UNIVERZITA PARDUBICE
Fakulta elektrotechniky a informatiky

Umělá neuronová síť jako nástroj pro modelování statických systémů

Martin Vančura

Bakalářská práce
2011

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: 2010/2011

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Martin VANČURA**
Osobní číslo:	**I08189**
Studijní program:	**B2646 Informační technologie**
Studijní obor:	**Informační technologie**
Název tématu:	**Umělá neuronová síť jako nástroj pro modelování statických systémů**
Zadávající katedra:	**Katedra informačních technologií**

Zásady pro vypracování:

Cíl práce: Cílem práce je vytvořit statický neuronový model konkrétního systému pomocí několika dostupných algoritmů trénování umělé neuronové sítě. Tyto algoritmy je třeba zhodnotit. Dalším cílem práce je porovnání tohoto způsobu statického modelování s klasickými přístupy (matematicko-fyzikální analýza, experimentální identifikace metodou nejmenších čtverců, ...).

Teoretická část: Student provede stručnou rešerši klasických přístupů ke statickému modelování a podrobnější rešerši informací o dopředné vícevrstvé neuronové síti a jejím využití ke statickému modelování.

Implementační část: V programovém prostředku Matlab student vytvoří grafické uživatelské prostředí sloužící k návrhu statického modelu pomocí neuronové sítě. V tomto prostředí pak navrhne model zadaného statického systému. Alternativně vytvoří model pomocí zmíněných klasických metod.

Rozsah grafických prací:

Rozsah pracovní zprávy:

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

1. HAYKIN, S. Neural Networks. New Jersey : Prentice Hall, 1999. 845 s. ISBN 0-13-273350-1
2. DRÁBEK, O., MACHÁČEK, J. Experimentální identifikace. Pardubice : VŠChT Pardubice, 1987. 275 s.

Vedoucí bakalářské práce: **Ing. Petr Doležel**
Katedra řízení procesů

Datum zadání bakalářské práce: **17. prosince 2010**

Termín odevzdání bakalářské práce: **13. května 2011**

L.S.

prof. Ing. Simeon Karamazov, Dr.
děkan

Ing. Lukáš Ogan, Ph.D.
vedoucí katedry

V Pardubicích dne 31. března 2011

Prohlášení autora

Prohlašuji, že jsem tuto práci vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Souhlasím s prezenčním zpřístupněním své práce v Univerzitní knihovně.

V Žehušicích dne 7. 8. 2011 Martin Vančura

Poděkování

Touto cestou bych chtěl poděkovat vedoucímu mé bakalářské práce, Ing. Petru Doleželovi, který mi věnoval svůj čas a poskytl mi cenné rady a doporučení pří zpracování mé bakalářské práce. A rodičům za jejich podporu při studiu.

Anotace

Tato bakalářská práce je zaměřena na umělé neuronové sítě, které jsou popsány v první části práce, a jejich využití při modelování statických systémů.

Ve druhé části je popsáno modelování především metodou nejmenších čtverců.

Ve třetí části je popsáno vlastní měření a zhodnocení výsledků.

Výsledkem této práce je grafické uživatelské prostředí v programovém prostředku MATLAB, které slouží k návrhu statického modelu pomocí neuronové sítě.

Klíčová slova

Umělé neuronové sítě, metoda nejmenších čtverců, statický systém, modelování

Title

Artificial neural network as a tool for modeling static systems.

Annotation

This thesis focuses on artificial neural networks, which are described in the first part of the work, and their use in modeling static systems.

In the second part is modeling above described method of least squares.

The third section describes the measurement and evaluation results.

The result of this work is a graphical user interface in MATLAB programming means that are used to design the static model using neural networks.

Keywords

Artificial neural network, least square method, static system, modeling

Obsah

Seznam zkratek

ADALINE	Adaptive Linear Neuron
BPG	Back-Propagation of Gradient
DARPA	Defense Advanced Research Projects Agency
INNS	International Neural Network Society
LBF	Lineární Bazické Funkce
LMS	Least Mean Square algorithm
MADALINE	Many Adaptive Linear Neuron
RBF	Radiální Bazické Funkce
UNS	Umělá Neuronová Síť

Seznam obrázků

Seznam tabulek

1 Úvodní informace

Tématem této práce jsou umělé neuronové sítě jako nástroj pro modelování statických systémů.

Hlavním cílem je vytvořit statický neuronový model konkrétního systému pomocí několika dostupných algoritmů trénování umělé neuronové sítě.

Dalším cílem práce je porovnání tohoto způsobu statického modelování s klasickými přístupy (matematicko-fyzikální analýza, experimentální identifikace metodou nejmenších čtverců).

V teoretické části mají být stručně popsány metody klasického přístupu ke statickému modelování a podrobněji zde mají být popsány dopředné vícevrstvé neuronové sítě.

V implementační části se má vytvořit grafické uživatelské prostředí sloužící k návrhu statického modelu pomocí neuronové sítě a v tomto prostředí se poté má navrhnout model zadaného statického systému.

2 Umělé neuronové sítě

Umělé neuronové sítě patří řadu let k oblastem lidského zkoumání. Inspirací pro jejich vznik byly biologické nervové sítě. Jejich základním kamenem jsou tzv. umělé neurony.

Míra zájmu o ně v průběhu minulých let narůstala i upadala. Největší rozvoj začal až s rozvojem počítačové techniky. Od té doby došlo k velkému vývoji na poli architektur a algoritmu učení, během ní si umělé neuronové sítě našli cestu k mnoha oborům.

Neuronové sítě se s úspěchem používají při aproximování požadovaných funkčních hodnot, při kontrole a řízeni různých fyzikálních veličin nebo tam, kde není znám algoritmus řešení. Využívají se ještě např. v ekonomických informačních systémech a ve zdravotnických systémech. [1]

2.1 Historie umělých neuronových sítí

V roce 1943 neurobiolog Warren McCulloch a statistik Walter Pitts popsali jednoduchý matematický model neuronu, který byl základem pro novou vědní disciplínu, k umělým neuronovým sítím. Roku 1949 poté Donald Hebb popsal zákon učení, který byl založen na změně synaptických vah při procesu učení. V roce 1958 Frank Rosenblatt zobecnil model neuronu McCulloha a Pittse pro reálný číselný obor parametrů, který pak nazval Perceptron. Navrhl také algoritmus učení pro vícevrstevnatou síť Perceptronů s dopředným šířením signálu.

Po objevení Perceptronu Bernard Widrow a Marcian E. Hoff roku 1959 odvodili a popsali neuronovou síť, která obsahovala jeden neuron s několika vstupy a doplňkovým jednotkovým signálem. Byla nazvána ADALINE (Adaptive Linear Neuron). Síť, která vznikla spojením několika těchto neuronů, dostala název MADALINE (Many Adaptive Linear Neuron).

I přes tyto úspěchy se obor umělých neuronových sítí potýkal s problémy. První problém byl, že většina badatelů přistupovala k neuronovým sítím pouze z experimentálního hlediska a jejich analytický výzkum se zanedbával. Druhý problém byl ten, že po prvotních úspěších vedlo nadšení některých vědeckých pracovníků k publikování neopodstatněných tvrzení jako např., že za několik málo let bude vyvinut umělý mozek.

Tyto skutečnosti odradili vědce a inženýry, kteří se dosud o neurovýpočty zajímali. Další rána pro neuronové sítě byla kampaň matematiků Minskeho a Paperta, kteří popsali meze zobecněných modelů, ze kterých vyvodili závěr, že neuronové sítě nemohou nahradit klasické metody. Argumentovali tím, že jeden perceptron nemůže řešit např. základní logickou funkci XOR. Připustili, že tyto nedostatky by se daly odstranit použitím

vícevrstvého perceptronu, ale parametry by se musely hledat ručně, protože v té době ještě neexistoval algoritmus učení. Tím se vývoj neuronových sítí téměř zastavil.

K novým obrovským investicím došlo až roku 1983 v USA, kde se o to zasloužila agentura DARPA(Defence Advance Research Project Association). V letech 1982-1984 Američan a světoznámý fyzik John Hopfield navrhnul nový model sítě, kde byly neurony propojeny ve stylu každý s každým, a použil názorné přirovnání s fyzikální teorií magnetických materiálů. Tato síť je známá jako tzv. Hopfieldova síť.

V roce 1986 nezávisle na sobě vědci David Rumelhart a LeCun odvodili nový algoritmus učení vrstevnatých sítí. Byl to dodnes nejpoužívanější algoritmus zpětného šíření chyby (Error Back-propagation of gradient – BPG). Dále také Fin Teuvo Kohonen přichází se samoorganizujícími neuronovými sítěmi, které ke svému učení nepotřebují učitele.

Roku 1987 se v San Diegu konala první větší konference zaměřená na umělé neuronové sítě, která měla přibližně 1700 účastníků. V tomto roce byla také založena mezinárodní společnost pro výzkum neuronových sítí INNS (International Neural Network Society). V roce 1988 začala INNS vydávat svůj časopis o Neural Networks. V této době univerzity zakládají nové výzkumné ústavy, které se zabývají neuronovými sítěmi.

V dnešní době je výzkum rozšířen po celém světě a je zaměřen hlavně na vývoj rychlejších variant algoritmů učení, na minimalizaci paměti a větším využití těchto aplikací v praxi. [1][5]

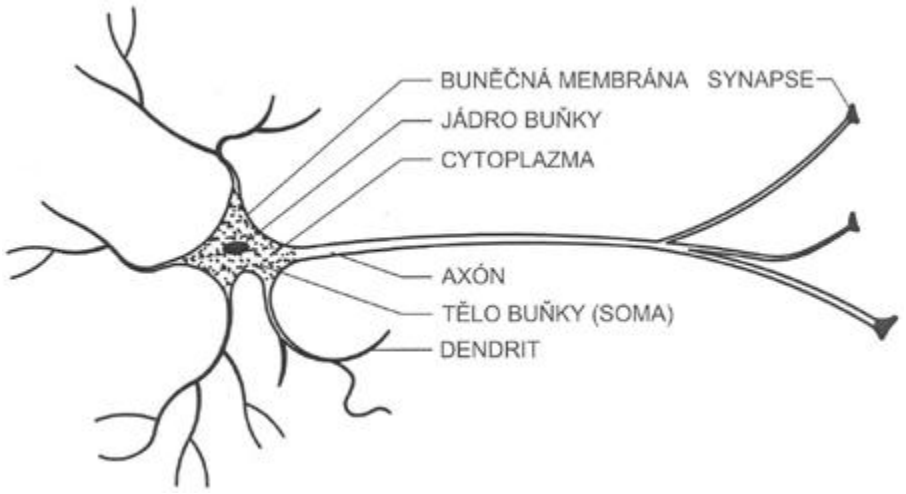
2.2 Biologický nervový systém

Biologický nervový systém tvoří celek, jenž zabezpečuje přenos, schromažďování, zpracování a následné vyhodnocení přenesených informací.

Základním prvkem biologické nervové sítě je neuron, což je vysoce specializovaná buňka sestávající z buněčného jádra, které je uloženo v těle neuronu. Dále jsou to krátké výběžky na těle neuronu zvané dendrity, které působí jako vstupy informací. V tomto případě je vstupní informací nervový vzruch od nějakého smyslového orgánu – receptoru. Dendritů je přibližně 10^4.

