BAKALÁŘSKÁ PRÁCE

Matlab simulátor pro autodráhovou platformu

David Zelenka

Květen 2015

Vedoucí práce: Ing. Dan Martinec

České vysoké učení technické v Praze
Fakulta elektrotechnická, Katedra řídicí techniky

České vysoké učení technické v Praze
Fakulta elektrotechnická

katedra řídicí techniky

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: **David Zelenka**

Studijní program: Kybernetika a robotika
Obor: Systémy a řízení

Název tématu: **Matlab simulátor pro autodráhovou platformu**

Pokyny pro vypracování:

1. Seznamte se s programováním autodráhového autíčka. Vytvořte seznam všech dostupných dílů autodráhy.
2. V Matlabu vytvořte simulátor, který bude simulovat pohyb vozidla po autodráze.
3. Ověřte funkčnost simulátoru. Porovnejte výstupy simulátoru s naměřenými daty z autodráhy.

Seznam odborné literatury:

[1] Martin Lád, Návrh a implementace řídicího systému pro autodráhové vozidlo, bakalářská práce FEL ČVUT, 2014
[2] Ashish Tewari, Modern Control Design: With Matlab and Simulink, Wiley, 2002
[3] Pavel Herout, Učebnice jazyka , KOPP, 2009

Vedoucí: Ing. Dan Martinec

Platnost zadání: do konce letního semestru 2015/2016

L.S.

prof. Ing. Michael Šebek, DrSc.
vedoucí katedry

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 12. 12. 2014

Poděkování

Tímto chci poděkovat vedoucímu práce Danovi Martincovi za ochotný přístup při konzultování problémů. Dále děkuji Martinu Ládovi za trpělivost při zasvěcování do projektu.

Prohlášení

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

V Praze, dne ....

Podpis

Abstrakt

Práce pojednává o implementaci simulátoru autodráhových vozidel v rámci projektu Slotcar platooning. Na začátku je řešen způsob reprezentace dráhy v počítači a kontrola její realizovatelnosti. Simulátor je napsán v jazyce Matlab a je rozložen do S-funkcí, které se používají jako bloky schématu v Simulinku. Bloky jsou tři: auto, dráha a zpětná vazba. Blok dráha umožňuje jízdu více autíček po jedné dráze, nicméně na sebe vzájemně nepůsobí. Matematický model autíčka zahrnuje jevy způsobené zatáčkami, které jsou do něj zavedeny pomocí zpětné vazby z výstupu dráhy. Výhodou použití více bloků je lepší přehlednost zapojení a modularita modelu. Nakonec je popsáno grafické rozhraní k vytváření různých tvarů dráhy.

Klíčová slova

S-funkce, autodráha Carrera, rotace souřadnic, Matlab GUI

Abstract

Thesis deal with implementation of slotcar simulator under the Slotcar platooning project. At first computer representation of the Carrera track and checking of its' feasibility are demonstrated. The simulator is written in Matlab language and is decomposed into three S-functions which are used as block in Simulink schemes. The three blocks are: slotcar, track and feedback. Multiple cars can be simulated on one track, although they don't interact with each other. Physical effects on cars caused by riding through curves is also included in the model. The advantage of using multiple block rather than just one is better modularity of the model. At last the graphical user interface is described.

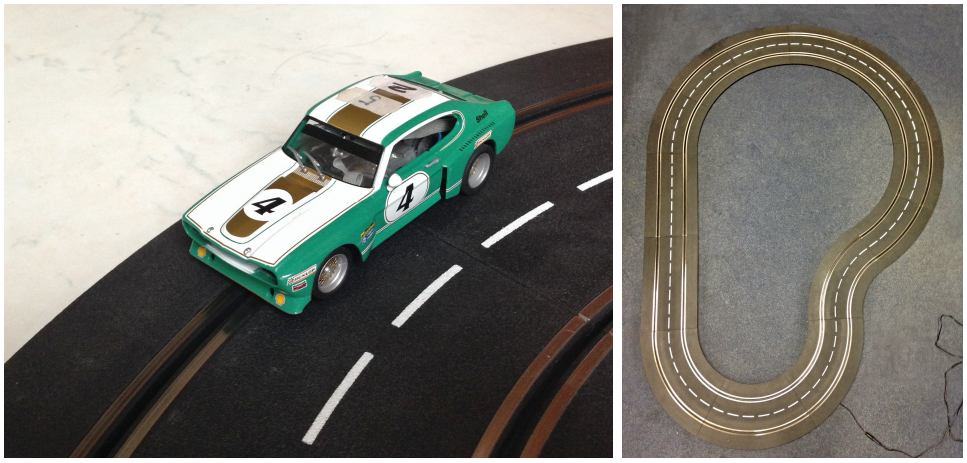
Key words

S-function, Carrera slotcar track, rotation of coordinates, Matlab GUI

Obsah

Část I
Úvod

Simulátor má sloužit projektu Slotcar platooning, který je zastřešený skupinou AA4CC. Cílem je vytvoření platformy pro testování distribuovaného řízení kolony aut. Jako základ se využívají autodráhová autíčka Carrera (obrázek I), díly autodráhy jsou od stejné firmy. Autíčka jsou vybavena řídící elektronikou a několika senzory, například senzory vzdálenosti ostatních autíček v koloně, senzor rychlosti nebo 3D gyroskop a akcelerometr. Hardware a jádro software byly navrženy Ing. Jaromírem Dvořákem. Nelineární model autíčka identifikoval Bc. Martin Lád a implementoval měření rychlosti.



Obrázek 0.1: Autodráhové autíčko a příklad skutečné dráhy

Tato práce se zabývá navržením simulátoru autíčka pohybujícího se po dráze se zatáčkami. Je napsán v jazyce Matlab. Hlavními body práce jsou:

1. Změření parametrů dostupných dílů dráhy a navržení její implementace v programovém jazyce. Navržení validátoru dráhy, který zaručí uskutečnitelnost dráhy. Simulace funguje pouze pro dráhy v 2D prostoru, čímž jsou při kontrole eliminovány dráhy s překryvy dílů a křiženími.

2. Implementace simulátoru, naprogramovaného jako S-funkce, kterou lze přidat jako blok do schématu v Simulinku. Základem je matematický model Bc. Martina Láda. K modelu budou přidány fyzikální jevy vznikající při průjezdu zatáčkou, což jsou boční zrychlení a úhlová rychlost. Jejich vliv na jízdu autíčka v podobě tření bude také započítán.

3. Vytvoření grafického rozhraní pro sestavení libovolného tvaru dráhy, který je možno ukládat a otevírat. Přídavnou funkcí bude například vytvoření obrázku dráhy.

Pokud není uvedeno jinak, jednotky všech veličin jsou ze soustavy SI. Příklad dráhy je na obrázku I

Část II
Model a software autíčka

Autíčko je model Ford Capri v měřítku 1:32 z řady Evolution od firmy Carrera. Do autíčka byla namontována řídící elektronika a senzory. Model byl již identifikován Martinem Ládem v jeho bakalářské práci[1]. Jeho model jsem použil pro simulátor a lehce ho upravil, neboť byl vytvořen pouze pro jízdu po rovince, nikoliv v zatáčkách. Pro větší přesnost jsem použil nelineární model.

Napájecí napětí motoru je vyjádřeno jako napětí zdroje U_{zdroj} s odečteným úbytkem napětí U_{dioda} na vstupní diodě. Tento rozdíl je dále vynásoben konstantou 0.96, což je úbytek napětí na odporu, na kterém je měřen protékající proud.

$$U_m = 0.96(U_{zdroj} - U_{dioda}) \tag{0.1}$$

Další rovnice popisuje třecí sílu, která je součtem dynamického a statického tření.

$$F_t(v(t)) = bv(t) + F_t^s \tag{0.2}$$

Samotný nelineární model autíčka má následující tvar:

$$\frac{dv(t)}{dt} = \frac{kU_m D(t)}{Rmnr} - \frac{k^2 v(t)}{Rmn^2 r^2} - \frac{F_t(v(t))}{m}, \tag{0.3}$$

$$y = v(t). \tag{0.4}$$

Výstupem y tohoto modelu je rychlost v v jednotkách metrů za sekundu. Jeho vstupem je D, což je střída řídícího PWM signálu.

Zbylé hodnoty jsou konstantní, byly také převzaty z modelu Martina Láda a jsou zaneseny v tabulce 1.

Název veličiny	Značka	Hodnota	Jednotka
Elektrický odpor vinutí motoru	R	5	Ω
Konstanta motoru	k	$2 \cdot 10^{-3}$	$\mathrm{N \cdot m \cdot A^{-1}}$
Hmotnost autíčka	m	0.294	kg
Poloměr kolečka	r	$10 \cdot 10^{-3}$	m
Konstanta převodu	n	$\frac{1}{3}$	-
Koeficient dynamického tření	b	0.5	-
Koeficient statického tření	F_t^s	0.55	-
Úbytek napětí na vstupní diodě	U_{dioda}	0.3	V

Tabulka 1: Parametry modelu autíčka

Chci upozornit, že značení veličin v tomto modelu je odlišné od značení používaného v implemetaci simulátoru.

Dále jsem převzal software autíčka, vytvořený Jaromírem Dvořákem, Janem Moravcem a Martinem Ládem, ve kterém je implementováno řízení motoru, čtení hodnot ze senzorů a komunikace s počítačem. V počítači jsem využil grafické rozhraní Martina Láda. Propojení těchto dvou programů umožňuje vzdáleně nastavovat autíčku mimo jiné hodnotu PWM signálu či referenční hodnotu rychlostního regulátoru a do počítače přenášet hodnoty ze senzorů. Naměřené údaje jsou zobrazovány do grafu závislého na čase, nicméně frekvence čtení těchto hodnot byla příliš pomalá, tudíž jsem program upravil pro svou potřebu.

Část III
Implementace simulátoru

1 Úprava software autíčka

Vzhledem k pomalému vyčítání údajů ze senzorů jsem provedl úpravy. Nejkratší zatáčkový díl má délku 13 cm. Nejvyšší bezpečná průjezdová rychlost touto zatáčkou je 1 m/s a průměrná frekvence výčítání ze senzorů je 9 vzorků za sekundu. Na jeden díl získáme při této rychlosti přibližně 1,2 vzorku. Cílem bylo maximalizovat frekvenci vyčítání.

Základem software autíčka je nekonečný cyklus, který provádí potřebné podprogramy vždy po určité době. Frekvenci opakování jednotlivých úkonů lze měnit nastavením doby, za kterou se mají opakovat. Úpravou je posílání paketu s daty z autíčka rovnou z tohoto cyklu. Do paketu jsem uložil následující hodnoty: čas od zapnutí autíčka, boční zrychlení, úhlovou rychlost a hodnotu střídy PWM signálu. Tento paket je označen identifikačním číslem 12, následně přidán do fronty paketů a odeslán.

Frekvence odesílání byla nastavena experimentálně. Zpočátku se zdálo možné odesílání dat s frekvencí 50 Hz. Ukázala se kolísající rychlost odesílání. První vzorky se odesílaly požadovanou frekvencí, ale následovala stejně dlouhá prodleva způsobená zaplněním fronty paketů. Snižováním frekvence odesílání se zmenšovala, až jsem se dostal na frekvenci 20 Hz, kdy prodleva vymizela úplně. Podmínkou u velkých tratí bylo zajištění přímé viditelnosti autíčka k anténě v počítači, jelikož pár paketů nebylo přijato vůbec z důvodu výpadku signálu.

V počítači jsou přijaté pakety uloženy do textového souboru pomocí upraveného grafického rozhraní. Úprava spočívá ve vytvoření funkce, která ukládá pakety označené identifikačním číslem 12 do tohoto souboru, ze kterého je možné importovat data do Matlabu.

2 Dráha

Použitá autíčka Carrera Evolution v měřítku 1:32 využívají traťové díly v měřítku 1:24. Nabídka dílů výrobce Carrera je velice rozsáhlá, od jednoduchých rovinek a zatáček po klopené zatáčky, zúžení, křížení a mosty. Má práce zahrnuje pouze základní rovinky a zatáčky, protože implementace složitějších dílů je pro účel celého projektu zbytečná.

Typ zatáčky	Středový úhel α	Radius R [m]
1/30	30°	0,297
1/60	60°	0,297
2/30	30°	0,495
3/30	30°	0,693
4/15	15°	0,891

Tabulka 2: Základní parametry zatáček

Typ rovinky	Délka d [m]
1	0,345
1/3	0,115
1/4	0,08625

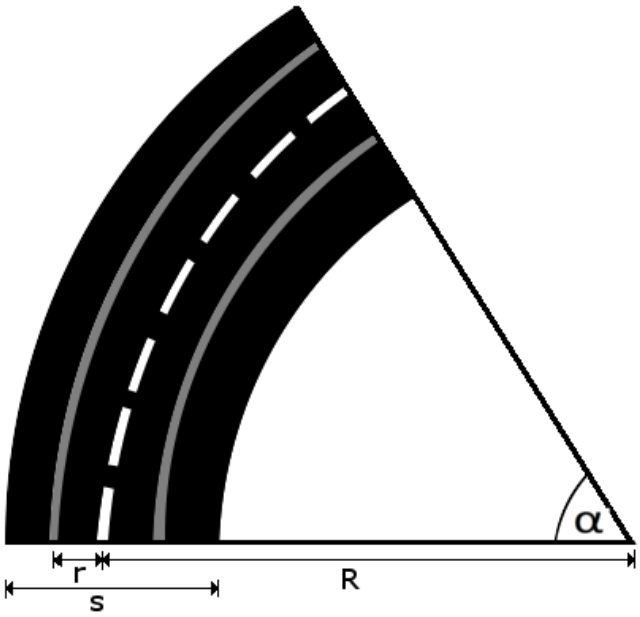
Tabulka 3: Základní parametry rovinek

2.1 Dostupné díly

V simulátoru je k dispozici 8 typů dílů, z toho jsou 3 rovinky, které se liší svou délkou, a 5 různých zatáček. Každý díl má pravý a levý slot, což je drážka vedoucí autíčko za jeho vodící lištu. Každý slot je lemován vodivými kolejemi, které jsou napájeny ze zdroje napětí. Z těchto kolejí je napájení přeneseno do autíčka pomocí vodivých kartáčků, umístěných po stranách vodící lišty.

