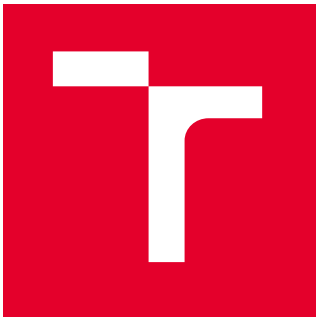


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY

DEPARTMENT OF CONTROL AND INSTRUMENTATION

ANALÝZA KINEMATIKY DIFERENCIÁLNÍHO ROBOTU

AN KINEMATIC ANALYSIS OF SKID-STEER WHEELED MOBILE ROBOT

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **Petr Šafránek**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. František Burian, Ph.D.**
SUPERVISOR

BRNO 2022

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní program **Automatizační a měřicí technika**

Ústav automatizace a měřicí techniky

Student: Petr Šafránek ***ID:*** 211180

Ročník: 3 ***Akademický rok:*** 2021/22

NÁZEV TÉMATU:

Analýza kinematiky diferenciálního robotu

POKYNY PRO VYPRACOVÁNÍ:

Úkolem studenta bude vytvořit výpočetní simulátor přímé i inverzní kinematiky obyčejného diferenciálního robota, při dopředu známé a snadno geometricky interpretovatelné trajektorii (přímka, kruh). V tomto simulátoru by mělo být možné demonstrovat vlastnosti výpočetních metod v závislosti na různých parametrech podvozku a metodách. Výpočet by měl být volitelně vs užitím různých datových typů, měl by mít možnost analyzovat kvalitativní výsledky odometrie pro různé statistické parametry senzorů odometrie vůči velikostem podvozku. Výsledné nalezené závislosti by měly být v práci graficky prezentovány i s odečtenými statistickými parametry, a to zejména na polárním a kartézském zobrazení. Forma práce nemusí být monolitický program, ale například generátor dat do souboru a následná analýza a vizualizace těchto dat v jiném nástroji (např MATLAB).

1. Proveďte rešerši v kinematice diferenciálních robotů
2. Spočítejte přímou a inverzní úlohu kinematiky pro typický diferenciální robot
3. Vytvořte parametrický simulátor, který do souboru vygeneruje sady dat pro pozdější analýzu
4. Analyzujte vygenerovaná data a prezentujte výsledky graficky
5. Zhodnoťte vlivy různých parametrů na výsledné hodnoty.

Termín zadání: 7.2.2022 ***Termín odevzdání:*** 23.5.2022

Vedoucí práce: Ing. František Burian, Ph.D.

doc. Ing. Václav Jirsík, CSc.
předseda rady studijního programu

Abstrakt

Tato práce se zabývá analýzou kinematiky diferenciálního robotu. V první kapitole projdeme teorii ke kinematice. V následující kapitole vypočítáme přímou a inverzní úlohu kinematiky, které následně implementujeme v simulátoru. Kapitola 3 se věnuje tvorbě simulátoru kinematiky a v kapitole 4 zhodnotíme dosažené výsledky. Samotný simulátor se pak nachází v příloze A a příloha B obsahuje m-file se zpracováním výsledků simulace.

Klíčová slova

Robot, diferenciální kolový robot, přímá úloha kinematiky, inverzní úloha kinematiky, simulátor

Abstract

In this thesis deals with an kinematic analysis of skid-steer wheeled mobile robot. In chapter we will focuse on theore about kinematice. In chapter two there is forward and inverse kinematice which will be implemented in the parametric simulator in chapter 3. Results of simulation are sumarized in chapter 4. The simulator is located in an attachment A and m-file with resullts processing is in attachment B.

Keywords

Robot, skid-steer wheeled robot, forward kinematice, inverse kinematice, simulator

Bibliografická citace

ŠAFRÁNEK, Petr. *Analýza kinematiky diferenciálního robotu.* Brno, 2022. Dostupné také z: https://www.vutbr.cz/studenti/zav-prace/detail/137675. Semestrální práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav automatizace a měřicí techniky. Vedoucí práce František Burian.

Prohlášení autora o původnosti díla

Jméno a příjmení studenta:	*Petr Šafránek*
VUT ID studenta:	*211 180*
Typ práce:	*Bakalářská práce*
Akademický rok:	*2021/22*
Téma závěrečné práce:	Analýza kinematiky diferenciálního robotu

Prohlašuji, že svou závěrečnou práci jsem vypracoval samostatně pod vedením vedoucí/ho závěrečné práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené závěrečné práce dále prohlašuji, že v souvislosti s vytvořením této závěrečné práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

.

V Brně dne: 23. května 2022 ----------

podpis autora

Poděkování

Děkuji vedoucímu diplomové (bakalářské) práce Ing. Františku Burianovi, PhD. za odbornou a morální podporu při zpracování mé bakalářská práce.

V Brně dne: 23. května 2022 ----------

podpis autora

Obsah

SEZNAM OBRÁZKŮ

Úvod

Nejjednodušším druhem robota je diferenciální kolový robot. První kapitola této práce pojednává o rozdělení robotů a principu jejich pohybu. Druhá kapitola se zaměřuje na analýzu kinematiky diferenciálního kolového robotu a následné zpracování přímé a inverzní úlohy kinematiky, kde se tyto úlohy řeší pro tři různé metody výpočtu.

Dále se tato kapitola zaměřuje na zdroje různých nejistot a jejich vliv na pohyb robota při pohybu po elementárních trajektoriích.

Ve třetí kapitole je rozebrán postup při konstrukci simulátoru přímé a inverzní úlohy kinematiky a poslední kapitola je věnována zhodnocení výsledků.

1. ROBOT

Termín „robot“ se poprvé objevil v dramatu Karla Čapka. Přesná definice tohoto výrazu není pevně dána. Obecně lze popsat robot jako stroj, pracující s určitou mírou samostatnosti, který vykonává zadané úkoly a současně je schopen interakce se svým okolím. Roboty lze rozdělit do několika kategorií, například podle úkonů, jenž mají vykonávat. V této práci nás budou zajímat pouze mobilní roboty [1].

1.1 Mobilní robot

Mobilní robot je zařízení schopné pohybu v určitém prostředí. Má schopnost pohybovat se ve svém okolí, a tudíž není vázán na jednu fyzickou polohu. Mobilní roboty
mohou být autonomní, což znamená, že jsou schopni samostatné navigace v prostoru, díky čemuž nevyžadují žádné vnější ovládání. Roboty se skládají ze čtyř hlavních komponentů: řídící jednotky, senzorů, aktorů a pohonného systému. Řídící jednotkou bývá nejčastěji mikrokontroler. Použité senzory se liší podle požadavků na robota. Pohonný systém bývá zpravidla napájen stejnosměrně z baterií. Podle použitých aktorů lze mobilní roboty rozdělit do tří skupin.

1.1.1 Chodící roboty

Jedná se o mobilní roboty, které ke svému pohybu využívají kloubové končetiny. Oproti kolovým a pásovým robotům mají výrazně lepší průchodnost terénem. Nevýhodami jsou větší spotřeba energie a komplexnost

1.1.2 Pásové roboty

Využívají k pohybu pásy, které poskytují lepší průchodnost terénem než kola, za cenu větší spotřeby energie. Z hlediska kinematiky jsou podobné robotům diferenciálním.

1.1.3 Kolové roboty

Využívají ke svému pohybu kola. Jejich výhodou je vysoká energetická efektivita a jednoduchá konstrukce. K vlastnímu pohybu robota je nutný kontakt kol se zemí. Nejjednodušším robotem kolové konstrukce je diferenciální robot. [1, 2]

2. DIFERENCIÁLNÍ KOLOVÝ ROBOT

2.1 Konstrukce

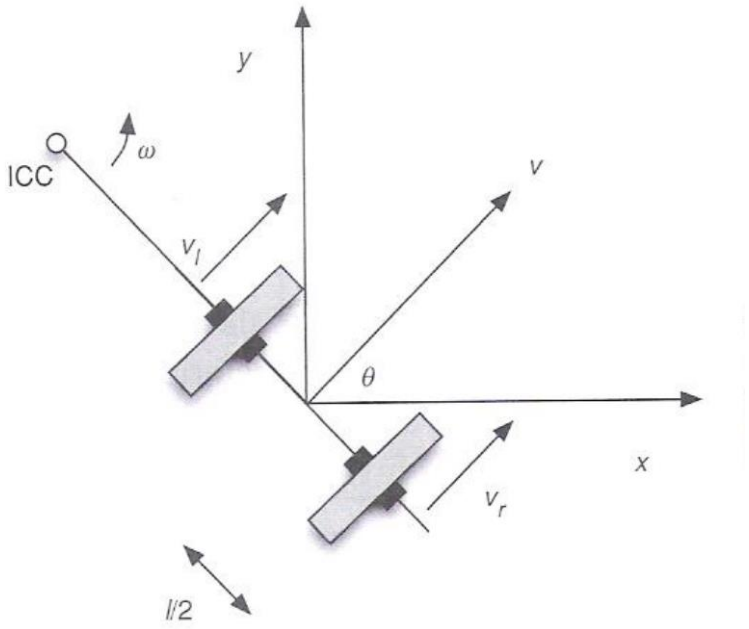
Diferenciální kolový robot se skládá ze dvou kol umístěných na společné nápravě, které jsou ovládány dvěma nezávislými motory. Pro zlepšení stability bývá robot vybaven opěrnou ploškou (případně pomocným kolem) které nemá vliv na výslednou kinematiku robotu. [1]

2.2 Kinematika

V každém bodě můžeme popsat polohu robota v prostoru pomocí souřadnic *x*, *y* a úhlu, který robot svírá s osou x. Tato soustava (x, y, θ) se označuje jako póza robota.

2.2.1 Odometrie

Základem pro popsání pohybu robotu je určení odometrie. Výsledná póza robotu závisí pouze na rychlostech kol a na šířce nápravy. Pro odvození změny polohy v určeném čase je možné použít následující kinematický model: [1, 2, 3]



Obrázek 1: Kinematický model diferenciálního robotu [1]

, kde bod *ICC* je střed otáčení, v je absolutní hodnota výsledné rychlosti v_l a v_r jsou rychlosti jednotlivých kol, θ je úhel natočení robotu, ω je úhlová rychlost a l představuje šířku nápravy.

Výslednou absolutní hodnotu rychlosti lze určit jednoduše jako průměr z rychlostí obou kol:

$$v = \frac{v_r + v_l}{2} \tag{1}$$

Na základě kinematického modelu můžeme rovněž sestavit rovnice pro rychlosti jednotlivých kol:

$$\begin{aligned} \omega(R + l/2) &= v_r \\ \omega(R - l/2) &= v_l \end{aligned} \tag{2}$$

, kde R je vzdálenost středu otáčení ke středu nápravy robotu.

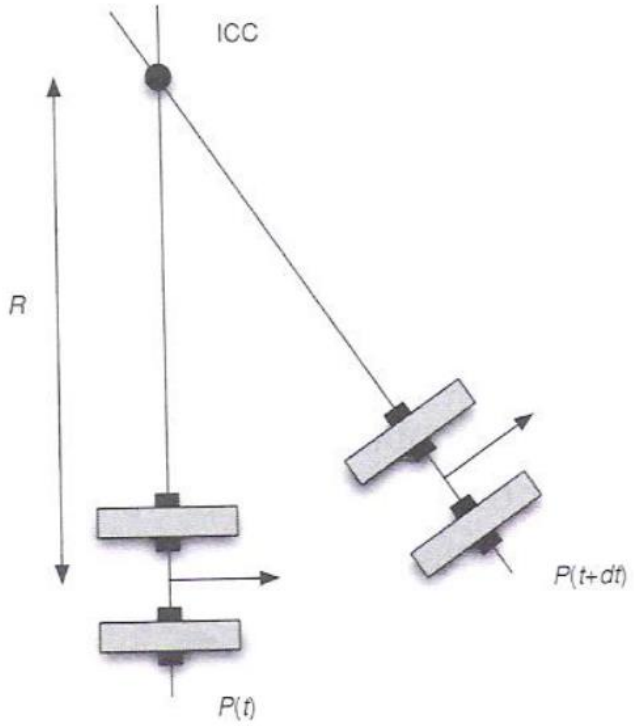
Z této soustavy rovnic můžeme následně vyjádřit vzorce pro výpočet R a ω:

$$\begin{aligned} R &= \frac{l}{2} \cdot \frac{v_r + v_l}{v_r - v_l} \\ \omega &= \frac{v_r - v_l}{l} \end{aligned} \tag{3}$$

Zde je vidět, že v případě shodných rychlostí obou kol vyjde poloměr otáčení roven nekonečnu, což odpovídá jízdě po přímce [1].

