U N I V E R Z I T A P A R D U B I C E

F a k u l t a e k o n o m i c k o – s p r á v n í

Oceňování finančních opcí pomocí RBF neuronových sítí

Michaela Vlková

Diplomová práce

2011

Univerzita Pardubice
Fakulta ekonomicko-správní
Akademický rok: 2010/2011

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	Bc. Michaela VLKOVÁ, DiS.
Osobní číslo:	E090496
Studijní program:	N6209 Systémové inženýrství a informatika
Studijní obor:	Informatika ve veřejné správě
Název tématu:	Oceňování finančních opcí pomocí RBF neuronových sítí
Zadávající katedra:	Ústav systémového inženýrství a informatiky

Zásady pro vypracování:

Charakteristika současného stavu oceňování finančních opcí.
Analýza vstupních dat a časové řady.
Všeobecná charakteristika RBF neuronových sítí.
Návrh modelu pro oceňování finančních opcí pomocí RBF neuronových sítí.
Verifikace navrženého modelu ve vybraném programovém prostředí.
Popis výsledků a jejich porovnání s dalšími modely na oceňování finančních opcí.

Rozsah grafických prací:

Rozsah pracovní zprávy: **cca 60 stran**

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

Seznam odborné literatury:

OLEJ, V. Modelovanie ekonomických procesov na báze výpočtovej inteligencie. [Vedecká monografia], Miloš Vognar, ISBN 80-903024-9-1, Hradec Králové, Česká republika, 2003, 160s.

NOVÁK, M. a kol. Umělé neuronové sítě teorie a aplikace. Praha: C.H.BECK, 1998. 382 s. ISBN 80-7179-132-6.

SINČÁK, P., ANDREJKOVÁ, G. Neuronové siete Inžiniersky prístup. Košice : TU Košice, 1996.

KVASNIČKA, V. a kol. Úvod do teórie neurónových sietí. Bratislava : IRIS, 1997. 285 s.

HAYKIN, S. S. Neural Network : A Comprehensive Foundation. Upper Saddle River : Prentice-Hall, 1999.

PARK, J., SANDBERG, I. W. Universal Approximation Using Radial-Basis-Function Networks. Neural Computation. 1991, vol. 3, no. 2, pp.246-257.

YAO, J., LI, Y., TAN, CH. L. Option Price Forecasting using Neural Networks. Omega. 2000, vol. 28, no. 4, pp.455-466.

Vedoucí diplomové práce: **Ing. Petr Hájek, Ph.D.**
Ústav systémového inženýrství a informatiky

Datum zadání diplomové práce: **5. října 2010**

Termín odevzdání diplomové práce: **6. května 2011**

L.S.

doc. Ing. Renáta Myšková, Ph.D.
děkanka

doc. Ing. Jiří Křupka, Ph.D.
vedoucí ústavu

V Pardubicích dne 5. října 2010

Prohlášení

Tuto práci jsem vypracovala samostatně. Veškeré literární prameny a informace, které jsem v práci využila, jsou uvedeny v seznamu použité literatury.

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

Souhlasím s prezenčním zpřístupněním své práce v Univerzitní knihovně Univerzity Pardubice.

V Pardubicích dne 6. 5. 2011

Michaela Vlková

Poděkování

Ráda bych touto cestou poděkovala vedoucímu diplomové práce panu Ing. Petru Hájkovi, Ph.D., za odbornou pomoc a za připomínky při vypracovávání této diplomové práce.

Dále děkuji svým rodičům za jejich pochopení, trpělivost a také za to, že mi umožnili studovat.

SOUHRN

Cílem diplomové práce je využít neuronovou síť typu RBF při oceňování finančních opcí a posoudit přesnost určení ceny opce. Práce je členěna do šesti částí. V první části je nastíněna obecná teorie finančních opcí a jejich oceňování pomocí Black-Schole-sova modelu a binomického modelu. Následuje popis problematiky neuronových sítí. V další části jsou rozebrána data, která jsou použita pro trénování a testování neuronové sítě typu RBF, dopředné sítě a pro výpočet dle Black-Schole-sova modelu. V poslední části jsou prezentovány navržené modely v prostředí GUI.

KLÍČOVÁ SLOVA

Oceňování finančních opcí, neuronové sítě typu RBF, návrh grafického uživatelského prostředí

TITLE

Option Pricing by RBF Neural Networks.

SUMMARY

The aim of this thesis is to use neuron network of RBF type in the option pricing and to assess the accuracy of the reached price. The work is divided into six parts. The first part outlines the general theory of financial option and their pricing the Black-Schole model and the binomial model. The first part is followed by a description of neuron networks. The third part analyzes the data, which are used for training and testing neuron network of RBF type , feedforward network and also used for the calculation using the Black-Schole model. The last section presents the proposed models in the GUI environment.

KEYWORDS

Pricing option, neural networks RBF, design of graphic user interface

Obsah

Úvod 9
1 Úvod do oceňování finančních opcí 10
1.1 Základní pojmy 10
1.2 Základní strategie v opčním obchodě 12
1.2.1 Koupě kupní oce - Long Call 12
1.2.2 Prodej kupní opce – Short Call 13
1.2.3 Koupě prodejní opce – Long Put 14
1.2.4 Prodej prodejní opce – Short Put 16
1.3 Opční prémie 16
1.3.1 Vnitřní hodnota opce 17
1.3.2 Časová hodnota 17
1.4 Faktory ceny opcí 18
2 Modely oceňování opcí 20
2.1 Black-Scholes-ův model oceňování opcí 20
2.1.1 Wiener-ův proces 20
2.1.2 Předpoklady Black-Schole- sova modelu 23
2.2 Binomický model oceňování pro akcie nevyplácející dividendy 26
3 Neuronové sítě 30
3.1 Učení neuronové sítě 32
4 Neuronové sítě typu RBF 34
4.1 Topologie neuronové sítě typu RBF 34
4.2 Učení neuronové sítě typu RBF 36
4.2.1 První fáze - určení počtu RBF center 36
4.2.2 Druhá fáze – nalezení center RBF neuronů 37
5 Modelování oceňování finančních opcí 40
5.1 Data 41
5.2 Předzpracování dat 44
5.3 Modelování pomocí neuronové sítě typu RBF 46
5.4 Modelování pomocí dopředné neuronové sítě 50
5.5 Aplikace Black-Scholes-ova modelu 52

5.6 Porovnání modelů pro oceňování finančních Call opcí 53
6 Tvorba grafického uživatelského prostředí..... 54
Závěr..... 59
Seznam použité literatury..... 61
Seznam tabulek..... 64
Seznam obrázků..... 65
Seznam příloh..... 66

Úvod

Finanční opce je cenný papír, jehož majitel má práva, nikoliv povinnosti, nákupu či prodeje podkladového aktiva v určitý čas a za stanovenou cenu. S opcemi se obchoduje především na mimoburzovních trzích. Problémem při obchodování je jak stanovit správně cenu opce. V teorii i praxi lze rozlišit několik základních oceňovacích metod. Neznámější metodou oceňování je Black-Scholes-ův model. Tento matematický model je založen na předpokladu, že je cena opce implicitně dána vývojem ceny podkladového aktiva, přičemž se cena aktiva vyvíjí jako stochastický proces, tj. náhodnou chůzí. Problém správného stanovení ceny opce je poslední dobou řešen pomocí neuronových sítí.

Problematika neuronových sítí patří k dynamicky se rovíjejícím oblastem výpočetní inteligence. Název neuronová síť je odvozen od základního principu, který je analogický s funkcí lidského mozku. V důsledku své struktury dokáže odhalit i závislosti, které nejsou na první pohled patrné. Velkým přínosem neuronových sítí pro finanční oblast je jejich predikční potenciál.

Diplomová práce se zabývá oceňováním finančních opcí pomocí neuronových sítí typu RBF. Je dělena do šesti hlavních kapitol. První kapitola vysvětluje základní pojmy a strategie související s oceňováním finančních opcí. Druhá kapitola seznamuje s modely oceňování opcí. Ve třetí kapitole je rozebrána teorie neuronových sítí, popsány základní stavební prvky neuronové sítě, jejich klasifikace, proces učení. Čtvrtá kapitola blíže charakterizuje neuronové sítě typu RBF. Vysvětluje princip radiálně bazické funkce ve skryté vrstvě neuronové sítě typu RBF. Pátá kapitola je věnována trénování a testování neuronové sítě typu RBF na použitých datech pro oceňování finančních opcí. Výsledky jsou porovnány s dalšími vybranými modely. Šestá kapitola popisuje návrh grafického uživatelského prostředí pro oceňování finančních opcí.

Cílem diplomové práce je charakterizovat současný stav oceňování finančních opcí, stručně charakterizovat neuronové sítě všeobecně a neuronovou síť typu RBF. Dále je cílem práce získat data, která budou podle potřeby předzpracována a použita pro návrh a verifikaci modelů pro oceňování finančních opcí. Následně budou data trénována a testována pomocí neuronové sítě typu RBF v programovém prostředí Matlab. Pro porovnání výsledků budou data dále trénována a testována pomocí dopředné neuronové sítě a pomocí Black-Scholes-ova modelu. Vytvořené modely budou mezi sebou porovnány a vyhodnoceny na základě vypočtených ukazatelů chyb.

1 Úvod do oceňování finančních opcí

V této kapitole jsou vysvětleny základní pojmy používané při oceňování finančních opcí a dále jsou popsány základní strategie v opčních obchodech a faktory ceny opcí.

1.1 Základní pojmy

Opce

Opce jsou "deriváty", což znamená, že jejich hodnota je odvozená od něčeho jiného, např. od akcie, akciového indexu nebo futures kontraktu. Základ, od kterého opci odvozujeme, se nazývá podkladové aktivum. Opce je vlastně právo koupit nebo prodat toto podkladové aktivum v nějaké době za předem stanovenou cenu. [7]

Opční obchody

Opční obchody se zařazují mezi tzv. podmíněné termínované obchody. Jejich základní odlišnost od nepodmíněných (pevných) termínových obchodů spočívá v tom, že pouze jeden z partnerů má povinnost na požádání sjednaný obchod splnit, zatímco druhý má možnost volby, to znamená požadovat plnění obchodu nebo od něj ustoupit. [4]

Opční smlouva

Opční smlouva obsahuje dohodu mezi dvěma subjekty – kupujícím a prodávajícím opce. Kupující (majitel) opce získává koupí opce za opční prémii (opční cenu) následující práva: koupit nebo prodat určité pevně sjednané množství stanoveného podkladového aktiva za předem pevně dohodnutou cenu v předem pevně stanovený den nebo kdykoliv ve lhůtě do tohoto dne. [4]

Spotová cena

Spotová cena je aktuální cena podkladového aktiva, za kterou může být podkladové aktivum v určený časový okamžik nakoupeno či prodáno. Spotová cena bývá označována jako promptní cena. [14]

Realizační cena (Strike price)

Realizační cena je fixní cena, za kterou má držitel opce právo koupit podkladové aktivum jedná-li se o Call opci nebo prodat podkladové aktivum jedná-li se o Put opci. Tato cena se v průběhu životnosti opce nemění a od této ceny se odvíjí její vnitřní hodnota. [14]

Volatilita ceny (Volatility)

Volatilita je vyjádřením častosti a velikosti výkyvů v pohybu ceny podkladového aktiva. Je dána rozptylem ceny podkladového aktiva za určité časové období s počtem těchto období. [12]

Opční prémie (Opční cena, Premium)

Opční prémie je cena opce, kterou kupující opce zaplatí za dané opční právo. [7]

Bod zvratu

Bod zvratu je bod, ve kterém je zisk roven přesně nule a opční strategie se dostává ze ztráty do zisku. [7]

Opce v penězích

Opce je v penězích, tzv. „in-the-money“, pokud má nějakou vnitřní hodnotu. Např. Call opce je in-the-money, pokud je aktuální spotová cena podkladového aktiva vyšší než realizační cena opce. Investor by při uplatnění takovéto opce obdržel od vypisovatele opce rozdíl mezi aktuální spotovou cenou podkladového aktiva a realizační cenou. [1]

Opce na penězích

Opce je na penězích, tedy „at-the-money“, pokud se spotová cena podkladového aktiva rovná nebo se téměř rovná realizační ceně opce. [1]

Opce mimo peníze

Opce mimo peníze, tedy „out-the-money“, je opce, která nemá žádnou vnitřní hodnotu. Od opce at-the-money se liší tím, že spotová cena se nerovná realizační ceně. Např. u Call opce to znamená, že realizační cena je vyšší než spotová cena a Put opce je naopak realizační cena nižší než spotová cena. V obou případech se nevyplatí opci realizovat, avšak opce není úplně bezcenná. [1]

1.2 Základní strategie v opčním obchodě

Pomocí opcí je možné vytvářet celou řadu strategií, které se hodí pro různé situace na trhu. Opce vydělávají jednak v prosperujícím trhu, stejně jako v trhu, který se vůbec nehýbe. Realizace zisku je možná i v takové situaci, kdy není nic zřejmé o dalším směru trhu, nicméně existuje velká pravděpodobnost, že cena podkladového aktiva (spotová cena) výrazně klesne dolů nebo stoupne nahoru.

Využití opcí z hlediska jednotlivých ekonomických subjektů závisí na jejich očekávání budoucího vývoje, které vedou právě k využití opcí. Je třeba upozornit na to, že analýza ziskovosti a analýza ztrátovosti jednotlivých strategií vychází pouze z možnosti využití či nevyužití opce. V praxi k reálnému využití (tj. fyzickému plnění) opce dochází jen zřídka. Obchody jsou vyrovnávány prostřednictvím zakoupení (prodeje) zrcadlové pozice. Ve skutečnosti jsou zisky nebo ztráty dány rozdílem nákladů a výnosů na otevření a uzavření pozice. Nicméně základní princip se nemění.

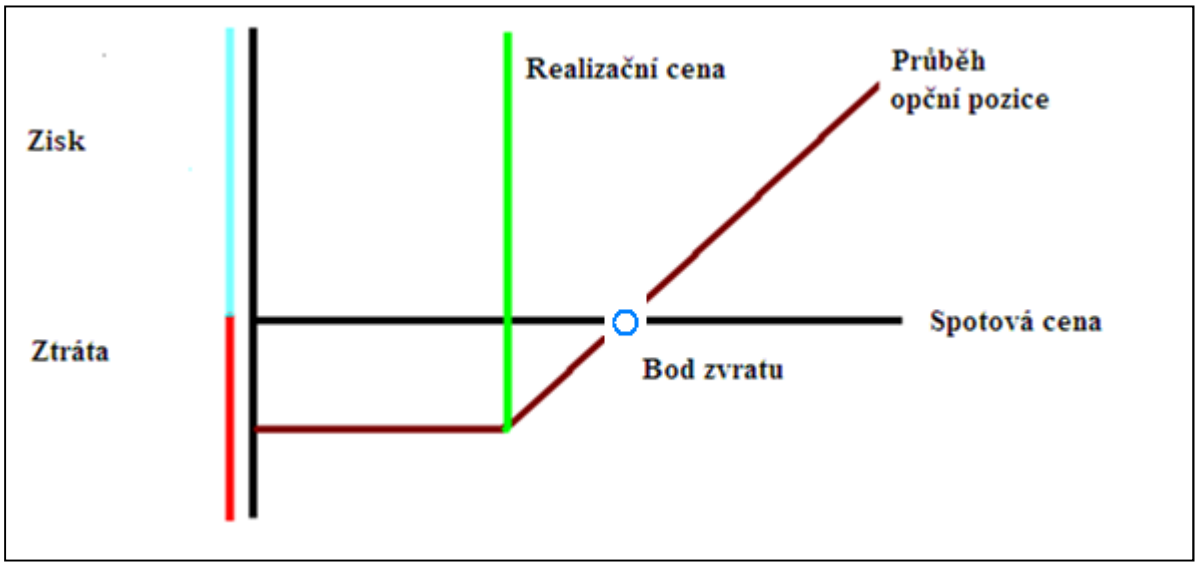
1.2.1 Koupě kupní oce - Long Call

Subjekt, který nakoupil kupní opci, má právo koupit určité množství podkladového aktiva za předem danou realizační cenu *X*. Za zakoupení pozice má kupující povinnost zaplatit prodávajícímu opční prémii *O*. Opční prémie *O* představuje jeho nejvyšší možnou ztrátu z této strategie. Naproti tomu pozice Log Call dává majiteli teoreticky neomezený ziskový potenciál, který se zvyšuje s růstem ceny podkladového aktiva [25].

Následující Obr. 1 zobrazuje zisk a ztrátu z pozice Long Call. Spotová cena *S* se nachází na horizontální ose a jak je vidět, tato cena podkladového aktiva roste směrem zleva doprava. Vertikální osa zobrazuje zisk nebo ztrátu z dané pozice. Tmavě červenou barvou je vyznačen průběh opční pozice. Počátek opční pozice se nachází v záporných hodnotách a to proto, že za nákup Call opce se musí zaplatit opční prémie *O*. Zaplacená opční prémie *O*, představuje maximální možnou ztrátu z dané pozice. Od úrovně realizační ceny *X*, začíná růst zisk z této pozice. Je třeba připomenout, že realizační cena *X* je cena, za kterou bude subjekt moci v budoucnu podkladové aktivum nakoupit. Bod zvratu identifikuje cenu podkladového aktiva při expiraci, při které bude pozice v nulovém zisku [3].

Na základě tohoto bodu se určuje vývoj zisku a ztráty z dané opční pozice. Je důležité vědět, od které ceny podkladového aktiva bude pozice zisková či ztrátová. K určení bodu zvratu je třeba znát realizační cenu *X* a opční prémii *O*. Opční prémie musí být vyjádřena v bodech. U Call opcí se bod zvratu získá součtem realizační ceny *X* a opční prémie O. U Put opcí

se vypočítá jako rozdíl mezi opční prémií O a realizační cenou opce *X*. Pokud je spotová cena *S* vyšší než realizační cena *X*, je výhodné Call opci využít. Tab. 1 uvádí, kdy majitel pozice Long Call dosahuje v takové situaci čistého zisku *CP* a kdy omezené ztráty *OZ*.



Obr. 1 - Pozice Long Call, zdroj: [25]

Tab. 1 - Využití Call opce, zdroj: [14]

Čistý zisk	Omezená ztráta
$S - X > O$	$S - X < O$
$tj. když : S > X + O$	$tj. když\ S \epsilon (S; X + O)$
$CP = S - X - O$	$OZ = O - (S - X)$

Pokud je spotová cena *S* nižší než realizační *X*, majitel dosahuje maximální možné ztráty *MZ* ve výši zaplacené opční prémie *O*. Proto se doporučuje nechat opci propadnout, jelikož zakoupení takového aktiva přímo na trhu bude výhodnější než prostřednictvím opce.

1.2.2 Prodej kupní opce – Short Call

Pozice Short Call je zrcadlovou pozicí k Long Call. Subjekt v pozici Short Call prodal opci, a proto má povinnost na požádání majitele opce prodat za realizační cenu *X* příslušné podkladové aktivum. Za tuto povinnost vyplývající z prodeje opce inkasuje opční prémii *O* [4].

Maximálního zisku *MP* subjekt Short Call pozice dosáhne v případě, kdy kupující (majitel) opce opční právo nevyužije a nechává jej propadnout. K tomu dochází tehdy, jestliže zakoupení podkladového aktiva na trhu je levnější než prostřednictvím opce, tj. kdy jeho spotová cena *S* je nižší než realizační cena *X*. Maximální zisk je limitován výší opční prémie *O*.

V případě, že kupující opce využije opci, prodávající dosahuje určité ztráty. Pokud tato ztráta je nižší než inkasovaná opční prémie *O*, jedná se o omezenou ztrátu *OZ*. Výše omezené ztráty *OZ* je vypočtena rozdílem *O – (S-X)*. O čisté ztrátě *CZ* se hovoří v případě, když inkasovaná opční prémie *O* nepokryje náklady vyplývající z využití opce, tj. $(S - X) > O$. Ztrátu lze lehce předpovídat, pokud je spotová cena *S* podkladového aktiva vyšší, než součet realizační ceny *X* a opční prémie O. Výše ztráty se vypočítá rozdílem $O - (S - X)$. Všechny jmenované situace jsou uvedeny v Tab. 2.

