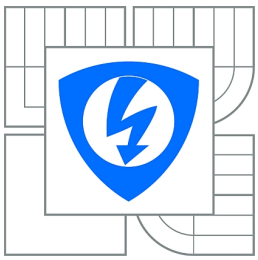


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav radioelektroniky

Diplomová práce

magisterský navazující studijní obor
Elektronika a sdělovací technika

Student:	Bc. Filip Kostka	***ID:***	125490
Ročník:	2	***Akademický rok:***	2013/2014

NÁZEV TÉMATU:

Umělá neuronová síť pro modelování polí uvnitř automobilu

POKYNY PRO VYPRACOVÁNÍ:

Seznamte se s principy fungování vybraných typů umělých neuronových sítí. Zaměřte se na typy, které jsou vhodné k pravděpodobnostnímu popisu rozložení polí uvnitř automobilů. S využitím Neural Network Toolboxu MATLABu studované sítě vytvořte a na jednoduchých testovacích příkladech ověřte jejich funkčnost.

Typy neuronových sítí, které vykázaly vlastnosti vhodné pro pravděpodobnostní modelování polí uvnitř automobilů, využijte k vytvoření komplexních modelů založených na měření intenzit polí uvnitř různých aut. Výstupy numerických modelů porovnejte s nezávislými měřeními.

DOPORUČENÁ LITERATURA:

[1] HAYKIN, S. Neural Networks: A Comprehensive Foundation, 2/E. Upper Saddle River: Prentice Hall, 1999.

[2] RAIDA, Z. Modeling EM structures in the Neural Network Toolbox of MATLAB. IEEE Antennas & Propagation Magazine, 2003, vol. 44, no. 6, p. 46-67.

Termín zadání:	10.2.2014	***Termín odevzdání:***	23.5.2014

Vedoucí práce: prof. Dr. Ing. Zbyněk Raida
Konzultanti diplomové práce:

doc. Ing. Tomáš Kratochvíl, Ph.D.
Předseda oborové rady

ABSTRAKT

Tato diplomová práce se zabývá umělými neuronovými sítěmi. Po navržení a odladění vzorové testovací a trénovací sady byly vytvořeny vícevrstvé perceptronové sítě v *Neural Network Toolbox*(NNT) Matlabu. Pro vytváření sítí byly využity různé trénovací algoritmy a algoritmy zlepšující generalizaci sítě. Při tvorbě radiální bázové sítě nebylo užito NNT. Tato síť byla vytvořena kódem v programu Matlab.

Funkčnost vytvořených sítí byla ověřena na jednoduchých trénovacích a testovacích vzorech. Reálná trénovací data byla získána simulací dvanácti monokónických anténpracujícíchna frekvencích 2 až 6 GHz. Antény byly rozmístěny uvnitř matematického modelu Octavia II. Simulací v programu CST Microwave Studio byla modelována elektromagnetická pole uvnitř automobilu.

U natrénovaných sítí zobrazujeme regresivní křivku přichycení trénovacích vzorů k síti, závislosti střední kvadratické chyby na počtu neuronů a na složitosti vstupního signálu a absolutní chybu sítě. Vlastnosti jednotlivých sítí jsou vzájemně porovnány a jsou určeny podmínky pro použití NN sítí pro modelování polí uvnitř automobilu.

KLÍČOVÁ SLOVA

Umělá neuronová síť, vícevrstvý perceptron, radiální bázová síť, Levenbergův-Marquardtův algoritmus, Bayesovská regularizace,gradientní sestup se spádem, FEKO Octavia II, učení sítě.

ABSTRACT

The project deals with artificial neural networks. After designing and debugging the test data set and the training sample set, we created a multilayer perceptron network in the *Neural NetworkToolbox* (NNT) of Matlab. When creating networks, we used different training algorithms and algorithms improving the generalization of the network. When creating a radial basis network, we did not use the NNT, but a specific source code in Matlab was written.

Functionality of neural networks was tested on simple training and testing patterns. Realistic training data were obtained by the simulation of twelve monoconic antennas operating in the frequency range from 2 to 6 GHz. Antennas were located inside a mathematical model of Octavia II. Using CST simulations, electromagnetic fields in a car were obtained.

Trained networks are described by regressive characteristics andthe mean square error of training. Algorithms improving generalization are applied on the created and trained networks. The performance of individual networks is mutually compared.

KEYWORDS

Artificial neural network, multilayer perceptron, radial basis network, Levenberg-Marquardt algorithms, Newton algorithm, learning process.

KOSTKA, F.*Umělá neuronová síť pro modelování polí uvnitř automobilu*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií. Ústav radioelektroniky, 2014. 38s., 4 s. příloh. Diplomová práce. Vedoucí práce: prof. Dr. Ing. Zbyněk Raida.

PROHLÁŠENÍ

Prohlašuji, že diplomovou práci na téma Umělá neuronová síť pro modelování polí uvnitř automobilu jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

V Brně dne ....

(podpis autora)

PODĚKOVÁNÍ

Děkuji vedoucímu semestrálního projektu prof. Dr. Ing. Zbyňku Raidovi za účinnou metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování mé diplomové práce. Dále bych chtěl poděkovat panu Ing. Vlastimilu Koudelkovi za účinnou metodickou a odbornou pomoct v problematice radiálních bázových sítí a panu Ing. Michalu Pokornému Ph.D. za odbornou pomoc při simulaci na modelu Octavia II.

V Brně dne ....

(podpis autora)

SIX research centre
sensor, information and communication systems

Faculty of Electrical Engineering
and Communication

Brno University of Technology
Technicka 12, CZ-61600 Brno, Czechia

http://www.six.feec.vutbr.cz

PODĚKOVÁNÍ

Experimentální část této diplomové práce byla realizována na výzkumné infrastruktuře vybudované v rámci projektu CZ.1.05/2.1.00/03.0072
Centrum senzorických, informačních a komunikačních systémů (SIX)
operačního programu Výzkum a vývoj pro inovace.

MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY

EVROPSKÁ UNIE
EVROPSKÝ FOND PRO REGIONÁLNÍ ROZVOJ
INVESTICE DO VAŠÍ BUDOUCNOSTI

2007–13
OP Výzkum a vývoj pro inovace

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

ÚVOD

Umělá neuronová síť je výpočetní systém, který sestává z menších dílčích výpočetních systémů (neuronů). Princip činnosti sítě je založen na soudobých poznatcích o struktuře a činnosti neuronů a nervových systémů živých organismů. V této mé diplomové práci jsou umělé neuronové sítě užity jako aproximační nástroj a následně je testována schopnost aproximovat různě složité množiny dat. Na natrénování tohoto nástroje jsou použity trénovací algoritmy typu učení s učitelem. Tyto většinou iterační algoritmy, hledají takové optimální nastavení sítě, aby byla minimalizována odchylka mezi skutečným a požadovaným výstupem sítě. Uživatel zadává parametry sítě, které jsou během učení algoritmem respektovány a značně ovlivňují kvalitu sítě

Toto téma jsem si pro svoji diplomovou práci vybral z důvodu čím dál větší aktuálnosti umělých neuronových sítí a jejich častějšímu použití v praxi. Zároveň mě zaujala podobnost umělých neuronových sítí s biologickou nervovou tkání. Neuronové sítě a trénovací algoritmy jsou známy již od 50.tých let 20. století, kdy se touto problematikou zabývali matematici McCulloch a Pitts. Avšak až v poslední době je realizace neuronových sítí možná. Došlo k tomu až v závislosti na dosažení potřebného výpočetního výkonu dnešní PC techniky.

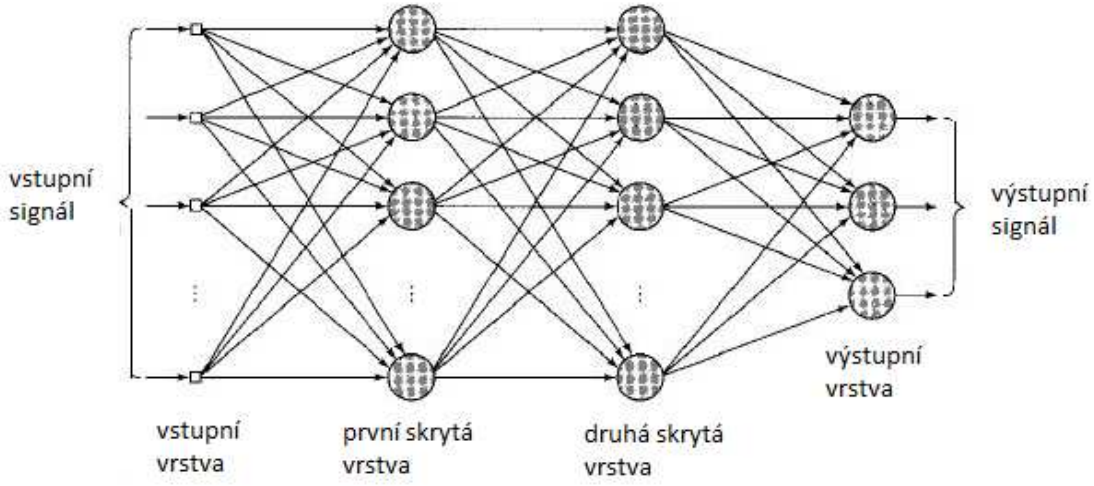
Cílem této diplomové práce je seznámit se s principy činnosti vybraných umělých neuronových sítí a jejich tvorbou pomocí Neural Network Toolbox Matlab. Dále také testováním funkčnosti sítí na jednoduchých vzorových příkladech dosáhnout možnosti následného užití sítí pro modelování polí uvnitř automobilu. Důležitou část práce tvoří vyvození závěrů o použitelnosti sítí pro modelování reálných systémů a jejich vzájemné srovnání.

1 ÚVOD DO UMĚLÝCH NEURONOVÝCH SÍTÍ A MATLAB NN TOOLBOXU

V následujícím textu je představena obecná struktura umělých neuronových sítí užitých v této diplomové práci.

1.1 Uměla neuronová síť

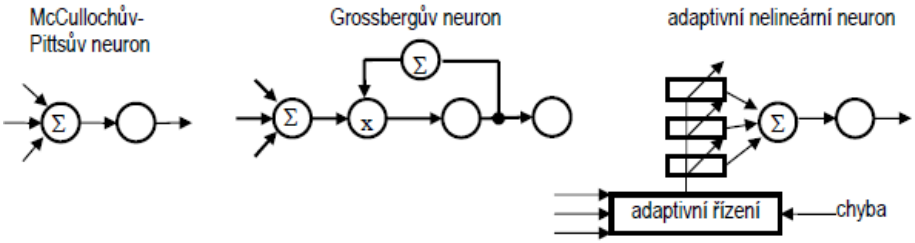
Princip činnosti umělé neuronové sítě vychází ze současných znalostí chování biologické neuronové sítě. Architektura dopředné sítě je znázorněna v grafu (viz Obr.1). Uzly grafu představují jednotlivé neurony. Spoje mezi uzly jsou chápány jako váhové spoje *w* (v biologické neuronové síti představuje synaptický spoj). Síť se dělí na několik vrstev (množin uzlů). Na vstupní, skryté a výstupní vrstvy [4].



Obr. 1 Graf architektury vícevrstvé sítě se dvěma skrytými vrstvami.

Obrázek 1 zobrazuje architekturu sítě se dvěma skrytými vrstvami. Ve většině aplikací postačuje architektura sítě pouze s jednou skrytou vrstvou. Obsahuje-li tato skrytá vrstva nelineární aktivační funkci, je dokázáno, že takovou sítí je možně zpracovávat libovolně složitou množinu dat. [1]

V současnosti existuje několik různých druhů neuronů (viz. Obr. 2). Základem všech je sumátor, který sčítá signály na jeho vstupu. Na výstupu je blok aktivační funkce neuronu. Aktivační funkce mohou být lineární nebo nelineární (viz. Obr. 7). U Grossbergova neuronu je navíc sumátor zapojen ve zpětné vazbě.



Obr. 2 Modely neuronů [podle 1]

Díky častému použití nelineárních aktivačních funkcí ve skrytých vrstvách vykazují umělé neuronové sítě velmi dobré aproximační vlastnosti. Každá úloha vyžaduje individuální přístup k výběru typu sítě a trénovacího algoritmu. Nastavováním vah spojů a prahů jednotlivých neuronů se trénovaní algoritmus snaží zajistit správnou konvergenci sítě. O tom, jak rychle a přesně síť konverguje, rozhoduje zvolený trénovací algoritmus, nastavení inicializačních parametrů sítě a složitost trénovací množiny dat. Základním požadavkem na přesnost sítě je dosažení co nejlepší odezvy sítě na případné chyby.

Princip činnosti dopředné sítě spočívá v šíření signálu od vstupu k výstupu. Tedy, vstupní vrstva neuronů přijímá vstupní signál x_i, $i = 1,\ldots,k$,, který je distribuován do skrytých vrstev. Zde dochází k jeho zpracování. Signál je dále předán do výstupní vrstvy, kde je rozhodnuto o jeho úrovni. Z této výstupní vrstvy je brán výstupní signál sítě y_i ,$i = 1,\ldots,n$. Neuronové sítě svou činností mapují (do jisté míry odhadují) závislost vstupních vzorů na výstupních vzorech. Děje se tak pomocí dané trénovaní množiny dat.

Použije-li se ve výstupní vrstvě lineární aktivační funkce, je možné pohlížet na síť jako na lineární systém i přesto, že skryté vrstvy obsahují nelineární aktivační funkce.[1]

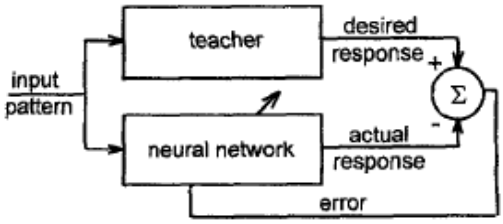
Jak již bylo řečeno, důležitou volbou při tvorbě umělé neuronové sítě je volba správného trénovacího algoritmu. V dnešní době již existuje mnoho způsobů trénování NN sítí, avšak základem všech těchto algoritmů je Hebbovo učící pravidlo. Principiálně se jedná o zesilování vazeb mezi aktivními neurony a zeslabování vazeb mezi neaktivními neurony.[1]

Správně natrénované sítě mohou vykazovat vysoké aproximační schopnosti a schopnost patřičně reagovat na chyby. Těchto schopností je možno vyžít například k návrhovým účelům.

1.2 Proces učení sítě

Pro neuronové sítě je příznačná schopnost učit se od okolí se snahou tuto nabytou zkušenost použít ve vlastní prospěch. Učení může nastávat od okolí. Takovéto učení se poté nazývá „bez učitele“. Naopak, jsou-li síti předkládány vstupní vzory s žádanou odezvou sítě, jedná se o „učení s učitelem“ (viz. Obr. 3). Vzhledem ke konkrétnímu zadání tato diplomová práce řeší pouze učení s učitelem. Během trénování je nová

informace ukládána do neuronové sítě změnou vah spojů a prahů jednotlivých neuronů.



