БЪЛГАРСКА АКАДЕМИЯ НА НАУКИТЕ
Институт по информационни и комуникационни технологии

Д И С Е Р Т А Ц И Я

на тема:

“Невронно-размити модели за целите на предсказващото управление”

за присъждане на образователна и научна степен „Доктор”
на маг. инж. Маргарита Николаева Терзийска
по научна специалност:
01.01.12 „Информатика”, професионално направление 4.6
„Информатика и компютърни науки”

научни ръководители:
доц. д-р Любка Дуковска
проф. д-р Михаил Петров

София, 2015 г.

Съдържание

УВОД

Невронните мрежи, размитата логика и хибридните невронно-размити структури са в основата на системите с изкуствен интелект. Поотделно и в комбинация тези техники са известни като универсални апроксиматори, поради което са широко използвани за моделиране и идентификация на нелинейни обекти. Това ги прави подходящи за включване в предсказващите регулатори в ролята на предсказващ модел. Последните са съществена и много важна част от предсказващите регулатори, тъй като от точността им на предсказване зависи качеството на управление. Друга основна част на предсказващите регулатори е оптимизаторът, който определя стойността на управлението, решавайки оптимизационна задача (често нелинейна). Определянето на бъдещия изход на обекта (т.е. осъществяването на структурна и параметрична идентификация) и на оптималната стойност на управлението са две изчислително тежки процедури, с които предсказващият регулатор трябва да се справи в реално време в рамките на един такт на дискретизация. Това е основната причина, поради която този вид регулатори се използват главно за бавни процеси. За да се преодолее този недостатък на предсказващите регулатори е необходимо да бъдат разработени начини за облекчаване на изчислителната процедура при моделирането и/или при оптимизацията. С оглед на това, настоящият дисертационен труд е насочен към проектиране на алгоритми за нелинейни предсказващи регулатори, базирани на невронно-размити модели, които са с редуциран брой размити правила и съответно имат малък брой параметри за настройка. В дисертационната работа са представени *Разпределен невронно-размит модел (DANFA - Distributed Adaptive Neuro Fuzzy Architecture)*, *Невронно-размит модел с частично размиване на входните сигнали (SFNN - Semi Fuzzy Neural Network)* и *Модифициран Нео-размит модел (MNFM - Modified Neo-fuzzy model)*. За да могат тези модели да се справят успешно в условията на неопределености те са реализирани във вариант с Тип 2 размита логика (Type-2 Fuzzy Logic) и във вариант с Интуиционистка размита логика (Intuitionistic Fuzzy Logic). За обучение на посочените по-горе невронно-размити структури са използвани градиентен алгоритъм от първи ред (двустъпков алгоритъм с обратно разпространение на грешката),

рекурентен метод на най-малките квадрати и градиентни алгоритми от втори ред.

За изчисляване на оптималната стойност на управлението са реализирани алгоритъм на Нютон и алгоритъм на Левенберг-Маркгуард, които осигуряват бързодействие на предсказващия регулатор и не допускат засядане в локален екстремум. Най-съществената изчислителна трудност при тези алгоритми е определянето на обратната матрица на Хесе (Hesse), което е преодоляно чрез използване на т.нар. LU декомпозиция.

Представен е и алгоритъм за супервайзорна адаптивна донастройка на невронно-размит обобщен предсказващ регулатор. Алгоритмите, описани в дисертацията, са реализирани в Matlab/Simulink® и са тествани както в симулационна среда, така и в реални условия. За целта предложените алгоритми за невронно-размити предсказващи регулатори са използвани за управление на химически реактор и лабораторна топлинна система.

Дисертационната работа е структурирана по следния начин: В **Глава I** е дадена постановката на моделното предсказващо управление (МПУ) и са представени по-известните предсказващи алгоритми. Направен е обзор на използваните в нелинейното предсказващо управление невронни, размити и невронно-размити модели, въз основа на който са формирани целта и задачите на дисертационния труд. В **Глава II** е представена теоретичната постановка на невронно-размит обобщен предсказващ регулатор. Предложените алгоритми са детайлно описани в **Глава III**, а резултатите от симулационните и реални експерименти са представени в **Глава IV**. Структурата и характерните особености на разработения програмен код са дискутирани в **Глава V.** В **Глава VI** са обобщени постигнатите резултати и са резюмирани направените изводи.

Основни съкращения

ANFIS	Adaptive Neuro Fuzzy Inference System	
ARIMAX	AutoRegressive Integrated Moving Average with eXogenous Input	Авторегресия интегрираща пълзяща средна стойност с външна променлива
ARMA	Autoregressive–moving-average	Авторегресия-пълзяща средна стойност
ARMAX	AutoRegressive Moving Average with eXogenous Input	Авторегресия пълзяща средна стойност с външна променлива
ARX	AutoRegressive eXogenous Input	Авторегресия с външна променлива
DANFA	Distributed Adaptive Neuro Fuzzy Architecture	Разпределен невронно-размит модел
DANFAGPC	Distributed Adaptive Neuro Fuzzy Architecture Generalized Predictive Controller	Обобщен предсказващ регулатор с разпределен невронно-размит модел
DMC	Dynamic Matrix Control	Динамично матрично управление
GPC	Generalized Predictive Control	Обобщено предсказващо управление
HIECON	Hierarhical constraint control	Йерархично управление с ограничения
IDCOM-M	Identification and Command – Multivariable	
LQR	Linear Quadratic Regulator	Линейно-квадратичен регулатор
MBPC	Model based predictive control	Моделно базирано предсказващо управление
MIMO	Multi Input Multi Output	Многомерна система

MISO	Multi Input Single Output	Система с много входове и един изход
MNFM	Modified Neo-fuzzy model	Модифициран нео-размит модел
MNFMGPC	Modified Neo-fuzzy model Generalized Predictive Controller	Обобщен предсказващ регулатор с модифициран нео-размит модел
MPHC	Model Predictive Heuristic Control	Моделно предсказващо евристично управление
MSE	Mean Squared Error	Средноквадратична грешка
MVC	Multivariable Control	Многомерно управление
NNFARX	Nonlinear neuro-fuzzy ARX	Нелинейна невронно-размита авторегресия с външна променлива
NNFIR	Neural Network Finite Impulse Response	
PFC	Predictive Functional Control	Предсказващо функционално управление
QDMC	Quadratic Dynamic Matrix Control	Динамично матрично управление с квадратично програмиране
RBF	Radial Basis Function	Радиално базисна невронна мрежа
RMPCT	Robust model predictive control technology	Робастно моделно предсказващо управление
RMSE	Root Mean Squared Error	Коренувана среднДоквадратична грешка
SFNN	Semi Fuzzy Neural Network	Невронно-размит модел с частично размиване на входните сигнали
SFNNGPC	Semi Fuzzy Neural Network Generalized Predictive Controller	Обобщен предсказващ регулатор с полу невронно-размит модел

SISO	Single Input Single Output	Едномерна система
SMOC	Shell Multivariable Optimizing Controller	
ЛМПУ	(Linear) Model Predictive Control	Линейно моделно предсказващо управление
МПУ	Model Predictive Control	Моделно предсказващо управление
НМПУ	Nonlinear Model Predictive Control	Нелинейно моделно предсказващо управление

Използвани означения

$y(k)$	Изход на обекта за управление
$\hat{y}(k+N_p)$	Предсказан изход на обекта за управление
$u(k)$	Управляващ сигнал
$\Delta u(k)$	Нарастък на управляващия сигнал
$r(k)$	Задание
m	Брой на използваните функции на принадлежност
p	Брой входове на невронно-размит модел
N	Брой размити правила
N_1	Минимален хоризонт на предсказване
N_u	Хоризонт на управление
N_p	Хоризонт на предсказване
$J(.)$	Целеви критерий
λ	Тегловен коефициент в целевия критерий
η	Скорост на обучение
I	Единична матрица

“Ако не можеш да обясниш нещо на 6-годишно дете,

значи самия ти не го разбираш.”

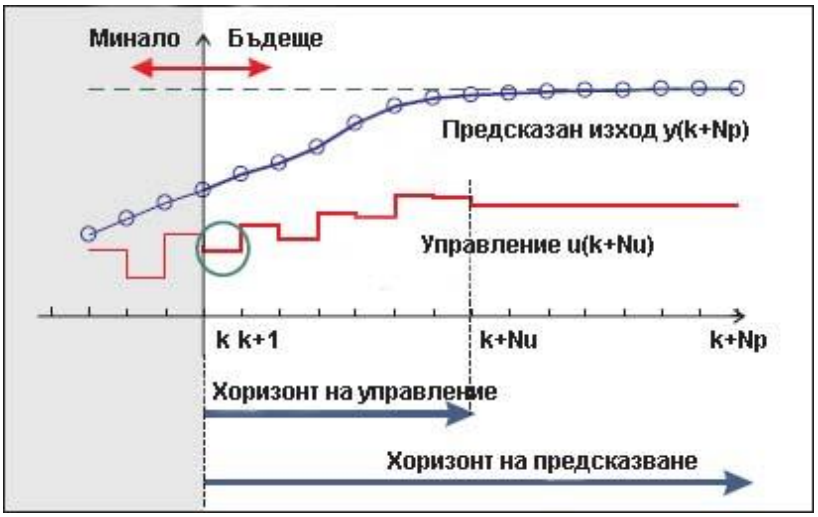
Алберт Айнщайн

Глава I

Съвременни алгоритми за нелинейно моделно предсказващо управление

1.1. Същност на Моделното предсказващо управление

Моделното предсказващо управление (МПУ), наричано още управление с плаващ или управление с удължен хоризонт, започва да се развива през 70-те години на миналия век и днес вече се е превърнало в класически метод за управление. Като основен мотив за развитието на предсказващото управление в производствените и научни среди може да се изтъкне възможността във вече проектирания регулатор да се инкорпорират ограничения на управляваните променливи. Друго основно предимство на предсказващите регулатори е възможността им да се справят с многомерни обекти с неизвестно или променливо времезакъснение, а също и да осигурят устойчива работа на неминималнофазови обекти за управление. Не без значение за успеха на предсказващото управление е и ясната процедура по проектиране на предсказващия регулатор. Основната идея на МПУ е представена на фиг.1.1.1.

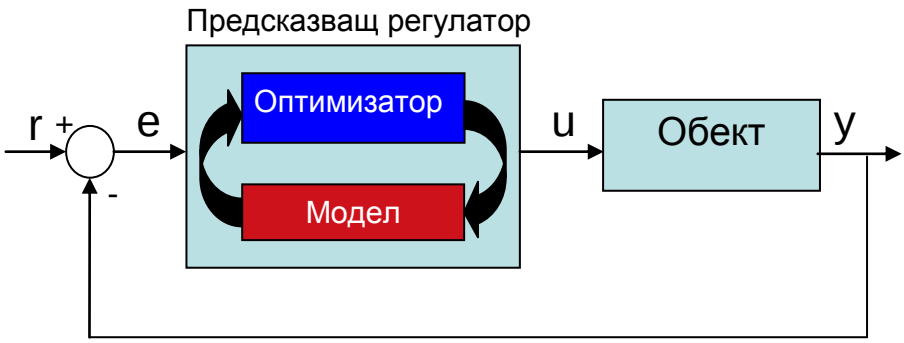


1.1.1. Същност на МПУ

При предсказващите регулатори въз основа на вграден модел, описващ динамиката на обекта, се предсказват бъдещите стойности на управляваните променливи в рамките на краен интервал от време, наречен хоризонт на предсказване N_p. Предсказаните стойности са функции на бъдещите управляващи променливи $\hat{u}(k+j)$, където $k = 0, 1, ..., , N_u–1$ и на текущото състояние на системата. С N_u е означен хоризонта на управление. Използвайки тези предсказани стойности се формира функционал, представляващ най-често квадратична разлика между грешката (разлика между действителната и предсказаната стойност на управляваната величина) и управляващото въздействие. Чрез минимизирането на този функционал се получава последователност от оптимални стойности на управлението. Към обекта на управление се прилага само първата от тези стойности $\hat{u}(k)$. В следващия период от време k+1 този процес се повтаря отново, но вече хоризонтът на предсказване е изместен с една стъпка напред. Това е причината, поради която този начин на управление често се нарича отдалечен хоризонт на управление.

МПУ може да се отнесе към адаптивните системи за управление и по-специално към класа на самонастройващите се регулатори. В литературата се среща още под наименованието моделно базирано предсказващо управление (МБПУ). То представлява клас от компютърни алгоритми за управление, които имат еднакви особености, а именно: употребата на вътрешен модел, чрез който се прави оценка на бъдещото състояние на системата за предварително определен краен интервал от време, наречен хоризонт на предсказване N_p; изчисляване на поредица от стойности на управлението, в рамките на хоризонта на управление, минимизирайки целева функция; и прилагане само на първата стойност от тази поредица, т.е. прилагане на удължен (отдалечен) хоризонт на предсказване.

На фиг. 1.1.2 е представена система за управление с предсказващ регулатор. Както се вижда от нея и в съответствие с гореизложеното предсказващият регулатор може да се представи като съвкупност от два взаимносвързани блока – предсказващ модел (предиктор) и оптимизатор.



Фиг 1.1.2 Блокова схема на система за управление с предсказващ регулатор

Съществуват редица модификации на предсказващите регулатори. Всяка една от тях има редица особености, с които се отличава от другите, например: по използвания модел; алгоритъма за оценка на параметрите в модела; алгоритъма за решаване на оптимизационната задача във всеки такт и т.н. Иначе казано, всички те се различават именно по структурата и алгоритмите, вложени в двата блока от фиг. 1.1.2 – модел и оптимизатор. Някои от по-известните модификации на предсказващите регулатори са:

- CGPC (Continuous-time Predictive Control);
- CRHPC (Constrained Receding Horizon Predictive Control);
- DMC (Dynamic Matrix Control);
- GPC (Generalized Predictive Control);
- GPCW (Generalized Predictive Control with end-point state weighting);
- MAC (Model Algorithmic Control);
- PFC (Predictive Functional Control);
- UPC (Unified Predictive Control) и др.

1.2. Нелинейно моделно предсказващо управление

Според това какъв модел използва МПУ се разделя на два големи класа – линейно МПУ (ЛМПУ) и нелинейно МПУ (НМПУ). Като цяло разликата между тях е малка (НМПУ използва нелинеен модел за предсказване), обаче в изчислителен и теоретичен план ЛМПУ е много по-елементарна задача от НМПУ, също както и линейната теория на управлението в сравнение с нелинейната.

В исторически план първо възниква ЛМПУ, поради което огромна част от приложенията на предсказващото управление са базирани на линеен модел. В същото време, обаче, по-голяма част от процесите са нелинейни и това става причина в края на 90-те години на миналия век да се развие НМПУ.

С НМПУ се означава управление на нелинейни обекти с използване на алгоритъм за МПУ, в който по подходящ начин са взети под внимание нелинейностите в обекта. По принцип НМПУ е аналогично на ЛМПУ, и характеристиките на системата за управление са същите. Следователно НМПУ има същите характеристики като ЛМПУ – използване на вътрешен модел; определяне на поредица от оптимални стойности на управлението посредством минимизиране на целеви критерий; прилагане на стратегията на отдалечения (плаващ) хоризонт. Случаите, в които се прибягва до използване на НМПУ са:

- при процеси със съществени нелинейности и подложени на чести и значителни смущения (например управление на pH);
- при процеси, където се наблюдава честа промяна на работната област и е необходимо да се обхване една сравнително широка област от нелинейната динамика на процеса.

Въведение в теорията на НМПУ може да бъде намерено в труда на (Findersen et al., 2002). В него е дадена математическата формулировка на НМПУ, представени са основните принципи и свойства на системите с предсказващо управление и са очертани ключовите му предимства и недостатъци. Редица теоретични и изчислителни аспекти на НМПУ са дискутирани в (Allgöwer et al., 2004). Тези автори също така посочват особеностите в практическото приложение на нелинейните предсказващи алгоритми, както и нуждата от надеждни и бързи методи за идентификация и оптимизация в реално време. Обзорни статии, посветени в по-голямата си част на ЛМПУ са тези на (Morari and Lee, 1999) и (Rawlings, 2000). В тях авторите коментират също практическата необходимост от развитие на алгоритми за НМПУ, техните теоретични и практически основи и дават някои насоки за развитието им. Изцяло посветени на НМПУ са статиите (Camacho, 2007; Magni, 2010), както и книгите (Kouvaritakis, 2001; Findeisen, 2007; Grune, 2011). В своята обзорна статия (Lee, 2011) прави обобщение на развитието на системите с предсказващ регулатор през последните три десетилетия. Според автора, през първите десет години след анонсирането на МПУ е наблюдавано едно широко възприемане на

идеята за предсказващото управление от индустрията и по-специално използването му в нефтени рафинерии и химически заводи. През второто десетилетие – е постигнат съществен напредък в теоретичен аспект, което включва разработване и интерпретиране на модели в пространство на състоянието, формулиране на критерии и доказване на устойчивост на системите с предсказващ регулатор. През третото десетилетие акцентът пада върху проектирането и разработването на т.нар. бързи МПУ алгоритми.

В последните години значително развитие получиха разпределените и йерархични системи с НМПУ. Те са подходящи за управление на сложни технологични обекти и дори на цели заводи. Различни йерархични структури с МПУ са представени в (Brdys et al., 2008) и (Falcone et al., 2008). Обзор на различните архитектури децентрализирани, разпределени и йерархични системи, използвани в системите с МПУ е представен в (Scattolini, 2009).

1.3. Нелинейни предсказващи модели

Основните изисквания при моделирането за целите на НМПУ, както и обзор на методите за моделиране и идентификация може да бъде намерен в (Lee, 2000). Трудът на (Pearson, 2003) представлява своеобразно ръководство за избора на подходяща структура на нелинеен модел. В него авторът описва някои основни класове нелинейни структури, класифицира ги и дава отговор на въпроса какво представлява "добрият" модел.

Според (Waller, 2000) използваните в НМПУ модели могат да се разделят на три групи. При първата група моделите се основават на фундаментални взаимовръзки, при втората моделите се базират на емпирични данни, а третият вид модели представлява комбинация от първите два и се наричат хибридни. Поради сложността на нелинейните обекти не е възможно разработването на методи за идентификацията им чрез директно разширяване на линейните методи. Като алтернатива, регулаторът при НМПУ може да бъде получен от модел базиран на фундаментални взаимовръзки. Основно предимство на тези модели е, че те са глобално валидни и следователно се очаква моделът да екстраполира към работните области, които не са представени в множеството данни,

използвани при разработването на модела. Съществена трудност е определянето на структурата на самия модел. Освен това полученият динамичен модел е твърде сложен (от висок ред), за да бъде полезен при проектирането на НМПУ.

Що се касае до емпиричните нелинейни модели, едно решение, което трябва да се направи е изборът на подходяща структура на модела. Като основен недостатък трябва да се подчертае, че валидността на тези модели е ограничена и зависи от входно-изходните данни, използвани при разработването им. Понастоящем за целите на НМПУ се използват предимно:

- Модели на Волтера;
- Модели на Хамерщайн и Винер;
- Невронни мрежи;
- Модели, базирани върху размитата логика;
- Хибридни модели (комбинация от изброените по-горе).

Невронните мрежи, размитатата логика и хибридните невронно-размити структури са известни като универсални апроксиматори, поради което са широко използвани за моделиране и идентификация на нелинейни обекти. За целите на предсказващото управление те се използват както поотделно, така и в комбинация с моделите на Волтера и тези на Винер и Хамерщайн. Пример за това е невронно-размития модел на Волтера, представен в (Todorov et al., 2010). Моделът е вграден в нелинеен предсказващ регулатор, използван за управление на технологичния процес лиофилизация. Голяма част от публикациите, посветени на използването на модела на Хамерщайн, докладват за използването му в комбинация със софт компютинг техники. В своята статия (Abonyi et al., 2000) предлагат подобна блокова структура като реализират нелинейната статична част с помощта на размита логика. За целта те описват поведението на системата в установен режим с размит модел на Такаги-Сугено от нулев ред. В статията си (Ngia et al., 1998) изгражда модел на Хамерщайн като комбинация от невронна мрежа с прави връзки с един скрит слой и линейна импулсна характеристика. На практика определянето на параметрите в импулсната характеристика е затруднена поради големия им брой. Комбинирайки свойствата на размитата логика и невронните мрежи, (Jia et al. 2004) предлагат невронно-размит модел на Хамерщайн. За целта авторите апроксимират нелинейния елемент с невронно-размит модел на Такаги-Сугено, базиран на радиално базисна невронна мрежа (RBF), а

линейната част е под формата на авторегресионен модел. В статията си (Hachino et al., 2004) също използват RBF мрежа като нелинейна статична част, но в допълнение предлага определянето на структурата на невронната мрежа (броя на Гаусовите функции, тяхната ширина и център) да става посредством генетични алгоритми.

Моделите на Винер се използват по-рядко за целите на НМПУ, като това е свързано както с избора на статична нелинейност, която трябва да може да се инвертира, така и с усложнената идентификационна процедура. Предсказващ алгоритъм, при който моделът на Винер е съставен от линейна импулсна характеристика и Такаги-Сугено размит модел от нулев ред, е предложен от (Abonyi et al., 1999). Моделът на Винер, описан в (Azhar et al.p 2002), представлява комбинация от ARMA модел и RBF невронна мрежа, а неговата параметрична идентификация е направена по метода на най-малките квадрати. Изчислително ефективен предсказващ регулатор, базиран на невронен Винер модел е представен в (Lawrynczuk, 2010).

1.3.1. Невронни предсказващи модели

Идеята за невронните мрежи е взаимствана от биологичните невронни мрежи и аналогично на тях те притежават следните особености:

- възможност за обучение – със или без учител;
- висока надеждност – при повреда на един или няколко неврона невронната мрежа продължава да работи ефективно;
- висока скорост на обработка на информацията.

А/ Видове невронни предсказващи модели

Изброените характеристики са причина невронните мрежи да намерят широко практическо приложение в системите за управление. Те се използват както за разработване на невронни регулатори, така и за моделиране и идентификация и са известни като универсални апроксиматори. Невронните мрежи са подходящи за моделиране на нелинейни, многомерни статични и динамични системи чрез метода на “черната кутия”. В (Nørgaard et al., 2000) задълбочено е описано изграждането на нелинейни модели с различни структури (NNFIR, NNARX, NNARMAX) с помощта на невронни мрежи. Както вече бе

споменато от съществено значение при предсказващото управление е точността на модела, т.е наличието на минимална грешка между изхода на технологичния процес и модела. При моделирането с невронни мрежи точността се свързва не толкова с използвания тип невронна мрежа, колкото с нейната структура (брой скрити слоеве, брой неврони в скритите слоеве). Като правило по-големият брой скрити слоеве и неврони в тях води до по-точно моделиране, но не съществува единна методология за определянето на структурата им. Обобщение на употребата на невронните мрежи в предсказващото управление е направено в (Lawrynczuk, 2009).

Моделирането на нелинейни технологични процеси в системите с НМПУ може да се реализира с помощта на различни по структура невронни мрежи, но най-често използваните са тези с прави връзки. Предсказващи алгоритми, базирани на невронни мрежи с прави връзки, са представени в (Hosen et al., 2011; Kittisupakom et al., 2009; Hajimolana et al., 2012; Yousef, 2012; Sendrescu et al., 2011; Vasičkaninová et al., 2011; Lawrynczuk et al., 2010; Hedjar, 2013). Обобщени предсказващи регулатори (Generalized Predictive Control - GPC), базирани на невронни мрежи с прави връзки, са докладвани в (Chidrawar, 2008; Niu and Li, 2013; Vasičkaninová, 2009). Две невронни мрежи с прави връзки са в основата на предсказващия регулатор NN-MPC (Neural Network-Model Predictive Control), предложен от (Mjalli, 2005). Едната представлява предсказващ модел, а другата се използва при оптимизацията на квадратичен целеви критерий. В статията си (Zamarreño et al., 1999) изграждат модела в предсказващия регулатор NPC (Neural Predictive Controller) като комбинация от описание в пространство на състоянието и невронна мрежа с прави връзки. Подобна технология използва и фирмата Aspen Tech при производството на регулаторите от серията Aspen Target.

Наред с тях, за целите на НМПУ, се използват също и радиално базисни невронни мрежи (Radial Basis Function - RBF). Те по същество са невронни мрежи с прави връзки, при които в ролята на активиращи функции са използвани Гаусови такива. Подобни предсказващи регулатори са описани в (Wang et al., 2006; Peng et al., 2004; Venkateswarlu et al., 2005; Samek et al., 2009).

Друг вид мрежи с прави връзки, които намират все по-голямо приложение са тези с авто-асоциативна памет (Cerebellar Model Arithmetic Computer - CMAC). Основната идея на този вид мрежи е да съхраняват

вече „научените" данни по такъв начин, че те да могат да бъдат извикани лесно за нова употреба. Това им действие наподобява работата на малкия мозък при човека. Основните характеристики на CMAC мрежите, които привличат учените, са възможността им за бързо обучение, добрата им обобщаваща способност и лесната хардуерна реализация. CMAC мрежите обаче са недостатъчно надеждни при апроксимацията на функции, поради което често се използват в комбинация с размита логика (Fuzzy CMAC - FCMAC). Идентификацията на нелинейни системи с помощта на FCMAC е дискутирана в (Lin and Peng, 2012). В (Lin et al., 2008) е представена параметрична FCMAC мрежа, която съчетава стандартна CMAC мрежа с размита система с механизъм на извеждане на Такаги-Сугено. Обзор на съществуващите FCMAC може да бъде намерен в (Mohajeri et al., 2009). Друг съществен недостатък на CMAC мрежите е голямата памет, която им е необходима за изчисление и съхранение на данни. Решението на този проблем се търси в изграждането на йерархични (Rodriguez et al., 2006) и самоорганизиращи се CMAC мрежи (Nguyen et al., 2006). Въпреки, че са намерени начини за преодоляване на недостатъците им, все още са малко представените в литературата предсказващи алгоритми, базирани на CMAC мрежите (Tang and Yan, 2007).

Друго актуално направление при моделирането на нелинейни процеси за целите на предсказващото управление е използването на wavelet функции в невронните мрежи, т.е. изграждане на wavelet невронни мрежи (Wavelet Neural Networks). Основни предимства на този тип мрежи са, че позволяват да се намали броя на скритите слоеве и невроните в тях, не допускат засядане в локален минимум и увеличават скоростта на сходимост. В литературата се срещат редица предсказващи алгоритми, базирани на wavelet невронни мрежи (Gu and Hu, 2000; Kim et al., 2005; Abdi et al., 2011; Xia et al., 2005). Тези мрежи често се комбинират с размита логика (Fuzzy Wavelet Neural Networks). Това позволява да се намали сложността на обработваните данни и да се работи в присъствието на неопределености. Размитите wavelet мрежи намират широко приложение – от предсказване на цените на стоковата борса (Abiyev, 2012) до идентификация на нелинейни системи (Esmaili et al., 2013; de Araújo Júnior et al., 2012) и управление на процеси (Zekri et al., 2008; Abiyev and Kaynak, 2008).

През последното десетилетие бурно развитие претърпяха рекурентните невронни мрежи. В тези мрежи, освен прави връзки се включват и обратни връзки (локални и/или глобални). Тези обратни връзки играят ролята на ”памет” и позволяват на модела по-добре да ”улови” динамиката на процеса. Ето защо се е наложило мнението, че рекурентните мрежи са по-добри апроксиматори от мрежите с прави връзки. В литературата са предложени голям брой различни структури на рекурентни невронни мрежи (Li, 2002; Patan, 2012; Rankovic et al., 2012). Предсказващо управление базирано на диференциална рекурентна невронна мрежа е предложено в (Seyab and Cao, 2008). Две невронни мрежи с различни структури са използвани при изграждането на предсказващия алгоритъм, представен в (Wang et al., 2006). Предсказващият модел е реализиран посредством диагонална рекурентна невронна мрежа, която представлява опростена форма на динамичната мрежа на Елман. При обучението ѝ е използвана модификация на метода с обратно разпространение на грешката, при която се работи не с текущата стойност на моделната грешка, а със сумата от квадрата на моделните грешки в рамките на хоризонта на предсказване. За оптимизацията е използвана трислойна невронна мрежа с прави връзки. В рекурентна форма се използват CMAC и уейвлет мрежите (Ortiz et al., 2007; Yoo et al., 2005; Kulkarni and Kumar, 2012).

Описаните дотук различни архитектури на невронни мрежи (MLP, RBF, RNN) в действителност нито по структура, нито като функция наподобяват биологичните неврони (Johnson et al., 2009). При тях в определен момент постъпват входните сигнали, всеки неврон се активира само по веднъж и след това се формират изходите, т.е. мрежата работи в дискретни тактове от време. Биологичните неврони работят непрекъснато – те получават на входа си последователност от напреженови пикове, наречени “потенциали на действие”. Ето защо започва използването т.нар. spiking невронни мрежи (Spiking Neural Networks - SNN) - биологично инспирирани невронни мрежи от трето поколение. SNN са специален клас невронни мрежи, при които невроните комуникират помежду си с последователност от напреженови пикове. Мрежите, изградени от spiking неврони могат да обработват значително количество информация, като използват относително малък брой пикове (Van Rullen et al., 2005). Поради тяхното функционално подобие с биологичните неврони, spiking моделите представляват мощен инструмент за анализ на елементарните процеси в

мозъка, включително обработка на невронна информация и обучение. В същото време SNN предлагат решение за широк набор от инженерни проблеми, като например обработка на изображения (Dolotov, 2009), класификация (Glackin et al., 2008; Mishra et al., 2006), разпознаване на говор, апроксимация (Mishra et al., 2006), идентификация и контрол (Johnson et al., 2009; Abiyev et al., 2012). SNN се използват и в комбинация с размита логика (Reid, 2008). В литературата са предложени също и еволюционни SNN (Schliebs and Kasabov, 2012; Kasabov et al., 2013). Обширна статия, посветена на SNN е представена от (Ponulak and Kasiński, 2011). В нея авторите акцентират върху свойствата на SNN, описват начина, по който спайкинг невроните обработват информация, методите за обучението им, а също и областите, в които намират приложение.

Б/ Обучение на невронни предсказващи модели

Характерна особеност на невронните мрежи е възможността им за обучение. Това е процес, свързан с адаптивната настройка на теглата на връзките и представлява нелинейна оптимизационна задача. Най-общо използваните методи за обучение могат да се класифицират като градиентни алгоритми от I-ви и II-ри ред. Към първата група се отнася алгоритъмът с обратно разпространение на грешката (backpropagation), както и неговите модификации (с въвеждане на моментум, с адаптивна скорост на обучение и др.), а втората група включва метода на Нютон, квази-Нютонови алгоритми, метод на Левенберг-Маркгуард. Основни изисквания към алгоритмите за обучение, както и кратък коментар относно градиентните алгоритми от I-ви и II-ри ред могат да бъдат намерени в Sørensen et. al (1999).

Като обучаващи алгоритми се използват също така и биологично инспирираните алгоритми за глобална оптимизация. Наред с вече утвърдилите се генетични алгоритми в последно време се използват също алгоритми за оптимизация по метода на мравките (Ant Colony Algorithm), по метода на пчелите (Artificial Bee Colony algorithm) и роящите (Particle Swarm Optimization). Характерно за тази група алгоритми е, че при тях не се изисква изчисляване на производни, но пък са значително по-бавни от градиентните методи, което не ги прави подходящи за целите на предсказващото управление.

1.3.2. Размити предсказващи модели

В предсказващото управление се използват също и размити модели. Това са математически модели, в които са използвани размити множества. В редица случаи размитите модели се предпочитат пред други нелинейни модели, тъй като се проектират лесно, имат сравнително опростена структура и в тях лесно може да се вгради човешка логика. Съществена трудност при изграждането им е определянето на размитите правила и функциите на принадлежност. Според (Peneva, V., and I. Popchev, 2009) е подходящо използването на размити модели в условията неточност, несигурност и частична истинност. Основните характеристики на моделирането с помощта на размита логика е обобщено в (Angelova, V., 2009). Изследване на различни размити предсказващи модели е направено от (Liutkevičius et al., 2005). Авторите сравняват модел на Мамдани и Такаги-Сугено модел. Според тях моделът на Мамдани е по-точен, но съдържа по-голям брой параметри, които трябва да бъдат определени при идентификацията. За да елиминират този недостатък предлагат изграждането на хибриден размит модел, който представлява комбинация от модел на Мамдани и модел на Такаги-Сугено.

За целите на предсказващото управление по-широко се използват моделите с механизъм на извеждане на Такаги-Сугено. Характерно за тях е, че в следствената част на правилата изходната величина се дефинира с конкретна функция (линейна или нелинейна), чийто аргументи са входните променливи. Модел на Такаги-Сугено е в основата на размития предсказващ алгоритъм предложен от (Sarimveis et al., 2003). В своя труд (Mollov, 2002) предлага управление на многомерни процеси с използването на предсказващ регулатор базиран на модела на Такаги-Сугено. Също така авторът прави задълбочено изследване на устойчивостта и робастността на системи с размито предсказващо управление. В предложения от (Abonyi et al., 2001) алгоритъм в ролята на предсказващ модел е използван модел на Такаги-Сугено от нулев ред, наречен още полу-размит модел. Функционално предсказващо управление (PFC), базирано на Такаги-Сугено размит модел от трети ред е предложено от (Lepetič et al., 2003). За идентификацията на предсказващия модел авторите използват метода на най-малките квадрати (МНК). В статията си (Huang et al., 2000) разглеждат нелинейният процес като съвкупност от множество квази-линейни модели, описани посредством размити правила на Такаги-Сугено. Последните от

своя страна са генерирани от импулсната характеристика на обекта. Този тип модели са известни като Fuzzy Convolution Model. Подобен Fuzzy Convolution Model е използван и при изграждането на предсказващия регулатор, представен от (Chakrabarty, 2013). За определянето на тегловните матрици в модела авторите използват безградиентен глобален стохастичен алгоритъм – Bacterial Foraging Optimization (BFO), който е биологично инспириран и имитира растежа на бактериалните колонии.

