

## THE USING SOME APPROXIMATING METHODS BY INVERSE MODELLING

P. Miček<sup>1</sup>, S. Věchet<sup>1</sup>, T. Březina<sup>2</sup>

**Summary:** *The creation of direct kinematic modell is relatively simply in contrast with creation of analytic inverse modell. With using of approximating methods is inverse modelling more simpler. For inverse modell creation with approximation methods is necessary to have direct kinematic modell or simulation data. This paper compares making of inverse kinematic modells by different approximation methods using.*

### 1. Úvod

Pro návrh, optimalizaci a následné řízení pedových mechanismů je nutné jejich modelování. Používané modely lze rozdělit na kinematické, kinetostatické a dynamické, přímé a inverzní. Obecně lze říci, že sestavit inverzní model je vždy mnohem obtížnější ve srovnání se sestavení přímého modelu.

Přímý model sestavený na základě transformačních matic je reprezentován soustavou rovnic, v nichž figurují goniometrické funkce. Právě ty pak značně komplikují řešení těchto rovnic (Mostýn & Skařupa (online 2003)). Aproximační metody modelování se snaží nahradit složité vztahy v soustavách rovnic přímého modelu aproximací polohy a natočení v prostoru v okolí známého bodu.

Příspěvek uvádí využití Neuro-fuzzy přístupu, metody Lazy Learning, metody Lokálně vážené regrese a umělé Neuronové sítě pro tvorbu inverzního modelu. Výsledkem aproximace je inverzní model soustavy, reprezentovaný fuzzy-inferenčním systémem, množinou tréninkových dat společně s metodou Lazy Learning event, metodou Lokálně vážené regrese a naučenou neuronovou sítí. Srovnání metod je provedeno na základě výsledků použití uvedených metod na příkladech dvou různých robotických pedových mechanismů.

### 2. 2D inverzní kinematický model

Vybrané aproximační metody (Neuro-fuzzy, Lazy Learning, Lokálně vážená regrese, Neuronové sítě) byly použity na příkladu tvorby inverzního modelu dvou pantografických noh, viz obrázek 1.

---

<sup>1</sup>Ing. Pavel Miček, Ing. Stanislav Věchet: ÚMT FSI VUT Brno; Technická 2; 616 69 Brno; e-mail: miček@umtn.fme.vutbr.cz, vechet@umtn.fme.vutbr.cz

<sup>2</sup>RNDr. Ing. Tomáš Březina, CSc.: UAI FSI VUT Brno; Technická 2; 616 69 Brno; e-mail: brezina@uai.fme.vutbr.cz

Mechanismus nohy 1 (obrázek 1(a)) má následující parametry:

$y_U$  - poloha motoru U  $\langle 5 \text{ mm}, 75 \text{ mm} \rangle$

$r_V$  - poloha motoru V  $\langle -50 \text{ mm}, 75 \text{ mm} \rangle$

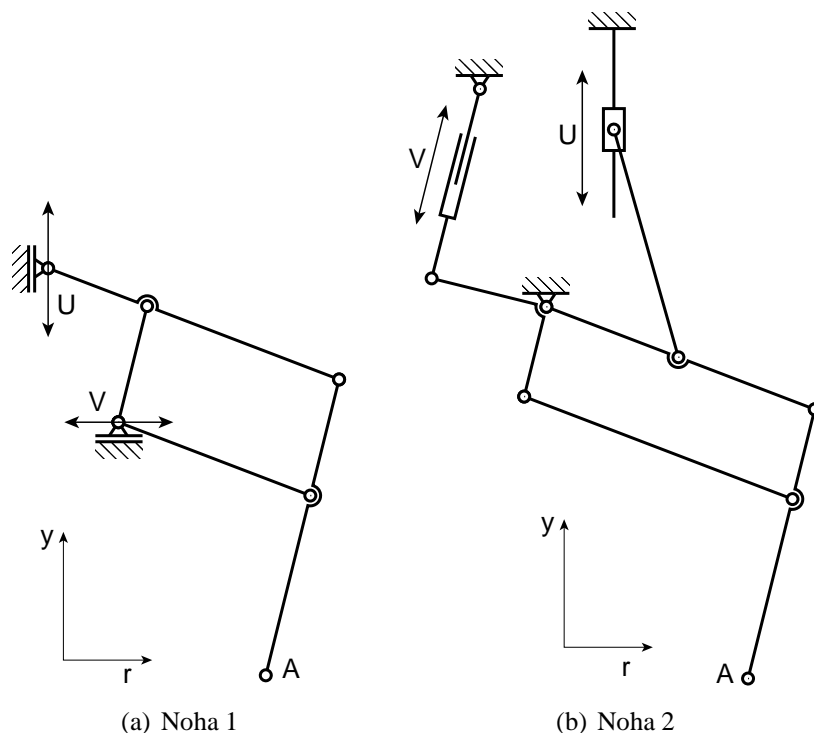
$r_A, y_A$  - poloha bodu A

Další typ pantografické nohy je uveden na obrázku 1(b) s parametry mechanismu:

$l_7$  - poloha motoru V  $\langle 104 \text{ mm}, 192 \text{ mm} \rangle$

$y_8$  - poloha motoru U  $\langle 224 \text{ mm}, 372 \text{ mm} \rangle$

$r_A, y_A$  - poloha bodu A



Obrázek 1: Modelované pantografické nohy

Vstupem do přímého kinematického modelu jsou polohy motorů  $[y_U, r_V]$ , výstupem přímého kinematického modelu jsou souřadnice koncového bodu mechanismu  $[r_A, y_A]$ . U inverzního kinematického modelu je tomu naopak, lze jej zapsat ve tvaru

$$[r_A, y_A] \rightarrow [y_U, r_V]. \quad (1)$$

Podprostor  $\mathbb{R}^2$  daný možným rozsahem pohonů  $U$  a  $V$  uvažujeme zjednodušeně čtvercový, jeho redukce se provede pomocí omezujících podmínek daných konstrukcí nohy robotu. Rozsahy poloh pohonů  $U$  a  $V$  jsou stejnoměrně rozděleny na  $n$  dílů a takto je určena matice  $n_U \times n_V$  vstupních bodů v  $\mathbb{R}^2$ . Řešením přímé úlohy jsou získány odpovídající matice souřadnic  $r_A$  a  $y_A$ , které jsou v inverzním modelu použity jako vstupy.

V následující kapitolách jsou popsány základní principy a výsledky jednotlivých použitých metod.

### 3. Neuro-fuzzy systém a jeho nastavení

Neuro-fuzzy systémy je možné použít k aproximaci funkcí, které je problematické popsat analyticky (Miček (2002)). Neuro-fuzzy systém využívá ideí fuzzy systému a učení pomocí neuronových sítí.

Typický fuzzy systém se skládá z příslušnostních funkcí, tabulky pravidel a inferenčního stroje. Na základě vstupů do fuzzy systému zpracovaných příslušnostními funkcemi do pojmů (termů) provádí fuzzy systém vyhodnocení tabulky pravidel a inferenční stroj pak převádí získané vyhodnocení do konkrétních hodnot výstupů fuzzy systému.

Neuro-fuzzy systém `anfis` (ADAPTIVE NETWORK BASED FUZZY INFERENCE SYSTEM) je oproti typickému fuzzy systému rozšířen o hledání parametrů Sugeno-fuzzy modelu pomocí hybridního učícího se mechanismu. Tento hybridní mechanismus je založen na využití metody nejmenších čtverců a neuronové sítě s Levenberg-Marquardt algoritmem. `Anfis` v průběhu učení pomocí tréninkových párů (vstupní hodnoty a požadované výstupní hodnoty) stanoví parametry fuzzy systému a vytvoří příslušnou FIS matici.

Pro oba neuro-fuzzy modely byly jako příslušnostní funkce použity zobecněné Bellovy funkce (Generalized Bell Function) viz The MathWorks, Inc. (2001) ve tvaru uvedeném v rovnici 2.

$$f(x; a, b, c) = \frac{1}{1 + \left| \frac{x-c}{a} \right|^{2b}} \quad (2)$$

kde  $x$  je nezávislá proměnná a  $a, b, c$  jsou parametry funkce.