Obr. 3 Blokové schéma trénování s učitelem [převzato z 5]

Při trénování pod dohledem (s učitelem) jsou trénovací vzory sekvenčně (nebo dávkově, *Batch mode*) předkládány síti a jejich odezvy jsou porovnány s požadovanou hodnotou. Toto je jeden z rozhodujících faktorů pro učící algoritmus při rozhodování o změně synaptických vah. Cílem algoritmů je minimalizovat chybovou funkci a přitom neuvíznout v lokálním minimu. Často je jako chybová funkce použita střední kvadratická chyba *mse* (*mean square error*) nebo *sse* (*sum of square errors*). Chybová funkce je počítána od výstupní vrstvy směrem ke vstupní vrstvě. Mluvíme tedy o zpětném šíření chyby (*Backpropagation).* U gradientních algoritmů je mezi každou sekvencí dat spočítán aktuální gradient chyby, který rozhoduje o tom, je-li nutné hodnoty patřičného neuronu dále měnit či je již ponechat beze změny.[5]

Vzhledem k rozmanitosti problémů, které jsou umělé neuronové sítě schopny řešit, není možné vytvořit jeden univerzální trénovací algoritmus pro všechny aplikace. Ze všech známých běžně používaných algoritmů se tato práce zaobírá vícevrstvými perceptronovými sítěmi, které vykázaly při testování na jednoduchém vzorovém příkladu nejlepších výsledků. Tato diplomová práce se tedy zaobírá vícevrstvými perceptronovými sítěmi trénovanými Levenbergovým-Marquardtovým optimalizačním algoritmem, Bayesovskou regularizací a gradientním sestupem se spádem. Také jsem převzal a modifikoval jednu radiální bázovou síť trénovanou Levenbergovým-Marquardtovým optimalizačním algoritmem.

1.3 NeuronovýToolbox Matlabu

Programový prostředek pro tvorbu umělých neuronových sití Neuronový toolbox (*Neural Network Toolbox*) firmy Matlab (dále jen NN Toolbox) poskytuje funkce a aplikace pro modelování nelineárních systémů pomocí umělých neuronových sítí. Poskytuje také algoritmy trénování sítě s učitelem (viztab. 1) i trénování bez učitele. NN Toolbox obsahuje grafické uživatelské rozhraní (GUI), pomocí něhož je tvorba, simulace a ladění neuronových sítí podstatně ulehčena oproti realizaci v programovacím jazyce (např. C++). Ovšem tvorba sítí v Matlabu je možná i pomocí textového kódu s využitím předdefinovaných funkcí.

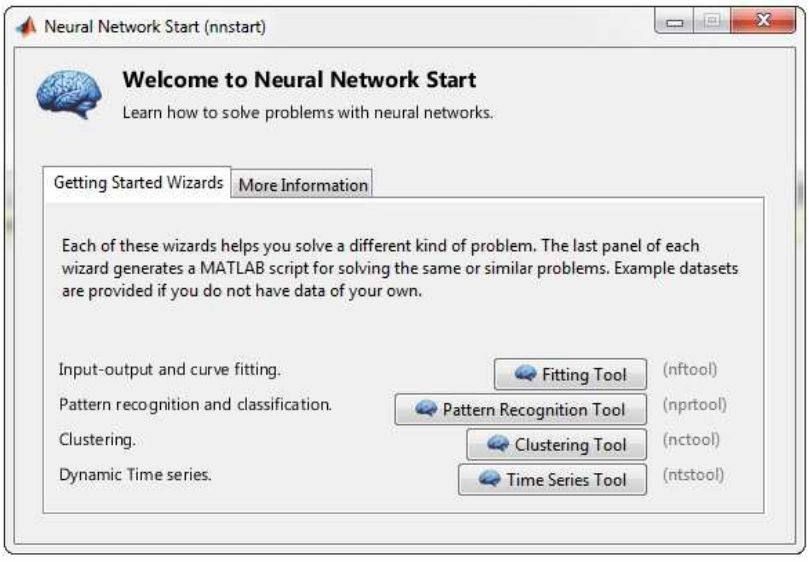
Předpokladem pro práci s NN Toolboxem jsou základní znalosti programování a teorie neuronových sítí. Uživateli je k dispozici dobře napsaná uživatelská příručka, případně nápověda, která obsahuje všechny podrobné informace, často i s ukázkou

vlastních postupů použití.

Funkce	Metoda
traingd	Steepest descent (SD)
traingdm	SD with momentum
traingda	SD with adaptive learning rate
traincfg	Conjugate gradient (Fletcher-Reeves)
traincgp	Conjugate gradient (Polak-Ribiére)
trainbfg	Quasi-Newton (QN) BFGS
trainlm	QN Levenberg-Marquardt
trainbr	Bayesian regularization

Tab. 1 Trénovací algoritmy nabízené NN Toolboxem

Spuštění NN Toolboxu je možné zadáním příkazu *nnstart* do příkazového okna Matlabu. Tímto příkazem se otevře grafické uživatelské rozhraní NN Toolboxu, kde je v první řadě potřeba zvolit, jakou problematiku bude síť řešit (viz. Obr. 4). Všechna další nastavení sítě či načtení trénovacích dat je provedeno prostřednictvím editačních oken. Pro modelování umělých neuronových sítí tedy není potřeba hlubokých znalostí programování.



Obr. 4 Úvodní okno grafického uživatelského rozhraní NN Toolboxu Matlabu

Vytvoření perceptronové sítě realizuje funkce *newp(…)*. Hodnoty volané funkce jsou parametry sítě. Např. vstupní a výstupní vzory, počet neuronů ve skryté vrstvě,

aktivační funkce a další. Po vytvoření sítě je vhodné ji inicializovat funkcí *init(…),* která realizuje reset vah a prahů sítě na defaultní hodnoty, a následně síť trénovat funkcí *train(…).* Tato funkce realizuje výpočetní smyčku, ve které jsou síti předkládány vstupní trénovací vzory. Dochází k počítání odezvy sítě, chyb a hodnot pro úpravu vah a prahů sítě pro každý vstupní vektor dat. Pro simulaci (použití) sítě je vytvořená funkce *sim(…).* Tato funkce předává hodnoty na vstup sítě a vrací k nim patřičnou odezvu sítě.[2]

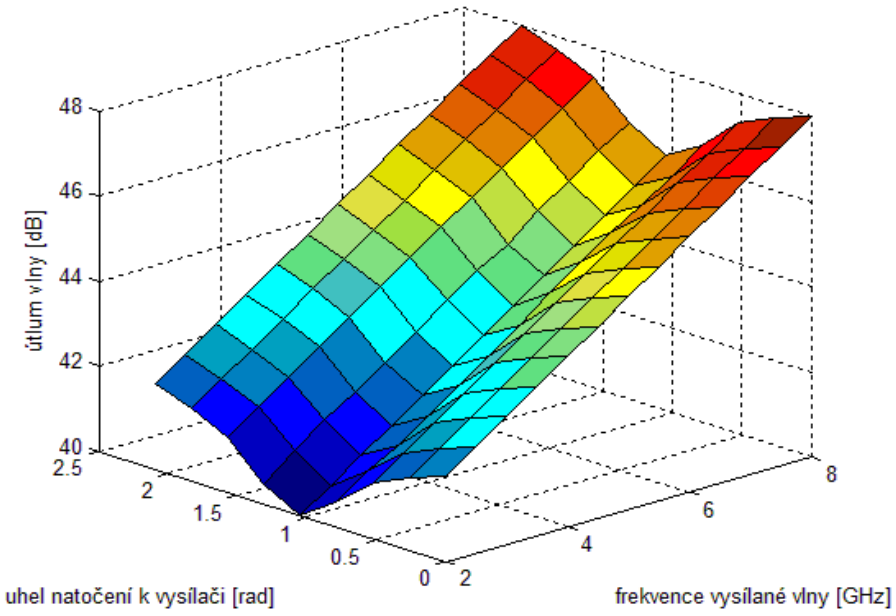
2 TRÉNOVACÍ A TESTOVACÍ VZORY

Proces trénování umělé neuronové sítě je chápán jako proces, při kterém jsou síti předkládány vstupní vzory s patřičnými výstupními odezvami. Správně natrénovaná a generalizovaná síť je potom schopna vracet i hodnoty mimo natrénované vzory a tedy vykazovat dobré aproximační vlastnosti.

2.1 Data pro ověření funkčnosti sítí

Pro potřeby otestování funkčnosti sítí byla vytvořena jednoduchá vzorová množina dat podle vzoru z článku [7]. Tato množina byla dále interpolována kubickým polynomem pro získání hodnot funkce mimo trénovací vzory, které sloužily ke zjištění kvality aproximačních vlastností sítí. V programu Matlab tuto aproximaci řeší funkce *griddata(…,'cubic')*.

Závislosti útlumu na úhlu natočení k vysílači a na frekvenci vysílané vlny, kterou chceme neuronovými sítěmi aproximovat, byla převzata z článku [7](viz obr. 5.)



Obr. 5 Graf závislosti útlumu vlny na úhlu natočení a frekvenci vlny. Vytvořen na základě [7].

Na základě testování trénovacích algoritmů vzorovou sadou dat byly vybrány algoritmy, které vykazovaly nejlepší výsledky. Vytvořené sítě trénované těmito algoritmy byly dále podrobněji testovány.

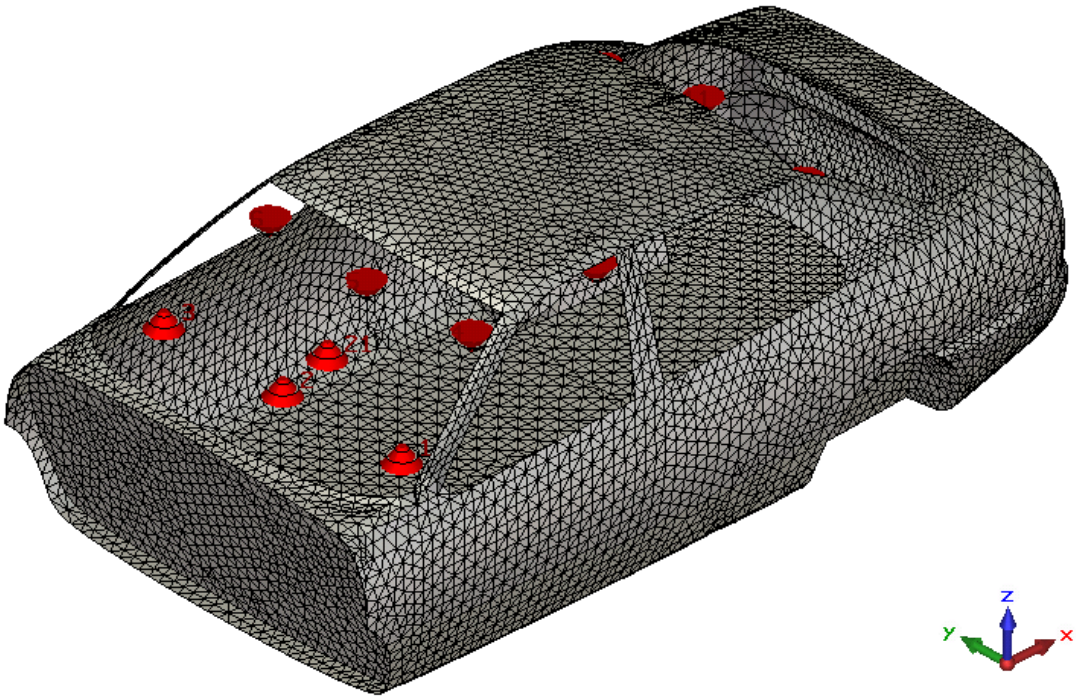
2.1 Výsledky simulace na matematickém modelu automobilu

Rozložení elektromagnetických polí uvnitř automobilu jsme reprezentovali pomocí přenosů mezi měřicími anténami. Antény uvnitř automobilu jsme simulovali na matematickém modelu vozu Octavia II v programu CST Microwave Studio. Simulace proběhla s dvanácti monokónickými anténami. Rozmístění antén uvnitř automobilu je uvedeno na Obr. 6. Výsledek simulace, tedy prvky matice 12 × 12 rozptylových parametrů, byly vyexportovány ze simulačního programu ve formátu Touchstone, a dále byly zpracovány v programu Matlab.

Prvky matice rozptylových parametrů jsou komplexní čísla. Pro uvažované neuronové modelování využijeme pouze moduly rozptylových parametrů, takže fázemi se nezabýváme.

Dále nás zajímají pouze přenosy mezi anténou na přístrojové desce vozu a anténami na pozicích cestujících.

Pro potřeby vytvoření bezdrátového spoje uvnitř automobilu, který by umožňoval přenos multimédií pro cestující a případně realizovat bezdrátovou počítačovou síť uvnitř automobilu, byla simulace provedena v pásmu 2 až 6 GHz s krokem 10MHz. Takový bezdrátový spoj, který pracuje na frekvencích blízkých zařízením WiFi (2,4GHz) a WiMax (3,5 až 5,825GHz), má předpoklad k přenosu velkých datových objemů s vysokou přenosovou rychlostí, což tato konkrétní aplikace vyžaduje.



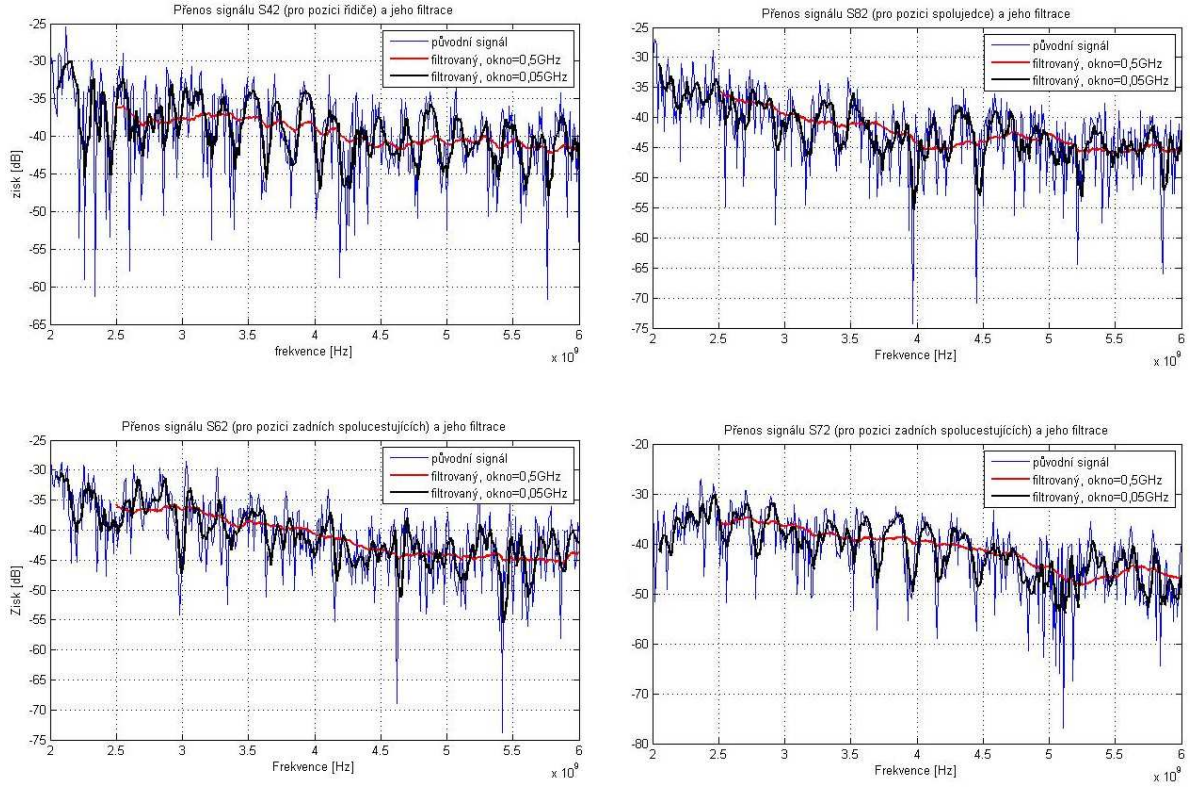
Obr. 6 Matematický model Octavia II s 12ti anténami.

