# TECHNICKÁ UNIVERZITA V LIBERCI

Fakulta mechatroniky a mezioborových inženýrských studií


BAKALÁŘSKÁ PRÁCE

# Inverzní úloha robotiky a její řešení

# Solving inverse task of robotics


Liberec 2008 Jan Opálka

# TECHNICKÁ UNIVERZITA V LIBERCI

Fakulta mechatroniky a mezioborových inženýrských studií


# Inverzní úloha robotiky a její řešení


Jan Opálka


| | |
|---|---|
| Studijní program: | B2612 – Elektrotechnika a informatika |
| Studijní obor: | 2612R001 - Elektronické informační a řídicí systémy |
| Pracoviště: | Ústav mechatroniky a technické informatiky |
| | Fakulta mechatroniky a mezioborových inženýrských studií |
| | Technická Univerzita v Liberci<br>Studentská 2, 461 17, Liberec 1 |
| **Vedoucí bakalářské práce:** | **doc. Ing. Mgr. Václav Záda, CSc.** |

## Abstrakt

Bakalářská práce se zabývá řešením inverzní kinematické úlohy robotiky. Inverzní úloha přepočítává polohu chapadla v systému základny na kloubové souřadnice robota. Pro bod je potřeba nalézt všechna možná řešení, u trajektorie je stěžejní rychlost výpočtu a použití vhodné interpolace.

V teoretické části práce je uveden návod pro návrh kinematických parametrů robota. Jsou zde nastíněny vybrané metody pro řešení inverzní úlohy. Dále je popsána problematika plánování trajektorie. Pozornost je věnována i přímé kinematické úloze.

V praktické části je popsán program pro plánování trajektorie. Naprogramovaná aplikace umožňuje řešit inverzní úlohu pro libovolnou kinematickou strukturu s maximálně šesti stupni volnosti. Inverzní úloha je řešena pomocí gradientní metody. Vypočtená trajektorie je graficky a tabelárně zpracována.

**Klíčová slova:** Robotika, Inverzní kinematická úloha, Kinematika

## Abstract

Bachelor work deals with solving inverse kinematic task of robotics. Inverse task converts location of grab in basic system to coordinate the robot. For point is mind to find all solvings, for trajectory is main task speed of calculation and right interpolation.

In theoretic part of work is described instruction for finding kinematic parameters of robot. You could find there chosen methods for solving inverse task. Next described problem is planning trajectory. Attention is given to direct task of robotics too.

In practical part is described program for planning trajectory. Application enables solving inverse task for arbitrary kinematics structure with six degrese of freedom. Inverse task is solved by gradient method. Computed trajectory is worked up by graphs and tables.

**Keywords:** Robotics, Inverse kinematic task, Kinematics

# Prohlášení

Byl jsem seznámen s tím, že na mou bakalářskou práci se plně vztahuje zákon č. 121/200 o právu autorském, zejména § 60 (školní dílo).

Beru na vědomí, že Technická Univerzita v Liberci má právo na uzavření licenční smlouvy o užití mé bakalářské práce a prohlašuji, že souhlasím s případným užitím mé bakalářské práce (prodej, zapůjčení apod.).

Jsem si vědom toho, že užít své bakalářské práce či poskytnout licenci k jejímu využití mohu jen se souhlasem Technické Univerzity v Liberci, která má právo požadovat ode mne přiměřený příspěvek na úhradu nákladů vynaložených univerzitou na vytvoření díla (až do jejich skutečné výše).

Bakalářskou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím bakalářské práce a konzultantem.

V Liberci dne: 12.5.2008 ……

Jan Opálka

## Poděkování

Rád bych poděkoval **doc. Ing. Mgr. Václavu Zádovi, CSc.** za odborné vedení při tvorbě bakalářské práce.

# Obsah

# Úvod

Bakalářská práce se zabývá inverzní úlohou robotiky pro průmyslové roboty a manipulátory. Inverzní úloha spadá do oblasti řízení robotů. Jedná se o problém, jak natočit či vysunout pohony robota tak, aby bylo docíleno požadované polohy a orientace chapadla robota, popřípadě jeho pracovního nástroje. Poloha a orientace je vztažena k nějakému referenčnímu souřadnému systému, nejčastěji k systému základny robota. Obecně se inverzní úloha řeší nejen pro polohu chapadla, ale i pro jeho rychlost a zrychlení. V této práci nás bude zajímat pouze kinematika robota. Proto se v práci objevuje pojem „inverzní kinematická úloha“.

Je zřejmé, že řešit inverzní kinematickou úlohu je v robotice velmi žádoucí. Každý řídící systém moderního robota ji má v sobě implementovanou. Pro člověka je přeci snazší programovat polohování robota v kartézském sytému základny, než v jeho zobecněných (někdy i kloubových) souřadnicích. Programujeme-li robota v kloubových souřadnicích, musíme přepočítávat aktuální nastavení pohonů zpět na polohu v souřadném systému základny – přímá kinematická úloha.

Představme si, že naše ruka symbolizuje nějaký manipulátor. Každému natočení našich kloubů bude jednoznačně příslušet nějaká poloha a orientace dlaně v prostoru (přímá úloha). Pokud ale budeme chtít nastavit klouby tak, abychom docílili předem požadované polohy a orientace dlaně v prostoru (inverzní úloha), zjistíme, že klouby můžeme nastavit více způsoby. Dokonce, pokud si zadáme nedostižný cíl, nenajdeme žádný způsob natočení kloubů. Z toho vyplývá, že inverzní úloha bude oproti přímé úloze znatelně komplikovanější, neboť její řešení není jednoznačné a vede k poměrně komplikovaným nelineárním soustavám rovnic.

Na metody řešení inverzní úlohy jsou kladeny různé nároky podle toho, v jaké aplikaci se používají. Ideální jsou takové robustní metody, které dokáží řešit inverzní úlohu v reálném čase, a to pro jakékoli kinematické struktury. Velká rychlost konvergence metody k řešení je požadována především v oblasti řízení a regulace. V práci jsou nastíněny vybrané metody pro řešení inverzní úlohy.

V praxi nám ale nepostačí řešit inverzní úlohu jenom pro bod. Potřebujeme, aby se chapadlo pohybovalo po dané trajektorii. Bakalářská práce se zabývá i plánováním a interpolací trajektorie robota, která je zadávána soustavou polohových bodů.

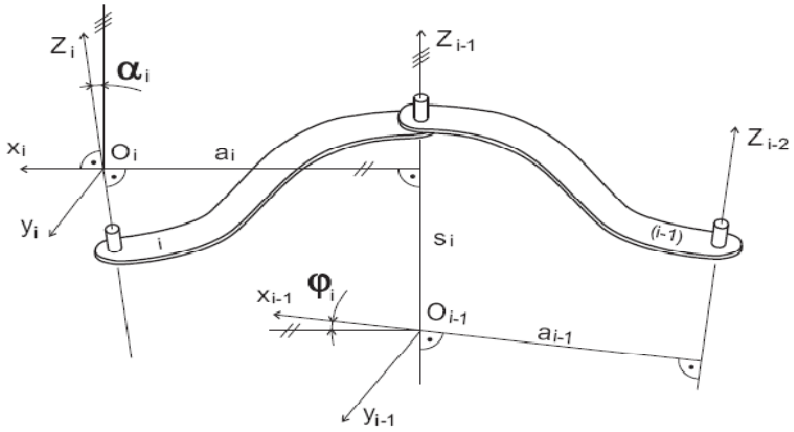
# Teoretická část

## 1 Návrh kinematických parametrů robota

### 1.1 Denavit-Hartenbergova konvence

Robot se skládá z několika kinematických dvojic, které tvoří kinematický řetězec robota. Počet těchto dvojic je dán počtem stupňů volnosti. Pro každý kinematický člen musí být zvolen souřadný systém. Ten se může obecně volit libovolně. Při libovolném zvolení souřadných systémů je ale komplikované sestavovat transformační matice pro přepočet souřadnic mezi systémy. Aby bylo nalezení těchto transformačních matic co možná nejjednodušší, je vhodné zavést pro zvolení souřadných systémů nějakou konvenci.

Jednou z těchto dohod je Denavit-Hartenbergova konvence (dále bude v práci Denavit-Hartenberg nahrazen zkratkou „DH“), pojmenovaná po svých zakladatelích (Denavit a Hartenberg, 1955). Abychom mohli sestavovat transformační matice automaticky, musíme při návrhu souřadných systémů dodržet následující pravidla:




*Obr. 1: Návrh souřadných systémů členů robota podle DH konvence.*

1. Očíslujeme jednotlivé kinematické členy robota od 0 do n, kde n je počet stupňů volnosti robota. Číslujeme od nepohyblivého členu – od základny.
2. Osa $z_i$ je osou rotace nebo posunu členu $i+1$ vzhledem k členu $i$. U posuvného spojení volíme kladný směr s kladným směrem posunutí.
3. Osa $x_i$ vznikne prodloužením společné kolmice os $z_{i-1}$ a $z_i$. Osu $x_0$ lze zvolit libovolně. Pokud jsou osy $z_{i-1}$ a $z_i$ totožné, je vhodné zavést osu $x_i$ orientovanou shodně s osou $x_{i-1}$.
4. Osa $y_i$ je kolmá na osu $x_i$ a $z_i$ tak, aby vzniklý souřadný systém byl pravotočivý.
5. Osu $z_n$ volíme jako přístupový (access) vektor chapadla a $y_n$ volíme jako vektor orientace (orientation) chapadla. Význam těchto vektorů je vysvětlen na obr. 2.

Podle těchto pravidel získáme $n+1$ souřadných systémů, kde $n$ je počet stupňů volnosti robota. Jsou-li tyto systémy správně navrženy, lze se dostat ze systému $i$ do systému *i-1* pomocí čtyř základních transformací:

1. Rotací podle osy $z_{i-1}$ o úhel $\varphi_i$. Osy $x_i$ a $x_{i-1}$ tak budou rovnoběžné.
2. Translací ve směru osy $z_{i-1}$ o vzdálenost $s_i$. Osy $x_i$ a $x_{i-1}$ tak budou totožné.
3. Rotací podle osy $x_i$ o úhel $\alpha_i$. Osy $z_i$ a $z_{i-1}$ tak budou rovnoběžné.
4. Translací ve směru osy $x_i$ o vzdálenost $a_i$. Systémy $i$ a *i-1* budou shodné.

Z uvedených transformací lze sestavit univerzální transformační matici mezi sousedními systémy, do níž se budou zadávat pouze 4 parametry $\varphi$, $s$, $\alpha$, $a$. Ukažme si tedy přesný význam jednotlivých parametrů:

$\varphi_i$ ... natočení osy $x_i$ vůči ose $x_{i-1}$ okolo osy $z_{i-1}$. U rotačního uložení je tento parametr proměnný.

$s_i$ ... vzdálenost os $x_{i-1}$ a $x_i$ podél osy $z_{i-1}$. U posuvného uložení je tento parametr proměnný.

$\alpha_i$ ... natočení osy $z_{i-1}$ vůči ose $z_i$ okolo osy $x_i$.

$a_i$ ... vzdálenost os $z_{i-1}$ a $z_i$ podél osy $x_i$.

Při pohybu kinematického členu robota se bude v transformační matici měnit pouze jeden parametr. Pro translaci to bude parametr $s_i$ a pro rotaci $\varphi_i$. Zbylé tři parametry jsou dány konstrukcí robota a budou konstantní.

Pro přechod z jednoho souřadného systému do druhého využívá DH homogenních transformačních matic 4x4. Homogenní transformační matice $A$ se skládá z matice rotace ($R$ 3x3) a z vektoru posunutí ($\vec{P}$):

$$A = \left(\begin{array}{ccc|c} & R & & \vec{P} \\ \hline 0 & 0 & 0 & 1 \end{array}\right) \tag{1}$$

Univerzální transformační matici $A_i^{i-1}$ pro přechod ze systému $i$ do systému *i-1* získáme vynásobením matic, které v sobě zahrnují čtyři výše zmiňované základní transformace:

$$A_i^{i-1} = \begin{pmatrix} \cos(\varphi_i) & -\sin(\varphi_i) & 0 & 0 \\ \sin(\varphi_i) & \cos(\varphi_i) & 0 & 0 \\ 0 & 0 & 1 & s_i \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 & a_i \\ 0 & \cos(\alpha_i) & -\sin(\alpha_i) & 0 \\ 0 & \sin(\alpha_i) & \cos(\alpha_i) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{2}$$

Výsledná transformační matice má pak tvar:

$$A_i^{i-1} = \begin{pmatrix} \cos(\varphi_i) & -\sin(\varphi_i)\cos(\alpha_i) & \sin(\varphi_i)\sin(\alpha_i) & a_i\cos(\varphi_i) \\ \sin(\varphi_i) & \cos(\varphi_i)\cos(\alpha_i) & -\cos(\varphi_i)\sin(\alpha_i) & a_i\sin(\varphi_i) \\ 0 & \sin(\alpha_i) & \cos(\alpha_i) & s_i \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3}$$

Použitím DH konvence takto získáme $n$ transformačních matic mezi sousedními souřadnými systémy, kde $n$ je počet stupňů volnosti robota.

## 1.2 Zobecněné souřadnice (kloubové souřadnice)

Každá DH transformační matice $A_i^{i-1}$ v sobě obsahuje jeden parametr, který je závislý na aktuálním natočení či posunu příslušného členu robota. Zavedeme vektor $\vec{q}$, jenž v sobě bude obsahovat pouze proměnné DH parametry, a nazveme ho vektorem zobecněných souřadnic (někdy i vektorem kloubových souřadnic). Tento vektor $\vec{q}$ v sobě ponese informaci o aktuální poloze všech kinematických členů robota.

Označme tedy:

$$\vec{q} = \left[q_1, q_2, ..., q_n\right]^T, \text{ kde } q_i = \sigma_i\varphi_i + \left(1-\sigma_i\right)s_i \tag{4}$$

| | | | |
|---|---|---|---|
| $\vec{q}$ | vektor zobecněných souřadnic | $\sigma_i = 0$ | pro rotační dvojici |
| $q_i$ | *i-tá* zobecněná souřadnice | $\sigma_i = 1$ | pro translační dvojici |
| $\varphi_i, s_i$ | jsou DH parametry | $i$ = 1, …, n | |

Vektor zobecněných souřadnic $\vec{q}$ je důležitý jak pro přímou, tak pro inverzní kinematickou úlohu. Jeho časová závislost $\vec{q}_{(t)}$ naznačuje pohyb jednotlivých členů robota, první časová derivace $\dot{\vec{q}}_{(t)}$ rychlost členů a druhá derivace $\ddot{\vec{q}}_{(t)}$ pak zrychlení.

## 1.3 Pracovní prostor robota

Pracovní prostor robota je dán množinou všech možných dosažitelných poloh a natočení chapadla. Vezmeme-li v potaz pouze polohu chapadla, pak tvar pracovního prostoru bude připomínat nějaké 3D těleso. Například jednoduchá kinematická struktura RTT bude mít pracovní prostor ve tvaru válce, popřípadě jeho části (dle rozsahů pohonů).

V této práci bude nadále pracovní prostor robota definován pomocí vektoru zobecněných souřadnic $\vec{q}$. A to tak, že pro každý z prvků vektoru $q_1, q_2, ..., q_n$ uvedeme jeho rozsah povolených hodnot.

# 2 Přímá kinematická úloha

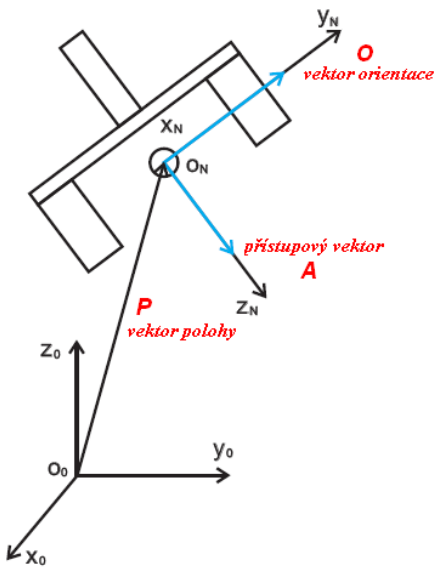
Úkolem přímé kinematické úlohy je nalézt polohu a natočení koncového zařízení robota, nejčastěji chapadla, v souřadném systému základny. Máme zadaný vektor zobecněných souřadnic $\vec{q}$ a známe DH parametry robotické struktury.

## 2.1 Poloha a natočení chapadla

Mezi souřadným systémem chapadla a souřadným systémem základny existuje homogenní transformační matice $A_n^0$. Tato matice obsahuje submatici rotace a vektor polohy (vztah 1). Rozeberme-li submatici rotace dále na jednotlivé vektory, získáme následující tvar matice.

$$A_n^0 = \left(\begin{array}{ccc|c} \vec{O}\times\vec{A} & \vec{O} & \vec{A} & \vec{P} \\ 0 & 0 & 0 & 1 \end{array}\right) = \left(\begin{array}{ccc|c} \vec{N} & \vec{O} & \vec{A} & \vec{P} \\ 0 & 0 & 0 & 1 \end{array}\right) \tag{5}$$

Složka $\vec{A}$ je přístupový vektor chapadla (access vector), $\vec{O}$ je vektor orientace (orientation vector, někdy i vektor stisku), třetí vektor je pouhým vektorovým součinem vektorů $\vec{O}$ a $\vec{A}$ (někdy i normálový vektor $\vec{N}$). Význam vektorů je znázorněn na následujícím obrázku č. 2.




*Obr. 2: Poloha a natočení koncového členu robota vůči systému základny.*

Všechny vektory, které jsou součástí submatice rotace (vektory $\vec{N}$, $\vec{O}$, $\vec{A}$), jsou jednotkové a tvoří ortonormální souřadný systém, jenž je natočený oproti souřadnému systému základny a posunutý o polohový vektor $\vec{P}$. Toto natočení lze popsat pomocí tří úhlů $\alpha, \beta, \gamma$.