Размитото моделиране и проектирането на нелинейни регулатори на базата на размити модели е дискутирано в (Babuska, 1998). В (Tuan, 2012) е представен робастен размит предсказващ регулатор с Такаги-Сугено механизъм на извеждане, който е подходящ за неминималнофазови обекти. Нова методология за автоматично извличане на Такаги-Сугено размит модел за предсказващ регулатор е описана в (Su, 2012). За целта е използван Artificial Bee Colony algorithm в комбинация с Fuzzy C-Mean клъстеринг техника. Размит предсказващ регулатор, който работи в присъствието на ограничения и на смущения, е разработен от (Bououden et al., 2009). Размит предсказващ R2R регулатор е представен в (Wang, 2011). В литературата са описани редица размити GPC регулатори в едномерен (Mendes, 2012) и многомерен (Huaguang, 2002; Li 2010) вариант. Размит GPC регулатор в комбинация с йерархичен генетичен алгоритъм е представен в (Mendes, 2013).

Описаните дотук размити модели са изградени въз основа на т.нар. обикновени (Тип 1) размити множества (РМ) – Гаусови, триъгълни, трапецовидни. През последното десетилетие много популярни станаха РМ Тип 2, по-конкретно интервалните множества Тип 2. Първоначално идеята за РМ Тип 2 е предложена от "бащата" на размитата логика (Zadeh, 1975), а впоследствие цялостната теория на РМ Тип 2 е разработена от (Karnik and Mendel, 1998). Основното предимство на РМ Тип 2 е способността им да се справят при наличие на шум и неопределености в информацията, постъпваща в размитите системи. Това ги прави все по-предпочитани за редица приложения, като например за моделиране (Ren et al., 2006) и предсказващо управление на нелинейни процеси (Liao et al., 2009). Съществен недостатък на системите, използващи Тип 2 размита логика е, че те имат по-голям брой параметри за настройка в сравнение с класическата (Тип 1) размита логика.

Наред с теорията на РМ Тип 2 се развиват и тези на т.нар. множества с неравна (неправилна) форма (Rough Fuzzy Sets) и интуционистки размити множества (Intuitionistic Fuzzy Sets). Те също могат да се справят с неопределености, но се използват значително по-рядко от РМ Тип 2. RANFIS - структура, която комбинира Rough Fuzzy Sets с ANFIS, е предложена от (Chandana and Mayorga, 2006). В статията си (Wang, 2008) представя GARSINFIS – невронно-размита структура с ГА и Rough Fuzzy Sets.

Интуиционистката размита логика е предложена от (Atanassov, 1999) и е друг начин за справяне с неопределеностите в системата. Теорията на Atanassov може да се разглежда като обобщение на класическата Тип 1 размита логика. Характерна особеност на интуиционистката размита логика е, че при нея освен степен на принадлежност $\mu(x)$ се определя и степен на непринадлежност $\nu(x)$.

1.3.3. Невронно-размити предсказващи модели

Като естествено продължение в развитието на невронните мрежи и системите с размита логика се появяват хибридните интелигентни архитектури. Те комбинират яснотата и прозрачността на размитите системи с възможността за обучение на невронните мрежи. Тяхната комбинация е сравнена от (Alavala, 2008) с човешкия мозък – невронната мрежа е структурата на мозъка, т.е. "хардуера", докато размитата логика се грижи за "софтуера". Някои от по-известните хибридни невронно-размити архитектури са: ANFIS (Jang, 1991), GARIC (Bherenji, 1992), NEFCON (Nauk and Kruse, 1992), FALCON (Lin, 1991), FUN (Sulzberger, 1993), SONFIN (Feng, 1998), EFuNN (Kasabov, 2001), dmEFuNN (Kasabov and Song, 1999).

ANFIS се отнася към хибридните структури, използващи механизма за извеждане на Сугено. За обучението на ANFIS се прилага хибридно обучаващо правило - комбинация от метода на обратно разпространение на грешката за настройка на функциите на принадлежност и МНК за параметрите в следствената част. ANFIS модел, разработен в Matlab, е в основата на предложения от (Yordanova et al., 2006) невронно-размит предсказващ регулатор. Архитектурата SONFIN също използва механизма за извеждане на Сугено. При нея системата от размити правила се

конфигурира в реално време посредством едновременна структурна и параметрична идентификация. За изграждането на GARIC са използвани две невронни мрежи - ASN (Action Selection Network) и AEN (Action State Evaluation Network). ASN представлява петслойна невронна мрежа с прави връзки, при която връзките между отделните слоеве не са претеглени. AEN мрежата адаптивно оценява действията на ASN. Хибридната структура NEFCON е реализирана с извеждащия механизъм на Мамдани. Тази структура се използва в случаите когато липсват предварителни знания за системата, за обучаване на начална система от правила или за оптимизиране на ръчно дефинирани правила. NEFCON има две модификации: NEFCLASS – за класификационни задачи и NEFPROX – за апроксимация на финкции. FALCON представлява петслойна невронна мрежа, в която за всяка изходна променлива има по две стойности (два изходни неврона). Едната стойност се използва в процеса на обучение като "желан изход", а другата стойност представлява действителния изход на FALCON. За обучението на тази структура се използва хибридна обучаваща техника, която комбинира обучение без учител за определяне на началните правила и функциите на принадлежност, и метода с обратно разпространение на грешката за оптималната им настройка. EFuNN, dmEFuNN принадлежат към еволюционните хибридни структури. Характерно за тях е, че в процеса на обучение може да се изменя както броя на невроните в отделните слоеве, така и броя на връзките между тях. EFuNN работи с механизма на извеждане на Мамдани и всичките неврони в тази структура се създават по време на обучението. Така например, ако в даден момент се работи с три неврона за размиване, но входния вектор съдържа стойности, които не принадлежат на нито една от трите функции на принадлежност, то тогава се добавя нов неврон със съответната функция на принадлежност. dmEFuNN използва механизма на извеждане на Сугено. В статията си (Kasabov and Song, 2002) предлагат еволюционната структура DENFIS, която позволява в процеса на работа да се създават и добавят нови размити правила, да се актуализират съществуващи или да се премахват такива, които не са активирани определен брой итерации. Тази структура също работи с Такаги-Сугено правила, а изходът й се формира въз основа на m най-често активирани правила. Авторите също така предлагат и еволюционен клъстеринг метод (ECM), който е специално проектиран за обучението на DENFIS. Невронно-размитата структура AnYa, предложена от (Angelov, 2013), също

принадлежи към еволюционните структури. Тя работи с т.нар. облаци вместо с размити множества, което премахва необходимостта от обучение на множествата. За формирането на облаците обаче е необходима предварителна информация. Предложената от (Su et al., 2006) ACSNFIS архитектура представлява петслойна невронна мрежа. Тя се изгражда и настройва в процеса на обучение, което започва без неврони в скритите слоеве, т.е. без набор от размити правила. Хибридните структури Falcon-AART (Tan et al., 2005) и GenSoFNN (Tung, 2002) също автоматично формулират размитите правила въз основа на обучаващите данни. Самоорганизираща се хибридна невронно размита структура (Self-organizing HNFN) е предложена от (Sung-Kwun Oh et al., 2003). Тя е изградена като комбинация от невронно-размита и самоорганизираща се полиномна невронна мрежа. Първата се използва за формиране на предпоставката на размитите правила, а втората – за следствената част. За настройката на параметрите на функциите на принадлежност, моментума и скоростта на обучение е използван генетичен алгоритъм. В статията си (Wang et al., 2004) също изграждат самоорганизираща се невронно-размита структура с механизъм на извеждане на Сугено (MS-TSKfnn). Тя представлява шест-слойна мрежа с прави връзки, а за получаването на размитите правила е реализиран DIC клъстеринг метод. В (Rutkowski et al., 2003) е дискутирано изграждането на гъвкави невронно-размити структури FLEXNFIS. Характерно за FLEXNFIS е, че въз основа на входно-изходни данни се определят не само параметрите на функциите на принадлежност, но също така се определя и механизмът на извеждане. За моделиране на нелинейни системи (Wen Yu et al., 2005) използват йерархична невронно-размита архитектура HFNN. Според авторите с подобна структура може да се постигне много точно моделиране при използване на по-малък брой правила.

Хибридните невронно-размити структури намират широко приложение при системите с НМПУ. Те се включват в системите за управление основно като предсказващи модели. Обширен материал, посветен на невронно-размитото моделиране за целите на предсказващото управление може да бъде намерен в (Castellano, 2000). Различни типове архитектури на модели, базирани на размита логика и невронни мрежи са дискутирани във (Fink et al., 2003) и (Vieira et al., 2004). Редица трудове са посветени на хибридните предсказващи регулатори – (Zhang et al., 1998; Waller and Toivonen, 2000; Sanchez et al., 1999; Хаджийски и Бошнаков,

2009). Невронно-размит обобщен предсказващ регулатор NFGPC за регулиране на температурата на пара е преставен в (Xiang-Jie Liu et al., 2006). В статията си (Ibarrola et al., 2006) изграждат нелинеен DMC регулатор с използване на рекурентен невронно-размит модел. Рекурентна невронно-размита структура е в основата и на предложения от (Zhang et al., 2000) предсказващ регулатор с удължен хоризонт на предсказване. В (Su et al., 2006) е реализирано робастно предсказващо управление на нестационарни процеси с голямо чисто закъснение, базирано на Такаги-Сугено невронно-размит модел. Нелинеен многомерен предсказващ регулатор, работещ в условията на ограничения, и базиран на невронно-размит модел е представен в (Liu, 2009). Регулаторът работи като супервайзор в система на две нива, която динамично оптимизира целевата функция и определя заданията. В статията си (Grosso et al., 2012) предлагат друг йерархичен предсказващ алгоритъм - с три нива, който също се основава на невронно-размит модел. Хибриден предсказващ регулатор въз основа на невронно-размит Convolution модел, който представлява комбинация от статична невронно-размита част и импулсна характеристика, е изграден в (Paulusová and Dúbravská, 2009). Рекурентни невронно-размити модели са в основата на предсказващите регулатори (Lu et al. 2007; Mendes et al., 2011; Lu et al., 2011).

Един съществен недостатък на размитите модели, който те пренасят и в невронно-размитите структури е големият брой правила, с който работят. По принцип, броя на правилата зависи експоненциално от броя на входовете и от броя на използваните функции на принадлежност (ФП). Ако **p** е броя на входовете, а **m** е броя на ФП, то необходимия брой размити правила е $\mathbf{m}^{\mathbf{p}}$. Големият брой правила изисква определянето на още по-голям брой параметри по време на процедурата по обучение, което всъщност значително отежнява изчислителната процедура. Например, ако системата е с 10 входа, всеки от които е размит с по 2 ФП, това означава да се работи с 1024 (=2^10) правила, което прави практическата реализация почти невъзможна.

За да се намали броят на размитите правила без да се прави компромис с точността на моделиране в литературата са предложени различни методи. Един от тях е използването на йерархични размити или невронно-размити структури (Wen Yu et al., 2005). Доказано е, че йерархичните размити/невронно-размити системи също са универсални

апроксиматори (Wei, 2000; Zeng et al., 2005). При тях обаче алгоритъмът за обучение е много сложен (Yu, 2007). В своя труд (Jamshidi, 1997) предлага използването на т.нар. sensory fusion – метод, при който от няколко входни сигнали се поучава един. Новият сигнал представлява линейна функция на съставящите го. По такъв начин размерът на базата правила намалява, тъй като намалява броят на входовете. Авторът предлага също и съчетаването на йерархични структури с sensory fusion. Недостатък на тази идея е, че реализацията й зависи главно от опита на оператора. Той е този, който трябва да определи каква да е структурата на йерархичната система, а също и съответните параметри. В опит да се преодолее този недостатък в своята дипломна работа (Ledeneva, 2006) предлага автоматично определяне на тези параметри като за целта се използва генетичен алгоритъм.

Друга възможност да се опростят размитите/невронно-размитите модели е използването на ортогонални трансформации, които всъщност представляват начин за определяне на най-значимите правила от базата правила. В (Yen, 1999) са сравнени няколко различни метода на ортогонални трансформации, а именно: метод на ортогоналните най-малки квадрати (Orthogonal Least Squares - OLS), метод на разлагане по собствените стойности (Eigenvalue Decomposition - ED), метод на тоталните най-малки квадрати (Total Least Squares - TLS) и метод на директното разлагане по дадена стойност (Direct Singular Value Decomposition - D-SVD). В (Alturki, 2000) е представена модифицирана версия на Gram-Schmidt трансформацията. В (Setnes, 2001) са дадени някои коментари, касаещи употребата на ортогоналните трансформации.

Различни начини за намаляване на размитите правила чрез премахване на излишните от тях са предложени в (Kaur et al., 2012; Ciliz, 2005; Jin, 2000). Условията за намаляване на правилата от гледна точна на линейните матрични неравенства (LMIs) са представени в (Taniguchi, 2000). Метод, основан на концепцията за подобие и размита интерполация е описан в (Bellaaj et al., 2013). При него първо се измерва подобието между правилата с цел да се определят “най-добрите”, а след това се осъществява интерполация, за да се повиши точността след премахването на незначимите правила. Обзор на методите за намаляване на размитите правила е направен в (Kaynak et al., 2002).

Една група от алгоритми, които са актуални в последните години, са т.нар само-конструиращи се и само-организиращи се невронно-размити структури (Allende-Cid et al., 2008; Ferreyra, 2006). При тях по време на обучението неактивните правила се премахват, което от своя страна води до намаляване на броя на обучаваните параметри. Метод, който компресира система с произволно голям брой правила до такава с малък брой правила е представена в (Gegov, 2007). По този начин значително се намаляват он-лайн изчисленията, като при това не се прави компромис с точността. Преглед на повечето от съществуващите методи за намаляване на броя на правилата, описание на техните характеристики, както и представяне на съвременните техники за формално представяне на размитите системи на базата на Булеви матрици и бинарни връзки, е направено в (Gegov et al., 2007).

1.4. Методи и алгоритми за оптимизация на предсказващи регулатори

Определянето на стойността на управляващото въздействие при предсказващите регулатори става посредством решаването на оптимизационна задача. Всъщност определя се не една, а поредица от оптимални стойности на управлението, като само първата от тях се прилага към процеса. За целта се минимизира целеви критерий, който обикновено има квадратична форма. Най-широко прилаган е следният целеви критерий:

$$J(u,k)=E\left\{\sum_{j=N_1}^{N_p}\left(\hat{y}(k+j)-r(k+j)\right)^2+\rho\sum_{j=1}^{N_u}\left(\Delta\hat{u}(k+j-1)\right)^2\right\} \qquad (1.4.1)$$

Използването му е предложено от (Clarke et al., 1987) при представянето на Обобщеното предсказващо управление (GPC). Ако този критерий се модифицира като се приеме, че $N_1=N_p=d$ и $\rho=0$, то се получава целевия критерий, използван от регулаторът с минимална дисперсия (Minimum Variance), предложен от (Astrem and Wittenmark, 1973):

$$J(u,k)=E\left\{\left(\hat{y}(k+d)-r(k+d)\right)^2\right\} \qquad (1.4.2)$$

Характерно за този регулатор е, че той е подходящ за управление само на минималнофазови обекти.

По аналогичен начин (N_1=N_p=d и ρ>0) се получава и целевата функция на регулаторът с обобщена минимална дисперсия (Generalized Minimum Variance):

$$J(u,k)=E\left\{\left(\hat{y}(k+d)-r(k+d)\right)^2+\rho.u^2(k)\right\} \qquad (1.4.3)$$

Той е разработен от (Clarke and Gawthrop, 1975), за да се преодолеят някои от недостатъците на регулатора с минимална дисперсия.

С цел осигуряване на интегрални свойства на предсказващият регулатор (C. H Lu and C. C Tsai, 2004) предлагат използването на следният целеви критерий:

$$J(u,k)=E\left\{\begin{array}{l}\sum\limits_{j=N_1}^{N_p}\left(\hat{y}(k+j)-w_j r(k+j)+h_{ij}(z^{-1})\Delta\hat{n}(k+j-1)\right)^2+\ldots\\ +\sum\limits_{j=d}^{d+N_u-1}\left(\bar{a}_{ij}(z^{-1})\Delta\hat{u}(k+j-d)\right)^2\end{array}\right\} \qquad (1.4.4)$$

където wj е тегловен коефициент, h_{ij}(z-1) е полином, чиято основна функция е да премахне влиянието на моделната грешка $\check{n}_{к-1}$, а $\bar{a}_{ij}$(z-1) е тегловен полином.

GPC алгоритъм в непрекъснатата му форма е дискутиран в (Gawthrop et al., 1998), като използвания критерии е във вида:

$$J(u,t)=E\left\{\int_{\tau_1}^{\tau_2}\left(\hat{y}(t+\tau)-r(t+\tau)\right)^2 d\tau+\rho\int_{\tau_1}^{\tau_2}(\hat{u}(t+\tau))^2 d\tau\right\} \qquad (1.4.5)$$

Представените по-горе целеви критерии са квадратични, но е възможно също така и използването на неквадратични. Подобен е предложен от (Chang et al., 1983):

$$J(u,k)=E\left\{\sum_{j=N_1}^{N_p}\sum_{i=1}^{m}q_i\left(\hat{y}(k+j)-r(k+j)\right)\right\} \qquad (1.4.6)$$

Проблеми, използващи на такъв целеви критерий, се основават на линейното програмиране (Linear Programming). Характерно за тях е, че са значително по-леки в изчислителен план и търсената оптимална стойност се достига за по-кратко време в сравнение с проблемите, решавани посредством квадратичното програмиране (Quadratic Programming).

Друг тип неквадратична целева функция е тази, използваща мин-максната формулировка (Min-max cost function), която е подробно описана в (Maciejowski, 2002):

$$J(u,k)=E\left\{\min_i \max_j \left(\hat{y}_j(k+i)-r_j(k+i)\right)q_j\right\} \tag{1.4.7}$$

Този тип целева функция се прилага в случаите, когато се цели минимизиране на максималното динамично отклонение. Прилагайки математическия апарат на мин-максната техника към стандартния GPC-критерий, (Ramirez et al., 2004) проектират мин-максен предсказващ регулатор (Min-max Model Predictive Controller - MMMPC).

Едно от основните предимства на МПУ е възможността да отработва ограничения, които се включват също към целевата функция. Те са породени главно от икономически и от съображения за сигурност. Ограниченията, използвани при МПУ, са твърди (Hard Constraints), които не трябва да бъдат нарушавани, и меки (Soft Constraints). Освен това те могат да бъдат областни и функционални, съответно във вид на равенства или неравенства:

- областни ограничения

$$\begin{aligned} u_{\min} &\le u(k) \le u_{\max} \\ \Delta u_{\min} &\le \Delta u(k) \le \Delta u_{\max} \\ y_{\min} &\le y(k) \le y_{\max} \\ x_{\min} &\le x(k) \le x_{\max} \end{aligned} \tag{1.4.8}$$

- функционални ограничения

$$\begin{aligned} \phi(k) &= Cx(k) + De(k) = 0 \\ \tilde{\psi}(k) &= Cx(k) + De(k) \le \psi(k) \end{aligned} \tag{1.4.9}$$

Въз основа на вида на използваната целева функция и ограничения се избира и начинът за изчисляване стойността на управлението, т.е. методът за оптимизация. При НМПУ на всеки такт на дискретизация трябва да се решава нелинеен оптимизационен проблем, известен още като задача на нелинейното програмиране. Основните изисквания, поставени пред оптимизационният алгоритъм, са бързата му сходимост и опростена изчислителна процедура, която да позволява реализацията му в реално време. Методи с намалена изчислителна тежест, гарантиращи в същото

време устойчивост и работоспособност на системата се предложени от (Chen и Allgöwer, 1998; De Nicolao et al., 1996; Tenny et al., 2002). Обширни трудове, посветени на изследването и разработването на алгоритми за решаването на нелинейната оптимизационна задача в реално време са предложени от (Diehl, 2001) и (Tenny, 2002).

Според (Zheng, 1997) НМПУ алгоритмите се разделят на две големи групи в зависимост от начина, по който се решава оптимизационния проблем. Едната група алгоритми реализират оптимизацията директно, а другата посредством апроксиматор (непреки методи за оптимизация). Подобен апроксиматор е използван в предложения от (Åkesson et al., 2006) предсказващ регулатор. Апроксиматорът представлява невронна мрежа с прави връзки, обучена посредством алгоритъма на Левенберг-Маргуард. За решаването на нелинейна оптимизационна задача с ограничения във вид на равенства (Wang et al., 2006) предлагат модифициран метод на Хесе, който е робастен и осигурява съкращаване на изчислителната процедура. Проектираните от (Kerrigan et al., 2002) предсказващи регулатори са изградени въз основа на многокритериален оптимизационен проблем, Последният се изчислява посредством решаването в реално време на редица еднокритериални целеви функции като се вземе в предвид съответният им приоритет, както и този на наложените ограничения. При определянето на управлението (Primož et al., 2002) използват генетичен алгоритъм, при който в ролята на фитнес функция е използван дефинираният целеви критерий. Реализирано е кодиране с плаваща точка, а кръстосването и мутацията са с наложени ограничения. Генетични алгоритми обаче, като метод за глобална оптимизация, са твърде бавни и неподходящи за приложение в реално време.

1.5. Методи и алгоритми за настройка на предсказващи регулатори

Целта на настройката на предсказващите регулатори е осигуряване на добро отследяване на заданието, робастност при несъответствие на модела и намаляване влиянието на смущенията. Изхождайки от целевия критерий (1.4.1), като основни параметри за настройка могат да се определят: N1 - минимален хоризонт на предсказване; Np – хоризонт на предсказване; Nu – хоризонт на управление и λ – тегловен фактор.

Минималният хоризонт на предсказване N1 се свързва с чистото закъснение в системата. Най-често той се избира равен на единица (N1=1), но може да приема и по-големи стойности в зависимост от закъснението. Стойността на параметъра Np се определя в зависимост от времето на преходния процес и трябва да е по-голяма от максималното възможно чисто закъснение. Като правило хоризонтът на управление е винаги по-малък от хоризонта на предсказване. Той приема стойности от интервала [1, Np]. Малкият хоризонт на управление придава робастни свойства на управляващата система. От друга страна използването на голям хоризонт осигурява гъвкавост на управляващият сигнал, а оттам и по-добро качество на управление, но увеличава в значителна степен броя на изчисленията. Тегловният фактор λ, наречен още наказателен, се избира в интервала [0,1). На практика неговата стойност трябва да е възможно най-малка, дори в редица случаи нула. При неминималнофазови системи нулевата стойност на λ обаче може да доведе до проблеми с устойчивостта.

Тъй като описаните по-горе предписания дават значителна свобода на избор и поради липсата на систематичен подход при определянето параметрите на регулатора, настройката се реализира основно по метода на “пробите и грешките” като се взимат предвид характеристиките на конкретния обект за управление. Настройката по един такъв начин е трудна и продължителна задача. Поради тази причина голям брой изследователи са насочили усилията си в разработването на методи за настройка на предсказващите регулатори. В своя труд (Rani et al., 1997) правят сравнение на съществуващи техники за настройка на предсказващи регулатори, в частност на DMC и GPC алгоритми. В резултат на направения от тях анализ и изхождайки от факта, че при системи с предавателна функция от първи ред времето за установяване е приблизително 3 до 4 пъти времеконстантата, авторите предлагат хоризонта на предсказване да се определя съгласно израза:

$$N_p = d_{\max} + ((t_s / T_s)/3.5) \qquad (1.5.1)$$

където dmax е максималната стойност на чистото закъснение, ts е времето за установяване, а Ts е периода на дискретизация. Shridhar and Cooper (1997, 1998) предлагат лесна за използване техника за настройка на DMC алгоритъм. Тя обаче се отнася за предсказващи алгоритми без ограничения. Освен това, за да се използва този подход е необходимо

процесът да се представи с предавателна функция от първи ред с чисто закъснение (FOPDT).

В (Ali and Zafiriou, 1993) е представена офлайн процедура за настройка на предсказващи алгоритми за управление на нелинейни процеси. По същество този начин за настройка представлява оптимизационна задача, която се решава с помощта на интерактивния софтуер CONSOLE. Друг офлайн алгоритъм е предложен от (Jiang and Jutan, 2000). Той също се свежда до оптимизационна задача, при която в ролята на целева функция е използвана интегрално-квадратичната грешка. Характерно за последните два алгоритъма е, че реализацията им е свързана с тежка изчислителна процедура, поради което не могат да бъдат реализирани в онлайн режим.

Онлайн адаптивна стратегия за настройка на линейни предсказващи алгоритми е разработена от (Al-Ghazzawi et al., 2001). Този метод е валиден както в случаите на наличие на ограничения, така и при липсата им. При него се настройват тегловните матрици Q(j) пред предсказаната грешка и R(j) пред управлението:

$$J(u,k)=E\left\{\begin{array}{l}\sum\limits_{j=N_1}^{N_p} Q(j)\left(\hat{y}(k+j)-r(k+j)\right)^2+\ldots \\ +R(j)\sum\limits_{j=1}^{N_u}\left(\Delta\hat{u}(k+j-1)\right)^2\end{array}\right\} \qquad (1.5.2)$$

За целта се изчисляват производните на предсказания изход по отношение на настройваемите параметри. Стойностите на хоризонтите на предсказване и управление са предварително определени въз основа на по-горе описаните правила. Този метод обаче не може да бъде развит за НМПУ поради трудността при получаване на аналитичните изрази за производните. Като надстройка на този алгоритъм се явява предложеният от (Ali, 2003) евристичен метод за онлайн настройка. При него авторът вместо производни използва размита логика за получаването на новите стойности на настройваемите параметри. Модифициран по такъв начин, алгоритъмът е много по-лек в изчислително отношение и е подходящ за целите на НМПУ.

Някои прости правила за автонастройка на предсказващ регулатор са описани във (Valencia-Palomo and Rossiter, 2009). Генетичен алгоритъм е

използван за оптималната настройка на описания в (De Almeida et al., 2009) DMC регулалтор с ограничения. Как да се използват генетични алгоритми за настройката на Обобщен предсказващ регулатор е описано в (De Almeida et al., 2006). В (Dougherty and Cooper, 2003) е предложен аналитичен израз за изчисляване на тегловния фактор λ за едномерни и многомерни системи с цел да получат автоматичен метод за настройка.

Като цяло обаче може да се направи изводът, че в областта на настройката на предсказващите регулатори не е работено задълбочено и също така не е проучена възможността за използването на интелигенти системи (в това число и невронно-размити структури) за настройката на предсказващите регулатори.

1.6. Изводи

Направеният обзор разкрива изключително голямо разнообразие на хибридни невронно-размити структури, подходящи за включване в системи с НМПУ в ролята на предсказващи модели. Многобройните изследвания и разработки в това направление говорят за засилен интерес и стремеж към усъвършенстване на предсказващите регулатори с невронно-размит модел. Един съществен недостатък, който трябва да бъде преодолян, е намаляване на изчислителната тежест на предсказващите регулатори. Това би могло да се постигне от една страна чрез използване на невронно-размити структури, работещи с редуциран брой размити правила, а от друга с оптимизационни алгоритми, осигуряващи бързодействие на предсказващите регулатори. Друг важен аспект е точността на предсказване, тъй като от нея зависи качеството на управление. Едно подходящо решение в това отношение е включването на Тип 2 или Интуиционистка размита логика в хибридните навронно-размити модели. Прави впечатление също така, че в областта на настройката на предсказващите регулатори не е работено задълбочено. Не е проучена и възможността за използването на невронно-размити структури за тази цел.

1.7. Формулиране на целта и задачите на дисертационния труд

С оглед на поставената тема и въз основа на направения обзор могат да бъдат формулирани следната цел и задачи на дисертационния труд:

Основна цел: Разработване на невронно-размити предсказващи модели с механизъм на извеждане Такаги-Сугено с намалена изчислителна тежест (с оптимизиран брой размити правила), подходящи за целите на нелинейното предсказващо управление.

Основни задачи:

1. Да се разработи и софтуерно реализира разпределен невронно-размит модел, работещ с редуциран брой размити правила.
2. Да се проектира, изследва и разработи програмен код на изчислително ефективен невронно-размит модел с частично размиване на входните сигнали.
3. Да се проектира, изследва и разработи програмно приложение за модифициран нео-размит модел.
4. Да се разработи софтуерно и изследва модифициран нео-размит модел във вариант с Тип 2 размита логика и с Интуиционистка размита логика.
5. Да се реализират програмно и изследват оптимизационни алгоритми от II ред (алгоритъм на Нютон и алгоритъм на Левенберг-Маргуард), които осигуряват бързодействие на предсказващия регулатор.
6. Оптимизационните алгоритми от II ред да се реализират софтуерно във вариант с LU декомпозиция – изчислително ефективен метод, който позволява да се избегне намирането на обратната матрица на Хесе на всеки такт.
7. Да се изследва възможността за използване на интелигентни структури в алгоритми за супервайзорна настройка на предсказващите алгоритми.

Глава II

Нелинейно невронно-размито обобщено предсказващо управление - теоретична постановка

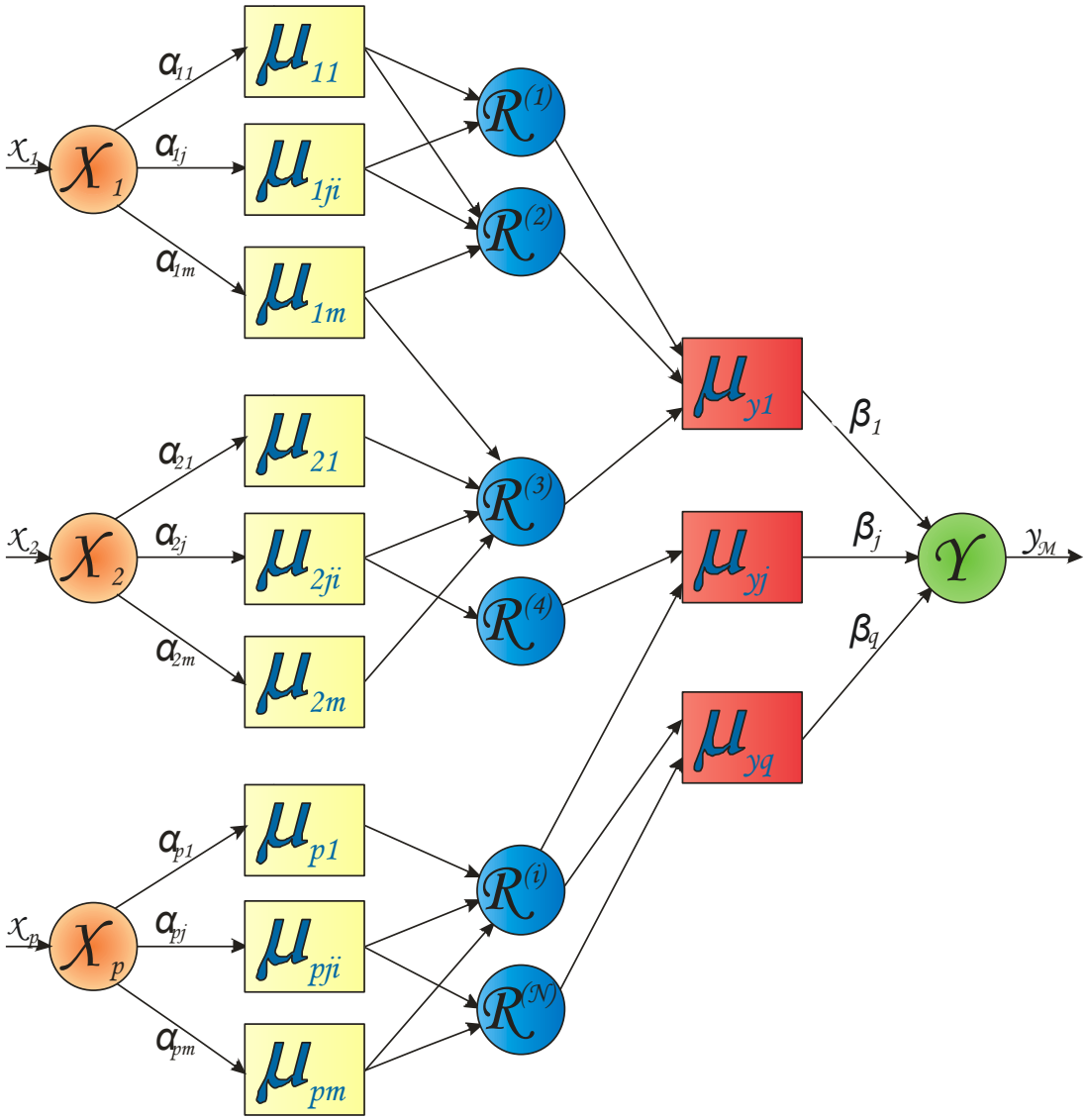
Предсказващото управление е научно направление, което се развива с бързи темпове през последните години. Разработени са различни предсказващи алгоритми, но най-широко приложение е намерило предложеното от (Clarke, 1987) Обобщено предсказващо управление (Generalized Predictive Control). Предсказващите алгоритми като цяло, се отнасят към класа на адаптивните самонастройващи се регулатори и като такива могат да се реализират в явна (непряка) и неявна (пряка) форма (Велев, 1994). В настоящата дисертация са разработени невронно-размити модели с механизъм на извеждане Такаги-Сугено с намалена изчислителна тежест (с оптимизиран брой размити правила), подходящи за включване в алгоритми за нелинейно обобщено предсказващо управление в неявна форма.

2.1. Невронно-размит предсказващ модел

Предсказващият модел е основен елемент и неразделна част от структурата на предсказващите регулатори (фиг. 1.1.2). Той има за цел да предскаже бъдещите стойности на изхода на обекта $\hat{y}(k+j)$, за $j=1,..,N_p$, въз основа на минали негови стойности и минали стойности на управляващия сигнал **u** до момента **k**. Точността на предсказване е от съществено значение за качеството на управление. Това твърдение следва непосредствено от факта, че грешката между моделния изход и заданието се използва като входен сигнал за оптимизатора, т.е на входа на блока, определящ стойността на управлението.

За целите на настоящата работа в ролята на предсказващ модел е използван нелинеен невронно-размит авторегресионен модел (NNFARX),

който представлява петслойна невронна мрежа и използва Такаги-Сугено механизъм на извеждане (фиг. 2.1.1).



