UNIVERZITA PARDUBICE
Fakulta elektrotechniky a informatiky

Hledání parametrů modelů dynamických systémů

Miroslav Moravec

Bakalářská práce
2011

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

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Miroslav MORAVEC**
Osobní číslo:	**I07723**
Studijní program:	**B2646 Informační technologie**
Studijní obor:	**Informační technologie**
Název tématu:	**Hledání parametrů modelů dynamických systémů**
Zadávající katedra:	**Katedra informačních technologií**

Zásady pro vypracování:

Cíl práce: Cílem je porovnat tři přístupy k hledání parametrů ARMA modelů dynamických systémů a to metodu nejmenších čtverců, jednoho zástupce klasických optimalizačních technik a jednoho zástupce evolučních optimalizačních technik.

Teoretická část: Popis a využití ARMA modelů, popis zmíněných optimalizačních technik.
Implementační část: Student v jazyku JAVA naprogramuje aplikaci, která alternativně třemi optimalizačními technikami stanoví parametry ARMA modelu na základě zadaných experimentálních dat. Očekává se přívětivé ovládání a podrobná vizualizace výsledků.

Rozsah grafických prací:

Rozsah pracovní zprávy:

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

Seznam odborné literatury:

1. DRÁBEK, O., MACHÁČEK, J. Experimentální identifikace. Pardubice : VŠChT Pardubice, 1987. 275 s.

2. ZELINKA, I. Umělá inteligence v problémech globální optimalizace. Praha : BEN - technická literatura, 2002. 193 s. ISBN 80-7300-069-5

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

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

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

Univerzita Pardubice Česká republika

L.S.

prof. Ing. Simeon Karamazov, Dr.
děkan

Ing. Lukáš Čegan, Ph.D.
vedoucí katedry

V Pardubicích dne 31. března 2011

Prohlášení autora

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

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

Souhlasím s prezenčním zpřístupněním své práce v Univerzitní knihovně.

V Pardubicích dne 9. 5. 2011 Miroslav Moravec

Anotace

Klasifikační práce se zabývá především problematikou odhadů parametrů modelů ARMA. Ve druhé kapitole jsou vysvětleny základní pojmy, nezbytné pro pochopení následného popisu ARMA modelů a dalších představitelů Boxovy-Jenkinsovy metodologie. Prostřední část práce je věnována výkladu o třech optimalizačních technikách, kterými jsou metody Nelder-Mead a nejmenších čtverců a hledání řešení s pomocí genetického algoritmu. V závěru práce je popsána aplikace určená pro odhad parametrů ARMA modelů prostřednictvím těchto metod a následně jsou na základě odhadů této aplikace dané techniky porovnány.

Klíčová slova

ARMA model, genetický algoritmus, metoda Nelder-Mead, metoda nejmenších čtverců

Title

Parameters tuning of dynamic systems models.

Annotation

The classification work is mainly concerned about the issue of parameter estimation of ARMA models. The second chapter explains the basic concepts necessary to understand the following describe the ARMA models, and other representatives of the Box-Jenkins methodology. The middle part is devoted to discussion of the three optimization techniques, which are methods of Nelder-Mead and the method of the least squares and finding solutions by using genetic algorithm. The conclusion describes an application designed to estimate the parameters of ARMA models via these methods and subsequently, based on estimates of the application these techniques are compared.

Keywords

ARMA model, genetic algorithm, Nelder-Mead method, the method of least squares

Obsah

Seznam proměnných

u	bílý šum / vstupní signál systému
ψ	parametr funkce
a	parametr funkce
b	parametr funkce
r	řád modelu
p	řád modelu
q	řád modelu
$\hat{p}$	vektoru odhadů parametrů
$\lvert A_{j,i} \rvert$	subdeterminant získaný z matice A, ve které byli vynechány řádky i a j
$\lvert A \rvert$	determinant matice A
P_i	pravděpodobnost výběru jedince
α	koeficient reflexe
γ	koeficient expanze
ρ	koeficient kontrakce
σ	koeficient redukce
e_j	jednotkový vektor
h_j	velikost kroku ve směru jednotkového vektoru e_j
x_0	těžiště simplexu
x_r	bod odrazu simplexu
x_e	bod expanze simplexu
x_c	bod kontrakce simplexu

Seznam zkratek

AR	Auto Regressive
MA	Moving Average
ARMA	Auto Regressive Moving Average
ARIMA	Auto Regressive Integrated Moving Average
SARIMA	Seasonal Auto Regressive Integrated Moving Average
AIC	Akaike Information Criterion
BIC	Bayesian Extension of Information Criterion
SBC	Schwartz Bayes Criterion
FPE	Final Prediction Error
ME	Mean error
MSE	Mean Squared Error
RMSE	Root Mean Squared Error
MAE	Mean Absolute Error
MPE	Mean Percent Error
MAPE	Mean Absolute Percent Error
GUI	Graphical User Interface
MNČ	Metoda nejmenších čtverců
GA	Genetický algoritmus
NMA	Nelder-Mead algoritmus
ACF	Výběrová autokorelační funkce
PACF	Parciální autokorelační funkce

Seznam obrázků

Seznam tabulek

1 Úvod

Každý systém má svůj stav, který se může v čase měnit. Lidstvo během své existence vytvořilo nespočet technologií na měření, uchování i analýzu těchto dat, přičemž každá z těchto činností je neméně důležitá než činnost druhá. Pouze v případě, kdy jsou všechny tyto tři části zvládnuty a je proveden potřebný rozbor, je možné se z dat, představujících stavy, poučit, porozumět chování zkoumaného systému a případně jej nakonfigurovat tak, aby toto jeho chování bylo více žádoucí, nebo výsledky analýzy využít při návrhu a tvorbě systému nového.

Tato bakalářská práce se zabývá právě analýzou dat, a to konkrétně analýzou prostřednictvím modelu ARMA. Přičemž je zaměřena především na část jeho výstavby, kterou lze nejlépe označit jako hledání parametrů ARMA modelu. Pro nalezení parametrů lze použít řadu odlišných postupů. Zde budou popsány a, především dle kritérií přesnosti a doby výpočtu, porovnány tři typy technik, kterými jsou evoluční optimalizace, klasická optimalizace a metoda nejmenších čtverců. Za představitele evolučních optimalizačních technik byl zvolen genetický algoritmus. Obor klasických technik zde zastupuje metoda Nelder-Mead.

V rámci bakalářské práce byla vytvořena grafická okenní aplikace v jazyku JAVA, která pomocí těchto tří metod odhadne parametry a poskytne dostatečné informace o jejich přesnosti. Tuto aplikaci lze použít pro odhad parametrů pro libovolný model ARMA, ovšem samotné porovnání bylo provedeno na základě tvorby modelů laboratorního simulátoru vrtulníku a dvou simulovaných dynamických systémů.

2 Důležité pojmy

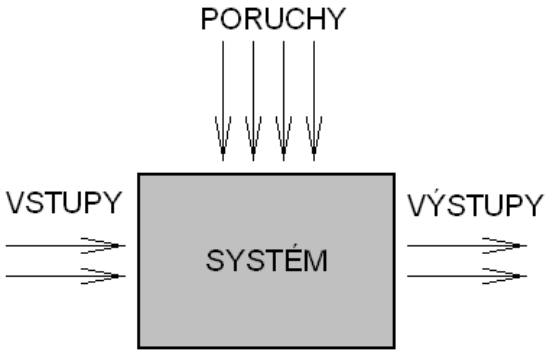
Popis pojmů v této kapitole vychází ze zdrojů [1], [2], [3], [4] a [5].

2.1 Systém

2.1.1 Definice

Definicí systému je mnoho, stejně jako jeho typů. Obecně lze systém označit jako souhrn souvisejících prvků, které dohromady tvoří smysluplný celek. Skládá se tedy z entit, které se podílejí na jeho chodu.

Na systém působí vstupní veličiny, které ovlivňují jeho chování a následné výstupní veličiny. Hodnoty vstupních a výstupních veličin lze označit jako signály.



Obrázek 1 - Systém

2.1.2 Rozdělení

Z hlediska této klasifikační práce je podstatné především trojí dělení, a to na systémy:

- deterministické a stochastické,
- statické a dynamické,
- spojité a diskrétní, případně také kombinované.

Dále lze systémy dělit také například dle vztahu k okolí na:

- uzavřené a otevřené,

nebo dle jejich složitosti:

- tvrdé a měkké.

2.2 Proces

2.2.1 Definice

Proces obecně označuje posloupnost určitých dějů či změn, které mění stav systému. I samotný sled stavů systému reprezentovaný určitými vstupními či výstupními hodnotami lze označit jako proces. Významy termínů systém a proces jsou v některých

případech natolik podobné, že je lze zaměnit. Například technologický systém lze označit jako technologický proces.

2.2.2 Rozdělení

Procesy lze podobně jako systémy dělit na deterministické, kde každý další stav zaručeně vyplývá ze stavu předchozího a jejich podobu lze tedy relativně přesně předvídat (např. chemické procesy) a procesy stochastické, jejichž stavy naopak lze předpovědět jen s určitou pravděpodobností, jelikož každý z nich je ovlivněn náhodnou složkou (např. meteorologické procesy).

2.3 Matematický model

2.3.1 Popis

Systémy a procesy jsou často předmětem matematického modelování, kdy se za použití modelu zkoumá jejich chování. S pomocí získaných poznatků lze provést optimalizaci, či dokonce předpovídat jejich chování v budoucnosti. Vytvářejí se například modely technologických zařízení, ekonomické, či modely představující organismus.

2.3.2 Rozdělení

Obor matematického modelování začleňuje dva základní přístupy při tvorbě modelu, kterými jsou:

- matematicko-fyzikální analýza,
- experimentální identifikace.

Matematicko-fyzikální analýza vychází za základních fyzikálních zákonů, systém je při ní rozložen na jednoduché části, u nichž lze závislost mezi vstupy a výstupy poměrně snadno popsat pomocí fyzikálních vzorců. Takový model je velmi realistický, ovšem jeho výstavba je náročná a vyžaduje podrobnou znalost řešeného problému.

Při experimentální identifikaci, ke které se řadí i řešení ARMA modelu, se vychází z naměřených vstupních a výstupních signálu, na jejichž základě je nejprve určen typ modelu a následně i jeho parametry.

Dále lze matematické modely dělit dle ustálenosti signálů v modelovaném systému:

- statické modely,
- dynamické modely.

Statický model popisuje systém, jehož vstupní veličiny (signály) jsou ustálené. Tento typ modelů bývá zpravidla jednodušší, než modely dynamické, které naopak popisují závislost mezi dynamickými, tedy v čase se měnícími, vstupy a výstupy.

2.4 Časová řada

2.4.1 Definice

Časová řada označuje posloupnost věcně a prostorově srovnatelných pozorování, vyjádřených určitými daty. Může tedy například reprezentovat proces, který představuje vstupy nebo výstupy systému v čase.

2.4.2 Analýza časových řad

Pokud je zapotřebí hlouběji pochopit chování systému, lze časové řady jeho vstupů či výstupů využít k tomuto typu rozboru. Mezi jednoduché způsoby analýzy patří například určení průměru hodnot, tempa růstu či vizuální analýza za pomoci grafu. Pro složitější rozbor časové řady je nutné vytvořit její model, například s využitím Boxovy-Jenkinsovy metodologie.

3 Boxova-Jenkinsova metodologie

Následující dvě kapitoly čerpají ze zdrojů [5], [6], [7], [8] a [9].

Tuto metodologii lze označit jako moderní soubor technik pro modelování a prognózování časových řad obsahujících náhodnou složku, tedy řad představujících určitý stochastický proces.

Náhodná složka procesu může být tvořena korelovanými náhodnými veličinami, přičemž Boxova-Jenkinsova metodologie začleňuje modely pro popis stacionárních i nestacionárních časových řad, tedy řad, kde se rozložení pravděpodobnosti této náhodné složky v čase nemění, respektive mění.

3.1 Stacionární modely časových řad

Všechny tyto modely vycházejí z tzv. lineárního procesu, který představuje vyjádření stacionárního procesu jako lineární kombinace řady stejně rozdělených náhodných veličin. Lze jej vyjádřit rovnicí ve tvaru:

$$y_t = u_t + \psi_1 u_{t-1} \cdots \psi_k u_{t-k} \tag{1}$$

kde u_t - je bílý šum,

ψ - jsou parametry funkce.

3.1.1 Model AR

Jedná se o model autoregresního procesu, což je v podstatě lineární proces s konečným počtem nenulových parametrů. Jeho označení je AR (p), kde „p“ představuje řád procesu. Každá hodnota y v čase t je dána součtem předchozích „p“ hodnot násobených parametry modelu.

Autoregresní proces prvního řádu AR (1) lze vyjádřit vztahem:

$$y_t = a_1 y_{t-1} + u_t \tag{2}$$

kde musí platit podmínka stacionarity $|a_1| < 1$,

u_t - značí bílý šum,

a - je parametr funkce.

Přestože je vzhledem ke složitosti a času potřebném k výpočtům vhodné, aby byl řád co nejnižší, používají se i modely procesů s vyššími řády, například proces druhého řádu AR (2) vyjádřený vztahem:

$$y_t = a_1 y_{t-1} + a_2 y_{t-2} + u_t \tag{3}$$

s podmínkami stacionarity procesu $a_1 + a_2 < 1$, $a_2 - a_1 < 1$ a $-1 < a_2 < 1$,

nebo proces obecného řádu „p“ AR (p):

$$y_t = a_1 y_{t-1} + \cdots + a_p y_{t-p} + u_t \tag{4}$$

kde u_t - je bílý šum,

a - jsou parametry funkce a

p - je řád modelu.

3.1.2 Model MA

Neboli model procesu klouzavých průměrů. Značí se jako MA (q), kde „q“ představuje řád procesu. Každá hodnota y v čase t je dána součtem předchozích „q“ hodnot bílého šumu násobených parametry modelu.

Proces prvního řádu MA (1) tedy vyjadřuje vztah:

$$y_t = u_t - b_1 u_{t-1} \tag{5}$$

omezený podmínkou stacionarity $|b_1| < 1$

a proces řádu obecného MA (q) lze zapsat takto:

$$y_t = u_t - b_1 u_{t-1} - \cdots - b_q u_{t-q} \tag{6}$$

kde u_t - je bílý šum,

b - značí parametry funkce a

q - je řád modelu.

3.1.3 Model ARMA

ARMA kombinuje dva výše uvedené modely AR a MA. ARMA (p, q) tedy představuje smíšený proces řádu „p“ a „q“.

Tento typ modelu je jedním z hlavních předmětů této práce, bude tedy podrobněji popsán v samostatné kapitole.

3.2 Nestacionární modely časových řad

3.2.1 Model ARIMA

Autoregresní integrovaný proces klouzavých průměrů řádu p, d, q, označený jako ARIMA (p, d, q), představuje nestacionární smíšený proces, který byl pomocí diference řádu „d“ transformován do takové podoby, že jej lze popsat pomocí stacionárního modelu ARMA (p, q).

Dále existují modely pro sezonní stacionární procesy SAR, SMA a SARMA či pro sezónní nestacionární procesy SARIMA. Vzhledem k tématu této práce ovšem není nutný jejich podrobnější popis.

3.3 Model systému

Důležité je říci, že v případě modelování systému pomocí modelu MA nebo jeho kombinací s jiným typem není vstupem označeným jako u samotný bílý šum, ale hodnoty určitého vstupního signálu z něho vycházející. Například napětí elektromotoru, otáčky motoru či průtok kapaliny potrubím.

3.4 Výstavba modelu

3.4.1 Identifikace a výběr modelu

Jedná se o nejdůležitější a také nejnáročnější etapu, v níž je třeba určit, jaký typ modelu bude vůbec použit a případně také, jaký bude jeho řád. Vychází se z charakteristik naměřených dat, která jsou případně transformována do vhodnější podoby. V případě volby mezi modely AR, MA a ARMA lze vycházet z grafů výběrové autokorelační a parcialní autokorelační funkce, jak je uvedeno v tabulce níže (Tabulka 1), nebo typ zvolit dle k tomu určených kritérií, kterými jsou:

- AIC (Akaike Information Criterion),
- BIC (Bayesian Extension of Information Criterion),
- SBC (Schwartz Bayes Criterion),
- FPE (Final Prediction Error).

Tyto techniky pracují na principu výběru modelu s nejmenším počtem parametrů a reziduálním rozptylem, kde rezidui jsou hodnoty chyb.

