# Doktorandský den '01

Ústav informatiky
Akademie věd České republiky

Praha, 25. říjen 2001

Obsah

# Doktorandský den ÚI AV 2001 – program

**— 25. říjen 2001 —**

**Časový rozvrh přednášek**

| | | |
|---|---|---|
| Petr Tichý | O odhadu A-normy chyby v metodě sdružených gradientů | $8^{30}$ |
| Ctirad Matonoha | Metody s lokálně omezeným krokem | $8^{55}$ |
| Petra Kudová | Učení neuronových sítí typu RBF | $9^{20}$ |
| Zuzana Petrová | Hybridní UI modely a Bang2 | $9^{45}$ |
| | přestávka 10 minut | |
| Milan Rydvan | Generalised Objective Functions for Multi-Layered Neural Networks with Genetic Training | $10^{20}$ |
| Terezie Šidlofová | Some Estimates of Rates of Approximation by Neural Networks Using Integral Representation | $10^{45}$ |
| Radek Pospíšil | Enhancements of Enterprise JavaBeans Component Model | $11^{10}$ |
| Stanislav Višnovský | Behavior Protocols and Component Lifecycle | $11^{35}$ |
| | přestávka na oběd | |
| Emil Kotrč | Překladač numerických funkcí zapsaných v C do Fortranu77 | $13^{00}$ |
| David Coufal | Incremental structure learning of Wang neuro-fuzzy system | $13^{25}$ |
| Zuzana Haniková | An interpretation of ZF in a fuzzy set theory | $13^{50}$ |
| Petr Cintula | On axiomatic systems of the various fuzzy logics | $14^{15}$ |
| | přestávka 10 minut | |
| Markéta Kyloušková | Statistical methods in genetic studies | $14^{50}$ |
| Petr Zavadil | Řízení a stabilizace v biosystémech | $15^{15}$ |
| | Slavností vyhlášení nejlepší prezentace – předseda vědecké rady ÚI AV ČR | $15^{45}$ |
| | Vyhodnocení doktorandského dne – ředitel ÚI AV ČR | $15^{50}$ |

# O odhadu $\mathbf{A}$-normy chyby v metodě sdružených gradientů


*doktorand:*
MGR. PETR TICHÝ
ÚI AVČR, Pod vodárenskou věží 2, Praha 8, ČR
petr.tichy@centrum.cz

*školitel:*
DOC. RNDR. JAN ZÍTKO, CSC.
MFF UK, Sokolovská 83, Praha 2, ČR
zitko@karlin.mff.cuni.cz


obor studia:
M6 – Vědecko-technické výpočty


**Abstrakt**

Metoda sdružených gradientů (CG) slaví své 50-té narozeniny. Byla srozumitelně popsána již svými stvořitely Hestenesem a Stiefelem [5], kteří odvodili všemožné algebraické vztahy panující mezi vektory a koeficienty algoritmu, ukázali souvislost CG s Gaussovou kvadraturou a minimalizací funkcionálu. Mohlo by se zdát, že již bylo řečeno vše. Nasazení CG na počítačích s konečnou aritmetikou však s sebou přineslo řadu dalších problémů. K jejich řešení bylo nezbytné pochopit chování a matematický model CG v konečné aritmetice a ukázala se nutnost řádné analýzy zaokrouhlovacích chyb ve formulích charakterizujících konvergenci. Jednou z těchto charakteristik je i $\mathbf{A}$-norma chyby, jejíž důležitost podtrhují aplikace ve fyzice a kvantové chemii (zde bývá $\mathbf{A}$-norma chyby nazývána energetickou normou) a která byla v samotné práci [5] označena za vhodného kandidáta na určení kvality počítané aproximace. Budeme prezentovat nový odhad $\mathbf{A}$-normy chyby odvozený algebraickou cestou a ukážeme, že nejjednodušší a numericky nejstabilnější odhad je "skryt" již v původní práci [5]. Vysvětlíme, že odhady $\mathbf{A}$-normy odvozené pomocí Gaussovy kvadratury jsou matematicky ekvivalentní odhadům odvozeným algebraickou cestou. Budeme diskutovat otázku fungování odhadů v aritmetice s konečnou přesností. Tento příspěvek je založen na práci [6].


## 1. Metoda sdružených gradientů (CG) a Gaussova kvadratura

Uvažujme systém lineárních rovnic

$$\mathbf{A}x = b,$$

kde $\mathbf{A} \in \mathbb{R}^{n\times n}$ je reálná symetrická pozitivně definitní matice a $b \in \mathbb{R}^n$ vektor pravé strany. Nechť $x_0$ je počáteční aproximace řešení. Potom $j$-tá aproximace řešení je určena podmínkou

$$\begin{aligned} x_j &\in x_0 + \mathcal{K}_j(\mathbf{A}, r_0) \\ \|x - x_j\|_{\mathbf{A}} &= \min_{u \in x_0 + \mathcal{K}_j(\mathbf{A}, r_0)} \|x - u\|_{\mathbf{A}}, \end{aligned}$$

t.j. minimalizuje $\mathbf{A}$-normu chyby $\|x - x_j\|_{\mathbf{A}} \stackrel{\text{def}}{=} \big((x - x_j), \mathbf{A}(x - x_j)\big)^{\frac{1}{2}}$ přes všechny aproximace z variety $x_0 + \mathcal{K}_j(\mathbf{A}, r_0)$, kde

$$\mathcal{K}_j(\mathbf{A}, r_0) \stackrel{\text{def}}{=} \text{span}\{r_0, \mathbf{A}r_0, \ldots \mathbf{A}^{j-1}r_0\}$$

je $j$-tý Krylovův prostor generovaný maticí $\mathbf{A}$ a počátečním residuem $r_0$, $r_0 = b - \mathbf{A}x_0$. Standardní implementace CG byla uvedena v [5, (3:1a)-(3:1f)]:

**Metoda sdružených gradientů** (CG)

**Input** $x_0$, $\mathbf{A}$, $b$
$r_0 = b - \mathbf{A}x_0$
$p_0 = r_0$
**do** $j = 0, 1, \ldots$

$$\gamma_j = \frac{(r_j, r_j)}{(p_j, \mathbf{A}p_j)}$$

$$\begin{aligned} x_{j+1} &= x_j + \gamma_j\, p_j \\ r_{j+1} &= r_j - \gamma_j\, \mathbf{A}p_j \end{aligned}$$

$$\delta_{j+1} = \frac{(r_{j+1}, r_{j+1})}{(r_j, r_j)}$$

$$p_{j+1} = r_{j+1} + \delta_{j+1}\, p_j$$

**end do**

Residuové vektory $\{r_0, r_1, \ldots, r_{j-1}\}$ tvoří ortogonální bázi a směrové vektory $\{p_0, p_1, \ldots, p_{j-1}\}$ $\mathbf{A}$-ortogonální bázi $j$-tého Krylovova prostoru $\mathcal{K}_j(\mathbf{A}, r_0)$. Uvedené ortogonální vztahy tvoří eleganci metody popsané v [5] a reprezentují podstatnou vlastnost, která spojuje CG s klasickým světem ortogonálních polynomů. Na residuum lze hledět jako na maticový polynom přenásobený vektorem $r_0$, $r_j = \varphi_j(\mathbf{A})r_0$ a lze ukázat, že polynomy $\varphi_j$ jsou ortogonální vzhledem k diskrétnímu skalárnímu součinu

$$(f, g) = \sum_{i=1}^{n} \omega_i f(\lambda_i) g(\lambda_i)\,, \tag{1}$$

s vahami $\omega_i$,

$$\omega_i \stackrel{\text{def}}{=} (v_1, u_i)^2, \qquad \sum_{i=1}^{n} \omega_i = 1,$$

kde $\lambda_i$ jsou vlastní čísla matice $\mathbf{A}$, $u_i$ příslušné normované vlastní vektory a $v_1 \stackrel{\text{def}}{=} r_0/\|r_0\|$. Víme, že kořeny ortogonálních polynomů hrají při aproximaci nějakého integrálu pomocí Gaussova kvadraturního pravidla roli uzlů. Souvislost mezi CG a Gaussovou kvadraturou je díky (1) nasnadě.

Definujme Riemann-Stieltjesův integrál následujícím způsobem

$$\int_\zeta^\xi f(\lambda)\, d\omega(\lambda) \stackrel{\text{def}}{=} \sum_{1}^{n} \omega_i\, f(\lambda_i),$$

kde $\omega(\lambda)$ je monotónní, po částech konstantní fukce mající $n$–bodů růstu $\lambda_1, \cdots, \lambda_n$ o velikosti individuálních skoků $\omega_1, \cdots, \omega_n$, $\langle\zeta, \xi\rangle$ je interval obsahující spektrum matice $\mathbf{A}$. Potom CG implicitně určuje v každém iteračním kroku $j$ po částech konstantní distribuční funkci $\omega^{(j)}(\lambda)$ $j$-bodové Gaussovy kvadratury

$$\int_\zeta^\xi f(\lambda)\, d\omega(\lambda) = \int_\zeta^\xi f(\lambda)\, d\omega^{(j)}(\lambda) + R_j(f), \tag{2}$$

$R_j(f)$ je zbytek Gaussovy kvadratury. Funkce $\omega^{(j)}(\lambda)$ (o $j$ schodech) optimálně aproximuje funkci $\omega(\lambda)$ (ve smyslu Gaussovy kvadratury). Rovnice (2) popisuje podstatu CG a je fundamentální pro pochopení fungování CG v konečné aritmetice.

**Ekvivalentní vyjádření Gaussovy kvadratury.** Uvažujme funkci $f(\lambda) \stackrel{\text{def}}{=} \frac{1}{\lambda}$. Nechť $C_n$ a $C_j$ jsou řetězové zlomky příslušné k jednotlivým Riemann-Stieltjesovým intergálům v (2). Potom

Gaussovovo kvardraturní pravidlo (2) nabývá tvaru (viz. např. [2])

$$C_n \;=\; C_j \;+\; \frac{\|x-x_j\|_A^2}{\|r_0\|^2}, \qquad C_n = \frac{\|x-x_0\|_A^2}{\|r_0\|^2}\,. \tag{3}$$

Zajímavé podoby vztahu (3) přenásobeného $\|r_0\|^2$ si povšiml Warnick [7], platí

$$r_0^T(x-x_0) = r_0^T(x_j-x_0) + \|x-x_j\|_A^2\,. \tag{4}$$

V [6] jsme ukázali další ekvivalentní vztah, odvozený algebraicky bez použití Gaussovy kvadratury

$$r_0^T(x-x_0) = r_0^T(x_j-x_0) + r_j^T(x_j-x_0) + \|x-x_j\|_A^2. \tag{5}$$

Jak vidíme, je (5) velmi podobné vztahu (4). Výraz $r_j^T(x_j-x_0)$, který je jakoby "navíc" v našem ekvivalentním vyjádření vztahu (3), bude mít v konečné aritmetice podstatný korekční efekt.

V neposlední řadě jsme si v [6] uvědomili jednu velmi podstatnou věc a tím jsme konečně "prohlédli" Gaussovo kvadraturní pravidlo a s ním i odhady založené jak na algebraickém tak na Gauss-kvadraturním pojetí. Gaussovo kvadraturní pravidlo není nic jiného, než vztah mezi druhou mocninou **A**-normy nulté a $j$-té chyby. Nutně tedy vztah uvedený v původní práci [5]

$$\|x-x_0\|_A^2 = \sum_{i=0}^{j-1} \gamma_i \|r_i\|^2 + \|x-x_j\|_A^2 \tag{6}$$

je další ekvivalentní formou k (3). Zatímco jsme v předchozích vztazích potřebovali k vyjádření hodnotu Gaussovy kvadratury buď složité vzorce (řetězové zlomky) nebo počítání dalších skalárních součinů (viz. (4) a (5)), ve vztahu (6) stačí sečíst čísla, která je nutné tak jako tak počítat v každém kroku CG.

## 2. Konstrukce odhadů A-normy chyby

Základní myšlenka pro konstrukci odhadů spočívá ve využití základního vztahu (3). Vyjádříme-li z (3) **A**- normu $j$-té chyby, máme

$$\|x-x_j\|_A^2 \;=\; \|r_0\|^2\,[C_n - C_j]\;. \tag{7}$$

Číslo $C_n$ je pro nás při počítání neznámé. Můžeme ho však aproximovat, počítáme-li dalších $d$ kroků metody CG, hodnotou $C_{j+d}$. Pokud chceme vědět, jak dobrá je naše aproximace, stačí uvažovat (7) pro iterační čísla $j$ a $j+d$ a oba vztahy odečíst. Dostáváme

$$\|x-x_j\|_A^2 = \|r_0\|^2\,[C_{j+d} - C_j] \;+\; \|x-x_{j+d}\|_A^2\,. \tag{8}$$

Pokud tedy platí $\|x-x_j\|_{\mathbf{A}}^2 \gg \|x-x_{j+d}\|_{\mathbf{A}}^2$, získáme na základě (8) dobrý (dolní) odhad pro $\|x-x_j\|_{\mathbf{A}}$ (všechny výrazy v (8) jsou kladné). Naprosto stejně lze postupovat i v případě ostatních ekvivalentních vztahů. Matematicky ekvivalentní odhady druhé mocniny **A**-normy $j$-té chyby příslušné k vztahům (3)–(6) (označme je postupně $\eta_{j,d}$, $\mu_{j,d}$, $\vartheta_{j,d}$ a $\nu_{j,d}$) mají tvar

$$\begin{aligned}
\eta_{j,d} &= \|r_0\|^2\,[C_{j+d} - C_j],\\
\mu_{j,d} &= r_0^T(x_{j+d}-x_j),\\
\vartheta_{j,d} &= r_0^T(x_{j+d}-x_j) - r_j^T(x_j-x_0) + r_{j+d}^T(x_{j+d}-x_0),
\end{aligned}$$

a

$$\nu_{j,d} = \sum_{i=j}^{j+d-1} \gamma_i \|r_i\|^2.$$

## 3. Aritmetika s konečnou přesností (FP - Finite Precision arithmetic)

**CG v aritmetice s konečnou přesností.** Přeneseme-li algoritmus CG do prostředí konečné aritmetiky, zjistíme, že ortogonalita mezi počítanými vektory se obvykle velmi rychle vytrácí, dochází dokonce k jejich lineární závislosti a následkem toho ke zpoždění konvergence a ke ztrátě finitnosti. Je nutné si uvědomit, že všechny formule a vztahy odvozené na základě Gaussovy kvadratury (a tedy i globální ortogonality) mohou přestat fungovat (nevydávat požadované informace). Ta totiž užívala k aproximaci integrálu

$$\int_{\zeta}^{\xi} \lambda^{-1}\, d\omega(\lambda) \tag{9}$$

implicitně kořenů ortogonálních polynomů. *Není vůbec zřejmé, zda hodnoty, které z formulí pro odhad dostáváme, skutečně aproximují hodnotu* $\|x - x_j\|_{\mathbf{A}}$, kde $x_j$ je aproximace spočtená CG v konečné aritmetice.

**Matematický model CG v konečné aritmetice.** Jak bylo ukázáno v [3] a [4], na CG v konečné aritmetice lze pohlížet jako na přesnou CG aplikovanou na systém $\hat{\mathbf{A}}\hat{x} = \hat{b}$, pro který je konvergence určena Riemann- Stieltjesovým integrálem

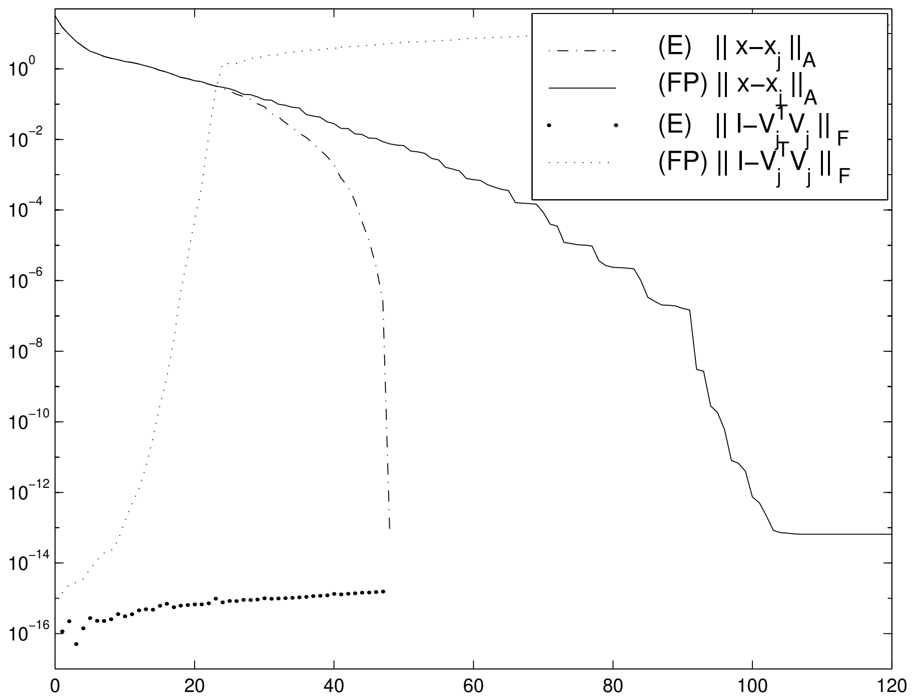
$$\int_{\zeta}^{\xi} \lambda^{-1}\, d\hat{\omega}(\lambda) \tag{10}$$

kde $\hat{\omega}(\lambda)$ vznikla z $\omega(\lambda)$ *rozmazáním* jednotlivých bodů růstu $\lambda_i$ do (pokud možno nekonečně) mnoha bodů růstu v blízkosti každého $\lambda_i$. Matice $\hat{\mathbf{A}}$ má o mnoho větší dimenzi než matice $\mathbf{A}$ a její vlastní čísla jsou nakupena v těsné blízkosti (v malých intervalech) okolo vlastních čísel matice $\mathbf{A}$. Celková výška skoku na okolí bodu $\lambda_i$ funkce $\hat{\omega}(\lambda)$ je stejná, jako výška skoku $\omega(\lambda)$ v bodě $\lambda_i$ a $\hat{\omega}(\lambda)$ velmi těsně aproximuje $\omega(\lambda)$. *CG v konečné aritmetice neztrácí dobré vlastnosti a lze ji modelovat pomocí přesné CG aplikované na modifikovaný systém.*

**Co odhady v konečné aritmetice počítají.** V [2] bylo vysvětleno, že odhad $\eta_{j,d}^{1/2}$, který byl odvozen na základě Gaussovy kvadratury pro integrál (9) počítá v konečné aritmetice hodnoty korespondující s Gaussovou kvadraturou pro integrál (10), tedy hodnoty příslušné k přesnému matematickému modelu CG v konečné aritmetice. Ostatní matematicky ekvivalentní odhady by měli počítat totéž, pokud však vztahy, na nichž jsou založeny, platí i v konečné aritmetice. To je nutné dokázat.

**Které odhady fungují v konečné aritmetice.** V [2] bylo ukázáno, že odhad založený na $\eta_{j,d}$ je funkční v konečné aritmetice pouze s jistými omezeními. Rozdíl $C_{j+d} - C_j$ je totiž nutné počítat tak, aby docházelo k vyrušení vlivu zaokrouhlovacích chyb. Tento problém byl chytře vyřešen v [1]. V práci [6] jsme ukázali následující fakta. Odhad založený na Warnickově vztahu (4) je silně závislý na globální ortogonalitě a tudíž nepoužitelný pro odhad $\mathbf{A}$-normy chyby. Náš vylepšený vztah (5) a s ním i $\vartheta_{j,d}$ jsme sice v [6] neanalyzovali, víme však, že je plně použitelný k odhadu až dokud $\mathbf{A}$-norma chyby nedosáhne hranice úměrné strojové přesnosti $\varepsilon$, počítáme-li rozdíly vektorů $x_{j+d} - x_j$ vhodným způsobem. Diskuse k tomuto vztahu včetně analýzy zaokrouhlovacích chyb bude provedena v mé disertační práci. V [6] jsme se místo toho věnovali zdá se nejjednoduššímu, nejstabilnějšímu a nejméně početně náročnému odhadu pomocí $\nu_{j,d}$ založeném na vztahu (6). Ukázali jsme, že informacím které poskytuje $(\nu_{j,d})^{1/2}$ je možno věřit až do okamžiku, kdy $\|x - x_j\|_{\mathbf{A}}$ dosáhne úrovně maximální přesnosti, t.j. hladiny úměrné strojovému epsilon a určitým konstantám, jež závisí na vstupních datech úlohy, tedy na vlastnostech matice $\mathbf{A}$, vektoru pravé strany $b$ a počátečním přiblížení $x_0$.

**O analýze formulí založených na algebraické manipulaci.** Analýza probíhá většinou následujím způsobem. Místo rekurentních vztahů metody CG předpokládáme porušené vztahy (s případným poruchovým vektorem, jehož norma je úměrná strojové přesnosti $\varepsilon$ a velikosti počítaných vektorů). Takto modifikované vztahy dosazujeme do vzorců a dostáváme většinou velmi komplikované formule. Potom přichází fáze, která by snad snesla označení “oddělení zrn od plev”. Jde o to pochopit vztah a vyloučit výrazy, které mají nepatrný vliv na jeho platnost (například vyšší mocniny $\varepsilon$). Zbydou výrazy, jež výrazně ovlivňují platnost vztahu v konečné aritmetice a na jejich základě můžeme odhadnout, do kdy vztah zůstává v platnosti a vydává potřebné informace.




**Obrázek 1:** CG v přesné (E) a konečné aritmetice (FP)

### 4. Numerické experimenty

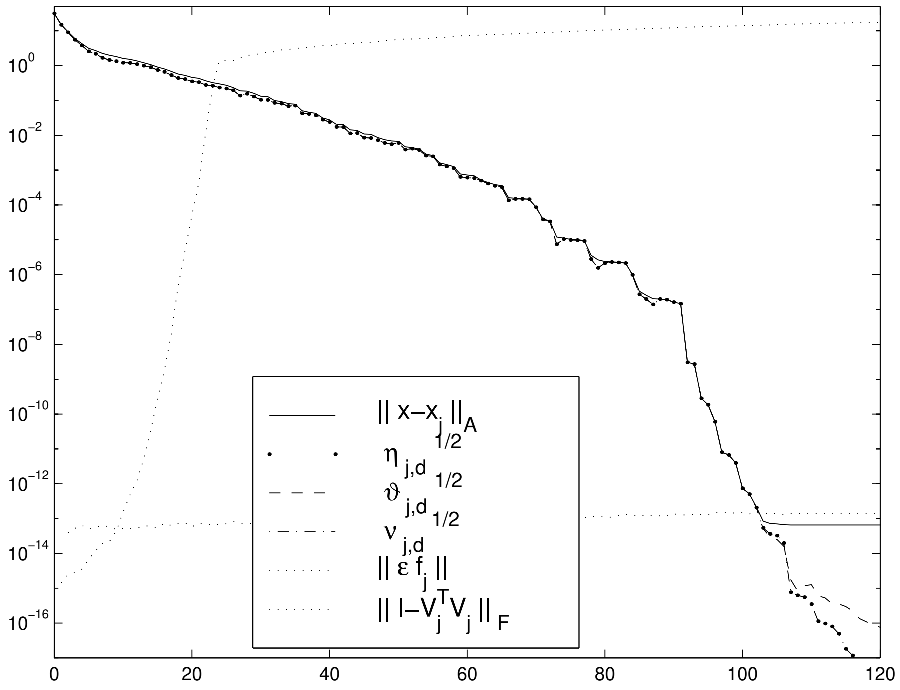
Pro naše numerické experimenty jsme si vybrali příklad, na kterém se silně projevují zaokrouhlovací chyby. Zvolili jsme matici $\mathbf{A} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^T$, kde $\mathbf{Q}$ je ortogonální matice, jež vznikla z (Matlab)QR-rozkladu náhodně generované matice (počítané Matlabovským příkazem `randn(n)`), a $\mathbf{\Lambda} = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$ je diagonální matice s vlastními čísly $\lambda_i$ počítanými podle předpisu

$$\lambda_i = \lambda_1 + \frac{i-1}{n-1}(\lambda_n - \lambda_1)\,\rho^{n-i}, \quad i = 2, \ldots, n-1. \tag{11}$$

Zvolili jsme $n = 48$, $d = 4$, $\lambda_1 = 0.1$, $\lambda_n = 1000$, $\rho = 0.9$, $x = (1, \ldots, 1)^T$, $b = \mathbf{A}x$ a $x_0 = (0, \ldots, 0)^T$. Pomocí $\|\varepsilon f_j\|$ jsme označili normu rozdílu skutečného a rekursivně počítaného residua, $\|\varepsilon f_j\| \overset{\text{def}}{=} \|(b - Ax_j) - r_j\|$. Experimenty byly provedeny za pomoci programu Matlab 5.1 na osobním počítači se strojovou přesností $\varepsilon \sim 1.1 \times 10^{-16}$.

Hodnoty počítané přesnou metodou CG jsme modelovali pomocí CG v konečné aritmetice s dvojnásobně reortogonalizovanými reziduovými vektory (viz. [4]) a budeme je značit pomocí (E). Hodnoty počítané v konečné aritmetice ponesou označení (FP). Na obrázku 1 vidíme, že $\mathbf{A}$-norma chyby (E)CG (čerchovaná čára) může být velmi odlišná od $\mathbf{A}$-normy chyby (FP)CG (normální čára). Ortogonalita mezi (FP) residuovými vektory (tečkovaná čára) měřená Frobeniovou normou $\|\mathbf{I} - \mathbf{V}_j^T\mathbf{V}_j\|_F$, kde $\mathbf{V}_j$ je matice o $j$ sloupcích složená z normovaných residových vektorů (jež by měli být vzájemně ortogonální), se vytrácí po pár iteracích. Ztráta ortogonality mezi residuovými vektory, jež jsou počítány CG algoritmem s dvojnásobně reortogonalizovanými residuovými vektory je zakreslena tečkami. Zůstává, jak je dobře vidět, na úrovni blízké strojovému epsilon.

Obrázek 2 ukazuje, že odhady $\eta_{j,d}{}^{1/2}$ (tečky) $\vartheta_{j,d}{}^{1/2}$ (čárkovaná čára) a $\nu_{j,d}{}^{1/2}$ (čerchovaná), které dávají velmi podobné a velmi dobré výsledky, fungují i za silného vlivu zaokrouhlovacích chyb na algoritmus CG v konečné aritmetice. Všechny čáry jsou v podstatě totožné dokud $\|x - x_j\|_A$ (plná čára) nedosáhne své maximální úrovně přesnosti. Ztráta ortogonality měřená $\|\mathbf{I} - \mathbf{V}_j^T\mathbf{V}_j\|_F$, je znázorněna příkře rostoucí tečkovanou čárou. Výraz $\|\varepsilon f_j\|$, který měří rozdíl mezi skutečným a rekursivním residuem (vodorovná tečkované čára) zůstává blízko úrovně úměrné strojové přesnosti po celou dobu výpočtu.




**Obrázek 2:** Aplikace odhadů v konečné aritmetice

### Závěry

Existující odhady **A**-normy chyby jsou matematicky ekvivalentní vztahu (6), který se objevil již v původní práci [5]. Každá aplikace nějakého odhadu v CG v konečné aritmetice musí být řádně ospravedlněna analýzou zaokrouhlovacích chyb. Jinak si nemůžeme být jisti, že odhad počítá hodnoty korespondující k přesnému matematickému modelu CG v konečné aritmetice a tím i hodnoty odhadující **A**-normu chyby v FP. Otevřenou a dosti důležitou otázkou v použití odhadů **A**-normy chyby zůstává adaptivní volba délky kroku $d$.

### References


[1] *Golub, G.H. and Meurant, G.,* Matrices, Moments and Quadrature II: How to Compute the Norm of the Error in Iterative Methods, BIT 37, pp. 687-705, 1997.

[2] *Golub, G.H. and Strakoš, Z.,* Estimates in Quadratic Formulas, Numerical Algorithms 8, pp. 241-268, 1994.

[3] *Greenbaum, A.,* Behavior of Slightly Perturbed Lanczos and Conjugate Gradient Recurrences, Lin. Alg. Appl. 113, pp. 7-63, 1989.

[4] *Greenbaum, A. and Strakoš, Z.,* Predicting the Behavior of Finite Precision Lanczos and Conjugate Gradient Computations, SIAM J. Matrix Anal. Appl. 13, pp. 121–137, 1992.

[5] *Hestenes, M.R. and Stiefel, E.*, Methods of conjugate gradients for solving linear systems. J. Res. Nat. Bureau Standarts 49, 409–435, 1952.

[6] *Strakoš, Z. and Tichý, P.*, On Error Estimation in the Conjugate Gradient Method and Why It Works In Finite Precision Computations, Submitted to ETNA, 25 p., June 2001.

[7] *Warnick, K.F.,* Nonincreasing Error Bound for the Biconjugate Gradient Method, Report, University of Illinois, 2000.

# Metody s lokálně omezeným krokem

*doktorand:*
MGR. CTIRAD MATONOHA
ÚI AVČR, Pod vodárenskou věží 2, Praha 8

*školitel:*
DOC. RNDR. JAN ZÍTKO, CSC.
Sokolovská 83, Praha 2

*konzultant:* ING. LADISLAV LUKŠAN, DRSC.
ÚI AVČR, Pod vodárenskou věží 2, Praha 8, ČR

obor studia:
M6 - vědecko-technické výpočty

**Abstrakt**

Metody s lokálně omezeným krokem jsou třídou nedávno vyvinutých algoritmů pro řešení optimalizačních problémů. Hledá se minimum kvadratické funkce vzhledem k určitému omezení, např. na sféře. Většina metod vede na řešení systému lineárních rovnic, což je neefektivní hlavně pro obsáhlé optimalizační problémy. V tomto článku ukážeme, že metoda s lokálně omezeným krokem s postupně generovanými Krylovovými podprostory má speciální strukturu, která nám dovolí řešit tento podproblém efektivně. Po konečném počtu kroků lze získat téměř optimální řešení, které splňuje podmínky optimalizace.

## 1. Základní optimalizační metoda

Nejprve definujeme základní optimalizační metodu, metodu s lokálně omezeným krokem a ukážeme některé vlastnosti lokálně omezeného kroku. Dále budeme generovat Krylovovy podprostory a hledat řešení splňující optimalizační podmínky.

Nechť je dána funkce $F(s) : \mathbb{R}^n \to \mathbb{R}$ v proměnné $s \in \mathbb{R}^n$ a hledáme její lokální minimum. Označme $g(s)$ její gradient a $\mathbf{G}(s)$ její Hessovu matici (matici druhých parciálních derivací). Spojitost druhých parciálních derivací implikuje symetrii matice $\mathbf{G}(s)$. Označme $\mathbf{A}$ matici, která aproximuje Hessovu matici $\mathbf{G}(s)$.