Zatáčky lze rozdělit na 4 poloměry, přičemž typ s nejmenším poloměrem existuje ve dvou délkách. Výrobce používá intuitivní značení každého dílu, které ho přesně definuje. Rozhodl jsem se ho použít. Například zatáčka 2/30 znamená, že se jedná o zatáčku s druhým nejmenším poloměrem a středový úhel opsaného oblouku odpovídá 30°. Ke kompletnímu kruhu vytvořeného z těchto zatáček proto potřebujeme $\frac{360°}{30°} = 12$ těchto dílů. Pokud by tato zatáčka byla levá, tak ji v simulátoru označím jako L2_30. Pravá zatáčka by měla tvar R2_30. Rovinky jsou značeny S1 až S3, S1 značí nejdelší rovný díl.

V tabulce 2 je výčet dostupných dílů zatáček. Radius udává poloměr kružnice vedoucí středem dráhy, tedy poloměr mezi pravým a levým slotem. Tabulka 3 zobrazuje základní parametry rovinek, což je jejich délka.

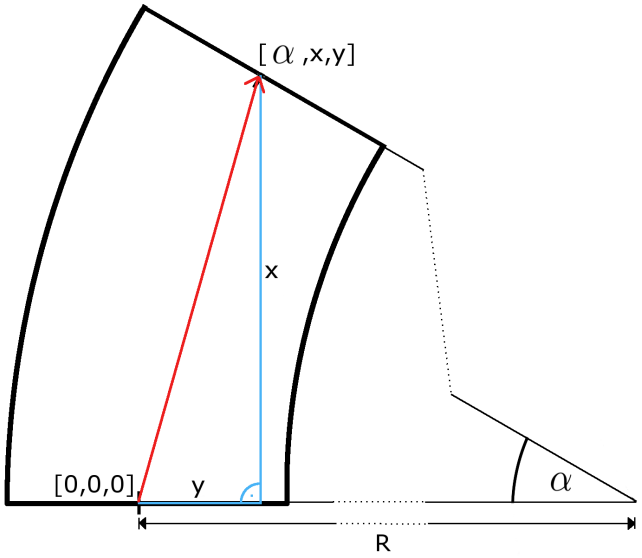


Obrázek 2.1: Základní parametry zatáček

Dalšími důležitými parametry při implementaci simulátoru je šířka dílu a pozice slotů. Na obrázku 2.1 jsou znázorněny tlustou šedou čarou, zatímco střed dráhy symbolizuje přerušovaná bílá čára. Šířky dílu budeme potřebovat při validaci dráhy, tedy při rozhodování, zda se díly nikde nekříží, či nepřekrývají. Díky tomu budeme schopni sestavit skutečnou dráhu bez mostů a vyvýšenin. Šířka všech traťových dílů je $s = 0,198$ m. Vzdálenost slotů od středu dílu $r = 49,5$ mm využijeme při výpočtu bočního zrychlení, které při jízdě působí na autíčko v zatáčce, neboť tato vzdálenost mění poloměr projeté zatáčky. Poloměr zatáčky je spolu s rychlostí autíčka hlavním parametrem pro výpočet bočního zrychlení a úhlové rychlosti otáčení autíčka.

2.2 Reprezentace dílů dráhy

Hlavním požadavkem pro formát parametrů jednotlivých dílů bylo definovat je co nejmenším počtem hodnot. Vektorové pojetí parametrů dílu má navíc zajistit kompaktnost a hlavně pohodlnost při výpočtech. Na základě těchto požadavků



Obrázek 2.2: Parametry dílů pro reprezentaci v simulátoru

jsem zvolil další a vytvořil vektor

$$par = \begin{bmatrix} \alpha \\ x \\ y \\ R \end{bmatrix}, \tag{2.1}$$

definující jeden díl, kde α [radian] je již dříve popsaný úhel, x [metr] a y [metr] jsou souřadnice vektoru spojujícího středy obou konců jednoho dílu a R [metr] je poloměr středu zatáčkového dílu. Jsou zakresleny do obrázku 2.2. Souřadnice x a y se vypočítají jednoduše

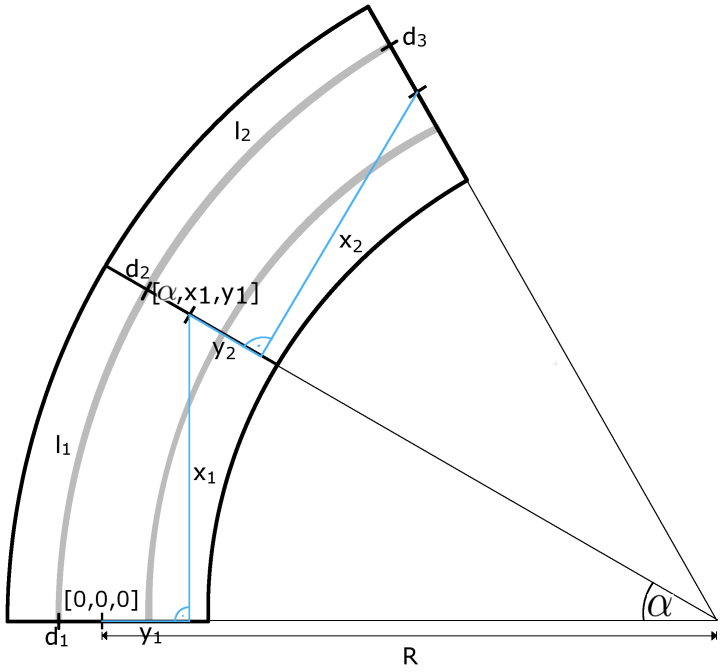
$$x = R \cdot \sin \alpha. \tag{2.2}$$

A platí, že $R = R \cdot \cos \alpha + y$, a proto

$$y = R - R \cdot \cos \alpha. \tag{2.3}$$

Pro pravé zatáčky bude znaménko y souřadnice opačné.

U rovných dílů je úhel α rovný nule, neboť se při průjezdu takovým dílem směr jízdy autíčka nezmění. Na souřadnice x a y je pak možné aplikovat vztahy



Obrázek 2.3: Zobrazení souřadnic pro dva díly
Parametry $l_{1,2}$ a $d_{1,2,3}$ platí pro levý slot dráhy

2.2 a 2.3 a odpadne tak souřadnice y. Poloměr zakřivení rovného dílu bychom si mohli představit jako nekonečný, proto by bylo i pro další počítání vhodné také tuto hodnotu na pozici parametru R uložit. Naneštěstí jsem se v Matlabu setkal s problémem, že výsledek násobení $0 \cdot nekonečno$ je NaN, proto jsem musel u rovinek poloměr reprezentovat nulou.

2.3 Reprezentace celé dráhy

Abychom mohli simulovat chování autíčka na dráze a vyčítat hodnoty, které snímají jeho senzory, musíme připravit parametry dráhy tak, aby bylo možné rychle zjistit, na kterém dílu se autíčko nachází. Usoudil jsem, že nejlepším způsobem bude ukládat parametry do matice P . Každý řádek této matice představuje jeden díl, přičemž má matice o řádek více, než je počet dílů. To je z toho důvodu, abych mohl do posledního řádku uložit hodnotu délky celé tratě.

Matice má šest sloupců, každý z nich reprezentuje jeden parametr popisující díl. Těchto parametrů by nyní mohla být pouze polovina, ale počet je pozůstatkem

vývoje programu a nelze vyloučit, že se v budoucnu zase využijí. Navíc jde o tak malá data, že by se úpravou efektivita programu výrazně nezlepšila. Tvar vektoru i-tého řádku matice je následující:

$$P_i = \begin{bmatrix} \alpha_i & x_i & y_i & R_i & l_i & d_i \end{bmatrix}. \tag{2.4}$$

První čtyři parametry již známe z definice dílu dráhy. Novými parametry, zobrazenými na obrázku 2.3, jsou l_i , což je délka dílu, a d_i , který představuje délku dráhy od jejího začátku k začátku i-tého dílu a lze ho popsat rovnicí

$$d_i = \sum_{j=1}^{i-1} l_j. \tag{2.5}$$

3 Inicializace simulace

Důležitá je funkce `init_sim`, která zpracovává řetězec posloupnosti dílů. Má dva úkoly, které probíhají současně. Pro přehlednost je budu popisovat odděleně. V příloze C je její vývojový diagram. Prvním úkolem je příprava dat pro schéma v Simulinku a druhým je validace, jež je důležitým krokem před samotnou simulací průjezdu autíčka dráhou. Musí být realizovatelná, abychom si výsledky ověřili na skutečné dráze.

Vstupními parametry funkce jsou:

1. pole řetězců *track* obsahující posloupnost kódů dílů
2. počet autíček, která chceme současně simulovat
3. volba pravého nebo levého slotu dráhy.

Na začátku je vytvořen MAT soubor `trackdata.mat`, do kterého se ukládají vstupní parametry funkce a dvě pomocné hodnoty, nazývající se step a tolerance. Popis těchto parametrů bude uveden v kapitole o validaci dráhy.

3.1 Příprava dat pro simulaci

Po vytvoření MAT souboru volám funkci `init_track`, které předávám jako parametr objekt otevřeného souboru `trackdata.mat`. Zde je možné použít přístup do souboru, protože se jedná pouze o jednorázové načtení proměnných, které není časově náročné.

Základem této funkce je for cyklus, ve kterém procházím pole řetězců *track* s posloupností kódů. Kód každého dílu pošlu jako parametr funkci `gen_part`, jejíž návratová hodnota je vektor parametrů jednoho dílu. Nejprve je upravím přepočítáním funkcí `choose_side`. Před tímto cyklem si inicializuji proměnnou d na hodnotu nula. Začátek prvního dílu je ve vzdálenosti nula od začátku celé dráhy. Na konci každého průchodu cyklem se proměnná d inkrementuje o délku i-tého dílu.

Funkce `gen_part`

Funkce na základě vstupního parametru, řetězce obsahujícího kód jednoho dílu, vrací vektor parametrů `par` daného dílu. Do parametrů přistupuje přes strukturu Map Container, jejíž konstruktor obsahuje pole klíčů, což jsou kódy dílů, a pole vektorů `par`. Vektory jsou uspořádány ve stejném pořadí jako kódy. Konstruktor vytváří objekt struktury Map Container, kterému předáme jako parametr náš klíč, kód dílu, a on vrátí vektor parametrů.

Funkce `choose_side`

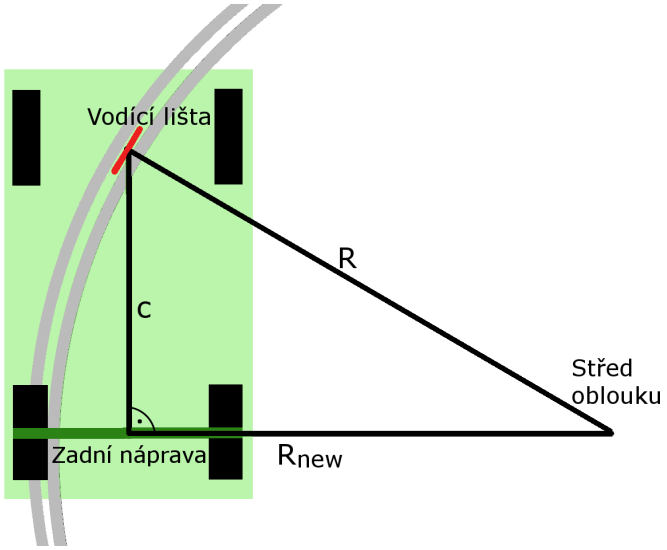
Tato funkce přičítá nebo odečítá vzdálenost slotu $\triangle r{=}r = 49,5\text{mm}$ od středu dráhy v závislosti na straně dráhy, kterou si uživatel simulátoru zvolil. Počítá také s vybočením zadní nápravy dovnitř zatáčky. Vznikne nová hodnota poloměru zatáčky R_{new}. Jestli je poloměr zvětšen nebo zmenšen je kromě volby slotu dáno také tím, zda je konkrétní slot blíže nebo dále od středu zatáčky. Podle definice dílů mají levé zatáčky kladný úhel α a pravé zatáčky záporný. Vyhodnocením tohoto úhlu matematickou funkcí signum a následným vynásobením změnou poloměru získám příspěvek k původnímu poloměru R. Mezivýsledky pro levý a pravý slot jsou popsány následujícími rovnicemi:

$$R_L = R - \text{sign}(\alpha) \cdot \triangle r, \tag{3.1}$$

$$R_R = R + \text{sign}(\alpha) \cdot \triangle r. \tag{3.2}$$

Druhá změna, kterou funkce provádí, je úprava poloměru podle toho, o kolik si zadní náprava autíčka zmenší poloměr obouku při průjezdu zatáčkou. To způsobuje vedení autíčka v přední části, zatímco zadní kolečka jsou volná. Výsledný poloměr, který zadní kolečka projedou, je popsán Pythagorovou větou

$$R_{newL,newR}{=}\sqrt{R_{L,R}^2 - c^2}, \tag{3.3}$$



Obrázek 3.1: Zmenšení poloměru oblouku zadní nápravy

kde c značí vzdálenost zadní nápravy od osy otáčení vodící lišty a celá situace je znázorněna na obrázku 3.1.

Dalším krokem po přepočítání poloměru zatáčky je výpočet délky dílu.