Фиг. 2.1.1 Хибридна невронно-размита структура

Реализираният NNFARX модел (фиг. 2.1.1) е MISO (много входове един изход) и представлява система от линейни подмодели, изградени във вид на отделни размити правила. Първият входен слой на NNFARX модела е непараметричен и служи за съгласуване и разпределение на входните сигнали, описани посредством вектор-регресора **x(k)**:

$$\mathbf{x}(k)=[u(k),...,u(k-n_u),y(k-1),..,y(k-n_y)]=[x_1,x_2,...,x_p] \quad (2.1.1)$$

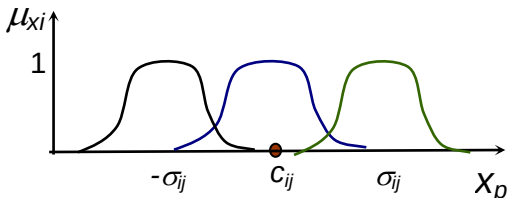
Както се вижда от горния израз той включва $\mathbf{n_u}$ на брой минали стойности на управлението и $\mathbf{n_y}$ на брой минали стойности на изхода на

обекта. Броят p=1,…,P на включените във вектор-регресора **x(k)** минали стойности (т.е. броят на входовете на NNFARX модела) се определя основно от реда на моделирания обект. Трябва да се има предвид също така, че по-големият брой елементи в регресионния вектор от една страна води до по-точно моделиране, но от друга - утежнява изчислителната процедура, тъй като се увеличава броят на размитите правила и съответно на параметрите за обучение.

Вторият слой на предсказващия модел е параметричен и реализира операцията размиване - преобразуване на входните сигнали от точни стойности в съответните лингвистични променливи определени от размитите множества, посредством дефинирани входни функции на принадлежност. За целите на нелинейното моделиране входните функции на принадлежност обикновено се избират да са нелинейни от тип Гаусова функция. Това се прави с цел да се постигне нелинейна апроксимация на входния диапазон. Гаусовите множества са представени на фиг. 2.1.2 и се описват със следната зависимост:

$$\mu_{Xp,m}^{(n)} = \exp\frac{-(x_p - c_{Xp,m})^2}{2\sigma_{Xp,m}^2} \qquad (2.1.2)$$

където $\mu_{p1}\ldots\mu_{pm}$ са степените на принадлежност на x_p входната променлива към m-то размито множество, а $c_{Xp,m}$ и $\sigma_{Xp,m}$ са съответно центъра и ширината на отделните множества.



Фиг. 2.1.2 Гаусови функции на принадлежност

Третият слой на NNFARX модела представлява своеобразен генератор на правила тъй като в него се формират размитите логически правила. Техният брой **N** е функция от броя на входовете на невронно-размития модел **p** и от броя на съответните им размити множества **m**=1,…,M и се изчислява от тяхната комбинация m^p. Размитите правила имат следната форма:

$$R^{(N)}: if\ \ x_1(k+j)\ is\ \tilde{X}_1^{(N)}\ \ and\ \ x_2(k+j)\ is\ \tilde{X}_2^{(N)}\ldots \\ if\ x_p(k+j)\ is\ \tilde{X}_p^{(N)}\ then\ \ f_y^{(N)}(k+j) \tag{2.1.3}$$

където с $\tilde{X}_p^{(N)}$, p=1…P, са означени активираните размити множества, а функцията на Сугено $f_y^{(N)}$, използвана в следствената част на размитите правила е:

$$f_y^{(N)}(k+j)=a_1^{(N)}y(k+j-1)+\ldots+a_{ny}^{(N)}y(k+j-n_y)+\ldots \\ +b_1^{(N)}u(k+j)+\ldots+b_{nu}^{(N)}u(k+j-n_u)+b_0^{(N)} \tag{2.1.4}$$

В горния израз свободният член b_0 играе ролята на филтър на смущенията. Това лесно може да бъде доказано ако изразът (2.1.4) бъде преобразуван в следната форма:

$$f_y^{(N)}(k+j)=a_1^{(N)}y(k+j-1)+\ldots+a_{ny}^{(N)}y(k+j-n_y)+\ldots \\ +b_1^{(N)}u(k+j)+\ldots+b_{nu}^{(N)}u(k+j-n_u)+d(k+j) \tag{2.1.5}$$

Това преобразуване е в сила при T(q-1)=1. В (2.1.5) d(k) е неизвестно смущение, което се определя от:

$$d(k)=\frac{T(q^{-1})}{\Delta(q^{-1})}v(k) \tag{2.1.6}$$

Последният израз от своя страна представлява част от ARIMAX модела, който е стандартно използван при обобщените предсказващи регулатори:

$$A(q^{-1})\hat{y}(k)=B^{(i)}(q^{-1})u(k)+\frac{T(q^{-1})}{\Delta(q^{-1})}v(k) \tag{2.1.7}$$

където v(k) е неизвестна променлива с нулево математическо очакване, която описва едновременно ефекта от смущенията и измеримия шум.

Последните два слоя на NNFARX модела са непараметрични и формират размития механизъм за извеждане. В четвъртия слой се реализира операцията импликация като за целта се използва средно претеглено произведение:

$$\mu_{yq}^{(n)}(k+j)=\mu_{x_1,m}^{(n)}(k+j)*\mu_{x_2,m}^{(n)}(k+j)*\ldots*\mu_{x_p,m}^{(n)}(k+j) \tag{2.1.8}$$

В петия последен слой се взима решението за извеждане, което се състои в определяне на стойността на изхода на модела посредством израза:

$$\hat{y}_M(k+j)=\frac{\sum_{i=1}^{q} f_y^{(i)}(k+j)\mu_y^{(i)}(k+j)}{\sum_{i=1}^{q} \mu_y^{(i)}(k+j)} \qquad (2.1.9)$$

Изразът (2.1.9) представлява дискретна форма на предсказания изход на обекта с q активирани правила. Опростен вариант на израза за изхода на модела се получава чрез въвеждането на нормализирана степен на принадлежност:

$$\hat{y}_M(k+j)=\sum_{i=1}^{q} f_y^{(i)}(k+j)\bar{\mu}_y^{(i)}(k+j) \qquad (2.1.10)$$

Параметричната идентификация е следващият етап от изграждането на предсказващия невронно-размит модел. Тя се състои в определяне на параметрите в предпоставката и в заключителната част на размитите логически правила, т.е. в параметричните слоеве. Това предполага изчисляване на две групи параметри - параметрите на размитите множества от втория слой (в конкретния случай Гаусовите функции) и коефициентите във функцията на Сугено от третия слой. Параметрите на размитите множества са нелинейни тъй като участват в нелинейната функция (2.1.2), а параметрите от функцията на Сугено са линейни. Броят на линейните параметри е в тясна връзка с броя на входните променливи. При P на брой входни променливи броят на линейните параметри за едно правило е (P+1), а за целия набор правила е N*(P+1). От друга страна всяка функция на принадлежност се характеризира с два параметъра – център $c_{Xp,m}$ и ширина $\sigma_{Xp,m}$. Поради това броят на нелинейните параметри е равен на 2*P*N. Следователно общият брой параметри, които трябва да се определят при обучението е N*(3P+1).

2.2. Неявна форма на нелинейно обобщено предсказващо управление

Неявната форма на нелинейното обобщено предсказващо управление е изградена на базата на описания в т.2.1. невронно-размит Такаги-Сугено модел. В него са включени две минали стойности на изхода на модела и две минали стойности на управлението. Използван е и стандартния GPC-критерий (1.4.1), като е прието, че не са наложени ограничения. При такава постановка изчисляването на поредицата от оптимални стойности на управлението може да бъде направено аналитично. За да се опростят математическите изрази, формулировката на неявната форма на нелинейното обобщено предсказващо управление е представена в матричен вид. По такъв начин целевият критерий (1.4.1) се преобразува в следния вид:

$$J[k,U(k)]=[R(k)-\hat{Y}(k)]^T[R(k)-\hat{Y}(k)]+\lambda\tilde{U}(k)^T\tilde{U}(k) \qquad (2.2.1)$$

Той включва векторите на заданието, предсказаният изход на модела, предсказаните стойности за сигнала на управлението и предсказаната грешка, които са дадени чрез съответните изрази:

$$R(k)=[r(k+N_1),\ldots,r(k+N_2)]^T$$

$$\hat{Y}(k)=[\hat{y}(k+N_1),\ldots,\hat{y}(k+N_2)]^T$$

$$\Delta\tilde{U}(k)=[\Delta u(k+N_1),\ldots,\Delta u(k+N_2)]^T$$

$$\hat{E}(k)=[\hat{e}(k+N_1),\ldots,\hat{e}(k+N_2)]^T$$

където $\hat{e}(k+i)=r(k+i)-\hat{y}(k+i) \quad i=N_1,N_1+1,\ldots,N_p$

Минимизацията на GPC-критерия се основава на изчислението на вектор градиента на целевата функция в k-тия такт по отношение на предсказаните стойности на управляващото въздействие:

$$\nabla J[k,U(k)]=\left[\frac{\partial J[k,U(k)]}{\partial u(k)},\frac{\partial J[k,U(k)]}{\partial u(k+1)},\ldots,\frac{\partial J[k,U(k)]}{\partial u(k+N_u-1)}\right] \qquad (2.2.2)$$

Спрямо матричното уравнение (2.2.1), елементите на вектор-градиента могат да се представят както следва:

$$\frac{\partial J[k,U(k)]}{\partial U(k)} = \left[-2[R(k) - \hat{Y}(k)]^T \frac{\partial \hat{Y}(k)}{\partial U(k)} + 2\lambda \hat{U}(k)^T \frac{\partial \hat{U}(k)}{\partial U(k)} \right] \quad (2.2.3)$$

От този израз става ясно, че е необходимо да се изчислят две групи частни производни. Първата група производни могат да бъдат записани в следната матрица:

$$\frac{\partial \hat{Y}(k)}{\partial U(k)} = \begin{bmatrix} \frac{\partial \hat{y}(k+N_1)}{\partial u(k)} \cdots\cdots \frac{\partial \hat{y}(k+N_1)}{\partial u(k+N_u-1)} \\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ \frac{\partial \hat{y}(k+N_2)}{\partial u(k)} \cdots\cdots \frac{\partial \hat{y}(k+N_2)}{\partial u(k+N_u-1)} \end{bmatrix}_{[(Np-N1+1)xNu]} \quad (2.2.4)$$

По-долу са показани зависимостите, по които става изчислението на елементите от първата колона на тази матрица:

$$\frac{\partial \hat{y}(k)}{\partial u(k)} = \sum_{i=1}^{N} b_1^{(i)} \overline{\mu}_y^{(i)}(k) \quad (2.2.5)$$

$$\frac{\partial \hat{y}(k+1)}{\partial u(k)} = \sum_{i=1}^{N} \left[a_1^{(i)} \frac{\partial \hat{y}(k)}{\partial u(k)} + b_2^{(i)} \right] \overline{\mu}_y^{(i)}(k+1) \quad (2.2.6)$$

$$\frac{\partial \hat{y}(k+2)}{\partial u(k)} = \sum_{i=1}^{N} \left[a_1^{(i)} \frac{\partial \hat{y}(k+1)}{\partial u(k)} + a_2^{(i)} \frac{\partial \hat{y}(k)}{\partial u(k)} \right] \overline{\mu}_y^{(i)}(k+2) \quad (2.2.7)$$

$$\frac{\partial \hat{y}(k+3)}{\partial u(k)} = \sum_{i=1}^{N} \left[a_1^{(i)} \frac{\partial \hat{y}(k+2)}{\partial u(k)} + a_2^{(i)} \frac{\partial \hat{y}(k+1)}{\partial u(k)} \right] \overline{\mu}_y^{(i)}(k+3) \quad (2.2.8)$$

$$\frac{\partial \hat{y}(k+N_1)}{\partial u(k)} = \sum_{i=1}^{N} \left[a_1^{(i)} \frac{\partial \hat{y}(k+N_1-1)}{\partial u(k)} + a_2^{(i)} \frac{\partial \hat{y}(k+N_1-2)}{\partial u(k)} \right] \overline{\mu}_y^{(i)}(k+N_1) \quad (2.2.9)$$

$$\frac{\partial \hat{y}(k+N_p)}{\partial u(k)} = \sum_{i=1}^{N} \left[a_1^{(i)} \frac{\partial \hat{y}(k+N_p-1)}{\partial u(k)} + a_2^{(i)} \frac{\partial \hat{y}(k+N_p-2)}{\partial u(k)} \right] \overline{\mu}_y^{(i)}(k+N_p) \quad (2.2.10)$$

По аналогичен начин могат да се изчислят и останалите елементи на матрицата (2.2.4).

Втората група производни са дадени посредством матрицата:

$$\frac{\partial \hat{U}(k)}{\partial U(k)} = \begin{bmatrix} \frac{\partial \Delta u(k)}{\partial u(k)} & \cdots\cdots & \frac{\partial \Delta u(k)}{\partial u(k+N_u-1)} \\ \cdots\cdots & \cdots\cdots & \cdots\cdots \\ \frac{\partial \Delta u(k+N_u-1)}{\partial u(k)} & \cdots\cdots & \frac{\partial \Delta u(k+N_u-1)}{\partial u(k+N_u-1)} \end{bmatrix}_{[NuxNu]} \tag{2.2.11}$$

Тъй като $\Delta u(k)=u(k)-u(k-1)$, то:

$$\frac{\partial \hat{U}(k)}{\partial U(k)} = \begin{bmatrix} 1 & 0 & \ldots & 0 \\ -1 & 1 & \ldots & 0 \\ \vdots & -1 & & \vdots \\ 0 & 0 & \ldots & 1 \end{bmatrix}_{NuxNu} \tag{2.2.12}$$

След като бъдат изчислени отделните градиентни елементи, те се приравняват на нула. В резултат на това се получава нова система от уравнения, която може да бъде решена по отношение на управляващите въздействия u(k), u(k+1), .., u(k+Nu+1):

$$\frac{\partial J[k,U(k)]}{\partial u(k)} = -2\hat{e}(k+N_1)\frac{\partial \hat{y}(k+N_1)}{\partial u(k)} - \ldots - 2\hat{e}(k+N_2)\frac{\partial \hat{y}(k+N_2)}{\partial u(k)} + 2\lambda\Delta u(k) - 2\lambda\Delta u(k+1) = 0 \tag{2.2.13}$$

$$\frac{\partial J[k,U(k)]}{\partial u(k+1)} = -2\hat{e}(k+N_1)\frac{\partial \hat{y}(k+N_1)}{\partial u(k+1)} - \ldots - 2\hat{e}(k+N_2)\frac{\partial \hat{y}(k+N_2)}{\partial u(k+1)} + 2\lambda\Delta u(k+1) - 2\lambda\Delta u(k+2) = 0 \tag{2.2.14}$$

$$\frac{\partial J[k,U(k)]}{\partial u(k+N_u-2)} = -2\hat{e}(k+N_1)\frac{\partial \hat{y}(k+N_1)}{\partial u(k+N_u-2)} - \ldots - 2\hat{e}(k+N_2)\frac{\partial \hat{y}(k+N_2)}{\partial u(k+N_u-2)} + 2\lambda\Delta u(k+N_u-2) - 2\lambda\Delta u(k+N_u-1) = 0 \tag{2.2.15}$$

$$\frac{\partial J[k,U(k)]}{\partial u(k+N_u-1)} = -2\hat{e}(k+N_1)\frac{\partial \hat{y}(k+N_1)}{\partial u(k+N_u-1)} - \ldots - \\ -2\hat{e}(k+N_2)\frac{\partial \hat{y}(k+N_2)}{\partial u(k+N_u-1)} + 2\lambda\Delta u(k+N_u-1) = 0 \qquad (2.2.16)$$

Тази система от уравнения може да бъде решена много просто като се започне в обратен ред от последното уравнение (2.2.16), от което се изчислява управляващото въздействие Δu(k+Nu-1). Получената стойност се замества в предходното уравнение и от него се определя Δu(k+Nu-2). По подобен начин се изчислява цялата последователност от управляващи въздействия в рамките на хоризонта на управление.

Програмната реализация на представения оптимизационен алгоритъм е опростена и се основава на следната система уравнения:

$$\Delta u(k+N_u-1) = \Delta u(k+N_u) + \\ +\lambda^{-1}\left[e(k+N_1)\frac{\partial \hat{y}(k+N_1)}{\partial u(k+N_u-1)} + \ldots + e(k+N_2)\frac{\partial \hat{y}(k+N_2)}{\partial u(k+N_u-1)} \right] \qquad (2.2.17)$$

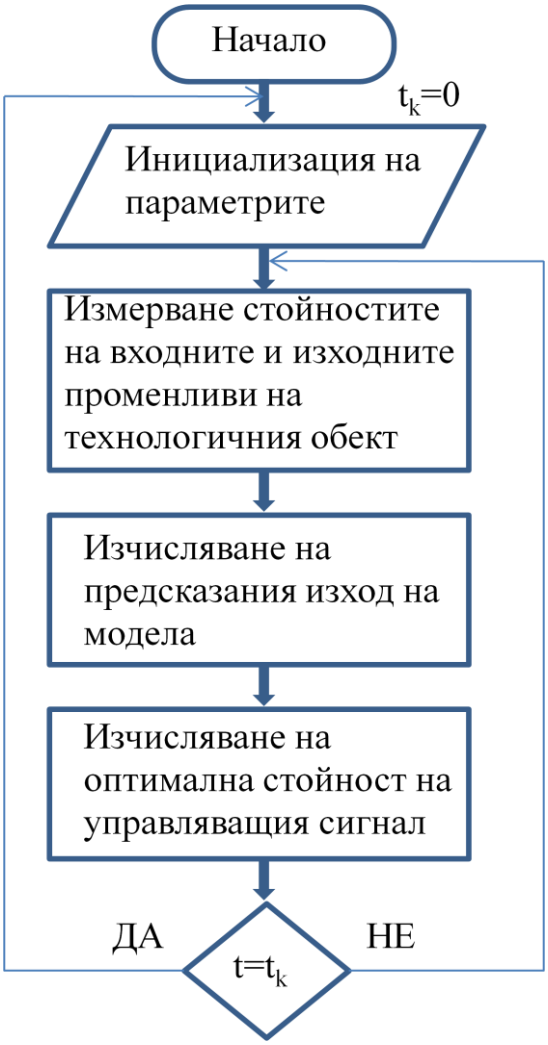
$$\Delta u(k+N_u-2) = \Delta u(k+N_u-1) + \\ +\lambda^{-1}\left[e(k+N_1)\frac{\partial \hat{y}(k+N_1)}{\partial u(k+N_u-2)} + \ldots + e(k+N_2)\frac{\partial \hat{y}(k+N_2)}{\partial u(k+N_u-2)} \right] \qquad (2.2.18)$$

$$\Delta u(k+1) = \Delta u(k+2) + \\ +\lambda^{-1}\left[e(k+N_1)\frac{\partial \hat{y}(k+N_1)}{\partial u(k+1)} + \ldots + e(k+N_2)\frac{\partial \hat{y}(k+N_2)}{\partial u(k+1)} \right] \qquad (2.2.19)$$

$$\Delta u(k) = \Delta u(k+1) + \\ +\lambda^{-1}\left[e(k+N_1)\frac{\partial \hat{y}(k+N_1)}{\partial u(k)} + \ldots + 2e(k+N_2)\frac{\partial \hat{y}(k+N_2)}{\partial u(k)} \right] \qquad (2.2.20)$$

Алгоритъмът за оптимизация, реализиран посредством подобно повтарящо се и последователно заместване, *представлява итеративен оптимизационен алгоритъм*. След приключване на изчисленията в текущия такт k индексите на елементите в уравнение (2.2.3) се променят и процедурата стартира отначало. Описаният тук оптимизационен алгоритъм представлява градиентен метод от първи ред.

Диаграмата, описваща последователността на действие на нелиеен невронно-размит обобщаващ регулатор, е представена на фиг. 2.2.1.



Фиг. 2.2.1 Алгоритъм на нелиеен обобщаващ невронно-размит предсказващ регулатор

2.3. Изводи

В настоящата глава е описан алгоритъмът на работа на нелинеен обобщен предсказващ регулатор. В ролята на предсказващ модел при него е използван невронно-размит авторегресионен модел (NNFARX), като е прието той да има четири входа (две минали стойности на изхода, миналата и текуща стойност на управлението) и всеки от тях да се размива с по три Гаусови множества. От тук произтича и един от основните недостатъци на този регулатор, а именно големият брой размити правила, с които работи предсказващият модел. Вече беше споменато, че техният

брой се определя съгласно израза $N=m^p$ и следователно в конкретния случай броят на правилата е 81. Това от своя страна означава, че на всеки такт по време на обучението на този модел трябва да бъдат определени стойностите на 1053 параметъра! Това го прави практически неприложим за работа в реално време, особено за процеси с бързо изменяща се динамика.

Глава III

Проектиране на невронно-размити модели и алгоритми за регулатори с невронно-размито предсказващо управление

Моделът представлява математическо описание на физична, биологична или информационна система. С помощта на модела системата може да бъде изследвана, анализирана, да се предскаже какво ще е бъдещото й състояние. В системите с НМПУ широко се използват емпиричните модели. Това се дължи на факта, че те са значително по-лесни за получаване в сравнение с моделите с фундаментални взаимовръзки. Емпиричните модели в системите с НМПУ се използват в различни форми – размита логика, невронни мрежи, невронно-размити структури, полиномни модели, Волтера серии и т.н.

Въз основа на направения в Глава I обзор и в съответствие с поставената цел в настоящата работа е използван описаният в Глава II, т. 2.1 нелинеен невронно-размит модел с механизъм на извеждане Такаги-Сугено. Посредством него е реализиран NNFARX модел, работещ с 81 размити правила и изискващ актуализация на 1053 параметъра на всеки такт по време на обучението. За да се преодолее този недостатък са разработени две модификации на описания по-горе NNFARX модел - *Разпределен невронно-размит модел (DANFA - Distributed Adaptive Neuro Fuzzy Architecture)* и *Невронно-размит модел с частично размиване на входните сигнали (SFNN - Semi Fuzzy Neural Network) модел*, а също и *модифициран нео-размит модел (Modified Neo Fuzzy Model),* които работят с редуциран брой правила.

3.1. Разпределен невронно-размит модел

Структурата на Разпределен невронно-размит модел (Distributed Adaptive Neuro Fuzzy Architecture – DANFA) е представена на фиг. 3.1.1. Основната идея при него е входните сигнали да бъдат „разпределени" (от където идва и наименованието му) към отделни невронно-размити структури от типа, представен на фиг. 2.1.1. По този начин се получава мрежа от невронно-размити структури. Всяка от тези невронно-размити структури играе ролята на отделен подмодел, а обобщеният (глобален) DANFA модел е съвкупност от *q* на брой подмодела.

Разпределеният невронно-размит модел [1] представлява шестслойна адаптивна архитектура с механизъм на извеждане на Такаги-Сугено, като отделните слоеве изпълняват както следва:

Слой 1: Невроните от този слой приемат входните сигнали и ги предават към следващия слой.

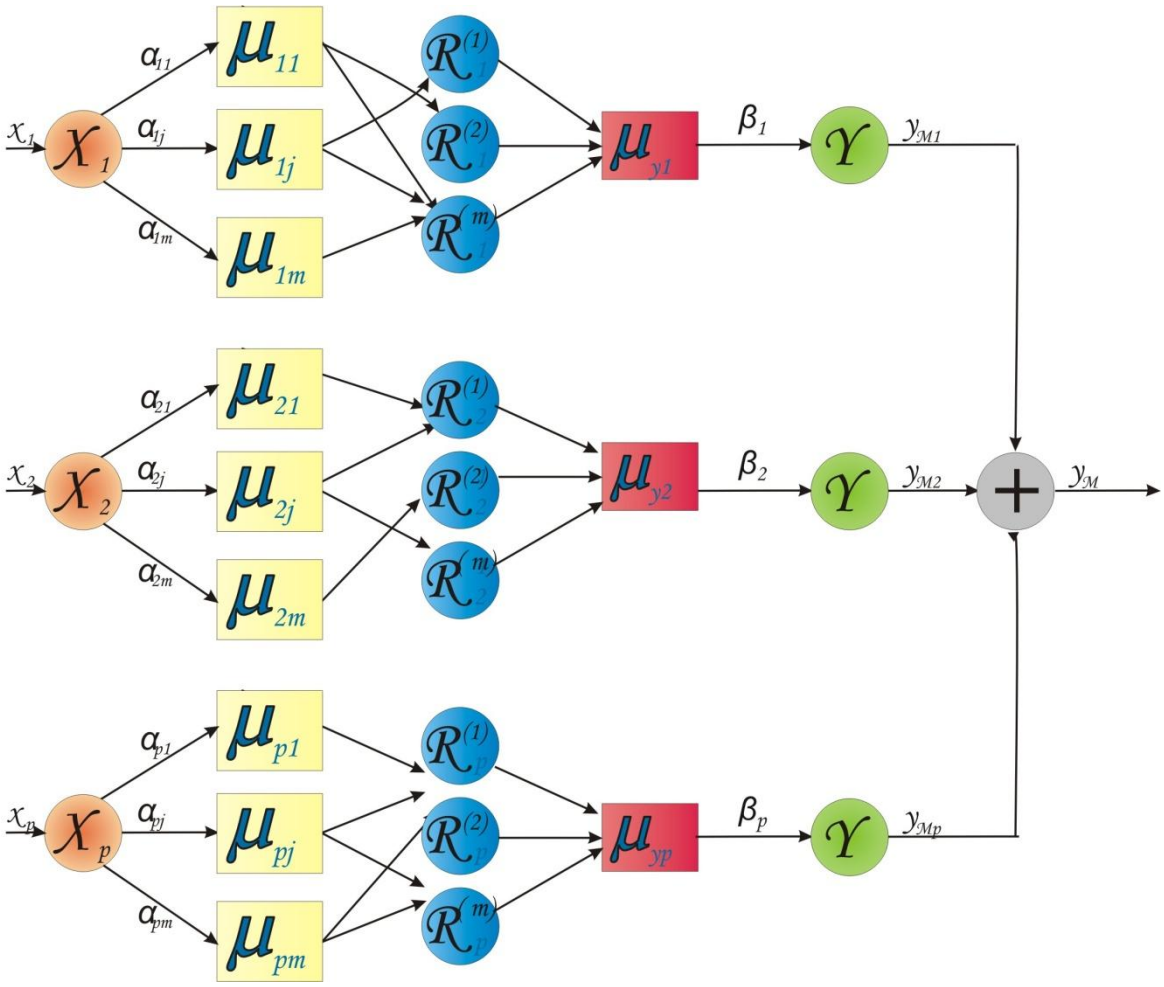
Слой 2: Тук се осъществява операцията размиване, като за целта е необходимо да се определи видът и броят на използваните функции на принадлежност. В конкретния случай са използвани три Гаусови функции на принадлежност, описани с формула (2.1.2).

Слой 3: Този слой представлява своеобразен генератор на правила, тъй като именно тук се формират размитите правила. Те имат формата, дадена с израз (2.1.3).

Слой 4: Реализира се операцията импликация. Тя се осъществява с помощта на формула (2.1.5).

Слой 5: Формират се изходите на отделните подмодели $\hat{y}_{M1}$, $\hat{y}_{M2}$, ... $\hat{y}_{Mq}$, съгласно изразите (2.1.6) и (2.1.7). С q е означен броят на използваните подмодели.

Слой 6: Получава се изходът на DANFA модела като сума от изходите на отделните подмодели $\hat{y}_{M1}$, $\hat{y}_{M2}$, ... $\hat{y}_{Mq}$.

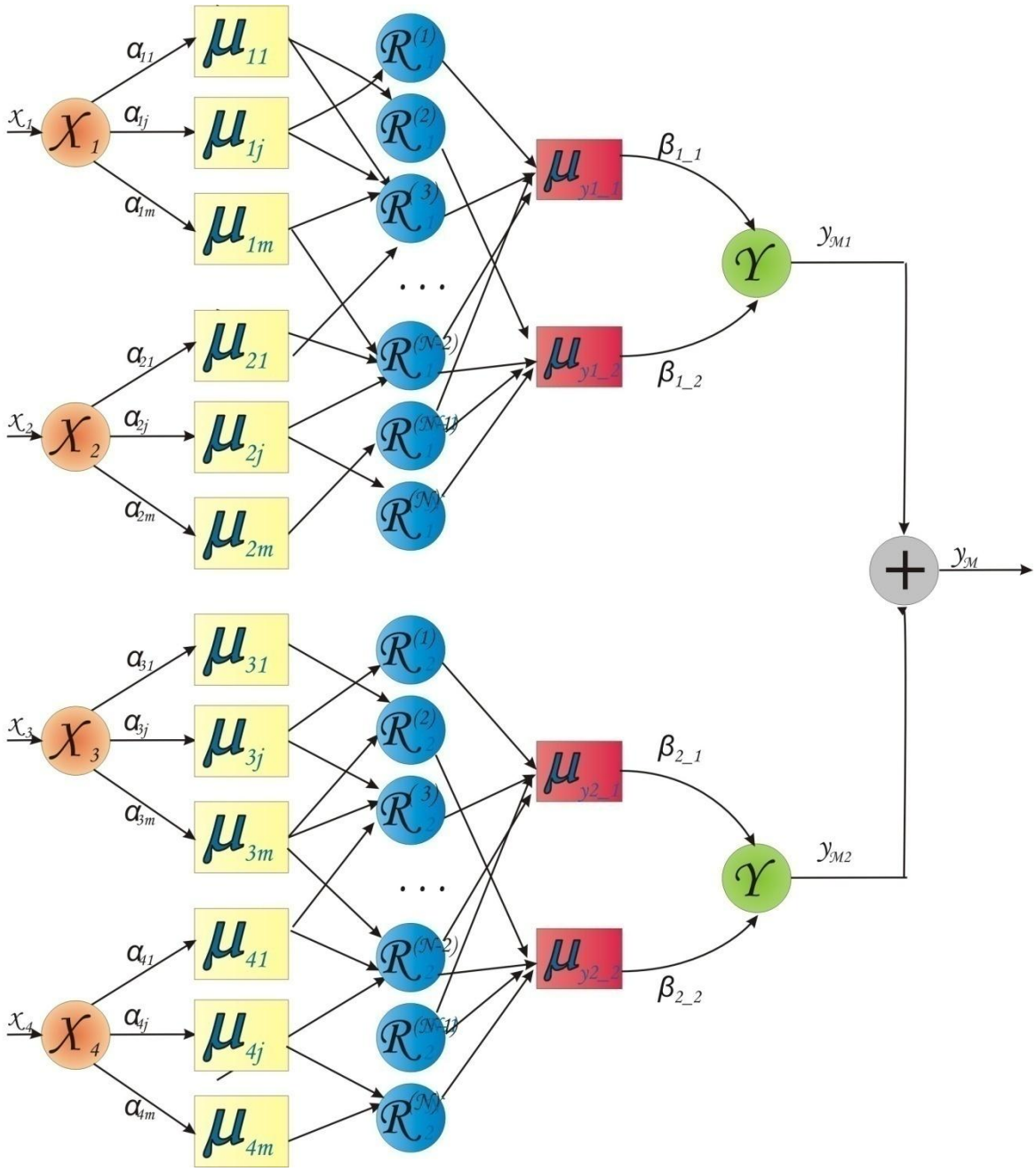


фиг. 3.1.1. Структура на DANFA модел

Основното предимство на DANFA модела е, че при него броят на размитите правила е значително редуциран в сравнение с модела, описан в т.2.1. При последния броят на размитите правила се определя от израза $N=m^{\mathbf{p}}$, докато при DANFA модела това става посредством формулата:

$$N = m_1^{p_1} + m_2^{p_2} + \ldots + m_q^{p_q} \qquad (3.1.1)$$

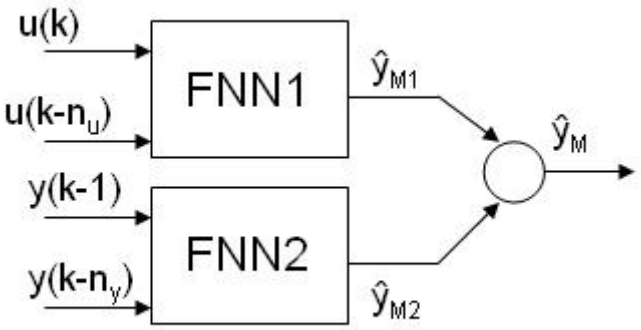
Така ако имаме 4 входа (две минали стойности на изхода, миналата и текуща стойност на управлението), размити с по три множества броят на правилата вече не е 81, а само 12. Това от своя страна ще намали драстично и броя на параметрите, които трябва да се определят на всеки такт при обучението. Тъй като общият брой параметри се определя от израза *N*(3P+1)*, то в този случай те ще бъдат само 156, а не 1053.



Фиг.3.1.2. Структура на DANFA модел с два подмодела

За целите на настоящото изследване разпределението на входните сигнали е направено на база тип на входните сигнали. При това положение моделът от фиг. 3.1.1 се трансформира до вида, показан на фиг. 3.1.2. В този случай DANFA моделът се състои от два подмодела, всеки от които се реализира чрез невронно-размита структура от типа, описан в т.2.1. За единия подмодел входове са стойностите на вектор-регресора **y** (т.е.

миналите стойности на изхода на обекта x1=y(k-1) и x2= y(k-2)), а за другия – стойностите на вектор-регресора на управлението **u**, т.е. x3=u(k) и x4=u(k-1). При тази постановка броят на размитите правила е 18, а на обучаваните параметри – 234. Обобщена блокова схема на този вариант на DANFA модела е представена на фиг. 3.1.3. В нея с FNN1 и FNN2 са означени двете невронно-размити структури от типа, представен на фиг. 2.1.1, които обработват стойностите на вектор-регресора на управлението **u** и стойностите на вектор-регресора **y**.

