papír pro razítko ústavu K611

České vysoké učení technické v Praze
Fakulta dopravní

DIPLOMOVÁ PRÁCE

Miroslav Vaniš

Ověření mikroskopických modelů dopravy na reálných dopravních datech

Ústav aplikované matematiky

Vedoucí bakalářské práce:	Dr. Ing. Jan Přikryl
Studijní program:	Technika a technologie v dopravě a spojích
Studijní obor:	Inženýrská informatika v dopravě a spojích

Praha 2015

Papír se zadáním diplomové práce

Poděkování

Rád bych poděkoval všem, kteří mně pomohli při psaní této diplomové práce.

Na prvním místě jsou to mí rodiče, kteří mě neustále podporovali během mého dlouhého studia.

Dále bych chtěl poděkovat mému vedoucímu Janu Přikrylovi, který se mnou moji diplomovou práci neustále konzultoval, kamarádovi Mikuláši Dítěti, který mi pomohl v prvních krocích se softwarem SUMO, Tomáši Třasákovi za diskuse nad tématy v diplomové práci, Krzystzofovi Urbańcovi za pomoc při hledání chyb a diskusích nejen o diplomové práci, Ivanovi Nagyovi při konzultacích o statistických metodách a celému Ústavu aplikované matematiky za prostředí k psaní diplomové práce.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Nemám závažný důvod proti užívání tohoto školního díla ve smyslu § 60 Zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon).

V Praze dne ..... Podpis autora .....

Abstract

In this diploma thesis we have studied selected mathematical models developed to mimic the behavior of the real traffic flow. In the theoretical part we introduced some basic ideas related to the models of the traffic flow. We introduce basic microscopic models which are used for traffic simulations. Next we familiarize with validation and calibration of these models, which are assumption for properly working model. In the practical part of the thesis we perform simulations in SUMO software. This software includes some of the models, the other ones will be implemented. After that, we started the simulations for each model. The results from these simulations we analyse with statistical methods and graphically.

Abstrakt

V této diplomové práci studujeme vybrané mikroskopické modely, které se snaží napodobit chování dopravního toku. V teoretické části představíme základní principy spojené s těmito modely. Představíme skupinu hlavních mikroskopických modelů, které se v dnešní době používají k simulaci dopravy. Dále se seznámíme s validací a kalibrací těchto modelů, které jsou předpokladem pro správně fungující model. V praktické části budeme provádět simulace v softwaru SUMO. Ten již některé z modelů obsahuje, ostatní do něj budou implementovány. Poté spustíme samotné simulace pro všechny modely. Výsledky z těchto modelů na základě statistických metod či graficky vyhodnotíme.

Obsah

Úvod

V dnešní době je na dálnicích velká snaha o dynamické řízení dopravy obzvláště v místech s uzavírkami jízdních pruhů či při vysoké hustotě provozu. Uzavření jízdního pruhu může nastat prakticky kdykoliv a to plánovaně (např. při pravidelné údržbě) a nebo neplánovaně (dopravní nehoda). Z toho vyplývá úkol pokusit se v tento moment řídit dopravu jak nejlépe to jde. Ani experti se ovšem neshodují, co znamená slovo nejlépe z předchozí věty. Může to být např. konkrétní hodnota intenzity či rychlosti, dojezdový čas či další parametry dopravního proudu.

Pokud jsme schopni získat surová data z daného úseku, např. intenzitu vztaženou na minutu pro každý pruh, a následně je vyhodnotíme, dostaneme přesné informace o úseku. Poté můžeme také tyto data vzít a pokusit se nasimulovat danou oblast. Otázkou zůstává jak. Ideálně tak, aby simulace byla blízká k již získaným informacím o úseku. V této práci se zaměříme na mikroskopické modely, podle kterých budeme simulace provádět. To znamená, že budeme simulovat každé jednotlivé vozidlo.

Po simulacích vyhodnotíme jednotlivé modely a zjistíme, které modely odpovídají reálným situacím. Předpoklad je takový, že některé modely budou přesnější např. při vysokých hustotách, některé zase naopak.

Ve chvíli, kdy získáme tento přehled modelů, můžeme již přesněji analyzovat danou situaci a do budoucna budeme vědět, že daný model simuluje tento stav nejlépe.

Hlavní cíle práce tedy spočívají v simulaci dálničního provozu na základě získaných reálných dat, a to pomocí různých mikroskopických modelů. Dalším krokem je porovnat výsledky z těchto simulací s reálnými daty. Tento způsob lze využít jako další možnost při řízení dané dálniční oblasti či při predikování např. plánovaných uzavírek.

Cíle práce

Cíle práce lze chronologicky formulovat následovně:

- seznámení se se základními typy mikroskopických modelů (hlavně jejich matematickým popisem) v dopravě pro použití na dálnicích,
- seznámit se s kalibrací těchto modelů,
- seznámit se simulačním softwarem SUMO a možnostmi jeho využití v dané problematice.

Tato témata jsou náplní první, teoretické části práce. V druhé, praktické, části jsou cíle práce následující:

- implementace všech mikroskopických modelů z teoretické části do simulačního softwaru SUMO,
- kalibrace mikroskopických modelů z teoretické části,

- nasimulovaní reálných dat, případně vytvoření vlastních scénářů pro mikroskopické modely,
- Makroskopický popis výsledků simulace.

1. Mikroskopické modely

Hlavní skupinou mikroskopických modelů jsou modely sledu vozidel (z angl. *car-following models*). Tyto modely vycházejí z předpokladu, že pokud n-té vozidlo následuje $n-1$ vozidlo na homogenní komunikaci, trajektorie n-tého vozidla bude stejná, jako je trajektorie vozidla před ním (tedy vozidla $n-1$), až na posuny v čase a místě (vozidlo bude ve stejném místě jako předcházející v jiný čas a ve stejný čas bude na jiném místě). Všechny další dispozice jednotlivých modelů v sobě tento předpoklad obsahují.

Na konci každého modelu budou pro přehlednost shrnuty parametry modelu a konečná rovnice, podle které se vypočítá budoucí hodnota rychlosti.

1.1 Newellův model

V teorii dopravního proudu je Newellův model jeden ze základních modelů popisujících chování vozidla na základě vozidla před ním. Řidič se snaží držet za vozidlem jedoucím před ním v konstantní vzdálenosti. Tento model vznikl již v roce 1961. Zde jej zmiňujeme hlavně z důvodu, že z Newellových úvah vychází mnohé novější modely, například *Intelligent Driver Model* (IDM), který si popíšeme později.

V článku [19] je přesně popsán nejen model, ale i jeho verifikace a porovnání s ostatními modely. Pro nás je důležitý popis modelu.

1.1.1 Popis modelu

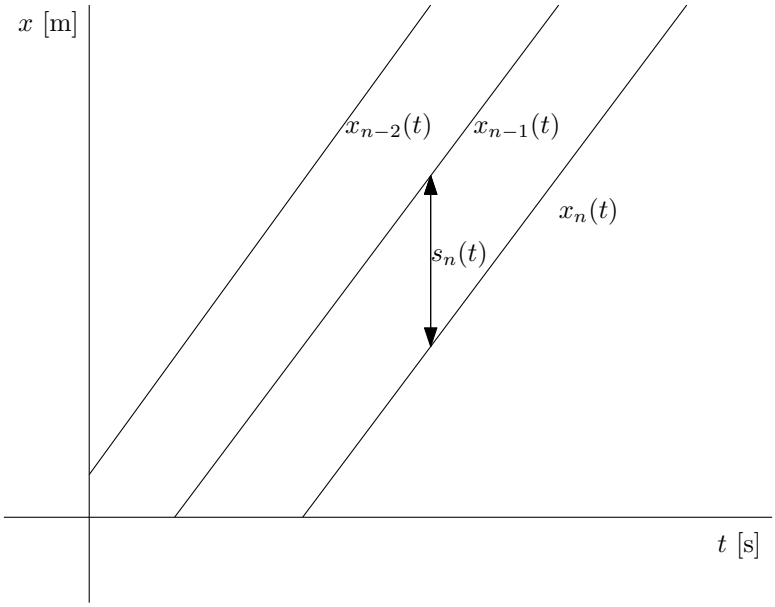
Jestliže n-té vozidlo jede za $n-1$ vozidlem (které jede za $n-2$ vozidlem atd.), cíl každého modelu sledu vozidel je zjistit závislost pozice n-tého vozidla $x_n(t)$ na $n-1$ vozidle, viz obrázek 1.1. Jestliže se $n-1$ vozidlo pohybuje konstantní rychlostí v,

$$x_{n-1}(t) = x_n(t) + vt,$$

n-té vozidlo také pojede průměrnou rychlostí v. Pokud by n-té vozidlo zrychlovalo, tak by došlo ke kolizi s $n-1$ vozidlem a naopak pokud by zpomalovalo, tak by se n-té vozidlo neustále vzdalovalo. Totéž platí pro všechna vozidla jedoucí za n-tým. Newellův model se nezabývá zjištěním hodnoty rychlosti v, předpokládá se, že je určena buďto na základě omezení rychlosti či podle konstrukčních možností vozidla.

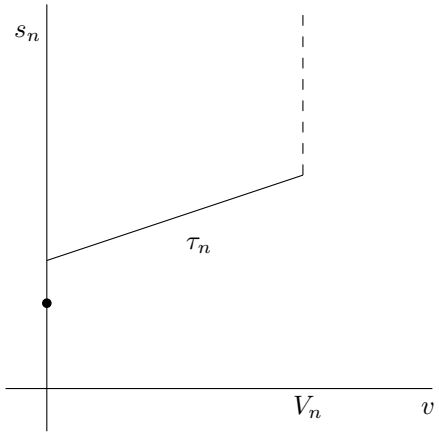
Vzdálenost $s_n = x_{n-1}(t) - x_n(t)$ mezi vozidly n a $n-1$ se může měnit v čase. Pokud je komunikace homogenní, tato vzdálenost zůstane téměř konstantní s mírnými oscilacemi v okolí s_n. Tato hodnota se mění na základě typu vozidla a závisí také na rychlosti v.

Předpokládejme, že existuje nějaký empirický vztah mezi rychlostí v a vzdáleností mezi vozidly s_n. Pokud rychlost v roste, je logické, že řidiči chtějí dosáhnout většího rozestupu mezi vozidly. Tato závislost mezi v a s_n je znázorněna na obrázku 1.2. Každý řidič má svoji preferovanou rychlost V_n. Jestliže rychlost v je u $n-1$ vozidla je vyšší než preferovaná rychlost u n-tého, tedy $v > V_n$, znamená to, že n-té vozidlo pojede svoji



Obrázek 1.1: Trajektorie vozidel s konstantní rychlostí, zdroj: [19]

preferovanou rychlostí (na obrázku 1.2 znázorněna čárkovanou čárou) a $n-1$ vozidlo mu ujede. Hodnota rychlosti v nemůže být záporná a vzdálenost mezi vozidly při nulové rychlosti by měla být níže než polopřímka lineární závislosti, viz černá tečka na obrázku 1.2 při rychlosti $v = 0$.



Obrázek 1.2: Vztah mezi rychlostí v a vzdáleností mezi vozidly s_n, zdroj: [19]

Nyní předpokládejme, že se $n-1$ vozidlo nějakou dobu t pohybuje konstantní rychlostí v[1] a potom náhle změní rychlost na hodnotu v'. Trajektorie vozidel n a $n-1$ mohou

[1] hodnota rychlosti osciluje okolo hodnoty v

poté vypadat jako na obrázku 1.3. Z obrázku lze také vypozorovat jak časovou τ_n, tak prostorovou d_n mezeru mezi vozidly n a $n-1$. Z čárkovaného obdelníku poté dostáváme vztah vzdálenost mezi vozidly před změnou rychlosti s_n a po změně rychlosti s'_n,

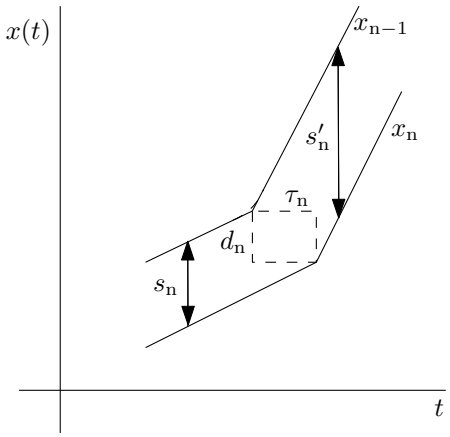
$$s_n = d_n + v\tau_n, \qquad s'_n = d_n + v'\tau_n.$$

Z toho vyplývá, že pokud leží v a v' na polopřímce z obrázku 1.2, sklon této přímky je právě τ_n a hodnota s_n při rychlosti $v = 0$ je d_n.

Ze vztahu mezi v a s_n, jak je zobrazeno na obrázku 1.2, plyne nezávislost mezer d_n a τ_n na rychlostech v, resp. v'. Pokud se tedy změní rychlost z hodnoty v' na v'', d_n a τ_n zůstanou při této změně rychlosti stejné. Lineární trajektorie vozidla $x_n(t)$ potom bude jednoduše posun v čase τ_n a místě d_n,

$$x_n(t + \tau_n) = x_{n-1}(t) - d_n. \tag{1.1}$$

U Newellova modelu platí, že n-té vozidlo bude (přibližně) kopírovat trajektorii n-1 vozidla dle vztahu (1.1) při vhodných hodnotách d_n a τ_n. Tím, jak se přesně dokáže n-té vozidlo dodržovat vztah (1.1), se Newellův model nezabývá, pouze předpokládá, že řidič je schopen se tímto vztahem řídit. To není tak těžké, protože pokud se změní rychlost vozidla $n-1$, n-tý řidič nemusí zareagovat okamžitě, ale může počkat dokud se vzdálenost s_n nezvýší (neklesne) na hodnotu, která odpovídá nové rychlosti vozidla $n-1$ (viz obrázek 1.2).



Obrázek 1.3: Lineární aproximace při změně rychlosti vozidel, zdroj: [19]

Při pozorování se došlo k závěru, že každý řidič nezkoumá všechna vozidla na komunikaci, ale pouze vozidla na vhodné „makroskopické“ části komunikace.

Hodnoty τ_n a d_n jsou přirozené, to vychází ze vztahu (1.1), kde postupným iterováním dostaneme

$$x_n(t + \tau_n + \tau_{n-1} + \cdots + \tau_1) = x_0(t) - d_n - d_{n-1} - \cdots - d_1. \tag{1.2}$$

Hodnoty τ_n a d_n se značně liší mezi jednotlivými vozidly v závislosti na typu řidiče. První skupina řidičů raději jede blíže vozidlu před sebou $(n-1)$ až na minimální bezpečnou vzdálenost, zatímco druhá skupina naopak preferuje delší vzdálenost pro klidnou reakci na nenadálou událost. Je rozumné, aby se hodnoty τ_n a d_n lišily, a to tak, že každému jednotlivému vozidlu budou hodnoty vygenerovány z nějakého pravděpodobnostního rozdělení, jehož variační koeficient se bude blížit jedné. Při generování rychlostí vozidel by naopak měl být variační koeficient zanedbatelný.

Položíme-li

$$\overline{\tau} = \frac{1}{n}\sum_{k=1}^{n} \tau_k, \qquad \overline{d} = \frac{1}{n}\sum_{k=1}^{n} d_k, \tag{1.3}$$

kde $\overline{\tau}$ a $\overline{d}$ jsou průměrné posuny v čase resp. místě, potom průměrnou vlnovou rychlost spočteme jako podíl $\overline{d}/\overline{\tau}$.

Tento model se dá ovšem popsat nejen mikroskopicky (rychlost, mezera mezi vozidly), ale i makroskopicky (hustota k, intenzita q). Stacionární stav nastane ve chvíli, kdy se všechna vozidla budou pohybovat stejnou konstantní rychlostí s rozdílnými posuny (časovými i prostorovými).

Jestliže

$$s_n = d_n + v\tau_n,$$

a rychlost vozidel je konstantní, tak platí

$$\overline{s} = \overline{d} + v\overline{\tau}.$$

Hustotu k lze potom vypočítat jako převrácenou hodnotu průměrné mezery mezi vozidly $k = 1/\overline{s}$ a rychlost jako podíl intenzity a hustoty, to jest $v = q/k$. Tudíž platí

$$q = \frac{1}{\overline{\tau}} - \frac{\overline{d}}{\overline{\tau}}k, \tag{1.4}$$

za předpokladu, že rychlost v je menší než preferovaná rychlost jakéhokoliv vozidla V_k.

Rovnice (1.4) propojuje Newellův model s klasickými makroskopickými modely. Problém nastává ve chvíli, kdy průměrná rychlost v je vyšší, než některá s preferovaných rychlostí jednotlivých vozidel. Vozidla se v tomto modelu nemohou předjíždět a proto vznikne kongesce za vozidlem s malou preferovanou rychlostí. V následujících řádcích předpokládáme, že aktuální rychlost vozidla bude menší než preferovaná rychlost V_n.

Uvažujme, že se vozidlo n pohybuje přesně podle rovnice (1.1) a vozidlo za ním $(n-1)$ jede plynule za ním.

Rovnici (1.1) můžeme přepsat do tvaru

$$x_n(t+\tau_n) = x_n(t) + \tau_n v_n(t+T), \tag{1.5}$$

kde T značí reakční čas řidiče n. Tvar rovnice (1.5) plyne z matematické analýzy[2], přičemž hodnota T se nachází někde mezi nulou a τ_n. Pokud je funkce hladká bude

[2] věta o střední hodnotě

přibližně

$$T = \frac{\tau_n}{2}. \tag{1.6}$$

Rovnici (1.5) lze přepsat na přibližný tvar

$$x_n(t + \tau_n) = x_n(t) + \tau_n v_n(t) + \tau_n T a_n(t), \tag{1.7}$$

který uvažuje zrychlení vozidla a_n[3]. Kombinací rovnic (1.7) a (1.1) dostáváme vztah pro rychlost

$$v_n(t + \tau_n) = \frac{1}{\tau_n} \left[x_{n-1}(t) - x_n(t)\right] - \frac{d_n}{\tau_n}. \tag{1.8}$$

Po zderivování rovnice (1.8) dostáváme vztah pro zrychlení vozidla

$$a_n(t + \tau_n) = \frac{1}{\tau_n} \left[v_{n-1}(t) - v_n(t)\right]. \tag{1.9}$$

Z rovnic (1.8) a (1.9) je vidět zřejmá závislost na vozidle, které jede před aktuálním, členem $v_{n-1}(t)$, resp. $x_{n-1}(t)$. Řidič volí svoji rychlost na základě odstupu od předchozího vozidla, resp. zrychlení na základě rychlostí.

Konečný vztah pro Newellův model je tedy

$$a_n(t) = \frac{\frac{1}{\tau_n} \left[v_{n-1}(t) - v_n(t)\right] - \frac{d_n}{\tau_n} - v_n(t)}{T}, \tag{1.10}$$

kde vztah mezi τ_n a T vychází z rovnice (1.6).

U Newellova modelu není vůbec definované čelní vozidlo a jeho vlastnosti. Je např. možné nastavit mu konstantní rychlost a vlastnosti následujících vozidel vypočítat poté rovnice (1.10).

1.1.2 Parametry Newellova modelu

Všechny proměnné a parametry Newellova modelu potom jsou:

- τ_n – časová mezera mezi vozidly,
- T – reakční čas řidiče,
- d_n – odstup mezi vozidly v metrech,

[3] Předpokládejme vozidlo v klidu, které se rozjíždí. Na začátku zvolíme $\tau_n = 5\,\mathrm{s}, t = 0\,\mathrm{s}, T = \tau_n/2 = 2{,}5\,\mathrm{s}$, potom platí

$$x_n(5) = x_n(0) + 5v_n(0 + 2{,}5).$$

Pro výpočet dráhy bereme hodnotu rychlosti v čase 2,5 s. Nemůžeme vzít hodnotu v nule, vozidlo by v tomto případě stálo na místě. Přibližná hodnota bude tedy opravdu v polovině hodnoty τ_n.

- $v_n(t)$ – rychlost vozidla n v čase t,
- $a_n(t)$ – zrychlení vozidla v čase t.

Všechny parametry s indexem $n-1$ mají stejný význam, ale vztahují se k vozidlu jedoucímu před aktuálním.

1.2 Gippsův model

Další model, kterým se zde budeme zabývat, navrhl v roce 1981 anglický specialista na dopravní modelování P. G. Gibbs [9] a podle něj se také model dnes jmenuje. Gibbsův model vychází z Newellova modelu a na rozdíl od svého jednoduchého předchůdce se i dnes tento model využívá pro simulaci dopravy.

Většina modelů, které vznikly před Gippsovým, jsou různé variace na rovnici (Newellův ovšem ne)

$$a_n(t+T) = l_n \frac{[v_{n-1}(t) - v_n(t)]^k}{[x_{n-1}(t) - x_n(t)]^m}, \tag{1.11}$$

kde jednotlivé proměnné mají stejný význam jako při popisu Newellova modelu v podkapitole 1.1. Proměnné l_n, k a m jsou parametry, které je nutno empiricky odhadnout. Tyto modely začaly vytvářet výzkumníci ve společnosti General Motors a z tohoto důvodu se také nazývají Modely General Motors. Blíže jsou rozebrány v mé bakalářské práci [26].

Ačkoliv tyto modely podávají v mnoha situacích dobré výsledky, je žádoucí, aby během přepočtů poloh, rychlostí a zrychlení výpočet závisel na reakčním času řidiče T. Další koncepční slabinou GM modelů je fakt, že empritické parametry l_n, k a m v rovnici (1.11) nemají přímou souvislost s vlastnostmi vozidla či řidiče.

Gipps se proto rozhodl, že vytvoří model, který bude splňovat následujicí podmínky:

- model by měl napodobovat chování opravdového provozu,
- parametry modelu by měly odpovídat vlastnostem řidiče a vozidla,
- model by se měl chovat správně i v případě, že doba mezi přepočítáváním polohy a rychlosti je stejná, jako reakční čas řidiče.

1.2.1 Popis modelu

Gippsův model je odvozen z omezení jak schopností řidiče, tak i vozidla a pomocí těchto omezení se dopočítává bezpečná rychlost podle předcházejícího vozidla. Předpokládá se přitom, že řidič volí svoji rychlost tak, aby zajistil bezpečné zastavení nebo aby dokázal náhle rychle zastavit při nenadálé události.

První podmínka, kterou aplikujeme na vozidlo n, souvisí s rychlostí vozidla. Ta nepřekročí jeho preferovanou rychlost V_n a jeho zrychlení na volné komunikaci[4] se zvyšuje

[4] tj. komunikace, na které se nevyskytuje žádné další vozidlo

stejně tak, jako rychlost, dokud se také zvyšuje krouticí moment motoru vozidla a poté začne klesat na nulovou hodnotu a té nabude ve chvíli, kdy se vozidlo bude pohybovat svoji preferovanou rychlostí V_n,

$$v_n(t+T) \leq v_n(t) + 2{,}5\, a\, T \left(\frac{1 - v_n(t)}{V_n}\right) \sqrt{\left(0{,}025 + \frac{v_n(t)}{V_n}\right)}. \tag{1.12}$$

Koeficienty v nerovnici (1.12) byly stanoveny na základě z měření na hlavních dopravních komunikacích při průměrném provozu. Využití nerovnice (1.12) pro tento model je považované za přijatelné až do chvíle, kdy se vozidlo přiblíží vozidlu jedoucím před ním. Od té chvíle důraz na tuto podmínku klesá. Naopak největší váhu má tato podmínka ve chvíli, kdy vozidlo nemá nikoho či vozidlo před ním se nachází velmi daleko. Vozidlo se tedy v tuto chvíli pohybuje volnou rychlostí (z angl. *free-flow speed*).

Další omezení, které je nutno zmínit, se týká brždění. Jestliže $n-1$ (první) vozidlo zahájí brždění v čase t, zpomalí a následně zastaví v místě x^*_{n-1}, které dostáváme rovnicí

$$x^*_{n-1} = x_{n-1}(t) - \frac{v_{n-1}(t)^2}{2b_{n-1}}, \tag{1.13}$$

kde člen b_{n-1} označuje nejvyšší kritickou decelaraci, kterou je řidič vozidla $n-1$ schopen vykonat. Její hodnota je vždy záporná.

Vozidlo n, jedoucí za ním, nebude reagovat na toto zpomalení ihned, ale až v čase $t+T$ a tudíž zastaví až v místě x^*_n, daném rovnicí

$$x^*_n = x_n(t) + [v_n(t) + v_n(t+T)]\,\frac{T}{2} - \frac{v_n(t+T)^2}{2b_n}. \tag{1.14}$$

Pro vlastní bezpečnost musí řidič vozidla n zajistit, že bude splněna podmínka

$$x^*_{n-1} - l_{n-1} < x^*_n, \tag{1.15}$$

kde l_{n-1} je délka vozidla $n-1$, sečtena s bezpečnou vzdáleností následujícího (n-tého) vozidla. Pokud podmínka (1.15) neplatí, n-té vozidlo nemá žádný prostor pro případnou řidičovu chybu. Z toho důvodu obsahuje Gippsův model ještě další podmínku, udávající dodatečnou časovou rezervu řidiče pro bezpečný odstup, θ. Jak již víme, řidič začne reagovat na vozidlo před sebou až v čase $t+T$. Od této chvíle tedy máme reakční čas T a bezpečný reakční čas $T+\theta$, který budeme ve výpočtech dále využívat. S využitím rovnic (1.13) a (1.14) můžeme podmínku (1.15) přepsat do nerovnice

$$x_{n-1}(t) - \frac{v_{n-1}(t)^2}{2b_{n-1}} - l_{n-1} \geq x_n(t) + \\ + [v_n(t) + v_n(t+T)]\,\frac{T}{2} + v_n(t+T)\theta - \frac{v_n(t+T)^2}{2b_n}. \tag{1.16}$$

Bez parametru θ by řidič vozidla byl nucen brzdit až ve chvíli, kdy opravdu musí, to znamená na maximální výkon brzd. Do té chvíle by se stále pohyboval svoji preferovanou rychlostí. Parametr θ slouží k dřívějšímu a přitom ne tak prudkému brždění.

Při skutečném pozorování dopravy jsme schopni změřit všechny parametry vozidla n v rovnici (1.16) kromě členu b_{n-1}. Tento člen nahradíme odhadnutou hodnotou $\hat{b}$, upravíme tuto rovnici a dostáváme

$$-\frac{v_n(t+T)^2}{2b_n} + v_n(t+T)\left(\frac{T}{2}+\theta\right) - \\ - [x_{n-1}(t) - l_{n-1} - x_n(t)] + v_n(t)\frac{T}{2} + \frac{v_{n-1}(t)^2}{2\hat{b}} \leq 0. \quad (1.17)$$

Relativní hodnoty T a θ jsou důležité v určování chování vozidel. Za normálních okolností se dosazuje za $\theta = T/2$. Díky tomu můžeme rovnici (1.17) přepsat do tvaru

$$-\frac{v_n(t+T)^2}{2b_n} + v_n(t+T)T - [x_{n-1}(t) - l_{n-1} - x_n(t)] + v_n(t)\frac{T}{2} + \frac{v_{n-1}(t)^2}{2\hat{b}} \leq 0. \quad (1.18)$$

Z toho po složitých úpravách plyne

$$v_n(t+T) \leq b_n T + \sqrt{\left(b^2T^2 - b\left(2\left[x_{n-1}(t) - l_{n-1} - x_n(t)\right] - v_n(t)T - \frac{v_{n-1}(t)^2}{2\hat{b}}\right)\right)}. \quad (1.19)$$

Z nerovnice (1.18) vychází, že bezpečné rychlosti v_n budou ležet mezi dvěma kořeny této nerovnice. Pokud bude mít spodní kořen zápornou hodnotu můžeme jej ignorovat, protože nás zajímají pouze kladné hodnoty rychlostí. Zároveň tedy musí platit

$$v_n \geq 0. \quad (1.20)$$

Podcenění plynulého brždění řidiče vozidla n nastane ve chvíli, kdy

$$v_n(t+T) < v_n(t) + b_n T.$$

V tuto chvíli musí vozidlo začít brzdit rychleji, než by sám řidič chtěl. Řidič tedy volí rychlost na základě preferované decelarace, ale může brzdit i silněji, pokud by to bylo nutné.