2.2.2 Přímá úloha kinematiky

Cílem přímé úlohy kinematiky (PKU) je zjistit pózu, ve které se robot ocitne po uběhnutí daného časového intervalu, na základě nastavení rychlosti kol. Předpokládejme, že se robot nacházel v póze *(x, y, θ)* a našim cílem je určit pózu, ve které se ocitne po určité době δt, při nastavení rychlostí v_l a v_r. Nutnou podmínkou je kontakt obou kol s podložkou.



Obrázek 2: Přímá úloha kinematiky [1]

Protože rychlosti kol, úhlová rychlost a poloměr otočení jsou funkcí času, je možné určit souřadnice středu otáčení, v daném čase, na základě výchozí pózy [1, 3]:

$$ICC = (x - R \cdot \sin(\theta)\,, y + R \cdot \cos(\theta)) \tag{4}$$

A následná póza robota v čase t + δt odpovídá:

$$\begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix} = \begin{bmatrix} \cos(\omega\delta t) & -sin(\omega\delta t) & 0 \\ sin(\omega\delta t) & \cos(\omega\delta t) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x - ICC_x \\ y - ICC_y \\ \theta \end{bmatrix} + \begin{bmatrix} ICC_x \\ ICC_y \\ \omega\delta t \end{bmatrix} \tag{5}$$

Tento výraz popisuje rotaci robota okolo středu otáčení. V závislosti na konstrukci robota se mění i předpis určující R a ω. Roznásobením a dosazením soustavy rovnic 3, pro případ diferenciálního kolového robota získáme předpis přímé úlohy kinematiky:

$$\begin{aligned} x' &= \mathrm{x} + \mathrm{R} \cdot \sin(\theta) \cdot \cos(\omega\delta t) + R \cdot \cos(\theta) \cdot \sin(\omega\delta t) - R \cdot \sin(\theta) \\ y' &= \mathrm{y} - \mathrm{R} \cdot \sin(\theta) \cdot \sin(\omega\delta t) + R \cdot \cos(\theta) \cdot \cos(\omega\delta t) + R \cdot \cos(\theta) \\ \theta' &= \theta + \omega\delta t \end{aligned} \tag{6}$$

Výše zmíněný popis označíme pro odlišení jako první metodu určení pozice robotu. Problém nastává při výpočtu jízdy po přímce, protože poloměr otáčení se v tomto případě blíží k nekonečnu. Z tohoto důvodu je výhodnější použít pro pohyb po přímce následující výraz vzniklý integrací z výrazu 5. Tento způsob výpočtu označíme jako druhou metodu [1]:

$$\begin{aligned} x(t) &= \int_0^t v(t)\cos\big(\theta(t)\big)\,dt \\ y(t) &= \int_0^t v(t)\sin\big(\theta(t)\big)\,dt \\ \theta(t) &= \int_0^t \omega(t)dt \end{aligned} \tag{7}$$

Nyní dosadíme rovnice popisující pohyb diferenciálního robotu 1 a 3 do rovnice 6:

$$\begin{aligned} x(t) &= \tfrac{1}{2}\int_0^t \big(v_r(t)+v_l(t)\big)\cos\big(\theta(t)\big)\,dt \\ y(t) &= \tfrac{1}{2}\int_0^t \big(v_r(t)+v_l(t)\big)\sin\big(\theta(t)\big)\,dt \\ \theta(t) &= \tfrac{1}{l}\int_0^t \big(v_r(t)-v_l(t)\big)dt \end{aligned} \tag{8}$$

Po integraci výrazu 7 získáme řešení přímé úlohy kinematiky za nulových počátečních podmínek:

$$\begin{aligned} x(t) &= \tfrac{1}{2}t(v_r(\mathrm{t})+v_l(\mathrm{t}))\cos\,(\theta(t)) \\ y(t) &= \tfrac{1}{2}t(v_r(\mathrm{t})+v_l(\mathrm{t}))\sin\,(\theta(t)) \\ \theta(t) &= \tfrac{v_r(\mathrm{t})-v_l(t)}{l}t \end{aligned} \tag{9}$$

Pro potřeby simulátoru určíme i předpis, pro výpočet přímé úlohy kinematiky z obecných počátečních podmínek:

$$\begin{aligned} x(t) &= x(t-\delta\mathrm{t}) + \tfrac{1}{2}\delta t\big(v_r(t)+v_l(t)\big)\cos\,(\theta(t-\delta\mathrm{t})+\theta(t)) \\ y(t) &= y(t-\delta\mathrm{t}) + \tfrac{1}{2}\delta t\big(v_r(t)+v_l(t)\big)\sin\,(\theta(t-\delta\mathrm{t})+\theta(t)) \\ \theta(t) &= \theta(t-\delta\mathrm{t}) + \tfrac{(v_r(t)-v_l(t))}{l}\delta t \end{aligned} \tag{10}$$

Nyní ještě zavedeme třetí metodu výpočtu, ze které budeme následně odvozovat vizualizaci vlivu neurčitostí: [1, 3]

$$\begin{aligned} x(t) &= x(t-\delta t) + \frac{1}{2}\delta t\big(v_r(t)+v_l(t)\big)\cos\left(\theta(t-\delta t)+\theta(t)/2\right) \\ y(t) &= y(t-\delta t) + \frac{1}{2}\delta t\big(v_r(t)+v_l(t)\big)\sin\left(\theta(t-\delta t)+\theta(t)/2\right) \\ \theta(t) &= \theta(t-\delta t) + \frac{\left(v_r(t)-v_l(t)\right)}{l}\delta t \end{aligned} \qquad (11)$$

2.2.3 Inverzní úloha kinematiky

Inverzní úloha kinematiky (IKU) se zabývá opačným problémem, kdy známe výchozí pózu a hledáme předpis pro výpočet rychlostí jednotlivých kol potřebných k dosažení konečné polohy v určeném čase. Protože hledání inverze k výrazu 6 by bylo příliš složité, přejdeme rovnou k řešení IKU z výrazu 9. Nejprve sepíšeme rovnice popisují celkovou rychlost a změnu úhlu natočení θ: [1]

$$v(t) = \frac{v_r(t)+v_l(t)}{2} \qquad \theta(t) = \frac{v_r(t)-v_l(t)}{l}t$$

Následně budeme předpokládat, že robot se nachází ve výchozí póze (0, 0, 0) a my se snažíme vyjádřit rovnice pro rychlost levého a pravého kola, které ho dostanou do souřadnic x(t) a y(t). Pro možnost určení nejen kladných, ale i záporných rychlostí musíme použít funkci arkus tangens rozšířenou o 2. a 3. kvadrant:

$$\begin{aligned} v_r(t) - v_l(t) &= \frac{l}{t}\mathrm{atan2}(y(t), x(t)) \\ v_r(t) + v_l(t) &= \frac{2}{t}\sqrt{x(t)^2 + y(t)^2} \end{aligned} \qquad (12)$$

A nyní už stačí vyjádřit rychlosti jednotlivých kol, čímž získáme řešení inverzní úlohy kinematiky:

$$\begin{aligned} v_r(t) &= \frac{2}{t}\sqrt{x(t)^2 + y(t)^2} + \frac{l}{t}\mathrm{atan2}(y(t), x(t)) \\ v_l(t) &= \frac{2}{t}\sqrt{x(t)^2 + y(t)^2} - \frac{l}{t}\mathrm{atan2}(y(t), x(t)) \end{aligned} \qquad (13)$$

Analogicky nyní můžeme vyjádřit řešení inverzní úlohy kinematiky pro třetí metodu: [3]

$$v_r(t) = \frac{2}{t}\sqrt{x(t)^2 + y(t)^2} + \frac{l}{2t}\text{atan2}(y(t), x(t))$$
$$v_l(t) = \frac{2}{t}\sqrt{x(t)^2 + y(t)^2} - \frac{l}{2t}\text{atan2}(y(t), x(t)) \quad (14)$$

2.3 Zdroje neurčitosti

Doposud jsme se zabývali pouze ideálními případy, avšak v reálných aplikacích musíme počítat i s různými nepřesnostmi, kterých se nelze zcela vyvarovat. Na zdroje některých nepřesností se zaměříme v této kapitole. Vzhledem k čistě kinematickému modelu se nebudeme věnovat vlivům souvisejícím s prokluzy a jinými silovými působeními. V této práci budeme počítat s nepřesnostmi vyvolanými vlastnostmi enkodéru, nepřesností povrchu kol a Gaussovým bílým aditivním šumem. Nejprve si popíšeme jednotlivé rozdělení chyb a následně jednotlivé zdroje neurčitostí [5].

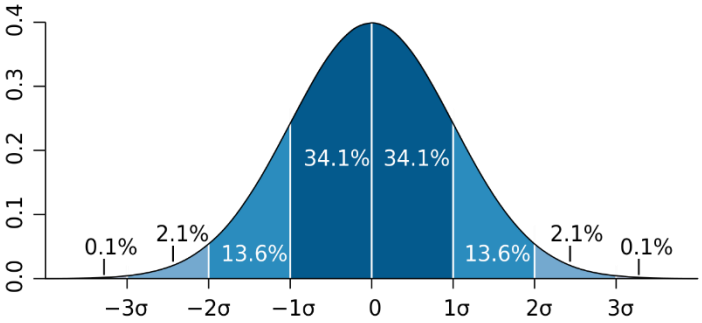
2.3.1 Gaussovo rozložení:

Jedná se o nejběžnější rozložení, které je popsáno následujícím vztahem:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\mu^2}} \quad (15)$$

, kde μ je střed rozložení a σ^2 je rozptyl.

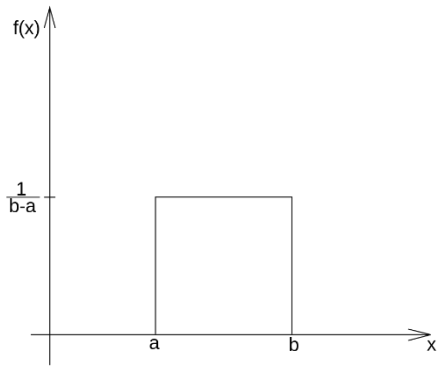
Běžně se toto rozdělení zapisuje ve tvaru $N(\mu, \sigma^2)$.



Obrázek 3: Normální rozdělení [5]

2.3.2 Rovnoměrné rozložení:

Toto rozložení se vyskytuje u chyb spojených s kvantizací dat, zejména při jejich prahování, kdy dochází k zaokrouhlení skutečné hodnoty signálu k nejbližší hodnotě. Typickým příkladem chyby s rovnoměrným rozložením je zaokrouhlování enkodéru při prahování signálu.



Obrázek 4: Rovnoměrné rozložení [5]

2.3.3 Enkodér

Enkodéry jsou nejběžnější snímače pro zjištění rychlosti jednotlivých kol robotu. V principu převádějí enkodéry rotační (případně posuvný) pohyb na elektrický signál.