Existuje více možností, jak definovat natočení pomocí tří úhlů. V této práci bude orientace chapadla definována pomocí Eulerových úhlů $\alpha, \beta, \gamma$ (rotace podle osy $z$, pak podle osy $y$ a nakonec opět podle nově vzniklé osy $z$). Matice rotace pak bude mít tvar:

$$R = \begin{pmatrix} \cos(\gamma) & -\sin(\gamma) & 0 \\ \sin(\gamma) & \cos(\gamma) & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos(\beta) & 0 & -\sin(\beta) \\ 0 & 1 & 0 \\ \sin(\beta) & 0 & \cos(\beta) \end{pmatrix} \begin{pmatrix} \cos(\alpha) & -\sin(\alpha) & 0 \\ \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad (6)$$

$$R = \begin{pmatrix} \cos(\gamma)\cos(\beta)\cos(\alpha) - \sin(\gamma)\sin(\alpha) & -\cos(\gamma)\cos(\beta)\sin(\alpha) - \sin(\gamma)\cos(\alpha) & \cos(\gamma)\sin(\beta) \\ \sin(\gamma)\cos(\beta)\cos(\alpha) + \cos(\gamma)\sin(\alpha) & -\sin(\gamma)\cos(\beta)\sin(\alpha) + \cos(\gamma)\cos(\alpha) & \sin(\gamma)\sin(\beta) \\ -\sin(\beta)\cos(\alpha) & \sin(\beta)\sin(\alpha) & \cos(\beta) \end{pmatrix} \quad (7)$$

## 2.2 Výpočet přímé kinematické úlohy

Známe-li všechny DH parametry robota a konkrétní vektor zobecněných souřadnic $\vec{q}$, známe také všechny transformační matice ${}^{i-1}_{i}A_{(qi)}$, kde $i = 0..n$ a $n$ je počet stupňů volnosti robota. Výslednou transformační matici $A^0_{n(\vec{q})}$ pro přepočet souřadnic z koncového systému chapadla do systému základny získáme tak, že postupně mezi sebou vynásobíme jednotlivé transformační matice ${}^{i-1}_{i}A_{(qi)}$ v přesně daném pořadí, protože násobení matic není komutativní. Vztah pro transformační matici $A^0_{n(\vec{q})}$ má tvar:

$$A^0_{n(\vec{q})} = {}^{0}_{1}A_{(q1)} A^1_{2(q2)} \ldots A^{n-1}_{n\ (qn)} \quad (8)$$

Z kapitoly (2.1) víme, že matice $A^0_{n(\vec{q})}$ v sobě uchovává informaci o poloze a natočení chapadla v souřadném systému základny. Zavedeme vektor $\vec{X} = \left[x, y, z, \alpha, \beta, \gamma\right]^T$, jehož složky $x, y, z$ vyjadřují polohu chapadla v kartézském souřadném systému základny. Složky $\alpha, \beta, \gamma$ definují natočení chapadla pomocí Eulerových úhlů. Cílem bude nalézt funkce, které přepočtou vektor zobecněných souřadnic $\vec{q}$ na vektor $\vec{X}$. Vznikne tedy vztah:

$$\vec{X} = \vec{F}\left(\vec{q}\right) \quad (9)$$

Víme, že poslední sloupec transformační matice $A^0_{n(\vec{q})}$ je vektorem polohy $\vec{P} = \left[ P_x, P_y, P_z \right]^T$ chapadla v systému základny (vztah 5). Můžeme tedy rovnou psát:

$$\begin{aligned} x &= F_{x(\vec{q})} = A^n_{0\,(1,4)(\vec{q})} = P_{x(\vec{q})} \\ y &= F_{y(\vec{q})} = A^n_{0\,(2,4)(\vec{q})} = P_{y(\vec{q})} \\ z &= F_{z(\vec{q})} = A^n_{0\,(3,4)(\vec{q})} = P_{z(\vec{q})} \end{aligned} \tag{10}$$

Jedná se o různé kombinace goniometrických funkcí. Tvar a složitost těchto funkcí záleží na konstrukci robota, především na počtu rotačních kinematických dvojic. S růstem rotačních dvojic roste složitost.

Určení orientace chapadla spočívá pouze v přepočtení submatice rotace $R$ transformační matice $A^0_{n(\vec{q})}$ na Eulerovo úhly $\alpha, \beta, \gamma$. Vyjdeme tedy ze vztahů (5) a (7).

$$\beta = \arccos\left( A^n_{0\,(3,3)(\vec{q})} \right); A^n_{0\,(3,3)(\vec{q})} \in \langle -1;1 \rangle \tag{11}$$

$$\begin{aligned} \cos(\alpha) &= -\frac{A^n_{0\,(3,1)(\vec{q})}}{\sin(\beta)}; \sin(\alpha) = \frac{A^n_{0\,(3,2)(\vec{q})}}{\sin(\beta)} \\ \cos(\gamma) &= \frac{A^n_{0\,(1,3)(\vec{q})}}{\sin(\beta)}; \sin(\gamma) = \frac{A^n_{0\,(2,3)(\vec{q})}}{\sin(\beta)} \end{aligned} ; \beta \neq k \cdot 180°; k \in \mathbb{Z}$$

Podmínka $A^n_{0\,(3,3)(\vec{q})} \in \langle -1;1 \rangle$ ve vztahu (11) je splněna vždy. Úhly $\alpha$ a $\gamma$ získáme pomocí funkce arcsin nebo arccos a výsledný úhel převedeme do správného kvadrantu dle znamének funkcí sin a cos.

Musíme ještě uvažovat nad výjimečnými případy, za kterých uvedené výrazy v (11) neplatí. Pro $\beta = k \cdot 180°$ dochází k nejednoznačnému určení úhlů $\alpha, \gamma$. Proto například úhel $\alpha$ volíme jako nulový a $\gamma$ vyjde po dosazení známých úhlů $\beta, \alpha$ do matice (7).

Je nutné si uvědomit, že přepočet vektoru zobecněných souřadnic na polohu a natočení koncového členu robota v systému základny je jednoznačný. Existuje vždy právě jedno řešení. Přímá kinematická úloha je tedy velmi jednoduše naprogramovatelná.

# 3 Inverzní kinematická úloha

Inverzní kinematická úloha je opakem úlohy přímé. Jejím úkolem je pro známou polohu a orientaci chapadla v souřadném systému základny spočítat vektor kloubových souřadnic $\vec{q}$. Jde nám tedy o určení natočení či posunutí všech pohonů robota tak, abychom dosáhli požadované polohy a orientace koncového členu. Opět známe konkrétní DH parametry robota.

Inverzní úloha je v robotice velmi důležitá a žádaná, protože je pro nás jednodušší a přirozenější programovat polohování robota v souřadném systému základny než v zobecněných souřadnicích. Je ale oproti přímé úloze složitější. Ve většině případů je totiž nutné řešit složité soustavy nelineárních rovnic. Přímá úloha vedla vždy k jednoznačnému řešení. U inverzní úlohy tomu ale tak není. Může existovat jedno, více, nekonečně mnoho, nebo dokonce žádné řešení.

## 3.1 Metody řešení inverzní úlohy

Na metody pro řešení inverzní úlohy jsou kladeny různé požadavky. Hlavní důraz je kladen na rychlost výpočtu, aby bylo možné řídit robota v reálném čase. Dále se požaduje, aby daná metoda byla univerzální, a mohla tak být použita pro jakoukoli robotickou strukturu s různým počtem stupňů volnosti.

Základní dělení metod:

- Analytické metody
- Numerické metody
    - Aproximační (metody pro řešení nelineárních soustav)
    - Optimalizační (metody převádějící soustavu na minimalizaci funkcionálu)

### 3.1.1 Analytické metody

Analytické metody jsou výhodné v tom, že naleznou řešení velmi rychle. Používají se proto v oblasti řízení v reálném čase. Nalezené řešení inverzní úlohy analytickou metodou je přesné – tedy za předpokladu, že se při jejím odvozování nepoužilo nějaké zjednodušení. Mají ale velkou nevýhodu. Pro každou konkrétní robotickou strukturu je

nutné odvozovat zvláštní vztahy pro výpočet kloubových souřadnic $\vec{q}$. Nejsou tedy univerzální. Odvozování je často velmi zdlouhavé, složité a komplikované. Využívá se různých úprav pomocí goniometrických vztahů. Musí se ošetřovat různé podmínky, protože ne za všech okolností odvozené vztahy platí (dělení nulou, apod.). Pro hledání analytického řešení neexistuje obecný postup. Může se dokonce stát, že se ani k analytickému řešení nedopracujeme. Nemusí totiž obecně existovat. Postačující podmínka pro řešitelnost robotické struktury je například to, že pro robota se šesti stupni volnosti tři po sobě jdoucí rotační klouby mají osy protínající se v jednom bodě nezávisle na pohybu, nebo tři po sobě jdoucí osy musí být rovnoběžné [3].

Vychází se z rovnice (12). Z této soustavy postupně odvozujeme vztahy pro jednotlivé prvky vektoru kloubových souřadnic $\vec{q}$. Názornou ukázku analytického řešení si ukážeme na kinematické struktuře RRT-R robota RHINO SCARA (kapitola 6.4).

**Příklad použití analytické metody pro robota RHINO SCARA:**

Vynásobíme-li transformační matice se známými DH parametry pro jednotlivé členy robota podle vztahu (8), získáme následující matici $A^0_{4(\vec{q})}$:

$$A^0_{4(\vec{q})} = \begin{pmatrix} \cos(q_1+q_2+q_4) & \sin(q_1+q_2+q_4) & 0 & a_2\cos(q_1+q_2)+a_1\cos(q_1)=x_D \\ \sin(q_1+q_2+q_4) & -\cos(q_1+q_2+q_4) & 0 & a_2\sin(q_1+q_2)+a_1\sin(q_1)=y_D \\ 0 & 0 & -1 & q_3+s_1=z_D \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Z této matice lehce získáme vztah pro $q_3$:

$$q_3 = z_D - s_1$$

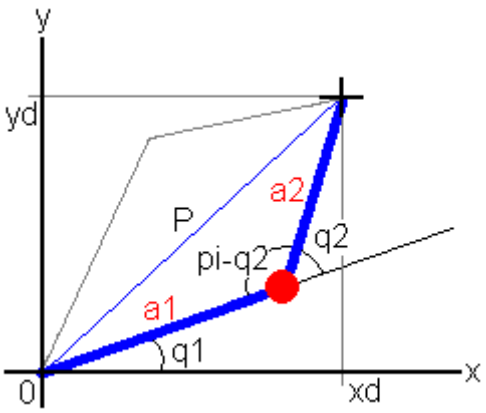
Pro odvození $q_4$ vyjdeme ze vztahu (7), který vyjadřuje submatici rotace pomocí Eulerových úhlů. Dosadíme-li za $\alpha = 0°, \beta = 180°$ (ty jsou pevně dané) a dorovnáme-li znaménka pomocí úhlu 180°, získáme vztah:

$$q_1 + q_2 + q_4 = \gamma + 180°$$
$$q_4 = 180° + \gamma - q_1 - q_2$$

Pro vyjádření $q_1$ a $q_2$ nám pak stačí vyřešit soustavu:

$$x_D = a_2 \cos(q_1 + q_2) + a_1 \cos(q_1)$$
$$y_D = a_2 \sin(q_1 + q_2) + a_1 \sin(q_1)$$

Nakresleme si obrázek situace, kterou tato soustava popisuje. S jeho pomocí si pomůžeme při řešení soustavy.




*Obr. 3: Schéma polohování pomocí q1 a q2 v rovině xy u robota RHINO SCARA.*

Pro výpočet $q_2$ nám pomůže trojúhelník daný stranami $P, a_1, a_2$. Parametry $a_1, a_2$ jsou známé z DH konvence. Stranu $P$ spočítáme z požadovaných souřadnic $x_D, y_D$ pomocí Pythagorovy věty ( $P = \sqrt{x_D{}^2 + y_D{}^2}$ ). Z kosinové věty nám pak vyjde vztah:

$$P^2 = x_D{}^2 + y_D{}^2 = a_1{}^2 + a_2{}^2 - 2a_1 a_2 \cos(180° - q_2)$$
$$q_2 = \arccos\left( \frac{x_D{}^2 + y_D{}^2 - a_1{}^2 - a_2{}^2}{-2a_1 a_2} \right)$$

Pro $q_2$ nám tedy vyjdou dvě řešení. Vyplývá to i z obr. 3, kde je možné se do požadované polohy dostat dvěma způsoby. Je ale nutné dodržet trojúhelníkovou nerovnost, aby argument funkce arccos byl v intervalu $\langle -1;1 \rangle$.

Souřadnici $q_1$ získáme dosazením již známého $q_2$ do původní soustavy rovnic. Upravením soustavy pomocí součtových goniometrických vztahů získáme:

$$x_D = \cos(q_1)\left(a_2 \cos(q_2) + a_1\right) + \sin(q_1)\left(a_2 \sin(q_2)\right) = \cos(q_1) \cdot A + \sin(q_1) \cdot B$$
$$y_D = \cos(q_1)\left(a_2 \sin(q_2)\right) + \sin(q_1)\left(a_2 \cos(q_2) + a_1\right) = \cos(q_1) \cdot B + \sin(q_1) \cdot A$$

Máme teď lehce řešitelnou soustavu lineárních rovnic pro členy $\cos(q_1)$ a $\sin(q_1)$. Lze tedy psát:

$$\cos(q_1) = \frac{Ax_D + By_D}{A^2 + B^2}; \sin(q_1) = \frac{y_D - B\cos(q_1)}{A}$$

Je ale nutné speciálně řešit soustavu pro $A = 0$ a $B = 0$. Pokud bude $A = 0$ i $B = 0$, soustava bude mít řešení pouze tehdy, pokud bude žádaná poloha $x_D = 0$ a $y_D = 0$.

$$A = 0, B \neq 0 \rightarrow \cos(q_1) = \frac{y_D}{B}; \sin(q_1) = -\frac{x_D}{B}$$
$$B = 0, B \neq 0 \rightarrow \cos(q_1) = \frac{x_D}{A}; \sin(q_1) = \frac{y_D}{A}$$

Souřadnici $q_1$ spočítáme buď pomocí funkce arccos nebo arcsin a úhel převedeme podle znamének $\cos(q_1)$ a $\sin(q_1)$ do správného kvadrantu. Vyjde nám jedno řešení pro každé $q_2$.

Nejčastěji nám teoreticky vyjdou dvě řešení (výjimečně jedno, pokud bude $q_2 = 0°$). Pokud ale nalezené řešení kloubových souřadnic nebude v pracovní oblasti robota, počet řešení se sníží. Musíme tedy u každé kloubové souřadnice kontrolovat, zda se nachází ve svém povoleném intervalu.

### 3.1.2 Úvod do numerických metod

Nevýhody analytických metod pro řešení inverzní kinematické úlohy vedly k nutnosti nalézt jiné robustní metody, které budou použitelné pokud možno pro libovolné kinematické struktury. Je také potřeba, aby dostatečně rychle a s co největší přesností konvergovaly k hledanému řešení. Tyto vlastnosti nám nabízejí právě numerické metody.

Numerické metody jsou často dobře algoritmizovatelné. Je ale potřeba dávat pozor na jejich stabilitu a dodržovaní podmínek, za kterých konvergují. Problémy často nastávají v singulárních bodech (kap. 3.2). Numerickou metodou je často nějaký iterační proces, jehož konečným opakováním se více a více přibližujeme k řešení. Je tedy časté, že nenalezneme úplně přesné řešení jako u analytických metod. Přesnost nalezeného řešení lze ale u numerických metod volit. Vybrané numerické metody jsou popsány v následujících kapitolách.

### 3.1.3 Aproximační numerické metody

Máme-li zadanou požadovanou polohu a orientaci chapadla, tedy vektor $\vec{X} = [x, y, z, \alpha, \beta, \gamma]^T$, můžeme číselně vyjádřit transformační matici mezi systémem chapadla a základny $A^0_{nD}$. Je to pro nás matice daná. Ze vztahu (8) známe také matici $A^0_{n(\vec{q})}$ vyjádřenou pomocí zobecněných souřadnic $\vec{q}$ a ostatních DH parametrů robota. Dáme-li obě matice do rovnosti, vznikne rovnice:

$$A^0_{nD} = A^0_{n(\vec{q})} \tag{12}$$

Dáme-li do rovnosti sobě odpovídající prvky matic, dostaneme soustavu šestnácti rovnic o $n$ neznámých, kde $n$ je počet stupňů volnosti robota. Protože jsou obě matice homogenními transformacemi, čtyři rovnice z posledního řádku nám v řešení nepomůžou. Zbývá nám tedy porovnat polohové vektory (3 rovnice) a submatice rotace (9 rovnic). Protože sloupce submatice rotace vyjadřují ortogonální souřadný systém, jsou rovnice vyplývající ze submatice rotace lineárně závislé. Ze vztahu (12) nám tedy vznikne maximálně 6 lineárně nezávislých rovnic. Aby byla soustava řešitelná, může být použit robot s maximálně šesti stupni volnosti.

Aproximační metody vycházejí ze známého bodu. Známý bod může být například výchozí poloha robota, kde lze vektor zobecněných souřadnic zjistit pomocí čidel polohy. Lze použít i referenční bod robota. Do hledaného bodu se dostaneme pomocí nějaké aproximace. Metody vycházejí z rovnice (12).

**Aproximace pomocí Taylorova rozvoje**

Víme, že matice $A^0_{n(\vec{q})}$ je funkcí vektoru zobecněných souřadnic $\vec{q}$. Protože jsou všechny tyto funkce diferencovatelné, lze funkce v okolí známého bodu $\vec{q}$ vyjádřit pomocí Taylorova rozvoje. Použijí se pouze první dva členy, ostatní se zanedbají.

$$A^0_{nD} = A^0_n(q_1 + \Delta q_1, ..., q_n + \Delta q_n) = A^0_n(q_1, ..., q_n) + \sum_{i=1}^{n} \frac{\partial A^0_n(q_1, ..., q_n)}{\partial q_i} \Delta q_i \tag{13}$$

Matici na levé straně rovnice číselně známe, je to žádaná poloha. Matici $A_n^0(q_1,...,q_n)$ známe také, je to výchozí poloha. Matici $\frac{\partial A_n^0(q_1,...,q_n)}{\partial q_i}$, $i=1,2,...,n$ lze také číselně vyjádřit. Neznámými jsou tedy pouze přírůstky $\Delta q_i$. Jedná se o lineární soustavu $n$ rovnic o $n$ neznámých, kterou lze řešit běžnými metodami. Vyčíslíme-li přírůstky $\Delta q_i$, snadno získáme nový bod $\vec{q}$, který bude novým výchozím bodem. Takto postupujeme, až metoda dokonverguje k požadovanému řešení. Omezením této medy je, že získaná soustava lineárních rovnic musí být regulární.