Nerovnice (1.12) pro vozidlo n na volné komunikaci a (1.19) pro vozidlo n před kterým se nachází vozidlo $n-1$ představují dvě hlavní podmínky pro rychlost v_n v čase $t+T$ a jestliže řidič přizpůsobí rychlost parametrům vozidla a tak, aby jízda byla bezpečná, dostáváme **konečný vztah pro Gippsův model:**

$$v_n(t+T) = \min\left(v_n(t) + 2,5\,a\,T\left(1 - \frac{v_n(t)}{V_n}\right)\sqrt{\left(0,025 + \frac{v_n(t)}{V_n}\right)}, \right. \\ \left. b\,T + \sqrt{\left(b^2T^2 - b\left(2\left[x_{n-1}(t) - l_{n-1} - x_n(t)\right] - v_n(t)T - \frac{v_{n-1}(t)^2}{2\hat{b}}\right)\right)}\right) \quad (1.21)$$

Pokud platí pro téměř všechna vozidla nižší hodnota v rovnici (1.21), vychází se z nerovnice (1.19), vozidla jedou blízko sebe, z čehož vyplývá vysoká hustota provozu. Pokud platí opak, tzn. nerovnice (1.12), dopravní proud je volný.

1.2.2 Parametry Gippsova modelu

Všechny proměnné a parametry Gippsova modelu potom jsou:

- T – reakční čas řidiče,
- $v_n(t)$ – rychlost vozidla n v čase t,
- a – maximální hodnota zrychlení, kterým je řidič ochoten zrychlovat - *nikoliv tedy zrychlení vozidla v čase t!*,
- V_n – preferovaná rychlost vozidla n,
- b – maximální decelerace vozidla, kterým je řidič ochoten brzdit,
- $x_n(t)$ – poloha vozidla n
- l_{n-1} – délka vozidla $n-1$ sečtena s bezpečnou vzdáleností n-tého vozidla
- $\hat{b}$ – odhadnutá hodnota decelerace b_{n-1}

1.3 Kraussův model

Tento model vytvořil S. Krauss na Universität Köln v rámci své disertační práce [17]. V popisu modelu v této podkapitole z tohoto zdroje také vycházím.

Tento model se hlavně zaměřuje na vlastnosti dopravního proudu, zkoumání chování řidiče zde není primární. Myšlenka autora je taková, že existují obecné vlastnosti dopravního proudu, ze kterých vyplývá chování jednotlivých účastníků silničního provozu a chování jednotlivce významně neovlivňuje vlastnosti dopravního proudu.

1.3.1 Obecný přístup

Pokud se dopravní proud modeluje mikroskopicky, musí být brány v úvahu dva typy pohybu vozidla. Prvním typem pohybu je vozidlo jedoucí po vozovce osamoceno, zatímco ve druhém případě vozidlo interaguje s ostatními vozidly. Na základě těchto pohybů předpokládáme následující podmínky. První podmínka má přímo souvislost s rychlostí vozidla. Ta je omezena nějakou maximální rychlostí $v_{\max}$,

$$v(t) \leq v_{\max}. \tag{1.22}$$

Za maximální rychlost může být například zvolena preferovaná rychlost vozidla.

Hlavní důvod, proč mezi sebou vozidla interagují, je fakt, že řidiči nemají v úmyslu srážet se s ostatními vozidly. Další podmínkou tedy bude tzv. „nekoliznost“ systému.

Budeme předpokládat, že řidič vozidla zvolí tedy takovou rychlost vozidla, která nebude vyšší než maximální bezpečná rychlost v_{safe},

$$v(t) \leq v_{\text{safe}}(t). \tag{1.23}$$

Z maximální bezpečné rychlosti poté plyne interakce mezi vozidly. Její stanovení bude uvedeno dále.

Je možné formulovat modely na základě podmínek (1.22) a (1.23). Nicméně je vcelku vhodné předpokládat také omezující podmínky pro akceleraci a a deceleraci b,

$$\begin{aligned} -b \leq \frac{\mathrm{d}v}{\mathrm{d}t} \leq a, \\ a, b > 0. \end{aligned} \tag{1.24}$$

Později bude ukázáno, že vztah (1.24) je nezbytnou podmínkou správného modelování dopravního proudu.

Předchozí podmínky můžeme shrnout do následující nerovnice

$$v(t + \Delta t) \leq \min(v_{\max}, v_{\text{safe}}, v(t) + a\Delta t), \tag{1.25}$$

kde maximální bezpečná rychlost v_{safe} musí splňovat podmínku

$$v(t + \Delta t) \geq v(t) - b\Delta t. \tag{1.26}$$

Celkové povědomí o tom, jak se jednotlivá vozidla pohybují a interagují mezi sebou, závisí na výpočtu maximální bezpečné rychlosti v_{safe}. Jakmile je stanovena hodnota v_{safe}, nerovnice (1.25) představuje schéma pro simulace dopravního proudu, pokud jsou splněny všechny podmínky zmíněné výše.

1.3.2 Interakce mezi vozidly

Každý sofistikovaný dopravní model by měl obsahovat interakci mezi vozidly. Představme si dvě vozidla na komunikaci, první vozidlo se nachází v bodě $x_1(t)$ a jede rychlostí $v_1(t)$, vozidlo, které ho následuje se nachází v bodě $x_f(t)$ a pohybuje se rychlostí $v_f(t)$. Jestliže délka prvního vozidla bude l, mezera mezi vozidly se vypočte vztahem

$$g(t) = x_1(t) - x_f(t) - l. \tag{1.27}$$

Jak již bylo zmíněno, hlavním důvodem, proč mezi sebou vozidla interagují, je fakt, že řidiči nemají v úmyslu srážet se s ostatními vozidly. Z toho plyne, že mezera mezi vozidly musí být nezáporná. Narozdíl od jiných modelovacích přístupů, zde nezačneme s předpokladem, že se zrychlení vozidla vypočte ze zrychlení vozidla jednoucího před ním, protože tím nekoliznost stejně není zajištěna automaticky.

Místo toho zavedeme pravidlo, že se ve spojitých modelech spolu interagující vozidla nesrazí tehdy, pokud je mezera mezi vozidly g větší než nějaká preferovaná mezera mezi vozidly g_{des} a splňuje tuto dynamickou nerovnici

$$\frac{\mathrm{d}g(t)}{\mathrm{d}t} \geq \frac{g_{\text{des}} - g(t)}{\tau_{\text{des}}}. \tag{1.28}$$

Preferovaný bezpečný časový odstup mezi vozidly τ_{des} a preferovaná mezera g_{des} by mohly být funkcemi odstupu mezi vozidly či jejich rychlostmi, ale v tomto případě to jsou konstanty. Fakt, že v tomto modelu nemohou nastat kolize, je zřejmý, protože pokud položíme g rovné nule, časová derivace v rovnici (1.28) bude vždy nezáporná neboť nutně $g_{\text{des}} > 0$ a $\tau_{\text{des}} > 0$.

Uvažujme stále případ dvou vozidel jedoucími rychlostmi $v_1(t)$ resp. $v_f(t)$ s odstupem $g(t)$. Druhé vozidlo jede svoji bezpečnou rychlostí a řidič tohoto vozidla může zastavit za jakýkoliv okolností tak, aby nedošlo ke kolizi s prvním vozidlem. To znamená, že pokud řidič má reakční čas T a brzdná vzdálenost vozidel jedoucích rychlostí $v(t)$ je dána funkcí $f(v(t))$, je situace bezpečná právě tehdy když

$$f(v_f(t)) + v_f(t)T \leq f(v_1(t)) + g(t). \tag{1.29}$$

Hodnota funkce $f(v_f(t))$ či $f(v_1(t))$ nemusí být nutně minimální možná brzdná vzdálenost, ale může to být funkce závisející na způsobu řízení řidiče. To samé platí pro reakční čas T.

Úprava nerovnice (1.29) vede k bezpečnostnímu pravidlu pro rychlost druhého vozidla. Nicméně my toto pravidlo neodvozujeme ze dvou důvodů. První z nich je ten, že po úpravě bychom potřebovali znát přesný předpis funkce $f(v_f(t))$ či $f(v_1(t))$, který, jak uvidíme, není možné zjistit. Druhým je nesmyslné a zbytečné počítat možnou strategii řidiče ve chvíli nenadálé události a tím pádem také vypočítávat decelaraci až k úplnému zastavení každý jednotlivý krok. Proto zavedeme Taylorův rozvoj brzdné vzdálenosti okolo průměrné rychlosti obou vozidel,

$$\overline{v}(t) = \frac{v_1(t) + v_f(t)}{2}. \tag{1.30}$$

Taylorův rozvoj je obecně definován následovně

$$f(x) = f(a) + \frac{f^{(1)}(a)}{1!}(x-a) + \frac{f^{(2)}(a)}{2!}(x-a)^2 + \frac{f^{(3)}(a)}{3!}(x-a)^3 \ldots, \tag{1.31}$$

kde a je bod, ve kterém budeme funkci rozvíjet a horní indexy u funkcí značí příslušnou derivaci.

Budeme pro zjednodušení uvažovat pouze první 4 členy Taylorova rozvoje. Potřebujeme rozvinout funkce $f(v_1(t))$ a $f(v_f(t))$ z rovnice (1.29) v okolí bodu $\overline{v}(t)$ z rovnice (1.30). Po vyjadření funkce $f(v_1(t))$ Taylorovým rozvojem, rovnice (1.31), dostáváme

$$\begin{aligned} f(v_1(t)) = f(\overline{v}(t)) &+ \frac{f^{(1)}(\overline{v}(t))}{1!}\left(v_1(t) - \frac{v_1(t)+v_f(t)}{2}\right) \\ &+ \frac{f^{(2)}(\overline{v}(t))}{2}\left(v_1(t) - \frac{v_1(t)+v_f(t)}{2}\right)^2 + \frac{f^{(3)}(\overline{v}(t))}{6}\left(v_1(t) - \frac{v_1(t)+v_f(t)}{2}\right)^3, \end{aligned} \tag{1.32}$$

kde po úpravách závorek dostaneme

$$f(v_1(t)) = f(\overline{v}(t)) + \frac{f^{(1)}(\overline{v}(t))}{1!}\left(\frac{v_1(t)-v_f(t)}{2}\right) + \frac{f^{(2)}(\overline{v}(t))}{2}\left(\frac{v_1(t)-v_f(t)}{2}\right)^2 + \frac{f^{(3)}(\overline{v}(t))}{6}\left(\frac{v_1(t)-v_f(t)}{2}\right)^3. \tag{1.33}$$

Stejný postup provedeme i pro $f(v_f(t))$

$$f(v_f(t)) = f(\overline{v}(t)) + \frac{f^{(1)}(\overline{v}(t))}{1!}\left(\frac{v_f(t)-v_1(t)}{2}\right) + \frac{f^{(2)}(\overline{v}(t))}{2}\left(\frac{v_f(t)-v_1(t)}{2}\right)^2 + \frac{f^{(3)}(\overline{v}(t))}{6}\left(\frac{v_f(t)-v_1(t)}{2}\right)^3. \tag{1.34}$$

a oba tyto rozvoje dosadíme do rovnice (1.29). Protože platí

$$\left(\frac{v_f(t)-v_1(t)}{2}\right)^k = \left(\frac{v_1(t)-v_f(t)}{2}\right)^k, \tag{1.35}$$

kde k je sudé číslo, vyruší se všechny sudé členy tohoto rozvoje a zanedbáním vyšších řádů dostáváme

$$f^{(1)}(\overline{v}(t))\frac{v_f(t)-v_1(t)}{2} + v_f(t)T \leq f^{(1)}(\overline{v}(t))\frac{v_1(t)-v_f(t)}{2} + g(t). \tag{1.36}$$

Po roznásobení a převedení členů rovnice na opačné strany získáme

$$f'(\overline{v}(t))v_f + v_f T \leq f'(\overline{v}(t))v_1 + g, \tag{1.37}$$

kde budeme nyní první derivaci značit f'.

Smysl derivace $f'(\overline{v})$ vysvětlíme jednoduše. Pokud se podívame na decelaraci vozidla b z rychlosti v na nulu (nemusí být nutně konstantní), tedy $\frac{\mathrm{d}\overline{v}}{\mathrm{d}t} = -b(v)$, kdy $b(v) > 0$, obrdžíme vztah

$$f'(\overline{v}(t)) = -\frac{\mathrm{d}}{\mathrm{d}\overline{v}(t)}\int_v^0 \frac{\overline{v}(t)'}{b(\overline{v}(t)')}\mathrm{d}v' = \frac{\overline{v}(t)}{b(\overline{v}(t))}. \tag{1.38}$$

Nyní můžeme přepsat podmínku bezpečnosti do tvaru

$$v_1 - v_f \geq \frac{v_1 T - g}{\frac{\overline{v}}{b(\overline{v})} + T}, \tag{1.39}$$

kde

$$v_1 - v_f = \frac{\mathrm{d}g}{\mathrm{d}t}. \tag{1.40}$$

Nerovnice (1.39) se rovná nerovnici (1.28) ve chvíli, kdy je preferovaný odstup $g_{\mathrm{des}} = v_1 t$ a preferovaný bezpečný časový odstup mezi vozidly $\tau_{\mathrm{des}} = \tau_b + T$, kde $\tau_b = \overline{v}(t)/b(\overline{v}(t))$ je stanoven na základě decelerací, které řidič využije.

1.3.3 Diskretizace modelu

Pro možnost simulace na počítači je potřeba převést spojité nerovnice odvozené v části 1.3.1, na diskrétní rovnice pro dráhy, rychlosti a zrychlení.

Vhodná cesta pro diskretizaci rovnic dynamického systému z bezpečnostní podmínky (1.28) je převést spojitou rychlost v_f ve výrazu $g'(t) = v_1(t) - v_f(t)$ jako rychlost diskrétní s diferencí či integračním krokem Δt. Potom nerovnici (1.28) přepíšeme do tvaru

$$v_f(t + \Delta t) \leq v_1(t) + \frac{g(t) - g_{\text{des}}}{\tau_{\text{des}}}. \tag{1.41}$$

Rovnice pro polohu vozidla se potom na stejném principu převede na

$$x(t + \Delta t) = x(t) + v(t + \Delta t)\Delta t. \tag{1.42}$$

Víme, že při $\Delta t \to 0$ a $g_{\text{des}} \geq 0$ nerovnice (1.41) a rovnice (1.42) garantují nekoliznost vozidel. Pro konečný čas Δt musí být tato garance opět dokázána.

Mezera $g(t)$ mezi dvěma vozidly je diskretizována vztahem

$$g(t + \Delta t) = g(t) + (v_1(t + \Delta t) - v_f(t + \Delta t))\Delta t. \tag{1.43}$$

Dosazení za $v_f(t + \Delta t)$ z rovnice (1.41) dostáváme

$$\xi(t + \Delta t) = \xi(t) + \left(1 - \frac{\Delta t}{\tau_b + T}\right) + \frac{g_{\text{des}} - v_1(t)\Delta t}{\tau_b + T}\Delta t, \tag{1.44}$$

kde

$$\xi(t) = g(t) - v_1(t)\Delta t.$$

Bezpečnost ($g \geq 0$) je tedy garantována tehdy, když platí:

$$\begin{aligned} &\xi(t = 0) \geq 0, \\ &\Delta t \leq T, \\ &g_{\text{des}} \geq v_1 \Delta t. \end{aligned} \tag{1.45}$$

To je výsledek, který se ovšem dal předpokládat. Záleží pouze na tom, zda je i po provedení iteračního kroku stále splněna podmínka bezpečnosti, tzn. kontrolovat jestli je náš iterační krok menší než reakční čas řidiče T.

1.3.4 Rovnice modelu

Předpokládáme tedy, že se každé vozidlo pohybuje nejvyšší rychlostí na základě podmínek, které jsou stanoveny níže. **Kraussův model tedy stanovíme následujícími rovnicemi:**

$$\begin{aligned} &v_{\text{safe}}(t) = v_1(t) + \frac{g(t) - g_{\text{des}}}{\tau_b + T}, \\ &v_{\text{des}} = \min\left(v_{\max}, v(t) + a(t)\Delta t, v_{\text{safe}}(t)\right), \\ &v(t + \Delta t) = \max(0, v_{\text{des}} - \eta). \end{aligned} \tag{1.46}$$

Nově zde také zavádíme proměnnou η, která je náhodnou výchylkou od běžného řízení. Preferovaná mezera g_{des} může být zvolena různě. Většinou se volí $g_{\text{des}} = v_1\tau$, kde τ je reakční čas řidiče. Hodnota τ_b je definována jako podíl $\overline{v}/b$, kde $\overline{v}$ plyne ze vztahu (1.30). Volba integračního kroku Δt a preferované mezery g_{des} je závislá na podmínkách (1.45).

1.3.5 Parametry Kraussova modelu

Všechny proměnné a parametry Kraussova modelu potom jsou:

- $g(t)$ – mezera mezi vozidly v čase t,
- g_{des} – preferovaná mezera mezi vozidly,
- τ_b – čas závislý na rychlosti decelerace,
- T – reakční čas řidiče,
- $v_1(t)$ – rychlost aktuálního vozidla v čase t,
- $a(t)$ – zrychlení v čase t,
- ΔT – integrační krok,
- η – náhodná výchylka od běžného řízení.

1.4 Rozšířený Kraussův model

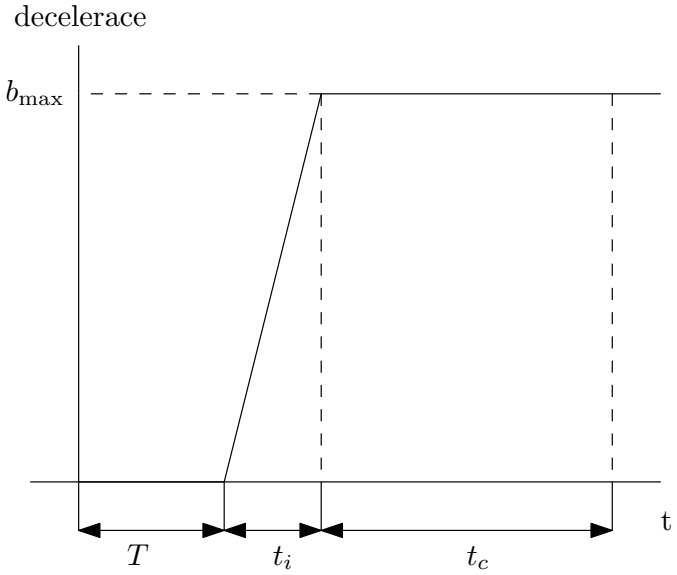
Hlavní zdroj pro vylepšení Kraussova modelu je přímo článek pro implementaci Rozšířeného Kraussova modelu [22]. Základ pro vznik tohoto vylepšení Kraussova modelu plyne z procesu brždění.

1.4.1 Proces brždění

Fáze brzdicího procesu na sebe plynule navazují, první z nich se nazývá doba reakce řidiče T, dále zvyšování decelarace t_{i} a maximální brždění t_{c}. Vše je graficky znázorněno na obrázku 1.4.

Doba reakce řidiče: Řidič potřebuje určitý čas T na to, aby zareagoval na změnu chování vozidla před sebou. V době této reakce se pohybuje rychlost vozidla v_0 na konstantní hodnotě, případně se lehce mění. Posunutí aktuálního vozidla je potom dán rovnicí

$$S_{\text{r}} = v_0 T. \tag{1.47}$$



Obrázek 1.4: Fáze brždění, zdroj: [22]

Zvyšování decelerace: Decelarace aktuálního vozidla se ve druhé fázi zvyšuje z nuly na hodnotu $b_{\max}$, která představuje nejvyšší hodnotu decelerace vozidla. Předpokládáme lineární zvyšování decelarace,

$$b = b_{\max}\frac{t}{t_{\mathrm{i}}}.$$

Aktuální rychlost vozidla se potom vypočte jako

$$v_i = v_0 - \int_0^{t_{\mathrm{i}}} b_{\max}\frac{t}{t_{\mathrm{i}}}\mathrm{d}t = v_0 - \frac{b_{\max}}{2t_{\mathrm{i}}}t^2, \tag{1.48}$$

a v této části vozidlo urazí vzdálenost

$$S_i = \int_0^{t_{\mathrm{i}}} \left(v_0 - \frac{b_{\max}}{2t_{\mathrm{i}}}t^2\right)\mathrm{d}t = v_0 t_i - b_{\max}\frac{b_{\max}t_{\mathrm{i}}^2}{6}. \tag{1.49}$$

Maximální brždění: V tuto chvíli již vozidlo brzdí svoji maximální decelerací $b_{\max}$, pohyb vozidla je potom dán vzorcem

$$S_c = \frac{v_0^2}{2b_{\max}} - \frac{v_0 t_i}{2} + \frac{b_{\max}t_i^2}{8}. \tag{1.50}$$

Celková dráha při brždění: Celková dráha při brždění vozidla se vypočítá jako součet tří předcházejících částí, tedy

$$S = S_r + S_i + S_c = v_0 T + v_0 t_i - b_{\max}\frac{b_{\max} t_i^2}{6} + \frac{v_0^2}{2b_{\max}} - \frac{v_0 t_i}{2} + \frac{b_{\max} t_i^2}{8},$$

a po úpravách dostaneme vztah

$$S = v_0 T + \frac{1}{2} v_0 t_i + \frac{v_0^2}{2b_{\max}} - \frac{b_{\max} t_i^2}{24} \approx v_0 \left(t_r + \frac{1}{2} t_i \right) + \frac{v_0^2}{2b_{\max}}, \tag{1.51}$$

kde člen $-\frac{b_{\max} t_i^2}{24}$ zanedbáme, jelikož má několikanásobně menší hodnotu než ostatní.

1.4.2 Matematický popis

Klíč ke správnému modelu sledu vozidel je dle Krausse je hlavně bezpečná rychlost. Pokud čelní vozidlo sníží rychlost, aktuální vozidlo musí začít brzdit. Pokud obě vozidla zastaví, jejich polohu bude možno vypočíst na základě vztahu

$$S_\mathrm{l} + g(t) = S_\mathrm{f}, \tag{1.52}$$

kde S_l představuje polohu čelního vozidla, $g(t)$ mezeru mezi oběma vozidly a S_f polohu aktuálního vozidla.

Pokud pomineme reakční čas čelního řidiče, brždění se skládá ze dvou částí probíhajících v čase: zvyšování decelarace t_li a doby maximálního brždění t_lc. Poloha čelního vozidla je v celém brzdicím průběhu dána rovnicí

$$S_\mathrm{l} = S_\mathrm{li} + S_\mathrm{lc} = \frac{1}{2} v_\mathrm{l} t_\mathrm{li} + \frac{v_\mathrm{l}^2}{2a_{\max}} - \frac{b_{\max} t_\mathrm{li}^2}{24} \approx \frac{1}{2} v_\mathrm{l} t_\mathrm{li} + \frac{v_\mathrm{l}^2}{2a_{\max}}, \tag{1.53}$$

kde $a_{\max}$ je maximální decelerace čelního vozidla.

Poloha aktuálního vozidla (vozidla jedoucího za čelním) je v celém brzdicím procesu rovna

$$S_\mathrm{f} = S_\mathrm{r} + S_\mathrm{i} + S_\mathrm{c} = v_\mathrm{safe} T + \frac{1}{2} v_\mathrm{safe} t_\mathrm{i} + \frac{v_\mathrm{safe}^2}{2b_{\max}} - \frac{b_{\max} t_\mathrm{fi}^2}{24} \approx v_\mathrm{safe} \left(T + \frac{1}{2} t_\mathrm{fi} \right) + \frac{v_\mathrm{safe}^2}{2b_{\max}}. \tag{1.54}$$

Pokud do rovnice (1.52) dosadíme za S_l z rovnice (1.53) a za S_f z rovnice (1.54) dostáváme

$$v_\mathrm{safe} \left(t_\mathrm{r} + \frac{1}{2} t_\mathrm{fi} \right) + \frac{v_\mathrm{safe}^2}{2b_{\max}} - \frac{1}{2} v_\mathrm{l} t_\mathrm{li} - \frac{v_\mathrm{l}^2}{2a_{\max}} - g = 0. \tag{1.55}$$

Z této rovnice už můžeme přes výpočet diskriminantu vyjádřit v_safe,

$$v_\mathrm{safe} = -b \left(T + \frac{t_\mathrm{fi}}{2} \right) + \sqrt{b^2 \left(t_\mathrm{r} + \frac{t_\mathrm{fi}}{2} \right)^2 + b \left(v_\mathrm{l} t_\mathrm{li} + \frac{v_\mathrm{l}^2}{a} + 2g \right)}. \tag{1.56}$$

V simulaci je potom čas decelerace stejný $t_i = t_{\mathrm{li}} = t_{\mathrm{fi}}$, a toho plyne

$$v_{\mathrm{safe}} = -b\left(T + \frac{t_{\mathrm{i}}}{2}\right) + \sqrt{b^2\left(T + \frac{t_{\mathrm{i}}}{2}\right)^2 + b\left(v_{\mathrm{l}}t_{\mathrm{i}} + \frac{v_{\mathrm{l}}^2}{a} + 2g\right)}. \tag{1.57}$$

Pokud čelní vozidlo stojí, je jeho rychlost $v_{\mathrm{l}} = 0\,\mathrm{ms}^{-1}$ a jeho poloha se nemění $S_{\mathrm{l}} = 0$, rovnici (1.57) můžeme zjednodušit na

$$v_{\mathrm{safe}} = -b\left(T + \frac{t_{\mathrm{i}}}{2}\right) + \sqrt{b^2\left(T + \frac{t_{\mathrm{i}}}{2}\right)^2 + 2bg}. \tag{1.58}$$

Tato rovnice (1.58) je jakýmsi doplňkem ke Kraussovu modelu, pouze přesněji definuje bezpečnou rychlost. Princip Kraussova modelu stále zůstává ve volbě preferované rychlosti, kterou se vozidlo bude pohybovat. Tato volba je dána hodnotou v_{des} v rovnici (1.46).

1.4.3 Parametry Rozšířeného Kraussova modelu

Všechny proměnné a parametry Kraussova modelu potom jsou:

- g – aktuální mezera mezi vozidly,
- b – aktuální decelerace,
- t_i – čas, při kterém je zvyšována decelerace,
- T – reakční čas řidiče.

1.5 Intelligent Driver Model

Intelligent Driver Model (dále jen IDM) je poměrně nový matematický model z roku 2000, který byl vymyšlen v Německu. Používá se jiný přístup výpočtu akcelerací vozidel, kde se nevybírá z několika hodnot (viz Gipps či Krauss), ale důležitost jednoho členu klesá v závislosti na vzdálenosti mezi vozidly.

V této sekci budeme primárně vycházet z prací [26] a [25].

