(i);
    X_act(rwi)=w0i;
end;
```

10.6 funmw()

```
function [ftk,Qdest_LDEL]=funmw(tk)
%Compute the right side of state equation
%State=X_act
%tk is in tact, which is t/T_Sim

global CONTR_MODE
global N_MWL N_ORL N_DEL N_SAFL N_DUL N_LFULL
global N_NODE N_CORIG N_VMSD N_WQUEUE N_CONTR N_VMSWAYS
global N_STATE N_STATEext
global MWL_Names ORL_Names DEL_Names SAFL_Names DUL_Names
global NODE_Names CORIG_Names VMSD_Names
global NODE_Table LINK_Table WQUEUE_Table CONTR_Table VMSWAYS_Table
global STATE_Table DEMAND_Table
global T_Sim T_Cont T_Full N_Cont N_Hor X_act X_extact TX_tran
global ORLtoNODE_Table DELfromNODE_Table

global RVQL_tran WQL_tran
global RVEC_OLD

%Precomputation of regular (1....m) data
for i=1:N_LFULL
    %
    r=STATE_Table(i,:);
    rs=r(2); re=r(3);
    Nm=r(4)-2; rsext=r(5)+1; reext=r(6)-1;
    rsrho=rs; rerho=rs+Nm-1; rsv=rerho+1; rev=rsv+Nm-1;
    data=LINK_Table{i}.data;
    Lambdam=data(3);
    vfm=data(7);
    rhocrm=data(8);
    am=data(9);
    rhom=X_act(rsrho:rerho);
    vm=X_act(rsv:rev);
    qm=rhom.*vm*Lambdam;
    Vrhom=vfm*exp(-1/am*(rhom/rhocrm).^am);
    X_extact(rsext:reext,1:4)=[rhom vm qm Vrhom];
    %
end;

% fprintf('Phase1\n');
% keyboard

%Administrate origin demands and waiting queues
for i=1:N_ORL
    %i is instead of o (origin)
    mu=ORLtoNODE_Table{i}.outlink;
    rhomax=LINK_Table{mu}.data(10);
    rhomucr=LINK_Table{mu}.data(8);
    Q0=ORLtoNODE_Table{i}.Q0;
    %--------------
    %Q0 in database has dimension veh/h/lane
    %However here Q0 should be in veh/h!!!
    lanemu=LINK_Table{mu}.data(3);
    Q0=Q0*lanemu;
    %--------------
    rsmu1=STATE_Table(mu,2);
    rhomu1=X_act(rsmu1);
    rwi=STATE_Table(N_LFULL+i,2);
```

```
    wi=X_act(rwi);
    %kdem=round(tk/T_sim);
    kdem=tk;
    di=DEMAND_Table(kdem+1,i+1);    %Time dependent origin demand
    qhati1=di+wi/(T_Sim/3600);
    qhati2=Q0*min([1; (rhomax-rhomu1)/(rhomax-rhomucr)]);
    qhati=min([qhati1; qhati2]);
    switch CONTR_MODE
        case 1  %Simulation
            ri=1;
        case 2  %ALINEA
            ri=RVEC_OLD(i);
            if rem(tk,N_Cont)==0
                icont=0;
                Kri=[];
                if N_CONTR~=0
                    for kk=1:N_CONTR
                        kkdata=CONTR_Table{kk}.data;
                        if kkdata(1)==i
                            icont=1;
                            Kri=kkdata(2);
                            break;
                        end;
                    end;
                end;
                if icont==0
                    ri=1;
                else
                    riprev=RVEC_OLD(i);
                    ri=riprev+Kri*(rhomucr-rhomu1);
                    %
                    rimin=0.001; rimax=1;
                    %
                    if ri<rimin, ri=rimin;
                    elseif ri>rimax, ri=rimax;
                    else
                        ri=ri;
                    end;
                end;
            end;
        otherwise
            ftk=[];
            fprintf('CONTR_MODE=%g is not developed
yet\n',CONTR_MODE);
            return;
    end;
    qi=ri*qhati;
    winew=wi+(T_Sim/3600)*(di-qi);
    rwiext=STATE_Table(N_LFULL+i,5);
    X_extact(rwiext,1)=winew;
    X_extact(rwiext,2)=qi;
    X_extact(rwiext,3)=ri;
    X_extact(rwiext,4)=qhati;
    ORLtoNODE_Table{i}.QORLact=qi;
end;