Tab. 2 - Pozice Short Call, zdroj: [14]

Maximální zisk	Omezená ztráta	Čistá ztráta
$X > S$	$X < S$	$X < S$
	$S - X < O$	$S - X > O$
	$S \epsilon (S, X + O)$	$S > X + O$
$CP = O$	$OZ = O - (S - X)$	$CZ = O - (S - X)$

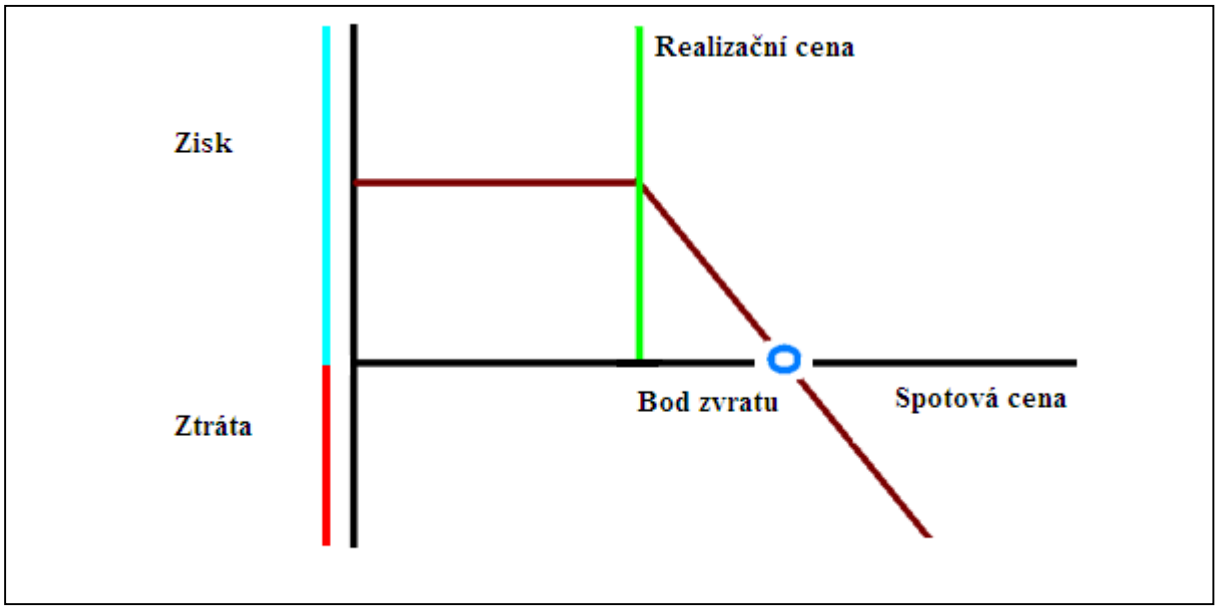
Následující Obr. 2 zobrazuje výše uvedené situace, které majitel pozice Short Call může dosáhnout.

1.2.3 Koupě prodejní opce – Long Put

Subjekt v pozici Long Put má právo prodat za předem stanovenou realizační cenu *X* podkladové aktivum. Za zakoupení této pozice zaplatí opční prémii *O*. Pokud spotová cena podkladového aktiva *S* klesá, roste tak výhodnost Put opce [4]. Opci je tedy výhodné využít, pokud aktuální spotová cena *S* podkladového aktiva je nižší než realizační cena *X*. Prodej podkladového aktiva prostřednictvím opce je výhodnější než přímo na spotovém trhu.

Pokud subjekt uplatní opci, může dosáhnout čistého zisku *CP* nebo omezené ztráty *OZ*. Čistý zisk *CP* dosáhne tehdy, pokud spotová cena *S* podkladového aktiva je nižší než rozdíl

mezi realizační cenou *X* a zaplacenou opční prémií *O*. Jeho výše je tak dána rozdílem *(X – S) – O*. Zisk bude maximální v případě nulové spotové ceny *S*.



Obr. 2 - Pozice Short Call, zdroj: [25]

Omezené ztráty *OZ* dosahuje majitel prodejní opce, pokud je zaplacená opční prémie *O* vyšší, než zisk z uplatnění opce, tj. *O > (X – S)*. Výše omezené ztráty se rovná rozdílu *O - (X - S)*.

Subjekt dosahuje maximální ztráty *MZ* v případě, kdy je spotová cena *S* podkladového aktiva vyšší než realizační *X*. Majitel opce nechává opci propadnout, protože je pro něj výhodnější prodat podkladové aktivum na trhu než jej prodat prostřednictvím opce. Maximální ztráta *MZ* je limitována výší zaplacené opční prémie *O*. Výše uvedené situace jsou shrnuty do přehledné Tab. 3.

Tab. 3 – Pozice Long Put, zdroj: [14]

Čistý zisk	Omezená ztráta	Maximální ztráta
$když\ S < X$	$když\ S < X$	$když\ S > X$
$pokud\ S < X - O$	$pokud\ 0 > (X - S)$	
$CP = (X - S) - O$	$OZ = O - (X - S)$	$MZ = O$

1.2.4 Prodej prodejní opce – Short Put

Subjekt v pozici Short Put prodal Put opci. Jeho povinností je na požádání majitele opce odkoupit za předem stanovenou realizační cenu *X* příslušné podkladové aktivum. Z této povinnosti mu vyplývá právo inkasovat opční prémii *O* [4].

Subjekt v pozici Short Put může dosáhnout maximálního zisku *MP*, omezeného zisku *OP* anebo ztráty *Z*. Maximální zisk *MP* je limitován výši inkasované opční prémie *O*. Jedná se o situaci, kdy majitel opce nechává opci propadnout, protože prodej podkladového aktiva přímo na trhu je pro něj výhodnější než prostřednictvím opce, tj. $S > X$. V případě, kdy je opce majitelem uplatněna, může vypisovatel opce dosáhnout omezeného zisku *OP* nebo ztráty *Z*. Omezeného zisku *OP* dosahuje v případě, kdy inkasovaná opční prémie *O* nepokryje ztrátu vyplývající z uplatnění opce. Ztráta *Z* je dosažena v případě, kdy je cena podkladového aktiva nulová. Výše uvedené situaci zpřehledňuje Tab. 4.

Tab. 4 - Pozice Short put, zdroj: [14]

Maximální zisk	Omezený zisk	Ztráta
$S > X$	$když\ S < X - O$	$když\ S = 0$
	$a\ pokud\ X - S > O$	
$MP = O$	$OP = O - (X - S)$	$Z = (X - S) - O$

1.3 Opční prémie

Opční prémie *O* je cena, za kterou jsou opce obchodovány, to znamená cena, kterou platí kupující opci prodávajícímu.[1] Představuje tedy (spolu s poplatky a provizemi) náklady na zakoupení opčního práva. Jak již bylo uvedeno, výše opční prémie *O* zároveň limituje maximální výši ztráty pro kupujícího opce a naopak maximálně možný zisk pro prodávajícího opce. Opční prémie *O* se skládá ze dvou komponent, z vnitřní a časové hodnoty[4].

[1] Pro opční prémii se využívá jako synonymum i termín opční cena nebo cena opce. Lze se ovšem setkat také s tím, že se oba termíny odlišují a pod opční prémií se rozumí pouze diference mezi opční cenou a vnitřní hodnotou opce, tzn. de facto časová hodnota opce.

1.3.1 Vnitřní hodnota opce

Vnitřní hodnota opce ukazuje na výhodnost okamžitého využití opce. Jedná se tedy o zisk, který by majitel opce docílil jejím okamžitým využitím, tzn. koupí či prodejem podkladového aktiva za realizační cenu X, a současným kompenzujícím obchodem na burze, tj. prodejem či koupí podkladového aktiva na burze. Opce má tedy vnitřní hodnotu v případě, že lze takovou ziskovou transakci provést [5].

Vnitřní hodnota pro Call opci je definována vztahem [1]:

$$max\ [S - X; 0]. \tag{1.1}$$

Vnitřní hodnota pro Put opci je definována vztahem [1]:

$$max\ \ [X - S; 0]. \tag{1.2}$$

U vnitřní hodnoty Call opce, lze předpokládat, že je rostoucí funkcí spotové ceny S pro $S > X$. Analogicky lze předpokládat, že vnitřní hodnota Put opce je naopak klesající lineární funkcí pro S v intervalu $(0, X)$.

Mezi faktory, které ovlivňují výši opční prémie, patří spotová cena S podkladového aktiva a realizační cena X a platí následující [5]:

- roste-li spotová cena S, roste opční prémie C_t u Call opce,
- roste-li spotová cena S, klesá opční prémie P_t u Put opce.

1.3.2 Časová hodnota

Podle [5] lze o časové hodnotě obecně říci, „že se v ní odráží vliv nabídky a poptávky po dané opci“. Časová hodnota opce představuje riziko změny spotové ceny S podkladového aktiva v průběhu doby splatnosti opce. Toto riziko se s ubývajícím časem do splatnosti opce snižuje a v den splatnosti je nulové [7]. Na základě vysvětlení vnitřní a časové hodnoty opce je zřejmé, že opce bude mít vždy kladnou nebo nulovou cenu ke dni splatnosti opce. Mezi faktory, které ovlivňují časovou hodnotu opce, patří realizační cena X, spotová cena S, doba do splatnosti opce t, tržní úroková míra r, volatilita ceny podkladového aktiva σ.

1.4 Faktory ceny opcí

V kapitole 1.3.2 se uvádí faktory, na kterých závisí cena opce *O*. Nástroje pro měření míry citlivosti opcí na těchto faktorech jsou pojmenovány podle řecké abecedy. Tyto nástroje by neměly být opomenuty při stanovení opční strategie. [14]

- Delta opce ΔO – závislost ceny opce *O* k pohybu ceny podkladového aktiva *S*,
- Gamma opce ΓO - závislost ceny opce *O* na pohybu delty,
- Theta opce θO – závislost ceny opce *O* na čase *t*,
- Vega opce ϑO – závislost ceny opce *O* na implikované volatilitě σ,
- Rho opce ρO – závislost ceny opce *O* na úrocích.

Delta opce ΔO měří závislost změny opční prémie *O* na malé změny spotové ceny *S*. Její hodnota vyjadřuje, o kolik procent se změní opční prémie *O*, pokud se cena podkladového aktiva *S* změní o jednotku za jinak nezměněných podmínek [5].

Ukazatel ΔO je definován jako první parciální derivace opční prémie *O* podle spotové ceny *S*. Podle [28] je pro

Call opci
$$\Delta O = \frac{\partial O}{\partial S} = N(d), \tag{1.3}$$

Put opci
$$\Delta O = \frac{\partial O}{\partial S} = N(d\ ) - 1. \tag{1.4}$$

Ukazatel ΓO vyjadřuje rychlost změny delty, když se spotová cena *S* podkladového aktiva změní o bod. Jako jediná nevyjadřuje závislost změny opční prémie *O* na některém z faktorů. Matematicky je definována jako druhá derivace opční prémie *O* podle spotové ceny *S* [5]:

$$\Gamma O = \frac{\partial^2 O}{\partial S^2}. \tag{1.5}$$

Ukazatel θO vypovídá o změně opční prémie v závislosti na změně doby do splatnosti opce *T*. Vyjadřuje, o kolik se změní opční prémie *O* při snížení doby do splatnosti o jeden den. Její hodnota nabývá záporných hodnot a to proto, že při zkrácení doby do splatnosti opce se snižuje časová hodnota opce a následně i opční prémie[5]:

$$\theta O = -\frac{\partial O}{\partial T}. \tag{1.6}$$

Ukazatel ϑO vyjadřuje závislost opční prémie O na změnách volatility σ ceny podkladového aktiva. Její hodnota vypovídá o výši změny v opční prémii při změně volatility podkladového aktiva o jeden procentní bod[5]:

$$\vartheta O = \frac{\partial O}{\partial \sigma}. \tag{1.7}$$

Rho ρO vyjadřuje změnu opční prémie v závislosti na změně úrokové sazby r [5]. Říká, o kolik se změní opční prémie při změně úrokové míry o jednotku:

$$\rho O = \frac{\partial O}{\partial r}. \tag{1.8}$$

2 Modely oceňování opcí

Modely oceňování opcí jsou vhodné k určování rovnovážných opčních prémií *O* v čase *t*. Obecně pracují s hlavními faktory, které ovlivňují opční prémii *O*, tj spotová cena podkladového aktiva *S*, realizační cena opce *X*, doba do splatnosti opce *t*, volatilita očekávaných výnosů podkladového aktiva σ, bezriziková úroková míra *r*.

V teorii i v praxi lze rozdělit řadu základních oceňovacích metod [9] např.

- analytické – Black-Scholes-ův model, Black-ův model,
- numerické – binomický model, trinomický model a metoda konečných prvků
- simulační – metoda Monte-Carlo, Quasi-Monte-Carlo.

K ocenění opcí se přistupuje diskrétním či spojitým způsobem. Tato práce se věnuje základním modelům, tj. spojitému Black-Scholes-ovu modelu a diskrétnímu binomickému modelu.

2.1 Black-Scholes-ův model oceňování opcí

Black–Scholes-ův model oceňování opcí poskytuje analytické řešení stanovení ceny vybraných typů opcí. Nespojitý proces je nahrazen spojitým a to za předpokladu, že časový úsek je rozdělen na nekonečně mnoho malých podúseků. Popis takovéhoto náhodného vývoje ceny aktiva vychází z teorie stochastických procesů.

Pro řešení řady praktických úloh jako je oceňování finančních instrumentů, odhadu budoucího vývoje ekonomických veličin (inflace, kurz, atd.), ale i pro řízení rizik, využívá moderní finanční matematika stochastického počtu. Pro zvládnutí této náročné problematiky je třeba znát teorii stochastických procesů.

Při odvozování základního tvaru Black-Scholes-ova vzorce pro opční prémii *O* evropských opcí na akcii nevyplácející dividendy se využívá model založený na speciálním typu Markov-ova procesu, tzv. Wiener-ův proces a jeho zevšeobecněný Brown-ův pohyb.[31]

2.1.1 Wiener-ův proces

Nejčastěji využívaným markovským procesem je Wiener-ův proces. Finanční ekonomie ho používá na modelování náhodné složky vývoje cen podkladového aktiva. Wiener-ův proces nachází uplatnění nejen v ekonomii, ale i ve fyzice pod označením Brown-ův pohyb.

Ten je z matematického hlediska považován za stochastickou veličinu, která popisuje neustálý a neuspořádaný pohyb molekul [30].

Protože se ceny aktiv na finančních trzích podle teorie dokonalých trhů chovají zcela náhodně a nezávisle na předchozím vývoji je Wiener-ův proces ideálním nástrojem popisující chování cen aktiv.

Wiener-ův proces je definován na intervale < 0, ∞) s následujícími vlastnostmi [31]:

- $W_0 = 0$; tzn., že proces začíná nulovou hodnotou;
- W (t); t > 0; W (0) = 0; tzn., že náhodný proces se spojitým časem;
- $W_{t2} - W_{t1} \approx N(0, t_2 - t_1)$ pro každé $t_2 > t_1$; tzn., že přírůstky jsou normálně rozdělené se střední hodnotou rovnou 0 a s rozptylem rovným $t_2 - t_1$. Pokud se zvolí $t_1 = 0$, potom $W_t \approx N(0,t)$, rozptyl je přímoúměrný času;
- náhodné proměnné $W_{t4} - W_{t3}$ a $W_{t2} - W_{t1}$ jsou nezávislé pro $t_4 > t_3 \geq t_2 > t_1$.

Aby se nějaká veličina řídila Wiener-ovým procesem, musí splňovat dva předpoklady. Tyto předpoklady lze vyjádřit pomocí přírůstku Δz této veličiny v krátkých časových intervalech Δt [31]:

$$\Delta z = z(t + \Delta t) - z(t). \tag{2.1}$$

Následující úvahy a odvození vychází z [31]. Pokud intervaly Δt přecházejí na intervaly *dt* nekonečně malé délky, pak uvedené předpoklady mají následující tvar:

(1) Mezi libovolnými odpovídajícími přírůstky Δ*t* a Δz platí vztah

$$\Delta z = \varepsilon\sqrt{\Delta t}. \tag{2.2}$$

kde ε je náhodná veličina se standardizovaným normálním rozdělením *N (0;1)*.

(2) Přírůstky Δz pro libovolné disjunktní časové přírůstky Δ*t* jsou navzájem nezávislé.

Z předpokladu (1) plyne, že přírůstek Δz má normální rozdělení s momenty [31]:

$$E(\Delta z) = 0, \tag{2.3}$$

$$var(\Delta z) = \Delta t, \tag{2.4}$$

$$\sigma(\Delta z) = \sqrt{\Delta t}. \tag{2.5}$$

A dále z předpokladu (2) plyne, že Wiener-ův proces je skutečně Mark-ovským procesem. V přírůstcích typu $\Delta y/\Delta x$ se provádí limitní přechod $\Delta x \rightarrow 0$ k hodnotám dy/dx. To platí také pro Wiener-ův proces při $\Delta t \rightarrow 0$:

$$dz = \varepsilon\sqrt{dt}. \tag{2.6}$$

Zobecněný Wiener-ův proces x lze potom definovat pomocí Wiener-ova procesu z jako

$$dx = a dt + b dz, \tag{2.7}$$

kde a je trendový koeficient a b je difúzní koeficient. Význam těchto koeficientů lze demonstrovat na následujícím příkladu kde $b = 0$, tj. $dx = adt$, potom při dané počáteční hodnotě $x(0) = x_0$ je řešením takovéto deterministické diferenciální rovnice zřejmě přímka $x = x_0 + at$ se směrnicí a. Pokud b není rovno nule, je na tuto přímku nabalena stochastická složka (šum) ve výši b-násobku Wiener-ova procesu. Pro diskrétní časové přírůstky Δt, lze rovnici $dx = a dt + b dz$ zapsat jako [31]:

$$\Delta x = a\Delta t + b\varepsilon\sqrt{\Delta t}. \tag{2.8}$$

A platí, že přírůstek Δx má zřejmě normální rozdělení s momenty[31]:

$$E(\Delta x) = a\Delta t, \tag{2.9}$$

$$var(\Delta x) = b^2\Delta t, \tag{2.10}$$

$$\sigma(\Delta x) = b\sqrt{\Delta t}. \tag{2.11}$$

Z uvedeného lze konstatovat, že koeficient a představuje nárůst trendu a koeficient b nárůst směrodatné odchylky zobecněného Wiener-ova procesu během časové jednotky. Zobecněný Wiener-ův proces x s a (trendovým) a b (difúzním) koeficientem, které nejsou konstantní v čase t, ale závisejí na hodnotě tohoto procesu x a na čase t, se někdy nazývá difúzní proces s trendovým koeficientem $a(x, t)$ a difúzním koeficientem $b(x, t)$:

$$dx = a(x,t)dt + b(x,t)dz. \tag{2.12}$$

Difúzní proces se označuje jako Itoovo lemma, které také souvisí s problematikou oceňování finančních instrumentů.

Uvažujme Itoovo lemma, tj. $dx = adt + bdz$, kde $dz = \varepsilon\sqrt{dt}$, je Wiener-ův proces. Přičemž se předpokládá, že náhodná proměnná **ε** má normální rozdělení *N (0,1)* [9]. Dále se uvažuje spojitá a diferencovatelná funkci *G* proměnných *x* a *t* s parciálními derivacemi potřebného stupně. Potom se dá dokázat[2], že platí [31]:

$$dG = \left(a\frac{\partial G}{\partial x} + \frac{\partial G}{\partial t} + \frac{1}{2}\, b^2 \frac{\partial^2 G}{\partial x^2}\right)dt + \; b\frac{\partial G}{\partial x}\, dz. \tag{2.13}$$

2.1.2 Předpoklady Black-Schole- sova modelu

Pro Black – Scholes-ův model platí následující předpoklady [35]:

1. Ceny akcií se mění spojitě. Pohyb ceny od jedné akcie k druhé musí zaznamenat všechny hodnoty ležící mezi těmito dvěma akciemi.
2. Trh je dokonalý, neexistují žádné transakční náklady, daně ani poplatky z obchodování, likvidita aktiv je okamžitá a absolutní.
3. Oceňují se evropské opce, jejichž podkladová akcie neplatí dividendu (pro americkou prodejní opci není použitelný).
4. Na trhu existuje jediná bezriziková úroková míra *r* na dobu do splatnosti opce
5. Podkladové aktivum a opce je možné libovolně dělit.
6. Trh je efektivní, neexistuje možnost arbitráže.
7. Nedochází k vyplácení dividend z podkladového aktiva.
8. Dynamika ceny podkladového aktiva se řídí Wienerovým procesem.

Black-Scholes-ův vzorec vyjadřuje matematický zápis opční prémii *O* jako funkci pěti proměnných:

- cena podkladového aktiva *S*,
- realizační cena akcie *X*,
- doba do splatnosti opce *t*,
- volatilita ceny akcie *σ*,
- bezriziková úroková míra *r*.

[2] Přesnější odvození a důkaz v [31]

Opční prémie *O* stanovená výpočetně pomocí oceňovacích opčních modelů takového typu bývá označena jako teoretická opční prémie a bývá využívána jako přijatelná aproximace skutečné opční prémie.

Při odvozování Black-Scholes-ova modelu se vychází podle [1], ze základní diferenciální rovnice, která je uvedena v následujícím tvaru

$$\frac{\partial C_t}{\partial t} + rS_t\frac{\partial C_t}{\partial S_t} + \frac{1}{2}\sigma^2 S_t^2\frac{\partial^2 C_t}{\partial {S^2}_t} = rC_t, \tag{2.14}$$

kde C_t je cena evropské Call opce v okamžiku t a S_t je spotová cena podkladového aktiva v okamžiku t, t je doba do splatnosti opce, σ je volatilita.

Při vypršení evropské Call opce v čase t musí platit následující podmínka, což je okrajová podmínka pro Black-Scholes-ovu diferenciální rovnici[1]:

$$C_t = \max(S_t - X, 0)\,. \tag{2.15}$$

Základní diferenciální rovnice (2.14) má za podmínky (2.15) pomocí následující substituce jednoznačné řešení [1], [28]

$$C\,(S.t) = e^{-rt}f(\xi,\eta),$$

$$\xi(S,t) = ln\frac{S}{X} + \left(r - \frac{1}{2}\sigma^2\right)t, \tag{2.16}$$

$$\eta(S,t) = \tfrac{1}{2}\sigma^2 t,$$

kde ξ je náhodná proměnná, která má logaritmicko normální rozdělení, r je roční bezriziková úroková míra.