### 4. Metoda Lazy Learning

Lazy Learning patří mezi metody lokálního modelování. Metoda nevytváří v režimu učení strukturu, která by pak prováděla aproximaci, tak jako je tomu u Neuro-fuzzy nebo Neuronové sítě. Veškeré výpočty potřebné pro aproximaci se vykonávají až v okamžiku obdržení dotazu, kdy je sestaven lokální model a provedena jeho strukturální a parametrická identifikace (Bontempi & Birattari & Bersini (1999)). Strukturální identifikace provádí výběr vhodné skupiny lokálních aproximátorů, výběr metriky pro posouzení vhodnosti aproximace a také velikosti oblasti, jejíž reprezentanti jsou použiti. Parametrická identifikace pak vykonává optimalizaci nastavení parametrů lokálních aproximátorů (Birattari (1999)).

THE LAZY LEARNING TOOLBOX FOR USE WITH MATLAB (LLT) používá jako lokální aproximátory polynomy různých stupňů, problém volby vhodné oblasti redukuje na problém volby  $k$  nejbližších sousedů. V okamžiku požadavku na predikci je na základě dotazu vytvořena množina lokálních modelů s polynomy různých stupňů a s různým počtem použitých sousedů. Každý model je ohodnocen a finální predikce je pak vytvořena kombinací nebo výběrem z množiny modelů, použit je algoritmus rekurzivní metody nejmenších čtverců.

Popsaný přístup se odlišuje od metod, které vytváří globální model a provádějí výpočty ve fázi učení. Příkladem jsou umělé neuronové sítě. Oba přístupy mají pochopitelně své výhody, limitujícím faktorem určujícím použitelnost LLT je čas potřebný pro výpočet po obdržení dotazu.

## 5. Lokálně vážená regrese

Mezi nejjednodušší algoritmy s lokálním lineárním modelem patří Lokálně vážená regrese (Locally weighted learning - LWR, Atkeson & Moore & Schaal (1997)). Predikce pro dotazovaný bod  $x_q$  je generována následujícím algoritmem vážené regrese:

### Algoritmus Lokálně vážené regrese

*Dáno:*

Dotazovaný bod  $x_q$  a  $p$  tréninkových bodů  $\{x_i, y_i\}$

*Výpočet predikce:*

a) výpočet diagonální matice vah  $W$

$$w_{ii} = \exp\left(-\frac{1}{2}(\mathbf{x}_i - \mathbf{x}_q)^T \mathbf{D}(\mathbf{x}_i - \mathbf{x}_q)\right)$$

b) sestavení matice  $X$  a vektoru  $y$

$$\mathbf{X} = (\tilde{\mathbf{x}}_1, \tilde{\mathbf{x}}_2, \dots, \tilde{\mathbf{x}}_p)^T \text{ kde } \tilde{\mathbf{x}}_i = [(\mathbf{x}_i - \mathbf{x}_q)^T \mathbf{1}]^T$$

$$\mathbf{y} = (y_1, y_2, \dots, y_p)^T$$

c) výpočet lokálního lineárního modelu

$$\beta = (\mathbf{X}^T \mathbf{W} \mathbf{X})^{-1} \mathbf{X}^T \mathbf{W} \mathbf{y}$$

d) predikce pro  $x_q$  je:

$$\hat{y}_q = \beta_{n+1}$$

$\beta_{n+1}$  značí (n+1)-tý element vektoru  $\beta$ . Výpočtová náročnost LWR je úměrná  $pn^2$ . Běžně je většina bodů z množiny  $p$  tréninkových dat aproximována nulovou vahou, protože jsou od dotazovaného bodu příliš vzdáleny což ve značné míře přispívá k redukci výpočtové náročnosti. LWR může být aplikováno na problémy v reálném čase pokud mají malý počet vstupů  $n$ .

Jediný neznámý parametr v algoritmu 1 je metrika vzdálenosti  $\mathbf{D}$ . Při značném množství dat, může být  $\mathbf{D}$  optimalizováno vynecháváním dotazovaného bodu tzv. leave-one-out cross validation.