**Newtonova aproximační metoda**

Newtonova metoda řeší nelineární soustavu ve tvaru:

$$\vec{F}(\vec{q})-\vec{X}=\vec{0}\text{, kde } \vec{X}=\left[x,y,z,\alpha,\beta,\gamma\right]^T \text{ a } \vec{F}=\left[f_{1(\vec{q})},...,f_{n(\vec{q})}\right]^T \tag{14}$$

Je tedy nutné upravit soustavu tak, aby byl na pravé straně nulový vektor. Při aproximačním řešení se využívá Jacobiovy matice, která v sobě obsahuje parciální derivace jednotlivých funkcí ze vztahu (14) podle jednotlivých zobecněných souřadnic:

$$J_{(\vec{q})}=\begin{pmatrix} \frac{\partial f_1}{\partial q_1} & \cdots & \frac{\partial f_1}{\partial q_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial q_1} & \cdots & \frac{\partial f_n}{\partial q_n} \end{pmatrix} \tag{15}$$

Aby Newtonova aproximační metoda konvergovala k řešení, je nutné znát přibližný odhad řešení. Tento odhad se použije jako počáteční bod metody $\vec{q}^{\,0}$. Dále se požaduje, aby determinant Jacobiovy matice byl v oblasti řešení nenulový, tedy aby nedocházelo k singularitám (kap. 3.2). Pokud jsou tyto podmínky splněny, lze napsat algoritmus Newtonovy metody takto:

1) $k=0$

   Nastavení iteračního parametru na nulu.

2) $\delta=-J(\vec{q}^{\,k})^{-1}\cdot\vec{F}\left(\vec{q}^{\,k}\right)$

   Vypočtení přírůstku $\delta$ z inverzní Jacobiovy matice a funkce $\vec{F}$ v bodě $\vec{q}^{\,k}$

3) $\vec{q}^{\,k+1}=\vec{q}^{\,k}+\delta$

   Výpočet nového vektoru $\vec{q}^{\,k+1}$

4) $\left|\vec{F}\left(\vec{q}^{k+1}\right)\right| < \varepsilon$

   Je nově získaný vektor $\vec{q}^{k+1}$ dostatečně blízko hledanému řešení? Pokud ano, iteraci ukončíme a $\vec{q}^{d} = \vec{q}^{k+1}$. Vektor $\vec{q}^{d}$ je řešením. Pokud ne, pokračujeme bodem 5)

5) $k = k + 1$

   Zvýšení iteračního parametru o 1 a pokračuje se znovu bodem 2)

Newtonova metoda konverguje k řešení kvadraticky. Pokud počáteční odhad $\vec{q}^{0}$ bude od hledaného řešení příliš daleko, může metoda divergovat. Pro příklad je v příloze této práce uveden jednoduchý program pro aplikaci Matlab, který řeší inverzní kinematickou úlohu pro robota RHINO SCARA (kapitola 6.4).

### 3.1.4 Optimalizační numerické metody

Optimalizační numerické metody mění zobecněné souřadnice $\vec{q}$ tak, aby došlo k co nejmenší chybě polohování. Změna $\vec{q}$ je prováděna fiktivním směrem. Po každé změně je vyhodnocována chyba polohování.

Mějme opět zadanou požadovanou polohu a orientaci chapadla robota. Známe tedy transformační matici $A^{0}_{n\,D}$ (matice daná) mezi souřadným systémem chapadla a souřadným systémem základny. Známe-li také obecnou transformační matici mezi systémem chapadla a základny $A^{0}_{n(\vec{q})}$, která je závislá na vektoru zobecněných souřadnic robota $\vec{q}$, můžeme inverzní kinematickou úlohu převést na řešení minimalizace funkcionálu daného vztahem:

$$I^{2}_{(\vec{q})} = \left| A^{0}_{n\,D} - A^{0}_{n(\vec{q})} \right|^{2} \tag{16}$$

Je zřejmé, že funkcionál nemůže nabývat záporných hodnot. Dále, pokud bude existovat takové $\vec{q}$ z pracovního rozsahu robota, pro které bude daný funkcionál nulový, je pro nás toto $\vec{q}$ řešením inverzní kinematické úlohy. Už tedy známe konkrétní velikost hledaného minima funkcionálu. Hodnota funkcionálu je pro nás chyba polohování – tedy hlavní kritérium pro optimalizační metody.

**Gradientní metody**

Jak nám název napovídá, gradientní metody pro hledání minima obecné funkce $f(\vec{x})$ více proměnných využívají gradientu funkce. Pro inverzní kinematickou úlohu bude funkcí $f(\vec{x})$ druhá mocnina funkcionálu $I_{(\vec{q})}$ a vektor $\vec{x}$ bude nahrazen zobecněnými souřadnicemi $\vec{q}$ (vztah 16). Záporný gradient funkce $-\nabla f(\vec{x})$ nám udává směr největšího spádu v bodě $\vec{x}$. Vydáme-li se tímto směrem do nějakého dalšího bodu $\vec{x}'$, který je ale od původního bodu vhodně daleko, bude hodnota funkce v bodě $\vec{x}'$ menší než v bodě $\vec{x}$. Zopakujeme-li tento postup pro bod $\vec{x}'$, dostaneme se do dalšího bodu $\vec{x}''$, pro který bude hodnota funkce ještě menší. Přechod mezi jednotlivými body ve směru záporného gradientu funkce lze popsat následujícím iteračním vztahem:

$$\vec{x}_{k+1} = \vec{x}_k - \lambda_k \cdot \nabla f(\vec{x}_k); \text{ kde } \lambda_k > 0, k = 0,1,2,... \tag{17}$$

Na začátku je nutné zvolit nějaký vhodný počáteční odhad řešení $\vec{x}_0$. Počáteční bod se volí odhadem podle toho, kde přibližně očekáváme minimum funkce. Má-li funkce více minim, zvolením počátečního odhadu se rozhodne, do jakého konkrétního minima gradientní metoda sklouzne. Chceme-li nalézt všechna minima, je nutné provézt gradientní metodu pro více počátečních odhadů. Dalším problémem je vhodné zvolení jednotlivých kroků $\lambda_k$. Příliš malá $\lambda_k$ zpomalují hledání minima, naopak příliš velká $\lambda_k$ vedou k oscilacím. Podle volby kroků $\lambda_k$ se gradientní metody dělí na dvě základní části:

a) Gradientní metoda s krátkým krokem
b) Gradientní metoda s dlouhým krokem

Gradientní metoda s krátkým krokem používá $\lambda_k$ velmi malá. Tato metoda hledání minima funkce je pomalá. Navíc, pokud je $\lambda_k$ v průběhu iterací konstantní, bude metoda v blízkosti minima funkce oscilovat. Proto se doporučuje průběžně kroky $\lambda_k$ zmenšovat.

Gradientní metoda s dlouhým krokem (známá také jako metoda největšího spádu nebo metoda s optimálním krokem) volí co největší krok $\lambda_k$ tak, aby došlo k co největšímu snížení hodnoty funkce $f(\vec{x})$. Jde vlastně o úlohu:

$$\min_{\lambda>0} \left( f\left[ \vec{x}_k - \lambda_k \cdot \nabla f(\vec{x}_k) \right] \right) \tag{18}$$

Optimální krok lze pak spočítat dle vztahu:

$$\lambda_{k\_opt} = \frac{\nabla^T f(\vec{x}_k) \cdot \nabla f(\vec{x}_k)}{\nabla^T f(\vec{x}_k) \cdot H(\vec{x}_k) \cdot \nabla f(\vec{x}_k)} \tag{19}$$

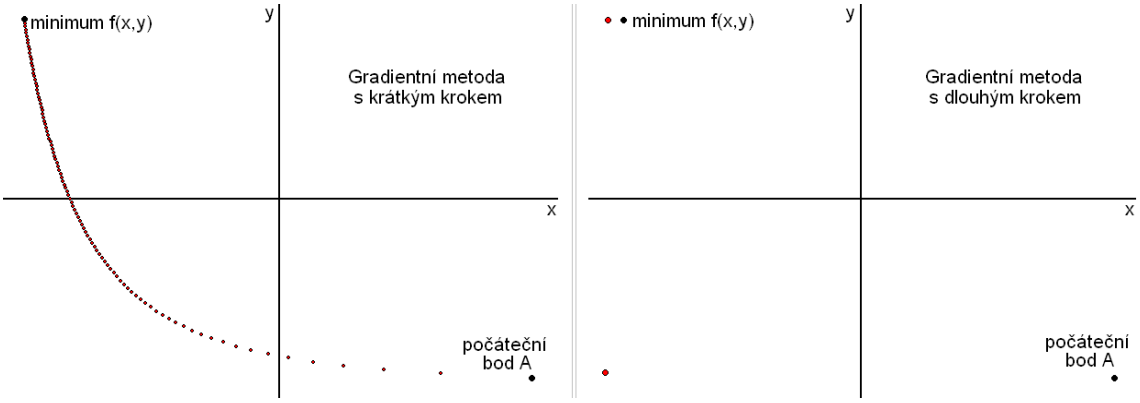
Člen $H(\vec{x}_k)$ ve vztahu (19) je tzv. Hessova matice. Je to matice druhých parciálních derivací podle složek vektoru $\vec{x}$.

$$H(\vec{x}) = \nabla\nabla f(\vec{x}) = \begin{pmatrix} \frac{\partial^2 f_{(\vec{x})}}{\partial x_1^2} & \cdots & \frac{\partial^2 f_{(\vec{x})}}{\partial x_1 \partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f_{(\vec{x})}}{\partial x_n \partial x_1} & \cdots & \frac{\partial^2 f_{(\vec{x})}}{\partial x_n^2} \end{pmatrix} \tag{20}$$

V každém kroku vztahu (17) je tedy nutné počítat optimální krok $\lambda_k$. Lze dokázat, že dva po sobě spočítané gradienty $\nabla f(\vec{x}_k)$ a $\nabla f(\vec{x}_{k+1})$ za použití optimálního kroku jsou na sebe kolmé. Jejich skalární součin je nulový. Platí tedy vztah:

$$\nabla^T f(\vec{x}_{k+1}) \cdot \nabla f(\vec{x}_k) = 0 \tag{21}$$

Příklad přibližování se z počátečního bodu $A = [x, y]$ do minima polynomiální funkce dvou proměnných $f(x, y)$ gradientními metodami s krátkým a dlouhým krokem je na obr. 4. Je zde vidět zdlouhavá konvergence u gradientní metody s krátkým krokem a navzájem ortogonální sousední gradienty u metody s dlouhým krokem.




*Obr. 4: Příklad průběhu konvergence u gradientní metody s dlouhým a krátkým krokem do minima funkce dvou proměnných f(x,y).*

**Newtonova metoda**

Newtonova metoda je další metodou využívající gradientu funkcionálu (16). Pro rychlejší konvergenci se zde ale využívá druhých derivací. Použijeme první dva členy Taylorova rozvoje minimalizované funkce $f(\vec{x})$ a položíme ho rovno nule, protože gradient funkce v místě minima je nulový. Získáme tedy:

$$\nabla f(\vec{x}+\Delta\vec{x}) = \nabla f(\vec{x}) + \nabla^2 f(\vec{x})^T \cdot \Delta\vec{x} = 0 \tag{22}$$

Z toho se $\Delta\vec{x}$ rovná:

$$\Delta\vec{x} = -\nabla^2 f(\vec{x})^{-1} \cdot \nabla f(\vec{x}) \tag{23}$$

Můžeme pak psát výsledný iterační vztah pro Newtonovu optimalizační metodu ve tvaru:

$$\vec{x}_{k+1} = \vec{x}_k - \nabla^2 f(\vec{x}_k)^{-1} \cdot \nabla f(\vec{x}_k) = \vec{x}_k - H(\vec{x}_k)^{-1} \cdot \nabla f(\vec{x}_k) \tag{24}$$

Člen $H(\vec{x}_k)^{-1}$ je inverzní Hessova matice (20). Je proto nutné, aby byla Hessova matice regulární. Dále je potřeba, aby počáteční bod $\vec{x}_0$ nebyl od extrému příliš vzdálen. Mohlo by pak dojít k divergenci. Metoda konverguje k řešení kvadraticky. Protože je výpočet inverzní matice poměrně zdlouhavý, používá se z počátku jiná metoda, která se k řešení přiblíží. Až poté se nasadí Newtonova metoda, jež pak v několika málo krocích dokonverguje do minima funkce.

**Heuristická Gaussova metoda**

Gaussova metoda je založena na fiktivním polohování jednotlivých členů robota. Nejprve se zablokují všechny pohony robota kromě posledního. V jeho rozsahu se nalezne poloha s nejmenší chybou polohy a orientace. Dále se zablokují všechny pohony kromě předposledního. Opět se nalezne poloha s nejmenší chybou. Takto se postupuje až k prvnímu pohonu. Celý postup se dále opakuje, dokud se nenalezne takový vektor zobecněných souřadnic, který docílí požadované polohy chapadla s danou přesností.

**Některé další numerické metody:**

Metoda zkušebního kroku, metoda konjugovaných gradientů, metoda Partan, Partan největšího spádu, Davidov-Fletcher-Powelová metoda a další…

## 3.2 Singulární body

Máme-li soustavu rovnic, ze které řešíme inverzní kinematickou úlohu, jedná se vlastně obecně o nějaké nelineární zobrazení z prostoru zobecněných souřadnic do prostoru požadované polohy a orientace. Každému vektoru zobecněných souřadnic je přiřazena jedna poloha a orientace. Zobecněné souřadnice jsou zde tedy vzorem, žádaná poloha a orientace symbolizuje obraz. Soustava pro robota se šesti stupni volnosti má tvar:

$$\begin{aligned} x &= f_1(q_1,\cdots,q_6) \\ y &= f_2(q_1,\cdots,q_6) \\ z &= f_3(q_1,\cdots,q_6) \\ &\vdots \\ \gamma &= f_6(q_1,\cdots,q_6) \end{aligned} \tag{25}$$

Je žádoucí, aby toto zobrazení bylo regulární. Například proto, že některé numerické metody pro řešení inverzní kinematické úlohy v singulárních oblastech selhávají. Zobrazení (25) je regulární tehdy, pokud funkce $f_1,...,f_6$ mají spojité parciální derivace prvního řádu podle všech zobecněných souřadnic $q_1,...,q_6$ a pokud Jacobiho funkcionální determinant je různý od nuly. Spojitost parciálních derivací je splněna z fyzikální podstaty polohování robota. Stačí nám tedy testovat nulovost Jacobiho determinantu. Jacobiova matice je definovaná vztahem (15). Je to matice prvních parciálních derivací funkcí $f_1,...,f_n$ podle zobecněných souřadnic $q_1,...,q_n$, kde $n$ je počet stupňů volnosti robota. Pro hledání singulárních bodů tedy obecně řešíme rovnici:

$$\det\left(J_{(\vec{q})}\right) = \det\begin{pmatrix} \frac{\partial f_1}{\partial q_1} & \cdots & \frac{\partial f_1}{\partial q_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial q_1} & \cdots & \frac{\partial f_n}{\partial q_n} \end{pmatrix} = 0 \tag{26}$$

Pokud je determinant (26) nenulový, zobrazení je regulární a v dostatečně malém okolí konkrétních zobecněných souřadnic $\vec{q}$ je zobrazení do prostoru požadované polohy a orientace navzájem jednoznačné. To však platí pouze lokálně, jednoznačnost nemusí platit v celém prostoru zobecněných souřadnic.

Ukažme si příklad vyhledávání singulárních bodů pro robotickou strukturu RRT-R robota RHINO SCARA (6.4).

Zobrazení pro robota RHINO SCARA je dáno soustavou:

$$\begin{aligned} x &= a_1 \cos(q_1) + a_2 \cos(q_1 + q_2) \\ y &= a_1 \sin(q_1) + a_2 \sin(q_1 + q_2) \\ z &= q_3 + s_1 \\ \gamma &= q_1 + q_2 + q_4 - 180 \end{aligned} \tag{27}$$

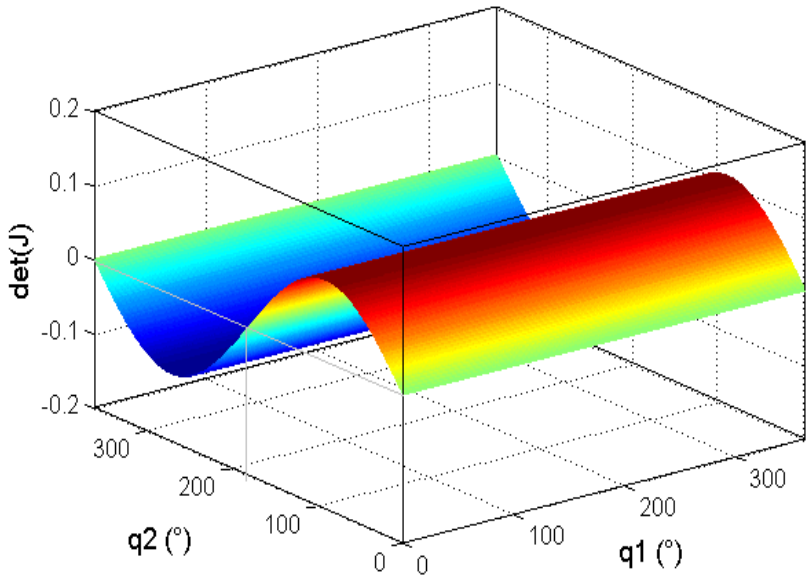
Jacobiho matice je rozměru 4x4, protože robot RHINO SCARA má čtyři stupně volnosti. Má tvar:

$$J = \begin{pmatrix} -a_1 \sin(q_1) - a_2 \sin(q_1 + q_2) & -a_2 \sin(q_1 + q_2) & 0 & 0 \\ a_1 \cos(q_1) + a_2 \cos(q_1 + q_2) & a_2 \cos(q_1 + q_2) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{pmatrix} \tag{28}$$

Počítáme-li determinant matice pomocí rozvoje podle posledního sloupce, zjistíme, že je závislý pouze na zobecněných souřadnicích $q_1$ a $q_2$:

$$\det(J) = a_1 a_2 \sin(q_1 + q_2)\cos(q_1) - a_1 a_2 \cos(q_1 + q_2)\sin(q_1) \tag{29}$$

Necháme-li si vykreslit průběh velikosti Jacobiho determinantu ve dvou dimenzionálním prostoru souřadnic $q_1$ a $q_2$, získáme následující graf:

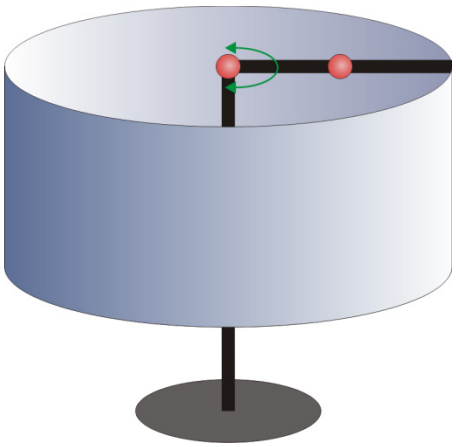



*Obr. 5: Průběh velikosti Jacobiho funkcionálního determinantu v závislosti na zobecněných souřadnicích q1 a q2 robota RHINO SCARA.*

Na obr. 5 sledujeme průchod plošného grafu nulou. Z grafu lze vyčíst, že robot se bude nacházet v singulárním bodě, pokud se $q_2 = 0 + k \cdot 180°$, kde $k$ jsou celá čísla. Na velikosti ostatních souřadnic $q_1, q_3, q_4$ nezáleží. Tuto vlastnost lze samozřejmě i analyticky odvodit. Upravíme-li vztah (29) pro determinant Jacobiánu pomocí goniometrických součtových vzorců a položíme ho rovno nule, vyjde nám:

$$\sin(q_2) = 0$$

To potvrzuje grafické řešení na obr. 5. Budeme-li uvažovat všechny možné kombinace $q_1, q_2, q_3, q_4$, pro které je determinant (29) nulový, zjistíme, že množina singulárních bodů je ve tvaru pláště válce, která tvoří hranici pracovního prostoru robota RHINO SCARA. Situační zjednodušené schéma singulární oblasti je na obr. 6. Uvažuje se zde pouze $q_2 = 0°$, protože jiná $q_2$, pro která je determinant (29) nulový, jsou mimo pracovní rozsah robota.