Pomocí této substituce lze získat následující tvar [1]:

$$\frac{\partial^2 f(\xi,\eta)}{\partial \xi^2} - \frac{\partial f(\xi,\eta)}{\partial \eta} = 0. \tag{2.17}$$

Z podmínky lze získat upravenou podmínku [1]:

$$f(\xi,0) = 0 \qquad pro\ \xi < 0, \tag{2.18}$$

$$= X(e^{\xi} - 1) \quad pro\ \xi \geq 0.$$

Řešením výše uvedené rovnice je Black – Schole-sův model pro rovnovážnou cenu evropské Call opce v čase *t* (C_t) [1]::

$$C_t = S_t N(d_1) - Xe^{-it} N(d_2), \tag{2.19}$$

kde e^{-rt} je spojitý diskontovaný faktor a N (d) je distribuční funkce normálního rozdělení a dále [1],[28]:

$$d_1 = \frac{ln\frac{S}{X} + (r + \frac{1}{2}\sigma^2)t}{\sigma\sqrt{t}}, \tag{2.20}$$

$$d_1 = \frac{ln\frac{S}{X} + (r - \frac{1}{2}\sigma^2)t}{\sigma\sqrt{t}} = d_1\sigma\sqrt{t}, \tag{2.21}$$

$$N(d) = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{d} e^{-\frac{x^2}{2}}dx. \tag{2.22}$$

Pro evropskou Put opci lze vycházet z rovnice [28]:

$$P + S = C + Xe^{-rt}. \tag{2.23}$$

Po dosazení lze získat

$$P_t = -S_t + S_t \times N(d_1) - Xe^{-rt} \times N(d_2) + Xe^{-rt},$$

neboli (2.24)

$$S_t \times (N(d_1) - 1) - Xe^{-rt} \times N\big((d_2) - 1\big).$$

Pro distribuční funkci normálního rozdělení platí *N(d) = 1 – N (d)*. Pokud se využije tento vztah, pak konečný tvar Black-Scholes-ova vzorce pro rovnovážnou cenu evropské Put opce v čase t (P_t) má [1]:

$$P_t = -S_t N(d_1) + Xe^{-rt} N(d_2). \tag{2.25}$$

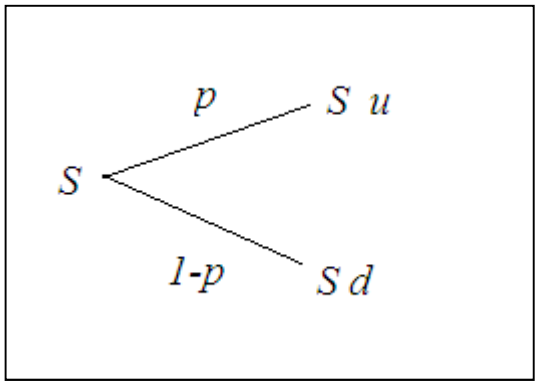
Existuje celá řada modelů pro oceňování finančních opcí, kterou jsou obdobou Black-Scholes-vova modelu, např. Merton-ův, Barnoe-Adesi, Whaley-ův apod. Garman, Kohlhagen a Gabber navrhli model pro oceňování měnové opce, který vychází právě z Black-Scholes-ova

modelu. Jejich model je založen na předpokladu tvorby bezrizikového zajišťovacího portfolia investováním do zahraničních bondů, domácích bondů a příslušné opce. Tyto měnové opce jsou obchodovány na tvz. Foreign exchange market – FX market, kde příslušné transakce probíhají nepřetržitě 24 hodin denně prostřednictvím telekomunikačních systémů [34].

Volba modelu ocenění závisí na typu opce. V jednoduchých opcích je možné se rozhodnout mezi opcí, která umožňuje získání hodnoty – analogie s finanční Call opcí, nebo opcí, která umožňuje zbavit se budoucí hodnoty – analogie s opcí put. Důležitým aspektem je i čas. Pokud jsou opční práva využitelná kdykoliv v době životnosti opce, pak jde o opci amerického typu a hodnota americké Put opce nemůže být stanovena Black-Scholes-ovým modelem, v tomto případě bude použit binomický model. I v případě, kdy jsou opční práva využitelná pouze v určitý okamžik, nemůže být použit Black-Scholes-ův model, ale pouze model binomický.

2.2 Binomický model oceňování pro akcie nevyplácející dividendy

Binomický model je stochastický model, který vychází z předpokladu, že životnost opce je rozdělena do velkého počtu malých časových intervalů délky Δt. Z Obr. 3 je patrné, že v každém takovém časovém intervalu se cena akcie pohybuje z počáteční hodnoty S do jedné ze dvou možných nových hodnot a to buď Su nebo Sd [28].



Obr. 3 - Binomický model pro akcii nevyplácející dividendy, zdroj: [1]

Hodnota Su představuje pohyb ceny akcie nahoru (up) z počáteční hodnoty S. Hodnota Sd znamená pohyb ceny akci dolů (down). Platí tedy, že index $u > 1$ a index $d<1$. Pravděpodobnost p zastupuje pravděpodobnost růstu ceny akcie, přičemž $1-p$ zastupuje pokles ceny akcie. [28]

Podmínky, které platí pro parametry p, u a d jsou následující:

1. Parametry musí být zvoleny tak, aby poskytly správné hodnoty pro střední hodnotu a rozptyl změny ceny akcie během časového intervalu Δt. Očekávaný výnos z akcie je roven bezrizikové úrokové míře r za předpokladu rizikově neutrálního světa, potom je očekávaná hodnota ceny akcie na konci časového intervalu Δt vypočtena jako [29]:

$$Se^{r\Delta t} = pSu + (1-p)Sd, \tag{2.26}$$

$$e^{r\Delta t} = pu + (1-p)d, \tag{2.27}$$

   kde S je cena akcie na počátku časového intervalu.

2. Je-li směrodatná odchylka proporcionální změny ceny akcie v krátkém časovém intervalu $\sigma\sqrt{\Delta t}$, znamená to, že rozptyl aktuální ceny v Δt je $S^2\sigma^2\Delta t$. Rozptyl náhodné proměnné Q je definován jako $E(Q^2) - E(Q)^2$, tedy platí [29].

$$\begin{aligned} S^2\sigma^2\Delta t &= pS^2u^2 + (1-p)S^2d^2 - S^2[pu + (1-p)d]^2, \\ \sigma^2\Delta t &= pu^2 + (1-p)d^2 - [pu + (1-p)d]^2. \end{aligned} \tag{2.28}$$

3. Jako třetí podmínka se obvykle používá vztah [29]:

$$u = \frac{1}{d}. \tag{2.29}$$

Uvedené podmínky jsou splněny za předpokladu, že Δt je malé, pomocí následujících rovnic[29]:

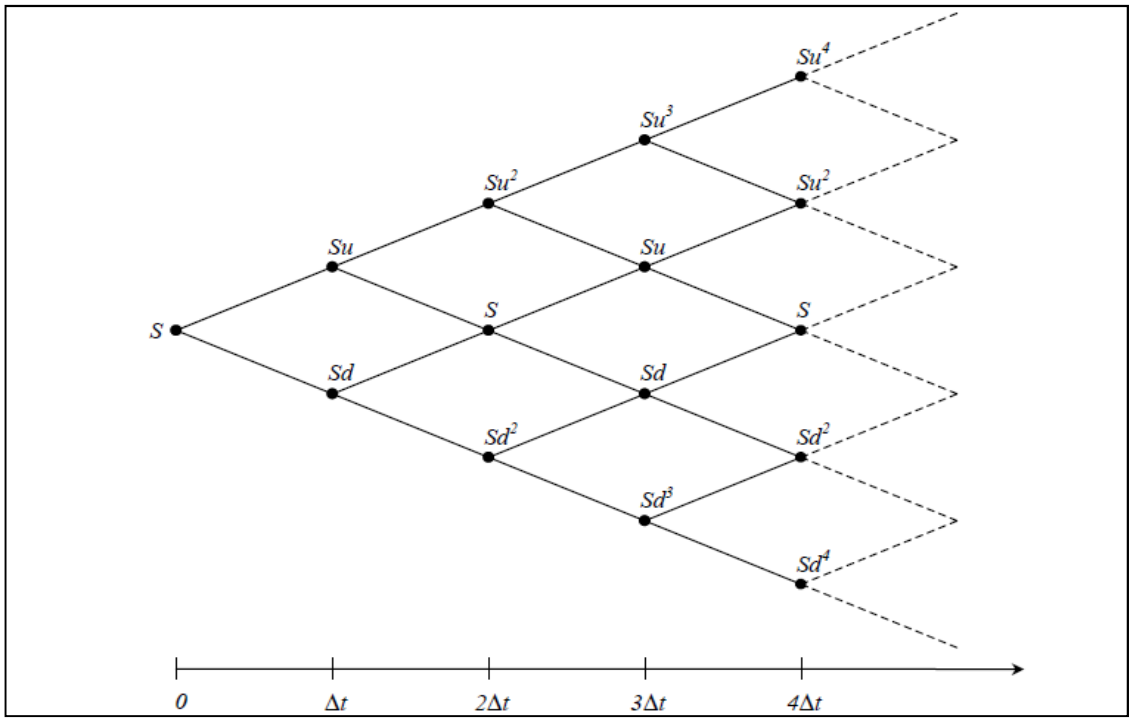
$$p = \frac{a-d}{u-d}, \tag{2.30}$$

$$u = e^{\sigma\sqrt{\Delta t}}, \tag{2.31}$$

$$d = e^{-\sigma\sqrt{\Delta t}}, \tag{2.32}$$

$$a = e^{r\Delta t}. \tag{2.33}$$

Orientační větvení binomického stromu je znázorněno na Obr. 4. Cena akcie klesá či roste exponenciálně nikoliv lineárně.



Obr. 4 - Binomický model, zdroj: [35]

V čase $t = 0$ je známá cena akcie S. Postupem času se cena akcie mění. Pro čas Δt existují dvě možnosti ceny akcie, tj. hodnota při růstu Su nebo při pokles Sd. Pro čas $2\Delta t$ existují již tři možné ceny akcie, tj. Su^2, S, Sd^2. Platí tedy, že v čase $i\Delta t$ existuje $i + 1$ možných cen akcie[1].

$$Su^j d^{i-j}, kde\ j = 0,1,..,i. \tag{2.34}$$

Vztah $u = \frac{1}{d}$ je použit v každém uzlu stromu. Každý pohyb nahoru následovaný pohybem dolů vede ke stejné ceně akcie jako pohyb dolů následovaný pohybem nahoru. Tato skutečnost podstatně snižuje počet uzlů stromu [18].

Při oceňování se postupuje od konce stromu k jeho počátku. Hodnota opce v čase t je známá. Např. hodnota Put opce je max $(X - S_t, 0)$ a hodnota Call opce max $(S_t - X, 0)$, kde S_t představuje cenu akcie v čase t a X zastupuje realizační cenu. Za předpokladu rizikově neutrálního světa lze hodnotu opce v každém uzlu v čase t-Δt určit jako očekávanou hodnotu v čase t diskontovanou sazbou r přes časovou periodu Δt. Podobně je i hodnota v každém uzlu v čase t - $2\Delta t$ kalkulována jako očekávaná hodnota v čase t-Δt diskontovaná přes časový interval Δt (k časovému okamžiku t-$2\Delta t$) bezrizikovou úrokovou sazbou r. Cestou jdoucí přes všechny uzle k počátku se zjistí hodnota opce v čase nula. Pokud je skutečná hodnota aktiva větší,

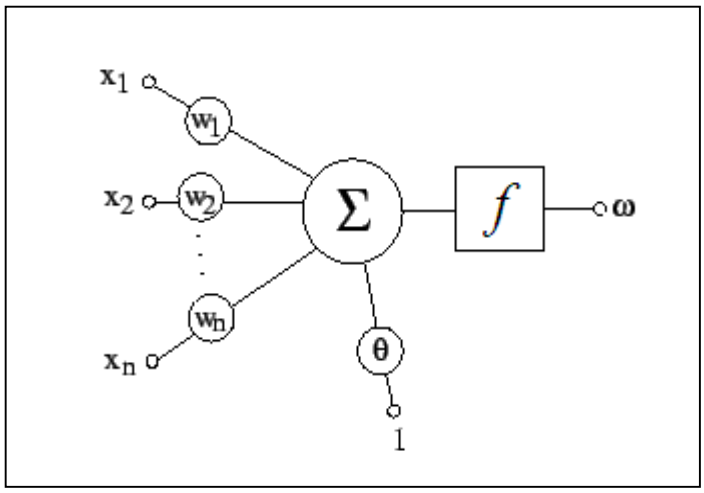
než předem smluvená realizační cena bude opce uplatněna, v opačném případě uplatněna nebude, neboť její hodnota je nulová [28].

3 Neuronové sítě

Neuronovou síť lze definovat jako masivně paralelní distribuovaný systém výkonných prvků, který modeluje biologické neurony [16]. Je účelně uspořádán tak, aby byl schopen získávat a zpracovávat informace, které jsou dále využívány. Díky získávání a uchovávání experimentálních znalostí představuje věrnou kopii lidského mozku. [6]

Neuron je základním stavebním elementem neuronové sítě. Biologický neuron se skládá z těla, dendridů, axionů a synapsí. Podle [16] jsou neurony „nervové buňky vzájemně hustě propojené a stýkající se v synapsích, malých výběžcích mezi jednotlivými neurony, které paralelně pracují s ostatními neurony v libovolné úrovni mozkové struktury“.

V roce 1943 McCulloch a Pitts definovali první „logický neuron“. Logický neuron pracoval s binárními vstupními a výstupními hodnotami. Dalším známým modelem byl Widrow-ův „adaptivní lineární neuron“ Adaline z roku 1960 [21], [22]. Strukturu McCulloch-Pitts-ova neuronu uvádí Obr. 5.



Obr. 5 - McCulloch-Pitts-ův neuron, zdroj: [21]

Vstupními podněty do neuronové sítě $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ mohou být výstupy z předcházejících neuronů nebo podněty z vnějšího okolí. Jsou-li vstupní podněty přijímány z vnějšího okolí, musí zdolat tzv. prahovou hodnotu neuronu θ. Práh neuronu rozhoduje o tom, kdy je neuron aktivní a kdy neaktivní. V případě kdy je vstupní hodnota neuronu nižší než prahová hodnota bude výstup z neuronu odpovídat neaktivnímu stavu neuronu. V opačném případě bude neuron v aktivním stavu a poroste až do určité maximální hodnoty, která je dána oborem hodnot příslušné aktivační funkce f [10]. Každý neuron obsahuje konečný počet vstupů x_n.

Vstup do neuronu musí být ohodnocen synaptickou váhou w_{ij}. Tato váha vyjadřuje citlivost, s jakou příslušný vstup ovlivňuje výstup z neuronu. Synapse mají svůj směr a právě ony spojují jednotlivé neurony do sítí.

V každém neuronu dochází k transformaci vstupních hodnot na výstupní hodnoty a to za pomocí minimálně dvou výpočetních procedur, tj. výpočtu vstupního potenciálu y_a a aktivační funkce f [10].

Každý podnět x_i je násoben váhou w_i, takovýto součin $x_i \times w_i$, pak vstupuje do součtového členu a následně je vytvořen vážený součet. Agregačním procesem je získán vstupní potenciál neuronu, který představuje vstup do aktivační funkce[11]:

$$y_a(t) = \sum_{i=1}^{n} x_i(t) \times w_i(t) + \theta. \tag{3.1}$$

Aktivační funkce je definována jako $f\ (y_a)$. Jejím úkolem je převést hodnotu vstupního potenciálu na výstupní hodnotu z neuronu. Podle [10] existují např. tyto následující typy aktivačních funkcí.

1. Prahová funkce

$$f(\,y_a\,) = \begin{Bmatrix} 1 & if & y_a \geq 0 \\ 0 & if & y_a < 0 \end{Bmatrix}, \tag{3.2}$$

2. Po částech lineární

$$f(\,y_a\,) = \begin{Bmatrix} 1 & if & y_a \geq +\frac{1}{2} \\ y_a & if & +\frac{1}{2} > y_a > -\frac{1}{2} \\ 0 & if & y_a \leq -\frac{1}{2} \end{Bmatrix}, \tag{3.3}$$

3. Sigmoidální funkce (logistická funkce)

$$f(y_a) = \frac{1}{1 + e^{-y_a}}. \tag{3.4}$$

Rovnice předešlých aktivačních funkcí jsou definovány na intervalu < 0;1 >. S ohledem na symetričnost lze aktivační funkce definovat na intervalu od < -1; 1 >. Potom prahová funkce (3.2) a hyperbolický tangens budou definovány takto

$$f(y_a) = \begin{Bmatrix} 1 & if & y_a > 0 \\ 0 & if & y_a = 0 \\ -1 & if & y_a < 0 \end{Bmatrix}, \tag{3.5}$$

$$f(y_a) = \tan y_a. \tag{3.6}$$

3.1 Učení neuronové sítě

Činnost neuronových sítí lze rozdělit na fázi učení a na fázi života. Ve fázi učení neuronová síť sbírá a uchovává znalosti. Znalosti jsou uchovány v synaptických vahách. Je tedy jasné, že se synaptické váhy v průběhu učení mění. Pokud se označí $\boldsymbol{W}$ jako matice všech synaptických vah neuronové sítě, potom pro stav učení neuronové sítě platí $\frac{\partial \boldsymbol{W}}{\partial t} \neq 0$. Po skončení fáze učení se neuronová síť nachází ve fázi života, ve které dochází k využívání získaných znalostí ve prospěch řešené problematiky. Ve fázi života se synaptické váhy nemění, platí tedy, že $\frac{\partial \boldsymbol{W}}{\partial x} = 0$ [23].

Důležitou vlastností neuronových sítí je tedy jejich schopnost učit se. Učením se myslí nastavení vah $\boldsymbol{W}$ podle předložených vzorů tak, aby neuronová síť co nejpřesněji zpracovala i neznámé příklady. Každá neuronová síť má jiný algoritmus učení. Vhodnost algoritmu je dána kvalitou a rychlostí učení na předložených datech [2].

Množina dat se většinou náhodně rozděluje na data v trénovací množině a data v testovací množině. Data v trénovací množině se používají ve fázi učení neuronové sítě. Na reprezentativnost trénovacích dat je kladen veliký důraz, neboť právě z nich jsou extrahovány znalosti do synaptických vah neuronové sítě ve fázi učení. Data v testovací množině slouží k otestování získaných znalostí ve fázi života neuronové sítě[23].

Proces učení může probíhat dvojím způsobem. V případě učení s učitelem má neuronová síť k dispozici informaci o požadovaném výstupu a v průběhu učení se snaží tomuto výstupu co nejvíce přiblížit. Taková neuronová síť vychází z historických dat. Tento způsob učení je někdy označován jako za kontrolovatelné učení či za induktivní metodu.

Proces učení je započat předložením dat vstupní vrstvě. Tato data jsou brána z trénovací množiny a představují vzory požadovaného chování. Cílem učení je najít takové optimální nastavení parametrů sítě (vektor vah, strmost, aktivační funkce), při kterém bude dosaženo co největší shody mezi skutečným a požadovaným výstupem[23]. Zavádí se tedy tzv. střední kvadratická chyba, která má pro *m* příkladů z trénovací množiny tvar [33].

$$E_t = \sum_{k=1}^{m} [\hat{y}_k(t_{tren}) - y_k]^2 , \qquad (3.7)$$

kde t je čas trénování, m představuje počet vzorů trénovací množiny, k je pořadové číslo vzoru trénovací množiny, y_k jsou požadované hodnoty výstupu z neuronové sítě, $\hat{y}_k\ (t)$ jsou hodnoty výstupu z neuronové sítě.

Pokud je ve výstupní vrstvě více neuronů, je třeba rovnici (3.7) upravit na tvar

$$E_t = \sum_{k=1}^{m} e_k(t), \qquad kde \qquad e_k(t) = \frac{1}{2}\sum_{h=1}^{M} \left[\hat{y}_k^h(t) - {y_k}^h\right]^2, \qquad (3.8)$$

kde $h = 1,2,..M$ je pořadové číslo výstupního neuronu a M představuje počet výstupních neuronů. Střední kvadratickou chybu E_t je třeba minimalizovat [33].

Nejpoužívanější metodou učení s učitelem je metoda zpětného šíření (Back-propagation), která minimalizuje střední kvadratickou chybu E_t.

V případě učení bez učitele nemá neuronová síť k dispozici informaci o požadovaném výstupu. Tuto informaci si sama odvozuje ze svého výstupu pomocí zpětné vazby. Neuronová síť rozezná ve svých vstupech vektory vykazující podobné vlastnosti a podle nich je sdruží do shluků nebo map. Princip učení je založen na výpočtu vzdálenosti mezi vzory a aktuálními hodnotami. Cílem je nalézt opět minimální vzdálenosti.

4 Neuronové sítě typu RBF

Tato kapitola se zabývá neuronovou sítí typu RBF, která se řadí mezi dopředné neuronové sítě s jednou skrytou vrstvou obsahující lokální jednotky.

Začátkem 80. let se v části numerické matematiky zabývající se interpolacía a proximací dat, začaly studovat radiálně bazické funkce (RBF) jako jeden z nových způsobů řešení aproximačních problémů. První, kdo navrhl využít radiálně bazické funkce (RBF) pro vytvoření nového modelu umělých neuronových sítí, byly Broomhead a Lowe v roce 1988. Další podstatný podíl na rozvoji RBF sítí měly práce Moody-ho a Darken-a, Poggi-ho a Girosi-ho [32].

4.1 Topologie neuronové sítě typu RBF

Neuronová síť typu RBF je třívrstvou neuronovou sítí. Vstupní vrstva neuronů slouží k přenosu vstupních hodnot. Druhá vrstva, označena jako skrytá vrstva, se skládá z RBF jednotek. Tyto jednotky realizují jednotlivé radiálně bazické funkce (RBF). Třetí vrstva je výstupní a je většinou lineární [32].