1.5.1 Matematický popis

Pro výpočet zrychlení v IDM je použit vztah

$$v_n(t + \Delta t) = v_n(t) + a_{n,\mathrm{free}}(t) + a_{n,\mathrm{int}}(t), \tag{1.59}$$

kde $v_n(t)$ značí rychlost vozidla v čase t a Δt je simulační krok. Aktuální zrychlení vozidla se tedy vypočítá jako součet zrychlení při volném proudu $a_{n,\mathrm{free}}$ a zrychlení, ve kterém je zahrnut i vozidlo jedoucí před ním $a_{n,\mathrm{int}}$. Zde je vidět určitá paralela s Gippsovým

modelem (kapitola 1.2), kde se ovšem vybírá pouze jedna z těchto dvou hodnot. Navíc i rovnice se od IDM odlišují. Pro volný proud je tedy zrychlení $a_{n,\text{int}} = 0$.

Zrychlení při volném proudu je definováno vztahem

$$a_{n,\text{free}}(t) = a\left[1 - \left(\frac{v_n(t)}{v_{n,\text{free}}}\right)^{\delta}\right], \tag{1.60}$$

kde a je maximální zrychlení, δ je exponent zrychlení, který napomahá zpřesnění výpočtu zrychlení $a_{n,\text{free}}$, v_n je aktuální rychlost vozidla a $v_{n,\text{free}}$ je rychlost, které chce vozidlo dosáhnout.

Pokud v rovnici (1.60) bude zlomek $v_n(t)/v_{n,\text{free}}$ roven jedné, vozidlo se již dostalo na svou preferovanou rychlost $v_{n,\text{free}}$ a celkové zrychlení $a_{n,\text{free}}(t)$ bude rovno nule. Naopak pokud bude $v_n(t)$ rovno nule, znamená to, že $a_{n,\text{free}}(t)$ bude přímo rovno a, a vozidlo pojede svým maximálním zrychlením. Pokud $v_n(t)$ bude menší než $v_{n,\text{free}}$, pak platí že, čím vyšší hodnotu bude mít nastavenou parametr δ, tím se dosáhne vyšší hodnota $a_{n,\text{free}}(t)$. Pokud $v_n(t)$ bude větší než $v_{n,\text{free}}$, bude platit pravý opak.

Podle rovnice (1.60) by bylo možné počítat celkové zrychlení za předpokladu, kdyby vozidlo jelo na naprosto prázdné komunikaci. Tomu ovšem tak být nemusí, proto se do rovnice (1.59) přidává interakční člen, který přímo závisí na parametrech vozidla jedoucí před naším účastníkem. Tento člen je roven

$$a_{n,\text{int}}(t) = -a\left(\frac{s_n^*(t)}{s_n(t)}\right)^2, \tag{1.61}$$

kde a je maximální zrychlení, $s_n(t) = x_{n-1}(t) - x_n(t)$ je rozdíl poloh vozidel a $s_n^*(t)$

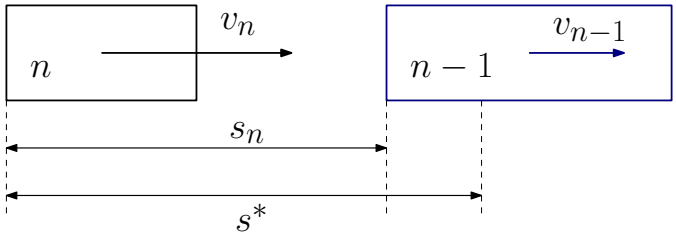
$$s_n^*(t) = s_0 + v_n(t)\tau + \frac{v_n(t)(v_n(t) - v_{n-1}(t))}{2\sqrt{ab}} \tag{1.62}$$

je preferovaný odstup. Hodnota s_0 se vypočte jako součet minimální bezpečné vzdálenosti mezi vozidly a délkou následujícího vozidla, τ je bezpečnostní časový odstup mezi vozidly a b je konstanta brzdění. Situace je schématicky znázorněna na obrázku 1.5, kde jsme pro přehlednost neuvedli závislost jednotlivých proměnných na čase.

Člen v čitateli zlomku rovnice (1.62) je kvadratický v proměnné $v_n(t)$ a obsahuje součin $v_n(t)v_{n-1}(t)$. Tyto nelineární členy popisují interakci, v tomto případě dvou vozidel. Pokud tedy bude vzdálenost $s_n(t)$ mezi vozidly vysoká, potom zlomek $s_n^*(t)/s_n(t)$ bude mít velmi malou hodnotu a interakční člen $a_{n,\text{int}}(t)$ lze zanedbat, takže vozidlo se pohybuje jako ve volném proudu.

Pokud rychlost vozidla $v_n(t)$ bude vyšší než rychlost vozidla $v_{n-1}(t)$, pak interakční člen $s_n^*(t)$ bude kladný a požadovaný odstup se zvětší, takže zrychlení vozidla $v_n(t+\Delta t)$ se musí snížit.

Dosazením do rovnice (1.59) za $a_{n,\text{int}}(t)$ z rovnice (1.61) a $a_{n,\text{free}}(t)$ z rovnice (1.60)



Obrázek 1.5: Proměnné jednotlivých vozidel, zdroj: [26]

dostáváme **konečný vztah pro zrychlení vozidla**:

$$v_n(t+\Delta t) = v_n(t) + a\left[1 - \left(\frac{v_n(t)}{v_{n\text{free}}}\right)^{\delta}\right] - a\left(\frac{s_0 + v_n(t)\tau + \frac{v_n(t)(v_n(t)-v_{n-1}(t))}{2\sqrt{ab}}}{s_n(t)}\right)^2. \tag{1.63}$$

1.5.2 Parametry IDM

Všechny proměnné a parametry IDM potom jsou:

- s_n – aktuální odstup mezi vozidly,
- τ – bezpečnostní časový odstup mezi vozidly,
- v_n – rychlost aktuálního vozidla,
- v_{n-1} – rychlost $n-1$ vozidla,
- a – maximální hodnota zrychlení, kterým je řidič ochoten zrychlovat,
- b – maximální decelerace,
- s_0 – bezpečnostní mezera mezi vozidly,
- $v_{n\text{free}}$ – preferovaná rychlost vozidla.

1.6 Wiedemannův model

V programu SUMO je také připraven Wiedemannův model, jehož první verze vznikla již v roce 1974 jako disertační práce tehdejšího studenta R. Wiedemanna. Tento model byl posléze několikrát vylepšován, v roce 1992 o tomto modelu vyšel aktualizovaný článek. Wiedemannova disertační práce je bohužel psána německy a navíc je v podstatě nedostupná, stejně jako zmiňovaný článek. V této sekci budeme hlavně vycházet z článku [12]. V něm najdeme poměrně přesný popis Wiedemannova modelu[5].

[5]Tento článek se přímo odkazuje na článek z roku 1992.

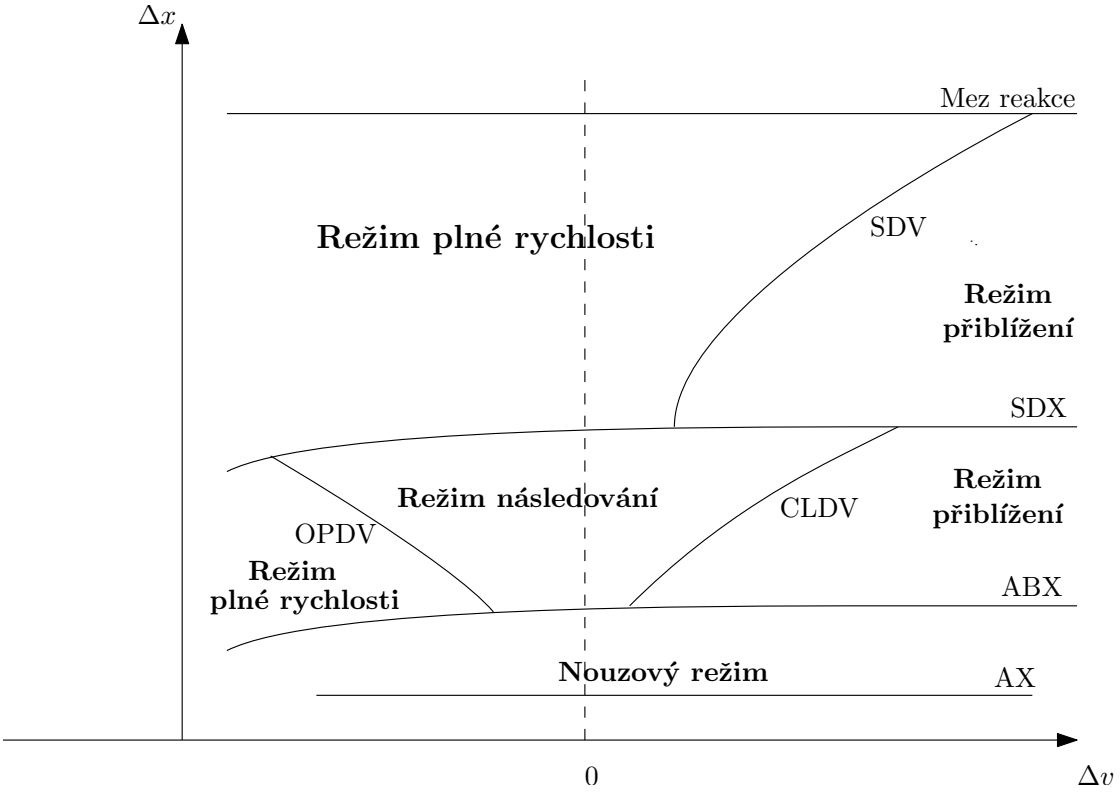
Tento model je odlišný oproti předcházejícím tím, že pracuje v tzv. režimech. Nejprve je nutné zjistit, v jakém režimu se vozidlo vlastně nachází a podle toho se určuje výpočet akcelerace či decelarace, která je pro každý režim jiná.

Klíčové pro pochopení Wiedemannova modelu je obrázek 1.6. Osa x zobrazuje rozdíl rychlostí vozidel Δv (aktuálního vozidla a vozidla jedoucího před ním) a na ose y se nachází vzdálenost mezi těmito vozidly Δx. Znamená to, že pokud rozdíl rychlostí a zároveň rozdíl poloh vozidel dosahuje určitých hodnot, lze tyto hodnoty přiřadit jednomu konkrétnímu režimu, podle kterého je vozidlo dále zpracováváno. K tomu, abychom mohli určit příslušný režim vozidla, musíme nejprve určit prahové hodnoty křivek AX, ABX, SDX, CLDV, OPDV a SDV, znázorněných na obrázku 1.6.

Preferovaný odstup mezi stacionárními[6] vozidly značíme AX. Je definován vztahem

$$\mathrm{AX} = L_{n-1} + \mathrm{AXadd} + \mathrm{RND1}_n \cdot \mathrm{AXmult}, \tag{1.64}$$

kde AXadd a AXmult jsou parametry modelu, $\mathrm{RND1}_n$ značí parametr závislý na řidiči generovaný z normálního rozdělení a L_{n-1} je délka čelního vozidla sečtená s preferovanou vzdáleností mezi vozidly.



Obrázek 1.6: Režimy Wiedemannova modelu, zdroj: [12]

[6] nezávislé na čase

ABX je preferovaná maximální vzdálenost mezi vozidly při malých rozdílech rychlosti. Ten se vypočítá následovně:

$$\mathrm{ABX} = \mathrm{AX} + \mathrm{BX}, \tag{1.65}$$

kde

$$\mathrm{BX} = (\mathrm{BXadd} + \mathrm{BXmult} \cdot \mathrm{RND1}_n) \cdot \sqrt{v(t)}, \tag{1.66}$$

kde BXadd a BXmult jsou opět parametry modelu. Rychlost $v(t)$ určujeme na základě rychlostí aktuálního a následujícího vozidla,

$$v(t) = \begin{cases} v_{n-1}(t) & \text{pro} \quad v_n(t) > v_{n-1}(t), \\ v_n(t) & \text{pro} \quad v_n(t) \leq v_{n-1}(t). \end{cases} \tag{1.67}$$

Maximální vzdálenost při které bere vozidlo na zřetel vozidlo před sebou se značí SDX. SDX se pohybuje mezi 1,5 až 2,5násoobkem hodnoty ABX a je definována následovně:

$$\mathrm{SDX} = \mathrm{AX} + \mathrm{EX} \cdot \mathrm{BX}, \tag{1.68}$$

kde EX stejně jako u ostatní členů je roven

$$\mathrm{EX} = \mathrm{EXadd} + \mathrm{EXmult} \cdot (\mathrm{NRND} - \mathrm{RND2}_n). \tag{1.69}$$

Parametry EXadd a EXmult se používají pro zpřesnění výpočtů, hodnota NRND je generována z normálního rozdělení a $\mathrm{RND2}_n$ značí parametr závislý na řidiči generovaný také z normálního rozdělení.

Další důležitou křivku určující práh, kdy řidič zjistí, že se přibližuje pomalejšímu vozidlu před sebou, značíme SDV. Vypočítá se vztahem:

$$SDV = \left(\frac{\Delta x - L_{n-1} - AX}{CX} \right)^2, \tag{1.70}$$

kde CX zavedeme jako

$$\mathrm{CX} = \mathrm{CXconst} \cdot (\mathrm{CXadd} + \mathrm{CXmult} \cdot (\mathrm{RND2}_n + \mathrm{RND2}_n)), \tag{1.71}$$

Parametry CXconst, CXadd a CXmult jsou zde opět pro zpřesnění výpočtů při určování režimu.

Pokud se jedná o klesající rozdíl rychlostí CLDV, v [12] je uvedeno, že v článku z roku 1992 od Wiedemanna a Reitera je křivka CLDV totožná s křivkou SDV. Jak již naznačil obrázek 1.6, v našem případě uvažujeme ve shodě s [12] různé průběhy.

Při zvyšujícím se rozdílu rychlostí, OPDV, řidič zjišťuje, že jede rychlostí nižší než vozidlo před ním. Práh OPDV se vypočítá jako součin SDV a parametrů modelu OPDVadd, OPDVmult a NRND, což je náhodné číslo z normálního rozdělení,

$$\begin{aligned} \mathrm{OPDV} = \mathrm{CLDV} \cdot (-\mathrm{OPDVadd} - \mathrm{OPDVmult} \cdot \mathrm{NRND}) = \\ \mathrm{SDV} \cdot (-\mathrm{OPDVadd} - \mathrm{OPDVmult} \cdot \mathrm{NRND}). \end{aligned} \tag{1.72}$$

Na základě těchto prahů stanovujeme příslušný režim Wiedemannova modelu. Tyto režimy jsou taktéž zobrazeny na obrázku 1.6.

Režim následování: Hodnota zrychlení vozidla je vždy počítána každou iteraci znovu od nuly. Důvodem je nepřesné zacházení s plynovým pedálem ve vozidle. Pokud vozidlo překoná práh SDV či ABX, jeho hodnota akcelerace bude rovna $-b_{null}$. Naopak při dosáhnutí prahů OPDV či SDX bereme kladnou hodnotu b_{null}. Akcelerace či decelerace je definována jako

$$b_{null} = \text{BNULLmult} \cdot (\text{RND4}_n + \text{NRND}), \tag{1.73}$$

kde BNULLmult je parametr modelu, hodnota NRND je generována z normálního rozdělení a RND4_n značí parametr závislý na řidiči; je generován také z normálního rozdělení.

Režim plné rychlosti: Vozidlo je umístěno nad všemi prahy v obrázku 1.6 a jede nezávisle na okolní dopravě. Vozidlo jede s maximálním zrychlením tak, aby dosáhlo své preferované rychlosti. Nepřesné zacházení s plynem ve vozidle je modelováno přiřazením akcelerace b_{null} či $-b_{null}$. Maximální zrychlení je definováno jako

$$b_{\max} = \text{BMAXmult} \cdot (v_{\max} - v \cdot \text{FaktorV}), \tag{1.74}$$

kde FaktorV zavedeme jako

$$\text{FaktorV} = \frac{v_{\max}}{v_{des} + \text{FAKTORVmult} \cdot (v_{\max} - v_{des})}, \tag{1.75}$$

kde $v_{\max}$ je maximální rychlost vozidla a FAKTORVmult parametr modelu.

Režim přiblížení: Pokud přesáhne vozidlo práh SDV, řidič zjišťuje, že se přibližuje pomalejšímu vozidlu. Řidič tedy začne zpomalovat, aby zabránil kolizi. Decelerace se vypočítá vztahem

$$b_n = \frac{1}{2} \frac{(\Delta v)^2}{\text{ABX} - (\Delta x - L_{n-1})} + b_{n-1}, \tag{1.76}$$

kde b_{n-1} je decelerace následujícího vozidla.

Nouzový režim: Jestliže vzdálenost čela aktuálního vozidla s nárazníkem vozidla před aktuálním je menší než ABX, decelerace se vypočítá následovně

$$b_n = \frac{1}{2} \frac{(\Delta v)^2}{\text{AX} - (\Delta x - L_{n-1})} + b_{n-1} + b_{\min} \frac{\text{ABX} - (\Delta x - L_{n-1})}{\text{BX}}, \tag{1.77}$$

kde $b_{\min}$ je maximální decelarace vozidla a zavedeme ji jako

$$b_{\min} = -\text{BMINadd} - \text{BMINmult} \cdot \text{RND3}_n + \text{BMINmult} \cdot v_n(t), \tag{1.78}$$

kde BMINadd a BMINmult jsou parametry modelu a RND3_n značí parametr závislý na řidiči generovaný z normálního rozdělení.

1.7 Shrnutí

Časově spojité mikroskopické modely jsou v zásadě definovány funkcí zrychlení. V nejstarších modelech, tj. modelech General Motors, může být zrychlení $a(t + \Delta t)$ vozidla n zapsáno jako

$$a(t + \Delta t) = \frac{-\lambda(v_n(t))^m \Delta v_n(t)}{s_n(t)^l}, \tag{1.79}$$

kde λ, l a m jsou parametry modelu, s je odstup mezi vozidly a $\Delta v_n(t)$ rozdíl rychlostí aktuálního vozidla a vozidla jedoucího před aktuálním.

Kromě toho může zrychlení může také záviset na vlastní rychlosti vozidla $v_n(t)$ a klesá se vzdáleností $s_n(t)$ k vozidlu před ním,

$$s_n(t) = x_{n-1}(t) - x_n(t) - l_n, \tag{1.80}$$

kde l_n je délka vozidla n.

V závislosti na rovnici (1.79), kdy zrychlení závisí na vedoucím vozidle, tyto modely není možné použít při velmi nízkých hustotách provozu. Pokud není určeno vedoucí vozidlo ($s_n \to \infty$), zrychlení buďto není možné stanovit (pro hodnotu $l = 0$) nebo je rovné nule ($l > 0$) bez ohledu na rychlost vozidla. Dá se předpokládat, že v tuto chvíli se vozidla budou snažit dosáhnout své preferované rychlosti. Chování vozidel v hustém provozu je ovšem poněkud nerealistické. Především odstup mezi vozidly s týkající se vedoucího vozidla nemusí nutně směřovat k rovnovážnému stavu. Ani malé mezery nepřimějí vozidlo k brždění pokud je rozdíl rychlostí Δv_n roven nule.

Tyto problémy byly vyřešeny v Newellově modelu (viz podkapitola 1.1), kde je z rovnice (1.10) vidět, že zrychlení není závislé pouze na rozdílů rychlostí. Newellův model je nekolizní, ale okamžitá závislost na rychlosti na hustotě provozu vede k velmi vysokým hodnotám zrychlení.[24]

Další model, který obsahuje realistické brzdné reakce řidičů, je Gippsův model, podkapitola 1.2 a Kraussův model, podkapitola 1.3. Navzdory své jednoduchosti, tyto modely realisticky popisují chování řidiče a jsou nekolizní. Bohužel u těchto modelů není možné nikdy dosáhnout stacionárního stavu a to z důvodu vybírání minima z několika hodnot (jedna hodnota by k němu směřovala, ale na základě volby ze dvou (Gipps) či tří (Krauss) možností se vybere jiná). V těchto případech potom není možné simulovat nestabilní stavy dopravy.

Tento problém se snaží odstanit IDM model, podkapitola 1.5, který rychlost určuje na základě zrychlení volného proudu a interakčního zrychlení, kde velikost interakčního zrychlení stoupá s klesající mezerou mezi aktuálním vozidlem a vozidlem jedoucím před aktuálním.

Kromě Newellova modelu a dalších modelů pro základní výzkum existují také vysoce komplexní modely jako Wiedemannův, podkapitola 1.6. Bohužel tyto modely mají mnoho parametrů.

2. Simulační a podpůrný software

Po předchozím uvedení modelů nastává otázka dalšího postupu. Nabízí se následující možnosti:

- pokračovat v aplikaci, kterou autor vytvořil pro svoji bakalářskou práci,
- využití softwaru AIMSUN,
- otevřený simulační software SUMO.

Po vyhodnocení všech hledisek se autor rozhodl zvolit platformu softwaru SUMO. Autor si je vědom toho, že software SUMO je pouze podpůrným nástrojem pro jeho práci, nicméně pro pochopení některých dílčích problémů, které jsou zde řešeny, je nutné zde vysvětlit proto i některé činnosti vlastního fungování tohoto softwaru.

V této kapitole rozebereme simulační software SUMO[1] a další, ať již autorem naprogramované či volně dostupné nástroje z nichž bude patrné proč autor vybral právě tento postup.

2.1 SUMO

SUMO je open-source nástroj pro dopravní simulace vyvíjený centrem DLR[2] v oddělení institutu pro dopravní systémy v Německu. Ve výzkumu se dá SUMO využít v mnoha oblastech např. vyhledání nejkratší cesty[3], možnost zkoušení signálních plánů nebo simulace komunikace mezi vozidly. SUMO bylo a je využíváno na mnoha zajímavých projektech[4].

Od roku 2001 se SUMO postupně rozšiřovalo a nyní je z něj již soubor několika programů. Umožňuje na základě dopravních měření a O/D matic využívat různé routovací algoritmy, simulování křižovatek. Obsahuje také rozhraní TraCI pro online simulaci.

Existují některé další volně dostupné simulační softwary, které se však dále nerozvíjely a přestaly být dříve či později podporovány. Toto je u SUMO vysoce nepravděpodobné z důvodu vývoje v rámci DLR.

Při vývoji SUMO existovaly dva hlavní důvody, proč by měl být otevřeným softwarem. První z nich je snaha podporovat společenství, které se zabývá simulacemi, volně dostupným nástrojem, ve kterém již budou zahrnuty základní algoritmy. Druhým důvodem je fakt, že pro dopravní simulace neexistuje na trhu nástroj, jenž by umožnil jednoduše porovnávat různé dopravní modely (ať už jde o modely sledu vozidel, předjížděcí modely, či řízení dopravy).

[1] Simulation of Urban MObility

[2] Deutsches Zentrum für Luft- und Raumfahrt - Německé středisko pro letectví a kosmonautiku

[3] či jiného kritéria

[4] např. předpověď dopravních proudů v době MS v Německu v roce 2006 v Kolíně nad Rýnem, využití komunitou V2X pro zjišťování dráhy vozidla atd.

Tento druhý důvod a také možnost vlastní implementace modelů vedly autora k volbě SUMO. K tomu, aby bylo možné vlastní modely vůbec implementovat, bylo třeba použít vývojovou verzi simulátoru. To znamená, že nestačí pouze verze SUMO, které obsahuje spustitelné soubory, ale potřebujeme přímo zdrojové kódy. Více o tom, jak SUMO zkompilovat, je uvedeno v příloze B.

2.1.1 Jak to funguje

Po zjištění, čeho všeho je SUMO přibližně schopno, je k tomu, aby byla spuštěna simulace, mít zapotřebí připraveny minimálně tři typy souborů, a to:

- konfigurační soubor,
- soubor s podkladem (dopravní síť),
- soubor s vozidly,
- další soubory.

2.1.2 Konfigurační soubor

V konfiguračním souboru je možné nastavit všechny možnosti, které se bez tohoto souboru musí zapsat při spuštění SUMO jako parametry příkazové řádky. To lze využít, když se SUMO spouští s mnoha parametry.

2.1.3 Soubor s podkladem

Pro diplomovou práci obdržel autor data, a to konkrétně intenzity a průměrné rychlosti z mýtných bran Silničního okruhu kolem Prahy (SOKP, Pražského okruhu). Bylo proto vhodné mít jako podklad pro simulaci síť v konkrétních úsecích mezi mýtnými branami. Pro mapu oblasti bylo využito volně dostupných map ze serveru OpenStreetMap, kde se konkrétní úsek stáhnul ve formátu OSM a vymazaly se všechny prvky nesouvisející s Pražským okruhem. Posléze byl použit jeden z nástrojů SUMO a to konkrétně *Netconvert*, který převádí data z předem definovaného formátu do SUMO formátu. Více k tomuto tématu v příloze C.

SUMO navíc také umožňuje využít vlastní nástroj na generování sítí, *JTRrouter* či *DFrouter*.

2.1.4 Vozidla pro simulaci

Vozidla mají v SUMO několik parametrů. Mezi základní patří

- id vozidla,
- typ vozidla,
- dráha, po které se bude vozidlo pohybovat,

- čas vjezdu do oblasti,
- místo vjezdu do oblasti,
- rychlost při vjezdu do oblasti,
- jízdní pruh při vjezdu do oblasti.

Můžeme tedy na začátku definovat typ vozidla, což znamená jeho parametry jako např. délka vozidla, zrychlení, max. rychlost atp. Poté u každého vozidla nastavíme tento typ a vozidlo automaticky dostane všechny přiřazené parametry.

Dráha, po které se bude vozidlo pohybovat, souvisí s podkapitolou 2.1.3, kde přímo tyto jízdní úseky vytváříme. Tento parametr může obsahovat více hodnot (např. jednotlivé úseky, po kterých se dané vozidlo bude pohybovat), v případě Pražského okruhu minimálně část mezi dvěma mýtnými branami.

2.1.5 Další soubory

Další soubor, který vytvoříme, se týká informací o indukčních smyčkách. Definujeme zde název výstupního souboru z daty, jejich polohu a časový úsek, ve kterém má proběhnout sběr dat. Po proběhnutí simulace budou data z těchto smyček uložena do souboru ve složce, ze které spouštíme simulaci. Indukční smyčka dokáže získat informaci o počtu vozidel za určený časový úsek, jejich průměrné rychlosti, obsazenosti a dalších parametrů. V našem případě nastavíme časový úsek sběru dat stejný jako v reálných datech, tzn. 1 minuta.

2.2 Příprava a vyhodnocení simulace – program

Aby bylo možné dostat se k jádru celé práce, je nutná příprava dat pro simulaci a také jejich následné vyhodnocení. K tomuto účelu byla v jazyce JAVA vytvořena konzolová aplikace jménem *AllInOne.jar*. Pro její správnou funkčnost je nutné mít nainstalované rozhraní pro Java aplikace (JRE).

Aplikace umožňuje 4 základní operace,

1. převedení formátu dat (MATLAB) z Pražského okruhu na formát podporovaný softwarem SUMO (XML),
2. vytvoření souboru vozidel pro simulaci (viz podkapitola 2.1.4), neboť SUMO nemá vlastní generátor vozidel,
3. vytvoření výstupů ze simulace (viz dále)

Důvodem pro volbu jazyka JAVA bylo jednak to, že autor s tímto jazykem dokáže pohodlně pracovat (na rozdíl od jazyka Python, jenž je standardním rozšiřovacím jazykem simulátoru) a také existence volně dostupného balíčku pro možnost práce s daty ze softwaru MATLAB [13]. Dále se autor rozhodl, že vytvoří raději jednu přehlednou aplikaci, než 5 malých.

2.2.1 Třída AllInOne

Třída AllInOne obsahuje dělení programu na jednotlivé podčásti. Ty pak vykonají příslušnou úlohu.

V hlavní třídě se nachází konstruktor, který na základě vstupních parametrů vytvoří objekt příslusné třídy, případně vytiskne chybovou hlášku s problémem.