Nejběžnější enkodéry využívají ke své funkce optický signál emitovaný z LED diody, který prochází skrze otvory v otáčejícím se disku dopadá na fotodiodu. Tyto snímače se dále dělí na absolutní a inkrementální enkodéry.

Absolutní enkodéry vysílají digitální kód, která je jedinečný pro jednotlivé úhly natočení.

Rotační inkrementální enkodéry jsou nejvíce využívaným typem rotačních enkodérů. Jejich hlavní výhodou je vysoké rozlišení, které může dosahovat až 10 000 impulsů na otáčku. Jejich konstrukce se od absolutních enkodérů odlišuje ve výstupu, který je tvořen dvěma signály A a B. Oba dohromady pak indikují nejen rychlost, ale i směr otáčení hřídele. Na rozdíl od absolutních enkodérů nelze z výstupu určit konkrétní poloha, ale pouze její změna. Díky vysoké citlivosti a rychlosti přináší údaje o změně polohy takřka v reálném čase, což je hlavní důvod jejich využití při přesném měření a řízení polohy a rychlosti [2, 8].

2.3.4 Nerovnost povrchu kol

Nerovnosti kol ovlivňují okamžitou rychlost robotu a tím se podílejí na výsledné poloze. Tyto nerovnosti můžeme tedy převézt na kolísání rychlosti s normálovým rozdělením. Menší přesnost rozměrů mají zejména kola vyrobená technologií 3D tisku.

$$v = \omega \cdot r \tag{16}$$

, kde *v* je rychlost kola, *ω* je úhlová rychlost hřídele a *r* je poloměr kola.

Protože úhlová rychlost zůstává konstantní, je kolísání rychlosti přímo úměrné kolísání poloměru kola. Tato nepřesnost bude reprezentována normálním rozdělením [3, 8].

2.3.5 Aditivní šum

Aditivní bílý Gaussovský šum je základním modelem šumu a jeho výskyt může mít celou řadu příčin. V případě diferenciálního robotu můžeme tento šum detekovat na A/D převodnících. Jak již název napovídá, bude mít v simulátoru rovněž normální rozdělení [3, 7, 8].

2.3.6 Sčítání chyb

Protože všechny výše uvedené neurčitosti ovlivňují rychlosti jednotlivých kol je vhodné tyto nejistoty sečíst. Při sčítání jednotlivých nejistot s normálním rozdělení lze postupovat následovně. Mějme dvě nezávislé proměnné popsané následujícími normálními rozděleními [5]:

$$X \sim N(\mu_x, \sigma_x^2)$$
$$Y \sim N(\mu_y, \sigma_y^2)$$

Výsledkem jejich součtu bude opět normální rozdělení s následujícím popisem [5]:

$$Z = X + Y$$
$$Z \sim N(\mu_x + \mu_y, \sigma_x^2 + \sigma_y^2) \tag{17}$$

2.4 Vizualizace neurčitosti

Nejprve vyjádříme rovnice popisující pohyb robota. K tomuto účelu využijeme soustavu rovnic 11, kde nahradíme rychlostní popis popisem vzdáleností ujetých jednotlivými koly:

$$\begin{aligned} x_t &= x_{t-1} + \Delta d \cdot \cos\left(\theta_{t-1} + \frac{\Delta\theta}{2}\right) \\ y_t &= y_{t-1} + \Delta d \cdot \sin\left(\theta_{t-1} + \frac{\Delta\theta}{2}\right) \\ \theta_t &= \theta_{t-1} + \Delta\theta \\ \Delta\theta &= \frac{\Delta d_r - \Delta d_l}{L} \\ \Delta d &= \frac{\Delta d_r + \Delta d_l}{2} \end{aligned} \tag{18}$$

, kde Δd_r a Δd_l představují vzdálenost ujetou pravým a levým kolečkem a Δd představuje aritmetický průměr rychlostí obou koleček, což představuje dráhu, kterou ujel střed robotu během pohybu [3, 7].

Nejistota pozice má normální rozložení se středem v bodě x_t a s kovarianční maticí Σ_t. Tu lze vypočítat následujícím způsobem:

$$\Sigma_t = J_{p(t-1)} \Sigma_{t-1} J_{p(t-1)}^T + J_d \Sigma_\Delta J_d^T \tag{19}$$

Matici Σ_t získáme sečtením předchozí hodnoty kovariance Σ_{t-1} a kovariance Σ_Δ, která odpovídá změně pohybu z místa x_{t-1} do místa x_t. Tu můžeme určit následujícím způsobem:

$$\Sigma_\Delta = \begin{bmatrix} k_r|\Delta d_r| & 0 \\ 0 & k_l|\Delta d_l| \end{bmatrix} \tag{20}$$

, kde k_r a k_l jsou konstanty zahrnující chyby způsobené chybami v měření rychlosti enkodéru, nepřesností povrchu koleček a šumem. Polohový jakobiján získáme následujícím způsobem, parciální derivací soustavy rovnic 18 [3]:

$$J_{p(t-1)} = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial \theta}\right] = \begin{bmatrix} 1 & 0 & -\Delta d \cdot \sin\left(\theta + \frac{\Delta\theta}{2}\right) \\ 0 & 1 & \Delta d \cdot \sin\left(\theta + \frac{\Delta\theta}{2}\right) \\ 0 & 0 & 1 \end{bmatrix} \tag{21}$$

Následně určíme pohybový jakobián obdobným způsobem [3]:

$$J_{p(t-1)} = \left[\frac{\partial f}{\partial \Delta d_r}, \frac{\partial f}{\partial \Delta d_l}\right]$$

$$J_{p(t-1)} = \begin{bmatrix} \frac{1}{2}\cos\left(\theta + \frac{\Delta\theta}{2}\right) - \frac{\Delta d}{2L} sin\left(\theta + \frac{\Delta\theta}{2}\right) & \frac{1}{2}\cos\left(\theta + \frac{\Delta\theta}{2}\right) + \frac{\Delta d}{2L} sin\left(\theta + \frac{\Delta\theta}{2}\right) \\ \frac{1}{2} sin\left(\theta + \frac{\Delta\theta}{2}\right) - \frac{\Delta d}{2L} cos\left(\theta + \frac{\Delta\theta}{2}\right) & \frac{1}{2} sin\left(\theta + \frac{\Delta\theta}{2}\right) - \frac{\Delta d}{2L} cos\left(\theta + \frac{\Delta\theta}{2}\right) \\ \frac{1}{L} & -\frac{1}{L} \end{bmatrix} \quad (22)$$

Pokud počítáme s neurčitostmi, tak by bylo vhodné nějakým způsobem zobrazit pravděpodobné pozice robota. Ve 2D prostoru předpokládáme rozložení ve tvaru elipsy. K určení parametrů elipsy vybereme z kovarianční matice pouze část, která udává polohu, tedy souřadnice *x* a *y* [3, 7, 8]:

$$\Sigma = \begin{bmatrix} \sigma_x^2 & \sigma_{yx} & \sigma_{\theta x} \\ \sigma_{xy} & \sigma_y^2 & \sigma_{\theta y} \\ \sigma_{x\theta} & \sigma_{y\theta} & \sigma_\theta^2 \end{bmatrix} \quad (23)$$

Úhel α mezi hlavní osou elipsy a osou x lze spočítat jako:

$$\alpha = 0.5 \cdot atan2(\sigma_{xy}, \sigma_x^2 - \sigma_y^2) \quad (24)$$

Délku hlavní a vedlejší osy určíme podle vztahů:

$$\alpha_{maj} = \sqrt{\frac{k}{2} \cdot \left(\sigma_x^2 + \sigma_y^2 + \tau\right)} \quad (25)$$

$$\alpha_{min} = \sqrt{\frac{k}{2} \cdot \left(\sigma_x^2 + \sigma_y^2 - \tau\right)} \quad (26)$$

, kde τ je rovno:

$$\tau = \sqrt{(\sigma_x^2)^2 + (\sigma_y^2)^2 - 2 \cdot \sigma_x^2 \cdot \sigma_y^2 + 4 \cdot \sigma_{xy}^2} \quad (27)$$

a k je konstanta odpovídající žádané pravděpodobnosti Pr [3, 7, 8]:

$$k = -2\ln(1 - \mathrm{Pr}) \quad (28)$$

3. PARAMETRICKÝ SIMULÁTOR

Samotný simulátor budeme vytvářet v programovacím prostředí CLion v jazyce c++. Tento simulátor vygeneruje data do souboru, která budeme později zpracovávat v programu MATLAB. V tomto simulátoru budou implementovány tři metody pro výpočet přímé úlohy kinematiky a dvě pro řešení inverzní úlohy kinematiky.

Nejprve vytvoříme třídu CRobot, která bude zajišťovat volání jednotlivých metod. Pro výchozí nastavení proměnných byl zvolen model popisující běžně dostupný robotický podvozek. Aby bylo možné provádět výpočty ve všech jednotlivých typech, budeme v programu uvádět délku v mm, rychlost v mm/s a čas v ms. Kompletní kód je přiložen v příloze.

3.1 Robot.h

V hlavičkovém souboru je uvedena deklarace jednotlivých metod třídy CRobot. Přepínání mezi jednotlivými typy je možné odkomentováním požadovaného typu a zakomentováním ostatních typů. Privátní proměnné představuji aktuální pózu robotu, veřejné proměnné pak představují ostatní parametry robotu. Proměnné iR_Left a iR_Right reprezentují průměry jednotlivých kol, iV_left a iV_right rychlosti jednotlivých kol, iL šířku nápravy, iAlpha_Left a iAlpha_Right nepřesnost povrchu kol, iBeta šum, iRes rozlišení enkodéru, iRes_Time čas integrace výstupu enkodéru, iSigma_Left a iSigma_Right představují složení jednotlivých chyb pro následný výpočet elipsy v programu MATLAB.

```
/*************************************************************************
* robot.h
*/
#ifndef ROBOTKINEMATICS_ROBOT_H
#define ROBOTKINEMATICS_ROBOT_H
#include <cmath>
#include <iostream>
#include <random>

// volba typu

//typedef int TYPENAME;
//typedef float TYPENAME;
typedef double TYPENAME;
//typedef long double TYPENAME;
class CRobot{
    TYPENAME iX, iY;
    double iTheta;
public:
    double iR_Left, iR_Right, iV_left, iV_right, iL, iAlpha_Left,
    iAlpha_Right, iBeta, iRes, iRes_time, iSigma_left, iSigma_right;
    CRobot();
    // Jednotlivé metody vypoctu
    void PKU_1(const double aTimeStep, const double aTime);
    void PKU_2(const double aTimeStep, const double aTime);
    void PKU_3(const double aTimeStep, const double aTime);
    void IKU_2(const double aX, const double aY, const double aTime);
    void IKU_3(const double aX, const double aY, const double aTime);
    // ziskani pozice x
    double GetX() const;
    // Ziskani pozice y
    double GetY() const;
    // Ziskani orientace theta
    double GetTheta() const;
    // Resetovani pozy
    void Reset();
};

#endif //ROBOTKINEMATICS_ROBOT_H
```

3.2 Robot.cpp

V souboru robot.cpp jsou popsány jednotlivé metody. Defaultní konstruktor vytváří třídu CRobot s výchozími parametry odpovídajícími modelu ze zdroje [].

3.2.1 Výpočet přímé úlohy kinematiky

K výpočtu přímé úlohy kinematiky byly využity všechny tři metody simulace. V rámci těchto jednotlivých úloh se současně provádí i výpočet celkového rozptylu rychlostí obou kol. Nejprve určíme vzorec pro přepočet rozlišení enkodéru na chybu rychlosti. Ten získáme vypočítáním rychlosti, která odpovídá jednomu impulzu z enkodéru:

$$\Delta v_{min} = \frac{2 \cdot \pi \cdot R}{Resolution \cdot t} \tag{29}$$

, kde Δv_{min} je rychlost odpovídající jednomu impulzu enkodéru.

