

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE  
FAKULTA STROJNÍ  
ÚSTAV TECHNIKY PROSTŘEDÍ

---

**STUDIE VYUŽITÍ NEURONOVÝCH SÍTÍ PRO PREDIKCI  
SPOTŘEBY KOMPLEXU BUDOV**

DIPLOMOVÁ PRÁCE

# České vysoké učení technické v Praze

## Fakulta strojní

Ústav techniky prostředí

Akademický rok: 2013/2014

## ZADÁNÍ DIPLOMOVÉ PRÁCE

- Jméno studenta:** Bc. Jakub ČEPELA
- Studijní program:** Inteligentní budovy
- Obor:** bez oboru
- Název česky:** Studie využití neuronových sítí pro predikci spotřeby komplexu budov
- Název anglicky:** Study of Neural Network Use for Energy Consumption Prediction of a Building

### Z á s a d y p r o v y p r a c o v á n í :

Nastudujte metody predikce spotřeby energie (PSE) definovaného objektu, tj. komplexu budov, s využitím neuronových sítí (NS) a navrhnete vhodný způsob PSE pro daný objekt.

- Proveďte rešerši publikovaných výsledků využití NS a používaných algoritmů jejich trénování, zejména z hlediska:
  - možností i omezení jednotlivých přístupů pro praktické využití, a
  - v rešerši se stručně zaměřte i na související metody předzpracování naměřených dat jako je filtrování šumu a nekorelovaných děje, odstranění chybných dat, apod.
- V případě potřeby, navrhnete použití vhodné metody úpravy dat (filtrace, extrakce PCA, ...) pro daný objekt.
- Naprogramujte lineární prediktivní model, polynomiální neuronové modely HONU (QNU a ev. CNU), a neuronovou síť typu MLP a porovnejte jejich možnosti výsledky pro PSE na daném objektu. (Podle okolností navrhnete jinou neuronovou architekturu pro PSE.).
- Na jednoduchém simulačním modelu a na teoretických datech ověřte funkčnost vašich programů a testovaných metod. Metody a neuronové sítě natrénujte a pak otestujte na dalších (jiných než trénovacích) datech.
- Proveďte experimentální analýzy na reálných datech daného objektu a vyhodnoťte a porovnejte přesnost prediktivních metod vhodnými kritérii (MSE, MAE, koeficient determinace,...).
- Diskuze a závěr.

**Rozsah grafických prací:** obrázky a tabulky max. 50 %

**Rozsah průvodní zprávy:** min. 50 stran práce a přílohy včetně kódů

**Seznam odborné literatury:**

- [1] Vladimír Malý : *Metoda PCA a vícerozměrová analýza falešných sousedů pro posouzení neurčitosti redukovaného stavového vektoru provozních dat energetického zařízení*, bakalářská práce, ČVUT v Praze , Fakulta strojní, Ústav přístrojové a řídicí techniky, 2010
- [2] Matouš Sláma: *Využití neuronových sítí při rekonstrukci termogramů spalovací komory práškových kotlů vysokých výkonů*, bakalářská práce, ČVUT v Praze , Fakulta strojní, Ústav přístrojové a řídicí techniky, 2011
- [3] Zhuldyz Assylova: *Comparison of Neural Network Models for Approximation of Pneumatic Muscle Actuator*, bachelor's thesis, CTU in Prague, FME, Department of Instrumentation and Control Engineering, Division of Automatic Control and Engineering Informatics, defended in 2013.
- [4] Ainur Duisenbayeva: *Data Pre-Processing for Adaptive Modelling of Heating System*, bachelor's thesis, CTU in Prague, FME, Department of Instrumentation and Control Engineering, Division of Automatic Control and Engineering Informatics, defended in 2013.

**Vedoucí diplomové práce:**

**Doc. Ing. Ivo Bukovský, Ph.D.**

**Konzultant:**

**Ing. Jan Široký**

**Datum zadání diplomové práce:**

**11.4. 2014**

**Termín odevzdání diplomové práce:**

**20.6. 2014**

Neodevzdá-li student bakalářskou nebo diplomovou práci v určeném termínu a tuto skutečnost předem písemně zdůvodnil a omluva byla děkanem uznána, stanoví děkan studentovi náhradní termín odevzdání bakalářské nebo diplomové práce. Pokud se však student řádně neomluvil nebo omluva nebyla děkanem uznána, může si student zapsat bakalářskou nebo diplomovou práci podruhé.


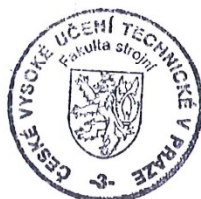
*Diplomant bere na vědomí, že je povinen vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.*

Zadání diplomové práce převzal dne: **23.4.2014**

.....  
student



Prof. Ing. Jiří Bašta, Ph.D.  
Vedoucí ústavu



Prof. Ing. Michael Valášek, DrSc.  
Děkan fakulty

V Praze dne 11.4. 2014

**Souhrn:** Cílem a výstupem této diplomové práce je návrh a naprogramování modelu spotřeby energie komplexu budov s využitím neuronových struktur. Predikce spotřeby energie dané oblasti je především důležitá pro zjištění možných úspor na vytápění a ohřev teplé vody. Základním požadavkem je schopnost předpovědět spotřebu na 24 hodin dopředu a případně vypočítat hodinovou spotřebu energie kterýkoliv den v roce na základě typického průběhu teplot. Návrh tohoto modelu se v první řadě skládá z nastudování již publikovaných výsledků využití neuronových sítí pro predikci spotřeby energie. Dalším krokem je příprava dostupných dat z uvažované kotelny, kde je nutné provést základní filtraci a určení vhodných vstupních vektorů, například pomocí metody hlavních komponent. Takto připravená data již slouží jako vstupy a výstupy pro neuronové modely. Práce se zabývá několika různými druhy neuronových modelů, které jsou vzájemně porovnány za pomoci hodnotících kritérií. Je vybrána nejvhodnější prediktivní metoda pro daný komplex budov při stávajících podmínkách.

**Klíčová slova:** neuronové sítě, predikce spotřeby energie, učící algoritmy, polynomiální neuronové modely, metoda hlavních komponent

**Summary:** Aim of this thesis is to design and program the model of energy consumption of a building complex using neural structures. Prediction of energy consumption of the given area can be used to find out possible savings on buildings heating and hot water heating. Main requirement is to be able to predict energy consumption for next 24 hours or eventually calculate hourly energy consumption for any day in year, given characteristic temperatures. Design of this model, in the first place, consists of studying published results of use of the neural networks for energy consumption prediction. Next step is preparation of available data from the boiler room, basic filtration is needed and also preparation of input vectors, for example using the principal component analysis. Prepared data are used as inputs and outputs of neural models. This thesis addresses several different neural models which are compared using evaluation criteria. At the end the most appropriate predictive method is selected, for the given building complex and conditions.

**Keywords:** neural networks, prediction of energy consumption, learning algorithms, higher order neural unit, principal component analysis



## **Prohlášení**

Prohlašuji, že jsem diplomovou práci s názvem: „Studie využití neuronových sítí pro predikci spotřeby komplexu budov“ vypracoval samostatně pod vedením Doc. Ing. Iva Bukovského, Ph.D., s použitím literatury, uvedené na konci mé diplomové práce v seznamu použité literatury.

V Praze .....

Jakub Čepela .....

## **Poděkování**

Především děkuji vedoucímu této diplomové práce Doc. Ing. Ivu Bukovskému, Ph.D. za jeho čas, který věnoval konzultacím a radám při vypracování této práce. Dále děkuji Ing. Janu Širokému a firmě Energocentrum Plus, s.r.o. za cenné rady v rámci konzultací a za poskytnutí dat použitých v této práci. Snadný a efektivní vývoj programů byl možný díky open-source programovacímu jazyku Python a podpoře komunity vyvíjející moduly Numpy a Matplotlib. Také samozřejmě děkuji své rodině, díky níž jsem mohl tuto práci vypracovat v klidu a pohodě.

## OBSAH

1. Soupis použitého značení a zkratek.....	8
2. Úvod .....	10
3. Využití neuronových sítí pro predikci energie v literatuře.....	11
4. Popis centrálního zdroje tepla.....	17
4.1. Kotle Loos UNIMAT .....	21
4.2. Kogenerační jednotka TEDOM Quanto.....	22
4.3. Kalorimetry Kamstrup MULTICAL.....	24
4.4. Teplotní čidlo Siemens QAE2120.010.....	26
4.5. Teplotní čidlo Domat ETF1.....	27
5. Použité metody.....	28
5.1. Získání a filtrování vstupních dat.....	28
5.2. Korelační analýza.....	29
5.3. Klouzavý průměr.....	29
5.4. Normalizace dat .....	30
5.5. Metoda hlavních komponent (PCA) .....	31
5.6. Použité neuronové modely .....	33
5.7. Učící algoritmy .....	39
5.8. Hodnotící kritéria .....	43
6. Ověření funkčnosti algoritmů .....	44
6.1. Simulační model .....	44
6.2. Testované algoritmy .....	44
6.3. Výsledky testování na simulačních datech.....	45
7. Experimentální analýza .....	47
7.1. Příprava dat .....	47
7.2. Analýza dat .....	48
7.3. Navržené neuronové modely .....	51
7.4. Optimalizace parametrů modelů .....	55
7.5. Dosažené výsledky na reálných datech.....	58
8. Závěr.....	61
9. Použité zdroje.....	64
10. Přílohy .....	68

# 1. SOUPIS POUŽITÉHO ZNAČENÍ A ZKRATEK

## Značení:

vektory jsou značeny tučně, např.  $\mathbf{x}$ , matice velkým znakem a tučně, např.  $\mathbf{X}$

$C_x$	kovarianční matice z matice $X$
$E$	chybová funkce
$E_{MJ}$	energie
$H$	Hessova matice
$I$	jednotková matice
$J$	Jakobiho matice
$N$	celkový počet vzorků
$P$	transformační matice
$R^2$	koeficient determinace
$RMSE$	odmocnina střední kvadratické chyby
$R_n$	poměr součtu všech vlastních čísel k součtu $n$ vlastních čísel
$T$	teplota
$V$	objem
$c_i$	$i$ -té centrum RBF sítě
$d$	Euklidovská vzdálenost vektorů
$e$	vlastní vektor
$e_k$	odchylka predikce pro $k$ -tý vzorek
$g$	gradient chybové funkce
$q$	redukční parametr učení ARPROP
$r$	korelační koeficient
$t$	časový údaj
$u$	vstupní veličina
$w$	váha modelu
$\Delta w$	přírůstek váhy modelu
$x$	prvek vstupního vektoru
$\bar{x}$	průměrná hodnota veličiny $x$
$y$	výstupní veličina
$z_i$	výstup $i$ -tého neuronu ve skryté vrstvě sítě MLP
$\alpha$	redukční parametr normalizace
$\beta$	šířka radiálně bázové funkce
$\Delta_i$	velikost přírůstku $i$ -té váhy
$\eta$	faktor změny vah modelu
$\lambda$	vlastní číslo
$\mu$	rychlost učení
$v$	výstup agregační funkce neuronu (synaptické operace)

---

$\sigma_x$	směrodatná odchylka veličiny $x$
$\tau$	velikost průměrovacího okna
$\phi(v)$	aktivační funkce neuronu (somatická operace)

**Indexy:**

CGT	veličina po aplikaci CGT
$k$	$k$ -tý vzorek průběhu veličiny
$n$	počet prvků
NORM	normalizovaná veličina
PCA	redukovaná data metodou PCA
$R$	skutečná hodnota veličiny
$T$	transpozice

**Zkratky:**

ARPROP	Advanced Resilient Propagation
CGT	Coarse Graining Technique
CNU	Cubic Neural Unit, kubická neuronová jednotka
CZT	Centrální Zdroj Tepla
GD	Gradient Descent
HONU	Higher Order Neural Unit
KISS	Keep It Simple, Stupid, zachovej to jednoduché, hlupáku!
LIFO	Last In First Out, poslední dovnitř, první ven
LM	Levenberg-Marquardt
LNU	Linear Neural Unit, lineární neuronová jednotka
MISO	Multiple-Input Single-Output, více vstupů, jeden výstup
MLP	Multi Layer Perceptron, vícevrstvá perceptronová síť
NN	Neural Network, neuronová síť
NS	Neuronová Síť
NU	Neural Unit, neuronová jednotka
PC	Principal Component, hlavní komponent
PCA	Principal Component Analysis, analýza hlavních komponent
PS	Předávací Stanice
QNU	Quadratic Neural Unit, kvadratická neuronová jednotka
RBF	Radial Basis Function
RMSE	Root Mean Square Error, odmocnina střední kvadratické chyby
SVM	Support Vector Machines
ÚT	Ústřední Topení



## 2. ÚVOD

Spotřeba tepelné energie na vytápění a ohřev teplé vody v domácnostech tvoří cca 14 % z energie celkem (včetně průmyslu a dopravy) spotřebované v rámci České republiky [1]. Proto i malé snížení spotřeby může znamenat významné finanční úspory. V zájmu snižování spotřeby energie lze udělat řadu opatření. Jedním z nich je snaha o předvídání budoucí spotřeby energie. Na základě této předpovědi lze založit například řízení a regulaci kotelny, která bude vyrábět vždy jen tolik tepla, kolik bude třeba.

Tato práce se zabývá návrhem modelu spotřeby energie malého města s cca šesti tisíci obyvateli, kde je energie vyráběna v centrální kotelně a následně distribuována přes výměňkové předávací stanice ke koncovým odběratelům.

V době zpracovávání této práce se centrální zdroj tepla skládal z kaskády tří plynových kotlů, plynové kogenerační jednotky a ze dvou akumulčních nádrží. Zařízení bylo osazeno pouze základní sestavou čidel, na základě jejichž hodnot byly zdroje tepla regulovány. U měření skutečně vyrobené energie, byl problém s přesností měřidel a také v absenci měření spotřeby plynu. Kaskáda kotlů je regulována na základě nastavitelné ekvitermní křivky s omezením na minimální teplotu vratné vody. Kogenerační jednotka je spínána dle proměnného časového programu, který se odvíjí především od cen elektrické energie. Akumulační nádrže jsou využívány převážně v letních měsících, kdy se do nich ukládají přebytky vyrobené kogenerační jednotkou a následně je energie uvolňována v době s menší potřebou tepla.

Regulace tradičními metodami není vždy nejúspornější způsob řízení tepelných zdrojů. Jednou z cest je optimální řízení na základě modelu spotřeby energie. Návrh tohoto modelu s nejlepší dosažitelnou přesností je cílem této práce. Úkolem je vypracovat takový systém řízení, který umožní předpovědět celkovou spotřebu města po hodinách na 24 hodin dopředu a také schopnost vypočítat typickou spotřebu pro kterýkoliv den v roce na základě typického průběhu venkovní teploty. Práce se zabývá možnostmi využití různých neuronových struktur pro tuto predikci. Nejprve je nutné získat vhodná vstupní data, která je třeba upravit (filtrovat) před použitím. Dále je naprogramováno několik různých modelů, které jsou vzájemně porovnány hodnotícími kritérii.

### 3. VYUŽITÍ NEURONOVÝCH SÍTÍ PRO PREDIKCI ENERGIE V LITERATUŘE

V devadesátých letech minulého století zažily neuronové sítě (dále NS) velký rozvoj díky několika významným objevům. Například učící algoritmus backpropagation [2], který umožňuje efektivně učit i vícevrstvé NS. V tomto období bylo napsáno mnoho prací zabývajících se využitím neuronových sítí. Mimo jiné i predikcí spotřeby energie. V této kapitole bude zmíněno několik odborných článků týkajících se této problematiky.

Autoři článku [3] zpracovali přehled článků za období 1991 až 1999 zabývajících se neuronovými sítěmi pro předpovídání spotřeby elektrické energie. Popisují v té době stále spíše skeptický přístup k využití NS pro predikci spotřeby energie, pramenící převážně z nedostatku systematicky zpracovaných studií. Většina popisovaných článků využívá sítě typu MLP<sup>1</sup>, které mají buď jeden výstup pro predikci spotřeby v následující hodině, nebo typicky 24 výstupů pro predikci spotřeby na 24 hodin dopředu. S ohledem na to, že neuronové sítě jsou velice citlivé na kvalitu vstupních dat, je příprava dat je nedílnou součástí návrhu modelu. Pro základní filtrování nevhodných vzorků je použit například Kalmanův filtr a data jsou často data rozdělena na menší lokální modely, které se je NS schopna lépe naučit než velký globální model. Klasickým příkladem takového rozdělení jsou pracovní dny a víkendy (svátky). Rozdělení je provedeno dvěma způsoby, buď jsou přímo data rozdělena a jednotlivé lokální modely se učí nezávisle, nebo pro rozdělení slouží speciální vstupy, které obsahují informaci o čase.

Článek	Parametrů	Počet NS	Celkem parametrů	Cvičná data	Test. data
Chen, Yu, 1992	163	1	163	336	14
Ho, Hsu, Yang, 1992	2881	1	2881	30	4
Kim, Park, 1995	97	1	97	?	120
Park, Marks II, 1991	81	1	81	840	30
Peng, Hubele, 1992	57	1	57	7	365
AlFuhaid, El-Sayed, 1997	6768	2	6931	365	365
Bakirtzis, Petridis, 1996	2136	1	2136	270	365
Chueiki, Ahalt, 1997	3132	1	3132	1460	365
Chow, Leung, 1996	8610	1	8610	136	40
Drezga, Rahman, 1998	171	2	342	144	56
Khotanzad, Abaye, 1997	421–1081	2	10104–25944	1095	182
Khotanzad, Maratukulam, 1998	3144–6264	2	6288–12528	1095	365
Kiartzis, Zoumas, 1997	4296	1	4296	1460	365
Lamedica, Prudenzi, 1996	2380	12	28560	31	21
Lee, Cha, Park, 1992	5384	6	32004	?	182
Lu, Wu, Vemuri, 1993	654	12	7848	365	84
Mohammed, Park, 1995	22–52	105	2310–5460	1095	365
Papalexopoulos, Hao, 1994	2472	1	2472	1825	365
Piras, Germond, Imhof, 1996	276	2	552	365	365
Srinivasan, Persaud, 1997	1860	3	5580	30	10
Liu, Subbarayan, Shoults, 1996	2361	1	2361	1440	?

Tab. 3.1 - Přehled velikostí NS popisovaných v [3]

<sup>1</sup> Multilayer Perceptron - vícevrstvá NS složenou z perceptronů, typicky s jednou skrytou vrstvou

V Tab. 3.1 jsou přehledně rozepsány konfigurace sítí, kterými se zabývá [3]. První sloupec je počet parametrů jedné MLP sítě, druhý je počet sítí pro predikční model, třetí je celkový počet parametrů modelu (váhy, prahové hodnoty). Poslední dva sloupce uvádí velikosti cvičné a testovací množiny ve dnech. Nevyplněné hodnoty nebyly ve zdrojovém článku uvedeny.

Na rozdíl od předchozího se článek [4] zabývá širším spektrem metod predikce spotřeby energie. Pro každou metodu vypisuje přehled nejdůležitějších prací, kde je použita. Od modelů založených na NS, přes SVM<sup>2</sup> a statistické modely až po fyzikální modely. Autor popisuje úspěšné aplikace NS pro předvídání nejen spotřeby tepelné energie, ale i například pro určení parametrů budovy (tepelná kapacita, tepelné ztráty). Většinou se jedná o NS typu MLP s učícím algoritmem backpropagation, případně je použita metoda PCA<sup>3</sup> pro snížení dimenze vstupních dat. V závěru přehledu je porovnání jednotlivých metod z hlediska snadnosti použití a přesnosti výsledků, viz Tab. 3.2.

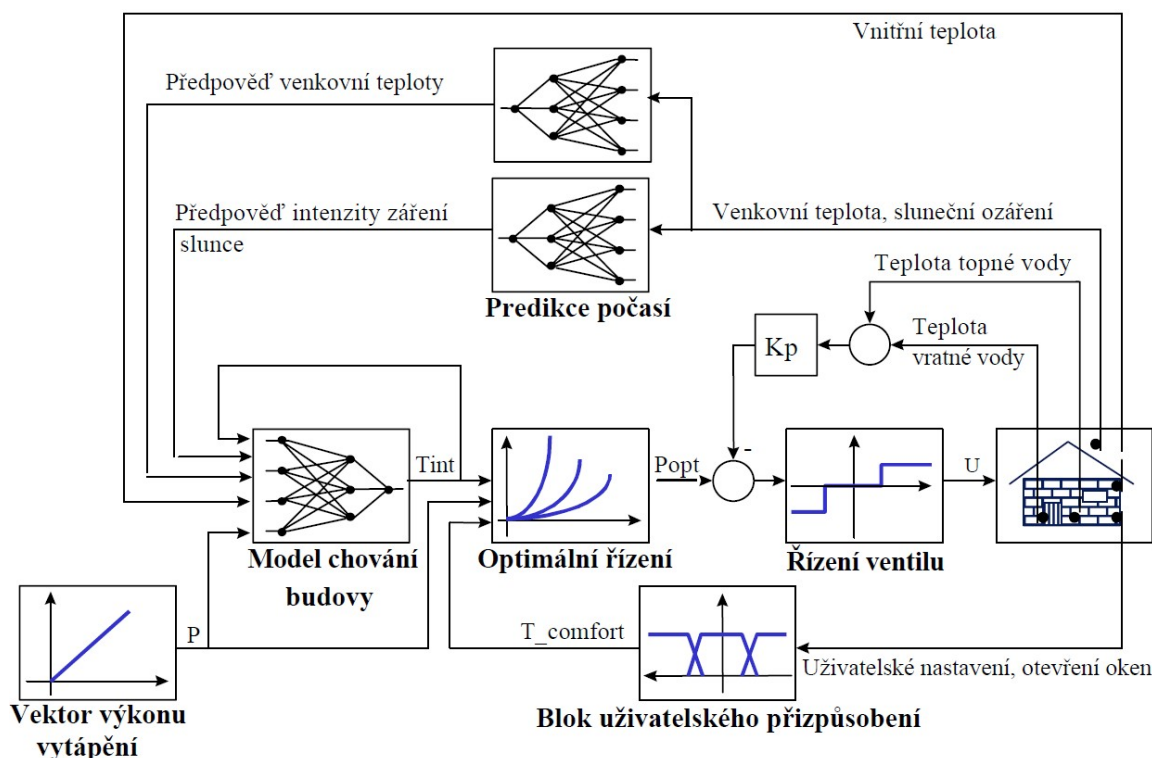
Metoda	Složitost modelu	Jednoduché použití	Rychlost	Vstupy	Přesnost
Fyzikální podrobná	Velmi velká	Ne	Nízká	Podrobné	Velmi velká
Fyzikální zjednodušená	Velká	Ano	Vysoká	Zjednodušené	Velká
Statistická	Střední	Ano	Střední	Historická data	Střední
Neuronová síť	Velká	Ne	Vysoká	Historická data	Velká
SVM	Velmi velká	Ne	Nízká	Historická data	Velmi velká

Tab. 3.2 - Porovnání nejpoužívanějších metod pro predikci spotřeby energie [4]

Článek [5] se zabývá návrhem systému NEUROBAT. Je to adaptivní systém řízení vytápění budovy na základě modelů vytvořených pomocí NS. Je to komplexní řídicí systém, který řeší celou problematiku od vytvoření modelu budovy až po samotné optimální řízení vytápění. Na Obr. 3.1 je vidět, kde se například může nacházet model (ať už budovy nebo města) v celkovém řídicím systému. Tato diplomová práce se zabývá pouze návrhem modelu, který lze ovšem dále úspěšně využít pro optimální řízení kotelny. Pro vytvoření prediktivního modelu autoři článku [5] používají NS typu MLP s jednou skrytou vrstvou a pro učení sítě je využit algoritmus Levenberg-Marquardt. Jako vstupy modelu jsou použity zpožděné hodnoty předchozích naměřených hodnot a jejich průměry za 24 hodin. Veškeré potřebné veličiny jsou snímány přímo ze senzorů, autoři se nezmiňují, zda použili filtrování či jiné metody úpravy dat. Navržený regulátor byl testován přes dvě topné sezóny a porovnáván s referenčním (běžně komerčně dostupným) regulátorem. Regulátor NEUROBAT dosáhl celkem 13% úspory energie, při mírně větším komfortu obyvatel (teplota a vlhkost se méně často nacházela mimo zónu komfortu).

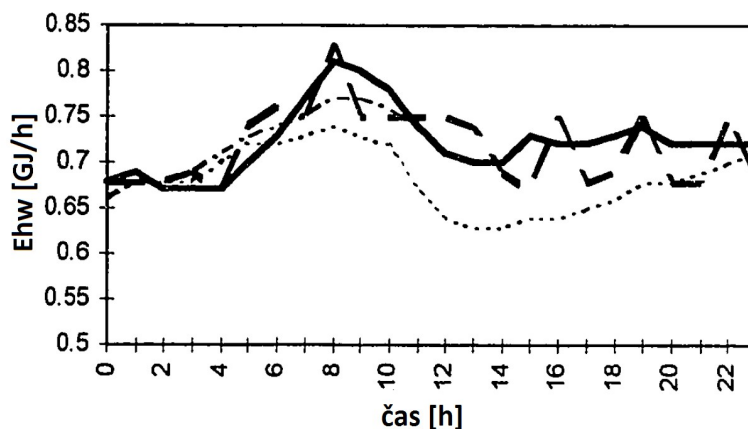
<sup>2</sup> Support vector machines - metoda strojového učení, více viz [7]

<sup>3</sup> Principal component analysis - metoda hlavních komponent, vysvětlena dále



Obr. 3.1 - Zařazení modelu chování budovy v řídicím systému vytápění [5]

Cílem disertační práce [6] je vývoj hodinového modelu spotřeby energie v komerčních budovách s použitím Fourierových řad a NS. Je navržena prediktivní NS s jednou skrytou vrstvou založenou na dvourozměrných waveletech<sup>4</sup>. Jako vstupy zde slouží pouze venkovní teplota a hodina dne. Výhodou tohoto přístupu je, že stačí pouze jedna vstupní měřená veličina, když měření vlhkosti nebo slunečního záření nejsou k dispozici. Wavelety byly pro tuto práci zvoleny z důvodu jejich dobrých lokalizačních schopností. NS založená na waveletech se nazývá WaveNet a využívá pokročilé učící algoritmy, které se ukázaly být značně rychlejší než backpropagation.

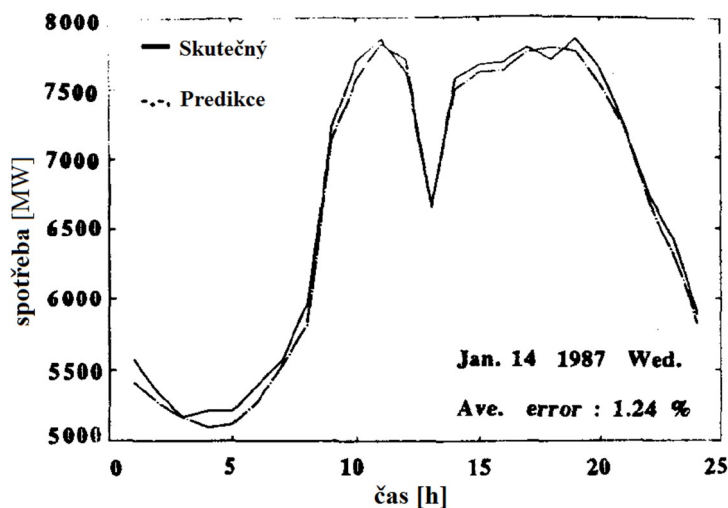


Obr. 3.2 - Průběh predikce tří modelů v porovnání se skutečným průběhem (čárkovaná čára) [6]

<sup>4</sup> Wavelet - vlnková transformace, používá se jako prostředek pro de Korelaci dat

Autor práce [6] navrhl tři verze modelu (Obr. 3.2), kde první (tečkovaná čára) je pouze lineární model, druhá (čerchovaná čára) a třetí (plná čára) jsou Wave-Net modely. Bylo také provedeno porovnání s tradiční NS s učícím algoritmem backpropagation. Parametry sítě pro porovnání byly následující: 3 vstupy (hodina dne, venkovní teplota, den v týdnu), dvě skryté vrstvy a sigmoidální aktivační funkce. Ukázalo se, že predikční model Wave-Net byl přesnější ve třech případech z osmi, proto je jeho hlavní výhodou pouze mnohonásobně vyšší rychlost učení oproti backpropagation. Jako trénovací data bylo použito 6 měsíců (leden až červen) a testování probíhalo na vybraných dnech v druhé polovině roku. Před použitím dat bylo nejdříve provedeno základní filtrování odstraňující chybné a nevhodné hodnoty. Dále proběhla kontrola průběhu venkovní teploty na základě porovnání s dostupnými daty z několika okolních meteorologických stanic.

Článek [8] ukazuje na nejčastější problémy, na které výzkumníci naráží při používání NS pro krátkodobou predikci spotřeby energie. Také se zabývá otázkou, zda a jak moc jsou prediktivní modely založené na NS závislé na systému (lze stejný model použít pro různé budovy?) nebo na období (ovlivnění modelu ročními obdobími, špatnými daty, svátky, apod.). Autoři založili prediktivní model na síti typu MLP s učícím algoritmem backpropagation. Vstupy jsou průměrné teploty za několik posledních dní a hodin, předpověď teploty, hodina dne, den v týdnu a několik posledních vzorků skutečné spotřeby energie. Počty zpožděných vzorků byly určeny pomocí korelační analýzy. Data ze dvou různých budov byla rozdělena na trénovací část (1 až 2 měsíce, pohyblivé okno) a testovací část (1 týden). Ukázalo se, že pro různé budovy bylo třeba navrhnout NS s jinou strukturou, kde navíc nebyla žádná přesně daná kritéria, podle kterých by se struktura měla změnit. Na druhou stranu se ukázalo, že jakmile je model jednou správně naučen, už je poměrně necitlivý na menší změny (svátky, víkendy) a chyby v datech, s výjimkou velkých výkyvů (změny ročních období), kdy už je nutné model přeučit.



Obr. 3.3 - Porovnání průběhů spotřeby energie pro predikci 24 hodin z článku [8]

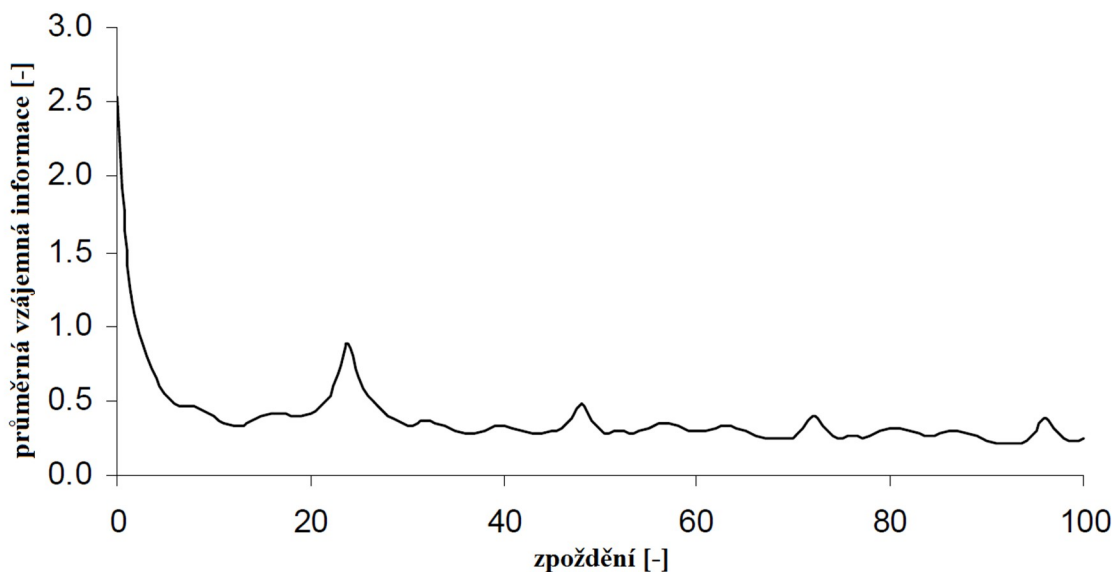


Predikcí spotřeby elektrické energie, se speciálním zaměřením na zvýšení přesnosti o víkendech a změnách ročních období, se zabývá článek [9]. Autorům nejde o přesný průběh spotřeby, ale o schopnost předpovědět hodnotu denní špičky, která je důležitá pro mnoho aplikací. Je využita síť typu MLP s jednou skrytou vrstvou, sigmoidální aktivační funkce a učení pomocí algoritmu backpropagation. Vstupní data jsou normalizována na interval od 0 do 1. NS má celkem 23 vstupů, kde:

- první je vždy jedničkový vstup,
- dalších pět je pro časové rozdělení (víkendy, svátky, roční období),
- sedmý a osmý vstup jsou minimální a maximální teplota pro den který se predikuje (hodnoty z předpovědi počasí),
- zbylých 15 vstupů jsou zpožděné hodnoty energetických špiček za posledních 15 dní.