**Definice 1** *Bod $s_\star \in \mathbb{R}^n$ je lokálním minimem funkce $F : \mathbb{R}^n \to \mathbb{R}$, jestliže existuje číslo $\varepsilon > 0$ takové, že*

$$F(s_\star) \leq F(s) \quad \forall s \in \mathcal{B}(s_\star, \varepsilon) = \{s \in \mathbb{R}^n; \parallel s - s_\star \parallel < \varepsilon\}.$$

*Je-li navíc $F(s_\star) < F(s)$ pro $s \neq s_\star$, pak bod $s_\star \in \mathbb{R}^n$ je ostrým lokálním minimem funkce $F$.*

**Lemma 1** *Nechť bod $s_\star \in \mathbb{R}^n$ je lokálním minimem funkce $F : \mathbb{R}^n \to \mathbb{R}$ a nechť $F \in \mathcal{C}^1$ (spojitě diferencovatelná) na $\mathcal{B}(s_\star, \varepsilon)$. Pak platí $g(s_\star) = 0$. Jestliže navíc $F \in \mathcal{C}^2$ (dvakrát spojitě diferencovatelná) na $\mathcal{B}(s_\star, \varepsilon)$, pak platí $\mathbf{G}(s_\star) \geq 0$ (matice $\mathbf{G}(s_\star)$ je positivně semidefinitní).*

**Lemma 2** *Nechť $F : \mathbb{R}^n \to \mathbb{R}$, $F \in \mathcal{C}^2$ na $\mathcal{B}(s_\star, \varepsilon)$ a nechť platí $g(s_\star) = 0$ a $\mathbf{G}(s_\star) > 0$ (matice $\mathbf{G}(s_\star)$ je positivně definitní). Pak bod $s_\star \in \mathbb{R}^n$ je ostrým lokálním minimem funkce $F$.*

**Definice 2** *Základní optimalizační metoda je iterační proces, jehož výsledkem je posloupnost bodů $s_k \in \mathbb{R}^n$, $k \in \mathbb{N}$, taková, že*

$$s_{k+1} = s_k + \alpha_k x_k,$$

*kde směrový vektor $x_k$ se určuje na základě hodnot $s_j$, $F(s_j)$, $g(s_j)$, $\mathbf{G}(s_j)$, $1 \leq j \leq k$, a délka kroku $\alpha_k > 0$ se určuje na základě chování funkce $F : \mathbb{R}^n \to \mathbb{R}$ v okolí bodu $s_k \in \mathbb{R}^n$.*

Mezi nejjednodušší a nejznámější optimalizační metody patří metoda největšího spádu a Newtonova metoda. Protože mají spoustu nevýhodných vlastností, byly z nich vyvinuty důmyslnější a tudíž i složitější metody, metody spádových směrů a metody s lokálně omezeným krokem.

## 2. Metody s lokálně omezeným krokem

Při výkladu metod s lokálně omezeným krokem budeme používat následující označení:

$$\psi_k(x) = \frac{1}{2} x^T \mathbf{A}_k x + g_k^T x$$

pro kvadratickou funkci, která lokálně aproximuje rozdíl $F(s_k + x) - F(s_k)$;

$$\omega_k(x) = \frac{\mathbf{A}_k x + g_k}{\| g_k \|}$$

pro přesnost určení směrového vektoru;

$$\rho_k(x) = \frac{F(s_k + x) - F(s_k)}{\psi_k(x)}$$

pro podíl skutečného a předpověděného poklesu funkce $F : \mathbb{R}^n \to \mathbb{R}$. Funkci $\psi_k(x)$ dostaneme z Taylorova rozvoje funkce $F$ v bodě $s_k$ :

$$F(s_k + x) = F(s_k) + g_k^T x + \frac{1}{2} x^T \mathbf{G}_k x + \text{zbytek},$$

kde matice $\mathbf{A}_k$ v jistém smyslu aproximují Hessovu matici $\mathbf{G}_k$.

**Definice 3** *Základní optimalizační metoda $s_{k+1} = s_k + \alpha_k x_k$, $k \in \mathbb{N}$, je metodou s lokálně omezeným krokem, jestliže se směrové vektory $x_k \in \mathbb{R}^n$, $k \in \mathbb{N}$, určují tak, že platí*

$$\| x_k \| \leq \Delta_k, \tag{12}$$

$$\| x_k \| < \Delta_k \quad \Rightarrow \quad \| \omega_k(x_k) \| \leq \overline{\omega}_k \leq \overline{\omega}, \tag{13}$$

$$-\psi_k(x_k) \geq \underline{\sigma} \, \| g_k \| \min\left\{ \| x_k \|, \frac{\| g_k \|}{\| \mathbf{A}_k \|} \right\}, \tag{14}$$

*kde $0 < \underline{\sigma} < 1$, $0 \leq \overline{\omega} < 1$ a délky kroku $\alpha_k \geq 0$, $k \in \mathbb{N}$, se vybírají tak, že*

$$\rho_k(x_k) \leq 0 \quad \Rightarrow \quad \alpha_k = 0, \tag{15}$$

$$\rho_k(x_k) > 0 \quad \Rightarrow \quad \alpha_k = 1. \tag{16}$$

*Přitom posloupnost $\Delta_k > 0,\ k \in \mathbb{N}$, se konstruuje tak, že*

$$\rho_k(x_k) < \underline{\rho} \quad \Rightarrow \quad \underline{\beta} \parallel x_k \parallel \leq \Delta_{k+1} \leq \overline{\beta} \parallel x_k \parallel, \tag{17}$$

$$\rho_k(x_k) \geq \underline{\rho} \quad \Rightarrow \quad \Delta_k \leq \Delta_{k+1} \leq \overline{\gamma}\Delta_k, \tag{18}$$

*kde $0 < \underline{\beta} \leq \overline{\beta} < 1 < \overline{\gamma}$ a $0 < \underline{\rho} < 1$. Posloupnost $\mathbf{A}_k \neq 0$ se konstruuje tak, aby matice $\mathbf{A}_k$ byly stejnoměrně omezené, tj.*

$$\parallel \mathbf{A}_k \parallel \leq M \quad \forall k \in \mathbb{N}, \tag{19}$$

*kde konstanta $M < \infty$ nezávisí na $k \in \mathbb{N}$.*

Načrtneme algoritmus metody s lokálně omezeným krokem. Symbolem $\mathbb{R}_S^{n \times n}$ označíme prostor symetrických matic řádu $n \times n$.

**Algoritmus 1** *Metoda s lokálně omezeným krokem.*
*Zvolíme $s_0 \in \mathbb{R}^n,\ 0 \neq \mathbf{A}_0 \in \mathbb{R}_S^{n \times n},\ \Delta_0 > 0,\ \varepsilon > 0$, spočítáme $F(s_0)$ a položíme $k = 0$.*

1. *Vypočítáme gradient $g(s_0)$. Je-li $\parallel g(s_0) \parallel < \varepsilon$, pak* **STOP**.
2. *Určíme vektor $x_k$ tak, aby byly splněny podmínky (12) až (14) a spočítáme $\psi_k(x_k)$.*
3. *Položíme $s_k^\dagger = s_k + x_k$ a vypočítáme hodnoty $F(s_k^\dagger)$ a $\rho_k(x_k) = \frac{F(s_k^\dagger) - F(s_k)}{\psi_k(x_k)}$.*
4. *Je-li $\rho_k(x_k) < \underline{\rho}$, pak určíme $\Delta_{k+1}$ tak, aby byla splněna podmínka (17).*
   *Je-li $\rho_k(x_k) \geq \underline{\rho}$, pak určíme $\Delta_{k+1}$ tak, aby byla splněna podmínka (18).*
5. *Je-li $\rho_k(x_k) \leq 0$, přejdeme na krok 2.*
   *Je-li $\rho_k(x_k) > 0$, určíme matici $\mathbf{A}_{k+1} \neq 0$ tak, aby byla splněna podmínka (19), položíme $s_{k+1} = s_k^\dagger,\ F(s_{k+1}) = F(s_k^\dagger),\ k = k + 1$ a přejdeme na krok 1.*

**Definice 4** *Metody s optimálním lokálně omezeným krokem používají směrové vektory $x_k,\ k \in \mathbb{N}$, takové, že*

$$x_k = \arg \min_{\|x\| \leq \Delta_k} \psi_k(x), \tag{20}$$

*přičemž bereme $\parallel x_k \parallel = \Delta_k$, pokud toto minimum není jediné.*

**Lemma 3** *Směrový vektor určený podle (20) vyhovuje podmínkám (12) až (14) pro $\overline{\omega} = 0$ a $\underline{\sigma} = \frac{1}{2}$.*

**Věta 1** *Vektor $x_k \in \mathbb{R}^n$ je řešením úlohy (20) ve smyslu definice 4 právě tehdy, když $\parallel x_k \parallel \leq \Delta_k$ a jestliže existuje číslo $\xi_k \geq 0$ takové, že matice $\mathbf{A}_k + \xi_k \mathbf{I}$ je positivně semidefinitní a platí*

$$(\mathbf{A}_k + \xi_k \mathbf{I}) x_k + g_k = 0, \quad (\parallel x_k \parallel - \Delta_k)\, \xi_k = 0.$$

Tato věta je speciálním případem tzv. Khun-Thuckerovy věty ([5]).

V další části se budeme zabývat podproblémem problému hledání minima funkce $F$, a sice určením optimálního lokálně omezeného kroku $x_k$. Protože k tomu použijeme iterační proces $\{x_{k,i}\}_{i \in \mathbb{N}}$, označíme optimální lokálně omezený krok v $k$-tém kroku jako $x_{k,\star}$. Protože se jedná o podproblém a $k$ je pevné, vynecháme pro jednoduchost index $k$. Hledáme tedy řešení podproblému

$$\min_{x \in \mathbb{R}^n} \psi(x) \quad \text{vzhledem k} \quad \parallel x \parallel \leq \Delta, \qquad \text{kde} \qquad \psi(x) = \frac{1}{2} x^T \mathbf{A} x + g^T x, \tag{21}$$

$\mathbf{A}$ je symetrická matice řádu $n \times n$, $g$ je $n$-dimenzionální vektor, $\Delta$ je dané kladné číslo a norma je Euklidovská. Pro řešení $x_\star$ platí věta 1. Z ní plyne, že řešit problém (21) je ekvivalentní nalezení správného $\xi_\star$ (tzv. Levenbergův-Marquardtův parametr) a řešení lineárního systému

$$(\mathbf{A} + \xi \mathbf{I})x = -g \tag{22}$$

pro $\xi = \xi_\star$. Jestliže $\| x_\star \| < \Delta$, pak je $\mathbf{A}$ positivně semidefinitní a $\mathbf{A}x_\star = -g$. To znamená, že v bodě $x_\star$ nabývá funkce $\psi(x)$ globálního minima přes $\mathbb{R}^n$. Z věty 1 plyne následující lemma.

**Lemma 4** *Nechť $\xi \in \mathbb{R}$, $x \in \mathbb{R}^n$ splňují (22), kde $\mathbf{A} + \xi\mathbf{I}$ je positivně semidefinitní. Pak platí následující tvrzení. Je-li $\xi = 0$ a $\| x \| < \Delta$, pak $x$ řeší (21). Je-li $\xi \geq 0$ a $\| x \| = \Delta$, pak $x$ řeší (21). Je-li navíc $\mathbf{A} + \xi\mathbf{I}$ positivně definitní, pak $x$ je jediné řešení (21).*

**Lemma 5** *Pro řešení $x_\star$ platí* $\quad g^T x_\star \leq 0 \quad$ *a* $\quad \psi(x_\star) \leq 0.$

Předpokládejme, že (21) má řešení $x_\star$ na hranici množiny $\{x; \| x \| \leq \Delta\}$ s odpovídajícím $\xi_\star \geq 0$. Tato $x_\star$ a $\xi_\star$ jsou řešením nelineární rovnice (viz věta 1)

$$\| x \| = \| (\mathbf{A} + \xi\mathbf{I})^\dagger g \| = \Delta, \quad \xi \geq 0$$

a protože $\mathbf{A} + \xi\mathbf{I}$ je positivně semidefinitní, je

$$\xi_\star \in \ \langle -\lambda_1, \infty),$$

kde $\lambda_1$ je nejmenší vlastní číslo matice $\mathbf{A}$ a symbol $^\dagger$ značí Moore-Penroseovu zobecněnou inverzi.

Není-li $g$ kolmé na prostor

$$\mathcal{S}_1 = \{z; \mathbf{A}z = \lambda_1 z, z \neq 0\},$$

pak je $\xi_\star \in (-\lambda_1, \infty)$. Zajímavá je situace, kdy je $g \perp \mathcal{S}_1$. V tomto případě může existovat opět $\xi_\star \in (-\lambda_1, \infty)$, ale může být i $\xi_\star = -\lambda_1$, jak ukáže následující lemma. Nejprve jedna definice.

**Definice 5** *Jestliže je vektor $g$ kolmý na prostor $\mathcal{S}_1$, pak řekneme, že pro problém (21) nastal „singulární případ“.*

**Lemma 6** *Pro řešení $x_\star$ problému (21) platí:*

1. $\{ \ \xi_\star = -\lambda_1 \quad$ *nebo* $\quad x_\star \perp \mathcal{S}_1 \ \} \quad \Rightarrow \quad g \perp \mathcal{S}_1,$
2. $g \perp \mathcal{S}_1 \quad \Rightarrow \quad \{ \ \xi_\star = -\lambda_1 \quad$ *nebo* $\quad \xi_\star \neq -\lambda_1 \ \ \& \ \ x_\star \perp \mathcal{S}_1 \ \}.$

Charakteristické pro „singulární případ“ je, že jestliže $\xi_\star = -\lambda_1$, pak je matice $\mathbf{A} + \xi_\star\mathbf{I}$ singulární, rovnice (22) nemá jediné řešení a norma vektoru $x = -(\mathbf{A}+\xi_\star\mathbf{I})^\dagger g$ je menší než $\Delta$. V tomto případě lze řešení (21) dostat nalezením libovolného řešení rovnice

$$(\mathbf{A} - \lambda_1\mathbf{I})x = -g,$$

kde $\| x \| \leq \Delta$, a určením vlastního vektoru $z \in \mathcal{S}_1$. Pak

$$(\mathbf{A} - \lambda_1\mathbf{I}) \cdot (x + \kappa z) = -g \ \text{ a } \ \| x + \kappa z \| = \Delta$$

pro nějaké $\kappa$. Nyní, protože $\mathbf{A} - \lambda_1\mathbf{I}$ je positivně semidefinitní matice, podle lemmatu 4 plyne, že $x_\star = x + \kappa z$ řeší (21).

Navíc platí $\ 0 \leq \xi_\star = -\lambda_1 \ \Rightarrow \ \lambda_1 \leq 0,\ $ což implikuje, že matice $\mathbf{A}$ není positivně definitní.

## 3. Metoda sdružených gradientů a Lanczosova metoda

Mezi metody, které hledají řešení problému (21), patří např. Newtonova metoda ([1], [4], [6]) nebo metody, které celý problém převedou na problém vlastních čísel ([3], [7]). Všechny tyto metody hledají řešení na celém prostoru $\mathbb{R}^n$. Jako Cauchyho bod je definováno řešení problému

$$\min_{x \in sp\{g\}} \psi(x) \equiv \frac{1}{2} x^T \mathbf{A} x + g^T x, \quad \| x \| \leq \Delta, \tag{23}$$

tedy minimum funkce $\psi$ je z 1-dimensionálního podprostoru $sp\{g\}$. Stejně tak lze uvažovat minimum $\psi$ na 2-dimensionálním podprostoru, atd. V těchto případech lze řešení snadno nalézt, protože tyto podprostory jsou malé. V obecném problému (21) je však hledaný prostor $\mathbb{R}^n$ velký. Budeme proto uvažovat řešení $x_k$ kompromisního problému

$$\min_{x \in K_k} \psi(x), \quad \| x \| \leq \Delta, \tag{24}$$

na Krylovově podprostoru $K_k$ generovaném vektorem $g$ a maticí $\mathbf{A}$ :

$$K_k = sp\{g, \mathbf{A}g, \mathbf{A}^2 g, ..., \mathbf{A}^k g\}. \tag{25}$$

Nejprve najdeme bázi $K_k$ a řešením problému (24) bude lineární kombinace báze.

Steihaug ([8]) byl první, kdo použil myšlenku sdružených gradientů. Vylepšení této myšlenky provedl Gould ([1], [2]). Pro vygenerování báze použijeme metodu sdružených gradientů a v případě jejího selhání - $(p_j, \mathbf{A}p_j) = 0$ - přejdeme na Lanczosovu metodu.

**Algoritmus 2** *Metoda sdružených gradientů (CGM).*
*Položíme* $r_0 = g,\ p_0 = -r_0$. *Pro* $j = 0, 1, ..., k-1$ *provedeme:*

1. $\alpha_j = \frac{(r_j, r_j)}{(p_j, \mathbf{A}p_j)}$.
2. $r_{j+1} = r_j + \alpha_j \mathbf{A} p_j$.
3. $\beta_j = \frac{(r_{j+1}, r_{j+1})}{(r_j, r_j)}$.
4. $p_{j+1} = -r_{j+1} + \beta_j p_j$.

**Algoritmus 3** *Lanczosova metoda (LM).*
*Položíme* $t_0 = g,\ q_{-1} = 0$. *Pro* $j = 0, 1, ..., k$ *provedeme:*

1. $\gamma_j = \sqrt{(t_j, t_j)}$.
2. $q_j = \frac{1}{\gamma_j} t_j$.
3. $\delta_j = (q_j, \mathbf{A}q_j)$.
4. $t_{j+1} = \mathbf{A}q_j - \delta_j q_j - \gamma_j q_{j-1}$.

Metoda CGM generuje $\mathbf{A}$-ortogonální bázi $sp\{p_0, p_1, .., p_k\}$, zatímco metoda LM generuje ortonormální bázi $sp\{q_0, q_1, .., q_k\}$. Lanczosovu iteraci lze psát v maticovém tvaru

$$\mathbf{A}\mathbf{Q}_k = \mathbf{Q}_k \mathbf{T}_k + \gamma_{k+1} q_{k+1} e_{k+1}^T, \tag{26}$$

kde $\mathbf{Q}_k = (q_0, ..., q_k)$, $\mathbf{Q}_k^T \mathbf{Q}_k = \mathbf{I}_{k+1}$ a $\mathbf{T}_k$ je 3-diagonální matice

$$\mathbf{T}_k = \begin{pmatrix} \delta_0 & \gamma_1 & & \\ \gamma_1 & \delta_1 & \ddots & \\ & \ddots & \ddots & \gamma_k \\ & & \gamma_k & \delta_k \end{pmatrix}.$$

Dále platí vztahy

$$\mathbf{Q}_k^T \mathbf{A} \mathbf{Q}_k = \mathbf{T}_k, \quad \mathbf{Q}_k^T g = \gamma_0 e_1 \quad \text{a} \quad g = \gamma_0 q_0. \tag{27}$$

Protože residua $\{r_k\}$ jsou ortogonální, můžeme Lanczosovy vektory získat z iterací sdružených gradientů jako $q_k = \frac{1}{\|r_k\|} r_k$ a matici $\mathbf{T}_k$ lze vyjádřit následujícím způsobem:

$$\mathbf{T}_k = \begin{pmatrix} \frac{1}{\alpha_0} & \frac{\sqrt{\beta_0}}{|\alpha_0|} & & & \\ \frac{\sqrt{\beta_0}}{|\alpha_0|} & \frac{1}{\alpha_1} + \frac{\beta_0}{\alpha_0} & \frac{\sqrt{\beta_1}}{|\alpha_1|} & & \\ & \frac{\sqrt{\beta_1}}{|\alpha_1|} & \ddots & \ddots & \\ & & \ddots & \ddots & \frac{\sqrt{\beta_{k-1}}}{|\alpha_{k-1}|} \\ & & & \frac{\sqrt{\beta_{k-1}}}{|\alpha_{k-1}|} & \frac{1}{\alpha_k} + \frac{\beta_{k-1}}{\alpha_{k-1}} \end{pmatrix}.$$

Nyní se zaměříme na problém (24) a budeme hledat řešení $x_k$ z Krylovova podprostoru $K_k$, tedy

$$x_k = \mathbf{Q}_k h_k, \tag{28}$$

kde $x_k$ řeší problém

$$\min_{x \in K_k} \{\frac{1}{2} x^T \mathbf{A} x + g^T x\} \quad \text{vzhledem k} \quad \| x \| \leq \Delta. \tag{29}$$

Ze vztahů (27) plyne po dosazení, že vektor $h_k$ řeší problém

$$\min_{h \in \mathbb{R}^{k+1}} \{\frac{1}{2} h^T \mathbf{T}_k h + (\gamma_0 e_1)^T h\} \quad \text{vzhledem k} \quad \| h \| \leq \Delta. \tag{30}$$

Základní věta 1 ukazuje, že platí

$$(\mathbf{T}_k + \xi_k \mathbf{I}_{k+1}) h_k = -\gamma_0 e_1, \tag{31}$$

kde matice $\mathbf{T}_k + \xi_k \mathbf{I}_{k+1}$ je positivně semidefinitní, dále $\xi_k \geq 0$ a $\xi_k(\| h_k \| - \Delta) = 0$. Odtud a ze vztahů (27) plyne, že

$$\mathbf{Q}_k^T (\mathbf{A} + \xi_k \mathbf{I}_n) x_k = \mathbf{Q}_k^T (\mathbf{A} + \xi_k \mathbf{I}_n) \mathbf{Q}_k h_k = (\mathbf{T}_k + \xi_k \mathbf{I}_{k+1}) h_k = -\gamma_0 e_1 = -\mathbf{Q}_k^T g$$

a dále, že

$$\xi_k(\| x_k \| - \Delta) = 0.$$

Vidíme, že $x_k$ je Galerkinova aproximace k $x_\star$ z prostoru generovaného maticí $\mathbf{Q}_k$. Ptáme se tedy, jak dobrá tato aproximace je, konkrétně jaká je chyba $(\mathbf{A} + \xi_k \mathbf{I}_n) x_k + g$.

**Věta 2** *Platí:*

$$(\mathbf{A} + \xi_k \mathbf{I}_n) x_k + g = \gamma_{k+1} \cdot e_{k+1}^T h_k \cdot q_{k+1}$$

*a tedy*

$$\| (\mathbf{A} + \xi_k \mathbf{I}_n) x_k + g \| = \gamma_{k+1} \cdot |e_{k+1}^T h_k|. \tag{32}$$

Můžeme tedy změřit chybu, známe-li číslo $\gamma_{k+1}$ a poslední složku vektoru $h_k$ a vůbec nepotřebujeme iterace $x_k$ ani matici $\mathbf{Q}_k$. Chyba bude malá, bude-li malé buď $\gamma_{k+1}$ nebo poslední složka $h_k$.

Dále budeme uvažovat problém (30). Začneme jednoduchou definicí.

**Definice 6** *Řekneme, že symetrická 3-diagonální matice je rozložitelná, jestliže alespoň jeden prvek mimo diagonálu je nulový. V opačném případě řekneme, že je nerozložitelná.*

**Lemma 7** *Nechť 3-diagonální matice* $\mathbf{T}$ *je nerozložitelná a* $v$ *je vlastní vektor matice* $\mathbf{T}$*. Pak první složka* $v$ *je nenulová.*

**Věta 3** *Nechť matice* $\mathbf{T}_k$ *je nerozložitelná. Pak singulární případ nemůže nastat pro problém (30).*

**Důsledek 1** *Nechť matice* $\mathbf{T}_{n-1}$ *je nerozložitelná. Pak singulární případ nemůže nastat pro původní problém (21).*

Tedy, nastane-li singulární případ pro problém (21), pak má Lanczosova 3-diagonální matice $\mathbf{T}_{n-1}$ alespoň jeden mimodiagonální prvek nulový. Následující věta stanovuje, kdy má rovnice (33) jediné řešení $h_k \in \mathbb{R}^{k+1}$.

**Věta 4** *Nechť* $\mathbf{T}_k$ *je nerozložitelná,* $h_k$ *a* $\xi_k$ *splňují*

$$(\mathbf{T}_k + \xi_k \mathbf{I}_{k+1}) h_k = -\gamma_0 e_1 \tag{33}$$

*a nechť* $\mathbf{T}_k + \xi_k \mathbf{I}_{k+1}$ *je positivně semidefinitní matice. Pak je positivně definitní.*

**Věta 5** *Nechť* $(e_{k+1}, h_k) = 0$*. Pak je matice* $\mathbf{T}_k$ *rozložitelná.*

**Věta 6** *Nechť pro problém (21) nenastane „singulární případ“ a nechť* $\gamma_{k+1} = 0$*. Pak* $x_k$ *řeší (21).*

Žádané řešení tedy dostaneme z prvního nerozložitelného bloku Lanczosovy 3-diagonální matice $\mathbf{T}_k$. Zbývá uvažovat „singulární případ“. Ten může nastat jen když je $\mathbf{T}_k$ rozložitelná. Nechť má tedy $\mathbf{T}_k$ následující $l$-blokový tvar

$$\mathbf{T}_k = \begin{pmatrix} \mathbf{T}_{k_1} & & & \\ & \mathbf{T}_{k_2} & & \\ & & \ddots & \\ & & & \mathbf{T}_{k_l} \end{pmatrix},$$

kde poslední blok $\mathbf{T}_{k_l}$ je první, který vede na nejmenší vlastní číslo $\lambda_1$ matice $\mathbf{A}$. Pak lze uvažovat dva případy.

**Věta 7** *Nechť pro problém (21) nastane „singulární případ“ a* $\mathbf{T}_k$ *má blokový tvar uvedený výše. Pak*

1. *Je-li* $-\lambda_1 \le \xi_{k_1}$*, pak řešení (21) je dáno takto:*

   $$x_k = \mathbf{Q}_{k_1} h_{k_1},$$

   *kde* $\mathbf{Q}_{k_1} = (q_0, ..., q_{k_1}) \in \mathbb{R}^{n \times (k_1+1)}$ *a* $h_{k_1} \in \mathbb{R}^{k_1+1}$ *je řešením positivně definitního systému* $(\mathbf{T}_{k_1} + \xi_{k_1} \mathbf{I}_{k_1+1}) h_{k_1} = -\gamma_0 e_1$.

2. *Je-li* $-\lambda_1 > \xi_{k_1}$*, pak řešení (21) je dáno takto:*

   $$x_k = \mathbf{Q}_k h_k, \quad \textit{kde} \quad h_k = (h, 0, ..., 0, \kappa v^T)^T,$$

   $h$ *je řešení regulárního 3-diagonálního systému* $(\mathbf{T}_{k_1} - \lambda_1 \mathbf{I}_{k_1+1}) h = -\gamma_0 e_1$*,* $v$ *je vlastní vektor podmatice* $\mathbf{T}_{k_l}$ *odpovídající nejmenšímu vlastnímu číslu* $\lambda_1$ *a* $\kappa$ *je zvoleno tak, že* $\| h_{k_1} \|^2 + \kappa^2 \| v \|^2 = \Delta^2$.

Pro řešení problému věty 7 však potřebujeme znát nejmenší vlastní číslo matice $\mathbf{A}$. Tento tvar řešení je proto užitečný pouze z teoretického hlediska.

## 4. Algoritmus

Nyní načrtneme algoritmus této metody. Ke generaci Lanczosových vektorů použijeme metodu sdružených gradientů (algoritmus 2). Jakmile bude skalární součin $(p_k, \mathbf{A}p_k)$ malý, přejdeme k Lanczosově metodě (algoritmus 3). Zavedeme parametr $INT$, který bude určovat, zde je řešení uvnitř oblasti nebo na hranici.

**Algoritmus 4** *Položíme*

$$x_0 = (0, ..., 0)^T,\ r_0 = g,\ \gamma_0 = \parallel g \parallel,\ q_0 = \frac{r_0}{\parallel r_0 \parallel},\ p_0 = -r_0,\ INT = \text{„}true\text{“},\ k = 0.$$

1. *Je-li* $|(p_k, \mathbf{A}p_k)| \leq \varepsilon$, *pak* $INT =$ *„false“ a jdeme na krok 10.*
2. *Položíme* $\alpha_k = \frac{(r_k, r_k)}{(p_k, \mathbf{A}p_k)}$.
3. *Aktualizujeme* $\mathbf{T}_k$ *z* $\mathbf{T}_{k-1}$ *přidáním řádku a sloupce.*
4. *Je-li* $INT =$ *„true“, ale* $\left\{\alpha_k \leq 0 \ \ nebo\ \parallel x_k + \alpha_k p_k \parallel \geq \Delta\right\}$, *pak* $INT =$ *„false“.*
5. *Je-li* $INT =$ *„true“, pak* $x_{k+1} = x_k + \alpha_k p_k$; *jinak řešíme 3-diag. problém (30) pro* $h_k$.
6. *Položíme* $r_{k+1} = r_k + \alpha_k \mathbf{A}p_k, \quad q_{k+1} = \frac{r_{k+1}}{\|r_{k+1}\|}$.
7. *Spočítáme* $\beta_k = \frac{(r_{k+1}, r_{k+1})}{(r_k, r_k)}$.
8. *Je-li* $INT =$ *„true“, pak: Je-li* $\parallel r_{k+1} \parallel \leq \varepsilon$, *pak* **STOP**, $x_\star = x_{k+1}$.
   *Je-li* $INT =$ *„false“, pak: Je-li* $\gamma_{k+1} \cdot |(e_{k+1}, h_k)| \leq \varepsilon$, *pak* **STOP**, $x_\star = \mathbf{Q}_k h_k$.
9. *Položíme* $p_{k+1} = -r_{k+1} + \beta_k p_k, \quad k := k+1$, *a jdeme na krok 1.*
10. *Položíme* $\delta_k = (q_k, \mathbf{A}q_k)$.
11. *Řešíme 3-diagonální problém (30) pro* $h_k$.
12. *Položíme* $t_{k+1} = \mathbf{A}q_k - \delta_k q_k - \gamma_k q_{k-1}$, *kde* $q_{-1} = 0$.
13. *Spočítáme* $\gamma_{k+1} = \parallel t_{k+1} \parallel$.
14. *Je-li* $\gamma_{k+1} \cdot |(e_{k+1}, h_k)| \leq \varepsilon$, *pak* **STOP**, $x_\star = \mathbf{Q}_k h_k$.
15. *Položíme* $q_{k+1} = \frac{t_{k+1}}{\gamma_{k+1}}, \quad k := k+1$ *a jdeme na krok 10.*

Nakonec zbývá uvažovat problém (30) pro $h_k$. Aplikujeme Newtonovu metodu na rovnici

$$\phi(\xi) \equiv \frac{1}{\Delta} - \frac{1}{\parallel h_k(\xi) \parallel} = 0, \quad \text{kde} \quad (\mathbf{T}_k + \xi\, \mathbf{I}_{k+1}) h_k(\xi) = -\gamma_0 e_1,$$

např. [1], [4], [6]. Protože matice $\mathbf{T}_k$ je třídiagonální, jedná se o jednoduchý problém.