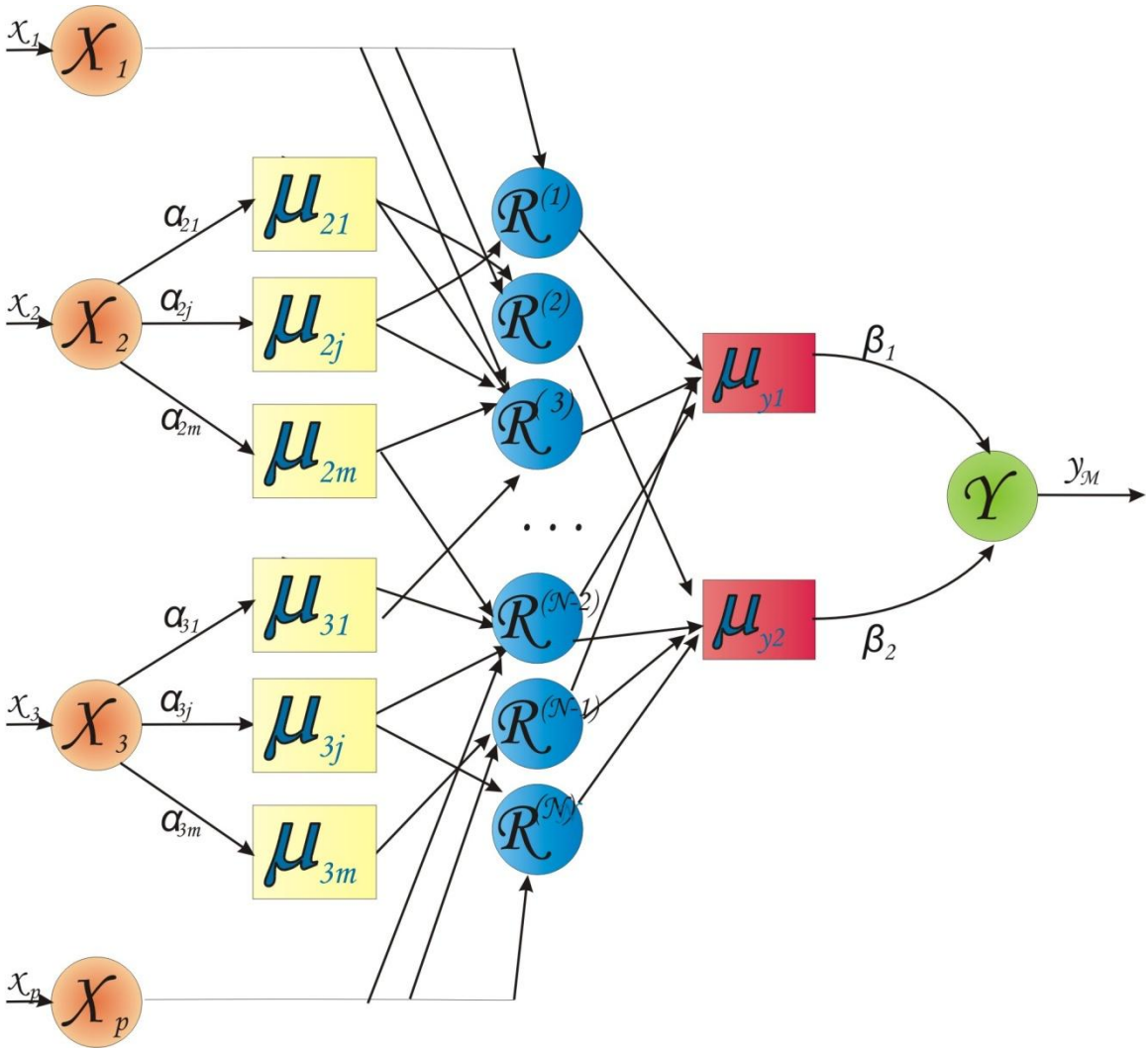


Фиг. 3.1.3. Обобщена блокова схема на DANFA модел

3.2. Невронно-размит модел с частично размиване на входните сигнали

Невронно-размитият модел с частично размиване на входните сигнали (Semi Fuzzy Neural Network - SFNN) е модификация на модела, описан в т.2.1. По същество той представлява петслойна структура с механизъм на извеждане на Такаги-Сугено. При SFNN модела [5] обаче част от входните сигнали не се размиват, а влизат с техните реални стойности, претеглени със съответен тегловен коефициент, директно в третия слой (слоят с размитите правила) на невронно-размитата структура (фиг.3.2.1), т.е. директно в следствената част на функциите на Сугено. По този начин от една страна се редуцира броят на размитите правила, с които моделът работи, а от друга – се намалява и броят на нелинейните параметри, които трябва да се определят в процеса на обучението му. Така например, ако

SFNN моделът има 4 входа (2 размити и 2 неразмити, които са съответно изходът на обекта и управлението), то тогава общият брой правила, които трябва да се генерират е 9. Общият брой параметри (линейни и нелинейни), които трябва да бъдат определени при обучението на модела е 64.



Фиг. 3.2.1. Структура на Невронно-размит модел с частично размиване на входните сигнали

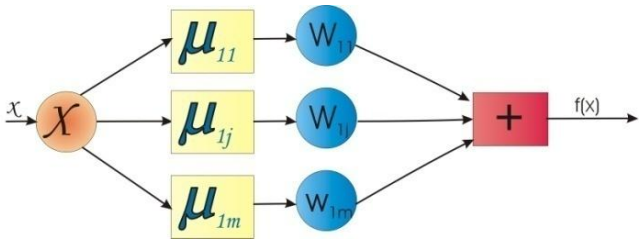
Освен това SFNN моделът се доближава по структура до линеен модел, поради което може да бъде наречен още полу- или квази- линеен. Това гарантира и намалена изчислителна тежест по време на оптимизационната процедура при изчисляване на стойността на управлението. Всичко това

прави SFNN моделът подходящ за работа в реално време и съответно за включването му в системи с НМПУ.

Основна трудност, която се явява при използването на SFNN модела, е начинът, по който да се определи кои от входните сигнали да бъдат размити и кои да се включат директно в следствената част на размитите правила. В търсене на подходящо решение на този проблем са разработени и изследвани ефективността и точността на моделиране на три разновидности на SFNN модела – SFNN Type I, при който се размиват миналите стойности на изхода на обекта, а стойностите на управлението влизат директно в следствената част на размитите правила, SFNN Type II, който е обратният вариант на SFNN Type I и SFNN Type III, наречен още модел с кръстосани връзки, при който са размити по една стойност от управлението и от изхода на обекта. Трябва да се подчертае, че тези три разновидности са разработени при допускането, че се реализира NNFARX модел с 4 входа, от които само два са размити. Резултатите, получени при предсказването на хаотични времеви серии с трите типа полу невронно-размити модели, са представени и анализирани в следващата глава.

3.3. Модифициран нео-размит модел

Нео-размитият неврон е предложен от проф. T. Yamakawa и екипът му (Yamakawa et al. , 1992). По същество той представлява радиално базисна невронна мрежа с механизъм на извеждане Такаги - Сугено от нулев ред. Структурата на нео-размития неврон е показана на фиг. 3.3.1.



Фиг. 3.3.1. Структура на отделен нео-размит неврон

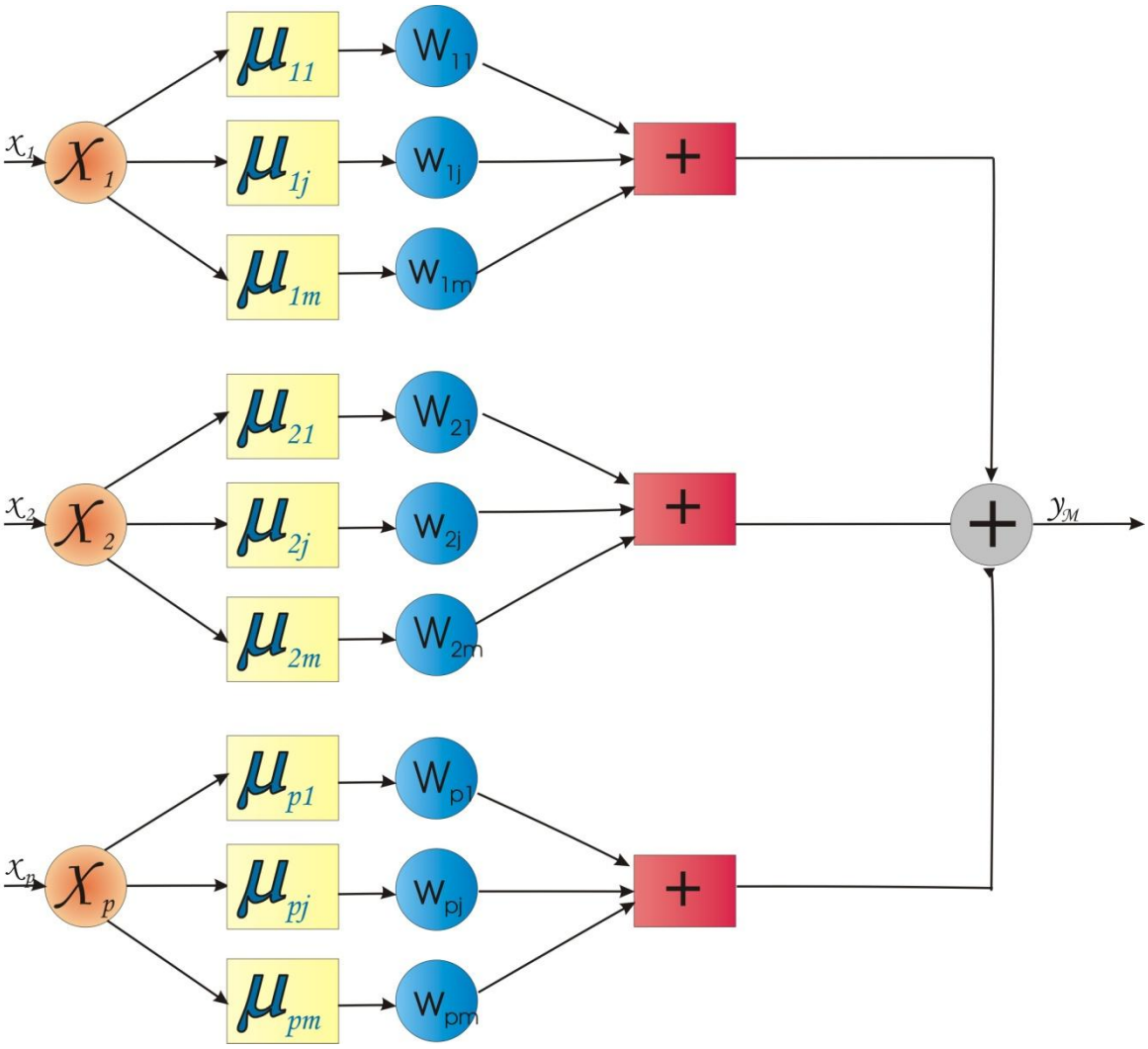
Изходът на нео-размития неврон се определя в съответствие с израза:

$$f(x) = \sum_{j=1}^{m} \mu_j(x(k)) w_j \qquad (3.3.1)$$

където x(k) е входният сигнал, w_j са тегловните коефициенти, а μ_j за j=1:m са степените на принадлежност към съответното размито множество. Размитите правила в този случай имат следната форма:

$$If\ x_i \text{ is } A_{ij}\ then \text{ the output is } f(x_i) \qquad (3.3.2)$$

Използвайки основната концепция на нео-размития неврон лесно може да се получи мрежа, съдържаща по-голям брой подобни неврони. Такава структура е представена на фиг. 3.3.2.



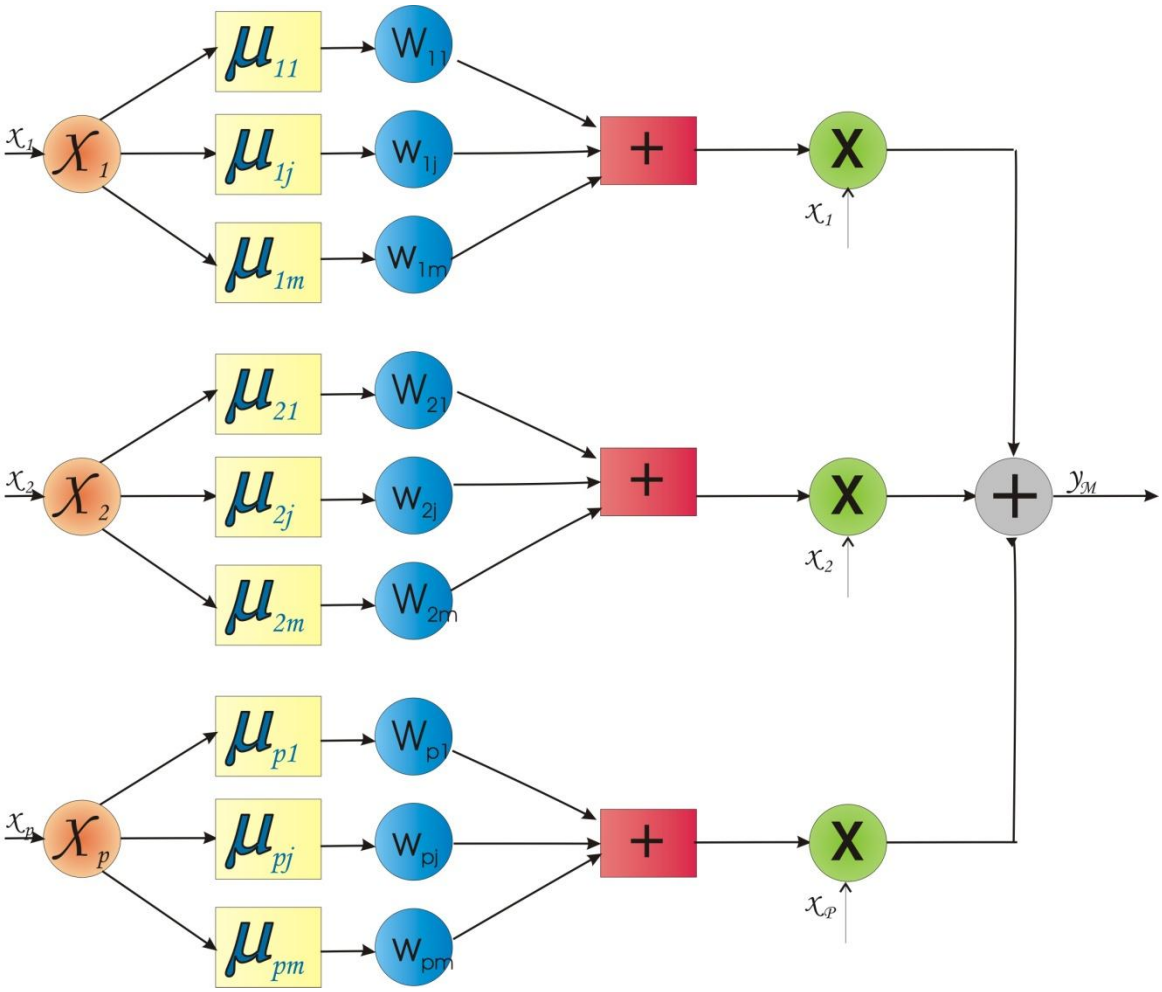
Фиг. 3.3.2. Структура на мрежа от нео-размити неврони

Изходът на подобна мрежа от нео-размити неврони се определя от израза:

$$y_m(k) = \sum_{i=1}^{p} f_i(x_i(k)) = \sum_{i=1}^{p}\sum_{j=1}^{m} \mu_{ij}(x_i(k)) w_{ij}(k) \qquad (3.3.3)$$

където **p** е броят на включените в мрежата неврони.

В настоящата дисертация е реализирана модификация на тази мрежа от нео-размити неврони, която е подходяща за целите на предсказващото управление. Структурата й е представена на фиг. 3.3.3.



Фиг. 3.3.3. Структура на модифициран нео-размит модел

От фиг. 3.3.3 става ясно, че модифицираният нео-размит модел представлява шестслойна архитектура, като отделните слоеве изпълняват както следва:

Слой 1: Невроните от този слой приемат входните сигнали и ги предават към следващия слой.

Слой 2: В него се реализира операцията размиване, като за целта е необходимо да се определи видът и броят на използваните функции на принадлежност. В конкретния случай са използвани три Гаусови функции на принадлежност, описани с формула (2.1.2).

Слой 3: В този слой се осъществява умножение на стойностите, получени от операцията размиване в предходния слой и тегловните коефициенти от размитите правила:

$$f_{ij}(k) = \mu_{ij}(k) * w_{ij}(k) \tag{3.3.4}$$

Слой 4: Осъществява се сумиране на изчислените в слой 3 стойности като се използва изразът (3.3.1).

Слой 5: Реализира се умножение на изходите на всеки нео-размит неврон с входния за него сигнал.

Слой 6: Получава се изходът на модифицирания нео-размит модел като сума от изходите на отделните неврони умножени по съответния входен сигнал, т.е.:

$$y_m(k) = \sum_{i=1}^{p} x_i(k)\, f_i(x_i(k)) = \sum_{i=1}^{p} x_i(k) \sum_{j=1}^{m} \mu_{ij}(x_i(k)) w_{ij}(k) \tag{3.3.5}$$

Модифицираният нео-размит модел е реализиран в многомерен вариант [4] като за целта е използвана структурата, представена на фиг.3.3.4. При него първият слой приема входните сигнали и ги предава директно към втория слой, където се осъществява операцията размиване. В третия слой получените степени на принадлежност се умножават по две групи тегловни коефициенти $w_{ji}(k)$ и $v_{ji}(k)$. В четвъртия слой се изчисляват две групи функции:

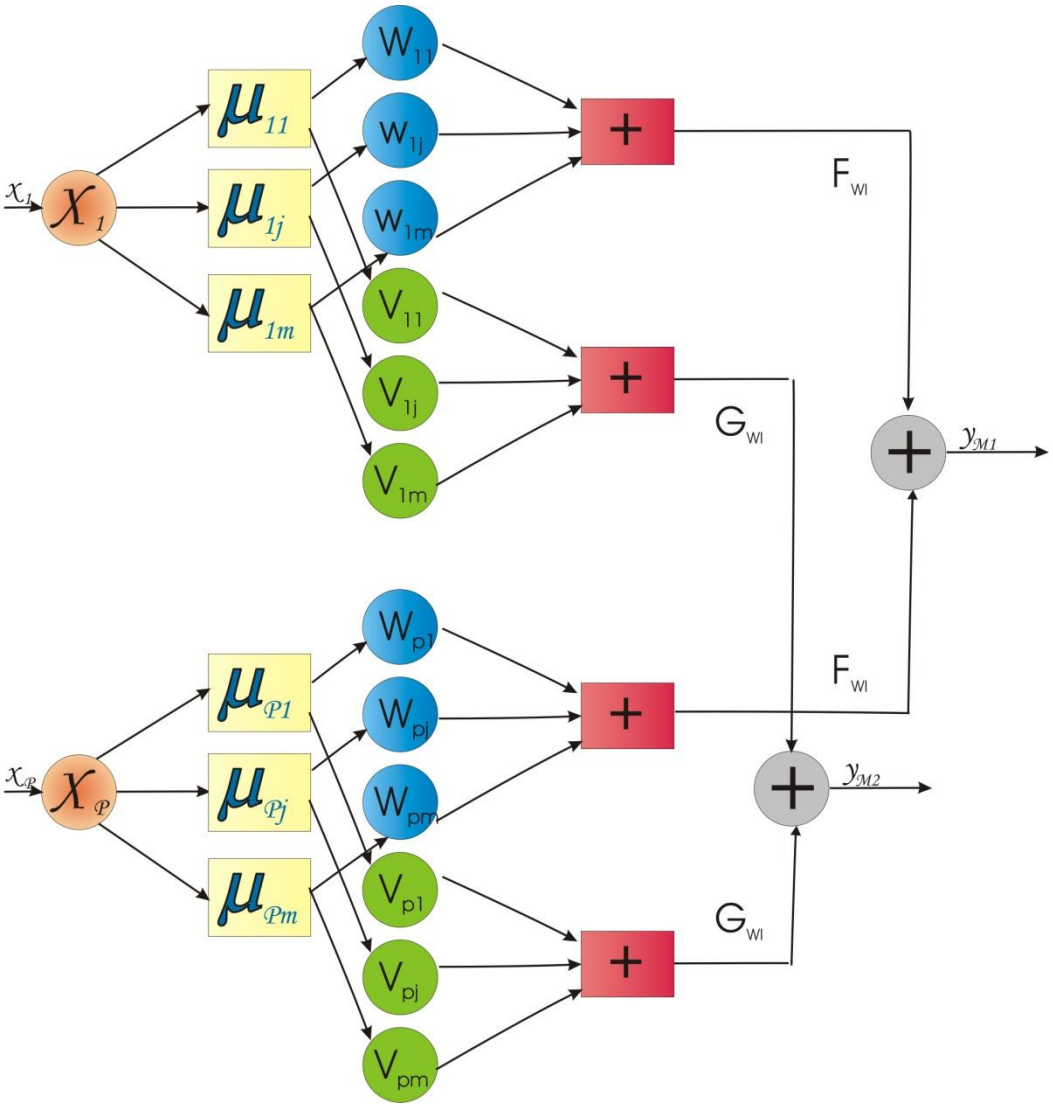
$$F_w(x) = \sum_{j=1}^{m} \mu_j(x(k)) w_j \tag{3.3.6}$$

$$G_w(x) = \sum_{j=1}^{m} \mu_j(x(k)) v_j \tag{3.3.7}$$

Изходите на многомерния нео-размит модел се получават в последния пети слой като се използват следните зависимости:

$$y_{m1} = \sum_{j=1}^{p} F_{wj}(x) \tag{3.3.8}$$

$$y_{m2} = \sum_{j=1}^{p} G_{wj}(x) \tag{3.3.9}$$



Фиг. 3.3.3. Структура на многомерен модифициран нео-размит модел

Обучението на многомерния нео-размит модел е направено с описания в т.3.6.1 двустъпков градиентен алгоритъм. В случая обаче е необходимо да се минимизират две моментни грешки, при което

актуализирането на двете групи тегловни коефициенти $w_{ji}(k)$ и $v_{ji}(k)$ става посредством изразите:

$$w_{ij}(k+1) = w_{ij}(k) + \Delta w = w_{ij}(k) - \eta\left(\frac{\partial E_1(k)}{\partial w(k)}\right) \\ = w_{ij}(k) + \eta e(k)\mu_{ij}(x_i(k)) \qquad (3.3.10)$$

$$v_{ij}(k+1) = v_{ij}(k) + \Delta v = v_{ij}(k) - \eta\left(\frac{\partial E_2(k)}{\partial v(k)}\right) \\ = v_{ij}(k) + \eta e(k)\mu_{ij}(x_i(k)) \qquad (3.3.11)$$

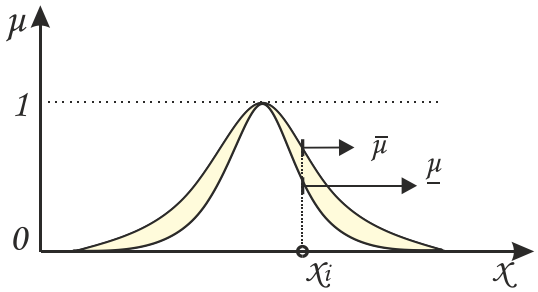
Въведена е адаптивна скорост на обучение **η**, чиято оптимална стойност се определя чрез следната зависимост:

$$\eta(k) = 0.1 * \left\| \mu_{ij}(x_i(k)) \right\|^2 \qquad (3.3.12)$$

3.4 Реализация на предложените невронно-размити модели с Тип 2 размита логика

През последните години в хибридните невронно-размити структури широко започва да се използва Тип 2 размита логика вместо класическата, известна още като Тип 1 размита логика. Тя е предложена от (Zadeh, 1975) в отговор на непрекъснатата критика, че размитатата логика Тип 1 не може да се справи в условията на неопределености. Неговата теория впоследствие е развита от (Karnik and Mendel, 1998).

Използваните за целите на дисертацията функции на принадлежност имат видът, представен на фиг.3.4.1, т.е центърът на гаусовата функция е един, но ширината е различна, при което се получават съответно и две степени на принадлежност в съответствие с израза (3.4.1). Оцветената област на фиг.3.4.1 се нарича област на неопределеност (Footprint of uncertainty - FOU).



Фиг. 3.4.1. Тип 2 Гаусово размито множество

$$\mu_{ij}(x_i) = \exp\left(\frac{-(x_i - c_{ij})^2}{2\sigma_{ij}^{\ 2}}\right) = \begin{cases} \bar{\mu}_{ij} \ за\ \sigma_{ij} = \bar{\sigma}_{ij} \\ \underline{\mu}_{ij} \ за\ \sigma_{ij} = \underline{\sigma}_{ij} \end{cases} \tag{3.4.1}$$

т.е. имаме степен на принадлежност $\bar{\mu}$ към горното множество и степен на принадлежност $\underline{\mu}$ към долното множество. При това положение изразът (2.1.8) за размита импликация се трансформира до вида:

$$\mu_{ij}^{\ *}(x_i) = \begin{cases} \bar{\mu}_{ij}^{*} = \prod_{i=1}^{n} \bar{\mu}_{ij} \\ \underline{\mu}_{\text{ij}}^{*} = \prod_{i=1}^{n} \underline{\mu}_{ij} \end{cases} \tag{3.4.2}$$

Стойността на изхода на модела при Тип 2 невронно-размита структура се определя съгласно следните формули:

$$\hat{y}_M(k+j) = \frac{1}{2}\left(\frac{\sum_{i=1}^{q} \bar{f}_y^{(i)}(k+j)\bar{\mu}_y^{(i)}(k+j)}{\sum_{i=1}^{q} \bar{\mu}_y^{(i)}(k+j)} + \frac{\sum_{i=1}^{q} \underline{f}_y^{(i)}(k+j)\underline{\mu}_y^{(i)}(k+j)}{\sum_{i=1}^{q} \underline{\mu}_y^{(i)}(k+j)}\right) \tag{3.4.3}$$

$$\hat{y}_M(k+j) = \frac{1}{2}\left(\sum_{i=1}^{q} \bar{f}_y^{(i)}(k+j)\bar{\mu}_y^{*(i)}(k+j) + \sum_{i=1}^{q} \underline{f}_y^{(i)}(k+j)\underline{\mu}_y^{*(i)}(k+j)\right) \tag{3.4.4}$$

От горните изрази става ясно, че при използване на Тип 2 невронно-размита структура е необходимо на всеки такт да се определят по-голям брой параметри, което се явява техен недостатък и „цена”, която трябва да се плати, за да се получи по-точно моделиране в условията на неопределености.

3.5 Реализация на предложените невронно-размити модели с Интуиционистка размита логика

Интуиционисната размита логика е друг начин за справяне с неопределеностите в системата. Теорията на Atanassov може да се разглежда като обобщение на класическата Тип 1 размита логика. Характерна особеност на Интуиционистката размита логика е, че при нея освен степен на принадлежност μ(x) се определя и степен на непринадлежност ν(x):

$$\mu_{ij}(x_i) = \exp\left(\frac{-(x_i - c_{ij})^2}{2\sigma_{ij}^{\ 2}}\right) \tag{3.5.1}$$

$$\nu_{ij}(x_i) = \left(1 - \exp\left(\frac{-(x_i - c_{ij})^2}{2\sigma_{ij}^{\ 2}}\right)\right)^k, k \geq 1 \tag{3.5.2}$$

където x_i е входният сигнал, i=1:p е броят на входовете на интуиционисната невронно-размита структура, c_{ij} и σ_{ij} са центърът и ширината на Гаусовите функции на принадлежност, j=1:m като m е броят на използваните функции на принадлежност, k е параметър, който трябва да бъде избран (Zhou, 2013).

Определя се също така и област на неопределеност като се използва следната зависимост:

$$\pi(x) = 1 - \mu(x) - \nu(x) \tag{3.5.3}$$

Този параметър определя степента на неопределеност, т.е. дали x принадлежи или не към съответното размито множество и е очевидно, че $0 \leq \pi_A \leq 1$ за всяко x. Ако k=1, тогава $\mu_A + \nu_A = 1$ и зоната на неопределеност π_A също ще е равна на нула.

При интуиционистки невронно-размити модели изходът се определя в съответствие с:

$$\hat{y}_M(k+1) = (1 - \pi(x))y_\mu + \pi(x)y_\nu \tag{3.5.4}$$

където y_μ и y_ν се изчисляват както следва:

$$y_{\mu}(k+j)=\frac{\sum_{i=1}^{q} f_{\mu}^{(i)}(k+j)\mu_{y}^{(i)}(k+j)}{\sum_{i=1}^{q}\mu_{y}^{(i)}(k+j)} \tag{3.5.5}$$

$$y_{\nu}(k+j)=\frac{\sum_{i=1}^{q} f_{\nu}^{(i)}(k+j)\nu_{y}^{(i)}(k+j)}{\sum_{i=1}^{q}\nu_{y}^{(i)}(k+j)} \tag{3.5.6}$$

Иначе казано y_{μ} и y_{ν} се определят както при невронно-размита система, използваща класическата Тип 1 размита логика. Разликата е, че това се прави както по отношение на степента на принадлежност μ_A, така и по отношение на степента на непринадлежност ν_A. И тук, както и при Тип 2 невронно-размитите структури, броят на параметрите, които следва да се определят по време на обучението е завишен. Като съществено предимство на интуиционистките невронно-размити модели пред тези с Тип 2 размита логика може да се изтъкне, че не е необходимо двукратно реализиране на операцията размиване. След като веднъж вече е опрелена степента на принадлежност, то от израза (3.5.2) лесно може да бъде определена степента на непринадлежност.

3.6 Алгоритми за обучение на предложените невронно-размити модели

Безспорно когато става дума за обучение на невронни мрежи и/или невронно-размити структури най-популярният метод е този с обратно разпространение на грешката. По същество той е градиентен метод от първи ред и като такъв има някои недостатъци – бавна сходимост и възможност от засядане в локален екстремум. С цел преодоляване на тези недостатъци в настоящата дисертация за обучението на предложените DANFA, SFNN и модифициран нео-размит модели освен алгоритъмът с обратно разпространение на грешката са разработени също така рекурентен метод на най-малките квадрати, метод на Нютон и метод на Левенберг-Маркгуард.

3.6.1 Двустъпков градиентен алгоритъм

Двустъпковият градиентен алгоритъм представлява модифициран алгоритъм с обратно разпространение на грешката. Основната разлика между двата алгоритъма е в броя на настройваемите параметри. При класическия алгоритъм за обучение грешката се разпространява в обратна посока последователно през всички слоеве на невронната (невронно-размита) мрежа и се актуализират теглата на връзките във всички слоеве. При двустъпковия градиентен алгоритъм се актуализират теглата на връзките само в два от слоевете. Това са т.нар. параметрични слоеве, които за структурата на фиг. 2.1.1 са съответно втори и трети слой.

Обучението с двустъпковия градиентен метод се основава на минимизирането на моментната моделна грешка (грешка между изхода на обекта и изхода на невронно-размития модел):

$$\varepsilon(k) = y(k) - \hat{y}_M(k) \tag{3.6.1}$$

където y е реалната стойност на изходната величина, а $\hat{y}$ е стойността на изхода на обекта изчислена от невронно-размития модел (DANFA, SFNN или модифицирания нео-размит модел).

Формираният целеви критерий е:

$$E = \frac{\varepsilon^2(k)}{2} \tag{3.6.2}$$

Най-напред се актуализират теглата на връзките в третия слой. Това са коефициентите в следствената част на размитите правила. За да се опрости записването въвеждаме векторът $\boldsymbol{\beta}$, в който са включени търсените параметри:

$$\begin{aligned}\boldsymbol{\beta} &= \{a_1^{(n)},\ldots,a_{ny}^{(n)},b_0^{(n)},b_1^{(n)}\ldots,b_{nu}^{(n)}\} = \\ &= \{\beta_1^{(n)},\beta_2^{(n)},\ldots\beta_{ny}^{(n)},\beta_{ny+1}^{(n)},\ldots,\beta_{ny+nu+1}^{(n)}\}\end{aligned} \tag{3.6.3}$$

По този начин обобщеното обучаващо правило за коефициентите във функцията на Сугено може да се представи със следния израз:

$$\beta(k+1) = \beta(k) + \eta(k)\left(-\frac{\partial E}{\partial \beta}\right) \tag{3.6.4}$$

където с $\eta(к)=\eta=const$ е означена скоростта на обучение. Производната от обучаващото правило може да се представи по следния начин:

$$\frac{\partial E}{\partial \boldsymbol{\beta}} = \frac{\partial E}{\partial \hat{y}} . \frac{\partial \hat{y}}{\partial \boldsymbol{\beta}} \tag{3.6.5}$$

където:

$$\frac{\partial E}{\partial \hat{y}} = -(y(k) - \hat{y}(k)) = -\varepsilon(k) \tag{3.6.6}$$

Стойността на втората частна производна е различна за свободния член и останалите коефициенти от функцията на Сугено и се определя съгласно изразите:

- за свободния член

$$\frac{\partial \hat{y}}{\partial \boldsymbol{\beta}_{0j}} = \bar{\mu}_y^{(n)}(k) \tag{3.6.7}$$

- за останалите коефициенти

$$\frac{\partial \hat{y}}{\partial \boldsymbol{\beta}_{ij}} = \bar{\mu}_y^{(n)}(k) x_p(k) \tag{3.6.8}$$

Замествайки (3.6.7) и (3.6.8) в (3.6.4) се получават окончателните изрази за параметрите в следствието:

$$\boldsymbol{\beta}_{0j}(k+1) = \boldsymbol{\beta}_{0j}(k) + \eta(k)\varepsilon(k)\bar{\mu}_y^{(n)}(k) \tag{3.6.9}$$

$$\boldsymbol{\beta}(k+1) = \boldsymbol{\beta}(k) + \eta(k)\varepsilon(k)\bar{\mu}_y^{(n)}(k) x_p(k) \tag{3.6.10}$$

Втората група настройваеми параметри са тези на размитите Гаусови множества $c_{Xp,m}$ и $\sigma_{Xp,m}$. Отново с цел опростяване на записа се въвежда векторът α, в който са включени нелинейните параметри:

$$\begin{aligned} \boldsymbol{\alpha} &= \{c_{11}^{(n)}, \ldots, c_{pm}^{(n)}, \sigma_{11}^{(n)}, \ldots, \sigma_{pm}^{(n)}\} = \\ &= \{\alpha_1^{(n)}, \alpha_2^{(n)}, \ldots \alpha_{pm}^{(n)}, \alpha_{pm+1}^{(n)}, \ldots, \alpha_{2pm}^{(n)}\} \end{aligned} \tag{3.6.11}$$

За определянето им се използва аналогичен на (3.6.4) израз:

$$\boldsymbol{\alpha}(k+1)=\boldsymbol{\alpha}(k)+\eta(k)\left(-\frac{\partial E}{\partial \boldsymbol{\alpha}}\right) \tag{3.6.12}$$

където производната може да се представи като:

$$\frac{\partial E}{\partial \boldsymbol{\alpha}}=\frac{\partial E}{\partial \hat{y}}\cdot\frac{\partial \hat{y}}{\partial \mu_{xp,m}}\cdot\frac{\partial \mu_{xp,m}}{\partial \boldsymbol{\alpha}} \tag{3.6.13}$$

Първата частна производна от (3.6.13) се определя с израза (3.6.6), а останалите две са както следва:

$$\frac{\partial \hat{y}}{\partial \mu_{xp,m}}=\frac{f_y(k)-\hat{y}(k)}{\sum_{i=1}^{q}\mu_{yq}^{(n)}(k)} \tag{3.6.14}$$

- за центъра $c_{Xp,m}$ на размитите множества

$$\frac{\partial \mu_{xp,m}}{\partial \boldsymbol{\alpha}}=2\mu_{xp,m}\frac{(x_p-c_{pm})}{\sigma_{pm}^2} \tag{3.6.15}$$

- за ширината $\sigma_{Xp,m}$ на размитите множества

$$\frac{\partial \mu_{xp,m}}{\partial \boldsymbol{\alpha}}=\mu_{xp,m}\frac{(x_p-c_{pm})^2}{\sigma_{pm}^3} \tag{3.6.16}$$

Заместващи изразите (3.6.6), (3.6.14), (3.6.15) и (3.1.16) в (3.6.12) се получава окончателният рекурентен израз за всеки един входен параметър за настройка (център $c_{Xp,m}$ или ширина $\sigma_{Xp,m}$):

$$\begin{aligned} c_{xp,m}(k+1)&=c_{xp,m}(k)+\ldots \\ &+\eta\varepsilon(k)\bar{\mu}_y^{(n)}(k)\left[f_y^{(n)}(k)-\hat{y}(k)\right]\frac{\left[x_p(k)-c_{xp,m}(k)\right]}{\sigma_{xp,m}^2(k)} \end{aligned} \tag{3.6.17}$$

$$\sigma_{xp,m}(k+1)=\sigma_{xp,m}(k)+\ldots$$
$$+\eta\varepsilon(k)\bar{\mu}_y^{(n)}(k)\left[f_y^{(n)}(k)-\hat{y}(k)\right]\frac{\left[x_p(k)-c_{xp,m}(k)\right]^2}{\sigma_{xp,m}^3(k)} \qquad (3.6.18)$$

Съществена особеност при обучението на DANFA модела, е че то се прави за всеки отделен подмодел (невронно-размита мрежа), влизащ в неговата структура. За целта се използват изразите от (3.6.1) до (3.6.18). Също въз основа на тези формули се осъществява обучението на SFNN модела и на модифицирания нео-размит модел. При обучението на SFNN модела трябва да се има предвид, че броят на нелинейните параметри, които трябва да се определят, е по-малък. Причината за това е, че част от входните сигнали не се размиват и следователно за тях отпада необходимостта да се определят параметрите на функциите на принадлежност. При обучението на модифицирания нео-размит модел броят на параметрите в следствената част е по-малък, тъй като функцията на Сугено е от нулев ред. Тези особености при обучението на трите модела са в сила и за останалите описани в дисертацията алгоритми за обучение.