2.2.2 Převedení formátu dat - třída MatlabToXML

Nejprve je nutné data Pražského okruhu převést do stejného formátu, který nám potom budou dávat indukční smyčky. Naprogramovaná třída, která toto v Javě umí, je nazvána *MatlabToXML*. V konstruktoru se vytvoří soubor, jehož název je jeden ze vstupních parametrů a poté ve funkci *createLoopDocument* se přímo vytváří jeho obsah.

- *public MatlabToXML(String fileNameMatlab, String fileNameCars)* – první vstupní parametr určuje soubor ve formátu .mat, který má předem definovanou strukturu, která je shodná s mýtnými branami z Pražského okruhu. Výstupní soubory se potom budou jmenovat na základě vstupní proměnné *fileNameCars*, kde dále bude následovat potržítko a číslo pruhu[5], ve kterém budou informace o rychlostech a intenzitách pouze pro ten daný pruh.
- *public Document createLoopDocument(int lane)* – vstupní parametr *lane* určuje pruh, pro který se právě obsah souboru vytváří.

Výstupní soubor bude shodný s výstupním souborem po simulaci SUMO, takže bude možné přesně jednotlivé hodnoty z mýtných bran porovnat s nasimulovanými.

Příkaz, kterým lze program spustit aby vytvořil tento dokument, je následující

```
1 java -jar AllInOne.jar -m sokp-0187-20121031.mat det0187.xml
```

Ze souboru *sokp-0187-20121031.mat* vytvoří v závislosti na počtu pruhů pod touto mýtnou branou stejný počet souborů, tedy *det0187_0.xml*, *det0187_1.xml* atd.

Tyto soubory lze následně použít jako vstup pro vytvoření souboru s informacemi o vjezdech vozidel pro SUMO a nebo pro vytvoření grafů nebo histogramů pro porovnání výsledků se simulací.

2.2.3 Vytvoření souboru vozidel pro simulaci - třída XMLCreator

Pro simulaci v SUMO je možné vytvořit si ručně jakýkoliv soubor vozidel. SUMO obsahuje přímo manuál, jak tento soubor vytvořit [23]. Není ovšem vhodné vytvářet ho ručně, nýbrž použít nějaký nástroj pro XML.

Další funkcí, kterou tedy aplikace disponuje, je vytvoření souboru s informací o vjíždějících vozidlech pro simulaci. Tento soubor je plně kompatibilní se vstupními soubory pro SUMO. Vstupním souborem pro generování vozidel bude opět soubor s příponou

[5] Např. pro vstupní parametr detektorkm0201.xml, budou výstupní soubory `detektorkm0201_0.xml` a `detektorkm0201_1.xml`

`.mat`, výstupním soubourem potom jeden soubor obsahující za sebou seřazená vozidla dle vjezdu do oblasti s rychlostí vygenerovanou z Gaussova rozdělení. Další parametry jsou předem dané či generované v aplikaci a určují se na základě volby modelu, který lze nastavit z příkazové řádky. Každému vozidlu je totiž nutné přiřadit model sledu vozidel, podle kterého se má v simulaci pohybovat.

Jediný problém nastává ve chvíli, kdy musíme jednotlivým vozidlům nastavit jejich cestu v simulaci. Pokud všechna vozidla pojedou po stejné dráze, stačí, když se každému vozidlu přiřadí cesta, která je shodná s názvy hran podkladu (viz podkapitola 2.1.3 a C.1 poslední odstavec).

Pro naší simulaci cestu pro vozidlo předem připravíme tak, aby všechna vozidla projela úsekem, který chceme simulovat. V konzolové aplikaci v tomto směru není nutné nic dalšího nastavovat.

Třída *XMLCreator* obsahuje následující konstruktor a metody

- *public XMLCreator(String fileNameMatlab, String fileNameCars, String model, String type)* – konstruktor, který přijímá název datového souboru, ze kterého budeme vozidla generovat, název výstupního souboru a způsob generování času vstupu vozidla do oblasti,
- *public Document createRouteEqualDocument(String model)* – naplnění souboru při konstantní časové vzdálenosti vozidel,
- *public Document createRouteExpDocument(String model)* – naplnění souboru při generování vozidel z exponenciálního rozdělení,
- *public Element createVType(Element element, int actualCar, String model, int iTime, int lane)* – vytvoření parametrů pro konkrétní vozidlo *actualCar*, nastavení pruhu *lane* a času *iTime*, ve kterém vjíždí do simulace v závislosti na modelu *model*,
- *public Element createVehicle(Element element, int actualCar, double time, int iTime, int lane, String type)* – vytvoření jednoho exempláře vozidla s příslušnými parametry, *actualCar* je pořadí vozidla od začátku simulace, *lane* pruh do kterého bude vozidlo vygenerováno, *time* je čas ve kterém vozidlo bude vpuštěno do simulace.

Lze přidat konkrétní model, třetí parametr příkazu, podle kterého chceme aby SUMO simulovalo provoz. Tento parametr je nepovinný a pokud ho nezadáme, program předpokládá volbu Kraussova modelu.

Základními parametry jsou:

- zrychlení a,
- decelerace b,
- reakční čas řidiče T.

Ty je zatím možné nastavit pouze ve zdrojovém kódu programu.
Zatím jsou k dispozici čtyři modely a k nim příslušné parametry:

- Gippsův model tak, jak je popsán v sekci 1.2
  - desiredSpeed (V_n): preferovaná rychlost daného vozidla, rychlost je generovaná z normálního rozdělení se střední hodnotou 130 km/h, což je maximální rychlost na rychlostní komunikaci a směrodatnou odchylkou 10% z maximální rychlosti, která byla stanovena na základě expertního odhadu.

```
1       java -jar AllInOne.jar -c sokp-0201-20121001.mat vozidla.rou.xml Gipps
```

- Kraussův model a Rozšířený Kraussův model, tak jak je popsán v sekci 1.3, resp. 1.4
  - η: náhodná výchylka od běžného řízení, hodnota 0,5 - předem nastavena v rámci SUMO

```
1       java -jar AllInOne.jar -c sokp-0201-20121001.mat vozidla.rou.xml Krauss
```

```
1       java -jar AllInOne.jar -c sokp-0201-20121001.mat vozidla.rou.xml
            KraussOrig
```

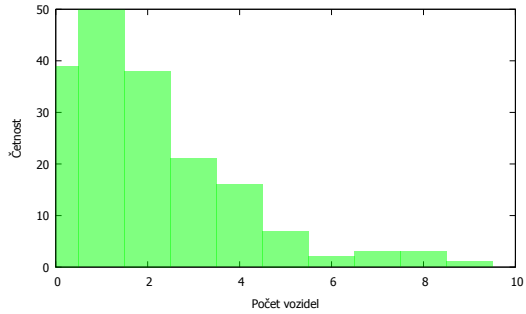
- IDM, Intelligent Driver Model ze sekce 1.5
  - delta (δ): exponent akcelerace dle rovnice (1.60) - hodnota 4, doporučena v rámci kalibrování autorů IDM

```
1       java -jar AllInOne.jar -c sokp-0201-20121001.mat vozidla.rou.xml IDM
```

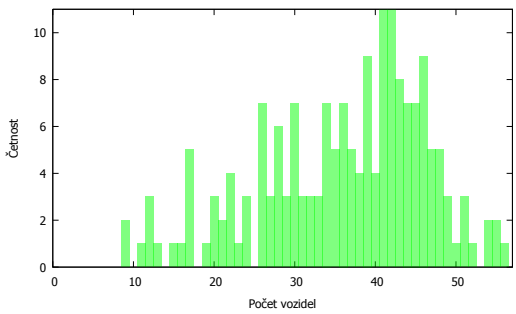
- Wiedemannův model - parametry použity pro výpočet křivek AX, BX atd. z obrázku 1.6
  - estimation - hodnota 0,5
  - security - hodnota 0,5

Výsledný soubor bude mít název `vozidla.rou.xml` – `.rou` je povinná druhá přípona, která pokud není uvedena, SUMO nespustí simulaci.

Implicitně je výchozí generování vozidel deterministické. Pokud nám v jedné minutě projede např. 10 vozidel, tak vozidla budou mít vstupní čas rovnoměrně rozdělen po 1/10. Tato varianta se zde nachází proto, aby bylo možné zkontrolovat správné generování vozidel a chování jednotlivých modelů sledu vozidel. Výsledný příkaz tedy bude:



Obrázek 2.1: Četnosti pro čas od 2:00-5:00



Obrázek 2.2: Četnosti pro čas od 6:00-9:00

```
1 java -jar AllInOne.jar -c sokp-0201-20121001.mat vozidla.rou.xml Gipps
```

Další varianta využívá znalost, že pokud rozdělení počtu vozidel za nějaký čas je Poissonovo, potom odstupy mezi vozidly mají exponenciální rozdělení. Z pohledu na obrázek 2.1 a 2.2[6] lze předpokládat, že se opravdu jedná o Poissonovo rozdělení.

Pokud chceme, aby se nám vozidla generovala z exponenciálního rozdělení, kde se každou minutu bude měnit jeho parametr λ (v závislosti na počtu vjíždějících vozidel za minutu), zvolíme možnost spuštění s dodatečným parametrem exp:

```
1 java -jar AllInOne.jar -c sokp-0201-20121001.mat vozidla.rou.xml Krauss exp
```

Pokud vynecháme označení modelu sledu vozidel, bude aplikace implicitně uvažovat standardní Kraussův model. Aplikace AllInOne.jar, zavolaná jako

```
1 java -jar AllInOne.jar -c sokp-0201-20121001.mat vozidla.rou.xml Gipps
```

generuje vozidla, která se budou pohybovat dle Gippsova modelu a budou mít deterministický časový odstup. Je možné vynechat i model, tz.

```
1 java -jar AllInOne.jar -c sokp-0201-20121001.mat vozidla.rou.xml exp
```

generuje vozidla z exponenciálního rozdělení. Pohyb bude na základě Kraussova modelu.

Soubor `vozidla.rou.xml` lze rovnou použít jako vstupní soubor vozidel pro simulaci v SUMO.

2.2.4 Výstupy ze simulací

Po simulaci SUMO vytvoří několik souborů. Hlavní soubor v XML formátu, jehož název určujeme při spouštění simulace, obsahuje informace o vozidlech, která se nacházejí v simulaci každou sekundu. Pro naše výstupy budou ovšem nejdůležitější soubory, dávající data z indukčních smyček (viz 2.1.5). Tento soubor se bude zpracovávat stejně jako soubor vzniklý pomocí aplikace *AllInOne* uvedeného v odstavci 2.2.2.

[6] vytvořeny také tímto programem

Pro porovnání reálných a nasimulovaných dat nabízí aplikace *AllInOne.jar* tři možnosti vyhodnocení:

- časová řada počtu vozidel každou minutu,
- počet vozidel za časový úsek, jehož délku je možné nastavit,
- histogram četností vozidel; v tomto případě máme možnost nastavit nastavení časového intervalu, ve kterém nás četnosti zajímají.

2.2.5 Časová řada – třída XMLReader

Jednou z možností, jak zobrazit výstupy ze simulace, je pomocí časové řady. Program vyhodnotí všechny vstupní soubory a předpřipraví soubor `gnuplot.txt` pomocí kterého je potom možné vykreslit tuto řadu za pomoci programu pro vykreslování grafů GNUplot [10]. Program vytvoří tento soubor pro každý pruh zvlášť a poté souhrný soubor, který obsahuje každý pruh a součet vozidel v jednom grafu.

Třída *XMLReader* obsahuje následující konstruktor a metody:

- *public XMLReader(String fileNameInput, String fileNameOutput) throws DocumentException* - první parametr je vstupní soubor ve formátu XML (buď to vytvořený programem z Matlab souboru nebo ze simulace SUMO), druhý parametr je potom název souboru, který je při instalovaném GNUplotu vygenerován automaticky,
- *public static void timeSeries(String fileNameInput,String fileNameOutput,int numberOfFiles) throws DocumentException* – stejné parametry jako u konstruktoru + navíc počet souborů, které je nutné zpracovat,
- *private static Calendar setStart(Calendar cal)* – nastavení časové řady od půlnoci do půlnoci z důvodu větší přehlednosti programu,
- *public static Document parse(String fileName) throws DocumentException* – rozdělení XML dokumentu na jednotlivé části

Je potřeba si uvědomit, že zadáváme jeden vstupní soubor, ale vstup je vlastně několik souborů zároveň. Např. příkaz

```
1 java -jar AllInOne.jar -x detektor.xml graf.pdf
```

prochází aktuální složku a kontroluje všechny XML soubory mající tvar `detektor_0.xml`, `detektor_1.xml` atd. Poté vytvoří pomocné soubory, které obsahují sadu příkazů pro vykreslení grafu GNUplotem. Pomocné textové soubory jsou vždy pro každý pruh dva, a to `pocetvozidelpruh_1.txt` a `gnuplot_1.txt` pro 1. pruh atd. Hlavní soubor, kde jsou obsaženy všechny pruhy, má název `gnuplot.txt` a ke svému spuštění potřebuje `pocetvozidelpruh.txt`.

2.2.6 Počet vozidel za daný čas – třída XMLReaderHist

Další variantou je vykreslení histogramu počtu vozidel za čas. Je možné vykreslit počet vozidel za minutu, kde výsledek bude stejný jako časová řada s tím rozdílem, že grafem bude histogram. Program tudíž umí nastavit i jinou časovou délku úseku, např. počet vozidel za hodinu.

Třída *XMLReaderHist* obsahuje následující konstruktor a metody

- *public XMLReaderHist(String fileNameInput, String fileNameOutput, String w) throws DocumentException* – stejná situace jako u XMLReaderu (viz odstavec 2.2.5), poslední parametr je potom délka úseku, za který chceme počítat vozidla
- *public static void hist(String fileNameInput, String fileNameOutput, int numberOfFiles, String w) throws DocumentException* – úplně stejná funkce jako u XMLReaderu doplněna o časový interval

Všechny ostatní funkce, které třída obsahuje, již byly vysvětleny v předcházejících třídách.

Pokud chceme histogram počtu vozidel během celého dne, necháme poslední parametr volání prázdný, tedy

```
1 java -jar AllInOne.jar -xh detektor.xml histogram.pdf
```

Další možností je nastavit si délku časového úseku. Tato délka musí být dělitelná počtem minut dne (1440) a udává se v minutách. Jinak program argument ignoruje a vytvoří histogram, jako kdyby žádný parametr nebyl zadán. Příkaz

```
1 java -jar AllInOne.jar -xh detektor.xml histogram.pdf 60
```

tedy vytvoří histogram, kde se sčítají vozidla za hodinu.

2.2.7 Histogram - třída Histogram

Poslední možností analýzy výstupu je histogram četností počtu vozidel.[7] Tento histogram můžeme udělat jak pro celý den, tak i pro určitou časovou oblast (např. ranní špička). Pokud nejsou zadány parametry od a do, histogram se vytvoří pro celý den. Je možné také vytvořit histogram přes půlnoc, tzn. od 22:00 do 3:00.

Třída Histogram obsahuje následující konstruktor a metody:

- *public Histogram(String fileNameInput, String fileNameOutput, String from, String to) throws DocumentException* – opět stejné parametry jako v předchozích třídách, tz. název vstupního a výstupního souboru, který bude vytvořen po spuštění GNUplotu, a dále časové hodnoty od kdy a do kdy chceme histogram vykreslit,
- *public static void hist(String fileNameInput, String fileNameOutput, int numberOfFiles, String from, String to) throws DocumentException* – vytvoření souborů pro GNUplot, všechny parametry jsou stejné jako u předchozích tříd.

[7] Obrázky 2.1 a 2.2 vznikly právě díky tomuto modulu.

Pokud chceme vytvořit histogram počtu vozidel za celý den, příkaz vypadá následovně

```
java -jar AllInOne.jar -h detektor.xml histogram.pdf
```

Pokud nás zajímá pouze určitý časový úsek, např. ranní špička, na konec příkazu zadáme příslušné meze

```
java -jar AllInOne.jar -h detektor.xml histogram.pdf 2:00 5:00
```

pro vytvoření histogramů počtu vozidel od 2 do 5 ráno.

2.3 Shrnutí práce s programem

Program byl vytvořen v jazyce JAVA jako podpůrný prostředek pro vytvoření jak vstupů, tak výstupů pro program SUMO při daném typu vstupních dat z Pražského okruhu. Program se spouští s příkazové řádky příkazem

```
java -jar AllInOne.jar
```

a dále zadáním příslušných parametrů.

Program se dělí na pět hlavních částí a pro každou z nich je uveden příklad pro spuštění:

- vytvoření vstupního souboru vozidel pro SUMO
  - pravidelné rozložení - např. IDM model

    ```
    java -jar AllInOne.jar -c sokp-0201-20121001.mat vozidla.rou.xml IDM
    ```

  - z exponenciálního rozdělení - Kraussův model (není nutno uvádět)

    ```
    java -jar AllInOne.jar -c sokp-0201-20121001.mat vozidla.rou.xml exp
    ```

  - z exponenciálního rozdělení - Wiedemannův model

    ```
    java -jar AllInOne.jar -c sokp-0201-20121001.mat vozidla.rou.xml
        Wiedemann exp
    ```

- převedení formátu dat z Matlabu do formátu SUMO

  ```
  java -jar AllInOne.jar -m sokp-0187-20121031.mat det0187.xml
  ```

- vytvoření grafu časové řady

  ```
  java -jar AllInOne.jar -x detektor.xml graf.pdf
  ```

- vytvoření histogramu počtu vozidel za časový úsek – v tomto případě 1 hodina

  ```
  java -jar AllInOne.jar -xh detektor.xml histogram.pdf 60
  ```

- vytvoření histogramu četností vozidel – v tomto případě od 2 do 5 hodin ráno

```
1 java -jar AllInOne.jar -xh detektor.xml histogram.pdf 2:00 5:00
```

3. Implementace nového modelu

V simulačním nástroji SUMO nejsou některé z modelů, uvedených v kapitole 1, implementovány. V této kapitole proto uvádím obecné informace, které s implementací nového modelu souvisí bez ohledu na to, o jaký model se konkrétně jedná.

Než začneme, je nutné mít připravené soubory pro nový model, pro Gippsův model jsou to soubory `MSCFModel_Gipps.cpp` a `MSCFModel_Gipps.h`. Postup, jak nové soubory vytvořit a implementovat do SUMO je uveden v příloze D.

3.1 MSCFModel.h

Ve složce `/src/microsim/cfmodels` v souboru `MSCFModel.h` (ne `MSCFModel_Gipps.h`) se nacházejí definice základních funkcí, které posléze v přidaném modelu implementujeme. Některé z nich přetížíme a poté použijeme u našeho implementovaného modelu. Ty, které nepoužijeme, zůstanou stejné a při použití našeho modelu během simulace se budou volat funkce implementované v `MSCFModel.cpp`, případně `MSCFModel.h`.

Při procházení zdrojových kódů je na první pohled vidět, že většina obecných funkcí je primárně vytvořena pro Kraussův model. Zbytek funkcí počítá nějakou obecnou vlastnost, která s modelem přímo nesouvisí. Tyto funkce nemají u své definice klíčové slovo virtual, které značí, že tuto funkci není možné nahradit při vlastní implementaci nového modelu. Dále jsou zde proměnné, které jsou při vytvoření nového potomka obecného modelu automaticky přístupné (např. zrychlení, reakční čas řidiče). **Tyto proměnné se používají i při jinde, nejen při výpočtech v třídě nového modelu, jako je např. vjezd vozidla do oblasti. Je proto nutné dbát na to, abychom do nich v implementaci nového modelu neuložili špatnou hodnotu (např. decelerace).**

Metody třídy *MSCFModel* obsahují většinou následující vstupní proměnné:

- *const MSVehicle* const veh* – ukazatel na aktuální vozidlo, přes něj je možné zjistit detailnější informace o vozidle
- *SUMOReal speed* – rychlost aktuálního vozidla
- *SUMOReal gap2pred* – vzdálenost mezi aktuálním a vedoucím vozidlem
- *SUMOReal predSpeed* nebo *SUMOReal vL* – rychlost vedoucího vozidla
- *SUMOReal predMaxDecel* – maximální decelerace vedoucího vozidla

Ostatní vlastnosti vedoucího vozidla nejsme schopni jednoduše zjistit, nicméně rychlost a vzdálenost by při implementaci nových modelů měly plně postačit.

Seznam metod, které je při implementaci nového modelu sledu vozidel přetížit, je následující:

- *virtual SUMOReal freeSpeed(const MSVehicle* const veh, SUMOReal speed, SUMOReal seen, SUMOReal maxSpeed, const bool onInsertion = false) const* – výpočet

rychlosti vozidla, které jede jako první, tzn. jeho poloha je nejdále od začátku simulace a nejede před ním žádné vozidlo,

- *virtual SUMOReal followSpeed(const MSVehicle* const veh, SUMOReal speed, SUMOReal gap2pred, SUMOReal predSpeed, SUMOReal predMaxDecel) const* – výpočet rychlosti vozidla na základě parametrů pohybu vedoucího vozidla,
- *virtual SUMOReal insertationFollowSpeed(const MSVehicle* const veh, SUMOReal speed, SUMOReal gap2pred, SUMOReal predSpeed, SUMOReal predMaxDecel) const* – výpočet rychlosti vozidla při vjezdu do oblasti na základě parametrů vozidla, které se nachází před ním; **SUMO nenechá vjet vozidlo do oblasti, pokud není zajištěna bezpečná vzdálenost pro zabrždění vozidla**,
- *virtual SUMOReal stopSpeed(const MSVehicle* const veh, const SUMOReal speed, SUMOReal gap2pred)* – výpočet rychlosti ve chvíli, kdy se před vozidlem nachází statický objekt[1],
- *virtual SUMOReal interactionGap(const MSVehicle* const veh, SUMOReal vL) const* – výpočet mezery, kdy ovlivňuje vedoucí vozidlo aktuální

Můžeme také využít další podpůrné funkce, které přímo nezávisí na modelu. Dají se použít při např. při kontrole bezpečné rychlosti. Tyto funkce nelze primárně přepsat v novém modelu. Jsou to:

- *SUMOReal maximumSafeFollowSpeed(SUMOReal gap, SUMOReal predSpeed, SUMOReal predMaxDecel) const* – výpočet maximální bezpečné rychlosti pro aktuální vozidlo na základě vedoucího
- *SUMOReal maximumSafeStopSpeed(SUMOReal gap) const* – výpočet maximální bezpečné rychlosti pro zastavení

3.2 Implementace Gippsova modelu

K tomu abychom mohli přidat do SUMO Gippsův model, je nejprve nezbytné vytvořit soubory pro nový model – viz příloha D. Spustíme kompilaci SUMO a pokud vše proběhne bez problémů můžeme postoupit dále. Jinak je ovšem chyba v nedodržení postupu v příloze a není možné postupovat dále.

Ve chvíli, kdy se nám SUMO zkompilovalo a máme k dispozici soubory `MSCFModel_Gipps.cpp` a `MSCFModel_Gipps.h`, můžeme začít programovat Gippsův model. V příloze D v sekci D.2 je k dispozici návod na vytvoření vlastních parametrů, my ho použijeme pouze pro preferovanou rychlost vozidla *desiredSpeed*. Při tvorbě konstruktoru by nám potom měla být k dispozici hodnota proměnné *desiredSpeed*, kterou můžeme přidat jako parametr při vytváření souboru vozidel.

[1] objekt, který má nulovou rychlost

Prvně zakomponujeme do souboru `MSCFModel_Gipps.h` všechny naše proměnné, které budou k dispozici našemu modelu. Proměnné deklarujeme jako chráněné, aby k nim měl přístup pouze Gippsův model, případně jeho potomek.

```
protected:
        SUMOReal myDesiredSpeed;
        SUMOReal myGippsDecel;
```

Kromě *myDesiredSpeed* jsme zde přidali ještě jednu proměnnou *myGippsDecel* , která bude udávat decelaraci aktuálního vozidla se záporným znaménkem. Není to nezbytný krok, činíme tak pouze z důvodu zpřehlednění kódu/výpočtu (viz dále).

Nyní již vytvoříme konstruktor, který bude naplní proměnné zděděné od obecného modelu a zároveň naše nově přidané proměnné:

```
MSCFModel_Gipps::MSCFModel_Gipps(const MSVehicleType* vtype, SUMOReal accel,
    SUMOReal decel, SUMOReal headwayTime, SUMOReal desiredSpeed): MSCFModel(
    vtype, accel, decel, headwayTime), myDesiredSpeed(desiredSpeed) {
                myGippsDecel = -decel;
}
```

Tento kód zavolá konstruktor obecného modelu, uloží nám hodnoty zadané při generování vozidel do simulace do jednotlivých proměnných. My si potom přidáme ještě hodnotu *desiredSpeed* do proměnné *myDesiredSpeed* a do proměnné *myGippsDecel* uložíme zápornou hodnotu decelerace.

V hlavičkovém souboru *MSCFModel_Gipps.h* je také nutné upravit definici stávajícího konstruktoru na

```
 MSCFModel_Gipps(const MSVehicleType* vtype, SUMOReal accel, SUMOReal decel,
     SUMOReal headwayTime, SUMOReal desiredSpeed);
```

Vraťme se nyní k proměnné myGippsDecel. Na první pohled se zdá nelogické zavádět novou proměnnou, obsahující zápornou hodnotu decelerace, zadanou do modelu a zdálo by se vhodné, tuto hodnotu uložit do proměnné myDecel. Tím bychom ovšem porušili definici třídy *MSCFModel*, u níž se předpokládá, že myDecel obsahuje kladnou hodnotu decelerace modelu. Na tuto hodnotu se také SUMO během simulace několikrát odkazuje. Zápornou hodnotu decelerace je proto třeba držet odděleně. **Není možné uložit do myDecel hodnotu -decel, protože na hodnotu myDecel se ptá SUMO v různých situacích a byla by v něm uložena záporná hodnota namísto kladné, které SUMO očekává.**

V tuto chvíli bychom měli mít k dispozici následující proměnné v celém těle Gippsova modelu. Uvedeme zde také proměnnou odpovídající parametrům Gippsova modelu - viz odstavec 1.2.2:

- *myAccel* – akcelerace a_n,
- *myGippsDecel* – decelerace b_n,
- *myDesiredSpeed* – preferovaná rychlost V_n,

- *myHeadwayTime* – reakční čas T_n.

Gippsův model pracuje tak, že vybírá minimální hodnotu z dvou možností: z rychlosti ve volném proudu či rychlosti při interakci z vedoucím vozidlem, jak uvádí rovnice 1.21. My si obě části této rovnice naprogramujeme, každá část rovnice bude jedna funkce. V souboru *MSCFModel_Gipps.h* je obě zadefinujeme,

```
private:
 virtual SUMOReal vFree(SUMOReal speed) const;
 virtual SUMOReal vInteraction(SUMOReal speed, SUMOReal predSpeed, SUMOReal
     gap2pred) const;
 virtual SUMOReal chooseBrakeConst(SUMOReal decel) const;
```

parametry těchto funkcí přímo odpovídají tomu, co pro výpočet daného členu potřebujeme. Funkce *vFree* potom vypadá následovně:

```
SUMOReal MSCFModel_Gipps::vFree(SUMOReal speed) const {
 SUMOReal actualSpeed = speed;
 SUMOReal result = actualSpeed + 2.5*myHeadwayTime*myAccel*(1-actualSpeed/
     myDesiredSpeed)*sqrt(0.025 + actualSpeed/myDesiredSpeed);
 return result;
}
```

Funkce přesně kopíruje vzorec (1.12), který je u Gippsova modelu vzorec pro rychlost při volném proudu. Pokud je preferovaná rychlost V_n rovna aktuální rychlosti $v_n(t)$, celý člen za proměnnou *actualSpeed* se vyruší a dostáváme, že rychlost v následujícím kroku bude přímo stejná jako aktuální.