*Obr. 6: Singulární oblast robota RHINO SCARA ve tvaru pláště válce.*

Další jednoduše představitelný případ singularity může být takový, pokud při pohybu robota splynou dvě osy rotačních (translačních) členů. Pak pro nastavení orientace (polohy) existuje nekonečně mnoho řešení, tedy zobrazení z kloubových souřadnic do systému základny je singulární.

Inverzní úloha se v singulárních oblastech řeší obtížně. Problémy nastávají zejména při plánování trajektorie (kapitola 4). Pokud v singulárních oblastech existuje nekonečný počet řešení, máme velmi malou pravděpodobnost, že nalezené řešení inverzní úlohy bude tvořit spojitou trajektorii. Lze to řešit například tak, že v singulárních oblastech budeme robota řídit v úrovni kloubových souřadnic, tedy pomocí přímé úlohy. Nejlepší je ale se singulárním oblastem vyhýbat a programovat pohyb robota pouze v oblastech regulárních.

# 4 Plánování trajektorie

V kapitole (3) se řešil problém inverzní kinematické úlohy pro jednotlivé body. V praxi je ale často potřeba, aby robot vykonával pohyb po přesně dané trajektorii definovanou rychlostí, popř. zrychlením. Toho se například využívá v oblasti automatizovaného svařování. Proto je žádoucí umět plánovat pohyb robota v čase, tedy plánovat jeho trajektorii.

## 4.1 Plánování trajektorie učením

Plánování trajektorie robota učením se v praxi hodně využívá. Operátor pomocí joysticku nebo jiného ovládacího panelu navádí pracovní nástroj robota po žádané trajektorii. Robot si jednotlivé body trajektorie s určitým krokem zaznamenává do paměti – učí se ji.

Takové plánování je ale nepřesné. Využívá se v oblastech, kde je sice potřeba, aby robot konal pohyb po dané trajektorii, ale s hodně velkou tolerancí – například při přemisťování výrobků. Zde je nutné se vyhýbat překážkám a na velkou přesnost zde není kladen důraz. Je pouze nutné přesně dosáhnout cílového bodu.

## 4.2 Plánování trajektorie soustavou polohových bodů

Při tomto plánování uživatel zadává trajektorii pomocí jednotlivých bodů v souřadném systému základny, kterými má koncový člen robota projet. Bod je složen z polohy a orientace koncového členu podle počtu stupňů volnosti robota. Ke každému bodu musí být zadán časový okamžik, ve kterém se má robot v bodě ocitnout. Také je možné zadávat trajektorii pomocí bodů a rychlosti. Časové okamžiky jsou potom k jednotlivým bodům dopočítány. Chceme-li zadat trajektorii přesně, musí být soustava zadávaných bodů od sebe dostatečně blízko.

| **Xk** | **X0** | **X1** | **X2** | **…** | **XN** |
|---|---|---|---|---|---|
| $t_k$ (s) | $t_0$ | $t_1$ | $t_2$ | … | $t_N$ |
| $x_k$ (m) | $x_0$ | $x_1$ | $x_2$ | … | $x_N$ |
| $y_k$ (m) | $y_0$ | $y_1$ | $y_2$ | … | $y_N$ |
| $z_k$ (m) | $z_0$ | $z_1$ | $z_2$ | … | $z_N$ |
| … | … | … | … | … | … |
| $\gamma_k$ (°) | $\gamma_0$ | $\gamma_1$ | $\gamma_2$ | … | $\gamma_N$ |

*Tab. 1: Trajektorie zadaná soustavou bodů.*

## 4.3 Interpolace

Pro každý bod trajektorie $X_k$ je spočítána inverzní kinematická úloha. Tím získáme soustavu vektorů zobecněných souřadnic $\vec{q}_0,...,\vec{q}_N$ – tedy informaci o tom, jak mají být v čase jednotlivé pohony natočeny či vysunuty pro docílení požadované trajektorie. Soustava těchto vektorů nám definuje trajektorii diskrétně. Co se ale děje mezi jednotlivými body, není definováno. Proto je nutné úseky mezi dvěma kloubovými souřadnicemi $\vec{q}_k$ a $\vec{q}_{k+1}$ prokládat vhodnými interpolačními funkcemi $Q_k(t)$. Interpolační funkce musí obsahovat krajní body úseku $\vec{q}_k$ a $\vec{q}_{k+1}$.

Pokud známe matematický popis všech interpolačních funkcí $Q_k(t)$ pro polohu jednotlivých pohonů, lze pomocí derivace spočítat rychlost a zrychlení v kterémkoli bodě trajektorie. Je tedy potřeba, aby byla spojitá první a druhá derivace výsledné funkce popisující trajektorii, která je složená z interpolačních funkcí $Q_k(t)$. Protože jednotlivé funkce $Q_k(t)$ jsou většinou polynomiální, stačí řešit spojitost pouze v uzlových bodech $\vec{q}_k$. Musí tedy platit:

$$\begin{aligned}
\lim_{t\to T_k-}\left(Q_k(t)\right) &= \lim_{t\to T_k+}\left(Q_k(t)\right)\\
\lim_{t\to T_k-}\left(\dot{Q}_k(t)\right) &= \lim_{t\to T_k+}\left(\dot{Q}_k(t)\right)\\
\lim_{t\to T_k-}\left(\ddot{Q}_k(t)\right) &= \lim_{t\to T_k+}\left(\ddot{Q}_k(t)\right)
\end{aligned} \tag{30}$$

Pokud máme všechny funkce $Q_k(t)$ definované, navzorkujeme trajektorii nějakou periodou $T_V$. Udává se, že by frekvence vzorkování měla být menší, než rezonanční frekvence vlastních kmitů kloubových mechanizmů. Zpravidla vyhovuje $T_V = 0,01s$.

### 4.3.1 Interpolace lineární funkcí

Než budeme interpolaci provádět, musíme mít spočtenou inverzní úlohu pro všechny uzlové body v systému základny. Interpolujeme v úrovni zobecněných souřadnic. Je nutné si ale uvědomit, že výsledný pohyb chapadla v základně nebude přímkový. Pokud interpolujeme uzlové body jednotlivých kloubových souřadnic přímkou, budeme mít spojitý průběh polohy. Rychlost bude mezi uzlovými body konstantní, ale bude se na zlomu měnit skokově. Zrychlení bude mít v uzlových bodech podobu nekonečně velkých impulsů. Fyzikálně je tedy tento pohyb nerealizovatelný.

Lineární interpolaci pro $k$-tý interval časového průběhu kloubové souřadnice $q$ počítáme dle vztahu:

$$q_{(t)} = \frac{q_{k+1} - q_k}{t_{k+1} - t_k}\left(t - t_k\right) + q_k \text{, kde } t \in \left\langle t_k ; t_{k+1} \right\rangle \tag{31}$$

Tato interpolace se používá u málo výkonných výpočetních systémů. Robot je nucen v uzlových bodech zpomalovat. Problém nespojitosti rychlosti a zrychlení v uzlových bodech řeší například tak, že se v blízkosti uzlového bodu proloží trajektorie křivkou alespoň druhého řádu, například kružnicí. Robot tedy neprojede zadaným uzlovým bodem, ale v jeho dostatečně blízkém okolí, které je možné definovat.

### 4.3.2 Interpolace kubickým splinem (spline-funkcí)

Interpolace kubickou funkcí, tedy polynomickou funkcí třetího řádu, se v praxi používá nejvíce. Funkce třetího řádu nám poskytuje nejen spojitou polohu, ale i rychlost a zrychlení. Pohyb po této křivce je tedy fyzikálně realizovatelný, neboť je splněna podmínka vyjádřená vztahem (30). V podstatě by šly použít i křivky vyšších řádů, ale s růstem řádu se komplikuje výpočet interpolačních funkcí a dochází ke ztrátě přesnosti kvůli vyšším mocninám polynomu.

Opět hledáme funkce pro každý interval $\left\langle q_k ; q_{k+1} \right\rangle$ zobecněné souřadnice $q$. Takže pokud máme $N$ bodů zobecněné souřadnice $q$ v čase, je třeba nalézt $N-1$ kubických funkcí. Pro $k$-tý interval to bude funkce:

$$q_{(t)} = a_k\left(t - t_k\right)^3 + b_k\left(t - t_k\right)^2 + c_k\left(t - t_k\right) + d_k \text{, kde } t \in \left\langle t_k ; t_{k+1} \right\rangle, k = 1,..,N-1 \tag{32}$$

Zavedeme si délku časového intervalu $h_k = t_{k+1} - t_k$ a moment $M_k = \ddot{q}_{(t_k)}$. Pak z návaznosti nultých, prvních a druhých derivací jednotlivých funkcí získáme vztahy pro koeficienty polynomu:

$$\begin{aligned} a_k &= \frac{M_{k+1} - M_k}{6h_k} \\ b_k &= 0,5 \cdot M_k \\ c_k &= \frac{q_{k+1} - q_k}{h_k} - \frac{M_{k+1} + 2M_k}{6} h_k \\ d_k &= q_k \end{aligned} \tag{33}$$

Koeficienty $a_k, b_k, c_k, d_k$ jsou tedy jednoznačně určeny pomocí zadaných bodů $q_1..q_N$, momentů $M_1..M_N$ a délek intervalů $h_1..h_{N-1}$. Zadané body a délky intervalů známe. Je tedy nutné spočítat momenty $M_1..M_N$. Dosadíme koeficienty $a_k, b_k, c_k, d_k$ do vztahu, který zaručuje spojitost první derivace celkové interpolované funkce. Získáme potom soustavu:

$$h_{k-1}M_{k-1} + 2\left(h_{k-1} + h_k\right)M_k + h_k M_{k+1} = 6\frac{q_{k+1} - q_k}{h_k} - 6\frac{q_k - q_{k-1}}{h_{k-1}} \tag{34}$$

Takto získáme $N-2$ rovnic o $N$ neznámých, kterými jsou momenty $M_1..M_N$. Je tedy potřeba zavést tzv. okrajové podmínky. Nejčastěji se volí $M_1 = 0$ a $M_N = 0$ (přirozený spline). Momenty nám představují velikosti zrychlení v zadaných bodech. Nulové okrajové podmínky nám vyhovují, protože většinou požadujeme nulová zrychlení na počátku a na konci trajektorie. Obecně ale můžeme volit nenulové okrajové podmínky nebo můžeme zadat hodnoty prvních derivací (rychlostí) na počátku a konci trajektorie. Přibyly by nám tak další dvě rovnice. Počet rovnic a neznámých by pak souhlasil.

Po zvolení okrajových podmínek (zvolení okrajových zrychleních) máme $N-2$ rovnic o $N-2$ neznámých. Výsledná matice soustavy (35) je třídiagonální a je ostře diagonálně dominantní – má právě jedno řešení. Získáme tedy lineární soustavu ve tvaru:

$$\begin{pmatrix} a_1 & c_1 & 0 & 0 & 0 \\ b_1 & a_2 & c_2 & 0 & 0 \\ 0 & b_2 & a_3 & \ddots & 0 \\ 0 & 0 & \ddots & \ddots & c_{n-1} \\ 0 & 0 & 0 & b_{n-1} & a_n \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \\ f_3 \\ \vdots \\ f_n \end{pmatrix} \tag{35}$$

Třídiagonální soustavy lze jednoduše řešit ve dvou krocích [4]. A to v kroku dopředném a zpětném. V dopředném kroku spočítáme koeficienty $\alpha_1..\alpha_{n-1}$ a $\beta_1..\beta_n$ podle vztahů:

$$\alpha_1 = -\frac{c_1}{a_1}, \beta_1 = \frac{f_1}{a_1} \rightarrow \alpha_i = -\frac{c_i}{b_{i-1}\alpha_{i-1} + a_i}, \beta_i = \frac{f_i - b_{i-1}\beta_{i-1}}{b_{i-1}\alpha_{i-1} + a_i} \tag{36}$$

Ve zpětném kroku pak rovnou získáváme jednotlivá řešení dle vztahů.

$$x_n = \beta_n \rightarrow x_i = \alpha_i x_{i+1} + \beta_i \tag{37}$$

Řešení touto metodou je velice rychlé i pro velký počet neznámých. Matice soustavy ale musí mít minimální rozměr 2x2.

# Praktická část

## 5 Struktura programu

Program pro inverzní kinematickou úlohu a plánování trajektorie robotů je napsán v programovacím jazyce *Delphi*. Skládá se z několika unit a formulářů. Důležitou unitou, která obsluhuje hlavní formulář programu, je *Unit1*. Následuje *Unit2*, která představuje formulář pro definici vlastní kinematické struktury robota. V *Unit3* se nachází formulář pro nastavení parametrů jednotlivých výpočtů v programu (chyba výpočtu inverzní úlohy, druh interpolace, apod.). V programu jsou ještě další tři formuláře: Pro zobrazení analytických řešení (*Zobr_A_R*), pro zobrazení informací o vybraném robotu (*InfoRobot*) a pro přidání bodu do trajektorie (*AddPoint*).

Vše, co se týče metody pro výpočet inverzní úlohy robota, je v unitě *Inv_ul*. V unitě *Inv_ul* je také naprogramovaná přímá úloha robota. Výpočty pro plánování trajektorie (řízení výpočtu inverzní úlohy trajektorie, interpolace, apod.) jsou naprogramovány v unitě *Trajektorie*. Grafické a tabelární zobrazování výsledných výpočtů je implementováno v unitě *Grafy*.

Následují důležité pomocné unity. Pro práci s Eulerovými úhly (převod z matice rotace na Eulerovy úhly a zpět) slouží unita *Eu_uhly*. Neméně důležitá je unita *Mytype*, která má v sobě nadefinované všechny používané datové typy v programu a která má v sobě naprogramované často používané funkce v programu. Jsou to funkce pro násobení matic a práci s tabulkami. Unita *ZpracujVyr* dokáže vyhodnocovat matematické výrazy s více proměnnými zapsané v textovém řetězci. Využívá se pro rozpoznání fixních vztahů pro Eulerovy úhly (6.7) u uživatelského robota se 4 nebo 5 stupni volnosti.

Robot je v programu představován instancí objektu *TRobot*, který je definován v unitě *Robot*. Objekt má v sobě uloženy důležité kinematické parametry (stupeň volnosti, DH parametry, apod.) a během výpočtů je poskytuje. Navíc dokáže při zadání konkrétního vektoru kloubových souřadnic vrátit jednotlivé transformační matice mezi kinematickými členy. Po spuštění programu se vytvoří instance všech pevně definovaných robotů (objektů). Kinematické parametry pro roboty jsou čerpány z unity *Konst*.

# 6 Robotické struktury definované v programu

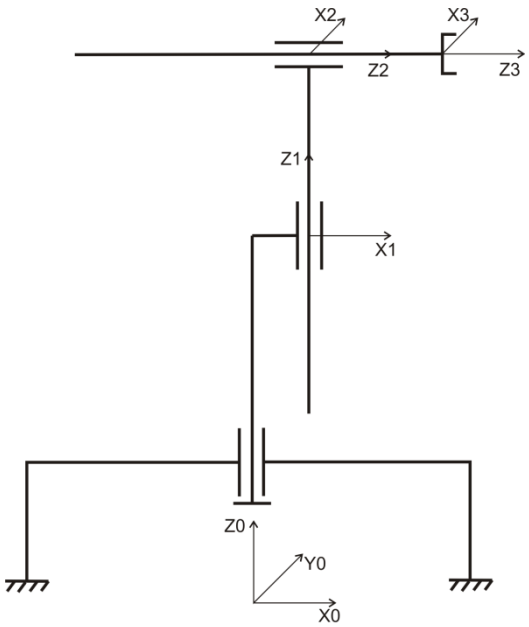
V programu je nadefinováno celkem šest robotů. První tři mají jednoduchou kinematickou strukturu. Umožňují pouze polohování chapadla. Nelze pomocí nich nastavovat orientaci. Do programu byly zavedeny hlavně kvůli testování správnosti výpočtů jednotlivých úloh, neboť si je lze v prostoru jednoduše představit. Nemají nastaveny fyzikálně realizovatelné rozsahy pohonů, aby bylo možné například názorně vidět, že použitá metoda pro inverzní úlohu nalezne všechna teoreticky možná řešení.

Další trojice robotů má složitější kinematickou strukturu. Roboty už umožňují nastavovat orientaci chapadla. Mají tedy více než 3 stupně volnosti. V programu je možno si zvolit mezi roboty se čtyřmi (RRT-R SCARA), pěti (RRR-RR) a šesti (RTR-TRR) stupni volnosti.

Všechny robotické struktury jsou pevně dané a nelze je v programu měnit. Je ale možné si nadefinovat jednu vlastní kinematickou strukturu od tří do šesti stupňů volnosti. Tato možnost je popsána v kapitole 6.7.

## 6.1 Robot RTT

Robot RTT pracuje v cylindrickém (nebo také válcovém) souřadném systému. Má tři stupně volnosti. První umožňuje rotaci kolem svislé osy, druhý umožňuje posuv ve vertikálním směru a třetí v horizontálním směru. Pracovní prostor robota je válec nebo jeho část – dle pracovních rozsahů pohonů.