3.6.2 Рекурентен метод на най-малките квадрати

Методът на най-малките квадрати (МНК) е един от най-широко използваните методи за оценка на параметрите на динамични модели. В настоящата дисертация той е използван, за да се реализира хибриден алгоритъм за обучение, при който посредством рекурентен МНК (РМНК) е направена оценка на параметрите във функцията на Сугено (2.1.4), а чрез алгоритъма с обратно разпространение на грешката се актуализират параметрите на Гаусовите функции на принадлежност (център и ширина).

За реализацията му първо се определят параметрите във функцията на Сугено. Това налага преобразуване на израза (2.1.4) в следната форма:

$$y(k)=a_1^{(i)}y(k-1)+a_2^{(i)}y(k-2)+\cdots+a_{ny}^{(i)}y(k-n_y)+\ldots$$
$$+b_1^{(i)}u(k)+b_2^{(i)}u(k-1)+\cdots+b_{nu}^{(i)}u(k-n_u)+b_0^{(i)} \qquad (3.6.18)$$

Последният израз от своя страна може да бъде записан във векторна форма по следния начин:

$$y(k) = \psi^T(k)\theta \tag{3.6.19}$$

където:

$$\theta = [b_0, b_1, \ldots, b_m, a_1, a_2, \ldots, a_n]^T$$

$$\psi(k) = [u(k-1), \ldots, u(k-n_u), y(k-1), \ldots, y(k-n_y)]^T$$

При подобна постановка рекурсивната оценка на параметрите по МНК става с използването на следните изрази:

$$\hat{\theta}(k) = \hat{\theta}(k-1) + K(k)[y(k) - \psi^T(k)\hat{\theta}(k-1)] \tag{3.6.20}$$

$$K(k) = P(k-1)\psi(k)[I + \psi^T(k)P(k-1)\psi(k)]^{-1} \tag{3.6.21}$$

$$\begin{aligned} P(k) &= P(k-1) - K(k)\psi^T(k)P(k-1) = \\ &= P(k-1) - P(k-1)\psi(k)[I + \psi^T(k)P(k-1)\psi(k)]^{-1}\psi^T(k)P(k-1) \end{aligned} \tag{3.6.22}$$

За определянето на параметрите от предпоставката на размитите правила се използват изразите (3.6.11 – 3.6.18).

РМНК има сравнително проста и лесна реализация. Оценката се получава посредством прибавянето на претеглената предсказана грешка към предходната стойност на оценката φ. Векторът K(k) се нарича Калманово усилване, а P(k) представлява симетрична ковариационна матрица. Изборът на началната й стойност е свързан с неопределеността в стойностите на оценките на параметрите.

3.6.3 Алгоритми за обучение, базирани на градиентни методи от втори ред

От теорията на оптимизацията е известно, че гредиентните методите от втори ред са по-бързо сходими от тези от първи ред. Освен това при тях отпада възможността от засядане в локален екстремум. Ето защо за целите на настоящата дисертация са разработени метод на Нютон и метод на Левенберг-Маркгуард. По същество представените алгоритми са хибридни, тъй като представляват комбинация от градиентен метод от втори ред (Нютон или Левенберг-Маркгуард метод) и алгоритъм с обратно

разпространение на грешката, т.е. параметрите във функцията на Сугено се определят посредством градиентен метод от втори ред, а нелинейните параметри на финкциите на принадлежност – с градиентен метод от първи ред. Това е направено с цел да се избегнат възможни осцилации при използването на алгоритъм за обучение, основан изцяло на градиентен метод от втори ред.

Обучението с градиентни алгоритми от втори ред се реализира въз основа на минимизирането на моментната моделна грешка (3.6.1). Този тип алгоритми се основават на квадратична апроксимация и това налага изчисляването на втората производна на целевия критерий (3.6.2). Актуализирането на стойностите в следствената част на размитите правила по метода на Нютон става чрез трансформирането на изразите (3.6.9) и (3.6.10) до следния вид:

$$\beta_{0j}(k+1)=\beta_{0j}(k)+\Delta\beta_{0j}(k) \tag{3.6.23}$$

$$\beta_{ij}(k+1)=\beta_{ij}(k)+\Delta\beta_{ij}(k) \tag{3.6.24}$$

където

$$\Delta\beta_{0j}(k)=-\left[J^T(\beta_{0j})J(\beta_{0j})\right]^{-1}J^T(\beta_{0j})\varepsilon(k) \tag{3.6.25}$$

$$\Delta\beta_{ij}(k)=-\left[J^T(\beta_{ij})J(\beta_{ij})\right]^{-1}J^T(\beta_{ij})\varepsilon(k) \tag{3.6.26}$$

и тъй като якобиянът за свободния член и за останалите коефициенти има вида:

$$J^T(\beta_{0j})=\bar{\mu}_y^{(j)} \qquad J(\beta_{0j})=\left[\bar{\mu}_y^{(j)}\right]^T \qquad J^T(\beta_{ij})=\bar{\mu}_y^{(j)}x_i \qquad J(\beta_{ij})=\left[\bar{\mu}_y^{(j)}x_i\right]^T$$

то окончателно изразите (3.6.23) и (3.6.24) приемат следната форма:

$$\beta_{0j}(k+1)=\beta_{0j}(k)+\eta\left[\bar{\mu}_y^{(j)}(\bar{\mu}_y^{(j)})^T\right]^{-1}\varepsilon(k)\bar{\mu}_y^{(j)} \tag{3.6.27}$$

$$\beta_{ij}(k+1)=\beta_{ij}(k)+\eta\left[\bar{\mu}_y^{(j)}x_i(k)(\bar{\mu}_y^{(j)}x_i(k))^T\right]^{-1}\varepsilon(k)\bar{\mu}_y^{(j)}x_i(k) \tag{3.6.28}$$

За определянето на параметрите от предпоставката на размитите правила се използват изразите (3.6.11 – 3.6.18).

По аналогичен начин се получават рекурентните зависимости за параметрите във функцията на Сугено при използване на алгоритъма на Левенберг-Маркгуард. В този случай изразите (3.6.25) и (3.6.26) се преобразуват до вида:

$$\Delta\beta_{0j}(k) = -\left[J^T(\beta_{0j})J(\beta_{0j}) + \lambda I\right]^{-1} J^T(\beta_{0j})\varepsilon(k) \qquad (3.6.29)$$

$$\Delta\beta_{ij}(k) = -\left[J^T(\beta_{ij})J(\beta_{ij}) + \lambda I\right]^{-1} J^T(\beta_{ij})\varepsilon(k) \qquad (3.6.30)$$

където I е единична матрица, а параметърът λ регулира направлението на движение към екстремума. И тъй като стойността на якобияна е същата както при алгоритъма на Нютон, то окончателните изрази за определяне на параметрите в следствената част на размитите правила са както следва:

$$\beta_{0j}(k+1) = \beta_{0j}(k) + \eta\left[\bar{\mu}_y^{(j)}(\bar{\mu}_y^{(j)})^T + \lambda I\right]^{-1}\varepsilon(k)\bar{\mu}_y^{(j)} \qquad (3.6.31)$$

$$\beta_{ij}(k+1) = \beta_{ij}(k) + \eta\left[\bar{\mu}_y^{(j)}x_i(k)(\bar{\mu}_y^{(j)}x_i(k))^T + \lambda I\right]^{-1}\varepsilon(k)\bar{\mu}_y^{(j)}x_i(k) \qquad (3.6.32)$$

За определянето на параметрите от предпоставката на размитите правила отново се използват изразите (3.6.11 – 3.6.18).

3.7. Оптимизационни алгоритми за определяне стойността на управляващия сигнал

Методът за оптимизация е от особено значение при МПУ, тъй като той оказва съществено влияние върху изчислителната ефективност на оптимизатора. В настоящата дисертация изборът пада върху алгоритми от втори ред, а именно тези на Нютон и на Левенберг-Маркгуард, тъй като при тях е намален броят на итерациите, свързани със сходимостта на алгоритъма, а това от своя страна води до по-голямо бързодействие на предсказващия регулатор [2].

3.7.1. Метод на Нютон

Както е известно, методът на Нютон се основава на квадратичната апроксимация на целева функция както следва:

$$\tilde{J}(x) = J(u^{(k)}) + \nabla^T J(u^{(k)})\Delta u^{(k)} + \frac{1}{2}(u^{(k)})^T \nabla^2 J(u^{(k)})\Delta u^{(k)} \quad (3.7.1)$$

От нея директно могат да бъдат изведени следните рекурентни зависимости за определяне стойността на управлението:

$$u^{(k+1)} = u^{(k)} - \left[\nabla^2 J(u^{(k)})\right]^{-1} \nabla J(u^{(k)}) \quad (3.7.2)$$

$$u^{(k+1)} = u^{(k)} - H^{-1}(u^{(k)})\nabla J(u^{(k)}) \quad (3.7.3)$$

където H е матрицата на Хесе, съдържаща частни производни от втори ред. За да бъде приложим методът на Нютон е необходимо целевата функция да е квадратична, а матрицата на Хесе да е положително определена.

При използване на квадратична целева функция и при липса на наложени ограничения оптимизацията може да бъде осъществена отново аналитично. В конкретния случай е използван стандартният GPC-критерий (1.4.1), като е прието, че не са наложени ограничения и следователно задачата за оптимизация по метода на Нютон може да се реши аналитично. За целта е необходимо да бъде изчислен вектор градиентът съгласно израза (2.2.2). Всеки елемент от вектор градиента може да бъде определен в съответствие с (2.2.3). От последния израз става ясно, че е необходимо да се изчислят две групи частни производни. Първата група производни могат да бъдат записани в матрицата (2.2.4), а втората група – в (2.2.11). При метода на Нютон е необходимо да се определят вторите частни производни на целевия критерий (1.4.1), а това става чрез преобразуването на (2.2.2) по следния начин:

$$H[k,U(k)] = [\frac{\partial^2 J[k,U(k)]}{\partial u^2(k)}, \frac{\partial^2 J[k,U(k)]}{\partial u^2(k+1)}, \ldots, \frac{\partial^2 J[k,U(k)]}{\partial u^2(k+N_u-1)}] \quad (3.7.4)$$

Всеки елемент от този вектор се определя както следва:

$$H[k,U(k)]=\left[-2\left[R(k)-\hat{Y}(k)\right]^T\frac{\partial\hat{Y}(k)}{\partial U(k)}+2\rho\hat{U}(k)^T\frac{\partial\hat{U}(k)}{\partial U(k)}\right]'= \tag{3.7.5}$$

$$=\begin{bmatrix}-2\left(\left[R(k)-\hat{Y}(k)\right]^T\right)'\frac{\partial\hat{Y}(k)}{\partial U(k)}-2\left[R(k)-\hat{Y}(k)\right]^T\left(\frac{\partial\hat{Y}(k)}{\partial U(k)}\right)'+\\ +2\rho\left(\hat{U}(k)^T\right)'\frac{\partial\hat{U}(k)}{\partial U(k)}+2\rho\hat{U}(k)^T\left(\frac{\partial\hat{U}(k)}{\partial U(k)}\right)'\end{bmatrix}$$

Тъй като стойностите на матриците $\left(\frac{\partial\hat{Y}(k)}{\partial U(k)}\right)'$ и $\left(\frac{\partial\hat{U}(k)}{\partial U(k)}\right)'$ са нулеви, то горният израз може да се опрости до следния вид [2]:

$$H[k,U(k)]=-2\left(\frac{\partial\hat{Y}}{\partial U(k)}\right)^2+2\rho\left(\frac{\partial\hat{U}(k)}{\partial U(k)}\right)^2 \tag{3.7.6}$$

След определяне на матрицата на Хесе е необходимо да бъде намерена обратната ѝ стойност, което внася допълнителна изчислителна тежест в метода на Нютон и се явява съществен негов недостатък. Разполагайки с обратната матрица на Хесе изразът (3.7.3) се преобразува до следните рекурентни зависимости:

$$\frac{\partial^2 J[k,U(k)]}{\partial U^2(k)}=\left[-2\left(\frac{\partial\hat{Y}(k)}{\partial U(k)}\right)^2+2\rho\left(\frac{\partial\hat{U}(k)}{\partial U(k)}\right)^2\right]^{-1}* \tag{3.7.7}$$
$$*\left[-2\left[R(k)-\hat{Y}(k)\right]^T\frac{\partial\hat{Y}(k)}{\partial U(k)}+2\rho\hat{U}(k)^T\frac{\partial\hat{U}(k)}{\partial U(k)}\right]$$

$$\Delta u(k+N_u-1)=\rho^{-1}H^{-1}\begin{bmatrix}e(k+N_1)\frac{\partial\hat{y}(k+N_1)}{\partial u(k+N_u-1)}+\dots\\ +e(k+N_2)\frac{\partial\hat{y}(k+N_2)}{\partial u(k+N_u-1)}\end{bmatrix} \tag{3.7.8}$$

$$\Delta u(k+N_u-2)=\Delta u(k+N_u-1)+$$
$$+\rho^{-1}H^{-1}\left[e(k+N_1)\frac{\partial\hat{y}(k+N_1)}{\partial u(k+N_u-2)}+\ldots+e(k+N_2)\frac{\partial\hat{y}(k+N_2)}{\partial u(k+N_u-2)}\right] \quad (3.6.9)$$

$$\Delta u(k)=\Delta u(k+1)+$$
$$\rho^{-1}H^{-1}\left[e(k+N_1)\frac{\partial\hat{y}(k+N_1)}{\partial u(k)}+\ldots+e(k+N_2)\frac{\partial\hat{y}(k+N_2)}{\partial u(k)}\right] \quad (3.7.10)$$

3.7.2. Метод на Левенберг-Маркгуард

Методът на Левенберг-Маркгуард също е градиентен метод от втори ред и като такъв се основава на квадратичната апроксимация на целева функция (3.7.1) . В действителност този метод представлява комбинация от градиентен метод от първи ред и метод на Нютон (Стоянов, 1990). При него движението към минимума се осъществява по формулата:

$$u^{(k+1)}=u^{(k)}-[H^{-1}(u^{(k)})+\lambda I]\nabla J(u^{(k)}) \quad (3.7.11)$$

където с I е означена единичната матрица, а параметърът λ регулира направлението на движение към екстремума, от градиентно – при голямо λ ($\lambda>10^3$) до направление, определено от метода на Нютон при λ=0. В началото на търсенето параметърът λ се избира голям (например $\lambda=10^4$), при което се подтиска влиянието на матрицата на Хесе и процесът на търсене се определя от градиентното направление. Ако още при първата стъпка се получи подобряване на целевия критерий, стойността на λ се намалява и се прави нова стъпка.

Изчисляването на вектор градиента и матрицата на Хесе става по същия начин както при метода на Нютон. Разликата е в рекурентните зависимости [6]:

$$\Delta u(k+N_u-1)=\rho^{-1}[H^{-1}+\lambda I]\begin{bmatrix}e(k+N_1)\dfrac{\partial\hat{y}(k+N_1)}{\partial u(k+N_u-1)}+\ldots\\ +e(k+N_2)\dfrac{\partial\hat{y}(k+N_2)}{\partial u(k+N_u-1)}\end{bmatrix} \quad (3.7.12)$$

$$\Delta u(k+N_u-2)=\Delta u(k+N_u-1)+\rho^{-1}[H^{-1}(x^{(k)})+\lambda I]*$$
$$*\left[e(k+N_1)\frac{\partial\hat{y}(k+N_1)}{\partial u(k+N_u-2)}+\ldots+e(k+N_2)\frac{\partial\hat{y}(k+N_2)}{\partial u(k+N_u-2)}\right] \quad (3.7.13)$$

$$\Delta u(k)=\Delta u(k+1)+\rho^{-1}[H^{-1}(x^{(k)})+\lambda I]*$$
$$*\left[e(k+N_1)\frac{\partial\hat{y}(k+N_1)}{\partial u(k)}+\ldots+e(k+N_2)\frac{\partial\hat{y}(k+N_2)}{\partial u(k)}\right] \quad (3.7.14)$$

При реализирането на метода на Левенберг-Маркгуард отново е използван стандартния GPC-критерий (1.4.1), с което е спазено изискването целевият критерий да е квадратичен, не са наложени ограничения и за да работи коректно алгоритъма е необходимо матрицата на Хесе да е положително определена. Основен недостък на този метод също е определянето на обратната матрица на Хесе.

3.7.3. LU декомпозиция

Характерно за описаните по-горе оптимизационни алгоритми за определяне стойността на управлението (метод на Нютон и метод на Левенберг-Маркгуард) е, че на всеки такт е необходимо намирането на обратната матрица на Хесе. Това е процедура, която изисква значителни изчисления и следователно води до намаляване на бързодействието на предсказващия регулатор. За да бъде преодолян този недостатък и да се намали изчислителната тежест е реализиран предложеният от Алън Тюринг метод на LU декомпозиция [2]. Това е широко използван в инженерната практика метод за решаване на матрични уравнения с помощта на т.нар. лентови матрици L и U т.е. матрици, в които единствено елементите върху главния и второстепенните диагонали са ненулеви (Доневска, 1994). Това разложение, когато съществува, е единствено, ако всички елементи в главния диагонал на матрицата U са единици, т.е. A= L*U, където:

$$L = \begin{pmatrix} l_{11} & 0 & 0 & \dots & 0 \\ l_{21} & l_{22} & 0 & \dots & 0 \\ l_{31} & l_{31} & l_{33} & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ l_{n1} & l_{n2} & l_{n3} & \dots & l_{nn} \end{pmatrix} \text{ и } U = \begin{pmatrix} 1 & u_{12} & u_{13} & \dots & u_{1n} \\ 0 & 1 & u_{23} & \dots & u_{2n} \\ 0 & 0 & 1 & \dots & u_{3n} \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & 1 \end{pmatrix} \quad (3.7.15)$$

Алгоритъмът, чрез който се определят елементите на матриците L и U, се нарича редукция на Краут (Crout Reduction Algorithm). При този метод първоначално се определят елементите от първия стълб на L, после от първия ред на U, следват елементите от втория стълб на L, втория ред на U, третия стълб на L, третия ред на U и т.н., докато се определят всички елементи. Алгоритъмът на Краут се състои от следните стъпки:

1) Инициализация. Ако $a_{11}=0$, стоп (не е възможно разложението). В противен случай, първият стълб на L съвпада с първия стълб на A, останалите елементи от първия ред на L са нули. Първият ред на U е първият ред на A, чиито елементи са разделени на $l_{11}=a_{11}$; останалите елементи в първия стълб на U са нули. Сега N=2.
2) За i=N,N+1,…,n се полага $L_i`$ на тази част от i-я ред на L, която вече е определена, т.е. $L_i`$ се състои от първите n-1 елемента на i-я ред на матрицата L.
3) За j= N,N+1,…,n се приема, че $U_j`$ е равна на тази част от j-я стълб на U, която вече е определена. Тогава $U_j`$ се състои от първите N-1 елемента на j-тия стълб на U.
4) Изчислява се N-тият стълб на L. За всеки елемент на този стълб, който се намира върху или под главния диагонал, се прилага формулата:

$$l_{iN} = a_{iN} - (L_i`)^T . U_N`, i = N, N+1, \dots, n. \quad (3.7.16)$$

Ако някой елемент $l_{NN}=0$, когато N≠n, изчисленията се спират, тъй като разложението не е възможно по-нататък. В противен случай, останалите елементи от N-тия ред на L се приемат за равни на нула.

5) Нека $u_{NN}=1$. Ако N=n, стоп (разложението е завършено). Иначе останалите елементи на N-тия стълб от U се приемат за равни на нула и се пресмятат елементите на N-тия ред на U. Всеки елемент от тези ред надясно от главния диагонал се изчислява по формулата

$$u_{Nj} = \frac{a_{Nj} - (\grave{L_N}).\grave{U_j}}{l_{NN}}, \; j = N, N+1, \ldots, n. \qquad (3.7.17)$$

6) Стойността на N се увеличава с единица и отново следва стъпка 2.

Методът на LU декомпозицията се използва за решаване на системи линейни уравнения, когато броя на неизвестните е равен на броя на уравненията. Матричният вид на тази система е A.X=B или L.(U.X)=B. Означава се U.X=Y и дадената ситема приема вида L.Y=B. Тогава за решението на това матрично уравнения спрамо $Y=L^{-1}.B$ и за неизвестната матрица се получава $U^{-1}.Y=X$, като се вземе предвид, че Y=U.X.

Недостатък на LU декомпозицията е, че матрицата A не може да се разложи, когато водещият елемент $a_{11}=0$. Това неудобство се отстранява лесно, като се разменят уравненията в системата, т.е. редовете на матрицата A.

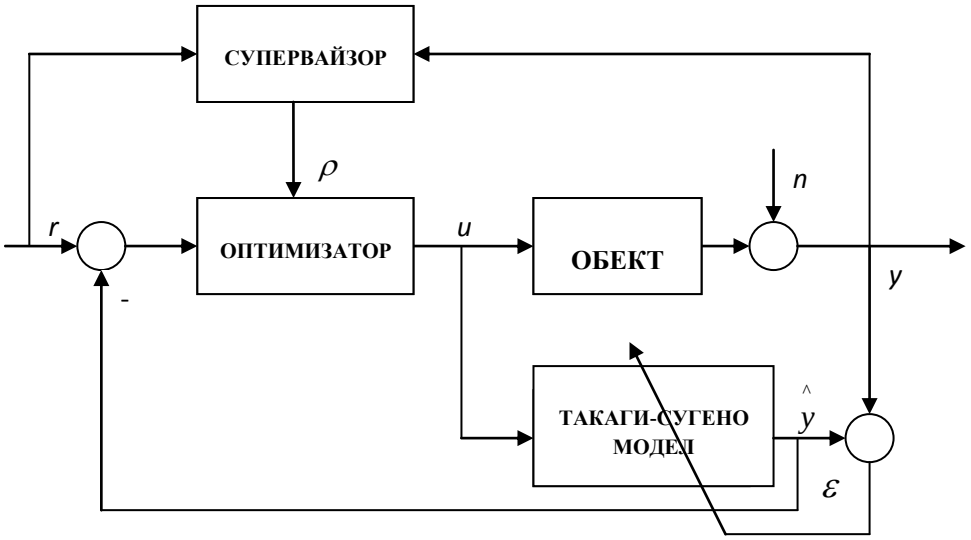
3.8. Алгоритъм за супервайзорна адаптивна донастройка на предсказващ регулатор

Въпросът с настройката на предсказващите регулатори е отворен проблем, който продължава да търси своето решение. По същество той се свежда до избора на три параметъра – хоризонт на предсказване, хоризонт на управление и тегловен коефициент в целевия критерий. Спорен е въпросът с избора на минималния хоризонт на предсказване. Според някои източници той се приема за единица, а според други се пределя въз основа на чистото закъснение в обекта за управление.

Както стана ясно от обзора, направен в Глава I, все още няма единна методология за определяне на настройваемите параметри на предсказващите регулатори. Ето защо, те обикновено се избират евристично в зависимост от вида на управлявания обект. Характерно е обаче, че изборът на конкретни „твърди” параметри, не винаги води до желания ефект и качествено управление в дадена система. Поради тази причина е полезно да бъде потърсен начин за адаптивната донастройка на

тези параметри в процеса на работа на системата и да се изследва влиянието на тези промени в структурата на управлението върху параметрите и показателите на процеса като цяло.

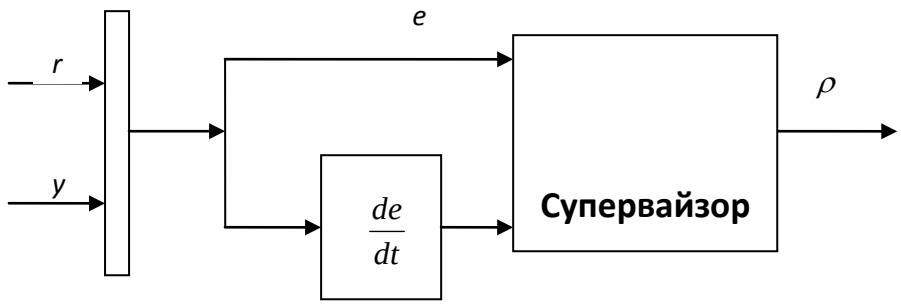
С оглед на това в настоящия дисертационен труд е разработен алгоритъм за супервайзорна адаптивна донастройка на тегловния коефициент в целевия критерий [3], като се запазват постоянни стойностите на хоризонтите на управление и на предсказване. Въвеждайки супервайзорния елемент структурата на системата с НМПУ вече има вида, показан на фиг.3.8.1.



Фиг.3.8.1 Структурна схема на система с НМПУ със супервайзор

Структурата на самия супервайзор е представена на фиг.3.8.2. От нея се вижда, че входове на супервайзорния елемент са грешката на регулиране и нейната производна. За реализирането на релацията между двата параметъра (тегловен коефициент и грешка на регулиране) се използва размит подход, който дава възможността за непрякото дефиниране на тази зависимост. Този подход е предпочитан, защото математическото описание на взаимовръзката е много трудна и често пъти дори непосилна задача. Тъй като при размитата логика се използват основно два механизма за извеждане – Мамдани и Сугено, то и супервайзорният блок е изграден в два варианта – с използване на размита логика с механизъм на извеждане

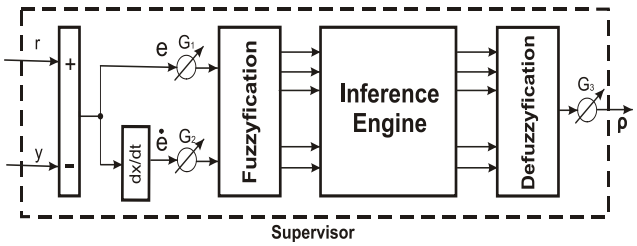
на Мамдани и чрез невронно-размита структура с механизъм на извеждане на Сугено.



Фиг.3.8.2 Структура на супервайзорния елемент

3.8.1. Супервайзорен блок с механизъм на извеждане на Мамдани

Структурата на супервайзорния блок с механизъм на извеждане на Мамдани е представена на фиг.3.8.3. От нея се вижда, че супервайзорът има структура като тази на класически размит регулатор – блок за размиване, логически блок и блок за деразмиване.



Фиг.3.8.3 Структура на супервайзорен блок с механизъм на извеждане на Мамдани

Основните настройваеми параметри на супервайзора са формата и броят на функциите на принадлежност, както и мащабиращите коефициенти на входните и изходни сигнали. Добре известно е, че броят на размитите правила зависи от броя на функциите на принадлежност. Размитите правила, използвани при реализацията на супервайзора, имат следната форма:

$$R^{(i)}: \text{if } e(k) \in \tilde{E}_i \text{ and } \Delta e(k) \in \Delta\tilde{E}_j \text{ then } \rho(k) \in \tilde{P}_k \qquad (3.8.1)$$

където $\tilde{E}_i$, $\tilde{E}_j$ and $\tilde{P}_k$ са активираните размити множества в $R^{(i)}$ правило. Таблицата с размитите правила, които се съдържат в логическия блок е:

Таблица 1. Таблица с размити правила

e / Δe	**NL**	**NM**	**NS**	**ZR**	**PS**	**PM**	**PL**
NL	**NL**	**NM**	**NM**	**NS**	**NS**	**ZR**	ZR
NM	**NM**	**NM**	**NS**	**NS**	**ZR**	**ZR**	ZR
NS	**NM**	**NS**	**NS**	**ZR**	**ZR**	**ZR**	PS
ZR	**NS**	**NS**	**ZR**	**ZR**	**ZR**	**PS**	PS
PS	**NS**	**ZR**	**ZR**	**ZR**	**PS**	**PS**	PM
PM	**ZR**	**ZR**	**ZR**	**PS**	**PS**	**PM**	PM
PL	ZR	ZR	PS	PS	PM	PM	PL

В блока за деразмиване се осъществява операцията по деразмиване като за целта е използван метода с центъра на гравитацията.

3.8.2. Супервайзорен блок с механизъм на извеждане на Такаги-Сугено

За реализирането на супервайзорния блок с механизъм на извеждане Такаги-Сугено е използвана невронно-размита структура, представена в т.2.1. В този случай обаче, за вход към невронно-размитата мрежа се прилага грешката и нейната производна, а изходът е тегловният коефициент, участващ в оптимизационния критерий. Функцията на Сугено се трасформира до вида:

$$\rho(k) = f_\rho(x(k)) \qquad (3.8.2)$$

където $x(k) = [\rho(k-1), \rho(k-2), e(k), e(k-1)]$; e е системната грешка. При тази постановка размитите правила имат следната форма:

$$Q^{(i)} : if \ \mathrm{x}_1 \text{ is } \tilde{\mathrm{C}}_1^{(i)} \text{ and } \mathrm{x}_\mathrm{p} \text{ is } \tilde{\mathrm{C}}_\mathrm{p}^{(i)} \text{ then } \mathrm{f}_\rho^{(i)}(k) \qquad (3.8.3)$$

$$f_\rho^{(i)}(k) = c_1^{(i)}\rho(k-1) + c_2^{(i)}\rho(k-2) + c_3^{(i)}e(k) + c_4^{(i)}e(k-1) + c_0^{(i)} \qquad (3.8.4)$$

(i)=1,2,…M, където M е броят на размитите правила, C_i е активираното размито множество дефинирано за входната величина x_i, а c_1, c_2, c_3, c_4, c_0 са коефициентите във функцията на Сугено f_ρ.

За обучение на представения невронно-размит супервайзор е реализиран двустъпков гредиентен алгоритъм, описан в т.3.6.1, за който са в сила изразите от (3.6.1) до (3.6.18).

3.9. Изводи

В Глава III от дисертационния труд са описани проектираните алгоритми за регулатори с невронно-размито предсказващо управление. Предложени са три невронно-размити модела, работещи с редуциран брой размити правила и подходящи за включване в системи с НМПУ. Това са Разпределен невронно-размит модел, Невронно-размит модел с частично размиване на входните сигнали и Модифициран нео-размит модел. В сравнение с невронно-размития модел от т.2.1, и трите предложени структури имат значително по-малък брой параметри за актуализиране по време на обучението, поради което може да се очаква съществено намаляване на времето за осъществяване на един такт от обучението.

DANFA, SFNN и Модифицираният нео-размит модел са реализирани във вариант с Тип 2 и с Интуиционистка размита логика. Това е направено с цел да се осигури възможността на моделите да се справят успешно в условията на неопределености. В този вариант обаче, се увеличава броят на параметрите, подлежащи на актуализиране при обучението, но все пак той е значително по-малък в сравнение с този на невронно-размития модел от т.2.1.

Едно от основните предимства, на което предсказващите регулатори дължат широкото си приложение, е възможността им да се справят с многомерни обекти с неизвестно или променливо времезакъснение. За да

бъде възможно това е нужно да се разработят многомерни невронно-размити модели, за които също е в сила изискването да осигуряват висока точност на предсказване и да са с малък брой параметри за настройка. Това е постигнато с реализираният в многомерен вариант модифициран нео-размит модел.

Реализирани са и различни алгоритми за обучение на предложените невронно-размити структури. Наред с превърналия се в класически метод с обратно разпространение на грешката, са разработени също и хибридни алгоритми, комбиниращи рекурентен метод на най-малките квадрати, метод на Нютон и метод на Левенберг-Маркгуард за обучение на параметрите във функцията на Сугено с градиентен метод от I ред за обучение на параметрите на размитите множества.

При системите с НМПУ методът за оптимизация е от особено значение, тъй като оказва съществено влияние върху изчислителната ефективност на оптимизатора. С оглед на това в настоящата дисертация са предпочетени алгоритми от втори ред, а именно тези на Нютон и на Левенберг-Маркгуард, тъй като при тях е намален броят на итерациите, свързани със сходимостта на алгоритъма, а това от своя страна води до по-голямо бързодействие на предсказващия регулатор.