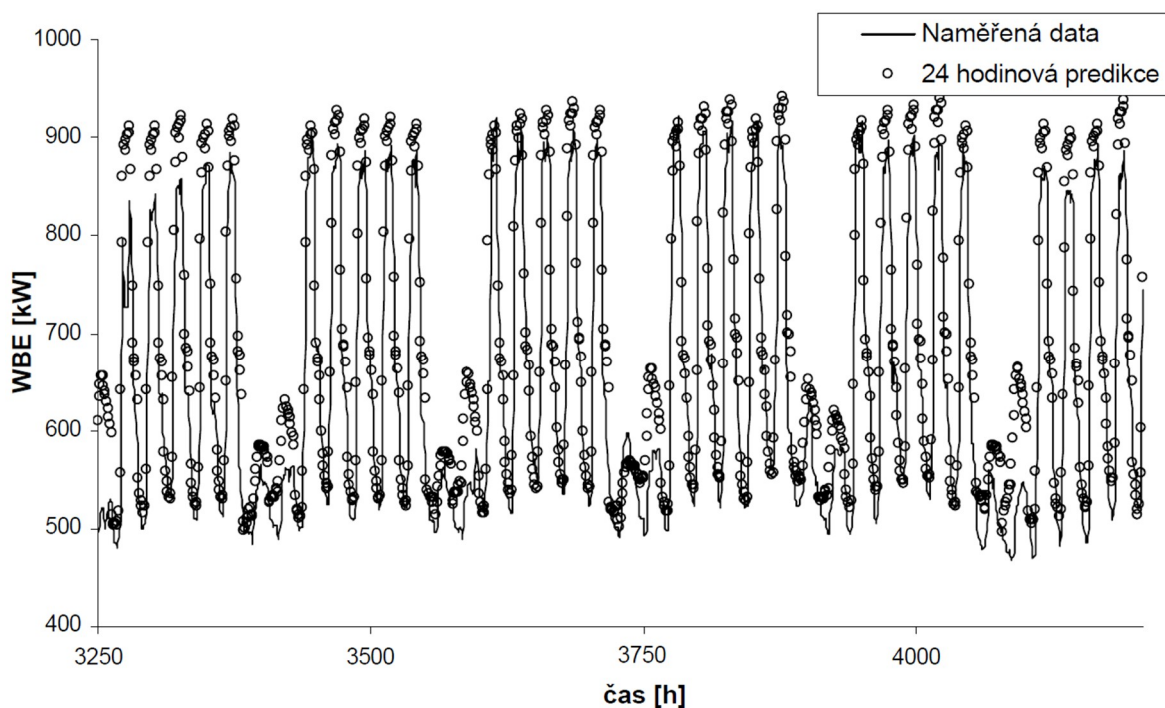
NS se skládá z 27 neuronů ve skryté vrstvě a trénovací data jsou vždy dána oknem o velikosti 15 dní, které se postupně posouvá. Horší předpověď se objevuje ve chvíli, kdy dojde k nenadálé změně počasí. Autoři tento problém řeší pomocí vstupů pro časové rozdělení a tím se snaží „vyhladit“ vzdálenost mezi vzorky. V závěru je porovnání navrženého modelu s modelem, který používal dodavatel elektrické energie (Southern Company), založeným na metodě nejmenších čtverců. Jedná se o předpověď na 7 dní dopředu a pouze o hodnoty špičkových odběrů, tedy 7 hodnot. Pro typický týden v srpnu 1991 je průměrná chyba neuronového modelu 1,4 %, oproti modelu Southern Company s chybou 4,7 %.

Poslední popisovaný článek [10] využívá metody pro analýzu chaotických časových řad pro určení základních parametrů NS. Predikci nejvíce ovlivňuje velikost okna trénovacích dat a počet zpožděných vzorků na vstupech NS. Motivace autorů pramení hlavně ze složitosti dostupných nástrojů pro tvorbu modelů, také



Obr. 3.4 - Časové zpoždění, první minimum pro  $\tau = 12$  [10]

často vyžadující řadu vstupních veličin, které nejsou vždy k dispozici, proto je navržen model nevyžadující žádné jiné vstupní veličiny než samotnou predikovanou. Model je navržen pro předpověď spotřeby energie ve velké budově na 24 hodin dopředu. Byla použita data ze soutěže Shootout I, organizované organizací ASHRAE. Standardní síť MLP se sigmoidálními aktivačními funkcemi, je učena pomocí algoritmu Levenberg-Marquardt. Časové zpoždění vstupního vektoru je určeno pomocí průměrné vzájemné informace, která odpovídá hodnotě prvního minima v grafu (Obr. 3.4). Pro určení vnořené dimenze je použita metoda falešných nejbližších sousedů. Rozhodující je dimenze, kde už je procento falešných nejbližších sousedů téměř nulové. Vnořená dimenze tedy odpovídá počtu zpožděných hodnot predikované veličiny na vstupu. Dále byl zaveden speciální vstup pro rozlišení víkendů a pracovních dní. Chyba modelu pro predikci spotřeby na 24 hodin dopředu byla 11 % (Obr. 3.5).

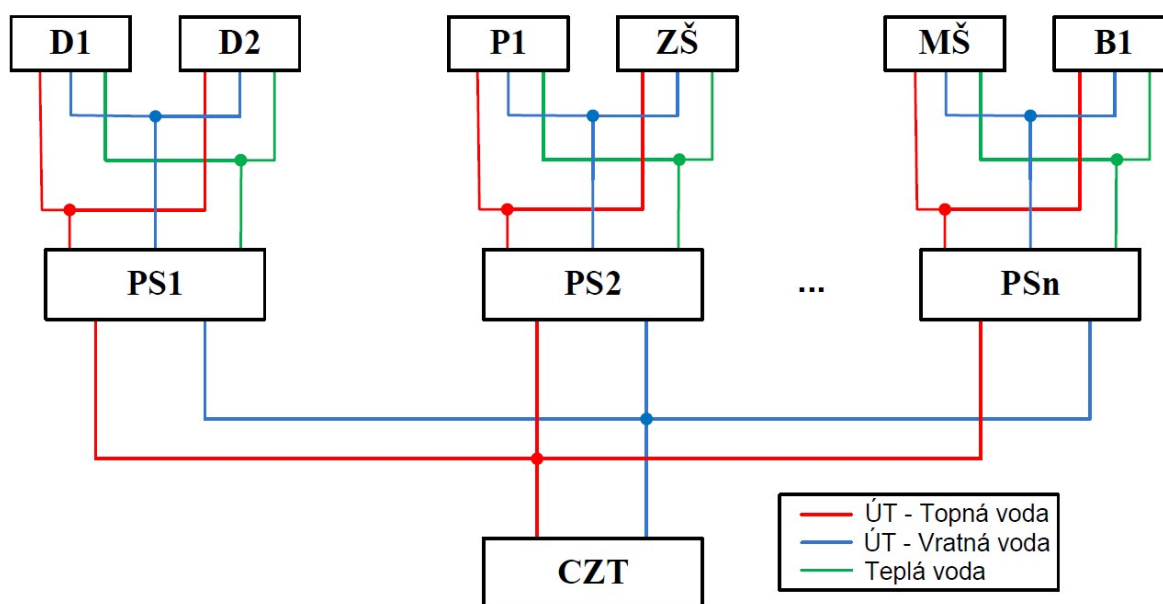


Obr. 3.5 - Porovnání průběhů predikované a skutečné spotřeby energie [10]

Výše uvedené práce ukazují možnosti využití neuronových sítí pro predikci spotřeby energie. Pro daný problém je možné NS úspěšně použít.

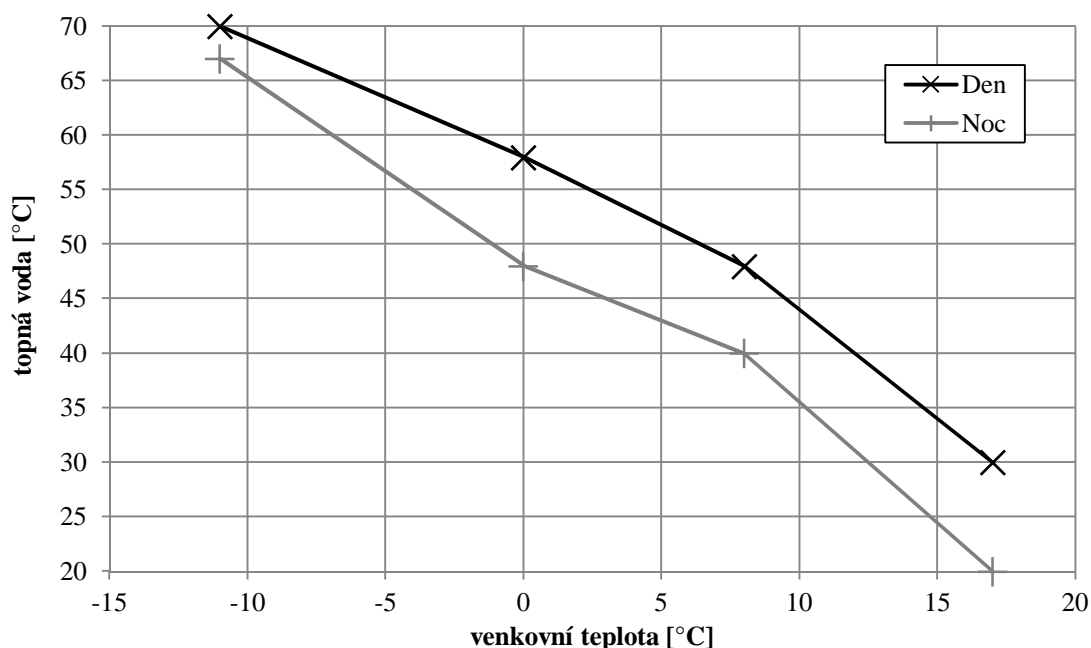
## 4. POPIS CENTRÁLNÍHO ZDROJE TEPLA

Menší město s cca 6000 tisíci obyvateli má zajištěné dodávky tepla (vytápění i teplou vodu) pomocí centrálního zdroje tepla. Tato práce se zabývá návrhem modelu spotřeby energie tohoto města. Dodávky teplé vody i tepla pro vytápění probíhají přes předávací stanice, kde se teplo z centrálního zdroje tepla rozdělí na vytápění a teplou vodu pomocí výměníků. Principiální schéma dodávek tepla do města viz Obr. 4.1.



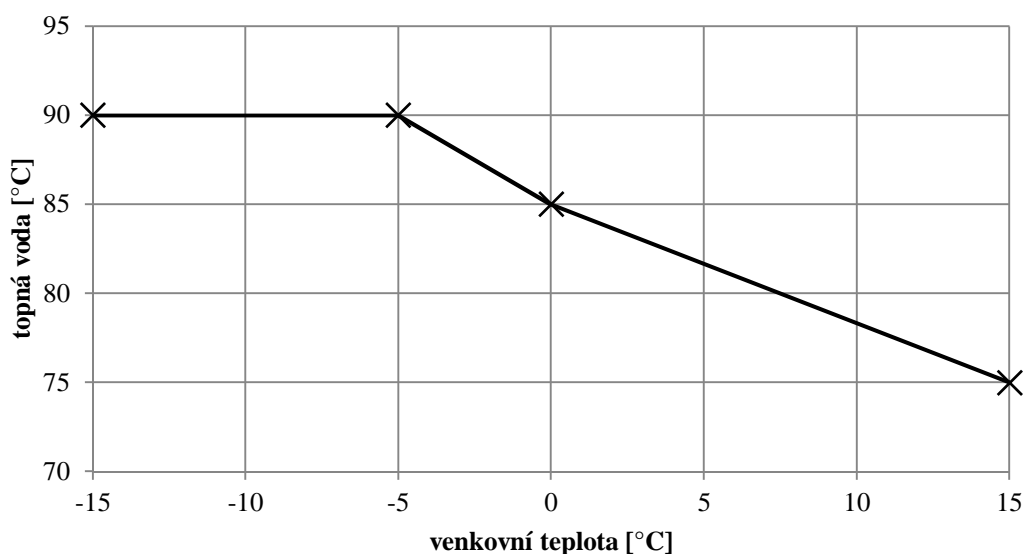
Obr. 4.1 - Blokové schéma dodávek tepla do města

Centrální zdroj tepla (CZT) dodává teplo pro celé město přes výměňkové předávací stanice (PSn) k různým odběratelům (D1, D2, P1, ZŠ, MŠ, B1,...). Každá předávací stanice má dle potřeby nastavené vlastní parametry pro teplou vodu i vytápění. Pro dodávky teplé vody je vždy nastavena teplota a časový program a také režim cirkulace. Dále má každá předávací stanice zabudovanou ochranu proti bakterii *Legionella pneumophila*, pomocí nastaveného časového programu (typicky mezi 23. hodinou a půlnocí), kdy se zvýší teplota na 75 °C. Druhou funkcí předávacích stanic je zajištění vytápění budov odběratelů. Některé předávací stanice mají více okruhů (až 5), ale většina stanic má pouze jeden. Teplota topné vody je dána ekvitermní křivkou a tedy venkovní teplotou. Ovládací software umožňuje nastavení křivky ve čtyřech bodech (Obr. 4.2). Další funkcí je časový program, který umožňuje například nastavit noční korekce, kdy je možné snížit teplotu topení. Celkem je na zdroj tepla připojeno 45 předávacích stanic. Výstup z kotelny je rozdělen na dva okruhy, na první je připojeno původních 35 předávacích stanic a na druhý zbývajících deset, které slouží pro novější část města.



Obr. 4.2 - Příklad ekvitermní křivky s noční korekcí pro předávací stanici

Centrální zdroj tepla se skládá z kaskády tří kotlů, kogenerační jednotky a dvou akumulčních nádrží. Zjednodušené schéma zapojení je v příloze 1. Hydraulicky je soustava (a tedy oběhová čerpadla) regulována na konstantní dopravní tlak. Pořadí kaskády kotlů lze nastavit ve třech režimech (123, 213, 312). V době zpracování této práce byl zvolen režim 312. Kotle 1 a 2 mají výkon každý 3,6 MW a třetí kotel 1,6 MW. Podrobný popis kotlů je níže. Žádané výstupní teploty z kaskády kotlů, která je dána vlastní ekvitermní křivkou (Obr. 4.3), je dosaženo kvalitativní regulací směřováním. Minimální teplota vratné vody je nastavena na 60 °C, pro zamezení nízkoteplotní korozi kotle. Tato ochrana je zajištěna spojitou regulací teploty vody ve vratném potrubí přes trojcestný směšovací ventil [11].



Obr. 4.3 - Ekvitermní křivka pro kaskádu kotlů

Kogenerační jednotka o výkonu 1,6 MW využívá pouze dvoupolohovou regulaci. Podrobné parametry jednotky jsou uvedeny níže. Její spínání je realizováno na základě časového programu, který je pravidelně manuálně upravován, převážně na základě tarifů elektrické energie. Jednotka je připojena paralelně ke kaskádě kotlů. Posledním zdrojovým prvkem jsou dvě akumulární nádrže o celkovém objemu 200 m<sup>3</sup>. Jejich nabíjení je možné pouze z kogenerační jednotky a to pomocí dvou klapek, které přepnou výstup z jednotky buď směrem do města, nebo do nádrží. Pokud je aktivní vybíjení nádrží, kotle a kogenerační jednotka jsou neaktivní. Vybíjení probíhá systémem LIFO<sup>5</sup>, tedy první jde do oběhu nejteplejší voda. Dále jsou nádrže vybaveny otopnými kabely, které v případě nutnosti udržují teplotu vody v nádržích nad 3 °C.

V době zpracovávání této práce došlo ke změně programu ovládání nádrží, proto jsou zde popsány oba přístupy. Původní postup byl založen na časovém programu, který každý den definoval nabíjecí čas (cca 4 hodiny), kdy se nádrže nabíjely do 100 % (odpovídá teplotě 85 °C). Následně probíhalo vybíjení až do 0 % (60 °C) v době s menší potřebou tepla (typicky přes noc). Používání nádrží bylo omezeno minimální venkovní teplotou 16 °C, tedy převážně mimo topnou sezónu. Většina dat, která používá tato práce, je právě z tohoto období. Nový program využívá nádrží pro odkládání tepla v době, kdy není třeba. Tedy jsou dané dvě nabíjecí podmínky, minimální venkovní teplota 16 °C a maximální teplota vratné vody 73 °C. Jakmile překročí teplota vratné vody tuto nastavitelnou mez, začnou se nádrže nabíjet. Vybíjení opět probíhá v době se sníženou potřebou tepla.

Z dalšího vybavení má kotelná automaticky řízené větrání. Zařízení pro automatické dopouštění systému na základě tlaku systému, který se pohybuje v rozmezí 360 až 450 kPa. V případě většího překročení nebo podkročení žádaného tlaku dojde k havarijnímu odstavení systému. Součástí vybavení je i vlastní vytápění celé budovy kotelny, které je připojeno jako třetí okruh na hlavní zdroj tepla.

Centrální zdroj tepla je vybaven řadou senzorů, které snímají veličiny potřebné pro regulaci. Důležitou kategorií je snímání poruch. Jsou sledovány poruchy komunikace (např. s kogenerační jednotkou), dodávky plynu, minimální tlak v systému, maximální teplota v prostoru kotelny, zaplavení prostoru kotelny, funkčnost ventilů i klapek. Pro regulaci je nepostradatelnou veličinou venkovní teplota. Její senzor se nachází přímo na budově kotelny včetně sledování poruch tohoto senzoru. V případě potřeby je teplota brána z náhradního senzoru. Systém doplňování média potřebuje pro svou funkci řadu senzorů i akčních členů. Probíhá

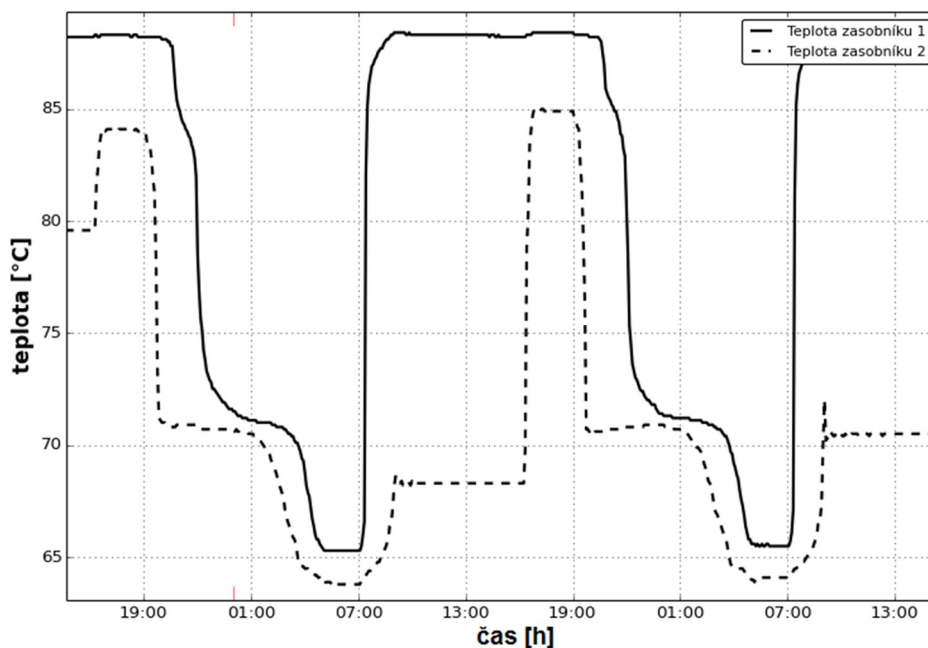
<sup>5</sup> LIFO - Last In, First Out, poslední dovnitř, první ven



zde snímání aktuálního tlaku v systému, na jehož základě jsou ovládána čerpadla a ventily pro dopouštění a odpouštění. Oběhové čerpadlo druhého okruhu pro 10 předávacích stanic je řízeno pomocí frekvenčního měniče podle žádané tlakové difference.

V prostoru kotelny je měřena teplota okolí, celková teplota topné a vratné vody, což umožňuje pomocí směšovacího ventilu udržovat minimální teplotu vratné vody. Na každém kotli je monitorována teplota vstupní a výstupní vody a jeho provozní hodiny. Je možné modulovat jeho výkon. Moderní kogenerační jednotka sleduje mnoho provozních parametrů. Základem jsou vstupní a výstupní teploty vody, dále stav motoru (otáčky, motohodiny, teplota a tlak oleje, napětí baterie) a samozřejmě parametry generátoru (činný výkon, frekvence, napětí mezi fázemi, vyráběný proud na každé fázi). U akumulčních nádrží jsou monitorovány teploty ve třech různých hladinách, ze kterých je vypočítávána průměrná teplota nádrže (Obr. 4.4). Je také měřena teplota potrubí za nádržemi pro případ „přebití“ nádrží.

U kogenerační jednotky a kaskády kotlů jsou připojeny kalorimetry, které měří vyrobenou energii na základě průtoku a rozdílu teplot. Měřidla počítají energii sumačním způsobem, stejně tak celkové množství proteklé tekutiny.



Obr. 4.4 - Typický průběh cyklu nabití a vybití zásobníků

#### 4.1. Kotle Loos UNIMAT

Jak už bylo řečeno, kaskáda kotlů je složena ze tří kotlů o výkonech 1,6 MW, a dvakrát 3,6 MW. Jedná se o kotle původně vyrobené firmou Loos, které jsou od 2. července 2012 nabízené pod značkou Bosch [12].



Obr. 4.5 - Loos UNIMAT UT-L [13]

Tepl vodní kotel UNIMAT UT-L (Obr. 4.5) v provedení dle směrnice pro plynová zařízení zajišťuje hospodárnou výrobu teplé vody a zároveň dosahuje pouze nízké emise ve spalinách. Tento plamencový žárotrubný kotel s rozsahem výkonu 750 až 19.200 kW produkuje teplou vodu na nízké teplotní a tlakové úrovni [14].

Typ	UNIMAT UT-L
Teplonosné médium	voda
Konstrukce	třítahový žárotrubný kotel
Výkon	750 až 19.200 kW
Tlak	do 16 bar
Účinnost	95 %
Max. teplota	120 °C
Min. teplota vratné vody	50 °C
Palivo	plyn, topný olej

Tab. 4.1 - Základní parametry kotle Loos UNIMAT UT-L [14]

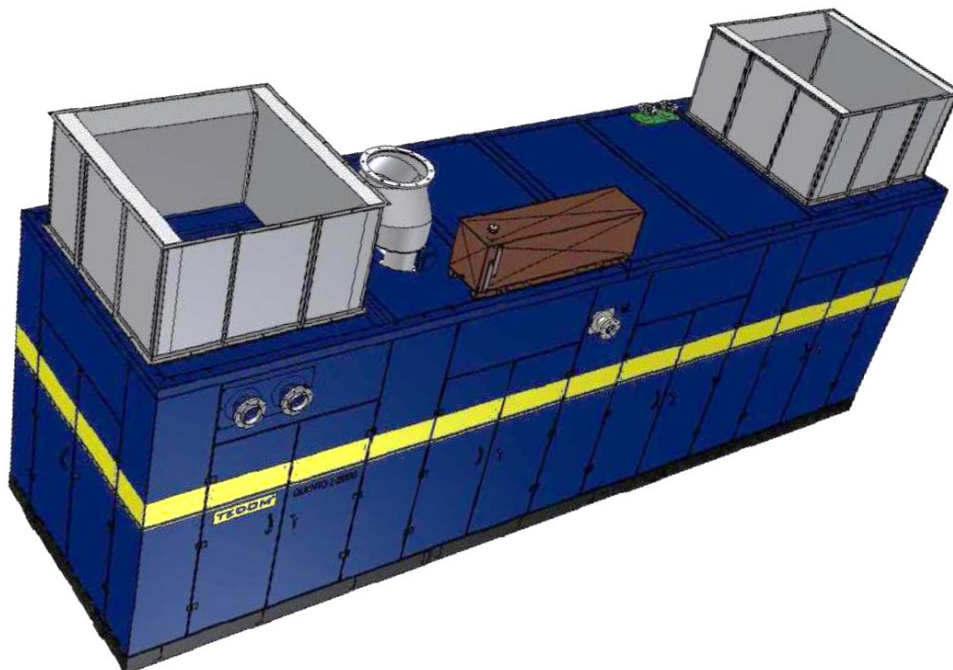
Konstrukce [14]:

- třítahová plamencová žárotrubná konstrukce,
- funkční válcová konstrukce pro optimální tlakovou stabilitu,
- speciální injektor, integrovaný v horní části kotle, pro účinné proudění a vnitřní zvýšení teploty vratné vody,

- konstrukční řada UT s integrovaným spalínovým výměníkem tepla k dosažení vyššího stupně využití tepla bez sirnatých spalin s obtokem nebo bez něj,
- geometrie spalovací komory je odsouhlasena se všemi předními výrobci hořáků,
- tepelná izolace rohožemi z minerální vlny pod ochranným hliníkovým pláštěm,
- kotlové dveře plně výkyvné – dle přání vpravo nebo vlevo, celkový průřez kotle je volně přístupný,
- jednoduchá a lehká kontrola, čištění a revize.

#### 4.2. Kogenerační jednotka TEDOM Quanto

V novější části centrálního zdroje tepla byla v roce 2010 instalována kogenerační jednotka Quanto D1600SP (Obr. 4.6) od české firmy TEDOM s doplňujícími akumulacími nádržemi o objemu 2x100 m<sup>3</sup> o průměru 3,2 m a výšce 15,65 m.



Obr. 4.6 - Kogenerační jednotka TEDOM Quanto v protihlukovém zakrytí

Protihlukové kapotované provedení je určeno především k instalaci do budov. Jeho výhodou je především rychlost instalace a nízká hlučnost. Jednotka Quanto D1600 je určena k instalaci do strojovny a je tvořena několika částmi. První z nich je modul motorgenerátoru obsahující soustrojí motoru s generátorem, umístěné na základovém rámu a opatřené protihlukovým krytem. Další částí je technologický modul, katalyzátor a dvojice tlumičů výfuku k volné zástavbě do spalínovodu strojovny. Dále volně stojící elektrické rozvaděče a plynová trasa určená k zástavbě do plynovodu. Jednotka je určena k provozu na zemní plyn, pro paralelní provoz se sítí 400V/50 Hz. Teplovodní okruh je přizpůsoben teplotnímu spádu 90/70 °C [15].

Pro ovládání kogenerační jednotky je použit řídicí systém ProCon Sight, který zajišťuje plně automatický chod soustrojí. Jedná se o víceprocesorový modulární systém, sestávající z centrální části, zobrazovací jednotky a rozšiřujících modulů analogových a binárních vstupů a výstupů [15].

Způsoby ovládání [15]:

- |                               |   |
|-------------------------------|---|
| Místní                        | - pomocí tlačítek na zobrazovací jednotce řídicího systému.   |
| Dálkové                       | - beznapěťovým kontaktem ,<br>- podle úrovně požadovaného výkonu či úrovně spotřeby objektu,<br>- z místního nebo vzdáleného PC,<br>- pomocí SMS zpráv. |
| Regulace dle spotřeby objektu | - informaci o spotřebě objektu řídicí systém získává z převodníku, který měří směr a velikost odběru/dodávky ze/do sítě.                                |
| Regulace na požadovaný výkon  | - analogovým signálem (např. 0/4 – 20 mA),<br>- datovou cestou – např. prostřednictvím protokolu MODBUS-RTU.  |

Typ	QUANTO D1600
Elektrický výkon	1560 kW
Tepelný výkon	1709 kW
Elektrická účinnost	43,3 %
Tepelná účinnost	47,5 %
Celková účinnost	90,8 %
Spotřeba plynu <sup>6</sup>	381 m <sup>3</sup> /h

Tab. 4.2 - Základní parametry kogenerační jednotky TEDOM Quanto D1600 [15]

K pohonu jednotky je použit plynový spalovací motor TCG 2020 V16, výrobek německé firmy MWM.

Typ	TCG 2020 V16
Vrtání / zdvih	170 / 195 mm
Zdvihový objem	70,8 dm <sup>3</sup>
Otáčky	1500 min <sup>-1</sup>
Délka x šířka x výška	5,43 x 1,81 x 2,21 m
Suchá hmotnost	13320 kg
Kompresní poměr	13,5 : 1

Tab. 4.3 - Základní parametry pohonu jednotky [16]

<sup>6</sup> Spotřeba zemního plynu je uvedena pro zemní plyn s výhřevností 34,2 MJ/m<sup>3</sup> a při fakturačních podmínkách: 15 °C, 101,325 kPa.

### 4.3. Kalorimetry Kamstrup MULTICAL

V kotelně jsou nainstalovány dva kalorimetry Kamstrup MULTICAL. První, pro kaskádu kotlů, je starší typ MULTICAL 66-C, který již trpí občasnými výpadky s dopadem na zpracování dat. Druhý, nainstalovaný pro kogenerační jednotku je MULTICAL 601 (Obr. 4.7). Oba typy kalorimetrů se již nevyrábí (aktuální výrobek je MULTICAL 602).



Obr. 4.7 - Kalorimetr Kamstrup MULTICAL 601

#### MULTICAL 66-C

MULTICAL 66-C je schopný měřit teplo i chlad ve všech aplikacích používajících vodu jako médium, kde se teploty pohybují v rozmezí 2 až 160 °C a průtoky 0,6 až 3000 m<sup>3</sup>/h.

Typ	MULTICAL 66-C
Měřicí teplotní rozsah q (teplo)	10–160 °C
Rozdílový rozsah Δq (teplo)	3–150 K
Měřicí teplotní rozsah q (chlad)	2–30 °C
Rozdílový rozsah Δq (chlad)	3–20 K
Přesnost	±(0,5+Δq <sub>min</sub> /Δq) %
Zobrazovací rozlišení	0,01 K
Teplotní senzory	Pt500
Max. frekvence pulzů průtokoměru	128 Hz

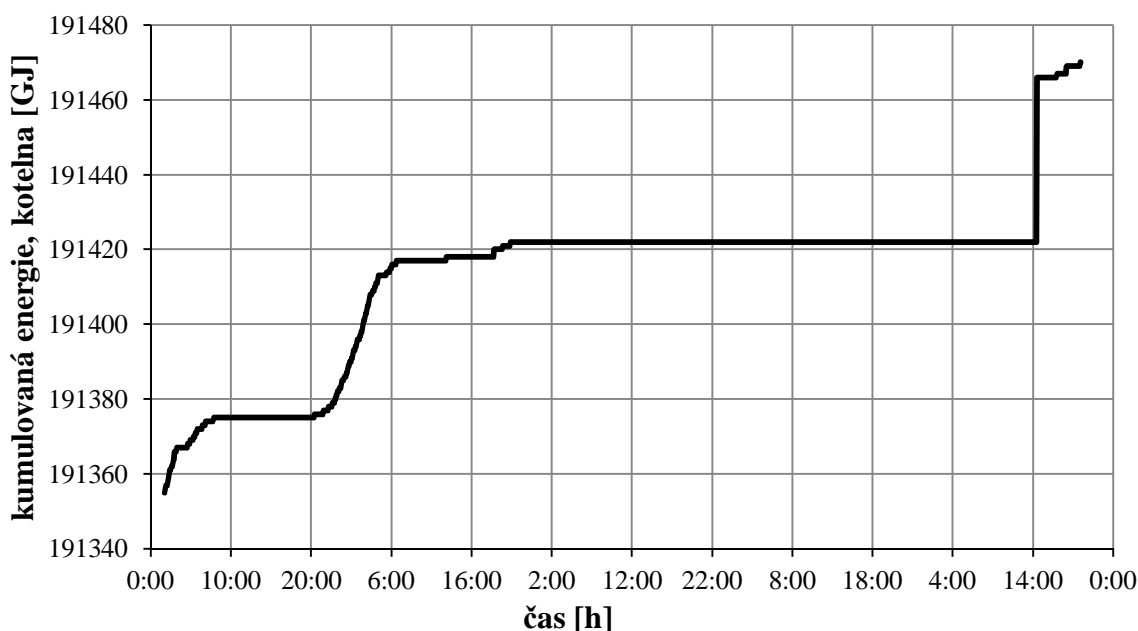
Tab. 4.4 - Základní parametry kalorimetru MULTICAL 66 [17]

Výpočet energie  $E_{MJ}$  (4.1) probíhá na základě proteklého objemu  $V$  za dobu integrace, rozdílu teplot vratné a topné vody a tepelného koeficientu  $k$  vody dle normy EN 1434-1:2004. Vypočtená energie je převedena na požadované jednotky.

$$E_{MJ} = V \cdot (T_{w1} - T_{w2}) \cdot k \quad [\text{MJ}] \quad (4.1)$$

Ke kalorimetru je připojen průtokoměr ULTRAFLOW II. V této části soustavy se dle zatížení pohybují průtoky v rozmezí 5 až 250 m<sup>3</sup>/h. Při tomto rozsahu je

rozlišení vypočítané energie pouze 1 GJ dané rozlišením průtokoměru, což v měření způsobuje znatelné skoky. Dalším problémem tohoto měřidla jsou velice časté výpadky měření, pravděpodobně způsobené dlouhodobým provozem. Naměřená data obsahují hned několik druhů chyb - občasné spadnutí měřené hodnoty na nulu nebo naopak vyskočení na tisícinásobky měřené veličiny, nebo „mrznutí“ hodnoty (Obr. 4.8), kdy se i několik hodin hodnota nemění a pak najednou vyskočí.



Obr. 4.8- Příklad mrznutí hodnoty energie u kalorimetru MULTICAL 66

Na Obr. 4.8 je jeden z horších výpadků měřidla, kdy komunikace vypadla na déle než den. Tento výpadek lze nahradit lineární interpolací, ovšem tyto úpravy pak následně působí velice nepříznivě na kvalitu predikce.

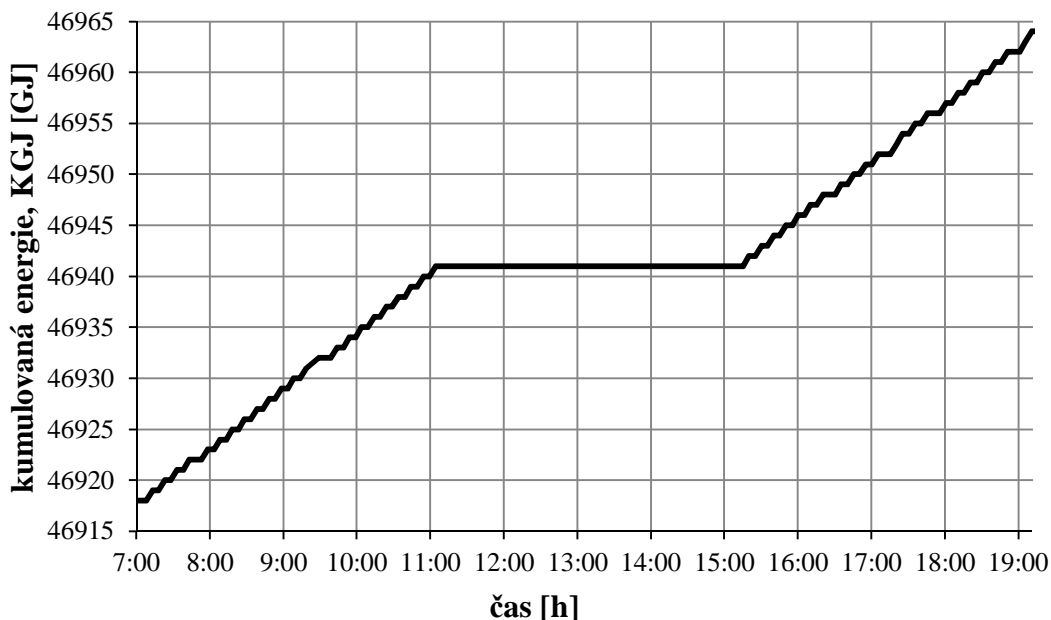
### MULTICAL 601

MULTICAL 601 je novější verzí předchozího kalorimetru a je určen k měření tepelné energie a chladu ve všech systémech s vodním médiem s teplotami od 2 do 180 °C. Rozsah všech průtokoměrů se pohybuje mezi 0,6 až 3000 m<sup>3</sup>/h.