Matematický model vozu je koncipován jako celokovové šasi vozu Octavia II. Uvnitř modelu automobilu dochází při simulaci k odrazům signálů a jejich vícecestnému šíření. Vícecestné šíření způsobuje velké kolísání přenosů. Toto kolísání lze eliminovat použitím průměrovacího filtru s různou velikostí průměrovacího okna (viz Obr. 6). Průměrováním signál ztrácí informaci o rychlých změnách, ovšem směrnici vývoje signálu popsané zpracování nemění. Ze znalosti kroku mezi jednotlivými hodnotami simulace (10MHz) je pak možné určit skutečnou velikost průměrovacího okna v GHz (viz Tab. 2) v závislosti na velikosti okna ve vzorcích.

Počet vzorků filtr.okna [-]	5	12	15	25	50
Skutečná velikost okna [GHz]	0,05	0,12	0,15	0,25	0,50

Tab. 2 Tabulka převodu velikosti filtračního okna na skutečnou velikost okna

Pro řešení tohoto konkrétního problému byly zvoleny přenosy mezi jednou vysílací anténou na palubové desce vozu a čtyřmi anténami (viz Obr. 7) rozmístěnými v prostoru vozu (pozice řidiče, spolujezdce a obou cestujících na zadních sedadlech). Předpokladem pro výběr pozic bylo, že každý cestující může mít na kolenou přenosný počítač, tablet či jiné multimediální zařízení s možností připojení na síť. Pokrytí interiéru vozu signálem sítě zajišťuje anténa umístěná na středu přístrojové desky vozu.



Obr. 7 Průběh přenosů a jejich filtrace pomocí oken různé velikosti

Filtrované signály byly dále rozloženy na liché a sudé vzory. Liché vzory byly použity jako trénovací a sudými byly testovány aproximační schopnosti sítí v závislosti na počtu neuronů skryté vrstvy a také velikosti filtračního okna.

2.2 Předzpracování trénovací množiny dat

Trénink umělé neuronové sítě je možné zefektivnit předzpracováním trénovací sady vzorů (vstupy a patřičné odezvy sítě). Podle konkrétní potřeby je možné použití několika různých metod normalizace dat:

- *Mapování na minimální a maximální hodnotu.* Toto mapování v programu MATLAB provádí předdefinovaná funkce *mapminmax(…).*
- *Předzpracování na střední nebo standardní odchylku.* Dalším přístupem předzpracování je normalizace střední nebo standardní odchylky trénovací množiny. Tato procedura je implementována ve funkci *prestd(…).*

Při použití jedné z výše popsaných metod dochází k transformaci celé množiny trénovacích dat a uložení transformačních koeficientů. Pomocí transformačních koeficientů je možná zpětná transformace normovaných výsledků na původní hodnotu (veličinu).[2]

V této práci se zabýváme pouze předzpracováním (mapováním) množiny dat na jejich minimální a maximální hodnotu. Toto rozhodnutí vychází z experimentálního zjištění pro tuto konkrétní aplikaci.

3 VÍCEVRSTVÉ PERCEPTRONOVÉ SÍTĚ

Vícevrstvé perceptronové sítě je možno podle jejich struktury rozdělit do několika kategorií. Tato práce se zabývá pouze dopřednými sítěmi se zpětným šířením chyby. Existují ovšem i rekurentní sítě, u kterých je výstup sítě přiveden zpět na její vstup.

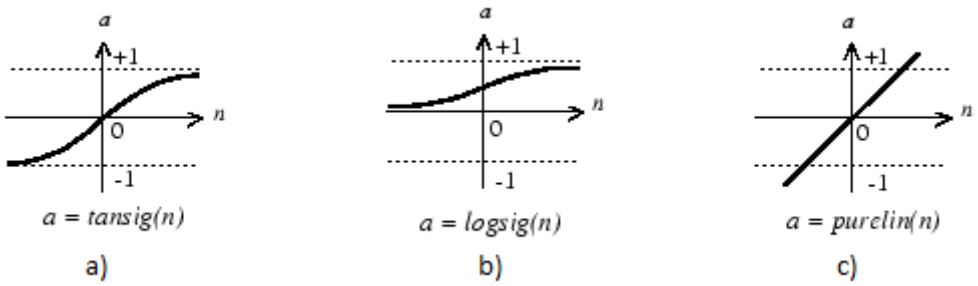
Princip činnosti dopředných sítí se zpětným šířením chyby spočívá ve dvou průchodech signálu sítí. V prvním průchodu se šíří signál od vstupu, přes skryté vrstvy, k výstupu, kde vyvolá výstupní signál (odezva sítě na vstupní signál). Váhy jednotlivých neuronů se nemění. V druhém průchodu sítí (tedy ve zpětném směru) jsou jednotlivé váhové spoje w_i měněny v závislosti na gradientu chybové funkce.

Nejrozšířenější jsou dvouvrstvé sítě, ale je možno setkat se i se čtyřvrstvými sítěmi. Bylo dokázáno, že pomocí jedné skryté vrstvy s nelineárními aktivačními funkcemi je možno aproximovat libovolně složité funkce. [1]

3.1 Uspořádání vícevrstvé perceptronové sítě

Architektury vícevrstvých perceptronových sítí (viz obr. 1), které jsou vytvářeny v této diplomové práci, odpovídají architektuře o dvou vstupních neuronech, jednom výstupním neuronu a různém počtu neuronů ve skryté vrstvě. Rozsah počtu neuronů ve skryté vrstvě je zvolen podle testu počtu efektivně využitých parametrů sítí (viz Obr. 8). Výpočet tohoto parametru má implementována Bayesovská regularizace (viz kapitola 3.4).

Na vlastnostech a možnostech umělých neuronových sítí se významně podepisuje typ použité aktivační funkce v neuronech skryté vrstvy. Při řešení tohoto konkrétního problému bylo experimentálně zjištěno, že sítě vykazují nejlepších výsledků při použití bipolární tangenciální aktivační funkce *tansig (*viz Obr. 8a). U výstupních neuronů byly zvoleny lineární funkce *purelin*(viz Obr. 5). Díky těmto lineárním aktivačním funkcím na výstupu je možno pohlížet na síť jako na lineární systém. Výběr vhodných aktivačních funkcí závisí od konkrétní aplikace.



Obr. 8 Aktivační funkce tansig(a), logsig (b) a lineární přenosová funkce (c); převzato z [2].

3.2 Trénování sítí se zpětným sířením chyby

V mnoha aplikacích s algoritmem zpětného šíření chyby je předkládáno sítí velké množství trénovací dat. Průchod celé dostupné trénovací množiny dat sítí se nazývá epocha. Trénování je založeno na průchodech sítě epochu za epochou se snahou minimalizovat střední kvadratickou chybu přes celou dostupnou množinu dat. Trénovací algoritmus hledá optimální nastavení prahů a vah sítě v souladu s nalezením minimální střední kvadratické chyby (*mse).* [1]

Pro danou trénovací množinu dat implementovanou algoritmem se zpětným šířením chyby existují dvě možnosti při výpočtu chyb:

- *Sekvenční mód*. V tomto případě je výpočet chyby *mse* počítán po průchodu každým trénovacím vzorem sítí. Následně dojde ke změně hodnot prahů a vah v souladu s hledáním minimální chyby. To se opakuje až do příchodu posledního trénovacího vzoru v epoše.
- *Dávkový mód (batch mode)*. Při tomto módu dochází k výpočtu chyby *mse*, úpravy hodnot vah a prahů až po průchodu všech trénovacích vzorů sítí, jež představují jednu epochu.

Oba dva módy mají své výhody i nevýhody. V dávkovém režimu představují problém redundantní data (opakování stejných učebních vzorů v epoše). Sekvenční mód je používán častěji, neboť zabírá méně prostoru v paměti pro každé synaptické spojení, avšak může uvíznout v lokálním minimu. Tomuto uvíznutí se dá zabránit zkoumáním konkrétního případu a následně správným nastavením parametrů trénovacího algoritmu.[1]

3.3 Zásady tvorby dopředných sítí se zpětným šířením chyby

Návrh architektury umělé neuronové sítě je do jisté míry více umění než věda. Existuje několik zásad, které by měly být při návrhu NN sítě dodrženy:

- *Sekvenční vs. dávkový mód.* Jak již bylo řečeno dříve, sekvenční předkládání trénovacích vzorů sítí je výpočetně rychlejší a je vhodné pro velké trénovací množiny s redundancí.
- *Aktivační funkce.* Při použití sigmoidální aktivační funkce (vizObr.8) v neuronech skryté vrstvy je dokázáno, že síť konverguje rychleji, je-li tato funkce navíc antisymetrická.
- *Výstupní hodnoty sítě.* Je důležité, aby výstupní hodnoty sítě (požadovaná odezva sítě) byly nalezeny v rozsahu aktivační funkce. Podle konkrétní aplikace je možné použít u výstupních neuronů lineární aktivační funkci, případně sigmoidální funkci. Je-li použita ve výstupní vrstvě sigmoidální aktivační funkce, je vhodné zajistit, aby požadovaná odezva sítě nebyla větší než limitní hodnoty této aktivační funkce.

- *Normalizace vstupních hodnot.* Před použitím trénovací množiny dat je vhodné tuto množinu předzpracovat (normovat). Jedná o transformaci množiny dat, která přiblíží střední hodnotu celé trénovací množiny nule. Další možností je normování vzorů v rozsahu 0 až 1, případně normování mezi minimální a maximální hodnotou trénovací sady vzorů.
- *Inicializace.* Správné nastavení inicializačních parametrů sítě je klíčový faktor, který přímo ovlivňuje konvergenci učícího algoritmu. Neexistuje obecné pravidlo pro nastavení parametrů. Hodnoty inicializačních parametrů jsou většinou voleny dle doporučení s korekcí pro konkrétní případ.

Podrobné informace o zásadách vytváření sítí jsou popsány v publikaci *Neural Networks: A comprehensive foundation.* [1]

3.4 Bayesovská regularizace

Na aproximační vlastnosti sítě má vliv i její zvolená architektura, tedy například počet neuronů ve skryté vrstvě. Počet skrytých neuronů je třeba zvolit tak, aby docházelo k efektivnímu využívání celé struktury sítě. K odhadu optimálního počtu skrytých neuronů je vhodné použít Bayesovskou regularizaci, která je implementována v neuronovém toolboxu MATLABu. Regularizace počítá počet efektivně využitých parametrů (prahů a vah) sítě. Je-li tento parametr příliš nízký, síť nevykazuje uspokojující aproximační vlastnosti a je potřeba zvýšit počet neuronů ve skryté vrstvě. Je-li naopak tento parametr příliš vysoký, pak se ve skryté vrstvě nachází nevyužité neurony, a v takovémto případě je vhodné počet neuronů ve skryté vrstvě snížit.[6]

Pro trénování sítě využijeme algoritmus Bayesovské regularizace, který mimo trénování sítě umožňuje také sledování počtu efektivně využitých parametrů. Podle článku [5] by neměl počet efektivně využitých parametrů u kvalitní sítě klesnout pod 50% a vzrůst nad 90%. Vývoj tohoto parametru byl sledován pro dvě různě dlouhé množiny dat o délce 201 (viz Tab. 4) a 804 (viz Tab. 3) trénovacích vzorů (viz Obr. 9). Efektivní parametr byl sledován také v závislosti na použitém filtru a počtu trénovacích vzorů. Bylo zjištěno, že tento parametr není v tomto případě závislý na velikosti průměrovacího okna ani na velikosti trénovací množiny.

Počet neuronů [-] / Velikost okna [GHz]	5	15	25	35	50
0,05	19,3	54,2	79,1	103	129
0,12	17,3	47,5	72,6	89,3	130
0,25	19,3	52	73,9	88,4	107
0,35	16	43	46,3	79,9	91,9
0,5	20,4	53,8	71,4	96,1	127
Průměr	18,5	50,1	68,7	91,3	117,0

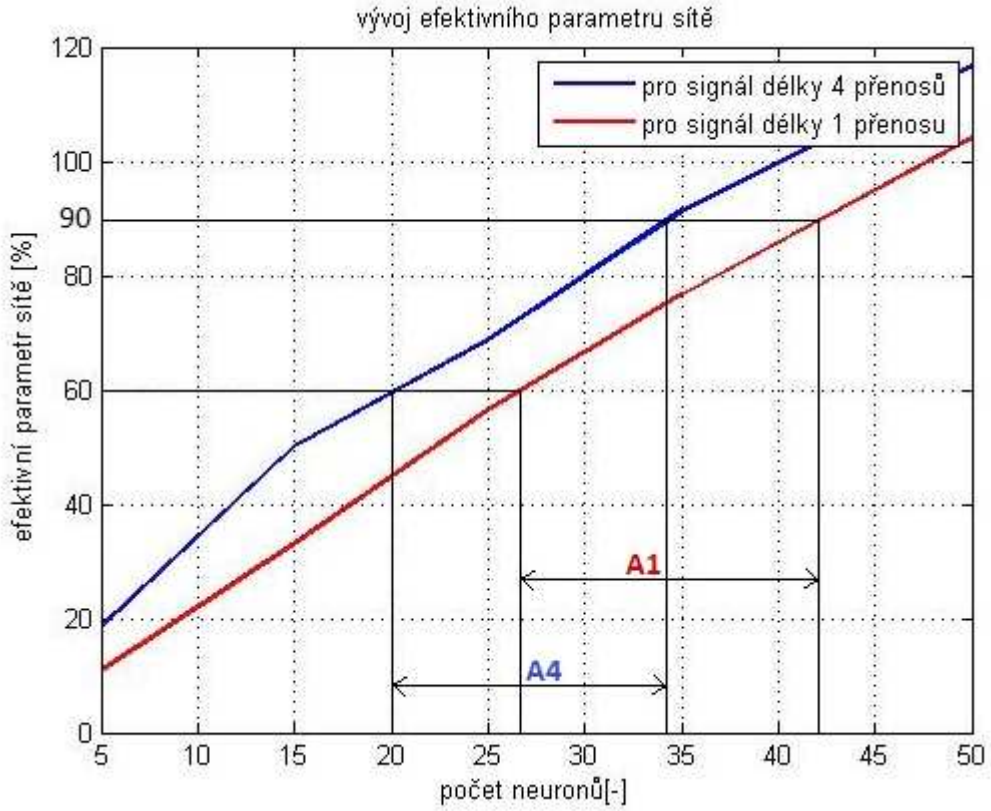
Tab. 3Tabulka vývoje počtu efektivně využitých parametrů sítě [%] na velikosti průměrovacího okna pro signál délky 201 trénovacích vzorů.