Nyní můžeme přejít k sečtení jednotlivých nejistot. Nepřesnost povrchu kolečka a šum mají normální rozdělení se středem v počátku, tudíž zde stačí prostý součet. Problém nastává u rovnoměrného rozložení. Jako nejjednodušší řešení se nabízí převod rovnoměrného rozdělení na normální. Vzhledem k tomu, že se pokoušíme předpovědět polohu, kde leží 99,8 % bodů, lze převézt toto rozložení na normální se středem v bodě nula a rozptylem odpovídajícím třetině b:

$$\sigma_{ekv}^2 = \frac{\frac{\Delta v_{min}}{2}}{3} = \frac{\Delta v_{min}}{6} \tag{30}$$

Nyní již můžeme získat celkový rozptyl součtem podle vzorce 17:

$$\sigma_{suma}^2 = \sigma_{ekv}^2 + \sigma_{beta}^2 + \sigma_{alpha}^2 \tag{31}$$

, kde σ_{beta}^2 odpovídá rozptylu rychlosti způsobenému šumem a σ_{alpha}^2 rozptylu, který je způsoben nepřesností povrchu jednotlivých kol. Pro simulaci náhodného veličiny byl použit výchozí generátor náhodných čísel z knihovny random.

```
// Vypocet prime kinematiky druhou metodou
    void CRobot::PKU_2(const double aTimeStep, const double aTime) {
        // prevod casu z vterin na ms
        TYPENAME Step = 1000 * aTimeStep;
        TYPENAME Time = 1000 * aTime;
        std::random_device rd;  // seed pro engine generatoru
        std::mt19937 gen(rd()); // standardni engine pro generatory
        // Prevod chyby enkoderu z rozliseni na zmenu rychlosti
        TYPENAME vl_error = 1000*iR_Left/iRes/2/3.1415;
        TYPENAME vr_error = 1000*iR_Right/iRes/2/3.1415;
        // pomocne promenne
        TYPENAME v_left, v_right, R;
        double Delta_Theta;
        // Vypocet celkoveho rozptylu rychlosti v m/s
        iSigma_left       =       iBeta*(iV_left+iV_right)/2000       +
iAlpha_Left*iV_left/1000 + vl_error /3000;
        iSigma_right      =      iBeta*(iV_left+iV_right)/2000      +
iAlpha_Left*iV_left/1000 + vr_error /3000;
        // generovani nahodnych velicin
        std::uniform_real_distribution<> encoder_left(-0.5 * vl_error,
0.5 * vl_error);
        std::uniform_real_distribution<>      encoder_right(-0.5      *
vr_error, 0.5 * vr_error);
        std::normal_distribution<>  NoiseDis{0,  iBeta  *  (iV_left  +
iV_right) / 2};
        std::normal_distribution<>   delta_R_left{0,   iAlpha_Left   *
iV_left};
        std::normal_distribution<>   delta_R_right{0,   iAlpha_Right   *
iV_right};
        // simulace pohybu
        for (int i = 0; i <Time/Step; ++i) {
            // zatizeni pozadovane rychlosti chybami
            v_left  =  iV_left  +  encoder_left(gen)  +  NoiseDis(gen)  +
delta_R_left(gen);
            v_right = iV_right + encoder_right(gen) + NoiseDis(gen) +
delta_R_right(gen);
            // vypocet pozy v nasledujicim case
            Delta_Theta = Step * (v_right - v_left) / iL/1000;
            iX  +=  Step  *  (v_right  +  v_left)/2000  *  cos(iTheta  +
Delta_Theta) ;
            iY  +=  Step  *  (v_right  +  v_left)  /  2000  *  sin(iTheta  +
Delta_Theta) ;
            iTheta += Delta_Theta;
        }
    };
```

3.2.2 Výpočet inverzní úlohy kinematiky

Pro výpočet inverzní úlohy kinematiky bylo využito výše uvedených rovnic. Vstupní parametr představuje poloha v metrech. Výstup se tiskne přímo do konzole a to v podobě rychlostí jednotlivých kol v m/s.

```
// vypocet rychlosti jednotlivych kol druhou metodou
void CRobot::IKU_2(const double aX, const double aY, const double
aTime) {
    double     vl     =     sqrt((aX*aX+aY*aY)/aTime)     -     0.5*iL*
atan2(aY,aX)/aTime;
    double     vr     =     sqrt((aX*aX+aY*aY)/aTime)     +     0.5*iL*
atan2(aY,aX)/aTime;
    std::cout << "\n Rychlost leveho kola: " << vl << "\n Rychlost
praveho kola: " << vr << std::endl;
}
```

3.3 main.cpp

V souboru main.cpp je příklad volání jednotlivých metod. Zde je také možné upravovat jednotlivé parametry podvozku robota.

Výstup ze simulátoru je tvořen dvěma soubory. V souboru „robot_data.txt“ se nachází finální póza robotu, kdy první sloupec odpovídá souřadnici x, druhý sloupec představuje souřadnici y a třetí sloupec reprezentuje finální úhel natočení θ. Ukázka prvního souboru:

```
// ukazka casti souboru s vyslednou pozou
0.970844 -0.256118 -0.515866
0.980904 0.158362 0.320128
0.988266 0.0846313 0.170855
1.00303 0.10701 0.212569
0.989702 -0.138321 -0.277722
1.0021 -0.0538359 -0.107343
1.00458 0.113251 0.224523
```

Druhý soubor obsahuje údaje potřebné pro vykreslení elipsy:

```
// ukázka casti souboru s konfiguracnimi udaji
vl   vr   L   t   Sigma_left Sigma_right
1 1 0.1 1 0.0286211 0.0286211
```

3.4 Zpracování dat

Když je simulátor hotov, je třeba přejít ke druhé části 3. bodu zadání a vytvořit m-file, který obstará zpracování dat vytvořených simulátorem.

Nejprve načteme data uložená ve vygenerovaných souborech pomocí funkce importdata. Chybové konstanty jednotlivých kol určíme následujícím způsobem:

$$k_r = 0.01 \cdot \sigma_r^2 \quad (32)$$

$$k_l = 0.01 \cdot \sigma_l^2 \quad (33)$$

, kde k_r a k_l jsou chybové konstanty jednotlivých kol, σ_r^2 a σ_l^2 jsou rozptyly rychlostí příslušných kol a konstanta 0.01 byla určena empiricky.

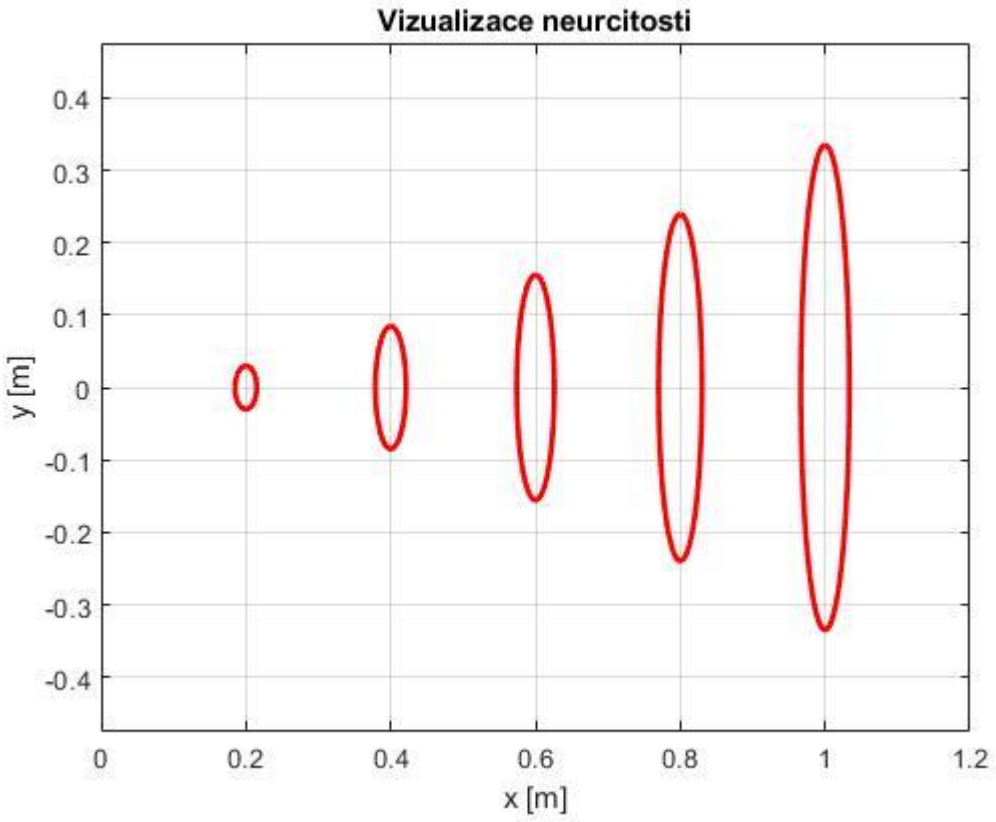
Následně naimplementujeme vzorce pro výpočet kovarianční matice, ze které následně vytvoříme elipsu.

4. ANALÝZA DAT

Nyní již můžeme přejít ke zpracování jednotlivých dat. Nejprve provedeme vizualizaci neurčitosti nejprve analyzujeme pohyb po přímce pro různé parametry simulátoru. Ve všech případech předpokládáme konstantní rychlost robotu.

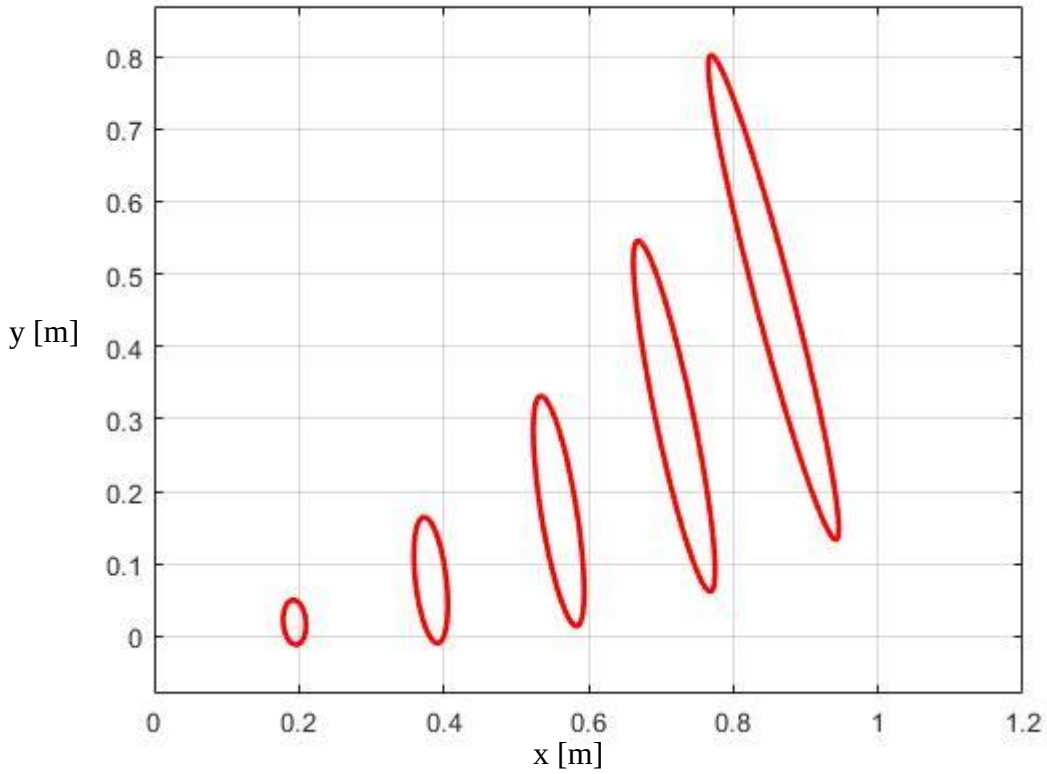
4.1 Vizualizace neurčitosti

Nejprve zkonstruujeme elipsy, které by měly pokrýt 99 % bodů.