Délka dílů

Pro účel simulace je délka dílů velmi důležitým parametrem. Na základě znalosti tohoto údaje můžeme říct, na kterém dílu se autíčko zrovna nachází. Výpočet délky musí následovat až po funkci `choose_side`, která mění poloměr oblouku. Počítám ji pomocí funkce `part_length`, jejíž vstupem je vektor parametrů dílu a výstupem je délka dílu. Na začátku funkce rozhodne, zda je vstupem zatáčka či rovinka. Délka rovného dílu je rovnou souřadnice x. U zatáček musíme provést výpočet, kdy pro délku oblouku kruhové výseče o středovém úhlu α platí

$$l = R \cdot |\alpha|, \tag{3.4}$$

za předpokladu, že je úhel α v jednotkách radian.

Dokončení přípravy

Po tomto kroku uložím do matice P nové parametry, vypočítanou délku dílu a vzdálenost dílu od začátku dráhy. Po proběhnutí cyklu všemi díly se do posledního

řádku matice uloží na pozici parametru d délka celkové tratě, což je součet délek všech dílů. V tuto chvíli je matice P kompletní a nebude se dále měnit. Navrátím se proto o úroveň výše do funkce `init_sim`, kam vracím tuto matici.

Příprava dat pro simulaci je dokončena voláním funkce `baseWorkspace` s parametry P, počtem autíček na dráze a napětím dráhy, což jsou proměnné, které potřebuje schéma v Simulinku pro simulaci průjezdu autíčka dráhou. Snažil jsem se, aby se nikde nemusela kopírovat žádná čísla a aby tyto úkony spojené se sdílením dat se Simulinkem udělala inicializační funkce za uživatele. Proto jsem vytvořil tuto funkci, která vstupní parametry uloží do zvláštního MAT-souboru `Pfile.mat`, ze kterého následně vyčte tyto proměnné do hlavního workspace Matlabu. K tomuto úkolu jsem využil zabudovanou funkci `evalin`, která vykoná příkaz, v našem případě `load('Pfile.mat')`, v námi vybraném workspace, zde 'base' workspace. V parametrech bloků v Simulinku musí být správně nastaveny vstupní parametry, aby byly načteny do simulace. Oba soubory vytvořené při přípravě jsou nakonec smazány.

3.2 Validace dráhy

Druhým krokem přípravy simulace je vyhodnocení, zda je možné trať postavit z dostupných dílů. Kromě toho je simulátor schopný vyhodnotit pouze jízdu autíčka v ploše bez vyvýšenin na dráze, z čehož plyne, že se díly nesmí křížit. Z těchto požadavků plynou tři podmínky, které musí dráha pro simulaci splňovat:

1. Součet úhlů zatáček je v absolutní hodnotě 360°, přičemž pravé a levé zatáčky mají v tomto součtu opačné znaménko.

2. Dráha musí být uzavřená, tedy trať končí ve stejném bodě jako začala.

3. Díly dráhy se nepřekrývají a nekříží. To je nutné k tomu, aby byla dráha realizovatelná ve dvou dimenzionálním prostoru.

3.2.1 Podmínky 1 a 2

Ověřování první a druhé podmínky probíhá současně s přípravou dat ve funkci `init_track`, ve které využívají stejný for cyklus, jako jsem popsal v kapitole 3.1. Nesplnění první podmínky, ale splnění druhé je na obrázku 3.2. Před započetím for cyklu je inicializována proměnná `pos`, což je vektor s totožným tvarem, jako má vektor `par`. Počáteční hodnota vektoru je nulová. Každý průchod cyklu znamená přičtení souřadnic `par` nového dílu k vektoru `pos`, a to se děje ve funkci `nextPos`. Přičítání je popsáno v následujícím odstavci, protože se nejedná o pouhý součet.

Funkce `nextPos`

Jak už bylo zmíněno, trať začíná v bodě $pos_0 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$, ke kterému můžeme přičíst vektor $par_1 = \begin{bmatrix} \alpha_1 \\ x_1 \\ y_1 \\ R_1 \end{bmatrix}$ nesoucí informace o prvním dílu tratě bez jakékoliv úpravy, protože ho přičítáme k nulovému vektoru. Výsledkem jsou souřadnice koncového bodu prvního dílu. Nyní vezměme vektor $par_2 = \begin{bmatrix} \alpha_2 \\ x_2 \\ y_2 \\ R_2 \end{bmatrix}$, což je vektor reprezentující druhý díl tratě. Pokud bychom tento vektor pouze přičetli k předchozím dvěma vektorům, tak by došlo k chybnému výsledku, protože souřadnice x_2 a y_2 jsou vztaženy k soustavě souřadnic pozice pos_0, nikoliv k soustavě, která je určena koncem předchozího dílu. Řešením je rotace souřadnic nově přidávaného dílu do soustavy konce předchozího dílu. Toho dosáhneme vynásobením vektoru par_i rotační maticí

$$Rot(\beta_i) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\beta_i & -\sin\beta_i & 0 \\ 0 & \sin\beta_i & \cos\beta_i & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \tag{3.5}$$

jejímž parametrem je úhel β, což je součet úhlů α všech předchozích dílů, tedy

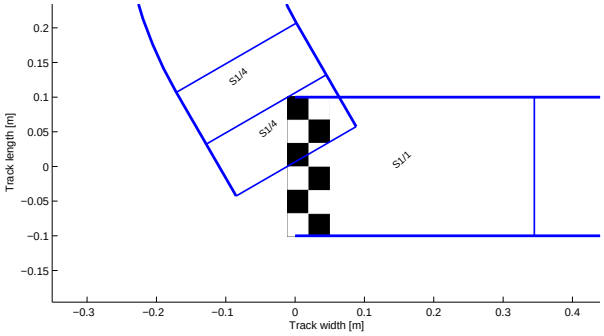
$$\beta_i = \sum_{j=1}^{i-1} \alpha_j. \tag{3.6}$$

Polohu středu konce i-tého dílu vypočítáme takto:

$$pos_i = pos_{i-1} + Rot(\beta_i) \cdot par_i. \tag{3.7}$$

Poslední nulový řádek eliminuje ve vektoru poloměr R_i, protože to je pro mě v tomto případě zbytečný údaj. Zde je místo, kde by docházelo při násobení nekonečnem nulou k výsledku NaN, kvůli kterému nelze poloměr reprezentovat nekonečnem.

Pokud je trať navržena správně, pak bude po dokončení algoritmu $|\beta| = 360°$ a $pos_0 = pos_{last}$.



Obrázek 3.2: Nesplnění podmínky 1

Při testování tohoto algoritmu jsem narazil na problém, kdy se koncový bod neshodoval s počátečním nebo součet úhlů nebyl přesně 360°. A přesto byla trať ve skutečnosti realizovatelná. Chyba se pohybovala řádově méně než 10^{-14} metru . Tento jev jsem přisoudil ztrátě přesnosti při výpočtu. Proto jsem se rozhodl zanést do algoritmu určitou toleranci v řádu centimetrů, která připouští odchýlení konce posledního dílu od začátku prvního při uzavření dráhy do okruhu. Tolerance je závislá na množství použitých dílů, protože spoje dílů umožňují jemně deformovat tvar dráhy. Podle zkušeností při sestavování různých drah jsem sestavil funkci

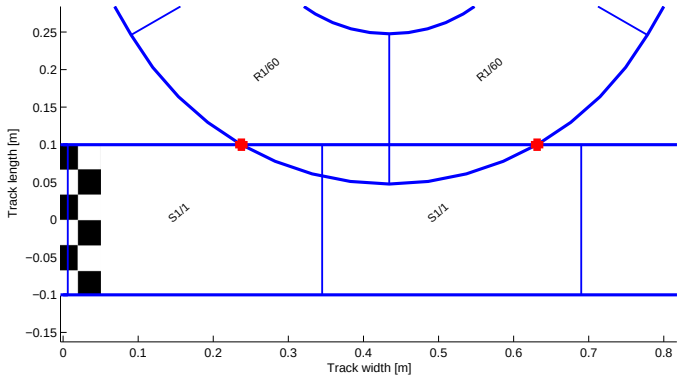
$$\text{tolerance} = \frac{\frac{\text{počet dílů}}{10} + 1,2}{100}. \tag{3.8}$$

Výsledkem je tolerance v metrech. Při 10 dílech je tolerance $2,2$ cm a při 30 dílech $4,2$ cm. To umožní postavit větší množství drah, jelikož je velice těžké sestavit dráhu z omezeného množství typů dílů, která by při validaci dráhu nepotřebovala. Toleranci na součet úhlů jsem zvolil přísnější, řádově kolem 10^{-3} rad. Vzhledem k délce dráhy jsou zanedbatelné, proto mají na výsledek simulace nepatrný vliv.

Je-li výsledek validace v těchto zvolených tolerancích, pak vrací funkce `init_track` příznačný bit 1 v proměnné `ret`. V opačném případě je vrácena nula. Kontrolu podmínky 3 provádím až v případě, kdy jsou splněny podmínky 1 a 2, čímž je inicializace více efektivní.

3.2.2 Podmínka 3

Jde o podmínku kontrolující křížení a překrývání dílů, ukázané na obrázku 3.3. Příprava probíhá také ve funkci `init_track` ve for cyklu. Princip je stejný jako u vytváření matice P, ale jsou zde dva rozdíly.



Obrázek 3.3: Překryt dílů

První je, že do připravených matic IL a IR se ukládají pouze čtyři parametry, které najdeme ve vektoru `par` každého dílu. Matice IL a IR definují parametry boční hranice dílu, jako P definuje střed dráhy. IL levou a IR pravou hranici.

Druhý rozdíl je ve funkci `choose_side,` která zde upravuje poloměr oblouku o $\triangle r = \frac{s}{2}$ a nepočítá s vybočením zadních koleček. Takto vytvořené matice jsou funkcí vráceny. Při splněných podmínkách 1 a 2 volám funkci `trackCross`. Ta vrací souřadnice překřížení dílů, pokud existují, a funguje následovně.

Nejprve jsou hranice vyhlazeny pomocí funkce `smooth_bound`, kterou jsem si vytvořil a popíši ve zvláštní kapitole. Z konce matic dále odebírám nejmenší počet vzorků, jejichž celková délka je stejná nebo větší, než je tolerance odchýlení konce posledního dílu od začátku prvního dílu. Důvod je předejití kolize i v případě, že by se počáteční a koncový díl křížily v rámci zadané tolerance.

Nakonec sloupce x a y upravených matic IL a IR vložím do funkce `intersections`, která vrací vektor souřadnic křížení jedné nebo dvou křivek. `Intersections` je volně dostupná na webové stránce Matlab Central v rubrice File exchange[2].

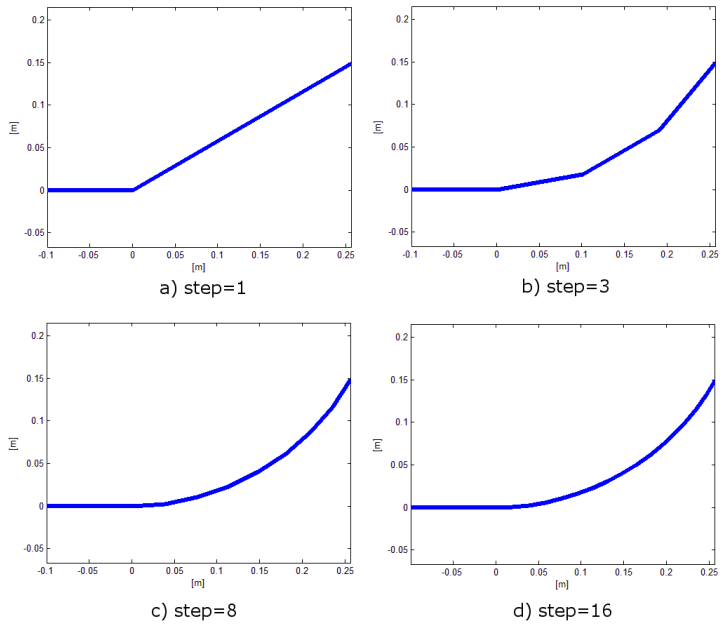
Funkci použiji třikrát. Pro každou z křivek IL a IR kontroluji, zda nekříží sama sebe a poté kontroluji křížení těchto křivek navzájem. Spojením souřadnic z těchto tří případů získávám vektor křížení všech křivek představujících ohraničení dílů. Prázdný vektor znamená, že se dráha nikde nekříží ani nepřekrývá a teprve tehdy jsou nahrány proměnné potřebné pro simulaci do hlavního workspace Matlabu.

3.3 Vyhlazování

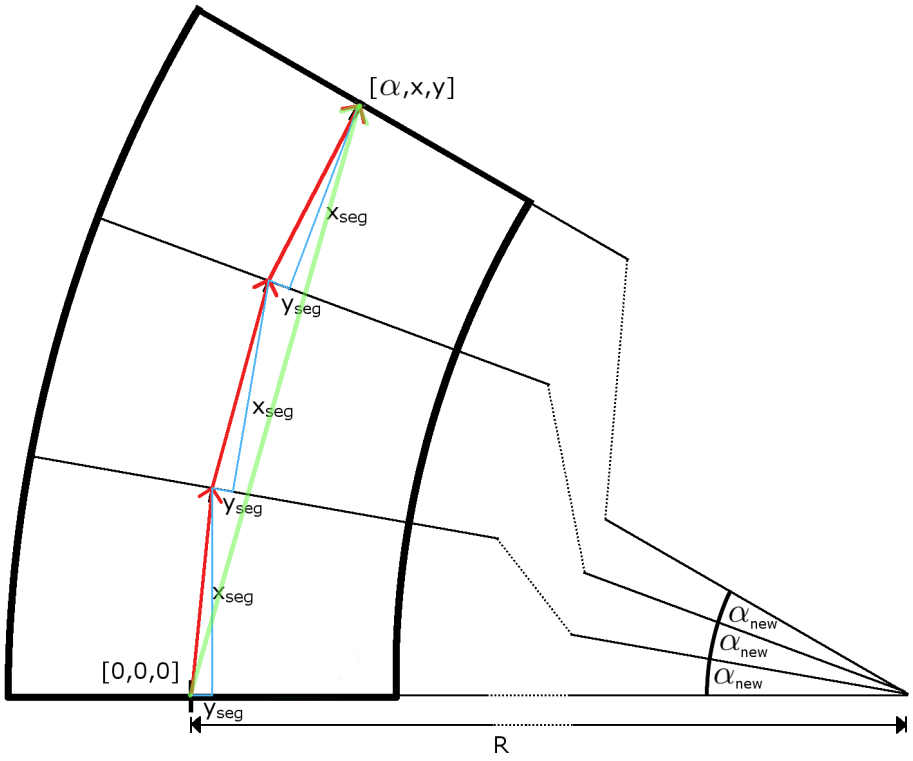
Vyhlazování dráhy je větší kapitola, protože se tento typ funkce objevuje v mém simulátoru vícekrát. Používám ho jako `smooth_bound` pro zjemnění okrajů dráhy při kontrole realizovatelnosti, objevuje se i v grafickém uživatelském rozhraní (GUI) pod názvem `smooth_part`, kde navyšuje uživatelskou přívětivost.