## References


[1] A.R.Conn, N.I.M.Gould, P.L.Toint: Trust-region methods, SIAM 2000

[2] N.I.M.Gould, S.Lucidi, M.Roma, P.L.Toint: Solving the trust-region subproblem using the Lanczos method, June 11, 1997

[3] W.W.Hager: Minimizing a quadratic over a sphere, May 4, 1999

[4] L.Lukšan: Metody s proměnnou metrikou, Academia Praha 1990

[5] S.Míka: Matematická optimalizace, Plzeň 1997

[6] J.J.Moré, D.C.Sorensen: Computing a trust region step, December 1981

[7] M.Rojas, S.A.Santos, D.C.Sorensen : A new matrix-free algorithm for the large-scale trust-region subproblem, September 1999

[8] T.Steihaug: The conjugate gradient method and trust regions in large scale optimization, June 18, 1982

# Učení neuronových sítí typu RBF

*doktorand:*
PETRA KUDOVÁ
Ústav informatiky AV ČR
Pod vodárenskou věží 2,
18 207 Praha 8
petra@cs.cas.cz

*školitel:*
ROMAN NERUDA
Ústav informatiky AV ČR
Pod vodárenskou věží 2,
18 207 Praha 8
roman@cs.cas.cz

obor studia:
Teoretická informatika

**Abstrakt**

Tato práce stručně popisuje model RBF sítě a možnosti jejího učení. Efektivita a použitelnost popsaných algoritmů je doložena výsledky experimentů.

## 1. Úvod

Neuronové sítě typu RBF (krátce RBF sítě) patří mezi nejmladší modely neuronových sítí a jsou alternativou klasických modelů, jako jsou vícevrstvé perceptronové sítě.

Motivaci k jejich studiu najdeme v numerické matematice, konkrétně ve studiu interpolací a aproximací dat. Řešení aproximačního problému se většinou hledá jako lineární kombinace bazických funkcí v nějakém konkrétním tvaru, například jako kombinace polynomů. V 80. letech se v této souvislosti dostávají do popředí tzv. *radiální bazické funkce* a následně se objevuje model neuronové sítě typu RBF.

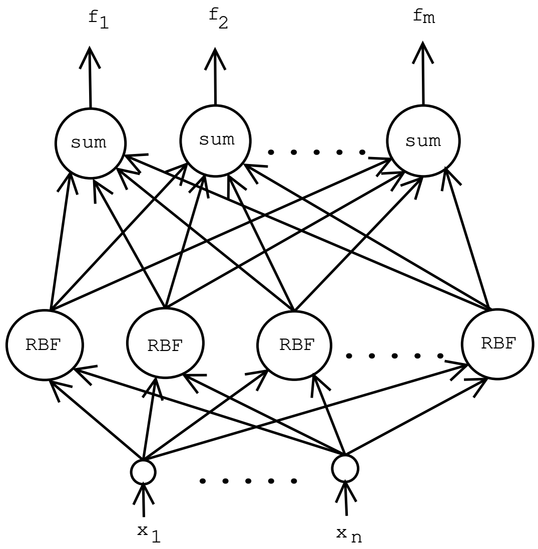
V následující kapitole tento model popíšeme, v kapitole 3 se seznámíme s postupy při učení této sítě, v kapitole 4 nastíníme implementaci učení v systému Bang2 a nakonec kapitola 5 přináší přehled výsledků provedených experimentů.

## 2. Neuronová síť typu RBF

Neuronová síť typu RBF (viz obr. 3) je dopředná neuronová síť s jednou skrytou vrstvou a lineární výstupní vrstvou. Skrytá vrstva je tvořena RBF jednotkami realizujícími funkci

$$y(\vec{x}) = \varphi\left(\frac{\parallel \vec{x} - \vec{c} \parallel_C}{b}\right), \tag{34}$$

kde $\vec{x}$ je vektor vstupů, $\varphi$ nějaká radiální funkce, parametr $\vec{c}$ střed RBF jednotky, parametr $b$ šířka a $\| \cdot \|_C$ označuje váženou normu ($\| \vec{x} \|_c^2 = (\mathbf{C}\vec{x})^T (\mathbf{C}\vec{x}) = \vec{x}^T \mathbf{C}^T \mathbf{C}\vec{x}$).




**Obrázek 3:** Neuronová síť typu RBF

Lineární výstupní vrstva počítá vážený součet výstupů RBF jednotek

$$f_s(\vec{x}) = \sum_{j=1}^{h} w_{js} \varphi \left( \frac{\| \vec{x} - \vec{c_j} \|_{C_j}}{b_j} \right), \tag{35}$$

kde $f_s$ je výstup $s$-té výstupní jednotky, $h$ je počet RBF jednotek a $w_{js}$ je váha spoje z $j$-té RBF jednotky do $s$-té výstupní jednotky.

V praxi nejpoužívanější radiální funkce je Gaussova funkce, RBF jednotky často nemají parametr šířka nebo používají euklidovskou normu (odpovídá vážené normě, kde $\mathbf{C}$ je jednotková matice).

## 3. Učení RBF sítě

Pro učení RBF sítí existuje řada algoritmů a metod. Nyní se stručně seznámíme s myšlenkami tří hlavních přístupů k tomuto problému, konkrétně s *gradientním učením*, *třífázovým učením* a *genetickým učením*.

### 3.1. Gradientní učení

Gradientní učení je inspirováno algoritmem *Back Propagation* pro vícevrstvé perceptronové sítě. Stejně jako u algoritmu Back Propagation minimalizujeme chybovou funkci sítě (36) gradientní metodou.

$$E = \frac{1}{2} \sum_{t=1}^{k} \| \vec{d}^{(t)} - \vec{f}^{(t)} \|^2 = \frac{1}{2} \sum_{t=1}^{k} \sum_{s=1}^{m} (d_s^{(t)} - f_s^{(t)})^2, \tag{36}$$

kde $k$ je počet tréninkových vzorů, $f_s^{(t)}$ označuje výstup sítě po předložení $t$-tého tréninkového vzoru a $\vec{d}^{(t)}$ požadovaný výstup sítě.

Protože má RBF síť pouze jednu skrytou vrstvu, nedochází při výpočtu derivací chybové funkce k zpětnému šíření chyby a výpočet je jednodušší než u vícevrstvých perceptronových sítí.

### 3.2. Třífázové učení

Parametry RBF sítě můžeme rozdělit podle charakteru do tří skupin. Do první skupiny patří středy RBF jednotek, které určují rozmístění RBF jednotek po vstupním prostoru. Druhá skupina obsahuje všechny ostatní parametry RBF jednotky (šířky, matice vážených norem), pokud jednotka nějaké má. Poslední skupinu tvoří váhy mezi skrytou a výstupní vrstvou.

Samotné učení rozdělíme do tří fází, v každé fázi učíme jednu skupinu parametrů metodou odpovídající významu této skupiny.

Úkolem **první fáze** je rozmístit středy RBF jednotek po vstupním prostoru tak, aby co nejlépe aproximovaly hustotu rozložení tréninkových vzorů. Vyžívají se různé heuristiky (rovnoměrné rozmístění, náhodné vzorky z tréninkové množiny) nebo metody vektorové kvantizace minimalizující funkci

$$E_{VQ} = \frac{1}{k} \sum_{t=1}^{k} \| \vec{x}^t - \vec{c}_c{}^2 \|, \quad c = \mathrm{argmin}_{i=1,..h} \{ \| \vec{x}^t - \vec{c}_i \| \}. \tag{37}$$

**Druhá fáze** hledá optimální hodnoty pro šířky a matice norem tak, aby oblasti působnosti RBF jednotek pokrývaly celý vstupní prostor a současně se příliš nepřekrývaly. Problém lze řešit gradientní minimalizací chybové funkce

$$E(b_1, \ldots, b_h; \mathbf{C}_1, \ldots, \mathbf{C}_h) = \frac{1}{2} \sum_{r=1}^{h} \left[ \sum_{s=1}^{h} \varphi \left( \frac{\| \vec{c}_s - \vec{c}_r \|_{C_r}}{b_r} \right) \left( \frac{\| \vec{c}_s - \vec{c}_r \|_{C_r}}{b_r} \right)^2 - P \right]^2, \tag{38}$$

kde $P$ je parametr překrytí.

V praxi se většinou této minimalizaci snažíme vyhnout různými heuristikami, např. nastavením šířky úměrně vzdálenosti několika nejbližších sousedů dané jednotky.

V **třetí fázi** zbývá nastavit hodnoty vah lineární kombinace tak, aby chybová funkce (36) byla minimální. Opět můžeme využít gradientní minimalizace, častěji se však používají známé numerické metody nejmenších čtverců (SVD-rozklad, QR-rozklad, pseudoinverse).

### 3.3. Genetické učení

Genetické algoritmy, používané při řešení různých optimalizačních problémů, lze použít i k učení neuronových sítí.

Genetický algoritmus pracuje s populací jedinců. Každý jedinec kóduje nějaké přípustné nastavení všech parametrů učené sítě. Uvažujeme-li různě dlouhé jedince, můžeme adaptovat i počet skrytých jednotek. Každý jedinec je ohodnocen hodnotou chybové funkce odpovídající sítě.

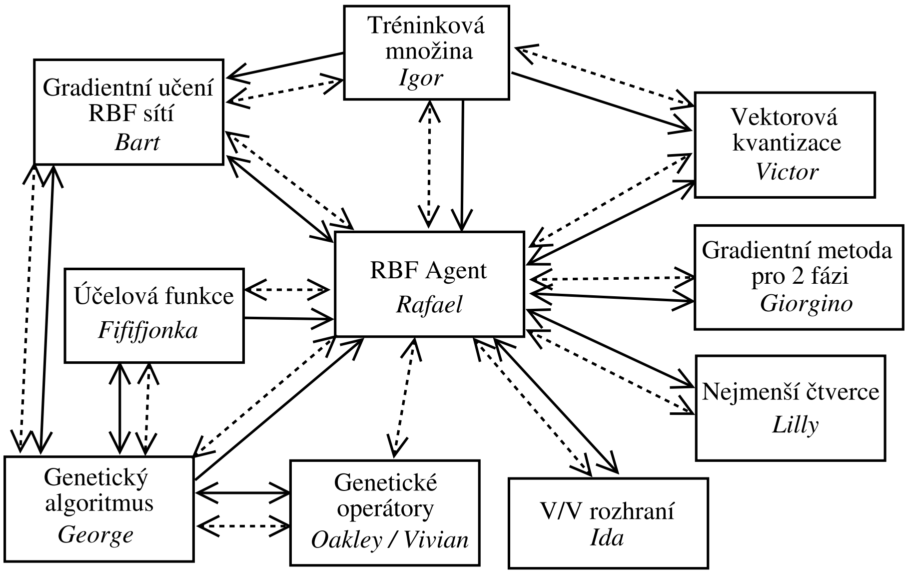
Začneme s populací náhodně vygenerovaných jedinců. V každém kroku pak generujeme pomocí operací *selekce*, *mutace* a *křížení* novou populaci. Operátor selekce zajišťuje, aby do následující generace byli s větší pravděpodobností vybíráni jedinci s menším ohodnocením. Operátor křížení vytváří nové jedince kombinací dvou vybraných jedinců, mutace pak vnáší do populace náhodné změny. Výpočet končí po nalezení jedince s dostatečně malou hodnotou chybové funkce.

Genetické algoritmy lze kombinovat s předcházejícími metodami. Lze je využít např. při řešení problému vektorové kvantizace nebo pomocí nich učit pouze skrytou vrstvu a jako ohodnocení použít několik iterací gradientního algoritmu.

Neruda v [4] popisuje *kanonickou* verzi genetického učení, která je rychlejší než klasický algoritmus.

## 4. Implementace v systému BANG2

Metody učení popsané v minulé kapitole byly implementovány pomocí adaptivních softwarových agentů v multiagentním systému Bang2 (viz [2, 1]). Každou metodu zajišťuje jeden agent, celé učení je pak řízeno RBF agentem, který reprezentuje samotnou RBF síť (viz obr. 4).




**Obrázek 4:** Agenti zapojení do učení RBF sítě. Čárkovaná šipka znázorňuje komunikaci mezi agenty, plná šipka přenos binárních dat.

Implementace pomocí více agentů přináší několik výhod. Učení RBF sítě využívá řady metod nezávislých na RBF síti (vektorová kvantizace, metody nejmenších čtverců, genetické algoritmy). Tyto metody jsou implementovány jako samostatní univerzálně použitelní agenti.

Dělba práce mezi jednotlivé agenty dále umožňuje snadno vytvářet nové kombinace již implementovaných metod, přidávat nové komponenty nebo nahradit některého agenta novým agentem se stejným rozhraním.

## 5. Výsledky experimentů

Metody uvedené v sekci 3 jsme otestovali pomocí popsané implementace na aproximačních a klasifikačních úlohách. Nyní shrneme výsledky dvou experimentů.

Pro aproximaci jsme zvolili funkci $sin(x)cos(y)$ rovnoměrně navzorkovanou na intervalu $(0, 2\pi) \times (0, 2\pi)$. Tréninková množina byla tvořena 100 tréninkovými vzory. Učili jsme RBF síť s 16 skrytými jednotkami.

Úlohu jsme řešili třemi různými způsoby. Všechny metody si s úlohou dokázaly poradit s přijatelnými výsledky. Nejrychlejší je třífázová metoda, nejpomalejší jsou dle očekávání genetické algoritmy.

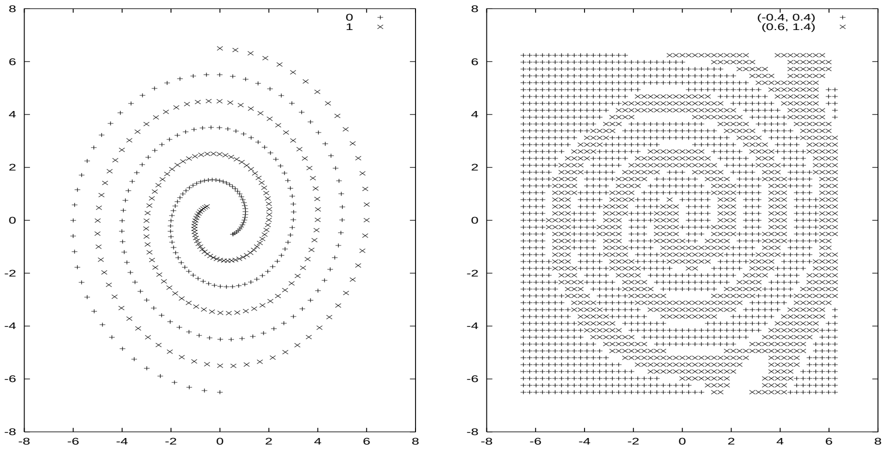
Nejpřesnějšího řešení jsme dosáhli pomocí gradientní metody, avšak za cenu vyšších časových nároků.

Tabulka 1 obsahuje časové srovnání uvedených metod. U třífázové metody jsme použili rovnoměrné rozmístění středů a metodu nejmenších čtverců. U genetického algoritmu se jedná o kanonickou verzi s populací 50 jedinců, elitou 2 a konstantní délkou jedinců.

Druhým experimentem byla klasifikační úloha, tzv. *problém dvou spirál.* Tréninková data obsahují 372 vzorků z dvou množin bodů (viz obr. 5). Úkolem sítě je separovat tyto dvě množiny.

Problém jsme řešili pomocí gradientního učení a třífázového učení s sítí o 150 jednotkách. Uvažovali jsme jak síť s euklidovskou normou tak s váženou normou. V obou případech nalezl gradientní algoritmus řešení s menší chybou. Třífázové učení si však poradí s problémem během několika minut, zatímco gradientní algoritmus potřebuje zhruba hodinu.

Obě metody nalezly lepší řešení pro síť, která používá vážené normy. Jedná se o dvourozměrný vstupní prostor a tak víme, že RBF jednotka s euklidovskou normou dává významné výstupy na kruhové oblasti kolem svého středu. Vážená norma umožňuje transformovat tuto oblast do tvaru elipsy, která může být v případě obecné matice normy ještě pootočena. Lepší výsledky v případě použití vážené normy lze vysvětlit tím, že vstupní vzory uspořádané ve spirálách je jednodušší pokrýt elipsami než kružnicemi.




**Obrázek 5:** Tréninková data — dvě spirály. Klasifikace bodů v rovině sítí naučenou gradientní metodou.

## 6. Závěr

Popsali jsme obecný model RBF sítě a seznámili se s myšlenkami tří základních přístupů k jejímu učení. Nastínili jsme implementaci učení v systému Bang2.

Z výsledků experimentů vidíme, že jednotlivé metody se od sebe liší časovou náročností i přesností řešení.

Časově nejnáročnější se ukázaly genetické algoritmy, které navíc v uvažovaném počtu iterací nepřinesly lepší řešení než druhé dvě metody. Jsou však snadno paralerizovatelné. Gradientní algoritmus je druhý v časové náročnosti, přináší přesnější řešení. Třífázové učení je nejrychlejší.

Použití vážené normy se ukazuje jako výhodné.

Vidíme, že RBF sítě se dají využít k řešení jak aproximačních tak klasifikačních problémů. Volba metody učení, velikost skryté vrstvy a jejích parametrů je závislá na velikosti a typu problému, na dostupných výpočetních možnostech a požadované přesnosti řešení.

| $\epsilon$ | tři fáze | gradientní metoda | genetické alg. |
|---|---|---|---|
| 2 | 1 s | 30 s | 9 min 40 s |
| 0.5 | 1 s | 52 s | – |
| 0.25 | 2 s | 1 min 15 s | – |
| 0.1 | – | 5 min 48 s | – |
| 0.05 | – | 26 min | – |

**Tabulka 1:** Srovnání uvedených metod. Čas potřebný k překonání prahu $\epsilon$.

| Euklidovská norma | | |
|---|---|---|
| průměr | minimum | maximum |
| 0.057 | 0.042 | 0.075 |
| Vážená norma | | |
| průměr | minimum | maximum |
| 0.012 | 0.010 | 0.014 |

**Tabulka 2:** Výsledná hodnota chybové funkce (po 10 000 iteracích gradientní metody).

| Euklidovská norma | Vážená norma |
|---|---|
| 1.101 | 0.051 |

**Tabulka 3:** Hodnota chybové funkce sítě naučené třífázovou metodou.

Podrobnější popis jednotlivých metod učení a úplnou dokumentaci prováděných experimentů lze najít v [3].

## References


[1] *On-line dokumentace Bang2.* (available in HTML form at `http://bang2.sourceforge.net/`).

[2] P. Krušina, R. Neruda, and Z. Petrová. Soft Computing Agents and Bang2. Technical Report V–819, Institute of ComputerScience, Prague, 2000.

[3] P. Kudová. Neuronové sítě typu RBF pro analýzu dat. Master's thesis, Charles University, Prague, Faculty of Mathematics and Physics, 2001.

[4] R. Neruda. *Functional Equivalence and Gentic Learning of RBF Networks.* PhD thesis, Institute of Computer Science, Academy of Sciences of the Czech Republic, Prague, Czech Republic, 1998.

# Hybridní UI modely a Bang2

*doktorand:*
ZUZANA PETROVÁ
Ústav informatiky AV ČR,
Pod vodárenskou věží 2, 182 07, Praha 8
petrova@cs.cas.cz

*školitel:*
ROMAN NERUDA
Ústav informatiky AV ČR,
Pod vodárenskou věží 2, 182 07, Praha 8
roman@cs.cas.cz

obor studia:
Teoretická informatika

**Abstrakt**

Popíšeme systém, který representuje hybridní výpočetní modely jako komunity spolupracujících autonomních softwarových agentů. Systém podporuje snadné vytváření nejrůznějších kombinací moderních metod umělé inteligence, například neuronových sítí, genetických algoritmů či fuzzy logických kontrolerů, a jejich distribuované spuštění na clusteru pracovních stanic. Paradigma autonomních agentů umožňuje poloautomatické vytváření modelů či dokonce automatický vývoj hybridních schémat.

## 1. Úvod

Hybridní modely, zejména kombinace metod umělé inteligence, jako jsou například neuronové sítě, genetické algoritmy či fuzzy logické kontrolery, se zdají být slibnou oblastí výzkumu a jsou též směrem v současné době poměrně rozšířeným [1]. My jsme s nimi experimentovali v předchozí implementaci našeho systému (popsán v práci [3]) s povzbuzujícími výsledky soudě alespoň podle srovnávacích testů [3].

Bang2 sestává z populace agentů žijících v prostředí, které jim poskytuje podporu pro vytváření nových agentů, komunikaci a distribuované spuštění v počítačové síti.

Naše pojetí inteligentního agenta vychází z výborného úvodního článku od pana Franklina [2]. Obecně, agent je entita (v našem případě počítačový program), která je autonomní, vnímá stav a změny ve svém okolí a odpovídá na ně, sledujíc tím vlastní cíle. Takovýto agent může být adaptivní či inteligentní, ve smyslu být schopen sbírat informace, které potřebuje, nějakým sofistikovaným způsobem. Naši agenti jsou navíc mobilní (mohou se přemístit na jiný počítač) a trvalí, tedy schopní přežít i poté, co přestane existovat jejich prostředí. Neuvažujeme o jiných "typických" vlastnostech agentů jako simulace lidských emocí, nálady atp.

Prostředí dává agentům možnost komunikovat a zprostředkovává nezbytné služby. Agenti komunikují pomocí zpráv. Zajímavou vlastností komunikace je tzv. location transparency, tedy skrytí rozdílů mezi tím jestli spolu komunikující agenti jsou na stejném či na různých počítačích, což značně zjednodušuje práci programátorům agentů. Na druhé straně je tu důležitá podpora pro synchronně i asynchronně posílané zprávy.

Z hlediska programátora je agent v Bangu2 třída v C++, odvozená od základní třídy Agent, která se umí připojit k prostředí a reaguje na několik nejdůležitějších zpráv. Programátor nového agenta pak jen připisuje triggery. Trigger je procedura reagující na jednu konkrétní zprávu. Prostředí je pak odpovědné za spuštění správného triggeru po příchodu zprávy.

## 2. Jazyk pro komunikaci mezi agenty

Agenti potřebují jazyk pro různá vyjednávání (např. neuronová síť si potřebuje dojednat, kdo ji bude učit) a pro přenos dat. Tento jazyk musí být schopen popsat základní typy komunikace, jako požadavek, přijetí, odepření, dotaz. Ale také popsat rozličné datové formáty od prostého celého čísla k vnitřnímu stavu neuronové sítě.

Několik jazyků pro agenty už existuje. Všechny, pokud vím, předpokládají, stejně jako my a Bang2, spolehlivé doručování zpráv dle protokolu TCP/IP. Nejpropracovanější z nich, ACL [5] a KQML ([4], de facto standard), velice pečlivě rozlišují hlavičku a obsah zprávy. Hlavička je v obou případech popsána podobným, Lispem inspirovaným, jazykem a obsahuje informace jako odesílatel, příjemce, identifikátor konverzace, jíž je zpráva součástí, jazyk, v němž je zpráva napsaná, ale i obor, jehož se zpráva týká a jehož znalost je důležitá pro pochopení zprávy (skupina kolem KQML k tomuto účelu vyvinula jazyk KIF, založený na predikátové logice, v němž zapisuje znalosti (predikáty). Oborem se pak míní soubor s predikáty). Obsah zprávy může být v libovolném jazyce, ale použije-li se jazyk KIF [6], je možné využít výše zmíněných aktivit skupiny KQML.

Z jazyků pro přenos dat stojí za zmínku PMML [7] a XSIL [8], používající XML [9].

Zprávy v Bangu2 přijaly syntaxi XML [9]. Hlavičky zpráv, jaké prosazuje ACL a KQML, nejsou nutné, potřebné informace se dozvíme jinak. První XML tag definuje typ zprávy:

- *request* (požadavek),
- *inform* (poskytnutí informace),
- *query* (sbírání informací),
- *ok* (odpověď, vše v pořádku),
- *ugh* (odpověď, někde je chyba).

Vlastní zpráva, tedy vše mimo vnějších tagů, obsahuje příkazy (typ request), informace (typ inform), dotazy (typ query), a odpovědi. Některým rozumí všichni agenti (protože na ně umí odpovědět základní třída agent), ostatní jsou srozumitelné jen určitým zájmovým skupinám agentů (naštěstí odpověď "nerozumím" je vždy po ruce).

Data se v Bangu2 mohou přenášet buď jako XML řetězce nebo binárně. XML řetězce jsou čitelné pro uživatele, a díky volnějšímu formátu dat vhodné při vyjednávání, bohužel výrazně pomalejší než binární přenos, který se tedy využívá při přenosu rozsáhlejších dat. Agenti zpočátku posílají zprávy v XML, pokud se dohodnou na přesném formátu, mohou komunikovat i binárně.

## 3. Agenti

V této sekci Vám představíme jednak příklad agenta, pro jakého byl Bang2 psán, a to agenta provozujícího genetické algoritmy, a potom Zlaté a Bílé stránky, které provozují služby užitečné ostatním agentům. Popis spousty dalších agentů můžete najít na stránkách Bangu2 [10].

```
<broadcast><halt/></broadcast>
<inform>
<created myid="!000000000001"
  name="Lucy"
  type="Neural Net.MLP"/>
</inform>

<ok>Agent Lucy, id=!000000000001,
type=Neural Net.MLP created</ok>

<request><ping/></request>
```

**Obrázek 6:** Příklady zpráv v jazyce Bang2, které rozumí jen Zlaté Stránky.

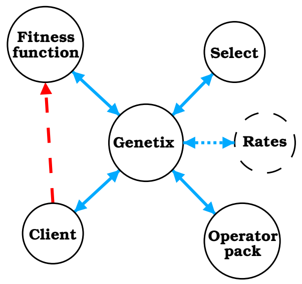
```
<query><vector row="45"/></query>
<query><vector/></query>
<ok><data separator=",">
Zde jsou binární data
</data></ok>
<query><bin><query>
<vector/>
</query></bin></query>
<ok session="5" funcnum="1"/>
```

**Obrázek 7:** Příklady zpráv v jazyce Bang2 pro přenos dat.

### 3.1. GA agent

Tedy on to není jeden agent, ale rodina vzájemně spolupracujících agentů. Genetický algoritmus začíná vytvořením populace jedinců daného typu, pokračuje změřením fitness, dále vytvoří novou populaci pomocí nějaké metody selekce a nakonec zavolá genetické operátory. Vše mimo prvního kroku potom opakuje až do splnění nějaké podmínky.

Genetix je agent provozující obecný genetický algoritmus. Dostane jedince kterého naklonuje v populaci příslušné velikosti. Je schopen vybírat jednice do nové populace a cyklit až do té chvíle než fitness dostatečného počtu jedinců je dost veliká. Protože je ale obecný nemůže dělat nic závislého na struktuře jedince a těchto krocích spolupracuje se specializovanějšími agenty, kteří umí náhodně nastavit proměnné v jedinci, počítají fitness a realizují genetické operátory.




**Obrázek 8:** Agenti spolupracující na učení pomocí GA.

### 3.2. Zlaté a Bílé stránky

Poskytují užitečné služby jako informace o jménech a typech spuštěných agentů, typech a schopnostech všech agentů, které má Bang2 k dispozici a také ukládají stav agentů.

Zlaté stránky se starají o živé (rozuměj běžící) agenty. Uchovávají informace o jejich jménech, číslech a typech. Každý agent se ihned po vytvoření registruje u Zlatých stránek a informuje je, když se přemisťuje na jiný počítač či je zabit. Každý agent má číslo, ze kterého se dá vyčíst, na kterém počítači agent je. Jméno slouží ke komunikaci s uživatelem namísto nezapamatovatelného čísla. Typ je jakýsi popis agentovy struktury či funkce, například genetika, RBF síť, či zlaté stránky.

Bílé stránky udržují informace o všech typech agentů, kteří mohou být kdykoliv spuštěni. Taktéž si ukládají stav agentů, kteří o to požádají. O agentech ukládají tyto informace:

- typ — struktura agenta (stejné jako položka typu Zlatých stránek), např. neuronová síť, genetika, atp. Podporuje dědičnost ve smyslu klasických programovacích jazyků jako např. C++ (to, co je vícevrstvý perceptron je samozřejmě také neuronová síť, tolik dědičnost),
- popis — schopnosti agenta, zejména pro uložené vnitřní stavy již naučených agentů (učení RBF sítí, rozpoznávání písmen, atp.),
- stav — vnitřní stav agenta (např. váhy neuronové sítě),
- pomocné informace.

Zlaté i Bílé stránky samozřejmě reagují na požadavky přidej, smaž a na dotazy.

## 4. Závěr a výhledy do budoucna

Nyní je hotova implementace prostředí a několik agentů. Připravuje se jednak naprogramování daleko většího počtu agentů, možná úpravy prostředí, spousta experimentů se složitějšími schématy, zaměříme se zejména na mirroring agenty, paralelní zpracování, automatické generování a vývoj schémat. Zajímá nás též koncept agenta jako mozku pro jiné agenty. Očekávají se i další podpůrní agenti, agent vyvažující zátěž na různých strojích a měnící poměr komunikace/počítání.

## References


[1] Pietro P. Bonissone, "Soft computing: the convergence of emerging reasoning technologies", *Soft Computing*, vol. 1, pp. 6–18, 1997.

[2] Stan Franklin and Art Graesser, "Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents", *Intelligent Agents III*, pp. 21–35, 1997.

[3] Roman Neruda and Pavel Krušina, "Creating Hybrid AI Models with Bang", *Signal Processing, Communications and Computer Science*, vol. I, pp. 228–233, 2000.

[4] Tim Finnin and Yannis Labrou and James Mayfield, "KQML as an agent communication language", *Software Agents*, 1997.

[5] "Agent Communication Language", Technical report, Foundation for Intelligent Physical Agents, 1998.

[6] Michael Genesereth and Richard Fikes, "Knowledge interchange format, v3.0 reference manual", Technical report, Computer Science Department, Stanford University, 1995.

[7] "PMML v1.1 Predictive Model Markup Language Specification", Technical report, Data Mining Group, 2000.

[8] Roy Williams, "JAVA/XML for Scientific Data", Technical report, California Institute of Technology, 2000.

[9] Extensible Markup Language (XML) 1.0 (Second Edition) W3C Recommendation, 2000.

[10] Bang2 homepage, http://bang2.sourceforge.net, 2001.

# Generalised Objective Functions for Multi-Layered Neural Networks with Genetic Training

*doktorand:*
MILAN RYDVAN
Ústav informatiky AV ČR, Pod vodárenskou vězi 2,
18 207 Praha 8
rydvan@cs.cas.cz

*školitel:*
LADISLAV ANDREJ
Ústav informatiky AV ČR, Pod vodárenskou vězi 2,
18 207 Praha 8
andre@cs.cas.cz

obor studia:
I1 - Teoretická informatika

**Abstrakt**

The aim of this paper is to outline the basic ideas of applying alternative objective functions for multi-layer neural networks. Special attention is paid to MNNs with genetic training, which allows the use of arbitrary objective functions. The approach is tested on the practical task of share price prediction, where a profit-based objective function is implemented in Matlab.