Neuronová síť typu RBF patří do kategorie dopředných neuronových sítí. Její signál se šíří od vstupů přes skrytou vrstvu k výstupům. Aby síť správně fungovala, musí být plně propojeny všechny neurony ve všech vrstvách. Čili neurony sousedících vrstev by měli být mezi sebou vzájemně propojeny tak, že výstup jednoho je distribuován do vstupů neuronů následující vrstvy.

Počet vrstev a počet neuronů udává parametry neuronové sítě. Parametry závisí na povaze řešeného problému. V případě, že je počet neuronů malý, neuronová síť nebude schopná postihnout všechny závislosti na trénovacích datech. V případě velkého počtu je prodloužena doba učení a při nadměrném počtu trénovacích dat síť vykazuje špatnou schopnost generalizace způsobenou tzv. přeučením neboli overfitting [23].

Neuronová síť typu RBF využívá jako aktivační funkci radiálně bazickou funkce RBF a je založena na učení s učitelem. Takto vymezené neuronové sítě jsou z pohledu aproximace přirozené, jelikož aproximujeme funkcemi, které ovlivňují výslednou funkci jen v okolí centra příslušného RBF neuronu a ne v celém rozsahu funkce [11]. Z pohledu klasifikace je použití RBF sítě taktéž vhodné, protože ve většině případů náleží určitá skupina vstupních vektorů do jedné z tříd, které jsou za pomocí této neuronové sítě hledány. Výstup neuronové sítě typu RBF je definován takto

$$f_j(\boldsymbol{x}, \boldsymbol{H}, \boldsymbol{w}) = \sum_{i=1}^{q} w_{i,j} \times h_i(x), \tag{4.1}$$

kde $\boldsymbol{H}$ je množina aktivačních funkcí radiálně bazických neuronů ve skryté vrstvě, tj. $\boldsymbol{H} = \{h_1(x), h_2(x), \dots, h_i(x), \dots, h_q(x),\}$ a $w_{i,j}$ označeny synaptické váhy. Každý z m prvků vstupního vektoru $\boldsymbol{x} = (x_1, x_2, \dots, x_m)$ je vstupní hodnotou pro q radiálně bazických funkcí $h_i(x)$. Celkový výstup $f_j(\boldsymbol{x}, \boldsymbol{H}, \boldsymbol{w})$ je lineární kombinací výstupů z q RBF neuronů a odpovídajících vah synapsí $w_{i,j}$ [11].

Vstupní vrstva neuronové sítě RBF zprostředkovává načítání jednotlivých vstupních vzorů $\boldsymbol{x}_i = (x_{i,1}, x_{i,2}, x_{i,3} \dots, x_{i,k} \dots x_{i,m})$. Neuronové sítě typu RBF mají jen jednu skrytou vrstvu. Není možné konstruovat neuronovou síť typu RBF s více než jednou skrytou vrstvou. Vysvětlení spočívá v omezení počtu skrytých vrstev. Každá z m hodnot vstupního vektoru $\boldsymbol{x}$ je použita jako parametr aktivační funkce $h_i(x)$, kde q je počet RBF neuronů ve skryté vrstvě. Celkový výstup nerunové sítě je pak lineární kombinací radiálně bazických funkcí a vah synapsí. Pokud by se použilo více vrstev, nebyla by tato kombinace lineární a proces učení by byl složitější. Neurony ve výstupní vrstvě jsou reprezentovány pouze váženou sumou všech vstupů předchozích ze skryté vrstvy.

RBF neurony se nachází ve skryté vrstvě. Radiální bazické funkce RBF se řadí do speciální třídy matematických funkcí. Tyto funkce monotónně klesají nebo rostou a to se stále zvětšující se vzdáleností od centra. Jsou použity jako aktivační funkce u neuronových sítí, které byly podle nich pojmenovány. Obecný zápis radiální bazické funkce RBF je [10]:

$$h_i(x) = \phi\left(\frac{\sqrt{(\boldsymbol{x} - \boldsymbol{c})^T \times (\boldsymbol{x} - \boldsymbol{c})}}{|sc|}\right), \tag{4.2}$$

kde $\boldsymbol{x}$ je vstupní vektor, $\boldsymbol{c}$ označuje centrum radiální funkce RBF a sc je poloměr radiální bazické funkce RBF.

Pokud je zadán jednorozměrný vstupní vektor, obecný zápis přechází do tvaru

$$h_i(x) = \phi\left(\left|\frac{\boldsymbol{x} - \boldsymbol{c}}{sc}\right|\right). \tag{4.3}$$

Za radiální bazickou funkci RBF $\phi : R \rightarrow R$ z předešlého vztahu lze dosadit jednu z následujících funkcí [10]:

lineární $$\phi(x) = x, \tag{4.4}$$

kubická $$\phi(x) = x^3, \tag{4.5}$$

Gauss-ova $$\phi(x) = e^{-x^2}, \tag{4.6}$$

multikvadratická $$\phi(x) = \sqrt{1 + x^2}, \tag{4.7}$$

inverzní multikvadratická $$\phi(x) = \frac{1}{\sqrt{1 + x^2}}, \tag{4.8}$$

Cauchy-ho $$\phi(x) = \frac{1}{1 + x}. \tag{4.9}$$

4.2 Učení neuronové sítě typu RBF

Učení neuronové sítě RBF probíhá ve dvou fázích. V první fázi dochází k určení počtu RBF center a nalezení jejich nejvhodnějších pozic. V druhé fázi dochází k určení poloměrů center, nastavení vah mezi skrytou a výstupní vrstvou a určení strmosti RBF.

4.2.1 První fáze - určení počtu RBF center

V 90. letech 20. století Niyogi a Girosi definovali optimální počet RBF neuronů ve skryté vrstvě. Podle [10] „je dána neuronová síť typu RBF používající Gauss-ovu funkci s m_0 vstupními neurony a q skrytými neurony. Dále, nechť *f(x)* představuje regresní funkci, která patří do Sobolev-ova prostoru.“ Potom lze předpokládat, že trénovací data jsou náhodným výběrem z funkce *f(x)*, jež je regresní funkcí. Pro všechny parametry $\delta \in (0,1)$ platí, že trénovací chyba E_{tren} je ohraničena shora s pravděpodobností vyšší než $1 - \delta$ [8], [10]:

$$\left\| f_0 - \hat{f}_{q,N} \right\|^2 \leq E_{tren}\left(\frac{1}{q}\right) + E_{tren}\left(\frac{m_0 \times q}{N} \times \log(q, N) + \frac{1}{N} \times \log\left(\frac{1}{\delta}\right)^{1/2}\right). \tag{4.10}$$

Z uvedených poznatků lze vyvodit následující závěry pro určení počtu RBF neuronů ve skryté vrstvě [8]:

- neuronové sítě jsou neefektivní pro řešení klasifikačních úloh, pokud se jednotlivé RBF neurony překrývají,
- počet neuronů ve skryté vrstvě ovlivňuje průběh trénovací chyby E_{tren},
- s růstem počtu RBF neuronů q ve skryté vrstvě konverguje trénovací chyba E_{tren} pro trénovací množinu dat k nule,
- velikost dat trénovací množiny N je optimální množství neuronů ve skryté vrstvě přibližně $\sqrt[3]{N}$, (zvýšení či snížení množství RBF neuronů vede k zvýšení chyby E_{tren}),
- testovací chyba E_{test} klesá, pokud počet neuronů q roste výrazně pomaleji než množství dat N ve vstupní množině.

4.2.2 Druhá fáze – nalezení center RBF neuronů

Existuje řada metod pro určení centra RBF neuronů, např. metoda segmentace, náhodná volba, samo-organizující se výběr center, kontrolovaný výběr center [24]. V práci je rozebrána metoda segmentace a metoda náhodného výběru. Přesná a správná volba center ovlivňuje možnost použití nižšího množství neuronů ve skryté vrstvě, přičemž přesnost bude zachována.

Metoda náhodného výběru využívá fixní sklon radiálně bazických funkcí RBF, jejichž poloha je zvolena náhodně z množiny trénovacích dat O_{tren}. Náhodně vybraná centra by měla vhodně reprezentovat data vstupující do neuronové sítě. Jako radiálně bazickou funkci RBF lze použít Gauss-ovu funkci. Její směrodatná odchylka je stanovena v závislosti na rozložení center[10]*:*

$$G\|\boldsymbol{x}-\boldsymbol{c_i}\|^2 = e^{\left(-\frac{q_1}{d_{max}^2}\|\boldsymbol{x}-\boldsymbol{c_i}\|^2\right)}; i = 1,2,\dots,q, \tag{4.11}$$

kde q zastupuje počet center a d_{max}, je maximální vzdálenost mezi zvolenými centry.

Sklon radiálně bazických funkcí (σ_r) je fixní a je zjištěn podle vztahu (4.12). Tento vztah vypovídá o průběhu radiálně bazické funkce RBF, který nebude ani příliš plochý ani příliš strmý[10]:

$$\sigma_r = \frac{d_{max}}{\sqrt{2q}}. \tag{4.12}$$

Metoda náhodného výběru představuje jednoduchý způsob jak určit centra RBF neuronů. Metoda není považována za efektivní při použití velkého množství dat, kde dochází k rychlému

a často zbytečnému nárůstu množství q RBF neuronů ve skryté vrstvě a tím i neopodstatněnému růstu složitosti výsledné neuronové sítě.

Použití metody segmentace vychází z předpokladu, že do neuronové sítě vstupují předem známá data. Tato data nesou informace, které mohou být využity k vyhledání vhodných míst jako center RBF oblastí. Pro segmentaci dat je možné použít řadu algoritmů, např. K-means.

K-means představuje nejznámější nehierarchickou metodu shlukové analýzy. K-means hledá pro datovou matici $\boldsymbol{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_{2,} \dots, \boldsymbol{x}_N\}$ vektory $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \dots, \boldsymbol{\mu}_k, kde\ (k < N)$, takové, že je minimalizována střední kvadratická chyba matice $\boldsymbol{X}$ od vektorů $\boldsymbol{\mu_1}, \boldsymbol{\mu}_2, \dots, \boldsymbol{\mu}_k$**.** Jinak řečeno, metoda K-means hledá *k* vektorů, které dobře aproximují danou množinu dat, tedy hledá takové vektory, ke kterým je euklidovská vzdálenost všech dat co nejmenší. K–means rozděluje vstupní matici dat do *k* shluků tím, že minimalizuje funkci *J*. Jeho vstupem je množina dat $\boldsymbol{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_N\}$ a informace – číslo *k* udávající počet vektorů $\boldsymbol{\mu}_j$. Funkce *J* je dána podle [17] takto:

$$J = \sum_{j=1}^{k} \sum_{i=1}^{l} \left(\boldsymbol{x}_i - \boldsymbol{\mu}_j\right)^2. \tag{4.13}$$

kde *k* zastupuje počet shluků, *l* udává počet objektů ve shluku, $\boldsymbol{x}_i$ je i-tý objekt ve shluku a $\boldsymbol{\mu}_j$ je reprezentant shluku.

Algoritmus K-means probíhá ve čtyřech následujících krocích[17].

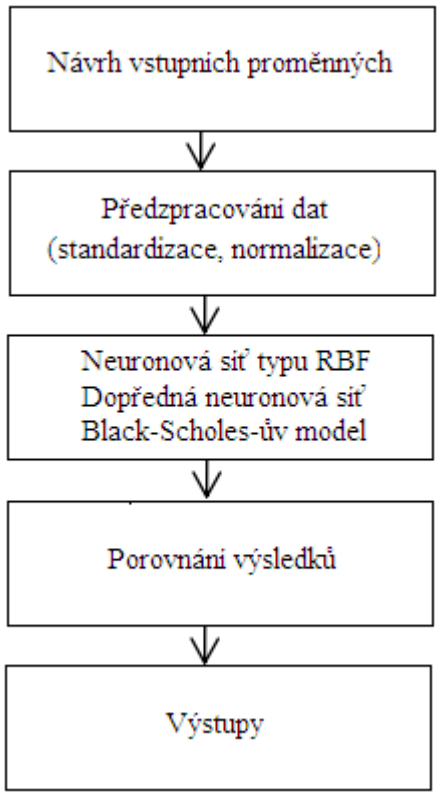
1. Volba reprezentantů daných oblastí: na začátku dochází k inicializaci vektorů $\boldsymbol{\mu}_j$, $j = 1,2,\dots,k$ na náhodně zvolenou hodnotu nebo použitím nějaké vhodně zvolené heuristiky (např. apriorní znalost o úloze).
2. K reprezentantům jsou přiřazeny jednotlivé prvky: všechna dat $\boldsymbol{x}_i$, $i = 1,2,\dots N$ se klasifikují do tříd určených vektory $\boldsymbol{\mu}_{j,}$, $j = 1,2,\dots$, k podle minima euklidovské vzdálenosti. Tedy vzor $\boldsymbol{x}_i$ je přiřazen do třídy $y_i = min \sum_{j=1}^{k} \left\| \boldsymbol{x}_i - \boldsymbol{\mu}_j \right\|$.
3. Pomocí výpočtu těžišť oblastí dojde k přepočtu reprezentantů těchto oblastí: vypočítají se nové hodnoty vektorů $\boldsymbol{\mu}_j$ jako střední hodnoty dat $\boldsymbol{x}_i$, které byly klasifikovány do třídy určené příslušným vektorem $\boldsymbol{\mu_j}$. Nová hodnota $\boldsymbol{\mu}_j$ se vypočítá jako $\boldsymbol{\mu}_j = \frac{1}{n_j} \sum \boldsymbol{x}_i$, kde n_j je počet vzorů $\boldsymbol{x}_i$ klasifikovaných v prvním kroku do třídy určené vektorem $\boldsymbol{\mu}_j$.

Kroky 1 a 2 se stále opakují do té doby, dokud se alespoň jeden vektor $\boldsymbol{x}_i$ klasifikuje do jiné třídy, než byl klasifikován v předcházejícím kroku, čili dojde ke změně poloze reprezentantů daných oblastí [16].

Každá takto vytvořená množina má svého zástupce, který je zároveň centrem RBF neuronových sítí. Tímto je zajištěno optimální přiřazení neuronů skryté vrstvy k daným skupinám dat, které vstupují do RBF neuronové sítě.

5 Modelování oceňování finančních opcí

Následující obr. 6 znázorňuje návrh modelu pro oceňování finančních opcí.



Obr. 6 - Návrh modelu pro oceňování finančních opcí

Cílem práce je predikce ceny Call opce *O*. Jako vstupní proměnné byly navrženy nejznámější faktory, které ovlivňují cenu opce *O*. Jedním z hlavních faktorů určující cenu opce *O* je volatilita σ. Volatilita σ označuje míru kolísání hodnoty aktiva. Lze ji chápat jako směrodatnou odchylku těchto změn během časového úseku. Obvykle se přepočítává na roční volatilitu σ a může se udávat buď v absolutních hodnotách nebo relativních hodnotách. U finančních instrumentů roste volatilita σ s odmocninou časového úseku, na němž je měřena. Cena opce *O* se odvíjí od implikované volatility, která označuje očekávanou volatilitu do budoucna na rozdíl od historické, která označuje hodnotu volatility vypočtenou na základě historických dat. Důležitou roli při stanovení ceny opce *O* hraje realizační cena *X*. a spotová cena *S*. Roste-li spotová cena *S*, roste opční prémie *O* u Call opce a klesá u Put opce. Dalším zásadním faktorem při stanovení ceny opce *O* je doba do splatnosti opce *t*. Doba do splatnosti opce určuje tzv. „ časovou hodnotu opce“. Ta ve finančním vyjádření představuje riziko změny spotové ceny podkladového aktiva *S* v průběhu doby splatnosti opce. Toto riziko

se s ubývajícícm čase do splatnosti opce snižuje a v den splatnosti (exspirace) opce je nulové. Dalším významným determinantem při určování ceny opce *O* je úroková míra *r*. Cena Call opce roste s růstem úrokové míry, zatímco u ceny Put opcí tomu je naopak.

Výjmenované faktory byly zvoleny jako vstupy. Za výstup modelu byla jednoznačně označena cena opce *O*, tj. $O = f(\sigma,t,S,X,r)$. Pro verifikaci navrženého modelu na získaných datech bylo zvoleno integrované programové prostředí Matlab 7.1. Před zpracováním dat neuronovou sítí je vhodné data předzpracovat. K předzpracování dat byla použita standardizace a normalizace. Standardizace a normalizace dat je vykonána před inicializací (počátečním nastavením synaptických vah $w_{i,j}$) neuronové sítě typu RBF a dopředné neuronové sítě. V průběhu trénování a testování neuronové sítě typu RBF byla měněna hodnota *sc* – hodnota poloměru $h_i(x)$. V případě trénování a testování modelu pomocí dopředné neuronové sítě byla měněna hodnota počtu epoch. Výstupy neuronových sítí byly mezi sebou porovnány na základně vypočtených chyb. Pro další porovnání byl použit Black-Scholes-ův model.

5.1 Data

Data byla získána z uměle vytvořené databáze Worden TC 2000 [27],[13]. Celkem bylo získáno 1530 hodnot pro vstupní množinu:

- volatilita σ [%] – minimální hodnota je σ = 20% a maximální hodnota je σ = 200%, volatilita se postupně zvyšuje o 20% pro 1530 měření,
- doba do splatnosti *t* – vyjádřena ve dnech, maximální doba splatnosti je 15 dní, minimální je 5 dní, doba se od minimální hodnoty navyšuje o 5 dní, dokud nedosáhne maxima,
- spotová cena podkladového aktiva *S* - pevně stanovena na 100 $,
- bezriziková úroková míra *r* [%] - nastavena na *r* = 0%,
- realizační cena *X* - se pohybuje v intervalu od 75 $ do 125 $, navyšuje se o 1 $.

Výstupní množina obsahuje 1530 hodnot pro cenu Call opce *O*. Ukázku dat zobrazuje Obr. 7.

Volatilita (σ)	Čas (t)	Strike cena (X)	Cena opce (O)
0,2	5	75	25,0000
0,2	5	76	24,0000
0,2	5	77	23,0000
0,2	5	78	22,0000
0,2	5	79	21,0000
0,2	5	80	20,0000
0,2	5	81	19,0000
0,2	5	82	18,0000
0,2	5	83	17,0000
0,2	5	84	16,0000
0,2	5	85	15,0000
0,2	5	86	14,0000
0,2	5	87	13,0000
0,2	5	88	12,0000
0,2	5	89	11,0000
0,2	5	90	10,0001
0,2	5	91	9,0003
0,2	5	92	8,0012
0,2	5	93	7,0043
0,2	5	94	6,0135
0,2	5	95	5,0372
0,2	5	96	4,0908
0,2	5	97	3,1976
0,2	5	98	2,3870
0,2	5	99	1,6886
0,2	5	100	1,1239
0,2	5	101	0,6992

Obr. 7 - Získaná data

Z Obr. 7 je patrné, že při růstu realizační ceny X roste cena Call opce O. Pro zjištění dalších vztahů byla vypočtena statistika ceny Call opce O a to vždy pro hodnotu volatility σ v čase t = 5, 10, 15. Základní popisnou statistiku uvádí obrázek Obr. 8.

Volatilita σ	Čas t	Průměr	Medián	Směr. odchylka	Minimum	Maximum	Počet
0.2	5	6.4520	1.1239	8.2584	0.0001	25.0000	51.0000
	10	6.5299	1.5893	8.2133	0.0001	25.0000	51.0000
	15	6.6079	1.9465	8.1721	0.0001	25.0000	51.0000
0.4	5	6.6859	2.2475	8.1337	0.0001	25.0000	51.0000
	10	6.9983	3.1780	7.9991	0.0068	25.0003	51.0000
	15	6.6859	2.2475	8.1337	0.0001	25.0000	51.0000
0.6	5	7.0763	3.3707	7.9691	0.0121	25.0006	51.0000
	10	7.7676	4.7655	7.7447	0.1607	25.0275	51.0000
	15	8.4269	5.8347	7.5774	0.4528	25.1178	51.0000
0.8	5	7.6163	4.4932	7.7886	0.1126	25.0166	51.0000
	10	8.7773	6.3510	7.5007	0.6516	25.1940	51.0000
	15	9.8264	7.7742	7.3045	1.3693	25.5238	51.0000
1.0	5	8.2836	5.6148	7.6110	0.3792	25.0922	51.0000
	10	9.9506	7.9340	7.2837	1.4635	25.5715	51.0000
	15	11.3938	9.7091	7.0691	2.6547	26.2308	51.0000
1.2	5	9.0495	6.7353	7.4455	0.8222	25.2657	51.0000
	10	11.2301	9.5139	7.0915	2.5122	26.1477	51.0000
	15	13.0604	11.6383	6.8607	4.1778	27.1686	51.0000
1.4	5	9.8887	7.8545	7.2940	1.4164	25.5475	51.0000
	10	12.5788	11.0900	6.9179	3.7259	26.8825	51.0000
	15	14.7871	13.5605	6.6705	5.8536	28.2699	51.0000
1.6	5	10.7817	8.9722	7.1551	2.1306	25.9300	51.0000
	10	13.9732	12.6618	6.7577	5.0542	27.7376	51.0000
	15	16.5504	15.4748	6.4926	7.6295	29.4857	51.0000
1.8	5	11.7144	10.0880	7.0264	2.9379	26.3989	51.0000
	10	15.3983	14.2286	6.6073	6.4631	28.6825	51.0000
	15	18.3356	17.3800	6.3235	9.4717	30.7819	51.0000
2.0	5	12.6766	11.2019	6.9061	3.8170	26.9397	51.0000
	10	16.8437	15.7897	6.4642	7.9295	29.6947	51.0000
	15	20.1330	19.2750	6.1609	11.3580	32.1349	51.0000

Obr. 8 - Statistika dat

Z vypočtené statistiky lze usuzovat, že s růstem volatility σ klesá směrodatná odchylka ceny opce *O*. S růstem volatility σ a času *t* roste cena opce *O*. Maximální naměřená hodnota je při volatilitě $\sigma = 2$ a čase $t = 15$, minimální hodnota byla naměřena při $\sigma=0.2$ kdy čas byl t = 5, 10, 15 a při $\sigma = 0{,}4$ kde čas t = 5. Maximální hodnoty ceny opce se při volatilitě σ v intervalu $< 0.2; 1>$ pohybují kolem 25$, od $\sigma = 1.2$ se zvyšují téměř vždy o 1$.