% fprintf('Phase2\n');
% keyboard

%Find Qin for nodes
Qin_NODE=zeros(N_NODE,1);
```

```
for i=1:N_NODE
    NTi=NODE_Table{i};
    if ~isempty(NTi.IsetORL)
        orl=NTi.IsetORL;
        rwiext=STATE_Table(N_LFULL+orl,5);
        qin=X_extact(rwiext,2);
        %qin=ORLtoNODE_Table{orl}.QORLact;
    else
        qin=0;
    end;
    IsetLF=NTi.IsetLF;
    qin=iset2qin(IsetLF,qin);
    Qin_NODE(i)=qin;
end;

Qindecr_NODE=Qin_NODE;
Qdest_LDEL=zeros(N_DEL,1);

% fprintf('Phase3\n');
% keyboard

%Distribute Qin for output links and administrate forward/backward
effects
for i=1:N_NODE
    Si=NODE_Table{i};
    OsetLF=Si.OsetLF;
    TsetLF=Si.TsetLF;
    OsetDEL=Si.OsetDEL;
    TsetDEL=Si.TsetDEL;
    IsetLF=Si.IsetLF;
    %Forward effect
    Qin=Qin_NODE(i);

[qindecr,qdesout]=otset2qv0(Qin,OsetDEL,TsetDEL,OsetLF,TsetLF,IsetLF)
;

%     fprintf('Phase4a\n');
%     keyboard

    Qindecr_NODE(i)=qindecr;
    Qdest_LDEL(OsetDEL)=qdesout;
    %Backward effect
    ioset2rhomNmp1(IsetLF,OsetLF);

%     fprintf('Phase4b\n');
%     keyboard

%     if i==1
%         i
%         qL00=X_extact(1,3)
%         qL01=X_extact(2,3)
%         keyboard
%     end;

end;

% fprintf('Phase5\n');
% keyboard

%Compute new states
ftk=NaN*ones(N_STATE,1);
```

```
for m=1:N_LFULL
    data=LINK_Table{m}.data;
    Lm=data(2);
    Lambdam=data(3);
    tau=data(4);
    kappa=data(5);
    nu=data(6);
    rsext=STATE_Table(m,5);
    reext=STATE_Table(m,6);
    rhomvec=X_extact(rsext+1:reext-1,1);
    rhomp1vec=X_extact(rsext+2:reext,1);
    qmvec=X_extact(rsext+1:reext-1,3);
    qmm1vec=X_extact(rsext:reext-2,3);
    vmvec=X_extact(rsext+1:reext-1,2);
    vmm1vec=X_extact(rsext:reext-2,2);
    Vrhomvec=X_extact(rsext+1:reext-1,4);
    newrhomvec=rhomvec+(T_Sim/3600)/(Lm*Lambdam)*(qmm1vec-qmvec);
    newvmvec=vmvec+T_Sim/tau*(Vrhomvec-vmvec)...
        +(T_Sim/3600)/Lm*vmvec.*(vmm1vec-vmvec)...
        -nu*T_Sim/(tau*Lm)*(rhomp1vec-rhomvec)./(rhomvec+kappa);
    %--------------------------
    %Extension is possible based on delta,phi and further members
    %--------------------------
    rs=STATE_Table(m,2);
    re=STATE_Table(m,3);
    ftk(rs:re)=[newrhomvec; newvmvec];
end;
for i=1:N_WQUEUE
    rw=STATE_Table(N_LFULL+i,2);
    rwext=STATE_Table(N_LFULL+i,5);
    ftk(rw)=X_extact(rwext);
end;