## 1. Introduction

First, let us very briefly introduce the model of *multi-layer neural networks*. It is based on a simplification of a biological neural network - the brain. An *artificial neuron* is a device, which computes the output $y$ from an input vector $x \in \mathcal{R}^n$ as follows:

$$y = f(\sum_{i=1}^{n} w_i x_i - t),$$

where $x$ is the input vector, $w \in \mathcal{R}^n$ are the weights of the neuron and $t$ is its threshold. Neurons in a MNN are divided into layers, where the first one contains the input of the whole network; outputs of the neurons in the $i$-th layer form the inputs of all the neurons in the $(i+1)$-st layer and the last layer produces the output of the whole network. Thus the input is propagated throughout the network, layer by layer.

This model of neural networks is based on supervised training. That means that we have a finite *training set* $T = \{(\vec{x}_i, \vec{d}_i)\}$ of pairs of input vectors $(x_i)$ and desired output vectors belonging to them $(d_i)$. The aim of the training is to modify the network's parameters (weights, thresholds) from the initial random values so that an *objective function* $\sum E(d_i, y_i)$ summed over all the output neurons and all the training patterns was minimised.

A commonly used training algorithm, called *Back-Propagation*, is based on the steepest descent minimisation method. It understands $E$ as a function of the network's parameters and in every step it moves in the direction of the gradient of $E$ in this space, until it reaches a (generally local) minima. There is very rich literature on the MNN model and the BP training algorithm; see for example [5].

This BP algorithm requires $E$ to be differentiable in $y$, so that partial derivatives according to the parameters existed. An example of such function is the widely used summed-square objective function $E = (y_i - d_i)^2$. This function is, however, too special. The problem is, that it depends only on the value $|y - d|$, i.e. it returns the same values for $y > d$ and $y < d$ when their distance is equal. There are examples, e.g. stock price prediction, where it is useful to use objective functions that do not meet this condition.

## 2. Alternative Objective Functions

The aim of my dissertation thesis is to study alternative objective functions and to propose functions that are more general in the above mentioned sense. Because the BP algorithm requires the objective function to be differentiable, which is not always true for the objective functions we would like to use, the first alternative functions proposed were differentiable approximations of the generally non-differentiable objective functions. Biquadratic approximations can be found in [3], while [4] uses a modular approach — the value of the objective function is computed using a separately trained neural network. Both of these approaches have shown improved behaviour and results overcoming the standard networks, when tested on a practical task.

This article examines truly general objective functions. As a trial task, we will use predicting the stock price behaviour on the Prague Stock Exchange. We will use a model of profit our network would achieve on a real market as the objective function. Rather than seeking minimisation of an error (i.e. the difference $|y - d|$), our objective is to maximise the profit. The model we employ is as follows:

$p = d - t$ iff $y > t$, (price rise prediction, recommendation to buy)

$p = -d - t$ iff $y < -t$, (price fall prediction, recommendation to sell)

$p = 0$ otherwise, (stagnation or small change prediction, no action recommended)

where $d$,$y$ and $t$ are the real stock price change achieved on the market, the change predicted by the network and the transaction costs, respectively, all in per cents. Note that the objective (or profit) function $p$ is asymmetric — it returns different results for pairs with the same distance $|y - t|$. It is not differentiable and we thus cannot employ the BP training algorithm when training a network using the profit objective function $p$.

## 3. Genetic Training

We will employ a genetic algorithm to train our network. For a description of this model, based on the rules of genetics, see e.g. [2]. The idea of genetic algorithms is that candidate solutions of the function that is to be optimised are represented as *chromozomes* - sequences of values (*genes*), either 1-bit, 2-bit (as in the case of real chromosomes) or even real values. Each of the chromosomes is assigned a *fitness* — the value of the function on the particular candidate solution, also called *individual*. The first *generation*, i.e. a set of individuals, is generated randomly. The $(i + 1)$-st generation is generated from the $i$-th one, by the operators of *selection* (chooses, which individuals enter the next population) , *crossover* (creates offspring of some of these individuals) and *mutation* (modifies some of them randomly) . Individuals with higher fitness have a higher chance of being selected, which ensures overall improvement of the population. Crossover ensures combination of patterns, while mutation represents a random factor necessary to avoid local minima.

When applied to neural networks, GAs can be used in several ways; in this article, we will use them to determine the best parameters (weights and thresholds) of the network. We will represent them as genes in real-valued chromosomes. Each chromosome will then represent a neural network, and its fitness will be defined as the average value of the objective function applied on the network and the training data. The GA will then run as usual, creating a first random population of neural networks and improving it using selection, crossover and mutation. As GAs search for better solution using a network of cooperative agents rather than employing the gradient descent method, it does not require differentiable objective function.

## 4. Application for Financial Data

This section describes the testing problem — the stock price prediction mentioned above — and the preliminary results we have obtained by employing the proposed approach.

The aim was to predict the next daily price change of a stock, given the five previous price changes, and several technical analysis information (volume of trade, supply/demand ratio etc.) from the previous day. The network architecture was 9-15-1 with a single hidden layer[1], which was represented by a chromosome with 166 genes. Matlab 6.0 was used for the experiments, together with the neural networks toolbox and a genetic algorithms toolbox developed by [1].

Regarding parameters of the GA, there were 80 individuals in the population; 20 pairs were selected for crossover and 4 individuals underwent mutation in every generation. The normalized geometric ranking was used for selection, where the probability of selecting the $i$-th individual equals

$$P_i = \frac{q}{1-(1-q)^P}(1-q)^{r-1},$$

where $q$ is the probability of selecting the best individual, $r$ is the rank of the $i$-th individual and $P$ is the population size. The parameter $q$ was set to 0.08. The heuristic crossover function was employed, which replaces parents $X$ and $Y$ by offspring $X' = X + r(X - Y)$ and $Y' = X$ if the fitness of $X$ was better than that of $Y$ and vice versa. The uniform mutation operator was applied; it randomly selects one gene and sets it equal to a uniform random number from the permitted space of values. The permitted set of values for the parameters (weights and thresholds) was set to $(-10, 10)$. The evolution continued for 200 generations or until the fitness of the best individual reached 0.35. The best individual then represented the trained network. For more detailed information on functions and parameters used by GAs see [1].

500 tests have been carried out, comparing the proposed model, which used the profit objective function and was trained by the genetic algorithms with networks trained by the standard BP algorithm and employing the summed-square objective function. During each of the tests, the set of known data was randomly divided into a training set (90% of them) and a test set (10%), unseen by the networks during training. Averaged results of the 500 tests are shown in Table 1.

| Err.f. | Set | Square error | Dir.corr. | Profit |
|---|---|---|---|---|
| Standard | Train | 0.033 | 81.7% | 46.9% |
| | Test | 0.051 | 73.1% | 17.7% |
| Profit | Train | 0.192 | 76.8% | 34.2% |
| | Test | 0.221 | 70.3% | 10.1% |

**Tabulka 4:** Results on financial data, separately for the training set and the test set. The first two rows contain the average result of networks trained by Back-Propagation and the standard objective function. The second two show the average result of networks trained genetically and employing the profit objective function. Several measures of success are presented - summed-square error, direction correctness (i.e. the percentage of correct prediction of price rise/decrease) and profit as defined above.

[1]This architecture has proven efficient for this task during earlier experiments.

Note that the results are quite good. Direction correctness (i.e. the ratio of successful trend predictions) over 70% and profit exceeding 0.1% per trading day are not bad, even though we have to take into account that we use only a limited model of profit (we do not take into account e.g. the possibility that our trading order was not fulfilled). On the other hand, these are net results, taking into account the transaction costs.

We can see that the profit achieved by the new objective function is lower than that achieved by the standard one. The direction correctness is also slightly worse. A positive result is that the margin, by which the standard error function defeated the alternative one, is smaller in the case of the test set than in the case of the training set. This might indicate that the generalisation ability of the profit-trained networks is higher. Also, predictions of networks trained by the alternative objective function were more "courageous", e.g. their prediction resulted either in purchasing or in selling stock more often than in the case of the standard networks (99% vs. 72%). This is similar to the behaviour of networks trained by other alternative functions ([3], [4]). The reason is that the profit-based objective function penalizes also the so-called opportunity loss, even though only indirectly. Another similarity is in the high summed-square errors when employing profit-based objective functions. It is related to the previous observation of frequent trading. These "alternative" networks often trade even if stagnation is quite probable because the loss of transaction costs in the case stagnation really occurs is smaller than the opportunity loss in the case they predicted stagnation and a price rise/decrease actually occurred. Apparently, their courage was even too high in this case.

Let us stress that the experimental results presented here are only preliminary. Presumably, after further experimenting with parameters of the genetic algorithm and with interconnecting of the two models, the profit achieved by the networks with the alternative objective function will rise.

## 5. Conclusion

We have discussed the possibility of using alternative objective functions for training neural networks. The standard summed-square error function limits us, as it is too general to include special knowledge we might have about the problem. Using an approximation, as discussed in previous papers, have proven suitable, but it suffers from the problems of approximations — it never fits the approximated function exactly.

The approach described in this article, i.e employing genetic training of neural networks, allows us to choose an arbitrary objective function, which no longer has to be differentiable. This may prove useful in solving tasks, where a better measure of success than the mere distance of the desired and the actual solution is known, but this measure is not smooth.

The example of stock price prediction was given to illustrate the proposed approach. The model of profit used as objective function was described, as well as the parameters and method employed by the combined GA-NN model and results of their implementation. The success of the achieved predictions is quite good, even though further improvement is possible and will be sought.

## References


[1] Chris Houck, Jeff Joines, Mike Kay, *"A Genetic Algorithm for Function Optimization: A Matlab Implementation"*, NCSU-IE TR 95-09, 1995, see `http://www.ie.ncsu.edu/mirage/#GAOT`.

[2] M. Mitchell, *"An Introduction to Genetic Algorithms"*, MIT Press, 1997.

[3] M. Rydvan, *"Biquadratic Error Functions for the BP-networks"*, TR No 98/10, Charles University Prague, MFF.

[4] I. Mrázová, M.Rydvan, *"Generalised Error Function for BP-Network"*, TR No 99/1, Charles University Prague, MFF.

[5] D. E. Rummelhart, G. E. Hilton,R. J. Williams, *"Learning representations by backpropagating errors"*, Nature (323), (1986), pp. 533-536, MFF.

# Some Estimates of Rates of Approximation by Neural Networks Using Integral Representations

*doktorand:*
TEREZIE ŠIDLOFOVÁ
Institute of Computer Science, Academy of Sciences of the Czech Republic
Pod vodárenskou věží 2, P.0. Box 5, 182 07 Prague 8, Czech Republic
terka@cs.cas.cz

*školitel:*
VĚRA KŮRKOVÁ
Institute of Computer Science, Academy of Sciences of the Czech Republic
Pod vodárenskou věží 2, P.0. Box 5, 182 07 Prague 8, Czech Republic
vera@cs.cas.cz

obor studia:
Teoretická Informatika

**Abstrakt**

We discuss approximation of functions by neural networks and rates of approximation derived using integral representation. We find conditions that guarantee approximation error rate of order $O\left(n^{1/q}\right)$ by one-hidden-layer networks with $n$ hidden units.

## 1. Introduction

In recent years approximation of multivariable functions by feedforward neural networks has been widely studied. The existence of an arbitrarily close approximation of any continuous or $\mathcal{L}_p$ function on a compact subset of $\mathcal{R}^d$ for one-hidden-layer network with perceptrons or radial-basis-function units with quite general activation functions have been proven (see e.g., Leshno et al., 1993; Park, Sandberg, 1993).

Currently neural networks are simulated on digital computers, and thus the number $n$ of hidden units necessary for a given accuracy of approximation is critical. Estimates derived from constructive proofs of approximation property grow exponentially with the number of input units (which correspond to the number $d$ of variables of the function $f$ to be approximated) (see e.g., Kůrková, 1992). The exponential growth of complexity with the number of variables is called the "curse of dimensionality" and it is a limiting factor in classical approximation methods. However in neural networks applications, functions with hundreds of variables were approximated sufficiently well using only reasonable number of hidden units (e.g., Sejnowsky, Rosenberg, 1987).

One method to cope with the problem of the curse of dimensionality is derived from Jones' construction (see Jones, 1992) of approximants with rates of convergence of the order of $O(1/\sqrt{n})$. Applying Jones' estimate Barron (1993) proved that it is possible to approximate any function satisfying certain condition on its Fourier transform within $\mathcal{L}_2$ error of $O(1/\sqrt{n})$ by a network with $n$ perceptrons with a sigmoidal activation function.

In this paper we extend the description of sets of functions to which rates of approximation by neural networks of the order $O(1/\sqrt{n})$ or $O(n^{-1/q})$ apply. Our results are derived by representing functions as networks with continuum of hidden units.

## 2. Approximation of Functions in Convex Closures

Let $\mathcal{R}, \mathcal{N}$ denote the set of real and natural numbers, respectively. Measurability will be considered with respect to Lebesgue measure in some subset of $\mathcal{R}^q$. We will denote by $\lambda_p(A)$ the $p$-dimensional Lebesgue measure of a set $A \subset \mathcal{R}^q$.

For a topological space $X$, $C(X)$ denotes the *set of all continuous real-valued functions* on $X$ and $\|.\|_C$ denotes the *supremum norm.* For $p \in \langle 1, \infty)$ and a subset $X$ of $\mathcal{R}^d$, $\mathcal{L}_p(X)$ denotes the space of $\mathcal{L}_p$ functions and $\|.\|_p$ denotes the $\mathcal{L}_p$*-norm.*

For any topological space $X$ with a topology $\tau$ and for any subset $A \subset X$ we write $cl_\tau A$ for the *closure* of $A$ (smallest closed subset containing $A$). Thus $cl_C$ denotes the closure in the topology of uniform convergence and $cl_{\mathcal{L}_p}$ the closure with respect to $\mathcal{L}_p$-topology. Closure of the convex hull is called the *convex closure.*

Let $G$ be a subset of $X$. Then the *linear span* of $G$, denoted by $span\, G$, is the set of all linear combinations of elements of $G$, i.e. $span\, G = \{\sum_{i=1}^n w_i g_i;\ w_i \in \mathcal{R}, g_i \in G, n \in \mathcal{N}_+\}$; $span_n G$ denotes the set of all linear combinations of at most $n$ elements of $G$, i.e. $span_n G = \{\sum_{i=1}^n w_i g_i;\ w_i \in \mathcal{R}, g_i \in G\}$. A *convex combination* of elements $s_1, \ldots, s_n$ $(n \in \mathcal{N})$ in a vector space is a sum of the form $\sum_{i=1}^n a_i s_i$, where $a_i$ are all non-negative and $\sum_{i=1}^n a_i = 1$. A subset of a vector space is convex if it contains every convex combination of its elements. We will denote by *convG* the set of all convex combinations of elements of $G$ calling it the *convex hull* of $G$. By analogy we define $conv_n G = \{\sum_{i=1}^n a_i g_i;\ a_i \in [0,1], \sum_{i=1}^n a_i = 1, g_i \in G\}$.

For a function $f : X \to \mathcal{R}$ the *support* of $f$ is defined as $supp(f) = cl_\tau\{x \in X; f(x) \neq 0\}$. For $f : X \to \mathcal{R}$ and $A \subset X$, $f|_A$ denotes the restriction of $f$ to $A$.

Jones (1992) estimated rates of approximation of functions from convex closures of bounded subsets of Hilbert spaces.

**Theorem 2.1** *Let $\mathcal{H}$ be a Hilbert space with a norm $\|.\|$ and $\mathcal{G}$ be a bounded subset of $\mathcal{H}$. Let us denote $s_G = \sup_{g \in G} \|g\|$. Then, for every $f \in cl\, conv\, \mathcal{G}$ and for every natural number $n$ the following holds:*

$$\|f - span_n G\|^2 \leq \|f - conv_n G\|^2 \leq \frac{s_G^2 - \|f\|^2}{n}.$$

Darken et al., 1993 extended Jones' theorem to $\mathcal{L}_p$ spaces, $1 < p < \infty$, with slightly worse rates of convergence, i.e. of order $O(n^{-1/q})$, where $q = \max\{p, p/(p-1)\}$.

We use these results to estimate the number of hidden units in neural networks, taking $G$ as the set of functions computable by computational units. The convex combinations of $n$ such functions can be thus computed by an $n$-hidden-unit network with one linear output unit.

Generally approximation by neural networks can be studied as follows: For $X, Y$ topological spaces, a function $\phi : X \times Y \to \mathcal{R}$, a positive real number $B$ and a subset $J \subseteq X$ define $\mathcal{G}(\phi, B, J) = \{f : J \to \mathcal{R};\ f(x) = w\phi(x,y),\ w \in \mathcal{R},\ |w| \leq B,\ y \in Y\}$. Thus $\mathcal{G}(\phi, B, J)$ are real-valued functions on $J$ parameterized by $y \in Y$ and scaled by a constant of at most $B$.

The following two theorems by Kůrková et al. (1997) characterize functions in convex closures of bounded subsets of Banach spaces.

**Theorem 2.2** *Let $d, k$ be any positive integers, $J$ be a compact subset of $\mathcal{R}^d$ and let $f \in C(J)$ be any function that can be represented as $f(x) = \int_Y w(y)\phi(x,y)dy$, where $Y \subseteq \mathcal{R}^k, w \in C(Y)$ is* *compactly supported and* $\phi \in C(\mathcal{R}^d \times Y)$. *Then* $f \in cl_C conv\, \mathcal{G}(\phi, B, J)$, *where* $B = \int_{J_\phi} |w(y)|dy$ *with* $J_\phi = \{y \in Y; \exists x \in J,\, w(y)\phi(x,y) \neq 0\}$.

This theorem can be applied to perceptron-type networks with any continuous activation function (a perceptron with an activation function $\psi$ computes function of the form $\psi(v \cdot x + b)$). Similar statement holds even for Heaviside-perceptrons, where a Heaviside $\vartheta$ function is taken as follows: $\vartheta : \mathcal{R} \to \mathcal{R}$, $\vartheta(x) = 0$ for $x < 0$, else $\vartheta(x) = 1$.

**Theorem 2.3** *Let* $d$ *be a positive integer,* $K \subseteq S^{d-1} \times \mathcal{R}$, *where* $S^{d-1}$ *denotes the unit sphere in* $\mathcal{R}^d$, $J$ *be a compact subset of* $\mathcal{R}^d$ *and* $f \in C(J)$ *be any function, that can be represented as* $f(x) = \int_K w(\mathbf{e}, b)\vartheta(\mathbf{e} \cdot \mathbf{x} + b)d(\mathbf{e}, b)$, *where* $w \in C(S^{d-1} \times \mathcal{R})$ *is compactly supported and* $supp(w) \subseteq K$. *Then* $f \in cl_C conv\, \mathcal{G}(P_\vartheta, B, J)$, *where* $\mathcal{G}(P_\vartheta, B, J) = \{g : J \to \mathcal{R}; g(x) = w\,\vartheta(\mathbf{e} \cdot \mathbf{x} + b), \mathbf{e} \in S^{d-1}, w, b \in \mathcal{R}, |w| \leq B\}$, *and* $B = \int_K |w(\mathbf{e}, b)|d(\mathbf{e}, b)$.

Further we will extend these results to more general functions.

**Theorem 2.4** *Let* $d, k$ *be any positive integers,* $J$ *be a compact subset of* $\mathcal{R}^d$. *Let* $(\mathcal{F}(J), \|.\|_p)$ *be a normed vector space of real-valued functions on* $J$ *and let* $f \in \mathcal{F}(J)$ *be any function that can be represented as* $f(x) = \int_Y w(y)\phi(x,y)dy$, *where* $Y \subseteq \mathcal{R}^k, w \in C(Y)$ *is compactly supported and* $\phi(x,y) = lim_{i\to\infty}\phi_i(x,y)$ *with respect to* $\lambda_{k+d}$, *where* $\phi_i(x,y) \in C(\mathcal{R}^d \times Y)$ *and* $max_i|\phi_i(x,y)| < \infty$. *Then* $f \in cl_{\|\cdot\|_p} conv\, \mathcal{G}(\phi, B, J)$, *where* $B = \int_{J_\phi} |w(y)|dy$ *with* $J_\phi = \{y \in Y; \exists x \in J, w(y)\phi(x,y) \neq 0\}$.

*Proof.* For $f(x) = \int_Y lim_{i\to\infty} w(y)\phi_i(x,y)dy$ we can use Lebesgue dominated convergence theorem to show that $f(x) = lim_{i\to\infty} \int_Y w(y)\phi_i(x,y)dy$ (note that $w(y)\phi_i(x,y)$ are bounded compactly supported continuous functions). Let us denote $f_i = \int_Y w(y)\phi_i(x,y)dy$. Using theorem 2.2, we have $f_i \in cl_C conv\, \mathcal{G}(\phi_i, B, J)$. Thus we know that $f(x) \in cl \bigcup_i cl_C conv\, \mathcal{G}(\phi_i, B, J)$. To conclude the proof of we will show that $cl \bigcup_i cl_C conv\, \mathcal{G}(\phi_i, B, J) \subseteq cl_{\|\cdot\|_p} conv\, \mathcal{G}(\phi, B, J)$. For every $f \in cl \bigcup_i cl_C conv\, \mathcal{G}(\phi_i, B, J)$ and for every $\varepsilon > 0$ we find $g \in conv\, \mathcal{G}(\phi, B, J)$, so that $\|f - g\| < \varepsilon$: As for $f = lim_{i\to\infty} f_i$, for every $\varepsilon_1 > 0$ there exists $n_1 \in \mathcal{N}$ such that for every $n \geq n_1$ $\left\|f - \int_Y w(y)\phi_n(x,y)dy\right\|_p < \varepsilon_1$. Further for every $\varepsilon_2 > 0$ there exists $m_2 \in \mathcal{N}$ such that for every $m \geq m_2$ and every $n \geq n_1$ $\left\|\int_Y w(y)\phi_n(x,y)dy - \sum_{i=1}^m w_i\phi_n(x,y_i)\right\|_p < \varepsilon_2$. And last, for every $\varepsilon_3 > 0$ there exists $n_3 \geq 0$ so that for every $n \geq n_3$ and for every $m \geq m_2$ $\|\sum_{i=1}^m w_i\phi_n(x,y_i) - \sum_{i=1}^m w_i\phi(x,y_i)\|_p < \varepsilon_3$. Now we take $\varepsilon_1, \varepsilon_2$ and $\varepsilon_3$ such that $\varepsilon_1 + \varepsilon_2 + \varepsilon_3 \leq \varepsilon$, find corresponding $n_1, m_2$ and $n_3$ and define $g = \sum_{i=1}^m w_i\phi(x,y_i)$, where $n \geq \max\{n_1, n_3\}$ and $m \geq m_2$. Q.E.D.

Theorem 2.4 can be applied to derive dimension independent rates of approximation for more general computational units.

## 3. Discussion

Jones-Barron Theorem in combination with integral formulas as a model of neural networks with continuum of hidden units helps to derive some estimates of rates of approximation by neural networks. We have extended the results given by Kůrková et al., (1997). We showed that dimension independent rates of approximation hold not only for continuous activation functions, but also for their $\mathcal{L}_p$ limits.

Theoretical properties of integral operators that correspond to continuum of hidden units in neural networks may be useful also in evolving of new hardware and software approaches thanks to the possibility of their physical implementation (holographic or analogue embodiments).

## References


[1] Barron, A. R., Neural net approximation, *In proceedings of the 7th Yale Workshop on Adaptive and Learning Systems*, pp. 69–72, 1992.

[2] Barron, A. R., Universal approximation bounds for superposition of a sigmoidal function, *IEEE Transactions on Information Theory*, vol. 39, pp. 930–945, 1993.

[3] Darken, C., Donahue, M., Gurvits, L., Sontag, E., Rate of approximation results motivated by robust neural network learning, *In proceedings of the 6th Annual ACM Conference on Computational Learning Theory*, pp. 303-309, 1993.

[4] Hornik, K., Stinchcombe, M., White, H., Multilayer feedforward networks are universal approximators, *Neural Networks*, vol. 2, pp. 251–257, 1989.

[5] Jones, L. K., A simple lemma on greedy approximation in Hilbert space and convergence rates for projection pursuit regression and neural network training, *The Annals of Statistics*, vol. 20, pp. 601-613, 1992.

[6] Kůrková, V., Kolmogorov's theorem and multilayer neural networks, *Neural Networks*, vol. 5, pp. 501-506, 1992.

[7] Kůrková, V., Kainen, P. C., Kreinovich, V., Estimates of the number of hidden units and variation with respect to half-spaces, *Neural Networks*, vol. 10, pp. 1061–1068, 1997.

[8] Leshno, M., Lin, V., Pinkus, A., Schocken, S., Multilayer feedforward networks with a non-polynomial activation function can approximate and function, *Neural Networks*, vol. 6, pp. 861-867, 1993.

[9] Park, J., Sandberg, I. W., Approximation and radial-basis-function networks, *Neural Computation*, vol. 5, pp. 305-316, 1993.

[10] Sejnowski, T. J., Rosenberg, C. R., Parallel networks that learn to pronounce English text, *Complex Systems*, vol. 1, pp. 145-168, 1987.

# Enhancements of Enterprise JavaBeans Component Model

*doktorand:*
Mgr. Radek Pospíšil
KSI MFF UK, Malostranské náměstí 25, 110 00 Prague 1
rpos@cs.cas.cz,pospisil@nenya.ms.mff.cuni.cz

*školitel:*
Prof. Ing. František Plášil, CSc.
KSI MFF UK, Malostranské náměstí 25, 110 00 Prague 1
plasil@cs.cas.cz,plasil@nenya.ms.mff.cuni.cz

obor studia:
I2 - Software Systems

**Abstrakt**

Contemporary component architectures development is getting more and more important in an industrial world. On the other hand there are differences between industrial and academic views on component architectures. Large companies implement their component models directly and push users to use them as-is. On the other hand academic world is spending long time on development and refinement of their component models, and they rarely implement them with poor support and development environment. The goal of this paper is to reuse the ideas gathered from academic component architectures at the Enterprise JavaBeans component architecture developed by Sun Microsystems.

## 1. Introduction

Headword of today's development is the reuse. There are many attempts proposed or under development, to reach this goal. But the fact is, that most of them is not as much as mature as users needs. On one side, there are industrial parties and groups proposing and implementing their vision of future development (COM/DCOM [11], EJB [10], CORBA Component model [5]). On the other side, there are academic research groups proposing their designs of component model development (Darwin, SOFA/DCUP). But industrial and academic streams are driven by different intentions. The industry needs to be quick in preparing the draft and implementation because of the market, but the academic one wants to have good design and implementation is not so important.

The goal of this paper is to present, enhance and repair Enterprise JavaBeans component model with features and ideas of academic component models. In the first section we present our view of component architecture. The second section presents EJB model. The third section presents our critique of EJB with respect to our view of architecture and component model. The third section contains the enhancements of EJB. In the fourth section we summarize the work and make a conclusion.

## 2. Component Architectures

One of the ways for describing applications or parts of an application based on component architecture is architecture description language (ADL) [4]. The ADL describe an application in terms of components (smallest parts of an application) and interconnection between them. This approach is not common in industrial component architectures ([9] [10] [11]), since an implementation language is used to describe the structure of outcome application ([9] [11]), or the XML descriptors are used to capture static references between components ([10]).

### 2.1. State-of-the-art of component architectures

All the component architectures are defined to meet two objectives. The first objective is to fulfill the common understanding of the terms architecture and component (there are many definitions of component architectures). The second objective is to fulfill specific features defined by component architecture provider (i.e., each specification of component architecture is different [2] [3] [6] [7] [8] [10] [11]). In this work we utilize experience acquired from our component architecture SOFA/DCUP [6] [7], and our projects, especially EJB comparison project [14].

**SOFA/DCUP component architecture** In SOFA/DCUP component architecture, a component is an instance of a component template (template for short). A component template is defined by a pair $< F, A >$ where F is a template frame (frame for short), and A is a template architecture (architecture for short). A frame represents the gray-box view of a component. It is defined by the behavior represented by the protocol [7], and by the set of interfaces that a component can possess and utilize as provides and requires interfaces. As frame defines a surface of a template, architecture defines an internal structure of a template. The internal structure is defined on the first level of nesting only - and recursively, each internal component template is defined by its pair $< F, A >$. Architecture is described by templates and their interconnections via interface ties. There are four types of ties: a) binding between requires interfaces and provides interfaces of internal template frames, b) delegating from a provides interface of a parent frame to its internal frame's provides interface, c) subsuming from an internal frame's requires interface to a parent frame's requires interface, d) exempting of an internal frame's interface from any ties. There are two types of architecture - primitive, where the component is implemented in target language, and composed, where the component has no implementation, but it is composed of components. An application in SOFA/DCUP is a component template, where its architecture does not make any ties on its frame's provides interfaces and requires interfaces.

**Server-side component architecture** The most widely used component architectures, Sun Microsystems's Enterprise JavaBeans [10] and Microsoft's COM+ [11], are so-called server-side component architectures. In state-of-the-art industrial software systems multi-tier architectures play the principal role. Applications are built of several layers - the server layer, intermediate layers (e.g., distribution, transactions, security), and the client layer. Thus an application is seen as a service provided by the server layer, where intermediate layers represent non-functional properties of a service. In the server-side component architecture, an application is a top-level component providing functionality and requiring nothing. There is no requirement on a client to be a component - a client can be both a standalone piece of a code, or a component located on a server.

### 2.2. Features of component architecture

The goal of our work is to strengthen component architecture of Enterprise JavaBeans. We present our view on the features of a server-side component architecture. This view is based on the study of component architectures presented above (SOFA/DCUP...), and current server-side component architectures (i.e., EJB, COM/DCOM, CCM). The discussed features cover three aspects of the whole system - component, architecture, deployment and runtime. In this section we do not map the features of component architecture into a real system - it will be done in the section 4.

A component represents basic unit of functionality. The functionality (we call it component type) is described by sets of provides and requires access points, and by a behavior description. Those access points can be of three types - a method-based interface, an event-base interface and a property. The behavior description complements sets of provides and requires of a more precise expression of a component type. The non-functional properties are the next way to specify components' behavior. They allow to setup transparent usage of a service (e.g., transaction service) without necessity of code modification.

Generally, the type is given by provides interfaces and requires interfaces. This definition is generally enhanced by description of behavior to component type (e.g., protocols [7]) - it defines compatibility of components as compatibility of behaviors. Briefly, to decide if a new component can replace an old component, the new component has to provide the same or even more, and require the same or less. This notion of sub-typing was adopted by many approaches (e.g., [13]).