Typ	MULTICAL 601
Měřicí teplotní rozsah q (teplo)	2–180 °C
Rozdílový rozsah Δq (teplo)	3–170 K
Měřicí teplotní rozsah q (chlad)	2–180 °C
Rozdílový rozsah Δq (chlad)	3–170 K
Přesnost	$\pm(0,5+\Delta q_{\min}/\Delta q)$ %
Zobrazovací rozlišení	0,01 K
Teplotní senzory	Pt500 (2 nebo 4 vodičové zapojení)
Max. frekvence pulzů průtokoměru	128 Hz

Tab. 4.5 - Základní parametry kalorimetru MULTICAL 601 [18]

Výpočet energie probíhá opět podle rovnice (3.1). V části systému u kogenerační jednotky je při zapnuté jednotce jmenovitý průtok  $71 \text{ m}^3/\text{h}$ , jinak je téměř nulový. Je použit průtokoměr ULTRAFLOW typ 65, který při tomto průtoku dosahuje také rozlišení pouze 1 GJ (Obr. 4.9). Samotné měřidlo je ovšem novější a komunikace je spolehlivější a proto je v nefiltrovaných datech minimum výpadků.



Obr. 4.9 - Schodovitost měření v důsledku rozlišení kalorimetru 1 GJ

#### 4.4. Teplotní čidlo Siemens QAE2120.010

Pro měření teploty vody jsou použita ponorná teplotní čidla Siemens QAE2120.010 (Obr. 4.10) uložená do jímky.



Obr. 4.10 - Teplotní čidlo Siemens QAE2120.010 [19]

Ponorné čidlo teploty se skládá z těchto částí [19]:

- plastové pouzdro s přípojovací svorkovnicí a odnímatelný kryt,
- ponorná měřicí trubka s měřicím článkem.

Přípojovací svorky jsou přístupné po odstranění krytu. U typu QAE2110.010 se kabel přivádí přes kabelovou průchodku M16.

Typ	QAE2110.010
Měřicí článek	Ni 1000
Délka ponoru	100 mm
Tlaková třída	PN 10
Stupeň krytí	IP 54
Rozsah měření	-30–130 °C
Časová konstanta s jímkou	cca 30 s
Kryt	polykarbonát
Měřicí trubka	nerezová ocel DIN 17 440
Ochranná jímka	mosaz CuZn37

Tab. 4.6 - Základní parametry teplotního čidla Siemens QAE2110.010 [19]

#### 4.5. Teplotní čidlo Domat ETF1

V druhé části kotelny jsou použita ponorná čidla ETF1 (Obr. 4.11) montovaná do jímky, vyrobená firmou Domat Control System, s.r.o.



Obr. 4.11 - Teplotní čidlo Domat ETF1 [20]

Typ	ETF1
Měřicí článek	Pt 1000
Délka ponoru	150 mm
Max. tlak	16 bar
Stupeň krytí	IP 65
Rozsah měření	-30–150 °C
Časová konstanta s jímkou	cca 30 s
Kryt	polyamid
Ochranná jímka	poniklovaná mosaz

Tab. 4.7 - Základní parametry teplotního čidla Domat ETF1 [20]



## 5. POUŽITÉ METODY

Tato kapitola popisuje použité metody a algoritmy pro přípravu a zpracování dat a také testované typy neuronových sítí, včetně příslušných učících algoritmů.

### 5.1. Získání a filtrování vstupních dat

Data byla poskytnuta společností Energocentrum Plus, s.r.o., která vyvíjí a provozuje dispečerský a řídicí systém RcWare. Do systému lze přistupovat odkudkoliv přes vizualizační část RcWare Vision a to i pomocí webového prohlížeče. Zde lze nejen získávat historická data, ale pokud má dotyčný dostatečná oprávnění, tak lze plně ovládat zdroj tepla [21].

The screenshot shows the RcWare Vision SCADA interface. On the left is a tree view of the system components, including 'Kotelna K1 nová', 'IPLC DOMAT', and 'Akumulace'. The main window displays a table of events with columns for 'Čas', 'Název', 'Stav', 'Hodnota', and 'Operace'. The table lists various events such as 'Povolení AKU od Venk teploty' and 'Nádrž NABITÉ'.

Čas	Název	Stav	Hodnota	Operace
<b>Akumulace</b>				
21.03.2014 pá	Povolení AKU od Venk teploty		16 °C	
19:20:20				
21.03.2014 pá	Povolení AKU od teploty zpátečk		73 °C	
21:51:38				
09:54:58	Nádrž NABITÉ		NE	
09:54:58	Nádrž VYBITÉ		ANO	
09:54:55	VYBÍJENÍ		NE	
09:54:56	NABÍJENÍ		NE	
28.07.2013 ne	Vynucené MAN NABITÍ		[empty]	
21:23:11	NABITÉ			
10.05.2013 pá	Režim klapky nabíjení AKU		AUT	
08:32:38				
13.05.2014 út	Povel na klapku AKU		0 %	
03:59:37				
08.05.2013 st	Režim klapky výstup KGJ		AUT	
10:41:04				
09:54:55	Poloha klapky výstup KGJ		OTV	
07.05.2013 út	Režim klapky zkrat		ZAV	
07:50:31				
09:54:55	Poloha klapky ZKRAT		ZAV	

Obr. 5.1 - Ukázka SCADA systému RcWare Vision [21]

Vybrané průběhy byly z databáze staženy pomocí modulu RcWare database client do Matlabu, kde i následně probíhalo základní filtrování. Jednotlivé veličiny nejsou všechny snímány ve stejný okamžik, a proto po stažení dat je prvním krokem převzorkování na žádanou vzorkovací frekvenci. V mém případě je nastavena vzorkovací perioda 5 minut. Přepočtení bodu do jiného časového okamžiku je provedeno lineární interpolací. Nejdůležitější částí prvotního filtrování je zpracování průběhů kumulované energie kaskády kotlů a kogenerační jednotky. Při tomto filtrování jsou kontrolovány a odstraněny tři typy chyb objevujících se v datech. Kontrola maximální velikosti skoku pro případ, kdy měřidlo několik hodin nekomunikovalo a potom následně došlo k velkému skoku kumulované energie. Další je odstranění propadů kumulativní hodnoty energie, kdy dochází ke snížení hodnoty, v některých případech až na nulu. Kumulativní hodnota se nemůže nikdy snížit, mezi vzorky může být pouze stejná nebo zvyšující se. Poslední

je kontrola správnosti rozložení vzorků v čase, protože někdy se v datech objeví vzorek, který má přiřazen nižší čas než předchozí. Tyto hodnoty jsou pak odstraněny. V průběhu těchto kontrol jsou všechny označené vzorky nahrazeny hodnotou NaN<sup>7</sup>. V závěru zpracování dat jsou všechny nalezené hodnoty NaN doplněny lineární interpolací (extrapolací v případě krajních hodnot). Takto připravená data jsou následně z Matlabu vyexportována do csv souboru.

## 5.2. Korelační analýza

Na základě korelační analýzy lze zjistit statistickou závislost mezi dvěma proměnnými. Tedy, zda lze pomocí dané množiny vstupních dat stanovit množinu výstupních dat lineárním vztahem. Korelační koeficient určuje stupeň vztahu mezi proměnnými a nabývá hodnot od -1 do +1, které indikují lineární vztah (kladný nebo záporný). V případě neexistence lineárního vztahu je  $r = 0$ . Korelační koeficient, někdy též nazývaný Pearsonův korelační koeficient, poprvé popsal Karl Pearson v roce 1895. Navržený vztah (4.1) je i v dnešní době užívaný v mnoha různých oborech [22].

$$r = \frac{\sum_{k=0}^N (u_k - \bar{u})(y_k - \bar{y})}{\sqrt{\sum_{k=0}^N (u_k - \bar{u})^2 \sum_{k=0}^N (y_k - \bar{y})^2}} \quad (5.1)$$

Dva průběhy proměnných jsou obvykle vzájemně časově posunuté. Tento posun je u aplikací vytápění důležitou charakteristikou. U vytápění je mnoho různých druhů zpoždění, ale nejvýznamnějším z nich je dopravní zpoždění dané vzdáleností a rychlostí teplonosného média [22]. Proto je důležité v rámci korelační analýzy prověřit velikost korelačních koeficientů mezi časově posunutými průběhy veličin. Korelační analýza mezi vzájemně časově posunutými veličinami se nazývá vzájemná korelace. Při analýze vzájemné korelace hledáme maximální hodnotu korelačního koeficientu při daném zpoždění. Za předpokladu, že  $u$  je vstupní veličina a  $y$  výstupní, lze pro kauzální systém předpokládat kladné zpoždění. Kladné zpoždění je nutnou, ovšem ne postačující podmínkou pro kauzalitu systému.

## 5.3. Klouzávy průměr

Pro filtraci a zároveň změnu vzorkování je použita technika klouzavého průměru. Tento způsob využití klouzavého průměru pro zpracování signálu se někdy nazývá Coarse Graining Technique (CGT) [22]. Původní data jsou zpracována se vzorkováním 5 minut, což je pro predikci nevhodné a také objem dat je příliš velký, proto je vhodné snížit vzorkování na 1 hodinu. Toto snížení vzorkování může

<sup>7</sup> Not a Number je v číslicové technice termín pro hodnotu, která nereprezentuje číslo.

být provedeno dvěma způsoby, prvním je prosté vybrání každého  $n$ -tého vzorku bez dalších úprav. Druhým způsobem je vybrání každého  $n$ -tého vzorku a následné určení průměru přes okno o šířce  $n$  vzorků ( 5.2 ) okolo toho vybraného (CGT). V této práci je použit druhý způsob dle rovnice

$$y_j^{CGT} = \frac{1}{\tau} \sum_{k=(j-1)\tau+1}^{j\tau} y_k, \quad 1 \leq j \leq \frac{N}{\tau}. \quad (5.2)$$

Délka zpracovaného průběhu veličiny odpovídá délce původního, vyděleného šířkou okna  $\tau$ . Pro  $\tau = 1$  metoda CGT odpovídá běžnému filtru založeném na klouzavém průměru. Pomocí této metody lze snížit vzorkování zdrojových dat a zároveň provést vyhlazení a redukci šumu při zachování dostatečné „ostrosti“ skokových hran i přes to, že se jedná o velice jednoduchou metodu.

#### 5.4. Normalizace dat

Velmi důležitým krokem, před použitím dat pro neuronovou síť, je normalizace. Jedním z hlavních důvodů je srovnání vlivů jednotlivých vstupů tak, aby vstup, který se pohybuje v řádu jednotek, měl stejnou váhu jako vstup v řádu tisíců. Neuronová síť je schopná se naučit na trénovací množinu i při nenormalizovaných vstupech, ovšem učicí proces je delší a zhorší se i predikční schopnost [24]. Jako nevýhodu lze zmínit nutnost přenosu hodnot směrodatné odchylky a střední hodnoty s modelem, protože při testování je nutné použít stejné hodnoty pro normalizaci jako při trénování.

V této práci je použita metoda Z-score [23], která vyjadřuje vzdálenost daného bodu od střední hodnoty množiny, viz rovnice ( 5.3 ). Tato vzdálenost je vyjádřena násobkem velikosti výběrové směrodatné odchylky.

$$u_k^{NORM} = \frac{u_k - \bar{u}}{\sigma_u \cdot \alpha}, \quad (5.3)$$

kde se obvykle volí parametr  $\alpha = 3$ , protože po provedení této normalizace je rozsah hodnot zhruba od -3 do +3, ale pro neuronovou síť je výhodnější rozsah od -1 do +1. Střední hodnota se získá vydělením součtu všech hodnot počtem hodnot a udává centrální umístění, podle rovnice ( 5.4 ). Směrodatná odchylka udává rozptyl dané množiny dat, podle rovnice ( 5.5 ) [23].

$$\bar{u} = \frac{1}{N} \sum_{k=0}^N u_k. \quad (5.4)$$

$$\sigma_u = \sqrt{\frac{1}{N-1} \sum_{k=0}^N (u_k - \bar{u})^2}. \quad (5.5)$$

### 5.5. Metoda hlavních komponent (PCA)

Předpokládáme, že odhad stavového vektoru systému je funkcí rozptylu dat, z kterých je odhad stavového vektoru složen, pak může být jeho původní dimenze zredukována do prostoru tzv. hlavních komponent (PC), jež je nižší o dimenzi než původní odhad stavového vektoru. Tento prostor hlavních komponent pokrývá většinu rozptylu v datech a také nízká hodnota rozptylu v datech je často považována za šum. Použití nižšího počtu hlavních komponent namísto vysokého počtu původních proměnných je obvyklým postupem v předzpracování dat, který mnohdy zlepšuje výsledky následných analýz pro potřeby klasifikace nebo predikce [25].

V této práci je metoda PCA použita ve dvou modifikacích na vektor zpožděných a budoucích hodnot venkovní teploty vzhledem k aktuálně predikovanému vzorku. První je klasické lineární PCA, kde je dimenze vstupního vektoru snížena přibližně na polovinu. Nevýhodou lineárního PCA je, že při redukci nebere v potaz závislosti vyšších řádů, proto je v druhé modifikaci provedeno rozšíření vstupního vektoru polynomem druhého stupně pro zachycení nelineárních závislostí. Tento rozšířený vektor je následně redukován stejným způsobem jako v předchozím případě.

Vstupní vektor ( 5.6 ) je tedy tvořen budoucími (počet dán  $n_1$ ) a minulými ( $n_2$ ) hodnotami vstupní veličiny s konstantním vzorkováním  $k$ .

$$\mathbf{x}(k) = \begin{bmatrix} u_1(k + n_1) \\ \vdots \\ u_1(k) \\ \vdots \\ u_1(k - n_2) \end{bmatrix}. \quad (5.6)$$

Na základě vstupních vektorů je vytvořena vstupní matice ( 5.7 ), která obsahuje kompletní dostupná data, tedy například 1 rok se vzorkováním 1 hodina odpovídá 8760 vstupním vektorům.

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}(k=0)^T \\ \vdots \\ \mathbf{x}(k=N)^T \end{bmatrix}. \quad (5.7)$$

Pro určení míry vzájemné vazby mezi veličinami je použita kovarianční matice ( 5.8 ) určená jako

$$\mathbf{C}_x = (\mathbf{X}\mathbf{X}^T)N^{-1}. \quad (5.8)$$

Následným krokem je nalezení vlastních vektorů  $\mathbf{e}_i$  a vlastních čísel  $\lambda_i$  kovarianční matice. Vlastní vektory jsou následně seřazeny podle odpovídajících

vlastních čísel od největšího po nejmenší. První vlastní vektor udává směr největšího rozptylu dat, druhý vlastní vektor udává směr největšího rozptylu dat kolmý na směr prvního vlastního vektoru [25]. Pro redukci dimenze vstupního vektoru  $\mathbf{x}$  již stačí vybrat pouze určitý počet největších vlastních vektorů a sestavit transformační matici  $\mathbf{P}$  ( 5.9 ).

$$\mathbf{P} = \begin{bmatrix} \mathbf{e}_0 \\ \vdots \\ \mathbf{e}_n \end{bmatrix}. \quad ( 5.9 )$$

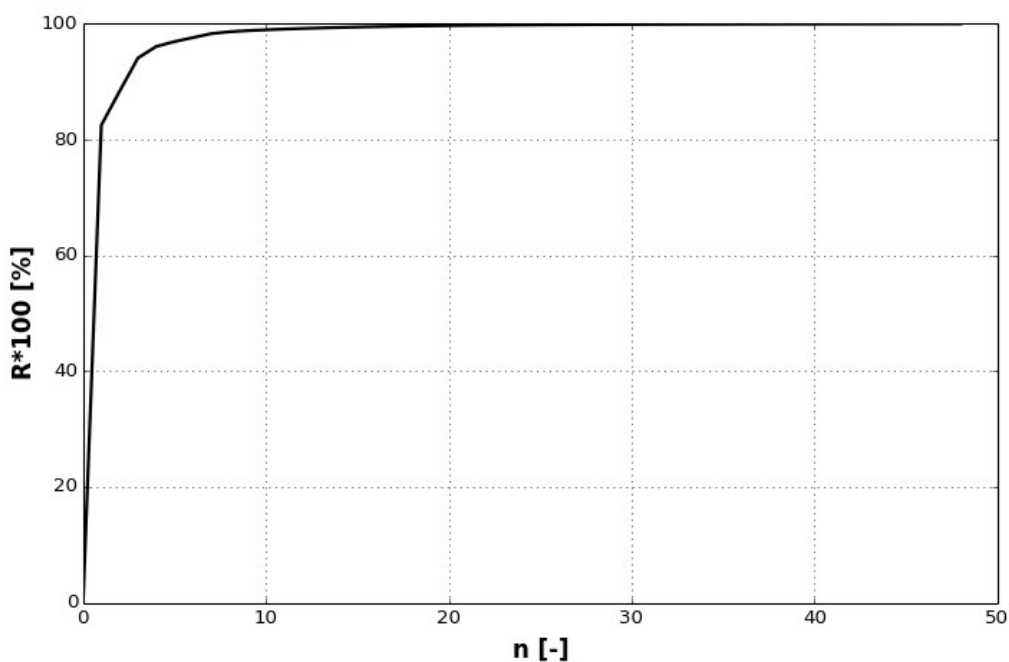
Kde  $n$  je požadovaná velikost redukováného vstupního vektoru  $\mathbf{x}$ . Redukovaná matice  $\mathbf{X}_{PCA}$  ( 5.10 ) vstupních vektorů vznikne jako lineární transformace původní matice  $\mathbf{X}$  přes transformační matici  $\mathbf{P}$ .

$$\mathbf{X}_{PCA} = \mathbf{P}\mathbf{X}. \quad ( 5.10 )$$

Pro určení vhodné dimenze redukce lze použít poměr  $R$  ( 5.11 ), jedná se o poměr součtu seřazených vlastních čísel odpovídajících vybraným  $n$  vlastním vektorům k součtu všech vlastních čísel, kde  $m$  je původní dimenze [25].

$$R = \frac{\sum_{i=0}^n \lambda_i}{\sum_{i=0}^m \lambda_i}. \quad ( 5.11 )$$

Například pro vstupní vektor sestavený z venkovních teplot ( $n_1=24$ ,  $n_2=24$ ), to je dimenze 50. Na Obr. 5.2 je průběh poměru  $R$ . Pro redukovanou dimenzi  $n=10$  je zachováno 99 % původního rozptylu dat.



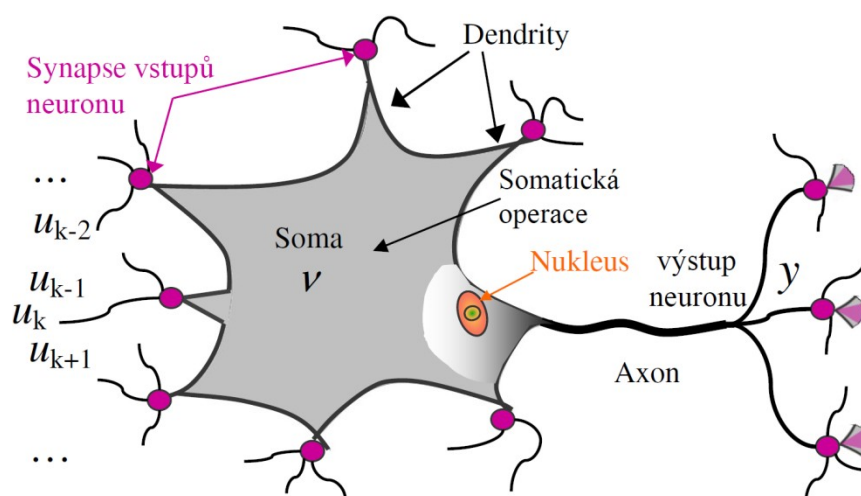
Obr. 5.2 - Příklad průběhu  $R$  pro vstupní vektor o původní dimenzi  $m=50$

## 5.6. Použité neuronové modely

V této práci je použito a porovnáno několik různých neuronových struktur, počínaje nejjednodušší lineární neuronovou jednotkou (LNU), přes modely HONU<sup>8</sup> (kvadratické a kubické neuronové jednotky) a klasickou neuronovou sítí typu MLP až po neuronovou síť RBF<sup>9</sup>. Všechny tyto neuronové struktury jsou implementovány ve své statické podobě.

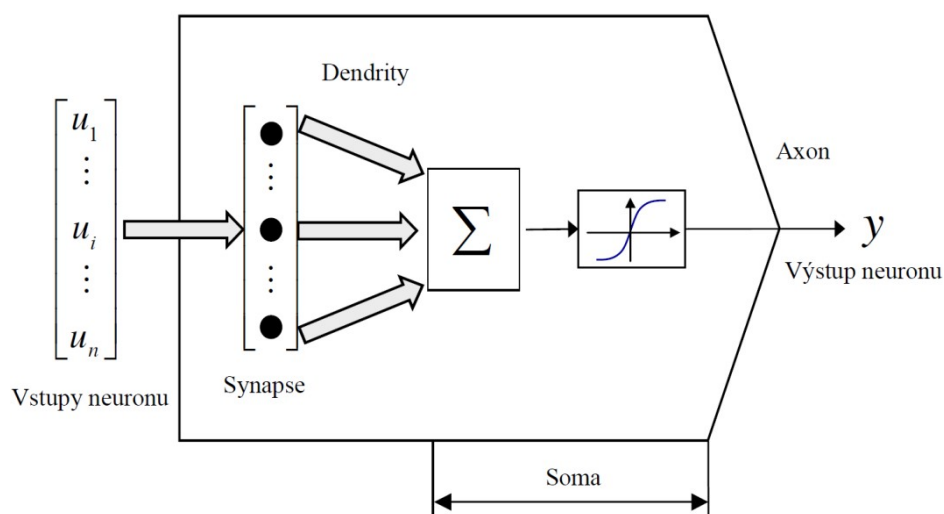
### Model formálního neuronu

Základním stavebním kamenem centrálního nervového systému, včetně mozku, sítnice i míchy, je neuron. Tato buňka přijímá a zpracovává informace a následně komunikuje s různými částmi těla. Ideální biologický neuron je zobrazen na Obr. 5.3.



Obr. 5.3 - Základní struktura biologického neuronu [26]

Tělo nervové buňky se nazývá *soma* a je obklopeno tenkou membránou naplněnou cytoplasmou. V těle neuronu je jádro (nukleus), které obsahuje DNA buňky.



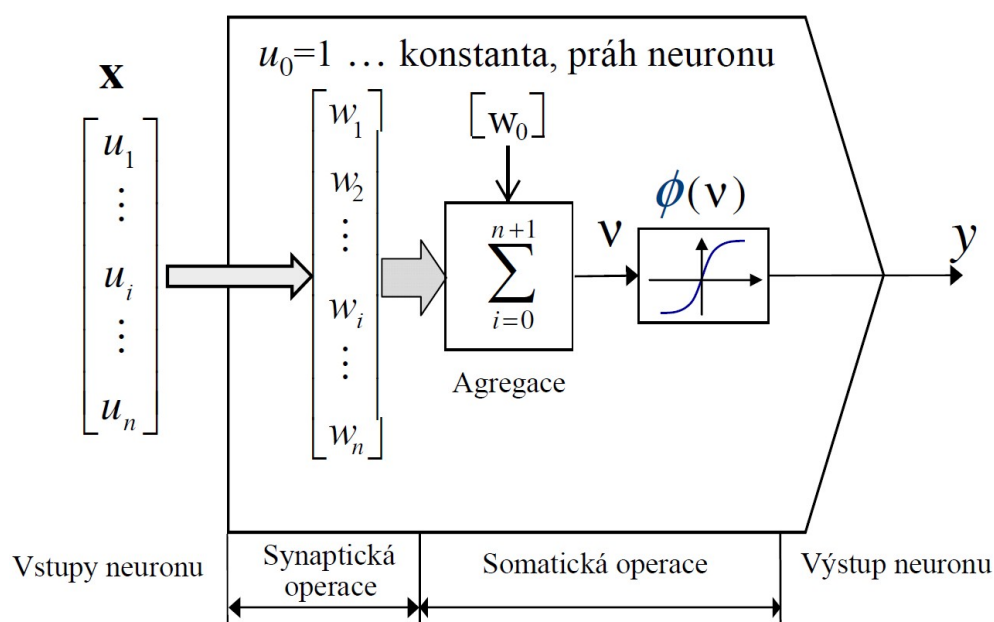
Obr. 5.4 - Jednoduchý model neuronu s více vstupy (dendrity) a jedním výstupem (axon) [27]

<sup>8</sup> HONU - Higher-Order Neural Unit, neuronová jednotka s nelinearitami vyšších řádů

<sup>9</sup> RBF - Radial Basis Function, neuronová síť na základě radiálně bázových funkcí

Nalezení matematického popisu jediného neuronu je prvním krokem k návrhu komplexních neuronových sítí, které lze využít například pro zpracování signálu, rozpoznávání vzorů, řízení složitých procesů, strojové vidění apod. Jednoduchý model neuronu je na Obr. 5.4. Z pohledu zpracování informace může být jeden neuron s dendrity jako terminály pro více vstupů a axonem jako terminálem pro jeden výstup považován za MISO<sup>10</sup> systém. Funkce jednotlivých částí neuronu pro zpracování informace lze rozdělit do následujících čtyř kategorií [27]:

- *dendrity*: přijímají vstupní informaci, skládají se z větvičích se vláken,
- *synapse*: místo pro ukládání zkušeností neuronu, poskytuje dlouhodobou paměť, přijímá informace ze senzorů i dalších neuronů,
- *soma*: tělo neuronové buňky, přijímá informace ze synapsí a provádí jejich další zpracování; téměř všechny logické funkce neuronu se odehrávají zde,
- *axon*: výstupní část, výstup je ve formě elektrického nervového vzruchu.



Obr. 5.5 - Analogie mezi biologickým a umělým neuronem [27]

Z matematického pohledu zpracování informace v neuronu zahrnuje dvě následující matematické operace [27]:

- *synaptická operace*: síla (váha) synapse reprezentuje paměť, synaptická operace určuje váhu každého přichozícího signálu na základě předchozích znalostí,
- *somatická operace*: provádí několik matematických operací, například agregaci, prahovou funkci, nelineární aktivaci a dynamické zpracování synapsí.

Zjednodušená reprezentace operací typického neuronu je na Obr. 5.5. Biologické neurony mají zajímavé matematické vlastnosti, díky jejich nelineární povaze a prahu neuronu. Pokud by byly neurony pouze lineární, rozpoznávací schopnosti lidí a robustnost neuronových systémů by zanikla.

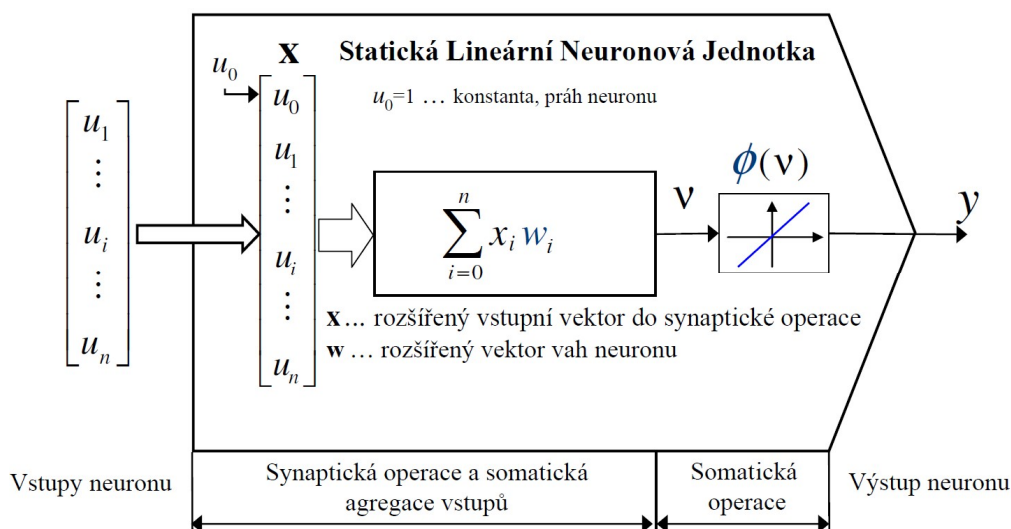
<sup>10</sup> MISO - Multiple-Input / Single-Output, více vstupů / jeden výstup

### Lineární neuronová jednotka (LNU)

Lineární neuronová jednotka je založena na modelu perceptronu, který navrhl v roce 1957 Frank Rosenbatt [28]. Stejně jako perceptron má LNU lineárně váženou agregační funkci, ale místo skokové aktivační funkce (výstup 0 nebo 1) má lineární aktivační funkci, viz Obr. 5.6. Matematicky lze vyjádřit lineární neuronovou jednotku v polynomiální formě ( 5.12 ).

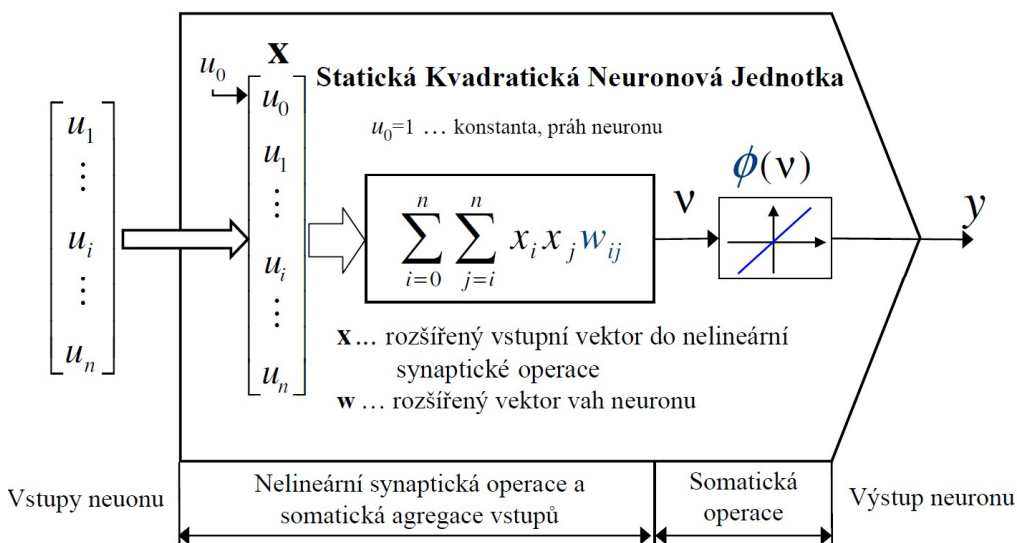
$$y = \sum_{i=0}^n x_i w_i = w_0 + w_1 x_1 + \dots + w_n x_n = [w_0 \ w_1 \dots w_n] \cdot \begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_n \end{bmatrix} = \mathbf{w} \cdot \mathbf{X}, \quad (5.12)$$

kde  $\mathbf{w}$  je vektor vah neuronu, které je nutné najít pomocí učícího algoritmu a  $\mathbf{x}$  je vektor vnějších vstupů, pro případ statického modelu.