5.2 Předzpracování dat

Z praktických důvodů je většinou třeba data před zpracováním pomocí neuronových sítí vhodně upravit. Po úpravě by se úroveň signálu přicházejícího do vstupní vrstvy měla vždy pohybovat v hodnotách, při kterých je neuron vstupní vrstvy nejcitlivější na vstupní signál.

Jelikož jsou vstupní data v jednotlivých jednotkách je zapotřebí provést standardizaci. Standardizace je metoda statistické analýzy. Provedením standardizace lze docílit souměřitelnosti všech parametrů, přiřadím jim stejnou váhu. Standardizované hodnoty mají střední hodnotu rovnu 0 a rozptyl 1. Standardizaci lze provést podle následujícího vztahu [26]:

$$z_{ij} = \frac{x_{ij} - \bar{x}_j}{s_j}, \tag{5.1}$$

kde i = 1, 2,…, m a j = 1, 2,…., n; dále $\bar{x}_j$ je střední hodnota, s_j je směrodatná odchylka a x_{ij} je prvek.

Velké rozdíly mezi prvky lze odstranit provedením normalizace. Pokud je známa norma a_i standardizované matice, mohou být dopočítány hodnoty normalizované matice X_{ij} podle vztahu[26]:

$$X_{ij} = \frac{x_{ij}}{a_i}, \tag{5.2}$$

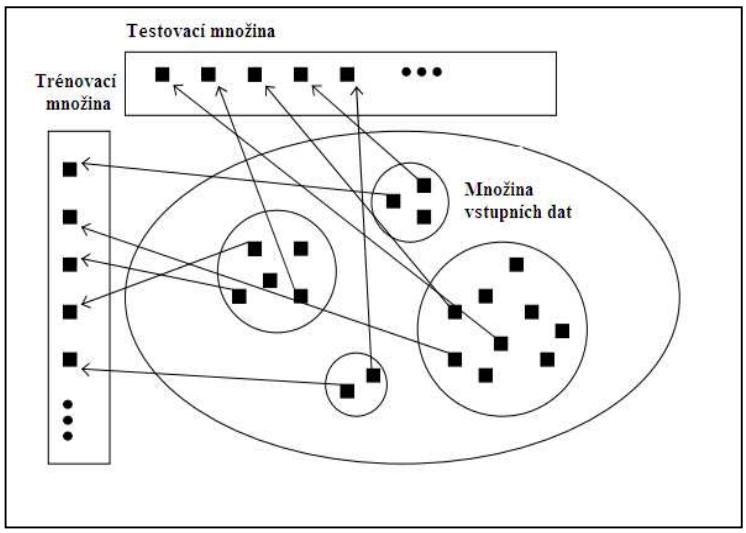
$$a_i = \sqrt{\left[\sum_{j=1}^{s} (x_{ij})^2\right]}. \tag{5.3}$$

Aby byl proces učení úspěšný, je nutné rozdělit data na množinu trénovacích *(O_{tren})* a testovacích dat.*(O_{test})* Rozdělení lze uskutečnit náhodným výběrem, výběrem každého n-tého řádku, pomocí shlukové analýzy, apod.

Data v této práci byla rozdělena na trénovací a testovací, podle následujícího kódu v programovém prostředí Matlab, náhodným výběrem v poměru 2 : 1. Příkaz *randperm (n)* vrací náhodné permutace celých čísel od 1 do 1530 [20].

```
%nahodne permutace
perm = randperm(1530)';
%mnozina vstupu
input = input(:,perm);
%mnozina vystupu
output = output(:,perm);
%mnozina vstupu trenovacih
input_tren = input(:,1:1020);
%mnozina vstupu testovacih
input_test  = input(:,1021:end);
%mnozina vystupu trenovacich
output_tren = output(:,1:1020);
%mnozina vystupu testovacih
output_test = output(:,1021:end);
```

Důležitou podmínkou procesu učení je, aby vstupní data v testovací množině reprezentovala všechna data v trénovací množině. Rozklad vstupní množiny je vysvětlen na Obr. 9. Proto byl proces náhodného rozdělení dat opakován pětkrát, aby se zamezilo nereprezentativnosti trénovací množiny dat.



Obr. 9 - Rozklad množiny vstupních dat, zdroj: [22]

5.3 Modelování pomocí neuronové sítě typu RBF

Podle následujícího strojového kódu v programovém prostředí Matlab (Neural Network Toolbox-u) je vytvořena neuronová síť typu RBF, která je trénována a následně testována.

```
% nastaveni pozadovane chyby oznacena jako eg
eg = 0.2;
% nastaveni hodnoty polomeru RBF funkce
sc = 1;
% navrh site RBF
net = newrb(input_tren,output_tren,eg,sc);
% vystup trenovani
ytren = sim(net, input_tren);
% vystup testovani
ytest = sim(net,input_test);
```

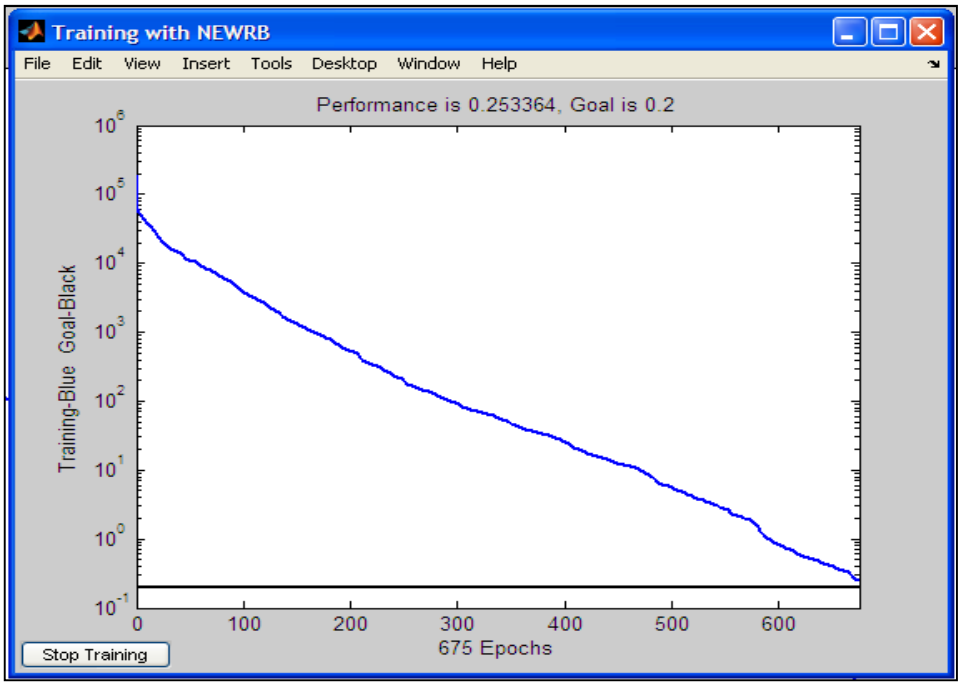
Pomocí funkce *newrb ()* je vytvořena neuronová síť typu RBF. Tento příkaz vyžaduje několik vstupních proměnných:

- *P* - vstupní množina: v tomto případě je to soubor input_tren,
- *T* - výstupní množinu: v tomto případě je to soubor output_tren,
- *eg* - požadovaná střední kvadratická chyba - nastavena na 0.2,
- *sc* - hodnota poloměru RBF funkce, hodnota nastavena na 1, následně bude měněna.

Příkaz *newrb* přidává neurony do skryté vrstvy RBF neuronové sítě dokud nedosáhne požadované střední kvadratické chyby.

Do proměnné *net* se uloží data vytvořeného objektu neuronové sítě. V objektu *net* jsou uloženy veškeré informace o neuronové síti, tj. informace o topologii, aktivačních a chybových funkcí, synaptické váhy, atd.

Trénování neuronové sítě typu RBF je spuštěno příkazem *sim()*, který vrací výstup ze sítě. Výstup neuronové sítě je uložen do proměnné *ytren* a výstup na testovacích datech je uložen do proměnné *ytest* . Křivka vývoje chyby ukazuje, jak se v průběhu trénování mění chyba dosažená na trénovacích datech, viz obr. 10.



Obr. 10 - Průběh chyby, sc = 1

V průběhu trénování a testování neuronové sítě byly vypočteny hodnoty různých chyb - odchylek. Chyby hodnotí neshodu mezi výstupem ze sítě a požadovaným výstupem. Jedná se zejména o tyto chyby:

Celkový součet čtverců chyb *(TE)* – lze vypočítat podle [15], [17]:

$$TE = \sum(\bar{y} - y)^2, \tag{5.4}$$

kde $\bar{y}$ zatupuje průměrnou hodnotu požadovaných hodnot a y jsou požadované hodnoty. Celkový součet čtverců chyb TE obecně vysvětluje variabilitu náhodných veličin.

Střední kvadratická chyba *(E)* – lze získat podle [15], [17]:

$$E = \frac{1}{n}\sum(y - \hat{y})^2, \tag{5.5}$$

kde $\hat{y}$ odhadovaná hodnota, ted výstup z neuronové sítě. Střední kvadratická chyba *(E)* vypovídá o tom, jak se v průměru liší odhad od odhadované hodnoty.

Podle hodnoty odmocniny střední kvadratické chyby *MRSE* lze také určit kvalitu modelu. Platí, že čím je nižší hodnota *MRSE*, tím je model kvalitnější a lépe prokládá očekávané výstupy[19]

$$MRSE = \sqrt{E}. \tag{5.6}$$

Koeficient determinace (R^2) udává, jaký podíl v pozorování závislé proměnné se podařilo vysvětlit použitým regresním modelem. Hodnota je vždy kladná a pohybuje se v intervalu od 0 do 1. Koeficient je vypočten podle [15] , [17]:

$$R^2 = 1 - \frac{\sum(y - \hat{y})^2}{\sum(\bar{y} - y)^2}. \tag{5.7}$$

Z absolutních chyb byla měřena průměrná absolutní chyba a maximální absolutní chyba. Průměrná absolutní chyba určuje aritmetický průměr absolutních hodnot chyb znaku všech prvků souboru od aritmetického průměru hodnot znaku. Hodnotu AAE a MAE lze získat podle rovnic[15]

$$AAE = \frac{\sum|y - \hat{y}|}{n}, \tag{5.8}$$

$$MAE = \max|y - \hat{y}|. \tag{5.9}$$

Strojový kód v programovém prostředí Matlab pro výpočet uvedených chyb je následující:

```
    % 1_celkovy soucet ctvercu chyb TE
TE_tren=sum((mean(output_tren) - output_tren).*(mean(output_tren) -
output_tren));
TE_test=sum((mean(output_test) - output_test).*(mean(output_test) -
output_test));

    % 2_stredni kvadraticka chyba E
E_tren=sum((output_tren - ytren).*(output_tren - ytren))/1020;
E_test=sum((output_test - ytest).*(output_test - ytest))/510;

    % 3_hodnota MRSE
MRSE_tren=sqrt(sum((output_tren - ytren).*(output_tren - ytren))/1020);
MRSE_test=sqrt(sum((output_test - ytest).*(output_test - ytest))/510);

    % 4_koeficient determinace R2
r2_tren=1-(sum((output_tren - ytren).*(output_tren - ytren))/TE_tren);
r2_test=1-(sum((output_test - ytest).*(output_test - ytest))/TE_test);

    % 5_prumerna absolutni chyba
AAE_tren=sum(abs(output_tren - ytren))/1020;
AAE_test=sum(abs(output_test - ytest))/510;

    % 6_maximální absolutní chyba MAE
MAE_tren=max(abs(output_tren - ytren));
```

```
MAE_test=max(abs(output_test - ytest));
```

Celý strojový kód je uveden v příloze A. Následně bylo provedeno pětkrát trénování / testování neuronové sítě typu RBF. Cílem bylo zjistit, jakým způsobem bude tato neuronová síť reagovat na změnu hodnoty poloměru RBF funkce, tj. *sc*. Hodnota parametru poloměru (*sc*) byla nastavena na 1. Obr. 10 zobrazuje průběh chybové funkce při *sc* = 1. Grafy porovnání požadovaných hodnot a hodnoty získaných neuronovou sítí pro trénování jsou uvedeny v příloze B a pro testování jsou uvedeny v příloze C.

Naměřené chyby (průměry za pět experimentů) uvádí Tab. 5. Nejnižší hodnoty kvadratických chyb byly naměřeny u testovacích dat při hodnotě *sc* = 1, kdy se E = 0.1261 a *MRSE* = 0.3551. Při této velikosti sc byly naměřeny i nejmenší hodnoty absolutních chyb, tj. *AAE* = 0.1432 a *MAE* = 3.9031. Nejmenší hodnota celkového součtu čtverců chyb byla naměřena u testovacích dat při nastavení *sc* = 0.8. Koeficient determinace R^2 vykazoval nejlepší výsledek opět při nastavení sc = 1, kdy dosahoval hodnoty R^2 = 0.99811.

Tab. 5 - Hodnoty naměřených chyb pro změnu parametrů neuronové sítě typu RBF

sc	*TE*		*E*	
	O_{tren}	O_{test}	O_{tren}	O_{test}
1	67973	33965	0.00019562	0.1261
0.8	69174	32666	0.00019077	0.8386
0.6	68421	33506	0.00019011	2.7279
0.4	67123	34867	0.00007081	9.1379
0.2	68844	33050	0.00019064	10.8781
sc	*MRSE*		*AAE*	
	O_{tren}	O_{test}	O_{tren}	O_{test}
1	0.0139	0.3551	0.0089949	0.1432
0.8	0.0138	0.7991	0.0068983	0.2459
0.6	0.0137	1.6516	0.0059887	0.6304
0.4	0.0084	3.0229	0.0026083	1.4904
0.2	0.0138	6.9913	0.0039987	4.8581
sc	R^2		*MAE*	
	O_{tren}	O_{test}	O_{tren}	O_{test}
1	1	0.9981	0.0882	3.9031
0.8	1	0.9900	0.1393	10.1380
0.6	1	0.9584	0.1472	12.6430
0.4	1	0.8663	0.0989	19.0790
0.2	1	0.2457	0.1148	29.4290

5.4 Modelování pomocí dopředné neuronové sítě

Předmětem této podkapitoly je učení pomocí dopředné sítě. Tyto neuronové sítě jsou díky svým vlastnostem nejpoužívanějším typem neuronových sítí. Využívají algoritmus zpětného šíření chyby (Back-propagation) v různých podobách.

Nová neuronová dopředná síť se v programovém prostředí Matlab (Neural Network Toolbox-u) vytváří pomocí funkce *neff()*. Tento příkaz vyžaduje několik vstupních proměnných:

- rozsah vstupních hodnot - většinou se uvádí min. a max. hodnota tj. minimax (input_tren),
- počet neuronů v jednotlivých vrstvách, nejčastěji se používají tři vrstvy (vstupní, skrytá a výstupní), počet není omezen, zde je zvolen počet neuronů ve skryté vrstvě na m - 1 = 2 neuronů, kde m je počet vstupních proměnných,
- název aktivační funkce neuronů jednotlivých vrstev - nastavená na *tansig* (hyperbolický tangens), ale může být nastavena i na *logsig*, *purelin* (lineární funkce) atd.
- název trénovacího algoritmu - nastaven na *tainlm* (Levenberg-Marquard) ale může být nastavena na *traingd* (gradientní metoda – Gradient descent backproopagation), *traingdx* (obdoba gradientní metody)
- název funkce pro výpočet chyby při trénování.

Do objektu *net2* se uloží data vytvořeného objektu neuronové sítě. V objektu *net2* jsou uloženy veškeré informace o dopředné neuronové síti, tj. informace o topologii, aktivačních funkcích, atd.

Kód pro trénování / testování dopředné neuronové sítě je následující:

```
%navrh neuronove site
net2 = newff(minmax(input_tren), [2 1], {'tansig','purelin'}, 'trainlm');

%nastaveni poctu epoch
net2.trainParam.epochs = 1000;
%nastaveni pozadovane chyby eg
net2. trainParam.goal = 0.2;

%trenovani site pomoci algoritmu Back-propagation
[net2,tr] = train(net2,input_tren,output_tren);
%vysledek trenovani
ytren_dopredna = sim(net2,input_tren);
%vysledek testovani
ytest_dopredna = sim(net2,input_test);
```

Proces trénování je spuštěn pomocí příkazu *train()*. Je nutné uvést jeho proměnné:

- *net* – název neuronové sítě, která bude použita pro trénování,
- *P* – vstupní množina trénovacích hodnot,
- *T* – výstupní množina požadovaných hodnot.

Je dobré, ještě před příkazem pro trénování neuronové sítě, uvést parametry trénování (funkce *train()*), které se mohou měnit:

- *net.trainFcn* – název trénovacího algoritmu, který bude použit, lze dosadit např. *trainlm* atd.,
- *net.trainParam.epochs* – udává, kolik epoch se neuronová síť bude učit, tento parametr byl měněn nejdříve na hodnotu 5000, poté 2000, 1000, 500 a 50,
- *net.trainParam.show* – udává, jak často se má chybová funkce vykreslovat v grafu,
- *net.trainParam.lr* – udává rychlost učení, čili jak se mění hodnoty synaptických vah a prahů v závislosti na chybové funkci, zde *lr* = 0.01.
- *net.trainParam.goal* – udává, při jaké hodnotě chybové funkce se má trénování ukončit, nastaveno na *eg* = 0.2.

Tyto parametry jsou uloženy v datové struktuře objektu *net2*. Celý strojový kód se nachází v příloze A.

Výsledek trénování a testování dopředné neuronové sítě je uložen do proměnné *ytren_dopredna a ytest_dopredna*. Při měření byl měněn parametr počtu epoch. Výsledky neuronové sítě byly opět porovnány s požadovanými hodnotami.

Hodnoty naměřených chyb (opět průměry za pět experimentů) uvádí Tab. 6.

Nejnižší hodnoty kvadratických chyb byly naměřeny u testovacích dat při počtu epoch = 5000, kdy se E = 0.8983 a $MRSE$ = 0.9478. Při tomto počtu epoch byly naměřeny i nejmenší hodnoty absolutních chyb, tj. AAE = 0.7595 a MAE = 2.9282. Nejlepší hodnota koeficientu determinace R^2 byla naměřena při stejném počtu epoch, kdy se rovnal hodnotě R^2 = 0.9861.

Tab. 6 – Hodnoty naměřených chyb u dopředné sítě při změně počtu epoch

Počet epoch	*TE*		*E*	
	O_{tren}	O_{test}	O_{tren}	O_{test}
5000	68628	32934	0.9459	0.8983
2000	67076	34795	3.8153	4.5740
1000	67772	34143	4.1458	3.7123
500	66516	35451	3.3960	3.5948
50	69476	32497	4.1369	3.9102
Počet epoch	*MRSE*		*AAE*	
	O_{tren}	O_{test}	O_{tren}	O_{test}
5000	0.9726	0.9478	0.7766	0.7595
2000	1.9533	4.5740	1.4733	1.5833
1000	2.0361	1.9267	1.5343	1.4566
500	1.8428	1.8960	1.4108	1.4526
50	2.0339	1.9774	1.6576	1.4778
Počet epoch	R^2		*MAE*	
	O_{tren}	O_{test}	O_{tren}	O_{test}
5000	0.9860	0.9861	3.3922	2.9282
2000	0.9420	0.9330	9.1910	8.1400
1000	0.9376	0.9446	9.1376	7.2891
500	0.9479	0.9483	7.6919	6.6360
50	0.9393	0.9386	8.9538	8.1901

5.5 Aplikace Black-Scholes-ova modelu

Následně byly ceny Call opcí měřeny pomocí Black-Scholes-ova modelu. Programové prostředí Matlab nabízí pro výpočet cen opcí Financial Toolbox. Pro realizaci modelu byla použita funkce blsprice(). Funkce vyžaduje zadat následující vstupní proměnné:

- spotová cena podkladového aktiva *S* - stanovena na 100 $,
- realizační cena *X*,
- bezriziková úroková míra *r* - stanovena na 0%,
- doba splatnosti opce *t*,
- volatilita σ – pro zjednodušené značení v programovém prostředí Matlab označena jako *v*.

Strojový kód pro výpočet ceny opce *O* pomocí Black-Scholes-ova modelu je následující

```
%BLACK SCHOLES-UV MODEL
%realizacni cena X;vyber z workspace output
X = input(3,1:1530);
%cas t musi byt prepocitan na dny;
t = input(2,1:1530)/ 365;
%hodnota volatility brana z input;
v = input(1,1:1530);
%black scholes-uv model pro vypocet ceny PUT/CALL opci
[Call, Put]= blsprice(100,X,0,t,v);
```

Díky této funkci lze získat jak cenu Call opce tak cenu Put opce. Předmětem tohoto měření je jen cena Call opce, která byla porovnána s požadovanými hodnotami (output). Průběh požadovaných hodnot a hodnot získaných pomocí Black-Scholes-ova modelu v Matlabu je znázorněn na grafu v příloze D a celý strojový kód i pro výpočet ukazatelů chyb je v příloze E. Naměřené chyby jsou uvedeny v Tab. 7.