Tabulka 1 - Určení typu modelu dle grafů ACF a PACF

Typ modelu	ACF	PACF
AR (p)	neexistuje bod useknutí *u*, graf exponenciálně klesá nebo představuje tlumenou sinusoidu	bod useknutí u = p
MA (q)	bod useknutí u = p	neexistuje bod useknutí *u*, graf exponenciálně klesá nebo představuje tlumenou sinusoidu
ARMA (p, q)	neexistuje bod useknutí *u*, graf exponenciálně klesá nebo představuje tlumenou sinusoidu od času *q*-*p*+1	neexistuje bod useknutí *u*, graf exponenciálně klesá nebo představuje tlumenou sinusoidu od času *p*-*q*+1

Volba řádu modelu

Volba řádu modelu se provádí obvykle až po identifikaci systému, tedy až poté, co jsme vypočetli parametry modelu, přičemž se dle testovacích kritérií porovnávají predikce vypočtené modely různých řádů. Postupuje se od menšího řádu k většímu a zkoumá se, zdali došlo ke zlepšení, či nikoliv. Hodnotící kritéria vycházejí z chyby predikce, tedy rozdílu mezi skutečnou naměřenou hodnotou, z níž model vychází a hodnotou určenou dle vypočtených parametrů. Nejčastěji používaná kritéria jsou uvedena dále v pododdílu 3.4.3.

Existují i metody pro odhad řádu před identifikací systému, ovšem s dnešní výpočetní technikou a jejími softwarovými nástroji obvykle není problémem vytvořit sérii pokusných modelů různých řádů a zjistit, při kterém řádu chování modelu nejvíce odpovídá skutečnému procesu.

3.4.2 Odhad parametrů

Jedná se o poměrně složitou proceduru, která často bývá předmětem optimalizace. Dnes je tato činnost většinou prováděna s pomocí specializovaného statistického softwaru. Princip odhadu spočívá v hledání té varianty, pro kterou je minimální suma kvadrátů rozdílů skutečných naměřených hodnot a predikcí, což jsou hodnoty pomocí parametrů vypočtené. Toto kritérium se nazývá chybová funkce.

Postupů k nalezení optimálních parametrů je více, tato bakalářská práce se zaměřuje na tři algoritmy, kterými jsou:

- metoda nejmenších čtverců,
- Nelder-Mead,
- genetický algoritmus,

které jsou podrobněji popsány v samostatných oddílech. Mezi další způsoby patří například:

- metoda maximální věrohodnosti;
- metoda stochastické aproximace.

3.4.3 Ověřování modelu

Během ověřování neboli verifikace modelu se provádí porovnání modelu s procesem, který představuje. Metod k této činnosti je celá řada. Nejjednodušším způsobem je vypočítat predikce a porovnat je s naměřenými hodnotami procesu, nejlépe pomocí jejich grafů.

Pro informace o přesnosti modelu lze dále využít statistické metody, které provádějí složitější analýzu reziduí, jimiž jsou hodnoty chyb, například:

- test normality nesystematické složky,
- metoda odhadnutých reziduí,
- Durbinův-Watsonův test.

Tyto testy hodnotí model dle různých kritérií a hypotéz. Moderní softwarové nástroje na řešení této problematiky samozřejmě také disponují funkcemi pro zhodnocení kvality modelu. K tomuto posouzení využívají kritéria uvedená v následující tabulce (Tabulka 2).

Tabulka 2 - Kritéria používaná statistickými programy

Kritérium	Vzorec
Střední chyba odhadu ME	$ME = \frac{1}{n}\sum_{t=1}^{n}(y_t - y'_t)$
Střední kvadratická chyba MSE	$MSE = \frac{1}{n}\sum_{t=1}^{n}(y_t - y'_t)^2$
RMSE	$\sqrt{MSE}$
Střední absolutní chyba MAE	$\frac{1}{n}\sum_{t=1}^{n}\lvert y_t - y'_t\rvert$
Střední procentuální chyba MPE	$MPE = \frac{100}{n}\sum_{t=1}^{n}\left(\frac{y_t - y'_t}{y_t}\right)$
Střední absolutní procentuální chyba MAPE	$MAPE = \frac{100}{n}\sum_{t=1}^{n}\left\lvert\frac{y_t - y'_t}{y_t}\right\rvert$

V případě, že je po provedení těchto testů přesnost modelu vyhodnocena jako nedostatečná, je třeba změnit řád modelu, metodu pro odhad jeho parametrů, pokud je shledána jako nepřesná, nebo dokonce použít jiný typ modelu.

4 Model ARMA

4.1 Popis

Jak již bylo uvedeno v pododdíle 3.1.3, model ARMA je kombinací modelu autoregresního procesu AR a procesu klouzavých průměrů MA. Představuje tedy smíšený proces AR (p) a MA (q). Pro obecný řád p a q lze proces ARMA (p, q) vyjádřit předpisem:

$$y_t = u_t + a_1 y_{t-1} - b_1 u_{t-1} + \cdots + a_p y_{t-p} - b_q u_{t-q} \tag{7}$$

kde u_t - je bílý šum,

a a b - jsou parametry funkce,

p a q - jsou řády procesů.

Používá se v těch případech, kdy modelovaný proces není vyhovující pro použití AR nebo MA modelů.

4.2 Praktické využití

Model ARMA spadá do matematického modelování, kdy se výstavba provádí metodou experimentální identifikace. Obecně lze říci, že slouží k analýze časových řad. Podmínkou pro jeho použití je, stejně jako u ostatních modelů Boxovy-Jenkinsovy metodologie, dostatečně dlouhá časová řada výchozích dat, která by dle zdroje [5] měla zahrnovat alespoň 50 hodnot. Pokud tedy je zapotřebí vytvořit model stochastického systému a je k dispozici dostatečný objem dat, lze aplikovat právě model ARMA. Své uplatnění nalézá v modelování například ekonomických, meteorologických či technologických systému.

5 Metody pro odhad parametrů ARMA modelu

5.1 Metoda nejmenších čtverců

Obsah tohoto oddílu vychází ze zdrojů [3], [4], [10] a [11].

Metoda nejmenších čtverců (dále jen MNČ) je matematicko-statistická metoda, určená k řešení nekompatibilních soustav lineárních rovnic. Využívá se zpravidla při zpracování nepřesných dat (například naměřených dat, které byly zaznamenány s určitou chybou), kdy je požadováno vytvořit co nejpřesnější matematický model procesu, který tyto data reprezentují, nebo odstranit chybu. Nejjednodušším způsobem je proložit naměřené jednorozměrné hodnoty přímkou. To znamená, že pokud je hledaná přímka vyjádřena rovnicí $y = ax + b$, pomocí MNČ jsou nalezeny takové koeficienty a a b, pro které přímka nejlépe vystihuje rozložení daných bodů. Při požadavku na větší přesnost modelu lze použít složitější typ proložení dat, kdy jsou naměřená data nahrazena parabolou, nebo vyjádřena polynomem předem daného stupně či obecnou lineární kombinací daných bázových funkcí. Proložení dat bývá také označováno jako jejich aproximace. MNČ lze také využít například k nalezení nejpravděpodobnějšího průsečíku daného počtu přímek.

MNČ je zřejmě nejpoužívanější metodou pro odhad parametrů modelů a své uplatnění nalézá v široké škále oborů, neboť s potřebou přijatelnějšího vyjádření dat znehodnocených určitou chybou se lze setkat téměř v každém oboru vědy.

5.1.1 Postup řešení

Při použití MNČ k odhadu parametrů modelu ARMA se vychází z kritéria minimálního součtu kvadrátů chyb $e(t)$, tedy ze splnění podmínky:

$$J(\hat{p}) = \sum_{t=r+1}^{n} e(t)^2 \rightarrow MIN \tag{8}$$

kde $\hat{p}$ - představuje odhad vektoru parametrů,

r - je řád modelu,

n - je počet měření.

Pro řešení odhadu parametrů modelu obecného řádu r daného předpisem

$$y_t = e_t + a_1 y_{t-1} - b_1 u_{t-1} + a_2 y_{t-2} - b_2 u_{t-2} + \cdots + a_r y_{t-r} - b_r u_{t-r} \tag{9}$$

lze vyjádřit chyby predikce e následovně

$$\begin{aligned} e_{r+1} &= y_{r+1} - \hat{y}_{1r+1} \\ e_{r+2} &= y_{r+2} - \hat{y}_{r+2} \\ &\vdots \end{aligned} \tag{10}$$

$$e_n = y_n - \hat{y}_n$$

Dále je třeba zavést vektory e, y, $\hat{p}$ a matici W ve tvaru:

$$e = \begin{bmatrix} e_{r+1} \\ e_{r+2} \\ \vdots \\ e_n \end{bmatrix};$$

$$y = \begin{bmatrix} y_{r+1} \\ y_{r+2} \\ \vdots \\ y_n \end{bmatrix};$$

$$W = \begin{bmatrix} y_r;\ u_r;\ y_{r-1};\ u_{r-1};\ \cdots;\ y_1;\ u_1 \\ y_{r+1};\ u_{r+1};\ y_r;\ u_r;\ \cdots;\ y_2;\ u_2 \\ \vdots \\ y_{n-1};\ u_{n-1};\ y_{n-2};\ u_{n-2};\ \cdots;\ y_{n-r};\ u_{n-r} \end{bmatrix};$$

$$\hat{p} = \begin{bmatrix} +\hat{a}_1 \\ -\hat{b}_1 \\ +\hat{a}_2 \\ -\hat{b}_2 \\ \vdots \\ +\hat{a}_r \\ -\hat{b}_r \end{bmatrix}.$$

Následně lze vyjádřit soustavu rovnic chyb v maticovém tvaru:

$$e = y - W\hat{p} \tag{11}$$

a hledaný vektor parametrů $\hat{p}$ je dán vztahem:

$$\hat{p} = (W^T W)^{-1} W^T y \tag{12}$$

5.2 Genetický algoritmus

Popis algoritmu vychází ze zdrojů [12], [13] a [14].

Genetické algoritmy se řadí ke skupině tzv. Evolučních algoritmů. Tyto postupy jsou inspirovány evolučními biologickými procesy, k nimž dochází v přírodě. Jedná se o relativně mladou metodu, jelikož první popis tohoto typu algoritmů se objevil v sedmdesátých letech minulého století.

Základním principem je snaha o aplikaci známé Darwinovy evoluční teorie vývoje druhů, jejíž stěžejní myšlenkou je, že přežijí pouze nejsilnější jedinci a tímto se zákonitě populace vyvíjí k lepší podobě.

Při aplikaci tohoto přístupu tedy každý jedinec reprezentuje určité řešení daného problému, přičemž kvalita tohoto řešení je ohodnocena tzv. fitness funkcí. Při simulaci přirozeného výběru jsou vybráni kvalitnější jedinci, jejichž křížením vzniknou potomci, kteří by měli být tvořeni, dle předpokladů, kvalitními geny svých předků. Opakováním inovováním populace by se měla průměrná kvalita jedinců daná fitness funkcí zvyšovat a řešení problému směřovat k žádanému optimu.

Jedná se tedy o algoritmus heuristický, který je vhodné uplatnit především v těch případech, kdy se nenabízí žádné zaručené exaktní řešení problému. Ač je svou povahou určen k hledání maximálních případů kritéria, hodnocení je plně určováno funkcí fitness, takže její vhodnou transformací lze hledat například minimum funkce, minimální počet určitých znaků řetězce, atd.

Genetický algoritmus (dále jen GA) je poměrně variabilní, jeho chování lze ovlivnit více variantami jeho provedení. Při tvorbě algoritmu je tedy třeba nejprve určit, jakým způsobem budou prováděny jeho úkony, kterými jsou:

- kódování řešení,
- způsob vytvoření počáteční populace,
- ohodnocovací („fitness“) funkce,
- operátor výběru,
- operátor křížení,
- operátor mutace.

5.2.1 Kódování řešení

Kódování řešení určuje, jak budou jedinci reprezentovat hledaná data, respektive jakým způsobem budou hledaná optimální data v jedinci zakódována.

Po vzoru biologických termínů lze jedince označit jako fenotyp, přičemž jeho genetickému kódu, tedy zakódovaným datům, se říká chromozom, případně genotyp či genom. Tento chromozom se skládá z genů, pro jejichž hodnoty se používá termín alely.

Jedním z jednoduchých způsobů zakódování dat do chromozomu jedince je například použití binárního řetězce. Další variantou je jedince vyjádřit jako chromozom složený z genů, jejichž alelami jsou reálná čísla. Tento způsob je vhodné použít, pokud jsou kladeny velké nároky na přesnost řešení.

V případě že pomocí GA hledáme například maximum funkce o dvou neznámých a použijeme kódování na reálná čísla, je vhodné tyto dvě neznámé vyjádřit jako chromozom jedince o dvou genech. Při této variantě také odpadají složité úkony při převádění skutečných hodnot na binární kódy.

Kódování je velmi důležitou složkou optimalizace pomocí GA a v případě, že nebude správně provedeno, mohou být výsledky velmi nepřesné, i když ostatní úkony byly zvládnuty dokonale.

5.2.2 Způsob vytvoření počáteční populace

V případě, že jsou známy přibližné hodnoty hledaných dat, zjištěné například z charakteristik zkoumaného procesu, nebo z výsledků dříve provedených optimalizačních testů, je samozřejmě vhodné z těchto dat vyjít a tím práci GA urychlit. Tento způsob je používán například v tzv. hybridních algoritmech, kde se k určení optima používají kombinace více algoritmů.

Naopak pokud neexistuje žádná představa o správném řešení problému, je běžným postupem volit chromozomy jedinců počáteční generace náhodným způsobem, pokud možno tak, aby tato generace byla co nejrůznorodější. Tedy aby obsahovala širokou škálu jedinců, jejichž chromozomy budou rovnoměrně pokrývat celý interval, v němž je očekáváno řešení.

5.2.3 Fitness funkce

Fitness funkce může mít různé podoby, dle povahy řešeného problému. Pokud například hledáme maximum funkce, lze jedince ohodnotit samotnou účelovou funkcí. V případě hledání minima je vhodným a jednoduchým řešením, aby hodnotící funkce vracela návratovou hodnotu účelové funkce násobenou mínus jednou.

Tato funkce společně s operátorem výběru zajišťuje, aby jedinci nových generací byli tvořeni ze silných jedinců generací minulých. Ovšem mělo by být také zajištěno, aby i jedinci slabší měli během výběru určitou šanci, protože kombinací genů silného jedince s geny slabého může vzniknout jedinec ještě silnější.

Pokud je použit ruletový výběr jedinců, kdy je výběr prováděn dle pravděpodobnosti určené jejich hodnocením, relativním vzhledem k ostatním jedincům, je nutné, aby hodnotící funkce vyjadřovala kvalitu reálným kladným číslem.

5.2.4 Operátor výběru

Jedná se o způsob, jakým GA simuluje přirozený výběr. Operátor výběru volí jedince pro následné křížení tak, aby tímto křížením byla vytvořena kvalitnější generace, než ta současná. Nejčastěji používanými operátory výběru jsou:

- turnajový výběr jedinců,
- ruleta,
- ruleta založená na pořadí jedinců.

Turnajový výběr jedinců

Princip turnajového výběru je odhadnutelný již z jeho názvu. Při této selekci je uspořádán „turnaj“, kterého se účastní N z celé generace náhodně zvolených jedinců. Vítěz, tedy nejlépe hodnocený, se stává prvním rodičem. Druhý rodič je zvolen stejným postupem.

Ruleta

Ruleta nevolí jedince přímo na základě jejich hodnocení, ale dle pravděpodobnosti výběru z něj odvozené. U všech jedinců pravděpodobnost, která je dána vztahem:

$$P_i = \frac{f(i)}{\sum_{j=1}^{N} f(j)} \tag{13}$$

kde f - je hodnotící funkce,

N - je počet jedinců,

představuje určitý úsek intervalu nula až jedna. Následně je vygenerováno náhodné desetinné číslo v tomto rozmezí a rodičem se stává ten jedinec, do jehož intervalu číslo spadá.

Při tomto řešení je nutné, aby hodnocení jedinců dané fitness funkcí bylo vždy kladné číslo. Další problém může nastat, pokud se vyskytne malé množství jedinců, kteří mají v porovnání se zbytkem generace vysoké hodnocení, tudíž i notně převyšují jejich pravděpodobnost výběru. V důsledku toho jsou ke křížení neustále vybíráni tito jedinci a vzhledem k tomu, že jejich hodnocení je relativně vysoké, ale od hledaného optima může být velmi odlišné, nemusí dojít k žádnému pokroku.