Další částí neuronu je axón, což je jediný výstup neuronu obalený myelinovou pochvou, která zde slouží jako izolant pro účinné a nezkreslené vedení signálu a je také příčinou barvy bíle hmoty mozkové a míchy. Tento obal je v určitých částech zúžen. Tomuto zúžení se říká Ranvierův zářez, který obnovuje sílu procházejícího signálu, protože jeho intezita klesá. Konec axonu se větví kvůli distribuci signálu, přes synapse na konci těchto větví do dendritů jiných neuronů. Tím vzniká neuronová síť.

Jeden neuron může být spojen až s 5000 dalšími neurony. Zjednodušené schéma bilogického neuronu je na obrázku Obrázek 2-1

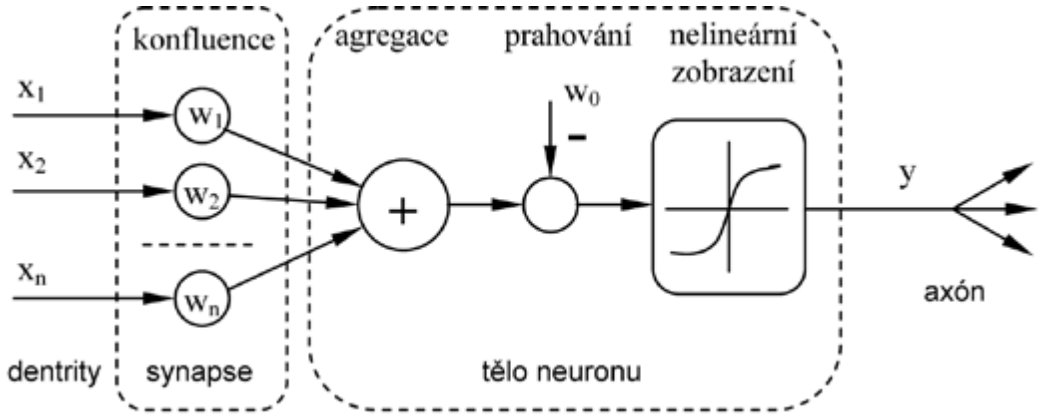


Obrázek 2-1: Zjednodušené schéma biologického neuronu[7]

K aktivaci neuronu dochází v okamžiku, kdy suma vstupních signálů přesáhne určitý práh. Neuron poté vyšle svůj výstupní signál k ostatním neuronům k nimž je připojen. Při opakovaném průchodu signálu se průchodnost synapsí zvyšuje. A naopak, když synapsí daný signál neprochází, snižuje se i průchodnost synapse. Tím je i realizován proces učení a paměti. [7][12][13][14][15][16]

2.3 Umělý neuron

Základem umělých neuronových sítí je stejně jako v biologických nervových sítích neuron. V UNS je to ale matematický model neuronu biologického. Schéma analogie biologického a umělého neuronu je zobrazeno na obrázku Obrázek 2-2.

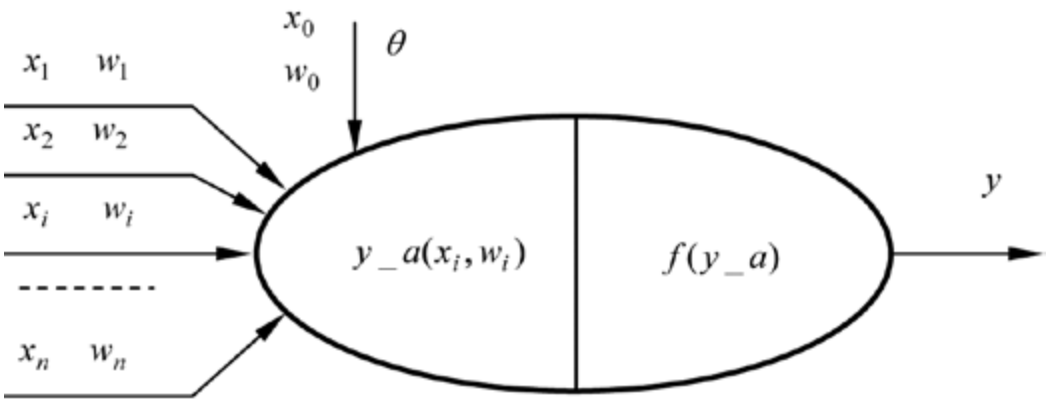


Obrázek 2-2: Schéma analogie biologického a umělého neuronu[2]

x_1, x_2,…x_n zde zobrazují vstupy modelující dendrity. w_1,w_2,...w_n jsou váhy modelující synapse a y je výstup, který simuluje činnost axonu. V těle neuronu pak dochází k agregaci vstupních signálů, porovnání s prahovou hodnotou a jejich nelineárním zobrazením aktivační funkcí, která je pak odvedena na výstup. [1][2]

2.3.1 Formální neuron

Tzv. formální model neuronu je dosud nejpoužívanějším modelem. Bývá také nazýván podle svých autorů McCulloch-Pittsuv neuron. Jeho základní schéma je uvedeno na obrázku Obrázek 2-3. [1][2]



Obrázek 2-3: Schéma umělého neuronu[2]

2.3.2 Agregační a aktivační funkce

Agregační funkce určuje, jak je v těle neuronu naloženo se vstupy. Agregační funkce určená vztahem (2.1) se používá v McCulloh-Pittsově modelu neuronu a patří do lineárních bazických funkcí. Rovnice (2.2) je další možná agregační funkce

Lineární bazické funkce (LBF)

$$u = (\sum_{i=1}^{n} w_i x_i + \Theta) \tag{2.1}$$

$$u = \prod_{i=1}^{n} w_i x_i \tag{2.2}$$

Radiální bazické funkce (RBF)

$$u = \sqrt{\prod_{i=1}^{n} (x_i - w_i)^2} \tag{2.3}$$

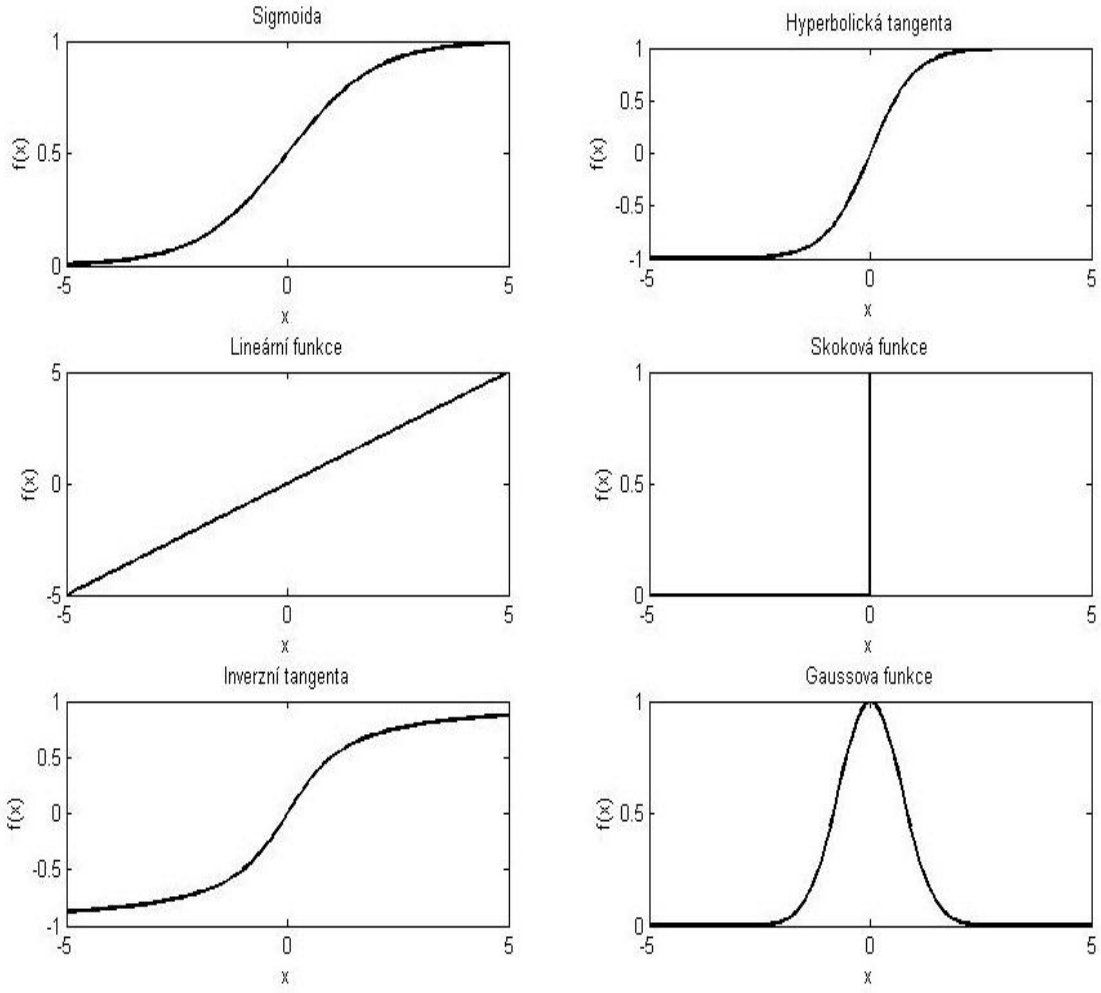
Aktivační funkce *f* určuje vztah mezi výstupem a agregační funkcí. Popsat se dá jako vztah (2.4)

$$y = f(u) \tag{2.4}$$

V tabulce Tabulka 2-1 lze vidět přehled aktivačních funkcí. Hodnota *k* určuje strmost funkce. Na obrázku Obrázek 2-4 lze vidět jejich průběhy při *k*=1. [1][2]

Tabulka 2-1: Aktivační funkce

Název funkce	Rovnice funkce
Sigmoida	$f(x) = \frac{1}{1 + e^{-kx}}$
Hyperbolická tangenta	$f(x) = \frac{2}{1 + e^{-kx}} - 1$
Lineární funkce	$f(x) = ax + b$
Gaussova funkce	$f(x) = e^{-kx^2}$
Skoková funkce	$f(x) = \begin{cases} 1, & x \geq 0 \\ 0, & x < 0 \end{cases}$
Inverzní tangenta	$f(x) = \frac{2}{\pi} \tan^{-1} kx$



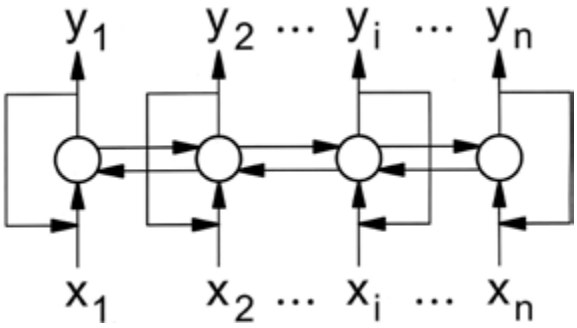
Obrázek 2-4: Průběhy aktivačních funkcí

2.4 Topologie neuronových sítí

Topologie, jinak také architektura, neuronových sítí je vlastně uspořádání a propojení neuronů v jedné vrstvě s ostatními neurony, propojení s ostatními vrstvami a propojení sama se sebou.