5.6 Porovnání modelů pro oceňování finančních Call opcí

Všeobecně se dává přednost modelu s nejnižšími hodnotami uvedených ukazatelů chyb. Je však důležité brát v potaz, že žádný z těchto ukazatelů nemá univerzální charakter, ale podává pouze dílčí informaci o kvalitě hodnoceného modelu. Pro komparaci těchto modelů lze v zásadě použít všech prezentovaných ukazatelů chyb.

Nejnižší hodnoty měřených chyb v tomto porovnání, vykazoval model neuronové sítě typu RBF. Model dopředné neuronové sítě vykazoval nižší hodnoty chyb než Black-Schole-sův model.

Tab. 7 – Hodnoty naměřených chyb – porovnání všech modelů

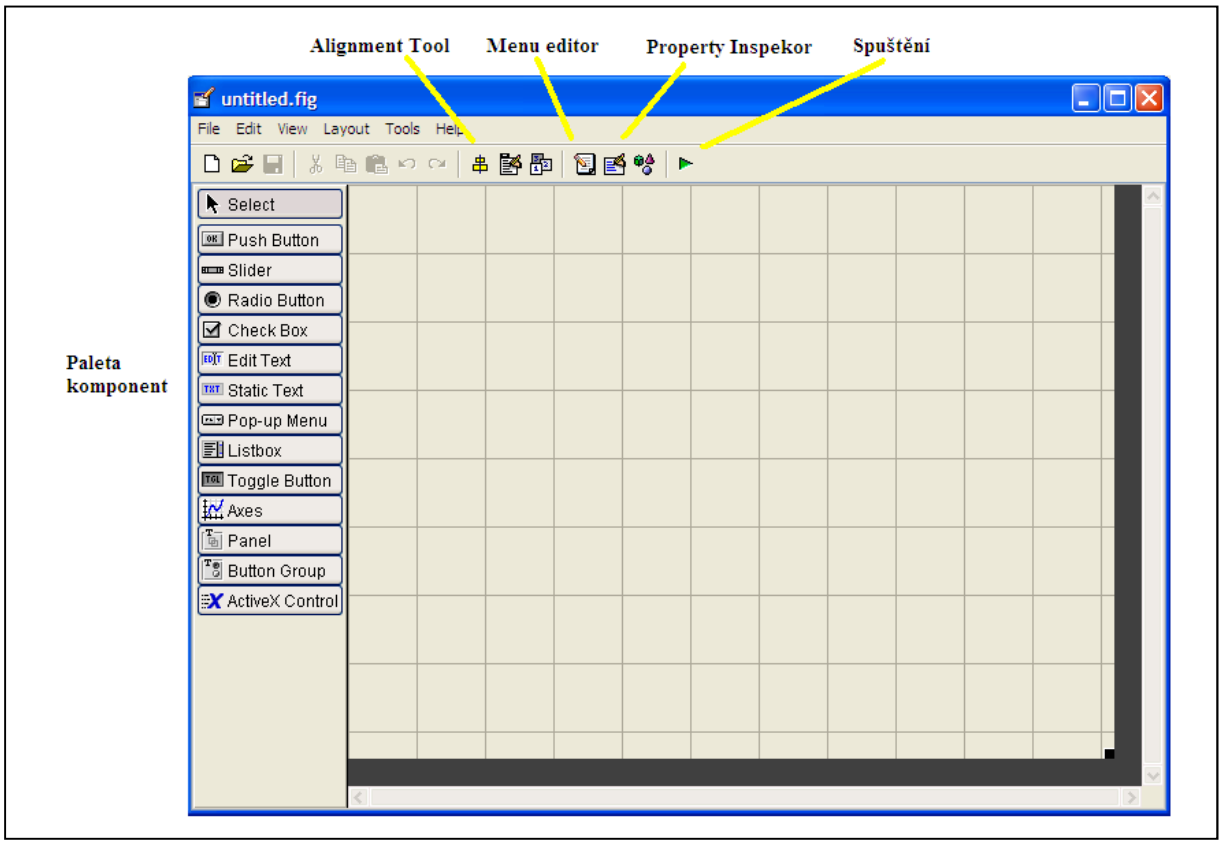
Chyba	Black-Scholes-ův model	RBF neuronová síť	Dopředná neuronová síť
TE	101 990.0000	33 965.0000	32 934.0000
E	2.0839	0.1261	0.8983
MRSE	1.4436	0.3551	0.9478
AAE	1.7058	0.1432	0.7595
R^2	0.9687	0.9900	0.9861
MAE	3.2906	3.9031	2.9282

6 Tvorba grafického uživatelského prostředí

Pomocí vývojového prostředí GUIDE v programovém prostředí Matlab 7.1 byla vytvořena aplikace „opce“. Tato aplikace je použita na navržené modely pro oceňování opcí pomocí neuronové sítě typu RBF, dopředné sítě a Black-Scholes-ovým modelem.

GUIDE umožňuje vytvářet, editovat uživatelský interface a to prostřednictvím list boxů, push buttonů, radio buttonů, checkboxů, atd. Dále obsahuje řadu nástrojů pro přidání a zakládání objektů v okně návrhu (*LayoutEditor*), nástroje pro zarovnání a rozmístění objektů s ohledem na různé části návrhu (*Alignemt Tool*), nastavení hodnot objektů a dohled nad nimi (*Property Inspektor*), vytváření menu (*Menu Editor).*

Zadáním příkazu *guide* se spustí prostředí GUIDE. Jako první se zobrazí okno *Layout Edit*or, který je vidět na Obr. 11.

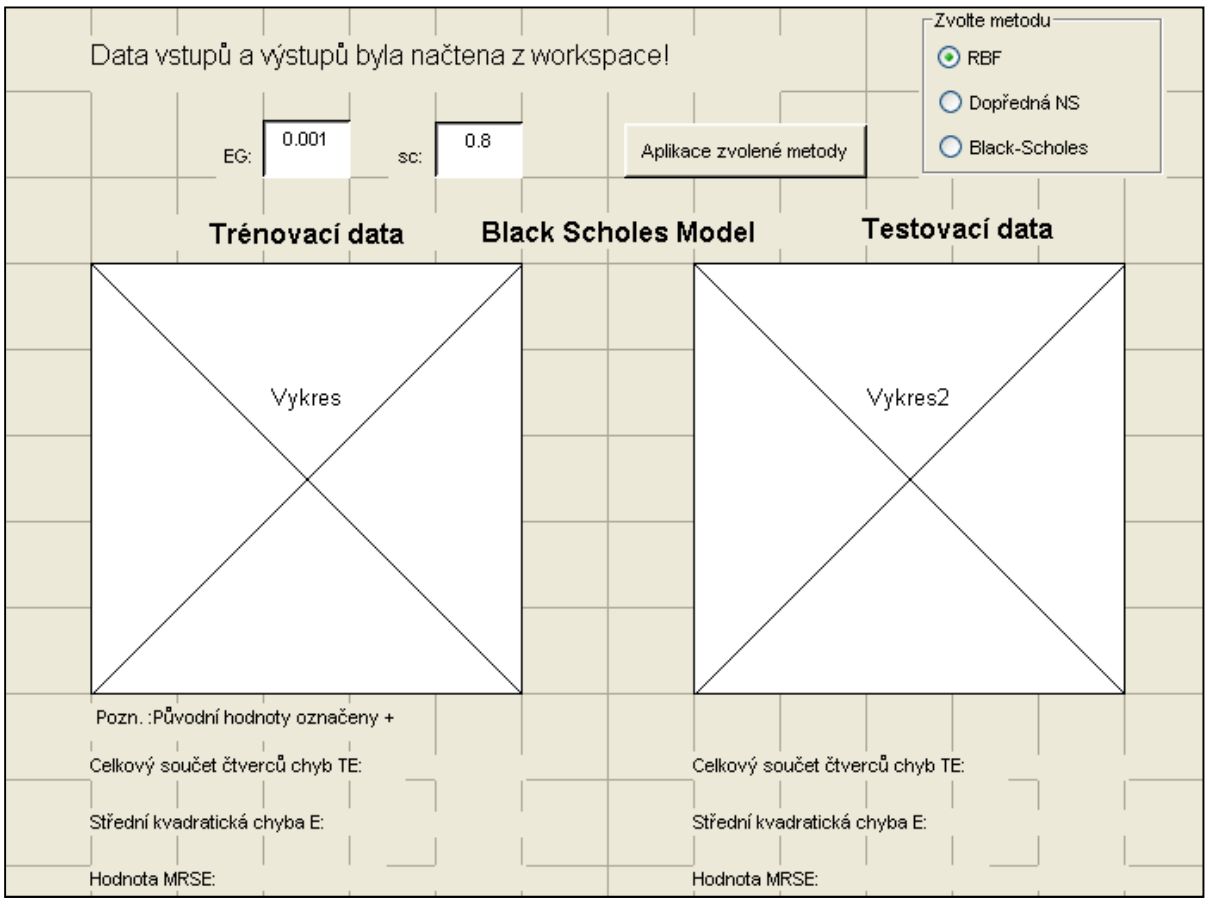


Obr. 11- Layout Editor

Ten slouží jako ovládací panel pro GUIDE. Díky němu lze jednoduše vybrat GUI komponenty z palety a uspořádat je v okně pro návrh. Pro úpravu vlastností vestavěných komponent slouží *Property Inspektor* a *Alignment Tool*, které se spustí podle vyznačení žlutou

čarou na Obr. 11. Vytvořený návrh musí být uložen jako soubor s příponou *.fig. Po uložení se automaticky generuje kód vestavěných komponent. K němu se lze vrátit přes *Menu editor*.

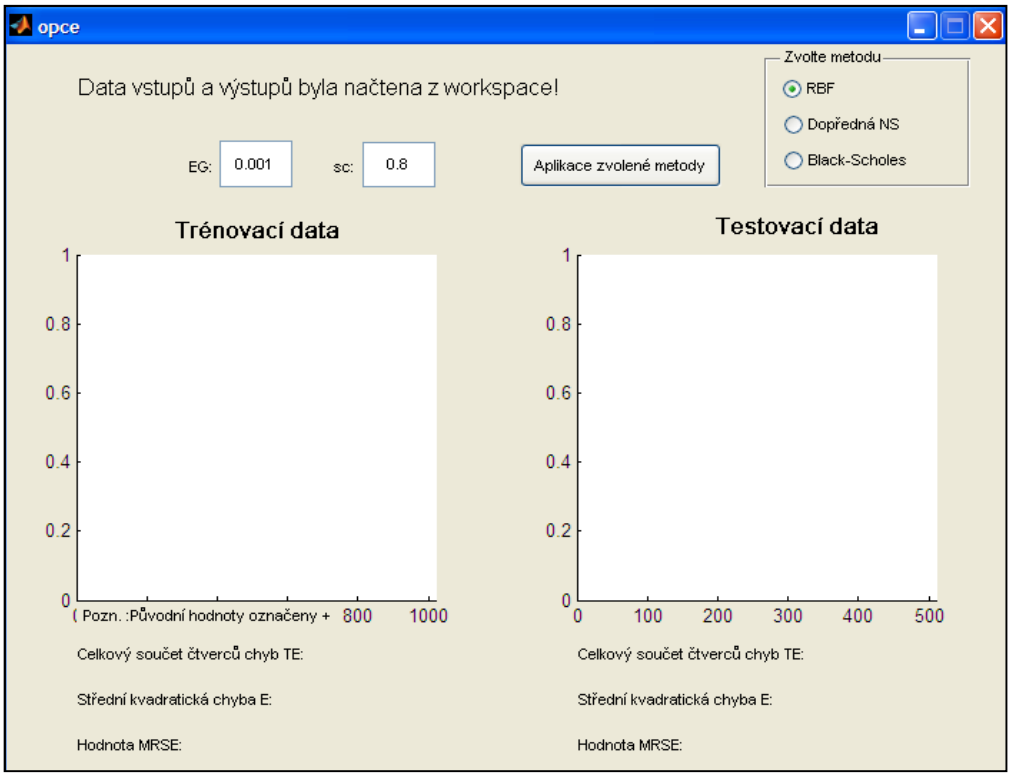
Návrh aplikace „opce“ je zobrazen na Obr. 12.



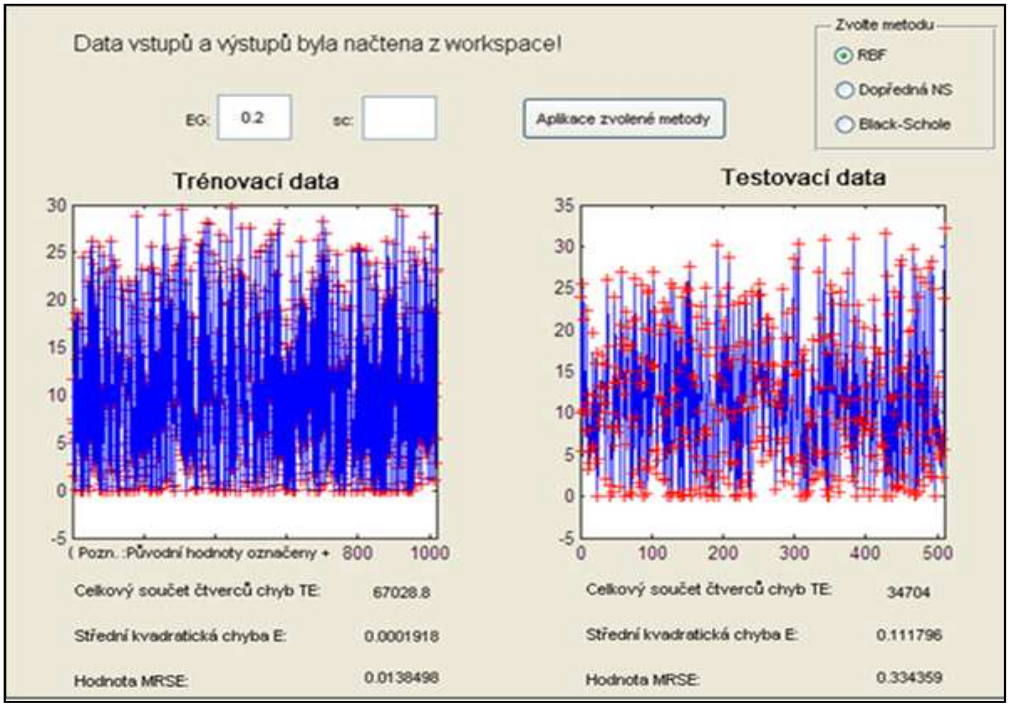
Obr. 12 - Návrh aplikace "opce"

Vytvořený strojový kód je uveden v příloze F. Finální podoba aplikace “opce” je znázorněna na Obr. 13.

Data jsou do aplikace „opce“ načtena z pracovní plochy (Workspace). Při zvolení tlačítka „Aplikace zvolené metody“ se provede nejen učení pomocí zvolené metody, ale i vše potřebné pro správné učení sítě, tj. předzpracování dat a rozdělení množiny na trénovací a testovací data. Pokud je zvoleno učení pomocí neuronové sítě typu RBF, je zapotřebí vyplnit hodnotu požadované chyby označenou jako *EG* a hodnotu poloměru RBF funkce označenou jako *sc*. Obr. 14 zobrazuje vykreslení trénovacích a testovacích dat, při zvolení RBF neuronové sítě, kde byla požadovaná hodnota *EG* nastavena na *EG* = 0.2 a hodnota poloměru RBF funkce na 1. Pod grafy jsou vypočteny chyby pro trénovací a testovací data.

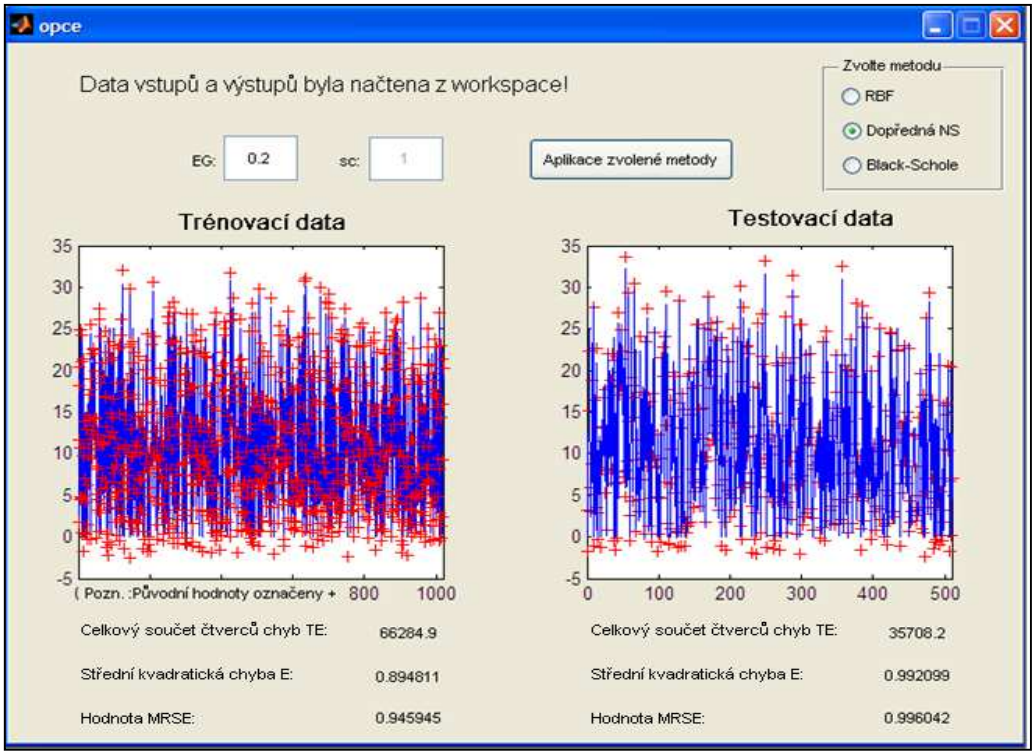


Obr. 13 - Finální podoba aplikace "opce"



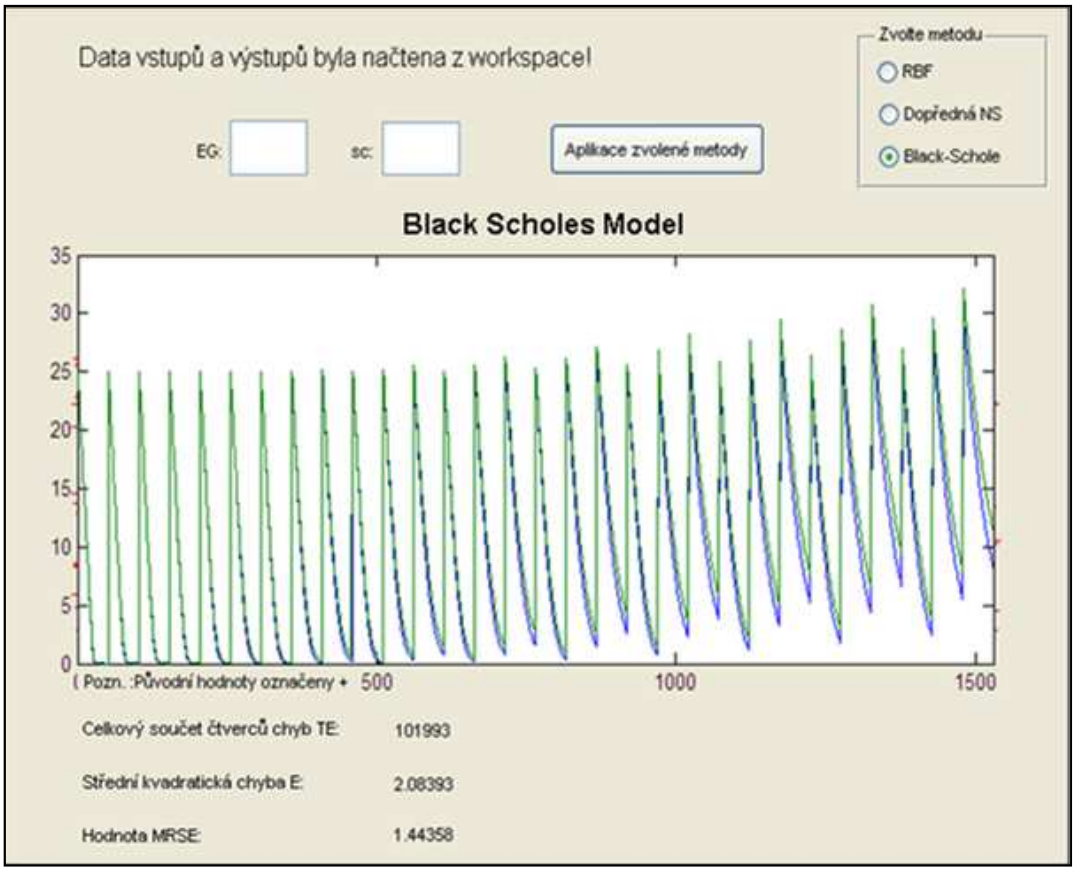
Obr. 14 - Aplikace "opce" při RBF

Pokud je zvolena dopředná neuronová síť, je nutné vyplnit pouze hodnotu požadované chyby *EG*. Hodnota poloměru RBF funkce *sc* nelze vyplnit, jak uvádí následující Obr. 15, na kterém byla hodnota opět nastavena na *EG* = 0.2.



Obr. 15 - Aplikace "opce" – dopředná neuronová síť

Obr. 16 obrazuje měření dat pomocí Black-Scholes-ova modelu. Opět je vykreslen průběh hodnot získaných výpočtem Black-Scholes-ova modelu a průběh požadovaných hodnot. Při zvolení modelu Black-Scholes se nezadává ani hodnota *sc* ani hodnota *EG*, nejedná se o neuronovou síť.