Obrázek 5: Graf neurčitosti pro pohyb po přímce

Na tomto grafu můžeme vidět, jak se neurčitost zvyšuje s ujetou vzdáleností. Obdobně lze vykrestlit graf v případě, kdy se jednotlivá kolečka otáčí rozdílnou rychlostí.



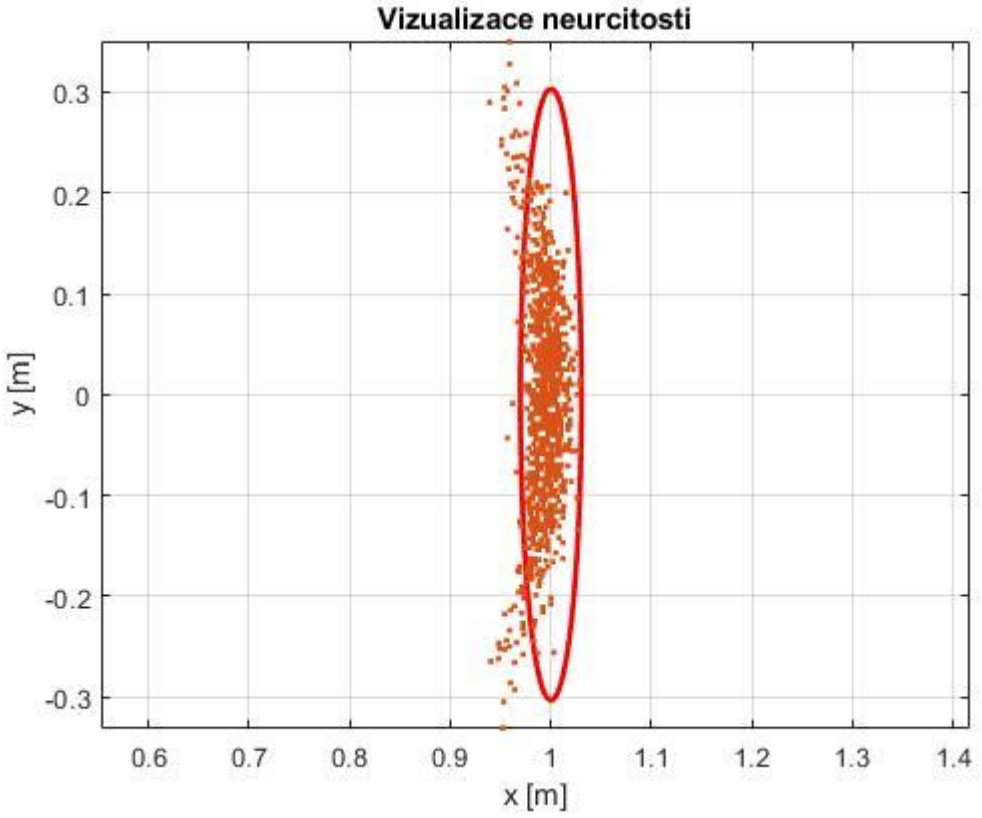
Obrázek 6: Graf neurčitosti pro jízdu po kružnici

V Tomto grafu si můžeme povšimnout zvětšující se elipsy a také nárůstu úhlu alpha, který svírá hlavní osa elipsy s osou x. Rychlost pravého kola byla 1 m/s, rychlost levého 9,5 m/s, šířka podvozku 10 cm, průměr koleček 6,5 cm.

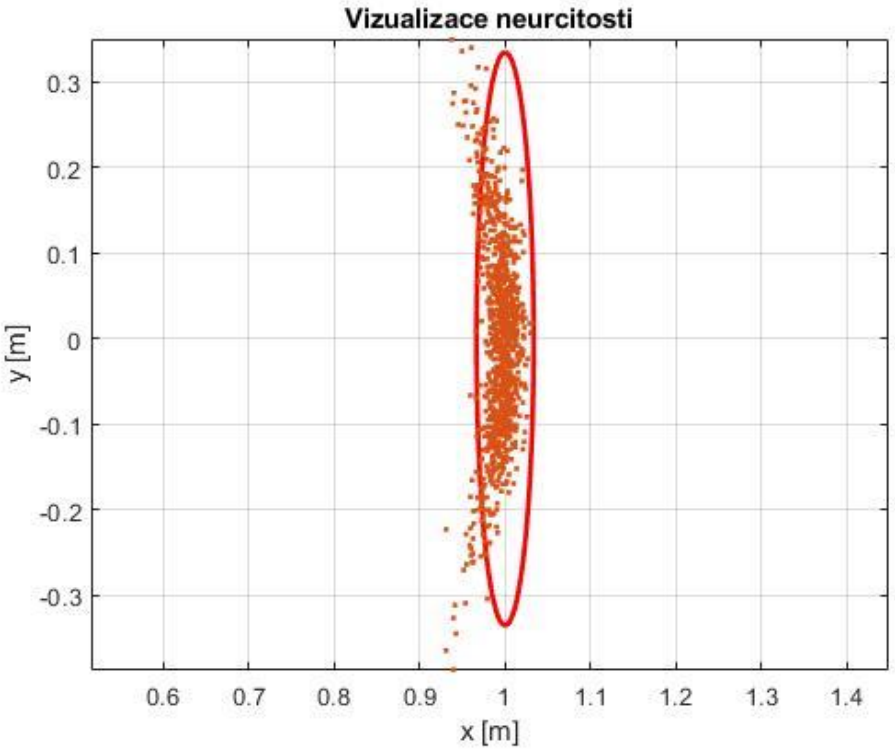
Nyní můžeme zkombinovat data ze simulátoru s předpovídanou neurčitostí a ověřit tak její platnost.

4.2 Jízda po přímce s konstantní rychlostí

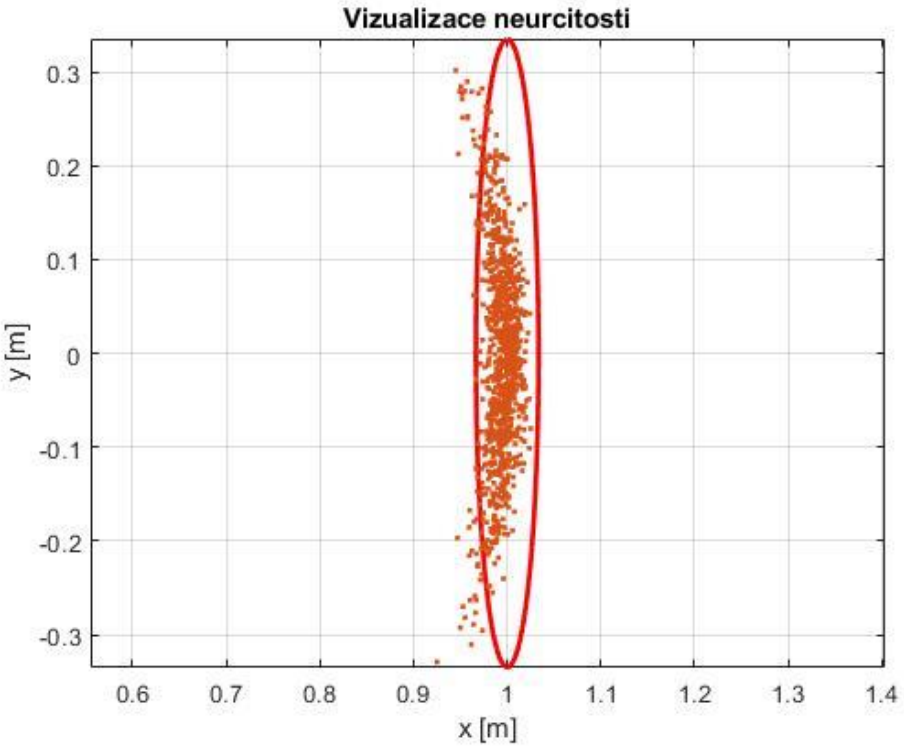
Nejprve zobrazíme grafy s tisíci body, do nichž se robot dostal po ujetí požadované vzdálenost. Výpočet byl prováděn metodou 3 v jednom kroku. Rychlosti levého a pravého kola byly rovny 1 m/s, šířka nápravy L = 10 cm, rozlišení enkodéru 400, rozptyl šumu 0.01 m/s a nepřesnost povrchu kolečka 1 %. V následujících grafech zobrazíme výsledky pro jednotlivé typy proměnných:



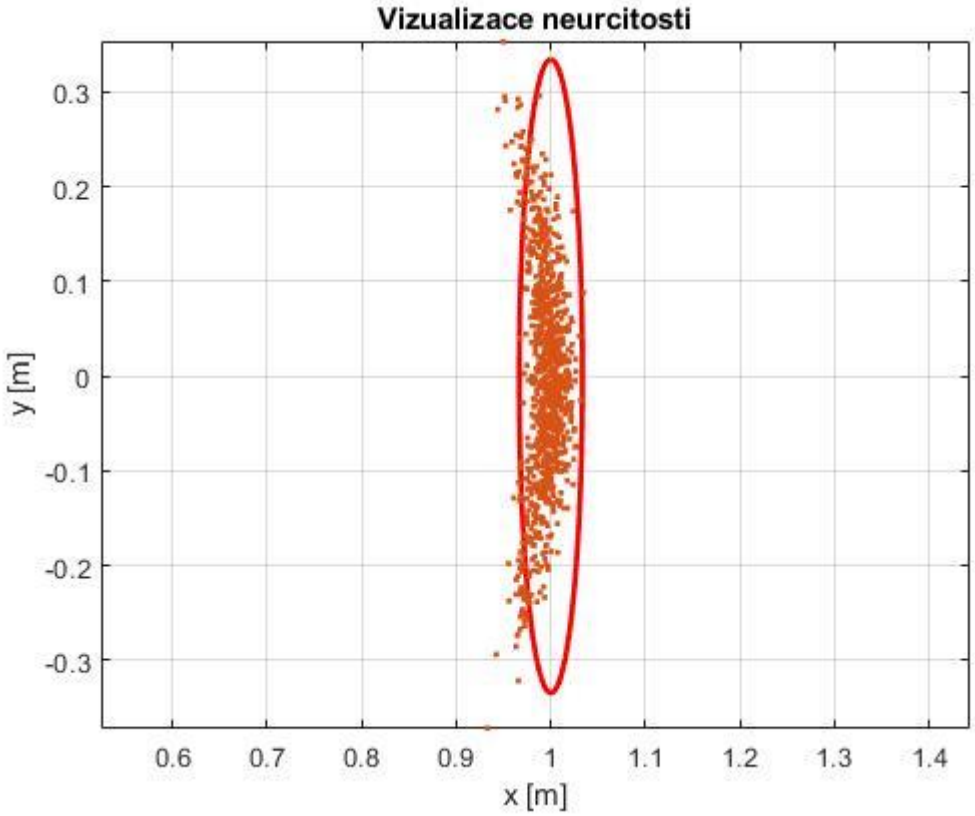
Obrázek 7: Graf neurčitosti polohy robota pro typ int