- Jednotlivé topologie se dají dělit podle šíření signálu:
  1) V přímém neboli dopředném směru
  2) V rekurentním směru, kde se signál, kromě dopředného směru, díky zpětným vazbám může vrátit opět na vstupy sítě. Rekurentní struktura sítě má nejméně jednu zpětnovazební smyčku.
- Nebo podle počtu vrstev:
  1) Jednovrstvé sítě

Příkladem může být Hopfieldova síť na obrázku Obrázek 2-5



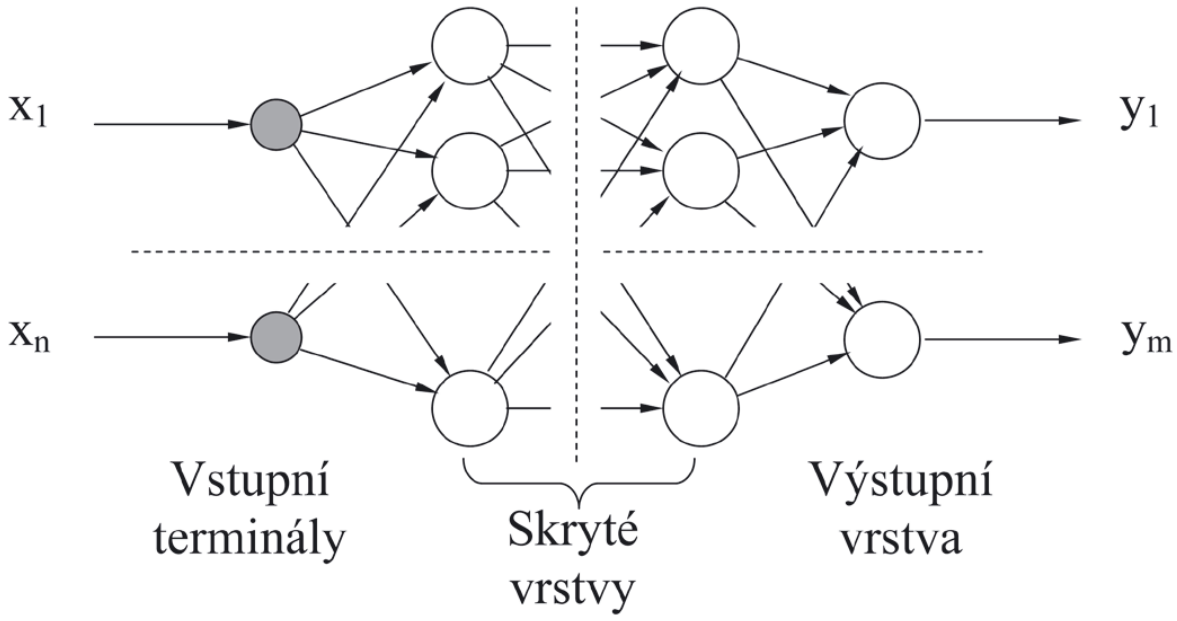
Obrázek 2-5: Hopfieldova síť[6]

2) Vrstevnaté sítě

Každá vrstevnatá síť je složena ze tří typů vrstev:

- Z jedné vstupní vrstvy
- Z jedné nebo více skrytých vrstev
- Z jedné výstupní vrstvy

Příklad vrstevnaté sítě lze nalézt na obrázku Obrázek 2-6



Obrázek 2-6: Vícevrstvá dopředná síť[11]

Konfigurace vrstevnaté sítě, která má dva neutrony ve vstupní vrstvě, dvě skryté vrstvy po čtyřech neuronech a dva neurony ve výstupní vrstvě se dá zapsat jako 2-4-4-2. [1][6]

2.5 Učení umělých neuronových sítí

Podstatnou vlastností umělých neuronových sítí je jejich schopnost se učit, která je dosažena díky analogii biologických neuronů posilováním nebo omezováním propustnosti synapsí (u UNS synaptických vah). Změnou vah a prahů se síť učí.

Způsoby učení se dají rozdělovat podle několika kriterií, ale jako základní rozdělení by se dalo označit:

1) Učení s učitelem

   V tomto případě jsou známy hodnoty požadovaných výstupů, které se během učení porovnávají s výstupem sítě, která pak nastavuje váhy a prahy, aby se výstupy přiblížily vzorům. Toto učení může probíhat dvěma způsoby, offline a online. Při učení offline jsou do sítě přivedeny všechny tréninkové vzory a až poté dochází ke změně vah. Při učení online dochází ke změnám vah po každém projití vzoru sítí.

2) Učení bez učitele

   Které se dá také popsat slovem samoorganizace, je založeno na schopnosti neuronových sítí rozeznat ve vstupních vzorech stejné nebo blízké vlastnosti a třídit přicházející vektory podle nich. Sobě podobné vektory sdružuje do shluků. Používá se na těch místech, kde není k dispozici tréninkový vzor. [1][6]

2.5.1 Hebbův zákon učení

Hebbův zákon učení se dá brát jako základ všech současných algoritmů učení. Vznikl analogií učení v biologických systémech. Pokud jsou aktivovány dva spolu propojené neurony, tak jejich vazby jsou zesíleny a naopak. Každý neuron má jen dva stavy – aktivní (1) a neaktivní (0). Tento zákon lze popsat vztahem (2.5)

$$w_{ij}(t+1) = w_{ij}(t) + \alpha y_i(t) x_j(t) \qquad (2.5)$$

Kde w_{ij} jsou prvky matice vah, α je koeficient rychlosti učení, x_j je presynaptický stav neuronu j, y_i je postsynaptický stav í-tého neuronu, t je čas. [1][6]

2.5.2 Chybové učení

Chybové učení je učení s učitelem, kde se hodnoty synaptických vah nastavují úměrně k velikosti chyby mezi požadovanými a vypočtenými hodnotami. Obecnou rovnici pro změnu vah v chybovém učení lze napsat ve tvaru (2.6)

$$\Delta w_{ij} = \alpha x_i (t_j - y_j) \tag{2.6}$$

Kde α je koeficient rychlosti učení, t_j je požadovaná hodnota, y_j je hodnota z výstupu sítě. [1]

2.6 Vícevrstvé dopředné sítě

Vícevrstvé dopředné sítě jsou tvořeny vrstvami neuronů McCulloh-Pittsova typu, které jsou propojeny jen s následující vrstvou v přímém směru šíření signálu. Mezi tyto sítě patří např. sítě typu Perceptron, ADALINE a MADALINE a sítě se zpětným šířením chyby. [1][11]

2.6.1 Vícevrstvé sítě typu Perceptron

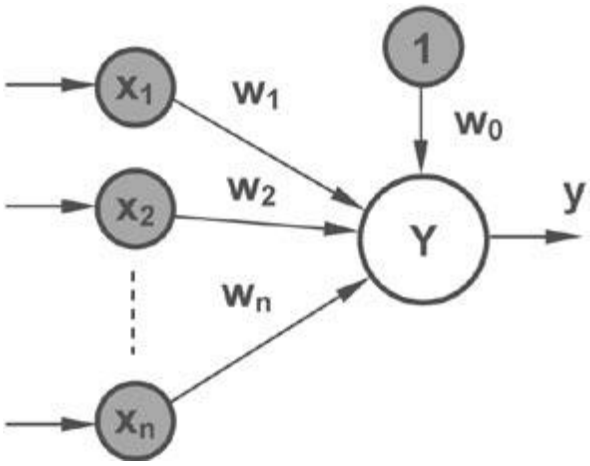
Tento model neuronu v roce 1958 popsal Frank Rosenblatt. Topologie této sítě může být tvořena buď jedním, nebo více neurony. Základ je McCulloh-Pittsův model neuronu, který používá agregační funkci (2.7)

$$u(x) = \Theta + \sum_{i=1}^{n} w_i x_i \tag{2.7}$$

a jako aktivační funkci používá skokovou funkci (2.8).

$$y(x) = \begin{cases} 1 & u(x) \geq 0 \\ 0 & u(x) < 0 \end{cases} \tag{2.8}$$

Jeho základní schéma je na obrázku Obrázek 2-7.



Obrázek 2-7: Základní schéma jednoduchého Perceptronu[6]

Učící algoritmus Perceptronu.

Po přivedení vstupního vektoru *x* na vstup sítě a vypočtení výstupu je tento výstup porovnán s požadovanou cílovou hodnotou, rozdílem je chyba *e* popsaná vztahem (2.9)

$$e = t - y \tag{2.9}$$

Je-li chyba $e = 0$, váhy se nezmění, je-li chyba $e = 1$, tak se k vahám přičte vstupní hodnota (zvýší se tak možnost klasifikace 1). Pokud je chyba $e = -1$, od vah se vstupní hodnota odečte (tím se zvýší pravděpodobnost klasifikace 0)

Adaptace vah a prahů se řídí podle vztahu (2.10) a (2.11)

$$w_{ij}(t+1) = w_{ij}(t) + e_j x_i \tag{2.10}$$

$$\Theta_{\mathrm{j}}(\mathrm{t}+1) = \Theta_{\mathrm{j}} + \mathrm{e_j} \tag{2.11}$$

Kde *t* označuje čas. [1][11]

2.6.2 ADALINE a MADALINE

ADALINE (Adaptive Linear Neuron) byla v roce 1960 odvozena a popsána pány Widrowem a Hoffem, tato síť obsahuje pouze jeden neuron, který jako aktivační funkci používá lineární funkci, takže výstup může nabývat libovolných hodnot.

Samotný tento neuron má jen omezené možnosti, proto spojením několika těchto neuronů vznikne síť, které se říká MADALINE (Many Adaptive Linear Neuron). Oproti Perceptronu má kromě aktivační funkce i jiný algoritmus učení, který se nazývá LMS algoritmus (Least Mean Squares), který je popsán rovnicemi (2.12) až (2.15):

$$w_{ij}(t+1) = w_{ij}(t) + \eta e_j(t) x_i(t) \tag{2.12}$$

$$e_j(t) = y_j(t) - c_j(t) \tag{2.13}$$

Kde $e_j(t)$ je chyba učení, $c_j(t)$ je požadovaná hodnota a $y_j(t)$ je výstupní hodnota pro kterou platí

$$y_j(t) = \Theta_j + w_{ij} x_i \tag{2.14}$$

$$\Theta_j(t+1) = \eta e_j(t) \tag{2.15}$$

Kde Θ_j je hodnota prahu, η je koeficient rychlosti učení. [1]

2.6.3 Vícevrstvé sítě se zpětným šířením chyby

Vícevrstvé sítě se zpětným šířením chyby se skládají ze vstupní vrstvy, minimálně jedné skryté vrstvy a vrstvy výstupní. Neurony v jednotlivých vrstvách mají vazby pouze na neurony v další vrstvě. Vícevrstvá dopředná síť je znázorněna na obrázku Obrázek 2-6.