Obr. 5.6 - Statická lineární neuronová jednotka (LNU) [26]

### Neuronové jednotky vyšších řádů (QNU, CNU)



Obr. 5.7 - Statická kvadratická neuronová jednotka (QNU) [26]



V této práci je použita kvadratická (Obr. 5.7) a kubická (Obr. 5.8) neuronová jednotka. Koncept nelineárních neuronových jednotek vyšších řádů poprvé představil v roce 2003 Madan M. Gupta [27]. Jednotky QNU i CNU sdílí koncept s jednotkou LNU, s tím rozdílem, že jsou již schopné lepší aproximace nelineárních dějů, díky polynomům vyšších řádů. Další výhodou těchto jednotek je schopnost naučit se žádaný děj s menší odchylkou než LNU nebo MLP síť (při srovnatelných parametrech sítě). Při použití pokročilých učících algoritmů jednotky HONU netrpí problémem lokálních minim chybové funkce (učení skončí ve falešném/lokálním minimu). Matematická struktura HONU s polynomiální nelinearitou stupně  $r$  může být s výhodou zapsána v dlouhé vektorové formě [29]

$$y = \sum w_{i,j,\dots,r} \cdot x_i \cdot x_j \dots x_r \quad kde \quad i = 0..n, j = i..n, \quad (5.13)$$

což lze také zapsat vektorovým součinem jako

$$y = \mathbf{w} \cdot \mathbf{x}^T, \quad (5.14)$$

kde  $y$  je výstup modelu a dlouhý řádkový vektor všech vah je

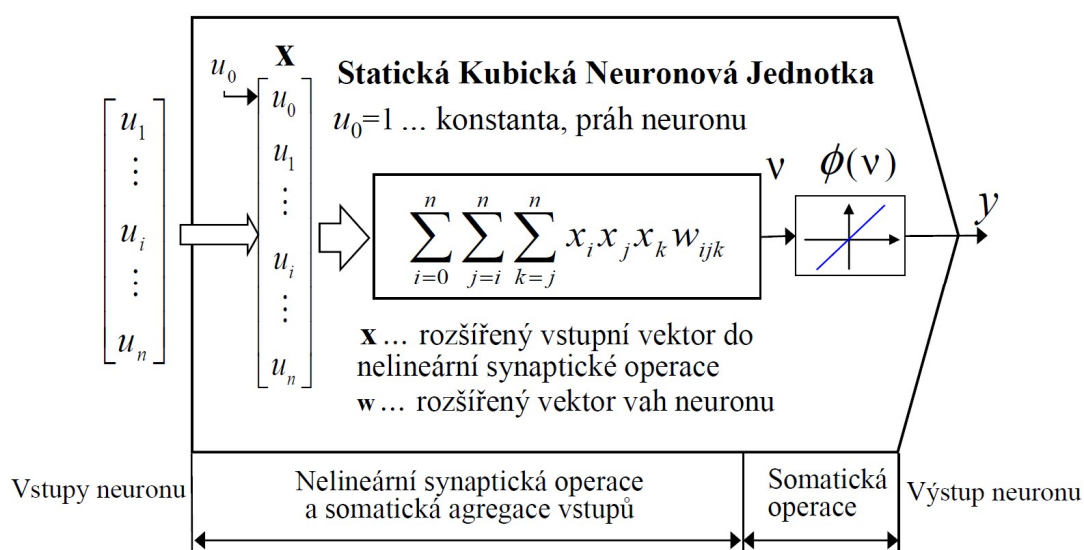
$$\mathbf{w} = [w_{i,j,\dots,r}] \quad kde \quad i = 0..n, j = i..n, \dots, \quad (5.15)$$

a dlouhý vektor polynomiálních členů je

$$\mathbf{x} = [x_i \cdot x_j \cdot \dots \cdot x_r] \quad kde \quad i = 0..n, j = i..n, \dots, \quad (5.16)$$

přičemž vstupní vektor neuronové jednotky HONU je definován jako

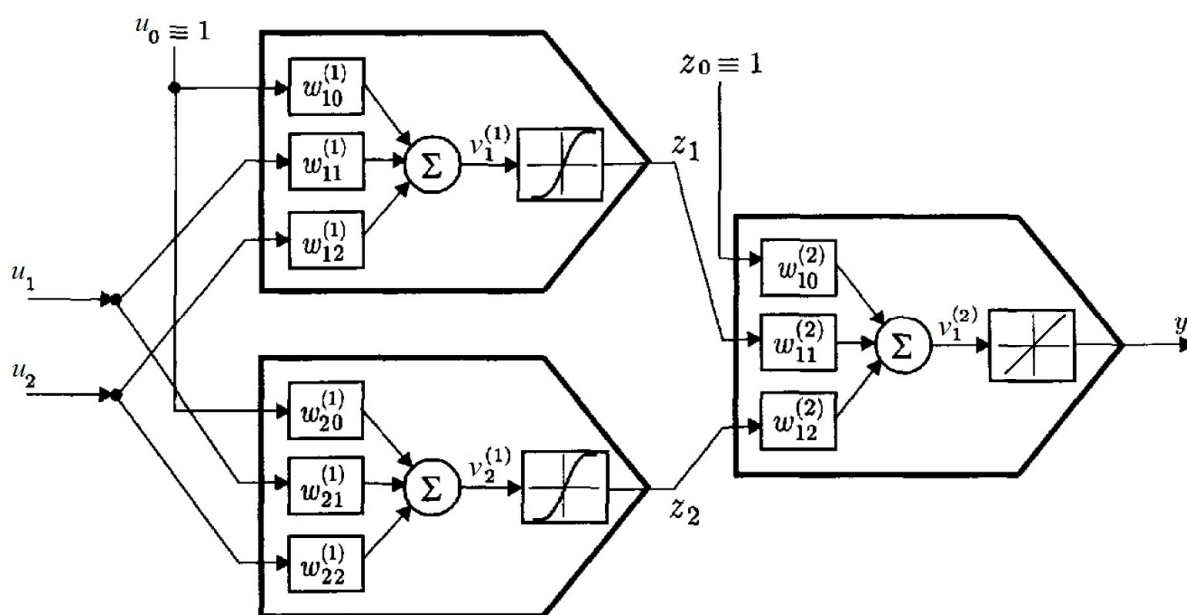
$$\mathbf{x} = \begin{bmatrix} u_0 = 1 \\ u_1 \\ u_2 \\ \vdots \\ u_n \end{bmatrix}. \quad (5.17)$$



Obr. 5.8 - Statická kubická neuronová jednotka (CNU) [26]

### Vícevrstvá neuronová síť (MLP)

Pro vytvoření modelu byly zvoleny dopředné vícevrstvé neuronové sítě. Typicky se síť skládá ze „senzorů“, které tvoří vstupní vrstvu (vektor), z jedné nebo více skrytých vrstev a výstupní vrstvy (Obr. 5.9). Vstupy prostupují sítí vždy dopředu, vrstva po vrstvě, proto se síť nazývá dopředná síť. Obecně každý neuron agreguje svoje vážené vstupy a vrací výstup přes nelineární aktivační funkci (typicky sigmoida). Síť MLP se dá popsat jako síť složená z jednotek LNU s nelineární aktivační funkcí. Neurony ve výstupní vrstvě obvykle mají lineární aktivační funkci, viz Obr. 5.9.



Obr. 5.9 - Nejjednodušší případ sítě MLP, jedna skrytá vrstva s dvěma neurony [27]

Nelineární výstupní funkcí je nejčastěji logistická funkce (sigmoida), její důležitou vlastností je její hladkost v celém intervalu (je diferencovatelná), na rozdíl od skokové funkce u jednoduchého perceptronu. Pro data normalizovaná v rozsahu -1 až 1, je vhodné upravit aktivační funkci na tento rozsah a tedy

$$\Phi(v) = \frac{2}{1 + e^{-v}} - 1. \quad (5.18)$$

Každý neuron ve skryté vrstvě agreguje svoje vážené vstupy a vrací výstup přes aktivační funkci. Lze zapsat jako

$$z_i = \Phi(v_i) = \Phi(\mathbf{w}_i^{(1)} \cdot \mathbf{x}). \quad (5.19)$$

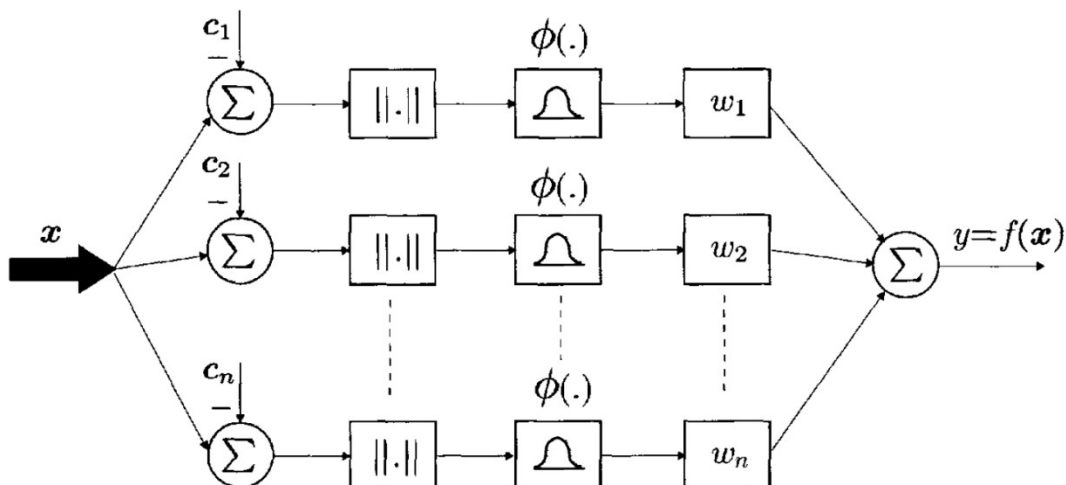
Pro výstupní neuron s lineární aktivační funkcí jsou vstupem výstupy neuronů ze skryté vrstvy a tedy

$$y = \mathbf{w}^{(2)} \cdot \mathbf{z} = \mathbf{w}^{(2)} \cdot \Phi(\mathbf{v}) = \mathbf{w}^{(2)} \cdot \Phi(\mathbf{W}^{(1)} \cdot \mathbf{x}), \quad (5.20)$$

kde  $\mathbf{w}^{(2)}$  je vektor vah výstupního neuronu a  $\mathbf{W}^{(1)}$  je matice vah všech neuronů ve skryté vrstvě.

### Sít' založená na radiálně bázových funkcích (RBF)

Posledním typem modelu je sít' založená na radiálně bázových funkcích, která řeší problém aproximace úplně jiným způsobem než předchozí. U této sítě je ekvivalentem učení hledání plochy ve vícedimenzionálním prostoru, která poskytuje nejlepší aproximaci trénovacích dat s co nejlepší schopností zobecňovat a tedy predikovat budoucí průběh veličiny. Jednou z prvních prací na toto téma zpracoval Powell v roce 1985 [30]. RBF sít' se díky svým zajímavým aproximačním schopnostem dostaly do hledáčku vědců po celém světě.



Obr. 5.10 - Neuronová sít' založená na radiálně bázových funkcích [27]

RBF sít' se skládá ze tří základních vrstev (Obr. 5.10). Vstupní vrstva je, stejně jako v předešlých případech, složena z jednotlivých vstupních veličin. Druhá vrstva je skrytou vrstvou, která je zde vždy pouze jedna a provádí nelineární transformaci ze vstupního prostoru do skrytého prostoru, který bývá vícedimenzionální. Čím větší je dimenze skryté vrstvy, tím lepší aproximaci sít' poskytuje. Výstupní vrstva je lineární, lze si ji představit jako jednotku LNU. Nejčastěji používanou radiálně bázovou funkcí je Gaussova funkce

$$\Phi(d) = e^{-\frac{1}{2}\left(\frac{d}{\beta}\right)^2}, \quad (5.21)$$

kde je nutné najít vhodný parametr  $\beta$ , který určuje šířku funkce a tím i překrytí funkcí ve skryté vrstvě. Váhy výstupního neuronu závisí na vzdálenosti mezi vstupním a středním vektorem [27], viz rovnice ( 5.22 ).

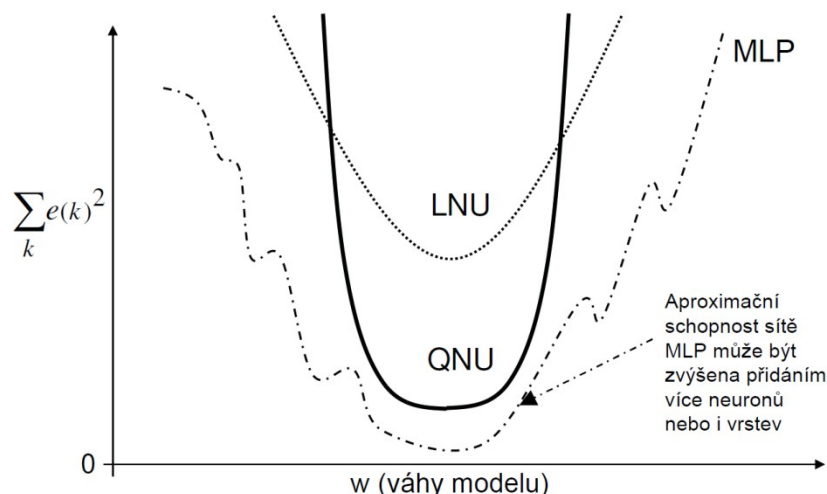
$$y \triangleq f(\mathbf{x}) = \sum_{i=0}^n w_i \Phi(d_i) = \sum_{i=0}^n w_i \Phi(\|\mathbf{x} - \mathbf{c}_i\|). \quad (5.22)$$

Vzdálenost vektorů  $d_i$  je určena jako Euklidovská vzdálenost, vektor  $\mathbf{c}_i$  představuje vektor center radiálně bázových funkcí. V případě velkého množství trénovacích dat je nutné nalézt vhodná centra reprezentující data, ovšem v případě menšího množství dat lze jako centra použít přímo trénovací data.

## 5.7. Učící algoritmy

Aby bylo možné neuronové modely používat, je nutné je nejdříve naučit závislosti vstupů na výstupu pomocí trénovacích dat. V této práci je využita metoda klesajícího gradientu (dále Gradient descent, GD) pro učení modelu LNU. Dále algoritmus Levenberg-Marquardt (LM) pro učení modelů QNU a CNU a pro síť MLP je použita metoda ARPROP<sup>11</sup>.

Obr. 5.11 názorně vystihuje porovnání chování různých neuronových modelů v průběhu učení. Lineární model lze hůře natrénovat, převážně díky neschopnosti postihnout nelineární vztahy, ovšem má vždy pouze jedno řešení a tedy netrpí problémem lokálních minim. Běžná síť MLP (učená algoritmem backpropagation) je schopná se naučit daný problém velice přesně, ovšem s rizikem přeučení sítě (zhoršení predikčních schopností modelu) a trpí problémem lokálních minim. Modely HONU se učí velmi rychle při zachování přesnosti aproximace, netrpí problémem lokálních minim [31].



Obr. 5.11 - Porovnání průběhů chybových funkcí různých modelů [31]

### Gradient descent (GD)

Na tomto optimalizačním algoritmu je založen i algoritmus backpropagation pro síť MLP, ovšem v této práci je použit pouze pro model LNU. Metoda GD [2] je založena na rovnici

$$w_i(k+1) = w_i(k) - \mu \frac{\partial E}{\partial w_i}, \quad (5.23)$$

kde  $E$  je chybová funkce a  $w_i$  váha modelu. Chybová funkce je záměrně definována jako

$$E = \frac{1}{2}(y_R - y)^2, \quad (5.24)$$

kde  $y_R$  je skutečný, naměřený výstup z procesu.

<sup>11</sup> ARPROP - Advanced Resilient Propagation, zdokonalený algoritmus backpropagation

Na základě řetězového pravidla lze parciální derivaci chybové funkce rozepsat

$$\nabla E = \frac{\partial E}{\partial w_i} = \frac{dE}{dy} \frac{dy}{dv} \frac{\partial v}{\partial w_i}, \quad (5.25)$$

kde  $v$  je výstup z agregační funkce, ještě před aktivační funkcí. V případě lineárního neuronu lze jednoduše derivovat jako

$$\nabla E = \frac{\partial E}{\partial w_i} = (y - y_R)(1)(x_i). \quad (5.26)$$

Proto pro případ LNU je pravidlo pro aktualizaci vah následující

$$w_i(k+1) = w_i(k) + \mu \cdot (y_R - y) \cdot x_i = w_i(k) + \mu \cdot e \cdot x_i, \quad (5.27)$$

Výhodami GD jsou hlavně velice jednoduchá implementace a rychlost výpočtu, díky které lze algoritmus využít pro rozsáhlé množiny dat. Nevýhodou je v případě sítě MLP pomalá a nezaručená konvergence, nevyužívání redundance v datech a vykazuje problém lokálních minim [27][31].

### Levenberg–Marquardt (LM)

LM algoritmus byl vyvinut nezávisle na sobě Levenbergem [32] a Marquardtem [33] pro řešení minimalizace nelineární funkce. Hlavním rozdílem oproti metodě GD je, že LM počítá aktualizace vah vždy přes celá trénovací data, kdy GD počítá aktualizace pro každý vzorek. Jedná se vlastně o kombinaci dvou základních metod, algoritmus nejdříve postupuje jako GD a při přiblížení se minimu začíná převažovat Gauss-Newtonova metoda [34], která značně urychlí konvergenci v blízkosti minima. Vzhledem k tomu že LM počítá aktualizace vah vždy přes celá data, je nutné upravit i chybovou funkci

$$E = \frac{1}{2} \sum_{k=0}^N e^2 = \frac{1}{2} \sum_{k=0}^N (y_R - y)^2. \quad (5.28)$$

LM algoritmus vychází z GD a tedy

$$w_i(k+1) = w_i(k) - \mu \frac{\partial E}{\partial w_i} = w_i(k) - \mu g_i, \quad (5.29)$$

kde  $g_i$  je gradient chybové funkce. Dále je využita Newtonova iterační metoda, která předpokládá, že gradient  $g_i$  je funkcí vah, které jsou lineárně nezávislé a tedy

$$g_i \approx g_{i,0} + \frac{\partial g_i}{\partial w_1} \Delta w_1 + \frac{\partial g_i}{\partial w_2} \Delta w_2 + \dots + \frac{\partial g_i}{\partial w_n} \Delta w_n. \quad (5.30)$$

Nyní lze do rovnice (5.30) dosadit definici gradientu z (5.29), a protože je cílem minimum, kde je gradient nulový, bude levá strana rovnice rovna nule

$$0 \approx g_{i,0} + \frac{\partial^2 E}{\partial w_i \partial w_1} \Delta w_1 + \frac{\partial^2 E}{\partial w_i \partial w_2} \Delta w_2 + \dots + \frac{\partial^2 E}{\partial w_i^2} \Delta w_n. \quad (5.31)$$

Za zbývající člen  $g_{i,0}$  lze opět dosadit z ( 5.29 ) a vše přepsat do maticové formy

$$\begin{bmatrix} -g_1 \\ -g_2 \\ \vdots \\ -g_n \end{bmatrix} = \begin{bmatrix} -\frac{\partial E}{\partial w_1} \\ -\frac{\partial E}{\partial w_2} \\ \vdots \\ -\frac{\partial E}{\partial w_n} \end{bmatrix} = \begin{bmatrix} \frac{\partial^2 E}{\partial w_1^2} & \frac{\partial^2 E}{\partial w_1 \partial w_2} & \dots & \frac{\partial^2 E}{\partial w_1 \partial w_n} \\ \frac{\partial^2 E}{\partial w_2 \partial w_1} & \frac{\partial^2 E}{\partial w_2^2} & \dots & \frac{\partial^2 E}{\partial w_2 \partial w_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 E}{\partial w_n \partial w_1} & \frac{\partial^2 E}{\partial w_n \partial w_2} & \dots & \frac{\partial^2 E}{\partial w_n^2} \end{bmatrix} \times \begin{bmatrix} \Delta w_1 \\ \Delta w_2 \\ \vdots \\ \Delta w_n \end{bmatrix}, \quad (5.32)$$

kde čtvercová matice druhých partiálních derivací chybové funkce je Hessova matice  $\mathbf{H}$ . Potom lze kombinací rovnic ( 5.29 ) a ( 5.32 ) přepsat jako pravidlo pro aktualizaci vah následovně do vektorové formy

$$-\mathbf{g} = \mathbf{H} \Delta \mathbf{w}, \quad (5.33)$$

$$\mathbf{w}(k+1) = \mathbf{w}(k) - \mathbf{H}^{-1} \mathbf{g}, \quad (5.34)$$

kde  $p$  je číslo iterace (epochy) učícího procesu. Gauss-Newtonův algoritmus je využit pro zjednodušení výpočtu Hessovy matice za pomoci Jacobiho matice (matice prvních partiálních derivací chyby  $e$ ). Integrací rovnic ( 5.28 ) a ( 5.29 ) lze získat

$$g_i = \frac{\partial E}{\partial w_i} = \frac{\partial \left( \frac{1}{2} \sum_{k=0}^N e^2 \right)}{\partial w_i} = \sum_{k=0}^N \left( \frac{\partial e}{\partial w_i} e \right), \quad (5.35)$$

kde lze již partiální derivace nahradit Jacobiho maticí a ve vektorové formě

$$\mathbf{g} = \mathbf{J} \mathbf{e}, \quad (5.36)$$

kde chybový vektor  $\mathbf{e}$  je sloupcový vektor. Pro výpočet členů Hessovy matice lze opět dosadit z ( 5.28 ) a získat

$$h_{i,j} = \frac{\partial^2 E}{\partial w_i \partial w_j} = \frac{\partial^2 \left( \frac{1}{2} \sum_{k=0}^N e^2 \right)}{\partial w_i \partial w_j} = \sum_{k=0}^N \frac{\partial e}{\partial w_i} \frac{\partial e}{\partial w_j} + S_{i,j}. \quad (5.37)$$

Základním předpokladem Newtonovy metody je, že člen  $S_{i,j}$  je téměř nulový a lze ho zanedbat. Potom je možné napsat, že vztah mezi  $\mathbf{H}$  a  $\mathbf{J}$  je

$$\mathbf{H} \approx \mathbf{J}^T \mathbf{J}. \quad (5.38)$$

Protože pro složitější chybové funkce nemusí existovat inverzní matice pro člen  $\mathbf{J}^T \mathbf{J}$ , LM navrhli aproximaci Hessovi matice následovně

$$\mathbf{H} \approx \mathbf{J}^T \mathbf{J} + \frac{1}{\mu} \mathbf{I}, \quad (5.39)$$

kde  $\mathbf{I}$  je jednotková matice a  $\mu$  koeficient rychlosti učení.

Marquardt [33] ještě zavedl nahrazení jednotkové matice pro zrychlení konvergence pro malý gradient (blízko minima)

$$\mathbf{H} \approx \mathbf{J}^T \mathbf{J} + \frac{1}{\mu} \text{diag}(\mathbf{J}^T \mathbf{J}). \quad (5.40)$$

Dosažením rovnic (5.40) a (5.36) do (5.34) získáme pravidlo pro aktualizaci vah dle LM

$$\mathbf{w}(k+1) = \mathbf{w}(k) - \frac{\mathbf{J}e}{\mathbf{J}^T \mathbf{J} + \frac{1}{\mu} \text{diag}(\mathbf{J}^T \mathbf{J})} = \mathbf{w}(k) - \frac{\mathbf{J}e}{\mathbf{H}}. \quad (5.41)$$

Jedná se o rychle a stabilně konvergující algoritmus, který je vhodný spíše pro malé a střední množiny dat [35].

### Advanced Resilient Propagation (ARPROP)

Advanced Resilient Propagation je heuristický učící algoritmus. Jeho první verze byla představena Riedmillerem a Braunem v roce 1993 [36]. Základní myšlenkou algoritmu je pracovat pouze se znaménkem (směrem) gradientu chybové funkce, ne s jeho velikostí. Rozdílný je také přístup ke každé váze individuálně. V této práci je implementována zdokonalená verze ARPROP [37]. Prvním krokem je určení velikosti přírůstku vah na základě znamének parciálních derivací

$$\Delta_{ij}(k) = \begin{cases} \eta^+ \cdot \Delta_{ij}(k-1) & , \text{když } \frac{\partial E(k-1)}{\partial w_{ij}} \cdot \frac{\partial E(k)}{\partial w_{ij}} > 0 \\ \eta^- \cdot \Delta_{ij}(k-1) & , \text{když } \frac{\partial E(k-1)}{\partial w_{ij}} \cdot \frac{\partial E(k)}{\partial w_{ij}} < 0, \\ \Delta_{ij}(k-1) & , \text{když } \frac{\partial E(k-1)}{\partial w_{ij}} \cdot \frac{\partial E(k)}{\partial w_{ij}} = 0 \end{cases} \quad (5.42)$$

kde  $0 < \eta^- < 1 < \eta^+$ ,  $\Delta_{ij}(k)$  je velikost přírůstku váhy v aktuálním kroku, který se určuje na základě přírůstku z minulého kroku (epochy) a změny znaménka parciálních derivací. Nyní je třeba rozhodnout o směru přírůstku

$$\Delta w_{ij}(k) = \begin{cases} -\Delta_{ij}(k) & , \text{když } \frac{\partial E(k)}{\partial w_{ij}} > 0 \\ +\Delta_{ij}(k) & , \text{když } \frac{\partial E(k)}{\partial w_{ij}} < 0. \\ 0 & , \text{když } \frac{\partial E(k)}{\partial w_{ij}} = 0 \end{cases} \quad (5.43)$$

V původním algoritmu v případě změny znaménka parciálních derivací mezi minulým a aktuálním krokem se nejen zmenší přírůstek, ale také se vrátí zpět poslední změna váhy. Ve zdokonaleném algoritmu [37] se nebere v úvahu jen změna

znaménka (vázána na danou váhu), ale také více globální informace o změně chyby učení celé sítě. Algoritmus dále postupuje metodou půlení intervalu (namísto vrácení změny váhy zpět), kdy je znám interval mezi aktuálním a posledním krokem, kde se pravděpodobně nachází minimum. Díky tomuto postupu je konvergence v oblasti minima rychlejší než u původního algoritmu. V případě zvýšení chyby celé sítě se nová váha počítá jako

$$w_{ij}(k+1) = w_{ij}(k) - \frac{1}{2^q} \Delta w_{ij}(k-1), \quad (5.44)$$

kde  $q$  je redukční parametr, který má vliv na počet potřebných iterací, většinou se volí roven jedné. Díky své rychlosti (nevyžaduje derivace druhého řádu) se jedná o algoritmus vhodný i pro učení na rozsáhlejších množinách dat.

## 5.8. Hodnotící kritéria

Pro hodnocení schopnosti modelů naučit se předložené trénovací vzory a také pro hodnocení kvality predikce již naučených modelů, jsou v této práci použita dvě kritéria.

Prvním je odmocnina z průměru kvadrátu chyby modelu, jinak Root Mean Squared Error neboli RMSE, která se určí podle rovnice ( 5.45 ).

$$RMSE = \sqrt{\frac{1}{N} \sum_{k=0}^N (y_R - y)^2}, \quad (5.45)$$

Vzhledem k tomu, že kritérium je závislé na měřítku a jednotkách hodnocené veličiny, je vhodné pro porovnávání různých modelů stejné veličiny, ale už ne pro různé veličiny nebo v různých jednotkách.

Druhým je koeficient determinace označovaný  $R^2$ , viz rovnice ( 5.46 ), který nehodnotí velikost chyby, ale kvalitu modelu. Udává kolik procent rozptylu vysvětlované proměnné je vysvětleno modelem a kolik zůstalo nevysvětleno. V případě mnohorozměrného normálního rozdělení je  $R^2$  konzistentním odhadem čtverce koeficientu mnohonásobné korelace, který je roven Pearsonovu koeficientu korelace mezi měřenou veličinou  $y_R$  a její predikcí  $y$  [38]

$$R^2 = r^2 = \frac{\left( \sum_{k=0}^N (y_{R,k} - \bar{y}_R)(y_k - \bar{y}) \right)^2}{\sum_{k=0}^N (y_{R,k} - \bar{y}_R)^2 \sum_{k=0}^N (y_k - \bar{y})^2}. \quad (5.46)$$

Koeficient determinace nabývá hodnot od nuly do jedné (teoreticky včetně těchto krajních mezí), přičemž hodnoty blízké nule značí špatnou kvalitu modelu a hodnoty blízké jedné značí dobrou kvalitu. Udává se většinou v procentech. Obecně hodnoty nad 70 % značí velice dobrý model.

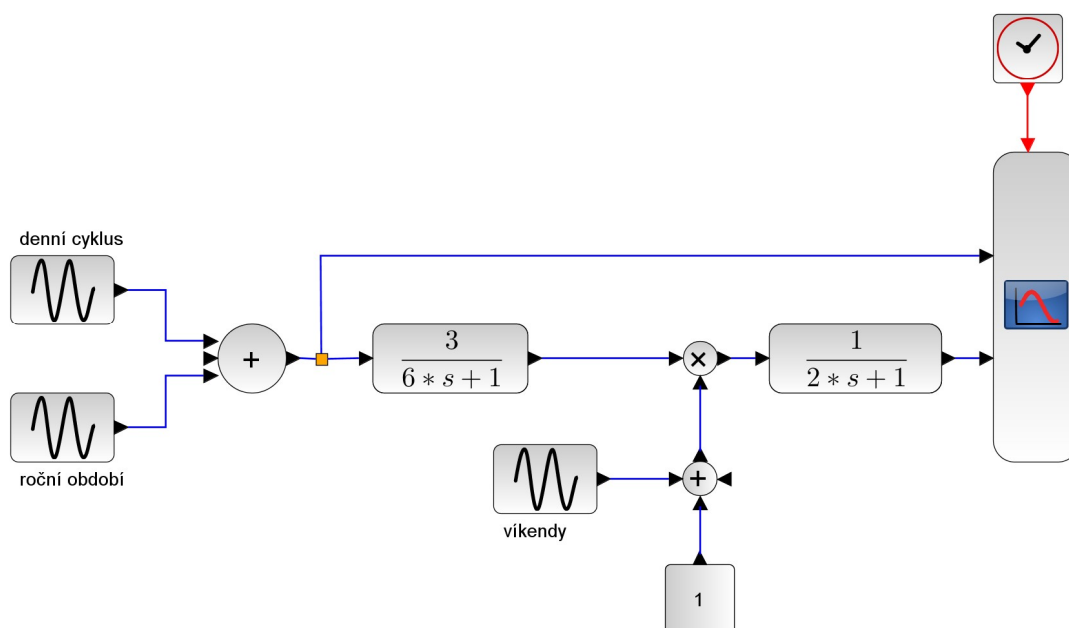


## 6. OVĚŘENÍ FUNKČNOSTI ALGORITMŮ

Před použitím naprogramovaných algoritmů z předchozí kapitoly je nutné ověřit jejich funkčnost a správnost na známém systému. V práci jsou použity celkem čtyři kombinace neuronových modelů a příslušných učících algoritmů. Jedná se o učící algoritmus GD pro LNU, LM pro QNU a CNU a algoritmus ARPROP pro síť MLP.

### 6.1. Simulační model

Ve volně šiřitelném programu pro numerické výpočty Scilab byl vytvořen jednoduchý simulační model (Obr. 6.1) pro testování algoritmů.

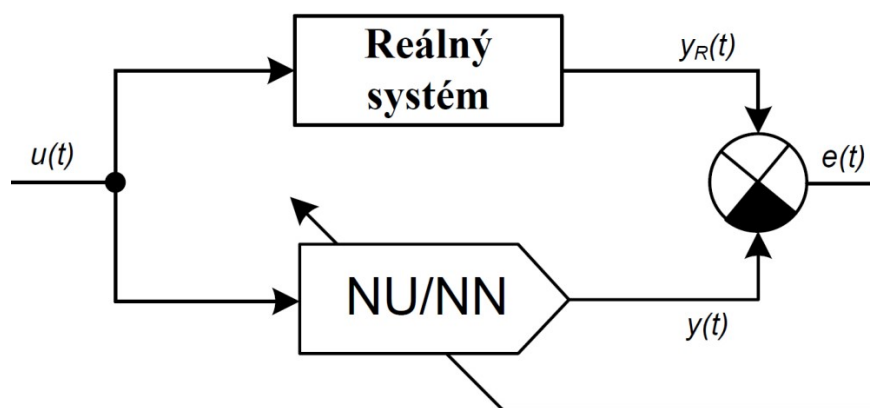


Obr. 6.1 - Simulační model vytvořený v programu Scilab

Jedná se o systém dvou soustav prvního řádu. Vstupem je teplota ve formě simulace cyklů den/noc pomocí sinusoidy, ke které je dále přičten druhý sinusový průběh s dlouhou periodou pro simulaci změny ročního období. Druhý vstup představuje poruchu ve formě druhé sinusoidy s rozdílnou amplitudou i periodou. Tento vstup si lze představit například jako změny ve spotřebě energie o víkendech. Výstupem je simulace spotřeby energie, pouze pro testovací účely. Tento model v žádném případě nepředstavuje model skutečné spotřeby města, slouží pouze pro testování naprogramovaných algoritmů.

### 6.2. Testované algoritmy

Proces učení neuronového modelu je v principu vždy stejný, bez ohledu na použitou neuronovou strukturu nebo učící algoritmus, viz Obr. 6.2, kde blok NU/NN představuje neuronovou jednotku nebo síť. V případě testování algoritmů je princip také shodný, ovšem s tím rozdílem, že místo reálného systému a reálných vstupních dat je použit simulační model a data.

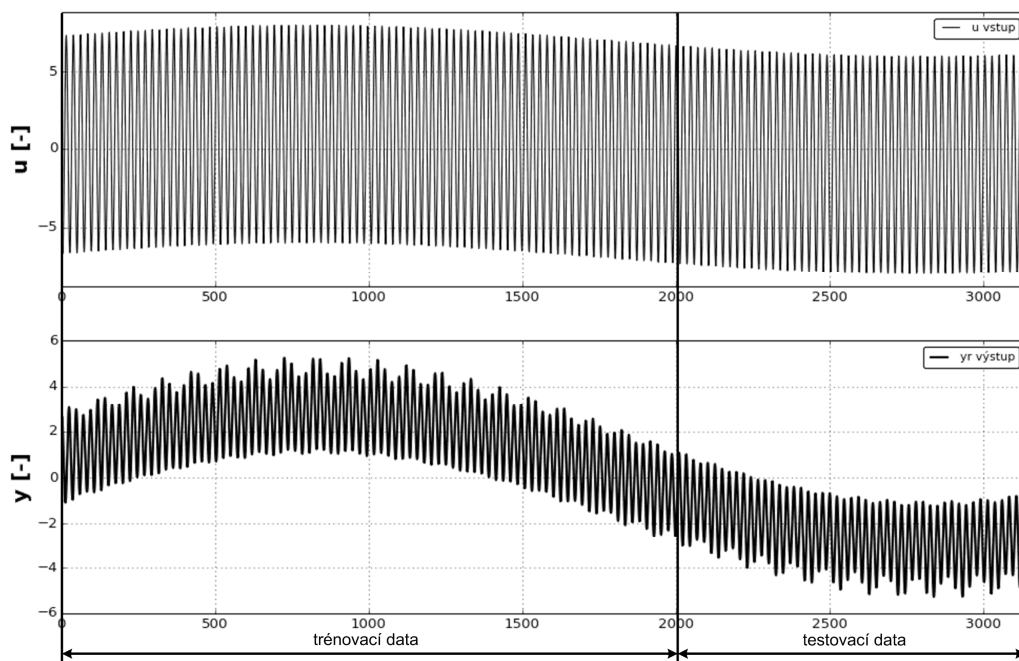


Obr. 6.2 - Schéma učení neuronových jednotek nebo sítí

Testovány jsou pouze kombinace neuronových struktur a učících algoritmů, které jsou využity v této práci. Testování zahrnuje v první řadě naučení se simulačního modelu a následné ověření pomocí vstupních dat, která nebyla zahrnuta v učící fázi. Pro testování jsou použity shodné naprogramované funkce jako pro návrh modelu spotřeby energie dle skutečných dat, kde je hlavním rozdílem zpracování vstupních dat pro dané modely. Jak již bylo řečeno, ověření funkce naprogramovaných algoritmů probíhá pro čtyři kombinace neuronových modelů a učících algoritmů. Jedná se o lineární neuronovou jednotku učenou algoritmem gradient descent, neuronové jednotky vyšších řádů (QNU, CNU) učené algoritmem Levenberg–Marquardt a neuronovou sítí MLP učenou algoritmem advanced resilient propagation.