There is one area little covered by current component architectures - if we do not reflect a component type (i.e., provides, requires and behavior), the following question arise - "what is a general behavior of a component?" COM/DCOM specifies a component as stateless, EJB offers four "component-types", CCM has four component categories, and SOFA/DCUP does not mention it. A component specification has to be enriched by another feature - we call it component category (this term is borrowed from CORBA Component Model). We consider two different approaches to specify component category. In the first approach, component category is represented by a set of properties (e.g., stateless/stateful, persistent/transient, reentrant/non-reentrant...), but the set of properties is fixed and cannot be widen. In the second approach, component category is represented by a piece of a code overseeing traffic on an instance, and controlling life cycle of an instance. In this case, such code is similar to CORBA's portable object adaptor.

We assume that a hierarchical structure of an application is important. Using hierarchy of components, applications become well structured and increase reusability of components. From this point of view, a component can be primitive (i.e., it does not contain another component, but it contains a code only), or composed (i.e., it has not a code, but it contains components).

**Architecture** Admitting hierarchical structure to components evoke need for architecture. Following the design of SOFA/DCUP architecture, we have to be aware of the fact that a server-side application, as stated before in the previous section, is not top-level component without requires and provides, but it is a top-level component offering provides interfaces to clients. Basically, a client is a piece-of-code accessing such top-level component. But a client can be a component.

**Dynamic Architecture:** Current trend in component architecture modeling is to add dynamic behavior. By dynamic architecture we mean ability to add, remove, rebind both component instances and components in run-time. There are approaches to deal with those goals, namely C2 [15][16], Wright [17], Darwin [18] and CHAM [19]. C2 describes architecture changes in terms of the architecture model - it uses shell-like language to execute changes. Wright describes modifications in CSP language. Darwin permits to parameterize initial architecture and during run-time components may be replicated. CHAM formalism is based on category theory. Our SOFA/DCUP is not dynamic, but it supports replacement of component in run-time. Simple server-side component architectures (i.e, EJB, COM/DCOM and CCM) use factory-pattern to create/remote instances.

Our understanding is, that every composed component has an initial configuration. An initial configuration defines contained components, bindings between them, and delegation and/or subsuming of contained components' interfaces from/to parent component. Exemption is allowed for top-level components representing applications. Dynamic change is triggered on two actions: on addition of a component instance or on removal of a component instance. As the first step, an instance of component is instantiated (e.g., using factory interface). Then the instance is bind into its parent component by resolving its requires and provides. Resolving new component's provides and requires can affect parent component requires and provides - a new provide and/or require is added. Thus addition could be non-local to a parent component if it affects parent's requires and provides. In the case of removal, an instance can be removed if another component instance or a client does not require it. Architecture of server-side application Integral part of the architecture is definition of living space for component instances. Such space offers a lifecycle manager for instances, and a set of basic services, which can be accessed directly or indirectly by instances (e.g., security libraries). The services can also be offered as special system primitive components, if they are complex (e.g., advanced transactions models). Indirect access to services is given by non-functional properties specified at the develop-time and/or the deploy-time.

**Deployment and Runtime:** The first phase of starting of an application is deployment - specification of the runtime environment and distribution of components' code onto dedicated computer nodes. The runtime environment is responsible for instantiation of component instances during startup and/or runtime of an application. It has to offer necessary services (both libraries and special primitive system components). On each computer node the runtime environment has to be provided. The deployment and runtime support of SOFA/DCUP is particularly proposed in the work [12].

## 3. Enterprise JavaBeans

The EJB is tightly bound to Java environment. The Java language serves for definition of interfaces, and it is the main implementation language. All the services required by EJB specification have their mapping and/or implementation in Java too. The EJB supports mapping to another languages via OMG's CORBA and it is incorporated into another component architecture - OMG's CORBA Component Model.

### 3.1. Component model

The EJB component can be viewed as an object with non-functional properties. It can be accessed as either a remote or local one. A remote access is supported by any distribution mechanism (e.g., CORBA, RMI, JavaServlets). A local access is in EJB since final draft and is proposed to provide local (i.e. faster) invocations between beans in the same container. For each access (either remote or local) there is a set of two provided interfaces - home and business interfaces. The business interface represents the functionality provided. The requirements are stored in the private naming space of the component (accessible via JNDI), and they are resolved at the deployment time. These are properties (i.e., value-name pairs), and references to interfaces.

The EJB offers four component categories (the EJB specification calls them "bean types"). There are three non-persistent component categories - stateful bean, stateless bean and message-driven bean, and one persistent - entity bean. Three of them provide method-based interface (stateless, stateful and entity), and one of them provides event-based interface (message-driven bean). The EJB does not support nesting of components, thus all components are primitive.

### 3.2. Architecture model

The EJB architecture does not support hierarchy. The EJB component is primitive, thus each component is the top-level component representing an application. The EJB component can reference another component's home interface (it does not represent an instances) used to obtain references to component instances, or it can reference an instance using relationship between bean instances managed by the EJB persistence manager.

### 3.3. Deployment and Runtime

The EJB component is defined in Deployment descriptor. The life space of EJB components is the EJB container. It provides all services (transaction, persistence and security), life-cycle managers of all component categories, and tools for deployment. Since a component is deployed, it is registered in the container's naming space, and clients can access its home interface to create and/or find component instances.

## 4. Enhanced EJB Component Architecture

Our intention is to enhance EJB with the features presented above. Here we present the main ideas of incorporation them.

### 4.1. Weaknesses of the Enterprise JavaBeans

Overall EJB component model is limited, because the EJB component is rather remote (possibly persistent) object with non-functional properties. It supports only one provided interface (called business interface), which implies no support for composed components. Also it is not possible to combine synchronous and asynchronous invocations. Each component can be implemented as one of the offered component categories (entity, sessions and message-drive), only. Developer cannot add new or modify features of component categories.

Architecture of the EJB is also very limited, due to limited underlying component model. The architecture model can be seen as a flat space, where a client can do (almost) anything. On the other hand, the dynamic architecture is supported, because a client can create and remove instances as he or she wishes. But he/she is not limited in any way. Since the EJB architecture does not work with instances (i.e., an application has to be explicitly created by a client), bindings are allowed between components, but not component instances.

### 4.2. Enhanced component

The role of a component in architecture is the basic building block, and architecture describes all components of an application, bindings between them. Component is described by nonfunctional properties and either architecture (composed component) or by its code (primitive component). If we consider an application as offering component, which provides functionality to clients, we can separate the whole system into two parts - an application part and a client part.

**Component type:** Component type of original EJB component is given by both home and business interface regardless of assuming them as local or remote one. Since a component can provide and require a number of interfaces, we adapted component type definition to reflect it. It has to be noted that the home interfaces are not bounded with any instance and they are a part of component type definition. The next part of component type is the component category as defined below.

**Component category:** Our intention is to support the properties-like implementation pattern, because it is more abstract, clearer and straightforward. The second implementation is to select basic properties defining component category. The properties are reflected in EJB as a bean-type (e.g., state-management, life-cycle management), and/or non-functional-properties of a component (i.e., transaction, security, persistence). There is a new non-functional property Concurrence, defining how concurrent calls from different clients are handled. This property contains reentrant property of EJB specification, and it adds sharing policy (i.e., multiple clients/single client access), and thread policy (i.e. concurrent execution of component code).

We define compatibility of component categories as full equality of the state management, concurrence and transaction properties. The reason to exclude lifecycle management and security is that they have rather influence on performance and on runtime configuration. The persistence was excluded since it represents internal state of the component.

**The Provides:** In our model the Provides are split into two classes - functional (interfaces) and value. The functional part is described as a set of both method-based and event-based interfaces. Primitive component's provided interfaces are directly mapped on implementation objects. Composed component's provided interfaces are delegated on internal components. In our enhanced EJB mode, provided interfaces are offered as single, an array of them or shared. Since clients cannot share one provided interface (but an instance can be shared), a mechanism has to be introduced to allow sharing of interfaces. It can be achieved by specifying an interface as an array.

The value part provides a set of scalar values of different types (e.g., string, integer, float, structure...). This abstraction is introduced as a part of requires. The intention of this kind of provides is to share and/or provide data for other components (e.g., references to datasource, URLs...) without binding and/or invocations on other components.

All provides will be mapped into a naming context that servers for both provides and requires.

On the provide side, it has specified structure - home subcontext stores all home interfaces (i.e., interfaces, that can be call without an instance), business subcontext stores all business interfaces (i.e., interfaces, that has to be called on an instance) and property subcontext stores all properties.

**Require interfaces:** Similarly to the Provides, the Requires are mapped into a naming context. In the naming context there are following types of objects: a) reference to required method-based interface; b) reference to required message queues for message-based interface; and c) property. The first two types (a,b) represent required functionality from other components. Type c is a representation of a value property (e.g., username, password, URL). The structure of the naming context is similar to the provide naming space - home subcontext contains references to provided home interfaces, business subcontext contains references to provided business interfaces of component instances and property subcontext contains required properties.

### 4.3. Enhanced Architecture

In previous subsections, the EJB component model was enhanced in functional direction. In this section we present enhancements of the structure. We intend to support both static and dynamic architectures. The static one is already supported in the EJB, we clarify its employment in the enhanced component model only. The dynamic architecture is new for the EJB, and it will be specified below.

There are two basic component styles - composed and primitive. Composed component is built of another components and it is described by architecture, while primitive component is indivisible. Composed components have no functional code, thus their role is architectural only. The roles we employ architecture are: composed components description and binding between components.

Since composed components do not contain any business functionality, they are used to capture logical structure. Of course, it has to implement home interface, where creation of the instance is coded. A composed component is built of a number of both primitive and composed components, and it defines a binding between them, delegation between parent's provides and internal component's provides, and subsuming between parent's requires and internal component's requires. Home interfaces of composed components are static interfaces and they are not bound to any instance. On the other hand, home interfaces of internal components delegated to parent's provide interfaces are not static, since they do not server to create an instance of parent component but they serve to create an instance of internal components.

**Static architecture** Static architecture we characterize as non-changeable during runtime - no ties are added, removed or changed, and no component instances are created or removed. This definition follows the original EJB specification of component (primitive) environment. As defined in the previous paragraph, the bindings can be between component and component instances. The component instance is created during deployment - to create it, we put a Java creation code into the deployment descriptor. We use Java for creation code specification, because it is the easiest and clearest way to describe it (execution of the code is guarded by Java security policies). Using this mechanism we can also create instance of application component, whose reference will be stored into client's naming context

**Dynamic architecture** we characterize as changeable during runtime - ties can be added, removed or changed, component instances can be created or removed - we allow all these changes. When a new component instance is created, provides and requires interfaces have to be successfully bounded, and then component's initialization code is executed. From this point the instance can be utilized. The problem is "how to specify binding of an instance". We decided to use a Java bind code, because it has greater expressive power than other languages. An example of such code is presented at Figure 6. It has to be noted, that we do not allow removing or changing of "static" instances (i.e., they are not created by any client) and/or "static" bindings.

### 4.4. Deployment and Runtime

At deployment phase component's code is inserted into runtime environment (EJB container) and the component provides and requires interfaces are mapped into naming context, but they are not resolved except home interfaces of application components (they are used to create instances). If an application component is required to be instantiated before the runtime phase begins, the appropriate Java creation code is executed and returned reference is mapped into a naming context.

During the runtime phase, clients are accessing naming context to obtain references to home interfaces or provided interfaces of instances. If an invocation on internal component home is invoked, then the new component instance is created, and Java bind code is executed (if it was specified). The runtime monitors if a component instances, that were instantiated automatically at the deployment are not removed, because an initial architecture has to be always a part of currently modified architecture.

## 5. Summary

This paper presents an enhancement of Enterprise JaveBeans component architecture. The approach is based on results from our component architecture SOFA/DCUP and on our experience with EJB. We enhanced component model by multiple interfaces, better component type specification (namely provides and requires), and we enhanced architecture model by composed component and dynamic architecture specification. There are problems not cover in this paper, and they are planned for future work. The most important problems are: a) persistence of composed component, b) compatibility of types of persistent composed components. Currently these enhancements have been implemented into JOnAS application server as a part of the PEPiTA project.

## 6. Acknowledgements

This work was partially supported by the Grant Agency of the Academy of Sciences of the Czech Repulic (project number A2030902), the Grant Agency of the Czech Republic (project number 201/99/0244), the PEPiTA/ITEA project (project number 2033).

## References

[1] Issarny, V., Bidan, C., Saridakis, T.: Achieving Middleware Customization in a Configuration-Based Development Environment: Experience with the Aster Prototype. In Proceedings of ICCDS '98, 1998, http://www.irisa.fr/solidor/work/aster.html.

[2] Jackson, M.: Software Requirements & Specifications: A Lexicon of Practice, Principles and Prejudices, ACM Press/Addison-Wesley, 1995.

[3] Luckham,D. C., Kenney, J. J., Augustin, L. M., Vera, J., Bryan, D., Mann, W.: Specification and Analysis of System Architecture Using Rapide. IEEE Transactions on Software Engineering, 21(4), 1995.

[4] Medvidovic, N., Taylor, R. N.: A Framework for Classifying and Comparing Architecture Description Languages. In Proceedings of the 6 th European Software Engineering Conference/5 th ACM SIGSOFT Symposium on the Foundations of Software Engineering, Zurich, Switzerland, 1997.

[5] OMG orbos/99-04-16, CORBA Component Model. Volume 1, 1999.

[6] Plasil, F., Balek, D., Janecek, R.: SOFA/DCUP Architecture for Component Trading and Dynamic Updating. In Proceedings of ICCDS '98, Annapolis, IEEE CS, 1998, pp. 43-52.

[7] Plasil, F., Besta, M., Visnovsky, S.: Bounding Component Behavior via Protocols. In Proceedings of TOOLS USA '99, Santa Barbara, USA, 1999.

[8] Purtilo, J. M.: The Polylith Software Bus. ACM Transactions on Programming Languages and Systems, 16(1), 1994.

[9] Sun Microsystems: JavaBeans 1.0 Specification. http://java.sun.com/beans/docs/spec.html.

[10] Sun Microsystems: Enterprise JavaBeans 2.0 final draft, http://java.sun.com/products/ejb/docs.html.

[11] Rogerson, D.: Inside COM. Microsoft Press 1997.

[12] D. Balek, F. Plasil: A Hierarchical Model of Software Connectors, TR 2000/2, Dept. of SW Engineering, Charles University, Prague

[13] Seco, J.C., Caires, L.: A Basic Model of Typed Components, TOOLS 2001

[14] Distributed Systems Research Group, EJB comparison, 2000, http://nenya.ms.mff.cuni.cz

[15] Oreizy, P., Issues in The Runtime Modification of Software Architectures, UCI-ICS-TR-96-35, 1996

[16] Oreizy, P., Medvidovic, N., Taylor, N.R., Architecture-Based Runtime Software Evolution, ICSE 1998

[17] Allan, R., Douence, R., Garlan, D., Specifying and Analyzing Software Architectures, Fundamental Approaches to Software Engineering 1998

[18] Jeff Magee and Jeff Kramer. Dynamic structure in software architectures. In Proceedings of the Fourth ACM SIGSOFT Symposium on the Foundations of Software Engineering, pages 3-14. ACM Press, 1996.

[19] Wermelinger, M., Fiadeiro, J.L., Algebraic Software Architecture Reconfiguration, Software Engineering-ESEC/FSE'99, volume 1687 of LNCS, pages 393-409. Springer-Verlag, 1999.

# Behavior Protocols and Component Lifecycle

*doktorand:*
Stanislav Visnovsky
Faculty of Mathamatics and Physics, Charles Univerity, Prague
visnovsky@nenya.ms.mff.cuni.cz

*školitel:*
Frantisek Plasil
Institute of Computer Science, Academy of Sciences of the Czech Republic, Prague
plasil@nenya.ms.mff.cuni.cz

obor studia:
I-2 Software systems

**Abstrakt**

In this paper, we employ description of component behavior into a lifecycle of hierarchical components. A description should allow for formal reasoning about the correctness of the specification refinement and also about the correctness of an implementation in terms of whether it adheres to the specification. As a proof of the concept, the behavior protocols used in the SOFA architecture description language are presented. We define bounded component behavior as a way to describe approximation of component behavior and protocol conformance relation to formalize specification refinement. Using these concepts, the designer can verify the adherence of a component's implementation to its specification at run time, while the correctness of refining the specification can be verified at design time.

## 1. Introduction

A lot of effort is being put into formal or semi-formal description of component semantics to help the development of the software systems based on the component concept. The target is to improve the specification by reducing the number of errors and by detecting errors earlier in the process.

In context of software architectures, a component is usually viewed as a black-box entity which provides and/or requires a set of services (accessed through interfaces). Components can be composed together by binding required to provided services to form a higher-level component. Typically, components and their compositions can be specified in ADL (Architecture Description Language [3]). For employing a behavior description of a component, the notation used has to strongly support description of the "call interplay" on the component's several interfaces, and to reflect step-by-step specification and refinement of the component during its design. Most of the approaches employ a black-box view of a component (no visible internals) together with a white-box view (all internals visible). However, to support a step-by-step specification, it is reasonable to include also specification of a "grey-box" view of the component (selected details visible only). In Wright language [1], the behavior of components and connectors is specified as a "computation", resp. a "glue", via a CSP-based notation (a system of recursive equations). In Wright, a component specification includes explicit behavior specification only for primitive components. As to composed components, their architecture behavior descriptions are generated via the composition operator, bottom-up, making the behavior description fully "white-box" based. Being strictly focused on design time, Wright does not address any behavior checks related to run time. In TRACTA [2], the behavior of a component is described in FSP (Finite State Processes), a formal vehicle similar to CSP in terms of being a system of recursive equations. A component specification includes explicit behavior specification only for primitive components. For composed components, TRACTA uses the same approach as Wright, i.e., generating an architecture process via the composition operator, bottom-up.

The goal of this paper is present how behavior protocols can support specification refinement in ADL and formal reasoning about the adherence of a component's implementation to the behavior specification and how the grey-box view can be employed in the process. In Section 2, we provide an overview of the SOFA component model and the component lifecycle. Section 3 introduces behavior of component, protocols and Section 4 formalizes compatibility of component behavior. Section 5 shows how the behavior compatibility can be used to support specification refinement. Further, in Section 6 an 7 we present this support for SOFA component model. Section 8 concludes the paper.

## 2. SOFA Components and Component Lifecycle

The SOFA (Software Appliances) project [4, 5] targets the issue of composing applications from components which can be deployed over a network. In the SOFA component model, an application is viewed as a hierarchy of nested software components. Analogously with the classical concept of object being an instance of a class, we introduce software component (component for short) as an instance of a component template. In principle, template can be interpreted as component type.

A template $T$ is a pair $< F, A >$ where $F$ is a template frame, and $A$ is a template architecture. The frame $F$ defines the set of individual (external) interfaces any component which is an instance of $T$ will possess. Basically, the frame $F$ reflects the black-box view on $T$. To support versioning, the frame $F$ can be implemented by more than one architecture. An architecture $A$ describes the structure of an implementation version of $F$ by (1) instantiating direct subcomponents of $A$ (those on the adjacent level of component nesting, subcomponents of $A$ for short), and by (2) specifying the subcomponents' interconnections via interface ties. Basically, the architecture $A$ reflects a particular grey-box view on the template $T$. An architecture can also be specified as primitive, which means that there are no subcomponents and its structure/implementation will be provided in an underlying implementation language, out of the scope of the component model.

A component's lifecycle is characterized by (potentially repeated) sequence of design time and run time phases. In a more detailed view, a design time phase is composed of the following design stages: development and provision, assembly, and deployment.

At the development and provision stage, a component is specified by its frame and potentially several architectures, each of them being a design version of the frame. For instance, the frame $F_{Main}$ is implemented by three different architectures: $A_1$, $A_2$, and $A_3$. While $A_1$ and $A_3$ are primitive, $A_2$ is composed of two subcomponents $Sub_1$ and $Sub_2$; these subcomponents are visible in $A_2$ only at the level of their frames $F_{Sub_1}$, $F_{Sub_2}$. It is important to emphasize that the actual specification of an architecture $A$ is always based on the frames of $A$'s subcomponents (and not on the architecture of those subcomponents). Reflecting top-down design, the specification of an application is factored this way into a hierarchy of alternating layers frame – architecture – frame – ..., forming a tree with nodes alternately of the frame and architecture types.

At the assembly stage of design time, the executable form of an application/component is determined by selecting an implementation architecture for each frame. In our example, this process starts at $F_{Main}$ by choosing one particular template $< F_{Main}, A_i >$. If $A_i$ is not primitive, the selection is applied recursively to all frames involved in $A_i$. Consequently, the executable form of the application/component is primarily based on all the primitive architectures involved recursively in the reduced subtree of $A_i$.

The executable is then deployed during the deployment stage, when the component run time configuration has to be devised. In particular, this includes distribution over computer network nodes and setting some of the component property parameters. At the end of the deployment, the component is ready to run.

## 3. Component Behavior and its Description

In this section we present a model of component behavior. By activity of a component $C$ we understand the finite sequence of actions which $C$ exhibits. The set of all possible activities of $C$ is referred to as the behavior of $C$. By convention, a sequence of actions is expressed as a trace of $C$; the trace is a sequence of action tokens, each of them representing exactly one action. The behavior of a component $C$ is represented as a set of words over $ACTs$ (the set of all action tokens) forming the language of $C$. By $L_A$ we denote the language of $C$ on all its interfaces. All the components connected to the external interfaces of $C$ form the environment $E$ of $C$. For a given set of interfaces $V$ (internal or external), the action tokens representing the actions on all $C$'s connections in $V$ form the alphabet $V_S$; $V_S = V_{S_{ext}} \cup V_{S_{int}}$, where $V_{S_{ext}}$ (resp. $V_{S_{int}}$) is the external alphabet (resp. internal alphabet) of $C$ defined as the $S$ set of all action tokens on $C$'s external (resp. internal) interfaces in $V$. Always, $V_{S_{ext}} = V_{S_{input}} \cup V_{S_{output}}$, where $V_{S_{input}}$ (input alphabet) is the set of enforced action tokens - the order in which its elements appear in the traces of $C$ is determined ("dictated") by $E$. On the contrary, the element of $V_{S_{output}}$ (output alphabet) appear in the traces in an order dictated by $C$. For simplicity, $V_{S_{input}}$, $V_{S_{output}}$ and $V_{S_{int}}$ are assumed to be disjoint.

The language of a component is typically an infinite. We assume that for each component there is a behavior description called "protocol". A protocol $Prot$ generates a set of traces $L(Prot)$.

## 4. Component Substition and Behavior Compliance

Assume a component $B$ with the behavior $L_B$ cooperating with its environment $E$ via an external alphabet $S_{ext}$. To capture the "dictated part" of a trace $t \in L_B$ , we say $t$ is with input $i$ if $t/S_{input} = i$, and the set $L_B/S_{input}$ forms the inputs of $B$ over $S_{ext}$. In a similar vein, we say that $t$ is with output $o$ if $t/S_{output} = o$ and the set $L_B/S_{output}$ forms the outputs of $B$ over $S_{ext}$.

With the intention to formally capture functional similarity of two components, consider their substitution: Should a component $A$ replace $B$ in $E$ ($A$ being put into the $B$'s environment $E$ by taking over all its tied external interfaces), $A$ needs to have the same $S_{ext}$ as $B$. Moreover, $A$ has to accept the inputs of $B$ as they are dictated by $E$; therefore we ask (1) $L_B/S_{input} \subset L_A/S_{input}$. For a given input $i$ of $B$, the environment $E$ expects a trace from $L_B/S_{ext}$ with the input $i$. By asking these traces to be also in $L_B/S_{ext}$ (i.e. (2) $L_B/S_{input}|S_{input}|L_A/S_{ext} \subset L_B/S_{ext}$) we do not allow $A$ to produce any other output for a given input $i$ except of those of $B$. Therefore, if both (1) and (2) hold, then, for every input $i$ of $B$, there exists at least one trace of $A$ with the input $i$. In this case, we conclude $A$ can replace $B$, since $E$ can dictate the same inputs it could for $A$, and will get outputs which could have been produced by $B$. If (1) and (2) holds, we say that a language $L_A$ is compliant with $L_B$, formally:

*Definition:* Let $B$ be a component, $L_B$ its behavior and $S_{ext}$ its external alphabet in an environment $E$. Also, let $A$ be another component, $L_A$ its behavior and $S_{ext}$ be also its external alphabet. We say that a language $L_A \subset ACTs*$ is compliant with $L_B \subset ACTs*$ on an alphabet $S_{ext} \subset ACTs$ if

1. $L_B/S_{input} \subset L_A/S_{input}$
2. $L_B/S_{input}|S_{input}|L_A/S_{ext} \subset L_B = S_{ext}$.

Similarly, compliance can be defined for two protocols by means of the languages they define. Furthermore, behavior compliance can be used for specifying a behavior of a component $A$, which cannot be directly described by a protocol. Our approach to the issue is based on $L_A$'s approximation, bounding, via a protocol determining a behavior "close enough" to $L_A$.

*Definition:* Let $Prot$ be a protocol, $A$ a component and $S$ its alphabet. We say $L_A$ is bounded by $L(Prot)$ on $S$, if $L_A/S$ is compliant with $L(Prot)$ on $S$.

In summary, if the behavior of a component $A$ is bounded by a protocol $Prot$, $A$ has to "react" to any input $i \in L(Prot)/S_{input}$, by at least one trace $t \in L_A/S$ such that $t/S_{input} = i$ and $t \in L(Prot)/S$ (narrower behavior of $A$ than $L(Prot)$). Even though the behavior of $A$ is "limited" by $Prot$, the condition (1) indicates that there can be protocol-neutral behavior of $A$, i.e., there can be a trace $t' \in L_A$, $t'/S_{input} \in L_A/S_{input}$ but $t'/S_{input} \notin L(Prot)/S_{input}$. The $A$'s activity represented by $t'$ is not limited by $Prot$ in any way.

## 5. Protocol-Based System Design

In this section, we address the issue of how a specification of components' hierarchy can be assisted by protocols. Typically, at early design stages, no component implementation is available, since an implementation should reflect the desired behavior of the component being designed. Thus, behavior specification is where to start. By employing protocols for this purpose, we can, at first, design the system as a collection of components with a hypothetical implementation, but well specified (a) behavior via protocols and (b) external alphabets. This idea leads to the following concept:

*Definition:* Let $Prot$ be a protocol and $S \subset ACTs$ an alphabet. A component $A$ with the alphabet $S$ such that $L_A = L(Prot)$ is called a model component of $Prot$ with $S$.

Thus, a system can be specified as a collection of cooperating model components, each of them with the behavior specified by a certain protocol and an alphabet $S$ reflecting an agreement for the components cooperation with its environment in the system. Via top-down design, the top model component is replaced by another, refined composed model component (with internal components at the next level of nesting) of an protocol which includes behavioral specification of the interplay of the internal components. Such refinement is recursively repeated until the low-level primitive model components are specified.

At any time during this process, a model component $B$ can be substituted by a composed model component $A$ only if the behavior of the composed component complies with the behavior of $B$.

After being composed top-down of model components, the system can be further elaborated by replacing a model component $M$ of a protocol $Prot$ by a component $A$ which is a "real, close enough" implementation of $M$ (and has the same external alphabet). Such a replacement does not destroy the bounded behavior of the components at the higher levels in the hierarchy (relatively to $M$) if this "close enough" means the behavior of $A$ is also bounded by $Prot$. By induction this can be generalized for any level of a component hierarchy if a primitive component is replaced:

*Theorem 1:* Let $A$, a component in a hierarchy, be bounded by a protocol $Prot_A$ on $S_{ext}$. Let a primitive component $B$ be a subcomponent of $A$ at any level of nesting and, at the same time, $B$ be a model component of a protocol $Prot_B$ with $S_{ext,B}$. If $B$ is substituted by a component $B'$ and the $B'$ behavior is bounded by $Prot_B$ on $S_{ext,B}$, $A$ becomes the component $A'$ and the behavior of $A'$ is also bounded by $Prot_A$ on $S_{ext}$.

As a direct consequence, replacement of all primitive components in the hierarchy preserves the structure and, in particular, bounding of behavior at higher level nodes of the component hierarchy. This is of utmost practical importance: A bottom-up elaboration of a hierarchy can be done as the replacement of the primitive model components by components with a "real implementation", since such a replacement induces an implicit elaboration of the components at higher levels in the hierarchy while preserving bounding of behavior of these higher-level components.

For runtime behavior evaluation, we introduce the concept of protocol obeying:

*Definition:* Let $Prot$ be a protocol, $A$ a component, $E$ its environment and $S_{ext}$ its external alphabet in $E$. Let $E$ dictate to $A$ inputs from $L(Prot)/S_{input}$ only. The component $A$ and the environment

$E$ obey a protocol $Prot$ if every trace of $A$ in $E$ is in $L(Prot)$.

The definition can be used practically for run-time checks of a system to identify communication not captured by a protocol, since if the environment dictates to $A$ inputs from $L(Prot)$ and behavior of $A$ is bounded, then it has to obey $Prot$.

## 6. Associating Behavior Protocols and SOFA Components

In SOFA, the behavior of a component template $T =< F, A >$ is described by means of behavior protocols [6]. Behavior protocols can describe regular languages in regular expression-like notation. Protocols are employed on two levels of abstraction. A frame protocol of $T$ is a protocol specifying the acceptable interplay of method invocations on the provides-interfaces and reactions on the requires-interfaces of the frame. An architecture protocol of $T$ is a protocol describing the grey-box behavior of $T$. It is based on the frames of the direct subcomponents specified in $A$. The protocol describes the interplay of the method invocations on the interfaces of $F$ and the outmost interfaces of the subcomponents in $A$. It is not specified in CDL directly, but it is generated by the CDL compiler by combining the frame protocols of the subcomponents via the composition operator [6].

Intuitively, in a template $T =< F, A >$, the architecture protocol of $A$ should follow the design intentions embodied in the frame protocol of $F$. Basically, employing protocol compliance is a natural way to reflect the desired correspondence in behavior description. However, the protocols associated with CDL incorporate interfaces at different levels of nesting which imply that reasoning on these protocols may require name unification. We denote such protocol $Prot_F$ unified against $A$ as ${}^{A}Prot_F$ and define CDL protocol conformance as follows:

*Definition:* Let $T =< F, A >$ be a template with the frame protocol $P_F$ and the architecture protocol $P_A$. We say that the architecture protocol $P_A$ conforms to the frame protocol $P_F$ if $P_A$ is compliant with ${}^{A}P_F$ on $S$ where $S$ is the alphabet associated with $F$.

## 7. Benefitting from Behavior Protocols

Behavior protocols can contribute to the correctness of a component design as follows: At the assembly design stage, an application is composed as a hierarchy of components; i.e. nested template instances. If these templates were associated with protocols during design in an ADL, as illustrated in Section 5, building up this hierarchy top-down can be viewed as a systematic, top-down composition/refinement of the model components of these protocols. Specific to SOFA, a model component of a frame protocol is replaced by a model component of an architecture protocol involving some internal frame protocols' components. These internal components get further replaced by some architecture protocol components, etc. As follows from Theorem 1, provided the frame – architecture – frame... protocols conformance has been successfully validated in such a model component hierarchy, the behavior of any component $C$ in the hierarchy is bounded by its frame protocol if the behavior of the primitive components recursively nested in $C$ is bounded by their frame protocols. The frame – architecture – frame ... protocols conformance can be advantageously verified beforehand at the development and provision stage by checking the frame- architecture protocol conformance for each template specification.