Tato síť používá k učení algoritmus zpětného šíření chyb, který je znám pod zkratkou BPG. Tento algoritmus je zobecněním algoritmu LMS. Je to učení s učitelem, které může probíhat online i offline. Rozděluje se na dvě části, dopřednou část, kdy je signál šířen směrem k výstupům a zpětnou část, kdy je chyba šířena směrem ke vstupům sítě a pomocí ní se modifikují synaptické váhy a prahy. Při testování se uplatňuje jen dopředná fáze. Matematický základ algoritmu učení je popsán rovnicí (2.16)

$$W_{k+1} = W_k - \alpha_k g_k \tag{2.16}$$

kde W_k je množina všech vah a prahů, α_k je rychlost učení a g_k je gradient. [1][11]

2.6.4 Algoritmus učení zpětného šíření chyby

Algoritmus je popsán pro síť s *n* vstupy, S_1 neurony ve skryté vrstvě a S_2 neurony ve výstupní vrstvě:

1) Inicializace vah a prahů všech neuronů v síti.
2) Na vstup do sítě je přivedena vstupní hodnota z tréninkového vzoru.
3) Každý neuron v první skryté vrstvě vypočítá svůj výstup podle vztahu.

$$v_1 = \sum_{k=1}^{n} x_k w_{ik}^1 + \Theta_i^1 , \qquad \text{i=1,2,…,}S_1 \tag{2.18}$$

$$h_i = \rho(v_1) \tag{2.19}$$

Kde h_i je výstupní hodnota i-tého neuronu, ρ je aktivační funkce skryté vrstvy, x_k je k-tý vstup do neuronu, Θ_i^1 je hodnota prahu v í-tém neuronu skryté vrstvy a w^1_{ij} označuje hodnotu váhy pro í-tý neuron a k-tý vstup, *n* označuje počet vstupů do neuronu.

4) Každý neuron ve druhé výstupní vrstvě vypočítá svůj výstup podle vztahů:

$$v_2 = \sum_{j=1}^{p} h_i w_{ji}^2 + \Theta_j^2 , \qquad \text{j=1,2,…,}S_2 \tag{2.20}$$

$$y_j = \sigma(v_2) \tag{2.21}$$

Kde σ je aktivační funkce výstupní vrstvy, w_{ji}^2 označuje hodnotu váhy pro j-tý neuron a i-tý vstup z předešlé vrstvy.

5) Spočítá se chyba na výstupu podle vztahu

$$E_j = (t_j^k - y_j)\sigma'(v_2) \tag{2.22}$$

Kde t_j^k je požadovaný výstup a σ' je derivací funkce

6) Vypočítá se chyba každého neuronu ve skryté vrstvě

$$e_i = \rho'(v1) \sum_{j=1}^{S_2} w_{ij} E_j \,, \qquad \text{i=1,2,…,S}_1 \qquad (2.23)$$

Kde e_i je í-tá chyba ve skryté vrstvě a ρ' je derivací funkce

7) Adaptace vah mezi první skrytou vrstvou a druhou výstupní vrstvou

$$\Delta w_{ij}^2 = \alpha h_i E_j \,, \qquad \text{i=1,2,…,S}_1,\ \text{j=1,2,…,S}_2 \qquad (2.24)$$

Kde Δw_{ij}^2 udává velikost změny vah, α je koeficient rychlosti učení

8) Změna prahů na výstupu

$$\Delta \Theta_j^2 = \alpha E_j \,, \qquad \text{j=1,2,…,S}_2 \qquad (2.25)$$

Kde $\Delta \Theta_j^2$ je velikost změny prahu ve výstupní vrstvě

9) Adaptace vah mezi první skrytou vrstvou a vstupní vrstvou

$$\Delta w_{ik}^2 = \alpha x_k e_i \,, \qquad \text{k=1,2,…n, i=1,2,…S}_1 \qquad (2.26)$$

Kde Δw_{ik}^2 udává velikost změny vah, x_k je k-tý vstup

10) Změna prahů ve skryté vrstvě

$$\Delta \Theta_i^1 = \alpha e_j \,, \qquad \text{i=1,2,…,S}_1 \qquad (2.27)$$

11) Opakovat od kroku 2) dokud není splněna podmínka pro dovolenou velikost chyby, nebo nebyl přesažen maximální počet dovolených iterací. [1][19]

3 Modelování systémů

3.1 Základní pojmy

3.1.1 Modelování

Cílem matematického modelování je popis určitého reálného systému za použití matematických rovnic a výrazů. Nejčastěji se využívá diferenciálních rovnic nebo jejich soustav. Součástí tohoto popisu mohou být i lineární a nelineární soustavy rovnic. Jeho hlavní výhodou je možnost ověření poznatků o systémech, před jejich možnou realizací bez nebezpečí např. havárie. [8][10][17]

3.1.2 Matematický model

Je to abstraktní systém, který je zobecněním nějakého reálného systému, jeho stavů a vztahů mezi veličinami tohoto systému. Samotný matematický model ještě neumožňuje provádění experimentů. Nejdříve se musí vyřešit jeho parametry např. metodou nejmenších čtverců. [8][10][17]

3.1.3 Statický systém

Je to systém, jehož stav je pouze funkcí vstupů tzn. je nezávislý v čase, protože jeho stav bude stejný v každém časovém okamžiku. Jinak lze tento systém taky popsat jako ustálený. Obecně lze tento systém zapsat jako vztah (3.1)

$$y = f(x) \tag{3.1}$$

Kde $x = x_1, x_2, \dots, x_n$ je vektor n vstupů, $y = y_1, y_2, \dots, y_m$ je vektor m výstupů a f je funkce popisující vztah mezi nimi. [9][20]

3.2 Metody modelování

3.2.1 Metoda nejmenších čtverců

Metoda nejmenších čtverců je matematicko-statická metoda, která byla poprvé popsána Carlem Friedrichem Gaussem kolem roku 1794, který tuto metodu použil pro

výpočet oběžné dráhy trpasličí planety Ceres. Využívá se zejména pro zpracování nepřesných dat. Základní podoba této metody je určena pro řešení nekompatibilních soustav lineárních rovnic, díky čemuž je fakticky ekvivalentem tzv. lineární regresi. Tato metoda slouží k eliminaci chyb odhadovaných parametrů.

Za předpokladu, že parametry modelu, které se budou odhadovat, jsou v těchto parametrech lineární a že model má obecný tvar (3.2)

$$y_d(k) = \sum_{i=1}^{r} a_i f_i(k) \tag{3.2}$$

Kde a_i jsou odhadované parametry a f_i (k) je funkcí vstupní anebo i výstupní veličiny podle druhu modelu, který byl použit. Obecný tvar modelu umožňuje jednotné řešení odhadu parametrů jak u statických modelů, tak i u různých typů stochastických modelů více rozměrových a nelineárních soustav. Model podle rov. (3.2) může tedy být např. ve tvaru polynomu (3.3)

$$y_d(k) = a_0 + a_1 u(k) + a_2 u^2(k) \tag{3.3}$$

Index *k* v obecném modelu (3.2) má buď význam *k*-tého měření, nebo *k*-té pořadnice v čase. Předpokládejme dále, že naměřené hodnoty výstupní veličiny obsahují výstupní signál n (k), takže funkční závislost (3.2) bude splněna s určitou chybou *e* (*k*),

$$y_d(k) = \sum_{i=1}^{r} a_i f_i(k) + e(k) \tag{3.4}$$

V chybě *e*(*k*) je obsažen i případný rozdíl mezi zvoleným druhem modelu a skutečným chováním reálné soustavy. Budeme-li do rovnice (3.4) postupně dosazovat všechny naměřené hodnoty, získáme soustavu *N* rovnic o *r* neznámých. Aby byla soustava řešitelná, musí být počet naměřených hodnot *N* minimálně roven počtu neznámých *r*. Pokud bude k dispozici větší počet naměřených hodnot, než bude počet neznámých *r* (*N*>>*r*), bude se tak jednat o přeurčenou soustavu rovnic.

Při metodě nejmenších čtverců se odhady parametrů a_i získají na základě kritéria minimálního součtu kvadrátů chyby *e* (*k*) popsané rovnicí (3.5),

$$S = \sum_{k=1}^{N} e^2\,(k) = \sum_{k=1}^{N} [y(k) - \sum_{i=1}^{r} a_i f_i(k)]^2 \tag{3.5}$$

Chybová funkce S nabývá minima, jsou-li parciální derivace podle jednotlivých parametrů rovny nule. Získá se tím tak soustavu *r* rovnic o *r* neznámých a jejím řešením jsou odhady parametrů a_i.

Soustava *N* rovnic, odvozená postupným dosazováním naměřených hodnot do rovnice (3.4), má tvar (3.6):

$$y(k) = a_1 f_1(k) + a_2 f_2(k) + \cdots a_n f_n(k) \tag{3.6}$$

Definují-li se vektory:

$$\begin{array}{lll} y(1) = a_1 f_1(1) + a_2 f_2(1) & \ldots & a_n f_n(1) + e(1) \\ y(2) = a_1 f_1(2) + a_2 f_2(2) & \ldots & a_n f_n(2) + e(2) \\ \vdots & & \vdots \\ y(N) = a_1 f_1(N) + a_2 f_2(N) & \cdots & a_n f_n(N) + e(N) \end{array}$$

a matice

$$\mathbf{Y} = [y(1)\ y(2) \ldots y(N)]^T$$

$$\mathbf{A} = [a_1\ a_2 \ldots a_N]^{\boldsymbol{T}}$$

$$\mathbf{E} = [e(1)\ e(2) \ldots e(N)]^{\boldsymbol{T}}$$

Může se soustava rovnic zapsat jako (3.7).

$$\mathbf{Y} = \mathbf{FA} + \mathbf{E} \tag{3.7}$$

Chybová funkce S je podle rovnice (3.5) skalár

$$S = \mathbf{E}^T\mathbf{E} = (\mathbf{Y} - \mathbf{FA})^T(\mathbf{Y} - \mathbf{FA})$$

A má minimum pro

$$\frac{\partial S}{\partial \mathbf{A}}|_{\mathbf{A}=\hat{\mathrm{A}}} = 0$$

Dle vztahu pro derivaci součinu vektoru (3.8)

$$\frac{\partial (x^T y)}{\partial a} = \frac{\partial x^T}{\partial a} y + \frac{\partial y^T}{\partial a} x \tag{3.8}$$

Obdrží se (3.9)

$$\frac{\partial S}{\partial \hat{\mathrm{A}}} = \frac{\partial \left(\mathbf{Y} - \mathbf{F}\hat{\mathrm{A}}\right)^{\mathrm{T}}}{\partial \hat{\mathrm{A}}}\left(\mathbf{Y} - \mathbf{F}\hat{\mathrm{A}}\right) + \frac{\partial \left(\mathbf{Y} - \mathbf{F}\hat{\mathrm{A}}\right)^{\mathrm{T}}}{\partial \hat{\mathrm{A}}}\left(\mathbf{Y} - \mathbf{F}\hat{\mathrm{A}}\right) = -\mathbf{F}^{\mathrm{T}}\left(\mathbf{Y} - \mathbf{F}\hat{\mathrm{A}}\right) - \mathbf{F}^{\mathrm{T}}\left(\mathbf{Y} - \mathbf{F}\hat{\mathrm{A}}\right) =$$

$$= -\mathbf{2}\mathbf{F}^{\mathrm{T}}\left(\mathbf{Y} - \mathbf{F}\hat{\mathrm{A}}\right) = 0 \tag{3.9}$$

řešením této rovnice se získá základní maticový tvar pro odhad parametrů metody nejmenších čtverců (3.10). [3][4]

$$\hat{\mathrm{A}} = (\mathbf{F}^{\mathrm{T}}\mathbf{F})^{-1}\mathbf{F}^{\mathbf{T}}\mathbf{Y} \tag{3.10}$$