Velikost okna [GHz] \ Počet neuronů [-]	5	15	25	35	50
0,05	19,3	54,2	79,1	103	129
0,12	17,3	47,5	72,6	89,3	130
0,25	19,3	52	73,9	88,4	107
0,35	16	43	46,3	79,9	91,9
0,5	20,4	53,8	71,4	96,1	127
Průměr	18,5	50,1	68,7	91,3	117,0

Tab. 4 Tabulka vývoje počtu efektivně využitých parametrů sítě [%] na velikosti průměrovacího okna pro signál délky 804 trénovacích vzorů.

Z grafu na obrázku 9 jsou patrné meze počtu použitých neuronů ve skryté vrstvě sítě, které zajišťují platnost podmínky uvedené v předchozím odstavci. Pro množinu dat velikosti 4 přenosů (tedy 804 trénovacích vzorů) je vhodné použít 20 až 34 neuronů, aby byly efektivně využity všechny parametry sítě (prahy a váhy). Pro druhou množinu dat o velikosti pouze 1 přenosu (201 trénovacích vzorů) je vhodná volba 27 až 42 neuronů ve skryté vrstvě sítě. Na základě tohoto poznatku jsou sítě testovány, s počty neuronů ve skryté vrstvě pod i nad doporučeným počtem neuronů.

Při použití Bayesovské regularizace (viz. m-file Bayesian_regularization.m, složka Bayesovská regularizace) jsou všechny parametry tohoto algoritmu (*show*, *epoch*, *goal*, *time*, *min_grad*, *max_fail*, *mu*, *mu_inc*, *mu_dec*) nastaveny podle experimentálních zjištění v souladu s nalezením minimální chybové funkce. Implementovaný algoritmus v NNT *trainbr(…)* mění nastavení vah a prahů v souladu s Levenbergovou-Marquardtovou optimalizací. Vhodné nastavení úrovní prahů a vah sítě vede ke zlepšení její generalizace. [2]



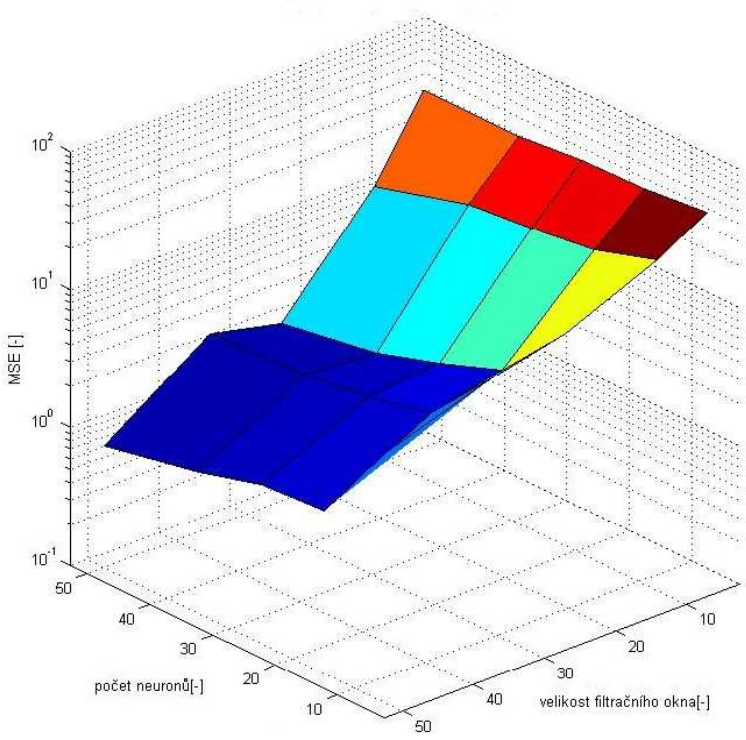
Obr. 9 Graf vývoje počtu efektivně využitých parametrů sítě pro dvě různě dlouhé množiny vstupních dat

Parametr *max_fail* určuje počet za sebou vzniklých chybných validačních kontrol. Po jeho překročení dochází k zastavení trénování.

Nastavení hodnoty *mu* je pouze inicializační hodnota. Vývoj tohoto parametru v průběhu tréninku závisí na parametrech *mu_inc a mu_dec*. Jedná se určení velikosti kroku vývoje parametru *mu* směrem nahoru (např. *mu_inc=1,9*) a směrem dolu (např. *mu_dec*=0,6).

Hodnota parametru *min_grad* určuje velikost vypočteného gradientu, při jehož překročení dochází k zastavení trénování sítě. Obvykle volíme tento parametr velmi malý (např. 10^6až 10^9). Na této hranici (nebo pod ní) můžeme uvažovat, že vývoj chyby je natolik malý, že již vývoj kvality sítě nijak neovlivní.

Byly vytvářeny sítě s 5ti až 50ti neurony ve skryté vrstvě (viz počet efektivně využitých parametrů sítě). Sítě byly trénovány množinou dat získanou průměrováním původních 4 přenosů (viz simulace Octavia II) okny velikostí 0,05 GHz až 0,50 GHz. Zaznamenávána byla minimální dosažená chybová funkce MSE a z ní následně složen graf (viz Obr. 10).

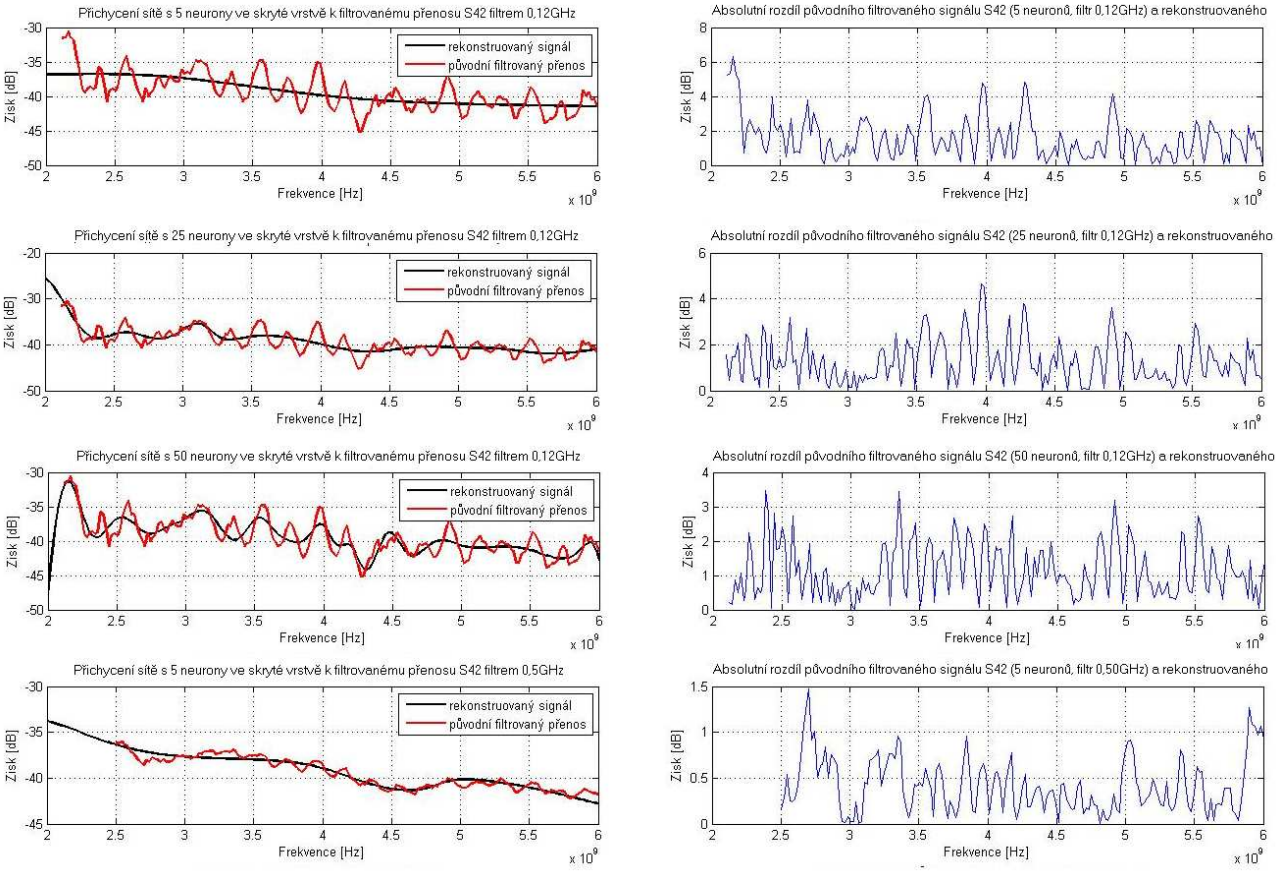


Obr. 10 Graf vývoje střední kvadratické chyby (*mse*)

Z grafu na obrázku 10 je patrný vývoj *mse* v závislosti na velikosti použitého průměrovacího okna a na počtu neuronů ve skryté vrstvě. Je zřejmé, že větší vliv na hodnotu *mse* má velikost použitého průměrovacího okna. Pokud tedy signál obsahuje méně rychlých změn, síť vykazuje dobré aproximační vlastnosti i s menším počtem neuronů.

Velikost nejnižší dosažené hodnoty *mse* je 18,5 pro architekturu 2-25-1 a trénovanou signálem průměrovaným oknem velikosti 0,5GHz. Síť byla učena lichými vzory 4 přenosů (pro pozici řidiče, spolujezdce a obou spolucestujících na zadních sedadlech) a poté byla testována sudými vzory.

Z odezvy sítě na testovací množinu je rekonstruován signál, který je následně porovnán s původním přenosem. Odečtením rekonstruovaného signálu od původního dostaneme absolutní odchylku těchto dvou signálu v celém dostupném frekvenčním pásmu (viz. Obr. 11). Rozdílu těchto dvou signálů je také možné porozumět jako chybě učení, a brát na ni ohled při rozhodování o kvalitě natrénování sítě. Stejným postupem testujeme i ostatní trénovací algoritmy.



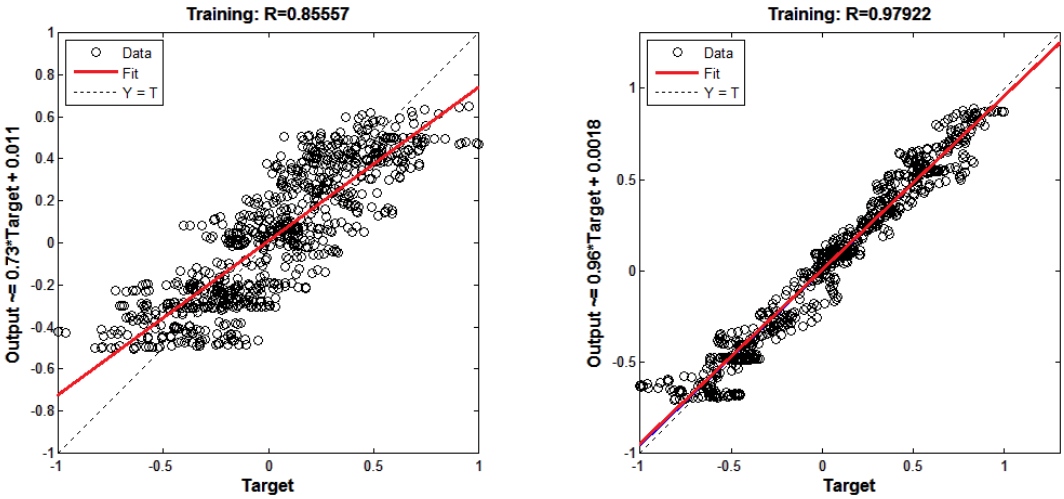
Obr. 11 Průběhy přichycení sítí k filtrovanému přenosu S42 a vzájemný rozdíl pro několik různých architektur sítě a velikostí průměrovacího okna

S těmito architekturami a velikostmi filtračních oken byly sítě testovány pro všechny 4 přenosy. Pro názornost byl použit pouze přenos S42, který odpovídá přenosu signál do pozice řidiče (viz Obr. 11).

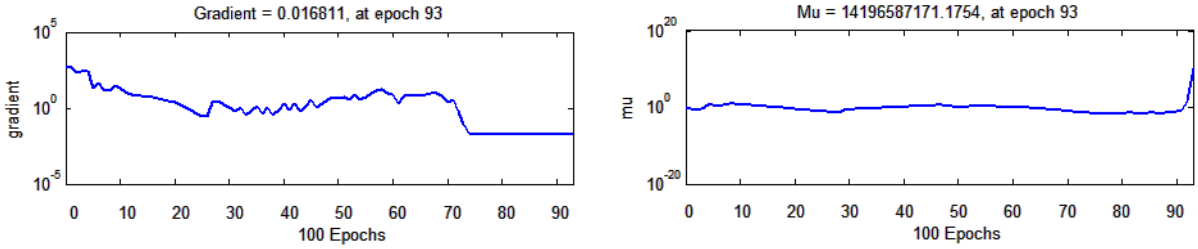
Obsahuje-li filtrovaný signál mnoho rychlých změn, síť trénovaná algoritmem Bayesovské regularizace se nedokáže k tomuto signálu dostatečně přichytit ani s větším počtem neuronů, který je v doporučeném rozmezí podle počtu efektivně využitých parametrů (viz Obr. 9). Naopak je-li signál dostatečné filtrovaný (velikost okna 0,5GHz a více), dochází ke ztrátě informace o rychlých změnách přenosu, ale tendence signálu zůstává zachována. Průběh je hladký a síť trénovaná tímto algoritmem se dokáže k filtrovanému signálu dobře přichytit i s malým počtem neuronů (viz Obr. 11 vlevo dole) a vykazovat tak malou chybu učení(viz Obr. 11 vpravo dole).

Síť s architekturou 2-5-1, trénována na signál průměrovaný oknem velikosti 0,12 GHz, prokázala nejhorší vlastnosti z testovaných variant (viz Obr. 11 vlevo nahoře). Kvalita přichycení sítě k trénovací sadě vzorů se dá také posuzovat podle regresivní křivky (viz Obr. 12). Směrnice regresivní přímky pro tuto konfiguraci (průměrovaný oknem 0,12 GHz) je 0,86 (viz Obr. 12 vlevo). Z tohoto výsledku je patrné špatné přichycení trénovacích vzorů k síti, které při testování způsobí velkou chybu. Pro případ nejlepšího přichycení sítě k trénovací sadě (viz Obr. 11 vlevo dole)

regresivní křivka dosahuje hodnoty přibližně 0,98 (viz Obr. 12 vpravo). Toto je velmi dobrá hodnota, která také potvrzuje dobré přichycení sítě k trénovacím vzorům(viz Obr. 11 vlevo dole). Takto natrénovaná síť je schopna z testovacích vzorů reprodukovat původní přenos s odchylkou okolo1 dB.