Aplikací funkce `nextPos` postupně na všechny díly dráhy získáme matici souřadnic spojů jednotlivých dílů. Vykreslením těchto bodů získáme obrázek, který často ani neukáže všechny zatáčky, a proto se s výsledkem nemůžeme spokojit.

Řešením je rozdělení každého dílu na několik stejných segmentů. Jejich následným spojením získáváme pseudo oblouk složený z úseček. Čím více segmentů vytvoříme, tím bude díl při zobrazení na obrazovce vypadat kulatěji. Nemá smysl vytvářet příliš velké množství segmentů kvůli zvyšující se výpočetní náročnosti. Volbou dělícího kroku chceme v první řadě dosáhnout při vykreslení na pohled hladké křivky. Dělící krok je vhodné volit větší než 6, ale na druhou stranu je zbytečné volit krok větší než 10, protože je vyhlazení už při této hodnotě dostačující. To je vidět na obrázku 3.4, na kterém je třikrát zobrazena nejdelší zatáčka z dostupných dílů, ale pokaždé s jiným počtem segmentů. Počet segmentů značím jako *step*.



Obrázek 3.4: Vyhlazení zatáček pro různou míru segmentace



Obrázek 3.5: Rozdělení dílů na segmenty. Červeně jsou vektory segmentů, zeleně je vektor celého dílu

Na začátku funkce je alokována matice o délce počtu dílů, které budeme zpracovávat, krát počet segmentů na jeden díl. Do této matice, kterou funkce vrací, budou na konci funkce uloženy souřadnice jednotlivých bodů.

Dále se podle parametru radius ve vektoru `par` rozhodnu, zda je daný díl rovinka nebo zatáčka. Vektor parametrů rovného dílu můžu rovnou vydělit počtem segmentů a získám dílčí segment. Tím vydělím pouze parametr x, který určuje délku rovinky, ostatní parametry jsou nulové.

U zatáček je nutné tyto souřadnice přepočítat pomocí goniometrických funkcí. Parametr úhlu zatáčky vypočítám jeho vydělením počtem segmentů

$$\alpha_{new} = \frac{\alpha}{step}. \tag{3.9}$$

Na obrázku 3.5 je vidět, že poloměr zůstává samozřejmě stejný. Zbylé parametry segmentu dopočítám. Pro levé zatáčky budou

$$par_{seg,L}= \begin{bmatrix} \alpha_{new} \\ R \cdot \sin|\alpha_{new}| \\ R - R \cdot \cos|\alpha_{new}| \\ R \end{bmatrix}. \tag{3.10}$$

Pravé zatáčky mají pouze opačné znaménko ve třetím řádku matice, neboť je zatáčka zahnuta na opačnou stranu,

$$par_{seg,R} = \begin{bmatrix} \alpha_{new} \\ R \cdot \sin|\alpha_{new}| \\ -R + R \cdot \cos|\alpha_{new}| \\ R \end{bmatrix}. \tag{3.11}$$

Nakonec z těchto parametrů vypočítám za pomoci funkce `nextPost` souřadnice jednotlivých segmentů. Postup použití funkce je stejný jako v případě evaluace dráhy, kde se počítaly souřadnice jednotlivých dílů. Segment je však zlomek dílu, proto se musí pomocí funkce `nextPos` přičíst k počáteční pozici tolikrát, na kolik segmentů je konkrétní díl rozdělen. Při každém přičtení pozice dalšího segmentu jsou souřadnice uloženy do matice. Sečtením všech segmentů dostáváme vždy původní vektor parametrů daného dílu.

4 Simulink

Simulace jízdy autíčka bude probíhat v prostředí Simulink, což je rozšíření Matlabu pro simulaci dynamických systémů. Pro tento program jsem se rozhodl, protože je na Fakultě elektrotechnické velmi používaný. Jeho obsluha je velmi jednoduchá. Základními prvky jsou symbolické bloky. Ty jsou spojeny virtuálními dráty, po kterých je předáván zpracovávaný signál. Cílem je vytvořit blok nebo systém bloků, jehož vstupem je PWM signál, kterým se řídí skutečné autíčko, a výstupem jsou měřené veličiny s odezvou odpovídající realitě. K začlenění dynamického systému autíčka do Simulinku použiji blok S-function.

4.1 S-function

Blok S-function, nebo-li systémová funkce, volá uživatelem vytvořenou S-funkci, což je dynamický systém implementovaný v počítačovém jazyce. Na výběr jsou

jazyky Matlab, C, C++, Ada a Fortran. Já si vybral jazyk Matlab, jenž poskytuje rozsáhlé knihovny a programování v něm je pohodlné. Tvar S-funkce má pevně danou syntaxi, která umožňuje spolupráci s řešičem rovnic v Simulinku a umožňuje simulovat jak spojité, tak i diskrétní a hybridní systémy. S-funkci lze pomocí řádku S-function parameters v bloku nastavit další parametry potřebné pro simulaci.

4.2 Schéma modelu

Při implementaci systému autíčka jedoucího po dráze je potřeba do funkce zanést všechny základní fyzikální vlivy, které působí na autíčko. Uvažoval jsem o dvou způsobech pojetí modelu.

První typ modelu, který jsem měl v počátku implementovaný, soustředil diferenciální rovnice modelu autíčka a program dráhy do jednoho bloku. Vstupem bloku byla posloupnost dílů dráhy *track* spolu s určením slotu, ve kterém autíčko jede, a počtem autíček na dráze. Tyto parametry byly výstupem grafického rozhraní (GUI). Výstupem bloku byly všechny zkoumané hodnoty.

Potřeba simulace více autíček na jedné dráze přinesla nápad rozdělit bloky na dva.

1. Blok obsahující model autíčka se vstupem PWM a zpětnou vazbou z dráhy. Výstupem je rychlost a poloha autíčka.

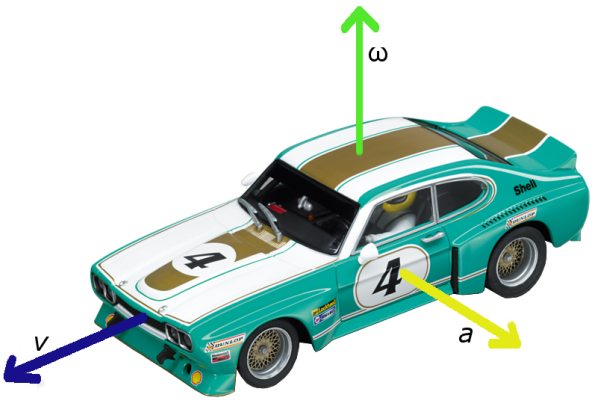
2. Blok dráhy se vstupy polohy a rychlosti, které generuje blok s modelem autíčka. Výstupem jsou zbylé měřené veličiny, boční zrychlení a úhlová rychlost otáčení kolem svislé osy.

Měřené veličiny jsou se svou orientací zakresleny na obrázku 4.1.

V prvním typu modelu probíhala validace a inicializace dráhy až při spuštění simulace. To se ukázalo jako nevýhodné, protože je velmi časté, že návrh nové dráhy není realizovatelný, a spuštění simulace v Simulinku není okamžité. Je proto efektivnější validovat trať v GUI a až poté v případě její realizovatelnosti předat potřebné parametry do hlavního workspace Matlabu, odkud si je získá schéma v Simulinku. Výhodou použitého modelu je také lepší přehlednost schématu a přístup k signálům

4.3 Bloček auta

Základním bločkem celého simulátoru je bloček představující fyzikální model autíčka. Jeho S-funkci jsem nazval `slotcar.m` a je poměrně jednoduchá. Jedná se



Obrázek 4.1: Měřené veličiny na autíčku

o nelineární model, který identifikoval Martin Lád. Pro simulaci jízdy po dráze libovolného tvaru je potřeba tento model upravit, respektive přidat fyzikální jevy, které vznikají při průjezdu zatáčkou a jež chceme v simulaci pozorovat. Vstupem bločku jsou střída PWM signálu a zpětná vazba z dráhy, což je boční zrychlení autíčka.

4.3.1 Úprava rovnice tření

Hlavním vlivem je tření vznikající mezi vodící lištou autíčka a slotem v dráze. Toto tření je popsané ve výše uvedené rovnici 0.2, ale počítá pouze s jízdou po rovném dílu. Při průjezdu zatáčkou působí odstředivá síla na vodící lištu autíčka a tím ji přitlačuje na stěnu slotu, čímž vzniká další třecí síla brzdící autíčko. Odstředivá síla je přímou reakcí na dostředivou sílu, která vznikla změnou směru autíčka v zatáčce. Třecí síla smykového tření je výsledkem součinu odstředivé normálové síly F_n a koeficientu tření b.

$$F_t = F_n \cdot b \tag{4.1}$$

Pro sílu F, působící na těleso o hmotnosti m, zrychlující rychlostí a platí:

$$F = m \cdot a. \tag{4.2}$$

m je hmotnost autíčka a a zde představuje boční zrychlení působící na autíčko, které můžeme vyjádřit jako změnu rychlosti za čas

$$a = \frac{\mathrm{d}v}{\mathrm{d}t} \tag{4.3}$$

a dále rozepsat

$$a = \frac{\mathrm{d}s}{\mathrm{d}t} \cdot \frac{\mathrm{d}v}{\mathrm{d}s} = \frac{\mathrm{d}s}{\mathrm{d}t} \cdot \frac{v}{R}, \tag{4.4}$$

protože

$$\frac{\mathrm{d}v}{\mathrm{d}s} = \frac{v}{R} \tag{4.5}$$

za předpokladu, že je středový úhel elementu dráhy s nekonečně malý. Dále do rovnice 4.4 dosadíme

$$\frac{\mathrm{d}s}{\mathrm{d}t} = \mathrm{v} \tag{4.6}$$

a zjistíme, že výsledek je

$$a = \frac{v^2}{R}, \tag{4.7}$$

proto bude výsledná síla F, přitlačující vodící lištu na stěnu slotu

$$F_n = m \cdot \frac{v^2}{R}. \tag{4.8}$$

Třecí sílu závislou na čase následně popíšeme takto

$$F_{t,odst}(t) = b_{odst} \cdot F(t) = b_{odst} \cdot m \cdot \frac{v^2(t)}{|R(t)|} = b_{odst} \cdot m \cdot |a(t)|. \tag{4.9}$$

Výsledek $F_{t,odst}$ je jedna ze složek třecí síly, která brzdí autíčko, a b_{odst} je koeficient tření, který jsem nalezl experimentálně a jeho hodnota je přibližně $b_{odst} = 0,08$. Tuto složku tření začlením do modelu přičtením do rovnice 0.2. Boční zrychlení je v absolutní hodnotě, protože je síla závislá pouze na velikosti odstředivé síly, nikoliv na směru.

Další změnou je závislost statického tření na rychlosti. Při dopředné rychlostí je výsledná třecí síla

$$F_t(v(t)) = bv(t) + F_t^s + b_{odst} \cdot |a(t)| \tag{4.10}$$

Pokud autíčko stojí, chceme mít výstupní rychlost také nulovou. Proto musí být hodnota statického tření $F_t^s = 0$, jinak autíčko začne couvat. Při jízdě vzad musí třecí síly působit proti směru jízdy, proto

$$F_t(v(t)) = bv(t) - F_t^s - b_{odst} \cdot |a(t)|. \tag{4.11}$$

Prvnímu členu tření, což je tření dynamické, otáčí znaménko hodnota rychlosti, která je při couvání záporná. Nesmíme zapomenout, že je vodící lišta autíčka v jeho přední části, kvůli čemuž se ve skutečnosti autíčko při couvání zkříží s drahou a vypadne, proto je couvání v simulátoru pouze teoretické.

4.3.2 Měření ujeté vzdálenosti

Kromě výstupní veličiny rychlosti, která je vypočítána diferenciální rovnicí 0.3 nelineárního modelu autíčka, potřebujeme měřit také ujetou vzdálenost, na základě které určíme přesnou polohu autíčka na dráze, respektive na jakém dílu se autíčko právě nachází. Okamžitá rychlost v se vypočítá podle rovnice 4.6, což je druhá diferenciální rovnice modelu, podle které je vypočítána ujetá vzdálenost s. Obě diferenciální rovnice jsou implementovány ve funkci `mdlDerivatives`.

Výstupem S-funkce autíčka je vektor $\begin{bmatrix} v \\ s \end{bmatrix}$.

4.3.3 Omezení vstupů

Maximální výkon motoru autíčka je dán fyzikálními vlastnostmi použitých součástek, což znamená potřebu zavést do modelu saturaci vstupu PWM, aby jeho hodnota nebyla větší, než je možné ve skutečném autíčku. Jízda vpřed je nastavována signálem v intervalu $(0; 1)$ a jízda vzad $(-1; 0)$. Nula znamená nulový výkon motoru.