Ruleta založená na pořadí jedinců

Je způsob ruletového výběru, při kterém není pravděpodobnost výběru určována přímo z hodnocení jedinců, ale z jejich pořadí v generaci, které je jím určeno. Odpadá tedy podmínka, že hodnocení musí být kladné, naopak je nutné jedince před započetím výběru seřadit. Dále také nedochází k problému s přehodnocením jedinců v důsledku relativní pravděpodobnosti, jelikož i když je například jeden jedinec ohodnocen o mnoho větším číslem než ostatní, pravděpodobnost se určuje dle pořadí, které se vždy liší maximálně počtem jedinců N menším o jedna. Bohužel tato vlastnost může být i nevýhodou, a to v opačném případě, kdy má celá generace jen minimálně odlišné hodnocení a v důsledku toho je pravděpodobnost výběru například nejhoršího jedince velmi malá, i když je jeho hodnocení jen nepatrně menší než jedince nejlepšího.

Kromě těchto metod lze použít i další metody výběru, jako například tzv. „síň slávy“, což je obdoba turnajového algoritmu, kde je jedinec porovnáván s nejlepšími jedinci z generací minulých.

5.2.5 Operátor křížení

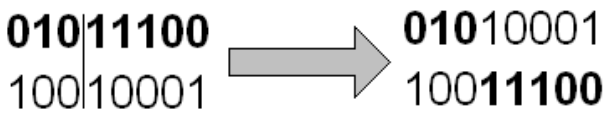
Ve fázi, kdy je dle operátoru výběru vytvořen pár jedinců, je třeba z tohoto rodičovského páru vytvořit jednoho či více potomků, a to takovým způsobem, aby vzniklý jedinec nesl část chromozomu obou předků. Tento proces je oproti svému skutečnému předobrazu značně zjednodušen.

Nejpoužívanějším typem křížení pro jedince reprezentované binárními chromozomy jsou metody označované jako:

- jednobodové křížení,
- vícebodové křížení,
- uniformní křížení.

Jednobodové křížení

Při jednobodovém křížení je zvolena jedna hranice v chromozomu, od které jsou data rodičů prohozena a tak vzniknou dva noví potomci. Princip je znázorněn na následujícím obrázku (Obrázek 2).



Obrázek 2 - Jednobodové křížení

Vícebodové křížení

Bývá také označováno jako k-bodové křížení. Ostatní metody z tohoto způsobu vycházejí. Princip této metody je tedy totožný jako u křížení jednobodového, jen počet hranic je větší než jedna.

Uniformní křížení

Uniformní křížení pracuje jako předchozí vícebodové křížení, kde se ovšem hranice nachází za každým jednotlivým genem. Při tvorbě prvního potomka se pro každý gen náhodným způsobem určí, zda mu bude přisouzena alela genu prvního či druhého rodiče. Druhý potomek je následně vytvořen z genů, jejichž hodnoty jsou dány alelami opačného rodiče. Uniformní křížení zpravidla vytvoří potomky s více odlišnou charakteristikou, než křížení jednobodové.

V případě, že není použita binární reprezentace jedinců, ale chromozom jedince je například tvořen geny, jejichž hodnotami jsou reálná čísla, lze použít následující operátory křížení:

- Aritmetický – alela vznikajícího chromozomu je dána aritmetickým průměrem hodnot genů předků.
- Geometrický – nová alela je odmocninou součtu rodičovských alel.
- Rozšiřující – rozdíl rodičovských alel je přičten k větší, respektive odečten od menší z nich.

5.2.6 Operátor mutace

Mutace označuje proces obvykle malých změn v genetickém kódu jedince. Přínos mutace není zaručený, někdy může mutace naopak hodnocení jedince značně zhoršit. Především u binárních reprezentací jedinců může zdánlivě nepříliš významná změna jednoho bitu způsobit velké odchýlení od původního hodnocení jedince. V genetických

algoritmech se používá kvůli zamezení přílišné jednotvárnosti a do řešení vnáší další prvek existující i ve skutečném předobrazu evolučních algoritmů.

Mutace se na jedince aplikuje zpravidla ihned po jejich vzniku křížením, přičemž k ní dochází vždy jen s předem danou, většinou velmi malou, pravděpodobností.

Pro mutaci binárně reprezentovaných jedinců se nabízí jednoduché řešení, a to změna alely náhodně zvoleného genu chromozomu z nuly na jednotku, respektive z jednotky na nulu. U reprezentací reálnými čísly lze mutaci provést například přičtením, čí odečtením samotné alely, vynásobené předem daným, malým poměrem.

5.2.7 Průběh algoritmu

Genetický algoritmus má dvě základní implementace, jejichž průběh se mírně liší, nazývají se:

- steady-state,
- generational.

Steady-state

Tato varianta v populaci uchovává jen jednu generaci[1], jejíž nejhorší jedinci jsou v rámci evoluce nahrazeni novými, pokud možno silnějšími. Algoritmus lze rozdělit do šesti kroků:

1. Vytvoření počáteční generace G (0).
2. Ohodnocení kvality jedinců ohodnocovací funkcí.
3. Výběr jedinců z předešlé generace G (T-1) prostřednictvím operátoru výběru. Následná aplikace křížení a mutace s pomocí příslušných operátoru. Vznik nového jedince.
4. Vyhodnocení kvality nově vzniklého jedince a nahrazení nejhoršího jedince generace G tímto novým jedincem.
5. Opakování algoritmu od kroku 2 do splnění ukončovací podmínky.
6. Nalezení nejlepšího řešení v jedinci, který má v rámci generace nejvyšší kvalitu.

Generational

Rozdíl oproti prvnímu přístupu spočívá v tom, že v každém cyklu je vytvořena generace nová, do které jsou vloženi vygenerovaní potomci. Postup je poté následující:

1. Vytvoření počáteční generace G (0).
2. Ohodnocení kvality jedinců ohodnocovací funkcí.
3. Vytvoření nové prázdné generace G (T).

[1] Termíny populace a generace jsou v problematice genetických algoritmů zaměnitelné, jelikož v případech, kdy je populace tvořena jednou generací je tato generace zároveň i celkovou populací

4. Výběr jedinců z předešlé generace G (T-1) prostřednictvím operátoru výběru. Následná aplikace křížení a mutace s pomocí příslušných operátoru. Vznik nového jedince.
5. Vyhodnocení kvality nového jedince a vložení tohoto jedince do nové generace G (T).
6. Nahrazení staré generace G (T-1) novou G (T).
7. Opakování algoritmu od kroku 2 do splnění ukončovací podmínky.
8. Nalezení nejlepšího řešení v jedinci z poslední generace G (T), který má v rámci této generace nejvyšší kvalitu.

5.2.8 Podmínka ukončení

Algoritmus může být například ukončen po provedení stanoveného počtu cyklů, nebo pokud je stávající nejlepší nalezené řešení shledáno dostačujícím, případně může být využita kombinace těchto podmínek.

5.3 Nelder-Mead algoritmus

Tento oddíl vychází ze zdrojů [15], [16] a [17].

Metoda Nelder-Mead, označována také jako „downhill simplex method“, se řadí ke klasickým optimalizačním technikám. Jedná se heuristický algoritmus určený k nalezení minima funkce ve vícerozměrném prostoru. Název metody je složen z příjmení Johna Neldera a Rogera Meada, kteří ji roku 1965 poprvé popsali ve své práci „A simplex method for function minimization“. Praktické využití nalézá tato technika především ve statistice.

Metoda k hledání minima využívá tzv. simplex, což je N-rozměrný polytop tvořený N plus jedna vrcholy. Kde N je počet neznámých optimalizované funkce. Každý vrchol tohoto polytopu má tedy N rozměrů.

Základní princip práce spočívá ve vyhodnocení polohy těžiště a vrcholů simplexu vzhledem k hledanému minimu, přičemž následně je dle zjištěných poznatků simplex transformován tak, že se k minimu přiblíží. Tento postup se opakuje až do splnění ukončovací podmínky, kterou může být například dosažení daného počtu cyklů, nebo docílení takového stavu, kdy se vrcholy simplexu a v důsledku toho i jejich funkční hodnoty liší jen minimálně, či jsou si dokonce rovny s předem danou přesností.

Nelder-Mead algoritmus má více implementací. V testovací aplikaci vytvořené v rámci této klasifikační práce je použit postup, který bude následně popsán. Lze jej rozdělit do pěti kroků:

1. Vytvoření počátečního simplexu.
2. Seřazení vrcholů.
3. Nalezení těžiště simplexu.
4. Transformace simplexu.
5. Ukončení optimalizace.

Pro provedení algoritmu je nutné předem stanovit koeficienty reflexe, expanze, kontrakce a redukce, zde označované jako α, γ, ρ a σ. Požadované rozmezí a standardně volené hodnoty těchto koeficientů jsou uvedeny v následující tabulce (Tabulka 3).

Tabulka 3 - Standardní hodnoty koeficientů pro metodu Nelder-Mead

Koeficient	α	γ	ρ	σ
Hodnota	1	2	$1/2$	$1/2$

V příkladech je popsáno hledání minima funkce o dvou neznámých $f(x, y)$, jelikož tento případ, kdy simplex tvoří tři dvourozměrné vrcholy, lze grafický znázornit nejlépe. Počet neznámých N bude tedy roven dvěma.

5.3.1 Vytvoření počátečního simplexu

Nejprve je třeba vygenerovat simplex, který bude tvořen N plus jedna vrcholy o rozměru N. Podobně jako u genetického algoritmu lze vyjít z dosavadních poznatků daného problému a vrcholy simplexu vytvořit na základě k optimu přibližné hodnoty, pokud je nám známa. Tato hodnota je dána vstupním bodem x_{in} a vrcholy simplexu $x_1, \cdots, x_n$ jsou generovány například dle následujících vztahů:

$$x_1 = x_{in} \tag{14}$$

$$x_j = x_1 + h_j e_j, \qquad j \in \{2, \cdots, n\} \tag{15}$$

kde e_j - je jednotkový vektor,

h_j - je velikost kroku ve směru jednotkového vektoru e_j.

Takto vznikne pravidelný počáteční simplex, jehož hrany mají stejnou délku.

V případě, že neexistuje přesnější představa o optimálním vrcholu, hodnoty se generují náhodně, například v daném rozmezí, ve kterém se nalezení optima očekává.

Důležité je, aby například v popisovaném případě, kdy simplex obsahuje tři vrcholy, neležely tyto vrcholy v jedné přímce a tvořily trojúhelník. Dále by počáteční simplex měl být relativně velký, aby se zamezilo nalezení pouze lokálního optima.

Po vytvoření výchozího simplexu se přejde k následujícím krokům, které jsou cyklicky opakovány až do splnění podmínky ukončení.

5.3.2 Seřazení vrcholů

Řadit vrcholy není bezpodmínečně nutné, ovšem jedná se o jednoduchý způsob, jak zjistit indexy vrcholů h, s, l pro něž platí:

$$f_h = \max_j f_j \tag{16}$$

$$f_s = \max_{j \neq h} f_j \tag{17}$$

$$f_l = \min_{j \neq h} f_j \tag{18}$$

Jedná se o indexy vrcholů s nejvyšší, druhou nejvyšší a nejnižší hodnotou funkce respektive o indexy nejhoršího, druhého nejhoršího a nejlepšího vrcholu vzhledem k optimálnímu řešení.

V případě seřazení vrcholů tedy platí, že:

$$f(x_1) \leq f(x_2) \leq \cdots \leq f(x_n) \leq f(x_{n+1}) \tag{19}$$

a indexy vrcholů lze určit jako:

$$h = n + 1 \,,$$

$$s = n \,,$$

$$l = 1 \,.$$

V dalším postupu se předpokládá, že vrcholy jsou po tomto kroku seřazené.

5.3.3 Nalezení těžiště simplexu

Toto těžiště zde označíme jako x_0. Určuje se dle všech vrcholů, kromě vrcholu nejhoršího, respektive vrcholu s nejvyšší hodnotou funkce. Vztah pro výpočet těžiště x_0 je tedy:

$$x_0 = \frac{\sum_{i=1}^{n} x_i}{n} \tag{20}$$

Nejedná se o geometrické těžiště. Výsledný bod je dán souřadnicemi, které jsou aritmetickými průměry souřadnic všech vrcholů, kromě vrcholu nejhoršího.

5.3.4 Transformace simplexu

V dalším kroku je třeba určit bod reflexe x_r, který je dán předpisem:

$$x_r = x_0 + \alpha(x_0 - x_{n+1}) \tag{21}$$

Na základě tohoto bodu x_r se algoritmus dále větví na následující úkony:

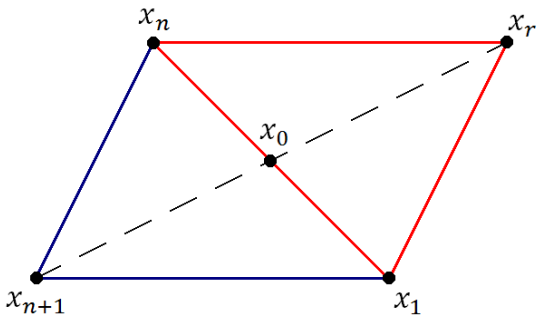
- Reflexe;
- Expanze;
- Kontrakce;
- Redukce;

Reflexe (Reflection)

Reflexe se provádí na základě splnění podmínky, kdy:

$$f(x_1) \leq f(x_r) < f(x_n) \tag{22}$$

Následně je nejhorší bod simplexu x_{n+1} (x_h) nahrazen bodem x_r a cyklus se opakuje od kroku č. 2.



Obrázek 3 - Reflexe simplexu

Expanze (Expansion)

V případě, že platí podmínka:

$$f(x_r) < f(x_1) \tag{23}$$

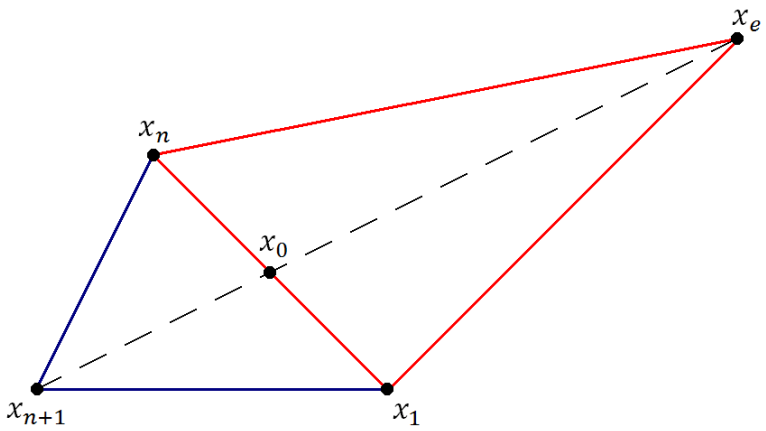
Je zapotřebí učit bod expanze x_e:

$$x_e = x_0 + \gamma(x_0 - x_{n+1}) \tag{24}$$

Pokud je bod expanze lepší, než bod odrazu, tj. platí vztah:

$$f(x_e) < f(x_r) \tag{25}$$

je simplex transformován nahrazením nejhoršího bodu x_{n+1} (x_h) bodem expanze x_e, jinak je bod x_{n+1} (x_h) nahrazen bodem reflexe x_r (dojde tedy k reflexi) a algoritmus dále pokračuje krokem č. 2.



Obrázek 4 - Expanze simplexu

Kontrakce (Contraction)

Pokud nejsou splněny podmínky reflexe a expanze, je jisté, že platí:

$$f(x_r) \geq f(x_n) \tag{26}$$

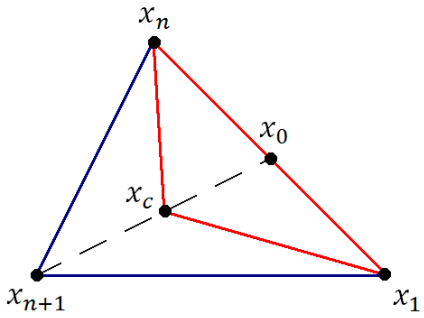
Je vypočten bod kontrakce x_c:

$$x_c = x_{n+1} + \rho(x_0 - x_{n+1}) \tag{27}$$

a testuje se podmínka:

$$f(x_c) < f(x_{n+1}) \tag{28}$$

Pokud platí, je nejhorší bod simplexu x_{n+1} (x_h) nahrazen bodem kontrakce x_c a opět se pokračuje druhým krokem, v opačném případě se provede následující blok algoritmu, tedy redukce.