Interakční člen nám dokumentuje funkce *vInteraction* obsahující výpočet rychlosti na základě vedoucího vozidla. Její kód je následující:

```
SUMOReal MSCFModel_Gipps::vInteraction(SUMOReal speed, SUMOReal predSpeed,
    SUMOReal gap2pred) const{
        SUMOReal wholeGap = (gap2pred+myType->getMinGap())*2;
        SUMOReal speeds = speed*myHeadwayTime + ((predSpeed*predSpeed)/(2*
            chooseBrakeConst(myGippsDecel)));
        SUMOReal sqrtPart = sqrt(myGippsDecel*myGippsDecel*myHeadwayTime*
            myHeadwayTime - myGippsDecel*(wholeGap - speeds));
        SUMOReal rV = myGippsDecel*myHeadwayTime + sqrtPart;
        if (rV < 0)
               rV = 0;
        return rV;
 }
```

Pro větší přehlednost jsme kód funkce *vInteraction* rozdělili na více částí. Podstatná je proměnná *myGippsDecel*, kterou jsme na začátku nastavili jako zápornou a proto zde figuruje s kladným znaménkem.

Proměnná *wholeGap* odpovídá části $[x_{n-1}(t) - s_{n-1} - x_n(t)]$. Tu bylo potřeba upravit pro účely SUMO. Nemáme totiž možnost zjistit parametr s_{n-1}, jenž závísí na délce vedoucího vozidla a k této informaci se jednoduše nedostaneme. Proto jsme upravili tento

člen jako součet vzdálenosti mezi vozidly (předek aktuálního vozidla a zadek vedoucího vozidla) a k ní jsme přičetli minimální bezpečnostní odstup. Tím bychom měli dostat stejný výsledek, jaký je uveden v Gippsově modelu.

Další zajímavostí je vypočet členu $\hat{b}$. Pro výpočet decelerace vedoucího vozidla $n-1$ zvolil Gipps na základě následující vztah

$$\hat{b} = \min\left(-3.0, \frac{b_n - 3.0}{2}\right) \mathrm{m/s^{-2}}. \tag{3.1}$$

Tato rovnice přepsaná do funkce se rovná přesně obsahu funkce *chooseBrakeConst* definovanou v části private, viz výše. Její kód je poměrně jednoduchý:

```
1 SUMOReal MSCFModel_Gipps::chooseBrakeConst(SUMOReal decel) const{
2         return MIN2(-3.0, (- decel - 3.0)/2);
3 }
```

Funkce *MIN2* je vlastní funkce SUMO, která hledá minimum ze dvou hodnot.

V tuto chvíli máme připravené obě funkce pro výpočet rychlostí pro vozidla vjíždějící do simulace.

V další fázi tyto funkce využijeme pro výpočty. Všechny následující funkce budeme pouze nahrazovat již stávajícími, které obsahuje *MSCFModel*. Vždycky je otázkou, zda je nutné danou funkci přepisovat a nebo nechat její stávající verzi.

Začneme funkcí *freeSpeed*. Tato funkce nám vypočítává rychlost při volném proudu, tj. ve chvíli, kdy před aktuálním vozidlem nejede žádné další. Mohli bychom tuto funkci nechat stejnou, tak jak je implementována v SUMO, ovšem daleko lepší bude tuto funkci přepsat a to z toho důvodu, že Gippsův model přímo uvažuje o rychlosti při volném proudu. Definice přepisovaných funkcí zůstanou stejné jako v *MSCFModel.h*, pro *freeSpeed* by definice[2] vypadala takto:

```
1 virtual SUMOReal freeSpeed(const MSVehicle* const /* veh */, SUMOReal /* speed
      */, SUMOReal seen, SUMOReal maxSpeed, const bool onInsertion) const;
```

Implementace naší vlastní funkce *freeSpeed* bude následující

```
1 SUMOReal
2 MSCFModel_Gipps::freeSpeed(const MSVehicle* const veh , SUMOReal speed,
      SUMOReal seen, SUMOReal maxSpeed, const bool onInsertion) const {
3         return vFree(speed);
4 }
```

Je vidět, že většina vstupních parametrů funkce není vůbec využita, nicméně musejí zde zůstat z důvodu kompatibility s ostatními modely. Gippsův model při výpočtu tyto parametry ignoruje.

Další funkcí, kterou je nutné v modelu implementovat, je *followSpeed*. Ta počítá rychlost v závislosti na vedoucím vozidle. My již víme, že tato funkce by měla vracet stejný výsledek jako rovnice (1.21). Její implementace vypadá následovně:

[2] Každá definice metody zde zmiňována bude umístěna do souboru `MSCFModel_Gipps.h` do části public. Tato definice je shodná s definicí uvedenou v souboru `MSCFModel.h`.

```
SUMOReal
MSCFModel_Gipps::followSpeed(const MSVehicle* const veh, SUMOReal speed,
    SUMOReal gap, SUMOReal predSpeed, SUMOReal predMaxDecel) const {
        return MIN2(vFree(speed), vInteraction(speed, predSpeed, gap));
}
```

Poslední metodou, kterou musíme ve tříde *MSCFModel_Gipps* vytvořit, je *stopSpeed*. Tato funkce má stejné parametry jako *followSpeed* až na parametr *predSpeed*, který je v tuto chvíli nulový, protože tato funkce počítá rychlost, pokud se před vozidlem nachazí statický objekt. Proto voláme stejné funkce jako u *followSpeed* s tím, že *predSpeed* má hodnotu nula:

```
SUMOReal
MSCFModel_Gipps::stopSpeed(const MSVehicle* const veh, const SUMOReal speed,
    SUMOReal gap) const {
        return MIN2(vFree(speed), vInteraction(speed, 0, gap));
}
```

Tím je nový model implementován.

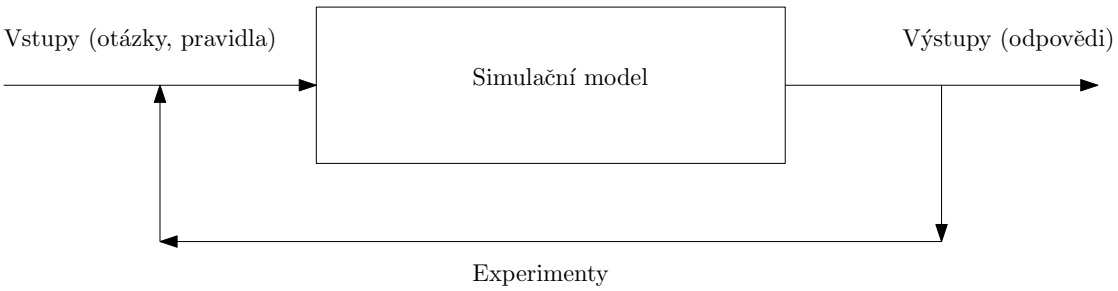
4. Kalibrace modelů

4.1 Stručný úvod do simulačních modelů

Často se stávají situace, že je potřeba vyzkoušet určitou možnost, a není možné zkoumat chování systému ve skutečnosti. Takovým systémem může být v našem případě dálnice a možností uzavření jednoho pruhu při dopravní nehodě. Model, který vytváříme, je jakési zjednodušení skutečnosti se snahou zachování co nejvíce vlastností systému.

V tuto chvíli se simulace jeví jako vhodná varianta jak provést experimentální test. Obecně je totiž daleko lepší možností provést pokus na modelu v počítači než na reálném systému a riskovat jeho poškození či zničení, navíc na některých reálných systémech nelze ani pokus provést. Výsledky tohoto pokusu na počítači mohou poskytnout mnoho užitečných informací, na jejichž základě je např. možné zvolit optimální variantu při rozhodování. Podle tohoto konceptu je tedy simulační model počítač, který je určen k řízení pokusů s modelem systému s cílem sestavení platných výsledků i pro reálný systém.

Simulace je tedy experiment na modelu reálného systému [21]. Za předpokladu, že vývoj modelu systému v čase vhodně napodobuje vývoj reálného systému, můžeme sbírat informace o modelu (např. proměnné) a posléze při použití statistických metod či jiného vhodného matematického aparátu a následných dalších experimentů můžeme díky tomu navrhnout i budoucí chování systému. Typickým příkladem může být liniové řízení na dálnici, kde model neustále zpracovává naměřené parametry dopravního proudu (proměnné) a na jejich základě může změnit rychlost proměnným dopravním značením (budoucí chování systému). Tato myšlenka je znázorněna na obrázku 4.1.



Obrázek 4.1: Simulační model, zdroj:[1]

Spolehlivost procesu rozhodování záleží na schopnosti vytvořit takový simulační model, který reprezentuje chování reálného systému s dostatečnou přesností. Ve chvíli, kdy je již model dostatečně přesný, může být reálný systém nahrazen modelem a na modelu je možné provádět různé experimenty. Proces určující, zda je simulační model dostatečně podobný reálnému systému, se nazývá **validace modelu** – jde o iterativní proces zahrnující **kalibraci parametrů modelu** a porovnání chování modelu s chováním reálného

systému (pokud je to možné). Nesoulady mezi systémem a modelem nutí systémového analytika k zjišťování dalších informací o systému. Tyto informace jsou potom použity k vylepšení modelu. Je nutné si uvědomit, že model by mohl být vylepšován neustále. Je proto nezbytné určit práh, kdy model již dává postačující výsledky, tj. kdy je jeho přesnost akceptována. Tento práh je právě dán validací. V tu chvíli již není nezbytné model dále vylepšovat. Validace simulačního modelu by měla být brána v potaz během celého procesu vytváření modelu.

4.2 Validace modelů

Jako hlavní zdroj informací k validaci modelů je použit text [15]. Problematika validace a verifikace modelů je ještě podrobněji rozebrána například v [2].

Verifikace je v první řadě rozhodnutí, zda simulační program vykonává přesně to, co bylo zamýšleno, to jest odstranění chyb z programu. Verifikace tedy kontroluje „přenesení“ simulačního modelu do správně fungujícího programu.

Verifikace obvykle znamená spuštění programu s implementovaným simulačním modelem s různým nastavením vstupních parametrů a kontrolou, zda jsou výsledky simulace přijatelné. V některých případech je možné některé výpočty provádět analyticky a posléze numerickou simulací výsledky porovnat a tím zjistit přijatelnost výstupů. To bohužel není případ modelů sledu vozidel.

Validace modelu je hotova tehdy, pokud simulační model je přesná reprezentace zkoumaného systému. Pokud je model validní, mohou být rozhodnutí, která byla provedena na základě simulačního modelu stejná, jako kdyby se prováděl experiment na reálném systému.

Model je **věrohodný** tehdy, když jsou jeho výsledky přijaty uživatelem a použity v rozhodovacím procesu.

Při validaci simulačního modelu by analytik neměl zapomenout, že

- simulační model je systém, který ovšem může být pouze aproximací reálného systému, bez ohledu na to, kolik práce bylo vloženo do vývoje modelu,
- simulační model by měl být vždy vyvíjen pro specifické účely,
- simulační model by měl být validován vzhledem k perfomačním indikátorům souvisejících s účely modelu a mohl být použit pro rozhodování
- vývoj a validace modelu by měly probíhat zároveň během celé analýzy systému a vytváření simulačního modelu.

Law and Kelton [15] dále navrhli postup, jak vyvinout validní a věrohodné simulační modely. Dělí se na tři části:

- vytvoření modelu se „zjevnou“ validitou[1],

[1] z anglického face validity

- testování předpokladů modelu empiricky,
- rozhodnutí, jak moc jsou výstupní data ze simulace reprezentativní.

Vytvoření modelu se zjevnou validitou Při vytváření modelu je důležité, aby model prozkoumal odborník, resp. člověk, které rozumí dané problematice. Pokud uzná, že simulační model vypadá racionálně, potom můžeme tvrdit že jde o model se zjevnou validitou.

Aby byl model označen za model se zjevnou validitou je dle [15] nutné, aby

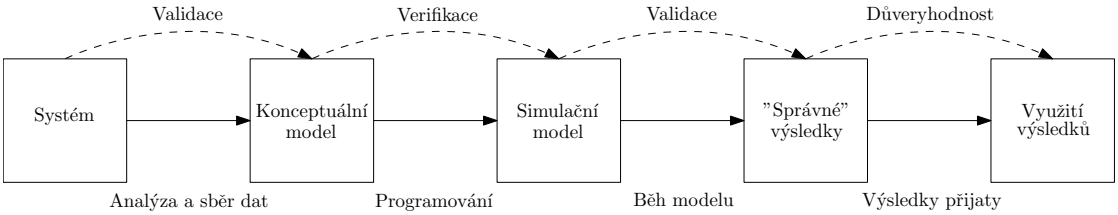
- analytik by měl spolupracovat s lidmi, kteří jsou důvěrně obeznámeni s reálným systémem,
- analytik musí být důsledný při získávání všech požadovaných informací,
- pokud existují historická data, měla by být získána a použita při tvorbě modelu,
- dávat pozor, aby tato data byla správná a znázorňovala to, co je modelováno.

Testování předpokladů modelu empiricky Jestliže je použito pravděpodobností rozdělení jako vstupní parametr simulačního modelu na základě shody s reálnými daty, u výstupu musí být tyto okolnosti zohledněny.

Z dalších okolností musí být zohledněna citlivost na počáteční podmínky. To nastává tehdy, když nepatrná změna vstupních parametrů způsobí signifikatní změnu dat.

Tyto dvě podmínky musí být při vyhodnocení testování také zahrnuty.

Rozhodnutí, jak moc jsou výstupní data ze simulace reprezentativní Hlavní test, kterým se posuzuje validita simulačního modelu, je porovnání výstupních dat. Pokud se data ze simulačního modelu porovnají s výstupními daty a výsledek je příznivý, lze model označit za validní. Model ovšem vždy zůstane pouhou aproximací systému.



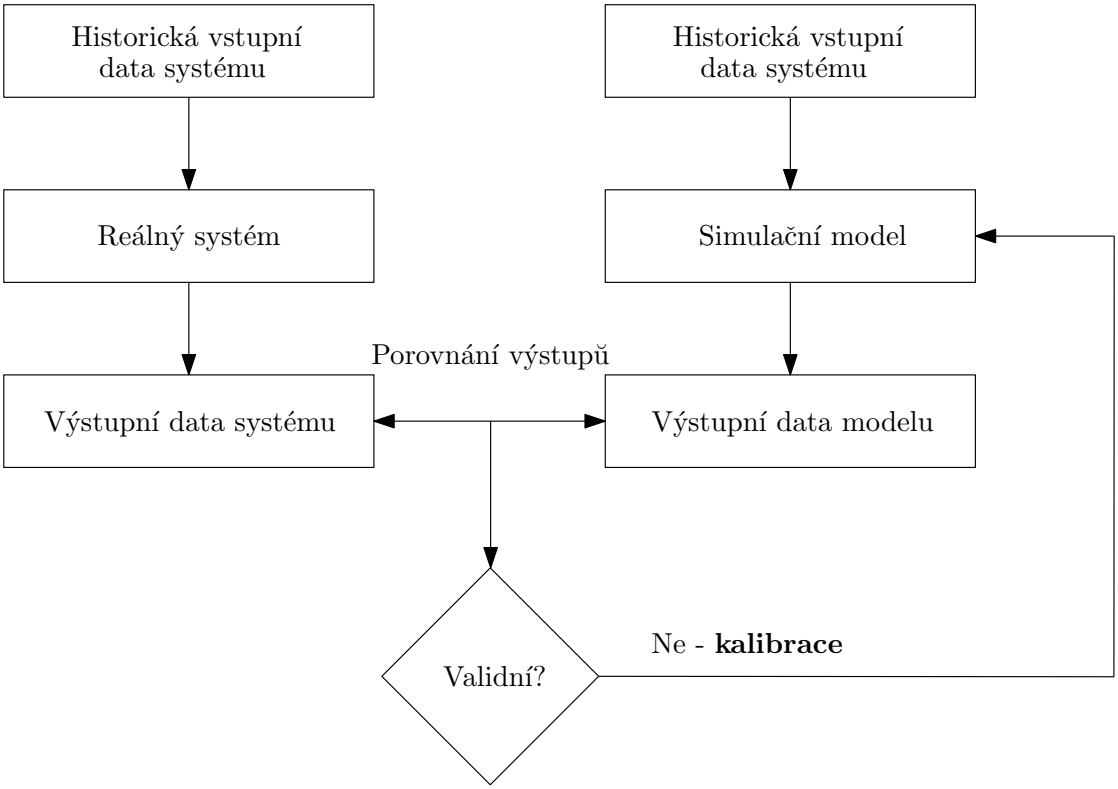
Obrázek 4.2: Vztahy mezi validací, verifikací a důveryhodností, zdroj:[1]

Shrnutí této části o validaci je názorně zobrazeno na obrázku 4.2. Obdelníky v tomto diagramu znázorňují stavy modelu a jejich vztah k procesu studia simulace. Přerušované čáry nad obdelníky ukazují, ve kterých situacích dochází k validaci, verifikaci a důveryhodnosti.

Ačkoliv to není v obrázku 4.2 zahrnuto , je důležité nezapomínat, že celý proces modelování je nutné brát jako iterativní a že některé postupy se můžou opakovat a to i několikrát.

Validace modelu tedy znamená proces testování, že model skutečně představuje uskutečnitelnou a použitelnou alternativu k experimentování se systémem. Jednou z částí je kalibrace modelu, což znamená přizpůsobování parametrů modelu do té chvíle, kdy se výstupní data nepodobají datům pozorovaných na systému. Další otázkou potom je, zda je model nakalibrován pouze na určitou sadu vstupních dat či je nakalibrován všeobecně pro různé typy vstupních dat. Simulační model je validován na základě porovnání dat mezi pozorovanými výstupními daty systému a výstupními daty vytvořenými počítačovým modelem.

Tento proces je schématicky znázorněn na obrázku 4.3.



Obrázek 4.3: Diagram validace modelu na základě porovnání dat systému a modelu, zdroj:[1]

4.3 Kalibrace modelů

K tomu, jak správně nakalibrovat model neexistuje jednotný postup a většinou je to obecné know-how jednotlivých autorů kalibrace. Existuje několik článků o kalibraci např. [4], kde se hlavně dozvíme informace o kalibrace modelů General motors, které pracují na principu rovnice (1.11), více viz. [26, str.11-13].

Je zde popsána část kalibrace věnující se i nekolizním Car-Following modelům, jako je např. Gippsův model, nicméně je zde stručně uvedeno, že Gipps model nekalibroval, nýbrž použil reálné hodnoty z provozu a jeho model s nimi rovnou fungoval, tím pádem neměl důvod se kalibrací nějakým způsobem dál zabývat.

V článku [14] se autoři snaží nakalibrovat jednotlivé modely metodou pokus omyl. Řeší zde posléze pouze různé možnosti kontroly validity (viz dále). Kalibrace parametrů na principu optimalizace je složitá z toho důvodu, že účelová funkce nemá analytické řešení.

Další možnost, jak kalibrovat modely, byla vyzkoušena institutem pro výzkum dopravy v Berlíně [5], kde vyzkoušely Nelder-Mead[2] metodu, která se numericky snaží najít maximum nelineárních problému v několikarozměrném prostoru. Výhodou pro autory byla informace o rychlosti a poloze vozidla na testovacím okruhu každou 0.1s s tím, že nepřesnost určení rychlosti a polohy byla zanedbatelná.

4.4 Statistické metody pro validaci a porovnání modelů

Statistické metody pro validaci modelů jsou vysvětleny hlavně v [15], [1] a [16]. Jako hlavní zdroj pro GEH formuli jsou použity zdroje [8] a [6]. F-test je obsažen v [20, str.111]. Střední absolutní chyba je dobře popsána v [27].

4.4.1 Statistický test střední hodnoty

Statistickým testem střední hodnoty budeme kontrolovat, zda model nemá střední hodnotu rozdílnou od nuly. V tom případě by to značilo nepřesnost v modelu či v jeho kalibraci.

Výstupem simulace bude soubor hodnot proměnných, v našem případě nejspíše intenzit, které zaznamenáváme každou vzorkovací periodu. Například, předpokládejme, že simulace má vzorkovací periodu 1 minuta - to znamená, že každou minutu přibývají hodnoty intenzit. Proměnná, která nás zajímá každou vzorkovací periodu, je intenzita w, výsledek simulačního modelu tedy bude soubor hodnot w_{ij}, kde indexem i značíme konkrétní detektor a j je čas zaznamenání. Index i probíhá od 1 do n,

$$i = 1, 2, \ldots, n,$$

kde n je celkový počet detektorů a čas j probíhá od 1 do m,

$$j = 1, 2, \ldots, m,$$

[2] též pojmenována Downhill simplexová metoda

kde m je počet vzorkovacích period v simulaci.

Pokud v_{ij} označíme naměřenou intenzitu, v případě Pražského okruhu mýtnou branou, kde indexy i a j budou stejné jako v případě modelu, klasická statistická technika k validaci modelu bude porovnání hodnot v_{ij} a w_{ij} a zjištění zda jsou u sebe dostatečně blízko.

Pro detektor i bude porovnání založeno na testování zda rozdíl

$$d_i = w_{ij} - v_{ij} \tag{4.1}$$

má střední hodnotu $\overline{d}_i$ významně vzdálenou od nuly či nikoliv. K tomu se využívá t-statistika

$$\bar{t}_{\mathrm{m}-1} = \frac{\overline{d}_i - \delta_i}{\frac{\overline{s}_d}{\sqrt{m}}}, \tag{4.2}$$

kde $\overline{s}_d$ je směrodatná odchylka od $\overline{d}_i$.

Jako nulovou hypotézu označíme

$$H_0 : \delta_i = 0. \tag{4.3}$$

Pokud zamítneme hypotézu (4.3) na hladině významnosti α, soudíme, že model není dostatečně přesný v chování. Musíme vyzkoušet jiné kombinace vstupních parametrů modelu.

Naopak pokud nezamítneme tuto hypotézu, tvrdíme, že chování simulace a systému jsou již natolik blízké, že simulaci považujeme za validní.

Tento proces musí být proveden pro každý z detektorů. Model je možné označit za validní ve chvíli, kdy hypotézu nezamítneme pro žádný z detektorů[3].

Pokud jde o tuto statistickou metodu validace, jsou zde další okolnosti, které je nutné vzít v úvahu [16] konkrétně při dopravních simulacích.

- Tento postup předpokládá stejná a nezávisle rozdělená pozorování[4]. Reálný systém a příslušné nasimulované výstupy nemusí tento předpoklad splňovat. Jednou z možností je vzít hodnoty z několika časových období a navíc zprůměrovat tyto hodnoty pro výpočet (4.1).
- Čím vyšší je vzorkovací perioda, tím nižší je kritická hodnota pro zamítnutí. To naznačuje, že simulační model má větší šanci být zamítnut, tím jak roste vzorkovací perioda. To znamená, že test za těchto podmínek není vhodný nástroj pro validaci daného modelu.

[3] Záleží ovšem na dalších okolnostech, musí být vzato v úvahu, o jaká data se jedná či že model nikdy nemůže být stejný jako simulovaný systém.

[4] i.i.d - independent and identically distributed random variables

4.4.2 F-test podílů rozptylů

Díky F-testu podílu rozptylů můžeme zjistit zda se rozptyl nasimulovaných hodnot významně liší od hodnot z mýtných bran. Nejprve je nutné vypočíst rozptyl z nasimulovaných hodnot s_1^2 a rozptyl z reálných dat s_2^2. Nulová hypotéza bude definovaná takto:

$$H_0 : s_1^2 = s_2^2. \tag{4.4}$$

Budeme testovat podíl rozptylů

$$F = \frac{s_1^2}{s_2^2}. \tag{4.5}$$

Jestliže platí, že

$$F > F_{\text{krit}} \tag{4.6}$$

zamítáme H_0 a rozptyl z nasimulovaných dat se významně liší od rozptylu z reálných dat. V opačném případě se rozptyly statisticky významně neliší. Hodnota F_{krit} také závisí na volbě hladiny významnosti a počtu stupňů volnosti obou rozptylů.

4.4.3 Střední absolutní chyba

Dalším důležitým ukazatelem je střední absolutní chyba[5], která je definována jako

$$\text{MAE} = \frac{1}{N} \sum_{i=1}^{n} |f_i - y_i| , \tag{4.7}$$

kde f_i je v našem případě nasimulovaná a y_i naměřená intenzita.

Nemůžeme bohužel použít střední procentuální chybu[6], protože některé naměřené hodnoty obsahují nulu.

4.4.4 GEH

GEH byla vymyšlena v 70. letech v Anglii. Jmenuje se dle iniciálů svého objevitele Geoffreyho E. Haverse a používá se pro zjištění informace o tom, zda je model dobře nakalibrován. Bere v úvahu absolutní i relativní rozdíly hodnot intenzit. GEH formule vypadá takto:

$$GEH_i = \sqrt{\frac{2(m-c)^2}{m+c}}, \tag{4.8}$$

kde c je reálná intenzita provozu a m intenzita nasimulovaná modelem a i je počet hodinových period přes který se počítá GEH.

[5] v odborné literatuře označována jako MAE (z angl. Mean absolute error)

[6] v odborné literatuře označována jako MAPE (z angl. Mean absolute percentage error)

Rovnice (4.8) je stejná, jako rovnice pro chí-kvadrát test, ale toto není test v pravém slova smyslu. Je to empirický vzorec vhodný pro analýzu dopravy. Ideálně funguje pro hodinové intenzity provozu, pro jiné časové úseky je doporučeno intenzity přepočítat.

Dle hodnoty GEH rozlišujeme tyto tři stavy:

- $GEH < 5$ - přijatelné, model je nakalibrován správně,
- $GEH > 5$ a $GEH < 10$ - možná chyba v kalibraci či v datech,
- $GEH > 10$ - vysoká pravděpodobnost chyby v kalibraci či v datech.

Na tomto základě je možné rozhodnout, zda výsledky, které dostaneme pomocí modelů jsou věrohodné či nikoliv.

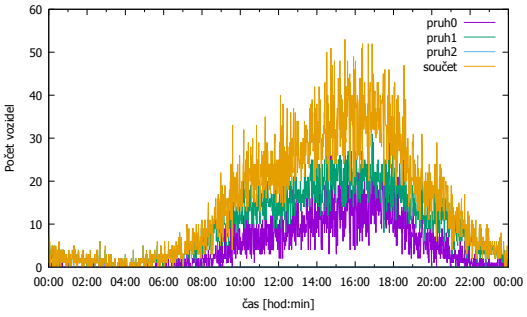
5. Simulace

V této kapitole se pokusíme zhodnotit nasimulované výsledky z modelů. Simulován bude osmikilometrový úsek na Pražském okruhu mezi 22. a 14. kilometrem. Data z mýtných bran máme k dispozici přibližně po dvou kilometrech, celkem jich je 6 - viz obrázek 5.1.

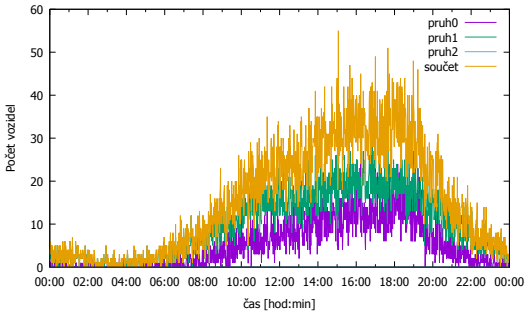
Obrázek 5.1: Simulovaný úsek dálnice–Pražský okruh km 22–14, Zdroj: [11]

Grafický podklad byl naimportován s OSM dle obrázku 5.1 - viz příloha C, nicméně po převedení do formátu SUMO některé úseky nejsou zobrazeny pro simulaci úplně přesně. Z důvodu nepřesnosti převodu infrastruktury mezi OSM a SUMO je u mýtné brány 201 o jeden jízdní pruh více než ve skutečnosti.