Dalším omezením vstupu je zavedení mrtvého pásma PWM okolo nuly. Při testování se stávalo, že pokud výkon motoru nebyl schopný překonat tření autíčka, adaptivní krok simulace se příliš zkrátil a simulace trvala dlouho. Výstupem rychlosti byl pilovitý signál s malou amplitudou a vysokou frekvencí, což neodpovídá realitě. Proto jsem z rovnice 0.3 odvodil funkci počítající hranici mrtvého pásma PWM v závislosti na napětí na motoru autíčka. Pokud autíčko stojí, pak jsou dv, v a a nulové. Ze zbytku rovnice jsem vyjádřil střídu a dosadil číselné hodnoty za konstanty. Hranice mrtvého pásma PWM potom bude

$$D_{dead} = \frac{2,1858}{U_m}. \tag{4.12}$$

Pokud je vstupní PWM v intervalu $(-D_{dead}; D_{dead})$, pak PWM nastavím na nulu a to se stane pouze v případě, kdy je rychlost v řádech jednotkách milimetrů za sekundu. V praxi to znamená, že se autíčko rozjede až po překonání tření silou motoru, ale pokud je autíčko již v pohybu, může být malá rychlost udržována i PWM signálem, který je ve zmíněném intervalu mrtvého pásma.

4.4 Bloček dráha

S-funkce tohoto bločku je nazvána `tracksim.m`. Jeho úkolem je na základě pozice a rychlosti autíčka vypočítání hodnot bočního zrychlení a úhlové rychlosti, které

jsou způsobeny tvarem dráhy. Vstupem bločku jsou rychlost a ujetá vzdálenost autíčka nebo autíček. Výstupem jsou boční zrychlení a úhlová rychlost otáčení autíček kolem svislé osy.

4.4.1 Boční zrychlení

Boční zrychlení je výstupem jedné ze tří os akcelerometru autíčka. Ve skutečnosti je výstup senzoru velmi zarušený. Srovnání simulovaných hodnot se skutečností tedy nemá význam, pokud není vhodně vyfiltrován. Účelem simulace parametru může být navrhnutí úprav autíčka, jež by vedly ke zlepšení měření parametru.

Hodnota bočního zrychlení autíčka je vypočítána ve funkci `mdlOutputs` podle vztahu

$$a(\mathrm{t}) = \frac{v(t)^2}{R(t)}, \tag{4.13}$$

který je odvozen v sekci 4.3.1 zabývající se třením. Jednotky jsou $\mathrm{m} \cdot \mathrm{s}^{-2}$

4.4.2 Úhlová rychlost

Úhlová rychlost je další sledovaná hodnota. V autíčku ji čteme pomocí gyroskopu. V zásadě definujeme úhlovou rychlost ω jako

$$\omega = \frac{\mathrm{d}\varphi}{\mathrm{d}t}, \tag{4.14}$$

s tím, že se jedná o změnu úhlu za čas. Pro element oblouku $\mathrm{d}\varphi$ pak platí

$$\mathrm{d}\varphi = \frac{\mathrm{d}s}{R}, \tag{4.15}$$

což můžeme dosadit do předchozí dovnice a dostaneme

$$\omega = \frac{\frac{\mathrm{d}s}{R}}{\mathrm{d}t} = \frac{1}{R}\frac{\mathrm{d}s}{\mathrm{d}t}, \tag{4.16}$$

z čehož vyplývá výsledný vztah pro úhlovou rychlost závislé na čase

$$\omega(t) = \frac{v(t)}{R(t)}, \tag{4.17}$$

vypočítávaný vedle bočního zrychlení také ve funkci `mdlOutputs` v jednotkách $\mathrm{rad} \cdot \mathrm{s}^{-2}$.

4.4.3 Pozice autíčka na dráze

V obou simulovaných veličinách se objevuje rychlost a poloměr oblouku, po kterém autíčko právě jede. Funkce `radius` nalezne a vrátí poloměr. Poloměr je vlastností dílu dráhy, na kterém se autíčko v danou chvíli nachází, a proto je potřeba zjistit parametry dílu. V přípravě dat pro Simulinkové schéma byly parametry všech dílů uloženy do matice P, ze které je získám na základě ujeté vzdálenosti autíčka.

Základem funkce je for cyklus, který postupně prochází řádky matice P a porovnává ujetou vzdálenost s parametrem d_i, jenž představuje souřadnice spoje dílů, pokud bychom považovali dráhu za přímku. Pokud je nalezen díl dráhy, do jehož intervalu souřadnic patří ujetá vzdálenost, pak víme, na kterém dílu se autíčko právě nachází.

Z uvedených důvodů v sekci 2.2 nemohla být uložena hodnota poloměru nekonečno v případě rovných dílů. Musíme tedy při vracení poloměru rozlišovat rovinky a zatáčky. V případě rovinek vrací funkce hodnotu nekonečno. Když narazí na zatáčku, vrací hodnotu

$$poloměr = R \cdot \mathrm{sgn}(\alpha), \tag{4.18}$$

protože je R v matici absolutní hodnotou poloměru. Vynásobením poloměru znaménkovou funkcí parametru α, který je zavislý na straně zatáčky, je zajištěno, že hodnoty bočního zrychlení a úhlové rychlosti mají v pravé a levé zatáčce opačné znaménko. Po nalezení správného dílu je for cyklus ukončen, protože je jisté, že poloha autíčka nemůže ležet v žádném intervalu příslušícímu některému z dalších dílů.

Musím také zmínit úpravu ujeté vzdálenosti na začátku funkce `radius`. Aby hodnota ujeté vzdálenosti byla vždy v intervalu od začátku do konce dráhy, vydělím ujetou vzdálenost délkou dráhy a se zbytkem dělení ve funkci pracuji. To umožní simulaci obkroužit dráhu autíčkem více než jednou.

Popsaný algoritmus je jednoduchý a může se zdát neefektivní. Ideálně by si uměl algoritmus zapamatovat, na kterém dílu se autíčko nachází a v příštím volání funkce by pouze zkontroloval, zda je autíčko stále na stejném dílu a v opačném případě přešel o jeden díl vpřed nebo vzad. Problém je, do jaké datové struktury tuto informaci uložit. Do lokální proměnné to možné není, neboť je celá S-funkce vždy volána v každém časovém kroku simulace, čímž by se hodnota ztratila. Druhou možností je uložení do proměnné v souboru. Je rychlejší procházet celou matici P, protože přístup do souboru je pomalý.

4.4.4 Více autíček na jedné dráze

Další vlastností popisovaného simulátoru je podpora jízdy více autíček na jedné dráze. V současnosti na sebe autíčka jedoucí po jedné dráze nemají žádný vliv. V budoucnu je možné implementovat interakci autíček například pro návrh regulátorů vzdálenosti. Počet autíček je dán vstupním parametrem bloku a jejich počet musí být ve schématu v Simulinku dodržen. Následující schéma ukazuje, jak jsou hodnoty seřazené ve vstupním i výstupním vektoru

$$\begin{bmatrix} v_1 \\ s_1 \\ v_2 \\ s_2 \\ \cdot \\ \cdot \\ v_n \\ s_n \end{bmatrix} \Longrightarrow \texttt{tracksim} \Longrightarrow \begin{bmatrix} a_1 \\ \omega_1 \\ a_2 \\ \omega_2 \\ \cdot \\ \cdot \\ a_n \\ \omega_n \end{bmatrix},$$

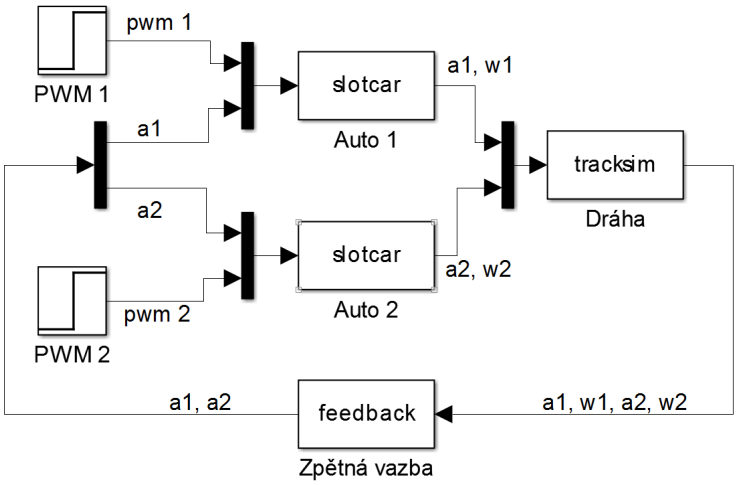
kde index n představuje počet autíček. Pro každé autíčko, kterému patří vždy dvojice vektorů na vstupu, jsou vypočítány výstupní hodnoty, které jsou zase seřazeny v pořadí autíček na vstupu.

4.5 Zpětná vazba

S-funkce zpětné vazby je nazvána `feedback.m`. Blok funguje tak, že z vektoru vstupního signálu, což je výstupní signál bločku `tracksim`, vybírá jen zpětnovazební signály, které bloky `slotcar` potřebují. Z kapitoly 3.3 víme, že se jedná o signál bočního zrychlení, jenž potřebujeme k výpočtu tření. Převod vstupu bloku na jeho výstup lze napsat názorně takto:

$$\begin{bmatrix} a_1 \\ \omega_1 \\ a_2 \\ \omega_2 \\ \cdot \\ \cdot \\ a_n \\ \omega_n \end{bmatrix} \Longrightarrow \texttt{feedback} \Longrightarrow \begin{bmatrix} a_1 \\ a_2 \\ \cdot \\ \cdot \\ a_n \end{bmatrix}.$$

Na obrázku 4.2 je názorně vidět zapojení schématu v Simulinku pro dvě auta.



Obrázek 4.2: Zapojení schéma v Simulinku pro dvě auta
Značení signálů: pwm - střída pwm signálu, v - rychlost, s - ujetá vzdálenost, a - boční zrychlení, w - úhlová rychlost. Čísla u signálů označují příslušnost k určitému autíčku.

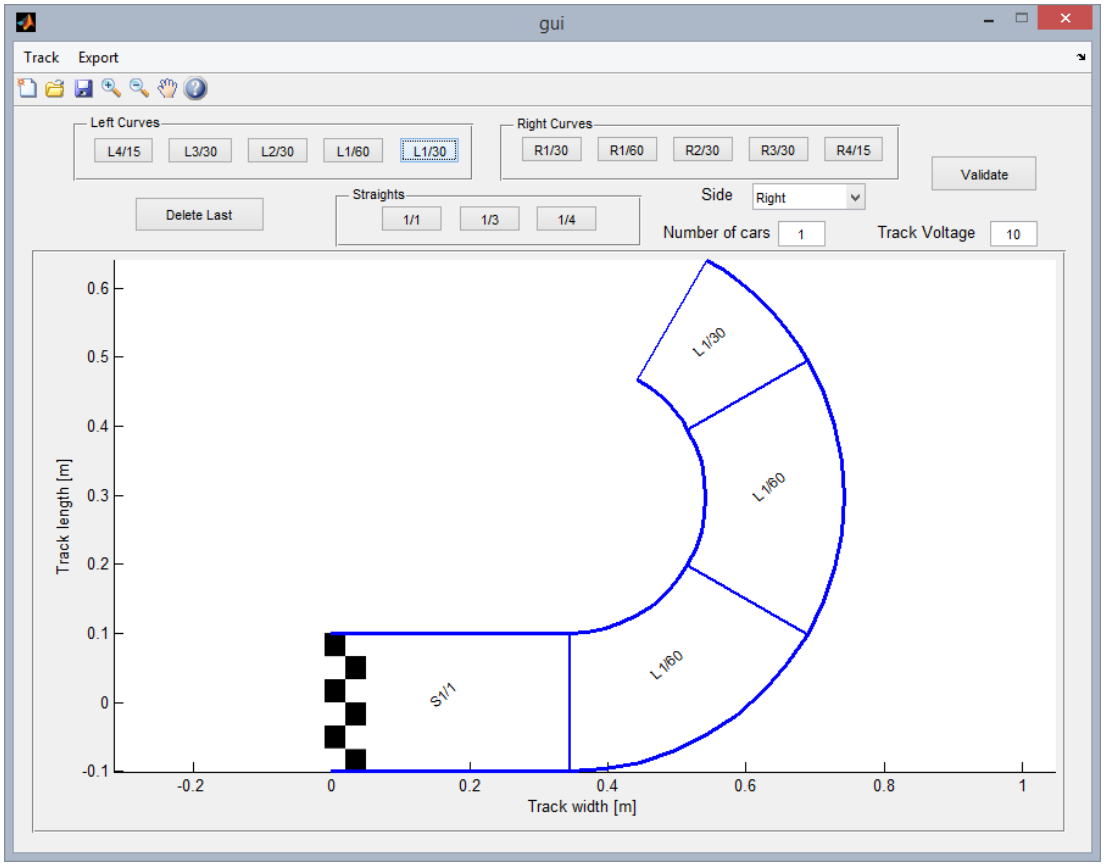
5 Grafické uživatelské rozhraní (GUI)

Součástí Simulátoru je grafické uživatelské rozhraní, nebo-li GUI. Slouží hlavně k jednoduchému vytváření tratí, které chceme simulovat. Dále umožňuje ukládání a otevírání návrhů tratě a exportování do obrazových formátů.

Návrh GUI jsem vytvořil v Matlabu pomocí prostředí Guide. Je to velice intuitivní nástroj, kde jsem zvolil velikost budoucího okna GUI a umístil do něj potřebné grafické prvky. Po rozložení prvků stačí navrhnuté okno uložit do souboru typu figure a Guide automaticky vygeneruje M-soubor, který obsahuje funkce volané při interakci s jednotlivými prvky GUI. Výhodou prostředí Guide je možnost úpravy okna, při které současně probíhá aktualizace M-souboru.