Obrázek 5 - Vnitřní kontrakce simplexu

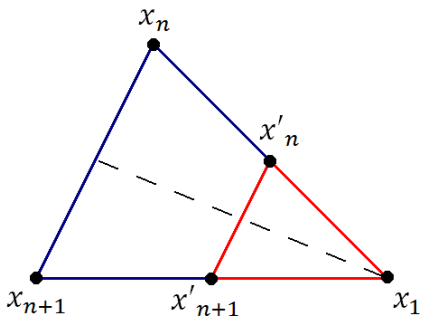
V některých implementacích Nelder-Mead algoritmu v tomto kroku může dojít i k tzv. vnější kontrakci simplexu, což je totéž, jako provedení reflexe a následné vnitřní kontrakce.

Redukce (Reduction)

Při redukci jsou všechny body simplexu, kromě bodu nejlépe hodnoceného (v případě seřazeného simplexu tedy prvního) x_1 (x_l), nahrazeny dle následujícího vztahu:

$$x_i = x_1 + \sigma(x_i - x_1), \qquad i \in \{2, \cdots, n+1\} \tag{29}$$

a přejde se ke kroku č. 2.



Obrázek 6 - Redukce simplexu

5.3.5 Ukončení optimalizace

Nejčastěji bývá ukončení optimalizace podmiňováno třemi typy podmínek, či jejich kombinacemi, jedná se o:

Term_i

Tato podmínka je označována jako doména konvergace nebo ukončení testu. Je splněna, pokud je simplex relativně malý. To znamená, že jeho vrcholy jsou si velmi blízko.

Term_f

Jedná se o funkční hodnotu testu konvergace. Při tomto způsobu je program ukončen, když jsou si funkční hodnoty zkoumané funkce pro všechny, nebo některé vrcholy, relativně blízko.

Fail

Podmínka fail ukončuje optimalizaci v případě, že byl proveden maximální počet daných cyklů, nebo funkční hodnota některého či všech vrcholů překračuje určitou mez. Toto znamená, že hledané řešení se nenachází ve stanoveném intervalu.

6 Popis aplikace

V rámci této klasifikační práce byla vytvořena GUI aplikace v programovacím jazyku java, která má za úkol dle tří výše popsaných optimalizačních metod určit parametry ARMA modelu. V této kapitole bude tato aplikace popsána, především s ohledem na konkrétní zvolené řešení daných optimalizačních algoritmů.

6.1 Struktura aplikace

Jednotlivé třídy jsou dle logických souvislostí rozmístěny do šesti balíčků:

- Funkce;
- Geneticky_algoritmus;
- Metoda_nejmensich_ctvercu;
- Nelder_Mead;
- Optimalizace;
- Graficka_aplikace.

Balíček Funkce

V tomto balíčku je obsažena pouze třída funkce, jejíž jediná metoda f je využívána jako účelová funkce pro algoritmus Nelder-Mead, a také je volána v hodnotící funkci Genetického algoritmu. Tato hodnotící funkce vrací návratovou hodnotu funkce f násobenou mínus jednou.

Balíček Geneticky_algoritmus

Druhý balíček obsahuje třídy genetického algoritmu, jimiž jsou:

- Jedinec;
- Generace;
- Geneticky_algoritmus.

Balíček Metoda_nejmensich_ctvercu

Tento balíček obsahuje pouze dvě třídy.

- Matice;
- Metoda_nejmensich_ctvercu.

Balíček Nelder_Mead

Následující balíček obsahuje tři třídy pro provádění optimalizace metodou Nelder-Mead.

- Simplex;
- Vrchol;

- Nelder_Mead_algoritmus.

Balíček Optimalizace

Balíček Optimalizace obsahuje pouze jednu třídu Optimalizace, která spojuje všechny tři použité techniky.

Balíček Graficka_aplikace

Tento balíček je nejrozsáhlejší. Obsahuje třídy grafických panelů:

- JPanelGraf,
- JPanelGrafBilySum,
- JPanelLegenda,
- JPanelLegendaBilySum,
- JPanelNastaveni,
- JPanelVstup,
- JPanelVystup,

hlavní třídu aplikace

- AplikaceJFrame

a třídu pro převod délky čísla na maximální požadovanou velikost

- Prevod.

6.2 Uživatelské rozhraní

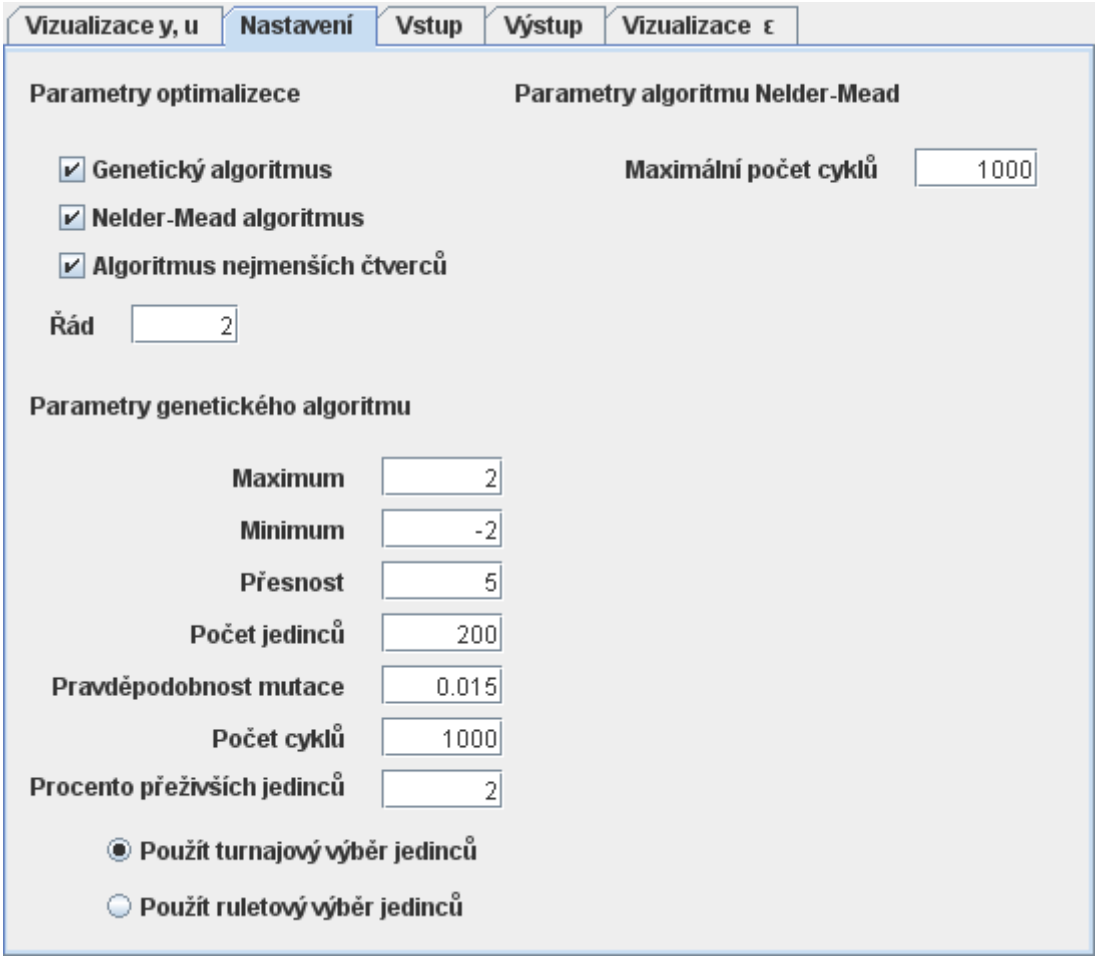
Základní volby programu lze provádět pomocí tlačítek umístěných v horní části okna. Dále aplikace obsahuje pět následujících záložek.

Vizualizace y, u

Tato záložka po načtení vstupních dat zobrazí graf vstupních hodnot y a u. Po provedení optimalizace následně zobrazí predikované časové řady.

Nastavení

V tomto panelu lze nastavit, jaké optimalizační algoritmy budou použity, případně bližší parametry genetického algoritmu a maximální počet cyklů algoritmu Nelder-Mead. Výchozí nastavení, při kterém byly odhadnuty parametry modelů v kapitole 7, je zobrazeno na následujícím obrázku (Obrázek 7):



Obrázek 7 - Výchozí nastavení aplikace

Vstup

Na kartě vstup se po načtení vstupních dat z textového souboru tato data zobrazí. Případně je zde lze dále upravit. Aby se tyto provedené změny projevily v grafu na záložce Vizualizace y, u, je třeba stisknout tlačítko Aktualizuj graf.

Výstup

Záložka výstup po nalezení parametrů tyto parametry zobrazí. Dále také vyobrazí hodnoty vstupů y, k nim se vztahujících predikcí a chyb ε, přičemž pod sloupcem s chybami jsou uvedeny hodnotící kritéria střední absolutní chyba MAE, střední kvadratická chyba MSE a střední absolutní procentuální chyba MAPE (viz pododdíl 3.4.3) pro jednotlivé modely.

Vizualizace ε

Poslední záložka zobrazuje graf hodnot chyb ε.

6.3 Použité implementace optimalizačních algoritmů

6.3.1 Metoda nejmenších čtverců

Konstruktor Metoda_nejmensich_ctvercu

```
public Metoda_nejmensich_ctvercu(   double nYmk[],
                                    double nU[],
                                    int nRad)
```

- nYmk [] - pole výstupních hodnot systému y pro výchozí matice MaticeY a MaticeW
- nU - pole vstupních hodnot do systému u pro výchozí matici MaticeW
- nRad - udává řád procesu

Postup odhadu parametrů

Algoritmus metody nejmenších čtverců vychází z postupu, který je popsán výše v pododdílu 5.1.1 Při požadavku na optimalizaci je provedena následující metoda:

```
public double[] provedAlgoritmus() {
      maticeWT = maticeW.dejTransponovanou();
      maticeWT_W = maticeWT.vynasob(maticeW);
      maticeWT_W_I = maticeWT_W.dejInverzni();
      maticeWT_Y = maticeWT.vynasob(maticeY);
      matice_P = maticeWT_W_I.vynasob(maticeWT_Y);
      nastavVysledky(matice_P.dejPrvky());
      return vysledky;
}
```

Nejobtížnější fází algoritmu je výpočet inverzní matice MaticeWT_W_I podle vztahu:

$$a_{i,j} = \frac{(-1)^{i+j}\left|A_{j,i}\right|}{|A|} \tag{30}$$

kde $\left|A_{j,i}\right|$ - je subdeterminant získaný z matice A, ve které byli vynechány řádky i a j a

$|A|$ – je determinant matice A.

K vypočtení inverzní matice slouží metoda dejInverzni třídy Matice:

```
public Matice dejInverzni() {
      double det = Determinant(prvky, pocetRadku);
      double I[][] = new double[pocetRadku][pocetSloupcu];
      for (int aktRadek = 0; aktRadek < pocetRadku; aktRadek++) {
            for (int aktSloupec = 0; aktSloupec < pocetSloupcu;
            aktSloupec++) {
                  I[aktRadek][aktSloupec] = (Math.pow(-1, (aktRadek + 1 +
                  aktSloupec + 1)) *
                  Determinant(dejMaticiSubdeterminant(aktSloupec,
                  aktRadek), pocetRadku - 1)) / det;
            }
      }
```

```
        return new Matice(I);
}
```

K výpočtu determinantů je používán rekurzivní algoritmus. V případě, že jsou hledány parametry modelu vyšších řádů, než pět, má matice MaticeWT_W rozměr větší než deset a v důsledku toho je optimalizace relativně časově náročná.

6.3.2 Genetický algoritmus

Konstruktor Geneticky_algoritmus

```
public Geneticky_algoritmus(  short nPocetJedincu,
                              double nPravMutace,
                              double nParPromennych[][],
                              Funkce nFunkce,
                              short nProcentoPrezivsich,
                              short nTypVyberu)
```

- nPocetJedincu - udává počet jedinců v generaci
- nPravMutace - představuje pravděpodobnost mutace jedinců
- nParPromennych[][] - je dvourozměrné pole o n řádcích a třech sloupcích. Řádky představují parametry pro jednotlivé geny, přičemž hodnota nParPromennych[n][0] je minimum genu, nParPromennych[n][1] je maximum genu a nParPromennych[n][2] představuje počet desetinných míst. Počet řádků n tohoto pole tedy udává počet genů jedince.
- nFunkce - je účelová funkce
- nProcentoPrezivsich - udává celočíselné procento nejlepších jedinců, kteří se automaticky přenášejí do další generace
- nTypVyberu - udává typ operátoru výběru, hodnota 1 představuje turnajový operátor výběru, hodnota 2 operátor ruletový.

Kódování

V aplikaci je použito kódování jedinců na binární chromozomy, přičemž vzhledem k tomu, že jedinci počáteční generace se generují náhodně na úrovni binárního chromozomu, není zajištěno zakódování, ale pouze dekódování chromozomu jedince z binární podoby na pole reálných čísel. Toto dekódování provádí následující metoda:

```
private double[] dejPoleGenuJedince(int index) {
      double[] poleGenu = new double[pocetGenuJedince];
      StringBuffer chromozomJedince = jedinci.get(index).dejChromozom();
      String genJedince;
      long genDekadicky;
      int pocetBituGenu;
      for (int aktGen = 0; aktGen < pocetGenuJedince; aktGen++) {
            pocetBituGenu = maxGenuJedinceBinarne[aktGen].length();
            genJedince = chromozomJedince.substring(aktGen *
            pocetBituGenu, (aktGen + 1) * pocetBituGenu);
            genDekadicky = Long.parseLong(genJedince, 2);
            poleGenu[aktGen] = genDekadicky / Math.pow(10,
            parametryJedince[aktGen][2]) + parametryJedince[aktGen][0];
      }
```

```
	return poleGenu;
}
```

Způsob vytvoření počáteční populace

Jak již je uvedeno výše, počáteční generace jedinců je vytvářena zcela náhodně na úrovni binárního chromozomu. Negenerují se tedy náhodná reálná čísla, která se následně převádějí na binární alely genů, ale přímo ony binární alely. Chromozomy jednotlivých jedinců generuje metoda:

```
private StringBuffer generujChromozomJedince() {
	int pocetBituGenu;
	StringBuffer novyChromozom = new StringBuffer();
	Random nahodneCislo = new Random();
	for (int aktGen = 0; aktGen < pocetGenuJedince; aktGen++) {
		pocetBituGenu = maxGenuJedinceBinarne[aktGen].length();
			for (int aktBit = 0; aktBit < pocetBituGenu; aktBit++)
			{
				novyChromozom.append(Integer.toString(nahodneCisl
				o.nextInt(2)));
			}
	}
	pocetBituJedince = novyChromozom.length();
	return novyChromozom;
}
```

Ohodnocovací („fitness“) funkce

Jako fitness funkce je použita metoda dejHodnotuJedince, která vrací návratovou hodnotu v konstruktoru zadané účelové funkce násobenou číslem mínus jedna.

```
private double dejHodnotuJedince(int index) {
	return -1 * fitness.f(dejPoleGenuJedince(index));
}
```

Toto řešení je postačující pouze pro turnajový operátor výběru. Pro operátor ruletový je třeba zajistit, aby hodnocení jedinců nebylo záporné. Tento požadavek je ošetřen v metodě nastavPravdepodobnost, která je volána v případě použití ruletového operátoru výběru.

Operátory výběru

Aplikace umožňuje zvolit procento nejlepších jedinců generace, kteří nezměněni přejdou do generace následující. Pro výběr ostatních jedinců lze zvolit mezi dvěma typy operátorů výběru. Tím prvním je turnajový výběr jedinců, který provádí metoda vyberJedinceTurnaj.

```
private int vyberJedinceTurnaj() {
	Random nahodneCislo = new Random();
	int indexSoupere1, indexSoupere2;
	indexSoupere1 = nahodneCislo.nextInt(pocetJedincu);
	do {
		indexSoupere2 = nahodneCislo.nextInt(pocetJedincu);
	} while (indexSoupere2 == indexSoupere1);
```

```
      if (dejHodnoceniJedince(indexSoupere1) >
      dejHodnoceniJedince(indexSoupere2)) {
            return indexSoupere1;
      } else {
            return indexSoupere2;
      }
}
```

Druhou možností je zvolit ruletový výběr, nezaložený na pořadí jedinců, pro který je určena metoda vyberJedinceRuleta:

```
private int vyberJedinceRuleta() {
      double intervalJedinceMin = 0;
      double intervalJedinceMax = 0;
      Random nahodneCislo = new Random();
      double ukazatel = nahodneCislo.nextDouble();
      for (int aktJedinec = 0; aktJedinec < pocetJedincu; aktJedinec++) {
            intervalJedinceMax = intervalJedinceMin +
            jedinci.get(aktJedinec).dejPravdepodobnost();
            if (ukazatel >= intervalJedinceMin && ukazatel <
            intervalJedinceMax) {
                  return aktJedinec;
            }
            intervalJedinceMin = intervalJedinceMax;
      }
      return 0;
}
```

Obě funkce jsou privátními metodami třídy Generace.