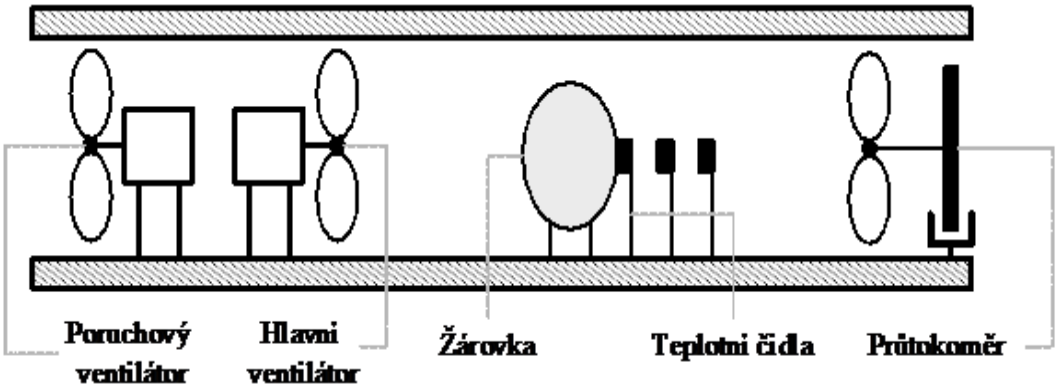
3.2.2 Matematicko-fyzikální analýza

Matematicko-fyzikální analýza se na základě podrobného zkoumání fyzikálních vlastností modelovaného systému snaží tento systém popsat a vytvořit tak matematický model, který by ho modeloval. Součástí této analýzy je také zjednodušování předpokladů a matematických závislostí. [8]

4 Vlastní modelování

4.1 Popis vybraného systému – teplovzdušného tunelu

Teplovzdušný tunel je tvořen žárovkou, která zde funguje jako zdroj tepla, hlavním a vedlejším (poruchovým) ventilátorem, který do soustavy vhání vzduch, třemi teplotními čidly v různé vzdálenosti od žárovky, a vrtulkovým průtokoměrem. Schéma soustavy je uvedeno na obrázku Obrázek 4-1.

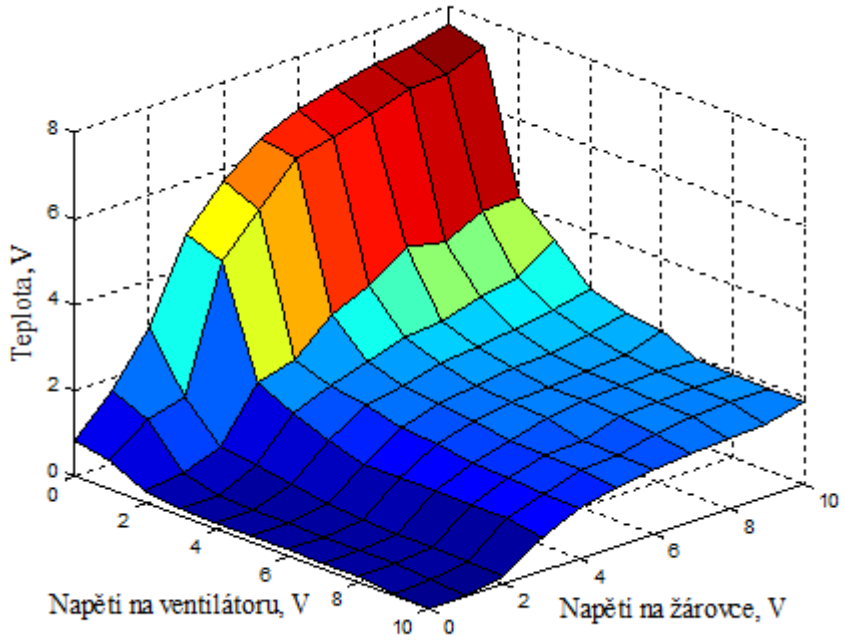


Obrázek 4-1: Schéma zvolené soustavy

U teplovzdušného modelu je možnost ovládat napětí přivedené na žárovku (0-10V), napětí přivedené na hlavní ventilátor (0-10V) a poruchový ventilátor (zapnuto/vypnuto). Snímat lze hodnoty teplot na všech třech teplotních čidlech (napětí 0-10V) a průtok vzduchu v tunelu měřený vrtulkovým průtokoměrem.

S řídicím počítačem je soustava propojena přes jednotku CTRL V3. Jednotka je napájena vlastním zdrojem elektrického napětí, je vybavena signálovým konektorem typu CANON25 a s počítačem komunikuje podle standardu RS232.

Statická charakteristika je na obrázku Obrázek 4-2. [18]



Obrázek 4-2: Statická charakteristika modelované soustavy

4.2 Modelování pomocí metody nejmenších čtverců

Modelovanou soustavu lze obecně zapsat jako rovnici (4.1) se dvěma vstupy a jedním výstupem.

$$T = f(U_z U_v) \tag{4.1}$$

U_z – napětí na žárovce
U_v – napětí na ventilátoru
T – teplota

Cílem je získat takový tvar funkce *f*, která by co nejlépe vystihovala modelovanou soustavu.

1) Tvar rovnice (4.2) pro funkci *f*

$$T = aU_z + bU_v + c \tag{4.2}$$

Matice **F** se může definovat tak, že

$$\mathbf{F} = \begin{pmatrix} U_{z1} & U_{v1} & 1 \\ \vdots & \vdots & \vdots \\ U_{zn} & U_{vn} & 1 \end{pmatrix}$$

A vektor parametrů *X*,

$$\boldsymbol{x} = \begin{pmatrix} a \\ b \\ c \end{pmatrix}$$

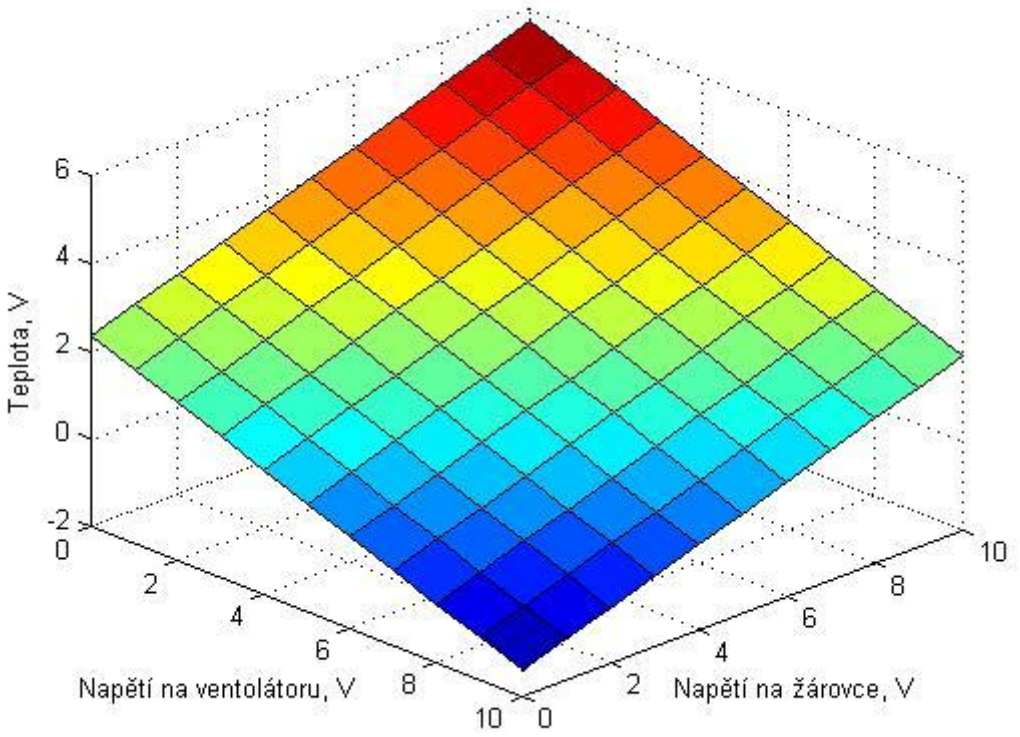
Dosazením do maticového tvaru pro odhad parametrů (3.10) je získán vektor těchto parametrů

$$\boldsymbol{x} = \begin{pmatrix} 0{,}3399 \\ -0{,}3795 \\ 2{,}3584 \end{pmatrix}$$

Potom suma nejmenších čtverců podle rovnice (3.5) bude

$$S = 143.8881$$

Na obrázku Obrázek 1 je pak možné vidět graf získaný pro tento model popsaný rovnicí (4.2)



Obrázek 4-3: Výstup získaný z modelu podle rovnice (4.2)

2) Tvar rovnice (4.3) pro funkci *f*

$$T = aU_z + bU_z^2 + cU_v + dU_v^2 + e \tag{4.3}$$

Matice **F** se může definovat tak, že

$$\mathbf{F} = \begin{pmatrix} U_{z1} & U_{z1}^2 & U_{v1} & U_{v1}^2 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ U_{zn} & U_{zn}^2 & U_{vn} & U_{vn}^2 & 1 \end{pmatrix}$$

A vektor parametrů ***x***

$$\boldsymbol{x} = \begin{pmatrix} a \\ b \\ c \\ d \\ e \end{pmatrix}$$

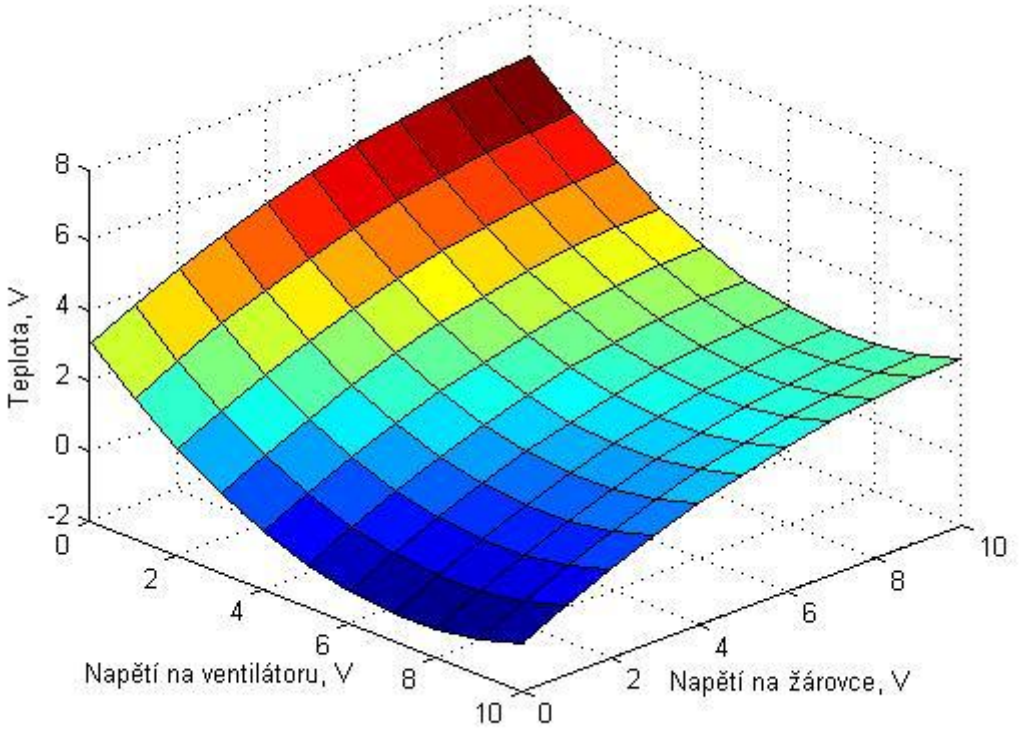
Dosazením do maticového tvaru pro odhad parametrů (3.10) je získán vektor těchto parametrů

$$\boldsymbol{x} = \begin{pmatrix} 0.6512 \\ -0.0311 \\ -1.1981 \\ 0.0819 \\ 3.1194 \end{pmatrix}$$

Potom suma nejmenších čtverců podle rovnice (3.5) bude

$$S = 71.4919$$

Na obrázku Obrázek 4-4 je pak možné vidět graf získaný pro tento model popsaný rovnicí (4.3).