### 6.3. Výsledky testování na simulačních datech

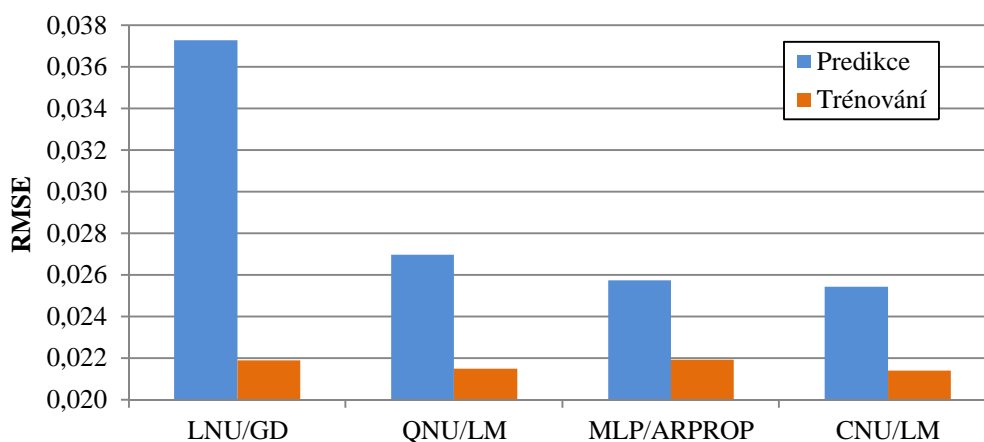
Data jsou rozdělena na trénovací část, která tvoří dvě třetiny a zbylá třetina je využita pro testování, viz Obr. 6.3.



Obr. 6.3 - Rozdělení simulačních dat na trénovací a testovací

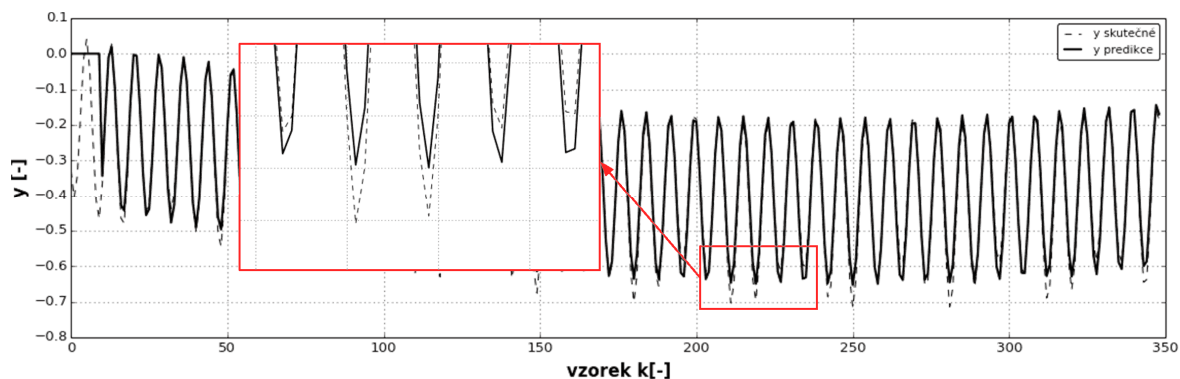
Vstupní vektor všech testovaných algoritmů je shodný, skládá se pouze z posledních deseti hodnot vstupního průběhu, a proto by naučené modely neměly být schopné postihnout víkendové změny, o kterých není ve vstupních datech žádná informace. Počet hledaných parametrů neuronových jednotek je dán rozměrem vstupního vektoru, pro LNU je to 11 vah (vstupní hodnoty + prahová hodnota), pro QNU 66 vah a pro CNU 286 vah. U neuronové sítě je počet parametrů dán i počtem neuronů ve skryté vrstvě. Testovaná síť MLP má 5 neuronů ve skryté vrstvě, to je 55 vah ve skryté vrstvě a k tomu dalších 6 vah ve výstupní vrstvě, což je celkem 61 vah.

Všechny testované modely jsou schopné se naučit trénovací data s žádanou odchylkou, ovšem jednotky vyšších řádů i síť MLP jsou schopné se naučit i na vyšší přesnost, kde už je nutné hlídat přeučení, při kterém se zhoršuje kvalita predikce.



Obr. 6.4 - Porovnání predikčních schopností jednotlivých modelů na simulačních datech

Na Obr. 6.4 je přehledné porovnání schopností jednotlivých ověřovaných algoritmů. Všechny testované modely jsou schopné se velice dobře naučit i predikovat jak denní cyklus, tak cyklus ročních období. Poruchový průběh ve formě víkendových změn nedokáže predikovat žádný z modelů (Obr. 6.5). Síť MLP je schopná dosáhnout velice dobré predikce i při nižší přesnosti natrénování a menším počtu hledaných vah, dokáže tedy dobře zobecňovat. Její další výhodou je rychlé a méně výpočetně náročné učení díky algoritmu ARPROP.



Obr. 6.5 - Predikce podle modelu CNU, „víkendové“ změny model neumí predikovat

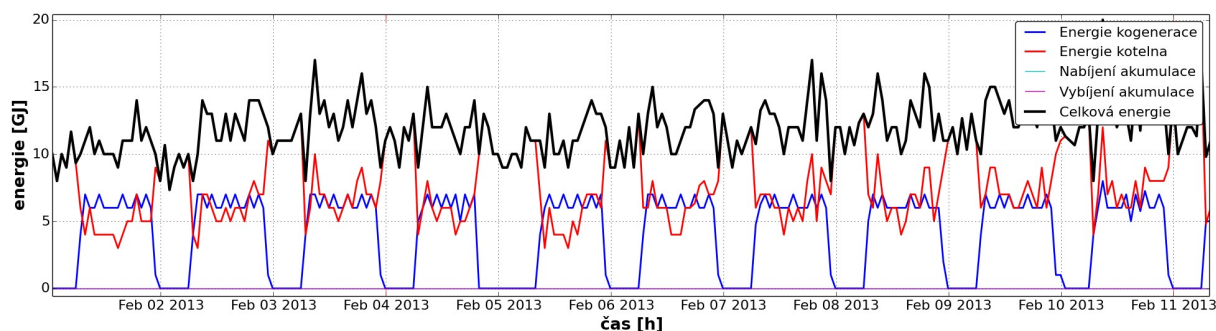
## 7. EXPERIMENTÁLNÍ ANALÝZA

Naprogramované a otestované neuronové modely a učící algoritmy je nyní možné použít na to, k čemu jsou určeny, tedy predikci spotřeby energie města. Nejdříve je ovšem nutné připravit vhodná vstupní data a provést jejich analýzu, pro správné nastavení parametrů modelů. V práci je navrženo několik různých neuronových modelů i přístupů pro vytváření vstupního vektoru, tyto různé přístupy i modely jsou dále popsány a vzájemně porovnány.

### 7.1. Příprava dat

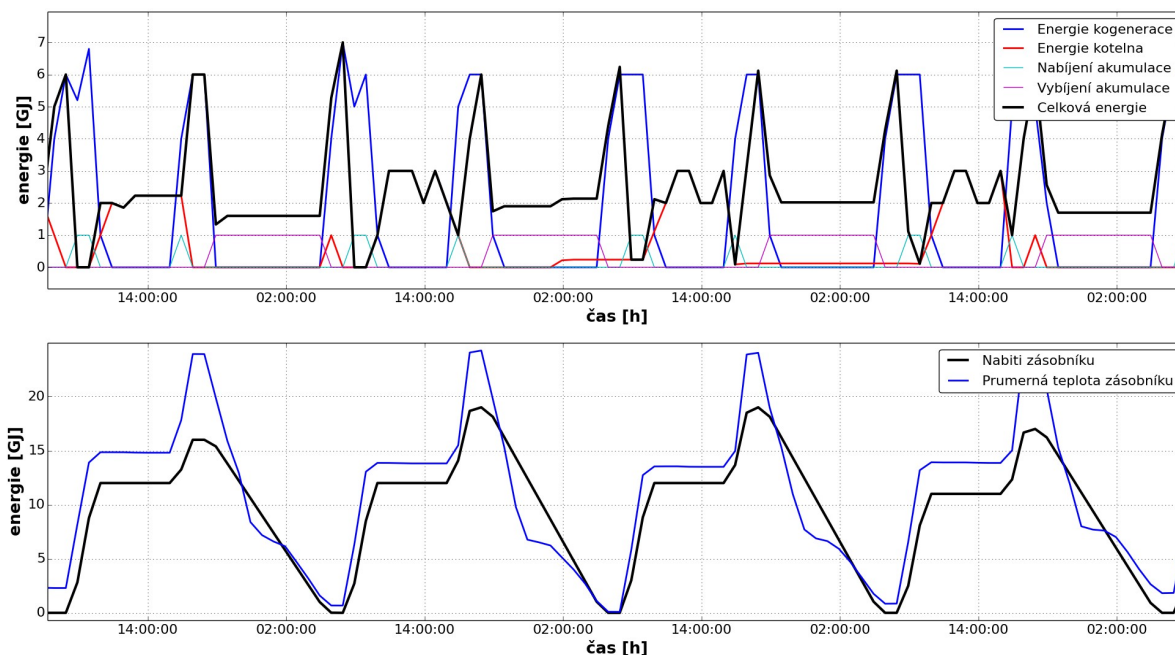
Z databáze RcWare byla stažena „surová“ data, která byla nejdříve filtrována dle kapitoly 5.1. Staženy byly průběhy všech potřebných veličin od 1. 2. 2013 do 26. 4. 2014, to je 1 rok, 2 měsíce a 26 dní dostupných dat. Všechny modely pracují jen s jednou vstupní veličinou a jednou výstupní. Jako vstupní veličina slouží pouze venkovní teplota, důvodem je hlavně dobrá dostupnost předpovědi venkovní teploty, proto je možné na jejím základě dobře předpovídat spotřebu energie. V případě použití jiných veličin by bylo nutné navrhnout vlastní model jejich predikce. Výstupem je spotřeba energie města v GJ.

Pro naučení modelů je nutné znát celkovou spotřebu energie, která ovšem není dostupná jako přímo měřená veličina. Výhodné by bylo stanovení celkové energie na základě spotřeby plynu měřené fakturačním měřidlem, ta ovšem není dostupná. Známa je pouze energie dodaná kaskádou kotlů (nespolehlivé měřidlo, kapitola 4.3) a kogenerační jednotkou, obě měřené pomocí kalorimetrů. Při tomto způsobu se chyby obou měřidel sčítají a vznikají tím značné nepřesnosti určení celkové energie. Dalším prvkem ovlivňujícím celkovou spotřebu energie jsou akumulární nádrže, u kterých je znám pouze čas nabíjení a vybíjení. Nemají vlastní kalorimetr. Proto je nutné dle logiky nabíjení/vybíjení nádrží (kapitola 4) určovat kolik energie je v nádržích uloženo a jak je v čase rozloženo jejich vybíjení. Díky těmto faktorům je určení celkové energie velice nepřesné a značně je tím ztížena úloha neuronových modelů. V zimním období, kdy se nepoužívají akumulární nádrže, je celková energie pouhým součtem energií vyrobených kaskádou kotlů a kogenerační jednotkou, viz Obr. 7.1.



Obr. 7.1 - Určení celkové energie v zimním období, zásobníky se nepoužívají

V letním období je nutné celkovou energii snížit při nabíjení zásobníků a povolit rozložit při vybíjení, kdy není v chodu kotelna ani kogenerační jednotka. Princip viz Obr. 7.2.

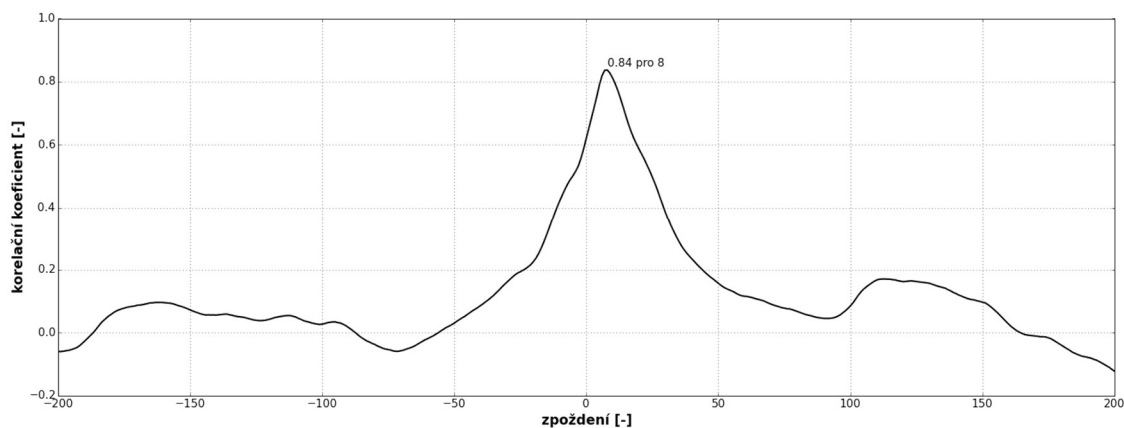


Obr. 7.2 - Určení celkové energie v letním období, nabití zásobníků

Pro ověření správnosti výpočtu je na spodním grafu na Obr. 7.2 vykresleno porovnání dvou metod určení nabití zásobníků. První metoda (černý průběh) je popsána výše. Jedná se o akumulaci energie kogenerační jednotky v době, kdy je indikováno nabíjení zásobníků. Druhá (modrý průběh), kontrolní metoda vychází z naměřených teplot nádrží. Tvar obou průběhů je velice podobný, ovšem u výpočtu založeném na teplotě nádrží je problém se skoky nabití podle toho, jak se posouvá hladina teplé vody, a to i přesto, že výpočet je založen na průměrné teplotě z několika čidel. Z těchto důvodů je v práci použita druhá metoda.

## 7.2. Analýza dat

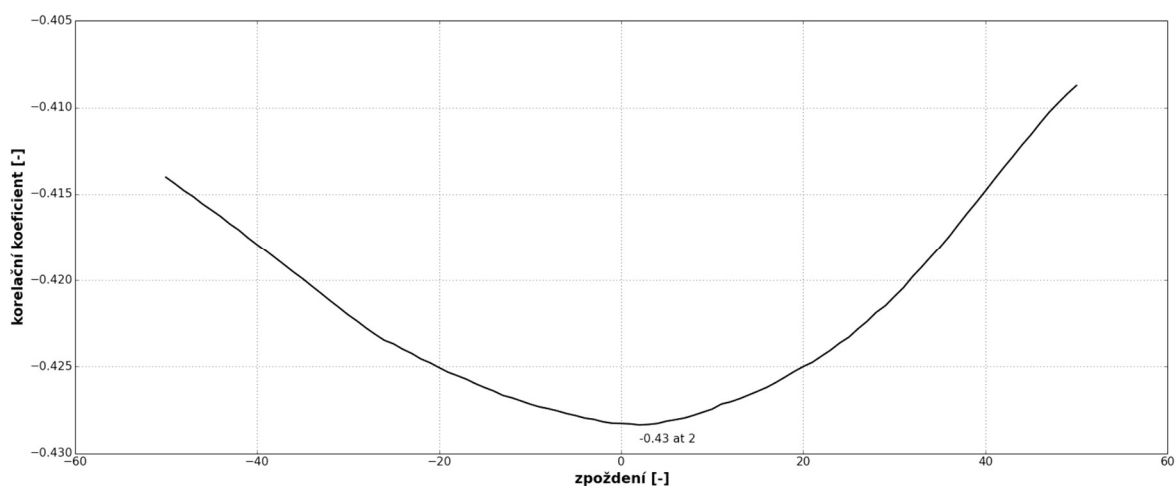
V rámci korelační analýzy je prověřeno zpoždění mezi teplotou topné vody a teplotou vratné vody (Obr. 7.3).



Obr. 7.3 - Vzájemná korelace mezi teplotou topné vody a teplotou vratné vody

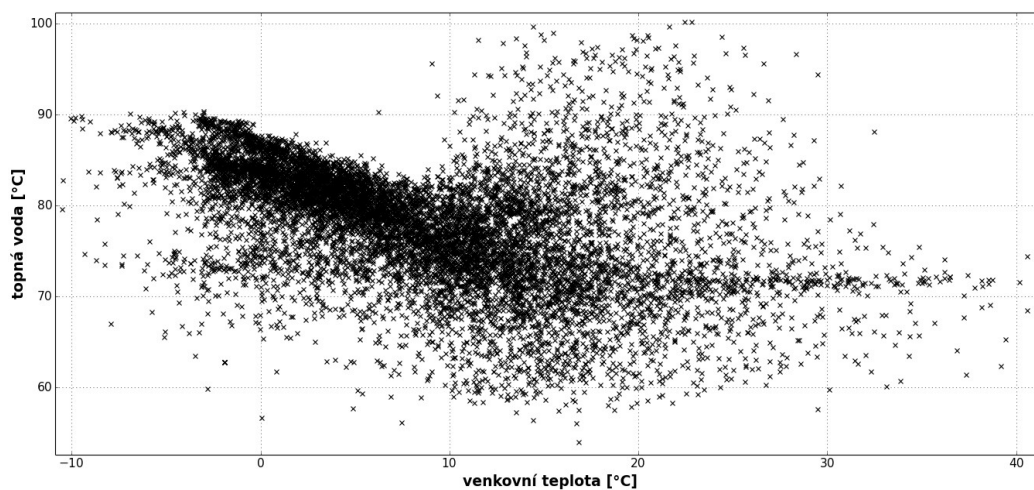
Maximální korelační koeficient je kladný při vzájemném posunu obou průběhů o 8 vzorků, tedy při vzorkování 5 minut je vzájemný posun 40 minut. To je doba, za kterou se topná voda vrátí zpět do kotelný jako vratná voda.

Dále byla provedena korelační analýza mezi venkovní teplotou a celkovou energií (Obr. 7.4). Maximální korelační koeficient je záporný s posunem 2 vzorky. Záporný korelační koeficient ukazuje, že s klesající venkovní teplotou roste spotřeba energie. Zpoždění pouhých 10 minut je dáno tím, že kaskáda kotlů je řízena ekvitermně a celková energie je stanovena na základě měření kalorimetrů v blízkosti kotlů.



Obr. 7.4 - Vzájemná korelace mezi venkovní teplotou a celkovou energií

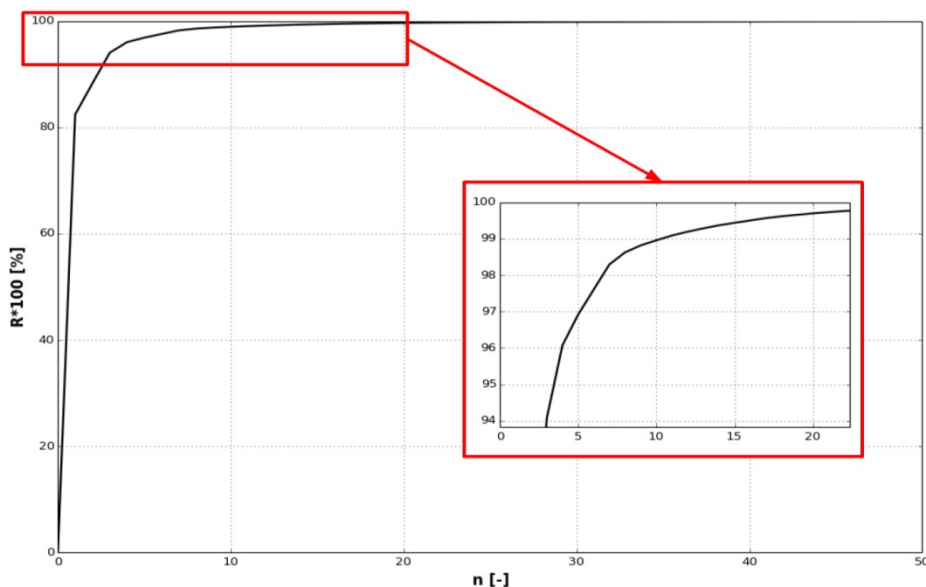
Na Obr. 7.5 je vykreslena závislost topné vody na venkovní teplotě ze všech stažených dat. Pro venkovní teploty od  $-10$  do  $15$  °C se potvrzuje ekvitermní řízení odpovídající křivce kaskády kotlů (Obr. 4.3). Další významná linie je pro teplotu topné vody kolem  $75$  °C, která odpovídá výstupní teplotě kotlů pro venkovní teploty od  $15$  °C. Velký rozptyl hodnot teploty topné vody v oblasti venkovních teplot od  $10$  do  $30$  °C je především způsoben používáním akumulčních nádrží, které v počátku vybíjení dodávají vodu o vysoké teplotě a vybíjejí se až k  $60$  °C.



Obr. 7.5 - Závislost teploty topné vody na venkovní teplotě vykreslená z dostupných dat

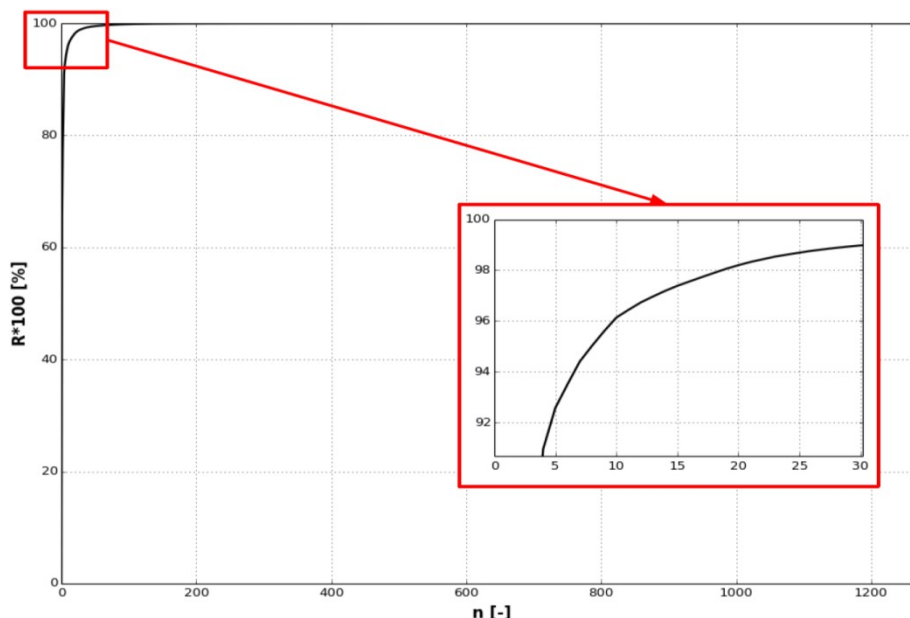


V rámci analýzy dat je také nutné určit, jak moc lze redukovat data pomocí lineární a rozšířené metody PCA. To lze určit na základě poměru  $R$  definovaného v kapitole 5.5. Vzhledem k tomu, že u modelů používajících metodu PCA je vstupní vektor vždy stejný, stačí vykreslit průběh poměru  $R$  jednou pro lineární a jednou pro rozšířené PCA.



Obr. 7.6 - Lineární PCA, redukce vektoru s 50 prvky

Pro lineární PCA (Obr. 7.6) lze vektor úspěšně redukovat na 20 prvků se zachováním 99,6 % informace. Pro jednotku CNU by takový vektor byl stále příliš velký, ale i pro dimenzi 10 je stále zachováno 99 % informace.



Obr. 7.7 - Rozšířené PCA, redukce vektoru s 1275 prvky

Při použití rozšířeného vektoru pro metodu PCA (Obr. 7.7) je zvolena opět redukce na vektor o velikosti 20 prvků, pro který je v tomto případě zachováno 98,2 % rozptylu původních dat.

### 7.3. Navržené neuronové modely

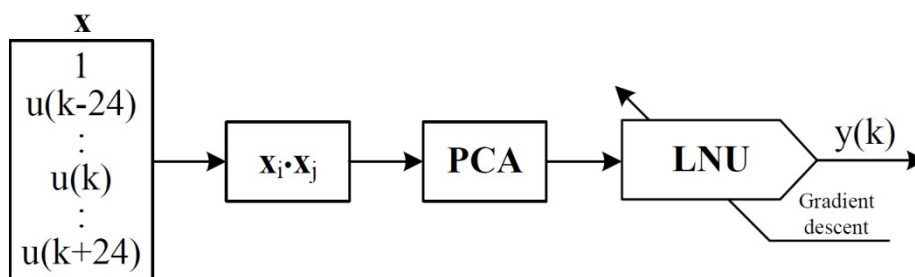
V průběhu zpracovávání této práce bylo navrženo a otestováno téměř dvacet různých modelů s různými vstupními vektory a jejich zpracováním. Také byly testovány různé neuronové struktury v kombinaci s různými učícími algoritmy. Zde je prezentováno a popsáno 8 modelů s nejlepšími výsledky, které jsou založené na metodách popsaných v kapitole 5.

Modely lze rozdělit do dvou kategorií, prvních 6 modelů je založeno na principu rozdělení vstupních dat podle času, proto není potřebný vstup pro určení časové příslušnosti daného vzorku. Jedná se o rozdělení dat podle měsíců, dnů (zda je pracovní den či volno) a podle hodin. Pro pokrytí celého roku je třeba rozdělit data na 576 částí, na základě kterých je nutné natrénovat 576 modelů, kde každý je použitelný jen pro danou hodinu v daný den daného měsíce. Výhodou tohoto přístupu je, že jednotlivé modely umí dobře postihnout specifika každé hodiny (např. ranní špička), ale pokud je požadován celoroční model, je nutné mít k dispozici data minimálně za 1 rok a natrénovat 576 modelů, což je výpočetně náročné. Pokud je požadována predikce na den dopředu, stačí data za poslední měsíc a 24 natrénovaných modelů.

Druhý přístup (poslední dva modely) již musí mít ve vstupním vektoru časovou informaci (den v týdnu a hodina), protože data nejsou nijak rozdělena. Je vždy použito posledních 28 dní dat před predikovaným dnem. Pro pokrytí celého roku stačí 12 modelů, kdy je každý naučen na příslušný měsíc. Jeden model je schopný predikovat celý den (24 hodin) a proto pro predikci jeden den dopředu stačí naučit pouze jeden model.

#### 1. LNU, gradient descent, rozšířené PCA

Lineární neuronová jednotka učená pomocí algoritmu gradient descent se vstupním vektorem redukováným pomocí rozšířené PCA. Princip je názorně vykreslen na Obr. 7.8.



Obr. 7.8 - LNU, gradient descent, rozšířené PCA

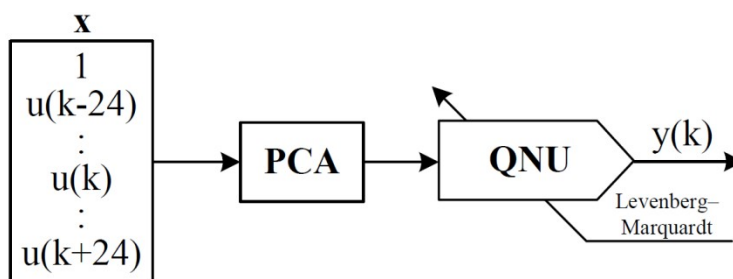
Vstupní vektor zahrnuje venkovní teploty 24 hodin historických hodnot a 24 hodin budoucích hodnot od aktuálního vzorku. Velikost dimenze vektoru, včetně prahu neuronu, je 50. Následně je vektor kvadraticky rozšířen na 1275 prvků.



Pomocí metody PCA je zredukován na 20 prvků a v této podobě slouží jako vstup pro LNU. Jednotka má celkem 21 vah, které jsou adaptovány pomocí algoritmu gradient descent. Normalizace dat je provedena před i po redukci, protože metoda PCA opět změní rozptyl hodnot vstupního vektoru.

## 2. QNU, Levenberg–Marquardt, PCA

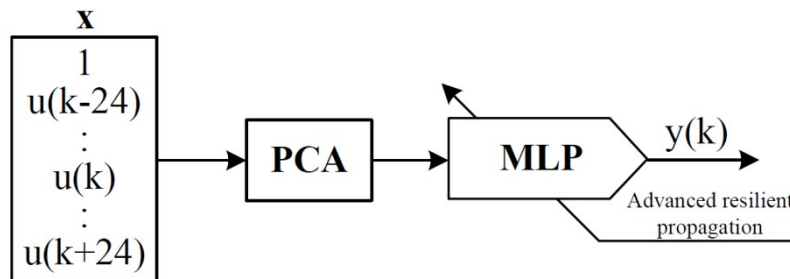
Kvadratická neuronová jednotka učená algoritmem Levenberg–Marquardt se vstupním vektorem redukováným metodou PCA. Princip je vykreslen na Obr. 7.9. Stejně jako v předchozím případě, vstupní vektor zahrnuje kromě aktuálního vzorku i 24 budoucích a 24 historických hodnot teploty. Tento vektor je redukován metodou PCA na vektor o dimenzi 20 a s přidáním prahu neuronu slouží jako vstup pro jednotku QNU, která vektor vnitřně kvadraticky rozšiřuje a proto má 231 vah. Normalizace je opět provedena před i po PCA.



Obr. 7.9 - QNU, Levenberg–Marquardt, PCA

## 3. MLP, advanced resilient propagation, PCA

Model založený na vícevrstvé perceptronové síti má shodné zpracování vstupního vektoru jako předchozí QNU. To je vektor o dimenzi 20. Princip je vykreslen na Obr. 7.10. Počet vah sítě je závislý na počtu neuronů ve skryté vrstvě. Například pro 20 neuronů ve skryté vrstvě, kde každý má 21 vah, má výstupní neuron 21 vah a celkem má síť 441 vah. Normalizace je opět provedena před i po PCA.

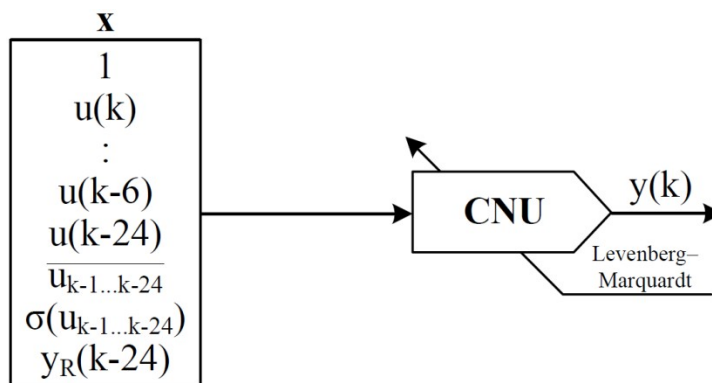


Obr. 7.10 - MLP, advanced resilient propagation, PCA

## 4. CNU, Levenberg–Marquardt

Kubická neuronová jednotka učená algoritmem Levenberg–Marquardt bez redukce vstupního vektoru. Princip je na Obr. 7.11. Jednotka CNU vstupní vektor rozšiřuje kubicky. V případě použití předchozího vstupního vektoru o dimenzi 20 by měla 1771 vah. To už je při použití algoritmu LM příliš mnoho a učení by bylo

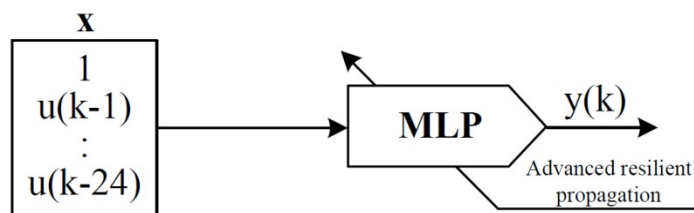
neúnosně pomalé a výpočetně náročné. Proto je použit náhradní vstupní vektor o dimenzi 12, model má potom 364 vah. Vstupní vektor zahrnuje údaje o venkovní teplotě za posledních 6 hodin a před 24 hodinami. Dále obsahuje statistické ukazatele o průměrné venkovní teplotě za poslední 24 hodin a stejně tak směrodatnou odchylku. Posledním členem je skutečná měřená hodnota energie před 24 hodinami. Vstupní vektor je normalizován před vstupem do neuronové jednotky.



Obr. 7.11 - CNU, Levenberg–Marquardt

## 5. MLP, advanced resilient propagation

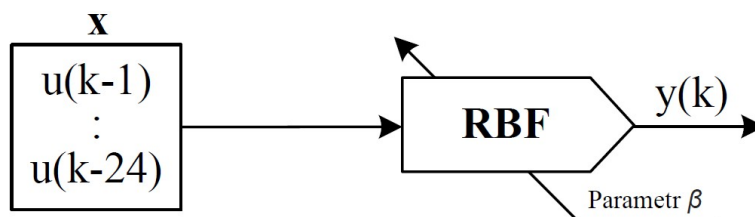
Vícevrstvá perceptronová síť bez redukce vstupního vektoru. Princip je na Obr. 7.12. Vzhledem k tomu, že síť MLP vnitřně nerozšiřuje vstupní vektor, není problém s delším vstupním vektorem, proto je v tomto případě použit velice jednoduchý vektor skládající se pouze z hodnot venkovní teploty za posledních 24 hodin. Počet vah je závislý na počtu neuronů ve skryté vrstvě. Například pro 20 neuronů s 25 vahami má výstupní neuron 21 vah a celkem má síť 521 vah. Vstupní vektor je normalizován před vstupem do neuronové sítě.