Operátor křížení

Křížení v aplikace je jednobodové (viz Obrázek 2), bez možnosti zvolit jiný typ.

Operátor mutace

Mutace je prováděna na úrovni binárního chromozomu. V případě, že má u daného jedince dojít k jeho mutaci, je znegován jeden náhodně vybraný bit z jeho chromozomu. Tento operátor zabezpečuje metoda mutaceChromozomu třídy Jedinec.

Průběh algoritmu

Genetický algoritmus pracuje na principu generational, kdy je v každém kroku cyklu vytvořena nová generace a do ní jsou vložení výběrem a křížením vytvoření potomci. Provedení algoritmu zajišťuje metoda provedAlgoritmus ve třídě Geneticky_algoritmus:

```
public double[] provedAlgoritmus(int pocetCyklu) {
      int cyklus = 0;
      Generace generace = new Generace(pocetJedincu, parPromennych,
      pravdepodobnostMutace, funkce, procentoPrezivsich, typVyberu);
      while (cyklus < pocetCyklu) {
            cyklus++;
            generace = generace.vytvorGeneraciJednobodovymKrizenim();
      }
```

```
	return generace.dejPoleGenuNejlepsihoJedince();
}
```

Nejdůležitější částí této funkce je volání metody vytvorGeneraciJednobodovymKrizenim, která provede potřebné úkony a vrátí nově vytvořenou generaci. Ukončovací podmínkou je dosažení stanoveného počtu cyklů.

6.3.3 Nelder-Mead algoritmus

Konstruktor

```
public Nelder_Mead_algoritmus(short nPocetNeznamych, Funkce nFunkce)
```

- nPocetNeznamych - počet hledaných neznámých funkce
- nFunkce - účelová funkce třídy Funkce

Vytvoření počátečního simplexu

Vrcholy počátečního simplexu jsou generovány náhodně pomocí metody generujSimplex třídy Simplex:

```
private void generujSimplex() {
	double noveHodnoty[] = new double[pocetHodnotVrcholu];
	Random nahodneCislo = new Random();
	for (int aktVrchol = 0; aktVrchol < pocetVrcholu; aktVrchol++) {
		for (int aktHodnota = 0; aktHodnota < pocetHodnotVrcholu;
		aktHodnota++) {
			noveHodnoty[aktHodnota] = nahodneCislo.nextDouble();
		}
		vrcholy.add(new Vrchol(noveHodnoty));
	}
}
```

Seřazení vrcholů

Vrcholy simplexu jsou uloženy v datové struktuře List. V rámci tohoto listu jsou řazeny pomocí Metody sort třídy Collections.

```
void seradVrcholy() {
	Collections.sort(vrcholy, pComparator);
}
```

Nalezení těžiště simplexu

Těžiště je určeno pomocí metody najdiTeziste ve třídě Simplex.

Transformace simplexu

Pro reflexi, expanzi, kontrakci a redukci jsou ve třídě Simplex vytvořeny odpovídající metody.

Ukončení optimalizace

Aplikace využívá dvě podmínky, a to podmínku typu term_f a fail (viz pododdíl 5.3.5), podmínka term_f je ověřována pomocí metody třídy Simplex jsouSiRovny, která vrací true v případě, že funkční hodnoty účelové funkce jsou si pro všechny vrcholy rovny. Podmínka fail ukončuje cyklus v případě, že je dosaženo stanoveného počtu cyklů.

Algoritmus je proveden zavoláním metody provedAlgoritmus, která je veřejnou metodou třídy Nelder_Mead_algoritmus. Tato metoda je pro svou rozsáhlost uvedena v příloze na konci práce (Příloha A).

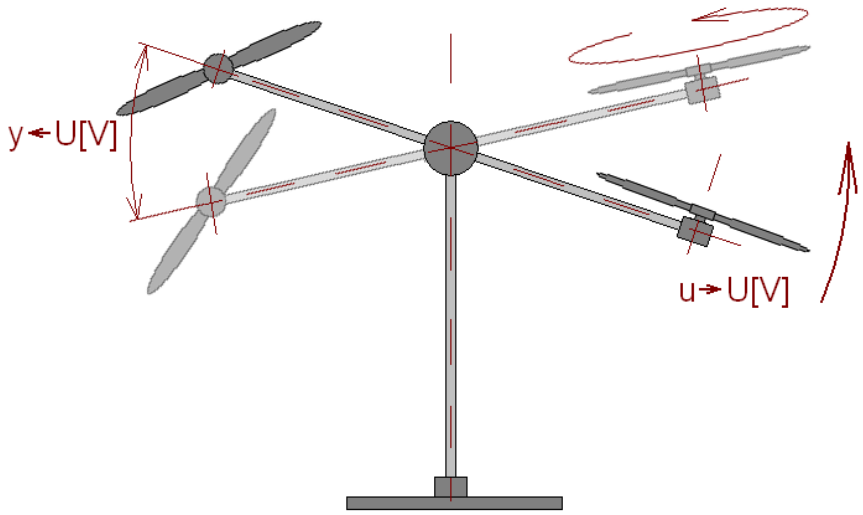
7 Porovnání metod pro odhad parametrů ARMA modelu

7.1 Modelované systémy užité při porovnání optimalizačních technik

Porovnání optimalizačních technik bylo provedeno na základě odhadu parametrů pro tři různé ARMA modely:

1. ARMA (1, 1) simulovaný lineární systém 1. řádu.
2. ARMA (2, 2) simulovaný lineární systém s šumem přičteným na výstupu.
3. ARMA (2, 2) výrazně nelineární laboratorní model vrtulníku.

Data systému č. 3 byla naměřena v laboratoři na skutečném modelu helikoptéry. Vstupem do systému, označeném jako u, bylo napětí elektromotorku horizontální vrtule naměřené ve voltech. Výstupem y bylo taktéž napětí ve voltech, udávající výškový úhel natočení helikoptéry. Pro vytvoření modelu byla k dispozici časová řada složená z 501 jednotlivých pozorování. Nákres tohoto modelu je uveden na následujícím obrázku (Obrázek 8).



Obrázek 8 - Nákres laboratorního modelu helikoptéry

Data systémů 1 a 2 byla získána simulací zvoleného dynamického systému v programu Simulink. V těchto případech byly vygenerovány časové řady o délce 201 pozorování.

Řády modelů 2 a 3 byly stanoveny na základě série pokusných odhadů. Následně byly provedeny pro každý z těchto modelů tři odhady parametrů. Konfigurace aplikace byla defaultní (viz Obrázek 7), jen pro model č. 1 byl nastaven 1. řád. Výsledkům těchto odhadů je věnován následující oddíl.

7.2 Výsledné modely

7.2.1 Parametry modelů

Odhady parametrů pro všechny modely a jednotlivé testy jsou uvedeny v následující tabulce (Tabulka 4).

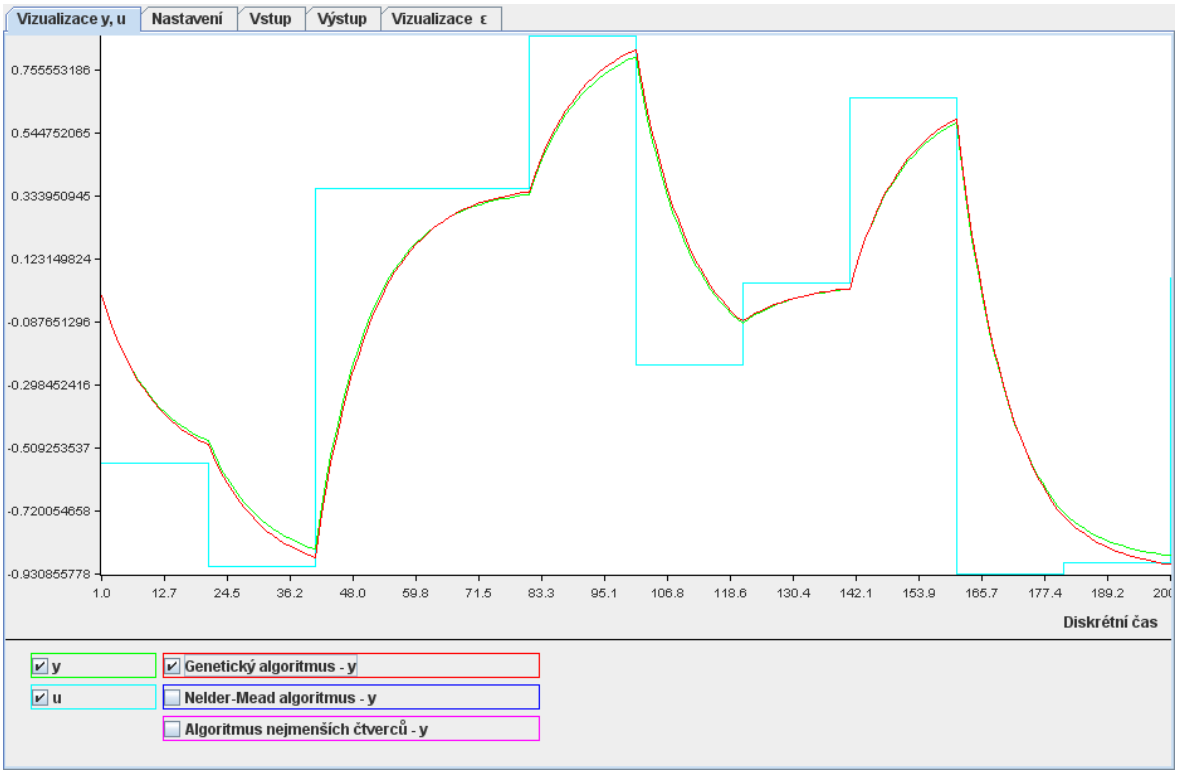
Tabulka 4 – Odhady parametrů modelů

Model		Číslo testu	Parametr	Odhady parametrů		
Číslo	Typ			GA	NMA	MNČ
1	ARMA (1, 1)	1	$\emptyset_1$	0.90816	0.90483741833	0.90483741833
			θ_1	-0.09536999999	-0.09516258166	-0.09516258166
		2	$\emptyset_1$	0.88255	0.90483741833	0.90483741833
			θ_1	-0.11585	-0.09516258166	-0.09516258166
		3	$\emptyset_1$	0.9184	0.90483741833	0.90483741833
			θ_1	-0.08512	-0.09516258166	-0.09516258166
2	ARMA (2, 2)	1	$\emptyset_1$	0.17087999999	0.3874079008	0.38740789016
			θ_1	-0.36159999999	-0.08570024302	-0.08570024792
			$\emptyset_2$	0.78528	0.43413346984	0.43413348234
			θ_2	0.29375999999	-0.07318583806	-0.07318583325
		2	$\emptyset_1$	0.45746999999	0.38740789293	0.38740789016
			θ_1	-0.19777	-0.08570023922	-0.08570024792
			$\emptyset_2$	0.37566999999	0.43413347981	0.43413348234
			θ_2	0.048	-0.07318584222	-0.07318583325
		3	$\emptyset_1$	0.48319	0.38740789705	0.38740789016
			θ_1	-0.17728999999	-0.08570024297	-0.08570024792
			$\emptyset_2$	0.33470999999	0.434133481	0.43413348234
			θ_2	0.01727999999	-0.07318583303	-0.07318583325
3	ARMA (2, 2)	1	$\emptyset_1$	1.27679	1.83158442723	1.83158442097
			θ_1	0.47807	-0.01262388454	-0.01262388765
			$\emptyset_2$	-0.42425	-0.8701634488	-0.87016344192
			θ_2	-0.60737	-0.01845606951	-0.01845606574
		2	$\emptyset_1$	2.02238999999	1.83158442443	1.83158442097
			θ_1	-0.09536999999	-0.01262388596	-0.01262388765
			$\emptyset_2$	-1.07840999999	-0.87016344543	-0.87016344192
			θ_2	0.04798999999	-0.01845606745	-0.01845606574
		3	$\emptyset_1$	1.20510999999	1.83158443365	1.83158442097
			θ_1	-0.52545	-0.01262389088	-0.01262388765
			$\emptyset_2$	-0.25903999999	-0.87016345161	-0.87016344192
			θ_2	0.47807999999	-0.01845606028	-0.01845606574

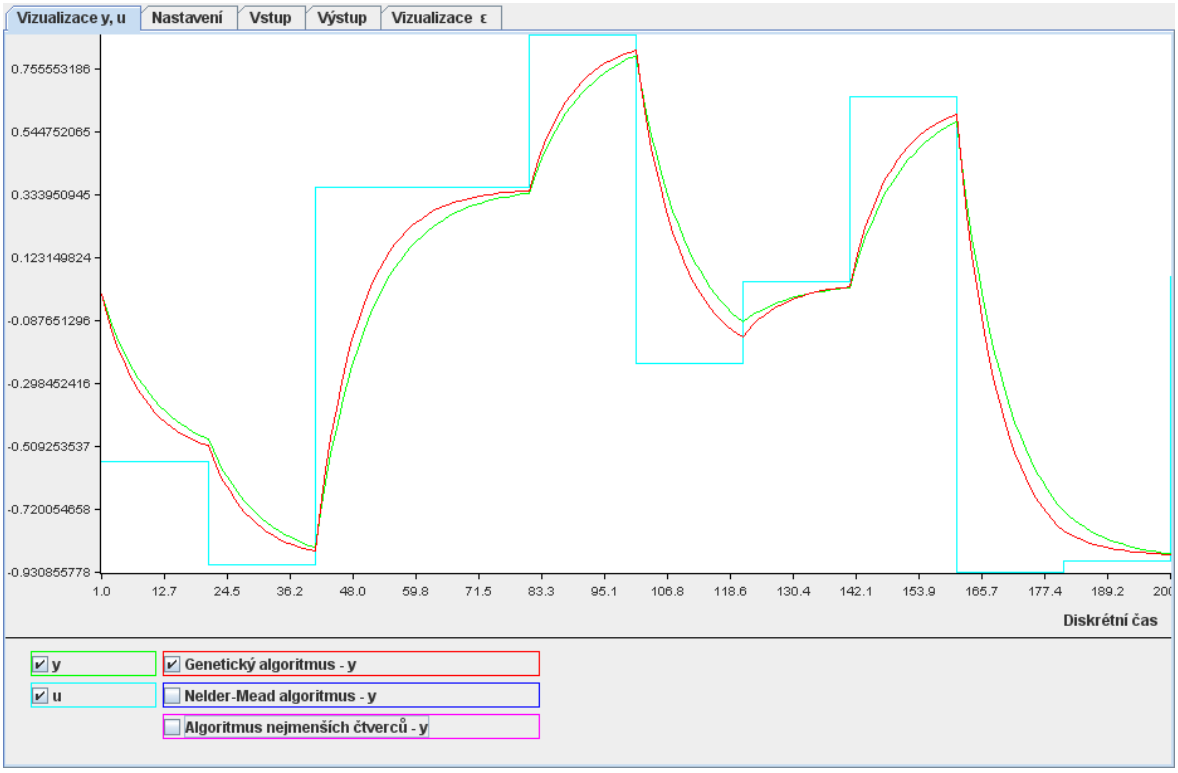
7.2.2 Grafické zobrazení výsledků

Porovnání výsledných predikcí s modelovanými časovými řadami je graficky vyobrazeno v následujících grafech.