Since there are no means to verify bounding behavior of a primitive component by its frame protocol statically, it has to be done at runtime by applying the protocol obeying concept; this means checking whether in a particular run the component its trace does not violate the protocol (and, consequently, whether the component does not violate the bounding). A primitive component $C$, assumed to obey a protocol $Prot$, is tested whether the trace of $C$ is in $L(Prot)$. If this is not the case, it can be so for one of the following reasons: (1) the input dictated by $C$'s environment is not in the inputs of $L(Prot)$ so that the violation is caused by an "incorrect" environment; (2) $C$'s behavior violates the bounding by $Prot$. Of course, similarly to testing, a run time check cannot ensure "full" correctness of the implementation. However, both the protocol guard construction and the obeying validation can be done algorithmically, and, in general, at any level of component nesting (not only for primitive components).

## 8. Conclusion

The paper presents how a behavior protocol-based description can be applied to hierarchical software components. The protocol compliance concept provides formal means for capturing component substitutability and adhering of a component implementation to the behavior specification via a protocol.

As a proof of the concept, we use behavior protocols built into the SOFA CDL language. The description at multiple abstraction levels (frame, and architecture protocols) supports the refinement design process, allowing one to reason about component behavior at different levels of information hiding. To target refinement correctness, the frame, and architecture protocols in a particular template are tied together by the protocol conformance relation based on the protocol compliance concept. The verification of protocol conformance can be done at design time at these abstraction levels which effectively factors the state space inherent to the verification. Moreover, by intercepting method invocations, it is possible to check whether a particular component implementation obeys the component's behavior specification at run time.

## 9. Acknowledgements

This work was partially supported by the Grant Agency of the Academy of Sciences of the Czech Republic (project number A2030902), the Grant Agency of the Czech Republic (project number 201/99/0244), the PEPiTA/ITEA project (the Eureka project no. 2033).

## References

[1] R.J. Allen, "A Formal Approach to Software Architecture", *doctoral dissertation, School of Computer Science, Carnegie Mellon University*, Pittsburgh, 1997.

[2] D. Giannakopoulou, "Model Checking for Concurrent Software Architectures", *doctoral dissertation, Imperial College, University of London*, Jan.1999.

[3] N. Medvedovic, R.N. Taylor, "A Classification and Comparison Framework for Software Architecture Description Languages", *TR UCI-ICS-97-02, Department of Information and Computer Science, University of California, Irvine*, Feb. 1997.

[4] F. Plasil, D. Balek, R. Janecek, "SOFA/DCUP Architecture for Component Trading and Dynamic Updating", *Proceedings of the ICCDS '98, Annapolis, IEEE Computer Soc.Press*, 1998, pp. 43-52.

[5] SOFA project, http://nenya.ms.mff.cuni.cz/thegroup/SOFA/sofa.html.

[6] F. Plasil, S. Visnovsky, M. Besta, "Behavior Protocols", *TR 7/2000, Charles University*, Prague, 2000.

# Překladač numerických funkcí zapsaných v C do Fortranu77

*doktorand:*
EMIL KOTRČ
pokoj 425, blok 11 Koleje Strahov, Vaníčova 5, Praha 6
kotrc@cs.cas.cz

*školitel:*
RNDR. PETR SAVICKÝ, CSC.
ÚI AV ČR, Pod Vodárenskou věží 2, Praha ?
savicky@cs.cas.cz

obor studia:
Matematické inženýrství

**Abstrakt**

Cílem této práce je implementovat překladač z podmnožiny jazyka C do jazyka Fortran77. Takový překladač by se mohl používat jako preprocesor optimalizačního systému UFO, takže by s jeho pomocí bylo možné optimalizovat i funkce, které jsou napsány v jazyce C. UFO totiž požaduje na vstupu jazyk Fortran77.

## 1. Úvod

Existují funkce, týkající se rozličných problematik, které jsou napsány v jazyce C, a které by bylo vhodné pomocí systému UFO optimalizovat. V podstatě jedinou možností jak toto provést, je pokusit se přeložit zdrojový text v jazyce C do jazyka Fortran77. Vstupem pro UFO totiž může být pouze zdrojový text v jazyce Fortran77, který je případně možné doplnit o speciální makra. ( UFO má ještě i jiné možnosti vstupu, například pomocí dialogu, ale tyto možnosti se netýkají naší problematiky ).

Zde se tedy budu velice stručně věnovat implementaci takového překladače a omezeními, kterými jsem vstupní jazyk zatížil.

## 2. Vstupní jazyk

V tomto odstavci bych se krátce věnoval vstupnímu jazyku. Jak jsem již v úvodu psal, vstupním jazykem je pouze podmnožina jazyka C. Implementovat překladač z jazyka C do Fortranu77 je v podstatě nemožné. Důvodem jsou některé konstrukce jazyka C, které nemají v jazyku Fortran77 ekvivalent. Například obtížné by bylo překládat strukturované typy nebo práci s dynamickou pamětí.

Protože jazyk Fortran je v podstatě velice jednoduchým jazykem a složitější kontrukce jazyka C by tak nebylo možné překládat, navrhl jsem vhodná omezení vstupního jazyka. Je ovšem důležité zdůraznit, že veškerá omezení jsou zcela v souladu s požadavky, které jsou na program kladeny. Vstupem pro tento program je nějaká numerická funkce, která je zapsána v jazyce C. V takovém případě vlastně odpadá používání strukturovaných typů i práce s dynamickou pamětí ( alokace a dealokace ) a jiné těžko přeložitelné konstrukce. Co však zůstává, je možnost používat pointerovou aritmetiku jazyka C. Tato vlastnost vstupního jazyka je totiž často využívána. Ovšem v jazyce Fortran77 opět nemáme žádný ekvivalent, což působí značné problémy. Ukazatele, které se ve vstupním souboru používají, se tak mohou jedině překládat na indexy do nějakého pole.

Příklad, kdy se taková konstrukce použije je následující. Jako parametr překládané funkce je ukazatel na nějaké pole. Dále ve funkci je deklarován ukazatel, který se inicializuje tak, aby ukazoval na nějakou část odkazovaného pole. Pak už můžeme pracovat s konkrétní částí pole pouze pomocí ukazatele. Toto ale není do Fortranu77 normálně přeložitelné. Parametrem funkce samozřejmě může být i proměnná typu pole, ve funkci dokonce ani nemusí být znám počet prvků takového pole. Tato konstrukce se proto přeloží jednoduše. Jakmile je ale někde deklarována proměnná typu ukazatel, vznikne problém, protože proměnné takového typu nejsou ve Fortranu77 možné. Ukazatel se proto musí přeložit jako index do pole. Jinými slovy, musí se vlastně převést na proměnnou celočíselného typu. Přiřazovací příkaz, který inicializuje deklarovaný ukazatel, se tak přeloží jako přiřazení vhodného čísla do indexové proměnné. Kdykoliv se pak dále pracuje s onou konkrétní částí pole prostřednictvím ukazatele, musí se takové konstrukce převést na práci s původním polem, kde ovšem se musíme pomocí indexové proměnné ( zastupující ukazatel ) přesunout do té části, s níž se právě pracuje. Zatím bohužel tato možnost překladu není možná, protože se nejedná o zcela triviální záležitost, ale na implementaci pracuji.

Další omezení, která jsem přidal do definice vstupního jazyk, se týkají některých příkazů. Jiné příkazy lze ale překládat bez omezení. Takovým příkazem je příkaz `if`. Ten je podporován v celém rozsahu, tedy podporována je i sekce `else`. Přiřazovací příkaz lze také přeložit bez omezení, a to včetně speciálních operátorů jako například `+=`. Ovšem příkaz `for` už jednoduše přeložit nelze. V jazyce C je tento příkaz velice mocný a všestranný, na druhou stranu, jeho ekvivalent v jazyce Fortran, příkaz `DO`, je poměrně jednoduchý. Cyklus `DO` ve Fortranu totiž může používat jako proměnnou cyklu pouze proměnnou celočíselného typu. Zde nebyla jiná možnost, než přizpůsobit cyklus `for` jazyka C tak, aby byl přeložitelný. Ovšem toto omezení opět není příliš radikální ( z našeho pohledu ), neboť potřebám, pro které je program psán, takový omezený cyklus `for` zcela postačuje. Nyní, jak je tedy příkaz `for` omezen. Normální syntaxe tohoto příkazu je následující
`for(výraz-start; výraz-stop; výraz-iter) příkaz`
Upravená, nebo zjednodušená syntaxe, vypadá takto
`for( i = výraz-start; i <= výraz-stop; výraz-iter) příkaz`
kde `výraz-iter` je typu `i++` nebo `i += výraz`
Takto upravený cyklus `for` se již na cyklus `DO` překládá poměrně snadno.

Omezení, která jsem výše popsal, nejsou zdaleka všechna. Například do jisté míry jsou omezeny i výrazy. Není podporován operátor *čárka* či ternární operátor `?:`, dále není implementován příkaz `switch`, ani zbývající cykly `while` a `do-while`. Omezeny jsou i deklarace, možné je deklarovat pouze jednoduché typy. Jak jsem ale již psal, navržená omezení jsou v souladu s požadavky na program.

### 3. Implementace

Nyní velice stručně popíši implementaci samotného překladače. Je známo, že překladače, i přes různou strukturu, se obecně skládají ze čtyř základních částí.

(1) Lexikální analýza,
(2) syntaktická analýza,
(3) zpracování sémantiky,
(4) generátor kódu.

Lexikální analyzátor je vlastně jednoduchý konečný automat, který rozpoznává symboly vstupního jazyka. V našem případě rozpoznává lexikální symboly ( *tokens* ) jazyka C. Lexikálními symboly se rozumí především identifikátory, klíčová slova, čísla, oddělovače a operátory. Drobný problém vzniká se standardními matematickými funkcemi. Jak známo, žádné takové funkce nejsou součástí jazyka C. Ovšem v jazyce Fortran tyto funkce součástí jsou. Proto jsem se rozhodl, že budu všechny standardní matematické funkce ( deklarované v souboru `math.h` ) chápat jako součást vstupního jazyka. Tím odpadá procházení hlavičkových souborů a generování je poté snadné. Tedy identifikátory jako například `sin` jsou v mém programu rozpoznávány jako klíčová slova.

Lexikálním analyzátorem nalezené lexikální symboly jsou vstupem pro syntaktický analyzátor. Ten je napsán objektově v jazyce C++. Použitá metoda je známá jako metoda rekurzivního sestupu (*recursive descent*), která spočívá v tom, že se každému neterminálnímu symbolu přiřadí jedna procedura či funkce, která jej zpracovává. Já jsem tuto metodu poněkud zobecnil, a každému neterminálnímu symbolu ( který je podporován ) jsem přiřadil jednu celou třídu v jazyce C++. Důsledkem takové implementace je velká jednoduchost, přehlednost a snadná rozšiřitelnost celého programu. V každé třídě tak jsou metody pro syntaktickou analýzu, pro generování kódu a případně i pro zpracování sémantiky. Tříd v programu je samozřejmě poměrně velké množství, ale všechny jsou odvozeny od základní třídy `ident`, která je vlastně schopna zpracovávat pouze jednoduché neterminální symboly, jako čísla, identifikátory či operátory.

Další částí překladače bývá zpracování sémantiky, kde se provádí některé doplňující kontroly. Například zda souhlasí skutečné parametry s formálními, apod. Zpracování sémantiky v mém programu je ale velice triviální, neboť předpokládám, že na vstupu je správně zapsaný program v jazyce C, a tudíž nějaká hlubší analýza není nutná. Proto, co se týče zpracování sémantiky, obsahuje program pouze jednoduchou dvouúrovňovou tabulku symbolů. To znamená, že jsou možné pouze dva rozsahy platnosti ( *scope* ), a to buď lokální nebo globální. Vnořené bloky z jazyka C a strukturované typy totiž nejsou podporovány, takže není potřeba mít víceúrovňovou tabulku. Do tabulky symbolů se tak ukládají základní informace o deklarovaných identifikátorech, hlavně zda se jedná o funkci či proměnnou ( aby se mohlo rozlišit volání funkce ). Zároveň se do tabulky ukládají některé dodatečné informace, které se využijí v budoucnosti při překladu proměnných typu ukazatel, jak jsem již výše zmiňoval.

Poslední částí překladače je generátor kódu. V mém programu je generátor rozložen vlastně do mnoha metod různých tříd. Každá třída obsahuje metodu, která se o generování konkrétního neterminálního symbolu postará. Generování se provádí do jazyka Fortran77, kde ale musí mít výsledný soubor i jistou stanovenou formu. Například řádky ( kromě komentářů ) mohou začínat až sedmým znakem od počátku řádku. Dále, jestliže je některý řádek delší než 72 znaků, musí se rozdělit. A to takovým způsobem, že pokračování řádku se vyznačí nebílým znakem na šesté pozici od počátku řádku. Všechny tyto záležitosti provádí k tomu určený modul, který rovněž zajišťuje vhodné odsazování a doplňující komentáře.

## 4. Závěr

V této fázi vývoje program umožňuje překládat jednoduché zdrojové texty. Umí překládat výrazy a příkazy, o kterých jsem psal v první části. Zatím není možné překládat práci s poli pomocí ukazatelů, lze ale přeložit takové konstrukce, kdy se ve vstupním souboru přistupuje do polí pouze pomocí indexů. Jak jsem zmiňoval, není možné, aby program uměl přeložit libovolný zdrojový text v jazyce C, ale pracuji na tom, aby omezení vstupního jazyka bylo co nejméně a aby program vyhovoval kladeným požadavkům.

## References


[1] Emil Kotrč, “Překlad z jazyka Pascal do C++”, *Diplomová práce*, 2001.

[2] Jiří Vogel, “Programování v jazyku Fortran”, *SNTL*, 1976.

[3] Bořivoj Melichar, Milan Češka, Karel Ježek, Karel Richta, “Konstrukce překladačů”, *Vydavatelství ČVUT*, 1999

# Incremental structure learning of Wang neuro-fuzzy system

*doktorand:*
ING. DAVID COUFAL
Institute of Computer Science
coufal@cs.cas.cz

*školitel:*
DOC. ING. STANISLAV KREJCI, CSC.
University of Pardubice
krejci@upce.cz

obor studia:
Technical cybernetics

**Abstrakt**

In the paper there is presented original incremental structure learning algorithm for Wang neuro-fuzzy system.

## 1. Incremental approach

There are two standard approaches to structure learning of (neuro)-fuzzy systems [1]. It is an decremental (exhaustive search) approach and approaches based on fuzzy clustering techniques [2]. The main drawback of decremental approach is its computational complexity which make it inapplicable on high-dimensional data. On the other hand in fuzzy clustering approaches we have to chose in advance number of rulenodes of neuro-fuzzy systems which is the main disadvantage of this class of algorithms. These both drawbacks can be solved by employing incremental approach to structure learning. The idea is simple. On base of value of some criterion add rulenodes until chosen criterion indicates satisfaction of predefined constrain.

In this paper we give description of this type of structure learning for Wang neuro-fuzzy system.

### 1.1. Incremental structure learning of Wang NFS

Computation of class of neuro-fuzzy systems we are interested in is considered to be generally given as

$$o(\boldsymbol{x}) = \sum_{j=1}^{m} A_j(\boldsymbol{x}) \cdot c_j, \tag{39}$$

where $\boldsymbol{x} \in \mathcal{R}^n$ and $c_j \in \mathcal{R}$.

As a special case we have antecedents $A_j$ represented by Gaussians which gives Wang neuro-fuzzy system [3] acting as

$$o(\boldsymbol{x}) = \sum_{j=1}^{m} c_j \cdot \exp\left[-\sum_{i=1}^{n} \frac{(x_i - a_{ji})^2}{2b_{ji}^2}\right], \tag{40}$$

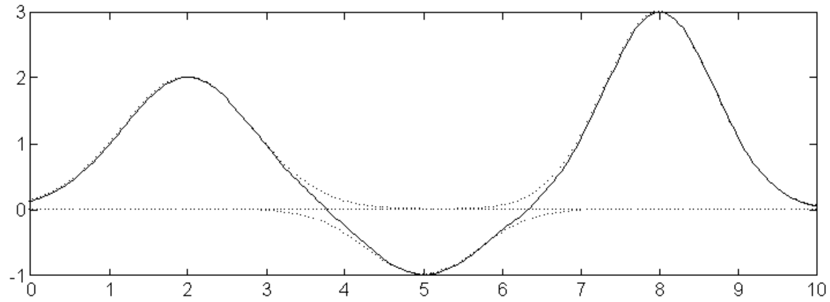
i.e., output is given by linear combination of Gaussians.

To present main idea of incremental structure learning we we will consider one-dimensional case, $n = 1$. The multidimensional one will be a straightforward extension presented later in the section.

In one-dimensional case, above equation has form

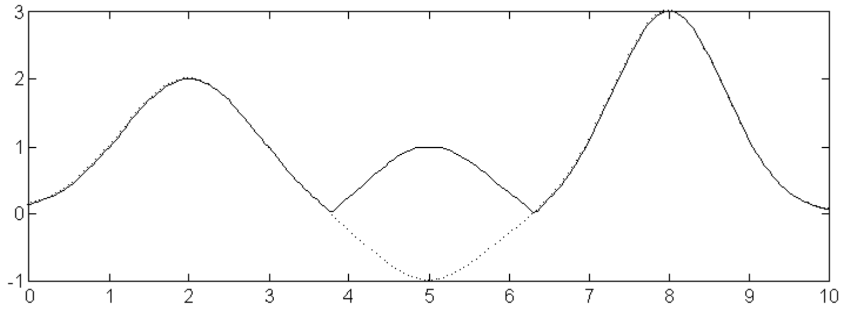
$$o(x) = \sum_{j=1}^{m} c_j \cdot \exp\left[-\frac{(x-a_j)^2}{2b_j^2}\right] = \sum_{j=1}^{m} c_j \cdot g_j(x, a_j, b_j), \tag{41}$$

where $g_j(x, a_j, b_j)$ denotes one-dimensional Gaussian with central point $a_j$ and width parameter $b_j$.




**Obrázek 9:** Sum of three Gaussians.

We proceed with graph given in Fig. 9. The graph is a graph of function of form (41) given by sum of three Gaussians. Particular Gaussians are presented here by dotted lines, their sum by solid one. Now a question is, how to determine particular Gaussians from the graph? Clearly, assuming small overlapping among Gaussians we can reconstruct them as the ones having central points at points where local maxima of graph's absolute value are reached. Width parameters can be then determined by local fitting of individual Gaussians to function graph. In Fig. 10, there is presented by dotted line graph of original function and by solid line its absolute value. We see that points at which local maxima are reached gives central points of Gaussians the function is formed from.




**Obrázek 10:** Absolute value of sum of three Gaussians.

The preceding example demonstrates the main idea of algorithm which is to set centers of rulenodes (here Gaussians) at points where graph of absolute value of learned function reaches its local maxima. Width parameters are then set on base of rulenodes membership functions' local fitting to learned function graph.

In a case of structural learning, we have at our disposal only numerical samples of learned function. From this reason we adopt in our algorithm cyclic process of local maxima identification. To describe this in details we will still consider one-dimensional case with set of samples given by training set $\mathcal{T} = \{(x_k, t_k)\}$.

We start description by formal renotation of $\mathcal{T} = \{(x_k, t_k)\}$ to $\mathcal{T}^1 = \{(x_k, t_k^1)\}$. Consider the first loop. It starts by identification of index $k_1^*$ for which maximum $\max_k \{|t_k^1|\}$[1], is reached, thus $k_1^* = \mathrm{argmax}_k \{|t_k^1|\}$. Regarding sample $(x_{k_1^*}, t_{k_1^*}^1)$, point $x_{k_1^*}$ then represents central point of the first identified Gaussian, i.e., $a_1 = x_{k_1^*}$ and $t_{k_1^*}^1$ gives value of $c_1$ parameter, i.e., $c_1 = t_{k_1^*}^1$. This is given by the fact that for point $x_{k_1^*}$ value of identified Gaussian is one, but value of learned function is $t_{k_1^*}^1$, so to have appropriate output of rulenode we are setting $c_1 = t_{k_1^*}^1$.

On base of parameters $a_1$ and $c_1$ Gaussian $c_1 \cdot g(x, a_1, b)$ is considered and value of its width parameter is found by one-dimensional minimization of suitable error function. Apparently, standard candidate for such a function is a sum of squares of particular errors

$$E_1(b) = \sum_{k=1}^{N} (t_k - \eta c_1 g_1(x_k, a_1, b))^2. \tag{42}$$

Actually, above formula is not generally standard sum of squares error function due to presence of $\eta \in \mathcal{R}$ parameter. We explain reasons for $\eta$ introduction later in the section, at this time will consider $\eta = 1$ which gives standard error function.

Optimal value of parameter $b$ found by (42) minimization is denoted by $b_1$. Setting of $b_1$ value finishes process of parameters setting for the first rulenode.

The last step of a loop is the update of $\mathcal{T}^1 = \{(x_k, t_k^1)\}$ to $\mathcal{T}^2 = \{(x_k, t_k^2)\}$ which is done according to formula

$$t_k^2 = t_k^1 - \eta c_1 g_1(x_k, a_1, b_1). \tag{43}$$

What is a rationale behind this update? Considering $\mathcal{T}^1$ as set of samples of some function $f_1$,

$$f_1(x) = \eta c_1 g_1(x, a_1, b_1) + \eta c_2 g_2(x, a_2, b_2) + \ldots + \eta c_m g_m(x, a_m, b_m), \tag{44}$$

we see, that above update gives $\mathcal{T}^2$ as set of samples of function $f_2$ of form

$$f_2(x) = \eta c_2 g_2(x, a_2, b_2) + \ldots + \eta c_m g_m(x, a_m, b_m). \tag{45}$$

Hence, update is actually subtraction of one member of linear combination the original function, characterized by set of samples $\mathcal{T}^1$, is assumed to represent. A remainder after subtraction gives new function, given also by sum of Gaussians, characterized by set of samples $\mathcal{T}^2$.

Formulation of $\mathcal{T}^2$ set finishes first loop of algorithm. The second loop consists of the same steps as the first one, the only difference is that we consider set $\mathcal{T}^2$ instead of $\mathcal{T}^1$. Other loops are then performed on $\mathcal{T}^j$ sets until all members of sum of original function are identified.

Regarding stopping of algorithm, in an ideal case it would be after $m$ loops where we will have $t_k^{m+1} = 0$ for all $k$, i.e., also $\max_k \{|t_k^{m+1}|\} = 0$. But this is not a real case because original function is not exactly given by linear combination of Gaussians but it is only approximated in this way. From this reason we stop algorithm not when $\max_k \{|t_k^{m+1}|\} = 0$ but already when $\max_k \{|t_k^{m+1}|\} \leq \tau |c_1|$ where $\tau$ is small number from $(0,1)$, e.g., $\tau = 0.2$. This finishes algorithm explanation.

In the following text we have to explain some other things. The first one is formulation of algorithm for multidimensional case, i.e., when $\mathcal{T} = \{(\boldsymbol{x}_k, t_k)\}$. We denote multidimensional Gaussian in short by $g(\boldsymbol{x}, \boldsymbol{a}, \boldsymbol{b})$. With respect to our algorithm, there are no problems with determination of $\boldsymbol{a}_j = (a_{j1}, \ldots, a_{jn})$ because these are given by points at which local maxima $\max_k \{|t_k^j|\}$ are found. Similarly for $c_j$. The other situation is for width parameters $\boldsymbol{b}_j = (b_{j1}, \ldots, b_{jn})$. To state their

[1] $|\cdot|$ is absolute value.

coordinates we should adopt in our algorithm multidimensional optimization of the following error function

$$E_j(\boldsymbol{b}) = \sum_{k=1}^{N} (t_k^j - \eta c_j g(\boldsymbol{x}_k, \boldsymbol{a}_j, \boldsymbol{b}))^2. \tag{46}$$

To preserve algorithm simple, together with the idea that outputs from structural learning are further tuned by parameter learning, we will consider $\boldsymbol{b}$ to have all coordinates equal in all dimensions, i.e., that $\boldsymbol{b} = (b, \ldots, b)$. Such a restriction on $\boldsymbol{b}$ transforms multidimensional Gaussian $g(\boldsymbol{x}, \boldsymbol{a}, \boldsymbol{b})$ to one-dimensional $g(\boldsymbol{x}, \boldsymbol{a}, b)$ with respect to width parameter. Hence, we can use to find $\boldsymbol{b} = (b, \ldots, b)$ values one-dimensional optimization of

$$E_j(b) = \sum_{k=1}^{N} (t_k^j - \eta c_j g(\boldsymbol{x}_k, \boldsymbol{a}_j, b))^2. \tag{47}$$

Regarding other steps of algorithm they are the same as in one-dimensional case. In Table 5 we get procedural transcription of whole algorithm for general multidimensional case.

Output of algorithm is given by $m$ Gaussians characterized by parameters $c_j, \boldsymbol{a}_j, \boldsymbol{b}_j$. These Gaussians then represents rulenodes of learned Wang neuro-fuzzy system.

denote $\mathcal{T} = \{(\boldsymbol{x}_k, t_k)\}$ as $\mathcal{T}^1 = \{(\boldsymbol{x}_k, t_k^1)\}$;<br>
denote $k_1^* = \mathrm{argmax}_k, \{|t_k^1|\}$; set $c_1 = t_{k_1^*}^1$;<br>
set $\tau \in (0,1)$; set $\eta = 1.001$; set $j = 1$;<br>
**while** $|c_j| > \tau |c_1|$ **do**<br>
  set $\boldsymbol{a}_j = \boldsymbol{x}_{k_j^*}$;<br>
  minimize $E_j(b)$ given by (47) w.r.t. $b$;<br>
  on base of found optimal $b$ state $\boldsymbol{b}_j = (b, \ldots, b)$;<br>
  update $\mathcal{T}^j$ to $\mathcal{T}^{j+1} = \{(\boldsymbol{x}_k, t_k^{j+1})\}$, where<br>
  $t_k^{j+1} = t_k^j - \eta c_j g(\boldsymbol{x}_k, \boldsymbol{a}_j, \boldsymbol{b}_j)$;<br>
  denote $k_{j+1}^* = \mathrm{argmax}_k, \{|t_k^{j+1}|\}$;<br>
  set $c_{j+1} = t_{k_{j+1}^*}^{j+1}$;<br>
  $j = j + 1$;<br>
**end**<br>
$m = j - 1$;

**Tabulka 5:** Additive structure learning algorithm.

To end the presentation we have to discuss the last thing which is a stating of bracketing triplet for minimization of error function (47). What does it mean? Standard one-dimensional optimization algorithmd such as golden search method [4] require as input parameter so called bracketing triplet. Considering minimization of continuos function $f(s)$, bracketing triplet is triplet of points $s_{left} < s_{mid} < s_{right}$ such that $f(s_{left}) > f(s_{mid}) < f(s_{right})$ holds. Stating such a triplet we have from continuity of $f$ assured that in interval $[s_{left}, s_{right}]$ there at most one local minimum of $f$.

## 2. Stating of bracketing triplet

To state bracketing triplet for error function (47) with respect to $b$ we start by discussion of its properties. The first remark is that it is a function defined on intervals $(-\infty, 0) \cup (0, +\infty)$ and its an even function which is given by fact that $b$ occurs in Gaussians, and therefore in $E_j(b)$, only in its second power.

The second observation is that although originally for $b = 0$, $E_j$ is not defined it can be defined here. This is given by the fact that at point $b = 0$ error function has discontinuity of the first order, hence we can continually define $E(0)$ by its limit at this point, i.e.,

$$E_j(0) = \lim_{b \to 0} E_j(b) = \sum_{k_p} (t^j_{k_p})^2 + \sum_{k_c} (t^j_{k_c} - \eta c_j)^2. \tag{48}$$

In this formula $k_p$ are indices of proper points of $\mathcal{T}^j$ which are the ones with $d^2(\boldsymbol{x}_{k_p}, \boldsymbol{a}_j) > 0$. Indices $k_c$ are indices of points coinciding with central point of Gaussian, i.e., these are points with $d^2(\boldsymbol{x}_{k_c}, \boldsymbol{a}_j) = 0$. In our algorithm we will consider error function enhanced on value of $b = 0$. That is, we will consider that $E(b)$ is defined for all $b \in \mathcal{R}$.

Let us now state the following assertion. For error function (47) there exists $b_{max} > 0$ such that inequality $E_j(2b_{max}) > E_j(b_{max})$ holds. This $b_{max}$ is given by

$$b_{max} = \sqrt{\frac{d^2_{max}}{2\ln(\eta)}}, \tag{49}$$

where $0 < d^2_{max} = \max_k\{d^2_k\}$, $d^2_k = \sum_i (x_{ki} - a_{ji})^2$.

**Proof:** We start by remainig two facts valid for Gaussian and by one lemma valid for inequalities. The first fact valid for Gaussians is that

$$\text{for fixed } d^2 \text{ and for } 0 < b_1 < b_2 \text{ inequality } g(b_1) < g(b_2) \text{ holds.} \tag{50}$$

The second fact is that

$$\text{for fixed } b \text{ and for } d_1^2 < d_2^2 \text{ inequality } g(d_2^2) < g(d_1^2) \text{ holds.} \tag{51}$$

The last fact we remain is that

$$\text{for } 0 \le x_1 < x_2 \text{ raising to second power retains inequality i.e., } x_1^2 < x_2^2. \tag{52}$$

Now, let $b_{max}$ be set in such way that for all $k$ and some $\eta > 0$

$$\eta|c|g_k(b_{max}) \ge |t_k| \tag{53}$$

inequality holds. In the following text we show that this assumpton implies $E(2b_{max}) > E(b_{max})$ inequality.