% fprintf('Phase6\n');
% keyboard
```

10.7 sym2type()

```
function [ix,type]=sym2type(s1,mode)
%Find symbol in table and convert to code
%type:  1=MWL, 2=ORL, 3=DEL, 4=SAFL, 5=DUL, 6=NODE, 7=CORIG, 8=VMSD
%mode:  1=MWL,ORL,DEL,SAFL,DUL,NODE; 2=CORI; 3=VMSD

global CONTR_MODE
global N_MWL N_ORL N_DEL N_SAFL N_DUL N_LFULL
global N_NODE N_CORIG N_VMSD N_WQUEUE N_CONTR N_VMSWAYS
global N_STATE N_STATEext
global MWL_Names ORL_Names DEL_Names SAFL_Names DUL_Names
global NODE_Names CORIG_Names VMSD_Names
global NODE_Table LINK_Table WQUEUE_Table CONTR_Table VMSWAYS_Table
global STATE_Table DEMAND_Table
global T_Sim T_Cont T_Full N_Cont N_Hor X_act X_extact TX_tran
global ORLtoNODE_Table DELfromNODE_Table

ix=[]; type=[];

if nargin<2, mode=1; end;

if mode==1
    if N_MWL~=0
        for i=1:N_MWL
            s2=MWL_Names(i);
            if strcmp(s1,s2)
                ix=i;
                type=1;
                return;
            end;
        end;
    end;
    if N_ORL~=0
        for i=1:N_ORL
            s2=ORL_Names(i);
            if strcmp(s1,s2)
                ix=i;
                type=2;
                return;
            end;
        end;
    end;
    if N_DEL~=0
        for i=1:N_DEL
            s2=DEL_Names(i);
            if strcmp(s1,s2)
                ix=i;
                type=3;
                return;
            end;
        end;
    end;
    if N_SAFL~=0
        for i=1:N_SAFL
            s2=SAFL_Names(i);
            if strcmp(s1,s2)
                ix=i;
                type=4;
                return;
```

```
            end;
        end;
    end;
    if N_DUL~=0
        for i=1:N_DUL
            s2=MWL_Names(i);
            if strcmp(s1,s2)
                ix=i;
                type=5;
                return;
            end;
        end;
    end;
    if N_NODE~=0
        for i=1:N_NODE
            s2=NODE_Names(i);
            if strcmp(s1,s2)
                ix=i;
                type=6;
                return;
            end;
        end;
    end;
    return
end; %End mode==1

if mode==2
    if N_CORIG~=0
        for i=1:N_CORIG
            s2=CORIG_Names(i);
            if strcmp(s1,s2)
                ix=i;
                type=7;
                return;
            end;
        end;
    end;
end;

if mode==3
    if N_VMSD~=0
        for i=1:N_VMSD
            s2=VMSD_Names(i);
            if strcmp(s1,s2)
                ix=i;
                type=8;
                return;
            end;
        end;
    end;
end;
```

10.8 tab2xdp()

```
function tab2xdp
%Find state dimensions and pointers from tables
global CONTR_MODE
global N_MWL N_ORL N_DEL N_SAFL N_DUL N_LFULL
global N_NODE N_CORIG N_VMSD N_WQUEUE N_CONTR N_VMSWAYS
global N_STATE N_STATEext
global MWL_Names ORL_Names DEL_Names SAFL_Names DUL_Names
global NODE_Names CORIG_Names VMSD_Names
global NODE_Table LINK_Table WQUEUE_Table CONTR_Table VMSWAYS_Table
global STATE_Table DEMAND_Table
global T_Sim T_Cont T_Full N_Cont N_Hor X_act X_extact TX_tran
global ORLtoNODE_Table DELfromNODE_Table