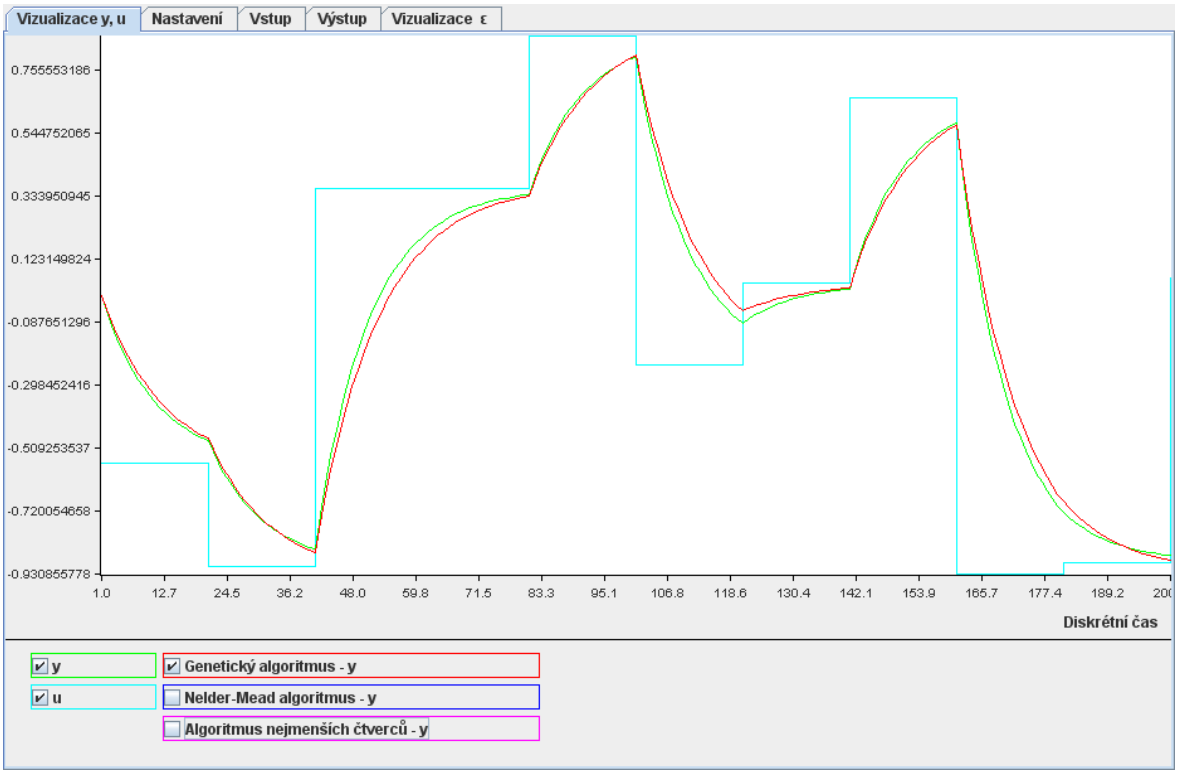
Model č. 1 – ARMA (1, 1)



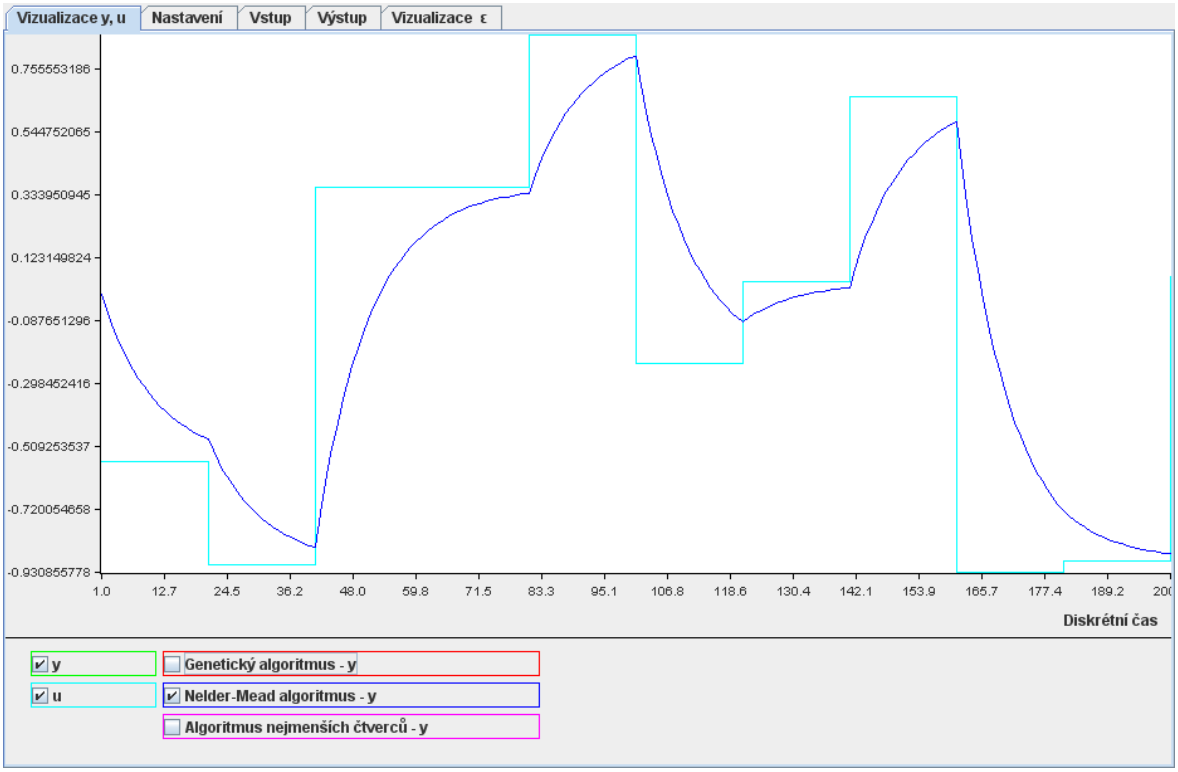
Obrázek 9 - Graf predikcí modelu č. 1 pro parametry odhadnuté GA, test č. 1



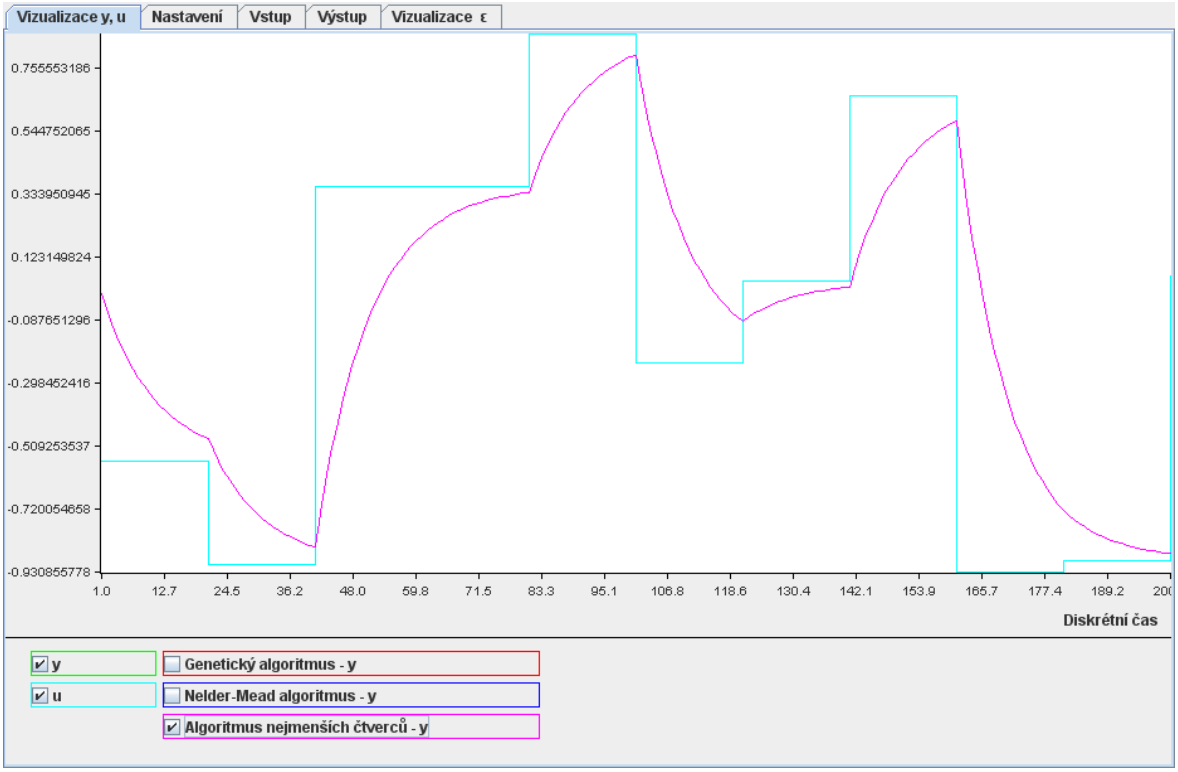
Obrázek 10 - Graf predikcí modelu č. 1 pro parametry odhadnuté GA, test č. 2



Obrázek 11 - Graf predikcí modelu č. 1 pro parametry odhadnuté GA, test č. 3



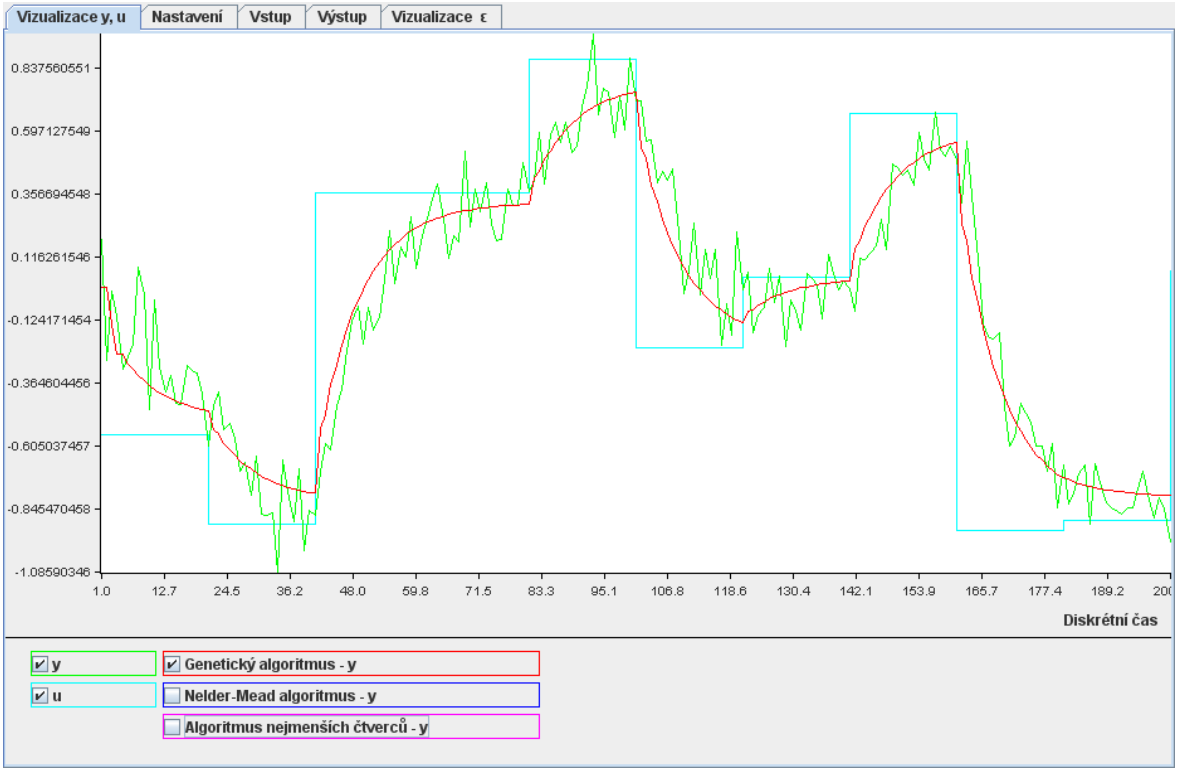
Obrázek 12 - Graf predikcí modelu č. 1 pro parametry odhadnuté NMA, test č. 1-3



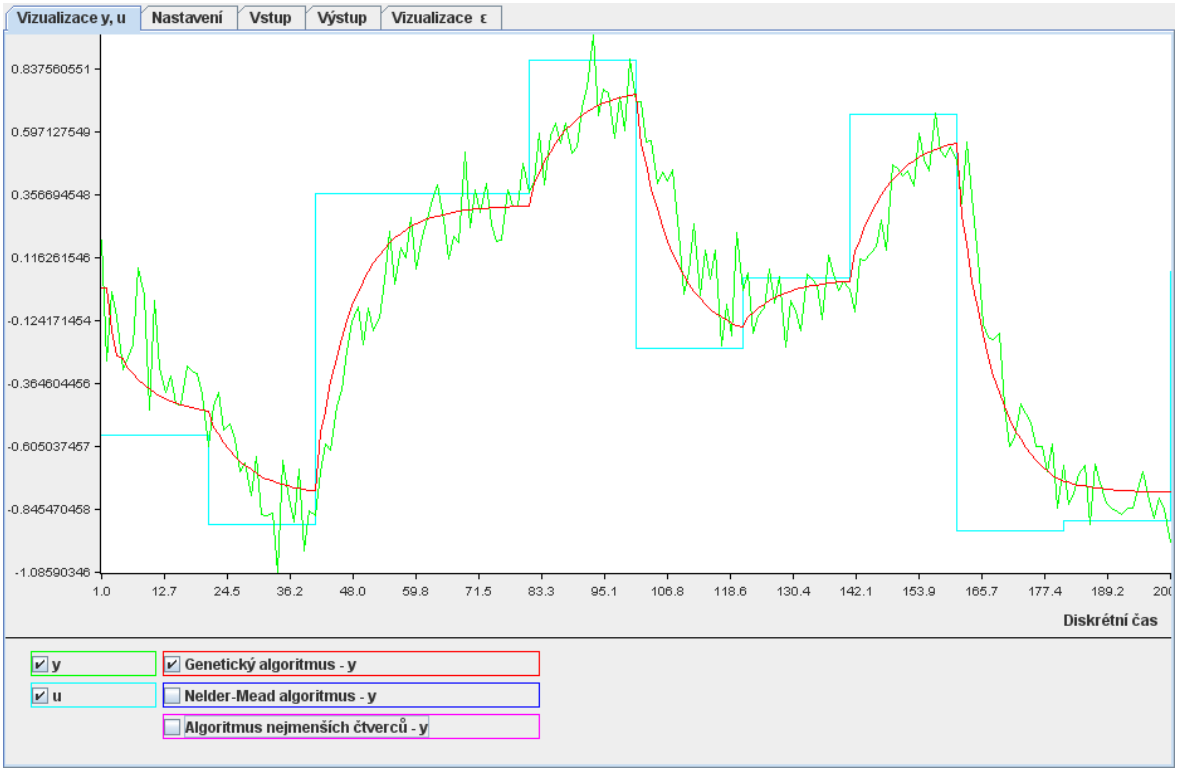
Obrázek 13 - Graf predikcí modelu č. 1 pro parametry odhadnuté MNČ, test č. 1-3

Model č. 2 – ARMA (2, 2)

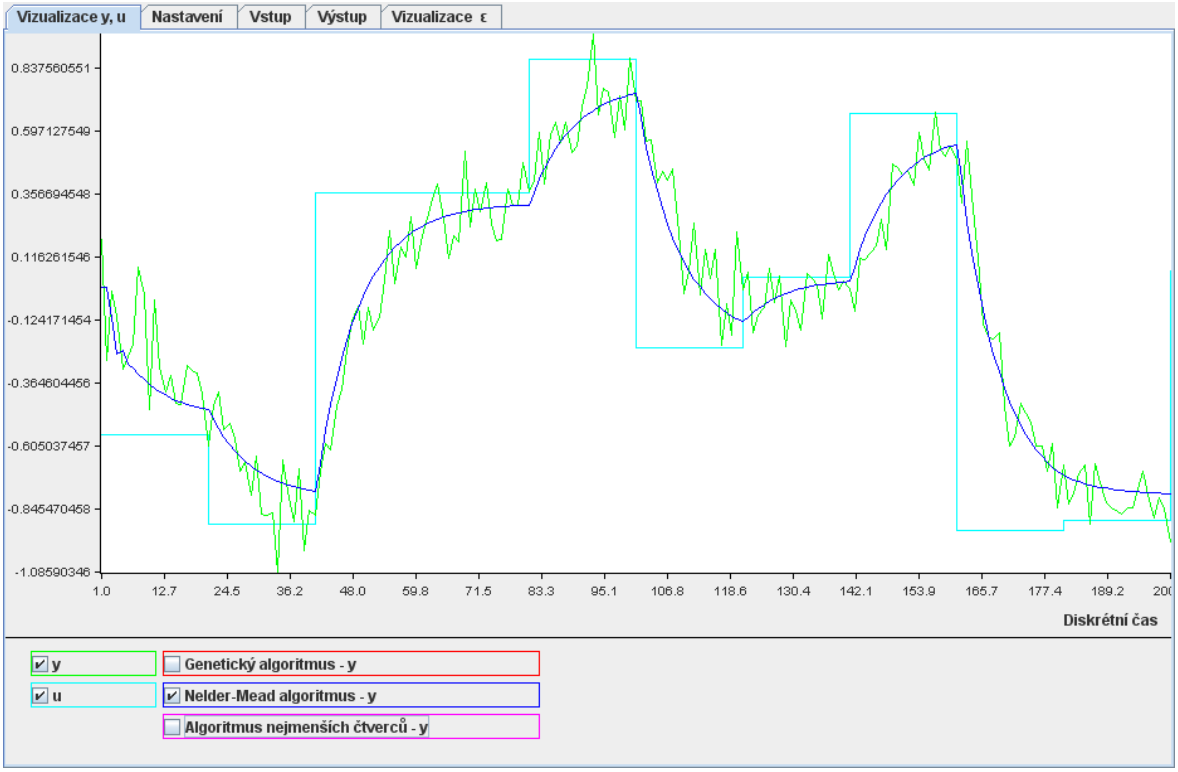
Obrázek 14 - Graf predikcí modelu č. 2 pro parametry odhadnuté GA, test č. 1