5.1 Popis rozložení prvků

Největší část okna GUI zabírá plocha pro zobrazení návrhu dráhy. Začátek dráhy je označen šachovnicovou čarou. Při návrhu dráhy je třeba brát v potaz, že se autíčko rozjíždí vždy doprava od této čáry. Pro představu o velikosti dráhy je plocha popsána osami s měřítkem v metrech. Obrázek je vytvořen stejně jako graf v Matlabu.



Obrázek 5.1: Screenshot GUI

Nad plochou nalezneme tlačítka pro vlastní navrhování dráhy. Přidání dílu se vykoná stisknutím tlačítka označeného kódem konkrétního dílu. Pro lepší přehlednost jsou tlačítka pro přidání rozdělena do tří rámečků na levé a pravé zatáčky a na rovinky. Na levé straně se nachází tlačítko Delete Last, které slouží pro vymazání posledního dílu dráhy. Napravo je tlačítko Validate, jehož stisknutím se spustí funkce `init_sim`, dále rozbalovací menu pro volbu slotu, pole pro zadání počtu simulovanách autíček a pole pro zadání napětí na dráze. Úspěšnost nebo neúspěšnost validace je indikována vyskakovacím oknem.

V horní části okna je lišta s nabídkou funkcí. Další funkce, včetně přiblížení obrázku nebo nápovědy, lze také volat kliknutím na ikonu v liště pod nabídkou. Pro představu je na obrázku 5.1 zobrazen screenshot GUI.

5.2 Inicializace GUI

Při spuštění je volána implicitní funkce `gui_OpeningFcn`, ve které jsou inicializovány téměř všechny proměnné využívané v programu. V první řadě to jsou matice popisující střed a hranice tratě v nevyhlazené a vyhlazené verzi, proměnná s počtem dílů, pole kódů dílů a počet segmentů na díl. Všechny proměnné inicializované v této funkci ukládám do objektu `handles`, který je předáván napříč všemi funkcemi GUI. Aktualizace nebo zapisování proměnné do objektu `handles` musí být následováno voláním funkce `guidata`, provádějící zápis proměnných do objektu GUI.

5.3 Popsání funkcí GUI

V této části popíši funkci jednotlivých prvků GUI.

5.3.1 Přidání dílu

Nejpoužívanější funkcí v programu je funkce `addPart`. Tuto funkci vždy volají funkce obsluhující činnost přidávacích tlačítek. Parametrem *part* je vždy kód přidávaného dílu. Jejím úkolem je zajistit přidání parametrů nového dílu do matic reprezentujících dráhu.

Nejprve jsou ze struktury `handles` načteny aktuální hodnoty proměnných a je inkrementována hodnota počtu dílů $i{=}i+1$.

Poté jsou načteny souřadnice konce středu a hranic předchozího dílu. Do proměnné *track* je na pozici i přidán kód dílu *part*, na jehož základě je funkcí `gen_part` vygenerován vektor *par*, který je po úpravách poloměru a vyhlazení přidán přes funkci nextPos do matic reprezentující střed a hranice dílů. Nakonec nahraji aktualizované proměnné zpět do `handles` a aktualizují obrázek funkcí `reinitPic`.

5.3.2 Odebrání dílů

Odebírání posledního dílu

Funkce `delLast` realizuje odebrání posledního dílu. Ze všech proměnných, do kterých jsou ve funkci `addPart` přidána data reprezentující díl, jsou parametry posledního dílu v této funkci nahrazena nulami. To proběhne v případě, že dráha obsahuje alespoň jeden díl. Je nutné odečíst díl z proměnné i.

Odebírání všech dílů

Pokud chceme vymazat všechny díly, například v případě vytvoření nového návrhu, volá se funkce `deleteAll`. Jsou zde vynulovány všechny matice se souřadnicemi, pole *track* a nula je přiřazena také do i. V obou funkcích odebírajících díly je na konci aktualizován obrázek.

5.3.3 Ukládání návrhu

V případě opakovaného použití jedné dráhy, nebo pokud pracujeme na větší trati a chceme ji dokončit jindy, poskytuje GUI funkci ukládání `saveTrack`. Kliknutím na ikonu diskety nebo kliknutím na Save v nabídce Track je vyvoláno okno, ve kterém vybereme umístění pro uložení a název souboru. Okno je otevřeno zabudouvanou funkcí Matlabu `uiputfile`, která vrací string s umístěním a názvem souboru po kliknnutí na tlačítko Uložit a prázdný řetězec po kliknutí na tlačítko Storno. Pokud název souboru a cesta k němu nejsou prázdné, uložím posloupnost dílů *track* do zvoleného souboru pomocí funkce `save`.

5.3.4 Otevírání návrhu

K otevření uloženého návrhu slouží funkce `open`. Na začátku je zobrazeno okno, které je podobně jako při ukládání otevřeno zabudouvanou funkcí Matlabu `uigetfile`, kde zvolíme požadovaný soubor a potvrdíme volbu tlačítkem Otevřít. Návratovou hodnotou je umístění a názvu souboru. V případě, že soubor obsahuje proměnnou *track*, vyskočí dialogové okno, zda chceme zahodit současný návrh, protože otevřením dráhy je přepsána dráha stávající. Až se souhlasem uživatele je stávající dráha vymazána a nahrazena novou. Načítání probíhá postupným přidáváním dílů funkcí `addPart`.

5.3.5 Výběr slotu, nastavení počtu autíček a napětí dráhy

O tom, zda bude simulovaná jízda autíčka probíhat v pravém nebo levém slotu dráhy, rozhodneme volbou Right nebo Left (pravý nebo levý) v rozbalovací nabídce Side. Výchozí hodnota je Right. Při změně je upravena hodnota proměnné *side* v objektu `handles`.

Počet autíček se zadává do kolonky Number of cars, jejíž výchozí hodnota je 1, a je uložen do proměnné *carNumber*. Je zde kontrolováno, aby byla hodnota přirozené číslo, protože jinak by došlo k selhání validace. Dodržení počtu autíček ve schématu v Simulinku je pro běh simulace též důležité.

Do kolonky Track Voltage se zadává napětí dráhy. Výchozí hodnotou je 10 V a musí být zadáno kladné číslo, které je následně uloženo v proměnné $trVolt$. Je vhodné zvolit napětí, které zvládá eletronika autíčka. Jmenovité napětí standartně dodávaného zdroje k autíčku je 12 V, proto doporučuji napětí blízké této hodnotě.

5.3.6 Aktualizace obrázku

Při každé změně tvaru dráhy je nutné aktualizovat její obrázek, což je vykonáno voláním funkce `reinitPic`. Původní obrázek je nejprve smazán, protože pouhá úprava obrázku na základě změn dráhy nefungovala vždy správně, a poté je načten obrázek startovní čáry. Pak jsou z matic IL a IR vykresleny okraje dráhy, které jsou doplněny o příčné čáry dělící jednotlivé díly a vykreslené ze souřadnic z matic GIL a GIR. Identifikace jednotlivých dílů v obrázku je zajištěna přidáním kódu dílu. Osy jsou nastaveny na stejné měřítko, aby nedocházelo k deformaci dráhy.

5.3.7 Validace

Chceme-li navrženou dráhu simulovat, pak stiskneme tlačítko Validate nebo klikneme na Validate v nabídce Track. Z `handles` jsou načteny proměnné *side*, *carNumber* a *track* a jsou předány funkci `init_sim`. V případě křížení dílů jsou souřadnice těchto bodů přijaty a vykresleny do obrázku.

5.3.8 Přiblížení obrázku

Pokud navrhujeme velkou dráhu a chceme si zobrazit její detail, pak se hodí funkce přiblížení obrázku. K tomu slouží ikonky v panelu nástrojů Lupa+ a Lupa-. Zpět do původního zobrazení celé dráhy se vrátíme kliknutím pravým tlačítkem do obrázku a vybráním Reset To Original View. Po přiblížení lze vybráním nástroje ruka obrázkem libovolně pohybovat, abychom si mohli zobrazit jinou část dráhy. Tyto funkce jsou zabudovány v Matlabu, proto je nebylo třeba implementovat.

5.3.9 Export dráhy

Při provádění simulací a následné analýze dat může být potřeba uložit obrázek dráhy. Za tímto účelem jsem GUI vybavil funkcí `exportTo`, která na základě typu, do kterého chceme exportovat vytvoří požadovaný soubor. Exportovat lze do PNG, PDF nebo do vektorového formátu EPS. Po kliknutí na jeden z těchto formátů v nabídce Export je zobrazeno okno, ve kterém je vybráno umístění ex-

portovaného souboru a stisknutím tlačítka Uložit je uložen obrázek dráhy. Uloženo je vždy aktuální zobrazení dráhy na obrázku, ne celá dráha.

5.3.10 Nápověda

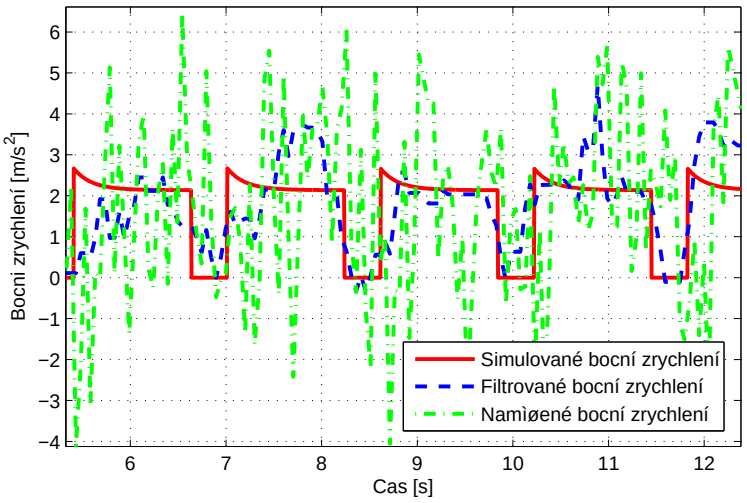
Program obsahuje nápovědu, která je otevřena kliknutím na ikonu otazník  . Otevře se nové okno, jehož obsahem je textový soubor `help.txt`, ve kterém jsou popsány funkce a ovládání GUI.

Část IV
Porovnání simulátoru se skutečností

Ověření simulátoru proběhlo na třech různých dráhách. Dvě složitější dráhy byly navrženy v GUI simulátoru a s úspěchem sestaveny v laboratoři. Vytváření nebylo jednoduché, protože je v laboratoři pouze omezené množství dílů typu 1/60, 2/30, 4/15 a rovinka 1/1. Jmenovité napětí napájecího adaptéru je 12 V, ale skutečné napětí dráhy je 12.34 V.

Porovnávat budu rychlost autíčka a úhlovou rychlost otáčení kolem svislé osy. Data naměřená akcelerometrem jsou velice zarušená, a proto je v porovnání vynechám. Podařilo se mi je částečně vyfiltrovat, ale výsledek stále nedosahuje kvality, jako data úhlové rychlosti. Obrázek 5.2 porovnává naměřená data akcelerometru vyfiltrovaná a nevyfiltrovaná s hodnotami ze simulátoru na oválné dráze. Na složitější dráze s rychlejšími změnami směru je výsledek filtrace ještě horší.

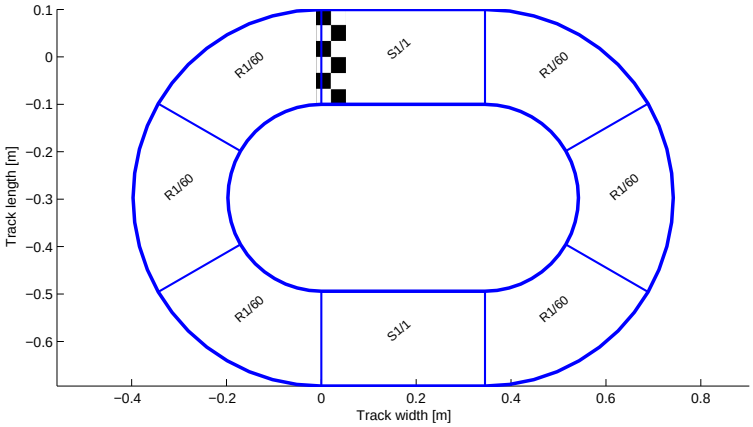
Filtrování provádím tak, že vezmu n naměřených vzorků, seřadím je podle velikosti a zahodím k maximálních a k minimálních hodnot. Podmínkou je nerovnice $n > 2k$. Tím se zbavím extrémních kmitů, které zřejmě způsobují jemné vibrace autíčka. Ze zbylých $n - 2k$ hodnot spočítám aritmetický průměr. Toto filtrování způsobuje zpoždění o $k/2$ vzorků, proto musí být data pro porovnání posunuta. To může být problematické při použití akcelerometru v reálném čase. Já jsem použil parametry filtru $n = 10$ a k=4. Vysoká hodnota k dokazuje velký šum.



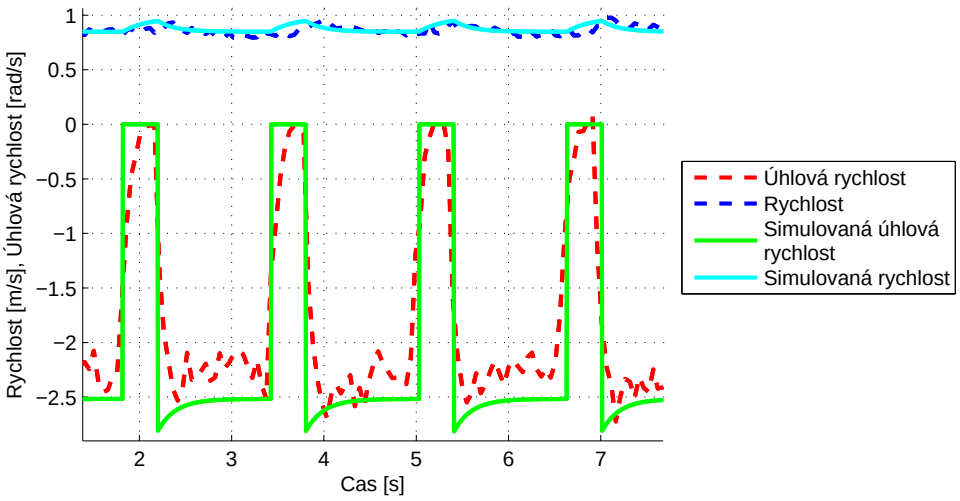
Obrázek 5.2: Data naměřená akcelerometrem a jejich filtrace

Dráha 1

První simulovaná dráha je jednoduchý ovál složený ze zatáček 1/60 a rovinek 1/1. Obrázek dráhy exportované z GUI je na obrázku 5.3.