Obr. 7.12 - MLP, advanced resilient propagation

## 6. RBF

Síť založená na radiálně bázové funkci (Obr. 7.13).

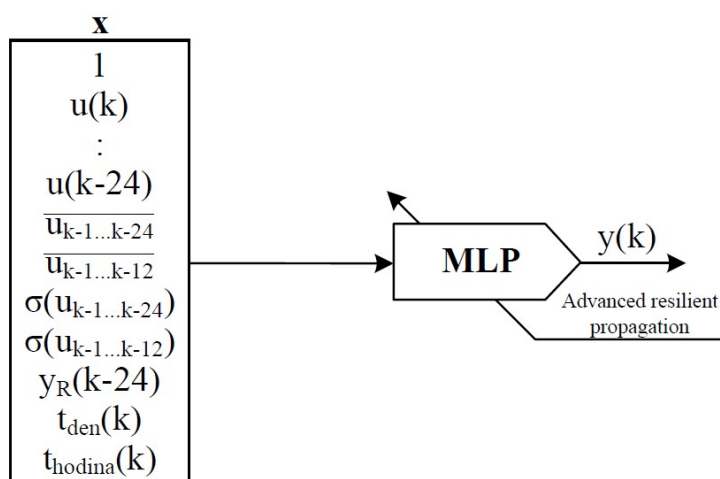
Obr. 7.13 - Síť RBF, hledáme parametr  $\beta$ 

Vstupní vektor je stejný jako u předchozího modelu, ale u RBF sítě se nepřidává prahový vstup. Při běžném použití je nutné na vstupní vektor aplikovat sesku-

povací algoritmus, který určí centra RBF vhodně reprezentující vstupní data. V případě, kdy jsou vstupní data už předem časově rozdělená a tedy vstupních vektorů není mnoho, je možné použít přímo vstupní data jako centra RBF. Proto je implementace sítě v této podobě velice jednoduchá a rychlá. Je nutné najít pouze vhodný parametr  $\beta$ , který určuje šířku jednotlivých funkcí kolem center, jeho hledání je popsáno v následující kapitole.

## 7. MLP, advanced resilient propagation, 28 dní

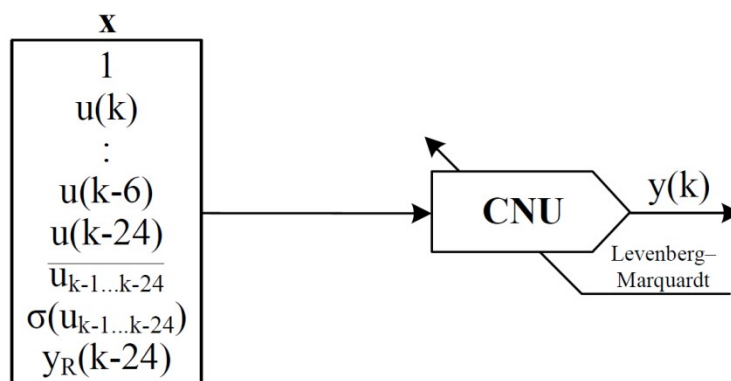
Vícevrstvá perceptronová síť se vstupním vektorem viz Obr. 7.14. V tomto případě není aplikováno rozdělení vstupních dat dle času, a proto je vhodné do vstupního vektoru přidat informaci o čase. Den v týdnu je ve formě čísla 1 až 7 a hodina 0 až 23. Vstupní vektor dále obsahuje venkovní teplotu za posledních 24 hodin včetně aktuální a jako statistické ukazatele jsou zařazeny průměrná teplota a její směrodatná odchylka za posledních 24 a 12 hodin. Také je zařazena skutečná spotřeba energie před 24 hodinami. Počet vah je dán počtem neuronů ve skryté vrstvě, pro 20 neuronů s 32 vahami má výstupní neuron 21 vah a síť má celkem 661 vah. Vstupní vektor je normalizován před vstupem do neuronové sítě.



Obr. 7.14 - MLP, advanced resilient propagation, 28 dní

## 8. CNU, Levenberg–Marquardt, 28 dní

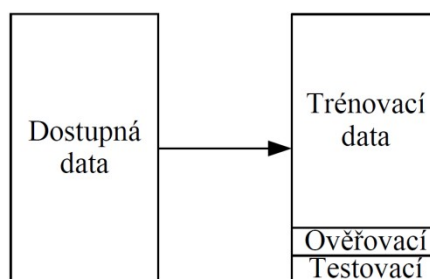
Kubická neuronová jednotka učená algoritmem Levenberg–Marquardt viz Obr. 7.15. I v tomto případě není aplikováno rozdělení vstupních dat dle času a proto je vhodné do vstupního vektoru přidat informaci o čase. Ovšem při testech měl lepší výsledky model se vstupním vektorem pouze s informací o teplotě a skutečné energii před 24 hodinami. Dimenze vstupního vektoru je 12, model CNU má tedy 364 vah. Vstupní vektor je normalizován před vstupem do neuronové jednotky.



Obr. 7.15 - CNU, Levenberg–Marquardt, 28 dní

#### 7.4. Optimalizace parametrů modelů

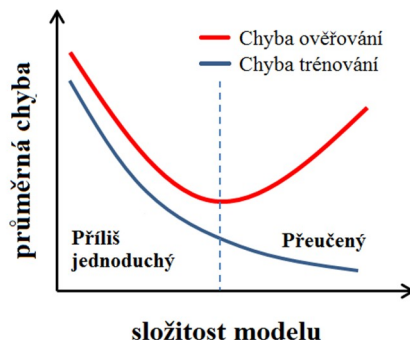
Po naprogramování a otestování všech uvažovaných modelů je nutné najít jejich vhodné parametry, při kterých bude mít model nejlepší predikční schopnosti. Pro každý druh modelu jsou tyto parametry jiné, ale u všech je nutné najít přesnost, na kterou daný model natrénovat, aby byl schopný co nejlépe zobecňovat. U učících algoritmů GD a LM je nutné najít vhodnou rychlost učení, tento parametr určuje velikost kroku učení. Pokud je příliš velký algoritmus může minimum minout a pokud příliš malý, učení je příliš pomalé. U sítí MLP je nutné najít vhodný počet neuronů ve skryté vrstvě, protože pokud má síť příliš mnoho neuronů, má tendenci se přeučovat a tedy hůře zobecňovat. Vzhledem k tomu, že síť RBF není třeba učit, hledá se pouze vhodná šířka radiálně bázových funkcí. Pro účely optimalizace parametrů jsou data rozdělena na trénovací a ověřovací, kdy se vždy model naučí trénovací data na danou přesnost a následně se zjistí přesnost modelu na ověřovacích datech. Tímto způsobem se najdou vhodné parametry modelu, na kterém se následně zjistí jeho kvalita pomocí testovacích dat (Obr. 7.16), na základě kterých se už model nijak neupravuje. Jako ověřovací data je zvoleno 6 dní z různých částí roku, včetně víkendů. Pro testování modelů je použito jiných 6 dní.



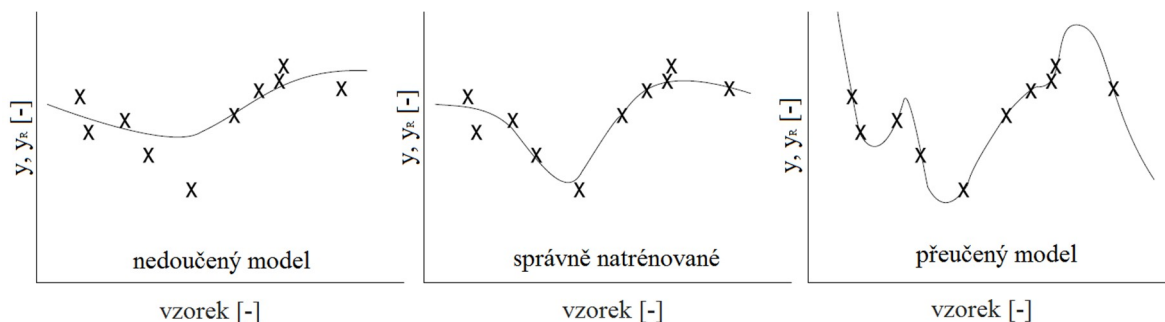
Obr. 7.16 - Rozdělení dostupných dat

Zobecňování znamená, jak kvalitně je model schopný predikovat na základě nových, nenaučených dat. Obr. 7.17 ukazuje typický průběh chyby modelu při trénování a ověřování v závislosti na složitosti modelu. Složitost modelu může znamenat buď počet trénovacích epoch, respektive dosaženou přesnost při trénování, nebo počet neuronů ve skryté vrstvě. Praktická ukázka jak může vypadat

přeučení nebo naopak příliš jednoduchý model je na Obr. 7.18. Touto problematikou se podrobně zabývá článek [40].



Obr. 7.17 - Princip hledání vhodných parametrů modelu [39]



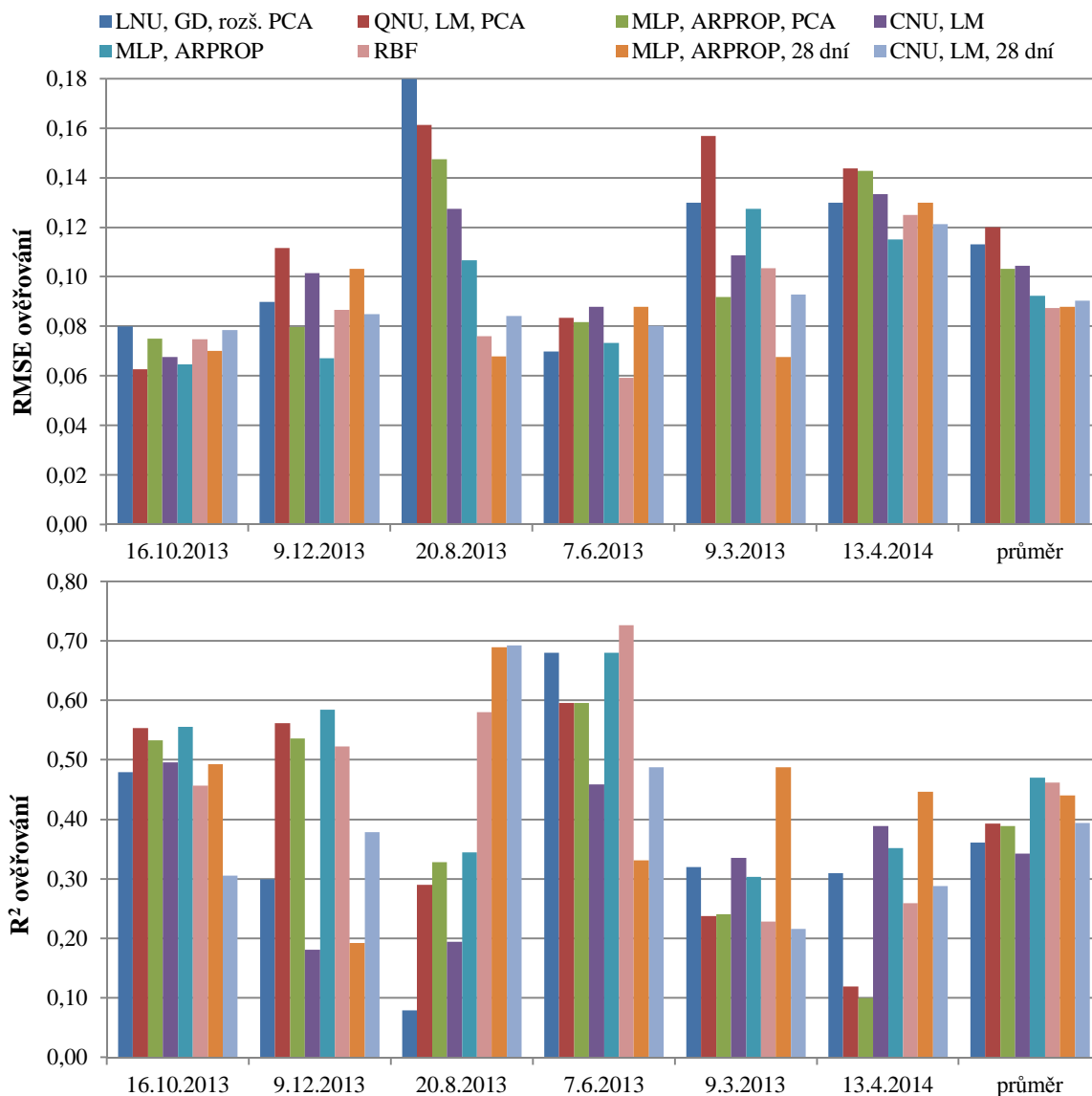
Obr. 7.18 - Problém přesnosti učení modelu [40]

Optimalizace probíhá podle heuristického algoritmu, kdy je na základě zkušeností určen rozsah parametrů, na kterém se pravděpodobně nalézá optimální hodnota parametru. Pro tento rozsah parametrů je následně model naučen a je určena chyba modelu na ověřovací množině dat. Jsou zvoleny parametry, pro které je nejnižší průměrná chyba přes všechny ověřovací dny. Tento postup byl naprogramován a proveden na všech uvažovaných modelech; výsledky viz Tab. 7.1.

model	param.	RMSE trénov.	ověř.	16. 10. 2013	9. 12. 2013	20. 8. 2013	7. 6. 2013	9. 3. 2013	13. 4. 2014	prům.
LNU, GD, rozš. PCA	$\mu$ 0,1	0,1	RMSE	0,080	0,090	0,180	0,070	0,130	0,130	0,113
			R <sup>2</sup>	0,480	0,300	0,080	0,680	0,320	0,310	0,362
QNU, LM, PCA	$\mu$ 0,01	0,1	RMSE	0,063	0,112	0,161	0,084	0,157	0,144	0,120
			R <sup>2</sup>	0,554	0,563	0,290	0,596	0,238	0,120	0,393
MLP, ARPROP, PCA	nh 20	0,125	RMSE	0,075	0,080	0,148	0,082	0,092	0,143	0,103
			R <sup>2</sup>	0,534	0,536	0,329	0,596	0,241	0,100	0,389
CNU, LM	$\mu$ 1	0,08	RMSE	0,068	0,102	0,128	0,088	0,109	0,134	0,105
			R <sup>2</sup>	0,496	0,181	0,195	0,460	0,336	0,389	0,343
MLP, ARPROP	nh 20	0,08	RMSE	0,065	0,067	0,107	0,073	0,128	0,115	0,092
			R <sup>2</sup>	0,555	0,585	0,345	0,681	0,304	0,352	<b>0,470</b>
RBF	$\beta$ 5	-	RMSE	0,075	0,087	0,076	0,059	0,104	0,125	<b>0,088</b>
			R <sup>2</sup>	0,457	0,523	0,581	0,727	0,228	0,260	0,463
MLP, ARPROP, 28 dní	nh 20	0,1	RMSE	0,070	0,103	0,068	0,088	0,068	0,130	0,088
			R <sup>2</sup>	0,494	0,193	0,689	0,331	0,488	0,447	0,440
CNU, LM, 28 dní	$\mu$ 1	0,1	RMSE	0,078	0,085	0,084	0,080	0,093	0,121	0,090
			R <sup>2</sup>	0,306	0,379	0,693	0,488	0,217	0,288	0,395

Tab. 7.1 - Nalezené parametry modelů na základě minimalizace chyby na ověřovacích datech

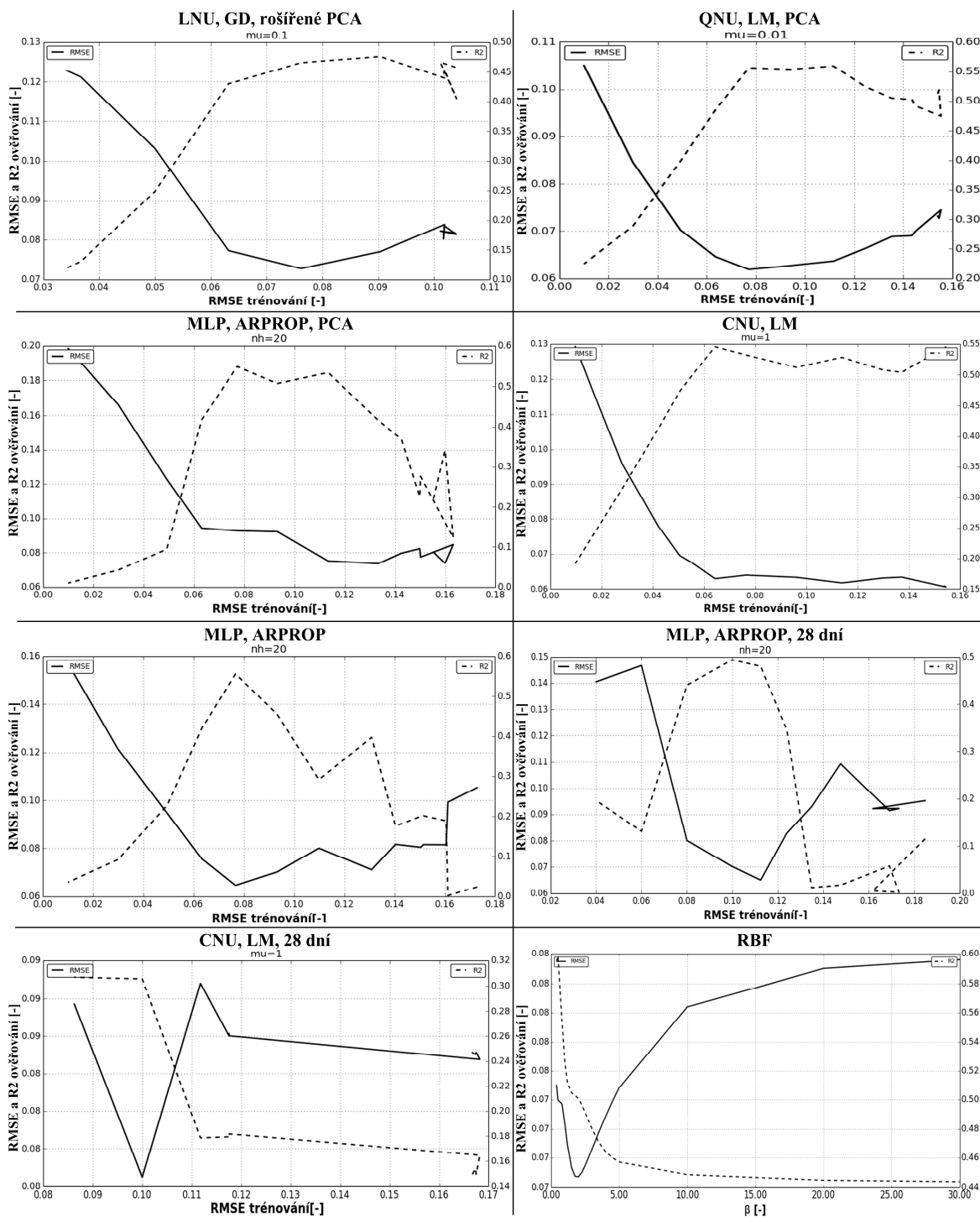
V tabulce výše jsou parametry  $\mu$  (rychlost učení),  $nh$  (počet neuronů ve skryté vrstvě) a  $\beta$  (šířka radiálně bázové funkce). Kompletní výsledky z optimalizace parametrů, včetně všech grafů průběhů ověřovací RMSE a  $R^2$  jsou na přiloženém CD. Na Obr. 7.19 jsou přehledně vykresleny výsledky z Tab. 7.1.



Obr. 7.19 – Nejlepší dosažené výsledky RMSE a  $R^2$  jednotlivých modelů na ověřovacích datech

Hodnoty RMSE ukazují jak blízko je predikovaný průběh skutečnému, ale už ne, jak dobře ho sleduje. Toto hodnotí  $R^2$ , která se v průběhu optimalizace ukázala být velice citlivá na parametry modelu, kdy už při malé změně klesala pod 15 %, což značí špatný model. Praktický význam  $R^2$  je, že ukazuje, jak dobře model sleduje skutečný průběh, tedy například kvalitu predikce ranních a večerních špiček. Na Obr. 7.20 jsou vygenerované průběhy RMSE a  $R^2$  všech modelů pro vybrané parametry pro středu 16. 10. 2013. Ve všech případech lze jasně najít při jak přesném natrénování modelu je RMSE minimální a  $R^2$  maximální. Oba parametry by měly jít vždy proti sobě, ale u všech ověřovacích množin tomu tak nebylo. RBF

sít se netrénuje, a proto je kvalita predikce vykreslena vzhledem k parametru šířky radiálně bázové funkce, který v tomto případě pro určitou hodnotu vykazuje jasné minimum, ovšem není tomu tak u všech ověřovacích dní.



Obr. 7.20 - Průběhy RSME a  $R^2$  všech modelů pro vybrané parametry pro střední 16. 10. 2013

## 7.5. Dosažené výsledky na reálných datech

Po nalezení vhodných parametrů všech modelů je možné přistoupit k závěrečnému hodnocení modelů na základě testovacích dat. Jako testovací data je zvoleno šest dní, náhodně vybraných z celé množiny dostupných dat. Jedná se o pracovní

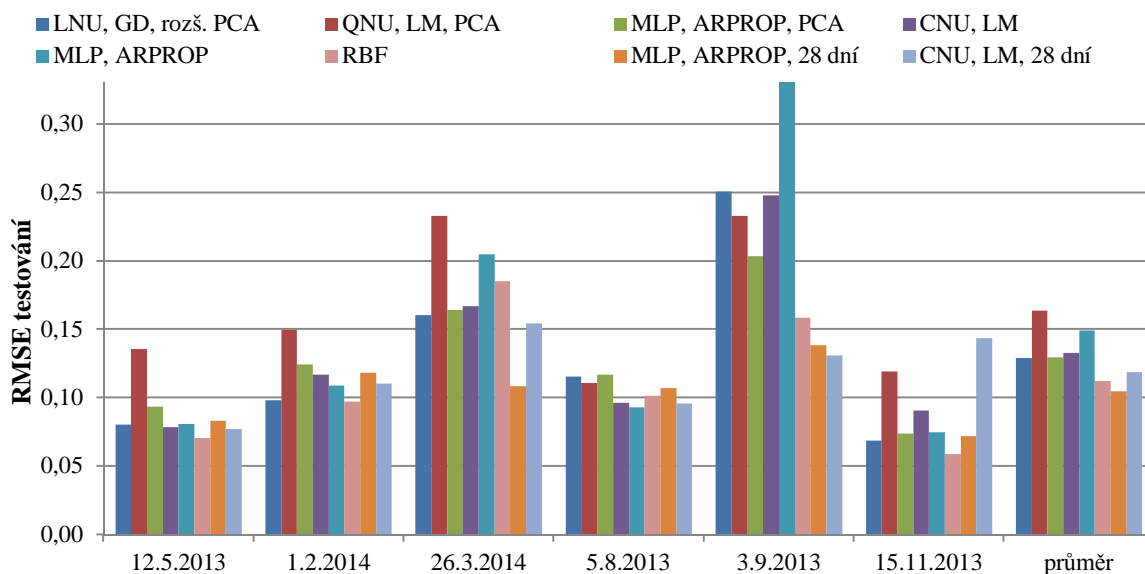


dny i víkendy z různých ročních období. Díky variabilitě naprogramovaných modelů pomocí funkcí lze pomocí malé programové změny využít programy pro optimalizaci i pro testování modelů. Po dokončení každý program vygeneruje výsledky do textového souboru a vykreslí průběhy predikované veličiny a chyby. Kompletní výsledky lze nalézt na přiloženém CD. Vykreslené průběhy testovacích dní jednotlivých modelů predikované energie v porovnání se skutečnou naměřenou energií jsou také vytištěné v příloze 2.

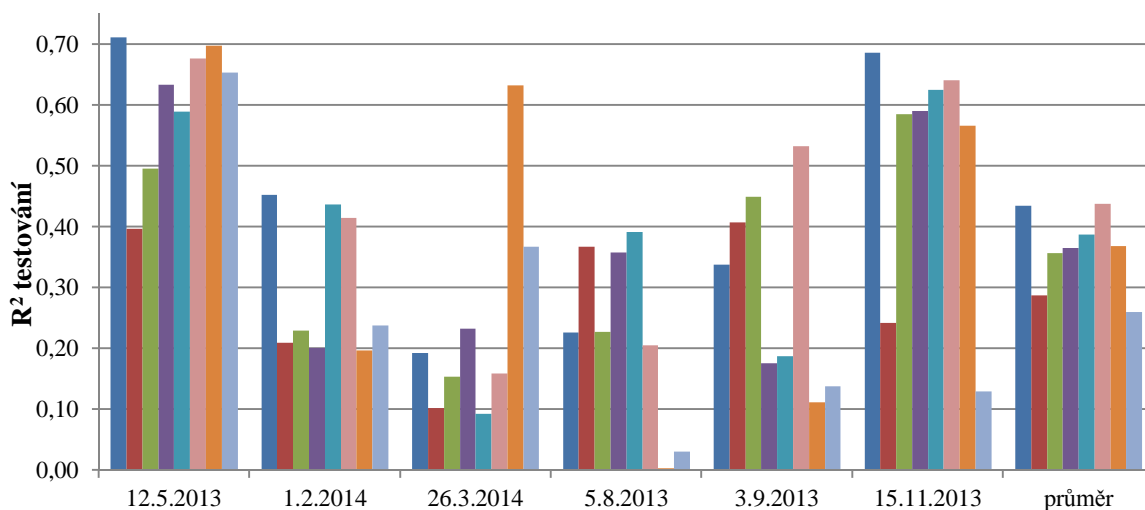
model	param.	RMSE trénov.	testov.	12. 5. 2013	1. 2. 2014	26. 3. 2014	5. 8. 2013	3. 9. 2013	15. 11. 2013	prům.
LNU, GD, rozš. PCA	$\mu$ 0,1	0,1	RMSE	0,080	0,098	0,161	0,116	0,251	0,069	0,129
			R <sup>2</sup>	0,712	0,453	0,193	0,227	0,338	0,686	0,435
QNU, LM, PCA	$\mu$ 0,01	0,1	RMSE	0,136	0,150	0,233	0,111	0,233	0,119	0,164
			R <sup>2</sup>	0,397	0,210	0,103	0,368	0,408	0,242	0,288
MLP, ARPROP, PCA	nh 20	0,125	RMSE	0,093	0,124	0,164	0,117	0,204	0,074	0,129
			R <sup>2</sup>	0,496	0,230	0,155	0,228	0,450	0,585	0,357
CNU, LM	$\mu$ 1	0,08	RMSE	0,079	0,117	0,167	0,096	0,248	0,091	0,133
			R <sup>2</sup>	0,635	0,201	0,233	0,358	0,177	0,591	0,366
MLP, ARPROP	nh 20	0,08	RMSE	0,081	0,109	0,205	0,093	0,333	0,075	0,149
			R <sup>2</sup>	0,590	0,437	0,093	0,392	0,188	0,626	0,388
RBF	$\beta$ 5	-	RMSE	0,071	0,097	0,185	0,102	0,159	0,059	0,112
			R <sup>2</sup>	0,678	0,415	0,159	0,206	0,533	0,641	<b>0,439</b>
MLP, ARPROP, 28 dní	nh 20	0,1	RMSE	0,083	0,118	0,108	0,107	0,138	0,072	<b>0,105</b>
			R <sup>2</sup>	0,698	0,198	0,633	0,003	0,112	0,567	0,368
CNU, LM, 28 dní	$\mu$ 1	0,1	RMSE	0,077	0,111	0,154	0,096	0,131	0,144	0,119
			R <sup>2</sup>	0,654	0,239	0,367	0,031	0,139	0,130	0,260

Tab. 7.2 - Dosažené výsledky RMSE a R<sup>2</sup> všech modelů na testovacích dnech

O něco vyšší průměrné hodnoty RMSE (Tab. 7.2) odpovídají tomu, že parametry modelů byly optimalizovány pro jiné dny (ověřovací data) a tedy pro testovací data nejsou parametry optimální.







Obr. 7.21 - Dosažené výsledky RMSE a  $R^2$  na testovacích datech

Přehledné porovnání jednotlivých modelů je vykresleno pomocí sloupcového grafu na Obr. 7.21. Jak se ukázalo, poslední dva modely bez rozdělování dat sice dosahují nejnižších hodnot RMSE, ale podle ukazatele  $R^2$  si nevedou příliš dobře. To je způsobeno především tím, že tyto modely nejsou schopné predikovat větší špičky spotřeby energie, ale sledují spíše střední hodnotu. Díky tomu dosahují nízkých hodnot RMSE, ale také nižších  $R^2$ .

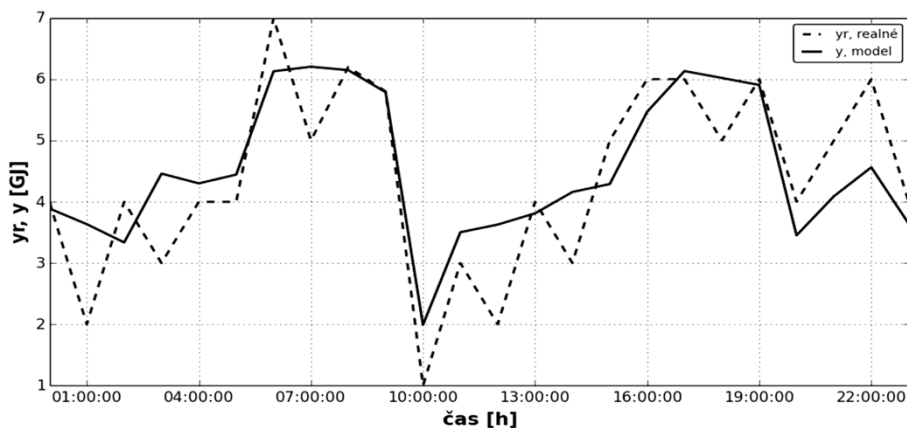
- První testovací den 12. 5. 2013 je jarní neděle z období těsně před začátkem používání akumulčních nádrží. Tento den dělал všem modelům nejmenší problém a všechny předvedly uspokojivou přesnost predikce.
- Sobota 1. 2. 2014, je zimním dnem, kde není výrazná ranní ani večerní špička, které jsou překryté spotřebou tepla na vytápění. Kvalita predikce všech modelů byla tento den průměrná.
- Podle kritéria  $R^2$  je nejhorším dnem středa 26. 3. 2014, jedná se o jeden z nečekaně teplých dnů a i proto zde jsou jasně ve výhodě modely bez rozdělování dat. Modely s rozdělováním dat totiž zahrnují i hodnoty z minulého roku ze stejného období, kdy byla větší zima.
- Pondělí 5. 8. 2013 je den uprostřed letních prázdnin, kdy se naplno využívají zásobníky, a dodávky tepla jsou prakticky pouze pro teplou vodu, proto jsou zde výrazné ranní a večerní špičky. Podle kritéria RMSE se jedná o průměrný den, ale  $R^2$  ukazuje velké zhoršení u modelů bez rozdělování dat. To je způsobeno tím, že tyto modely mají tendenci průběh spíše vyhlazovat a tyto velké špičky predikují hůře. Další zajímavostí tohoto dne je značně snížená ranní špička oproti běžnému dni. Všechny modely proto tuto špičku predikovaly i přes její absenci.
- Úterý 3. 9. 2013 je jedním z posledních teplejších dnů, kdy se ještě používaly akumulční nádrže, proto zde mají navrch modely bez rozdělování dat, modely s rozdělováním dat totiž predikují na základě dat z měsíce září, kde je většina dnů bez používání nádrží.
- Posledním testovacím dnem je pátek 15. 11. 2013, podle obou hodnotících kritérií tento den patří mezi dobře predikovatelné. To je dáno tím, že se jedná v rámci měsíce o typický den.

## 8. ZÁVĚR

Cílem a výstupem této diplomové práce byl návrh a naprogramování modelu spotřeby energie komplexu budov s využitím neuronových struktur. Základním požadavkem byla schopnost předpovědět spotřebu na 24 hodin dopředu a případně vypočítat hodinovou spotřebu energie kterýkoliv den v roce na základě typického průběhu teplot. Pro návrh tohoto modelu byly využité neuronové struktury doporučené v zadání a jejich výsledky byly vzájemně porovnány. Testování neuronových modelů předcházela vhodná příprava dat, jejich případné další zpracování a nalezení vhodných parametrů každého modelu. Následně bylo možné hodnotit kvalitu predikce jednotlivých modelů na testovacích datech a zvolit vyhovující model.

V práci popsané modely (kapitola 7.3) predikují spotřebu energie na základě vstupních vektorů skládajících se převážně z aktuální historie (typicky 24 hodin) a předpovědi venkovní teploty, případně dalších pomocných veličin. Modely jsou navrženy pro hodinové vzorkování vstupních dat. Z osmi zde představených modelů (Tab. 7.2) je obtížné zvolit jeden nejlepší, který je schopný popsat většinu dat. Výběr záleží především na konkrétním cílovém použití modelu. Proto doporučím jeden model každého typu, tedy jeden s rozdělováním dat a druhý bez.

Na každém testovacím dni se dle hodnotících kritérií ukázal být nejlepším jiný model, rozhodující je však schopnost dobré predikce pro kterýkoliv den, a proto jsou modely vybírány na základě průměrných hodnot přes všechny testovací dny (Obr. 7.21). Jako zástupce kategorie modelů s rozdělováním dat jednoznačně vítězí RBF síť (Obr. 8.1), která dokázala podle kritéria  $R^2$  v průměru vysvětlit 44 % rozptylu predikovaného průběhu na testovacích dnech. Podle RMSE je druhou nejlepším s chybou RMSE rovnou 0,112.