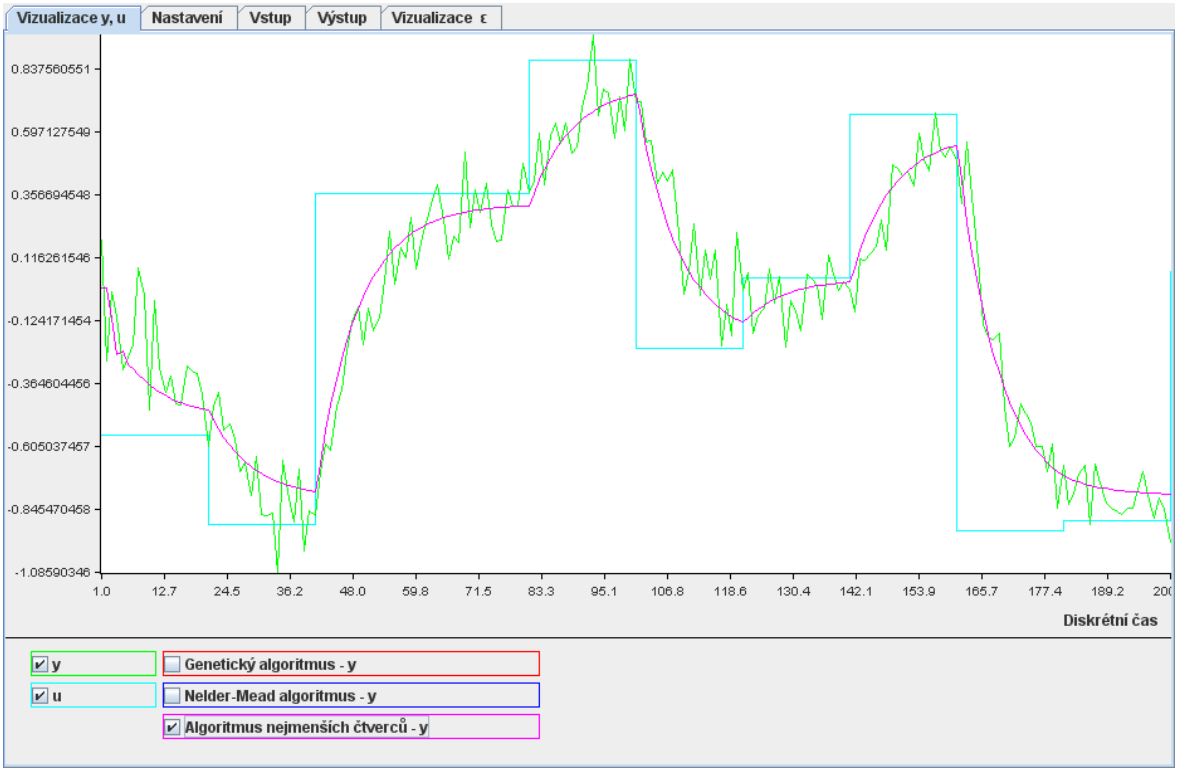
Obrázek 15 - Graf predikcí modelu č. 2 pro parametry odhadnuté GA, test č. 2



Obrázek 16 - Graf predikcí modelu č. 2 pro parametry odhadnuté GA, test č. 3

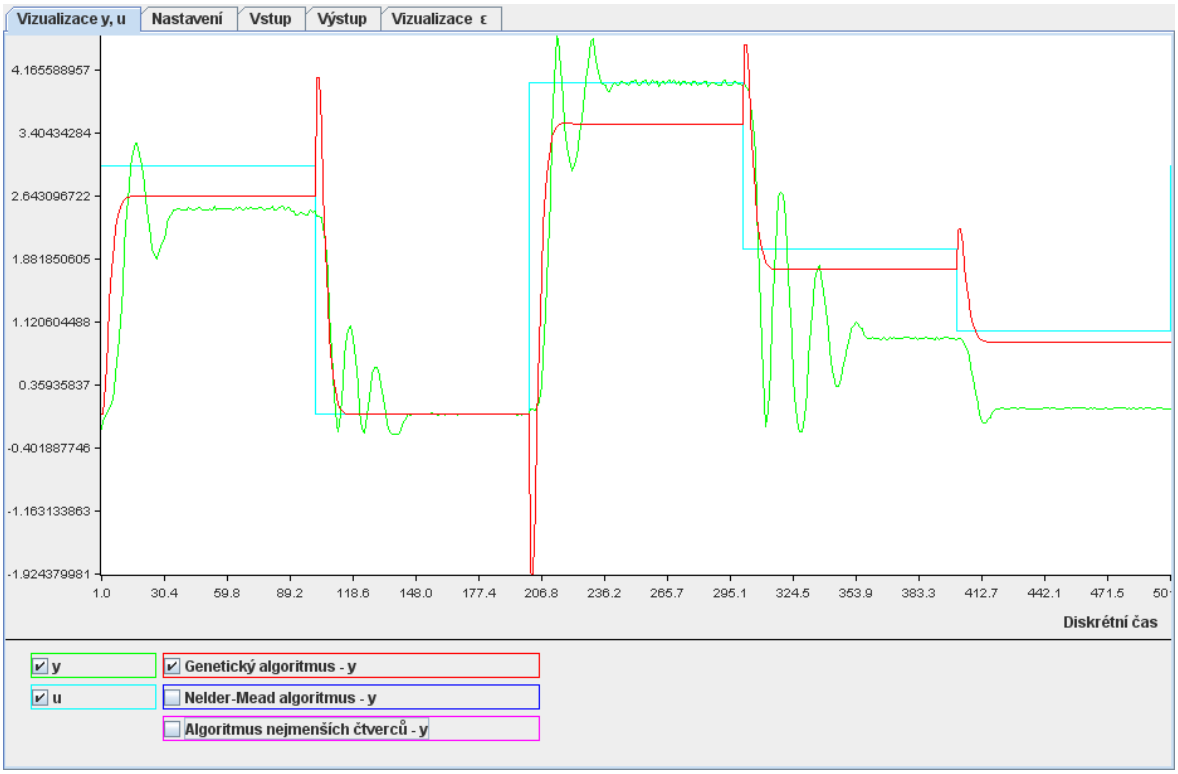


Obrázek 17 - Graf predikcí modelu č. 2 pro parametry odhadnuté NMA, test č. 1-3

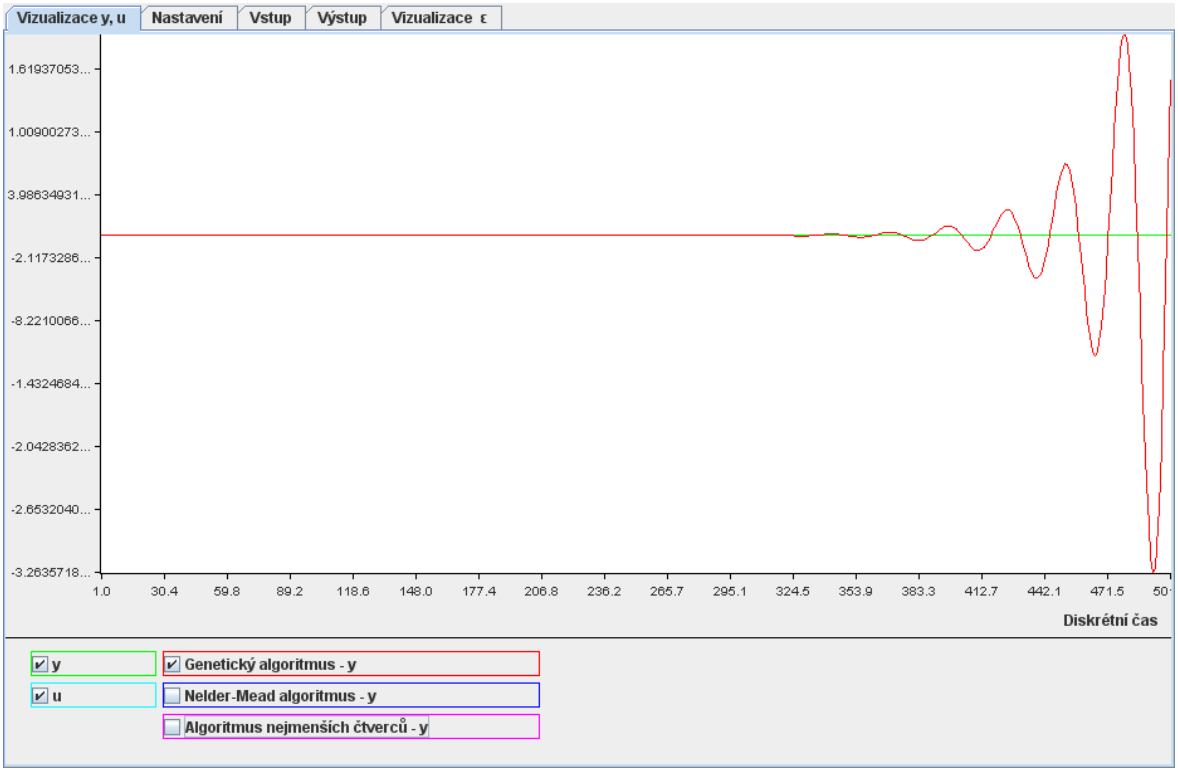


Obrázek 18 - Graf predikcí modelu č. 2 pro parametry odhadnuté MNČ, test č. 1-3

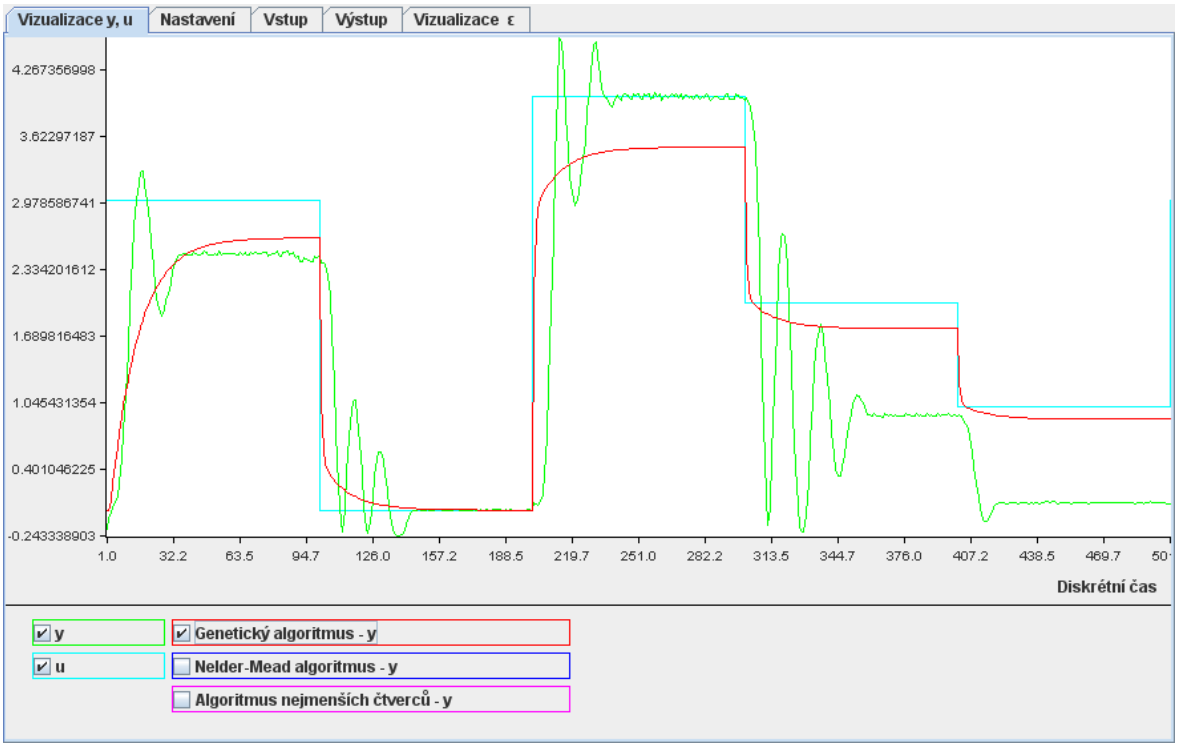
Model č. 3 – ARMA (2, 2)



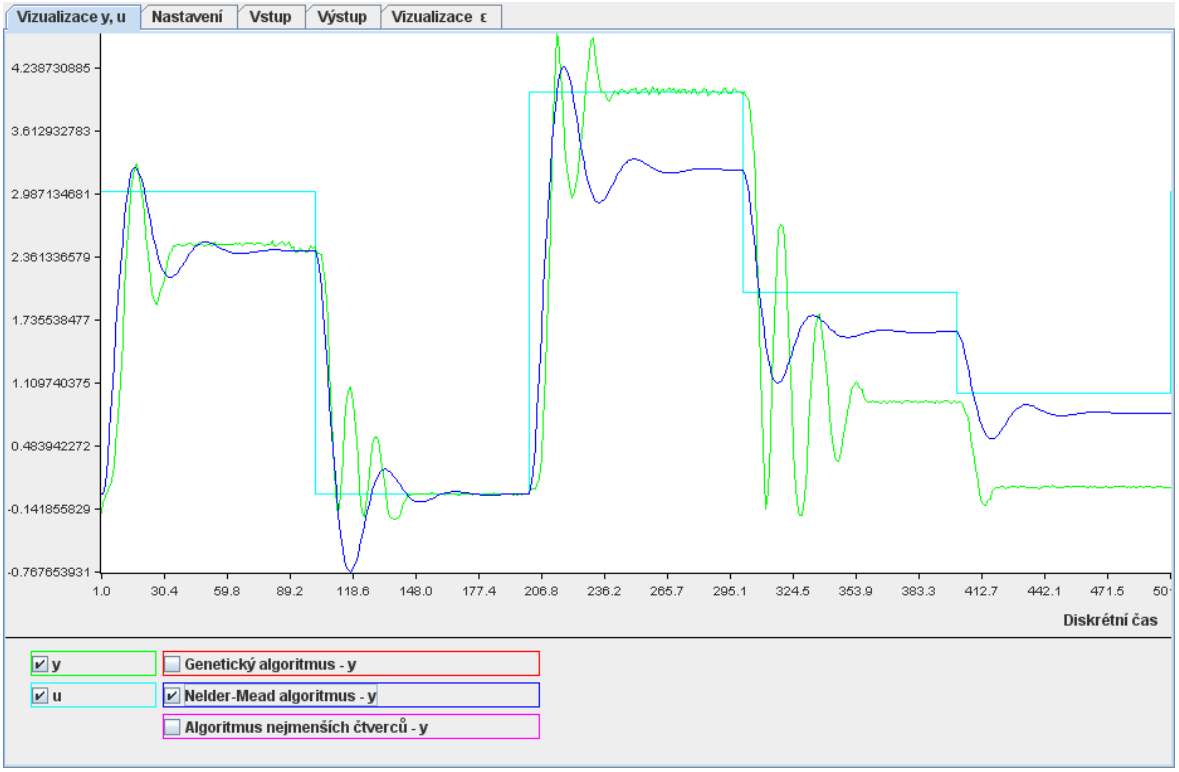
Obrázek 19 - Graf predikcí modelu č. 3 pro parametry odhadnuté GA, test č. 1