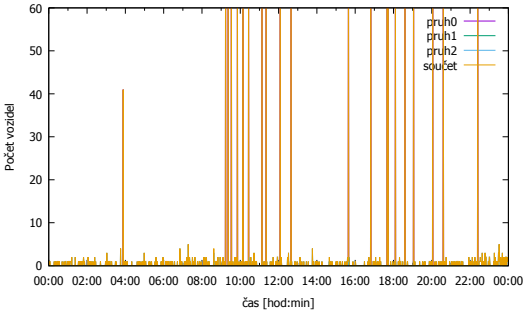
Nejprve jsme provedli kontrolu dat z mýtných bran. Bohužel touto kontrolou bylo zjištěno, že brána 157 (viz obrázky 5.4 a 5.5) a brána 145 (viz obrázky 5.6 a 5.7) obsahují nekvalitní data. Obrázek znázorňuje časovou osu, kde na ose x je čas (počítán po minutách) a na ose y je vynášen počet vozidel. Problém u brány 145 není na první pohled vidět, nicméně pokud se podíváme na intenzitu provozu na bráně 170 (viz obrázky 5.2



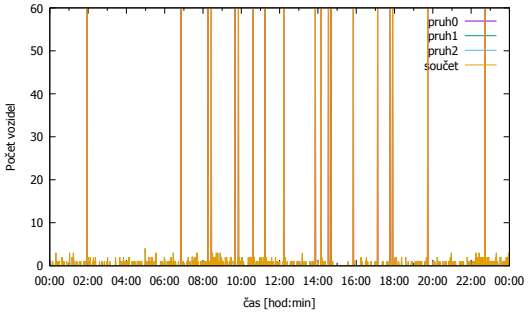
Obrázek 5.2: Brána 0170, 28.10.2012



Obrázek 5.3: Brána 0170, 7.10.2012



Obrázek 5.4: Brána 0157, 28.10.2012

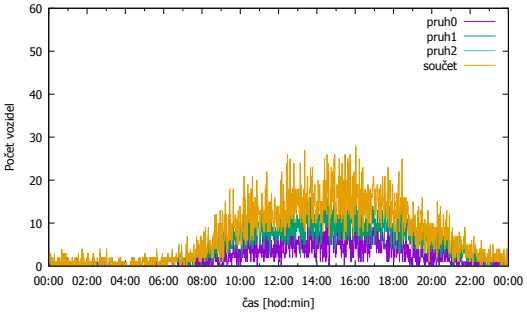


Obrázek 5.5: Brána 0157, 7.10.2012

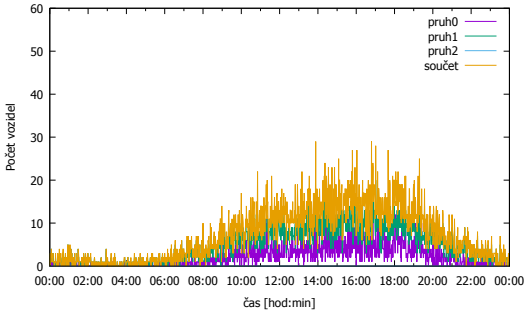
a 5.3), vidíme, že více jak o 50% vyšší než u brány 157, což je zjevný nesmysl.

Zaměříme se proto hlavně na úsek mezi branami 218 a 170, což je pro zkoumání modelů plně postačující. Velmi důležité je ovšem data zkontrolovat ještě předtím, než mají být použity např. pro simulaci, aby se nehledala chyba někde, kde nakonec vůbec není.

Postupně nasimulujeme tento úsek Gippsovým, Kraussovým, IDM a Wiedeman-



Obrázek 5.6: Brána 0145, 28.10.2012



Obrázek 5.7: Brána 0145, 7.10.2012

novým modelem a poté porovnáme jednotlivé modely. Tyto modely začneme kalibrovat tak, že vezmeme hodnoty doporučené pro simulaci autory a dále je budeme zpřesňovat.

Jako kritérium pro validní model použijeme formuli GEH uvedený v podkapitole 4.4.4. Máme totiž k dispozici data z detektorů (c) a nasimulovaná data pomocí jednotlivých modelů (m) z rovnice (4.8). Intenzity počítáme jako součet počtu vozidel za hodinu pro reálná i nasimulovaná data.

Čas vjezdu vozidel byl vždy generován z exponenciálního rozdělení, kde parametr rozdělení byl počet vozidel v danou minutu.

V programu SUMO je implementován vlastní předjížděcí model „DK2008“, který je primárně naprogramován pro Kraussův model. Nicméně i ostatní modely dávaly uspokojivé výsledky, proto nebylo nutné se tímto modelem přímo zabývat.

Ve všech obrázcích je zobrazena intenzita provozu v každém pruhu a dále také součet všech intenzit. Pruh 0 odpovídá pruhu nejvíce napravo při víceproudé dálnici a každý další má vždy o jedničku vyšší hodnotu. Pruh 2 tedy odpovídá pruhu nejvíce vlevo na tříproudé dálnici.

Generovány byly dva typy rychlostí, preferovaná a při vjezdu do oblasti. Preferovaná rychlost byla vždy generována z normálního rozdělení se střední hodnotou 36,11ms^{-1}[1] bez ohledu na rychlosti na mýtné bráně. Naopak rychlosti vozidel při vjezdu do oblasti byly generovány z náhodného rozdělení se střední hodnotou rovné hodnotě průměrné rychlosti na mýtné bráně.

Některé parametry modelů či vozidel jsou pro všechny modely stejná, konkrétně

- reakční čas $T_n = 1s$,
- délka vozidla $l = 4,5m$,
- bezpečnostní odstup (minimální mezera mezi vozidly) $s = 2m$.

Pro vyzkoušení funkčnosti simulací a správné kalibrace modelů byl jako první den zvolen svátek 28.10.2012.

Pokud se jedná o samotné simulace, tato kapitola bude rozdělena na simulace podle jednotlivých modelů. Nejprve budou uvedeny rozdíly mezi teorií (kapitola 1) a implementací daného modelu, dále budou následovat hodnoty parametrů a poté výsledky samotné simulace s informacemi o hodnotách GEH.

Rozdíly mezi definicí a implementací modelů jsou většinou způsobeny samotným simulačním nástrojem či velkým množstvím parametrů. SUMO umožňuje jednoduše zjistit pouze rychlost předního vozidla a odstup mezi aktuálním a předním vozidlem, kdežto v modelech může být tato vzdálenost skryta v určitém členu rovnice.

[1] to odpovídá rychlosti 130 kmh^{-1}

5.1 Gippsův model

5.1.1 Rozdíly oproti teorii

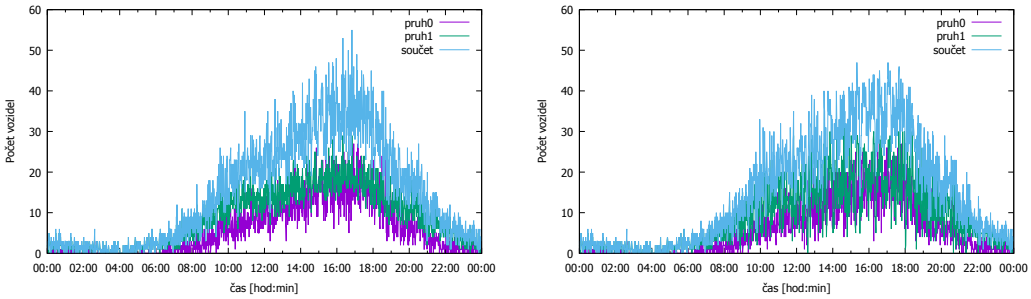
Gippsův model je naimplementován stejně tak, jak zní rovnice (1.21). Jediná drobná změna nastává u výpočtu členu $[x_{n-1}(t) - s_{n-1} - x_n(t)]$, kde není možné získat parametr s_{n-1}. Proto jsme upravili tento člen jako součet vzdálenosti mezi vozidly (předek aktuálního vozidla a zadek vedoucího vozidla) a k ní jsme přičetli minimální bezpečnostní odstup.

5.1.2 Nastavení parametrů a simulace

Nejprve jsme nechali parametry stejné jako u Kraussova modelu, nicméně to nevedlo k úspěchu. Podstatná byla informace z [4], kde bylo řečeno pokusit se nastavit hodnoty „reálně“, což v tomto případě přineslo úspěch a po dalším zpřesnění GEH nepřesáhlo nikde hodnotu 5. Simulační krok zůstal na výchozí hodnotě 1s.

Hodnoty parametrů tedy byly následující:

- zrychlení $a = 1,5\text{ms}^{-1}$
- decelerace $b = 5\text{ms}^{-1}$

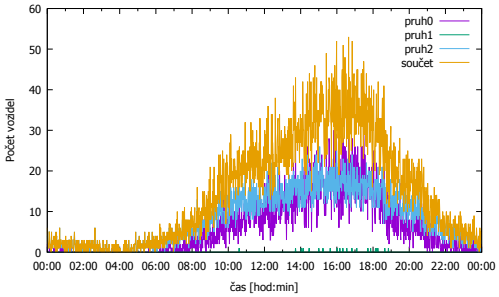


a) Reálná data b) Gippsův model

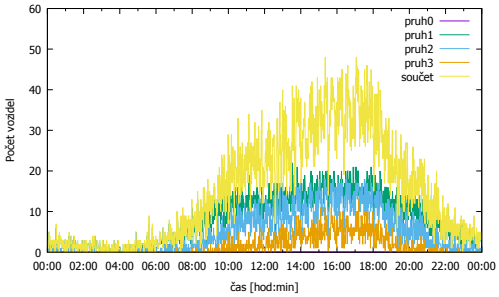
Obrázek 5.8: Brána 0218, 28.10.2012

5.1.3 Simulace

Na levé straně je vykreslena časová řada dle detektoru a na pravé časová řada nasimulovaná Gippsovým modelem. Na obrázcích 5.8a) a 5.8b) je přímo vidět určitá nepřesnost vjezdu vozidel při vysokých intenzitách. To je způsobeno podmínkou SUMO pro vjezd do oblasti. Mohli bychom např. zvýšit decelaraci k přesnění tohoto rozdílu, ovšem tím

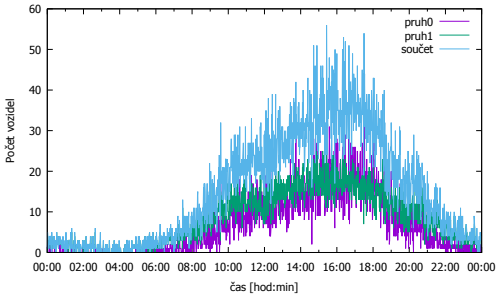


a) Reálná data

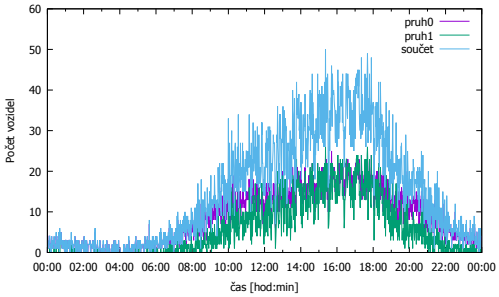


b) Gippsův model

Obrázek 5.9: Brána 0201, 28.10.2012



a) Reálná data



b) Gippsův model

Obrázek 5.10: Brána 0187, 28.10.2012

bychom snížili reálnost simulace. Na dalších mýtných branách (např. obrázky 5.9a) a 5.9b) tento jev přetrvává, což je očekávatelné.

Koeficient GEH má nejvyšší hodnotu v době mezi 15. a 16. hodinou - viz tabulka A.1, nicméně hodnota nedosahuje ani 4, čímž se dá konstatovat, a tím pádem model dává uspokojivé výsledky.

5.2 Kraussův model

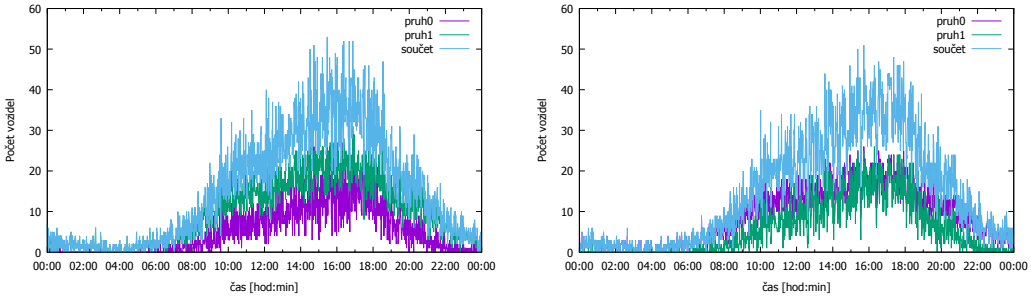
5.2.1 Rozdíly oproti teorii

Kraussův model je přímo implementován v SUMO. Byl to vůbec první model, který bylo možné v SUMO vyzkoušet. Nicméně přes zkoumání se nepodařilo nalézt přímou implementaci shodnou s (1.46), konkrétně výpočet hodnoty v_{safe}.

5.2.2 Nastavení parametrů a simulace

Simulace běžela s iteračním krokem 1s. Výsledné hodnoty nakalibrovaného Kraussova modelu byly:

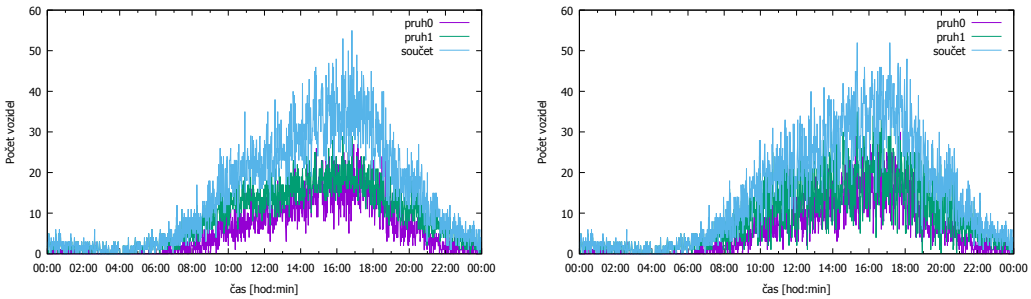
- zrychlení $a = 0,8\text{ms}^{-1}$



a) Reálná data b) Gippsův model

Obrázek 5.11: Brána 0170, 28.12.2012

- decelerace $b = 5\text{ms}^{-1}$
- náhodná výchylka η - nenastavována dopředu, generována programem SUMO, hodnota mezi 0 a 1

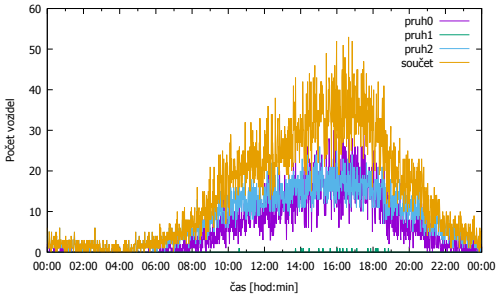


a) Reálná data b) Kraussův model

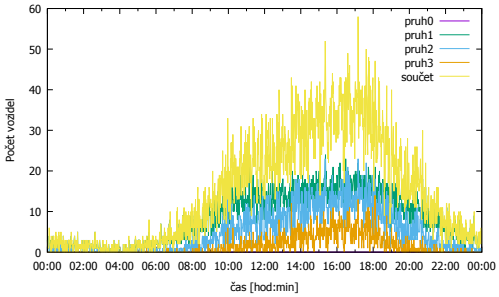
Obrázek 5.12: Brána 0218, 28.10.2012

5.2.3 Simulace

Z obrázků časových řad je jasně vidět, že Kraussův model dokázal věrně kopírovat provoz a to i v poměrně vysokých intenzitách. Jak již bylo řečeno, velkou výhodou pro Kraussův model je implementace do SUMO jako vůbec prvního modelu. Jak bude dále řečeno, nebyl úplně nejlepší z uvedených modelů. To je způsobeno tím, že intenzita, která je nasimulována Kraussem může i odpovídat skutečné intenzitě, ovšem až o 2-3 minuty, což je problém, protože brána má v tu chvíli už skutečnou hodnotu jinou. Může to být dáno generováním vozidel s exponenciálního rozdělení či generováním preferované

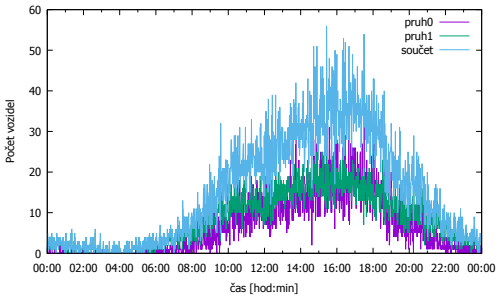


a) Reálná data

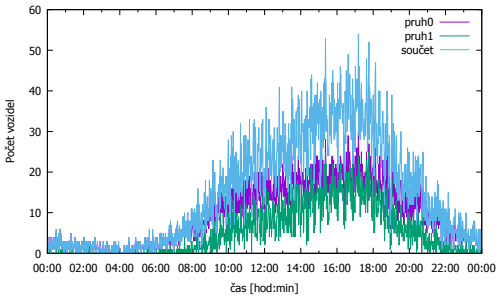


b) Kraussův model

Obrázek 5.13: Brána 0201, 28.10.2012



a) Reálná data



b) Kraussův model

Obrázek 5.14: Brána 0187, 28.10.2012

rychlosti nižší, než aktuálně byla. Co se týká maximální hodnoty GEH - viz tabulka A.2, pouze jednou překročila hodnota nepatrně hodnotu 3, což je velmi dobrý výsledek.

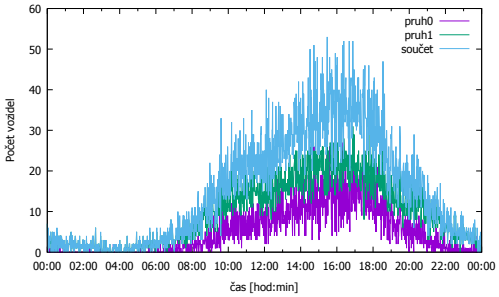
5.3 Rozšířený Kraussův model

5.3.1 Rozdíly oproti teorii

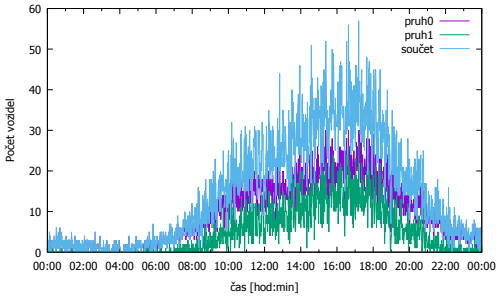
Rozšířený Kraussův model je naimplementován naprosto stejně jako v odstavci 1.4, konkrétně výpočet v_{safe} – rovnice (1.58).

5.3.2 Nastavení parametrů a simulace

Všechny parametry byly ponechány stejné jako u Kraussova modelu. Kraussův a Rozšířený Kraussův model pracují se stejnými parametry a zároveň není doporučeno měnit jejich hodnoty. [22]

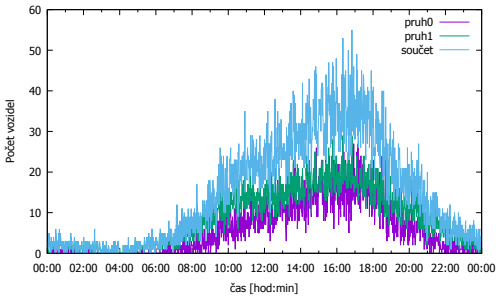


a) Reálná data

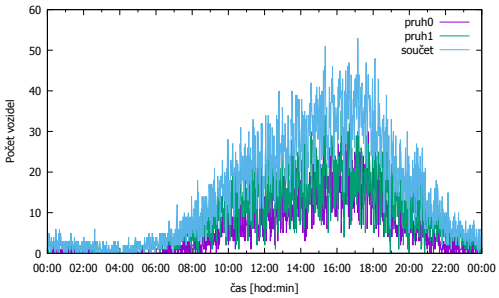


b) Kraussův model

Obrázek 5.15: Brána 0170, 28.10.2012



a) Reálná data



b) Rozšířený Kraussův model

Obrázek 5.16: Brána 0218, 28.10.2012

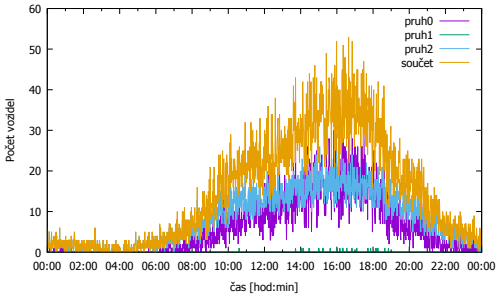
5.3.3 Simulace

Ze všech modelů dokázal Rozšířený Kraussův model dosahovat nejvyšších intenzit, až k 60 vozidlům, a to bez toho, aby byla narušena podmínka bezpečnosti. Bohužel to bylo často na základě snížení přesnosti modelu, protože skutečná intenzita nikdy tuto hodnotu nedosáhla. Je otázkou, čím je takto vysoká intenzita způsobena. Může to být z důvodu nahuštění provozu při předjíždění dvou vozidel a následně krátký silný provoz, nicméně hlavní průběh průjezdu vozidel je generován v normě – viz hodnoty GEH v tabulce A.3.

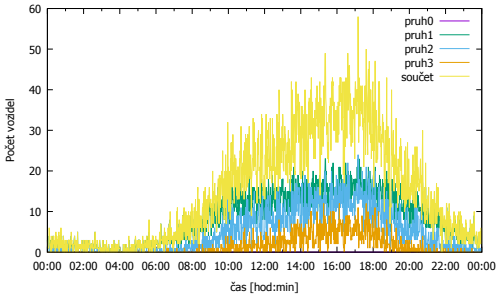
5.4 Intelligent Driver Model

5.4.1 Rozdíly oproti teorii

IDM v SUMO bylo naimplementováno poměrně nešťastně. Nebyla tam totiž uvedena verze z teoretické sekce, pouze některé její části. IDM byl naprogramován ještě se smyčkou, která mohla zrychlení vozidla počítat až několikrát. Výsledky neodpovídaly realitě a model i při sníženém simulačním kroku způsoboval kolize. Po vyzkoušení tohoto modelu jsem došel k závěru, že lepší bude naprogramovat IDM přesně tak, jak je uveden v teoretické části v podkapitole 1.5 a to i z toho důvodu, že jsem ho již zkoušel v [26].

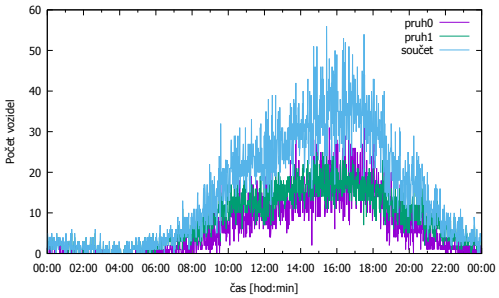


a) Reálná data

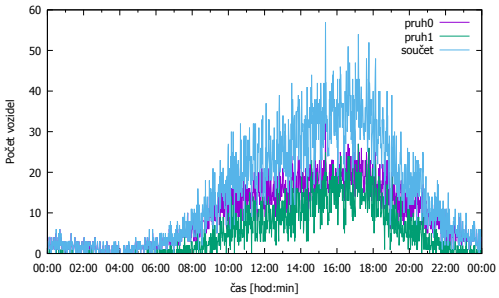


b) Rozšířený Kraussův model

Obrázek 5.17: Brána 0201, 28.10.2012



a) Reálná data



b) Rozšířený Kraussův model

Obrázek 5.18: Brána 0187, 28.10.2012

Přeprogramována je pouze ta část modelu, která počítá interakční zrychlení. Posléze již byly výsledky postačující.

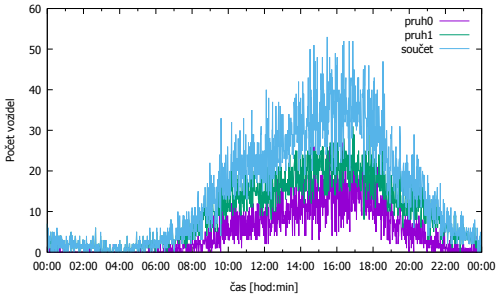
5.4.2 Nastavení parametrů a simulace

Velmi zajímavým poznatkem pro mě byla skutečnost, že když jsem si programoval IDM ve svém vlastním programu, vjezd vozidel jsem si řídil jednoduchou podmínkou, aby se mi vozidla nesrazila. Naopak v SUMO existuje sofistikovaná kontrola vjezdu vozidel a tím pádem jsem musel zvýšit hodnotu decelerace na $b = 5\text{ms}^{-1}$.

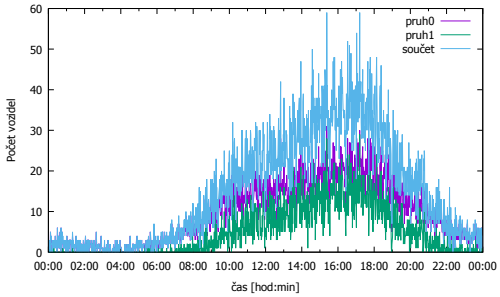
Simulační krok jsem tentokrát zvolil 0,1 s. Byl to kompromis mezi 1 s nastavenou SUMO a 0,01 s stanevou v [26]. Na logaritmické stupnici se tato hodnota nachází přes v polovině. U IDM se tímto způsobem velice zlepšily výsledky, ale také i doba výpočtu se zvýšila desetinásobně.

Parametry modelu byly stanoveny na hodnoty:

- zrychlení $a = 0,8\text{ms}^{-1}$,
- decelerace $b = 5\text{ms}^{-1}$,
- exponent zrychlení $\delta = 4$.