According to (50) we have $g_k(2b_{max}) > g_k(b_{max})$, for all $k$. Hence

$$\begin{aligned} g_k(2b_{max}) &> g_k(b_{max}), & (54)\\ \eta|c|g_k(2b_{max}) &> \eta|c|g_k(b_{max}), & (55)\\ \eta|c|g_k(2b_{max}) - |t_k| &> \eta|c|g_k(b_{max}) - |t_k|. & (56) \end{aligned}$$

Since from our assumption (53) we have $(\eta|c|g_k(b_{max}) - |t_k|) \ge 0$ we can last equation rewrite according to (52) as

$$(\eta|c|g_k(2b_{max}) - |t_k|)^2 > (\eta|c|g_k(b_{max}) - |t_k|)^2. \tag{57}$$

Left side of (57) can be written as

$$\eta^2 c^2 g_k^2(2b_{max}) - 2\eta|c|g_k(2b_{max})|t_k| + t_k^2 \tag{58}$$

which is equal to

$$(\eta c g_k(2b_{max}) - t_k)^2 + 2\eta c g_k(2b_{max}) t_k - 2\eta|c|g_k(2b_{max})|t_k|. \tag{59}$$

Similarly, rigth side of (57) can be rewritten as

$$(\eta c g_k(b_{max}) - t_k)^2 + 2\eta c g_k(b_{max}) t_k - 2\eta|c|g_k(2b_{max})|t_k|. \tag{60}$$

Denoting $E_k(2b_{max}) = (\eta c g_k(2b_{max}) - t_k)^2$ and $E_k(b_{max}) = (\eta c g_k(b_{max}) - t_k)^2$ we have (57) in form

$$E_k(2b_{max}) - E_k(b_{max}) >$$
$$2\eta|c|g_k(2b_{max})|t_k| - 2\eta c g_k(2b_{max})t_k + 2\eta c g_k(b_{max})t_k - 2\eta|c|g_k(2b_{max})|t_k| \quad (61)$$

Term (61) can be rewritten as

$$2\eta g_k(2b_{max})(|c||t_k| - ct_k) + 2\eta g_k(b_{max})(ct_k - |c||t_k|) \quad (62)$$

which is

$$2\eta(|ct_k| - ct_k)(g_k(2b_{max}) - g_k(2b_{max})). \quad (63)$$

Since $|x| - x \geq 0$ for all $x \in \mathcal{R}$ and $(g_k(2b_{max}) - g_k(2b_{max})) > 0$ we have term (63) $\geq 0$. That is, for all $k$ holds $E_k(2b_{max}) - E_k(b_{max}) > 0$ which gives, summing through $k$, $E(2b_{max}) - E(b_{max}) > 0$. Obviously, this is

$$E(2b_{max}) > E(b_{max}). \quad (64)$$

Now, we aim on task how to set $b_{max}$ to condition (53) holds. Considering $0 < d^2_{\max} = \max_k\{d^2_k\}$. Then setting $b_{max}$ in such a way that

$$\eta|c|g(d^2_{\max}, b_{max}) \geq |c| \quad (65)$$

solves he problem. This is due to fact that with respect to our algortihm we have $|c| \geq |t_k|$, see point 02 or 09 of Table 5. Clearly, (65) can be rewritten as

$$\eta g(d^2_{\max}, b_{max}) \geq 1 \quad (66)$$

Since for all $k$, $d^2_k \leq d^2_{\max}$ we have $g(d^2_k, b_{max}) \geq g(d^2_{\max}, b_{max})$ it is sufficient to consider only equality

$$\eta g(d^2_{\max}, b_{max}) = 1. \quad (67)$$

This gives for $b_{max}$ expression

$$\frac{d^2_{max}}{2b^2_{max}} = -\ln(\frac{1}{\eta}), \quad (68)$$

$$b_{max} = \sqrt{\frac{d^2_{max}}{2\ln(\eta)}}.\square \quad (69)$$

Since this assertion is crucial for setting bracketing triplet we see from (49) that $b_{max}$ can be defined only for $\eta > 1$. To minimize impact of $\eta$ on sum of squares error (47) we set $\eta$ value very close the one, we use $\eta = 1.001$ as it is done in Table 5.

On base of above assertion we can set bracketing triplet in the following way. Firstly, we compute values $E_j(0)$ and $E_j(b_{max})$. Now we have three cases possible.

For $E(0) < E(b_{max})$ we have $E(-b_{max}) > E(0) < E(b_{max})$ which gives bracketing triplet as $b_{left} = -b_{max}$, $b_{mid} = 0$, $b_{right} = b_{max}$.

For $E(0) > E(b_{max})$ then because we have $E(b_{max}) < E(2b_{max})$ we can set triplet as $b_{left} = 0$, $b_{mid} = b_{max}$, $b_{right} = 2b_{max}$.

In the case of equality $E(0) = E(b_{max})$ we have $E(b_{max}) < E(2b_{max})$ hence $E(0) < E(2b_{max})$ which is in fact the first case. Therefore we can set triplet as $b_{left} = -2b_{max}$, $b_{mid} = 0$, $b_{right} = 2b_{max}$.

Having bracketing triplet set minimization can be performed by some one-dimensional optimization algortihm [4].

By stating above formula we finish determination of all parameters to algorithm given in Table 5 can be used for incremental structure learning of Wang neuro-fuzzy system.

## 3. Conclusion

We end by short discussion of presented approach. The main idea assumption of incremental approach is to consider learned function as linear combination of Gaussians. Central points of these Gaussians are considered at local maxima and minima of learned function.

The main advantage of incremental learning is that in does not require in advance specification of rulenodes (Gaussians here) number. Computational extensity is here mainly driven by number of points in training set because particular distances $d^2$ have to be computed, but this is a case of all algorithms. Number of dimensions here have not so crucial effect as in decremental approaches.

## References


[1] Nauck D., Klawonn F., Kruse R. Foundations of Neuro-Fuzzy Systems, John Wiley & Sons, 1997

[2] Höppner F., et al., Fuzzy cluster analysis, John Wiley & Sons, 1999

[3] Wang L.X., A Course in Fuzzy Systems and Control, Prentice Hall, 1997

[4] Press W.H., Teukolsky S.A., Vetterling W.T, Flannery B.P., Numerical Recipes in C, The Art of Scientific Computing, Second Edition, Cambridge University Press, 1992; *internet version is available at* http://www.nr.com

# An interpretation of ZF in a fuzzy set theory

*doktorand:*
ZUZANA HANIKOVÁ
Ústav informatiky AV ČR, Pod Vodárenskou věží 2, 182 07 Praha 8
zuzana@cs.cas.cz

*školitel:*
PROF. RNDR. PETR HÁJEK, DRSC.
Ústav informatiky AV ČR, Pod Vodárenskou věží 2, 182 07 Praha 8
hajek@cs.cas.cz

obor studia:
Matematická logika

**Abstrakt**

A set theory over a many-valued logic in the style of [1] is presented by listing its axioms and discussing the spelling of some of them as this has been subject to non-trivial modification compared to ZF. In FST we define a class of hereditarily "crisp" sets, which we prove to be an inner model of ZF in FST.

This paper presents a first order theory in a language $\{\in\}$, over the logical system BL$\forall\Delta$; we presume $=$ is a logical symbol of BL$\forall\Delta$. For a detailed treatment of BL$\forall\Delta$ the reader should consult [1] and the paper [2] on introducing function symbols.

An analysis of the state of art respecting set theories in non-classical logics has been presented in [5]. The theory defined here, and referred to as FST ('fuzzy set theory'). has first been introduced in [4], together with a non-crisp model for FST, which guarantees that FST is distinct from ZF or its fragments in the classical logic.

## 1. The theory FST

The following axioms have been assembled using a class model with many-valued semantics, namely a BL$\Delta$-valued universe. All FST axioms are valid in this universe.

In the formulation of several of the axioms we use functions like $\emptyset$, $\{x\}$ or $x \cup y$, whose existence is guaranteed by some other axioms, as is common in ZF.

The reader will note that ours is a "crisp" $=$, i.e. admits only 0-1 interpretations in the (potential) models. This is forced by the fact that e.g. in Łukasiewicz and product logic, the axiom of separation together with equality axioms imply crispness of $=$ anyway. Moreover, under the "standard" formulation of extensionality, crispness of $=$ implies crispness of $\in$. These facts (which have been proved in [3]) force a reformulation of the extensionality axiom, and we take over the solution from [6].

The weak form of $\in$-induction we include only helps to interpret the classical $\in$-induction and does not play a crucial role. The original spelling, missing the $\Delta$'s, was not valid in the BL$\Delta$-valued universe.

The power set axiom has been weakened using $\Delta$, to the following form: for any set $x$ there is a "crisp power set" $z$, which contains all sets that are subsets of $x$ *in degree* 1.

**Definition 1** *FST is a first order theory in the language* $\{\in\}$, *with the following axioms:*
(i) (extensionality) $\forall x \forall y(x = y \equiv (\Delta(x \subseteq y)\,\&\,\Delta(y \subseteq x)))$
(ii) (empty set) $\exists x \Delta \forall y(y \notin x)$
(iii) (pair) $\forall x \forall y \exists z \Delta \forall u(u \in z \equiv (u = x \vee u = y))$
(iv) (union) $\forall x \exists z \Delta \forall u(u \in z \equiv \exists y(u \in y\,\&\,y \in x))$
(v) (crisp power) $\forall x \exists z \Delta \forall u(u \in z \equiv \Delta(u \subseteq x))$
(vi) (infinity)$\exists z \Delta(\emptyset \in z\,\&\,\forall x \in z(x \cup \{x\} \in z))$
(vii) (separation) $\forall x \exists z \Delta \forall u(u \in z \equiv (u \in x\,\&\,\varphi(u,x))$, for any formula not containing $z$ as a free variable
(viii) (collection) $\forall x[\forall u \in x \exists v\, \varphi(u,v) \to \exists z \Delta \forall u \in x \exists v \in z \varphi(u,v))]$ for any formula not containing $z$ as a free variable
(ix) (support) $\forall x \exists z(\mathrm{Crisp}(z)\,\&\,\Delta x \subseteq z))$,
(x) ($\in$-induction) $\Delta \forall x(\forall y \in x \varphi(y) \to \varphi(x)) \to \Delta \forall x(\varphi(x))$, for any FST-formula $\varphi$.

where $\mathrm{Crisp}(x) \equiv \forall y \Delta(x \in y \vee \neg x \in y)$.

## 2. Interpretation of ZF

In FST we define a class of hereditarily crisp sets. In the FST universe, these form a subuniverse on which membership is crisp, so that the law of the excluded middle holds in it (and so does all classical logic) and all axioms of ZF hold likewise.

**Definition 2 (Hereditarily crisp transitive set)** $HCT(x) \equiv \mathrm{Crisp}(x)\,\&\,\forall u \in x(\mathrm{Crisp}(u) \& u \subseteq x)$

**Lemma 8** $\mathrm{FST} \vdash \mathrm{Crisp}(x) \equiv \forall u \Delta(u \in x \to \Delta(u \in x))$

We say that $\varphi$ is crisp (or, more precisely, FST proves $\varphi$ to be crisp) iff $\mathrm{FST} \vdash \Delta\varphi \vee \neg\Delta\varphi$.

**Lemma 9 (Crispness of** Crisp**)** $\mathrm{FST} \vdash \mathrm{Crisp}(x) \to \Delta\mathrm{Crisp}(x)$

**Definition 3 (Hereditarily crisp set)** $H(x) \equiv \mathrm{Crisp}(x)\,\&\,\exists x' \in \mathrm{HCT}(x \subseteq x')$.

This is equivalent to $\ldots \Delta(x \subseteq x')$ since the definition postulates that both $x$ and $x'$ are crisp.

We show that FST proves H to be an inner model of ZF. In more detail, for $\varphi$ a formula in the language of ZF, define $\varphi^H$ inductively as follows: $\varphi^H = \varphi$ for $\varphi$ atomic; $\varphi^H = \psi^H \& \chi^H$ for $\varphi = \psi \& \chi$; $\varphi^H = \psi^H \to \chi^H$ for $\varphi = \psi \to \chi$; $\varphi^H = (\forall x \in H)\psi$ for $\varphi = (\forall x)\psi$.

**Theorem 2.1** *Let* $\varphi$ *be a theorem of ZF. Then* $\mathrm{FST} \vdash \varphi^H$.

To prove this theorem, we first show that the law of the excluded middle (LEM) holds in H; since BL$\forall$ together with LEM yield classical logic, we will have proved that all logical axioms of ZF are provable relativized to H. Then we prove the H-relativized versions of all the axioms of ZF.

**Lemma 10 (Crispness of H)** $\forall x(x \in H \vee \neg x \in H)$.

*Proof.* Note that for any $\alpha$ the following formulas are equiprovable over BL$\forall\Delta$: $\forall x(\alpha \vee \neg\alpha)$; $\Delta\forall x(\alpha \vee \neg\alpha)$; $\forall x\Delta(\alpha \vee \neg\alpha)$; $\forall x\Delta(\alpha \to \Delta\alpha)$ (this is actually equivalent to the preceding formula); $\Delta\forall x(\alpha \to \Delta\alpha)$; $\forall x(\alpha \to \Delta\alpha)$; $\alpha \to \Delta\alpha$.

Substituting $x \in H$ for $\alpha$, we prove the last formula; by definition of H it is the formula

$$(\mathrm{Crisp}(x)\&\exists y \in \mathrm{HCT}(x \subseteq y)) \to \Delta(\mathrm{Crisp}(x)\&\exists y \in \mathrm{HCT}(x \subseteq y)).$$

We know $\mathrm{Crisp}(x) \to \Delta\mathrm{Crisp}(x)$. It suffices to prove $(\mathrm{Crisp}(x)\&\exists y \in \mathrm{HCT}(x \subseteq y)) \to \Delta(\exists y \in \mathrm{HCT}(x \subseteq y))$ (since $\mathrm{Crisp}(x)$ is idempotent we may use it twice in the antecedent).

To do so, prove $(\mathrm{Crisp}(x)\&y \in \mathrm{HCT}\&x \subseteq y) \to \Delta(y \in \mathrm{HCT}\&x \subseteq y)$. By ?? $y \in \mathrm{HCT} \to \Delta y \in \mathrm{HCT}$, and by semantic check $(\mathrm{Crisp}(x)\&\mathrm{Crisp}(y)\&x \subseteq y) \to \Delta(x \subseteq y)$ (the $\mathrm{Crisp}(y)$ in the antecedent comes from $y \in \mathrm{HCT}$, which may again be used repeatedly).

Now generalize: $\forall y((\mathrm{Crisp}(x)\&y \in \mathrm{HCT}\&x \subseteq y) \to \Delta(y \in \mathrm{HCT}\&x \subseteq y))$; hence $(\mathrm{Crisp}(x)\&\exists y(y \in \mathrm{HCT}\&x \subseteq y)) \to \exists y\Delta(y \in \mathrm{HCT}\&x \subseteq y)$; and the succedent implies $\Delta\exists y(y \in \mathrm{HCT}\&x \subseteq y)$.

This concludes the proof.

**Lemma 11** *Let $\varphi(x_1, \ldots, x_n)$ be a ZF-formula whose free variables are among $x_1, \ldots, x_n$. Then FST proves $\forall x_1 \in H \ldots \forall x_n \in H(\varphi^H(x_1, \ldots, x_n) \vee \neg\varphi^H(x_1, \ldots, x_n))$.*

Proof: we consider a formula $\varphi$ with (at most) one free variable $x$, the modification for multiple free variables being easy. We assume that $\varphi$ only contains the connectives & and $\to$ and the universal quantifier. The formula to be proved in FST is $\forall x(x \in H \to (\varphi^H(x) \vee \neg\varphi^H(x))$. It suffices to prove $\forall x\Delta(x \in H \to (\varphi^H(x) \vee \neg\varphi^H(x))$. Suppose $\alpha$ is a formula which has crisp evaluation in all l.o. BL$\Delta$-algebras, and consider the propositional formula $(\alpha \to \Delta\beta) \to \Delta(\alpha \to \beta)$. Under the condition this formula holds in all l.o. BL$\Delta$-algebras, and by completeness, the formula is provable in any theory which proves $\alpha$ to be crisp. Hence, to prove our implication, it is enough to prove $\forall x(x \in H \to \Delta(\varphi^H(x) \vee \neg\varphi^H(x))$, and this is the same as $\forall x(x \in H \to \Delta(\varphi^H(x) \to \Delta\varphi^H(x))$. The universal quantifier can of course be dropped in the proof.

The proof proceeds by induction on the complexity of $\varphi$.

Atomic subformulas: $=$ is a crisp predicate, for $\in$ we have to prove $x \in y \to \Delta x \in y$ assuming $x, y \in H$. In fact $y \in H$ implies $\mathrm{Crisp}(y)$, which entails $\forall x(x \in y \to \Delta x \in y)$.

Conjunction is very easy: for a subformula $\psi_1(x,y)\&\psi_2(x,z)$ of $\varphi$ (we assume one free variable in common, and one distinct free variable for each subformula, and we henceforth omit their explicit listings, for legibility's sake) assume $x, y \in H \to (\psi_1^H \to \Delta\psi_1^H)$ and $x, z \in H \to (\psi_2^H \to \Delta\psi_2^H)$. Then $(x \in H)^2\&(y, z \in H) \to ((\psi_1^H\&\psi_2^H) \to \Delta(\psi_1^H\&\psi_2^H))$, and since $x \in H$ is idempotent, this completes the induction step for conjunction.

Implication (the same simplified notation): for a subformula $\psi_1(x,y) \to \psi_2(x,z)$ of $\varphi$, assume $x, y \in H \to (\psi_1^H \to \Delta\psi_1^H)$ and $x, z \in H \to (\psi_2^H \to \Delta\psi_2^H)$. Thus $(\psi_1^H \to \psi_2^H)\&(x, z \in H) \to (\psi_1^H \to \Delta\psi_2^H)$ (by transitivity of $\to$, and since $x, y \in H$ implies crispness of $\psi_1^H$, we get $x, y \in H \to ((\psi_1^H \to \Delta\psi_2^H) \to \Delta(\psi_1^H \to \psi_2^H))$. Thus $x, y, z \in H \to ((\psi_1^H \to \psi_2^H) \to \Delta(\psi_1^H \to \psi_2^H))$.

The universal quantifier: for a subformula $\forall y\psi(x,y)$ of $\varphi$, the induction hypothesis is $x, y \in H \to (\psi^H(x,y) \to \Delta\psi^H(x,y))$. Generalize in $y$: $x \in H \to \forall y(y \in H \to (\psi^H(x,y) \to \Delta\psi^H(x,y))$; now since for a crisp $\alpha$, BL proves $(\alpha \to (\beta \to \gamma)) \to ((\alpha \to \beta) \to (\alpha \to \gamma))$, we may modify the succedent to $\forall y((y \in H \to \psi^H(x,y)) \to (y \in H \to \Delta\psi^H(x,y))$, and distributing $\forall y$, we have proved: $x \in H \to (\forall y \in H\psi^H(x,y) \to \forall y \in H\Delta\psi^H(x,y))$. To flip the $\Delta$ and the $\forall y \in H$ in the succedent, use again the fact that for crisp $\alpha$, BL$\Delta$ proves $(\alpha \to \Delta\beta) \to \Delta(\alpha \to \beta)$.

**ZF-axioms in H**

We consider ZF with the following axioms: empty set, pair, union, power set, infinity, separation, collection, $\in$-induction. The exact spelling of these axioms is given separately when proving in FST their H-versions.

**Theorem 2.2** *For $\varphi$ being any of the abovementioned axioms of ZF, FST proves $\varphi^H$.*

*Proof.* At this stage it is useful to recall the definitions of HCT and of H: to verify that a set is in H, it is enough to show that it is crisp and that it is a subset of a set in HCT.

**(empty set)** $\exists z \forall u \neg u \in z$.

The H-translation, which reads $\exists z \in H \forall u \in H \neg u \in z$, is absolute in H; in fact $\emptyset \in HCT$.

**(pair)** $\forall x, y \exists z \forall u (u \in z \equiv (u = x \vee u = y))$.

The H-translation $\forall x, y \in H \exists z \in H \forall u \in H (u \in z \equiv (u = x) \vee (u = y))$ is absolute: the set $\{x, y\}$ is crisp (since $=$ is crisp); to show that it is a subset of a set which is in HCT, consider $x' \in HCT$ a witness for $x \in H$ and $y' \in HCT$ a witness for $y \in H$. Then $\{x, y\} \cup x' \cup y'$ is a crisp transitive set with crisp elements, hence in HCT, and thus $\{x, y\}$ is in H.

**(union)** $\forall x \exists z \forall u (u \in z \equiv \exists y (u \in y \in x))$.

The H-translation $\forall x \in H \exists z \in H \forall u \in H (u \in z \equiv \exists y \in H (u \in y \in x))$ is absolute: let $x' \in HCT$ witness $x \in H$. Then $\bigcup x$ is a crisp set with crisp elements (since $x \subseteq x'$), and $\bigcup x \subseteq \bigcup x' \in HCT$, which witnesses $\bigcup x \in H$.

**(power set)** $\forall x \exists z \forall u (u \in z \equiv u \subseteq x)$

Let $x'$ be the witness for $x \in H$, and let $P(x)$ denote the *crisp* power of $x$. For $x \in H$, $P(x)$ is crisp, but its elements need not be (it only follows from the definition that they are subsets of x in degree 1). Define $P'(x) = \{u \in P(x); \mathrm{Crisp}(u)\}$; then for $x \in H$ it holds that $\forall u \in H (u \in P'(x) \equiv \Delta u \subseteq x \equiv u \subseteq x)$, $P'(x)$ is crisp, and $P'(x) \subseteq P'(x') \cup x'$, which is a transitive crisp set with crisp elements, thus in HCT and a witness for $P'(x) \in H$.

**(separation)** $\forall x \exists z \forall u (u \in z \equiv (u \in x \& \varphi(u))$ for a ZF-formula not containing $z$ freely.

The H-translation is absolute: let $x' \in HCT$ witness $x \in H$ and set $z = \{u \in x; \varphi^H(u)\}$, then $z$ is a crisp set and $z \subseteq x \subseteq x' \in HCT$ (i.e., $x'$ is a witness for $z \in H$).

**(infinity)** $\exists z (0 \in z \& \forall u \in z (u \cup \{u\} \in z))$.

It suffices to prove that there is a set $z \in H$ s.t. $0 \in z \& \forall u \in z (u \cup \{u\} \in z)$. Let $z_0$ be the set postulated by the axiom of infinity in FST and define $z_1 = \{x \in z_0 : \Delta(x \in z_0) \& \mathrm{Crisp}(x)\}$. Then $z_1$ is a subset of $z_0$ and $0 \in z_1$ and $u \in z_1 \to u \cup \{u\} \in z_1)$. Now let $z = \{x \in z_1 : \forall y \in x (y \in z_1)\}$, i.e., a transitive subset of $z_1$. Obviously $0 \in z$, let us prove $x \in z \to (x \cup \{x\}) \in z$, that is by definition of $z$, $[x \in z_1 \& \forall y \in x (y \in z_1)] \to [x \cup \{x\} \in z_1 \& \forall a (a \in x \vee a = x \to a \in z_1)]$. We know $x \in z_1 \to (x \cup \{x\} \in z_1)$, and since $x \in z_1$ is crisp, we may use it repeatedly and thus get $x \in z_1 \& (y \in x \to y \in z_1) \to (y \in x \vee y = x \to y \in z_1)$.

**(extensionality)** $\forall xy (x = y \equiv \forall z (z \in x \equiv z \in y))$.

The H-translation $\forall x, y \in H (x = y \equiv \forall z \in H (z \in x \equiv z \in y))$ follows from extensionality in FST by H being transitive and by the crispness of its elements (the $\Delta$'s may be left out).

**(collection)** $\forall u (\forall x \in u \exists y \varphi(x, y) \to \exists v \forall x \in u \exists y \in v \varphi(x, y))$ for $\varphi$ not containing $v$ freely.

The H-translation reads $\forall u \in H(\forall x \in H(x \in u \to \exists y \in H\varphi^H(x,y)) \to \exists v \in H\forall u \in H(u \in x \to \exists y \in H(y \in v\&\varphi^H(x,y))))$. Fix $u \in H$; we want to find a corresponding $v \in H$ s.t. the above is true. First define $v_0$ by collection in FST for $u$ and the formula $y \in H\&\varphi^H(x,y)$; separate $v_1 = \{x \in v_0 : \Delta(x \in v_0)\&x \in H\}$. Then $v_1 \subseteq H$ is a crisp set and, since $u$ is crisp, satisfies the collection axiom. By collection on $v_1$, let $v_2 = \{x : \exists x_0 \in v_1(x_0 \subseteq x\&x \in HCT\}$ ($v_2$ is a set of witnesses for each member of $v_1$ being in H). $v_2$ is a crisp set of crisp elements. Thus $\bigcup v_2$ is crisp, and also transitive because $b \in a \in \bigcup v_2$ implies $\exists y(b \in a \in y \in v_2)$, and since $y \in HCT$ is transitive, $b \in y \in v_2$ gives $b \in \bigcup v_2$. Thus $\bigcup v_2 \in HCT$. Now consider $P'(\cup v_2)$ (the crisp elements of the crisp power set of $\bigcup v_2$). This set is in H because it is crisp and a subset of $P'(\bigcup v_2) \cup \bigcup v_2$, which is a transitive crisp set of crisp elements, thus in HCT. To conclude, for each $x_0 \in v_1$ we found $x \in v_2$ s.t. $x_0 \subseteq x$, thus $x_0 \in P'(\bigcup v_2)$, and $P'(\bigcup v_2)$ is the desired set.

**($\in$-induction)** $\forall x(\forall y \in x\varphi(y) \to \varphi(x)) \to \forall x\varphi(x)$ for any ZF-formula $\varphi$.

The H-translation is $\forall x \in H(\forall y \in H(y \in x \to \varphi^H(y)) \to \varphi^H(x)) \to \forall x \in H\varphi^H(x)$. Fix a ZF-formula $\varphi$, and assume $x \in H$. Consider the instance of $\in$-induction in FST for the formula $x \in H \to \varphi^H(x)$: this is $\Delta\forall x(\forall y \in x(y \in H \to \varphi^H(y)) \to (x \in H \to \varphi^H(x))) \to \Delta\forall x(x \in H \to \varphi^H(x))$. This formula is our aim except for the $\Delta$'s; our desire now is to omit both of them, and we are allowed to do so considering that both the antecedent and the succedent (without the respective $\Delta$'s) are semantically crisp.

## References


[1] P. Hájek, "Metamathematics of Fuzzy Logic", *Kluwer Academic Publishers*, 1998.

[2] P. Hájek, "Function symbols in fuzzy predicate logic", *Proceedings East West Fuzzy Colloquium 2000, Zittau/Gorlitz*, pp.2-8.

[3] P. Hájek, "What we cannot have in fuzzy set theory", unpublished notes.

[4] P. Hájek, Z. Haniková, "A set theory within fuzzy logic", *Proceedings of the ISMVL'01 Warsaw* (in print).

[5] Z. Haniková, "Teorie množin ve vícehodnotové logice", *Doktorandský den '00, Sborník příspěvků*, pp.27-31.

[6] M. Shirahata, "Phase-valued models of linear set theory", preprint.

[7] G. Takeuti, S. Titani, "Intuitionistic fuzzy logic and intuitionistic fuzzy set theory", *J. Symb. Logic* 49 (1984), pp. 851-866.

[8] G. Takeuti, S. Titani, "Fuzzy logic and fuzzy set theory", *Arch. Math. Logic*, 32(1992), pp. 1-32.

[9] S. Titani, "A lattice-valued set theory", Arch. Math. Logic, 38(1999), pp. 395-421.

# On axiomatic systems of the various fuzzy logics

*doktorand:*
Petr Cintula

Pokoj 435, Blok 11, Vaníčkova 5, Praha 6, 16017

Cintula@cs.cas.cz

*školitel:*
Petr Hájek

Ústav informatiky, Akademie věd České Republiky, Pod vodárenskou věží 2, 182 07 Prague 8

hajek@cs.cas.cz

obor studia:
Matematické inženýrství

**Abstrakt**

The goal of this short work is to summarize and present my affords in simplifying axiomatic systems of the various fuzzy logics based on Hájek's Basic Logic BL. We will start with product logic, then we will continue with product involutive logic and at last but certainly not at least we will discuss alternative axiomatic systems for the ŁΠ logic. Results appearing in this work are from my papers ([1], [2], [3], [4]) and also are parts of my diploma thesis [5].

## 1. Introduction

Fuzzy (or many-valued) logics have been widely studied during the last decade of the last century. These logic are a direct generalization of the classical logic by an assumption that the possible truth value of the formula can be other than Truth or False. There are many different approaches towards many-valued logics. Very important class of these approaches is based on Petr Hájek's Basic Logic BL.

Each logic consists of two different parts (or aspects): syntax (what is "provable"?) and semantic (what is "true"?). We will deal only with the syntactical aspects of various many-valued logics based on already mentioned Basic Logic BL. Before we start presenting our results in this field we need to recall some basic definitions and we will also introduced logic BL and some of its extensions.

## 2. Preliminaries

The syntax of each logic is given by three sets: the set of connectives, the set of axioms and the set of deduction rules. The set of connectives tells us how to build formulas of our logic. The set of axioms tells us which formulas are considered to be "valid obviously" in our logic (this set is usually referred as an axiomatic system). Finally the set of deduction rules tells us how to derive another "valid" formulas from axioms of our logic. Having this three sets we may introduce a notion of a "provable" formula.

The formula is said to be provable in some logic if there is a proof of this formula within that logic. The proof is a sequence of the formulas such that each member of this sequence is either an axiom or can be derived from the previous members of the proof by using some deduction rule.

Two differen syntaxes are said to be equivalent if each formula provable in the first syntax is provable in the second one and other way round (if the sets of connectives and deduction rules are the same we speak about the equivalence of the axiomatic systems rather than about the equivalence of syntaxes).

The axiomatic system should be consistent, complete and "simple". The first two properties doesn't interests us in this work. We will deal only with the third one. The property "simple" means that the system should be small and the axioms should be short, well comprehensible and should express some logical properties of our logic.

### 3. Basic logic and its extension

The basic logic BL was introduced by Petr Hájek is heavily studied in his work [6]. This logic has one logical constant $\overline{0}$ (this is in fact a nullary connective) and two binary connectives: conjunction & and implication $\rightarrow$. There are more connectives (negation $\neg$, disjunction $\vee$, equivalence ($\equiv$), etc.) definable using these ones but we will not need them. BL has seven axioms (there is no need to write them down explicitly in this work) and one deduction rule (modus ponens: from $\varphi$ and $\varphi \rightarrow \psi$ derive $\psi$). BL can be extended into the three famous logics by adding some axioms.

| | | | |
|---|---|---|---|
| Łukasiewicz logic: | BL | + | $\neg\neg\varphi \rightarrow \varphi$ |
| Gödel logic : | BL | + | $\varphi \rightarrow (\varphi \& \varphi)$ |
| product logic : | BL | + | $\neg\neg\varphi \rightarrow ((\varphi \& \chi \rightarrow \varphi \& \psi) \rightarrow (\chi \rightarrow \psi))$ |
| | | + | $\neg\neg\varphi \& \neg\varphi \rightarrow \overline{0}$ |

You can notice that there are two axioms of product logic and the first one is quite complicated one, while the axiomatic system of the other two logics are rather simple. There arise two obvious questions. Could that axiomatic system be simplified ? Is there an axiom with only one variable, which would axiomatize product logic (just like there exists such axioms for the Łukasiewicz and Gödel logic)? We answer these question in the following theorems. These theorems were stated and proven in [3].