Изследвана е възможността за използване на интелигентни невронно-размити структури за настройката на предсказващите регулатори. За целта е реализиран алгоритъм за супервайзорна адаптивна донастройка на тегловния коефициент в целевия критерий (1.4.1), като се запазват постоянни стойностите на хоризонтите на управление и на предсказване. Една интересна идея за бъдещи изследвания е използване на многомерния модифициран нео-размит модел за цялостна настройка на предсказващите регулатори, т.е. за определянето както на тегловния коефициент, така и на хоризонтите на управление N_u и на предсказване N_p.

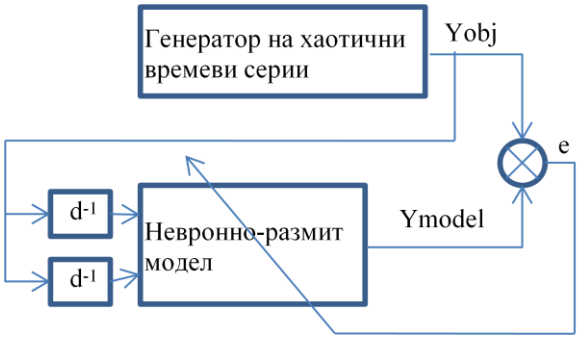
Глава IV

Експериментално изследване на предложените невронно-размити модели и алгоритми за регулатори с невронно-размито предсказващо управление

Предложените в предходната Глава III на настоящата дисертация невронно-размити модели с редуциран брой размити правила и алгоритми за предсказващи регулатори, изградени въз основа на тях, са софтуерно реализирани в средата *Matlab*. Една част от тях са тествани в симулационна среда, а други в реални условия.

Първоначално е изследвана способността на DANFA, SFNN и нео-размития модел да предсказват хаотични времеви серии. За обучението им е използван двустъпков градиентен алгоритъм с постоянна скорост на обучение η=0.05. Направено е сравнение на тези модели с невронно-размития модел, описан в т.2.1. Получените резултати са обобщени и е определен най-добрият модел в съответствите със заложените критерии. Избраният модел е реализиран във вариант с Тайп 2 размита логика и с интуиционстка размита логика в присъствието на шум, за да се провери как се справя в условия на неопределености.

При реализирането на тези симулационни изследвания посредством DANFA, SFNN и модифициран нео-размит модел е разработен MISO NNFARX модел като е използвана следната блокова схема:

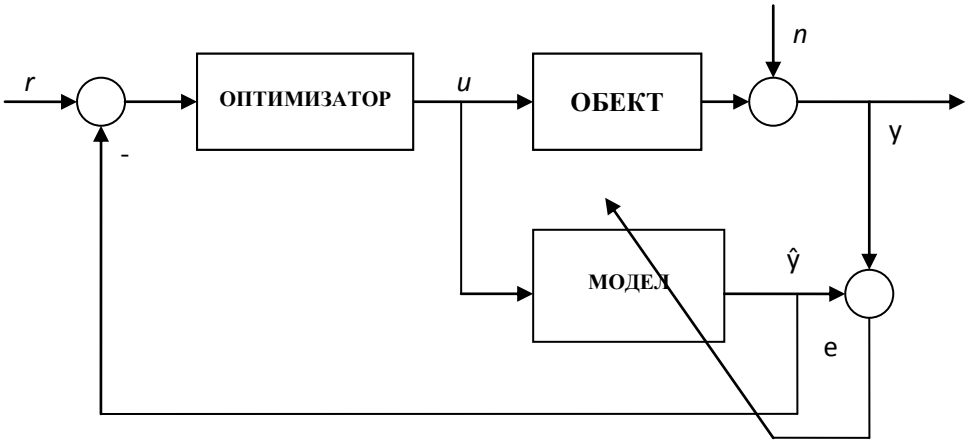


Фиг.4.1 Блокова схема за реализиране на предсказване на хаотични времеви серии

Във фиг.4.1 с $\mathbf{d^{-1}}$ е означено закъснението с един такт на съответния сигнал. Зад блока „Невронно-размит модел" стои всеки един от невронно-

размитите модели с редуциран брой размити правила - DANFA, SFNN или нео-размит модел. В блока „Генератор на хаотични времеви серии" е реализирана някоя от описаните по-долу времеви серии, чието поведение се стремим да моделираме посредством модела. Изходите на тези два блока се сравняват, при което се получава сигналът на моделната (предсказана) грешка, означена на фиг.4.1 с **e**. Въз основа на тази грешка е реализирано обучението на съответния модел.

При втората част от изследванията тестваните вече невронно-размити модели с редуциран брой размити правила са вградени в алгоритми на обобщени предсказващи регулатори, които са използвани за управлението на химически реактор и лабораторна топлинна система. Експериментите са реализирани въз основа на блок-схемата, представена на фиг.4.2. В блока „Оптимизатор" става изчисляването на оптималните стойности на управлението като за целта са използвани методът на Нютон и този на Левенберг-Маркгуард, описани съответно в т.3.7.1 и т.3.7.2.



Фиг. 4.2 Блок-схема за реализация на система с МПУ

4.1. Изследване способността на предложените модели за предсказване на хаотични времеви серии

Тестването на описаните в Глава III невронно-размити модели е направено с помощта на някои превърнали се в класически хаотични времеви серии, а именно Макгий-Глас (Mackey-Glass), Рослер (Rossler) и Лоренц (Lorenz). Хаотични динамични системи се срещат навсякъде в природата – от атмосферните условия до фондовата борса. Като цяло техните функции са различни при различни ситуации и са силно чувствителни към началните условия. Изучаването, разбирането на хаоса и поставянето му в служба на човешкото общество е много важна задача (Liu, 2010). Например, хаотичното поведение на торнадото е вредно за хората. То трябва да бъде внимателно проучено, за да могат да бъдат избегнати или контролирани този вид природни стихии. Основният проблем е, че хаотичните динамични системи не се описват с динамично уравнение и поради това те могат да бъдат разбрани само с помощта на хаотичните времеви серии. По-долу са дадени изразите, описващи съответно хаотичните серии на Макгий-Глас (4.1.1), Рослер (4.1.2) и Лоренц (4.1.3):

$$x(i+1) = \frac{x(i) + ax(i-s)}{(1+x^c(i-s)) - bx(i)} \tag{4.1.1}$$

където a=0.2; b=0.1; C=10; а началните условия са x(0)=0.1 и s= 17s.

$$\begin{aligned} \frac{\partial x}{\partial t} &= -y - z \\ \frac{\partial y}{\partial t} &= x + ay \\ \frac{\partial z}{\partial t} &= b + z(x - c) \end{aligned} \tag{4.1.2}$$

където a=0.2; b=0.4; c=5.7; при следните начални условия x0=0.1; y0=0.1; z0=0.1

$$\frac{\partial x}{\partial t} = s(y - x)$$
$$\frac{\partial y}{\partial t} = rx - y - xz \qquad (4.1.3)$$
$$\frac{\partial z}{\partial t} = xy - bz$$

където s=16; b=4; r=45.92; при следните начални условия x0=0.1; y0=0.1; z0=0.1.

За оценка на точността на предсказване е прието да се използва средноквадратична грешка (Mean Squared Error - MSE), дефинирана посредством израза:

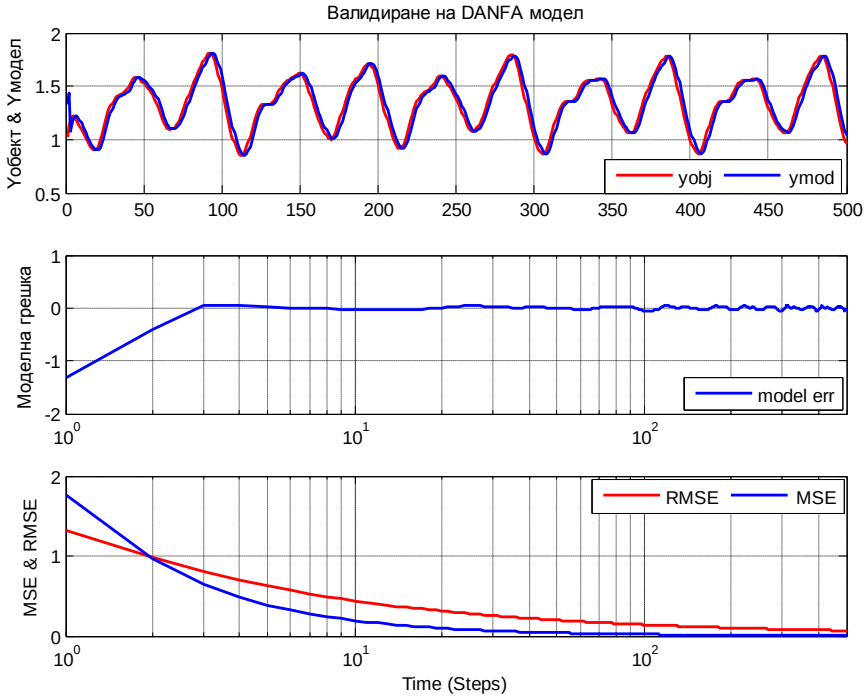
$$MSE = \frac{1}{n}\sum_{i=1}^{n}(\hat{Y}_i - Y_i)^2 \qquad (4.1.4)$$

и коренуваната средноквадратична грешка, определена чрез:

$$RMSE = \frac{1}{n}\sqrt{\sum_{i=1}^{n}(\hat{Y}_i - Y_i)^2} \qquad (4.1.5)$$

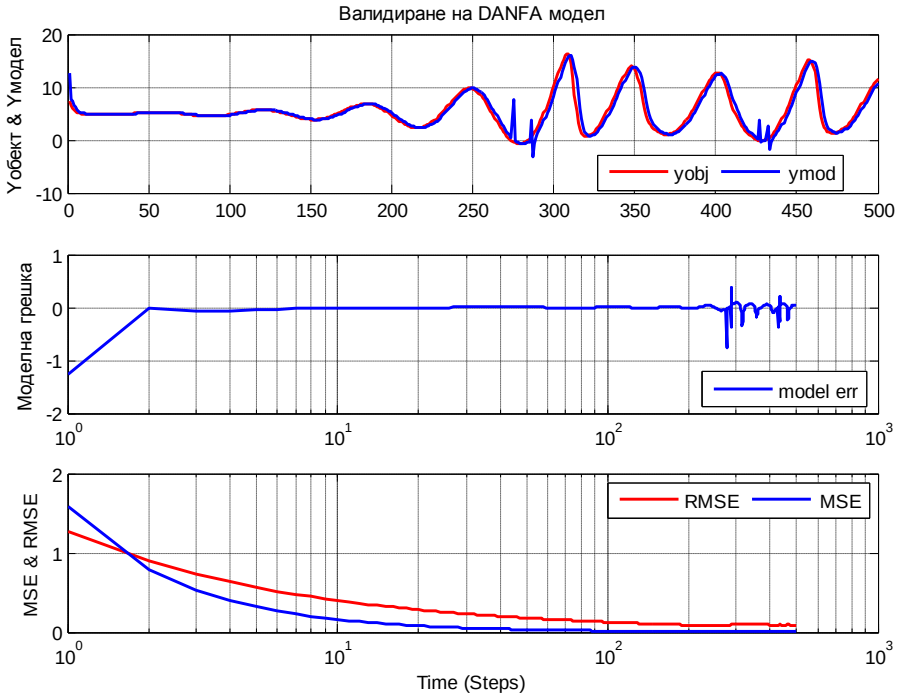
4.1.1. DANFA модел за предсказване на хаотични времеви серии

Разработеният DANFA модел е изследван във варианта, представен на фиг.3.1.3, т.е. като съвкупност от два подмодела. Способността на DANFA модела да предсказва хаотична времева серия на Макгий-Глас е демонстрирана на фиг.4.1.1.1. На нея освен желания и моделния изход са представени също изменението на моделната грешка, на RMSE и на MSE. Последните за по-голяма яснота се дадени в логаритмичен мащаб.



Фиг.4.1.1.1 Валидиране на DANFA модел чрез използване на Макий-Глас хаотична времева серия

По аналогичен начин на фиг. 4.1.1.2 е представена способността на DANFA модела да предсказва хаотична времева серия на Рослер.



Фиг.4.1.1.2 Валидиране на DANFA модел чрез използване на Рослер хаотична времева серия

Изследваният DANFA модел е сравнен с описания в т.2.1 невронно-размит модел като резултатите са обобщени в Таблица 4.1.1.1. В нея са представени данните при предсказване на Макий-Глас хаотична времева серия. Вижда се, че моментните стойности на моделната грешка (Model_err) на DANFA модела в отчетените стъпки са по-малки от тези на невронно-размития модел. Това се отразява и върху стойностите на MSE и RMSE и в момента на приключване на обучението в стъпка 500 те са по-малки при DANFA модела. Следователно може да се направи извода, че предложеният DANFA модел освен, че работи с по-малък брой правила и има по-малко параметри за определяне в процеса на обучение, е и по-точен. По-високата точност на DANFA модела има просто обяснение. В израза (2.1.4) свободният параметър b_0 играе ролята на филтър на смущенията. Това беше показано в т.2.1. В изследвания DANFA модел вместо един имаме два филтъра, което е и причина за по-точното моделиране.

Таблица 4.1.1.1

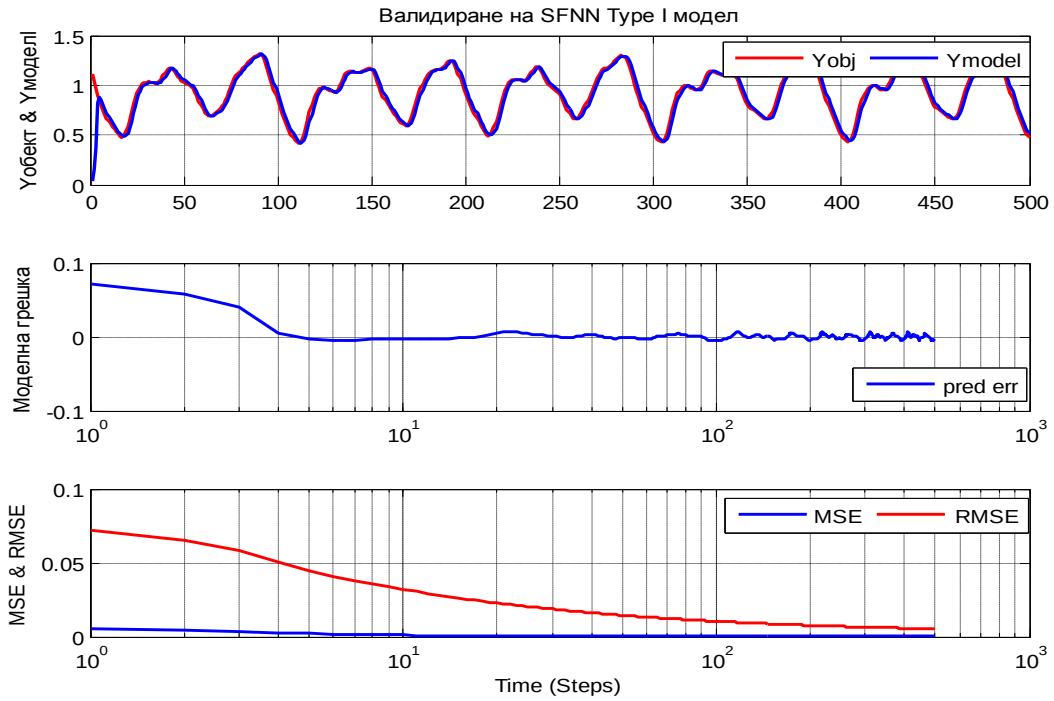
Стъпки	DANFA модел			Невронно-размит модел		
	Model_err	MSE	RMSE	Model_err	MSE	RMSE
50	-0.019613	0.053553	0.23141	-0.04702	0.0091595	0.095705
100	-0.067288	0.027756	0.1659	-0.15122	0.0077459	0.088011
150	-0.01297	0.01893	0.13759	-0.051306	0.0079593	0.089513
200	-0.058875	0.014637	0.12098	-0.12744	0.077992	0.088313
250	-0.02409	0.012033	0.1097	-0.051261	0.0077583	0.088081
300	-0.04841	0.010343	0.10156	-0.12544	0.0079122	0.08895
350	-0.039738	0.0090874	0.095328	-0.10179	0.0077017	0.087759
400	-0.045243	0.0081819	0.090454	-0.11864	0.007838	0.088471
450	-0.03445	0.0074437	0.086277	-0.095164	0.007659	0.087516
500	-0.040763	0.0068844	0.082973	-0.11223	0.007769	0.088143

4.1.2. SFNN модел за предсказване на хаотични времеви серии

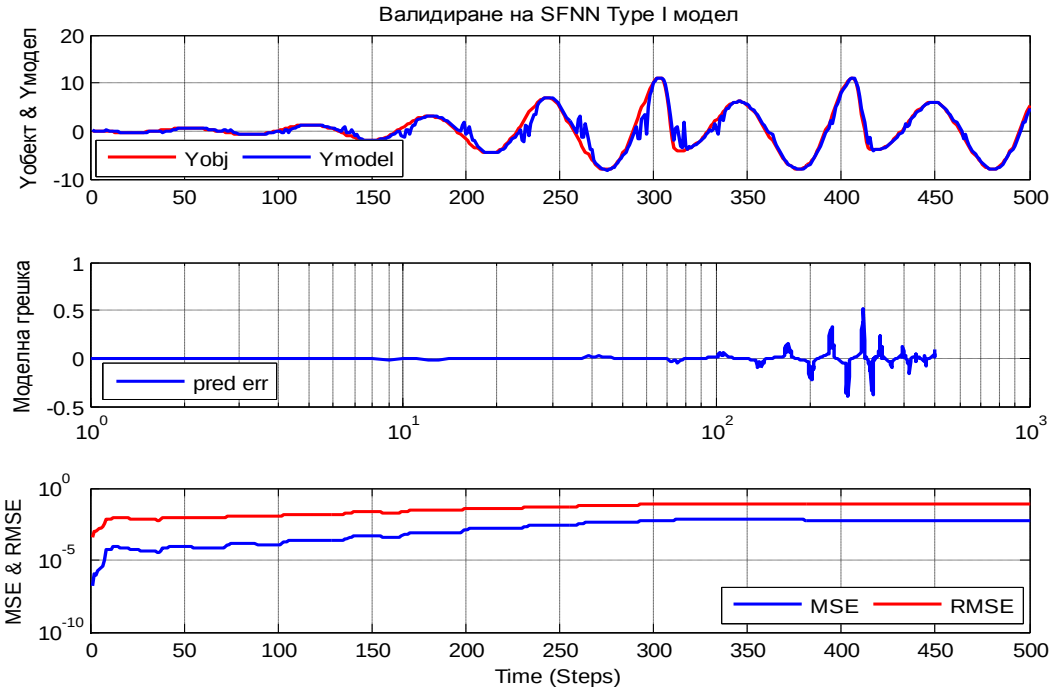
Невронно-размитият модел с частично размиване на входните сигнали е разработен и изследван в три варианта. При първия от тях SFNN Type I се размиват миналите стойности на изхода на обекта, а стойностите на управлението влизат директно в следствената част на размитите правила. На фиг.4.1.2.1 и фиг.4.1.2.2 са представени графично резултатите от валидирането на SFNN Type I модел съответно с Макий-Глас хаотична времева серия и с хаотична времева серия на Рослер. Моделната грешка, MSE и RMSE имат малки стойности, поради което за по-голяма яснота преходните им процеси са представени в логаритмичен мащаб.

При втория изследван SFNN Type II модел се размиват стойностите на управлението, а миналите стойности на изхода на обекта влизат директно в следствената част на размитите правила. На фиг.4.1.2.3 и фиг.4.1.2.4 са представени графично резултатите от валидирането му с хаотичните времеви серии съответно на Макий-Глас и Рослер.

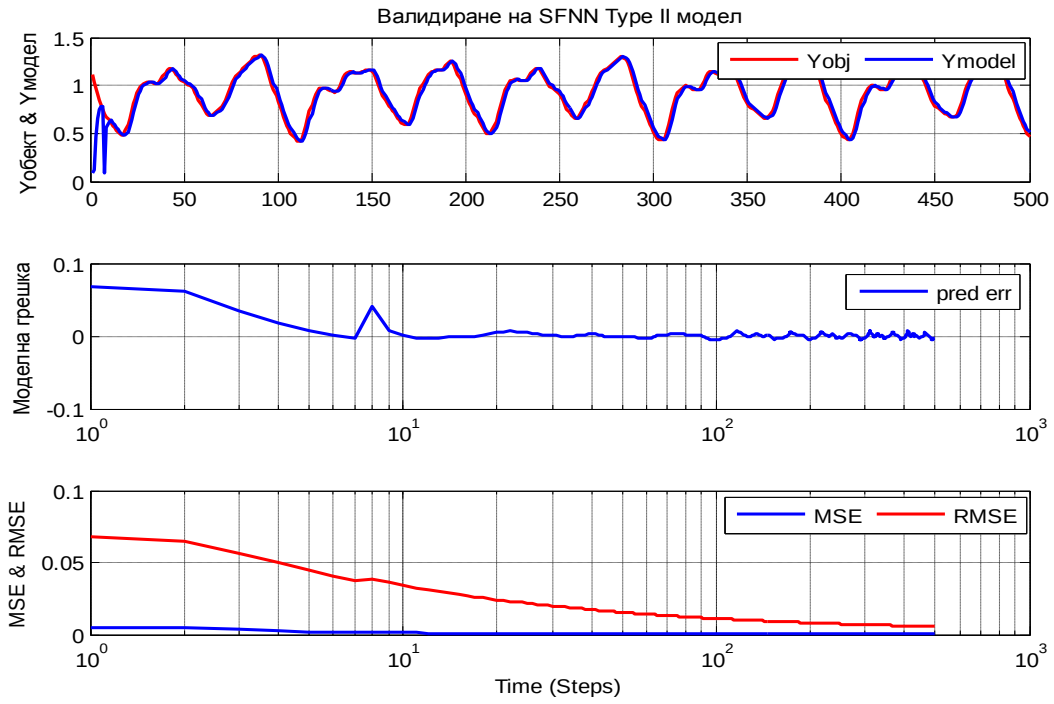
Третият разработен и реализиран вариант SFNN Type III модел, е този, при който са размити по една стойност от управлението и от изхода на обекта. Резултатите от тестването му с хаотичните времеви серии ва Макий-Глас и Рослер са представени съответно на фиг.4.1.2.5 и фиг.4.1.2.6.



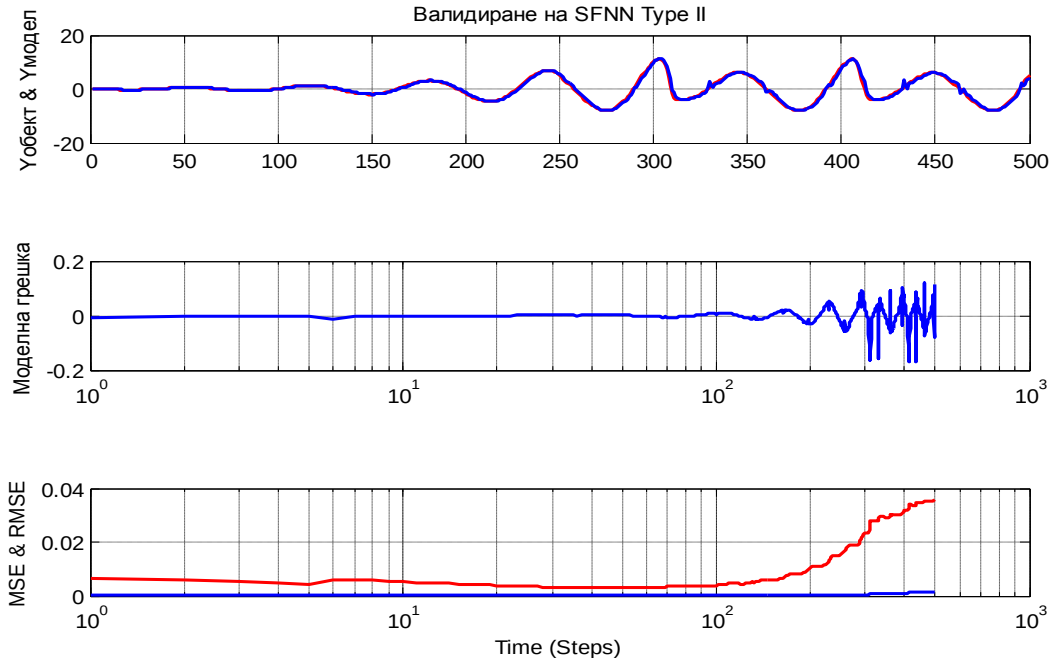
Фиг.4.1.2.1 Валидиране на SFNN Type I модел с Макий-Глас хаотична времева серия



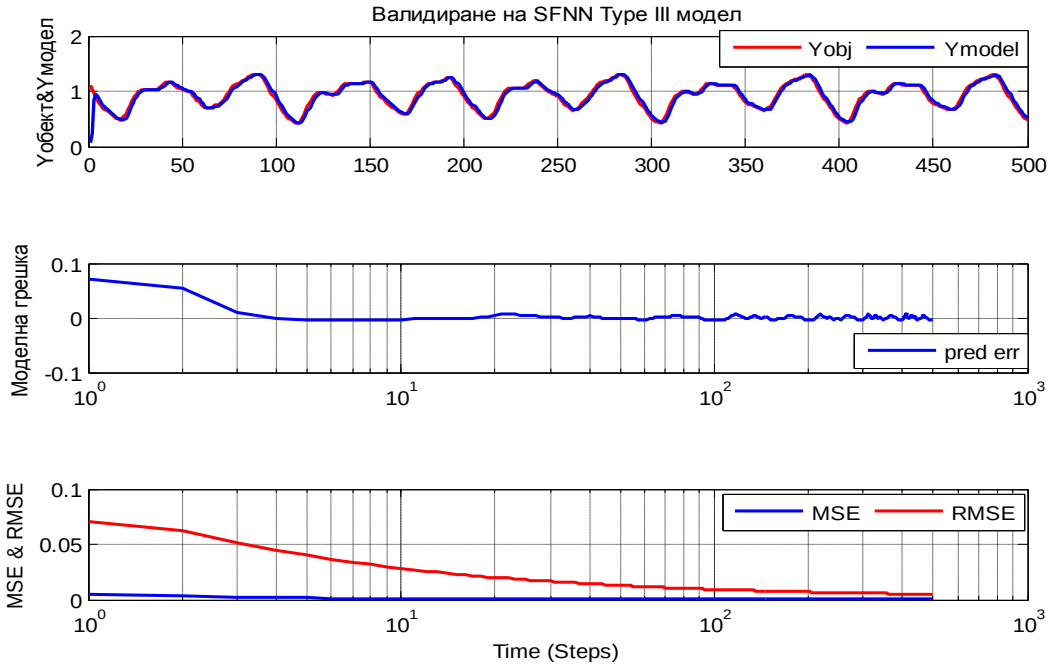
Фиг.4.1.2.2 Валидиране на SFNN Type I модел с Рослер хаотична времева серия



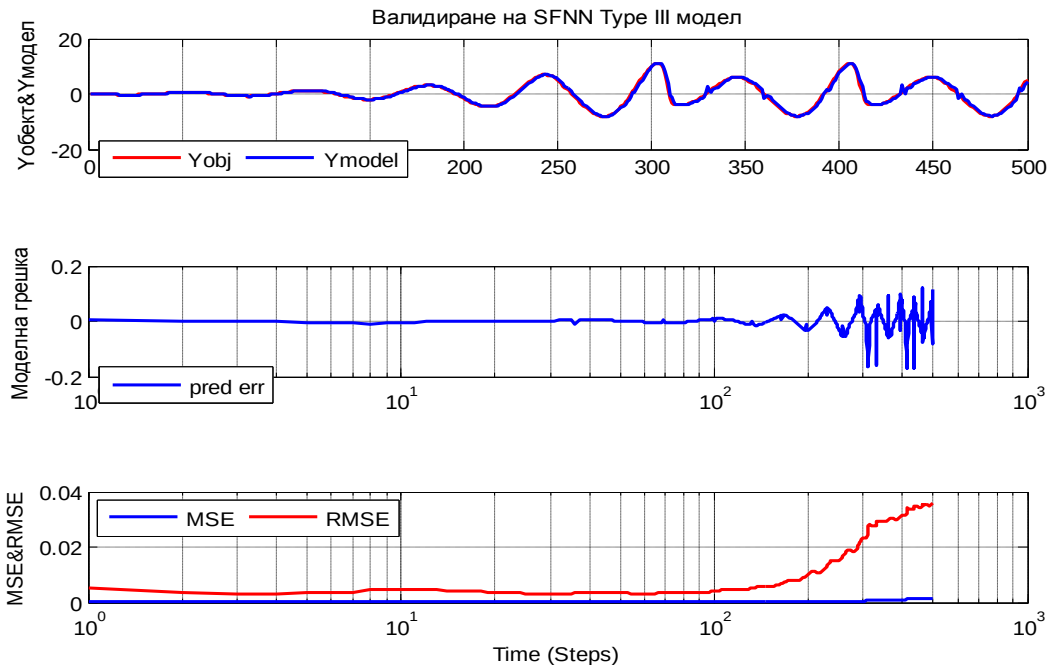
Фиг.4.1.2.3 Валидиране на SFNN Type II модел с Макий-Глас хаотична времева серия



Фиг.4.1.2.4 Валидиране на SFNN Type II модел с Рослер хаотична времева серия



Фиг.4.1.2.5 Валидиране на SFNN Type III модел с Макий-Глас хаотична времева серия



Фиг.4.1.2.6 Валидиране на SFNN Type III модел с Рослер хаотична времева серия

Сравнението на изследваните три типа невронно-размити модели с частично размиване на входните сигнали не би могло да се направи въз основа на броя използвани размити правила и/или броя параметри, подлежащи на актуализация по време на обучението, тъй като той е еднакъв и за трите модела. Следователно времето за реализиране на една стъпка (такт) от обучението също е еднакво за трите модела. Следващ критерий, по който можем да определим кой от тях е по-добър е точност на моделиране (на предсказване). За да се направи оценка по този критерий в Таблица 4.1.2.1 са обобщени данните за стойностите на MSE и RMSE, получени по време на обучението на трите модела с Макий-Глас хаотична времева серия. Видно е, че при SFNN Type III модел, стойностите на тези грешки са най-малки не само в края на обучението, но от самото му начало. И тъй като при предсказващите регулатори точността на предсказване е от изключително важно значение, то при работа с SFNN модел се препоръчва да се използва именно моделът с кръстосани връзки.

Друг важен извод, който може да бъде направен е, че и трите типа невронно-размити модели с частично размиване на входните сигнали са по-точни в сравнение с описания в т.2.1 невронно-размит модел. Това лесно може да се провери при анализ на данните за стойностите на MSE и RMSE от Таблица 4.2.2 и тези от Таблица 4.2.1 (колонката, отнасяща се за невронно-размит модел).