*Obr. 7: Kinematické schéma robota RTT.*

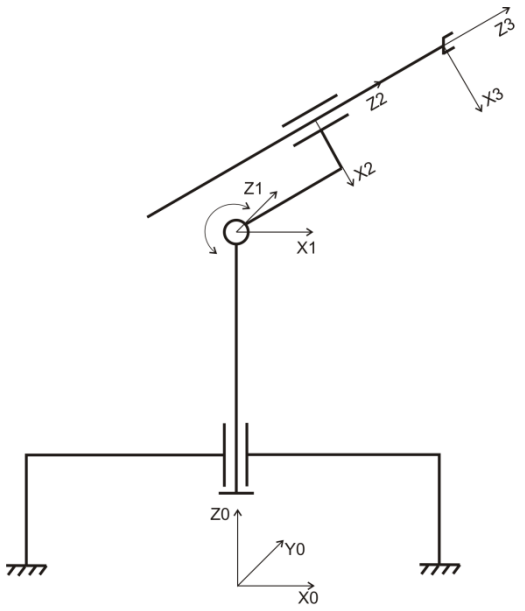
| RTT | DH parametry | | | | Rozsah | |
|---|---|---|---|---|---|---|
| | φ (°) | s (m) | α (°) | a (m) | Od | Do |
| $q_1$ | **Q** | 0 | 0 | 0 | 0° | 359° |
| $q_2$ | 90 | **Q** | 90 | 0 | -1m | 1m |
| $q_3$ | 0 | **Q** | 0 | 0 | -1m | 1m |

*Tab. 2: DH parametry a rozsahy pohonů robota RTT.*

Je potřeba upozornit na to, že robot RTT (ale i RRT) nadefinovaný v programu nemá konstrukčně možné DH parametry a rozsahy pohonů. Například, pokud budou všechny pohony v nulové pozici, bude i chapadlo v nulové pozici.

## 6.2 Robot RRT

Robot RRT pracuje ve sférických (nebo také v kulových) souřadnicích. Má opět tři stupně volnosti. První natáčí systém kolem svislé osy, druhý umožňuje rotaci kolem vodorovné osy a třetí vysouvá koncový člen ve směru přístupového vektoru chapadla (obr. 2). Pracovní prostor robota představuje část koule.



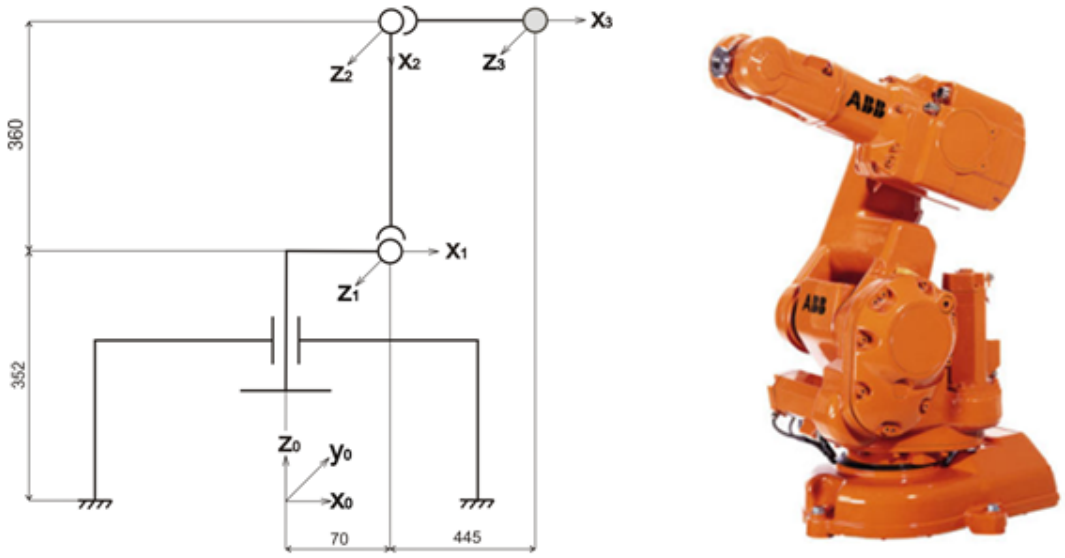

*Obr. 8: Kinematické schéma robota RRT.*

| RRT | DH parametry | | | | Rozsah | |
|---|---|---|---|---|---|---|
| | φ (°) | s (m) | α (°) | a (m) | Od | Do |
| $q_1$ | **Q** | 1 | -90 | 0 | 0° | 359° |
| $q_2$ | **Q** | 0 | 90 | 0 | 0° | 359° |
| $q_3$ | 0 | **Q** | 0 | 0 | -1m | 1m |

*Tab. 3: DH parametry a rozsahy pohonů robota RRT.*

## 6.3 Robot RRR

Robot RRR se třemi stupni volnosti pracuje v angulárních souřadnicích. Robot RRR definovaný v programu tvoří polohovací systém robota ABB IRB 140 (6 st. vol.). Umožňuje tedy pouze nastavení polohy, ne orientace. První pohon robota natáčí systém okolo svislé osy, zbylé dva pak ve vodorovné ose. Ačkoli jsou poslední dvě osy rotace rovnoběžné, pokryjí dvojdimenzionální prostor (rovnoběžné osy posuvných členů nerozšiřují stupeň volnosti robota).




*Obr. 9: Kinematické schéma polohovacího systému robota ABB IRB 140 (RRR).*

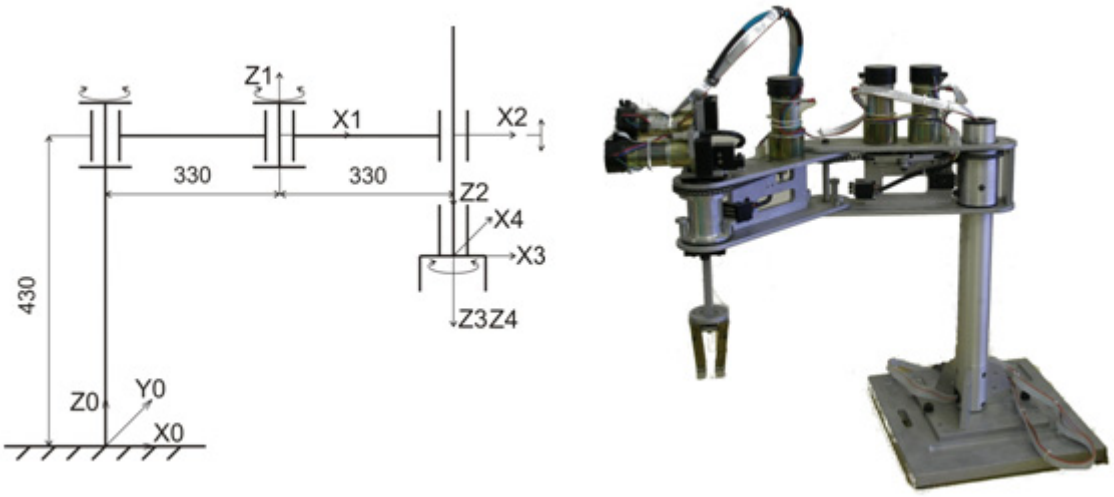
| RRR | DH parametry | | | | Rozsah | |
|---|---|---|---|---|---|---|
| | φ (°) | s (m) | α (°) | a (m) | Od | Do |
| $q_1$ | **Q** | 0,352 | 90 | 0,07 | 0° | 359° |
| $q_2$ | **Q** | 0 | 0 | -0,36 | 0° | 359° |
| $q_3$ | **Q** | 0 | 0 | 0,445 | 0° | 359° |

*Tab. 4: DH parametry a rozsahy pohonů robota RRR (ABB IRB 140).*

## 6.4 Robot RHINO SCARA

Robot RHINO SCARA má 4 stupně volnosti. Polohovací systém robota má strukturu RRT. První dva pohony se otáčí kolem svislé osy, pokryjí tedy vodorovnou rovinu. Třetí pohon polohuje vysouváním ve svislém směru. Jednoduchou orientaci chapadla pak umožňuje nastavit poslední rotační člen, který se otáčí také kolem svislé osy. Robot má tedy kinematickou strukturu RRT-R.

Výhodou SCARA struktury je to, že polohovací rotační členy robota nemusí překonávat gravitační sílu. Umožňují tedy rychlé přemisťování předmětů v rovině $xy$. Pracovním prostorem robota je část válce (jen co se týče polohování).




*Obr. 10: Kinematické schéma robota RHINO SCARA.*

| RHINO SCARA | DH parametry | | | | Rozsah | |
|---|---|---|---|---|---|---|
| | φ (°) | s (m) | α (°) | a (m) | Od | Do |
| $q_1$ | **Q** | 0,43 | 0 | 0,33 | 0° | 359° |
| $q_2$ | **Q** | 0 | 180 | 0,33 | -170° | 170° |
| $q_3$ | 0 | **Q** | 0 | 0 | -0,25m | 0,25m |
| $q_4$ | **Q** | 0,15 | 0 | 0 | 0° | 359° |

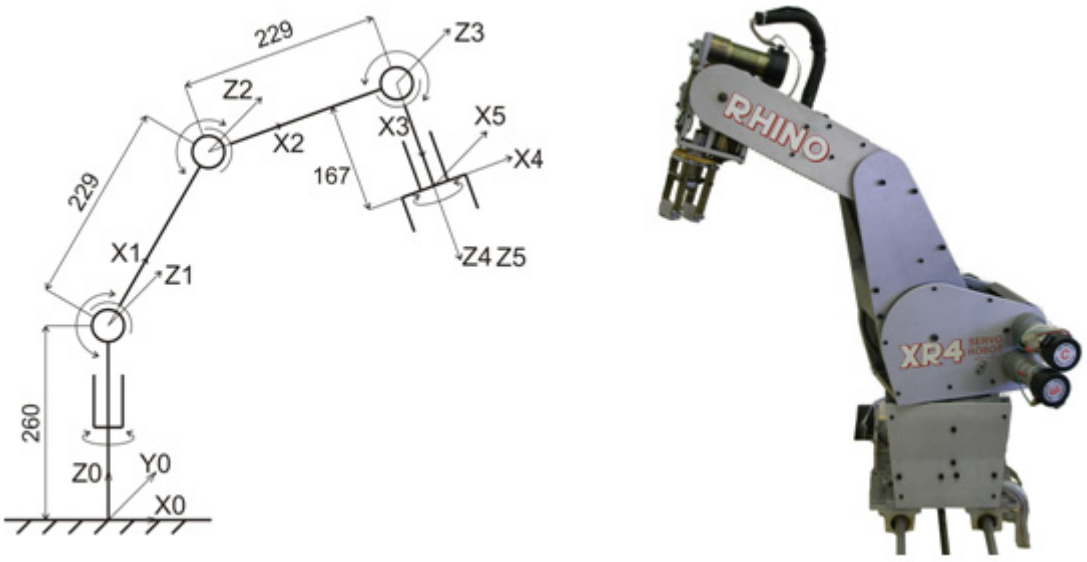
*Tab. 4: DH parametry a rozsahy pohonů robota RHINO SCARA.*

Přístupový vektor chapadla (obr. 2) SCARA struktury směřuje vždy dolů kolmo k rovině $xy$. To odpovídá konstantnímu Eulerovu úhlu $\beta = 180°$. Osy otáčení Eulerových úhlů $\alpha$ a $\gamma$ jsou pak shodné a existuje tak nekonečně mnoho kombinací úhlů $\alpha$ a $\gamma$ popisujících aktuální natočení chapadla. V programu je tak volen $\alpha = 0°$ a úhel $\gamma$ je nastavitelný. Z toho vyplývají fixní vztahy pro úhly $\alpha$ a $\beta$:

$$\alpha = 0°, \beta = 180°$$

## 6.5 Robot RHINO XR4

Robot RHINO XR4 má 5 stupňů volnosti. Skládá se z pěti rotačních dvojic, má tedy strukturu RRR-RR. První rotační člen se otáčí kolem svislé osy. Následující tři pohony se otáčí kolem rovnoběžných vodorovných os. Umožňují tedy polohování chapadla s více variantami natočení úhlu $\beta$. Poslední člen se otáčí kolem přístupového vektoru chapadla a nastavuje tedy pouze orientaci. Jeden Eulerův úhel ale nejde libovolně nastavovat, a musíme tedy znát jeho fixní vztah. Je jím úhel $\gamma$, který se vždy rovná natočení kloubu $q_1$. Máme tedy rovnost: $\gamma = q_1$.



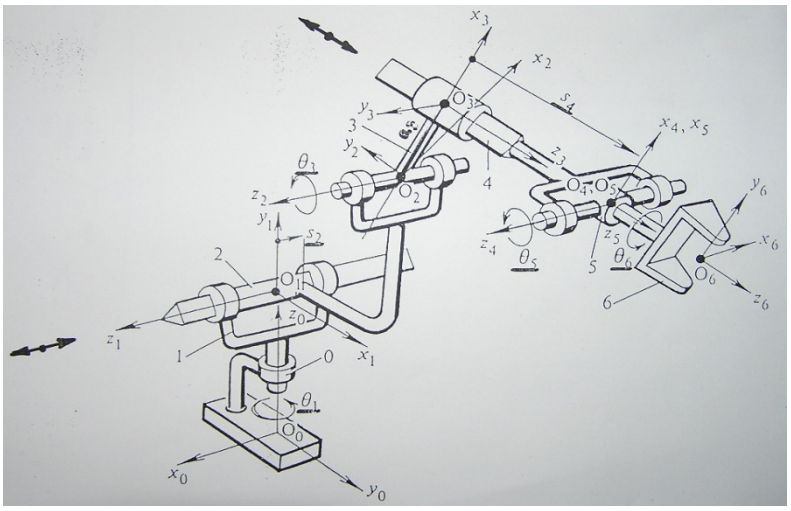

*Obr. 11: Kinematické schéma robota RHINO XR4.*

| RHINO XR4 | DH parametry | | | | Rozsah | |
|---|---|---|---|---|---|---|
| | φ (°) | s (m) | α (°) | a (m) | Od | Do |
| $q_1$ | **Q** | 0,26 | -90 | 0 | 5° | 355° |
| $q_2$ | **Q** | 0 | 0 | 0,229 | 0° | 135° |
| $q_3$ | **Q** | 0 | 0 | 0,229 | -60° | 135° |
| $q_4$ | **Q** | 0 | -90 | 0 | -80° | 260° |
| $q_5$ | **Q** | 0,167 | 0 | 0 | 0° | 359° |

*Tab. 5: DH parametry a rozsahy pohonů robota RHINO XR4.*

## 6.6 Robot se 6 stupni volnosti

Robot má strukturu RTR-TRR. Umožňuje ve svém pracovním prostoru nastavovat libovolnou polohu a orientaci chapadla. Z analytického řešení, které je naimplementované v programu, vyplývá, že každou polohu a orientaci dokáže nastavit až čtyřmi způsoby. Samozřejmě vyjma některých případů, jako jsou například singulární body. Robot byl převzat z přednášek doc. Mgr. Ing. Václava Zády, CSc. [2].



*Obr. 12: Kinematické schéma robota se 6 st. vol. (RTR-TRR).*

| RHINO XR4 | DH parametry | | | | Rozsah | |
|---|---|---|---|---|---|---|
| | φ (°) | s (m) | α (°) | a (m) | Od | Do |
| $q_1$ | **Q** | 0,5 | 90 | 0 | 0° | 359° |
| $q_2$ | 45 | **Q** | 0 | 0,2 | -1m | -1m |
| $q_3$ | **Q** | 0 | 90 | 0,2 | 0° | 359° |
| $q_4$ | 0 | **Q** | -90 | 0 | -1m | 1m |
| $q_1$ | **Q** | 0 | 90 | 0 | 0° | 359° |
| $q_6$ | **Q** | 0,5 | 0 | 0 | 0° | 359° |

Tab. 6: *DH parametry a rozsahy pohonů robota se 6 st. vol. (RTR-TRR).*

## 6.7 Uživatelsky definovatelný robot

V programu je umožněno definování vlastní kinematické struktury robota. Robot může mít minimálně tři stupně volnosti a maximálně šest stupňů. Pokud má robot pouze tři stupně volnosti, předpokládá se, že budeme moci nastavovat pouze polohu. Nelze tedy nastavovat například dvě kartézské souřadnice a jeden Eulerův úhel apod. S růstem počtu stupňů volnosti se přidávají nastavitelné Eulerovy úhly.

Prvním spíše formálním parametrem je název robota. Pokud si pak robota vybereme, v dolním informačním panelu programu se jeho název zobrazí.

Následují důležitější parametry, především počet stupňů volnosti robota. Je to klíčový kinematický parametr. Definuje počet kinematických dvojic (rotačních nebo translačních) s jedním stupněm volnosti. Podle počtu stupňů volnosti se mění rozměr tabulky pro zadávání DH parametrů a rozsahů pohonů.

Dále je nutné určit druh jednotlivých kinematických dvojic – tedy kinematickou strukturu robota. Zadává se pomocí písmen „R“ pro rotační člen a „T“ pro translační člen. Dle struktury se definuje vektor zobecněných (kloubových) souřadnic. Dokud nevyplníme počet stupňů volnosti a kinematickou strukturu, nelze zadávat ostatní potřebné parametry.

Po zadání počtu stupňů volnosti a kinematické struktury je potřeba zadat DH parametry robota. Zadávají se pouze konstantní DH parametry. Proměnné DH parametry tvořící kloubové souřadnice se nevyplňují a ignorují se. Program políčka kloubových souřadnic zabarví červeně.

Tabulka dále vyžaduje zadat rozsahy pohonů robota – tedy rozsahy příslušných kloubových souřadnic. Pokud zadáváme rozsah pro translační člen, údaj je v metrech,

pokud pro translační, údaj je ve stupních. Rozsahem pohonů se určí pracovní prostor robota.