**Theorem 1** *If we add the formula $\neg\neg\varphi \rightarrow ((\varphi \rightarrow \varphi \& \psi) \rightarrow (\neg\neg\psi \& \psi))$ to the axioms of BL we obtain product logic (i.e. those axiomatic systems are equivalent).*

**Theorem 2** *There is no formula with single variable such that if we add it to BL we would get product logic.*

There are of course many other extensions of BL. Some of them has extended not only an axiomatic system but also the set of connectives. One of those extensions is called STR$_\sim$ logic (strict involutive basic logic - introduced by F. Esteva, L. Godo, P. Hájek and M. Navara in [7]). We obtain it by adding a new unary connective ($\neg_{\L}$ this is in fact the second negation, we change the symbol for normal negation from $\neg$ to $\neg_{\Pi}$ and we also define a new connective $\triangle$ as $\neg_{\Pi}\neg_{\L}$), one additional deduction rule (necessitation of $\triangle$: from $\varphi$ infer $\triangle\varphi$) and following additional axioms:

| | |
|---|---|
| (STR) | $\neg_{\Pi}(\varphi \& \psi) \rightarrow (\neg_{\Pi}\varphi \vee \neg_{\Pi}\psi)$ |
| ($\sim$1) | $\neg_{\L}\neg_{\L}\varphi \equiv \varphi$ |
| ($\sim$2) | $\neg_{\Pi}\varphi \rightarrow \neg_{\L}\varphi$ |
| ($\sim$3) | $\triangle(\varphi \rightarrow \psi) \rightarrow (\neg_{\L}\psi \rightarrow \neg_{\L}\varphi)$ |
| ($\triangle 1'$) | $\triangle\varphi \vee \neg_{\Pi}\triangle\varphi$ |
| ($\triangle 2'$) | $\triangle(\varphi \vee \psi) \rightarrow (\triangle\varphi \vee \triangle\psi)$ |
| ($\triangle 5'$) | $\triangle(\varphi \rightarrow \psi) \rightarrow (\triangle\varphi \rightarrow \triangle\psi)$ |

There is one interesting result about this logic, a redundancy of some axioms. This result will be published in my upcoming paper [4].

**Theorem 3** *The axioms ($\triangle 1'$), ($\triangle 2'$) and ($\triangle 5'$) are redundant in the definition of the $STR_\sim$ logic, i.e. there are provable using the remaining axioms of the $STR_\sim$.*

## 4. The ŁΠ logic

Another very interesting logic is the ŁΠ logic. This logic was defined by F. Esteva, L. Godo and F. Montagna in [8]. It is basically defined as union of another logics, namely Łukasiewicz and product logic. All connectives from both of these logics are definable using three basic connectives (Łukasiewicz implication $\rightarrow_Ł$, product implication $\rightarrow_\Pi$ and product conjunction $\odot$) and a truth constant $\overline{0}$. It has two deduction rules (Modus Ponens and necessitation, just like the $\mathrm{STR}_\sim$ logic) and the following axioms:

- (ŁΠ1) Axioms of the Łukasiewicz logic with $\triangle$ [1]
- (ŁΠ2) $\triangle(\varphi \equiv_Ł \psi) \wedge \triangle(\chi \equiv_Ł \delta) \rightarrow_Ł ((\varphi * \chi) \equiv_Ł (\psi * \delta))$, for $* \in \{\rightarrow_\Pi, \odot\}$
- (ŁΠ3) $(\varphi \odot \psi) \rightarrow_Ł (\psi \odot \varphi)$
- (ŁΠ4) $(\varphi \odot \psi) \odot \chi \equiv_Ł \varphi \odot (\psi \odot \chi)$
- (ŁΠ5) $\varphi \wedge \neg_\Pi \varphi \rightarrow_Ł 0$
- (ŁΠ6) $\varphi \odot (\psi \ominus \chi) \equiv_Ł (\varphi \odot \psi) \ominus (\varphi \odot \chi)$
- (ŁΠ7) $\triangle(\varphi \rightarrow_Ł \psi) \rightarrow_Ł (\varphi \rightarrow_\Pi \psi)$
- (ŁΠ8) $\triangle(\psi \rightarrow_Ł \varphi) \rightarrow_Ł (\varphi \odot (\varphi \rightarrow_\Pi \psi) \equiv_Ł \psi)$

The main result here is a following equivalent axiomatic system. This result is a part of my thesis [5] and will be published in my upcoming paper [2]. This system is much simpler and shows in a very good way how product and Łukasiewicz logic are tight together.

**Theorem 4** *The following is the axiomatic system of the* ŁΠ *logic*

- (M1) Axioms of Łukasiewicz logic
- (M2) Axioms of product logic[2]
- (Ł△) $\triangle(\varphi \rightarrow_Ł \psi) \rightarrow_Ł (\varphi \rightarrow_\Pi \psi)$
- (Π△) $\triangle(\varphi \rightarrow_\Pi \psi) \rightarrow_Ł (\varphi \rightarrow_Ł \psi)$
- (Cor) $\varphi \odot \neg_Ł \psi \equiv_Ł \neg_Ł(\varphi \rightarrow_Ł \neg_Ł(\varphi \odot \psi))$

Another interesting result here is the proof of a connection between the ŁΠ logic and so-called product involutive logic $\Pi_\sim$, which is an extension of the $\mathrm{STR}_\sim$ by additional axiom of product logic. It turns out that in this logic we can easily define all connectives of the ŁΠ logic and that the ŁΠ logic is just an extension of the $\Pi_\sim$ by a single axiom! This result is a part of my thesis [5] and will be published in my upcoming paper [4].

[1] Lukasiewicz logic with $\triangle$ is just another extension of BL, we don't need to go into further details here
[2] It can be even proven that we don't need all of these axioms, at least two of them can be omitted

**Theorem 5** *The following is the axiomatic system of the* ŁΠ *logic (Łukasiewicz implication* $\varphi \to_Ł \psi$ *is defined as* $\neg_Ł(\varphi \odot \neg_Ł(\varphi \to_\Pi \psi))$.)

(Π) *Axioms of product logic*[3]

(∼1) $\neg_Ł\neg_Ł\varphi \equiv_\Pi \varphi$

(∼2) $\neg_\Pi\varphi \to_\Pi \neg_Ł\varphi$

(∼3) $\triangle(\varphi \to_\Pi \psi) \to_\Pi \triangle(\neg_Ł\psi \to_\Pi \neg_Ł\varphi)$

(Ł2) $(\varphi \to_Ł \psi) \to_Ł ((\psi \to_Ł \chi) \to_Ł (\varphi \to_Ł \chi))$[4]

## References

[1] P. Cintula, "The ŁΠ and ŁΠ$\frac{1}{2}$ propositional and predicate logics," *Fuzzy Sets and Systems*, vol. 124/3, pp. 21–34, 2001.

[2] P. Cintula, "Advances in the ŁΠ and ŁΠ$\frac{1}{2}$ logics." Paper in preparation.

[3] P. Cintula, "About axiomatic systems of product fuzzy logic," *Soft Computing*, vol. 5, pp. 243–244, 2001.

[4] P. Cintula, "Product involutive fuzzy logic," *To appear in a Special Issue on SOFSEM2001 of Neural network = world.*

[5] P. Cintula, "The ŁΠ and ŁΠ$\frac{1}{2}$ logics," Master's thesis, Czech Technical University, FNSPE, Dept. of Math., 2001.

[6] P. Hájek, *Metamathematics of fuzzy logic.* Dordrecht: Kluwer, 1998.

[7] F. Esteva, L. Godo, P. Hájek, and M. Navara, "Residuated fuzzy logics with an involutive negation," *Archive of math. logic*, vol. 40, pp. 103–124, 2000.

[8] F. Esteva, L. Godo, and F. Montagna, "The ŁΠ and ŁΠ$\frac{1}{2}$ logics: two complete fuzzy systems joining łukasiewicz and product logics." To appear in Archive of Math. Logic.

[3]Observe that the axiom (STR) of the STR$_\sim$ logic indeed holds in the product logic, and thus a logic satisfying this set of axioms together with the following three axioms of $\sim$ is indeed an extension of the STR$_\sim$. It also extends a product logic - this is in fact the product involutive logic.

[4]It can be shown that this axioms is not provable in the product involutive logic. Thus the ŁΠ logic is a non-conservative extension of the product involutive logic.

# Statistical methods in genetic studies

*doktorand:*
Mgr. Markéta Kyloušková
EuroMISE Center, Charles University and Academy of Sciences, Pod vodárenskou věží 2, Praha 8
kylouskova@euromise.cz

*školitel:*
prof. RNDr. Jana Zvárová, DrSc.
EuroMISE Center, Charles University and Academy of Sciences, Pod vodárenskou věží 2, Praha 8
zvarova@euromise.cz

obor studia:
Statistical genetics

**Abstrakt**

The article gives a quick introduction to the interesting and rapidly developing field of statistical genetics. It presents a brief explanation of the concept of linkage analysis and an overview of different types of studies employed in genetical research. In the closing part a focus of interest of the author is described.

## 1. Short introduction to genetic analysis topics

Genetics (as a science about heredity) and statistics are closely related since their beginning – it suffices to mention two famous names as sir Francis Galton ("regression towards the mediocrity/mean") and R.A. Fisher (design of experiments, agricultural and biometrical studies). In a field where systematic (genetic) and random (environment) components play an important role it is vital for biologists (medical doctors) and statisticians to cooperate and to improve methods of detection of genome regions that are responsible for a studied phenotypic trait.

Most of the genetic analyses aim at the following task: estimate the way and the extent a particular trait is inherited. A genetist faces usually two types of traits: **Mendelian** traits and **complex** traits. A statistician distinguishes between a **qualitative** (discrete) and a **quantitative** (continuous) trait.

Qualitative traits are those which show large and distinguishable phenotype effects (discrete pattern). The important fact is that although small random modifications due to environment may occur, they are not of great importance. These traits would be classified as Mendelian by a genetist as they are usually caused by an action of a single gene (single mutation at one locus) and the model of inheritance (autosomal dominant/recessive, sex linked) is usually easy to observe in pedigrees. Huntington's disease, cystic fibrosis or sickle cell anemia may serve as well known examples. Most of these conditions in humans were already explored.

Unlike for qualitative traits, phenotype of a quantitative trait provides very little information on underlying genotype. Quantitative traits (often denoted QTL, which stands for quantitative trait loci) are of continuous nature, and are supposed to result from a joint action of several (two to tens) genes, that may act additively, in synergy, in opposite directions or in other setting. This joint action is summarized in the term "complex trait" in the genetical language. Body height, predisposition to infections, all complex diseases as atherosclerosis, cancer or schizophrenia may serve as examples of quantitative traits. The environmental (non genetic) part is always present and plays an important role; it may contribute to more than one half of the total phenotypic variability.

While nowadays the genetic research is mostly human-disease oriented, historically, quantitative traits were of interest in animal and plant breeding first. Different methods have been developed; they have been oriented towards finding out how much a particular trait is heritable (i.e. to assist in decisions whether it is worth the pain to breed animals/plants to obtain a strain of the desired property). The methods included ANOVA, mixed models and partition of variance, restricted maximum likelihood methods etc. [1]. The detection or localization of the underlying genes were not of importance.

Only in last decades, due to the discovery of DNA structure and quick analytical tools, the attention has shifted towards direct examination of effects and positions of individual genes and gene systems. This change brought forward a concept of **linkage** and of linkage analysis.

## 2. Linkage analysis – biological concept

For the "purity" of scientific research it is preferable that the gene responsible for a particular trait is found via a deep biological knowledge of the trait. This path may lead to the goal if e.g. the protein responsible for the disease is known, and the only task consists in finding which spot on the DNA corresponds to the protein. It is clear that there is no place a statistician had to help.

However, only seldom it is possible to find suitable candidate loci for QTL based on biological knowledge of the trait. QTLs are more often assayed indirectly by using linked **marker** loci, i.e. genes of known genotype and genome position that are or are thought to be strongly associated with the trait. The part of statistical genetics that searches for a location of genes influencing a trait in question using marker data is called linkage analysis.

The basic biological scheme of linkage is as follows: Each individual's chromosomes come in pairs where one chromatid is maternal and one is paternal. During gamete (ovum, sperm) formation a mixing of paternal and maternal information occurs in two steps: (1) On most chromosomes at least one recombination occurs. This means that one copy of the maternal chromatid exchanges its part with the paternal chromatid, resulting to a mix of maternal and paternal information. (2) The chromosomes segregate independently into newly formed gametes, thus providing another mixing.

It is clear that any pair of traits (or a trait and a marker) located on different chromosomes is inherited from the same grandparent with probability one half. If no recombination occurred, any pair of traits located on the same chromosome would be inherited from the same grandparent with probability 1, and no other possibility except for 1 and 0.5 could occur. On the other hand, with recombination an entire scale of probability values from 0.5 to 1 is possible, hence giving the researcher an opportunity to estimate the distance of the gene(s) responsible for the trait in question from the marker.

## 3. Linkage analysis – types of research [3, 4]

Three main areas of statistical genetics' research work with the concept of linkage. The first and the most rigorous is the **experimental** research. In this field, on-purpose bred strains of animals or plants with extreme trait manifestations are mated and their offspring examined for marker genotypes and trait value. Then a wide range of association models are employed to find markers most associated with the trait and hence most probable to lie close to the location of "true" responsible genes. These methods are quite developed and reliable, but nonetheless unapplicable for human-disease research.

**Family studies** represent a tool closest to experimental design applicable for humans. Basically they try to track down markers inherited together with the disease by using family pedigrees. Affected sib-pair analysis, transmission-disequilibrium test and allele-sharing analysis belong to these methods. Results are usually quite reliable but the studies are extremely demanding as far as data extent and precision are concerned.

The least informative but the most popular are **population studies**, also known as association studies. They are quite similar to epidemiological studies well known to medical audience. The core of association studies are case-control studies, where the frequency of marker allele in diseased people is compared to the frequency of marker alleles in healthy controls. Basically a $2 \times 2$ contingency table test can be and is used, although a logistic regression (possibly multivariable) approach fights for its place in the sun recently.

The association studies are most meaningful when it involves alleles with possible direct biological relevance. The findings then open the door for biological explanations and permit to design relevant biochemical experiments.

The main disadvantage of the population studies is the fact that the discovered association might be an artifact of population admixture. It also might not be clear where is the right spot of the gene in the cause-effect path.

## 4. My focus

In my doctoral studies I've focused on population studies, as they represent a type of study most employed by medical researchers who contact our centre. Firstly I participated in a practical task of analyzing a real genetic study [2]. During one year we have been analyzing a large case-control study on atherosclerotic patients. This study fulfilled all requirements for an association study, including the fact that the examined genes produced enzymes of the homocysteine metabolism cycle, homocysteine being one of the chemicals known to be closely associated to atherosclerosis. We opted for the use of multivariable logistic regression to account for different environmental factors known to influence the risk of atherosclerosis, an approach that is still not standard in similar analyses. The research results are to be sent to review in short time and we hope that our approach will be accepted and approved by the scientific audience.

Secondly I've started a theoretical study of detection of disease-causing allele inheritance mode, based on article [5]. The article features elaborate models designed for better understanding of disease inheritance and my aim is to extend the work to the cases where more genes are involved.

In fine I also participate in development of a modular system that would help medical doctors and researches better and easily analyze genetical data.

## References


[1] D.S. Falconer, "Introduction to Quantitative Genetics", Longman Scientific & Technical, England, 1989.

[2] B. Janošíková, M. Kaloušková, D. Kocmanová, A. Vítová, K. Veselá, L. Meixnerová, J. Krijt, P. Kraml, J. Hyánek, J. Zvárová, M. Anděl, V. Kožich, "Metabolism of Aminothiols and the Risk of Coronary Artery Disease: Modulation by Genetic Variants in the Methionine Cycle", (in preparation)

[3] M. Lynch, B. Walsh, "Genetics and Analysis of Quantitative Traits", Sinauer Associates Inc., Sunderland, MA, 1997.

[4] P.D. Markel, "Lecture Notes on Statistical Genetics", *Student copies at LUC*, Belgium, 2000.

[5] G. Thomson, "Investigation of the Mode of Inheritance of the HLA Association Disease by the Method of Antigen Genotype Frequencies among Individuals", *Tissue Antigens*, vol. 21, pp. 81–104, 1983.

# Řízení a stabilizace v biosystémech

*doktorand:*
ING. PETR ZAVADIL

EuroMISE
Ústav informatiky AV ČR
Pod Vodárenskou věží 2
Praha 8

zavadil@euromise.cz

*školitel:*
DOC. ING. VLADIMÍR ECK, CSC.

Katedra kybernetiky
FEL ČVUT v Praze
Technická 2
Praha 6

eck@labe.felk.cvut.cz

obor studia:
Umělá inteligence a biokybernetika

**Souhrn**

Práce je zaměřena na oblast řízení a stabilizace v biosystémech. Biosystém je chápán jako systém popsatelných psychofyziologických koincidencí u člověka. Aplikační a experimentální část je zaměřena na oblasti identifikace psychofyziologických stavů člověka, ovlivňování těchto stavů a predikci na základě znalosti fyziologických parametrů.

## 1. Úvod

**Biosystém** v tomto kontextu zúžíme na řadu popsatelných a do jisté míry ovlivnitelných psychofyziologických stavů člověka, každý stav pak chápeme intraindividuálně, není definován přesně ani ostře a může se překrývat se stavy ostatními. V dalším postupu budou tyto stavy zjednodušeny na stavy operátora (člověka, který je zapojen do experimentu, sledován a měřen).

**Řízením** rozumíme ovlivňování systému vnějšími podněty a **stabilizaci** udržování systému ve zvoleném stavu. V případě psychofyziologického přístupu je počet volitelných stavů pro stabilizaci značně omezen.

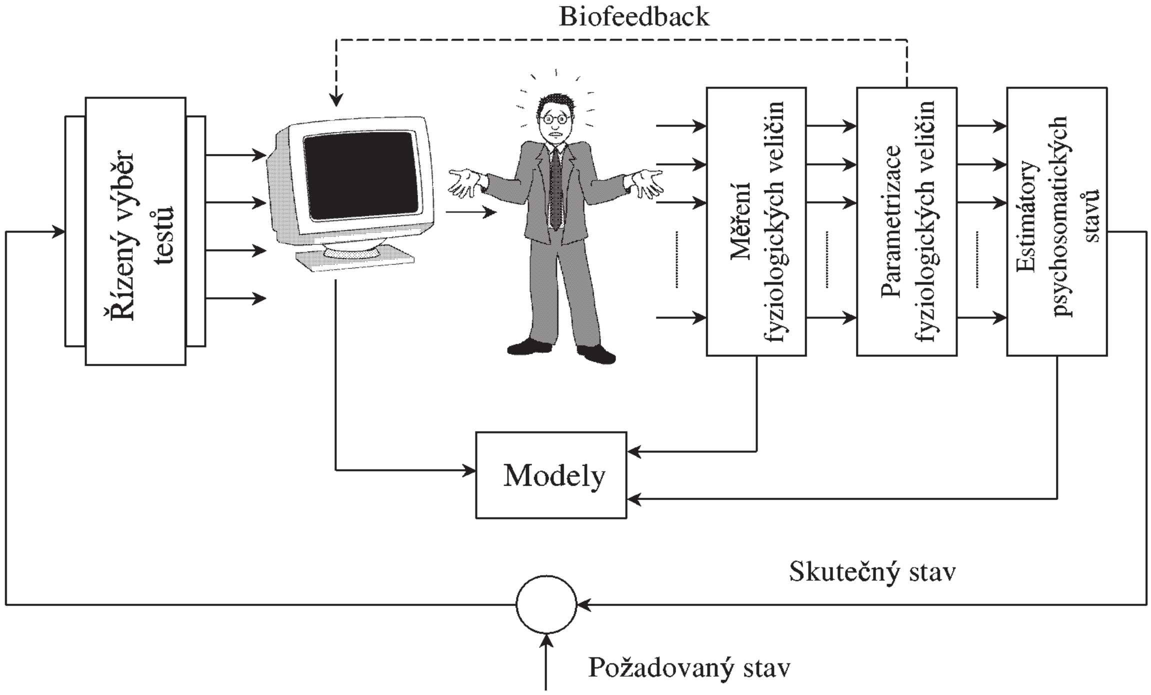
Pro řízení a stabilizaci pak předpokládáme možnost uzavření zpětné vazby (ZV) na systému, konkrétně u člověka.

## 2. Uzavření zpětné vazby u člověka

Uzavření zpětné vazby u člověka vychází ze dvou předpokladů:

- psychický stav člověka lze ovlivňovat a tím ho do určité míry měnit,
- tento stav je úzce svázán s parametry fyziologických (případně dalších) veličin objektivně zjistitelných u člověka.

Míra ovlivnění psychického stavu a psychofyziologická provázanost je individuální, proto je nutné před realizací ZV o každém člověku získat dostatečné množství informací, podle kterých se systém ZV člověku přizpůsobí a případně vymezí možnost požadavků na ovlivňování psychického stavu. Struktura systému ZV u člověka je ilustrována v blokovém schématu na obr. 2.




**Obr. 2:** Blokové schéma uzavření zpětné vazby u člověka

Blok *měření fyziologických veličin* odpovídá úrovni senzorů a měřících přístrojů.
*Parametrizace fyziologických veličin* je částečně řešena také na úrovni přístrojů a dále již matematického zpracování, které je spolu s ostatními bloky řešeno softwarově v počítači.
*Estimace psychosomatických stavů* a následná *komparace* s požadovaným stavem odpovídá dvěma logickým systémům. Tyto systémy pracují na základě nejefektivnějšího umělointeligenčního principu, vybraného z několika alternativ.
*Řízený výběr testů* předkládá (na základě standardizované informace z komparátoru) test z otevřené databáze klasifikovaných zátěžových a relaxačních testů.
K těmto základním blokům můžeme připojit blok *modelů* (fyziologických i psychologických). Jedná se jak o porovnávání aktuálního průběhu s dosud známými a implementovanými modely, tak o tvorbu nových modelů.
Pro názornost je doplněna větev *biofeedbacku*, která toto pojetí ZV uvádí do kontextu s obecně známým pojmem z oblasti psychologické a psychiatrické terapie.

S obdobně pojatým problémem sledování psychofyziologických stavů člověka se můžeme v současné době setkat u skupiny japonských vědců (Yoshikawa, Takahashi, Kitamura a další) z Institutu pro jadernou energii na Kyotské univerzitě. Jedná se o systém "man-machine interface" (MMI) vyvíjený pro řízení jaderné elektrárny [1, 2, 3].

### 3. Identifikace psychofyziologických stavů

Název stavu *psychofyziologický* můžeme nahradit též *psychosomatický*, nebo *psychofyzický*. Pro jeho popis využíváme maximum relevantních informací o psychických a fyziologických projevech člověka. Vzhledem k převažujícím intraindividuálním vlastnostem těchto stavů je nutné u člověka tyto stavy identifikovat dlouhodobou testovací fází. Výsledek by měl směřovat k možnosti odhadu těchto stavů na základě neúplných vstupních informací, např.: fyziologických parametrů měřených telemetricky a reakčních parametrů při předložení zátěžového testu.

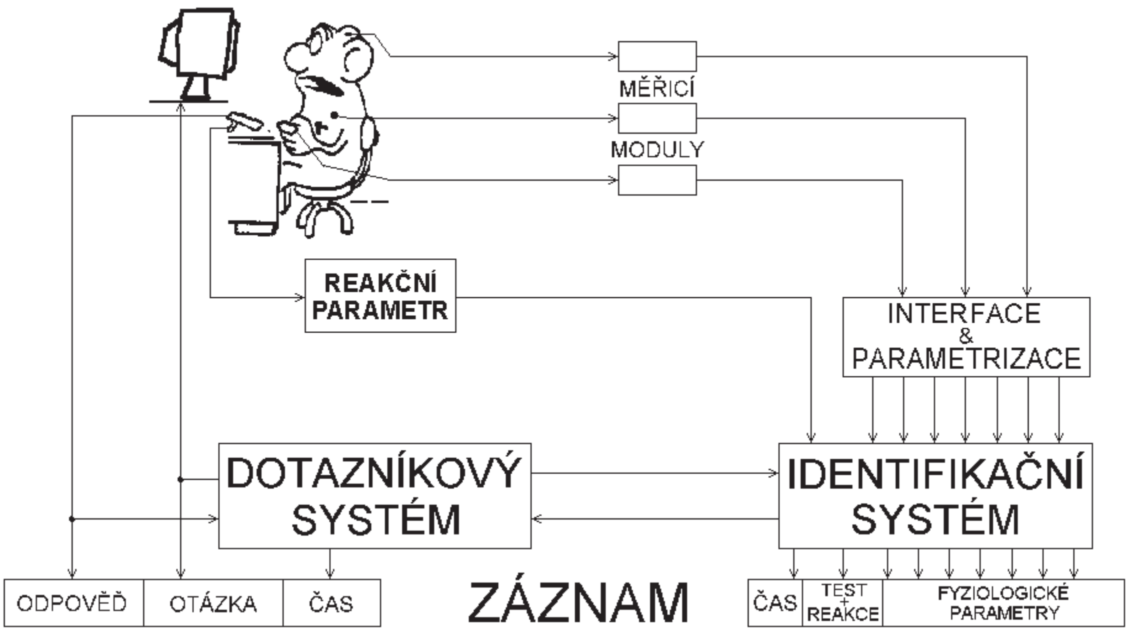
Pro praktické použití identifikovaných stavů a zkrácení fáze identifikace je vhodné jejich intraindividualitu částečně potlačit a vytvořit charakteristické skupiny osob, každý další člověk je pak testován pouze ve specifických oblastech významných pro dané skupiny.

Při identifikaci jsou mapované parametry a informace rozdělené na dvě části:

- fyzickou (fyziologickou, somatickou)
- psychickou

Dosud publikované práce při testování využívaly pro identifikaci psychické oblasti různé druhy zátěže. Parametry fyziologických signálů byly přiřazovány k psychickým stavům, které zátěž vybudila a jejich existence byla implicitně předpokládána.

Autor pro identifikaci psychické části zvolil metodiku jinou. Orientoval se na postupy standardní psychologie, které využívají sebediagnostické dotazníky. Konkrétně škálové dotazníky z oblasti krátkodobých emočních stavů při zátěži a stresu [4]. Návrh postupu při identifikaci psychosomatických stavů dotazníkovou metodou je na obr. 3.




**Obr. 3:** Identifikace psychosomatických stavů člověka

Takto navržená identifikace předpokládá několikanásobné měření v standardizovaných podmínkách. Testovaná osoba vyplní krátký klasifikačně-osobnostní dotazník a škálový dotazník MIND. Druhý emoční dotazník obsahuje otázky s 5-ti stupňovými škálami míry ztotožnění s otázkou. Originální dotazník obsahuje 40 otázek a bývá předkládán v papírové formě. Pro vícenásobný postup identifikace je vhodné předložit dotazník v elektronické formě, zaznamenávat zároveň čas a případně redukovat počet emočních otázek.

## 4. Navržené cíle a stav rozpracovanosti

Identifikace psychofyziologických stavů a uzavření zpětné vazby u člověka jsou značně rozsáhlé problematiky. Autor si stanovil tyto dílčí cíle:

- navrhnout redukci počtu otázek emočního dotazníku
- otestovat skupinu osob předložením emočních dotazníků a měřením vybraných fyziologických signálů s předpokládanou psychofyziologickou vazbou a jednoduchou aplikací při testech
- rozdělit testované osoby na charakteristické podskupiny za předpokladu maximálního zjednodušení a minimalizace počtu stavů u jednotlivců
- navrhnout několik realizací pro estimaci stavů z částečných - fyziologických vstupů u testovaných osob
- navrhnout další postup pro faktickou realizaci ZV u člověka

Dosud autor vytvořil softwarový dotazníkový systém s následným grafickým výstupem umožňujícím přehledně informovat testovanou osobu bezprostředně po vyplnění o aktuálním emočním stavu.

Dále otestoval skupinu 40-ti osob. Při testování byly použity 2 standardní dotazníky s plným počtem otázek. Na základě studie zpracování psychologických dotazníků ve švédštině [5] zpracoval výsledky faktorovou analýzou a navrhl redukci počtu otázek. Metodika zpracována v prostředí Matlab podle [6, 7]. Omezení relevantnosti výsledků je nesplnění normálního rozložení vstupního souboru dat a zároveň nedostatečné množství testovaných osob.

Pro další testování navrhl měření vybraných fyziologických signálů s psychofyziologickou vazbou - další realizace závisí na dostupnosti měřících přístrojů.

Pro estimaci autor navrhl několik alternativních řešení fuzzy systém, neuronovou síť, expertní systém a regresní model.

Dílčí výsledky byly prezentovány ve výzkumných zprávách na Katedře kybernetiky ČVUT FEL, při několika prezentacích Workshop ČVUT, Poster ČVUT FEL, na konferencích ASIS a MEDSOFT v tuzemsku a EUFIT v Aachenu.

## 5. Závěr

Identifikace psychofyziologických stavů člověka a případné uzavření ZV nachází využití v oblasti kontroly a optimálního využití potenciálu operátorů v řídících centrech (např. elektráren), pilotů či řidičů. Na druhé straně může pomoci optimálně využít paměťové a intelektuální schopnosti studentů, nebo v psychiatrii rozvinout možnosti již používaného biofeedbacku.

Počáteční konzultace postupů mapujících oblast psychofyziologických vazeb s odborníky z psychologie a fyziologie umožňuje využít dílčí výsledky nejen v oblasti uzavřené ZV u člověka, ale i ve zmíněných oborech při řešení specifických úkolů. Např. redukce počtu otázek emočních dotazníků pro snadnější testování zaměstnanců na rizikových pracovištích ve studiích psychické náročnosti.

## Literatura


[1] ***Yoshikawa, H.; Nakagawa, T.; Nakatani, Y.; Furuta, T.; Hasegawa, A.*: Development of an analysis support system for man-machine system design information**; Journal Paper, Control Engineering Practice, Elsevier, UK, 1997: Vol: 5 Iss: 3 p. 417-25.

[2] ***Washio, T.; Sakuma, M.; Kitamura, M.*: A new approach to quantitative and credible diagnosis for multiple faults of components and sensors**; Journal Paper, Artificial Intelligence, Elsevier, Netherlands, 1997: Vol: 91 Iss: 1 p. 103-30.

[3] ***Nakanishi, T.; Yikai, K.; Satoh, J.; Miyoshi, I.; Satoh, A.; Takahashi, M.*: The development of a road traffic simulation system in broad areas**; Conference Paper in Journal, Mathematics and Computers in Simulation, Elsevier, Netherlands, 1995: Vol: 39 Iss: 3-4 p. 207-12.

[4] ***Hladký, A. a spolupracovníci*: Zdravotní aspekty zátěže a stresu**; Praha, UK Karolinum, 1993.

[5] ***Kjellberg, A.; Iwanowski, S.*: Stress/Energi Formulärt: Utveckling av en metod för skattming av sinnesstmäning i arbetet**; Solna, NIOH, Undersöknigsrapport 26, 1989.

[6] ***Überla, K.*: Faktorová analýza**; Bratislava, ALFA, 1976.

[7] ***Blahuš , P.*: Faktorová analýza a její zobecňení**; Praha, SNTL, 1985.