switch CONTR_MODE
    case {1,2}  %simul,ALINEA
        Nmvec=zeros(N_LFULL,1);
        Nmextvec=zeros(N_LFULL,1);
        Pxs=zeros(N_LFULL,1);
        Pxv=zeros(N_LFULL,1);
        %Links
        for i=1:N_LFULL
            Nmi=LINK_Table{i}.data(1)*2; %rho,v:1,...,Nm
            Nmiext=LINK_Table{i}.data(1)+2; %rho,v,q: 0,1,...,Nm,Nm+1
            Nmvec(i)=Nmi;
            Nmextvec(i)=Nmiext;
        end;
        cs=cumsum(Nmvec);
        csext=cumsum(Nmextvec);
        DIM_X=Nmvec;
        DIM_Xext=Nmextvec;
        PNT_Xs=[1; 1+cs(1:N_LFULL-1)];
        PNT_Xe=cs;
        PNT_Xexts=[1; 1+csext(1:N_LFULL-1)];
        PNT_Xexte=csext;
        %Waiting queues
        %No extension for w
        wvec=ones(N_WQUEUE,1);
        csw=cumsum(wvec);
        pnt_ws=[1; 1+csw(1:N_WQUEUE-1)];
        pnt_we=csw;
        pxlast=PNT_Xe(N_LFULL);
        pxextlast=PNT_Xexte(N_LFULL);
        DIM_X=[DIM_X; wvec];
        DIM_Xext=[DIM_Xext; wvec];
        PNT_Xs=[PNT_Xs; pxlast+pnt_ws];
        PNT_Xe=[PNT_Xe; pxlast+pnt_we];
        PNT_Xexts=[PNT_Xexts; pxextlast+pnt_ws];
        PNT_Xexte=[PNT_Xexte; pxextlast+pnt_we];
        %
        N_STATE=sum(DIM_X);
        N_STATEext=sum(DIM_Xext);
        STATE_Table=[DIM_X PNT_Xs PNT_Xe DIM_Xext PNT_Xexts
PNT_Xexte];
    otherwise
        fprintf('CONTR_MODE=%g actually not
elaborated\n',CONTR_MODE);
end
```

10.9 od2node()

```
function [nname,n]=od2node(ix,goal)
%Find node belonging to ORL or DEL link
%Input: i is index in ORL_Names or DEL_Names used in NODE_Table

global CONTR_MODE
global N_MWL N_ORL N_DEL N_SAFL N_DUL N_LFULL
global N_NODE N_CORIG N_VMSD N_WQUEUE N_CONTR N_VMSWAYS
global N_STATE N_STATEext
global MWL_Names ORL_Names DEL_Names SAFL_Names DUL_Names
global NODE_Names CORIG_Names VMSD_Names
global NODE_Table LINK_Table WQUEUE_Table CONTR_Table VMSWAYS_Table
global STATE_Table DEMAND_Table
global T_Sim T_Cont T_Full N_Cont N_Hor X_act X_extact TX_tran
global ORLtoNODE_Table DELfromNODE_Table

nname=[]; n=[];

for i=1:N_NODE
    NTi=NODE_Table{i};
    if strcmp(goal,'orig')
        set=NTi.IsetORL;
    elseif strcmp(goal,'dest')
        set=NTi.OsetDEL;
    else
        fprintf('Error in function od2node, goal=%s\n',goal);
        return;
    end;
    if (length(set)==1) & (set(1)==ix)
        n=i;
        nname=NODE_Names{i};
        return;
    end;
end;

fprintf('Error in NODE_Table during od2node calling\n');
return;
```

10.10 sec2fun()

```
function [yi,xi]=sec2fun(nfpars)
%Create time function from sections

x=nfpars(:,1);
y=nfpars(:,2);
x0=x(1);
xmax=x(length(x));
xi=[x0:1:xmax]';
yi=interp1(x,y,xi);
```

10.11 iset2qin()