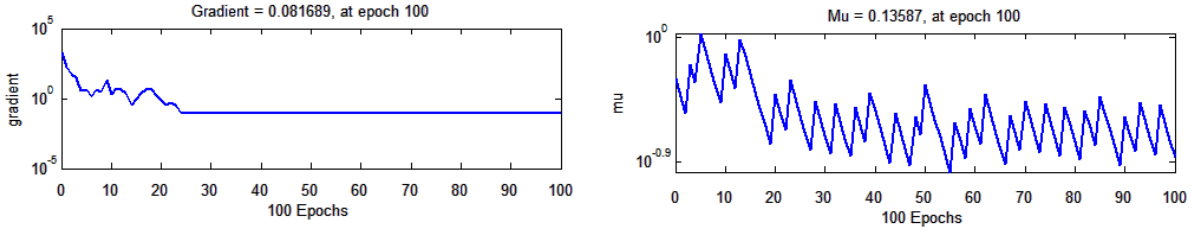
Obr. 12 Regresivní křivky pro přichycení sítě architektury 2-5-1 k přenosu průměrovaného filtrem velikosti 0,12 GHz (vlevo) a 0,50 GHz (vpravo)



Obr. 13 Vývoj gradientu a Marquardtova parametru pro síť 2-5-1 trénovanou Bayesovskou regularizací na přenosy průměrovanými oknem velikosti 0,50 GHz

Bayesovská regularizace patří mezi algoritmy, které počítají gradient (směr růstu/klesání) mezi aktuální a předchozí hodnotou chybové funkce. Je-li tento gradient příliš malý, chybová funkce se již téměř nezmenšuje. V takovém případě je vhodné učení sítě zastavit. K tomu slouží nastavitelný parametr *min_grad*. Pokud dosáhne hodnota aktuálně spočítaného gradientu zvolenou minimální mez (případně nedochází k další změně gradientu), učení je zastaveno (viz Obr. 13 vlevo). Stejně jako je zvolena hodnota minimálního gradientu, je nutné nastavit i hodnotu Marquardtova parametru *mu*. Zde je vhodné experimentálně nastavit maximální možnou hodnotu pro tento parametr (viz Obr. 13 vpravo). Za tímto maximem již nedochází k žádnému zlepšení vlastností sítě.

Pro stejnou architekturu sítě 2-5-1, ovšem trénovanou na přenosy průměrované oknem velikosti 0,12 GHz, je vývoj Marquardtova parametru (viz. kapitola 3.5) odlišný od vývoje na obrázku 13. Signál obsahuje příliš mnoho rychlých změn, síť obsahuje ve skryté vrstvě příliš málo neuronů a tedy i přichycení sítě k trénovací množině není vyhovující (viz Obr. 11 vlevo nahoře). Z vývoje tréninku této sítě jsou patrné oscilace parametru *mu*, což způsobuje špatnou konvergenci sítě.



Obr. 14 Vývoj gradientu a Marquardtova parametru pro síť 2-5-1 trénovanou přenosy průměrovanými oknem velikosti 0,12GHz

Ze zjištěných průběhů a hodnot sítí trénovaných Bayesovskou regularizací je patrné, že v případě signálu s rychlými změnami funkčních hodnot vykazují sítě nepříliš uspokojující výsledky. Použije-li se k trénovaní sítě signál, který neobsahuje rychlé změny (v našem případě průměrování oknem 0,50 GHz), je i výsledné přichycení sítě k trénovacím vzorům vyhovující (regresivní rovnice $R \geq 0{,}95$).

Největší výhodou algoritmu Bayesovské regularizace je počítání efektivního parametru sítě, který určuje počet efektivně využitých prahů a vah sítě. Toto zjištění umožňuje prvotní určení architektury sítě bez předchozích pokusů s různými velikostmi architektur.

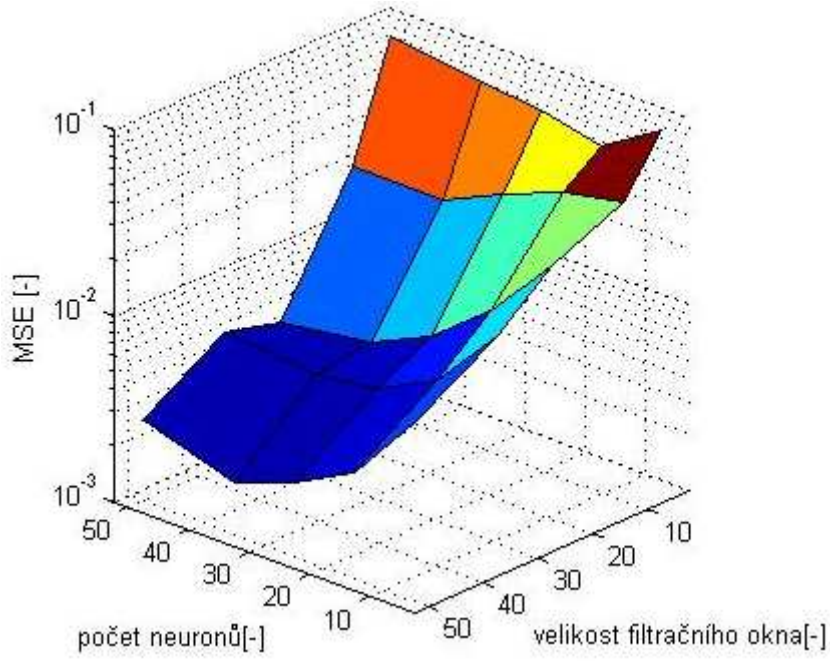
3.5 Levenbergův-Marquardtův algoritmus

Algoritmus LM patří do množiny kvazi-newtonovských algoritmů. Tyto algoritmy nepočítají přímo Hessovu matici, ale postupně zpřesňují její odhad. Tím je dosaženo vysoké efektivnosti a výborných konvergenčních vlastností současně.

Inicializace parametrů Levenbergova-Marquardtova algoritmu (viz m-file *Levenberg_Marquardt.m*, složka Levenberg-Marquardt) probíhá obdobně jako u předchozího algoritmu. Ovšem tento učící algoritmus pracuje s parametry *show*, *epochs*, *goal*, *time*, *min_grad*, *max_fail*, *mu*, *mu_inc*, *mu_dec*, *mu_max*, *mem_reduc*. Prvních devět parametrů bylo probráno v předchozí kapitole. Pro tuto metodu je příznačné počítání parametru *mu*. Je-li tento parametr roven nule, je algoritmus totožný s Newtonovým algoritmem. Je-li *mu* velmi vysoké, algoritmus bude pracovat jako algoritmus nejstrmějšího sestupu s malým krokem (*gradient descentwithsmall step size*). Parametr *mu* se po každém úspěšném kroku (snížení chybové funkce) vynásobí zadanou hodnotou *mu_dec*. Naopak při neúspěšném kroku je *mu* vynásobeno konstantou *mu_inc*. Při dosažení hranice *mu_max*algoritmus trénování ukončí. [2]

Levenbergova-Marquardtova metoda je vhodná v aplikacích, kde není pro trénovací algoritmus dostatek místa v paměti. Určit omezení na paměť lze v tomto algoritmu pomocí parametru *mem_reduc*. Pokud je *mem_reduc* rovný 1, nedochází k žádné úspoře paměti. Je-li hodnota rovna 2, dojde k rozdělení Jacobiho matice na dvě matice o poloviční počtu řádků.[2]

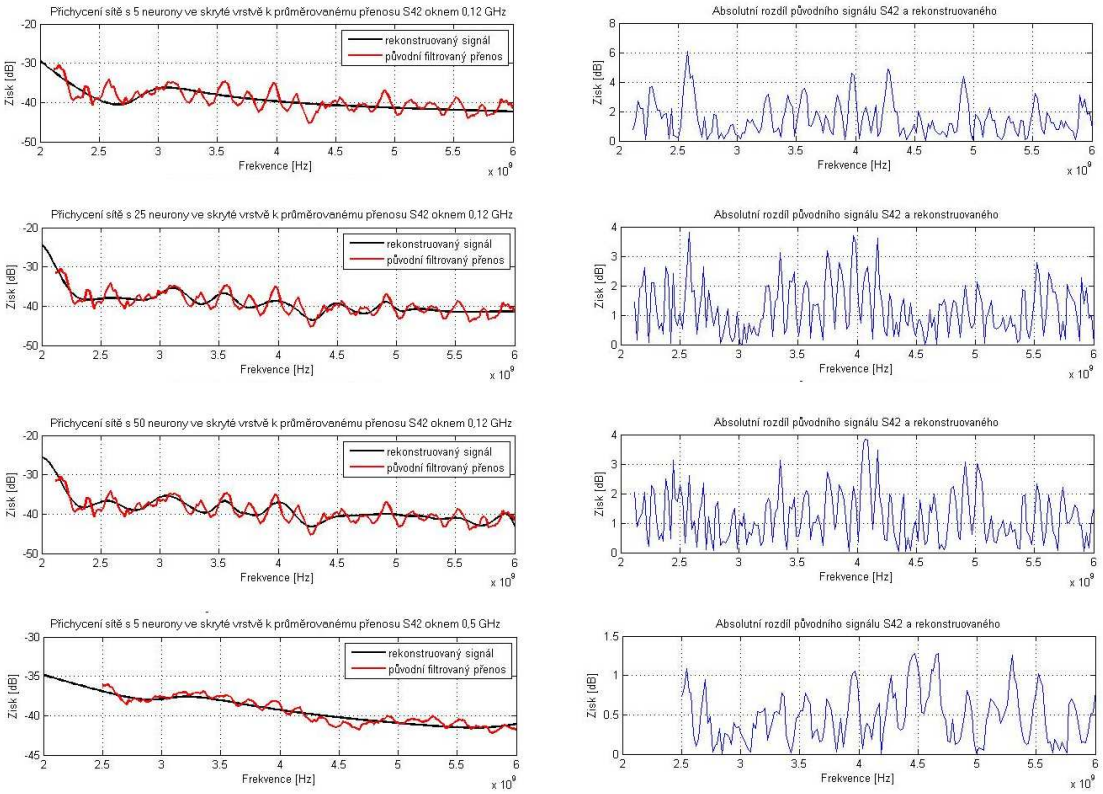
Stejně jako předchozí síť trénovaná algoritmem Bayesovské regularizace tak i síť trénována Levenbergovým-Marquardtovým optimalizačním algoritmem je trénována s 5 až 50 neurony ve skryté vrstvě na trénovací množinu dat průměrovanou okny velikostí 0,05 GHz až 0,50 GHz.



Obr. 15 Graf vývoje střední kvadratické chyby (*mse)* pro sítě trénované Levenbergovým-Marquardtovým algoritmem

Vývoj minimální chyby sítě (viz Obr. 15) vykazuje podobnou tendenci jako síť trénovaná Bayesovskou regularizací, ovšem minimální chyba učení této sítě dosahuje o 2 řády nižší hodnoty. Minimální hodnota chybové funkce mse byla zjištěna pro architekturu 2-35-1 trénovanou signálem průměrovaným oknem velikosti 0,50 GHz.

Chybová funkce sítě v řádu 10^{-3} je již dostatečně nízká. V takovém případě dochází k podstatnému zlepšení přichycení sítě k trénovacím vzorům oproti předchozímu případu (kapitola 3.2) a rekonstruovaný signál je téměř věrnou kopií původního průměrovaného signálu.



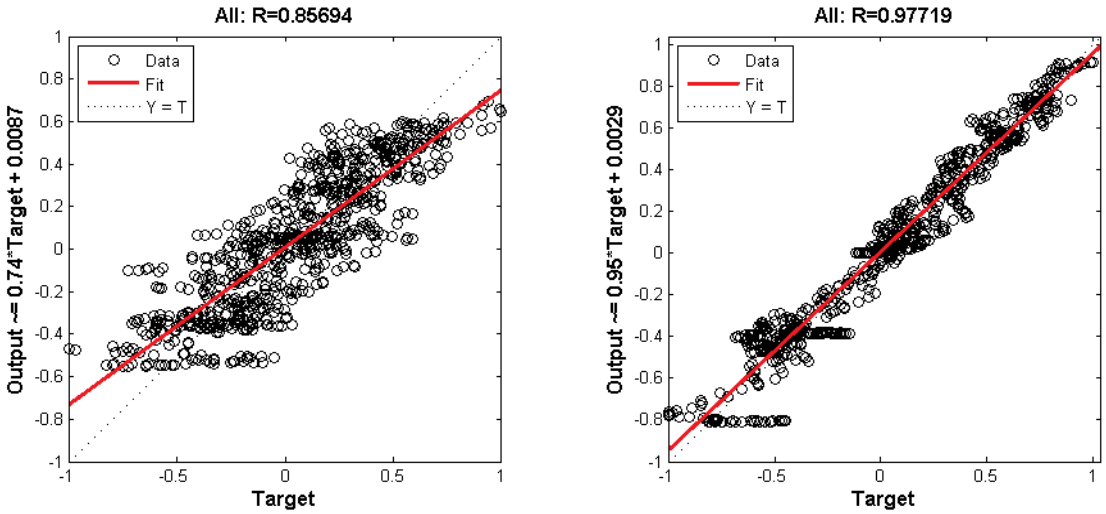
Obr. 16 Průběhy přichycení sítí k průměrovanému přenosu S42 a vzájemný rozdíl pro několik různých architektur sítě a velikostí průměrovacího okna

Opět pro názornost je použit pouze přenos S42. Z grafů přichycení sítí k průměrovaným přenosům (viz Obr.16) je na první pohled patrné lepší přichycení než u sítí trénovaných Bayesovskou regularizací. Obsahuje-li přenos příliš mnoho rychlých změn a málo neuronů ve skryté vrstvě, síť trénovaná Levenbergovým-Marquardtovým optimalizačním algoritmem nemá možnost dokonalejšího přichycení k přenosu. Při zvyšování počtu neuronů ve skryté vrstvě dochází k velmi dobrému přichycení sítě k průměrovanému přenosu a tedy i k malé odchylce rekonstruovaného signálu oproti skutečnému průměrovanému přenosu.

Při použití tohoto trénovacího algoritmu vede zvyšování počtu neuronů ve skryté vrstvě k lepšímu přichycení sítě k trénovacím vzorům v porovnání s algoritmem Bayesovské regularizace.

Průměrujeme-li signál oknem 0,50 GHz a použijeme-li pro trénování síť s architekturou 2-5-1, budou výsledné přichycení sítě k přenosu i odchylka rekonstruovaného signálu od skutečného přenosu vykazovat stejné výsledky jako při použití algoritmu Bayesovské regularizace. V takovém případě se odchylka opět nachází okolo hodnoty 1 dB a menší v celém dostupném frekvenčním rozsahu.

Pro další srovnání výhod a možností Levenbergova-Marquardtova optimalizačního algoritmu je testována síť architektury 2-5-1, která byla trénována na přenos průměrovaný oknem 0,12 GHz a 0,50 GHz.