Pokud zadáváme robota se čtyřmi nebo s pěti stupni volnosti, nastává problém, který z Eulerových úhlů dokáže robot nastavovat. K tomu slouží položka „Fixní vztahy pro nenastavitelné Eulerovy úhly“. Robot se 4 stupni volnosti dokáže nastavovat pouze jeden Eulerův úhel. Zbylé dva jsou konstantní nebo jsou funkcemi vektoru kloubových souřadnic. U robota s 5 stupni volnosti lze nastavovat dva Eulerovy úhly a zbylý jeden je dán strukturou robota. Protože program využívá gradientní metodu (kapitola 3.1.4) pro výpočet inverzní úlohy, kde se minimalizuje funkcionál $I^2_{(\vec{q})} = \left| A^0_{n\,D} - A^0_{n(\vec{q})} \right|^2$, je potřeba dosadit do submatice rotace dané transformační matice $A^0_{n\,D}$ jednotlivé Eulerovy úhly $\alpha, \beta, \gamma$ podle vztahu (7). Protože u robotů se 4 nebo 5 stupni volnosti nemáme všechny tyto úhly zadané, musíme je získat právě z fixních vztahů pro nenastavitelné Eulerovy úhly. U robota se třemi stupni volnosti není potřeba tyto vztahy znát, protože nastavujeme jenom polohu. Poloha lze z transformační matice odseparovat, takže submatici rotace lze ve funkcionálu (16) ignorovat. Jednotlivé Eulerovy úhly ale ze submatice rotace odseparovat nelze. U robotů se 6 stupni volnosti jsou nastavitelné všechny Eulerovy úhly, není tedy potřeba fixní vztahy zadávat.

Fixní vztahy lze získat ze znalosti matice $A^0_{n(\vec{q})}$ v obecném tvaru. Dáme do rovnosti submatici rotace matice $A^0_{n(\vec{q})}$ s maticí rotace vyjádřenou pomocí Eulerových úhlů (7). Z této rovnice odvodíme vztahy pro nenastavitelné úhly. Pokud fixní vztahy pro daného robota neznáme, program umožňuje řešit alespoň jeho polohování. Pak je sice dále od uživatele vyžadováno při výpočtech zadávat požadovanou orientaci, ale musí si být vědom toho, že výsledky budou odpovídat pouze poloze, ne zadané orientaci. Závěrem je důležité podotknout, že případné úhly zadávané do fixních vztahů musejí být v radiánech.

# 7 Inverzní a přímá úloha pro bod (počáteční bod)

V záložce „Počáteční bod“ hlavního formuláře programu lze řešit přímou a inverzní kinematickou úlohu pro jednotlivé body. Přímou úlohou (kapitola 2) se příliš zabývat nebudeme, je v programu naimplementována spíše pro kontrolu správnosti výpočtu inverzní úlohy. Je jen dobré podotknout, že přímá úloha v programu vypočte pro konkrétní kloubové souřadnice kompletní polohu a orientaci chapadla v systému základny i u robotů s méně než šesti stupni volnosti.

Zajímavější je ale inverzní kinematická úloha. Program počítá inverzní úlohu numericky pro všechny zadané roboty a analyticky pro vybrané robotické struktury. Analytické řešení je duální kontrolou konvergence numerické metody ke správnému řešení inverzní kinematické úlohy. Jestli se bude analytické řešení počítat, nebo nebude, lze zvolit v nastavení programu.

## 7.1 Použitá numerická metoda pro výpočet inverzní úlohy

Program využívá pro řešení inverzní úlohy gradientní numerickou metodu popsanou v kapitole (3.1.4). Protože bylo snahou naprogramovat univerzální metodu použitelnou pokud možno pro libovolnou kinematickou strukturu, tak program nepočítá gradient funkcionálu $I^2_{(\vec{q})}$ (vztah 16) analyticky. Využívá ale numerické derivování pro výpočet gradientu podle vztahu:

$$\frac{\partial I^2(\vec{q})}{\partial q_i} = \frac{I^2(q_1,...,q_i+h_i,...,q_n) - I^2(q_1,...,q_i-h_i,...,q_n)}{2h_i} \tag{38}$$

V praxi se stává, že numerické derivování je nestabilní a je tedy dobré se mu pokud možno vyhýbat. Bývá to hlavně u skokových změn derivované veličiny. Nicméně v programu se numerické derivování osvědčilo. Volba velikosti kroku $h_i = 0,0001$ byla provedena experimentálně.

**Algoritmus postupného zdvojnásobování nebo půlení délky kroku**

Dalším problémem bylo optimalizovat velikost kroku gradientní metody $\lambda_k$ (17). Pro výpočet optimálního kroku je potřeba určit Hessovu matici (20), tedy matici druhých parciálních derivací. Bylo by zřejmě možné použít opět numerické derivování, nicméně v programu je použita jiná optimalizace velikosti kroku $\lambda_k$. Spočívá v tom, že se v kroku $k$ zvolí dostatečně malý krok $\lambda_k$. Jím se pokusně provede jeden krok gradientní metody podle vztahu (17). Získáme tedy z původních kloubových souřadnic $\vec{q}_k$ souřadnici $\vec{q}_{k+1}$. Oba body dosadíme do funkcionálu $I^2_{(\vec{q})}$ a porovnáním zjistíme, zda po provedení kroku s $\lambda_k$ došlo ke zvýšení či snížení hodnoty funkcionálu.

a) **Pokud došlo ke snížení** funkcionálu, uložíme si funkční hodnotu $I^2(\vec{q}_{k+1})$ a vynásobíme krok $\lambda_k$ dvěma. Opět provedeme krok gradientní metody, porovnáme novou funkční hodnotu s hodnotou $I^2(\vec{q}_{k+1})$ a případně po dalším snížení funkce si uložíme novou funkční hodnotu a opět násobíme krok $\lambda_k$ dvěma. Takto krok zdvojnásobujeme, dokud nedojde ke zvýšení funkce oproti předešlé hodnotě. Až dojde ke zvýšení funkce, víme, že optimální krok leží mezi aktuální a poloviční hodnotou kroku $\lambda_k$.

b) **Pokud došlo ke zvýšení** funkcionálu, dělí se krok $\lambda_k$ dvěma, dokud nedojde ke snížení funkcionálu oproti původní hodnotě. Až dojde ke snížení, dostaneme se teprve na rozhraní mezi zvýšením a snížením funkcionálu. Nejedná se o optimální krok. Je nutné dále krok dělit dvěma, dokud funkcionál více a více nesnižujeme. Ukládáme si tedy postupně nové snížené hodnoty funkcionálu. Až zase dojde k zvýšení funkcionálu, víme, že optimální krok leží mezi aktuální a dvojnásobnou hodnotou $\lambda_k$.

Máme tedy intervaly, ve kterých zaručeně leží optimální krok $\lambda_k$. Teď intervaly postupně sužujeme, dokud nebudou dostatečně úzké. Pak budeme považovat prostřední hodnotu intervalu za „optimální krok“.

## 7.2 Přesnost řešení inverzní úlohy

Gradientní numerická metoda se svým konečným počtem opakování přiblíží k řešení inverzní úlohy s nějakou přesností. Iterace metody se ukončí tehdy, pokud bude hodnota funkcionálu $I^2(\vec{q})$ dostatečně malá, v ideálním případě nulová. Hodnota, při které se ukončí průběh gradientní metody, souvisí s přesností nalezeného řešení. Zavedeme si tedy nějakou chybu minimalizace $e$. Protože minimalizujeme druhou mocninu funkcionálu $I$, budeme gradientní metodu ukončovat, pokud bude hodnota $I^2(\vec{q})$ menší než druhá mocnina chyby $e$. Kloubové souřadnice $\vec{q}$ pro nás tedy budou řešením, pokud bude platit:

$$I_{(\vec{q})}^2 < e^2 \tag{39}$$

V programu lze chybu minimalizace $e$ zadat v nastavení. Teoreticky lze chybu $e$ nekonečně snižovat, ale v programu ne. Pokud zadáme $e$ příliš malé, může se stát, že metoda nenalezne žádná řešení. Je pak nutné chybu $e$ zvýšit. U každého robota je minimální možné $e$ různé. Nelze tedy přesně stanovit univerzální minimální chybu minimalizace. Referenčně je nastavena na hodnotu $e = 10^{-4}$.

## 7.3 Hledání všech řešení inverzní úlohy

Je žádoucí nalézt všechny možné způsoby nastavení pohonů, které docílí požadované polohy a orientace. Ne každé řešení musí vyhovovat dané situaci a je dobré, pokud si uživatel může zvolit, které mu nejvíce vyhovuje.

Gradientní metoda nalezne pro zvolený počáteční odhad jedno řešení inverzní úlohy. Může se ale také stát, že pokud zvolíme nevhodný počáteční odhad, nenalezne žádné řešení. Abychom tedy našli všechna možná řešení, musíme pro gradientní metodu zvolit více počátečních odhadů z různých míst pracovního prostoru robota. V podstatě prohledáme celý jeho pracovní prostor. Čím jemněji ho prohledáváme, tím větší je pravděpodobnost, že nalezneme všechna řešení.

V programu je prohledávání prostoru provedeno tak, že si z pracovního intervalu každé kloubové souřadnice rovnoměrně vybereme $K$ bodů. Získáme tak $K^N$ počátečních odhadů, kde $N$ je počet stupňů volnosti robota. Každý z těchto odhadů dosadíme do gradientní metody. Získáme tak několik řešení inverzní kinematické úlohy. Řešení pak musíme protřídit a vybrat z nich pouze ta, která se od sebe liší. Tato metoda samozřejmě nezaručuje nalezení úplně všech řešení, nicméně, čím vyšší $K$ zvolíme, tím větší pravděpodobnost úspěchu máme. Je ale nutné si uvědomit, že s rostoucím $K$ velice rychle roste počet počátečních odhadů a s rostoucím počtem stupňů volnosti jejich počet roste exponenciálně. Prohledávání pak bude velice pomalé. V programu je referenčně nastaveno $K = 2$ a lze ho měnit v intervalu od 1…5. Není však doporučeno používat $K > 3$ zejména u robotů s větším počtem stupňů volnosti, neboť doba výpočtu je velice dlouhá.

I s klesající hodnotou chyby minimalizace $e$ (kapitola 7.2) samozřejmě roste doba výpočtu inverzní úlohy. Proto je dobré zvolit na počátku vyšší hodnotu $e$, prohledat pracovní prostor robota a nalézt tak jednotlivá řešení inverzní úlohy. Pomocí tlačítka <zpřesnit> pak „dokonvergujeme“ blíže k hledanému řešení. Tlačítko <zpřesnit> provede několik dalších kroků gradientní metody pouze ale pro nalezená řešení.

# 8 Plánování trajektorie v programu

## 8.1 Editování trajektorie

Trajektorie v programu je definována soustavou polohových bodů (kapitola 4.2). Ke každému bodu trajektorie v systému základny je přiřazen časový okamžik. Editování trajektorie v programu umožňují následující tlačítka.

Tlačítko <*Přidat bod*> vyvolá formulář, ve kterém je uživatel vyzván k zadání požadovaného bodu. Po potvrzení se bod vloží do trajektorie a automaticky se zařadí chronologicky podle času. Zadá-li uživatel do formuláře bod, jehož časový okamžik už trajektorie obsahuje, starý bod se přepíše novým. Při každém vložení se vymaže trajektorie v kloubových souřadnicích.

Tlačítko <*Ubrat bod*> odebere z trajektorie všechny označené body. Opět se případně smaže trajektorie v kloubových souřadnicích.

Další možností, jak vložit body do trajektorie, je načtení ze souboru. Soubor je v textovém formátu **.txt*. Body jsou zadávány ve sloupcích a musejí být odděleny tabulátorem. První řádek souboru je rezervován pro hlavičku popisující jednotlivé sloupce a program jej ignoruje. Následující řádky už nesou informaci o zadávané trajektorii. Program po načtení vyhodnotí správnost načtení a zkontroluje, zda jsou časové okamžiky bodů v souboru seřazeny vzestupně. Pokud soubor obsahuje více sloupců než je potřeba (například načteme-li trajektorii pro robota se 6 stupni volnosti u robota se 4 stupni volnosti), program nepotřebné sloupce ignoruje.

Vytvořenou trajektorii je také možné uložit do textového souboru. Po stisku příslušného tlačítka je nutné zadat název souboru a cíl, kam ho chceme uložit.

Zadanou trajektorii můžeme dodatečně zjemňovat. Zjemňování je prováděno tak, že se mezi každou označenou dvojici bodů vloží několik dalších bodů zvolenou aproximací. Aproximace může být lineární nebo kubická (kapitola 4.3). Počet vkládaných bodů lze volit v nastavení. Zjemněním sice zadáme trajektorii přesněji, ale zase prodlužujeme výpočet inverzní úlohy trajektorie. Také je nutné si dávat pozor na to, že opakovaným zjemňováním vybraného časového úseku počet bodů trajektorie exponenciálně roste.

## 8.2 Inverzní kinematická úloha pro trajektorii

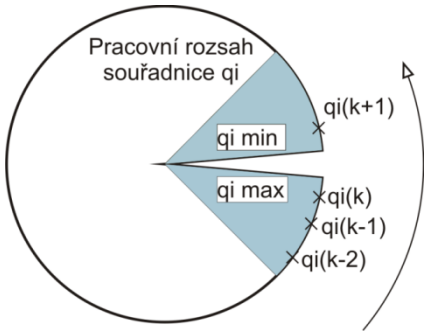
Máme-li zadanou trajektorii posloupností polohových bodů, můžeme ji přepočítat na kloubové souřadnice. Nejdříve je ale nutné definovat počáteční bod $Q_0$ v kloubových souřadnicích, tedy výchozí pozici robota. Počáteční bod $Q_0$ je počítán z prvního bodu trajektorie. Po stisku tlačítka <*Inv. úloha pro poč. bod*> se spustí výpočet inverzní úlohy (kapitola 7.1) a jsou hledána všechna možná řešení (kapitola 7.3). Uživatel si pak vybere, jaké výchozí natočení robota chce použít a vloží ho do trajektorie. Je vhodné volit takový bod, který je pokud možno co nejvíce uprostřed pracovního rozsahu jednotlivých pohonů robota. Sníží se tak riziko, že se robot při pohybu po trajektorii dostane mimo pracovní oblast.

Po zvolení výchozí polohy robota $Q_0$ lze přepočíst zadanou trajektorii do kloubových souřadnic pomocí tlačítka <Přepočet>. Pro výpočet inverzní úlohy se opět používá gradientní metoda (kapitoly 3.1.4 a 7.1). Už se zde ale nehledají všechna řešení, protože vyplývají jednoznačně z výchozí polohy $Q_0$.

U výpočtu inverzní úlohy trajektorie je kladen důraz na rychlost výpočtu, aby bylo možné řídit robota pokud možno v reálném čase. Rychlost výpočtu gradientní metody je tím menší, čím je počáteční bod iterace blíže k hledanému řešení. Toho se u přepočtu trajektorie velmi využívá. Předpokládá se totiž, že jednotlivé body trajektorie nejsou od sebe příliš vzdálené. Použijeme-li pro výpočet bodu $Q_k$ jako počáteční bod iterace předchozí vypočtený bod $Q_{k-1}$, „dokonverguje“ gradientní metoda k řešení velmi rychle v několika málo krocích. Výpočet inverzní úlohy trajektorie je tedy mnohem rychlejší než hledání všech řešení inverzní úlohy pro jeden bod.

Pokud některý ze zadaných bodů trajektorie je mimo pracovní oblast robota, naprogramovaná gradientní metoda pro něj nenalezne řešení. Program pak příslušný řádek v tabulce vymaže a nechá ho prázdný. Problém lze vyřešit tak, že zvolíme jiný výchozí bod. Pokud ale na určitou část trajektorie robot svým rozsahem i při změně výchozího bodu nedosáhne, nelze trajektorii pro daný typ robota spočítat.

Může se také stát, že program sice nalezne pro všechny body zadané trajektorie řešení inverzní úlohy, ale robot se přesto při pohybu dostane mimo pracovní oblast. Tuto situaci názorně popisuje následující obrázek č. 13.




*Obr. 13: Ukázka pohybu robota, při kterém jeden z pohonů narazí na svůj doraz.*

Souřadnice $qi_k$ i $qi_{k+1}$ leží v pracovním prostoru robota, ale při přesunu z bodu $qi_k$ do $qi_{k+1}$ by pohon robota narazil na koncový spínač, nebo by se otočil zpět na začátek svého rozsahu a pokračoval dále v pohybu. Ani jedna z uvedených možností však není žádoucí. Program tuto situaci dokáže za určitých okolností rozeznat a uživatele případně upozorní. Aby program situaci rozeznal, musí platit:

$$\begin{aligned} qi_k &< qi_{\min} + \frac{Zona1}{100}\cdot\left|qi_{\max} - qi_{\min}\right| \\ qi_{k+1} &> qi_{\max} - \frac{Zona1}{100}\cdot\left|qi_{\max} - qi_{\min}\right| \end{aligned} \quad \text{nebo} \quad \begin{aligned} qi_k &> qi_{\min} + \frac{Zona1}{100}\cdot\left|qi_{\max} - qi_{\min}\right| \\ qi_{k+1} &< qi_{\max} - \frac{Zona1}{100}\cdot\left|qi_{\max} - qi_{\min}\right| \end{aligned} \tag{40}$$

Znamená to pouze, že předchozí a následující bod souřadnice $qi$ musí ležet v určité zóně za začátkem a před koncem pracovního rozsahu nebo naopak. Parametr „Zóna1“ je uváděn v procentech z velikosti pracovního rozsahu souřadnice $qi$. Lze jej v programu měnit v rozsahu od 1% do 49%. Referenčně je nastaven na 10%. Na obr. 13 je zóna vyznačena modrou barvou. Tento problém nastává pouze u rotačních kinematických dvojic. Jak by mohl pohyb robota v této situaci vypadat, je ukázáno v následující kapitole na obr. 15.

Program také uživatele upozorní, pokud je vypočtená trajektorie v kloubových souřadnicích blízko konce pracovního prostoru robota. Všechny přepočtené body jsou uvnitř pracovního prostoru, nedochází k situaci, kterou popisuje obr. 13, ale část trajektorie se nebezpečně ke konci rozsahu blíží. Problém pak může následně nastat při interpolaci bodů v kloubových souřadnicích (kapitola 8.3). Program vyhodnotí souřadnici $qi$ jako blížící se ke konci svého rozsahu, pokud bude $qi$ náležet jednomu z následujících intervalů:

$$\begin{aligned} qi_k &\in \left\langle qi_{\min}; qi_{\min} + \frac{Zona2}{100}\cdot\left|qi_{\max} - qi_{\min}\right| \right\rangle \\ qi_k &\in \left\langle qi_{\max} - \frac{Zona2}{100}\cdot\left|qi_{\max} - qi_{\min}\right|; qi_{\max} \right\rangle \end{aligned} \tag{41}$$

Parametr „Zóna2“ je opět v procentech z velikosti pracovního rozsahu pohonu $qi$ a lze jej měnit v rozsahu od 1% do 10%.