```
function qout=iset2qin(Iset,qin)
%Input flow computation
%Before call: X_extact should contain q(m,1),....,q(m,Nm) for MWL,
SAFL and DUL links

global CONTR_MODE
global N_MWL N_ORL N_DEL N_SAFL N_DUL N_LFULL
global N_NODE N_CORIG N_VMSD N_WQUEUE N_CONTR N_VMSWAYS
global N_STATE N_STATEext
global MWL_Names ORL_Names DEL_Names SAFL_Names DUL_Names
global NODE_Names CORIG_Names VMSD_Names
global NODE_Table LINK_Table WQUEUE_Table CONTR_Table VMSWAYS_Table
global STATE_Table DEMAND_Table
global T_Sim T_Cont T_Full N_Cont N_Hor X_act X_extact TX_tran
global ORLtoNODE_Table DELfromNODE_Table


if isempty(Iset)
    qout=qin;
    return;
end;

L=length(Iset);
q=qin;
for i=1:L
   mu=Iset(i);
   reextmu=STATE_Table(mu,6);
   qNmu=X_extact(reextmu-1,3);
   q=q+qNmu;
end;
qout=q;
```

10.12 oset2qv0()

```
function
[qindecr,qdesout]=otset2qv0(qin,OsetDEL,TsetDEL,OsetLF,TsetLF,IsetLF)
%Distribute qin for node i to destination link and compute qm0, vm0

global CONTR_MODE
global N_MWL N_ORL N_DEL N_SAFL N_DUL N_LFULL
global N_NODE N_CORIG N_VMSD N_WQUEUE N_CONTR N_VMSWAYS
global N_STATE N_STATEext
global MWL_Names ORL_Names DEL_Names SAFL_Names DUL_Names
global NODE_Names CORIG_Names VMSD_Names
global NODE_Table LINK_Table WQUEUE_Table CONTR_Table VMSWAYS_Table
global STATE_Table DEMAND_Table
global T_Sim T_Cont T_Full N_Cont N_Hor X_act X_extact TX_tran
global ORLtoNODE_Table DELfromNODE_Table

%Destination link
if isempty(TsetDEL)
    qdesout=0;
else
    qdesout=TsetDEL*qin;
end;
qindecr=qin-qdesout;

%Remaining output links
Lout=length(OsetLF);
if Lout==0, return; end;
for i=1:Lout
    m=OsetLF(i);
    betanm=TsetLF(i);
    qm0=betanm*qin;
    Nm=STATE_Table(m,4)-2;
    rsmext=STATE_Table(m,5);
    Lin=length(IsetLF);
    if Lin==0
        vm1=X_extact(rsmext+1,2);
        %------------------
        vm0=vm1; %?????????
        %------------------
    else
        vmuvec=zeros(Lin,1); qmuvec=zeros(Lin,1);
        for k=1:Lin
            mu=IsetLF(k);
            Nmu=STATE_Table(mu,4)-2;
            rsmuext=STATE_Table(mu,5);
            vmuNmu=X_extact(rsmuext+Nmu,2);
            qmuNmu=X_extact(rsmuext+Nmu,3);
            vmuvec(k)=vmuNmu;
            qmuvec(k)=qmuNmu;
        end;
        vm0=sum(vmuvec.*qmuvec)/sum(qmuvec);
    end;
    X_extact(rsmext,3)=qm0;
    X_extact(rsmext,2)=vm0;
end;
```

10.13 ioset2rhomNmp1()

```
function ioset2rhomNmp1(IsetLF,OsetLF);
%Administrate backward effect of output links to input links

global CONTR_MODE
global N_MWL N_ORL N_DEL N_SAFL N_DUL N_LFULL
global N_NODE N_CORIG N_VMSD N_WQUEUE N_CONTR N_VMSWAYS
global N_STATE N_STATEext
global MWL_Names ORL_Names DEL_Names SAFL_Names DUL_Names
global NODE_Names CORIG_Names VMSD_Names
global NODE_Table LINK_Table WQUEUE_Table CONTR_Table VMSWAYS_Table
global STATE_Table DEMAND_Table
global T_Sim T_Cont T_Full N_Cont N_Hor X_act X_extact TX_tran
global ORLtoNODE_Table DELfromNODE_Table