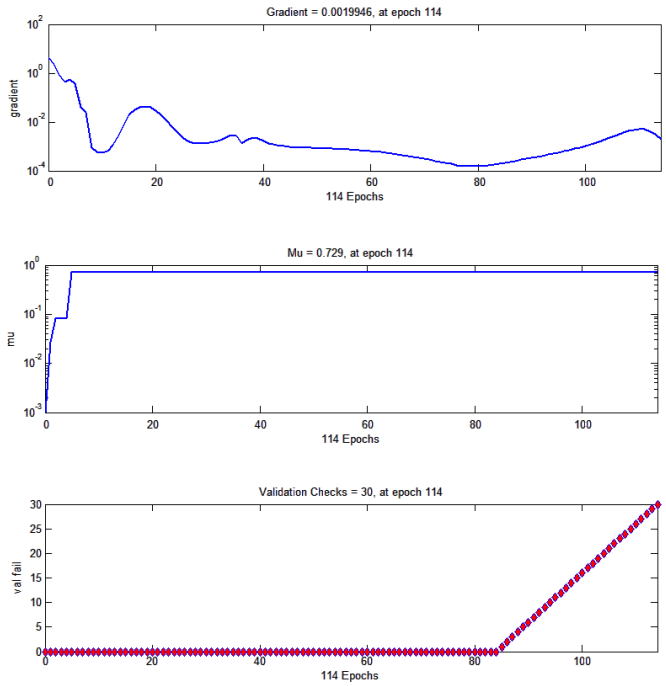


Obr. 17 Regresivní křivky pro přichycení sítě architektury 2-5-1 k přenosu průměrovaného signálu oknem 0,12 GHz (vlevo) a oknem 0,50 GHz (vpravo)

Rovnice regrese pro tyto dva případy (viz Obr. 17) se opět blíží k předchozímu případu (vizObr.12). Při trénovaní přenosem průměrovaným oknem 0,50 GHz je přichycení i výsledná odchylka rekonstruovaného signálu a původního průměrovaného přenosu podstatně lepší než při využití okna 0,12 GHz. V takovém případě signál obsahuje příliš mnoho rychlých změn funkčních hodnot a síť s 5 neurony ve skryté vrstvě opět nemá možnost přichytit se dokonale k tomuto signálu.

Z obrázku 18 je patrný vývoj gradientu a Marquardtova parametru *mu* pro architekturu 2-5-1 a průměrovací okno 0,50 GHz. Gradient prudce klesá a *mu* naopak prudce roste. Oba tyto parametry dosahují svých limitních hodnot již po několikáté epoše trénování což znamená, že síť konverguje rychle. Rychlá konvergence sítě je také jedna z hlavních výhod oproti algoritmu Bayesovské regularizace.

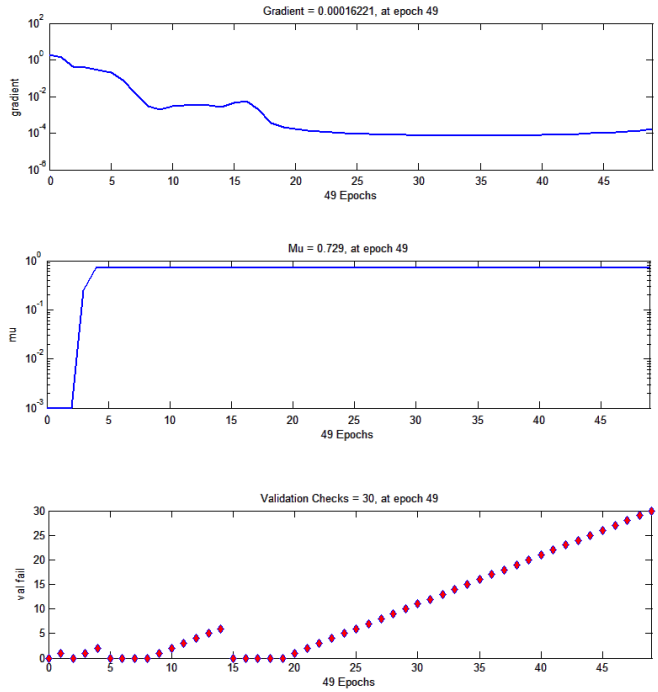
Je zde vidět také vývoj validační kontroly sítě. Při dosažení 30-ti po sobě následujících chybných validačních kontrol dochází k zastavení tréninku. Hodnotu je potřeba experimentálně nastavit v závislosti na konkrétních potřebách sítě (rychlá konvergence, nízká hodnota *mse*, atd.).



Obr. 18 Vývoj gradientu,Marquardtova parametru a validační kontroly pro síť architektury2-5-1 trénovanou přenosy průměrovanými oknem velikosti 0,50 GHz

V případě signálu s více rychlými změnami je i výsledné přichycení sítě a konvergence horší než v předchozím případě (viz Obr. 19).

Ze zjištěných hodnot je patrné, že sítě trénované Levenbergovým-Marquardtovým algoritmem rychleji konvergují a také vykazují menší chybu rekonstruovaného signálu oproti Bayesovské regularizaci. Algoritmus LM je výhodný použít v případě omezené paměti pro ukládání hodnot nastavení sítě. V aplikacích, kde je potřeba velkého počtu neuronů ve skryté vrstvě, bude pravděpodobně výhodné užití tohoto algoritmu.



Obr. 19 Vývoj gradientu, Marquardtova parametru a validační kontroly pro síť architektury 2-5-1 trénovanou přenosy průměrovanými oknem velikosti 0,12 GHz

3.6 Gradientní sestup s adaptivním parametrem učení

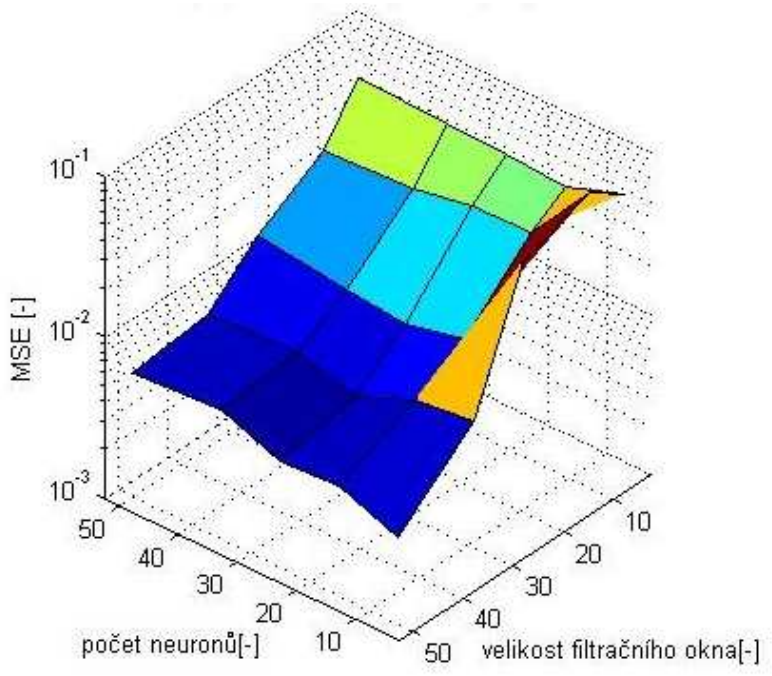
Tento algoritmus je oproti standardní metodě nejstrmějšího sestupu, kde je konstanta učení (*lr, learning rate*) držena na stejné hodnotě v průběhu celého procesu učení, mnohem pružnější. Příliš vysoká hodnota konstanty učení vede k oscilacím a síť je tedy nestabilní. Je-li naopak konstanta *lr* příliš malá, algoritmus konverguje velmi pomalu. Není příliš vhodné konstantu *lr* určovat experimentálně. Vhodným řešením je použití algoritmu s adaptivním procesem učení, který ji mění podle předem vykonaných kroků (*lr_inc, lr*_dec) během učení sítě.

Algoritmus na začátku procesu trénování spočítá výstup sítě a chybu sítě. V každé další epoše jsou nové váhy a prahy počítány s použitím aktuální hodnoty konstanty učení *lr*. Opět dojde ke spočítání výstupů sítě a chyb a rozhodnutí, zdali bude vhodné konstantu učení zvýšit o *lr_inc*, případně ji snížit o *lr_dec*.

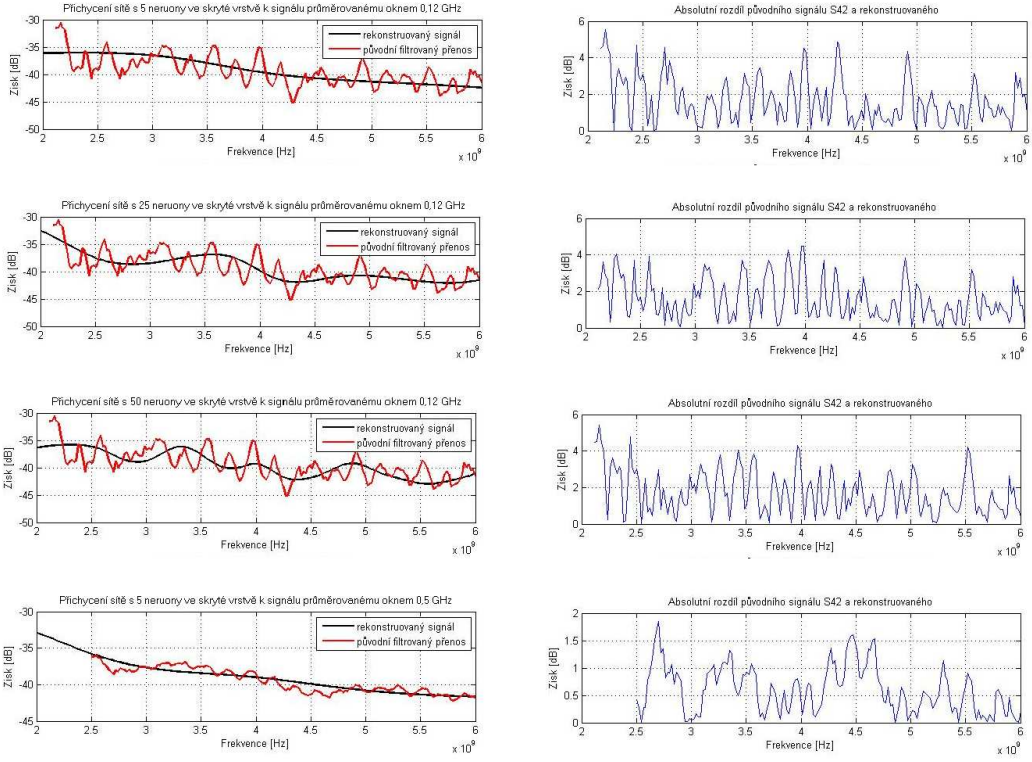
V NN Toolboxu MATLABu je tento algoritmus implementován pomocí funkce *traingda(…)*, který pracuje s parametry *show, epochs,time,lr, lr_inc, lr_dec,goal,min_grad a max_fail (*viz. m-file *gradientni_sestup.m*, složka Gradientní sestup se spádem)

Tento algoritmus byl testován na stejných závislostech jako předchozí dva

algoritmy. Závislost střední kvadratické chyby *mse* na velikosti průměrovacího okna a počtu neuronů ve skryté vrstvě vykazuje podobný charakter jako u sítí trénovaných Levenbergovým-Marquardtovým algoritmem (viz Obr.20).

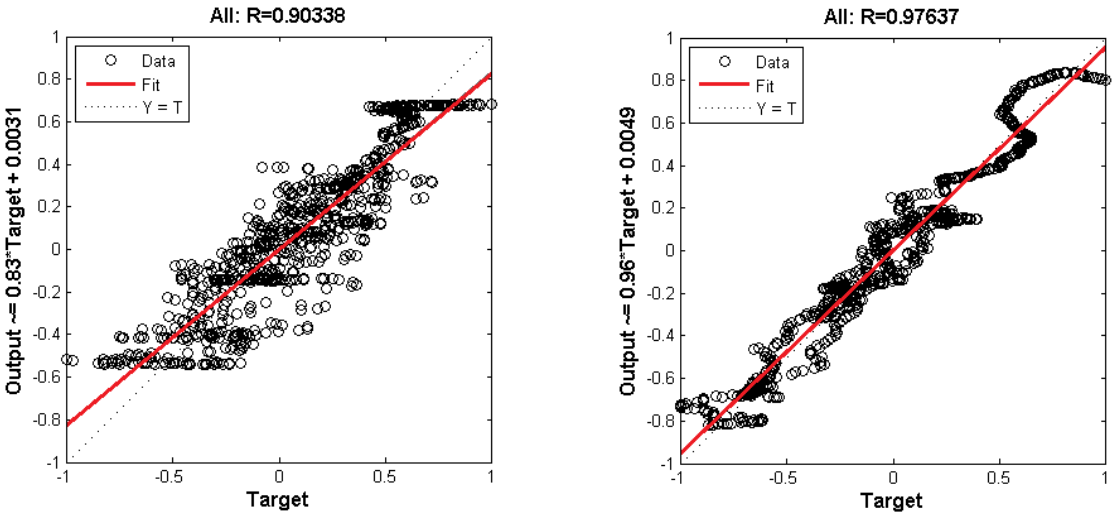


Obr. 20 Graf vývoje střední kvadratické chyby *mse*



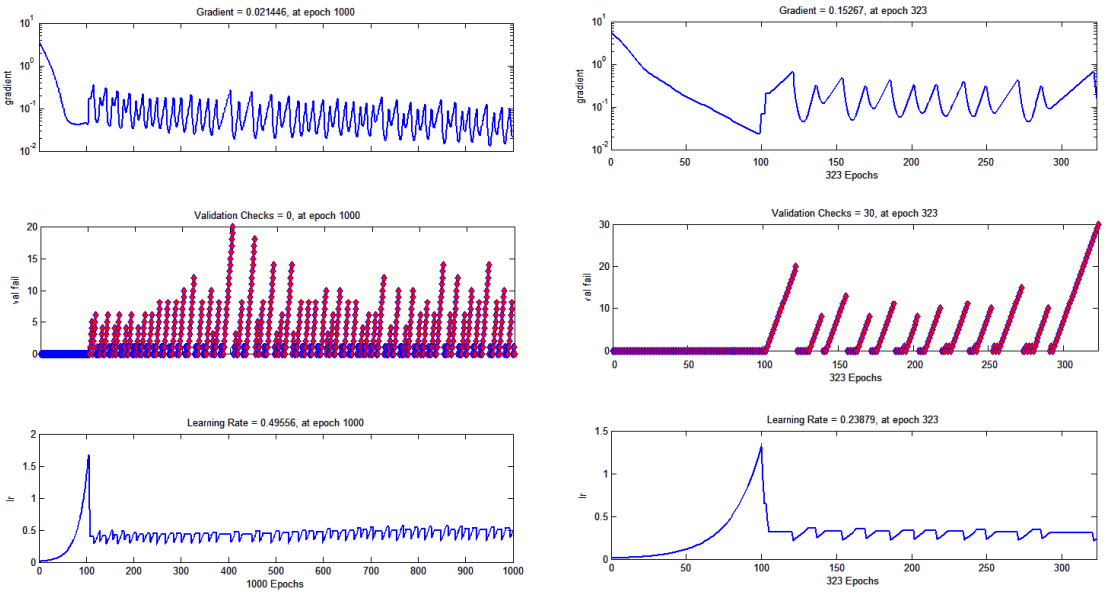
Obr. 21 Průběhy přichycení sítí k průměrovanému přenosu S42 a vzájemný rozdíl pro několik různých architektur sítě a velikostí průměrovacího okna

Přichycení sítí k průměrovaným přenosům je obdobné jako u trénování Levenbergovým-Marquardtovým optimalizačním algoritmem. Metoda gradientního sestupu však vyžaduje mnohem více trénovacích epoch (v řádů stovek až jednotek tisíců epoch), a tedy i mnohem pomaleji konverguje. Rychlost konvergence je závislá na vhodném nastavení parametrů *lr_inc* a *lr_dec*. Neexistuje univerzální stanovisko pro nastavení těchto dvou hodnot, a proto je potřeba hodnoty určit experimentálně. V tomto případě bylo nastaveno *lr_inc*=1,05 a *lr_dec*=0,7. Z důvodu pomalé konvergence sítě je vhodnější použít některý z tzv. rychle konvergujících algoritmů.