## 8.3 Interpolace trajektorie

Přepočtenou trajektorii do kloubových souřadnic je nutné interpolovat, aby byl pohyb robota dostatečně hladký. Program umožňuje volbu mezi lineární interpolací, nebo mezi interpolací kubickým splinem (kapitola 4.3). Než začneme interpolaci provádět, musíme si zvolit časový krok pro interpolaci $T$. Čím menší krok interpolace zadáme, tím spojitější bude pohyb robota. V následujícím vztahu je ukázán výpočet počtu bodů, které vložíme mezi dvě vypočtené kloubové souřadnice.

$$N = \frac{t_{k+1} - t_k}{T}; N \in \mathbb{N} \tag{42}$$

Protože potřebujeme, aby bylo $N$ přirozené číslo, uvažujeme z podílu (42) pouze jeho celočíselnou část zaokrouhlenou vždy směrem dolů. Časové rozestupy mezi body pro interval $k$ pak budou:

$$T_v = \frac{t_{k+1} - t_k}{N} \tag{43}$$

Je potřeba, aby velikost kroku interpolace $T$ nikdy nebyla větší nebo rovna než časové rozestupy mezi jednotlivými body. Musí tedy platit:

$$T < t_{k+1} - t_k \tag{44}$$

Program podmínku (44) vyhodnocuje a při jejím nesplnění vyzve uživatele k zmenšení kroku $T$. V programu je referenčně nastaveno $T$ na $0,01s$.

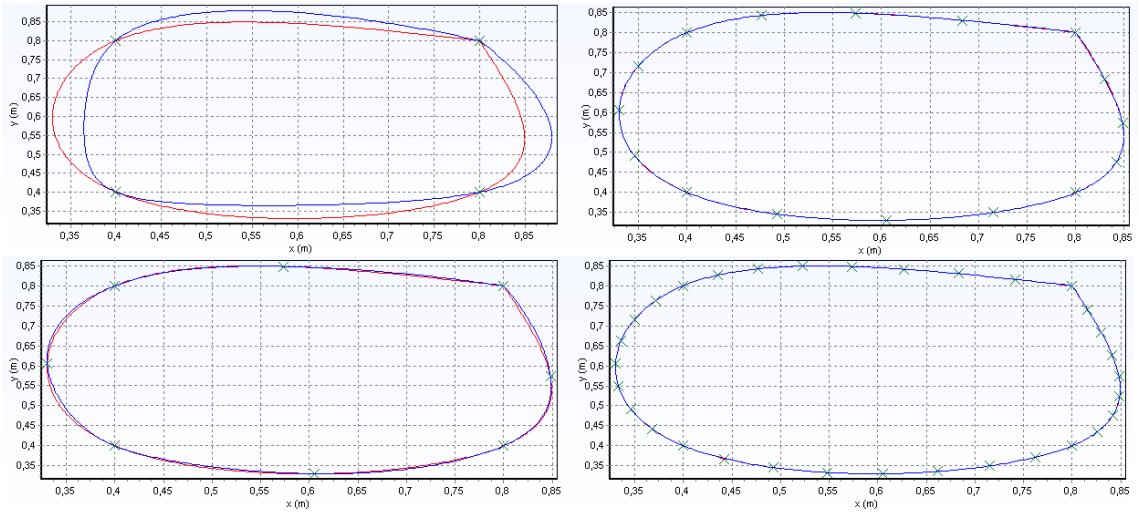
Po provedení vybrané interpolace program opět vyhodnocuje, zda všechny vložené body kloubových souřadnic leží v pracovním prostoru robota, nebo jestli nejsou v blízkosti jeho hranice. Při případném problému je uživatel informován. Při použití interpolace kubickým splinem se totiž může trajektorie dostat mimo pracovní rozsah robota, ačkoli všechny body pro interpolaci v pracovní oblasti leží.

V programu je prováděna nejen interpolace v úrovni kloubových souřadnic, ale i interpolace zadaných bodů v systému základny. Zadaná interpolovaná trajektorie v systému základny a inverzní interpolovaná trajektorie v kloubových souřadnicích jsou pak dále zdrojem pro grafické a tabelární zpracování výsledků.

## 8.4 Grafické a tabelární zpracování trajektorie robota

Je-li vypočtena inverzní úloha trajektorie a je-li provedena její interpolace, program získané výsledky automaticky graficky a tabelárně zpracuje. Je umožněno sledovat časové průběhy pohybu jednotlivých kloubových souřadnic robota $q_1 ... q_n = f(t)$, časové průběhy souřadnic základny $x, y, z, \alpha, \beta, \gamma = f(t)$ (podle počtu stupňů volnosti robota se některé časové průběhy orientace nezobrazují) a také funkce $x = f(y), x = f(z), y = f(z)$. V tabulce lze jednotlivě zobrazit zadanou trajektorii v systému základny, interpolovanou trajektorii v systému základny, trajektorii v kloubových souřadnicích a interpolovanou trajektorii v kloubových souřadnicích.

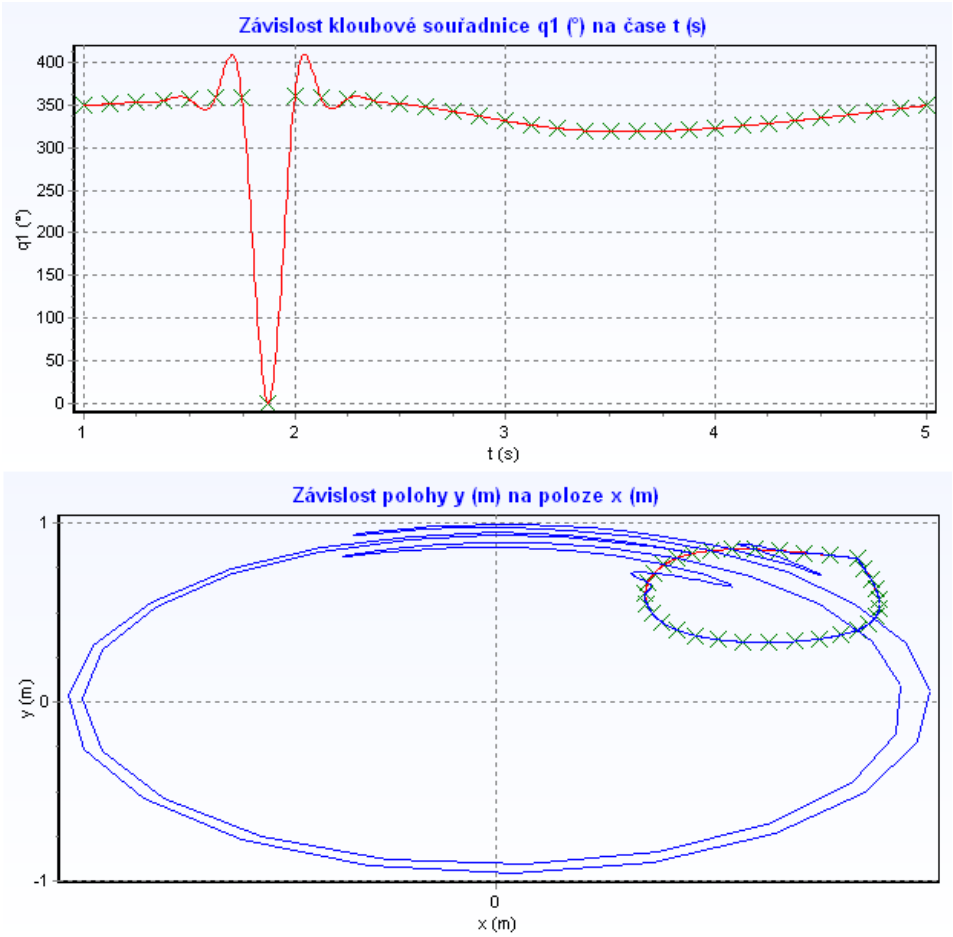
Abychom mohli graficky zobrazit skutečný pohyb robota, program také vypočte přímou úlohu přepočtené trajektorie v kloubových souřadnicích do systému základny. Uvidíme tak názorně, jak moc se pohyb robota blíží požadovanému. Vybereme-li z požadované trajektorie několik málo bodů, které jsou od sebe relativně dosti vzdálené, robot jimi projede (resp. s minimální chybou), ale pohyb mezi zadanými body nelze předpovídat. Přidáváme-li ale k trajektorii pomocí zjemňování (kapitola 8.1) stále více bodů, výsledný pohyb robota konverguje k požadované trajektorii. To je ukázáno na obrázku č. 14.




*Obr. 14: Ukázka konvergence pohybu robota k požadované trajektorii. V grafech je zobrazena závislost y=f(x), kde červená křivka je požadovaná trajektorie a modrou křivkou je vykreslen skutečný pohyb robota.*

Na obrázku č. 14 je první trajektorie definována pouze čtyřmi body, a pohyb robota je tak od požadované trajektorie poměrně vzdálen. Pod ní je už nadefinováno osm bodů. Pohyb robota se už výrazně trajektorii přiblížil, nicméně jsou stále vidět drobné odchylky. Při zadání 16 a 32 bodů je už pohyb robota s požadovanou trajektorií téměř shodný. Je tedy zřejmé, že od určitého počtu bodů není potřeba dalšího zjemňování, protože výsledný efekt zpřesnění pohybu robota by byl mizivý.

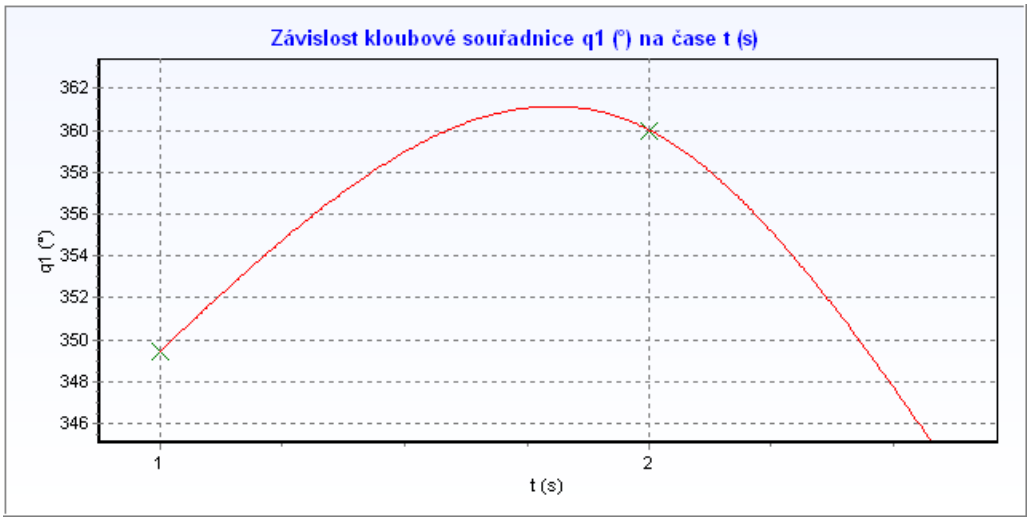
Dále se můžeme podívat, jak by se mohl robot chovat v situaci popsané obrázkem č. 13. Zde se robot snaží dostat z jednoho konce pracovního rozsahu na druhý.




*Obr. 15: Pohyb robota v situaci demonstrované obrázkem č. 13. Robot se snaží dostat z jednoho konce pracovního rozsahu na druhý. Prac. rozsah pohonu je (0° - 360°)*

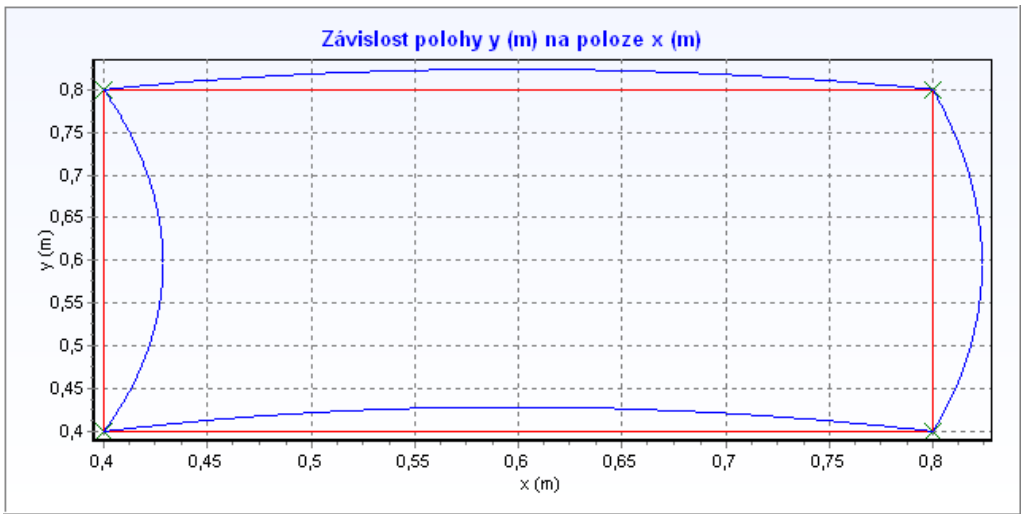
V horním grafu obr. č. 15 je zobrazena časová závislost pohybu pohonu, který je reprezentován kloubovou souřadnicí $q_1$. Je zde vidět, jak se během krátkého časového okamžiku musí pohon dostat z jedné krajní polohy do druhé (výrazný propad v charakteristice). Navíc je pohon nucen překročit svůj pracovní rozsah (0°-360°). V dolním grafu je pak modrou křivkou zobrazen pohyb robota v rovině $xy$. V ideálním případě by měl robot projet body označené křížkem podobně jako je tomu na obr. 14. Tento problém lze vyřešit jinou volbou počátečního (výchozího) bodu. Robot tak začne pohyb z jiné části pracovního prostoru a nedostane se za jeho hranici.

V předchozí kapitole byl uveden problém, kdy se přepočtená trajektorie do kloubových souřadnic velmi blíží ke konci pracovního rozsahu robota. Zdánlivě by tedy nemělo dojít k žádným komplikacím, protože jsou všechny body uvnitř pracovního prostoru robota a nedochází k problému z obrázků č. 13 a 15. Přesto se díky aproximaci kubickým splinem můžeme dostat za hranici pracovního prostoru. Podívejme se na následující obr. č. 16.




*Obr. 16: Ukázka interpolace kubickým splinem, kde dochází k překmitu mimo pracovní oblast robota (rozsah pohonu $q_1$ je 0°-360°).*

Naposledy se podíváme na průběh pohybu robota za použití lineární interpolace. V kapitole 4.3.1 bylo uvedeno, že lineární interpolace vede ke spojitému pohybu, ale k nespojité rychlosti a zrychlení. Na následujícím obrázku č. 16 můžeme vidět, jak se v zadaných bodech láme křivka pohybu robota. Při náhlé změně směru pohybu se skokově mění rychlost, a k tomu potřebujeme nekonečně velký impuls zrychlení. Je tedy zřejmé, že použití lineární interpolace je v praxi nevhodné. Dále si můžeme všimnout toho, že ačkoli na úrovni kloubů používáme lineární interpolaci, výsledný pohyb robota není přímkový (modrá křivka).




*Obr. 17: Ukázka pohybu robota za použití lineární interpolace. Pohyb robota je vykreslen modrou křivkou.*

Grafický a tabelární výstup programu lze ukládat i do souborů. Grafické závislosti se ukládají do bitmapového typu (*.bmp), tabulky bodů trajektorie se ukládají do textového souboru. Pokud ukládáme tabulky, program vygeneruje celkem čtyři soubory. První soubor obsahuje zadanou trajektorii, druhý interpolovanou zadanou trajektorii, třetí trajektorii v kloubových souřadnicích a konečně čtvrtý obsahuje interpolovanou trajektorii v kloubových souřadnicích.

Program ještě vyhodnocuje absolutní chyby polohy, popř. orientace, pro zadané body trajektorie. Počítá ji ze zadaných bodů a z přímé úlohy inverzní trajektorie pro každou souřadnici polohy a orientace systému základny. Jednotlivé chyby lze zobrazit v tabulce. Program ale navíc nalezne maximální chybu polohy a orientace a zobrazí je zvlášť. Chyba polohy je uváděna v milimetrech a chyba orientace je ve stupních. Jejich velikost je závislá na hodnotě chyby minimalizace $e$ (kap. 7.2), kterou lze měnit v nastavení programu. Za použití $e = 0,0001$ se maximální chyba polohy pohybuje řádově v setinách milimetrů a chyba orientace v setinách stupně. Samozřejmě záleží na použitém robotu.

# Závěr

Problém řešení inverzní kinematické úlohy a plánování trajektorie robotických manipulátorů a robotů vůbec je v současnosti na vysoké úrovni. Nelze tedy očekávat, že bakalářská práce vnese do této problematiky revoluční poznatky. Nicméně při řešení inverzní kinematické úlohy a plánování trajektorie robotů vzniká velké množství problémů, se kterými je nutné se seznámit, a není jednoduché sestavit algoritmus, který by řešil inverzní úlohu dostatečně rychle a zároveň byl univerzální. Není ani jednoduché plánovat trajektorii tak, aby se pamatovalo na veškerá možná kritéria (optimální trajektorie, trajektorie s nejnižší spotřebou energie, apod.).

Navržený program v bakalářské práci v jazyku Delphi splňuje univerzálnost použití metody řešení inverzní úlohy pro libovolnou kinematickou strukturu s maximálně šesti stupni volnosti. Je ale navržen pro reálné rozměry robotů. Nelze tedy předpokládat stabilní chování programu při zadávání nesmyslných hodnot. Rychlost výpočtu inverzní úlohy v reálném čase je splněna u plánování trajektorie. Pokud však hledáme všechna řešení inverzní úlohy pro jeden bod v systému základny, prohledávání pracovního prostoru robota může být v závislosti na použité kinematické struktuře robota a jemnosti prohledávání zdlouhavější.

Řízení pohybu robota samozřejmě nekončí kinematikou robota. Je nutné dále řešit jeho dynamiku. Také je dobré zavádět zpětnou vazbu, abychom znali skutečný pohyb robota a vyregulovali tak pomocí řídících obvodů případné odchylky způsobené například pružnou konstrukcí robota.

V příloze této práce naleznete stručný návod pro obsluhu programu, který je přiložen včetně zdrojového kódu na CD. Návod je sestaven tak, aby uživatel co nejrychleji pochopil logiku ovládání programu. U popisků jednotlivých částí programu, kde je potřeba podrobnějšího vysvětlení, je uveden odkaz na příslušnou kapitolu práce.