Lin=length(IsetLF);
Lout=length(OsetLF);
if Lin==0, return; end;
if Lout==0
    for k=1:Lin
        mu=IsetLF(k);
        remuext=STATE_Table(mu,6);
        X_extact(remuext,1)=X_extact(remuext-1,1);   %Node has no
link outputs
    end;
else
    rhom1vec=zeros(Lout,1);
    for k=1:Lout
        m=OsetLF(k);
        rsmext=STATE_Table(m,5);
        rhom1vec(k)=X_extact(rsmext+1); %rhom1
    end;
    sumrho1norm=sum(rhom1vec.^2)/sum(rhom1vec);
    for k=1:Lin
        mu=IsetLF(k);
        remuext=STATE_Table(mu,6);
        X_extact(remuext,1)=sumrho1norm;
    end;
end;
```

10.14 costfun()

```
function [TTS,TTT,TWT,QDC]=costfun(RVQ,WQR,Ts,Nc,af)
%Compute cost function of motorway control
%Ts in sec

global CONTR_MODE
global N_MWL N_ORL N_DEL N_SAFL N_DUL N_LFULL
global N_NODE N_CORIG N_VMSD N_WQUEUE N_CONTR N_VMSWAYS
global N_STATE N_STATEext
global MWL_Names ORL_Names DEL_Names SAFL_Names DUL_Names
global NODE_Names CORIG_Names VMSD_Names
global NODE_Table LINK_Table WQUEUE_Table CONTR_Table VMSWAYS_Table
global STATE_Table DEMAND_Table
global T_Sim T_Cont T_Full N_Cont N_Hor X_act X_extact TX_tran
global ORLtoNODE_Table DELfromNODE_Table

global RVQL_tran WQL_tran
global RVEC_OLD

[m,n]=size(RVQ);
[m1,n1]=size(WQR);
m=min([m m1]');
n=min([n n1]');
K=m-1;
ts=[0:K-1]';
ic=find(~rem(ts,Nc));

Nmvec=NaN*ones(N_LFULL,1);
Lmvec=NaN*ones(N_LFULL,1);
lambdamvec=NaN*ones(N_LFULL,1);
for m=1:N_LFULL
    Nm=LINK_Table{m}.data(1);
    Lm=LINK_Table{m}.data(2);
    lambdam=LINK_Table{m}.data(3);
    Nmvec(m)=Nm;
    Lmvec(m)=Lm;
    lambdamvec(m)=lambdam;
end;

sum1=0;
col=1;
for m=1:N_LFULL
    Nm=Nmvec(m);
    Lm=Lmvec(m);
    lambdam=lambdamvec(m);
    rs=col+2;
    re=col+Nmvec(m);
    rmatm=RVQ(1:K,rs:re);
    rvecm=rmatm(:);
    summ=sum(rvecm)*Lm*lambdam;
    sum1=sum1+summ;
    col=col+3*(Nm+2);
end;

sum2=0;
col=1;
for orig=1:N_ORL
    rws=col+1;
    wveco=WQR(1:K,rws);
```

```
    sumo=sum(wveco);
    sum2=sum2+sumo;
    col=col+3;
end;

sum3=0;
col=1;
for orig=1:N_ORL
    rcs=col+3;
    cveco=WQR(1:K,rcs);
    dcvec=cveco(2:K)-cveco(1:K-1);
    dcvec2=norm(dcvec,2)^2;
    sumc2=dcvec2;
    sum3=sum3+sumc2;
    col=col+3;
end;