Obrázek 8: Graf neurčitosti robota pro datový typ float



Obrázek 9: Graf neurčitosti polohy robota pro datový typ double

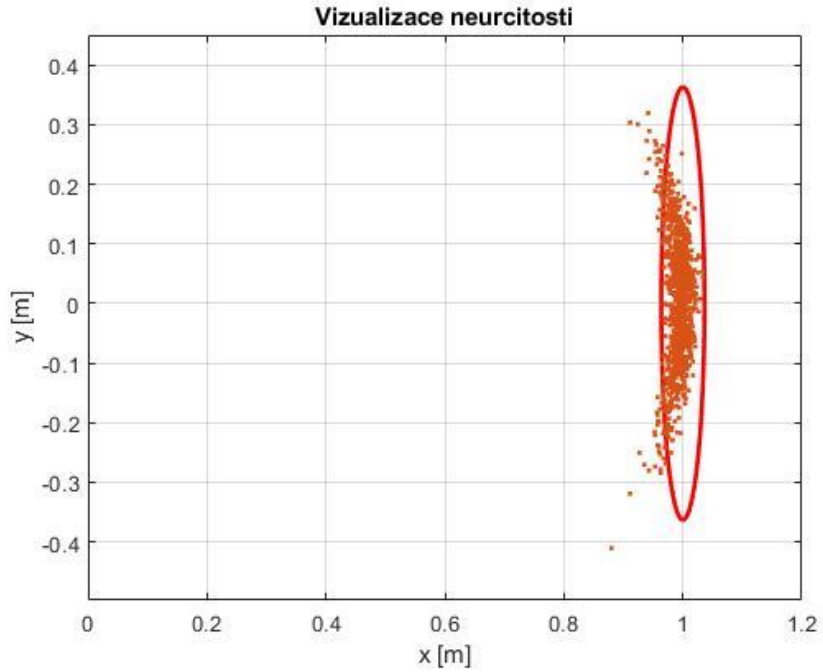


Obrázek 10: Graf neurčitosti pozice robota pro datový typ long

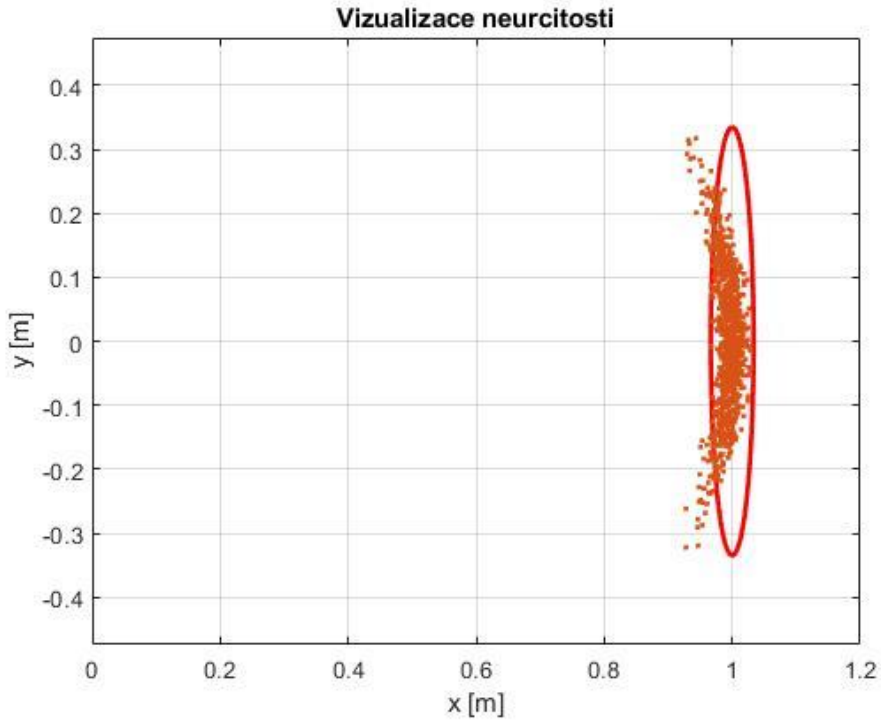
Z výše uvedených grafů vyplývá, že popis pomocí elipsy nemusí být úplně přesný. Daleko více připomíná rozložení bodů v kartézských souřadnicích prohnutou elipsu. Můžeme si povšimnout, že u středu elipsy se body nachází uvnitř elipsy a s rostoucí vzdáleností od středu k vrcholu elipsy se body rozchází.

Nyní můžeme zobrazit pohyb po přímce s konstantní rychlostí také druhou metodou a výsledky porovnat. Parametry zůstávají stejné jako v předchozích případech, jenom počet kroků simulace byl zvýšen na 5.

Z grafů vyplývá, že v případě zvýšení množství kroků simulace dochází k zahuštění prostoru, na kterém se koncové body nachází.



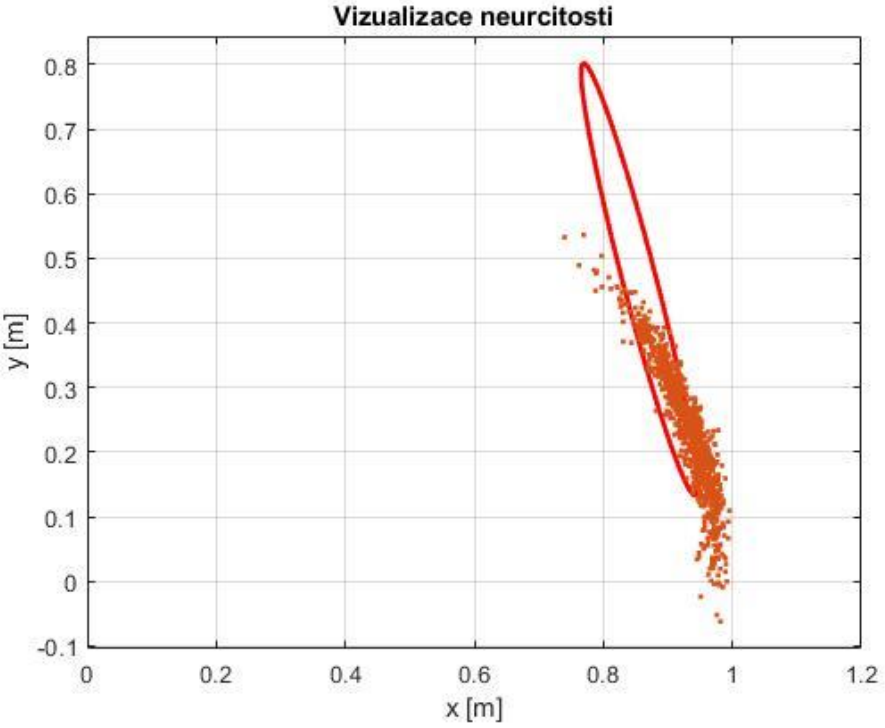
Obrázek 11: Graf neurčitosti výsledné polohy robotu druhou metodou



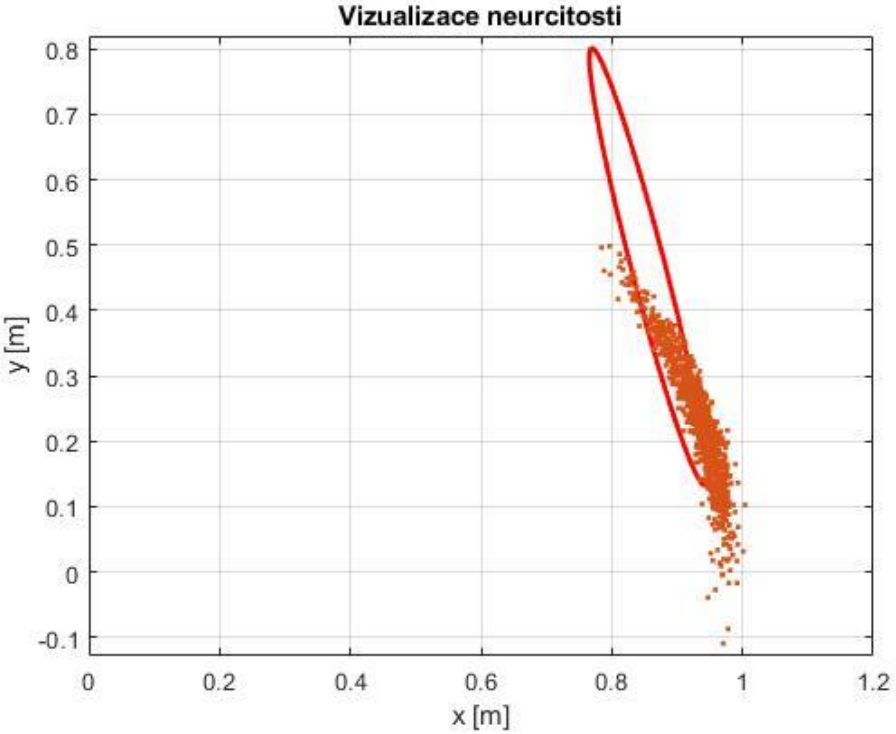
Obrázek 12: Graf neurčitosti výsledné polohy třetí metodou

4.3 Jízda po kružnici s konstantní rychlostí

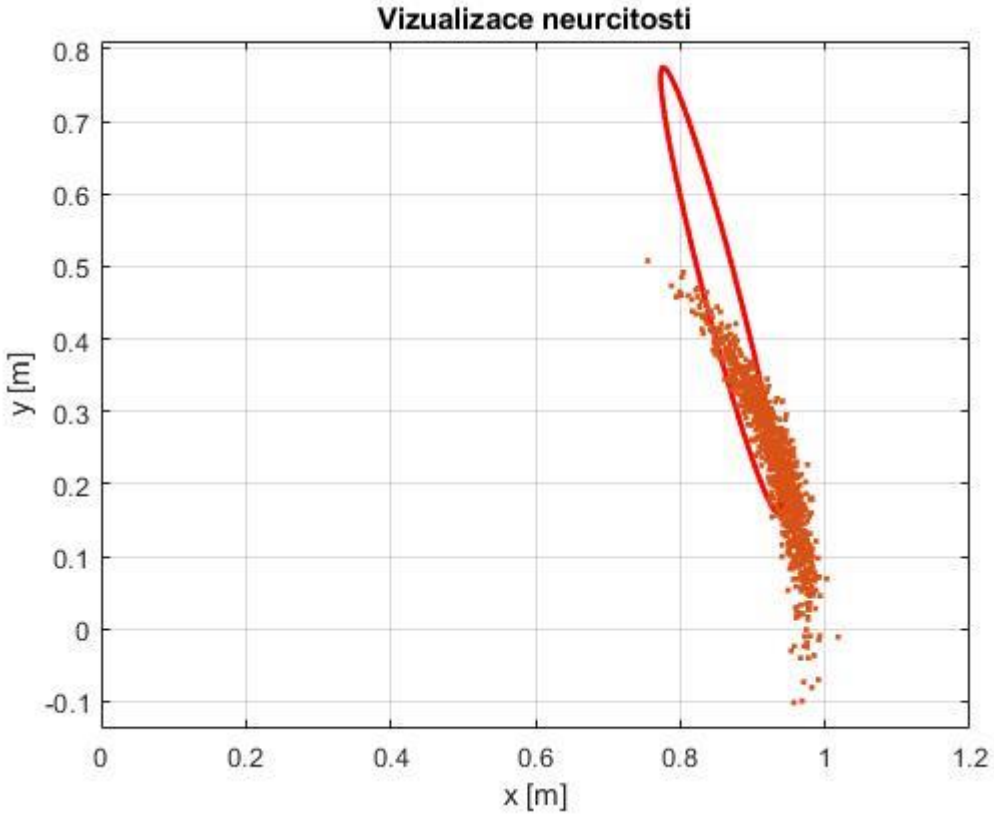
Nyní provedeme porovnání jednotlivých metod i pro jízdu po kružnici. V tomto případě zvolíme rychlost pravého kola 1 m/s a rychlost levého kola 0,95 m/s.