Obr. 22 Regresivní křivky pro přichycení sítě architektury 2-5-1 k přenosům průměrovaným oknem velikosti 0,12 GHz (vlevo) a 0,50 GHz (vpravo)

Regresivní křivky odpovídají oběma předešlým případům (viz Obr. 21). Je tedy zřejmě, že pro všechny testované vícevrstvé perceptronové sítě je hlavní problém s přichycením k signálům s rychlými změnami funkčních hodnot.



Obr. 23 Vývoj gradientu, validační kontroly a parametru *lr* pro architektury 2-5-1 průměrované oknem 0,50 GHz (levý sloupec) a oknem 0,12 GHz (pravý sloupec)

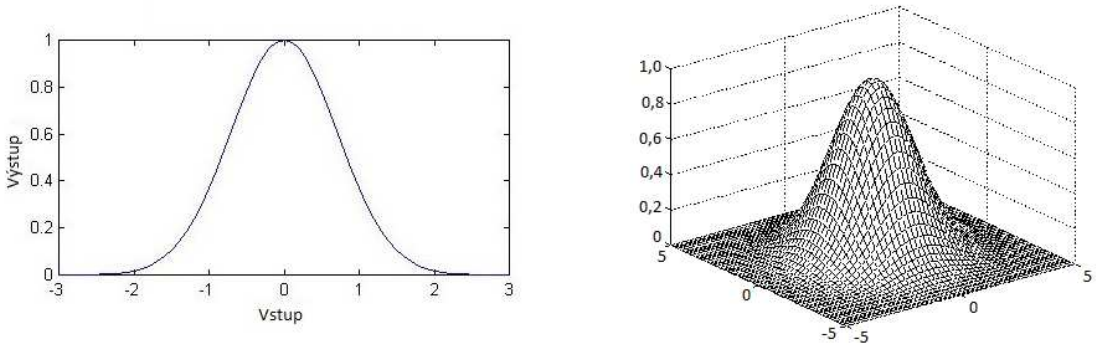
Z obou dvou vývojů znázorněných na obrázku 23 jsou patrné oscilace gradientu i parametru *lr*. I přes oscilace gradientu je jeho tendence stále klesající. S těmito oscilacemi souvisí validační chyba, která po dosažení předvoleného počtu za sebou následujících validačních chyb ukončí trénovací proces. V našem případě dochází k přerušení trénování po 30-ti za sebou neúspěšných validacích (viz Obr. 23). Pokud by byl zvolen větší počet neúspěšných validačních kontrol, došlo by i u vývoje gradientu k větším oscilacím. Bude-li naopak počet neúspěšných validačních kontrol nízký, síť nestihne dostatečně dobře konvergovat z důvodu brzkého zastavení trénování touto validační kontrolou.

Použití toho algoritmu není příliš časté, a to hlavně z důvodu dlouhého trénovacího času. Metody tohoto typu jsou v dnešní době vytlačeny rychle konvergujícími algoritmy.

4 RADIÁLNÍ BÁZOVÉ SÍTĚ

4.1 Uspořádání radiální bázové sítě

Radiální bázová síť je dopředná síť, která obsahuje radiální bázové funkce (RBF). Hodnota radiální bázové funkce monotónně klesá od maxima v počátku souřadné soustavy (viz Obr. 24). Radiální bázová funkce má většinou tvar Gaussovy křivky.[3].



Obr. 24 Přenosová funkce radiální báze pro 1D vstupní prostor se středem v bodě [0] (vlevo) a pro 2D vstupní prostor se středem v bodě [0,0] (vpravo)

Matematicky Gaussovu křivku v jednorozměrném případě popisujeme vztahem

$$h(x) = \exp(-[(x-c)]^2/r^2), \quad (1)$$

kde *c* je souřadnice středu, *x* je nezávislá proměnná a *r* je poloměr (rádius) křivky.

Skládáním Gaussovských RBF je možno vytvořit libovolnou spojitou funkci definovanou nad vstupním prostorem. Počet neuronů a jejich parametry závisejí na konkrétní aplikaci. Je vhodné znát průběh signálu před samotným trénováním sítě. Hustější rozmístění středů neuronů nad místy s rychlými změnami funkčních hodnot vede k lepšímu přichycení sítě k tomuto signál.

Pro tvorbu RBF sítí nebyl využit NN Toolbox. Sítě byly převzaty v textovým kódu a dále modifikovány v programu MATLAB.

Architektura RBF sítě odpovídá architektuře diskutované na začátku tohoto projektu (viz obr.1). Ve skryté vrstvě jsou použity neurony s radiální bázovou funkcí. V našem případě nás zajímá šířka, pozice a počet radiálních bázových funkcí, které jsou nezbytné k vytvoření funkce, jež reprezentuje závislost výstupních hodnot sítě na hodnotách vstupních.

4.2 Levenbergův-Marquardtův algoritmus

Levenbergův-Marquardtův algoritmus (LM) patří mezi kvazi-Newtonovy optimalizační metody. Cílem LM optimalizace je minimalizovat během procesu učení odchylku mezi trénovacími výstupními vzory a skutečnými odezvami sítě.

Vstupními parametry trénovací funkce (viz. m-file *RBF_LM.m*, složka RBF siť) jsou vektor (matice) pozic středů radiálních bázových funkcí a typ použitých jader. Pro algoritmus je důležitá znalost Jacobiho matice, která obsahuje první derivace kriteriální funkce podle stavových proměnných. V průběhu trénování dochází ke zpřesňování odhadu hodnot Jacobiho matice. Algoritmus LM je schopen na základě znalosti této matice v průběhu trénování nalézt správné velikosti prahů a vah výstupní vrstvy tak, aby přichycení k trénovacím vzorům bylo co nejlepší.

V našem případě algoritmus vrací vektor **W**, jehož hodnoty reprezentují vhodné nastavení vah výstupní vrstvy sítě, vektor **b**, jenž je roven nastavení prahových hodnot radiálních bázových funkcí a hodnotu *SF* (*spread factor*), která určuje šířku radiálních bázových funkcí [1].

Odpovídá-li model sítě funkci

$$\mathbf{f(x)} = \sum_{j=1}^{m} \mathbf{w_j \cdot h_j(x)} \quad , \tag{2}$$

a trénovací množina je $\{(x_i, \hat{y}_i)\}$, $i = 1,\ldots,p$, potom se algoritmus snaží nastavením SF, prahů a vah snížit střední kvadratickou chybu (*MSE*, *mean square error*) (viz obr.16) [3]. Střední kvadratická chyba se počítá jako

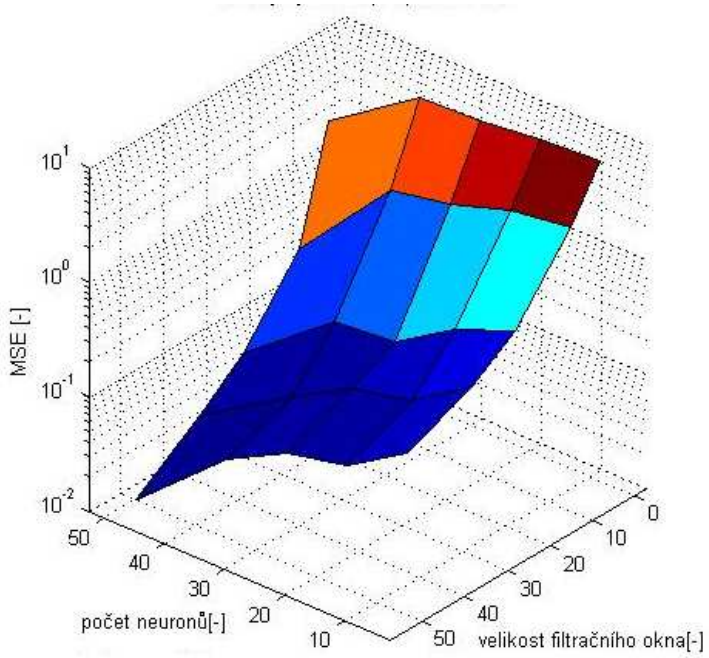
$$\mathbf{mse} = \sum_{i=1}^{p} (\hat{y}_i - \mathbf{f(x_i)})^2 \tag{3}$$

,kde w_j – váha j-tého neuronu

h_j – hodnota prahu j-tého neuronu

$f(x_i)$ – odezva sítě na i-tý vzor

$\hat{y}_i$ - požadovaná odezva na i-tý vzor.



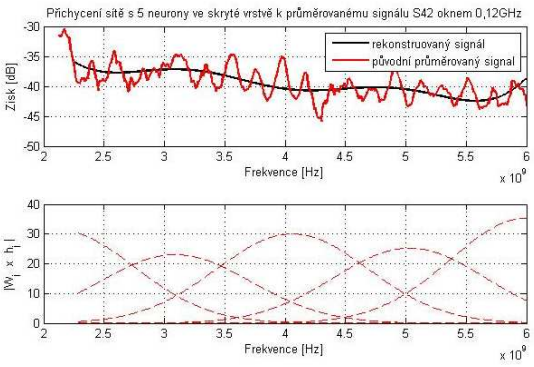
Obr. 25Graf vývoje střední kvadratické chyby (*mse*)

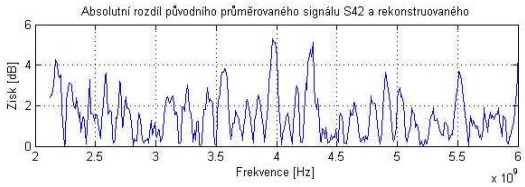
Při trénování RBF sítí, oproti vícevrstvým perceptronovým sítím, byla pro trénování užita pouze množina dat představující přenos S42. Jedná se tedy o přenos signálu z vysílací antény (na přístrojové desce) do přijímací antény 4 (prostor řidiče).

RBF sítě trénované Levenbergovým-Marquardtovým optimalizačním algoritmem byly testovány ve stejném smyslu jako předchozí vícevrstvé percetptronové sítě.

Z vývoje *mse* (viz Obr. 25) je patrné, že použitelnost RBF sítí má také limity. Přichycení sítí k signálu, který obsahuje mnoho rychlých změn funkčních hodnot, je realizovatelné pouze s velkým počtem neuronů ve skryté vrstvě. Toto ovšem není žádoucí, neboť s počtem neuronů a složitosti trénovací množiny roste také potřebná výpočetní náročnost a doba trénování se značně prodlužuje.

Pro bližší testování, stejně jako u vícevrstvých perceptronových sítí, byly použity různé architektury a signál průměrovaný oknem 0,12 GHz a 0,50 GHz (viz. Obr. 26).



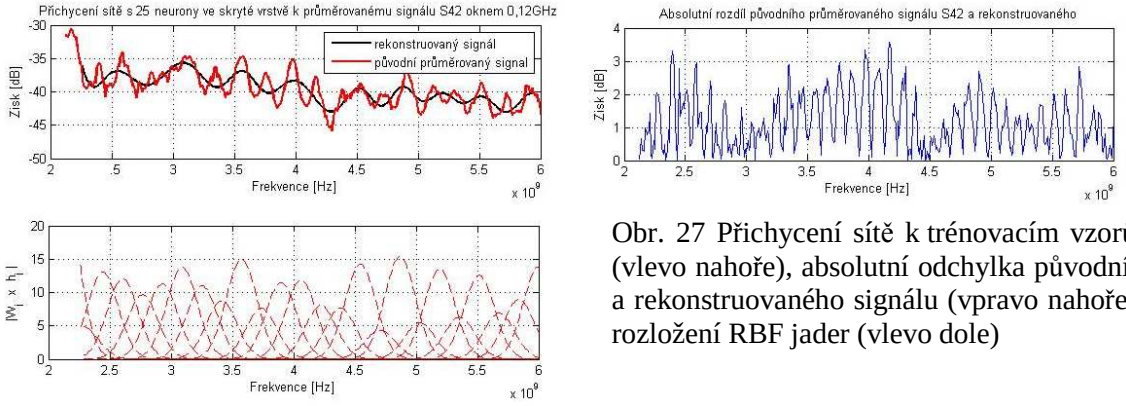


Obr. 26 Přichycení sítě k trénovacím vzorům (vlevo nahoře), absolutní odchylka původního a rekonstruovaného signálu (vpravo nahoře) a rozložení RBF jader (vlevo dole)

Stejně jako tomu bylo u perceptronových sítí tak i RBF sítě potřebují dostatečný počet neuronů ve skryté vrstvě, aby se dokázaly dostatečně přichytit k trénovacím vzorům. Průběh přichycení sítě k průměrovanému signálu na obrázku 26 je zobrazuje přichycení RBF sítě s 5 neurony ve skryté vrstvě trénované na signál průměrovaný oknem 0,12 GHz. Přichycení je nedostačující a tedy i výsledný rozdíl těchto dvou signálů je značný.

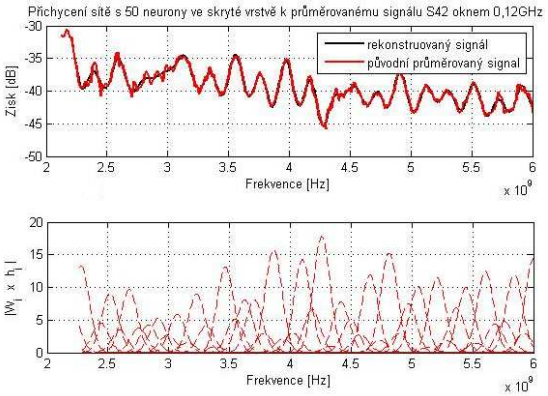
Přichycení je pro představivost doplněno o rozmístění RBF jader přes dostupný frekvenční interval. V tomto případě vidíme 5 jader rozmístěných lineárně přes dostupný rozsah. Levenbergův-Marquardtův optimalizační algoritmus počítá šířku (*SF,spread factor)*a výšku (*bias)* každého neuronu tak, aby bylo přichycení co nejdokonalejší a tedy i chybová funkce *mse* co nejnižší (viz Obr. 26 vlevo dole).

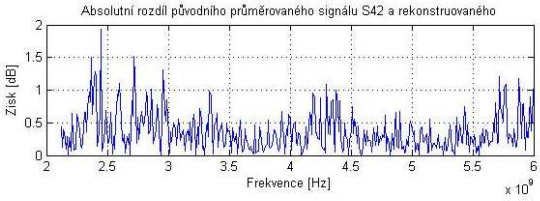
Na obrázku 27 je patrná schopnost RBF sítí přichytit se k trénovacím vzorům. Při použití architektury 2-25-1 a trénovacího signálu průměrovaného oknem 0,12 GHz je síť schopna se velmi dobře k této množině přichytit a vykazovat tedy menší chybu. Síť s těmito parametry dosahovala poměrně dost chyb srovnatelných s chybami vícevrstvých perceptronových sítí trénovaných stejným Levenbergovým-Marquardtovým optimalizačním algoritmem.