Obrázek 4-4: Výstup získaný z modelu podle rovnice (4.3)

3) Tvar rovnice (4.4) pro funkci *f*

$$T = aU_z + bU_z^2 + cU_v + dU_v^2 + eU_zU_v + g \tag{4.4}$$

Matice **F** se může definovat tak, že

$$\mathbf{F} = \begin{pmatrix} U_{z1} & U_{z1}^2 & U_{v1} & U_{v1}^2 & U_{z1}U_{v1} & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ U_{zn} & U_{zn}^2 & U_{vn} & U_{vn}^2 & U_{zn}U_{vn} & 1 \end{pmatrix}$$

A vektor parametrů ***x***

$$\boldsymbol{x} = \begin{pmatrix} a \\ b \\ c \\ d \\ e \\ f \end{pmatrix}$$

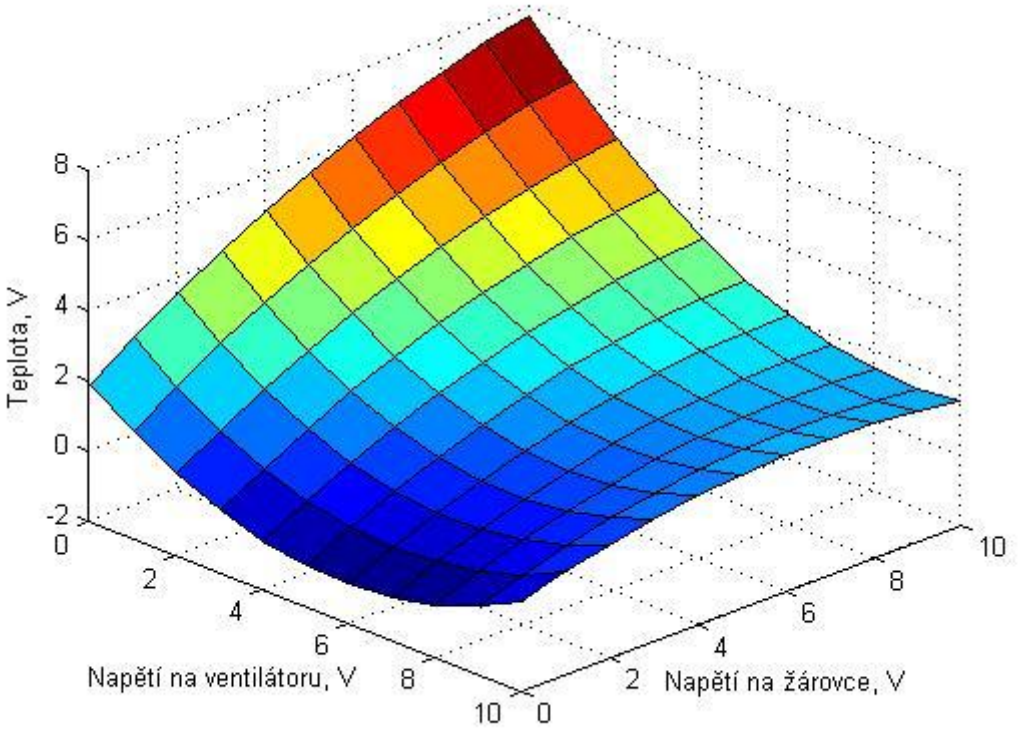
Dosazením do maticového tvaru pro odhad parametrů (3.10) je získán vektor těchto parametrů

$$\boldsymbol{x} = \begin{pmatrix} 0.8898 \\ -0.0311 \\ -0.9595 \\ 0.0819 \\ -0.0477 \\ 1.9265 \end{pmatrix}$$

Potom suma nejmenších čtverců podle rovnice (3.5) bude

$$S = 43.9428$$

Na obrázku Obrázek 4-5 je pak možné vidět graf získaný pro tento model popsaný rovnicí (4.4)



Obrázek 4-5: Výstup získaný z modelu podle rovnice (4.4)

4.3 Modelování pomocí umělých neuronových sítí

Síť měla dva vstupy a 1 výstup. Ve skryté vrstvě byla jako aktivační funkce použita sigmoida a ve výstupní vrstvě jako aktivační funkce byla použita funkce lineární. Tréninkové vzory byly vytvořeny z naměřených dat. Počet epoch byl nastaven na 20000 a při vybírání topologie sítě byl koeficient rychlosti učení $\alpha = 0{,}1$. Jako učící algoritmus byla použita základní verze algoritmu zpětného šíření chyby (BPG) s online učením, což znamená, že se váhy měnily po projití každého vzoru. Přehled topologií trénovaných sítí a jejich kvadrátů chyb spočítaných podle rovnice (4.5) lze vidět v tabulce Tabulka 4-1.

$$E = \sum_{k=1}^{n} ({}^{k}t - {}^{k}y)^2 \qquad (4.5)$$

Kde *k* označuje číslo vzoru, *t* je požadovaný výstup a *y* je získaný výstup

Tabulka 4-1: Přehled trénovaných sítí a jejich sumy kvadrátů chyb

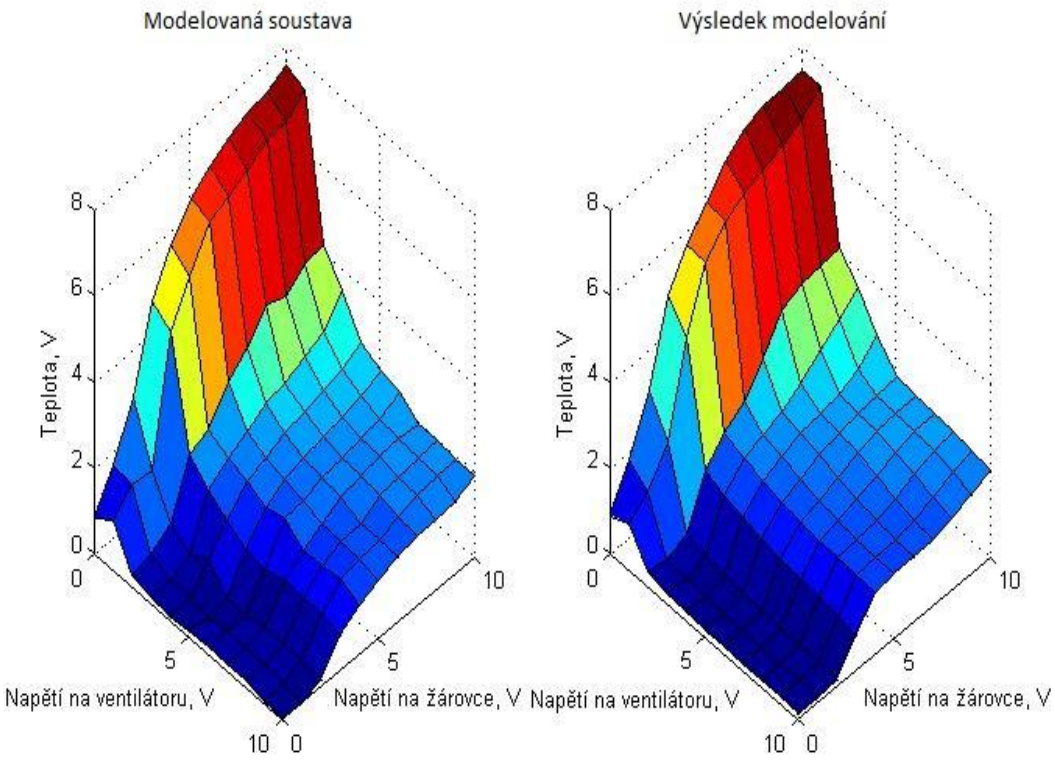
Topologie	Rychlost učení	Počet epoch	Suma kvadrátů chyb
2-6-1	0,1	20000	39,2318
2-8-1	0,1	20000	22,4779
2-10-1	0,1	20000	14.9914
2-12-1	0,1	20000	19,3365
2-15-1	0,1	20000	23.0114

Podle velikosti sumy kvadrátů chyb se zvolila síť s topologií 2-10-1. Dalším krokem bylo určit koeficient rychlosti učení. Všechny sítě s touto topologií měly stejné počáteční hodnoty synaptických vah, kvůli citlivosti algoritmu BPG na jejich inicializační hodnoty.

Tabulka 4-2: Přehled vlivu velikosti koeficientu rychlosti učení na sumu kvadrátů chyb

Topologie	Rychlost učení	Počet epoch	Suma kvadrátů chyb
2-10-1	0.01	20000	2,70319
2-10-1	0.02	20000	3,65209
2-10-1	0.05	20000	6,58096
2-10-1	0.1	20000	14,9914
2-10-1	0.15	20000	5,88305
2-10-1	0.2	20000	15,5405

Podle výsledků v tabulkách Tabulka 4-1 a Tabulka 4-2 je pro modelování této soustavy nejlepší konfigurace sítě s 10 neurony ve skryté vrstvě a koeficientem rychlosti učení 0,01. Pro grafické porovnání je na obrázku Obrázek 4-6 vlevo statická charakteristika modelované soustavy a vpravo výstup z natrénované sítě.



Obrázek 4-6: Porovnání modelované soustavy a výsledku modelování

4.4 Porovnání a zhodnocení obou metod

Tabulka 4-3: Porovnání výsledků získaných modelů

Způsob modelování	Čas realizace, s	Velikost chyby
Metoda nejmenších čtverců	0,015	43,9428
Neuronové sítě	142,2	2,7032

V tabulce

Tabulka 2-1 je možno vidět dva parametry, podle kterých budou tyto metody hodnoceny. Čas realizace u metody nejmenších čtverců udává čas potřebný pro výpočet parametrů. Čas realizace pro neuronovou síť udává čas, potřebný pro natrénování sítě. Oba časy byly změřeny pomocí profileru, který je součástí prostředí MATLAB s využitím procesoru Athlon II Phenom X4 945 3GHz, který byl mimo MATLAB zatížen jen operačním systémem.

Obě metody byly realizovány v prostředí MATLAB. Z výsledků modelování je vidět, že se umělé neuronové sítě projevily jako mocný nástroj při modelování statických systémů. Hlavní nevýhoda UNS je časová náročnost při hledání správné konfigurace sítě,

když každá síť musí projít učícím procesem. Na výslednou přesnost modelování pomocí umělých neuronových sítí může mít vliv také citlivost základního algoritmu učení BPG na inicializační hodnoty vah a prahů a koeficientu rychlosti učení. Metoda nejmenších čtverců je v tomto případě sice méně přesná, ale výpočet parametrů matematických modelů pomocí této metody trvá jen zlomky sekundy.

Modelování druhou zmíněnou metodou matematického modelování, což je matematicko-fyzikální analýza, nebylo možno provést vzhledem ke zvolené soustavě.