## 6. Vícevrstvá dopředná Neuronová síť s učícím algoritmem Levenberg-Marquardt

Umělé neuronové sítě jsou inspirovány skutečnými neuronovými sítěmi vyšších organismů, z technického hlediska je umělá Neuronová síť mnoha navzájem bohatě propojených procesorů nejčastěji realizovaných softwarově. Vlastnosti sítě jsou určeny její architekturou a typem nelinearit neuronů, kterou tvoří neurony a jejich propojení a zvoleným trénovacím algoritmem.

Jedno z velmi častých uspořádání je právě vícevrstvá dopředná síť a trénovací algoritmus Levenberg-Marquardt. Počet neuronů ve vstupní a výstupní vrstvě je dán počtem vstupů a výstupů modelu, počet skrytých vrstev a počet neuronů v každé z nich je volitelný. Je matematicky dokázáno, že při použití jedné skryté vrstvy je síť schopna aproximovat libovolně složitou funkci, přesto někdy dostaneme lepší numerické výsledky při použití dvou skrytých vrstev. Neexistuje univerzální doporučení jak volit počet neuronů ve skryté vrstvě.

Síť pracuje ve dvou základních režimech, v režimu učení, kdy jsou síti předkládány vstupy a požadované výstupy a podle těchto informací jsou upravovány váhové koeficienty a prahy jednotlivých neuronů. V pracovním režimu je síti předložen neznámý vstup na který síť při správném natrénování správně odpoví.

K vytvoření a naučení neuronové sítě jsme použili **The Matlab Neural Network Toolbox**.

Síť má dva vstupy a dva výstupy.

Pro první model se jako nejvýhodnější ukázala konfigurace: dvě skryté vrstvy s počty neuronů 6,5 a nelinearitou neuronu typu `tansig`, trénovací funkce `trainlm` - Levenberg–Marquardt algoritmus, počet trénovacích epoch = 1000.

Pro druhý model jsme použili stejné parametry, pouze počty neuronů ve skrytých vrstvách byly 5,5.

## 7. Dosažené výsledky modelování

Pro natrénování a testování modelů byla využita data, získaná simulací přímého kinematického modelu. Aproximační metody byly srovnávány na základě následujících hledisek:

- velikost matic trénovacích dat jednotlivých modelů (tabulka 1)
- doba zpracování je pro použité metody uvedena (tabulka 2<sup>3</sup>)
- velikost SIGE (Signed Error, viz rovnici 4) a RMSE (Root Mean Square Error) (tabulka 3)

Pro Neuro-fuzzy modely a Neuronové sítě je dobou zpracování myšlena doba učení, pro metody Lazy Learning a Lokálně váženou regresi je doba zpracování čas potřebný k výpočtu parametrů specifických pro jednotlivé metody. Matice testovacích dat měla rozměr 100×100 (tj. rozsahy motorů U a V byly rovnoměrně rozděleny na 100 dílů).

$$\Delta = p - p_a \quad (3)$$

$$\text{SIGE} = \text{sgn}(\Delta) \max(|\Delta|) \quad (4)$$

kde  $p$  je přesná hodnota  $p_a$  aproximovaná hodnota.

Typ modelu	Noha	Velikost tréninkových matic	
		Motor U	Motor V
Neuro-fuzzy	1	9	9
	2	17	17
Lazy Learning	1	9	9
	2	55	55
Lokálně vážená regrese	1	13	13
	2	55	55
Levenberg-Marquardt	1	9	9
	2	45	45

Tabulka 1: Velikost tréninkových matic

Z tabulky 2 lze usuzovat, že pro mechanismus, který je jednodušší vychází ze srovnání doby zpracování nejlépe Neuro-fuzzy, pro složitější mechanismus se jako výhodnější ukazuje

<sup>3</sup>Simulace a výpočty byly prováděny na počítači s procesorem Pentium III 600MHz, 256MB RAM v prostředí MATLAB 5.4

Typ modelu	Noha	Doba zpracování [s]		
		Motor U	Motor V	Celkem
Neuro-fuzzy	1	0.11	0.111	0.221
	2	6.179	6.379	12.558
Lazy Learning	1	0.59	0.631	1.221
	2	4.126	4.146	8.272
Lokálně vážená regrese	1	6.99	7	13.99
	2	131.79	132.32	264.11
Levenberg- -Marquardt	1	23	-	23
	2	115	-	115