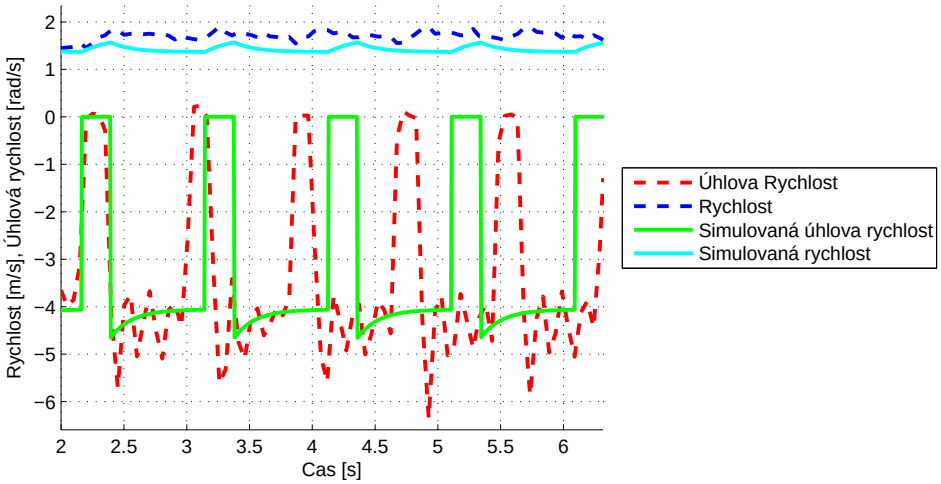
Obrázek 5.3: Tvar dráhy 1



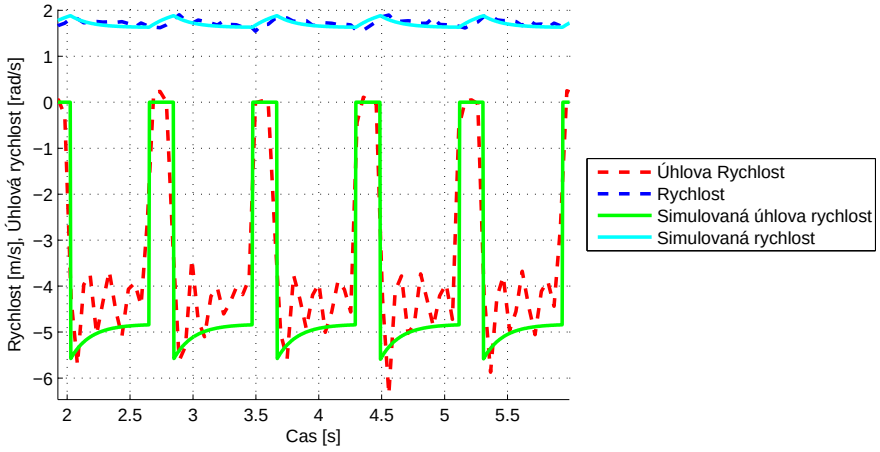
Obrázek 5.4: Ověření simulátoru na dráze 1. PWM= 0, 4

Nejprve porovnám jízdu autíčka ustálenou rychlostí se vstupem PWM= 0, 4. Při tomto vstupu by měla být ustálená rychlost autíčka 0, 98m/s. Na obrázku 5.4 je vidět vliv třecí síly brzdící autíčko v zatáčce. Hodnota úhlové rychlosti ve špičkách odpovídá teoretické hodnotě, ale nejsou příliš zřetelná zpomalení na začátku každé zatáčky.

Dále jsem na stejné dráze testoval autíčko s PWM= 0, 6. Při vyšší rychlosti už autíčko vylétávalo v prudkých zatáčkách z dráhy. Velký rozdíl mezi skutečností a simulací jsem zaznamenal v rychlosti autíčka při PWM větším, než 0, 4. Teoretická rychlost podle modelu na rovných dílech je 1, 7 m/s. Ve skutečnosti autíčko dosahovalo na rovince rychlosti 1, 85 m/s. V obrázku 5.5 je vidět, že je skutečné autíčko znatelně rychlejší. To způsobí projetí kola za kratší čas a také zvýšení hodnot úhlové rychlosti. Na dalším obrázku 5.6 je zvýšena hodnota PWM na 0, 7, při které jede simulované autíčko stejně rychle jako skutečné s PWM 0, 6. Zlepšila se synchronizace průjezdu zatáček, pokles rychlosti v zatáčkách i hodnoty úhlové rychlosti. Tuto skutečnost přisuzuji stáří autíčka, které již nemusí mít vlastnosti jako v době, kdy byl model vytvořen.



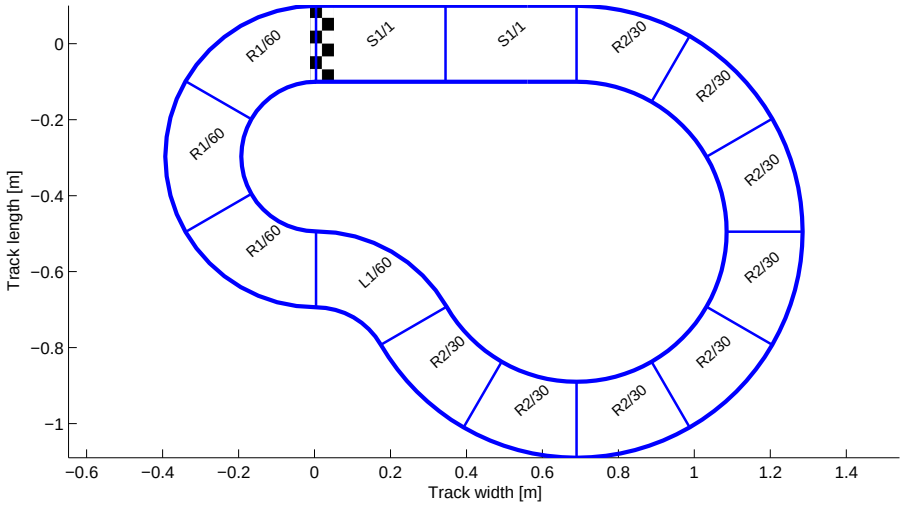
Obrázek 5.5: Ověření simulátoru na dráze 1. PWM= $0,6$



Obrázek 5.6: Ověření simulátoru na dráze 1. Korekce PWM v simulaci na $0,7$, PWM skutečného autíčka $0,6$

Dráha 2

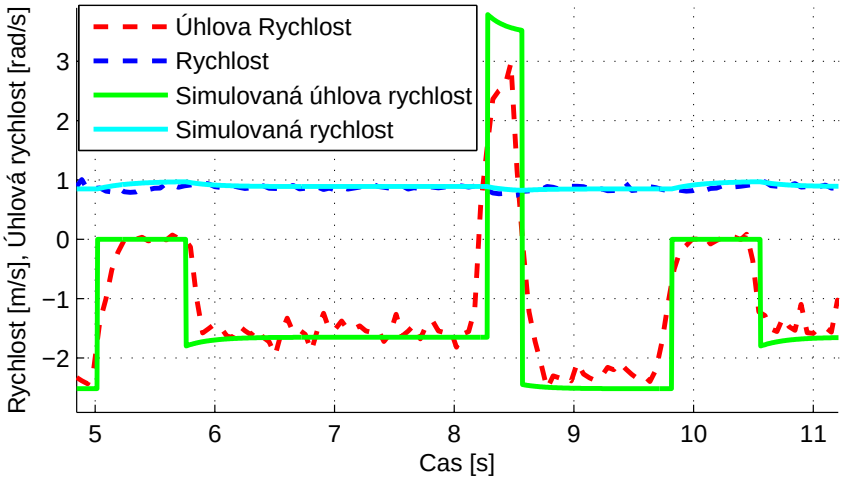
Druhá testovaná dráha má přibližně tvar trojúhelníku. Její přesný tvar je na obrázku 5.7. Při střídě PWM= 0,4 průběh simulované rychlosti i úhlové rychlosti hezky odpovídá průjezdu skutečného autíčka (obrázek 5.8). Při vyšším PWM=0,6 byla i zde rychlost simulovaného autíčka nižší, proto rovnou na obrázku 5.9 porovnávám simulaci s PWM= 0,7 se skutečným autíčkem, kterému jsem nastavil PWM= 0,6.



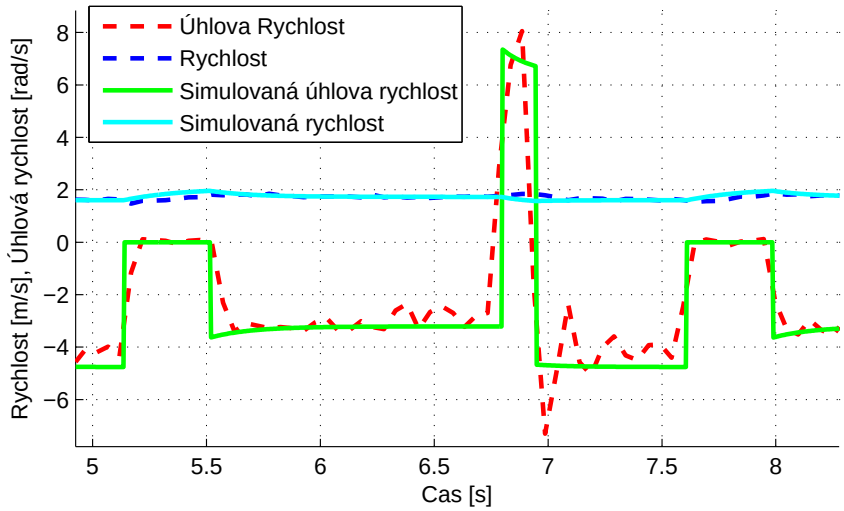
Obrázek 5.7: Tvar dráhy 2

Dráha 3

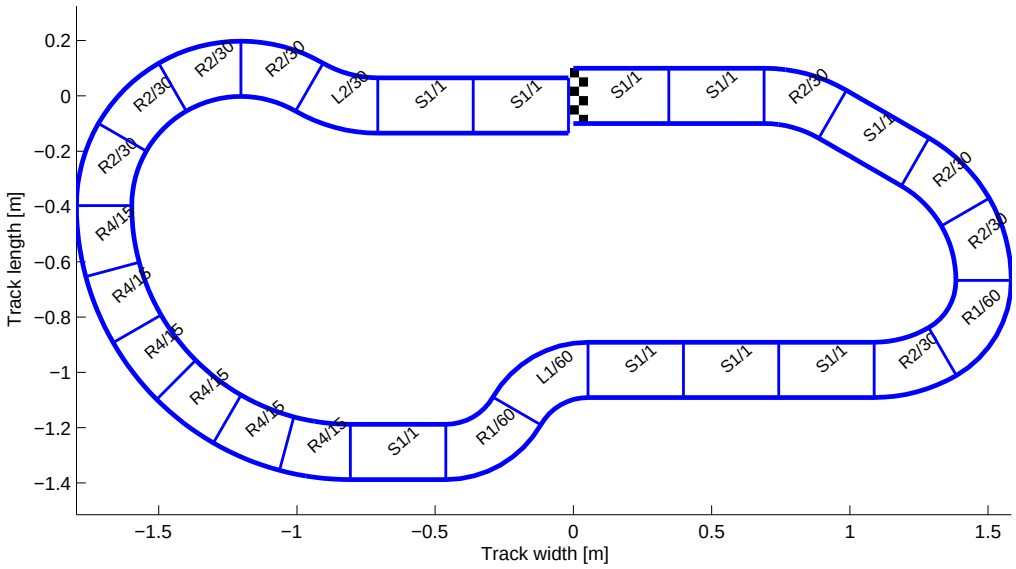
Tvar poslední testované dráhy je na obrázku 5.10 a navrhoval jsem ji s důrazem na využití všech typů dílů v laboratoři. Na setkání začátku a konce dráhy v cílové čáře je hezky vidět, jak při validaci funguje tolerance. Teoreticky začátek a konec dráhy nesedí o 3,5 cm, ale je ve skutečnosti sestavitelná bez problémů. Průjezd je zaznamenán na obrázku 5.11. Při hodnotě PWM= 0,6 již autíčko vylétlo z dráhy, proto není průjezd změřen. Vylétnutí bylo způsobeno smykem v šikaně ve spodní části dráhy.