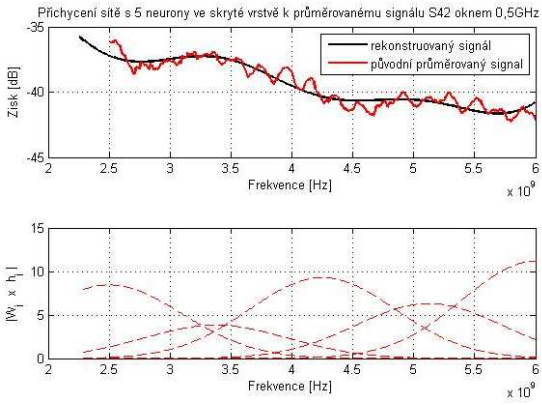
Obr. 27 Přichycení sítě k trénovacím vzorům (vlevo nahoře), absolutní odchylka původního a rekonstruovaného signálu (vpravo nahoře) a rozložení RBF jader (vlevo dole)

Při použití architektury 2-50-1 a signálu průměrovaného oknem 0,12 GHz (viz.Obr.28) je přichycení sítě k průměrovanému signálu téměř dokonalé. Síť má dostatek neuronů, a v relativně krátkém čase je schopna najít ideální nastavení, aby minimalizovala odchylku původního průměrovaného přenosu a rekonstruovaného signálu.





Obr. 28 Přichycení sítě k trénovacím vzorům (vlevo nahoře), absolutní odchylka původního a rekonstruovaného signálu (vpravo nahoře) a rozložení RBF jader (vlevo dole)



Obr. 29 Přichycení sítě k trénovacím vzorům (vlevo nahoře), absolutní odchylka původního a rekonstruovaného signálu (vpravo nahoře) a rozložení RBF jader (vlevo dole)

Poslední testovaná konfigurace byla architektury 2-5-1 ovšem trénována signálem průměrovaným oknem 0,50 GHz. Signál je podstatně vyhlazenější než předchozí tříprůměrované signály. I síť s 5 neurony ve skryté vrstvě dokáže dostatečně přesně kopírovat tendenci tohoto signálu. Rozdíl původního průměrovaného přenosu a rekonstruovaného signálu je podstatně menší oproti použití průměrovacích oken menší velikosti. Menší rozdíl je způsoben především malými změnami funkčních hodnot v celém frekvenčním rozsahu.

4.3 Křížová kontrola

Pokud máme k dispozici dostatek dat, je možné vykonat křížovou kontrolu (*cross-validation*). Křížová kontrola spočívá v natrénování sítě pomocí trénovací množiny a v následném testování sítě disjunktní testovací množinou.

Ve vylepšené variantě křížové kontroly (*leave-one-out*) je trénovací množina velikosti p–1, kde p je celkový počet trénovacích vzorů. Testovací množina má jednotkovou velikost (pouze jeden vzor ze všech je použit k testování). Tímto způsobem algoritmus projde celou množinu dat a zprůměruje kvadratickou chybu

$$\sigma^2_{LOO} = \frac{\hat{\mathbf{y}}^{\mathbf{T}} \cdot \mathbf{P} \cdot \mathbf{diag}(\mathbf{P})^{-2} \cdot \mathbf{P} \cdot \hat{\mathbf{y}}}{\mathbf{p}}, \tag{4}$$

kde projekční matice Pje

$$\mathbf{P} = \mathbf{I_p} - \mathbf{H} \cdot \mathbf{A}^{-1} \cdot \mathbf{H}^{\mathbf{T}}, \tag{5}$$

A je kovarianční matice a I_p čtvercová diagonální jednotková matice velikosti p[3].

5 ZÁVĚR

Prvním krokem řešení problematiky aproximace funkce pomocí umělých neuronových sítí byla příprava vzorové trénovací a testovací sady dat. Pomocí těchto vzorových sad byla testována většina dostupných trénovacích algoritmů pro vícevrstvé perceptronové sítě. V NNT MATLABu byly sítě vytvořeny a vhodnými trénovacími algoritmy trénovány pomocí trénovací sady vzorů. U natrénovaných sítí byla testována jejich odezva na hodnoty mimo trénovací data pomocí vytvořené vzorové testovací sady. Nejlepších výsledků dosahovaly sítě trénované Levenbergovým-Marquardtovým optimalizačním algoritmem, Bayesovskou regularizací a gradientním sestupem s adaptivním procesem učení. Tyto typy sítí byly dále užity pro modelování polí uvnitř automobilu.

Další část této práce je zaměřena na data potřebná pro modelování polí uvnitř automobilu. Byly uváženy a vybrány vhodné pozice pro zkoumání přenosů uvnitř vozu. V programu CST Microwave Studio 2013 byla provedena simulace přenosů mezi 12-ti anténami rozmístěnými uvnitř matematického modelu vozu Octavia II. Pro konkrétní aplikaci bylo vhodné užití radiofrekvenčních spojů na vysokých frekvencích 2 až 6 GHz.

Ze všech pozic antén byly vybrány pouze 4 přenosy do prostorů cestujících a tyto signály byly dále průměrovány okny různé velikosti. Na těchto průměrovaných signálech bylo provedeno následující testování NN sítí.

Nejlepších výsledků dosahovala síť trénovaná pomocí Levenbergova-Marquardtova optimalizačního algoritmu. Tento algoritmus vedl k nejlepšímu přichycení trénovací sady vzorů k síti a také čas potřebný pro konvergenci byl ve většině případu nejkratší. Další výhoda je možnost úspory paměťového prostoru během výpočtů, které provádí samotný algoritmus.

Výhoda Bayesovské regularizace spočívá v počítání počtu efektivně využitých parametrů sítě (prahů a vah) potřebných pro určení vhodné architektury sítě. Tento algoritmu je použitelný pouze pro aproximaci jednoduchých signálu.

Poslední testovaný algoritmus gradientního sestupu se skluzem prokázal při použití většího průměrovacího okna výsledky srovnatelné s ostatními algoritmy, ovšem čas potřebný pro konvergenci sítě byl podstatně delší než u předešlých algoritmů, které patří do skupiny tzv. rychle konvergujících.

Při tvorbě RBF sítě nebylo NNT užito, tato síť byla vytvořena kódem v programu MATLAB. Vytvořené sítě různých architektur byly trénovány pomocí Levenbergova-Marquardtova optimalizačního algoritmu. Tyto sítě vykazovaly obdobné kvality jako vícevrstvé perceptronové sítě. Výhodou RBF sítě je možnost vlastní volby středů RBF neuronů podle konkrétní potřeby. Tímto je možné docílit hustějšího rozložení RBF neuronů v místech s velkou změnou aproximované funkce. Naopak v místech, kde se aproximovaná funkce příliš nemění, a proto není potřeba příliš velkého počtu RBF neuronů.

Všechny testované algoritmy mají dostačující předpoklady pro modelování polí uvnitř automobilu. Použití konkrétního typu algoritmu vyplívá z požadavků pro danou aplikaci. V tomto případě by pro modelování polí uvnitř automobilu postačovalo průměrovat přenosy oknem 0,50 GHz. V takovém případě je tedy vhodné zvolit jednoduchou architekturu sítě 2-5-1 a výběrem vhodného algoritmu zajistit rychlou a správnou konvergenci sítě.

LITERATURA

[1] HAIKIN, S. NeuralNetworks: A ComprehensiveFoundation, 2/E. UpperSaddle River: PrenticeHall, 1999. 842 s.

[2] DEMUTH, H. BEALE, M. Neural Network Toolbox User´sGuide, 4. version. TheMathWorks, 2004. 842 s.

[3] ORR, M. J. L.Introduction to RadialBasisFunctionNetworks. 2. vydání. Edinburgh: University of Edinburgh, 1996. 67 s.

[4] HAKL, F. HOLEŇA, M. Úvod do teorie neuronových sítí. Praha: Editační středisko ČVUT Praha, 1997. 210 s.

[5] RAIDA,Z. Modeling EM structures in theNeural Network Toolbox of MATLAB. IEEE Antennas& Propagation Magazine, 2003, vol. 44, no. 6,p. 46-47.

[6] ČERNOHORSKÝ, D., RAIDA, Z., ŠKVOR, Z., NOVÁČEK Z. Analýza a optimalizace mikrovlnných struktur. Brno: VUTIUM Publishing, 1999.

[7] SCHACK, M., JEMAI, J., PIESIEWICZ, R., GEISE, R., SCHMIDT, I., KÜRNER, T. UWB Measurements and analysisofan in-car UWB channel. Braunschweig: Technische Universität Braunschweig. 2008, 459-463 s.

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

A	Kovariantní matice (variance matrix)
c	Střed Gausovy křivky
e	Vektor chyb sítě
g	Gradient
$h(x)$	Odezva na signál xv časové oblasti
H	Hessova matice
I_p	Čtvercová diagonální jednotková matice
J	Jacobiho matice
P	Projekční matice
r	Poloměr Gausovy křivky
w_j	j-tý váhový spoj
x_i	Vstupní signál
y_i	Výstupní signál
$\hat{y}_i$	Odhad výstupního signálu

LOO*(Leave-one-out)*	Křížová kontrola
RBF*(Radialbasisfunction)*	Radiální bázova funkce
NN*(Neural Network)*	Neuronová síť
LM*(Levemberg-Marqurdt)*	Levenbergův-Marquardtův optimalizační algoritmus
SF*(Spreadfactor)*	Konstanta rozprostření RBF
mse*(Mean square error)*	Střední kvadratická chyba
sse*(Sum of square error)*	Suma kvadratické chyby

SEZNAM PŘÍLOH

A PROGRAMY

root\Matlab\Trenovaci_mnozina - Obsahuje program pro přípravu a ladění trénovací a testovací množiny dat

Skript *prenosy.m* připravuje množinu vstupních a výstupních dat pro trénování sítě. Skript také realizuje průměrování extrahovaných přenosů a přípravu testovacích dat.

root\Matlab_NN_Toolbox\Levenberg-Marquardt - Obsahuje programy pro realizaci a trénování vícevrstvých perceptronových sítí

Skript *Levemberg_Marquardt.m* realizuje tvorbu vícevrstvé perceptronové sítě trénované Levenbergovým-Marquardtovým optimalizačním algoritmem. V tomto skryptu dochází k počáteční inicializaci, trénování a testování sítě.

Skript *Chyby.m* realizuje zpracování a grafickou vizualizaci výsledků testování sítě.

root\Matlab_NN_Toolbox\Bayesian_regularization - Obsahuje programy pro realizaci a trénování vícevrstvých perceptronových sítí

Skript *Bayesian_regularization.m* realizuje tvorbu vícevrstvé perceptronové sítě trénované algoritmem Bayesovské regularizace. V tomto skryptu dochází k počáteční inicializaci, trénování a testování sítě.

Skript *Chyby.m* realizuje zpracování a grafickou vizualizaci výsledků testování sítě.

root\Matlab_NN_Toolbox\Gradient_descent_adapt_learning- Obsahuje programy pro realizaci a trénování vícevrstvých perceptronových sítí

Skript *Gradient_descent_adapt_learning.m* realizuje tvorbu vícevrstvé perceptronové sítě trénované algoritmem Gradientního sestupu s procesem adaptivního učení. V tomto skryptu dochází k počáteční inicializaci, trénování a testování sítě.

Skript *Chyby.m* realizuje zpracování a grafickou vizualizaci výsledků testování sítě.

root\Matlab\RBF_site - Obsahuje programy, které vytváří, trénují, zobrazují a testují RBF sítě

Skript *RBF.m* realizuje načtení dat a počáteční nastavení parametrů sítě. Tento skrypt volá skritp *RBF_train.m* pomocí něhož je síť trénovaná a pomocí skriptu *RBF_sim.m* je síť simulována. Grafické zobrazení probíhá ve skriptu *vizualize.m*.

Skript *Chyby.m* realizuje zpracování a grafickou vizualizaci výsledků testování sítě.

B TABULKY

Tabulka 1 Závislost střední kvadratické chyby *mse* [-] na počtu neuronů ve skryté vrstvě a velikosti průměrovacího okna sítí trénovaných Bayesovskou regularizací

Počet neuronů [-] / Velikost okna [GHz]	5	15	25	35	50
0,05	48,3	43	41,9	37,7	37,7
0,12	29,7	21,1	18,5	16,5	10,4
0,25	15,5	5,1	3,5	2,5	1,9
0,35	11,8	3,9	3,1	2,7	2,5
0,5	6,7	1,5	1,4	1,1	0,8

Tabulka 2 Závislost střední kvadratické chyby *mse* [-] na počtu neuronů ve skryté vrstvě a velikosti průměrovacího okna sítí trénovaných Levenbergovým-Marquardtovým optimalizačním algoritmem.

Počet neuronů [-] / Velikost okna [GHz]	5	15	25	35	50
0,05	8,7E-02	5,4E-02	6,5E-02	6,9E-02	8,0E-02
0,12	4,5E-02	3,8E-02	2,8E-02	2,0E-02	2,0E-02
0,25	3,0E-02	1,3E-02	7,2E-03	5,1E-03	4,3E-03
0,35	1,7E-02	7,5E-03	5,1E-03	4,6E-03	5,1E-03
0,5	9,2E-03	3,8E-03	2,5E-03	1,9E-03	2,7E-03

Tabulka 3 Závislost střední kvadratické chyby *mse* [-] na počtu neuronů ve skryté vrstvě a velikosti průměrovacího okna sítí trénovaných Levenbergovým-Marquardtovým optimalizačním algoritmem.

Počet neuronů [-] / Velikost okna [GHz]	5	15	25	35	50
0,05	5,6E-02	4,2E-02	4,4E-02	4,6E-02	4,8E-02
0,12	8,1E-02	3,0E-02	3,0E-02	2,5E-02	2,4E-02
0,25	5,6E-02	1,2E-02	1,0E-02	1,1E-02	1,3E-02
0,35	9,0E-03	8,1E-03	5,8E-03	7,2E-03	6,7E-03
0,5	3,5E-03	4,8E-03	4,6E-03	6,4E-03	5,9E-03

Tabulka 4 Závislost střední kvadratické chyby *mse* [-] na počtu neuronů ve skryté vrstvě a velikosti průměrovacího okna RBF sítí trénovaných Levenbergovým-Marquardtovým optimalizačním algoritmem.

Počet neuronů [-] / Velikost okna [GHz]	5	15	25	35	50
0,05	9,4E+00	8,8E+00	7,5E+00	7,1E+00	2,1E+00
0,12	3,5E+00	3,0E+00	2,0E+00	1,6E+00	2,2E-01
0,25	8,6E-01	5,3E-01	2,4E-01	2,2E-01	5,3E-02
0,35	4,6E-01	2,1E-01	1,6E-01	7,8E-02	2,6E-02
0,5	2,6E-01	1,2E-01	9,1E-02	4,9E-02	1,0E-02