Tabulka 2: Časy učení modelů

Typ modelu	Noha	SIGE [mm]		RMSE [mm]	
		Motor U	Motor V	Motor U	Motor V
Neuro-fuzzy	1	-0.338112	0.15229	0.0706741	0.0438406
	2	-0.847539	2.42708	0.13415	0.442405
Lazy Learning	1	0.648677	0.789859	0.265332	0.257262
	2	0.807471	-0.511772	0.108053	0.0556702
Lokálně vážená regrese	1	0.277382	-0.519917	0.0976769	0.0624279
	2	1.03114	0.115752	0.213278	0.115752
Levenberg- -Marquardt	1	0.0521	-0.0410	0.0114	0.0092
	2	7.9037	-6.4150	1.7327	1.2128

Tabulka 3: Chyby modelů

metoda Lazy Learning. Příčina vyššího času učení u Neuro-fuzzy je ve značném nárůstu času učení pro vyšší počet příslušnostních funkcí. Pro komplikovanější datové struktury by tento vliv byl patrně ještě markantnější. Je třeba říci, že všechny metody vyžadovaly jisté období pokusů, při hledání vhodných parametrů jednotlivých metod.

Z hlediska paměťových nároků při implementaci v mikrokontroleru všechny testované metody modelování vyžadují implementaci příslušného algoritmu a dále prostor pro potřebná data. V případě Neuro-fuzzy jde o parametry příslušnostních a defuzifikčních funkcí a tabulku pravidel. V případě Lazy Learningu a Lokálně vážené regrese jde o celou tréninkovou množinu (viz tabulka 1) a v případě neuronové sítě pak hodnoty vah a prahů jednotlivých neuronů. Lazy Learning a Lokálně vážená regrese jsou z tohoto pohledu pro složitější modely nevýhodné.

Z hlediska přesnosti vybraných metod (tabulka 3) testovaných na uvedených příkladech vychází metody Lazy Learning a Lokálně vážená regrese s nejvyrovnanějšími výsledky, Neuro-fuzzy a neuronová síť mají lepší výsledky pro jednodušší model a horší výsledky pro složitější model. Všechny metody však mají přesnost přijatelnou po potřeby řízení.

## 8. Závěr

Celkově lze tedy usuzovat, že pro jednodušší mechanismus je vhodnější využít Neuro-fuzzy systém, nebo neuronovou síť. Pro složitější mechanismus je výhodnější použití Lazy Learningu event. Lokálně vážené regrese v případě, že paměťové nároky při implementaci nejsou podstatné. Je-li vliv paměťových nároků při implementaci podstatný, je výhodnější použití Neuro-fuzzy systému i za cenu nezanedbatelné doby učení.

## 9. Poděkování

Práce vznikla za podpory pilotního projektu ÚT AV ČR č. 52020 „Řízení kráčivého robotu s využitím metod umělé inteligence“, výzkumného záměru MŠMT MSM 262100024 „Výzkum a vývoj mechatronických soustav“ a výzkumného záměru CEZ:J22/98:261100009 „Netradiční metody studia komplexních a neurčitých systémů“.

## 10. Literatura

- Atkeson, C. G. & Moore, A. W. & Schaal, S. (1997) *Locally Weighted Learning* Artificial Intelligence Review, pp. 11-73.
- Bontempi, G. & Birattari, M. & Bersini, H. (1999) *Lazy learning for modeling and control design*, International Journal of Control, pp. 643-658.
- Birattari, M. (1999) *The Lazy Learning Toolbox for Matlab*, <http://iridia0.ulb.ac.be/~lazy/>.
- The MathWorks, Inc. (2001) *Fuzzy Logic Toolbox User's Guide*, The MathWorks, Inc.
- Miček P. (2002) *Omezení vlivu singularit při posuzování neuro-fuzzy modelů*, in CD-ROM: Proceedings of the Engineering Mechanics.
- Mostýn, V. & Skařupa, J. (online 2003) *Teorie průmyslových robotů*, [http://robot.vsb.cz/uvodni\\_strana/E\\_skripta.htm](http://robot.vsb.cz/uvodni_strana/E_skripta.htm).