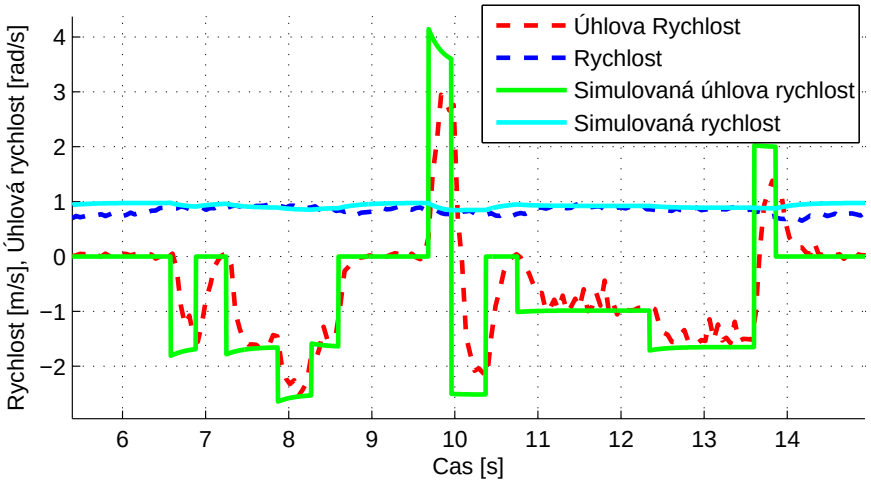
Obrázek 5.8: Ověření simulátoru na dráze 2. PWM$= 0,4$



Obrázek 5.9: Ověření simulátoru na dráze 2. Korekce PWM v simulaci na $0,7$, PWM skutečného autíčka $0,6$



Obrázek 5.10: Tvar dráhy 3



Obrázek 5.11: Ověření simulátoru na dráze 3. PWM= 0, 4

Část V
Závěr

Prvním řešeným úkolem práce byla úprava software autíčka a rozhraní v počítači pro co nejrychlejší vyčítání měřených dat. Výsledná stabilní rychlost je 20 vzorků za sekundu. Při vyšších rychlostech se data ztrácela a jejich rozložení v čase nebylo rovnoměrné.

Druhým úkolem bylo vytvořit v počítači reprezentaci jednotlivých dílů a následně celé dráhy. Změřil jsem údaje dostupných dílů a z nich vytvořil vektor parametrů jednoznačně určující jak rovné, tak zatáčkové díly. Vektor se skládá z poloměru středu dílu, středového úhlu oblouku a souřadnic konce dílu vůči počátku.

Ve třetí sekci je rozebrána inicializace dráhy. Dělí se na přípravu dat, se kterými pracuje schéma v Simulinku, a na validaci dráhy. Validace kontroluje tři podmínky a ukazuje se, že splnění všech je nutné k vytvoření skutečné dráhy. Při validaci bylo třeba zavést toleranci těchto podmínek hlavně kvůli chybě zaokrouhlování při přidávání dílů. Tolerance také umožňuje navrhnutí a simulaci více tvarů tratí, protože ve skutečnosti je možné spoje dílů lehce deformovat. Přidávání dílů za sebe je prováděno rotací původních souřadnic do soustavy souřadnic konce předchozího dílu, což zajišťuje jejich návaznost. V programu je velmi využíváno vyhlazování dílu, tedy rozdělení dílu na mnoho malých segmentů k získání jeho přesnější prostorové reprezentace. Ukázalo se, že bohatě stačí 8-10 segmentů.

Další část se zabývá implementací modelu do S-funkcí, jež se používají jako bloky schématu v Simulinku. Vytvořil jsem tři bloky: auto, dráhu a zpětnou vazbu. Rozdělení modelu do více bloků je přehlednější a umožňuje lepší úpravy simulátoru. Blok auto má výstupy rychlost a ujetou vzdálenost, jako vstup je zavedena střída PWM a boční zrychlení, na jehož základě je vypočítána nová třecí síla. Na testovacích datech se ukázal nezanedbatelný rozdíl v rychlosti mezi skutečným autíčkem a matematickým modelem. Boční zrychlení a úhlová rychlost otáčení autíčka jsou výstupy bloku dráhy. Simulátor neumí poskytovat informaci o vylétnutí autíčka z dráhy, neboť je tento jev kromě bočního zrychlení a úhlové rychlosti též závislý na posloupnosti dílů. Poslední blok plní úkol separátoru bočního zrychlení z výstupu dráhy. Simulátor podporuje jízdu více autíček, které na sebe vzájemně nereagují. Přestože vydavatel Matlabu doporučuje lépe vybavené S-funkce Level-2, použil jsem S-funkce Level-1, jelikož jsem se o této možnosti nedozvěděl včas. Nabízí se možnost přeprogramování S-funkcí do nové verze. Nicméně ve verzi Level-1 jsem neshledal pro tuto práci žádné nedostatky.

Poslední část práce je věnována návrhu uživatelského prostředí pro pohodlné vytváření různých tvarů dráhy jak pro simulátor, tak i pro stavbu skutečné dráhy. Instrukce ke spuštění simulátoru jsou v příloze B.

Literatura

[1] Lád, Martin: Návrh a implementace řídícího systému pro autodráhové vozidlo. 2014. `http://support.dce.felk.cvut.cz/mediawiki/images/a/a4/Bp_2014_lad_martin.pdf` (cit. 17.12.2014).

[2] Schwarz, Douglas M.: Fast and Robust Curve Intersections. 2010. `http://www.mathworks.com/matlabcentral/fileexchange/11837-fast-and-robust-curve-intersections` (cit. 23.3.2015)

[3] Tewari, Ashish: Modern Control Design With MATLAB and SIMULINK. Wiley. 2002. ISBN: 978-0471496793

[4] Herout, Pavel: Učebnice jazyka C. KOPP. 2009. ISBN: 9788072323838

[5] Bednařík, Michal: Fyzika 1. České vysoké učení technické v Praze. 2012. ISBN: 978-80-01-04834-4

Přílohy

A Obsah přiloženého CD

CD nalepené na vnitřní zadní obálce práce obsahuje text bakalářské práce a složku ***sim*** s programem simulátoru.

Obsah jednotlivých adresářů:

sim/

gui.m spouštěcí soubor uživatelského rozhraní

onecar.slx Simulinkové schéma pro simulaci jednoho autíčka

twocar.slx Simulinkové schéma pro simulaci dvou autíček

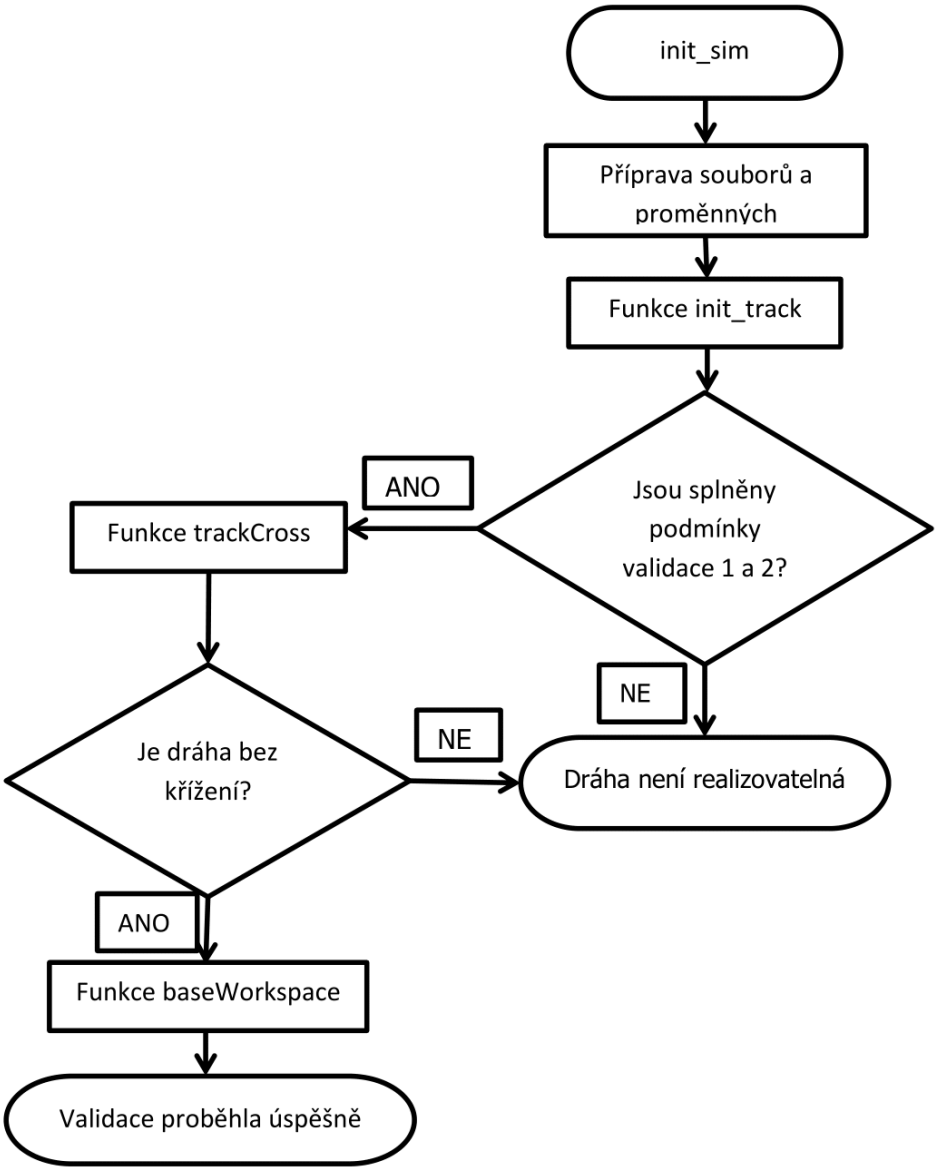
sim/simfiles/ S-funkce a pomocné soubory

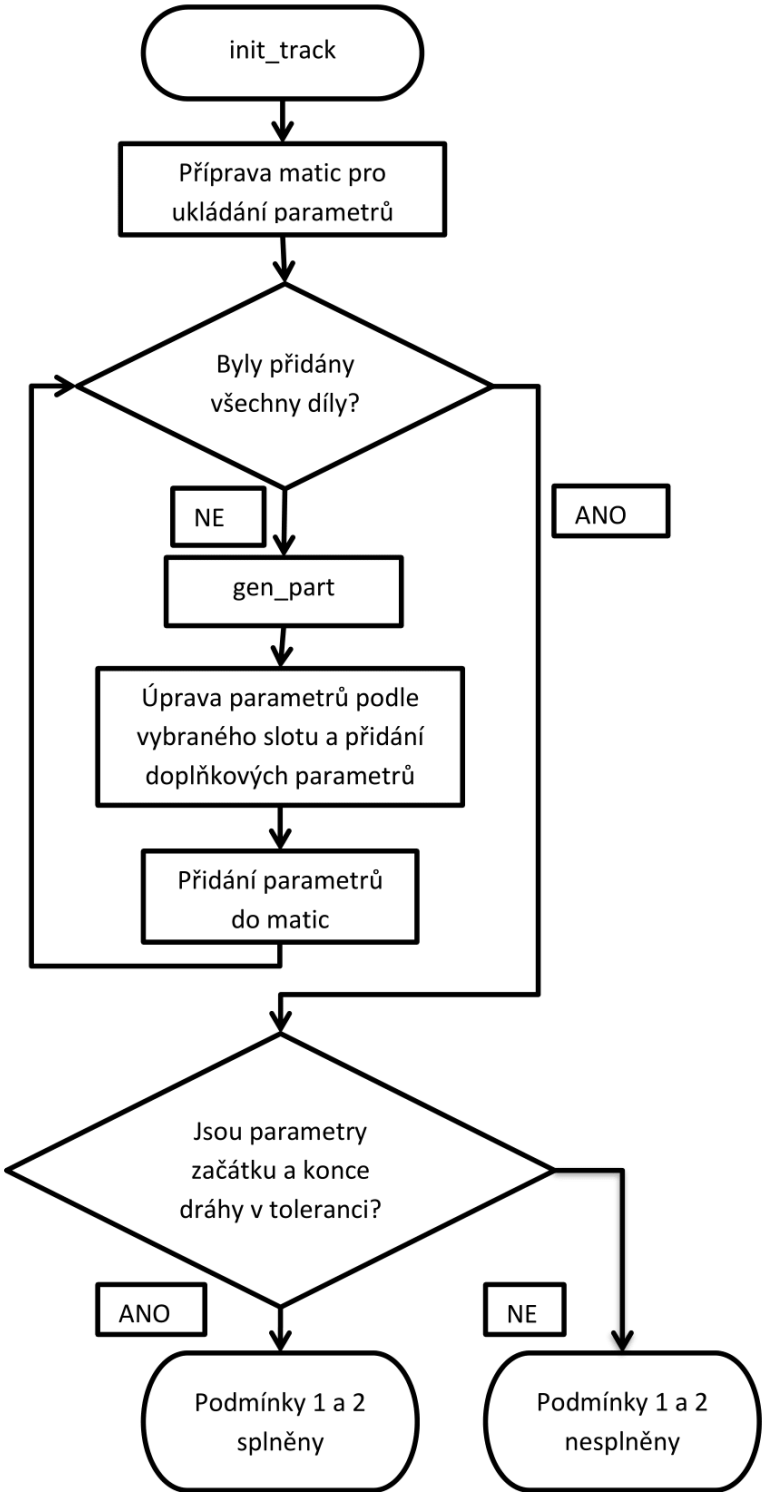
sim/tracks/ uložené dráhy

B Spuštění simulátoru

Nejprve v Matlabu přidáme adresář ***sim*** s podsložkami mezi pracovní složky. GUI se otevře spuštěním souboru **gui.m**. Dráhu lze buď sestavit novou pomocí tlačítek s kódy dílů, nebo ji můžeme otevřít kliknutím na ikonu složky a vybrat některou z připravených drah v adresáři ***tracks***. Před validací, která je spuštěna tlačítkem Validate, je vhodné zkontrolovat či změnit parametry side (volba slotu), number of cars (počet autíček) a track voltage (napětí na dráze). Po úspěšném proběhnutí validace stačí otevřít připravené simulinkové schéma jednoho autíčka **onecar.slx** nebo dvou autíček **twocars.slx**, nastavit požadovaný vstup PWM a spustit simulaci.

C Vývojový diagram inicializační funkce




init_track
Příprava matic pro ukládání parametrů
Byly přidány všechny díly?
NE
ANO
gen_part
Úprava parametrů podle vybraného slotu a přidání doplňkových parametrů
Přidání parametrů do matic
Jsou parametry začátku a konce dráhy v toleranci?
ANO
NE
Podmínky 1 a 2 splněny
Podmínky 1 a 2 nesplněny