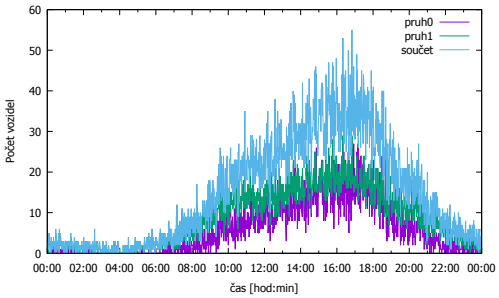


a) Reálná data

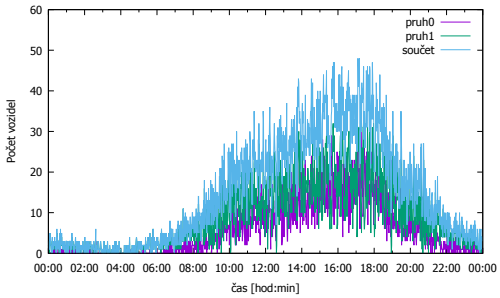


b) Rozšířený Kraussův model

Obrázek 5.19: Brána 0170, 28.10.2012



a) Reálná data



b) Intelligent Driver Model

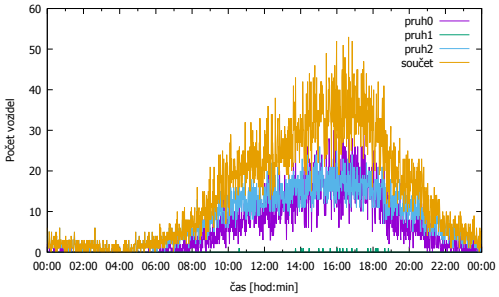
Obrázek 5.20: Brána 0218, 28.10.2012

5.4.3 Simulace

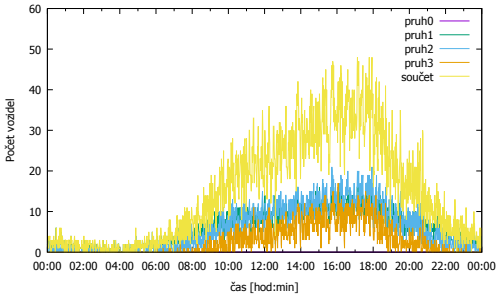
Maximální počet vozidel, která vjíždí do oblasti, se za minutu při těchto parametrech pohybuje okolo 40, viz obrázek 5.20. V další části se již tento počet zvýšil až k 50 vozidlům. Je to zaprvé způsobeno vjezdovou podmínkou a dle mého názoru předjížděcím modelem, který není zcela přizpůsoben IDM modelu. IDM model pracuje ideálně s předjížděcím modelem MOBIL [26]. Obrázek brány 5.21 vypadá tak, že součet pruhů neodpovídá celkovému součtu vozidel. Je to zapříčiněno nepřesností v infrastuktuře (viz výše), křivka pruhu 1 je skoro totožná s pruhem 2. Na další bráně, obrázek 5.22, jsou výsledky nepřesné primárně z důvodu předjížděcího modelu. U poslední brany, zobrazené na 5.23, se intenzity již zpřesnily. Nicméně stále nastává situace, že intenzita je nejdříve poměrně malá, náhle se velmi zvýší a následně opět sníží a to pro oba pruhy. Dle hodnot GEH je IDM v normě – viz A.4.

5.5 Wiedemannův model

Wiedemannův model je v SUMO naimplementován skoro stejně jako v teoretické části. Jediná ale zároveň velmi podstatná změna nastává při zadávání parametrů – Wiedemannův model jich má sám o sobě mnoho. SUMO dokázalo těchto více jak 10 parametrů

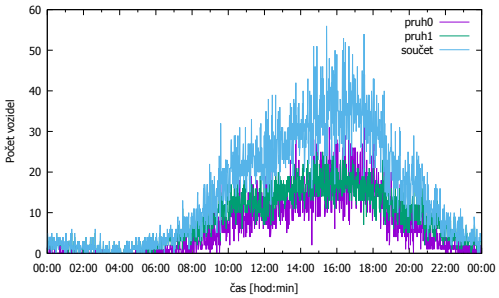


a) Reálná data

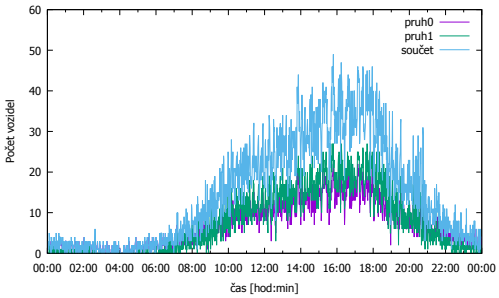


b) Intelligent Driver Model

Obrázek 5.21: Brána 0201, 28.10.2012



a) Reálná data

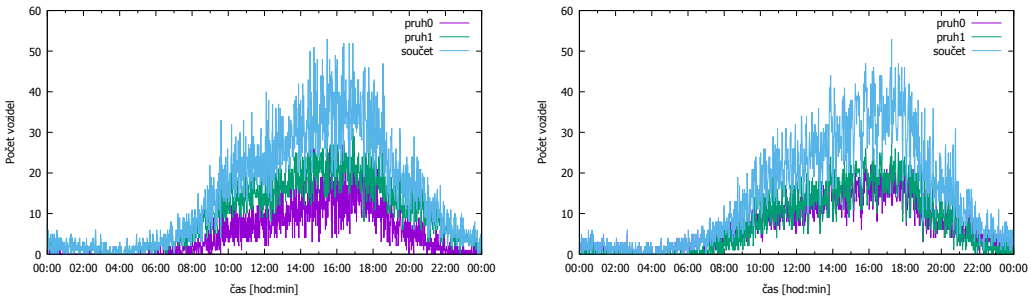


b) Intelligent Driver Model

Obrázek 5.22: Brána 0187, 28.10.2012

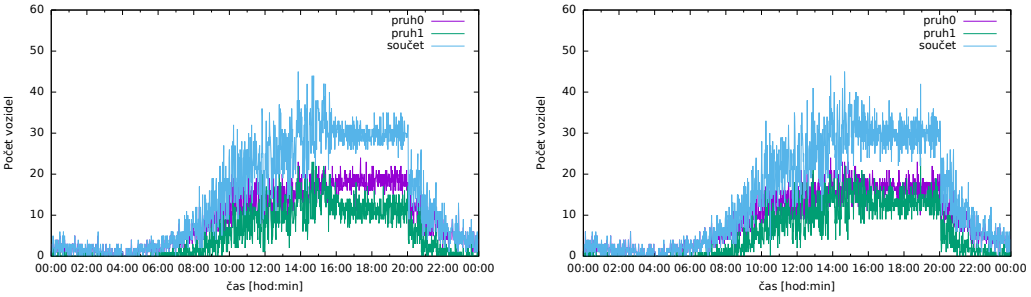
modelu shrnout pouze do dvou. Bohužel pouze nastavení těchto parametrů nepostačuje k tomu, abychom byli schopni spustit simulaci na přijatelné úrovni. Navíc nikde není vysvětleno, jak tyto dva parametry souvisí s hranicemi režimů. Proto bohužel výpočty křivek AX, SDX atd. nejsou v SUMU tak přesné, aby bylo možné simulovat reálný provoz. Jinak zde existují stejné podmínky pro určení režimu jako v obrázku 1.6. Vozidla se během simulace sráží a nepodařilo se mi přesně zjistit, kde se nachází problém. Jednou z variant je výpočet decelerace, protože Wiedemannův model ji jako jediný z modelu nemá přímo jako parametr.

Některé hodnoty v GEH tabulce přesahují hodnotu 17, což také dokazuje, že v modelu je někde chyba. Pro ilustraci přikládám na obrázku 5.24 počty vozidel na detektorech u mýtných bran 170 resp. 187, kde je jasně vidět, že model selhává a to bohužel i při nízkých intenzitách.



a) Reálná data b) Intelligent Driver Model

Obrázek 5.23: Brána 0170, 28.10.2012



a) Brána 0187 b) Brána 0170

Obrázek 5.24: Wiedemannův model

6. Porovnání modelů

Hlavním cílem této práce je porovnání mikroskopických modelů s reálnými daty. Nástroje, které k tomu využijeme, jsou uvedeny v podkapitole 4.4. Dalšími možnostmi porovnání modelů jsou grafická znázornění intenzit či absolutních chyb v průběhu dne.

Jednotlivé modely budeme porovnávat na nasimulovaných datech uvedených v kapitole 5. Wiedemannův model není v tomto porovnání vůbec obsažen, protože vozidla se v tomto modelu srážejí a výsledky by němely žádnou vypovídající hodnotu.

6.1 Střední hodnota

Prvním nutným testem byla kontrola nulové střední hodnoty u absolutní chyby modelů, tzn. že ji model nemá vychýlenou do kladných či záporných hodnot.

Díky výsledkům od všech modelů nebylo ani nutné tento test provádět, protože průměr absolutní chyby ze všech hodnot ze dne 28.10.2012 vychází nulový.

Jak již bylo řečeno, pokud by byla hypotéza (4.3) zamítnuta, tj. že střední hodnota absolutní chyby nebude rovna nule, značilo by to nepřesnost v modelu či v jeho kalibraci.

6.2 Rozptyl a směrodatná odchylka

Z důvodu kontroly, zda se rozptyl naměřených intenzit modelů významně neliší od rozptylu naměřených dat, byl proveden F-test podílu rozptylů vysvětlený v podkapitole 4.4.2. Pro hladinu významnosti 95% má kritická hodnota F-testu hodnotu

$$F_{\text{krit}} = 1{,}090. \tag{6.1}$$

Výsledky jednotlivých modelů jsou shrnuty v tabulce 6.1.

Model:	**Hodnota F**
Gipps	1,0124
Krauss	1,0233
Rozšířený Krauss	1,0234
IDM	1,0187

Tabulka 6.1: Výsledná statistika pro jednotlivé modely

Je tedy patrné, že pro všechny modely rovnice (4.6) neplatí a tím pádem se rozptyly jednotlivých modelů významně statisticky neliší. Je to také kontrola dávají modely přijatelné výsledky.

Směrodatné odchylky σ rozdílů mezi naměřenými a nasimulovanými daty jednotlivých modelů jsou znázorněny v tabulce 6.2. Je tedy vidět, že nejlepší výsledky podle

tohoto ukazetele měl Gippsův model před IDM a Kraussovými modely, které jsou vyrovnané.

Model	σ [voz]
Gipps	5,97601
Krauss	6,26314
Rozšířený Krauss	6,26967
IDM	6,02885

Tabulka 6.2: Směrodatné odchylky jednotlivých modelů

6.3 Střední absolutní chyba

Pro všechny modely jsme posléze vypočítali střední absolutní chybu mezi skutečnou a nasimulovanou intenzitou dle rovnice (4.7). Výsledné hodnoty pro všechny modely jsou shrnuty v tabulce 6.3.

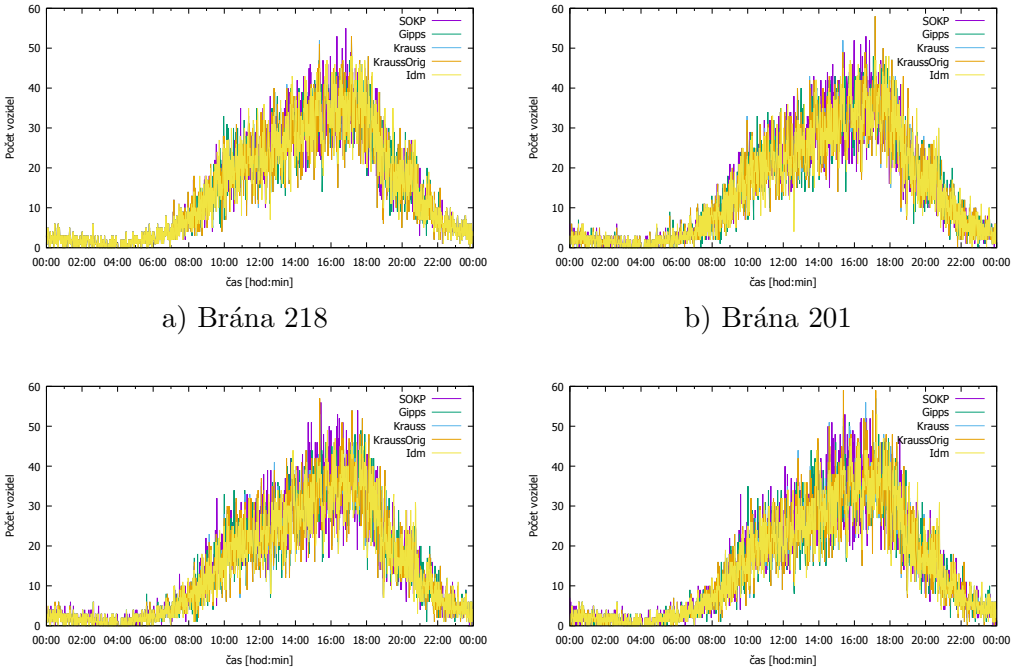
Model	MAE [voz]
Gipps	3,579
Krauss	3,665
Rozšířený Krauss	3,665
IDM	3,590

Tabulka 6.3: Střední absolutní chyba (MAE) jednotlivých modelů

I podle tohoto ukazatele je nejlepší Gippsův model, nicméně rozdíly hodnot chyb nejsou nějak vysoké, což značí možnost použití všech modelů. Mezní hodnota střední absolutní chyby, kdy už model neposkytuje data v požadované přesnosti, není striktně stanovena. Dle mého názoru by chyba neměla přesahnout 25%. Pokud budeme uvažovat průměrnou intenzitu 30 voz/s, jedná se o hodnotu 7,5 vozidla.

6.4 Grafická znázornění

Jako další variantu porovnání modelů jsme zvolili průběhy intenzit na jednotlivých mýtných branách, viz obrázek 6.1. Z těchto obrázku vyplývá jistá podobnost modelů, žádný z nich se výrazně nevymyká. Nejvíce nepřesný v počtu vozidel dle obrázků je Rozšířený Kraussův model, což je matematicky doloženo v předchozích podkapitolách (směrodatná odchylka, střední absolutní chyba).



a) Brána 218

b) Brána 201

c) Brána 187

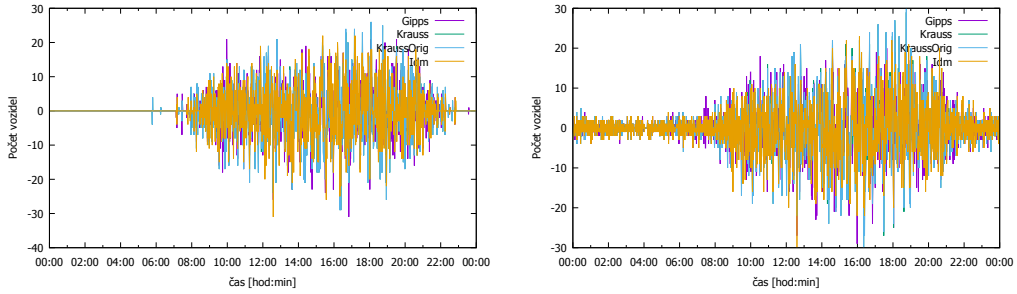
d) Brána 170

Obrázek 6.1: Porovnání jednotlivých modelů mezi sebou - konkrétní hodnoty

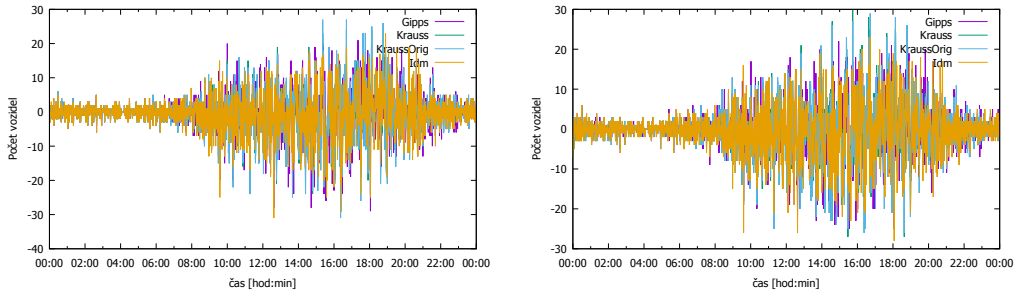
Dále jsme vynesli absolutní chybu jako časovou řadu a to nejprve pro všechny brány, obrázek 6.2 a také pro každý jednotlivý model zvlášť na poslední mýtné bráně 170 - obrázek 6.3. Pokud se zaměříme na výsledky modelů na bráně 170 je vidět i pouhým okem, že Gippsův model dává asi nejlepší výsledky. IDM by mohl jistě dávat lepší výsledky, pokud by chyba někdy až příliš často nestoupala do kladných hodnot, a hned potom rychle klesla a to až o 30 vozidel. Tento problém je souhrou několika okolností. Hlavní dvě z nich jsou již zmíněny výše, a to předjíždění vozidel a zároveň vjíždění vozidel do oblasti. Kraussovy modely jsou poměrně dost rozkmitány, což by se postupným dalším kalibrováním mohlo částečně vylepšit.

O trochu větší chyby během první noční hodiny u mýtné brány 170 jsou způsobeny tím, že ještě nejsou žádná vozidla v simulaci. Simulace startuje o půlnoci, ale v reálných datech zde již nějaká vozidla projela. Ta ale vyjela ještě před půlnocí minulého dne.

Dalším důvodem vzniku chyb jsou nájezdy a výjezdy z Pražského okruhu, které neměříme a nejsme schopni je ani realisticky simulovat. Další nepřesností je generování vozidel do simulace z exponenciálního rozdělení - viz podkapitola 2.2.3. Dále nesmíme zapomenout na ne úplně přesný import podkladových dat. Ovlivnění výsledků také může plynout z předjížděcího modelu, který je v SUMO implementován. Závěrečný vliv, který nesouvisí přímo se samotným modelem, je vjezd vozidel do simulace na základě vstupní podmínky, tedy pouze v okamžiku, kdy má vozidlo dostatečný časový i bezpečnostní



a) Brána 218 b) Brána 201



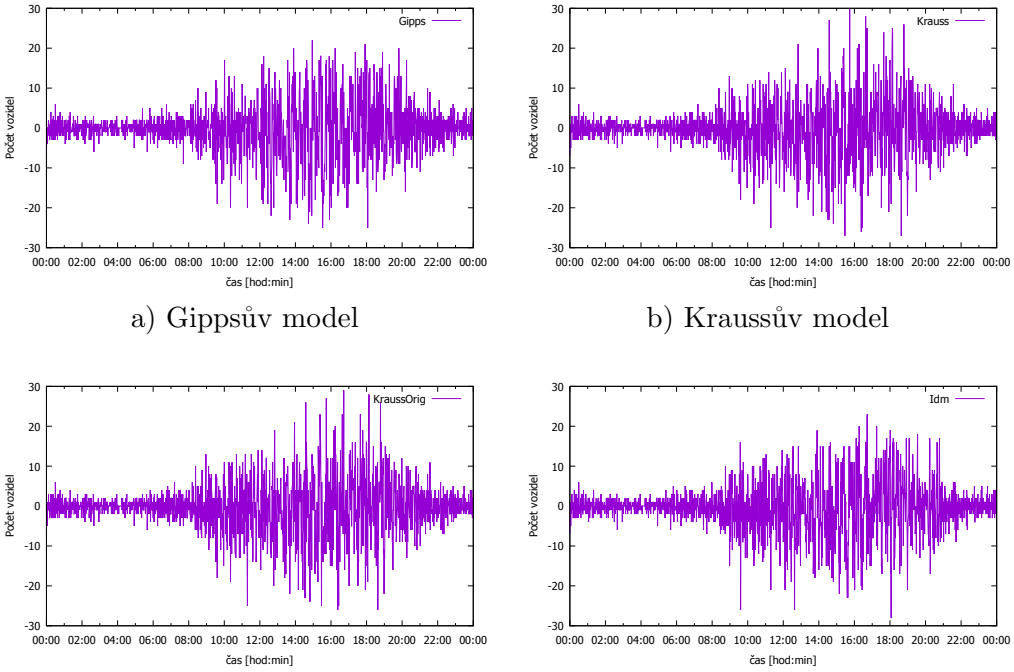
c) Brána 187 d) Brána 170

Obrázek 6.2: Porovnání jednotlivých modelů mezi sebou - absolutní chyby

odstup a neohrozí vozidlo před sebou.

Pokud se podíváme na průběhy všech modelů, vidíme, že absolutní chyba se zvyšuje se zvyšující se intenzitou dopravy. To je také dáno i tím, že je velice těžké do modelů zakomponovat chování řidiče. Řidič má z tohoto pohledu nepředvídatelné chování a je velice těžké ho do výpočtů zahrnout. Jako příklad můžeme uvést omezenou rychlost na Pražském okruhu na 130 km/h. Pokud bychom nastavili podmínku, že vozidlo nemůže jet rychleji, než je tato rychlost, potom by naše výsledky byly ještě více zkreslující. Ne každý řidič totiž maximální povolenou rychlost na dálnici dodržuje. U nízkých intenzit (do 8 hodiny ranní) všechny modely simulovaly dopravu poměrně dobře – absolutní chyba není větší než 5 vozidel.

První větší chybu pozorujeme u všech modelů až před 10. hodinou, kdy se zvyšuje i skutečná intenzita. Jak již bylo řečeno, aby modely nezpůsobily srážku vozidel, raději vozidla nepustí do oblasti pokud není zajištěn dostatečný odstup. Před 10. hodinou tedy vjelo do oblasti méně vozidel než ve skutečnosti. Tím pádem hned, jak to bude možné, se do simulace posílají další vozidla. To ale trvá řádově minuty a v tu chvíli už má brána jiný údaj o intenzitě – většinou nižší než pro stejnou minutu nasimuluje model. Díky tomu se chyba modelu o počtu vozidel překlopí ze záporných (větší počet reálných vozidel než nasimulovaných) do kladných hodnot. Z toho důvodu byl pro validaci modelů zvolen jako primární ukazatel GEH. Je totiž vidět, že pokud se počty vozidel sečtou hodinově,



a) Gippsův model

b) Kraussův model

c) Rozšířený Kraussův model

d) Intelligent Driver Model

Obrázek 6.3: Porovnání jednotlivých modelů mezi sebou. Mýtná brána 0170

problémy vyjmenované výše v tuto chvíli nemají velkou váhu a ovlivní GEH pouze na začátku a konci dané hodiny.

Všechny modely nicméně poměrně přesvědčivě potvrdily tvrzení, že je v podstatě jedno, který z nich použijeme. V našem případě ukazuje nejlepší výsledky Gippsův model. Ze všech modelů, které byly uvedeny v teoretické části, má takové parametry, které se dají přímo nastavit na reálné hodnoty (je to také z důvodu určených koeficientů u jednotlivých členů rovnice (1.21)). Při stejně nastavených hodnotách, jako má Gippsův model, nedosahuje žádný z modelů stejně kvalitní výsledky.

Jak již bylo řečeno v podkapitole 5.4.1, IDM byl v SUMO primárně naprogramován v určitém módu, kde ovšem dosahoval velmi špatných výsledků. Bylo tedy nutné model přeprogramovat, aby dosahoval přijatelných výsledků. Přeprogramování proběhlo nahrazením výpočtu interakčního zrychlení v SUMO za interakční zrychlení dle rovnice (1.61). Po přeprogramování již IDM dosahoval poměrně kvalitních výsledků. Na obrázku 6.3 d) pozorujeme zajímavý jev a sice, že u tohoto modelu jsou daleko vyšší hodnoty výchylek do záporna. Částečně to může být způsobeno problémem s vjížděním vozidel do oblasti. Intenzity se snaží nekolísat, zatímco intenzita reálné dopravy se mění z minuty na minutu i o desítky vozidel. Střední hodnota po velkém výkyvu do záporných čísel se u IDM na pár minut zvýší o 2–3 vozidla.

Oproti tomu porovnání podle GEH, který zkoumá hodinové intenzity, je vidět, že si v určitých částech dne vedly jednotlivé modely různě. Nejlepších výsledků dosáhl

Kraussův model, jehož maximální hodnota GEH byla 3,25. Je také vidět, že největší problém měly modely mezi 17. a 18. hodinou. Toto tvrzení ovšem neplatí pro Gippsův model, který měl největší problémy mezi 14. a 15. hodinou.

Závěr

Modelování dopravy se v dnešní době využívá hlavně při predikování dopravy. Díky němu můžeme, na základě zjištěných dopravních parametrů dopravního proudu detektory, tento proud také ovlivňovat, ať již změnou rychlosti či např. prodloužením fáze volno. Výhodou mikroskopického přístupu je zkoumání každého jednotlivého vozidla a hlavně proto se používají mikroskopické modely pro simulaci při vzniku dopravních nehod, dočasného uzavření jízdního pruhu ap.

Mikroskopických modelů popisujících chování vozidel na komunikaci existuje mnoho. Některé z nich se více či méně hodí na situace, které lze modelovat. Vždy zůstane otázkou, zda by jiný mikroskopický model nedával lepší výsledky než aktuální či zda daný model nemohl být lépe nakalibrován.

Modely, které budou mezi sebou dále porovnávány, jsou odvozeny již od základní myšlenky autorů v kapitole 1. Na konci každé podkapitoly je uveden seznam parametrů modelu. To je vhodné ve chvíli, kdy by měl být model použit přímo pro implementaci a postup, jakým ho autor odvodil, není důležitý. Na konci této kapitoly je krátké shrnutí těchto modelů, jejich výhod či nevýhod pro použití při simulování dopravy.

Tato diplomová práce si za hlavní cíl kladla porovnat v této chvíli nejpoužívanější mikroskopické modely na stejném úseku rychlostní komunikace. Ještě než vůbec byly spuštěny první simulace, byl naprogramován podpůrný program, který obsahuje funkce nutné k spuštění simulace – generování vozidel a dále také funkce k vyhodnocování těchto simulací, konkrétně tvorbu grafů časových řad a harmonogramů. Detailně se jím zabývá podkapitola 2.2.

Dále bylo nutné vhodně zvolit simulační nástroj, který by umožňoval měnit mezi jednotlivými modely a zároveň uměl nastavovat všechny parametry modelů. Těmto požadavkům vyhovoval opensource simulační nástroj SUMO - jeho detailnější popis lze nálezt v podkapitole 2.1. Ten již některé modely určené na porovnání obsahoval. Bohužel tento nástroj neobsahuje žádnou možnost využití makroskopického nástroje a proto nebylo možné na něm vyzkoušet některý z makroskopických modelů. Část modelů bylo potřeba přeprogramovat a některé úplně od začátku naimplementovat. Tvorbou implementace nového modelu do SUMO se nachází v kapitole 3.

Data získaná pro účely této diplomové práce jsou z mýtných bran Pražského okruhu z roku 2012. Z toho důvodu byl do SUMO naimportován právě úsek, ve kterém se nacházejí tyto mýtné brány. Postup, jak naimportovat jakýkoliv úsek z OSM je uveden v příloze C.

Ve chvíli, kdy jsme měli připraveny všechny vstupy do simulace a následné vyhodnocení, bylo nutné provést kalibraci modelů a s tím související validaci. S tím souvisela otázka nalezení vhodné metody validace. Podrobnosti k této otázce jsou shrnuty v kapitole 4.

Pokud máme k dispozici validované modely, bylo možné již spustit samotnou simulaci s příslušnými parametry. Průběh simulace, hodnoty parametrů u jednotlivých modelů a hlavně graficky znázorněné výsledky simulací jsou blíže popsány v kapitole 5.

V závěrečné kapitole porovnáváme modely z nasimulovaných dat. Porovnání probíhá jak matematicky (použité metody jsou teoreticky popsány v 4.4), tak i graficky na základě grafů časových řad.

Pro porovnání jednotlivých modelů jsme si vzali jednodušší případ a to konkrétně svátek 28. října. Bylo to z toho důvodu, abychom vyzkoušeli, zda je vůbec možné modely porovnávat a zda dávají přijatelné výsledky. V tuto chvíli existují další možnosti, jak dále pokračovat.

V první řadě je to pokusit se nasimulovat, ať už z reálných dat nebo z předpřipravené situace, nějakou složitější událost - např. uzavření jednoho jízdního pruhu. Důležité v tomto ohledu je brát zřetel na infrastrukturu na které se bude daná situace programovat.

Dále by bylo jistě vhodné zamyslet se nad způsobem vjezdu vozidel do jakékoliv simulované oblasti. Dle mých znalostí nyní neexistuje žádný ideální způsob jak toto dělat a může to způsobit i chybu, která se v první chvíli jeví jako chyba modelu.

Literatura

[1] *Aimsun 7 Dynamic Simulators User's Manual*. TSS-Transport Simulation Systems, 2012.

[2] BALCI, Osman. Verification validation and accreditation of simulation models. In: *Proceedings of the 29th conference on Winter simulation*. IEEE Computer Society, 1997. p. 135-141.