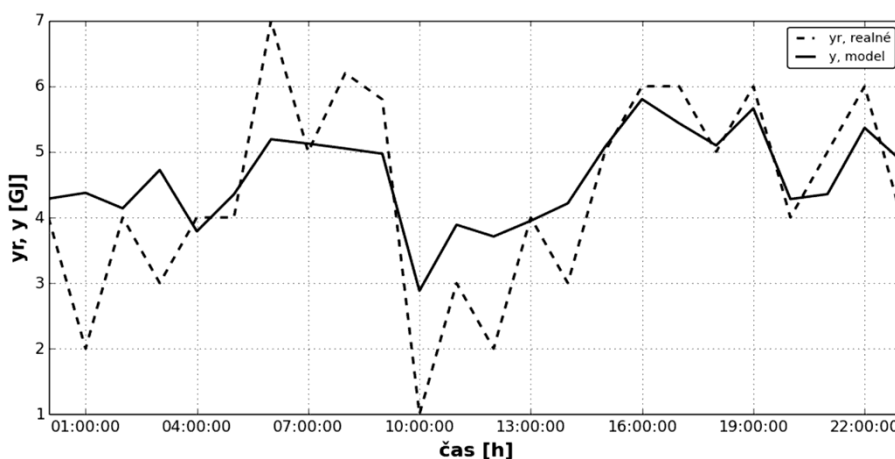


Obr. 8.1 - Průběh predikce modelu RBF pro 12. 5. 2013 s rozdělováním dat

V druhé kategorii bez rozdělování dat se osvědčil model založený na MLP síti učený algoritmem ARPROP (Obr. 8.2), který dosáhl nejlepší průměrné hodnoty RMSE na testovacích dnech a to 0,105. Podle kritéria  $R^2$  se umístil až na čtvrtém

místě s hodnotou 36,8 %, což je dáno právě povahou modelu bez rozdělování dat, který má tendenci vyhlazovat predikovaný průběh, viz Obr. 8.2.

Model založený na síti RBF podával v průběhu testování konzistentní výsledky pro všechny testovací i ověřovací dny, proto ho v rámci této práce lze zvolit jako nejvhodnější metodu pro predikci spotřeby energie daného města. Další směr vývoje tohoto modelu může být rozšíření vstupního vektoru o další veličiny ovlivňující spotřebu energie. Dále by byla také vhodná úprava programu modelu, aby se byl schopný adaptovat v reálném čase a díky tomu podával vždy nejpřesnější možnou předpověď spotřeby energie.



Obr. 8.2 - Průběh predikce modelu MLP, ARPROP pro 12. 5. 2013 bez rozdělování dat

Všechny navržené modely jsou schopné v dostatečné kvalitě predikovat spotřebu energie ve dny, ve kterých je spotřeba typická v rámci daného období (například neděle 12. 5. 2013, viz příloha 2). Pokud se objeví náhlá změna chování samotného systému (města), například událost, které se účastní velká část obyvatel města a jiné těžko predikovatelné nepravidelnosti, již se projeví kvality jen některých modelů. Stejně tak některé mají problémy v případě změny režimu, kdy se používají akumulční nádrže a kdy ne, příkladem může být úterý 3. 9. 2013. Kvalitu predikce lze v těchto situacích zlepšit přidáním dalšího vstupu, který by odlišoval tyto specifické dny. To by znamenalo vytvoření tohoto vstupu pro trénovací data a i jeho přidání do řídicího systému zdroje tepla, proto není v této práci zahrnut. Zahrnout však šlo odlišení pravidelných změn ve spotřebě energie, jako jsou především víkendy. Toto odlišení je zahrnuto u modelů s rozdělováním dat, které neodlišují pouze víkendy, ale i dny pracovního klidu.

Článek [41] zabývající se porovnáním tří učících algoritmů na klasické síti MLP pro predikci průtoku řeky, ukazuje, že nejlepší výsledky predikce má algoritmus resilient propagation následovaný LM a poslední je gradient descent. Kvalita predikce je závislá na mnoha faktorech, ale ve většině případů toto pořadí platí. V této práci se toto pořadí plně nepotvrdilo. Jednou z příčin je, že modely

s rozdělováním dat trpěly nedostatkem trénovacích vzorů. Při rozdělení dostupných dat podle měsíce, typu dne a hodiny má model pro každou hodinu průměrně 20 trénovacích vzorů a to je velmi málo, vzhledem ke složitosti systému. I přes tuto nevýhodu si vedly velice dobře v porovnání s druhým typem modelů. Pozitivně překvapil lineární model založený na jednotkách LNU, který při testování vykázal druhou nejvyšší hodnotu  $R^2$  a čtvrtou nejnižší RMSE. Malý počet vah LNU jednotky v tomto případě působí pozitivně na zobecňovací schopnosti modelu, protože model není schopen se přeučit trénovací data. Při porovnání všech modelů se potvrdilo pravidlo KISS<sup>12</sup> [42], kdy jednodušší, méně komplikované mají lepší výsledky než modely komplexní. Zejména dobré výsledky prokázal velice jednoduchý model založený na síti RBF.

Negativní vliv na přesnost předvídání spotřeby energie má také nespolehlivost použitých kalorimetrů (kapitola 4.3). Jedná se o problémy s nestálostí měřené veličiny a také s nízkým rozlišením měření (1 GJ). Pokud by se modely učily na základě spolehlivých měření skutečné spotřeby, lze předpokládat i zlepšení kvality predikce.

Jedním z hlavních možných budoucích využití navrženého modelu spotřeby energie města může být optimální řízení centrálního zdroje tepla, viz Obr. 3.1. Na základě hodnot předpovězených modelem lze řídit výkon jednotlivých tepelných zdrojů a tím dosáhnout reálných úspor energie a nižší spotřeby plynu. Při řízení na základě modelu je vyžadována jistá spolehlivost predikovaných hodnot, proto by bylo nutné nejdříve model podrobit širšímu testování v reálném provozu s případnými úpravami modelu pro zvýšení jeho přesnosti. Také modernizace měřícího vybavení centrálního zdroje tepla by přinesla zvýšení spolehlivosti predikce a tedy širší možnosti úspor energie.

---

<sup>12</sup> KISS - Keep It Simple, Stupid, Zachovej to jednoduché, hlupáku! [42]

## 9. POUŽITÉ ZDROJE

- [1] MIKŠ, Lubomír. *Energetická spotřeba budov – dogmata a fakta*. časopis Stavebnictví, 2012, roč. 6, č. 1, s. 62–64. ISSN 1802-2030.
- [2] RUMELHART, D. E., HINTON, G. E. a WILLIAMS, R. J. *Learning representations by back-propagating errors*. Nature, 1986, č. 323, s. 533–536. ISSN 0028-0836.
- [3] HIPPERT, H. S., PEDREIRA, C. E. a SOUZA, R. C. *Neural Networks for Short-Term Load Forecasting: A Review and Evaluation*. IEEE Transactions on Power Systems, 2001, roč. 16, č. 1, s. 44–55. ISSN 0885-8950.
- [4] ZHAO, Hai-xiang a MAGOULÉS, Frédéric. *A review on the prediction of building energy consumption*. Renewable and Sustainable Energy Reviews, 2012, č. 16, s. 3586-3592. ISSN 1364-0321.
- [5] MOREL, N., BAUER, M., EL-KHOURY, M. a KRAUSS J. *NEUROBAT, a Predictive and Adaptive Heating Control System Using Artificial Neural Networks*. International Journal of Solar Energy, 2001, č. 21, s. 161–201. ISSN 0142-5919.
- [6] DHAR, Amitava. *Development of Fourier Series and Artificial Neural Network Approaches to Model Hourly Energy Use in Commercial Buildings*. College Station, 1995. Disertační práce. Texas A&M University. Mechanical Engineering.
- [7] SCHÖLKOPF, B. a SMOLA, A. J. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. Cambridge: MIT Press, 2002. 626 s. ISBN 02-621-9475-9.
- [8] LU, C. N., WU, H. T. a VEMURI, S. *Neural Network Based Short Term Load Forecasting*. IEEE Transactions on Power Systems, 1993, roč. 8, č. 1, s. 336–342. ISSN 0885-8950.
- [9] MOHARARI, N. S. a DEBS, A. S. An Artificial Neural Network Based Short Term Load Forecasting with Special Tuning for Weekends and Seasonal Changes. In: *Neural Networks to Power Systems, 1993. ANNPS '93*. Yokohama, Japonsko 19.–22. dubna 1993. IEEE, 1993, s. 279–283. ISBN 0-7803-1217-1.
- [10] KARATASOU, S., SANTAMOURIS, M. a GEROS, V. Prediction of Energy Consumption in Buildings with Artificial Intelligent Techniques and Chaos Time Series Analysis. In: *Energy Performance and Environmental Quality of Buildings, International Workshop (EPEQUB 2006)*. Milos, Řecko 6.–7. července 2006. INIVE eeig, 2006, 5 s.
- [11] BAŠTA, Jiří. *Hydraulika otopných soustav*. Praha: Vydavatelství ČVUT, 2003. 252 s. ISBN 80-01-02808-9.
- [12] *Dvakrát více zkušeností a inovační síly: Z Loos se stává Bosch* [online]. TZB-info. [cit. 8. 5. 2014]. Aktualizováno: 2. 7. 2012. Dostupné z: <http://vytapeni.tzb-info.cz/8779-dvakrat-vice-zkusenosti-a-inovacni-sily-z-loos-se-stava-bosch>

- [13] *Водогрейные котлы LOOS Unimat (750 - 38000 кВт)* [online]. Новая Энергия, ООО. [cit. 8. 5. 2014]. Aktualizováno: 2009. Dostupné z: <http://novaenergo.ru/kotly-loos-vod.html>
- [14] *Tepl vodní a horkovodní kotle* [online]. Loos International. [cit. 8. 5. 2014]. Aktualizováno: 11. 10. 2005. Dostupné z: [www.loos.cz/cz/informace/Tepl-HorkovodniKotleCZ2005.pdf](http://www.loos.cz/cz/informace/Tepl-HorkovodniKotleCZ2005.pdf)
- [15] *Kogenerační jednotky - soubory ke stažení* [online]. TEDOM a.s. [cit. 9. 5. 2014]. Aktualizováno: 2010. Dostupné z: <http://kogenerace.tedom.com/kogeneracni-jednotky-download.html>
- [16] *Gas engine TCG 2020* [online]. MWM. [cit. 9. 5. 2014]. Aktualizováno: 9. 4. 2013. Dostupné z: <http://www.mwm.net/en/products/gas-engines-power-generators/tcg-2020/>
- [17] *Ukončené produkty teplo a chlad* [online]. Kamstrup A/S. [cit. 9. 5. 2014]. Aktualizováno: 2011. Dostupné z: <http://www.kamstrup.cz/18735/ukoncene-produkty>
- [18] *MULTICAL 601* [online]. Kamstrup A/S. [cit. 9. 5. 2014]. Aktualizováno: 2012. Dostupné z: <http://kamstrup.com/2736/MULTICAL-601>
- [19] *Immersion temperature sensor 100 mm LG-Ni1000, with protection pocket* [online]. Siemens Switzerland Ltd. [cit. 10. 5. 2014]. Aktualizováno: 2013. Dostupné z: [https://hit.sbt.siemens.com/HIT/fs\\_global.aspx?&MODULE=Catalog&ACTION=ShowProduct&KEY=BPZ:QAE21..](https://hit.sbt.siemens.com/HIT/fs_global.aspx?&MODULE=Catalog&ACTION=ShowProduct&KEY=BPZ:QAE21..)
- [20] *Ponorné čidlo teploty* [online]. Domat Control System s.r.o. [cit. 10. 5. 2014]. Aktualizováno: 6. 8. 2013. Dostupné z: <http://domat-int.com/wp-content/uploads/KL/CZ/domat ETF cz.pdf>
- [21] *RcWare* [online]. Energocentrum Plus, s.r.o. [cit. 14. 5. 2014]. Aktualizováno: 2014. Dostupné z: <http://rcware.eu/public/rcware>
- [22] DUSENBAYEVA, Ainur. *Data Pre-Processing for Adaptive Modelling of Heating System*. Praha, 2013. Bakalářská práce. ČVUT v Praze. Fakulta strojní. Ústav přístrojové a řídicí techniky.
- [23] MEŠKO, Dezider. *Normalizace dat pro neuronovou síť GAME*. Praha, 2008. Bakalářská práce. ČVUT v Praze. Fakulta elektrotechnická. Elektrotechnika a informatika.
- [24] JAYALAKSHMI, T. a SANTHAKUMARAN, A. *Statistical Normalization and Back Propagation for Classification*. International Journal of Computer Theory and Engineering, 2011, roč. 3, č. 1, s. 89–93. ISSN 1793-8201.
- [25] MALÝ, Vladimír. *Metoda PCA a vícerozměrová analýza falešných sousedů pro posouzení neurčitosti redukovaného stavového vektoru provozních dat energetického zařízení*. Praha, 2010. Bakalářská práce. ČVUT v Praze. Fakulta strojní. Ústav přístrojové a řídicí techniky.

- 
- [26] BUKOVSKÝ, Ivo. *Modeling of Complex Dynamic Systems by Nonconventional Artificial Neural Architectures and Adaptive Approach to Evaluation of Chaotic Time Series*. Praha, 2007. Disertační práce. ČVUT v Praze. Fakulta strojní. Ústav přístrojové a řídicí techniky.
- [27] GUPTA, M. M., JIN, L. a HOMMA, N. *Static and Dynamic Neural Networks, From Fundamentals to Advanced Theory*. Hoboken: Wiley, 2003. 722 s. ISBN 04-712-1948-7.
- [28] ROSENBLATT, F. *The perceptron: A probabilistic model for information storage and organization in the brain*. Psychological Review. 1958, roč. 65, č. 6, s. 386–408. ISSN 1939-1471.
- [29] BUKOVSKÝ, I. a SLÁMA, M. Platforma pro návrh adaptivního regulátoru, implementaci a testování algoritmů. In: *Automatizácia a riadenie v teórii a praxi (ARTEP 2014)*. Stará lesná, Slovensko 5.–7. února 2014. Technická univerzita v Košiciach, 2014, 11 s.
- [30] POWELL, M. J. D. Radial basis functions for multivariable interpolation: a review. In: *Proceedings of the IMA Conference on Algorithms for the Approximation of Functions and Data*. Shrivensham, Anglie 1985. RMCS.
- [31] BUKOVSKÝ, I., HOMMA, N., SMETANA, L., RODRIGUEZ, R., MIRINOVOVA, M. a VRANA, S. Quadratic Neural Unit is a Good Compromise between Linear Models and Neural Networks for Industrial Applications. In: *9th IEEE Int. Conf. on Cognitive Informatics (ICCI'10)*. Peking, Čína 7.–9. července 2010. IEEE, 2010, s. 556–560. ISBN 978-1-4244-8041-8.
- [32] LEVENBERG, K. *A method for the solution of certain problems in least squares*. Quarterly of Applied Mathematics. 1944, roč. 2, s. 164–168. ISSN 1552-4485.
- [33] MARQUARDT, D. *An algorithm for least-squares estimation of nonlinear parameters*. Journal of the Society for Industrial and Applied Mathematics. 1963, roč. 11, č. 2, s. 431–441. ISSN 0036-1399.
- [34] NOCEDAL, Jorge. *Numerical Optimization*. New York: Springer, 1999, 636 s. ISBN 03-879-8793-2.
- [35] YU, H. a WILAMOWSKI, B. M. *Levenberg–Marquardt Training*. Industrial Electronics Handbook, vol. 5 – Intelligent Systems. Boca Raton, Florida: CRC Press, 2011. s. 12/1–12/5. Druhé vydání. ISBN 978-143-9802-892.
- [36] RIEDMILLER, M. a BRAUN, H. A direct adaptive method for faster back-propagation learning: the RPROP algorithm. In: *IEEE International Conference on Neural Networks*. San Francisco, Kalifornie 28. 3. – 1. 4. 1993. IEEE, 1993, s. 586-591. ISBN 0-7803-0999-5.
- [37] ANASTASIADIS, A. D., MAGOULAS, G. D. a VRAHATIS, M. N. An efficient improvement of the Rprop algorithm. In: *Artificial Neural Networks in Pattern Recognition (ANNPR-03)*. Florencie, Itálie 12.–13. září 2003. Florentská univerzita, 2003, s. 197–202.

- 
- [38] ZVÁRA, K. Koeficient determinace v regresi s chybami v obou proměnných. In: *ROBUST 94: Sborník prací*. Malenovice, Česká Republika 17.–21. ledna 1994. Jednota českých matematiků a fyziků, 1994, s. 222–233. ISBN 80-7015-492-6.
- [39] *Predictive Analytics: Evaluate Model Performance* [online]. Ricky Ho. [cit. 29. 5. 2014]. Aktualizováno: 11. 6. 2012. Dostupné z: <http://horicky.blogspot.cz/2012/06/predictive-analytics-evaluate-model.html>
- [40] LAWRENCE, S., GILES, C. L., a TSOI, A. C. *What size neural network gives optimal generalization? Convergence properties of backpropagation*. College Park, Md, University of Maryland, 1996. 35 s. UMIACS-TR-96-22.
- [41] KIŞI, Özgür a UNCUOĞLU, Erdal. *Comparison of three back-propagation training algorithms for two case studies*. Indian Journal of Engineering & Materials Sciences. 2005, č. 12, s. 434–442. ISSN 0975-1017.
- [42] THORBURN, W. M. *The Myth of Occam's Razor*. Mind. 1918, roč. 27, č. 107, s. 345–353. ISSN 0975-1017. ISSN 0026-4423.



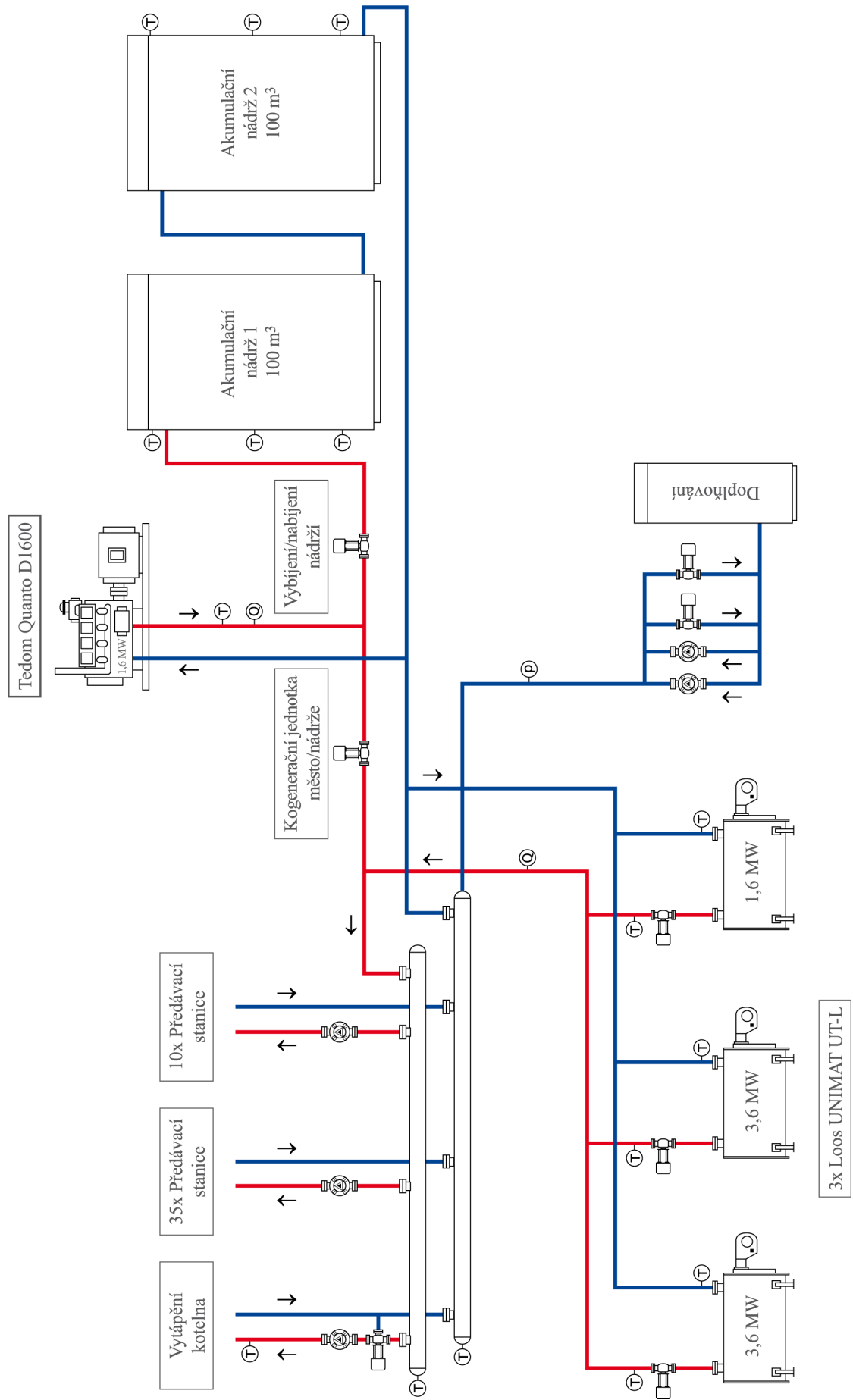
---

## 10. PŘÍLOHY

### Seznam příloh

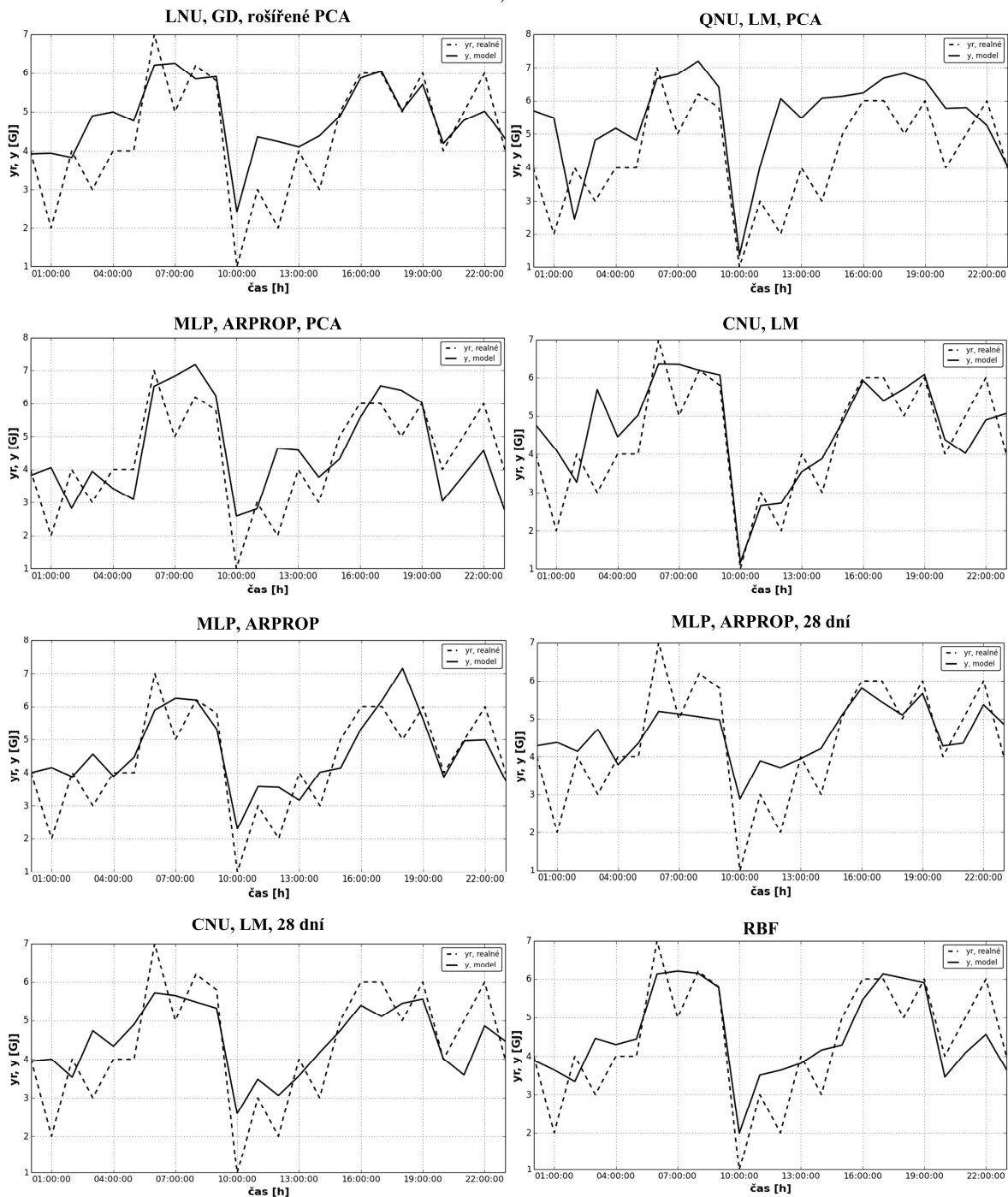
1. Schéma centrálního zdroje tepla.
2. Průběhy predikce spotřeby energie všech testovacích dní podle všech modelů.
3. Zdrojové kódy použitých neuronových struktur.
4. CD s prací v elektronické podobě, naprogramovanými modely a přílohami.