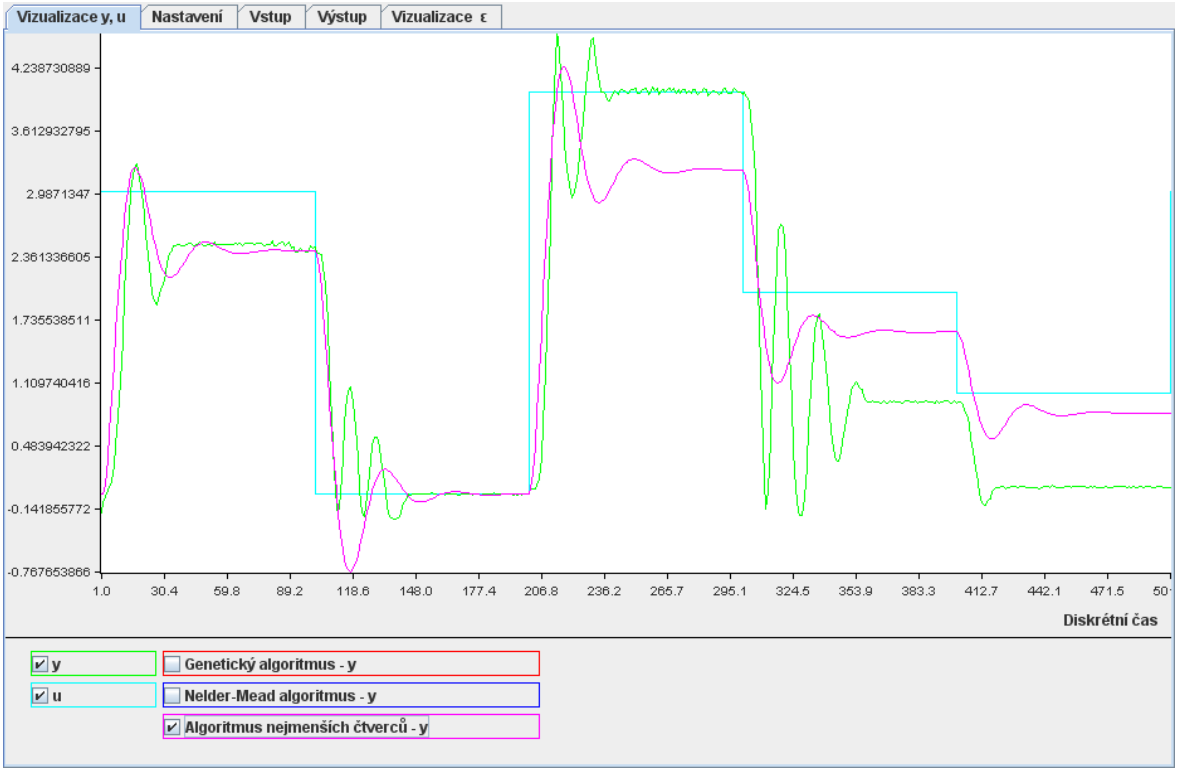
Obrázek 20 - Graf predikcí modelu č. 3 pro parametry odhadnuté GA, test č. 2



Obrázek 21 - Graf predikcí modelu č. 3 pro parametry odhadnuté GA, test č. 3



Obrázek 22 - Graf predikcí modelu č. 3 pro parametry odhadnuté NMA, test č. 1-3



Obrázek 23 - Graf predikcí modelu č. 3 pro parametry odhadnuté MNČ, test č. 1-3

7.2.3 Hodnotící kritéria

Tabulka 5 - Hodnotící kritéria verifikace modelů

Model		Kritérium	Číslo testu	Čas odhadu [ns]		
Číslo	Typ			GA	NMA	MNČ
1	ARMA (1, 1)	*MAE*	1	0.01249239647	4.0E-11	4.0E-11
			2	0.0406444208	4.0E-11	4.0E-11
			3	0.02450277538	4.0E-11	4.0E-11
			Průměr	0.03257359809	4.0E-11	4.0E-11
		MSE	1	0.041759515	0	0
			2	0.0023474576	0	0
			3	0.182847537	0	0
			Průměr	0.0756515032	0	0
		MAPE	1	9.59479552716	5.677E-8	5.677E-8
			2	45.2014461701	5.677E-8	5.677E-8
			3	40.1882580039	5.677E-8	5.677E-8
			Průměr	31.66149990038	5.677E-8	5.677E-8
2	ARMA (2, 2)	*MAE*	1	0.19250281141	0.09204746448	0.0920474635
			2	0.09965487115	0.09204746291	0.0920474635
			3	0.10494148124	0.09204746216	0.0920474635
			Průměr	0.13236638793	0.092047463183	0.0920474635
		MSE	1	0.05318383146	0.01387401821	0.0138740179
			2	0.01601559622	0.01387401775	0.0138740179
			3	0.01779571722	0.01387401757	0.0138740179
			Průměr	0.02899838163	0.013874017843	0.0138740179
		MAPE	1	149.606297331	68.3363447023	68.3363440842

Model		Kritérium	Číslo testu	Čas odhadu [ns]		
Číslo	Typ			GA	NMA	MNČ
			2	73.0295931252	68.336344034	68.3363440842
			3	79.2961506504	68.3363423658	68.3363440842
			Průměr	100.6440137022	68.3363437007	68.3363440842
3	ARMA (2, 2)	*MAE*	1	0.55290357315	0.54155593551	0.54155593225
			2	1587.788928	0.54155593441	0.54155593225
			3	0.58023913953	0.54155593847	0.54155593225
			Průměr	529.6406902375	0.54155593613	0.54155593225
		MSE	1	0.48320680791	0.46118600452	0.46118599609
			2	2.31174204661E15	0.46118599981	0.46118599609
			3	0.59378315915	0.46118600202	0.46118599609
			Průměr	2517143.7356633	0.46118600211	0.46118599609
		MAPE	1	388.782779908	514.418266782	514.418209423
			2	2.063039755E10	514.418232925	514.418209423
			3	435.815121654	514.418232833	514.418209423
			Průměr	15399.705813854	514.41824418	514.418209423

7.2.4 Časy odhadů parametrů

Údaje zaznamenané pomocí funkce nanoTime třídy System zobrazuje tabulka níže (Tabulka 6).

Tabulka 6 - Časy odhadů parametrů modelů

Model		Číslo testu	Čas odhadu [ns]		
Číslo	Typ		GA	NMA	MNČ
1	ARMA (1, 1)	1	2549657568	32977935	228521
		2	2012405208	33862683	70400
		3	2643287904	93541777	69841
		Průměr	2401783560	53460798.33	122920.6667
2	ARMA (2, 2)	1	3583629483	120522301	862400
		2	4003921169	28848639	213714
		3	3610189564	23913654	138844
		Průměr	3732580072	57761531.33	404986
3	ARMA (2, 2)	1	4704071886	82797395	249194
		2	4685126462	177673978	211200
		3	4585941789	199587098	241092
		Průměr	4658380046	153352823.7	233828.6667

7.2.5 Vyhodnocení

Metoda nejmenších čtverců

Z výše uvedených grafů a tabulek je jednoznačné, že po všech stránkách nejvhodnějším řešením pro odhad parametrů modelů ARMA je nejpoužívanější metoda nejmenších čtverců, která pro každý ze tří testů odhadů parametrů určila stejné parametry modelu. Tato metoda je nejpřesnější a pro dané modely také nejrychlejší. V případě, že

jsou hledány parametry modelu s vyšším řádem, je doba odhadu parametrů s pomocí této metody relativně dlouhá, ovšem tento nedostatek je vyvážen její přesností, která je v případě vhodně zvoleného řádu zaručená.

Nelder-Mead algoritmus

Z heuristických algoritmů lze za vhodnější řešení považovat algoritmus Nelder-Mead, který je přesností srovnatelný s metodou nejmenších čtverců. Parametry jednotlivých testů se od sebe liší jen minimálně.

Doba potřebná k nalezení řešení je vyšší než u výše hodnocené techniky, ale přesto velmi nízká.

Genetický algoritmus

Genetický algoritmus se ukázal jako jednoznačně nejhorší řešení. Především odhady parametrů modelu č. 3 byly značně nepřesné. Kritérium MAPE pro první a třetí test odhadů pro tento model sice naznačuje přesnější výsledky než zbylé dvě techniky, ovšem při pohledu na odpovídající grafy (Obrázek 19 a Obrázek 21) a jejich porovnání s grafy predikcí určených dle parametrů odhadnutých NMA (Obrázek 22) a MNČ (Obrázek 23) je zřejmé, že charakter zkoumaného modelu lépe vystihují modely, u nichž byly parametry stanoveny pomocí NMA a MNČ. Při opakovaném testování této techniky také poměrně často docházelo k výrazně nepřesným predikcím, podobně jako u druhého testu modelu č. 3 (Obrázek 20).

Také z hlediska časové náročnosti je genetický algoritmus vzhledem k ostatním technikám výrazně nejhorší. Doba potřebná k odhadu parametrů pro testované modely přesahuje cca třicetkrát čas potřebný pro odhad metodou Nelder-Mead a dokonce cca dvacettisíckrát dobu odhadu metodou nejmenších čtverců. Přesto se při tvorbě tří testovacích modelů jednalo o řádově sekundy a tak je nevýhodou především výše zmíněná nepřesnost, pro kterou je tento typ algoritmu při větších nárocích prakticky nepoužitelný.

8 Závěr

Po sérii provedených testů odhadů parametrů prostřednictvím vytvořené aplikace byla jako nejlepší optimalizační technika vyhodnocena metoda nejmenších čtverců. Naopak jednoznačně nejhorších výsledků dosáhl genetický algoritmus. Ovšem nelze jej odsoudit pouze na základě v rámci této klasifikační práce zjištěných poznatků. Lze jen říci, že při konkrétních zvolených implementacích optimalizačních algoritmů, jejich použitých konfiguracích a pro konkrétní modelované systémy je jeho využití vhodné nejméně.

Především právě práci genetického algoritmu lze ovlivnit mnoha způsoby provedení jeho dílčích úkonů, které jsou popsány v oddíle 5.2, přičemž jeho neúspěch ve zde provedeném srovnání s velkou pravděpodobností zapříčiňuje především nevhodně zvolené binární kódování. V případě použití kódování na reálná čísla by zřejmě bylo dosaženo větší přesnosti odhadů a také by se zkrátil čas potřebný k výpočtu, neboť by odpadla nutnost při každém ohodnocení jedince poměrně složitě převádět jeho geny z binární podoby do podoby reálných čísel.

I v případě zbývajících dvou lépe hodnocených metod by mohlo být dosaženo jejich zrychlení. U metody nejmenších čtverců je jedinou nevýhodou významnějšího rázu právě delší doba výpočtu v případě vyšších řádů modelu, která je způsobena náročným výpočtem inverzní matice kdy je nutné určit, například pro odhad parametrů modelu šestého řádu, matici maticeWT_W_I (viz pododdíl 6.3.1) o 144 prvcích, přičemž k získání každého tohoto prvku je třeba vypočíst determinant submatice o 121 prvcích. Při použití efektivnějšího způsobu výpočtu inverzní matice obecné velikosti by se tento typ optimalizace výrazně zrychlil.

V případě metody Nelder-Mead, která byla vyhodnocena jako velmi přesná, by určitého zrychlení mohlo být dosaženo odstraněním řazení všech vrcholů simplexu (viz pododdíl 6.3.3) v případě, že v předchozím kroku došlo k jeho reflexi, expanzi či kontrakci. Protože během dlouholetého studia této metody bylo zjištěno, že k redukci simplexu, kdy jsou nahrazeny všechny jeho vrcholy kromě nejlepšího, dochází poměrně málo a v ostatních případech, během kterých je nahrazen vždy pouze nejhorší vrchol simplexu, je třeba nový vrchol pouze zařadit na správné místo a tím je dosaženo seřazeného simplexu.

Literatura

[1] Systém. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 20. 9. 2004, last modified on 7. 2. 2011 [cit. 2011-05-08]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/Systém>.

[2] Proces. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 4. 2. 2005, last modified on 1. 4. 2011 [cit. 2011-05-08]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/Proces>.

[3] DRÁBEK, Oldřich; MACHÁČEK, Jiří. *Experimentální identifikace a řízení systémů*. VŠChT Pardubice : SNTL, 1983.

[4] DRÁBEK, Oldřich; MACHÁČEK, Jiří. *Experimentální identifikace*. Pardubice : VŠChT, 1987. 275 s.

[5] Časová řada. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 30. 4. 2008, last modified on 1. 5. 2011 [cit. 2011-05-08]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/Časová_řada>.

[6] Box jenkins. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 3. 10. 2008, last modified on 15. 4. 2011 [cit. 2011-05-08]. Dostupné z WWW: <http://en.wikipedia.org/wiki/Box_jenkins>.

[7] KRIŠTOF, Aleš. *Nové metody a přístupy k analýze a prognóze časových řad*. Praha, 2006. 113 s. Dizertační práce. Česká zemědělská univerzita v Praze.

[8] ARTL, Josef. *Moderní metody modelování ekonomických časových řad*. Praha : Grada Publishing, 1999. 312 s.

[9] CHATFIELD, Chris. *The Analysis of Time Series*. New York : Chapman & Hall, 1996.

[10] Metoda nejmenších čtverců. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 29. 6. 2006, last modified on 22. 4. 2011 [cit. 2011-05-08]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/Metoda_nejmenších_čtverců>.

[11] PAIGE, Christopher; STRAKOŠ, Zdeněk. *Scaled Total Least Squares Fundamentals*, Numerische Mathematik. 2002, 91, s. 117-146. Dostupný také z WWW: <http://www.cs.mcgill.ca/~chris/pub/PaiS02b.pdf>.

[12] Genetický algoritmus. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 6. 4. 2005, last modified on 22. 2. 2011 [cit. 2011-05-08]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/Genetický_algoritmus>.

[13] ZELINKA, Ivan. *Umělá inteligence v problémech globální optimalizace*. Praha : BEN-Technická literatura, 2002. 193 s. ISBN 80-7300-069-5.

[14] HYNEK , Josef. *Genetické algoritmy a genetické programování* . 1. Praha : Grada Publishing, 2008. 182 s. ISBN 978-80-247-2695-3.

[15] Nelder-Mead method. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 9. 12. 2009, last modified on 31. 1. 2011 [cit. 2011-05-08]. Dostupné z WWW: <http://en.wikipedia.org/wiki/Nelder-Mead_method>.

[16] WRIGHT, Margaret. *Direct Search Methods: Once Scorned, Now Respectable*. Numerical Analysis, 1995. s. 191–208.

[17] LAGARIAS, Jeffrey; REEDS, James; WRIGHT, Margaret; WRIGHT, Paul. *Convergence Properties of the Nelder-Mead Simplex Method in Low Dimensions*. SIAM Journal on Optimization. 1998, 1998, 9, s. 112-147. Dostupný také z WWW: <http://epubs.siam.org/sam-bin/dbq/article/30347>.

Příloha A – Metoda provedAlgoritmus třídy Nelder_Mead_algoritmus

```
public double[] provedAlgoritmus(int pocetCyklu) {
      cyklus = 0;
      while (cyklus < pocetCyklu && !simplex.jsouSiRovny()) {
            cyklus++;
            simplex.seradVrcholy();
            simplex.najdiTeziste();

            double f_nejlepsiho, f_druh_nejhorsiho, f_nejhorsiho,
            f_bodu_xr, f_bodu_xe, f_bodu_xc;
            f_nejlepsiho = funkce.f(simplex.dejHodnotyVrcholu(0));
            f_druh_nejhorsiho =
            funkce.f(simplex.dejHodnotyVrcholu(pocetVrcholu - 2));
            f_bodu_xr = funkce.f(simplex.dejHodnotyBoduXR());
// Reflexe
            (f_nejlepsiho <= f_bodu_xr && f_bodu_xr < f_druh_nejhorsiho)
            {
                  simplex.reflexe();
                  continue;
            }
// Expanze
            if (f_bodu_xr < f_nejlepsiho) {
                  f_bodu_xe = funkce.f(simplex.dejHodnotyBoduXE());
                  if (f_bodu_xe < f_bodu_xr) {
                        simplex.expanze();
                        continue;
                  } else {
                        simplex.reflexe();
                        continue;
                  }
            }
// Kontrakce
            f_bodu_xc = funkce.f(simplex.dejHodnotyBoduXC());
            f_nejhorsiho =
            funkce.f(simplex.dejHodnotyVrcholu(pocetVrcholu - 1));
            if (f_bodu_xc < f_nejhorsiho) {
                  simplex.kontrakce();
                  continue;
            } else {
// Redukce
            simplex.redukce();
            continue;
            }

      }
      simplex.seradVrcholy();
      return simplex.dejHodnotyVrcholu(0);
}
```