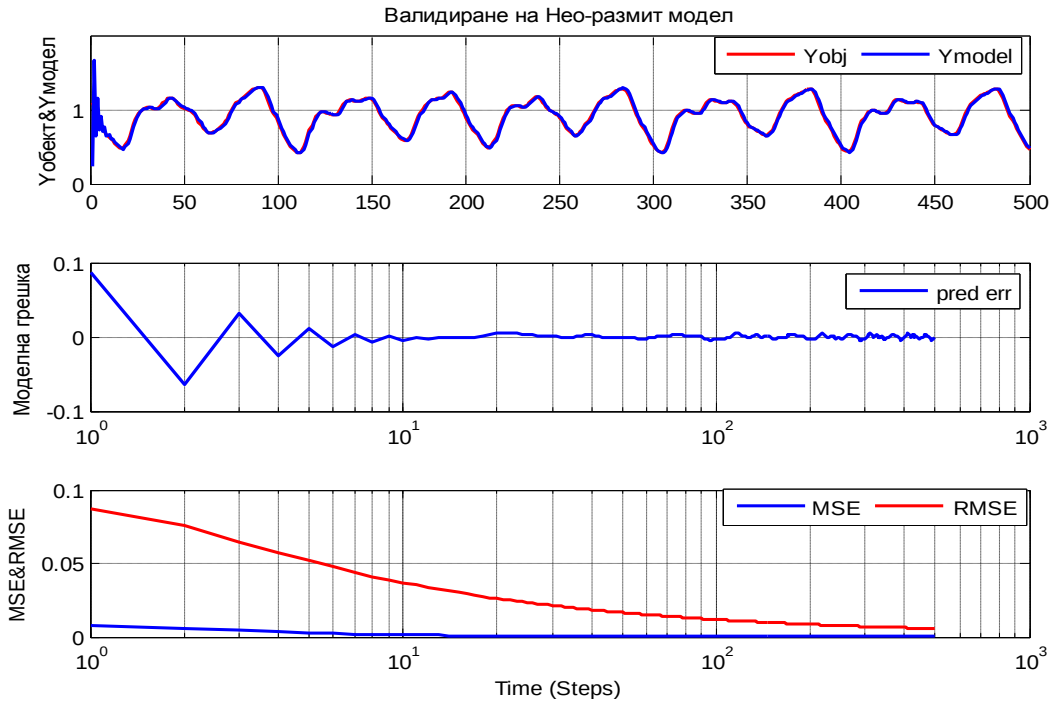
Таблица 4.1.2.1

Стъпки	SFNN Type I модел		SFNN Type II модел		SFNN Type III модел	
	MSE	RMSE	MSE	RMSE	MSE	RMSE
50	$2,4e^{-4}$	0,0145	$2,9e^{-4}$	0,0156	$1,6e^{-4}$	0,0129
100	$1,09e^{-4}$	0,0104	$1,2e^{-4}$	0,0111	$8,7e^{-5}$	0,0093
150	$7,6e^{-5}$	0,0087	$8,5e^{-5}$	0,0092	$6,04e^{-5}$	0,0078
200	$5,9e^{-5}$	0,0077	$6,6e^{-5}$	0,0081	$4,8e^{-5}$	0,0069
250	$4,9e^{-5}$	0,0070	$5,4e^{-5}$	0,0074	$3,97e^{-5}$	0,0063
300	$4,2e^{-5}$	0,0065	$4,7e^{-5}$	0,0068	$3,5e^{-5}$	0,0059
350	$3,7e^{-5}$	0,0061	$4,1e^{-5}$	0,0064	$3,1e^{-5}$	0,0056
400	$3,4e^{-5}$	0,0058	$3,7e^{-5}$	0,0061	$2,8e^{-5}$	0,0053

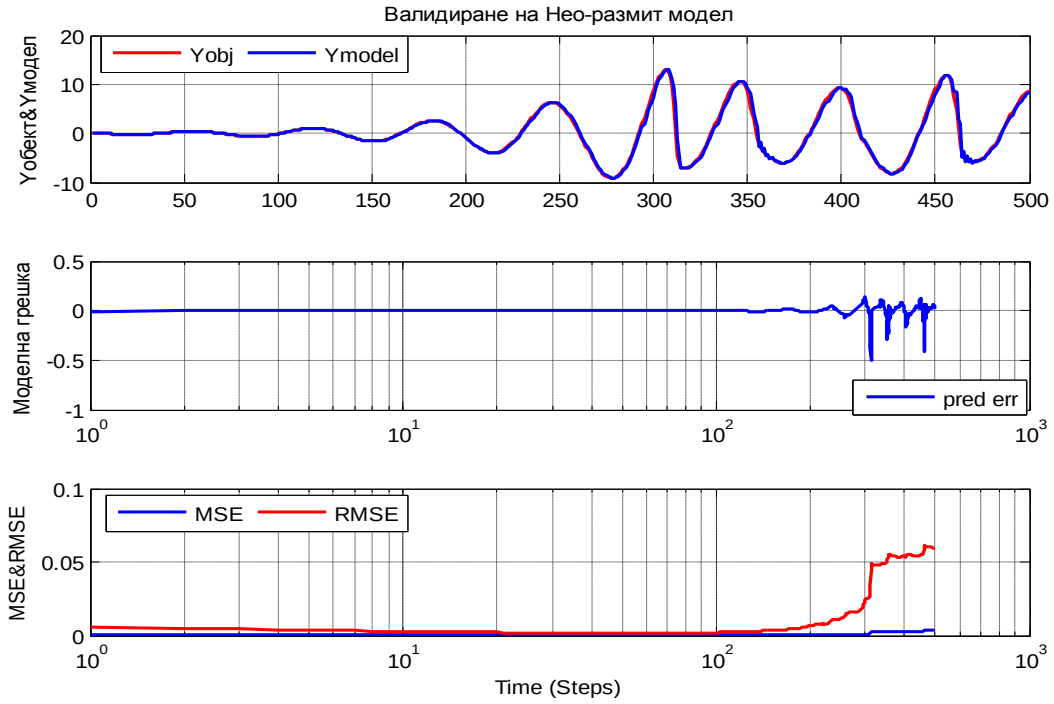
450	$3,1e^{-5}$	0,0056	$3,4e^{-5}$	0,0058	$2,6e^{-5}$	0,0051
500	$2,8e^{-5}$	0,0054	$3,2e^{-5}$	0,0056	$2,3e^{-5}$	0,0049

4.1.3. Модифициран нео-размит модел за предсказване на хаотични времеви серии

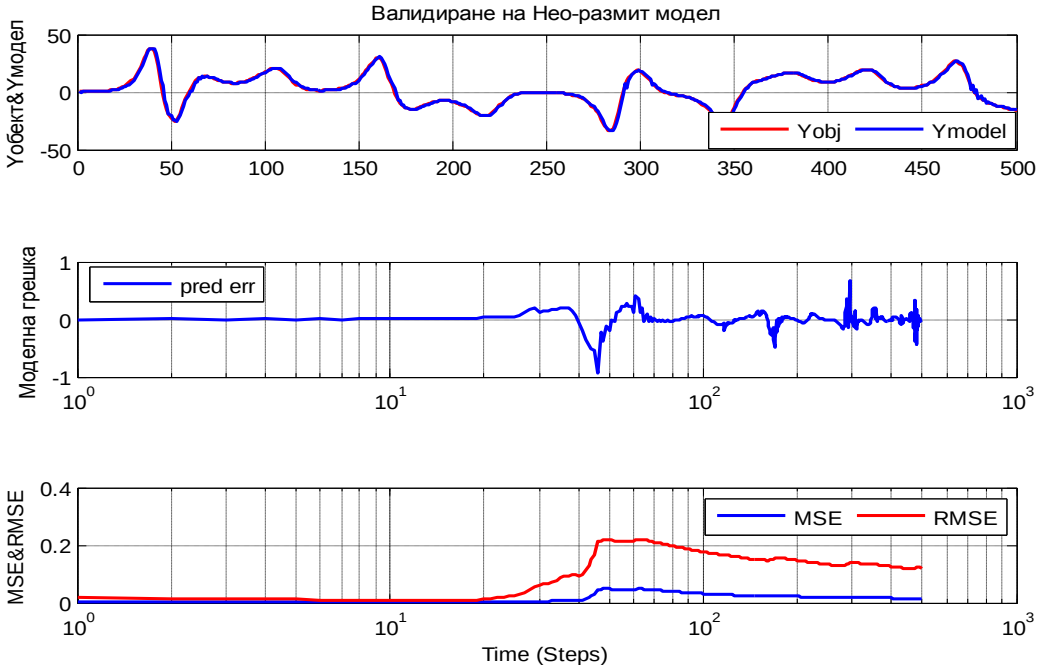
Способността на нео-размития модел да предсказва различни хаотични времеви серии е демонстрирана на фиг. 4.1.3.1, фиг. 4.1.3.2 и фиг. 4.1.3.3. От първата се вижда, че обучението на нео-размития модел става много бързо – в рамките само на десетина стъпки, след което моделната грешка приема стойности много близки до нула, т.е. характерна за този модел е много бързата сходимост на обучението. В същото време рязкото увеличаване на амплитудата на хаотичната серия води и до увеличаване както на моделната грешка, така и на MSE и RMSE (фиг. 4.1.3.2 и фиг. 4.1.3.3). Стойностите на последните, отчетени по време на обучението с Макий-Глас хаотична времева серия са обобщени в табл. 4.1.3.1. Трябва да се отбележи, че дадените графики са получени при актуализиране само на тегловните коефициенти в нео-размит модела. В случая, когато се променят и параметрите на размитите множества се получават подобни резултати, поради което е счетено за излишно представянето им на отделни фигури. Отчетени са обаче стойностите на MSE и RMSE в този случай и те също са представени в табл. 4.1.3.1. От тях се вижда, че точността на нео-размития модел в случая когато се актуализират само тегловните коефициенти и в случая когато се актуализират тегловните коефициенти и параметрите на размитите множества е приблизително еднаква. Може да се очаква обаче, че времето за осъществяване на една стъпка от процеса на обучение във втория случай ще е по-голямо и това дава основание да направим извода, че този вариант не винаги е удачен и оправдан. Той следва да бъде предпочетен за обекти с по-бавно изменяща се динамика.



Фиг.4.1.3.1 Валидиране на нео-размит модел с Макий-Глас хаотична времева серия



Фиг.4.1.3.2 Валидиране на нео-размит модел с Рослер хаотична времева серия



Фиг.4.1.3.3 Валидиране на нео-размит модел с Лоренц хаотична времева серия

Таблица 4.1.3.1

Стъпки	Нео-размит модел без обучаване на множества		Нео-размит модел с обучаване на множества	
	MSE	RMSE	MSE	RMSE
50	$9,5e^{-5}$	0,0097	$9,35e^{-5}$	0,0097
100	$7,3e^{-5}$	0,0085	$7,06e^{-5}$	0,0084
150	$5,9e^{-5}$	0,0077	$5,6e^{-5}$	0,0075
200	$4,8e^{-5}$	0,0069	$4,5e^{-5}$	0,0067
250	$4,1e^{-5}$	0,0064	$3,9e^{-5}$	0,0062
300	$3,5e^{-5}$	0,0059	$3,4e^{-5}$	0,0058
350	$3,2e^{-5}$	0,0057	$3,12e^{-5}$	0,0056
400	$2,9e^{-5}$	0,0054	$2,92e^{-5}$	0,0054
450	$2,5e^{-5}$	0,0050	$2,33e^{-5}$	0,0048
500	$2,25e^{-5}$	0,0047	$2,17e^{-5}$	0,0047

4.1.4. Сравнителен анализ

За оценка на предложените невронно-размити модели са използвани няколко критерии, а именно: брой размити правила, използвани от модела; брой параметри, които трябва да се определят в процеса на обучението му; времето, необходимо за реализиране на всички изчисления в рамките на един такт (стъпка) и точност на предсказване, определена в съответствие с изразите (4.1.4) и (4.1.5). Обобщение на тези критерии за изследваните модели е представено в табл. 4.1.4.1.

Таблица 4.1.4.1

Модел	Бр. правила	Бр. параметри	Точност			t
			стъпки	MSE	RMSE	
Невронно-размит модел, описан в т.2.1	81	1053	50	0.0091595	0.095705	$2,82e^{-4}$
			100	0.0077459	0.088011	
			150	0.0079593	0.089513	
			200	0.077992	0.088313	
			250	0.0077583	0.088081	
			300	0.0079122	0.08895	
			350	0.0077017	0.087759	
			400	0.007838	0.088471	
			450	0.007659	0.087516	
			500	0.007769	0.088143	
DANFA модел	18	126	50	0.053553	0.23141	$1.83e^{-4}$
			100	0.027756	0.1659	
			150	0.01893	0.13759	
			200	0.014637	0.12098	
			250	0.012033	0.1097	
			300	0.010343	0.10156	
			350	0.0090874	0.095328	
			400	0.0081819	0.090454	
			450	0.0074437	0.086277	

			500	0.0068844	0.082973	
SFNN Type III модел	9	63	50	$1,6e^{-4}$	0,0129	$1.58e^{-4}$
			100	$8,7e^{-5}$	0,0093	
			150	$6,04e^{-5}$	0,0078	
			200	$4,8e^{-5}$	0,0069	
			250	$3,97e^{-5}$	0,0063	
			300	$3,5e^{-5}$	0,0059	
			350	$3,1e^{-5}$	0,0056	
			400	$2,8e^{-5}$	0,0053	
			450	$2,6e^{-5}$	0,0051	
			500	$2,3e^{-5}$	0,0049	
Нео-размит модел с обучаване параметрите на размитите множества	12	36	50	$9,35e^{-5}$	0,0097	$1.67e^{-4}$
			100	$7,06e^{-5}$	0,0084	
			150	$5,6e^{-5}$	0,0075	
			200	$4,5e^{-5}$	0,0067	
			250	$3,9e^{-5}$	0,0062	
			300	$3,4e^{-5}$	0,0058	
			350	$3,12e^{-5}$	0,0056	
			400	$2,92e^{-5}$	0,0054	
			450	$2,33e^{-5}$	0,0048	
			500	$2,17e^{-5}$	0,0047	
Нео-размит модел без обучаване параметрите на размитите множества	12	12	50	$9,5e^{-5}$	0,0097	$1.14e^{-4}$
			100	$7,3e^{-5}$	0,0085	
			150	$5,9e^{-5}$	0,0077	
			200	$4,8e^{-5}$	0,0069	
			250	$4,1e^{-5}$	0,0064	
			300	$3,5e^{-5}$	0,0059	
			350	$3,2e^{-5}$	0,0057	
			400	$2,9e^{-5}$	0,0054	
			450	$2,5e^{-5}$	0,0050	
			500	$2,25e^{-5}$	0,0047	

Времето, което е дадено в последната колона на табл. 4.1.4.1 е отчетено с помощта на командите *tic* и *toc* в Matlab. Стойностите са получени при използване на компютърна конфигурация с Intel Pentium CPU 850 2,90GHz и 4GB RAM памет.

Въз основа на обобщените данни от табл. 4.1.4.1 може да се каже, че модифицираният нео-размит модел е най-добър по смисъла на зададените критерии. Въпреки, че работи с по-голям брой размити правила в сравнение с SFNN Type III, нео-размит модела има по-малък брой параметри за актуализиране в процеса на обучение (дори и в случая с обучаване параметрите на размитите множества), по-точен е и по-бърз. Ето защо именно нео-размития модел е реализиран във вариант с Тип 2 размита логика и с Интуиционистка размита логика в присъствието на шум, за да се види как се справя в условия на неопределености. Той е разработен и в MIMO вариант, тъй като едно от предимствата на предказващите регулатори е способността им да се справят с многомерни обекти.

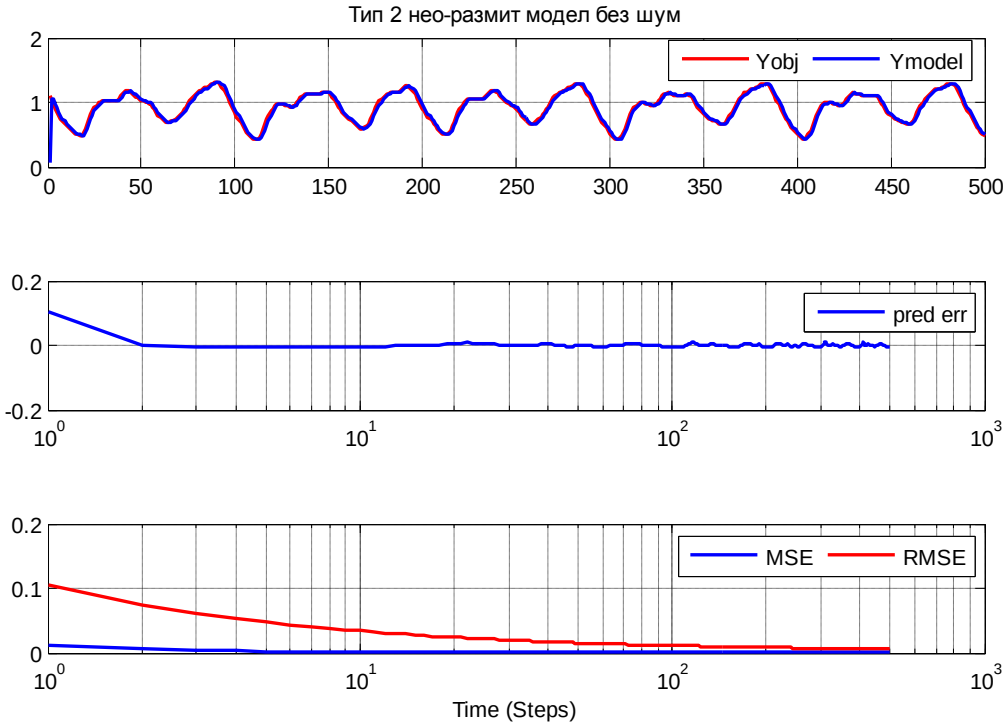
4.1.5. Тип 2 и Интуиционистки нео-размити модели

Тип 2 и Интуиционистката размита логика се явяват допълнение на теорията на Заде за класическата (Тип 1) размита логика. През последните години невронно-размити структури, използващи Тип 2 размита логика се използват широко в системите за управление и в частност при предсказващите регулатори. Основна причина за това е способността им да моделират нелинейни процеси с висока точност дори и в присъствието на неопределености. Въпреки, че Интуиционистката размита логика се явява алтернатива на Тип 2 размитата логика, то тя е значително по-слабо използвана. Може да се каже, че за целите на предсказващото управление Интуиционистки невронно-размити структури не са използвани.

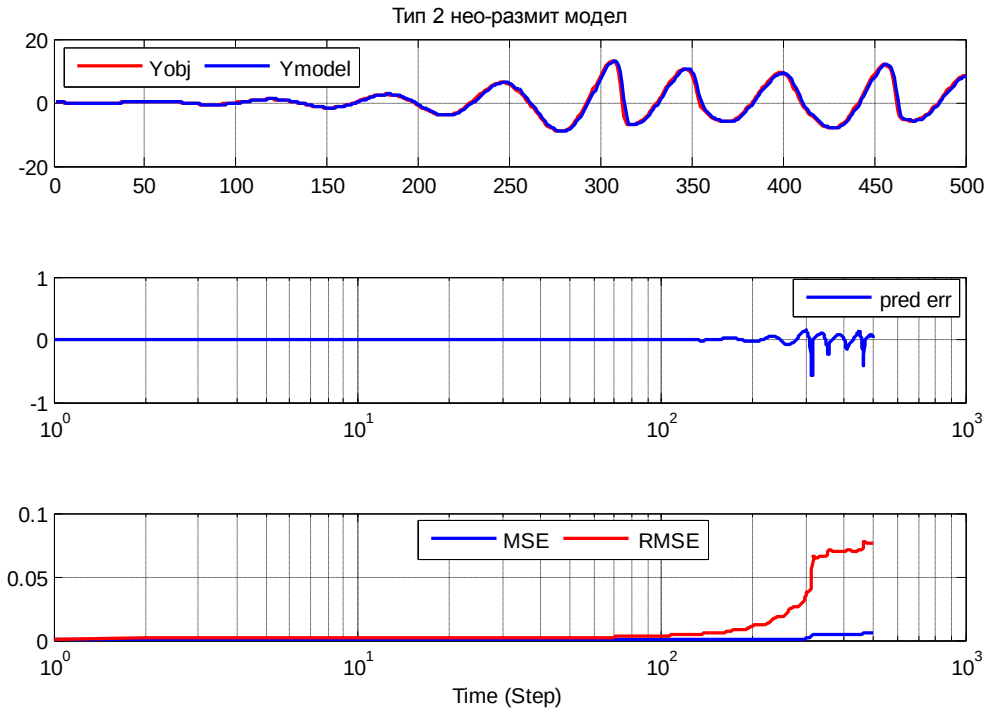
В настоящия дисертационен труд са реализирани Тип 2 нео-размит модел [7] и Инстуиционистки нео-размит модел. Тествана е способността им за предсказване на хаотични времеви серии в случаите със и без добавен бял шум, а получените резултати са представени на фиг. 4.1.5.1 до

фиг. 4.1.5.8. За точността на моделиране се съди по стойностите на MSE и RMSE, които са обобщени в табл. 4.1.5.1.

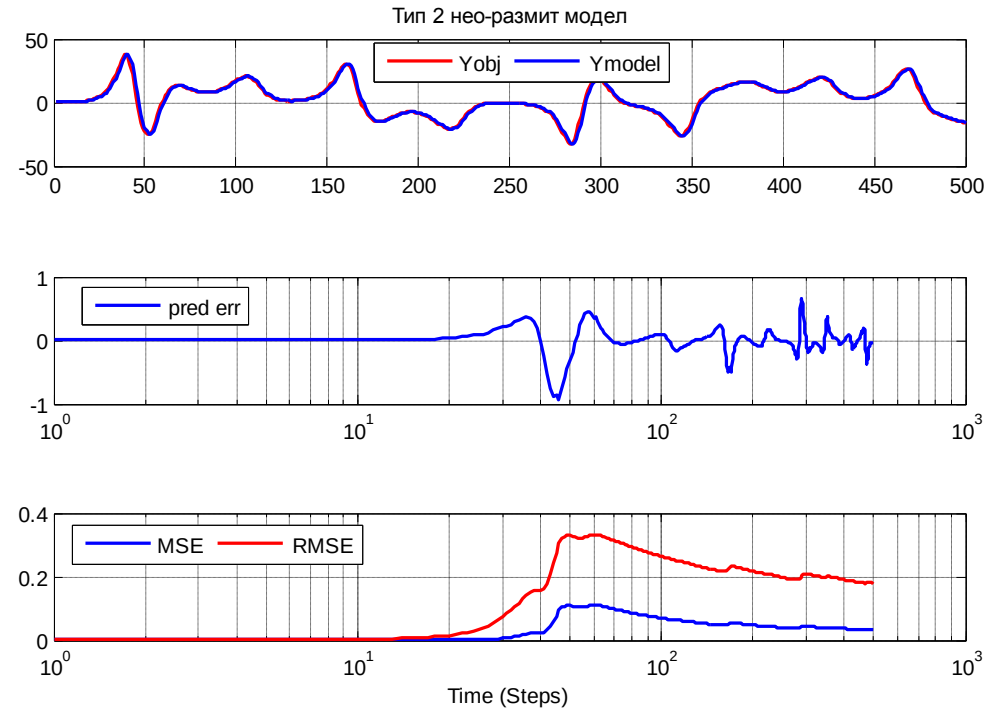
От получените резултати става ясно, че Интуиционисткият нео-размит модел работи с по-малка грешка в сравнение с Тип 2 нео-размития модел. Времето за осъществяване на необходимите изчисления за една стъпка от обучението при Интуиционисткия нео-размит модел е съответно $1.48e^{-4}$ за случая без шум и $1.50e^{-4}$ с шум. При Тип 2 нео-размития модел това време е по-голямо – съответно $1.93e^{-4}$ и $1.96e^{-4}$ за случаите без и с шум. Следователно Интуиционисткият нео-размит модел е по-точен и по-бърз и това го прави подходящ за включване в предсказващ регулатор.



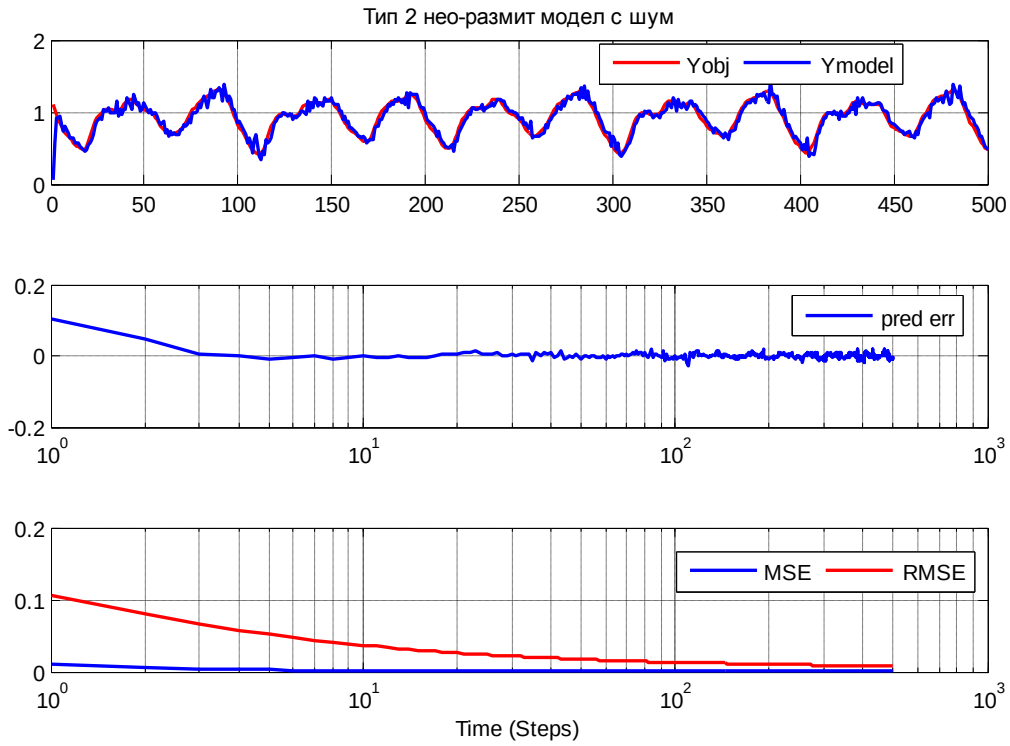
Фиг.4.1.5.1 Предсказване на Макий-Глас хаотична времева серия с Тип 2 нео-размит модел



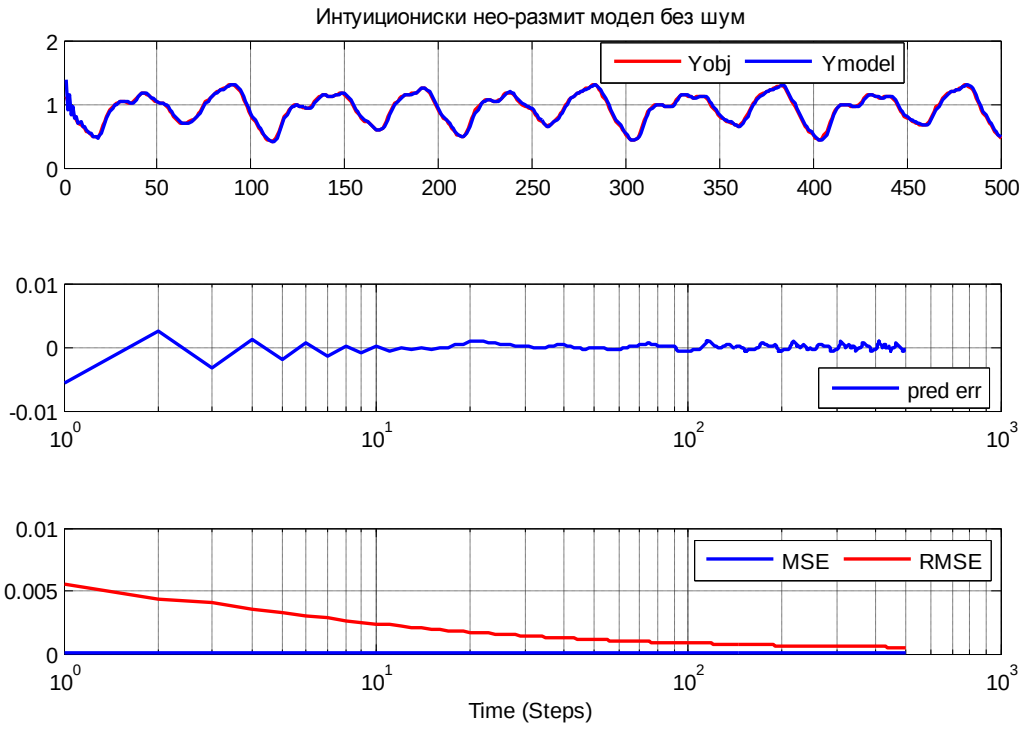
Фиг.4.1.5.2 Предсказване на Рослер хаотична времева серия с Тип 2 нео-размит модел



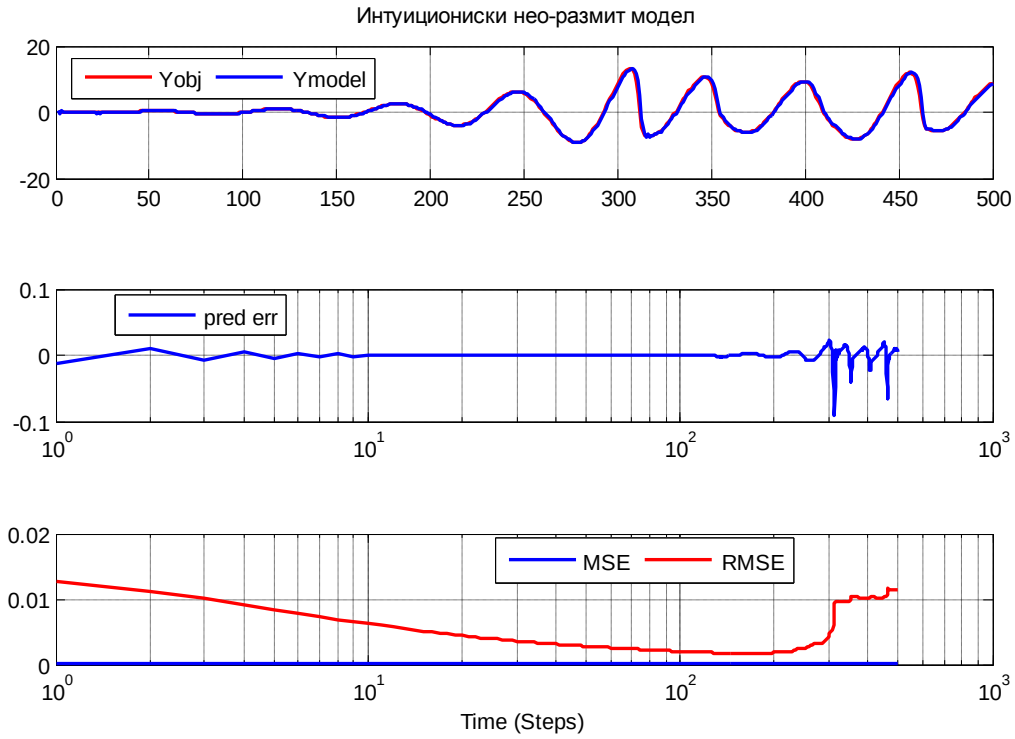
Фиг.4.1.5.3 Предсказване на Лоренц хаотична времева серия с Тип 2 нео-размит модел



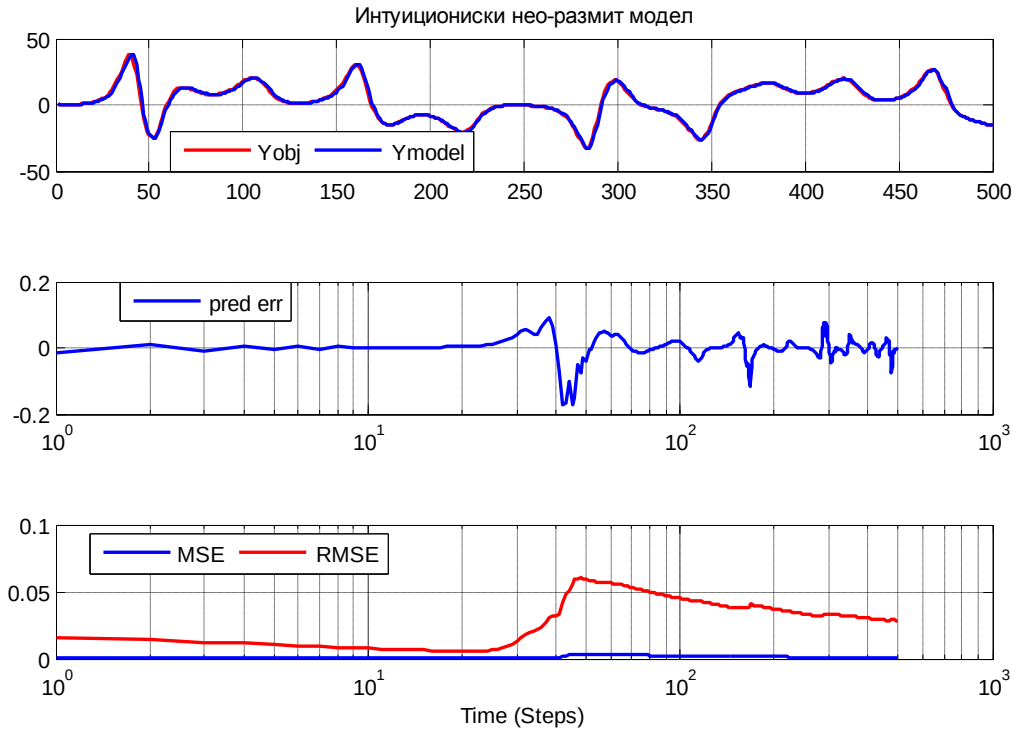
Фиг.4.1.5.4 Предсказване на Макий-Глас хаотична времева серия с добавен бял шум с Тип 2 нео-размит модел



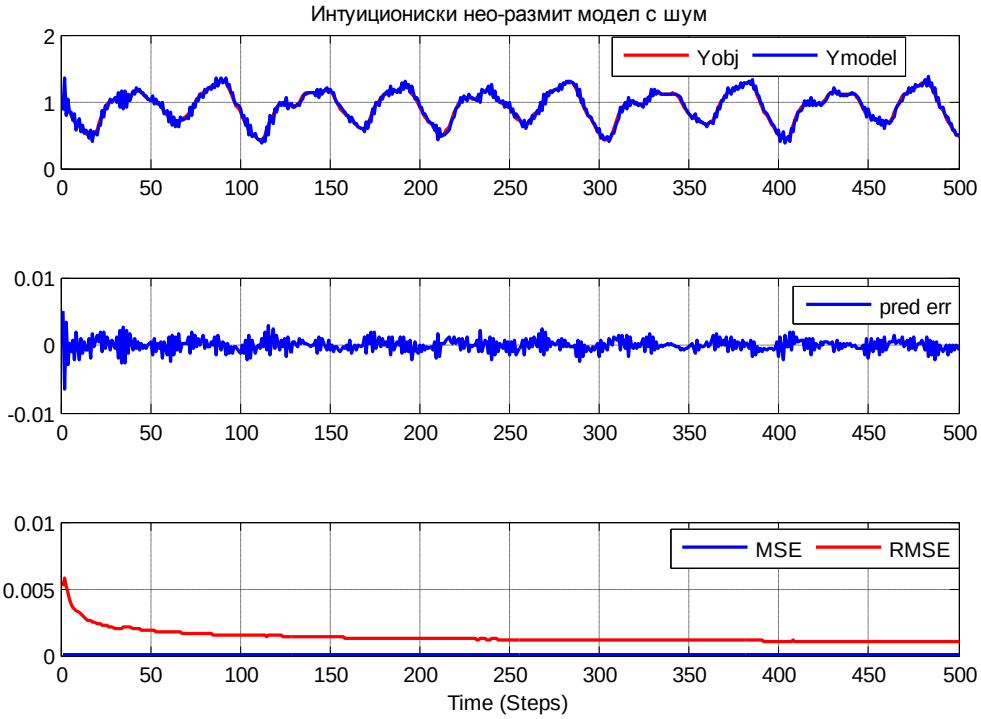
Фиг.4.1.5.5 Предсказване на Макий-Глас хаотична времева серия с Интуиционистки нео-размит модел



Фиг.4.1.5.6 Предсказване на Рослер хаотична времева серия с Интуиционистки нео-размит модел



Фиг.4.1.5.7 Предсказване на Лоренц хаотична времева серия с Интуиционистки нео-размит модел



Фиг.4.1.5.8 Предсказване на Макий-Глас хаотична времева серия с добавен бял шум с Интуиционистки нео-размит модел