Obr. 16 - Aplikace "opce" - Black-Scholes-ův model

Závěr

V práci je charakterizován současný stav oceňování finančních opcí, zejména pak v praxi nejčastěji používané modely, Black-Scholes-ův a binomický. Ke splnění stanoveného cíle bylo zapotřebí nastudovat problematiku finančních opcí a jejich oceňování. Dále jsou v práci charakterizovány základní vlastnosti neuronových sítí a podrobně popsána topologie a učení neuronové sítě typu RBF. Dále byl cílem diplomové práce návrh modelu pro oceňování finančních opcí. Tento model byl v práci navržen a zahrnuje sběr dat, jejich předzpracování, návrh vhodných struktur neuronové sítě typu RBF a porovnání výsledků s dalšími modely pro oceňování finančních opcí. Pro porovnání výsledků byly použity modely dopředné neuronové sítě a Black-Scholes-ův model. Modely byly verifikovány v programovém prostředí Matlab 7.1. Oproti stanoveným cílům bylo navíc pro uvedenou verifikaci navrženo grafické uživatelské prostředí „opce“.

Jako vstupy modelů bylo použito 1530 hodnot proměnných získaných z uměle vytvořené databáze Worden TC 2000. Za vstupní proměnné byly zvoleny faktory, které zásadně ovlivňují cenu opce tj. volatilita, čas do splatnosti opce, realizační cena, spotová cena a úroková míra. Výstupem modelu byla jednoznačně určena cena opce. Data použitá pro tvorbu modelu byla vhodně předzpracována. Pro realizaci návrhu modelu oceňování finančních opcí pomocí neuronové sítě typu RBF bylo využito programového prostředí Matlab 7.1. Nejdříve byla vytvořena neuronová síť typu RBF funkcí *newrb()*, která byla následně trénována a testována. V průběhu trénování a testování neuronové sítě typu RBF, byla měněna hodnota poloměru funkce RBF $h_i(x)$. Výstupy z neuronové sítě byly posouzeny z hlediska naměřených chyb. Jednalo se o celkový součet čtverců chyb *TE*, střední kvadratickou chybu *E* a její odmocninu *RMSE*, absolutní průměrnou, maximální odchylku a koeficient determinace. Nejlepších výsledků bylo dosaženo pro velikost okolí RBF funkce $sc = 1$.

Pro objektivní posouzení byly vytvořeny další modely, a to model oceňování finančních opcí za pomocí dopředné neuronové sítě a Black-Scholes-ův model. Dopředná neuronová síť byla vytvořena pomocí funkce *newff()* a trénována pomocí funkce *train ()*. V průběhu trénování a testování byl měněn parametr počtu epoch. Nejlepších výsledků bylo pro danou rychlost učení $lr = 0.01$ dosaženo po 5000 epochách. Programové prostředí Matlab nabízí pro výpočet cen pomocí Black-Scholes-ova modelu funkce obsažené ve Financial Toolbox. Cena Call opce podle Black-Scholes-ova modelu byla získána zadáním funkce *blsprice()*. I pro tento model byly vypočteny jmenované ukazatele chyb.

Všechny vypočtené ukazatele chyb byly mezi sebou pozouzeny a vyhodnoceny. Nejlepší model, který určil cenu Call opce s nejnižšími hodnotami chyb, byl model oceňování finančních opcí za pomocí neuronové sítě typu RBF, poté následoval model využívající dopřednou neuronovou síť. Nejvyšší hodnoty chyb vykazoval Black-Scholes-ův model.

Navržené modely byly implementovány do aplikace „opce“. Aplikace byla vytvořena ve vývojové prostředí GUIDE programového prostředí Matlab 7.1. Tato aplikace umožňuje uživateli zvolit metodu určení ceny finanční opce. Aplikace vykreslí průběh požadovaných hodnot a hodnot získaných ze zvolené metody a vypočte ukazatele chyb.

Seznam použité literatury:

[1] AMBROŽ, Luděk. Oceňování opcí. Praha : C.H.Beck, 2002. 313 s. ISBN 80-7179-531-3

[2] BERKA, Petr. Neuronové sítě. In Dobývání znalostí. Praha : Accademia, 2003. s. 213. ISBN 80-200-1062-9.

[3] CIMLER, JIří. OptionLock [online]. 2009 [cit. 2010-11-23]. Opce. Dostupné z WWW: <http://www.optionslock.cz/opce/opce.php>.

[4] DVOŘÁK, Petr. Finanční deriváty. druhé. Praha : Vysoká škola ekonomická v Praze, 1996. Finanční opce, s. 217. ISBN 80-7079-139.

[5] DVOŘÁK, Petr. Deriváty. Praha : Nkladatelství Oeconomica, 2006. Opce, s. 296.

[6] ENDORF, Carl. Detekce a prevence počítačového útoku [online]. Praha : Grada Publishing a.s., 2005 [cit. 2011-03-23]. Dostupné z WWW: http://books.google.cz/books

[7] Finance : Kapitálový trh [online]. 21.5.2009 [cit. 2010-11-21]. Opce - specifický druh derivátů. Dostupné z WWW: <http://www.finance.cz>

[8] GIROSI, F., NIYOGI, P. On the Relationship Between Generalization Error, Hypothesis Complexity, and Sample Complexity for Radial Basis Functions[online]. 1994 [cit. 2010-02-12]. Dostupný z WWW: <ftp://publications.ai.mit.edu/ai-publications/pdf/AIM-1467.pdf>.

[9] GREGOR, Leoš. Ověření ocenění opcí metodou Quasi-Monte-Carlo. 5.mezinárodní konference Fiannční řízení podniku a finančích institucí [online]. 2005, 5, [cit. 2011-05-01]. Dostupný z WWW: http://www.ekf.vsb.cz

[10] HAYKIN, S. Neural Networks: A Comprehensive Foundation. 2nd edition, New Jersey, Prentice-Hall, Inc., 1999, 842s. ISBN 81-7808-300-0.

[11] HÁJEK, P.: Řízení znalostí. Pardubice : Univerzita Pardubice, 2011

[12] Investopedia News and Articles [online]. 2011 [cit. 2011-04-12]. Investopedia. Dostupné z WWW: <http://books.google.cz/books>.

[13] KATZ, Jeffrey Owen. Option pricing models : An Empirical Approach to Valuing Options. New York : Scientific Consultatn Services, 2001. 437 s.

[14] KOŠŤÁL, Josef; TUREK, Ludvík. Opce : Jak na obchodování s opcemi a výběr správné strategie. první. Brno : Computer Press, a.s., 2009. Historie novodobých opcí, s. 152. ISBN 978-80-251-2223-5.

[15] KUBANOVÁ, Jana. Statistické metody pro ekonomickou a technickou praxi. Bratislava : STATIS, 2004. 249 s. ISBN 80-86659-37-9.

[16] KVASNIČKA, V. a kol. Úvod do teorie neurónových sietí. Bratislava: IRIS, 1997. 285 s.

[17] KELBA, J., ŠILHÁN, D. Shluková analýza [online]. 2009 [cit. 2010-02-12]. Dostupný z WWW: <http://staff.utia.cas.cz

[18] MALÝ, Josef. Oceňování průmyslového vlastnictví. Praha : C.H.Beck, 2007. 179 s.

[19] Math interactive [online]. 2007 [cit. 2011-03-20]. Math interactive. Dostupné z WWW: <http://www.math-interactive.com>.

[20] MathWorks [online]. 1994-2011 [cit. 2011-05-01]. MathWorks. Dostupné z WWW: <http://www.mathworks.com/index.html>.

[21] NOVÁK, M. a kol. Umělé neuronové sítě teori a aplikace. Praha: C.H.Beck, 1998. 382 s. ISBN 80-7179-132-6.

[22] OLEJ, Vladimír; HÁJEK, Petr. Úvod do umělé inteligence. Pardubice : Univerzita Pardubice, 2010. 98 s.

[23] OLEJ, V. Modelovanie ekonomických procesov na báze výpočtovej inteliegencie. [Vedecká monografia], Miloš Vognar MV, ISBN 80-903024-9-1, Hradec Králové, Česká republika, 2003, 160s

[24] PARK, J., SANDBERG, I. W. Universal Approximation Using Radial-Basis-Function Networks. Neural Computation. 1991, vol. 3, no.2, pp. 246-257

[25] ROHRBACHER, Jan. Finance [online]. 28.05.2009 [cit. 2010-11-21]. Kapitálový trh. Dostupné z WWW: http://www.finance.cz/zpravy/finance

[26] ŘEZANKOVÁ, H., HÚSEK, D., SNÁŠEL, V. Shluková analýza dat. Praha: Professional Publishing, 2007. 196 s. ISBN 978-80-86946-26-9

[27] Scientific consultants services inc. Black-Scholes Test Data For Neural Networks [online]. 2009 [cit. 2011-05-02]. Scientific consultants services inc. Dostupné z WWW: <http://www.scientific-consultants.com/nnbd.html>.

[28] SCHOLLEOVÁ, Hana. Hodnota flexibility: Reálné opce. 1. vydání. Praha : Nakladatelství C H Beck, 2007. 171 s

[29] SCHOLLEOVÁ, H. Hodnota flexibility: Reálné opce. 1. vydání. Praha : Nakladatelství C H Beck, 2007. 171 s

[30] SOUČEK, Ivan. Podnikatelský záměr a investiční rozhodování. Praha : Grada, 2005. 356 s.

[31] Stochastické modelování úrokových sazeb. In PAPEŽ, Michal. Stochastické modelování úrokových sazeb [online]. Praha : UnitCreditGroup, 2010, 2010 [cit. 2011-04-13]. Dostupné z WWW: <http://www2.humusoft.cz/www/papers>

[32] ŠENČOVIČ Daniel. Analytické a numerické metódy oceňovania finančných derivátov, Nakladateĺstvo STU, Bratislava 2009, 200 s. 209, ISBN 978-80-227-3014-3

[33] ŠÍMA, Jiří; NERUDA, Roman. Teoretické otázky neuronových sítí. Praha : MATFYZPRESS, 1996. 390 s. Dostupné z WWW: http://www2.cs.cas.cz/~sima

[34] TAUFER, I. , et al. UMĚLÉ NEURONOVÉ SÍTĚ – ZÁKLADY TEORIE A APLIKACE (14). CHEMagazín. 2009 - XIX, XIX, 1, s. 26-27.

[35] WAJSZCZUK, Jerzey. Międzynarodowe środowisko finansowe: kierunki instytucjonalizacji [online]. Warszawa : Key Text Wydawnictwo, 2005 [cit. 2011-04-13]. Dostupné z WWW: <http://books.google.com/books>.

[36] ZÁŠKODNÝ, Přemysl, et al. Finanční deriváty a jejich oceňování. první. Praha : Vysoká škola finanční a správní v edici EUPRESS, 2007. 162 s

Seznam tabulek:

Tab. 1 - využití Call opce, zdroj: [3] 13
Tab. 2 - Pozice Short Call, zdroj: [3] 14
Tab. 3 – Pozice Long Put, zdroj: [3] 15
Tab. 4 - Pozice Short put, zdroj: [3] 16
Tab. 5 - Hodnoty naměřených chyb pro změnu parametrů neuronové sítě typu RBF 49
Tab. 6 – Hodnoty naměřených chyb u dopředné sítě při změně počtu epoch 50
Tab. 7 – Hodnoty naměřených chyb – porovnání všech modelů 53

Seznam obrázků:

Obr. 1 - Pozice Long Call, zdroj: [25] 13
Obr. 2 - Pozice Short Call, zdroj: [25] 15
Obr. 3 - Binomický model pro akcii nevyplácející dividendy, zdroj: [1] 26
Obr. 4 - Binomický model, zdroj: [35] 28
Obr. 5 - McCulloch-Pitts-ův neuron, zdroj: [21] 30
Obr. 6 - Návrh modelu pro oceňování finančních opcí 40
Obr. 7 - Získaná data 42
Obr. 8 - Statistika dat 43
Obr. 9 - Rozklad množiny vstupních dat, zdroj: [22] 45
Obr. 10 - Průběh chyby, sc = 1 47
Obr. 11- Layout Editor 54
Obr. 12 - Návrh aplikace "opce" 55
Obr. 13 - Finální podoba aplikace "opce" 56
Obr. 14 - Aplikace "opce" při RBF 56
Obr. 15 - Aplikace "opce" – dopředná neuronová síť 57
Obr. 16 - Aplikace "opce" - Black-Scholes-ův model 58

Seznam příloh:

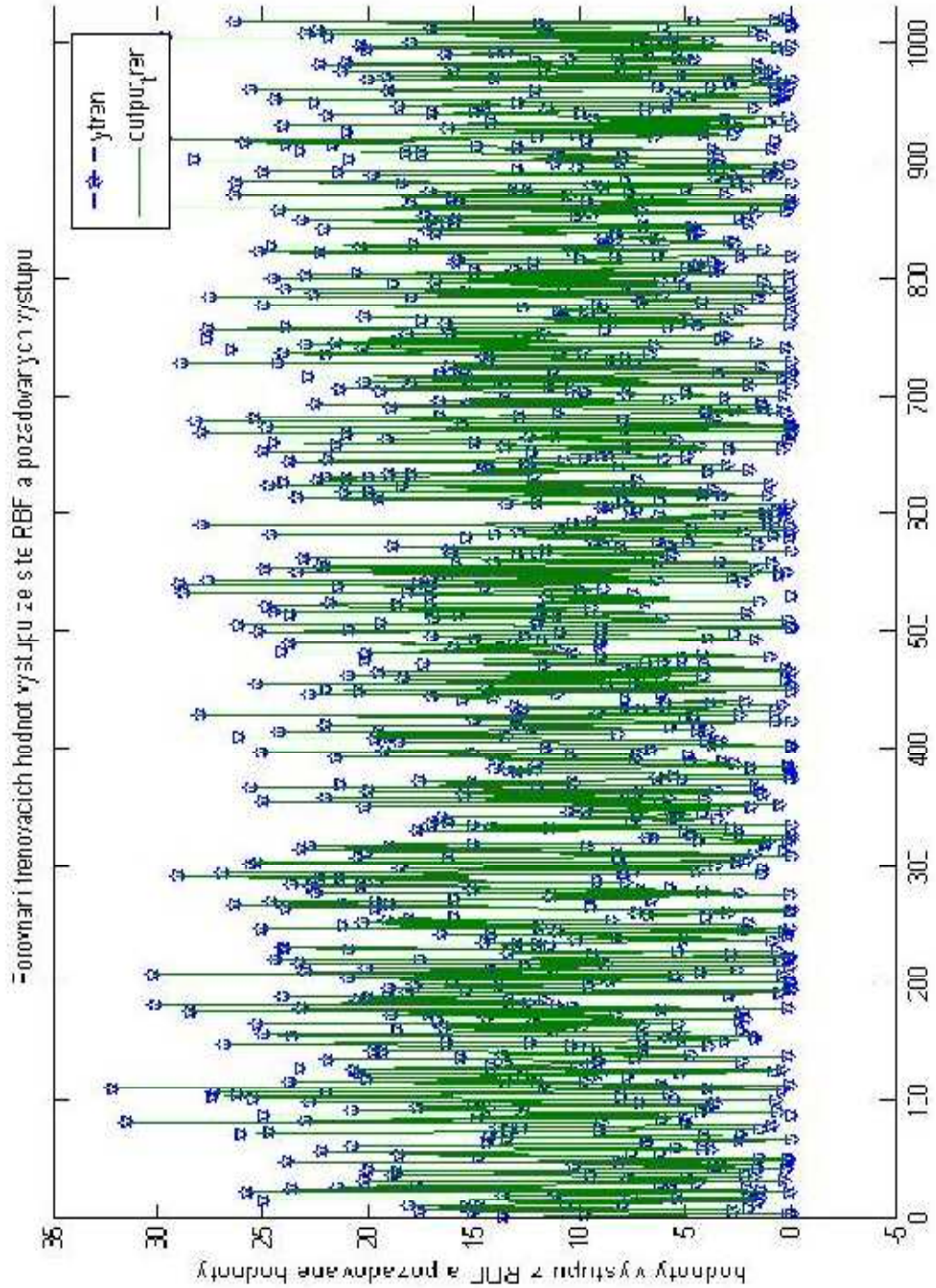
PŘÍLOHA A 67
PŘÍLOHA B 69
PŘÍLOHA C 70
PŘÍLOHA D 71
PŘÍLOHA E 72
PŘÍLOHA F 73

PŘÍLOHA A

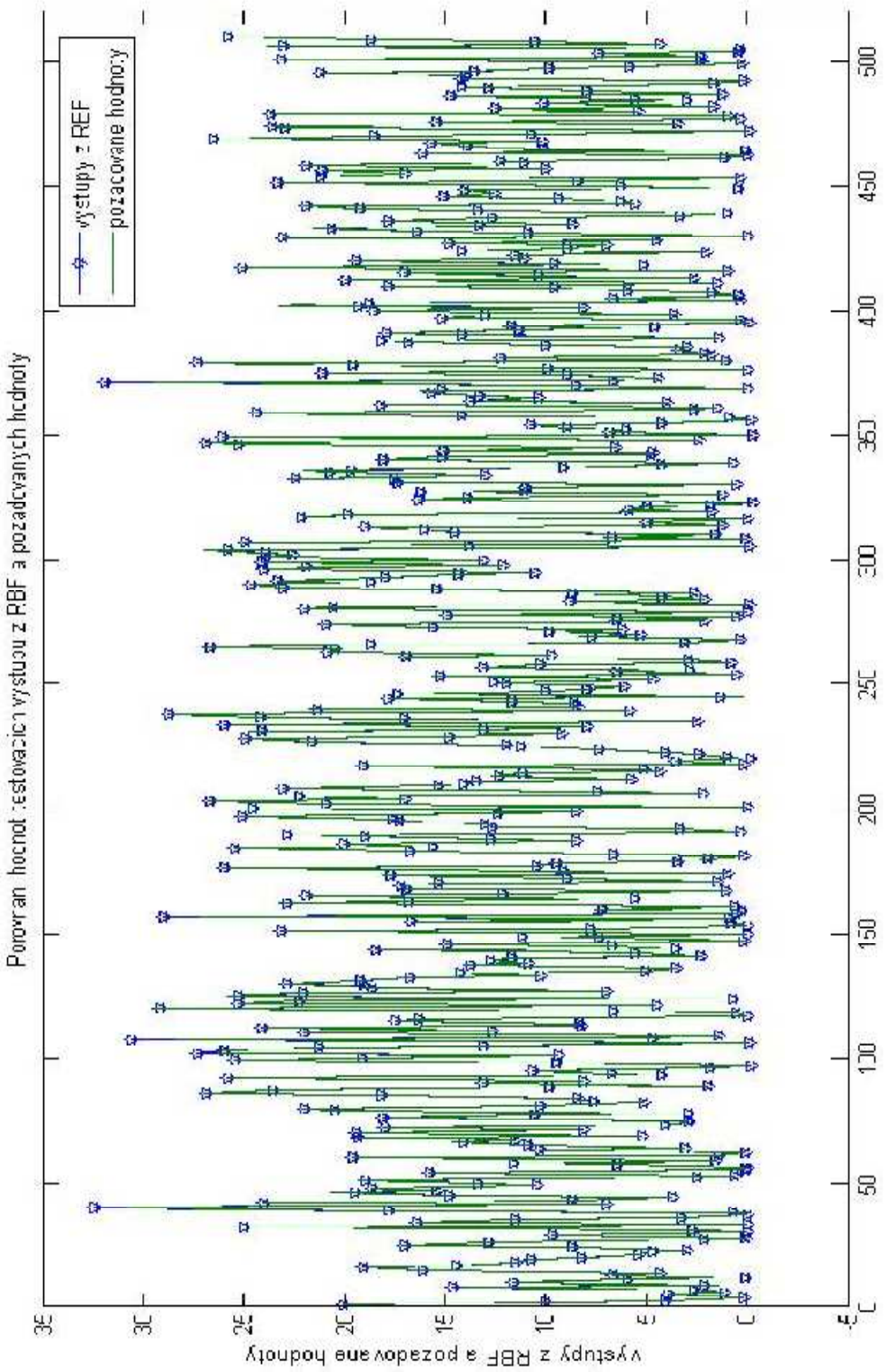
```
% ROZDELENI DAT NA TRENOVACI A TESTOVACI MNOZINU
%   nahodna permutace
perm = randperm(1530)';
%vstupni dat
input = input(:,perm);
%vystupni dat
output = output(:,perm);
%   vstupni trenovaci
input_tren = input(:,1:1020);
%   vstupni testovaci
input_test  = input(:,1021:end);
%   vystupni trenovaci
output_tren = output(:,1:1020);
%   vystupni testovaci
output_test  = output(:,1021:end);

%hodnota pozadovanee chyby
eg = 0.2;
%hodnota sireni pro RBF
sc = 0.8; %hodnota polomeru RBF funkce

%TESTOVANI/TRENOVANI SITE RBF
%tvorba neuronove site RBF
net = newrb(input_tren,output_tren,eg,sc);

ytren = sim(net, input_tren);               %   vysledek trenovani
ytest = sim(net,input_test);                %   vysledek testovani

    % 1_celkovy soucet ctvercu chyb TE
TE_tren=sum((mean(output_tren) - output_tren).*(mean(output_tren) -
output_tren));
TE_test=sum((mean(output_test) - output_test).*(mean(output_test) -
output_test));

    % 2_stredni kvadraticka chyba E
E_tren=sum((output_tren - ytren).*(output_tren - ytren))/1020;
E_test=sum((output_test - ytest).*(output_test - ytest))/510;

    % 3_sMRSE
MRSE_tren=sqrt(sum((output_tren - ytren).*(output_tren - ytren))/1020);
MRSE_test=sqrt(sum((output_test - ytest).*(output_test - ytest))/510);

    % 4_koeficient determinace pro RBF
r2_tren=1-(sum((output_tren - ytren).*(output_tren - ytren))/TE_tren);
r2_test=1-(sum((output_test - ytest).*(output_test - ytest))/TE_test);

    % 5_prumerna absolutni chyba pro RBF
AAE_tren=sum(abs(output_tren - ytren))/1020;
AAE_test=sum(abs(output_test - ytest))/510;

    % 6_maximální absolutní chyb MAE dopsat do textu
MAE_tren=max(abs(output_tren - ytren));
MAE_test=max(abs(output_test - ytest));
```

```
%-----------------------------------------------------------------
%TRENOVANI  / TESTOVANI DOPREDNE NEURONOVE SITE

    %tvorba neuronové sítě FF
net2 = newff(minmax(input_tren), [2 1], {'tansig','purelin'}, 'trainlm');
%dopředná síť, tansig je hyperbolicky tangens, purelin je linearni fce
%nastavení počtu epoch
net2.trainParam.epochs = 100;
%nastavení pozadovane chyby
net2. trainParam.goal = 0.2;
%trénování sítě
[net2,tr] = train(net2,input_tren,output_tren);
%výsledek trénování
ytren_dopredna = sim(net2,input_tren);
%výsledek testování
ytest_dopredna = sim(net2,input_test);

    % VYPOCET CHYB
    % 1_vypocet celkovych souctu ctvercu chyb TE pro FF
mtren_dopredna=sum((mean(output_tren) - output_tren).*(mean(output_tren) -
output_tren));
mtest_dopredna=sum((mean(output_test) - output_test).*(mean(output_test) -
output_test));

    % 2_strední kvadratickA chyba E pro FF
mse_tren_dopredna=sum((output_tren - ytren_dopredna).*(output_tren -
ytren_dopredna))/1020;
mse_test_dopredna=sum((output_test - ytest_dopredna).*(output_test -
ytest_dopredna))/510;

    % 3_RMSE pro FF
mrse_tren_dopredna=sqrt(sum((output_tren - ytren_dopredna).*(output_tren -
ytren_dopredna))/1020);
mrse_test_dopredna=sqrt(sum((output_test - ytest_dopredna).*(output_test -
ytest_dopredna))/510);

    % 4_koeficient determinace pro FF
r2_tren_dopredna=1-(sum((output_tren - ytren_dopredna).*(output_tren -
ytren_dopredna))/mtren_dopredna);
r2_test_dopredna=1-(sum((output_test - ytest_dopredna).*(output_test -
ytest_dopredna))/mtest_dopredna);

    % 5_pumerna absolutni chyba pro FF
aae_tren_dopredna=sum(abs(output_tren - ytren_dopredna))/1020;
aae_test_dopredna=sum(abs(output_test - ytest_dopredna))/510;

    % 6_maximální absolutní chyba MAE
mae_tren_dopredna=max(abs(output_tren - ytren_dopredna));
mae_test_dopredna=max(abs(output_test - ytest_dopredna));
```