[3] BEHRISCH, Michael, Laura BIEKER, Jakob ERDMANN a Daniel KRAJZEWICZ. SUMO - Simulation of Urban MObility - an Overview. In: *ThinkMind (TM)*. Barcelona: IARIA, 2011, s. 55-60. Dostupné také z: http://sumo.dlr.de/pdf/simul_2011_3_40_50150.pdf

[4] BRACKSTONE, Mark a Mike MCDONALD. Car-following: a historical review. *Transportation Research Part F: Traffic Psychology and Behaviour*, 1999, 2.4: 181-196. Dostupné také z: http://www.researchgate.net/profile/Mark_Brackstone/publication/222495203_Car-following_a_historical_review/links/542002a50cf241a65a1af53a.pdf

[5] BROCKFELD, Elmar, Reinhart D. KÜHNE a Peter WAGNER. Calibration and validation of microscopic traffic flow models. *Transportation Research Record: Journal of the Transportation Research Board*. 2004, **1876**(1): 62-70. Dostupné také z: http://core.ac.uk/download/pdf/11091639.pdf

[6] Model Calibration. *WisDOT Information Wiki* [online]. 2009 [cit. 2015-04-23]. Dostupné z: http://www.wisdot.info/microsimulation/index.php?title=Model_Calibration

[7] SUMO Documentation. SUMO - Simulation of Urban MObility [online]. 2001 [cit. 2015-05-23]. Dostupné z: http://www.sumo.dlr.de/doxygen/index.html

[8] FELDMAN, Olga. The GEH Measure and Quality of the Highway Assignment Models. In: *European Transport Conference 2012* [online]. 2012 [cit. 2015-05-23]. Dostupné z: http://www.researchgate.net/profile/Olga_Feldman2/publication/263140653_THE_GEH_MEASURE_AND_QUALITY_OF_THE_HIGHWAY_ASSIGNMENT_MODELS/links/00b4953a02a66492fe000000.pdf

[9] GIPPS, Peter G. A behavioural car-following model for computer simulation. *Transportation Research Part B: Methodological*. 1981, **15**(2): 105-111. Dostupné také z: http://turing.iimas.unam.mx/sos/sites/default/files/Gipps_ABehaviouralCarFollowingModel.pdf

[10] GNUplot [online]. 2011 [cit. 2015-05-27]. Dostupné z: http://www.gnuplot.info/

[11] Mapy Google. *Mapy Google* [online]. 2015 [cit. 2015-05-20]. Dostupné z: https://www.google.cz/maps

[12] HIGGS, Bryan, Montasir M. ABBAS a Alejandra MEDINA. *Analysis of the Wiedemann Car Following Model over Different Speeds using Naturalistic Data* [online]. 2011 [cit. 2015-05-23]. Dostupné z: http://onlinepubs.trb.org/onlinepubs/conferences/2011/RSS/3/Higgs,B.pdf

[13] JMATIO. JMatIO - Matlab's MAT-file I/O in JAVA - File Exchange - MATLAB Central [online]. 2013 [cit. 2015-05-27]. Dostupné z: http://www.mathworks.com/matlabcentral/fileexchange/10759-jmatio-matlab-s-mat-file-i-o-in-java

[14] KANAGARAJ, Venkatesan, Gowri ASAITHAMBI, C.H. Naveen KUMAR, Karthik K. SRINIVASAN a R. SIVANANDAN. Evaluation of Different Vehicle Following Models Under Mixed Traffic Conditions. In: *Procedia - Social and Behavioral Sciences.* Amsterdam: Elsevier, 2013, s. 390-401. DOI: 10.1016/j.sbspro.2013.11.132. ISSN 18770428. Dostupné také z: http://linkinghub.elsevier.com/retrieve/pii/S1877042813045230

[15] KELTON, David a Averill LAW. *Simulation modeling and analysis.* New York: McGraw-Hill, 1991.

[16] KLEIJNEN, Jack P.C. Statistical validation of simulation models. In: *European Journal of Operational Research.* Amsterdam: Elsevier, 1995, s. 21-34. DOI: 10.1016/0377-2217(95)00132-a.

[17] KRAUSS, Stefan. *Microscopic Modeling of Traffic Flow: Investigation of Collision Free Vehicle Dynamics.* Köln, 1998. PhD. Thesis. Universität Köln.

[18] MAY, Adolf D. *Traffic flow fundamentals.* Upper Saddle River: Prentice Hall, 1990, xi, 464 s. ISBN 01-392-6072-2.

[19] NEWELL, G. F. A simplified car-following theory: a lower order model. *Transportation Research Part B: Methodological.* Amsterdam: Elsevier, 2002, **36**(3): 195-205.

[20] PAVLÍK, Jiří. *Aplikovaná statistika.* Praha: Vysoká škola chemicko-technologická, 2005, 172 s. ISBN 80-708-0569-2.

[21] PIDD, Michael. *Computer simulation in management science.* New York: Wiley, 1984, xiv, 237 p. ISBN 04-719-0281-0.

[22] SONG, Jie, Yi WU, Zhexin XU a Xiao LIN. *Research on car-following model based on SUMO.* The 7th IEEE/International Conference on Advanced Infocomm Technology [online]. 2014 [cit. 2015-05-23]. DOI: 10.1109/icait.2014.7019528.

[23] SUMO - Wiki. SUMO [online]. 2013 [cit. 2015-05-27]. Dostupné z: http://sumo.dlr.de/wiki/Main_Page

[24] TREIBER, Martin, Ansgar HENNECKE a Dirk HELBING. *Congested traffic states in empirical observations and microscopic simulations.* Physical Review E. 2000, **62**(2): 1805-1824. DOI: 10.1103/physreve.62.1805.

[25] TREIBER, Martin. *Microsimulation of Road Traffic Flow* [online]. 2011 [cit. 2015-05-23]. Dostupné z: www.traffic-simulation.de

[26] VANIŠ, Miroslav. *Matematické modelování vybraných problémů v dopravě v jazyce Java*. Praha, 2013. bakalářská práce. FD ČVUT.

[27] WILLMOTT, CJ a K MATSUURA. Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance. *Climate Research.* 2005, **30**(1): 79-82. DOI: 10.3354/cr030079.

Seznam obrázků

Seznam tabulek

A. Hodnoty GEH pro jednotlivé modely

V následujících tabulkách jsou uvedeny hodnoty GEH pro všechny brány. Výpočet byl proveden pro každou bránu v daném časovém úseku. GEH 218 značí hodnotu GEH pro bránu 218, GEH 201 pro bránu 201 atd.

Čas od:	Čas do:	GEH 218	GEH 201	GEH 187	GEH 170
0:00	0:59	0	0,346844	0,86711	1,549193
1:00	1:59	0	0,603023	0,594089	0,591198
2:00	2:59	0	0,322562	0,106904	0
3:00	3:59	0	0,149906	0,14825	0,149906
4:00	4:59	0	0	0,27735	0,966092
5:00	5:59	0	0,569495	0,654177	0,188144
6:00	6:59	0	0,439941	0,799076	0,511208
7:00	7:59	0	0,274618	0,054841	0,111629
8:00	8:59	0,470541	0,685994	2,281916	1,872764
9:00	9:59	0,372828	1,519109	2,578874	1,920924
10:00	10:59	0,465581	1,132697	3,132882	2,334866
11:00	11:59	0,027592	0,497131	0,683586	0,164584
12:00	12:59	0,446365	1,051903	1,700837	2,278846
13:00	13:59	0,786595	1,203928	2,100061	1,738182
14:00	14:59	1,730314	1,943682	3,123043	**3,374375**
15:00	15:59	1,697956	1,389913	2,034519	1,101632
16:00	16:59	0	0,043581	1,409219	1,323431
17:00	17:59	1,99029	2,789693	2,247377	2,289925
18:00	18:59	2,041722	1,739164	1,80896	2,524103
19:00	19:59	1,299185	1,888078	1,348284	1,055179
20:00	20:59	0,710047	0,707107	0	0
21:00	21:59	0	0,593199	0,625271	0,579821
22:00	22:59	0	0,053877	0,107058	0,322097
23:00	23:59	0	0,065026	0,458094	0,525226

Tabulka A.1: Vyhodnocení GEH pro Gippsův model - 28.10. 2012

Čas od:	Čas do:	GEH 218	GEH 201	GEH 187	GEH 170
0:00	0:59	0	0,346844	0,86711	1,549193
1:00	1:59	0	0,603023	0,594089	0,591198
2:00	2:59	0	0,428845	0,106904	0
3:00	3:59	0	0,149906	0,14825	0,149906
4:00	4:59	0	0	0,27735	0,966092
5:00	5:59	0	0,569495	0,654177	0,188144
6:00	6:59	0	0,439941	0,799076	0,585018
7:00	7:59	0	0,16502	0,275033	0,279727
8:00	8:59	0,427569	0,599694	1,936949	1,483664
9:00	9:59	0,202375	1,076276	2,544668	2,092399
10:00	10:59	0,758175	1,337978	2,693421	2,100774
11:00	11:59	0,248708	0,276606	1,207407	0,688233
12:00	12:59	0,525407	1,184416	2,072142	2,358641
13:00	13:59	0,860728	1,030849	1,976492	1,515806
14:00	14:59	1,118398	1,353604	2,390506	2,853913
15:00	15:59	1,631207	1,434386	2,522452	1,390251
16:00	16:59	0,65341	0,590102	1,540387	1,323431
17:00	17:59	2,464982	**3,28581**	2,0748	1,879809
18:00	18:59	1,375215	1,097602	1,66181	2,255698
19:00	19:59	1,794506	2,496224	1,724512	1,490044
20:00	20:59	0,388311	0	0,513994	0,192154
21:00	21:59	0,209703	0,928841	0,290785	0,496564
22:00	22:59	0	0,053877	0,160701	0,322097
23:00	23:59	0	0,065164	0,393073	0,590243

Tabulka A.2: Vyhodnocení GEH pro Kraussův model - 28.10. 2012

Čas od:	Čas do:	GEH 218	GEH 201	GEH 187	GEH 170
0:00	0:59	0	0,346844	0,86711	1,549193
1:00	1:59	0	0,603023	0,594089	0,591198
2:00	2:59	0	0,428845	0,106904	0
3:00	3:59	0	0,149906	0,14825	0,149906
4:00	4:59	0	0	0,27735	0,966092
5:00	5:59	0	0,569495	0,654177	0,188144
6:00	6:59	0	0,439941	0,799076	0,585018
7:00	7:59	0	0,16502	0,275033	0,279727
8:00	8:59	0,427569	0,599694	1,936949	1,483664
9:00	9:59	0,202375	1,076276	2,544668	2,092399
10:00	10:59	0,758175	1,337978	2,693421	2,100774
11:00	11:59	0,248708	0,276606	1,207407	0,688233
12:00	12:59	0,525407	1,184416	2,072142	2,358641
13:00	13:59	0,860728	1,030849	1,976492	1,515806
14:00	14:59	1,118398	1,353604	2,390506	2,853913
15:00	15:59	1,631207	1,434386	2,522452	1,390251
16:00	16:59	0,65341	0,590102	1,540387	1,323431
17:00	17:59	2,464982	**3,28581**	2,0748	1,879809
18:00	18:59	1,375215	1,097602	1,66181	2,255698
19:00	19:59	1,794506	2,496224	1,724512	1,490044
20:00	20:59	0,388311	0	0,513994	0,192154
21:00	21:59	0,209703	0,928841	0,290785	0,496564
22:00	22:59	0	0,053877	0,160701	0,322097
23:00	23:59	0	0,065164	0,393073	0,590243

Tabulka A.3: Vyhodnocení GEH pro Rozšířený Kraussův model - 28.10. 2012

Čas od:	Čas do:	GEH 218	GEH 201	GEH 187	GEH 170
0:00	0:59	0	0,346844	0,955619	1,460425
1:00	1:59	0	0,603023	0,594089	0,691411
2:00	2:59	0	0,428845	0,106904	0
3:00	3:59	0	0,149906	0,14825	0,149906
4:00	4:59	0	0	0,27735	0,966092
5:00	5:59	0	0,381385	0,749269	0
6:00	6:59	0	0,513956	0,799076	0,511208
7:00	7:59	0	0,16502	0,38563	0,279727
8:00	8:59	0,513553	0,945854	2,804071	2,265081
9:00	9:59	0,168694	1,144191	2,442168	1,886688
10:00	10:59	0,670266	1,484982	2,898162	2,188467
11:00	11:59	0,16571	0,22137	1,207407	0,715898
12:00	12:59	0,736587	1,503387	2,365147	2,812383
13:00	13:59	0,910188	1,253446	1,902439	1,441815
14:00	14:59	1,048072	1,37713	2,696945	3,35065
15:00	15:59	1,475649	1,323243	2,233815	1,013009
16:00	16:59	0,697136	0,524348	1,649836	1,498408
17:00	17:59	2,894384	**3,672186**	2,57008	2,655313
18:00	18:59	1,151801	0,79975	0,971515	1,593046
19:00	19:59	1,445245	1,391715	0,882353	0,705273
20:00	20:59	0,549388	0,547108	0,064018	0,287128
21:00	21:59	0	0,719345	0,583333	0,704976
22:00	22:59	0	0,107833	0,214423	0,161398
23:00	23:59	0	0,065026	0,393073	0,655122

Tabulka A.4: Vyhodnocení GEH pro Intelligent Driver Model - 28.10. 2012

B. Kompilace simulačního softwaru SUMO

B.1 SUMO

SUMO je volně šiřitelný dopravní simulátor, který vznikl v roce 2001. SUMO umožňuje modelování intermodálních dopravních systémů. Ty mohou obsahovat dopravní prostředky, městskou hromadnou dopravu či chodce. Dále v SUMO existují nástroje pro vyhledávání cesty, visualizace, import či export map, počítání emisí a další. Nejdůležitější vlastností pro nás je možnost implmentování vlastního dopravního modelu. SUMO samo o sobě již obsahuje některé modely - Kraussův model 1.3, Intelligent Driver Model 1.5 a Wiedermanův model.

B.1.1 SUMO download

Na stránkách projektu SUMO[1] se stáhne aktuální verze SUMO na pravé straně. Uvažujeme pouze případ dopravního simulátoru SUMO pod Windows, v době vzniku byla na světě verze SUMO 0.22. Ideální je varianta stáhnutí tří zkomprimovaných souborů. Pro potřeby kompilace stačí pouze soubor SUMO 0.22 sources (sumo-src-0.22.0,14 MB). V souboru SUMO 0.22 manual, tutorial and examples (sumo-doc-0.22.0.zip, 35 MB) se nachází podrobná dokumentace ke všem naprogramovaným součástem SUMO a SUMO 0.22 for Windows (sumo-winbin-0.22.0.zip, 33 MB) obsahuje již zkompilované SUMO, které je možné rovnou použít. V tomto souboru se ve složce *docs/userdoc* nachází uživatelská příručka, která obsahuje i návod na zkompilování SUMO pod Windows. Tento návod ovšem v některých částech není dostatečně přesný.

B.2 Visual Studio

Pro zkompilování SUMO je zapotřebí mít nainstalované Visual Studio. To je možné získat přímo na stránkách Visual Studia[2] a po zaregistrování je možno získat Expres edici Visual Studia. Autor pracuje s Visual Studio 2010 Pro. Kompilace SUMO je tedy vyzkoušena na této verzi. **Pro správné fungování Visual Studia je nutné mít nainstalován Microsoft .NET Framework verze 4 ne vyšší!**

B.3 Ostatní závislosti

SUMO ke svému zkompilování potřebuje další knihovny. V první řadě je to **Xerces-C** (xerces-c-3.1.1-x86-windows-vc-10.0.zip, 23 MB)[3], které už je samo o sobě zkompilované.

[1] http://www.dlr.de/ts/sumo/en
[2] http://www.visualstudio.com
[3] http://mirror.hosting90.cz/apache//xerces/c/3/binaries/xerces-c-3.1.1-x86-windows-vc-10.0.zip

Podstatné je stáhnout verzi x86 a ne 64bitovou, aby s ní byly schopni další součásti pracovat.

Další závislostí jsou **knihovny FOX**, kterou si budeme muset ve Visual Studiu zkompilovat sami. V návodu u SUMO je uvedeno, že je vyzkoušena pouze verze 1.6.36[4]. Přesně pro tuto jedinou verzi se mi povedlo provést kompilaci v pořádku.

Poslední částí pro zdárnou kompilaci jsou FWTools. Verze doporučená SUMO ovšem nefunguje a ani nikde na internetu se ji nepodařilo nalézt. Funkční je ovšem verze 2.4.7[5].

Důrazně se doporučuje ani jednu z těchto částí neinstalovat do klasického adresáře Program Files, může to způsobit problémy.

Nutné je také mít nainstalovaný Python (pro něj neplatí předchozí odstavec), ovšem pozor rozhodně ne nejnovější verzi[6], nýbrž verzi 2.7[7]. U Pythonu není nutné žádné další nastavení.

B.3.1 Xerces-C

Nejprve rozbalíme obsah archivu a posléze překopírujeme ze složky *bin* obě dll knihovny[8] do složky *Windows/System32*. Přes veškerou snahu se nepodařilo XERCES naimportovat všem projektům. Tím pádem se musí celý obsah složky *include* překopírovat do složky, kde je nainstalované Visual Studio do složky *vc/include* [9]. Výsledkem tedy bude překopírovaná složka *Xercesc* ze složky *Xerces/include/* do složky */vc/include*. Pro kompilaci a následné spuštění zkompilovaných programů je nutno vytvořit systémovou proměnnou XERCES[10], jejíž hodnota bude nastavena na kořenový adresář Xerces. Poté vyhledáme v systémových proměnných proměnnou PATH a nakonec hodnoty této proměnné přidáme %XERCES%/bin/.

B.3.2 FOX

FOX klasicky rozbalíme z archivu a poté musíme Visual Studiem zkompilovat. Otevřeme v rozbaleném archivu složku *windows/vcpp* a v ní otevřeme Visual Studiem soubor *win32.dsw*. Dle verze Visual Studia se může objevit hlášení o nutnosti konvertování. Potvrdíme pro všechny (možno zaškrtnout) a necháme Visual Studio načíst FOX. Po načtení musíme vybrat možnost kompilace **Release** a zkompilujeme **pouze** projekt foxdll. Pak provedeme to samé s tím rozdílem, že vybereme možnost **Debug**. Při kompilaci můžou vzniknout chyby, ty nás ovšem netrápí, protože to důležité by se mělo vytvořit. V kořenovém adresáři se vytvoří složka *lib* ve které se nacházejí zkompilované knihovny FOXDLL-1.6.dll a FOXDLLD-1.6.dll, které také překopírujeme do složky *Windows/System32*. Do systémových proměnných je nutné přidat proměnnou z názvem FOX16 s ces-

[4] http://ftp.fox-toolkit.org/pub/fox-1.6.36.zip
[5] http://home.gdal.org/fwtools/FWTools247.exe
[6] 3 a výše
[7] https://www.python.org/ftp/python/2.7.8/python-2.7.8.msi, 16 MB
[8] xerces-c_3-1.dll a xerces-c_3_1D
[9] např. c:/Program Files/Microsoft Visual Studio 10.0/VC/include/
[10] Start - Počítač - Vlastnosti systému - Upřesnit nastavení systému (nabídka na pravé straně) - Proměnné prostředí... - Systémové proměnné - Nová...

tou do kořenového adresáře FOX[11] a následně na konec hodnoty systémové proměnné PATH přidat %FOX16%/lib.

B.3.3 FWTools

FWTools klasicky nainstalujeme (ideálně ne do složky Program Files). Vytvoříme stejně jako pro FOX systémovou proměnnou s názvem PROJ_GDAL, která bude opět odkazovat do kořenového adresáře FWTools. Poté vyhledáme v systémových proměnných proměnnou PATH a nakonec hodnoty této proměnné přidáme %PROJ_GDAL%/bin.

B.3.4 SUMO

V kořenovém adresáři SUMO ve složce **/build/msvc10** se nachází soubor s názvem *prj.sln*. Ten otevřeme Visual Studiem. Pokud je vše nastaveno správně, je možno spustit kompilaci[12] SUMO, to znamená zkompilovat všechno (všech 54 projektů). Pokud kompilujeme v režimu Release, dostaneme výstup, který se nachází v kořenovém adresáři ve složce *bin*. Tento výstup je v sumo-winbin-0.22.0.zip.

[11] např. C:/fox-1.6.36/
[12] Build → Build Solution

C. Vytvoření sítě pro simulaci v SUMO

C.1 OpenStreetMap

Pro vytvoření sítě bylo využito serveru OpenStreetMap, který obsahuje kromě pozemních komunikací spoustu dalších objektů. Nejdříve bylo na tomto serveru vybrána oblast, ve které se nacházejí mýtné brány. Posléze tato mapa byla vyexportována do souboru OSM. Ten ovšem obsahoval i další objekty, které nemají pro simulování žádnou relevanci. Tyto objekty bylo nutné odstranit.

K tomu se využil free software JOSM[1], kde se vybraly všechny prvky, které bylo nutné odstanit. Tyto prvky byly sice odstraněny, ovšem velikost souboru se ještě zvýšila, protože ke každému objektu byl pouze přirazen parametr skrytí.

Aby se prvky opravdu smazali byl využit software XMLStartlet[2], který přímo pracuje se souborem XML. Ve stejné složce, kde se nachází XMLStartlet se překopíruje soubor OSM s již odstraněnými prvky (*input.osm*) a s příkazové řádky se spustí příkaz

```
1 xmlstarlet ed -d "/osm/*[@action='delete']" < input > output.osm
```

kde *output.osm* je název výstupního souboru s přímo smazanými prvky.

C.2 netConvert

Pokud máme připravený soubor z odstavce C.1, zkopírujeme ho do složky se souborem *netconvert.exe*. Z příkazové řádky spustíme příkaz

```
1 netconvert --guess-ramps --remove-geometry --junctions.join --osm-files output.
    osm --output-file nodes.net.xml
```

Tento příkaz nám vytvoří soubor *nodes.net.xml*, který obsahuje již přesnou strukturu sítě, kterou bude možné otevřít v SUMO. Existuje mnoho parametrů netconvert, po několika zkouškách autor usoudil, že tyto parametry jsou nejblíže tomu, čeho se chce dosáhnout. Program *netconvert.exe* se pokusí uhádnout nájezdy a sjezdy s Pražského okruhu (*–guess-ramps*), které nejsou zřejmé ze souboru *output.osm*, s tím souvisí vytvoření propojení (*–junctions.join*) těchto nájezdů a sjezdů a vytvoření dalších uzlů (*–remove-geometry*). V tuto chvíli máme již použitelný soubor jako podklad pro simulaci.

SUMO pouští vozidla do oblasti tak, že vezme jeden z parametrů vozidla, který určuje vjezd do oblasti, v dalším parametru jsou informace o dalších hranách, které vozidlo projíždí. Problém výstupního souboru *nodes.net.xml*, se kterým jsou tyto parametry vozidla srovnávány, spočívá v tom, že hrany a uzly jsou 8 až 9ti místná čísla, která nemají na první pohled danou spojitost. Je tedy nutné je přejmenovat.

[1] https://josm.openstreetmap.de/
[2] http://xmlstar.sourceforge.net/

Nejlepší způsob je asi v použití SUMO-GUI, kde při najetí na prvek zjistíme jeho ID a potom v souboru *nodes.net.xml* nahradíme předem vymyšleným ID z nějakého vlastního systému. Autor zvolil číslování od 1 a v závislosti na směru bud písmeno a nebo b. Takže hrana má např. název 1a, 3b.

D. Vytvoření souborů pro nový model

D.1 Implementace nového modelu

Ideální volbou pro implementaci nového modelu je začít s modelem, který již existuje. Pokud se nacházíme v kořenovém adresáři simulačního nástroje SUMO, potom ve složce */src/microsim/cfmodels* nalezneme všechny dosud naprogramované modely.

Nejjednoduší způsob je zkopírování souborů s již existujícím modelem (např. *MSCFModel_KraussOrig2.h* a *MSCFModel_KraussOrig2.cpp* a jejich následné přejmenování na jméno našeho modelu (např. Gipps). Výsledkem tedy budou dva soubory *MSCFModel_Gipps.h* a *MSCFModel_Gipps.cpp*. Nyní otevřeme postupně oba soubory a nahradíme každý výskyt *MSCFModel_KraussOrig1* za *MSCFModel_Gipps*.

Velmi důležitá věc je potom přidat tyto soubory do projektu[1]. To se ve Visual Studiu provede volbou *Project* → *Add Existing Item* a vyberou se námi vytvořené soubory, tedy např. *MSCFModel_Gipps.h* a *MSCFModel_Gipps.cpp*.

Posléze otevřeme soubor */src/utils/xml/SUMOXMLDefinitions.h* a přidáme do seznamu parametrů[2].

V souboru */src/utisk/xml/SUMOXMLDefinitions.cpp* poté přidáme opět k sekci modelů Gippsův:

```
1 { "carFollowing-GIPPS", SUMO_TAG_CF_GIPPS},
```

Závěrečnou fází přidání modelu je otevření souboru */src/microsim/MSVehicleType.cpp*, kde se ve funkci *build* nachází dělení, podle toho jaký model je nadefinován v souboru .rou.xml. Na tomto základě je zvolen vybraný model. Do tohoto dělení je nutné vložit následující kód:

```
1 case SUMO_TAG_CF_GIPPS:
2         vtype->myCarFollowModel = new MSCFModel_Gipps(vtype, accel, decel, sigma
              , tau);
```

Pokud chceme zvolit náš model při generování vozidel, existuje více možností jak to udělat. Nejjednodušší je vložit mezi tagy v souboru s příponou .rou.xml <*vtype*> a </*vtype*> tag <*carFollowing-Gipps*>. V něm poté budou následovat jednotlivé parametry modelu.

[1] Postup, jak se otevírá a kompiluje projekt SUMO, se nachází v příloze B v sekci B.3.4

[2] ideálně vyhledat nějaký stávající model - pro Krausse SUMO_TAG_CF_KRAUSS_ORIG1 a za něj (ne místo něj!) přidat SUMO_TAG_CF_GIPPS

D.2 Implementace nových parametrů

Např. u Gippsova modelu 1.2 je nutné přidat parametr preferované rychlosti V_n. Proto je zde uveden stručný návod pro přidání jednoho parametru. To je nezávislé na modelu do té doby dokud při volání konstruktoru nepřipojíme tento parametr jako vstupní.

Pokud se nacházíme v kořenovém adresáři SUMO, potom ve složce */src/utils/xml/* otevřeme soubory *SUMOXMLDefinitions.h* a *SUMOXMLDefinitions.cpp*. V prvně jmenovaném souboru *SUMOXMLDefinitions.h* do části *enum SumoXMLAttr* přidáme název nového atributu ve tvaru[3]

```
1 SUMO_ATTR_DESIREDSPEED,
```

V souboru *SUMOXMLDefinitions.cpp* v sekci *StringBijection<int>::Entry SUMOXMLDefinitions::attrs[]* přiřadíme tyto parametry proměnným pro SUMO a to ve tvaru

```
1 { "desiredSpeed", SUMO_ATTR_DESIREDSPEED},
```

Další částí nutnou vykonat je přidání parametrů do souboru */src/utils/xml/SUMOVehicleParserHelper.cpp*, konkrétně do metody *getAllowedCFModelAttr()*. Pro Gippsův model bude kód následující:

```
1 std::set<SumoXMLAttr> gippsParams;
2 gippsParams.insert(SUMO_ATTR_ACCEL);
3 gippsParams.insert(SUMO_ATTR_DECEL);
4 gippsParams.insert(SUMO_ATTR_TAU);
5 gippsParams.insert(SUMO_ATTR_CF_DESIREDSPEED);
6 allowedCFModelAttrs[SUMO_TAG_CF_GIPPS] = gippsParams;
```

[3] Je vhodné najít sekci parametrů Car Following Modelů a přidat ji tam, nicméně při vložení např. hned na začátek to bude fungovat také. Ideální je vyhledat text „SUMO_ATTR_CF_“