5 Závěr

V první části práce je podrobná rešerše o umělých neuronových sítích a dopředných vícevrstvých sítích, pomocí nichž se vytvářely statické modely určené soustavy, která je popsána v kapitole 4.1.

Ve druhé části je stručně popsáno matematické modelování a jedna z jeho metod, matematicko-fyzikální analýza. V téže kapitole je podrobněji popsána metoda nejmenších čtverců, jež byla také použita pro modelování soustavy.

Ve třetí části jsou popsány vlastní modely vytvořené pomocí UNS a zmíněné metody nejmenších čtverců. V závěru této třetí části je také zhodnocení těchto dvou metod přístupu.

V příloze A je pak stručný popis a návod k programu, vytvořeném pro implementační část zadání, ve kterém se navrhovaly neuronové modely statického systému.

Dle mého názoru se cíle vytyčené v úvodu podařilo splnit.

6 Použitá literatura

[1] TUČKOVÁ, J. Úvod do teorie a aplikací umělých neuronových sítí. Praha: Vydavatelství ČVUT, 2003 [cit. 2011-08-07]. 103s. ISBN 80-01-02800-3.

[2] TAUFER, I.; DRÁBEK, O.; SEIDL, P. Umělé neuronové sítě – teorie a aplikace (3). *CHEMagazín*, 1 (XVI), 2006 [cit. 2011-08-07], s. 12–14. ISSN 1210-7409.

[3] Metoda nejmenších čtverců. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 29. 6. 2006, last modified on 3. 8. 2011 [cit. 2011-08-07]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/Metoda_nejmen%C5%A1%C3%ADch_%C4%8Dtverc%C5%AF >.

[4] DRÁBEK, O.; MACHÁČEK, J. Experimentální identifikace. Pardubice: VŠChT Pardubice, 1987. 275 s.

[5] TAUFER, I.; DRÁBEK, O.; SEIDL, P. Umělé neuronové sítě – teorie a aplikace (1). *CHEMagazín*, 4 (XV), 2005 [cit. 2011-08-07], s. 32–34. ISSN 1210-7409.

[6] TAUFER, I.; DRÁBEK, O.; SEIDL, P. Umělé neuronové sítě – teorie a aplikace (4). *CHEMagazín*, 2 (XVI), 2006 [cit. 2011-08-07], s. 33–36. ISSN 1210-7409.

[7] TAUFER, I.; DRÁBEK, O.; SEIDL, P. Umělé neuronové sítě – teorie a aplikace (2). *CHEMagazín*, 6 (XV), 2005 [cit. 2011-08-07], s. 10–12. ISSN 1210-7409.

[8] HAVLÍČEK, Libor. *Modelování a řízení vícerozměrové soustavy*. Pardubice, 2010. 89 s. Dizertační práce. Univerzita Pardubice.

[9] HEŘMAN, Petr. *Wikiskripta.eu* [online]. Praha: 2006 [cit. 2011-08-07]. Biosignály z pohledu biofyziky. Dostupné z WWW: <http://www.wikiskripta.eu/index.php/Biosign%C3%A1ly_z_pohledu_biofyziky#Stochastick.C3.A9_syst.C3.A9my>.

[10] Počítačová simulace. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 6. 5. 2005, last modified on 22. 7. 2011 [cit. 2011-08-07]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/Počítačová_simulace>.

[11] TAUFER, I.; DRÁBEK, O.; SEIDL, P. Umělé neuronové sítě – teorie a aplikace (5). *CHEMagazín*, 5 (XVI), 2006 [cit. 2011-08-07], s. 29–31. ISSN 1210-7409.

[12] Nervová soustava. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 30. 3. 2005, last modified on 28. 6. 2011 [cit. 2011-08-07]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/Nervov%C3%A1_soustava>.

[13] Neuron. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 31. 10. 2005, last modified on 21. 7. 2011 [cit. 2011-08-07]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/Neuron>.

[14] Dendrit. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 8. 9. 2009, last modified on 17. 1. 2011 [cit. 2011-08-07]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/Dendrit>.

[15] Myelin. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 16. 8. 2009, last modified on 29. 7. 2011 [cit. 2011-08-07]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/Myelin>.

[16] Axon. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 30. 6. 2008, last modified on 21. 7. 2011 [cit. 2011-08-07]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/Axon>.

[17] Vědecké modelování. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 27. 9. 2008, last modified on 23. 12. 2009 [cit. 2011-08-07]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/V%C4%9Bdeck%C3%A9_modelov%C3%A1n%C3%AD>.

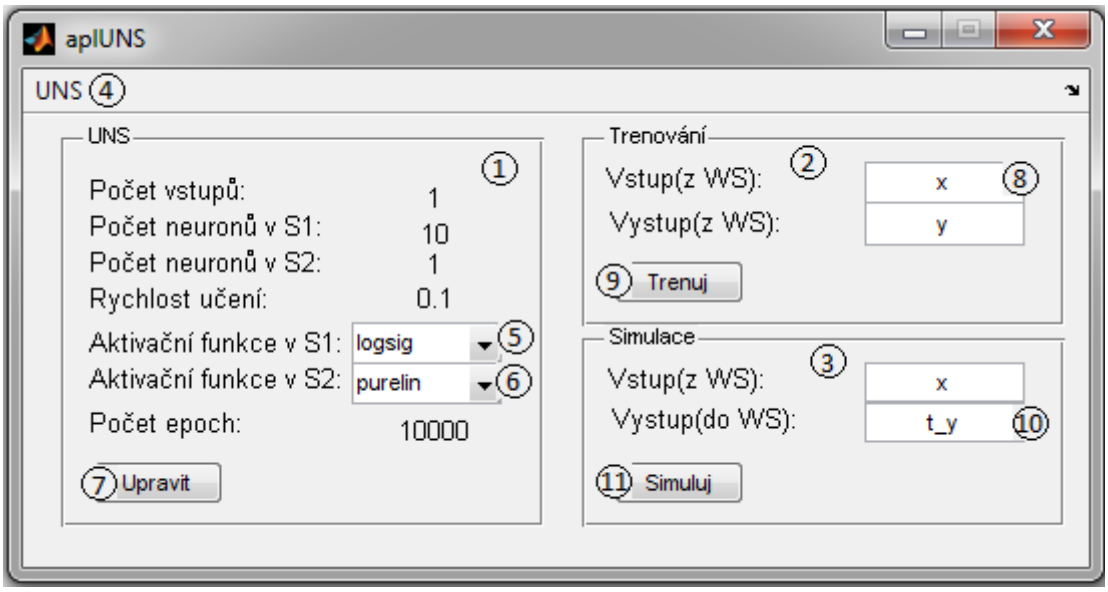
[18] *Server reálných aplikací "352LAB.vsb.cz"* [online]. 2005 [cit. 2011-08-7]. Teplovzdušný model. Dostupné z WWW: <http://www.352.vsb.cz/uc_texty/Identifikace/str/lmtva.htm>.

[19] TAUFER, I.; DRÁBEK, O.; SEIDL, P. Umělé neuronové sítě – teorie a aplikace (6). *CHEMagazín*, 6 (XVI), 2006 [cit. 2011-08-07], s. 31–33. ISSN 1210-7409.

[20] TAUFER, I.; DRÁBEK, O.; SEIDL, P. Umělé neuronové sítě – teorie a aplikace (7). *CHEMagazín*, 3 (XVII), 2007 [cit. 2011-08-07], s. 2–7. ISSN 1210-7409.

Příloha A – Stručný popis a návod k použití aplikace aplUNS

V rámci zadání této bakalářské práce byla vytvořena aplikace sloužící k návrhu statických modelů pomocí umělých neuronových sítí. Proměnné se načítají přímo z Workspace Matlabu a v případě provedení simulace se zase přímo do Workspace ukládají pod názvem určeným v panelu Simulace. Při jakékoliv změně konfigurace UNS, která se dá provést tlačítkem Upravit nebo změnou aktivačních funkcí, se vektory synaptických vah zinicializují na malé náhodné hodnoty. Natrénovanou síť lze pomocí menu UNS také uložit nebo načíst. Aplikace byla navržena a testována ve verzi MATLAB R2008a.

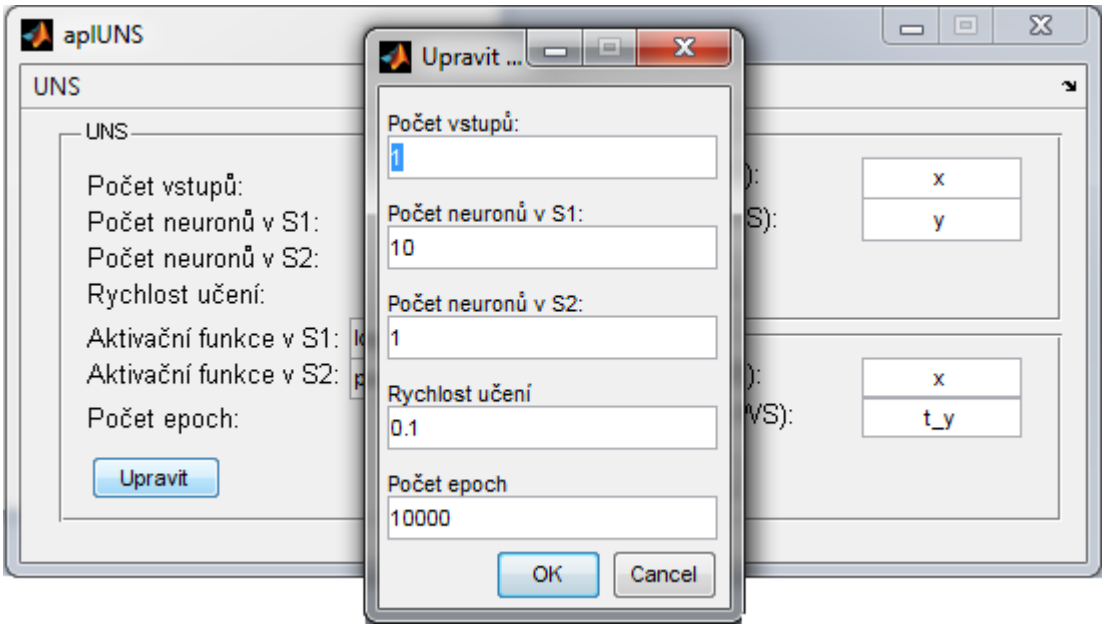


Obrázek 1:Hlavní okno aplikace

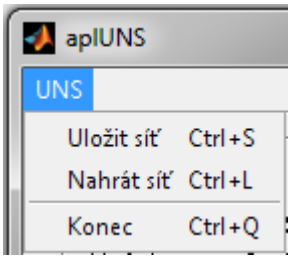
Tabulka 1: Popis komponent označených čísly na obrázku Obrázek 1

Označení	Popis
1	Panel, kde jsou zobrazeny informace o aktuální konfiguraci UNS
2	Panel pro zadání tréninkových vzorů a zahájení trénování
3	Panel pro zadání vstupních vzorů a výstupů
4	Menu, s jehož pomocí lze uložit aktuální nebo načíst UNS. Menu je možné vidět na obrázku Obrázek 3
5	Pop-up menu pro výběr aktivační funkce neuronů skryté vrstvy
6	Pop-up menu pro výběr aktivační funkce neuronů výstupní vrstvy
7	Tlačítko, které vyvolá modální dialogové okno pro úpravu neuronové sítě. Toto okno je možno vidět na obrázku Obrázek 2
8	Komponenta typu Edit pro zadání vstupní proměnné
9	Tlačítko, které zahajuje trénování