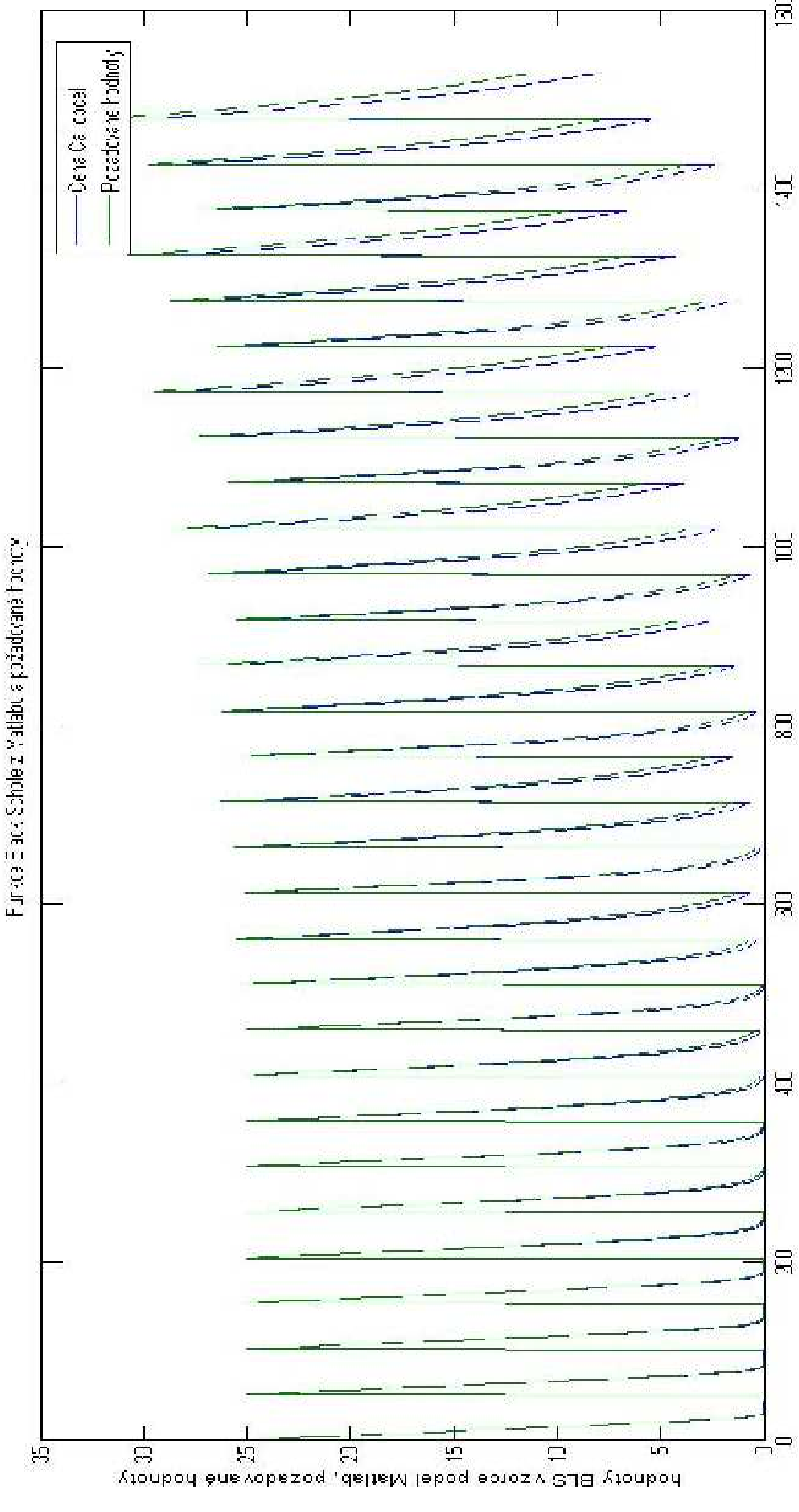
PŘÍLOHA B


Porovnání trenovacich hodnot výstupu ze sítě RBF a požadovaných výstupu
ytran
outputrar
hodnoty výstupu z RBF a požadované hodnoty
-5
0
5
10
15
20
25
30
35
0
100
200
300
400
500
600
700
800
900
1000

PŘÍLOHA C


Porovnání hodnot testovacích výstupů z RBF a požadovaných hodnoty
výstupy z REF
požadované hodnoty
výstupy z RBF a požadované hodnoty

PŘÍLOHA D



PŘÍLOHA E

```
%BLACK SCHOLES MODEL
%realizacni cena;vyber z workspace output
X = input(3,1:1530);
%cas musi byt prepocitan na dny;
t = input(2,1:1530)/ 365;
%hodnota volatility brana z input;
v = input(1,1:1530);
%black schole-suv vzorec pro vypocet ceny PUT/CALL opci
[Call, Put]= blsprice(100,X,0,t,v);

% NAMERENE ODCHYLKY
%výpočet celkových čtverců odchylek pro BLS
TEbls=sum((mean(output)-output).*(mean(output)-output));
% 2_střední kvadratickA odchylka E pro FF
Ebls = sum((Call - output).*(Call - output))/1530;
% 3_střední odchylka odmocnini E = RMSE pro FF
MRSEbls=sqrt(sum((Call - output).*(Call - output))/1530);
% 4_koeficient determinace pro FF
r2_bls=1-(sum((Call - output).*(Call - output))/TEbls);
% 5_pumerna absolutni odchylka pro FF
AAEbls=sum(abs(Call - output))/1020;
% 6_maximální absolutní odchylka MAE dopsat
MAEbls=max(abs(Call - output));
```

PŘÍLOHA F

```
function varargout = opce(varargin)
% OPCE M-file for opce.fig
%      OPCE, by itself, creates a new OPCE or raises the existing
%      singleton*.
%
%      H = OPCE returns the handle to a new OPCE or the handle to
%      the existing singleton*.
%
%      OPCE('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in OPCE.M with the given input arguments.
%
%      OPCE('Property','Value',...) creates a new OPCE or raises the
%      existing singleton*.  Starting from the left, property value pairs
are
%      applied to the GUI before opce_OpeningFunction gets called.  An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to opce_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Copyright 2002-2003 The MathWorks, Inc.

% Edit the above text to modify the response to help opce

% Last Modified by GUIDE v2.5 26-Apr-2011 23:00:27

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @opce_OpeningFcn, ...
                   'gui_OutputFcn',  @opce_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before opce is made visible.
function opce_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to opce (see VARARGIN)

set(handles.Metoda, 'SelectionChangeFcn', @Metoda_SelectionChangeFcn);
%set(handles.jaka, 'String', 1);

% Choose default command line output for opce
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes opce wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = opce_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in rozdel.
function rozdel_Callback(hObject, eventdata, handles)
% hObject    handle to rozdel (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

i = evalin('base', 'input');    %čtení z workspace inputy
o = evalin('base', 'output');   %čtení z workspace outputy

perm = randperm(1530)';                     %   nahodna permutace
vstup = i(:,perm);
vystup = o(:,perm);
input_tren = vstup(:,1:1020);               %   vstupni trenovaci
input_test  = vstup(:,1021:end);            %   vstupni testovaci
output_tren = vystup(:,1:1020);             %   vystupni trenovaci
output_test  = vystup(:,1021:end);          %   vystupni testovaci

eg = str2num(get(handles.edit1, 'String')); % ošetření nulová chyba
if (isempty(eg))
    eg = '0'
end

sc = str2num(get(handles.edit2, 'String')); %ošetření sc
if (isempty(sc))
    sc = '0'
end

switch get(handles.jaka, 'String')
    case '2'
```

```
        %TRENOVANI  / TESTOVANI DOPREDNE NEURONOVE SITE

        %tvorba neuronové sítě FF
        net2 = newff(minmax(input_tren), [2 1], {'tansig','purelin'},
'trainlm');         %dopředná síť, tansig je hyperbolicky tangens, purelin
je linearni fce
        %nastavení epoch
        net2.trainParam.epochs = 100;
        %trénování
        [net2,tr] = train(net2,input_tren,output_tren);
        %výsledek trénování
        ytren_dopredna = sim(net2,input_tren);
        %výsledek testování
        ytest_dopredna = sim(net2,input_test);

        % VYPOCET  ODCHYLEK
            % 1_vypocet celkovych souctu ctvercu odchylek TE pro FF
        mtren_dopredna=sum((mean(output_tren) -
output_tren).*(mean(output_tren) - output_tren));
        mtest_dopredna=sum((mean(output_test) -
output_test).*(mean(output_test) - output_test));

            % 2_střední kvadratickA odchylka E pro FF
        mse_tren_dopredna=sum((output_tren - ytren_dopredna).*(output_tren
- ytren_dopredna))/1020;
        mse_test_dopredna=sum((output_test - ytest_dopredna).*(output_test
- ytest_dopredna))/510;

            % 3_střední odchylka odmocnini E = RMSE pro FF
        mrse_tren_dopredna=sqrt(sum((output_tren -
ytren_dopredna).*(output_tren - ytren_dopredna))/1020);
        mrse_test_dopredna=sqrt(sum((output_test -
ytest_dopredna).*(output_test - ytest_dopredna))/510);

            % 4_koeficient determinace pro FF
        r2_tren_dopredna=1-(sum((output_tren -
ytren_dopredna).*(output_tren - ytren_dopredna))/mtren_dopredna);
        r2_test_dopredna=1-(sum((output_test -
ytest_dopredna).*(output_test - ytest_dopredna))/mtest_dopredna);

            % 5_pumerna absolutni odchylka pro FF
        aae_tren_dopredna=sum(abs(output_tren - ytren_dopredna))/1020;
        aae_test_dopredna=sum(abs(output_test - ytest_dopredna))/510;

            % 6_maximální absolutní odchylka MAE dopsat
        mae_tren_dopredna=max(abs(output_tren - ytren_dopredna));
        mae_test_dopredna=max(abs(output_test - ytest_dopredna));

            % vykreslení grafu tren
        x = 1:1020;

        axes(handles.Vykres)

        plot(x,output_tren,x,ytren_dopredna);
        set(handles.Vykres, 'XLim', [0 1020]);

        % vykreslení grafu test
        v = 1:510; %jako xová osa
```

```
        axes(handles.Vykres2)

        plot(v,ytest_dopredna,v,output_test);
        set(handles.Vykres2, 'XLim', [0 510]);

        % výpis sledovaných parametrů do labelů tren
        set(handles.E, 'String', mse_tren_dopredna);
        set(handles.TE, 'String', mtren_dopredna);
        set(handles.RMSE, 'String', mrse_tren_dopredna);

        % výpis sledovaných parametrů do labelů test
        set(handles.E2, 'String', mse_test_dopredna);
        set(handles.TE2, 'String', mtest_dopredna);
        set(handles.RMSE2, 'String', mrse_test_dopredna);

        guidata(hObject, handles);

    case '3'
        %Black Schole model
        v = i(1,1:1530);                            %realizacni cena;
        t = i(2,1:1530) / 365;                      %čas musí být
přepočítaný na dny;
        s = i(3,1:1530);                            %hodnota volatility;
        [Call, Put]= blsprice(100,s,0,t,v);   %black schole-sův vzorec pro
výpočet ceny PUT/CALL opcí
           Call=[Call];
        %DOPLNIT DEFINICI NA VÝSTUP
        % NAMERENE ODCHYLKY
            %výpočet celkových čtverců odchylek pro BLS
        TEbls=sum((mean(o) - o).*(mean(o) - o));
            % 2_stredni kvadratickA odchylka E pro FF
        Ebls = sum((Call - o).*(Call - o))/1530;
            % 3_stredni odchylka odmocnini E = RMSE pro FF
        RMSEbls=sqrt(sum((Call - o).*(Call - o))/1530);
            % 4_koeficient determinace pro FF
        r2_bls=1-(sum((Call - o).*(Call - o))/TEbls);
            % 5_pumerna absolutni odchylka pro FF
        AAEbls=sum(abs(Call - o))/1020;
            % 6_maximální absolutní odchylka MAE dopsat
        MAEbls=max(abs(Call - o));

        % vykreslení grafu bls

        xx = 1:1530;    %jako ixová osa

        axes(handles.Vykres3)

        plot(xx,Call,xx,o);
        set(handles.Vykres3, 'XLim', [0 1530]);

        % výpis sledovaných chyb - odchylek
        set(handles.E, 'String',Ebls);
        set(handles.TE, 'String',TEbls);
        set(handles.RMSE, 'String',RMSEbls);

        guidata(hObject, handles);
```

```
    otherwise

        %TESTOVANI/TRENOVANI SITE RBF
        %tvorba neuronove site RBF
        net = newrb(input_tren,output_tren,eg,sc);
        %   vysledek trenovani
        ytren = sim(net, input_tren);
        %   vysledek testovani
        ytest = sim(net, input_test);

            % 1_celkovy soucet ctvercu odchylek TE
        TE_tren=sum((mean(output_tren) - output_tren).*(mean(output_tren) -
output_tren));
        TE_test=sum((mean(output_test) - output_test).*(mean(output_test) -
output_test));

            % 2_stredni kvadraticka odchylka E
        E_tren=sum((output_tren - ytren).*(output_tren - ytren))/1020;
        E_test=sum((output_test - ytest).*(output_test - ytest))/510;

            % 3_stredni odchylka odmocnini E = RMSE
        RME_tren=sqrt(sum((output_tren - ytren).*(output_tren -
ytren))/1020);
        RME_test=sqrt(sum((output_test - ytest).*(output_test -
ytest))/510);

            % 4_koeficient determinace pro RBF
        r2_tren=1-(sum((output_tren - ytren).*(output_tren -
ytren))/TE_tren);
        r2_test=1-(sum((output_test - ytest).*(output_test -
ytest))/TE_test);

            % 5_prumerna absolutni odchylka pro RBF
        AAE_tren=sum(abs(output_tren - ytren))/1020;
        AAE_test=sum(abs(output_test - ytest))/510;

            % 6_maximální absolutní odchylka MAE dopsat do textu
        MAE_tren=max(abs(output_tren - ytren));
        MAE_test=max(abs(output_test - ytest));

            % 7_stredni absolutni procentická odchylka pro RBF
        MAPE_tren=(sum(abs(output_tren - ytren)./output_tren)/1020)*100;
        MAPE_test=(sum(abs(output_test - ytest)./output_test)/510)*100;

        % vykreslení grafu tren
        x = 1:1020; %jako xová osa

        axes(handles.Vykres)

        plot(x,output_tren,x,ytren);
        set(handles.Vykres, 'XLim', [0 1020]);

        % vykreslení grafu test
        v = 1:510; % jako xová osa
```

```
        axes(handles.Vykres2)

        plot(v,ytest,v,output_test);
        set(handles.Vykres2, 'XLim', [0 510]); %rozsah od 0 do 510

        % výpis sledovaných chyb - odchylek do labelů tren
        set(handles.E, 'String', E_tren);
        set(handles.TE, 'String', TE_tren);
        set(handles.RMSE, 'String', RME_tren);

        % výpis sledovaných chyb - odchylek do labelů test
        set(handles.E2, 'String', E_test);
        set(handles.TE2, 'String', TE_test);
        set(handles.RMSE2, 'String', RME_test);

        guidata(hObject, handles);

end

function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1 as a
double


% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit2 as text
%        str2double(get(hObject,'String')) returns contents of edit2 as a
double


% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function rozdel_CreateFcn(hObject, eventdata, handles)
% hObject    handle to rozdel (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called


% --- If Enable == 'on', executes on mouse press in 5 pixel border.
% --- Otherwise, executes on mouse press in 5 pixel border or over rozdel.
function rozdel_ButtonDownFcn(hObject, eventdata, handles)
% hObject    handle to rozdel (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes during object deletion, before destroying properties.
function rozdel_DeleteFcn(hObject, eventdata, handles)
% hObject    handle to rozdel (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes during object creation, after setting all properties.
function FlashMessage_CreateFcn(hObject, eventdata, handles)
% hObject    handle to FlashMessage (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called


% --- Executes during object deletion, before destroying properties.
function FlashMessage_DeleteFcn(hObject, eventdata, handles)
% hObject    handle to FlashMessage (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% --------------------------------------------------------------------
function Metoda_SelectionChangeFcn(hObject, eventdata, handles)
% hObject    handle to Metoda (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

handles = guidata(hObject);

switch get(eventdata.NewValue,'Tag')
    case 'radiobutton1'
        set(handles.jaka, 'String', 1);            %hierarchie
        set(handles.Vykres, 'Visible', 'on');
        set(handles.Vykres2, 'Visible', 'on');
```

```
        set(handles.Vykres3, 'Visible', 'off');
        set(handles.text14, 'Visible', 'on');
        set(handles.text15, 'Visible', 'on');
        set(handles.text16, 'Visible', 'on');
        set(handles.text17, 'Visible', 'on');
        set(handles.text18, 'Visible', 'on');
        set(handles.text25, 'Visible', 'off');
        set(handles.edit1, 'Enable', 'on');
        set(handles.edit2, 'Enable', 'on');
    case 'radiobutton2'
        set(handles.jaka, 'String', 2);
        set(handles.Vykres, 'Visible', 'on');
        set(handles.Vykres2, 'Visible', 'on');
        set(handles.Vykres3, 'Visible', 'off');
        set(handles.text14, 'Visible', 'on');
        set(handles.text15, 'Visible', 'on');
        set(handles.text16, 'Visible', 'on');
        set(handles.text17, 'Visible', 'on');
        set(handles.text18, 'Visible', 'on');
        set(handles.text25, 'Visible', 'off');
        set(handles.edit1, 'Enable', 'on');
        set(handles.edit2, 'Enable', 'off');
    case 'radiobutton3'
        set(handles.jaka, 'String', 3);
        set(handles.Vykres, 'Visible', 'off');
        set(handles.Vykres2, 'Visible', 'off');
        set(handles.Vykres3, 'Visible', 'on');
        set(handles.text14, 'Visible', 'off');
        set(handles.text15, 'Visible', 'off');
        set(handles.text16, 'Visible', 'off');
        set(handles.text17, 'Visible', 'off');
        set(handles.text18, 'Visible', 'off');
        set(handles.text25, 'Visible', 'on');
        set(handles.edit1, 'Enable', 'off');
        set(handles.edit2, 'Enable', 'off');


end
guidata(hObject, handles);


% --- Executes during object creation, after setting all properties.
function Vykres3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Vykres3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: place code in OpeningFcn to populate Vykres3
```