**Příloha 1. - Schéma centrálního zdroje tepla**

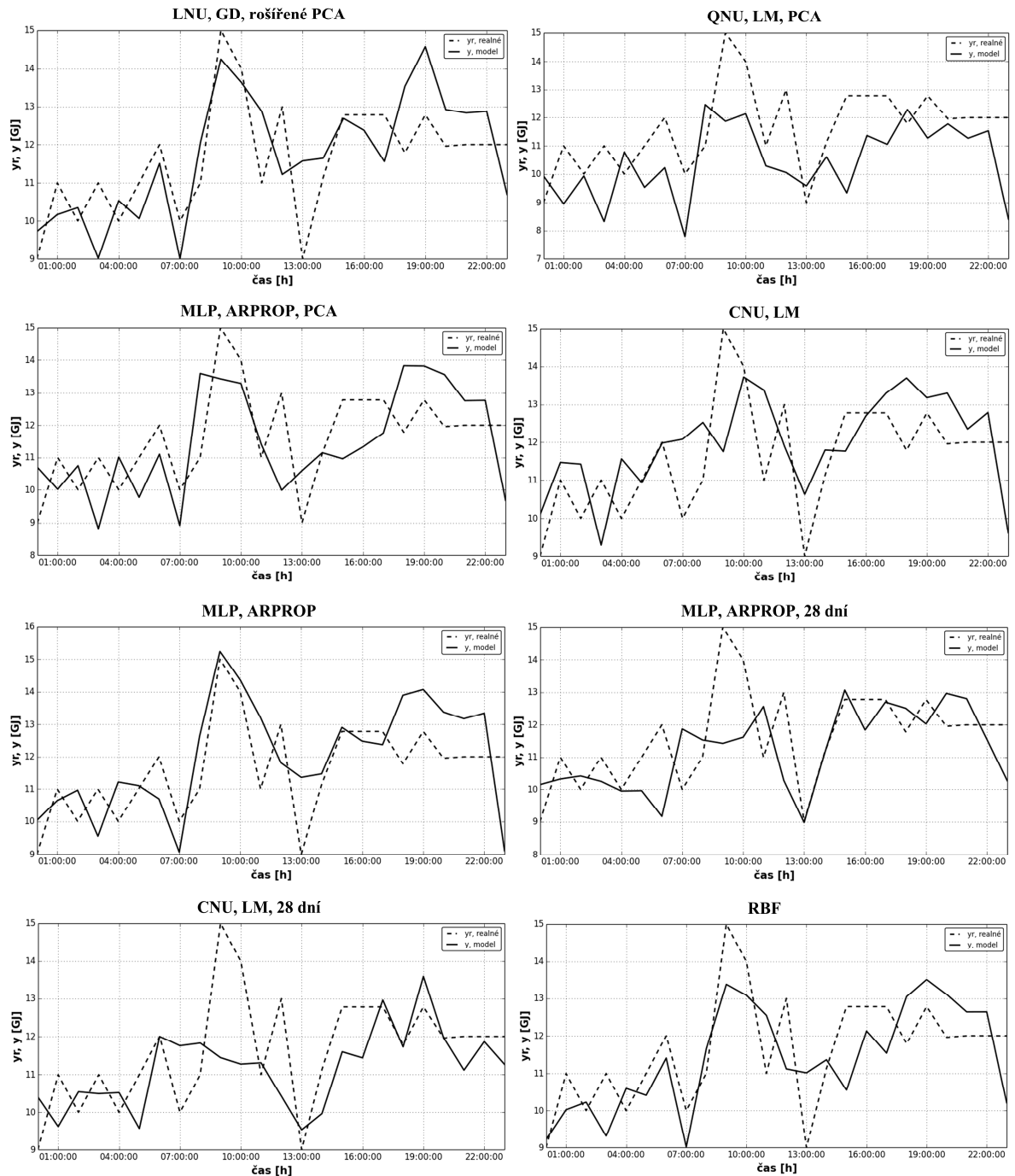


## Příloha 2. – Průběhy predikované spotřeby energie

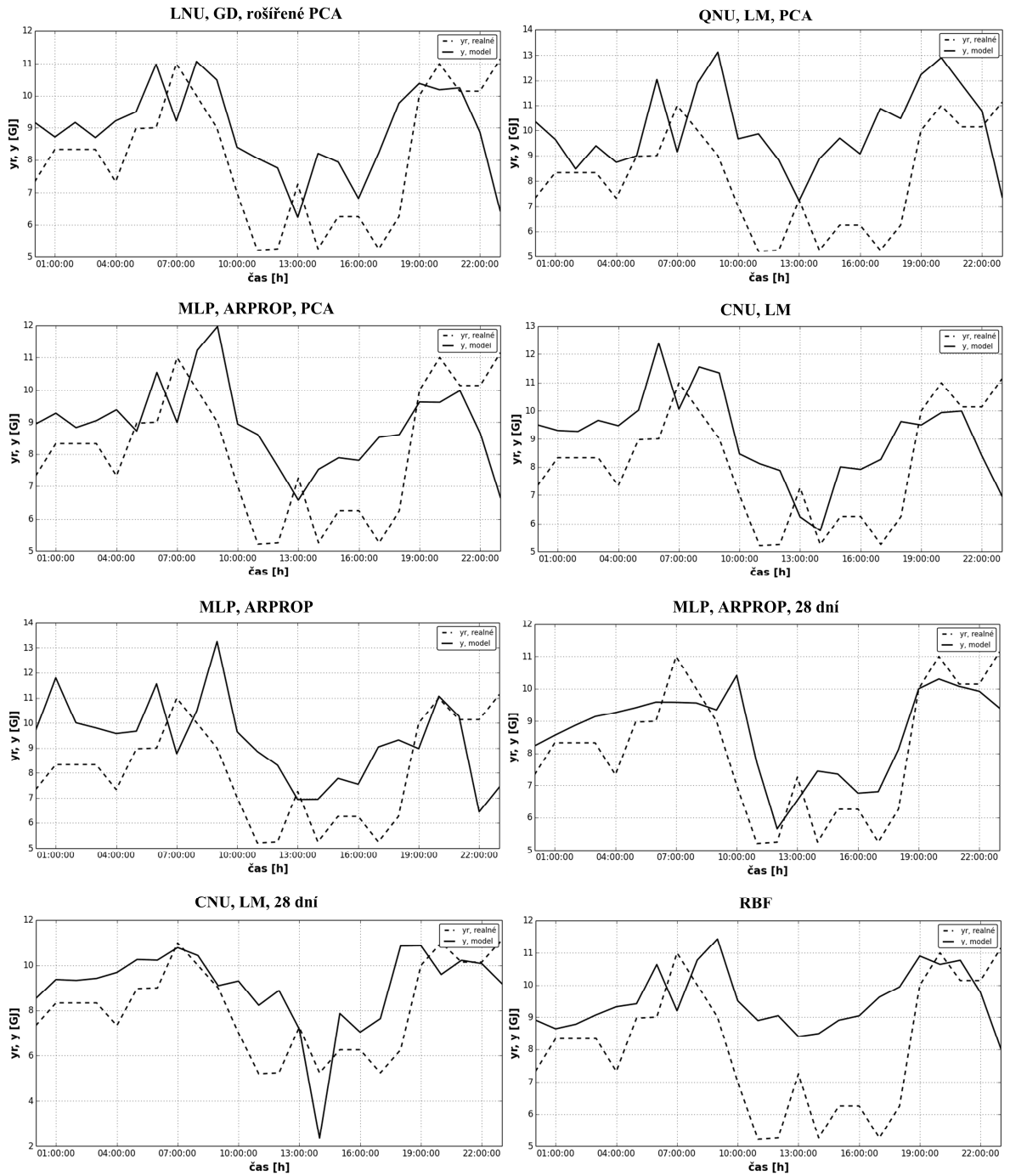
Neděle, 12. 5. 2013



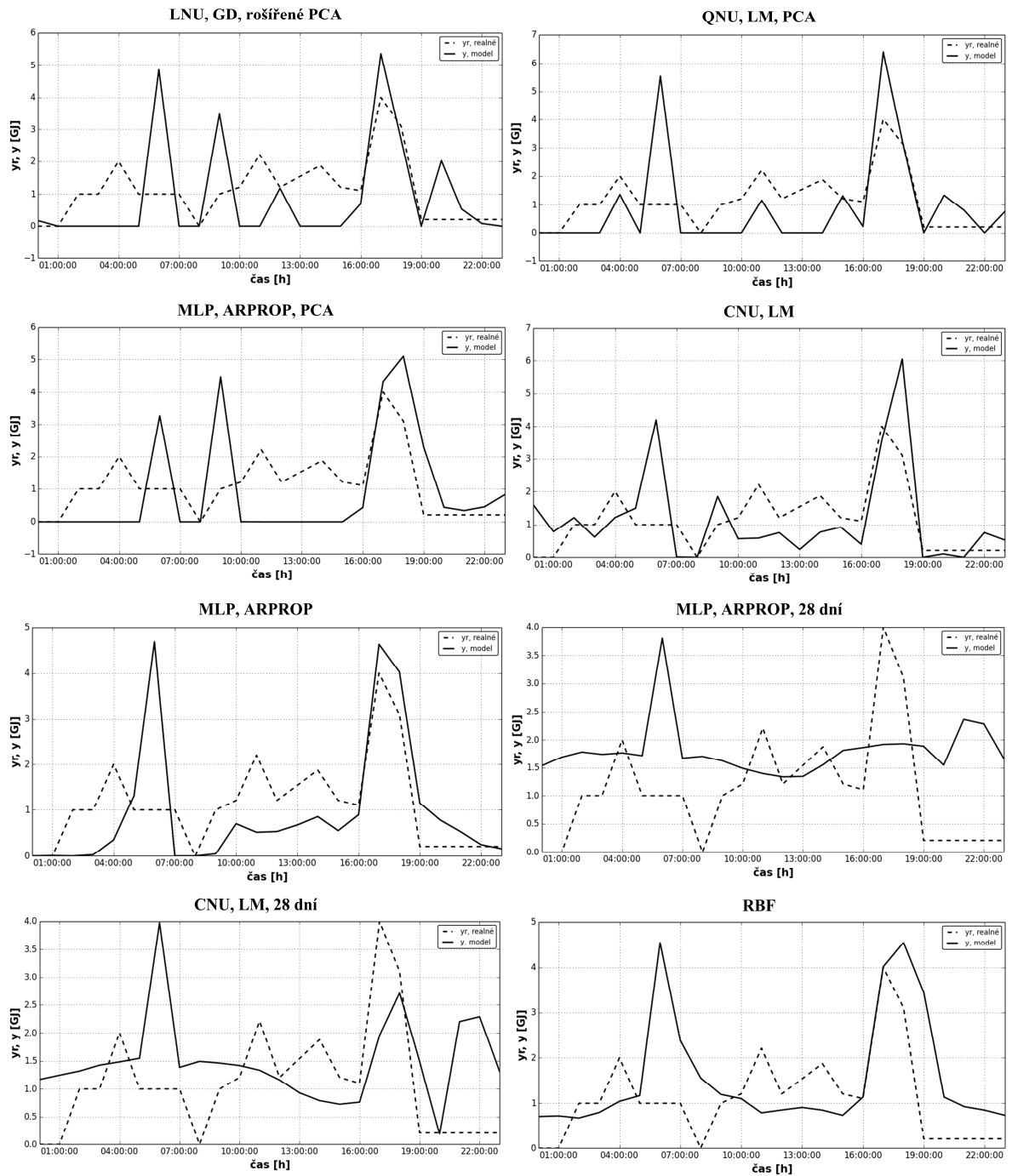
Sobota, 1. 2. 2014



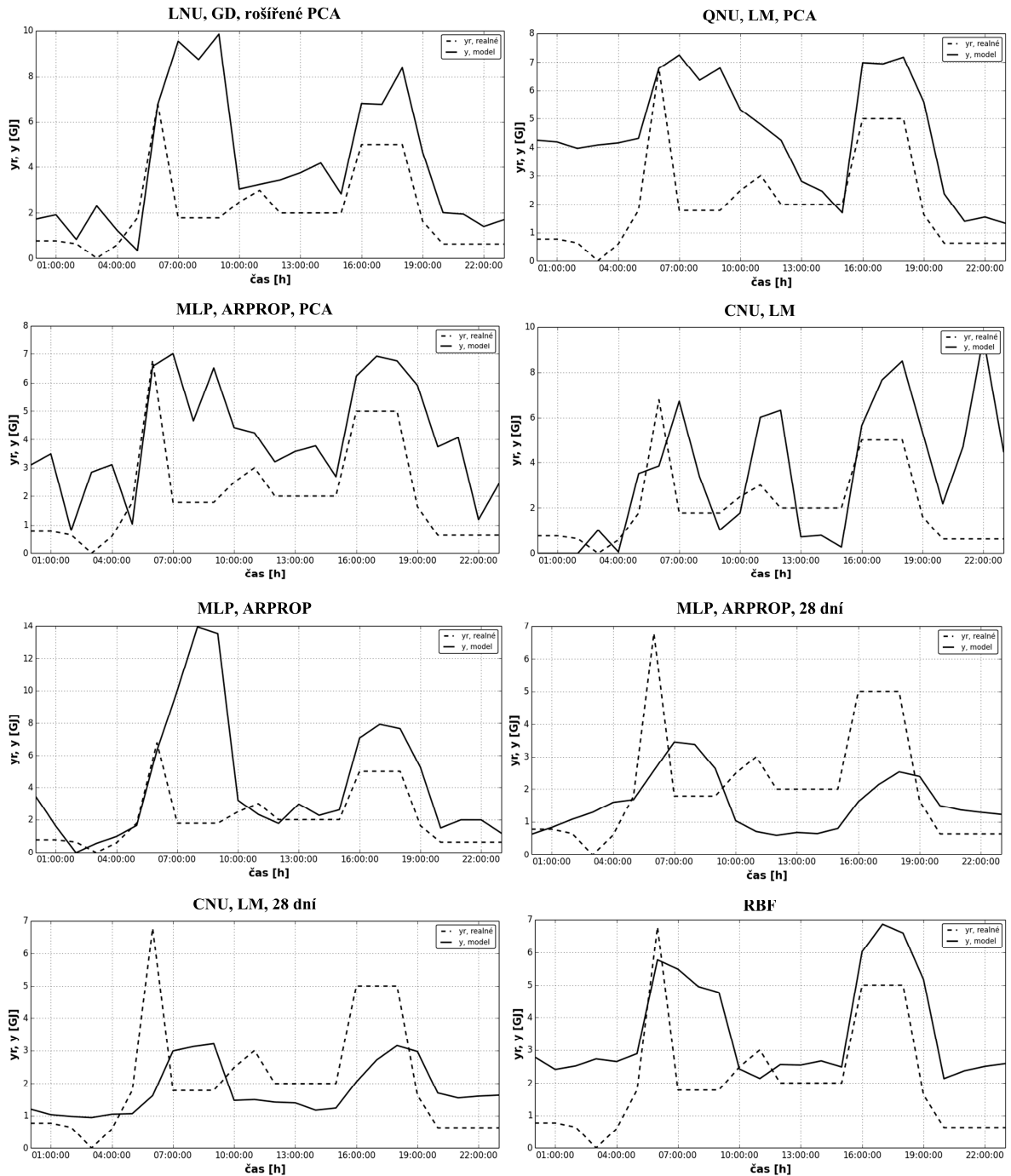
Středa, 26. 3. 2014



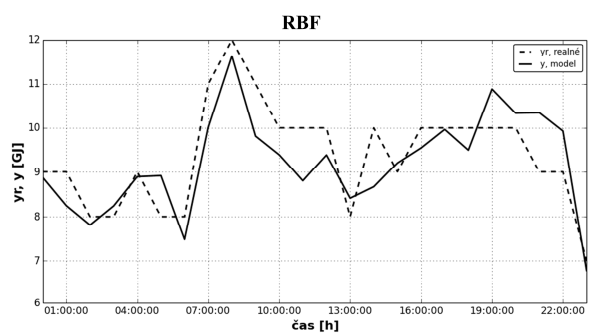
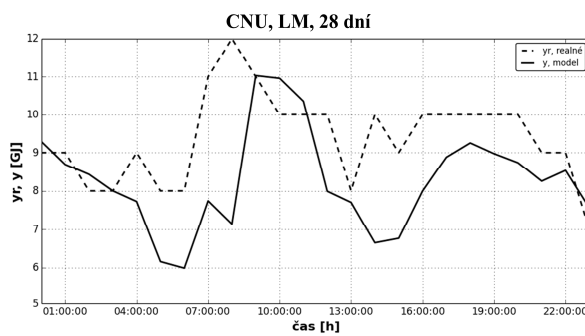
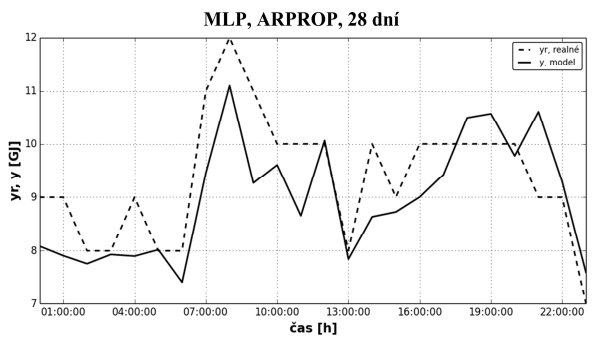
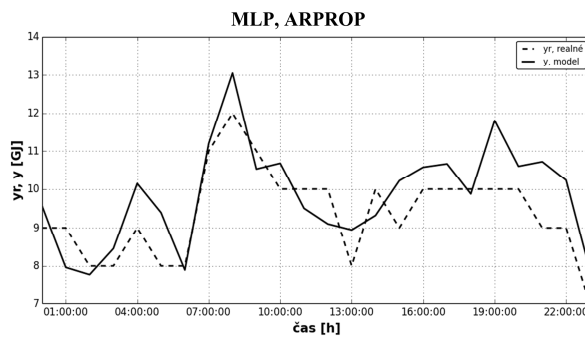
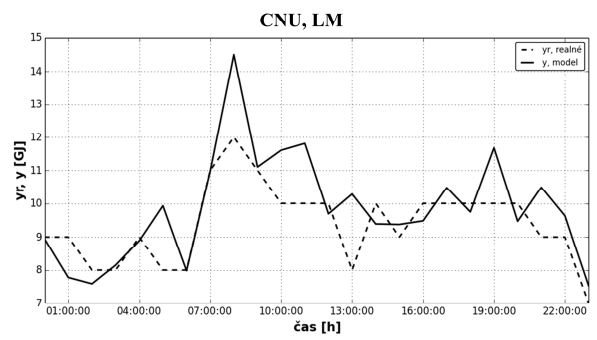
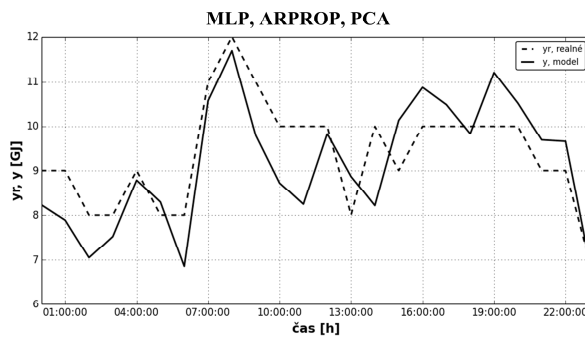
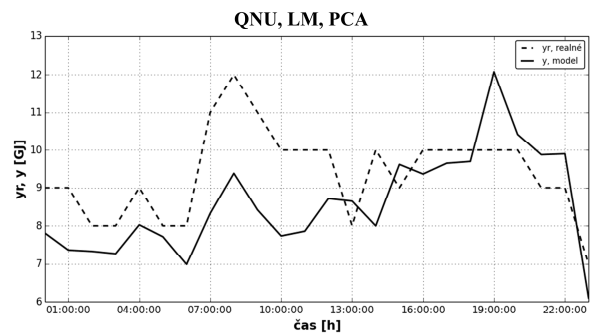
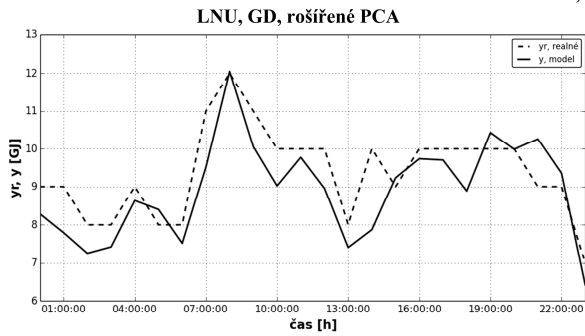
Pondělí, 5. 8. 2013



## Úterý, 3. 9. 2013



Pátek, 15. 11. 2013





### Příloha 3. - Zdrojové kódy použitých neuronových struktur

#### Jednotka LNU učená algoritmem GD, model s rozdělováním dat

```

import numpy as np

def train(mu, RMSEd, epochMAX, u, yr, hours, months, days, day_type, day, month):
# vstupy funkce: rychlost uceni, zadane RMSE, max pocet epoch, vstupni data,
#               vystupni data k nauceni, prislusne hodiny, dny, poradí dne v tydnu
#               typ dne a mesic který je zadany den pro predikci
    nw=len(u[0])+1 # pocet vah / velikost vstupniho vektoru
    N=len(yr) # delka dat
    x=np.ones(nw) # vektor vstupu
    e=np.zeros(N) # chyba modelu
    yn=np.zeros(N) # nauceny prubeh
    wall=np.zeros((N,nw)) # vahy
    wNU=[] # prumerne vahy pro kazdy neuron (GD)
    EPOCHlist=np.zeros(len(range(0,24))) # pro ukladani potrebného počtu epoch k nauceni

    eps=1
    epochMIN=5 # minimalni pocet epoch

    if month == 1: volno=[1] # vyber dni pracovniho volna
    elif month == 4: volno=[21]
    elif month == 5: volno=[1,8]
    elif month == 7: volno=[5,6]
    elif month == 9: volno=[28]
    elif month == 10: volno=[28]
    elif month == 11: volno=[17]
    elif month == 12: volno=[24,25,26]
    else: volno=[]
    for hour in range(0,24): # cyklus iteruje pres hodiny
        if day == 0: # pracovni dny nebo vikendy
            day_start=1
            day_end=5
        else:
            day_start=6
            day_end=7
        list_k=[]
        for i in range(0,N): # samotne rozdeleneni dat, mesic, den, hodina
            if int(hours[i]) == hour and int(months[i]) == month and \
                day_start <= int(day_type[i]) <= day_end:
                list_k.append(i)
            elif int(hours[i]) == hour and int(months[i]) == month and \
                days[i] in volno and day == 1:
                list_k.append(i) # vybrani dni pracovniho volna

        w=np.random.randn(nw)/nw # nahodne pocatecni vahy
        RMSE=[] # vektor pro ulozeni kvadratu chyby pro jednotlivé průchody
        epoch=0
        while True: # uci se dokud není splněno zadane RMSE nebo max pocet epoch
            for k in list_k: # cyklus pro vypocet kazdeho vzorku modelu
                x[1:]=(u[k]) # naplneni vstupniho vektoru
                yn[k]=sum(w*x) # agregacni funkce
                e[k]=yr[k]-yn[k] # chyba nauceného modelu
                nu=mu/(sum(x*x)+eps) # adaptace rychlosti uceni
                w=w+nu*e[k]*x # gradient descent
                wall[k,:]=w # matice historie vah
            RMSE.append(np.sqrt(sum(e[list_k]**2)/len(e[list_k]))) # honoceni dle RMSE
            if RMSE[epoch] <= RMSEd and epoch >= epochMIN or epoch == epochMAX:
                break # podminka ukonceni uceni
            epoch += 1
        print str(RMSE[epoch]) + ' for '+str(epoch)+' epoch - '+str(month)\
            + ' - ' + str(day) + ' - ' + str(hour)

        w_temp=[]
        for weight in range(0,nw): w_temp.append(np.mean(wall[list_k][:,weight]))
        wNU.append(w_temp) # vypocet a ulozeni konecných vah
        EPOCHlist[hour]=epoch

    RMSEt=np.sqrt(sum(e[np.nonzero(e)]**2)/len(e[np.nonzero(e)])) # konecne RMSE
    print 'Total train error ' + str(RMSEt)
    epochs=np.mean(EPOCHlist) # prumerny pocet epoch potrebný k natrenovani
    return(wNU,RMSEt,epochs) # funkce vrati vahy, chybu nauceného modelu a pocet epoch

```

## Jednotka QNU učená algoritmem LM, model s rozdělováním dat

```

import numpy as np

def cx(x): # funkce pro kvadraticke rozsireni vstupniho vektoru
    nx=len(x)
    pom=0
    colx=np.zeros((nx*nx+nx)/2)
    for i in range(nx):
        for j in range(i,nx):
            colx[pom]=x[i]*x[j]
            pom+=1
    return(colx)

def train(muBPTT, RMSEd, epochMAX, u, yr, hours, months, days, day_type, day, month):
# vstupy funkce: rychlost uceni, zadane RMSE, max pocet epoch, vstupni data,
# vystupni data k nauceni, prislusne hodiny, dny, poradí dne v tydnu
# typ dne a mesic který je zadany den pro predikci
    nw=len(u[0])+1 # pocet vah / velikost vstupniho vektoru
    N=len(yr) # delka dat
    yn=np.zeros(N) # nauceny prubeh
    e=np.zeros(N) # chyba modelu
    dydw=np.zeros((N,nw)) # derivate vystupu modelu podle vah
    wNU=[] # vektor pro ukladani vah
    EPOCHlist=np.zeros(len(range(0,24))) # pro ukladani potrebného počtu epoch k nauceni
    if month == 1: volno=[1] # vyber dni pracovniho volna
    elif month == 4: volno=[21]
    elif month == 5: volno=[1,8]
    elif month == 7: volno=[5,6]
    elif month == 9: volno=[28]
    elif month == 10: volno=[28]
    elif month == 11: volno=[17]
    elif month == 12: volno=[24,25,26]
    else: volno=[]
    for hour in range(0,24): # cyklus iteruje pres hodiny
        if day == 0: # pracovni dny nebo vikendy
            day_start=1
            day_end=5
        else:
            day_start=6
            day_end=7
        list_k=[]
        for i in range(0,N): # samotne rozdeleni dat, mesic, den, hodina
            if int(hours[i]) == hour and int(months[i]) == month and \
                day_start <= int(day_type[i]) <= day_end:
                list_k.append(i)
            elif int(hours[i]) == hour and int(months[i]) == month and \
                days[i] in volno and day == 1:
                list_k.append(i) # vybrani dni pracovniho volna
    w=np.random.randn(nw)/nw # nahodne pocatecni vahy
    w[0]=0 # vynulovani vahy prahu neuronu
    RMSE=[] # vektor pro ulozeni kvadratu chyby pro jednotlivé průchody
    epoch=0
    while True: # uci se dokud není splněno zadane RMSE nebo max pocet epoch
        for k in list_k: # cyklus pro vypocet kazdeho vzorku modelu
            x[1:]=u[k] # naplneni vstupniho vektoru
            colx=cx(x) # rozsireni vstupniho vektoru
            yn[k]=np.dot(w,colx) # nauceny vystup modelu
            e[k]=yr[k]-yn[k] # chyba nauceného modelu
            dydw[k]=colx # naplneni matice derivaci
        J=dydw[list_k,:] # vytvoreni Jakobiho matice pro dana data
        Hw=np.dot(J.T,J)+1./muBPTT*(np.diag(np.diag(Hw))) # aproximace Hessovi matice
        dw=np.dot(np.dot(np.linalg.inv(Hw),J.T),e[list_k]) # Levenberg-Marquardt
        w=w+dw # aktualizace vah
        RMSE.append(np.sqrt(sum(e[list_k]**2)/len(e[list_k]))) # honoceni dle RMSE
        if RMSE[epoch] <= RMSEd or epoch == epochMAX:
            break # podminka ukonceni uceni
        epoch += 1
    print str(RMSE[epoch]) + ' pro ' + str(epoch) + ' epoch - ' + str(month) \
        + ' - ' + str(day) + ' - ' + str(hour)

    wNU.append(w) # ulozeni konecných vah
    EPOCHlist[hour]=epoch
    RMSEt=np.sqrt(sum(e[np.nonzero(e)]**2)/len(e[np.nonzero(e)])) # konecne RMSE
    print 'Total train error ' + str(RMSEt)
    epochs=np.mean(EPOCHlist) # prumerny pocet epoch potrebný k natrenovani
    return(wNU,RMSEt,epochs) # funkce vrati vahy, chybu nauceného modelu a pocet epoch

```

## RBF síť s rozdělováním dat

```

import numpy as np

def average(x): # funkce pro vypočet průměru
    assert len(x) > 0
    return float(sum(x)) / len(x)

def pearson(x, y): # funkce pro vypočet korelačního koeficientu
    assert len(x) == len(y)
    n = len(x)
    assert n > 0
    avg_x = average(x)
    avg_y = average(y)
    diffprod = 0
    xdifff2 = 0
    ydifff2 = 0
    for idx in range(n):
        xdifff = x[idx] - avg_x
        ydifff = y[idx] - avg_y
        diffprod += xdifff * ydifff
        xdifff2 += xdifff * xdifff
        ydifff2 += ydifff * ydifff

    return diffprod / np.sqrt(xdifff2 * ydifff2)

def gaussian(z): # Gaussova aktivací funkce
    phi=np.exp(-0.5*z**2)
    return(phi)

# vstupy funkce: vstupní data, výstupní data, vstupní data pro predikovaný den, šířka RBF
def RBF(X,yr,x_test,r): # Radial Basis Function
    meanX=np.mean(X,0) # průměrná hodnota jednotlivých vstupů
    stdX=np.std(X,0) # směrodatná odchylka jednotlivých vstupů
    X=(X-meanX)/stdX # normalizace jednotlivých vstupů
    x_test=(x_test-meanX)/stdX # normalizace vstupu predikovaného dne
    dx=np.sqrt(np.sum((X-x_test)**2,1)) # ||X-x_test||, Euklidovka vzdálenost
    phi=gaussian(dx/r) # průchod aktivací funkce
    yn=sum(phi*yr)/sum(phi) # výstup modelu
    return(yn)

def test(r, u, yr, uT, yrT, time, hours, months, days, day_type):
    # vstupy funkce: šířka RBF, vstupní data, výstupní data, vstupní data predikovaného dne,
    # výstupní data predikovaného dne, hodiny, měsíc, dny, pořadí dne v týdnu
    N=len(yr) # délka dat
    N_test=len(yrT) # délka dat predikovaného dne
    yn=np.zeros(N_test) # načtený průběh
    e=np.zeros(N_test) # chyba modelu
    RMSE=[] # vektor pro uložení kvadrátu chyby
    for test in range(0,N_test): # cyklus iteruje přes hodiny
        month=time[test].month # určení měsíce
        day=time[test].day # určení dne
        weekday=time[test].weekday()+1 # určení dne v týdnu
        hour=time[test].hour # určení hodiny

        if month == 1: volno=[1] # vyber dny pracovního volna
        elif month == 4: volno=[21]
        elif month == 5: volno=[1,8]
        elif month == 7: volno=[5,6]
        elif month == 9: volno=[28]
        elif month == 10: volno=[28]
        elif month == 11: volno=[17]
        elif month == 12: volno=[24,25,26]
        else: volno=[]

        if 1 <= weekday <= 5: # pracovní dny nebo víkendy
            day_start=1
            day_end=5
        else:
            day_start=6
            day_end=7
        list_k=[]
        for i in range(0,N): # samotné rozdělení dat, měsíc, den, hodina
            if int(hours[i]) == hour and int(months[i]) == month and \
                day_start <= int(day_type[i]) <= day_end:

```

```

        list_k.append(i)
    elif int(hours[i]) == hour and int(months[i]) == month and \
          days[i] in volno and 6 <= weekday <= 7:
        list_k.append(i)      # vybrani dni pracovniho volna

    yn[test]=RBF(u[list_k],yr[list_k],uT[test],r) # vystup modelu
    e[test]=yrT[test]-yn[test]                  # chyba modelu
    RMSE.append(np.sqrt(e[test]**2))            # honoceni dle RMSE

RMSEt=np.sqrt(sum(e**2)/len(e))                # konecne RMSE
R2=pearson(yrT,yn)**2                          # konecne R2
print 'Total test error ' + str(RMSEt)
return(yn,e,RMSEt,R2) # funkce vrati prubeh vystupu modelu, chybu modelu, RMSE a R2

```

### Jednotka CNU učená algoritmem LM, model bez rozdělování dat

```

import numpy as np

def cx3(x): # funkce pro kubické rozšíření vstupního vektoru
    nx=len(x)
    pom=0
    colx=np.zeros((nx**3+3*nx**2+2*nx)/6)
    for i in range(nx):
        for j in range(i,nx):
            for k in range(j,nx):
                colx[pom]=x[i]*x[j]*x[k]
                pom+=1
    return(colx)

def train(muBPTT, RMSEd, epochMAX, u, yr):
    # vstupy funkce: rychlost uceni, zadane RMSE, max pocet epoch, vstupni data,
    #                vystupni data k nauceni

    nw=len(u[0])+1          # pocet vah / velikost vstupniho vektoru
    N=len(yr)               # delka dat
    yn=np.zeros(N)         # nauceny prubeh
    e=np.zeros(N)          # chyba modelu
    dydw=np.zeros((N,nw)) # derivace vystupu modelu podle vah
    w=np.random.randn(nw) # nahodne pocatecni vahy
    w[0]=0                  # vynulovani vahy prahu neuronu
    RMSE=[]                 # vektor pro ulozeni kvadratu chyby pro jednotlivé pruchody
    epoch=0

    while True: # uci se dokud není splněno zadane RMSE nebo max počet epoch
        for k in range(N): # cyklus pro výpočet každého vzorku modelu
            x[1:]=u[k]     # naplnění vstupního vektoru
            colx=cx3(x)    # rozšíření vstupního vektoru
            yn[k]=np.dot(w,colx) # nauceny vystup modelu
            e[k]=yr[k]-yn[k] # chyba nauceneho modelu
            dydw[k:]=colx  # naplnění matice derivací
            J=dydw         # vytvoření Jakobiho matice pro dana data
            Hw=np.dot(J.T,J)+1./muBPTT*(np.diag(np.diag(Hw))) # aproximace Hessovi matice
            dw=np.dot(np.dot(np.linalg.inv(Hw),J.T),e) # Levenberg-Marquardt
            w=w+dw # aktualizace vah
            RMSE.append(np.sqrt(sum(e**2)/len(e))) # honoceni dle RMSE
            if RMSE[epoch] <= RMSEd or epoch == epochMAX:
                break # podminka ukonceni uceni
            epoch += 1

    wNU=w # ulozeni konecnych vah
    RMSEt=np.sqrt(sum(e[np.nonzero(e)]**2)/len(e[np.nonzero(e)])) # konecne RMSE
    print 'Total train error ' + str(RMSEt) + ' pro '+str(epoch)+' epoch'
    return(wNU,RMSEt, epoch) # funkce vrati vahy, chybu nauceneho modelu a pocet epoch

```

## MLP síť učená algoritmem ARPROP, model bez rozdělování dat

```

import numpy as np

def phi(ny): # logisticka funkce, sigmoida
    fi=(2./(1+np.exp(-ny)))-1
    return(fi)

def dphi(ny): # derivace sigmoidy
    dfi=2.*np.exp(-ny)/(1+np.exp(-ny))**2
    return(dfi)

def train(nh, RMSEd, epochMAX, u, yr):
# vstupy funkce: pocet neuronu ve skryte vrstve, zadane RMSE, max pocet epoch,
#                vstupni data, vystupni data k nauceni

    eta_minus=0.5 # faktor snizovani rychlosti uceni
    eta_plus=1.4 # faktor zvysovani rychlosti uceni
    deltaMAX=50 # max krok aktualizace vah
    deltaMIN=0 # min krok aktualizace vah
    deltaINIT=0.0125 # pocatecni hodnota kroku aktualizace vah
    q=1.1 # redukcní parametr uceni
    epochMIN=0 # minimalni pocet epoch uceni

    nu=len(u[0]) # pocet vstupu
    nx=1+nu # pocet vah / velikost vstupniho vektoru
    no=1+nh # velikost vystupniho neuronu
    nxo=no # vektor vstupu do vystupniho neuronu
    N=len(yr) # delka dat

    e=np.zeros(N) # chyba modelu
    yn=np.zeros(N) # nauceny prubeh
    x=np.ones(nx) # vektor vstupu
    xo=np.ones(nxo) # vektor vstupu do vystupniho neuronu

    dEdv=np.zeros((N,no)) # derivace chybove funkce podle vah vystupniho neuronu
    dEdW=np.zeros((N,nx,nh)) # derivace chybove funkce podle vah skrytych neuronu
    dxodny=np.zeros(no) # derivace skryte agregacni fce aktivacni funkce
    vNU=[] # ukladani vah vystupniho neuronu
    WNU=[] # ukladani vah skrytych neuronu
    RMSE=[] # vektor pro ulozeni kvadratu chyby pro jednotlivé průchody

    RMSEnan=True # kontrola přetečení vytostu
    while RMSEnan==True: # cyklus ochrany pro přetečení výpočtu
        W=np.random.randn(nx,nh)/nx # náhodně počáteční vahy
        v=np.random.randn(no)/no # náhodně počáteční vahy vystupniho neuronu
        dEdv_all=np.zeros(no) # vektor pro ukládání derivací vystupniho neuronu
        dEdW_all=np.zeros((nx,nh)) # matice pro ukládání derivací skrytych neuronu
        delta_v=np.zeros(no) # vektor kroku aktualizace vah vystupniho vektoru
        delta_W=np.zeros((nx,nh)) # matice kroku aktualizace vah skrytych vektoru
        delta_v.fill(deltaINIT) # naplnění počáteční hodnotou
        delta_W.fill(deltaINIT) # naplnění počáteční hodnotou
        dv=np.zeros(no) # vektor aktualizace vah vystupniho neuronu
        dW=np.zeros((nx,nh)) # matice aktualizace vah skrytych neuronu
        RMSE=[] # vektor pro ulozeni kvadratu chyby
        epoch=0
        while True: # uci se dokud není splněno zadane RMSE nebo max pocet epoch
            for k in range(N): # cyklus pro výpočet každého vzorku modelu
                x[1:]=u[k] # naplnění vstupniho vektoru
                ny=np.dot(W.T,x) # výstup ze skryte vrstvy
                xo[1:]=phi(ny) # aktivacní funkce
                yn[k]=np.dot(v,xo) # nauceny výstup modelu
                e[k]=yr[k]-yn[k] # chyba nauceného modelu
                # ----- adaptace vah neuronu
                dEdv[k,:]=-e[k]*xo # derivace chybove funkce vystupniho neuronu
                dxodny[1:]=dphi(ny) # derivace výstupu skryte vrstvy
                for h in range(1,no): # derivace chybove funkce skrytych neuronu
                    dEdW[k, :,h-1]=-e[k]*v[h]*dxodny[h]*x
                dEdv_old=dEdv_all # ulozeni gradientu z minule epochy pro vystupni neuron
                dEdW_old=dEdW_all # ulozeni gradientu z minule epochy pro skryte neuronny
                dEdv_all=sum(dEdv) # nove gradienty vystupniho neuronu
                dEdW_all=sum(dEdW) # nove gradienty skrytych neuronu
                RMSE.append(np.sqrt(sum(e**2)/len(e))) # RMSE modelu pro každou epochu
            for weight in range(0,no): # cyklus iteruje přes vahy vystupniho neuronu
                if dEdv_old[weight]*dEdv_all[weight] > 0: # když se nezmění směr g
                    delta_v[weight]=np.minimum(delta_v[weight]*eta_plus,deltaMAX)

```

```

        dv[weight]=-np.sign(dEdv_all[weight])*delta_v[weight] # smer zmeny
        v[weight]=v[weight]+dv[weight] # aktualizace vahy
    elif dEdv_old[weight]*dEdv_all[weight] < 0: # kdyz se smer g
        delta_v[weight]=np.maximum(delta_v[weight]*eta_minus,deltaMIN)
        dEdv_all[weight]=0 # umela zmena gradientu vahy na 0
        # v pristim cyklu skoci na else
    else: # kdyz je gradient nulovy
        dv[weight]=-np.sign(dEdv_all[weight])*delta_v[weight] # smer zmeny
        v[weight]=v[weight]+dv[weight] # aktualizace vahy
    if RMSE[epoch] > RMSE[epoch-1]: # pokud se zvetsila celkova chyba modelu
        v=v-(1/2**q)*dv # redukce kroku uceni
    for h in range(0,nh): # cyklus iteruje skryte neurony
        for weight in range(0,nx): # cyklus iteruje pres vahy skrytych neuronu
            if dEdW_old[weight,h]*dEdW_all[weight,h] > 0: # nezmenil se smer g
                delta_W[weight,h]=np.minimum(delta_W[weight,h]*eta_plus,deltaMAX)
                dW[weight,h]=-np.sign(dEdW_all[weight,h])*delta_W[weight,h] # smer
                W[weight,h]=W[weight,h]+dW[weight,h] # aktualizace vahy
            elif dEdW_old[weight,h]*dEdW_all[weight,h] < 0: # zmenil se smer g
                delta_W[weight,h]=np.maximum(delta_W[weight,h]*eta_minus,deltaMIN)
                dEdW_all[weight,h]=0 # umela zmena gradientu vahy na 0
            else: # kdyz je gradient nulovy
                dW[weight,h]=-np.sign(dEdW_all[weight,h])*delta_W[weight,h]
                W[weight,h]=W[weight,h]+dW[weight,h] # aktualizace vahy
        if RMSE[epoch] > RMSE[epoch-1]: # pokud se zvetsila celkova chyba modelu
            W=W-(1/2**q)*dW # redukce kroku uceni

    if RMSE[epoch] <= RMSEd and epoch >= epochMIN or epoch == epochMAX or \
        np.isnan(RMSE[epoch]) == True:
        break # podminka ukonceni uceni
    epoch += 1
    RMSEnan=np.isnan(RMSE[epoch]) # kontrola preteceni pri uceni

vNU=v # ulozeni konecných vah vystupniho neuronu
WNU=W # ulozeni konecných vah skrytych neuronu

RMSEt=np.sqrt(sum(e[np.nonzero(e)]**2)/len(e[np.nonzero(e)])) # konecne RMSE
print 'Total train error ' + str(RMSEt) + ' pro '+str(epoch)+' epoch'
return(WNU,vNU,RMSEt,epoch) # funkce vrati vahy, chybu nauceného modelu a pocet epoch

```

#### Příloha 4. - CD s prací v elektronické podobě, naprogramovanými modely a přílohami

Obsah CD podle názvů složek:

- *naprogramované modely*, všechny popsané modely, včetně optimalizačních a testovacích programů a příslušných výsledků,
- *příloha 1*, schéma centrálního zdroje tepla; informace o zařízení,
- *příloha 2*, průběhy predikce spotřeby energie všech testovacích dní podle všech modelů,
- *python*, instalace použité verze Pythonu včetně potřebných modulů,
- *testování algoritmů*, programy použité pro testování algoritmů
- *text*, práce v elektronické podobě ve formátech docx a pdf.