Obrázek 13: Graf neurčitosti při jízdě po kružnici první metodou



Obrázek 14: Graf neurčitosti při jízdě po kružnici druhou metodou

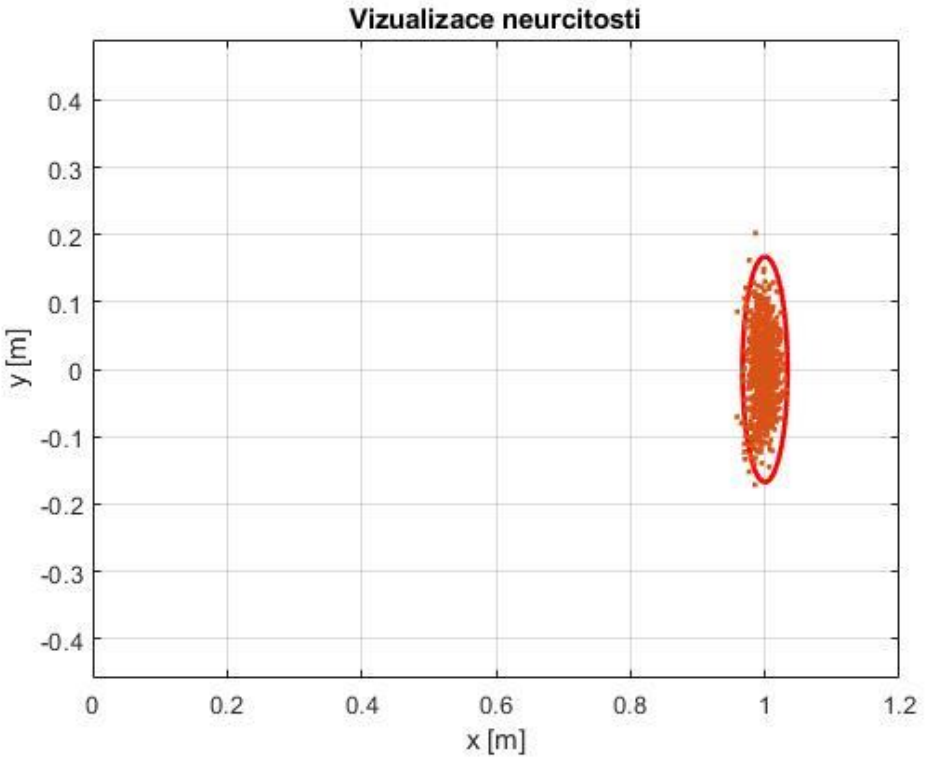


Obrázek 15: Graf neurčitosti při jízdě po kružnici třetí metodou

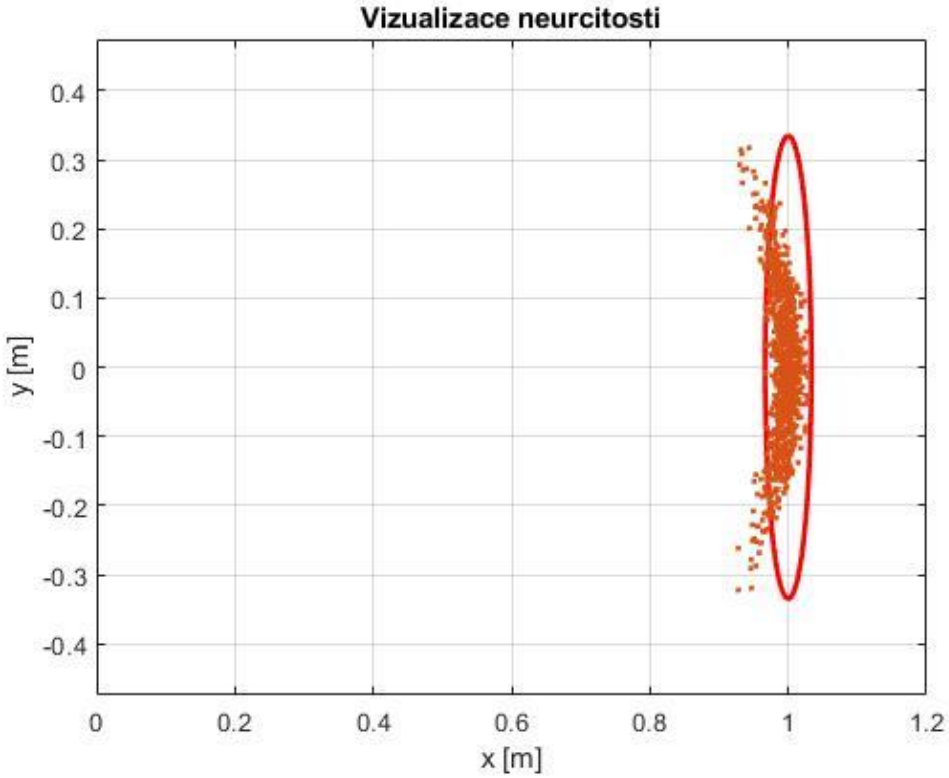
V tomto případě opět výsledná pozice nepadají do elipsy, ale kopírují tvar prohnuté elipsy, podobný rohlíku. Simulace byla provedena s tisíci 10ms kroky.

Při této aproximaci lze pozorovat, že výsledný graf neurčitosti není závislý na zvolené metodě simulace.

Dále můžeme sledovat změnu neurčitosti při změně šířky nápravy, kdy rozšíření nápravy zkrátí hlavní osu elipsy.



Obrázek 16: Graf neurčitosti šířka nápravy L = 20 cm



Obrázek 17: Graf neurčitosti šířka nápravy L = 10 cm

5. ZÁVĚR

Diferenciální kolový robot lze popsat několika různými rovnicemi. V této práci jsou uvedeny tři různé metody výpočtu přímé kinematické úlohy a dvě metody pro výpočet inverzní úlohy kinematiky. První metoda je zaměřena na popis robota pohybujícího se po kružnici o poloměru R (rovnice 6), zbylé dvě metody používají integraci rychlostí jednotlivých kol k získání polohy (rovnice 10 a 11).

Všechny tyto metody jsou implementovány v parametrickém simulátoru i s příslušnými neurčitostmi. V simulátoru lze jednoduše zavolat kteroukoli metodu a provézt simulaci se zadanými parametry.

Při předpovídání neurčitosti jsme vycházeli z disertační práce pana doktora Jaroslava Rozmana. Při porovnání předpověděné plochy neurčitosti jsme došli k závěru, že robot se s předepsanou pravděpodobností neocitne v ploše o tvaru elipsy, ale v ploše tvaru elipsy s prohnutou hlavní osou. Hlavním důvodem může být způsob vyjádření změny polohy v kartézských souřadnicích, jako vhodnější by se mohlo jevit použití polárního souřadného systému.

LITERATURA

[1] DUDEK, Gregory a Michael JENKIN. *Computational principles of mobile robotics*. 2nd ed. New York: Cambridge University Press, 2010. ISBN isbn978-0-521-69212-0.

[2] RIPKA, Pavel. Senzory a převodníky. Praha: Vydavatelství ČVUT, 2005. ISBN isbn80-01-03123-3.

[3] Jaroslav Rozman: Navigace mobilních robotů, disertační práce, Brno, FIT VUT v Brně, 2011

[4] *ČSN EN ISO 80000-2: Veličiny a jednotky - Část 2: Matematické znaky a značky užívané v přírodních vědách a technice*. Praha: Úřad pro technickou normalizaci, metrologii a státní zkušebnictví, 2020.

[5] HEBÁK, Petr a Jana KAHOUNOVÁ. Počet pravděpodobnosti v příkladech. 7., nezměn. vyd. Praha: Informatorium, 2014. ISBN 978-80-7333-10-92.

[6] Siegwart, R. a Nourbakhsh, I. R. Introduction to Autonomous Mobile Robots. Scituate, MA, USA: Bradford Company, 2004. ISBN 026219502X.

[7] Sim, R., Elinas, P. a Griffin, M. Vision-based SLAM Using the Rao-Blackwellised Particle Filter. In IJCAI Workshop on Reasoning with Uncertainty in Robotics. 2005.

[8] Simmons, R. a Koenig, S. Probabilistic Robot Navigation in Partially Observable Environments. In Proceedings of IJCAI-95. [b.m.]: IJCAI, Inc, 1995. S. 1080–1087.

SEZNAM PŘÍLOH

Příloha A - Naměřené hodnoty

A.1 main.cpp

```
// main.cpp
#include <iostream>
#include <fstream>
#include "robot.h"
using namespace std;
int main() {
    // volani tridy
    CRobot Robot;
    //Robot.iV_left=950;
    Robot.iL = 200;
    // priprava souboru pro ulozeni dat
    ofstream DataFile("robot_data.txt");
    ofstream ConfigFile("robot_config.txt");
    srand(time(nullptr));
    // zadani parametru simulace
    double MaxTime = 1;
    double Step = 0.01;
    // volani prime ulohy kinematiky tisickrat
    for (int i = 0; i < 1000; ++i) {
        Robot.PKU_3(Step, MaxTime);
        // zapis vysledne pozy do souboru
        DataFile << Robot.GetX() << " " << Robot.GetY() << " " <<
Robot.GetTheta() << endl;
        Robot.Reset();
    }
    // volani inverzni ulohy kinematiky
    Robot.IKU_2(10, 0, 1);
    // zapis parametru pro vypocet elipsy:
    ConfigFile << "vl   vr   L   t   Sigma_left Sigma_right" <<endl;
    ConfigFile << Robot.iV_left/1000 << " " << Robot.iV_right/1000 <<
" " << Robot.iL/1000 << " " << MaxTime << " ";
    ConfigFile << Robot.iSigma_left << " " << Robot.iSigma_right;
    // zavreni souboru
    ConfigFile.close();
    DataFile.close();
    return 0;
}
```