Označení	Popis
10	Komponenta typu Edit, která určuje, do jaké proměnné se mají výstupy UNS uložit
11	Tlačítko, které zahajuje výpočet



Obrázek 2:Dialogové okno pro úpravu neuronové sítě



Obrázek 3:Menu pro ukládání a načítání UNS vytvořených tímto programem

Příloha B – zdrojový kód souboru vrstvaUNS.m

```
classdef vrstvaUNS < handle
  properties(GetAccess='public',SetAccess='public')
    pVahy
    pPrahy
    pVystupyNeuronu
  end
  methods
    function obj=vrstvaUNS(pocetNeuronu,pocetVstupu)
      if(pocetNeuronu>0)
      for i=1:pocetNeuronu
        for j=1:pocetVstupu
          B(i,j)=-1+rand*2;
        end
        obj.pPrahy(i,1)=-1+rand*2;
      end
       obj.pVahy=B;
       obj.pVystupyNeuronu=zeros(1,pocetNeuronu);
      end
    end
    function obj=aktivacniFce(obj,funkce,vstupy)
      a=size(obj.pVahy);
      obj.pVystupyNeuronu=zeros(1,a(1));
      for i=1:a(1)
        b=0;
        for j=1:a(2)
          b(1,1)=b(1,1)+obj.pVahy(i,j).*vstupy(j);
        end
        suma=b+obj.pPrahy(i,1);
        switch(funkce)
          case('logsig')
            obj.pVystupyNeuronu(1,i)=logsig(suma);
          case('tansig')
            obj.pVystupyNeuronu(1,i)=tansig(suma);
          case('purelin')
            obj.pVystupyNeuronu(1,i)=purelin(suma);
        end
      end
    end
    function obj=upravVahy(obj,rychlostUceni,velikostChyby,vstupy)
      a=size(obj.pVahy);
      for i=1:a(1)
        for j=1:a(2)
obj.pVahy(i,j)=obj.pVahy(i,j)+rychlostUceni.*vstupy(j).*velikostChyby(i);
        end
obj.pPrahy(i)=obj.pPrahy(i)+rychlostUceni.*velikostChyby(i);
      end
    end
```

```
    function obj=inicializujVahy(obj)
        a=size(obj.pVahy);
        for i=1:a(1)
            for j=1:a(2)
                B(i,j)=-1+rand*2;
            end
            obj.pPrahy(i,1)=-1+rand*2;
            obj.pVahy=B;
        end
    end
  end
end
```

Příloha C – zdrojový kód souboru UNS.m

```
classdef UNS

  properties(GetAccess='public',SetAccess='public')
     sit=vrstvaUNS(0,0);
     param=0;
     alfa=0.2;
     pocetEpoch=10000;
     AFV1='logsig';
     AFV2='purelin'
  end

  methods
     function obj=UNS(Parametry)
           obj.sit(1,1)=vrstvaUNS(Parametry(2),Parametry(1));
           obj.sit(2,1)=vrstvaUNS(Parametry(3),Parametry(2));
           obj.alfa=(Parametry(4));
           obj.param=Parametry;
     end
     function v=simuluj(obj,vstup)
        a=size(vstup);
        b=size(obj.sit(2).pVahy);
        vystup=zeros(b(1),a(2));
        for i=1:a(2)
          obj.sit(1).aktivacniFce(obj.AFV1,vstup(:,i)');
          obj.sit(2).aktivacniFce(obj.AFV2,obj.sit(1).pVystupyNeuronu);
          vystup(:,i)=obj.sit(2).pVystupyNeuronu';
        end
        v=vystup;
     end
     function trenuj(obj,vstupy,vystupy)
        a=size(vstupy);
        b=size(obj.sit(2).pVahy);
           i=1;
           h=waitbar(0,'Please wait..','WindowStyle', 'modal');
           l=h;
           for k=1:obj.pocetEpoch
              waitbar(k/obj.pocetEpoch);
              if(i>a(2))
                 i=1;
              end
              vys=simuluj(obj,vstupy(:,i));

              E=(vystupy(:,i)-vys).*funkce(obj,obj.AFV2,vys);
              obj.sit(2).upravVahy(obj.alfa,E,obj.sit(1).pVystupyNeuronu);

              e=zeros(1,1);
```

```
            for j=1:b(2)
                h=obj.sit(1).pVystupyNeuronu(1,j);
                w=obj.sit(2).pVahy(:,j);
                e(j,1)=funkce(obj,obj.AFV1,h).*sum(w.*E');
            end
            vst=vstupy(:,i);
            obj.sit(1).upravVahy(obj.alfa,e,vst);
            i=i+1;
        end
        close(l);
    end
    function inicializujVahy(obj)
        obj.sit(1).inicializujVahy();
        obj.sit(2).inicializujVahy();
    end
  end
  methods(Access = private)
    function vystup=funkce(obj,funkce,h)
      switch(funkce)
            case('logsig')
                vystup=h.*(1-h);
            case('tansig')
                vystup=(1-h.^2);
            case('purelin')
                vystup=1;
            otherwise
                vystup=h.*(1-h);
      end
    end
  end

end
```

Příloha D – zdrojový kód souboru aplUNS.m

```
function varargout = aplUNS(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @aplUNS_OpeningFcn, ...
                   'gui_OutputFcn',  @aplUNS_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end

function aplUNS_OpeningFcn(hObject, eventdata, handles, varargin)
pVstupy=str2double(get(handles.text8pVstupy,'String'));
pS1=str2double(get(handles.text9pS1,'String'));
pS2=str2double(get(handles.text10pS2,'String'));
pAlfa=str2double(get(handles.text11pAlfa,'String'));
pEpoch=str2double(get(handles.text14pEpochy,'String'));
handles.sit=UNS([pVstupy pS1 pS2 pAlfa]);
handles.sit.pocetEpoch=pEpoch;
string_list1 = get(handles.popupmenu1,'String');
handles.sit.AFV1=char(string_list1(get(handles.popupmenu1,'Value')));
handles.sit.AFV2=char(string_list1(get(handles.popupmenu2,'Value')));
handles.output = hObject;
guidata(hObject, handles);

function varargout = aplUNS_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;

function listbox1_Callback(hObject, eventdata, handles)

function listbox1_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),...
    get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function pushbutton1_Callback(hObject, eventdata, handles)

function edit1_Callback(hObject, eventdata, handles)
```

```
function edit1_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function pushbuttonUpravit_Callback(hObject, eventdata, handles)
prompt = {'Počet vstupů:','Počet neuronů v S1:','Počet neuronů v S2:','Rychlost
učení','Počet epoch'};
dlg_title = 'Upravit UNS';
num_lines = 1;
def                                                                    =
{get(handles.text8pVstupy,'String'),get(handles.text9pS1,'String'),get(handles.text10pS2,'St
ring'),...
    get(handles.text11pAlfa,'String'),get(handles.text14pEpochy,'String')};
answer = inputdlg(prompt,dlg_title,num_lines,def);
if size(answer)==0
    return;
end
vstupy=str2double(getfield(answer,{1}));
S1=str2double(getfield(answer,{2}));
S2=str2double(getfield(answer,{3}));
rychlost=str2double(getfield(answer,{4}));
epochy=str2double(getfield(answer,{5}));

set(handles.text8pVstupy,'String',vstupy);
set(handles.text9pS1,'String',S1);
set(handles.text10pS2,'String',S2);
set(handles.text11pAlfa,'String',rychlost);
set(handles.text14pEpochy,'String',epochy);

handles.sit=UNS([vstupy S1 S2 rychlost]);
handles.sit.pocetEpoch=epochy;
guidata(hObject, handles);

function Untitled_1_Callback(hObject, eventdata, handles)

function popupmenu1_Callback(hObject, eventdata, handles)
string_list1 = get(handles.popupmenu1,'String');
handles.sit.AFV1=char(string_list1(get(handles.popupmenu1,'Value')));
handles.sit.inicializujVahy();
guidata(hObject, handles);

function popupmenu1_CreateFcn(hObject, eventdata, handles)
if              ispc            &&              isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function popupmenu2_Callback(hObject, eventdata, handles)
string_list1 = get(handles.popupmenu2,'String');
handles.sit.AFV2=char(string_list1(get(handles.popupmenu2,'Value')));
handles.sit.inicializujVahy();
guidata(hObject, handles);

function popupmenu2_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end

function menuSave_Callback(hObject, eventdata, handles)
[filename,pathname] = uiputfile('mojeSit.mat','Uložit síť...');
if pathname == 0
   return
end
saveDataName = fullfile(pathname,filename);
s=handles.sit;
save(saveDataName,'s');

function menuLoad_Callback(hObject, eventdata, handles)
[filename, pathname] = uigetfile('*.mat', 'Nahrát síť...');
if pathname == 0
   return
end
loadDataName = fullfile(pathname,filename);
l=load(loadDataName,'s');
handles.sit=l.s;

set(handles.text8pVstupy,'String',l.s.param(1));
set(handles.text9pS1,'String',l.s.param(2));
set(handles.text10pS2,'String',l.s.param(3));
set(handles.text11pAlfa,'String',l.s.alfa);
set(handles.text14pEpochy,'String',l.s.pocetEpoch);
v1=1;
v2=1;
switch(l.s.AFV1)
   case('purelin')
      v1=2;
   case('logsig')
      v1=1;
   case('tansig')
      v1=3;
end
switch(l.s.AFV2)
   case('purelin')
      v2=2;
```

```
    case('logsig')
        v2=1;
    case('tansig')
        v2=3;
end
set(handles.popupmenu1,'Value',v1);
set(handles.popupmenu2,'Value',v2);
guidata(hObject, handles);

function edit2_Callback(hObject, eventdata, handles)

function edit2_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function pushbutton5Tren_Callback(hObject, eventdata, handles)
try
   x=evalin('base',get(handles.edit1,'String'));
   y=evalin('base',get(handles.edit2,'String'));
catch
   errordlg('Proměnné pravděpodobně neexistují.', 'Chyba.', 'modal');
   return;
end
a=size(x);
b=size(y);
if(isnumeric(x)&&isnumeric(y)&&(a(1)==handles.sit.param(1))&&(b(1)==handles.sit.par
am(3)))
    handles.sit.trenuj(x,y);
    guidata(hObject, handles);
else
    errordlg('Něco není v pořádku', 'Chyba.', 'modal');
end
guidata(hObject, handles);

function Untitled_2_Callback(hObject, eventdata, handles)
close

function edit3_Callback(hObject, eventdata, handles)

function edit3_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit4_Callback(hObject, eventdata, handles)
```

```
function edit4_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function pushbutton6Simulace_Callback(hObject, eventdata, handles)
try
   x=evalin('base',get(handles.edit3,'String'));
catch
   errordlg('Proměnné pravděpodobně neexistují.', 'Chyba.', 'modal');
   return;
end
a=size(x);
if(isnumeric(x)&&(a(1)==handles.sit.param(1)))
    y=handles.sit.simuluj(x);
    assignin('base',get(handles.edit4,'String'),y);
else
    errordlg('Něco není v pořádku', 'Chyba.', 'modal');
end
```