Dále je v příloze uveden jednoduchý program pro řešení inverzní úlohy Newtonovou aproximační metodou pouze pro robota RHINO SCARA (kap. 6.4). Je zde pouze nastíněna jiná numerická metoda pro řešení inverzní úlohy, než která je použita v hlavní aplikaci této bakalářské práce.

## Použitá literatura


[1] ZÁDA, V. *Robotika*, Přednášky TUL 2008.

[2] ZÁDA, V. *Základy robotiky*, Přednášky TUL 2007.

[3] MOSTÝN, V. *Mechatronika* [online]. [cit. 2008-04-20]. Dostupné z WWW: <http://robot.vsb.cz/elekskripta/mechatronika.pdf>.

[4] REKRORYS, K. *Přehled užité matematiky I, II*. Praha : Prometheus, 1995.

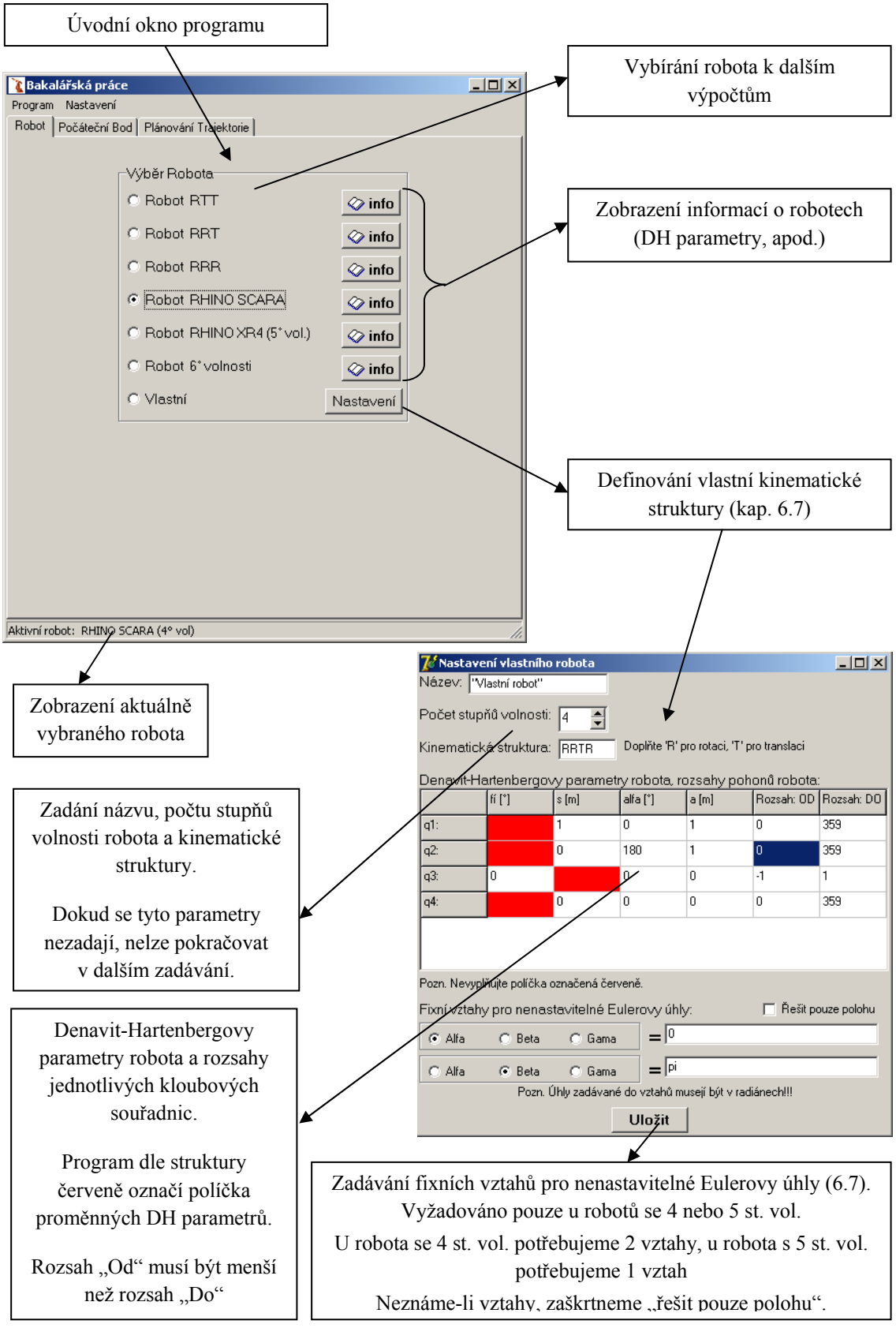
[5] SMUTNÝ, V. *Robotika* [online]. 2000 [cit. 2008-04-20]. Dostupné z WWW: <http://cmp.felk.cvut.cz/cmp/courses/ROB/ROB08S/rob.html>.

[6] VUKOBRATOVIČ, M, KIRČANSI, M. *Scientific Fundamentals of Robotics*. New York : Springer-Verlag, 1986.

[7] ŠUSTEK, M. *Vyhodnocování výrazů* [online]. 2004 [cit. 2008-04-29]. Dostupné z WWW: <http://www.pcsvet.cz/art/article.php?id=5358>.

# Přílohy bakalářské práce

## A Stručný návod pro ovládání programu



Nastavení přesnosti výpočtu inverzní úlohy prostřednictvím chyby minimalizace „p“ (kap. 7.2)

Nastavení na kolik částí se rozdělí pracovní rozsah pohonů robota při hledání všech řešení inverzní úlohy (kap. 7.3)

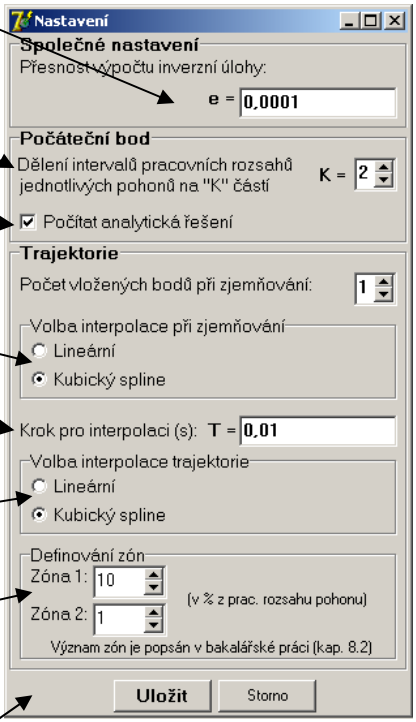
Nastavení, zda se mají navíc počítat analytická řešení inverzní úlohy u vybraných robotů

Nastavení druhu interpolace trajektorie v úrovni systému základny při zjemňování trajektorie (8.1)

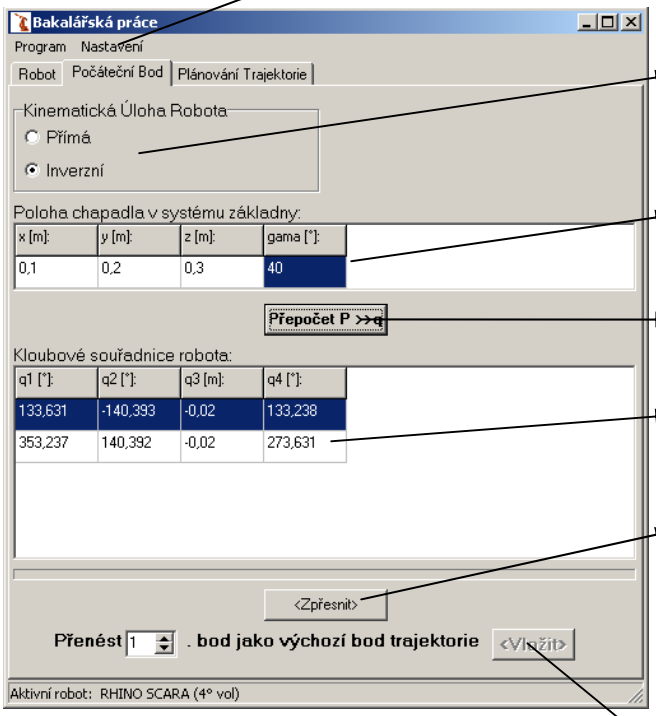
Časový krok interpolace. Čím menší krok, tím hladší bude průběh interpolované křivky

Nastavení druhu interpolace trajektorie v úrovni kloubů robota a systému základny

Nastavení velikosti zón v % z velikosti pracovního rozsahu pohonu (podrobně kap. 8.2)




Nastavení programu




Volba kinematické úlohy robota (přímá / inverzní úloha)

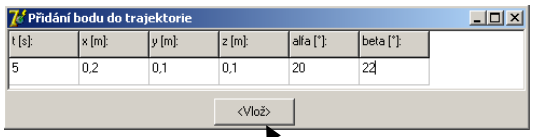
Vstup pro inverzní (přímou) úlohu žádaná poloha (natočení kloubů).

Spouštěcí tlačítko výpočtu kinematické úlohy

Řešení inverzní (přímé) úlohy

Tlačítko pro zpřesnění výsledků inverzní úlohy. Je provedeno několik dalších kroků gradientní metody

Tlačítko pro vložení vybraného počátečního bodu do trajektorie. Volbu bodu provádíme pomocí šipek. Tlačítko „Vložit“ je aktivní, pokud byl výpočet aktivován ze záložky „Plánování trajektorie“

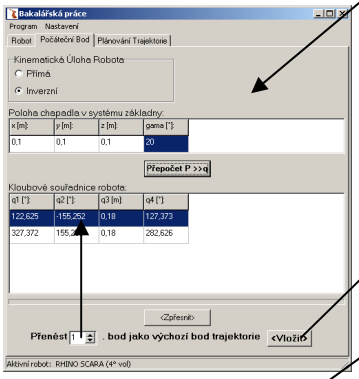


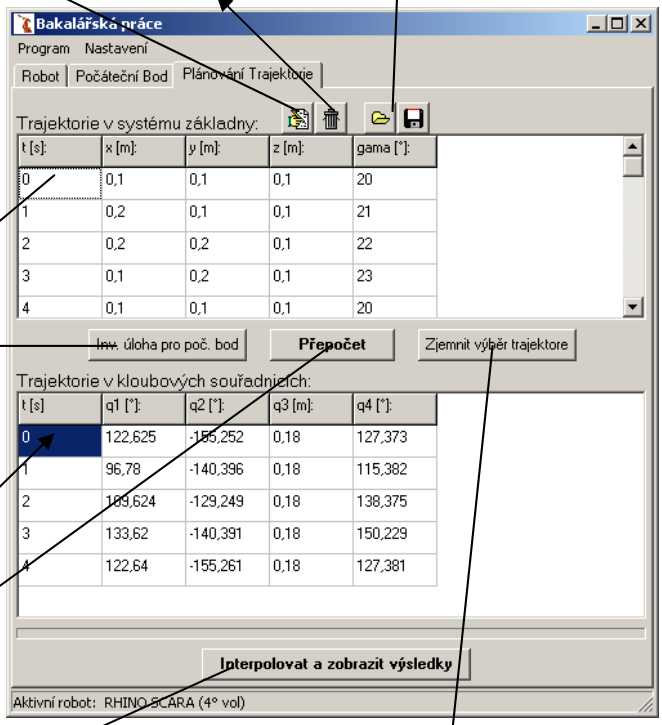

Tlačítko pro vložení nového bodu do trajektorie

Tlačítka pro načtení/uložení zadávané trajektorie z/do textového souboru

Tlačítko pro odebrání označených bodů z trajektorie

Výpočet inverzní úlohy počátečního bodu trajektorie. Proces se spustí automaticky po stisku tlačítka. Po výběru výchozího bodu se program vrátí zpět do záložky „Plánování trajektorie“.

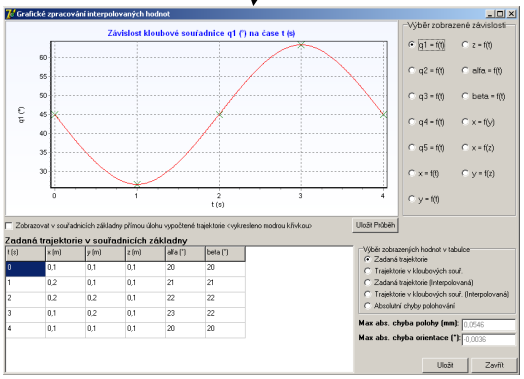



Spuštění výpočtu inverzní úlohy zadané trajektorie

Interpolace vypočtené inverzní trajektorie a zobrazení grafického a tabelárního zpracování výsledků

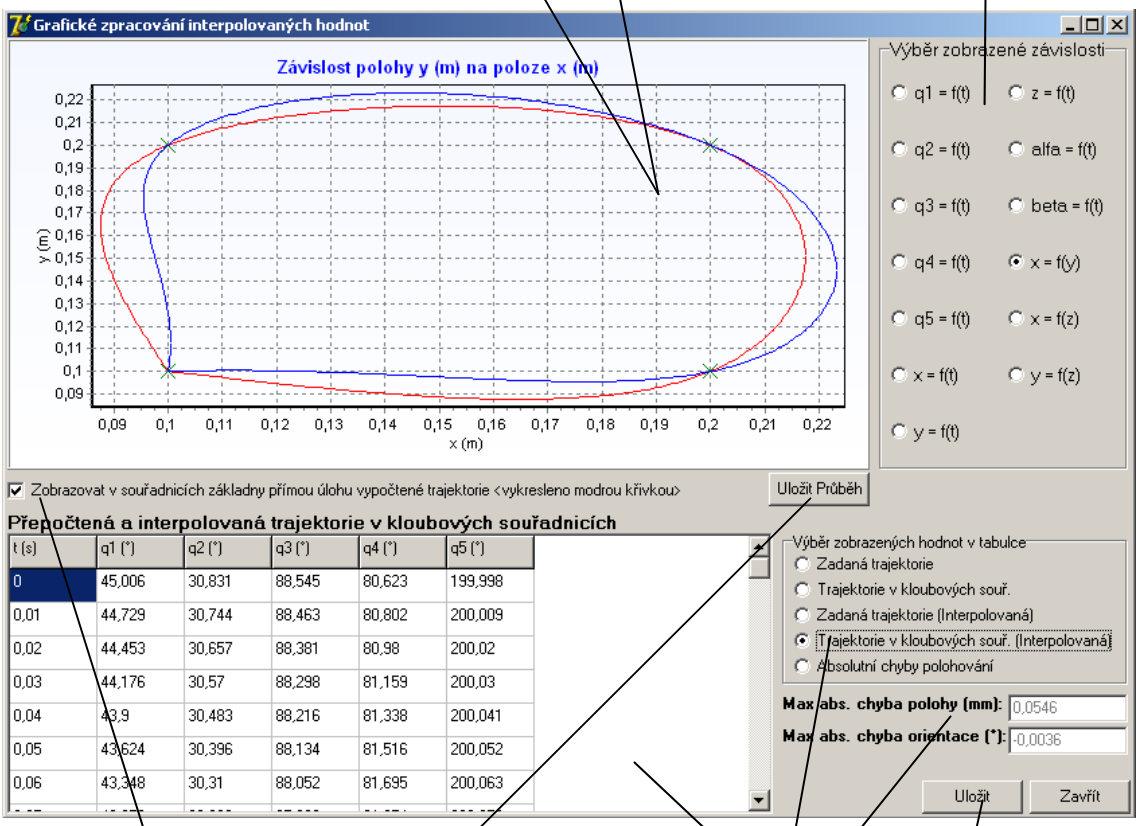
Tlačítko pro zjemnění trajektorie zvolenou aproximací (kap. 8.1)



Zadané body trajektorie jsou v grafu vyznačeny křížkem. Křivky jsou tvořeny interpolovanými trajektoriemi.

Grafické zpracování vybrané závislosti. Lze použít funkci zoom tak, že myší označíme zleva doprava část grafu ke zvětšení. Zoom se zruší označením části grafu zprava doleva.

Vybrání zobrazené závislosti v grafu




Po zaškrtnutí políčka se modrou křivkou zobrazuje přímá úloha inverzní trajektorie v závislostech systému základny.

Tabelární zpracování výsledků. Význam hodnot v tabulce je dle výběru.

Zobrazení maximální chyby polohy a orientace

Tlačítko pro uložení grafu do obrázku formátu *.bmp

Tlačítko pro uložení všech tabulek do textového souboru

# B Program pro řešení inverzní úlohy Newtonovou aproximační metodou

Program je napsán pro aplikaci Matlab. Řeší inverzní úlohu pouze pro robotickou strukturu RRT-R robota RHINO SCARA (kapitola 6.4). Jeho vstupem je požadovaná poloha a orientace chapadla a prvotní počáteční odhad řešení inverzní úlohy, který nesmí být od přesného řešení příliš vzdálený. Metoda by pak mohla divergovat. Výstupem je jedno řešení inverzní kinematické úlohy. Algoritmus je napsán podle kapitoly 3.1.3, kde je vysvětlen princip Newtonovy aproximační metody. V programu však není definovaná chyba, při které se výpočet ukončí, ale provádí se předem definovaný počet kroků metody.

Zdrojový kód programu (M-filu):

```
%Známé konstantní DH parametry
s1=0.43;a1=0.33;a2=0.33;s4=0.15;
%Požadovaná poloha a orientace
x=0.1;
y=0.2;
z=0.3;
gama=40;
%Počáteční odhad
q=[120;-120;0;120];
for i=1:1000
  %Soustava nelineárních rovnic pro výpočet polohy a orientace
  %pomocí zobecněných souřadnic q
  F=[a2*cosd(q(1)+q(2))+a1*cosd(q(1))-x;
     a2*sind(q(1)+q(2))+a1*sind(q(1))-y;
     -q(3)+s1-s4-z;
     q(4)-q(1)-q(2)+180+gama];
  %Jakobián funkce F (parciální derivace podle složek q)
  J=[-sind(q(1)+q(2))-sind(q(1))  -sind(q(1)+q(2))    0  0;
      cosd(q(1)+q(2))+cosd(q(1))   cosd(q(1)+q(2))    0  0;
                               0                  0  -1  0;
                              -1                 -1   0  1];
  %Iterační vzorec Newtonovy aproximační metody
  dQ=-inv(J)*F;
  q=q+dQ;
end
%Výsledný vektor zobecněných souřadnic pro požadovanou
%polohu a orientaci
q
```

Po spuštění programu vyjde následující řešení pro kloubové souřadnice:

q1 = 133,5853°

q2 = -140,3321°

q3 = -0,02m

q4 = 133,2532°