TTT=(Ts/3600)*sum1;
TWT=(Ts/3600)*sum2;
QDC=(Ts/3600)*Nc*af*sum3;
TTS=TTT+TWT;
```

10.15 plotrvq()

```
function plotrvq(RVQ_tran)
%Plot rho,v,q transients of sections

global CONTR_MODE
global N_MWL N_ORL N_DEL N_SAFL N_DUL N_LFULL
global N_NODE N_CORIG N_VMSD N_WQUEUE N_CONTR N_VMSWAYS
global N_STATE N_STATEext
global MWL_Names ORL_Names DEL_Names SAFL_Names DUL_Names
global NODE_Names CORIG_Names VMSD_Names
global NODE_Table LINK_Table WQUEUE_Table CONTR_Table VMSWAYS_Table
global STATE_Table DEMAND_Table
global T_Sim T_Cont T_Full N_Cont N_Hor X_act X_extact TX_tran
global ORLtoNODE_Table DELfromNODE_Table

col=1;
nt=size(RVQ_tran,1)-2;
tt=RVQ_tran(1:nt,1);
for m=1:N_LFULL
    dm=STATE_Table(m,4);
    rhomat=RVQ_tran(1:nt,col+2:col+dm-1);
    vmat=RVQ_tran(1:nt,col+dm+2:col+2*dm-1);
    qmat=RVQ_tran(1:nt,col+2*dm+2:col+3*dm-1);
    sm=[' of Link ' LINK_Table{m}.n];
    figure
    subplot(311);
    plot(tt,rhomat);
    title(['rho' sm ' (veh/km/lane)']);
    subplot(312);
    plot(tt,vmat);
    title(['v' sm ' (km/h)']);
    subplot(313);
    plot(tt,qmat);
    title(['q' sm ' (veh/h)']);
    xlabel(['step (sec/' num2str(T_Sim) ')'])
    col=col+3*dm;
end;
```

10.16 plotwq()

```
function plotwq(WQ_tran)
%Plot w and q for origins

global CONTR_MODE
global N_MWL N_ORL N_DEL N_SAFL N_DUL N_LFULL
global N_NODE N_CORIG N_VMSD N_WQUEUE N_CONTR N_VMSWAYS
global N_STATE N_STATEext
global MWL_Names ORL_Names DEL_Names SAFL_Names DUL_Names
global NODE_Names CORIG_Names VMSD_Names
global NODE_Table LINK_Table WQUEUE_Table CONTR_Table VMSWAYS_Table
global STATE_Table DEMAND_Table
global T_Sim T_Cont T_Full N_Cont N_Hor X_act X_extact TX_tran
global ORLtoNODE_Table DELfromNODE_Table

col=1;
nt=size(WQ_tran,1)-2;
tt=WQ_tran(1:nt,1);
for io=1:N_ORL
    do=3;
    wvec=WQ_tran(1:nt,col+1);
    qvec=WQ_tran(1:nt,col+2);
    rvec=WQ_tran(1:nt,col+3);
    sm=ORL_Names{io};
    figure
    subplot(311);
    plot(tt,wvec);
    title(['w of origin ' sm ' (veh)']);
    subplot(312);
    plot(tt,qvec);
    title(['q of origin ' sm ' (veh/h)']);
    subplot(313);
    plot(tt,rvec);
    title(['r of origin ' sm ]);
    xlabel(['step (sec/' num2str(T_Sim) ')'])
    col=col+3;
end;
```

10.17 plotqdest()

```
function  plotqdest(QD_tran,Ndel,Ts)
%Plot w and q for origins

nt=size(QD_tran,1)-2;
tt=QD_tran(1:nt,1);
qdmat=QD_tran(1:nt,2:Ndel+1);
qdsum=sum(qdmat')';

figure
subplot(211);
plot(tt,qdmat);
title('q of destinations (veh/h)');
subplot(212);
plot(tt,qdsum);
title('q total outflow (veh/h)');
xlabel(['step (sec/' num2str(Ts) ')'])
```