A.2 robot.cpp

```
//
// Created by petrs on 17.05.2022.
//

#include "robot.h"




// defaultni konstruktor s hodnotami bezne dostupneho modelu
CRobot::CRobot() {
iR_Left=iR_Right=65;
iL=100;
iRes = 400;
iV_right=iV_left = 1000;
iAlpha_Left = iAlpha_Right = 0.01;
iBeta = 0.01;
iX = iY = iTheta = 0;
}
// vrati hodnotu souradnice X v metrech
double CRobot::GetX() const {
    double X = iX;
    return X/1000;
}
// vrati hodnotu souradnice Y v metrech
double CRobot::GetY() const {
    double Y = iY;
    return Y/1000;
}
// vrati hodnotu uhlu Theta v radianech
double CRobot::GetTheta()const {
    return iTheta;
}
// vynuluje pozu robotu
void CRobot::Reset() {
    iX = iY = iTheta = 0;
}
// Vypocet prime kinematiky prvni metodou
void CRobot::PKU_1(const double aTimeStep, const double aTime) {
    // osetreni hazardniho stavu
    if(iV_right==iV_left){
        iV_right=iV_left=0;
        printf("/n Rychlosti kol nesmi byt shodne!/n/n");
    }
    // prevod casu z vterin na ms
    TYPENAME Step = 1000*aTimeStep;
    TYPENAME Time = 1000*aTime;
    std::random_device rd;  // seed pro engine generatoru
    std::mt19937 gen(rd()); // standardni engine pro generatory
    // Prevod chyby enkoderu z rozliseni na zmenu rychlosti
    TYPENAME vl_error = 1000*iR_Left/iRes/2/3.1415;
    TYPENAME vr_error = 1000*iR_Right/iRes/2/3.1415;
```

```
    // pomocne promenne
    TYPENAME v_left, v_right, R;
    double Delta_Theta;
    // Vypocet celkoveho rozptylu rychlosti v m/s
    iSigma_left        =        iBeta*(iV_left+iV_right)/2000        +
iAlpha_Left*iV_left/1000 + vl_error /3000;
    iSigma_right       =        iBeta*(iV_left+iV_right)/2000        +
iAlpha_Left*iV_left/1000 + vr_error /3000;
    // generovani nahodnych velicin
    std::uniform_real_distribution<>        encoder_left(-0.5*vl_error,
0.5*vl_error);
    std::uniform_real_distribution<>       encoder_right(-0.5*vr_error,
0.5*vr_error);
    std::normal_distribution<> NoiseDis{0,iBeta*(iV_left+iV_right)/2};
    std::normal_distribution<> delta_R_left{0,iAlpha_Left*iV_left};
    std::normal_distribution<> delta_R_right{0,iAlpha_Right*iV_right};
    // simulace pohybu
    // zatizeni pozadovane rychlosti chybami
    v_left   =   iV_left   +   encoder_left(gen)   +   NoiseDis(gen)   +
delta_R_left(gen);
    v_right   =   iV_right   +   encoder_right(gen)   +   NoiseDis(gen)   +
delta_R_right(gen);
    for (int i = 0; i < Time/Step; ++i) {
        // vypocet pozy v nasledujicim case
        Delta_Theta = 0.001*Step * (v_right - v_left) / iL;
        R = 0.5 * iL * (v_right + v_left) / (v_right - v_left);
        iX  +=  R  *  (sin(iTheta)  *  cos(Delta_Theta)  +  cos(iTheta)  *
sin(Delta_Theta) - sin(iTheta));
        iY  +=  R  *  (sin(iTheta)  *  sin(Delta_Theta)  -  cos(iTheta)  *
cos(Delta_Theta) + cos(iTheta));
        iTheta += Delta_Theta;
        }
    };
// Vypocet prime kinematiky druhou metodou
    void CRobot::PKU_2(const double aTimeStep, const double aTime) {
        // prevod casu z vterin na ms
        TYPENAME Step = 1000 * aTimeStep;
        TYPENAME Time = 1000 * aTime;
        std::random_device rd;  // seed pro engine generatoru
        std::mt19937 gen(rd()); // standardni engine pro generatory
        // Prevod chyby enkoderu z rozliseni na zmenu rychlosti
        TYPENAME vl_error = 1000*iR_Left/iRes/2/3.1415;
        TYPENAME vr_error = 1000*iR_Right/iRes/2/3.1415;
        // pomocne promenne
        TYPENAME v_left, v_right, R;
        double Delta_Theta;
        // Vypocet celkoveho rozptylu rychlosti v m/s
        iSigma_left       =       iBeta*(iV_left+iV_right)/2000       +
iAlpha_Left*iV_left/1000 + vl_error /3000;
        iSigma_right      =       iBeta*(iV_left+iV_right)/2000       +
iAlpha_Left*iV_left/1000 + vr_error /3000;
        // generovani nahodnych velicin
        std::uniform_real_distribution<> encoder_left(-0.5 * vl_error,
0.5 * vl_error);
        std::uniform_real_distribution<>      encoder_right(-0.5      *
vr_error, 0.5 * vr_error);
        std::normal_distribution<>  NoiseDis{0,  iBeta  *  (iV_left  +
iV_right) / 2};
        std::normal_distribution<>   delta_R_left{0,   iAlpha_Left   *
```

```
iV_left};
        std::normal_distribution<>  delta_R_right{0,  iAlpha_Right  *
iV_right};
        // simulace pohybu
        // zatizeni pozadovane rychlosti chybami
        v_left  =  iV_left  +  encoder_left(gen)  +  NoiseDis(gen)  +
delta_R_left(gen);
        v_right  =  iV_right  +  encoder_right(gen)  +  NoiseDis(gen)  +
delta_R_right(gen);
        for (int i = 0; i <Time/Step; ++i) {
            // vypocet pozy v nasledujicim case
            Delta_Theta = Step * (v_right - v_left) / iL/1000;
            iX  +=  Step  *  (v_right  +  v_left)  /  2000  *  cos(iTheta  +
Delta_Theta) ;
            iY  +=  Step  *  (v_right  +  v_left)  /  2000  *  sin(iTheta  +
Delta_Theta) ;
            iTheta += Delta_Theta;
        }
    };
// Vypocet prime kinematiky treti metodou
        void CRobot::PKU_3(const double aTimeStep, const double aTime)
{
            // prevod casu z vterin na ms
            TYPENAME Step = 1000 * aTimeStep;
            TYPENAME Time = 1000 * aTime;
            std::random_device rd;  // seed pro engine generatoru
            std::mt19937   gen(rd());   //   standardni   engine   pro
generatory
            // Prevod chyby enkoderu z rozliseni na zmenu rychlosti
            TYPENAME vl_error = 1000*iR_Left/iRes/2/3.1415;
            TYPENAME vr_error = 1000*iR_Right/iRes/2/3.1415;
            // pomocne promenne
            TYPENAME v_left, v_right, R;
            double Delta_Theta;
            // Vypocet celkoveho rozptylu rychlosti v m/s
            iSigma_left     =     iBeta*(iV_left+iV_right)/2000     +
iAlpha_Left*iV_left/1000 + vl_error /6000;
            iSigma_right     =     iBeta*(iV_left+iV_right)/2000     +
iAlpha_Left*iV_left/1000 + vr_error /6000;
            // generovani nahodnych velicin
            std::uniform_real_distribution<>    encoder_left(-0.5    *
vl_error, 0.5 * vl_error);
            std::uniform_real_distribution<>    encoder_right(-0.5    *
vr_error, 0.5 * vr_error);
            std::normal_distribution<> NoiseDis{0, iBeta * (iV_left +
iV_right) / 2};
            std::normal_distribution<>  delta_R_left{0,  iAlpha_Left  *
iV_left};
            std::normal_distribution<> delta_R_right{0, iAlpha_Right *
iV_right};
            // simulace pohybu
       // zatizeni pozadovane rychlosti chybami
            v_left  =  iV_left  +  encoder_left(gen)  +  NoiseDis(gen)  +
delta_R_left(gen);
            v_right = iV_right + encoder_right(gen) + NoiseDis(gen) +
delta_R_right(gen);
            for (int i = 0; i < Time/Step; ++i) {
                // vypocet pozy v nasledujicim case
                Delta_Theta = Step * (v_right - v_left) / iL/1000;
```

```
                iX += Step * (v_right + v_left) / 2000 * cos(iTheta +
Delta_Theta/2) ;
                iY += Step * (v_right + v_left) / 2000 * sin(iTheta +
Delta_Theta/2) ;
                iTheta += Delta_Theta;
            }
        }
// vypocet rychlosti jednotlivych kol druhou metodou
void CRobot::IKU_2(const double aX, const double aY, const double
aTime) {
    double     vl     =     sqrt((aX*aX+aY*aY)/aTime)     -     0.5*iL*
atan2(aY,aX)/aTime;
    double     vr     =     sqrt((aX*aX+aY*aY)/aTime)     +     0.5*iL*
atan2(aY,aX)/aTime;
    std::cout << "\n Rychlost leveho kola: " << vl << "\n Rychlost
praveho kola: " << vr << std::endl;
}
// vypocet rychlosti jednotlivych kol treti metodou
void CRobot::IKU_3(const double aX, const double aY, const double
aTime) {
    double vl = sqrt((aX*aX+aY*aY)/aTime) - iL* atan2(aY,aX)/aTime;
    double vr = sqrt((aX*aX+aY*aY)/aTime) + iL* atan2(aY,aX)/aTime;
    std::cout << "\n Rychlost leveho kola: " << vl << "\n Rychlost
praveho kola: " << vr << std::endl;
}
```

A.3 robot.h

```
//
// Created by Petr Safranek on 17.05.2022.
//

#ifndef ROBOTKINEMATICS_ROBOT_H
#define ROBOTKINEMATICS_ROBOT_H
#include <cmath>
#include <iostream>
#include <random>

// volba typu

//typedef int TYPENAME;
//typedef float TYPENAME;
typedef double TYPENAME;
//typedef long double TYPENAME;
class CRobot{
    TYPENAME iX, iY;
    double iTheta;
public:
    double iR_Left, iR_Right, iV_left, iV_right, iL, iAlpha_Left,
    iAlpha_Right, iBeta, iRes, iSigma_left, iSigma_right;
    CRobot();
    // Jednotlivé metody vypoctu
    void PKU_1(const double aTimeStep, const double aTime);
    void PKU_2(const double aTimeStep, const double aTime);
    void PKU_3(const double aTimeStep, const double aTime);
    void IKU_2(const double aX, const double aY, const double aTime);
    void IKU_3(const double aX, const double aY, const double aTime);
    // ziskani pozice x
    double GetX() const;
    // Ziskani pozice y
    double GetY() const;
    // Ziskani orientace theta
    double GetTheta() const;
    // Resetovani pozy
    void Reset();
};

#endif //ROBOTKINEMATICS_ROBOT_H
```

Příloha B - m-file pro zpracování dat

```
% m-file pro zpracovani dat z kinematickeho simulatoru
% autor: Petr Safranek ID: 211180
% datum vytvoreni: 20. 5. 2022

% Udrzba
clc
clear
close
% Nacteni dat ze souboru
FileConfig = "robot_config.txt";
FileData = "robot_data.txt";

Config = importdata(FileConfig).data;
Data = importdata(FileData);
Steps = 1;
vl = Config(1,1);
vr = Config(1,2);
L = Config(1,3);
t = Config(1,4)/Steps;
kl = 0.01*Config(1,5);
kr = 0.01*Config(1,6);
% Vypocet ujete vzdalenosti
% Inicializace kovariancni matice polohy
Kov_matrix = zeros(3,3);
for i = 1:Steps
dsr = vr*t(1);
dsl = vl*t(1);
ds = 0.5*(dsr+dsl);
Theta = t(1)*(vr-vl)/L;
% Vypocet polohoveho jakobianu
    Jp = [1 0 -ds*sin(0.5*Theta); 0 1 ds*cos(0.5*Theta); 0 0 1];
% Vypocet pohyboveho jakobianu
Jd               =               [0.5*cos(Theta)-0.5*ds*sin(0.5*Theta)/L
0.5*cos(Theta)+0.5*ds*sin(0.5*Theta)/L;
    0.5*sin(Theta)-0.5*ds*cos(0.5*Theta)/L
0.5*sin(Theta)+0.5*ds*cos(0.5*Theta)/L; 1/L -1/L];


% Vypocet kovariancni matice pohybu
Kov_matrix_delta = [kl*abs(dsr) 0; 0 kr*abs(dsl)];
    Kov_matrix =Jp*Kov_matrix*Jp'+ Jd * Kov_matrix_delta * Jd';
Pr = 0.99;
k = -2*log(1-Pr);
alpha = -0.5*atan2(2*Kov_matrix(1,2),Kov_matrix(1,1)-Kov_matrix(2,2));
Tau             =             sqrt(Kov_matrix(1,1)^2+Kov_matrix(2,2)^2-
2*Kov_matrix(1,1)*Kov_matrix(2,2)+4*Kov_matrix(1,2)^2);
alpha_maj = sqrt(0.5*k*(Kov_matrix(1,1)+Kov_matrix(2,2)+Tau));
alpha_min = sqrt(0.5*k*(Kov_matrix(1,1)+Kov_matrix(2,2)-Tau));

time = linspace(0,2*pi);
x = alpha_maj * sin(time+alpha);
y = alpha_min * cos(time+alpha);
Matrix_rot = [cos(alpha) -sin(alpha); sin(alpha) cos(alpha)];
Ellipse_rot = Matrix_rot * [x; y];
```

```
x_r = Ellipse_rot(1,:)+ds*cos(Theta);
y_r = Ellipse_rot(2,:)+ds*sin(Theta);

end
figure(1)
plot(x_r, y_r, 'r-', 'LineWidth', 2);
axis equal
grid on;
hold on;
plot(Data(:,1), Data(:,2), '.')
title('Vizualizace neurcitosti')
xlabel('x [m]')
ylabel('y [m]')
xlim([0 1.2])
```