Таблица 4.1.5.1

Стъпки	Тип 2 нео-размит модел				Интуиционистки нео-размит модел			
	Без шум		С шум		Без шум		С шум	
	MSE	RMSE	MSE	RMSE	MSE	RMSE	MSE	RMSE
50	$2.38e^{-4}$	0.0156	$2.98e^{-4}$	0.0173	$1.26e^{-6}$	0.0011	$3.42e^{-6}$	0.0019
100	$1.26e^{-4}$	0.0112	$1.69e^{-4}$	0.0130	$7.05e^{-7}$	$8.39e^{-4}$	$2.25e^{-6}$	0.0015
150	$8.85e^{-5}$	0.0094	$1.31e^{-4}$	0.0115	$5.19e^{-7}$	$7.2e^{-4}$	$1.87e^{-6}$	0.0013
200	$7.01e^{-5}$	0.0084	$1.07e^{-4}$	0.0104	$4.32e^{-7}$	$6.57e^{-4}$	$1.61e^{-6}$	0.0012
250	$5.89e^{-5}$	0.0077	$8.4e^{-5}$	0.0091	$3.75e^{-7}$	$6.12e^{-4}$	$1.48e^{-6}$	0.0011
300	$5.19e^{-5}$	0.0072	$7.22e^{-5}$	0.0085	$3.41e^{-7}$	$5.85e^{-4}$	$1.37e^{-6}$	0.0011
350	$4.63e^{-5}$	0.0068	$6.72e^{-5}$	0.0082	$3.13e^{-7}$	$5.6e^{-4}$	$1.26e^{-6}$	0.0010
400	$4.26e^{-5}$	0.0065	$5.91e^{-5}$	0.0077	$2.96e^{-7}$	$5.44e^{-4}$	$1.19e^{-6}$	0.0010
450	$3.92e^{-5}$	0.0063	$5.33e^{-5}$	0.0073	$2.78e^{-7}$	$5.28e^{-4}$	$1.13e^{-6}$	0.0010
500	$3.69e^{-5}$	0.0061	$4.76e^{-5}$	0.0069	$2.67e^{-7}$	$5.17e^{-4}$	$1.09e^{-6}$	0.0010

4.1.6. Многомерен нео-размит модел

За тестването на многомерния нео-размит модел е използвана многомерна нелинейна система, представена от Pan (2012) и описана с изразите:

$$y_1(k) = \frac{y_1^2(k-1)}{y_1^2(k-1)+1} + 0.5y_2(k-1)$$

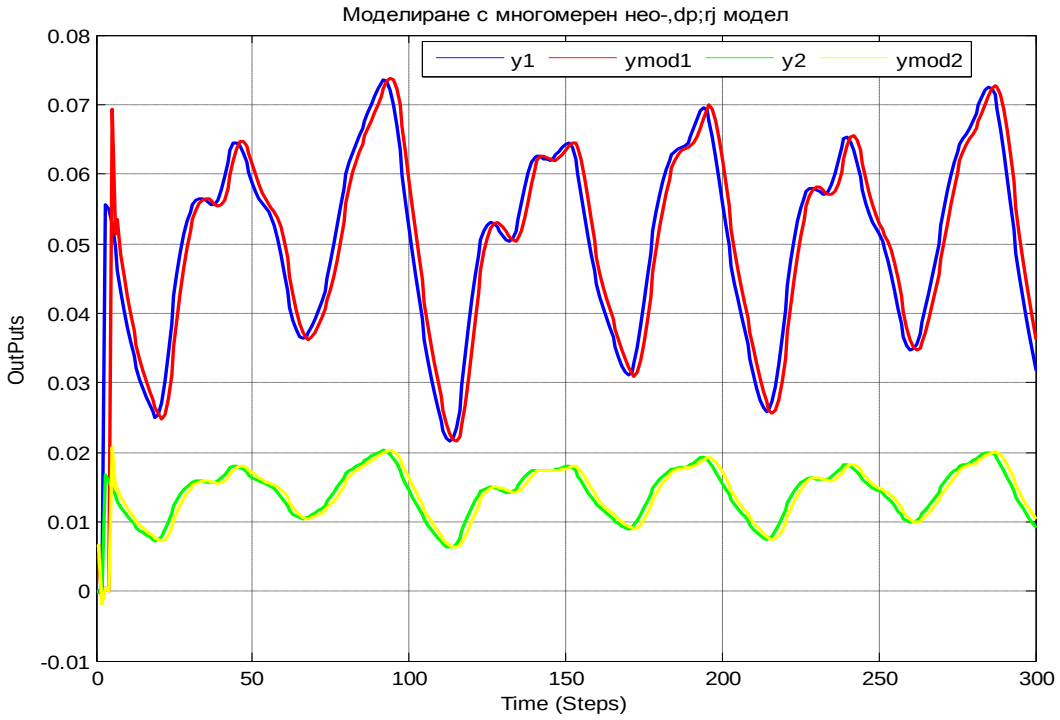
$$y_2(k) = \frac{y_1^2(k-1)}{y_2^2(k-1)+y_3^2(k-1)+y_4^2(k-1)} + u_1(k-1) \qquad (4.1.6.1)$$

$$y_3(k) = \frac{y_3^2(k-1)}{y_3^2(k-1)+1} + 0.3y_4(k-1)$$

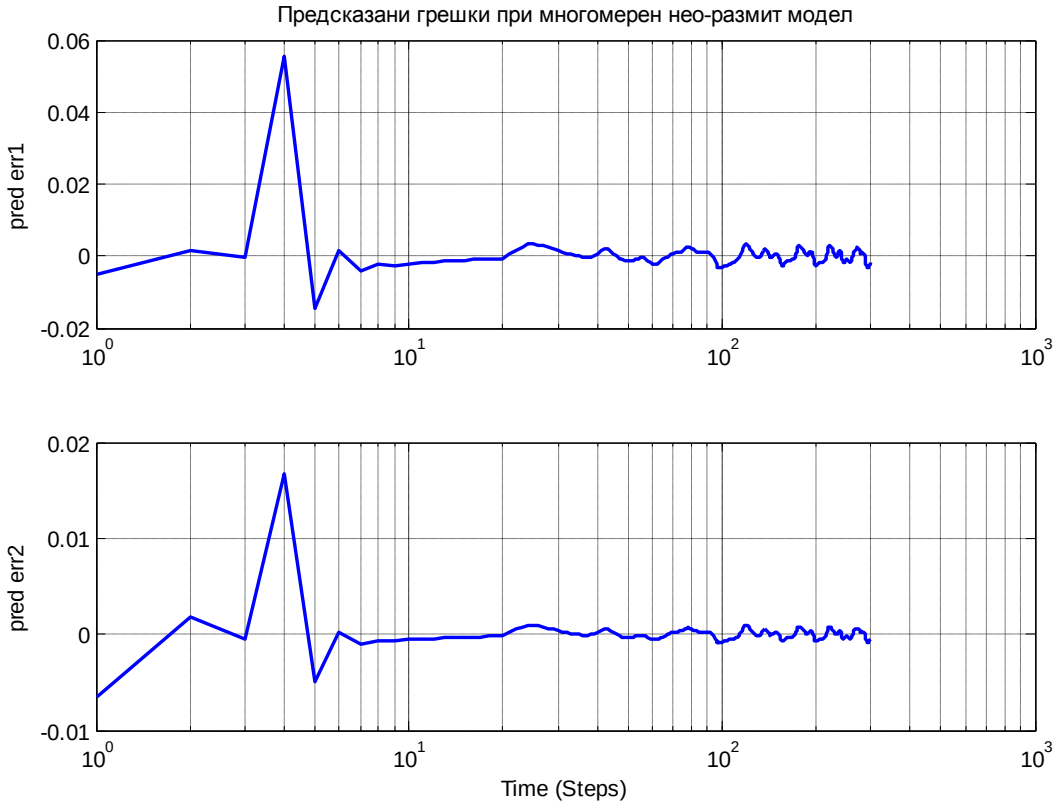
$$y_4(k) = \frac{y_3^2(k-1)}{y_1^2(k-1)+y_2^2(k-1)+y_4^2(k-1)} + 0.5u_2(k-1)$$

където $u_1(k)$ и $u_2(k)$ са входовете, а $y_1(k)$ и $y_3(k)$ са изходите. В ролята на входове са използвани хаотичните времеви серии на Макий-Глас и Рослер. Получените резултати са представени на фиг. 4.1.6.1 до фиг. 4.1.6.4.

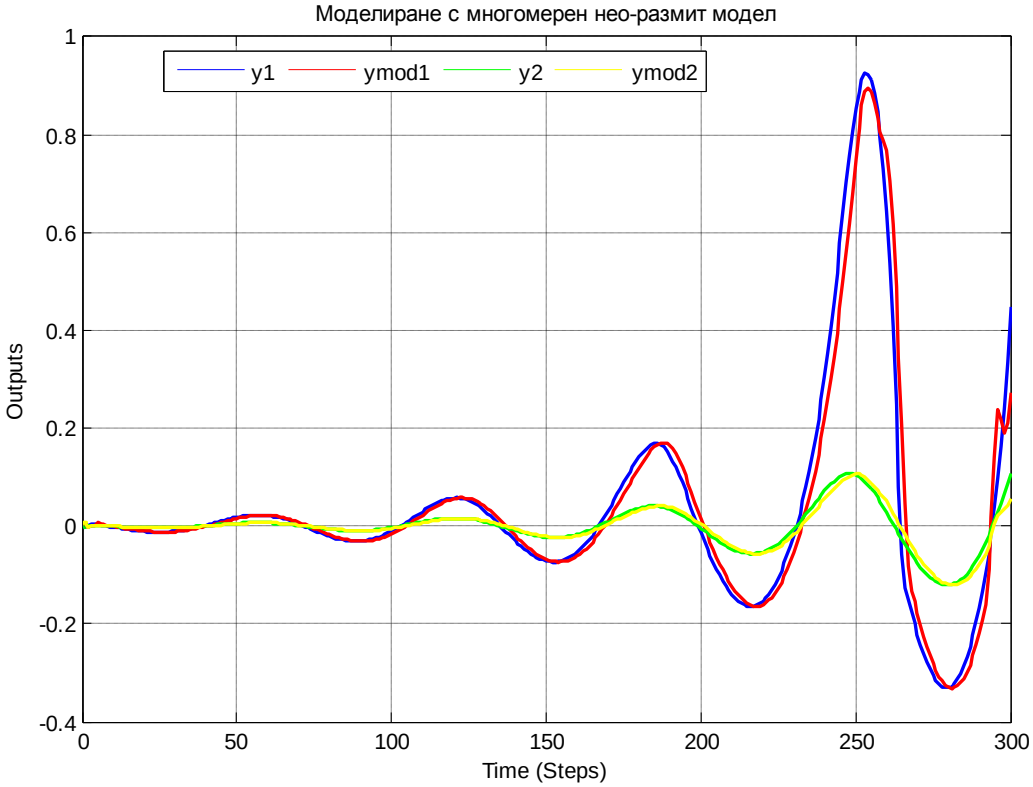
Обучението на многомерния нео-размит модел е направено с описания в т.3.6.1 двустъпков градиентен алгоритъм, като за целта са използвани изразите (3.3.10) и (3.3.11). Въведена е и адаптивна скорост на обучение **η**, чиято стойност се определя чрез зависимостта (3.3.12).



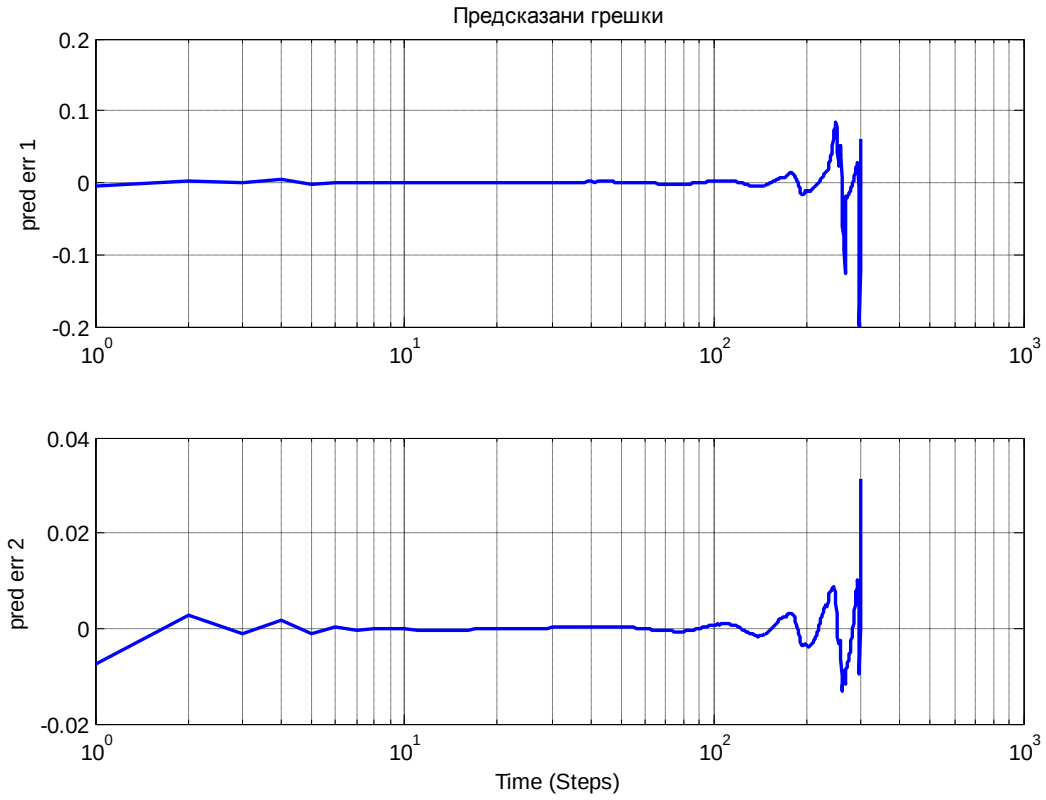
Фиг.4.1.6.1 Моделиране на Макий-Глас хаотична времева серия с многомерен нео-размит модел



Фиг.4.1.6.2 Предсказани грешки при многомерен нео-размит модел



Фиг.4.1.6.3 Моделиране на Рослер хаотична времева серия с многомерен нео-размит модел



Фиг.4.1.6.4 Предсказани грешки при многомерен нео-размит модел

4.2. Реализиране на обобщен предсказващ регулатор с използване на проектираните модели

4.2.1. Използвани обекти за управление

А/ Химичен реактор

Използваният химичен реактор представлява опростен модел на полимерен реактор (Continuous Stirred Tank Reactor - CSTR). Той е подробно разгледан в Ray (1981) и може да бъде описан със следната система нелинейни диференциали уравнения:

$$\dot{x}_1 = x_1 + D_a(1-x_1)e^{\left(\frac{x_2}{1+\frac{x_2}{\varphi}}\right)}$$
$$\dot{x}_2 = -(1+\delta)x_2 + BD_a(1-x_1)e^{\left(\frac{x_2}{1+\frac{x_2}{\varphi}}\right)} + u\delta \qquad (4.2.1.1)$$

където x_1 е концентрацията на продуктите в реактора, x_2 е температурата в реактора, u е темпаратурата на охлаждането на реактора, а B, D_c δ ,φ са физичеки параметри на реактора и имат следните стойности: $D_a = 0{,}072; \varphi = 20; B = 8; \delta = 0{,}69$.

Б/ Лабораторна топлинна система

Схемата на използваната лабораторна топлинна система е представена на фиг.4.1.1. Тя се намира в Машинния факултет на Техническия университет в Прага и се състои от два топлинни контура, през които циркулира вода. Движението на водата се осъществява посредством две помпи, съответно по една за двете вериги. Топлинният източник на системата е електрически нагревател, разположен по главния (първичен) контур. В топлообменика, който се намира в центъра на схемата се осъществява топлинният обмен между двата контура. Друг основен компонент е охладителят, който принадлежи към втория контур. Както се вижда от схемата, отделните части на системата са свързани по между си чрез тръби, което е причина за наличието на чисти закъснения.

Измерваните температури в системата са следните: ϑ_h - температура на водата на входа на топлообменика; ϑ_a - температура на водата на

изхода на топлообменика ; ϑ_d, ϑ_c - входната и изходна температура на водата към вентилатора. Управляващият вход на системата е напрежението u . То се подава към сервомеханизъм, регулиращ позицията на вентил. Положението на вентила определя съотношението на вода от нагревателя и вода от топлообменика, като по този начин се регулира температурата ϑ_h. Целта е управлението на температурата на водата на входа на топлообменника- ϑ_h. Други входове на системата са скоростта на перката на вентилатора и топлинното изпълнение на нагревателя.

Тъй като енергийната ефективност на топлообменника е много висока, загубата на топлина по време на обмяната може да се пренебрегне. Балансирайки транспортирането, акумулирането и обмяната на топлината, моделът от този участък на системата може да се представи чрез следния израз:

$$T_a \frac{d\Delta\vartheta_a(t)}{dt} = K_a[\Delta\vartheta_h(t) - \frac{1+q}{2}\Delta\vartheta_a(t) - \frac{1-q}{2}\Delta\vartheta_c(t-\tau_e)] - \dots \\ -[\Delta\vartheta_a(t) - \Delta\vartheta_c(t-\tau_e)] \quad (4.2.1.2)$$

където T_a е константата на акумулиране (натрупване), K_a е топлинен предавателен коефициент и τ_e е времето на закъснение.

Като втори участък от системата може да се разгледа свързващата тръба между топлообменика и вентилатора, който се описва със следното уравнението:

$$T_d \frac{d\Delta\vartheta_d(t)}{dt} = -\Delta\vartheta_d(t) + K_d\Delta\vartheta_a(t-\tau_d) \quad (4.2.1.3)$$

където T_d е времева константа, τ_d е закъснението и K_d е статичен усилващ коефициент.

Следващото уравнение описва втория контур на системата:

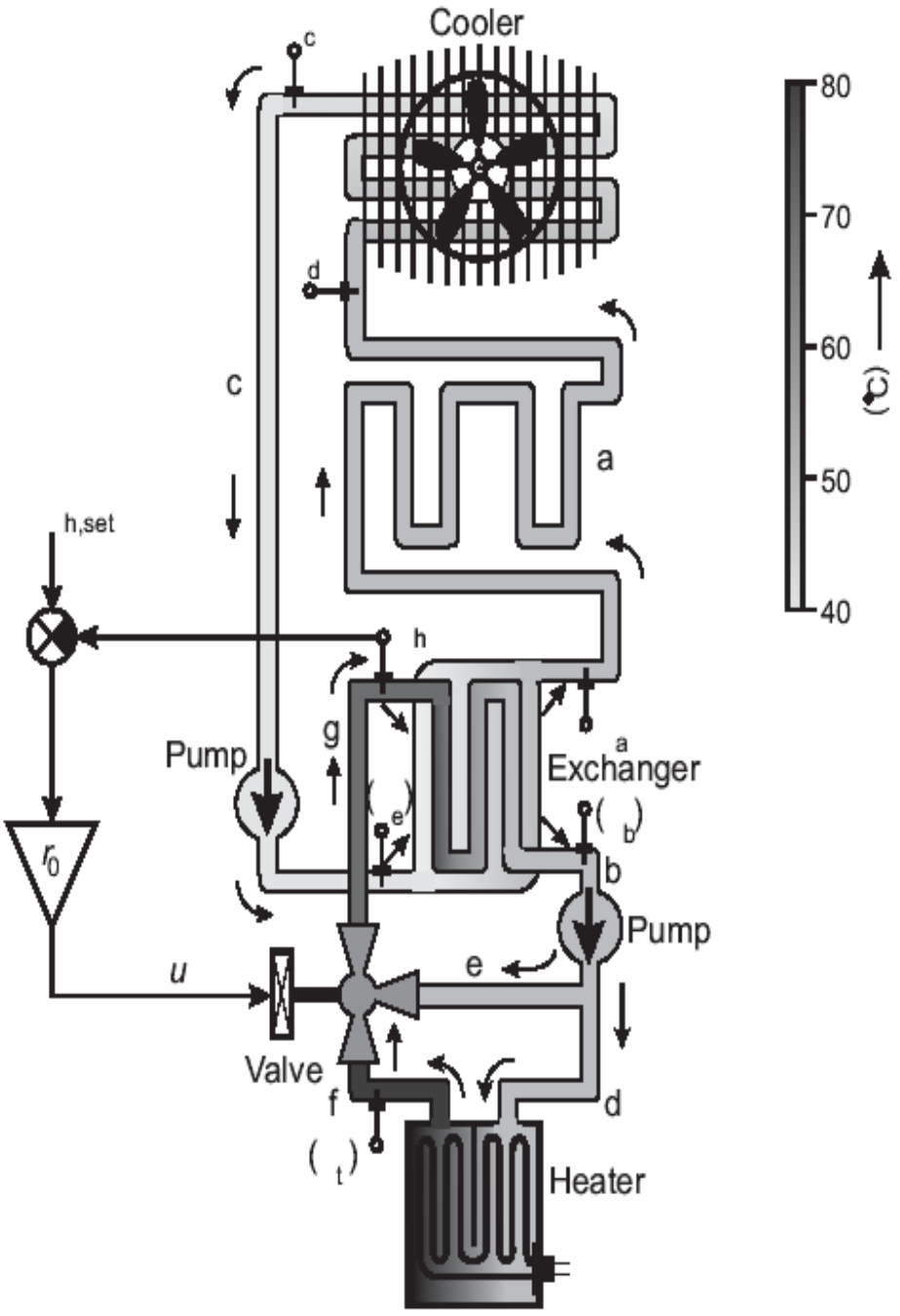
$$T_c \frac{d\Delta\vartheta_c(t)}{dt} = -\Delta\vartheta_c(t-\eta_c) + K_c\Delta\vartheta_d(t-\tau_c) \quad (4.2.1.4)$$

където T_c е времеконстанта, τ_c и η_c са закъснения и K_c е статичния коефициент на усилване.

Последното уравнение на системата описва по-голяма част от главния (първичен) контур:

$$T_h \frac{d\Delta\vartheta_h(t)}{dt} = -\Delta\vartheta_h(t-\eta_h) + K_b\Delta\vartheta_a(t-\tau_b) + K_u\Delta\vartheta_{h,set}(t-\tau_u) \quad (4.2.1.5)$$

$\Delta\vartheta_{h,set}$ е заданието, T_h е времеконстанта, τ_b, τ_u и η_h са закъснения, K_b и K_u са статични коефициенти на усилване.



Фиг. 4.2.1.1 Структурна схема на единия контур от топлинната система

Прилагайки Лапласова трансформация към описаните по-горе уравнения, моделът на лабораторната топлинна система може да се запише в следния вид:

$$sx(s) = A(s)x(s) + B(s)\Delta\vartheta_{h,set}(s) \qquad (4.2.1.6)$$

където $x(s) = [\Delta\vartheta_h(s)\Delta\vartheta_a(s)\Delta\vartheta_d(s)\Delta\vartheta_c(s)]^T$ е вектор на променливите на състоянието,

$$A(s) = \begin{bmatrix} \frac{-\exp(-\eta_h s)}{T_h} & \frac{K_b \exp(-\tau_b s)}{T_h} & 0 & 0 \\ \frac{K_a}{T_a} & \frac{-(1+0.5K_a(1+q))}{T_a} & 0 & \frac{(1-0.5K_a(1-q))\exp(-\tau_e s)}{T_a} \\ 0 & \frac{K_d \exp(-\tau_d s)}{T_d} & \frac{-1}{T_d} & 0 \\ 0 & 0 & \frac{K_c \exp(-\tau_c s)}{T_c} & \frac{-\exp(-\eta_c s)}{T_c} \end{bmatrix}$$

$$\mathbf{B}(s) = \begin{bmatrix} \frac{K_u \exp(-\tau_u s)}{T_h} & 0 & 0 & 0 \end{bmatrix}^T$$

Стойностите на използваните параметри са:

$T_h = 14s$, $K_b = 0.24$, $K_u = 0.39$, $\eta_h = 6.5s$, $\tau_b = 40s$, $\tau_u = 13.2s$

$T_a = 3s$, $K_a = 1$, $\tau_e = 13s$, $q = 1$ ($m_1 = m_2 = 0.08m^3/hour$)

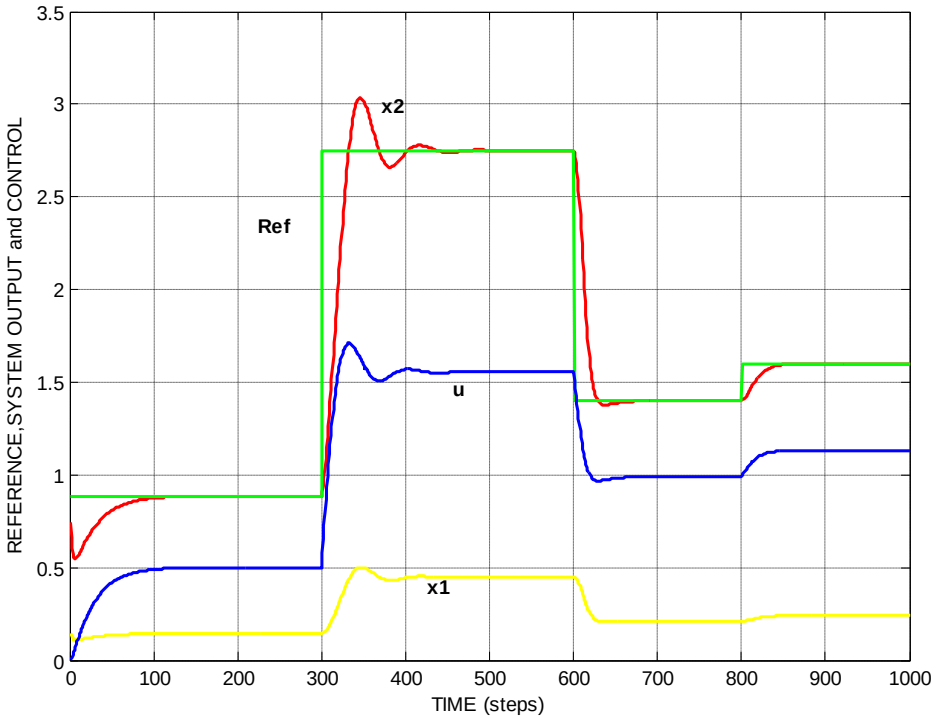
$T_d = 3s$, $K_d = 0.94$, $\tau_d = 18s$

$T_c = 25s$, $K_c = 0.81$, $\eta_c = 9.2s$, $\tau_c = 2.8s$

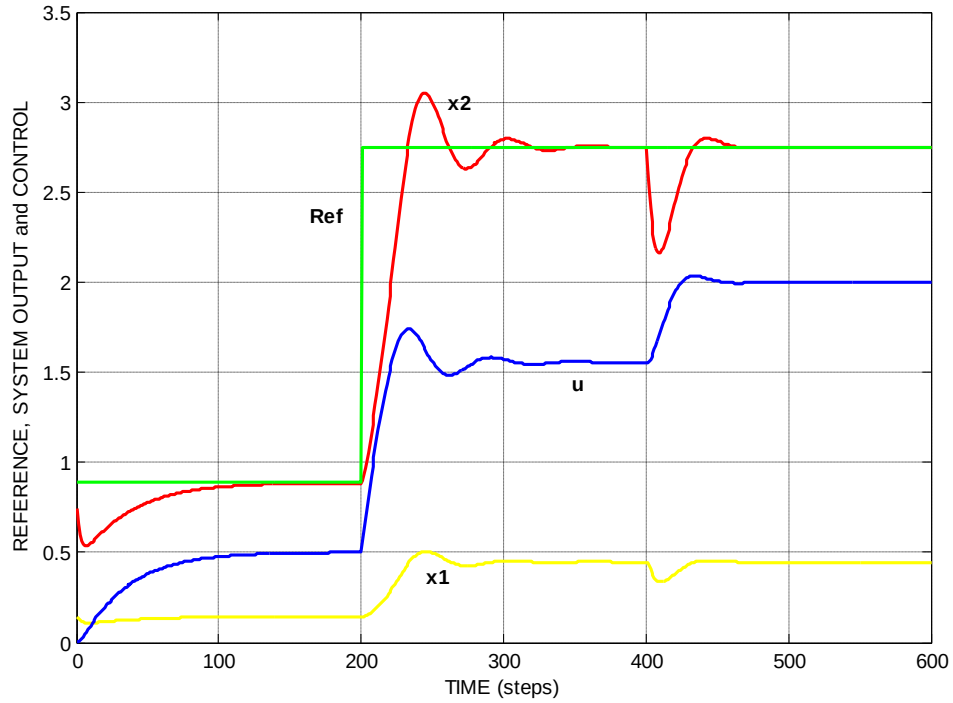
Фиг. 4.2.1.2 Лабораторна топлинна система

4.2.2. Нелинейно невронно-размито обобщено предсказващо управление

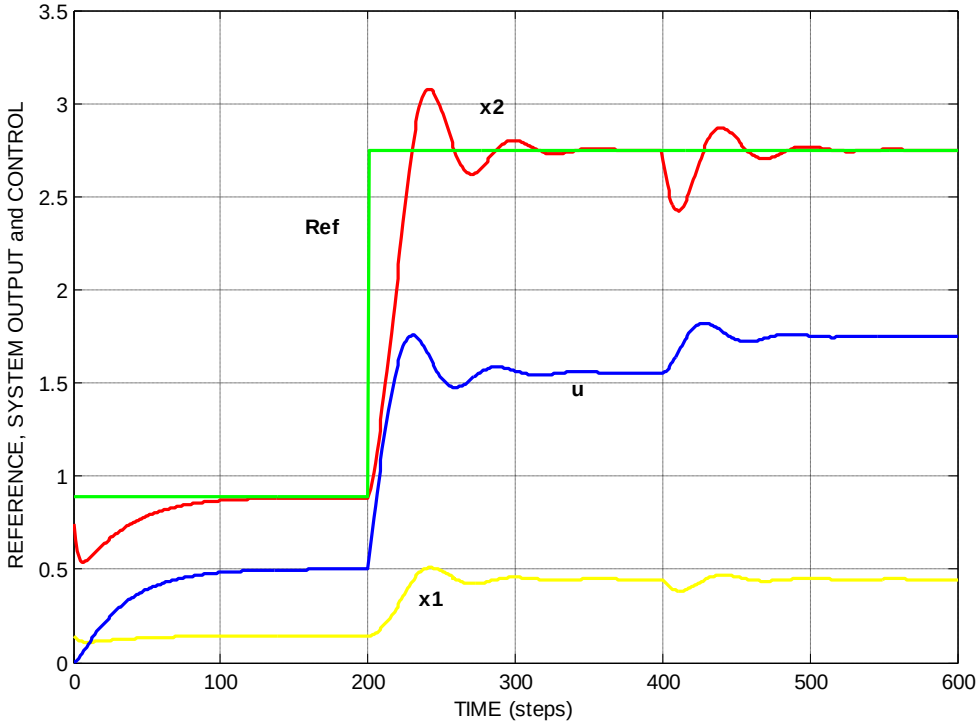
Всеки един от представените по-горе модели е включен в обобщен предсказващ регулатор, реализиран чрез итеративния оптимизационен алгоритъм от първи ред, описан в т.2.2. GPC регулатор базиран на DANFA модела със следните параметри N_1=1; N_p=7; N_u=3; ρ=0.1 е използван за управление на химичен реактор. Получените резултати са представени на фиг.4.2.2.1 до фиг.4.2.2.4. Всички фигури показват поведението на обекта и стойността на управлението. На фиг.4.2.2.1 е представен преходният процес при променливо задание. Симулационните резултати при промяна на параметрите на обекта са показани на фиг.4.2.2.2 В този случай на стъпка 400 стойността на топлинния коефициент δ и на B са едновременно променени съответно на 0.8 и 7.5. Изследвана е възможността на регулатора да се справя със смущения (фиг.4.2.2.3), като за целта на стъпка 400 е подадено стъпално смущение с амплутида 0.5. На фиг.4.2.2.4 е показано как се справя регулаторът в най-тежкия случай, когато едновременно имаме промяна в параметрите на обекта и добавено смущение.



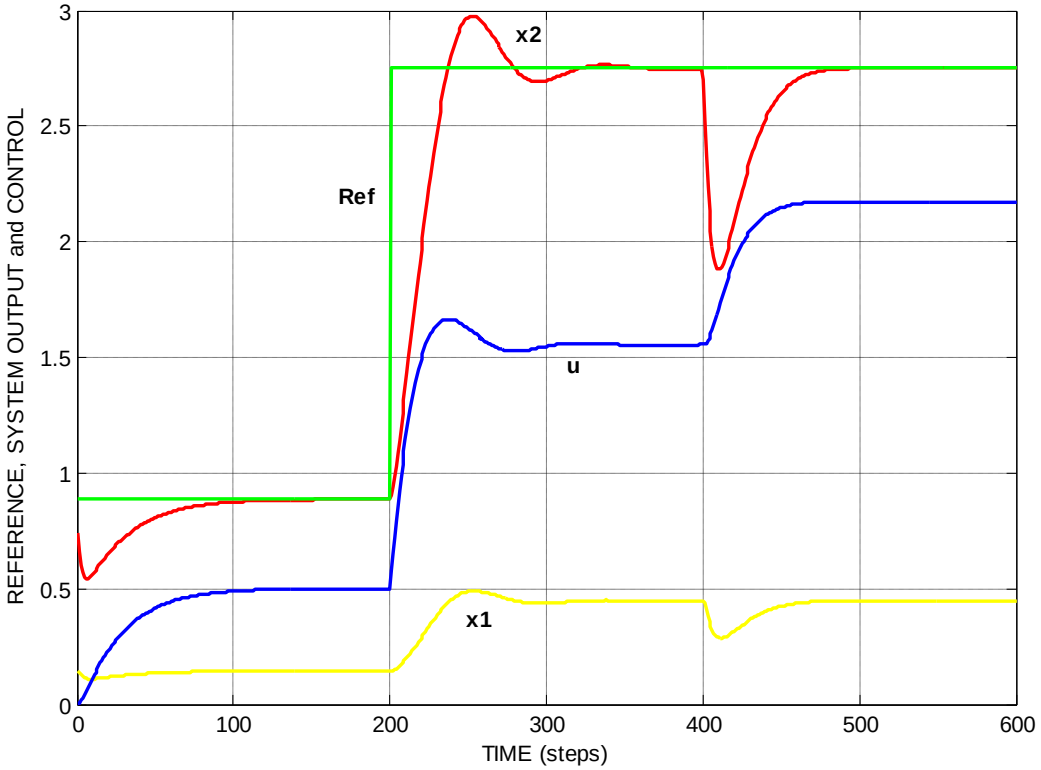
Фиг. 4.2.2.1 Преходен процес при променливо задание



Фиг. 4.2.2.2 Преходен процес при променливо задание и промяна в параметрите на CSTR - B=7.5, δ=0.8 в стъпка 400



Фиг. 4.2.2.3 Преходен процес при променливо задание и добавяне на смущение в стъпка 400

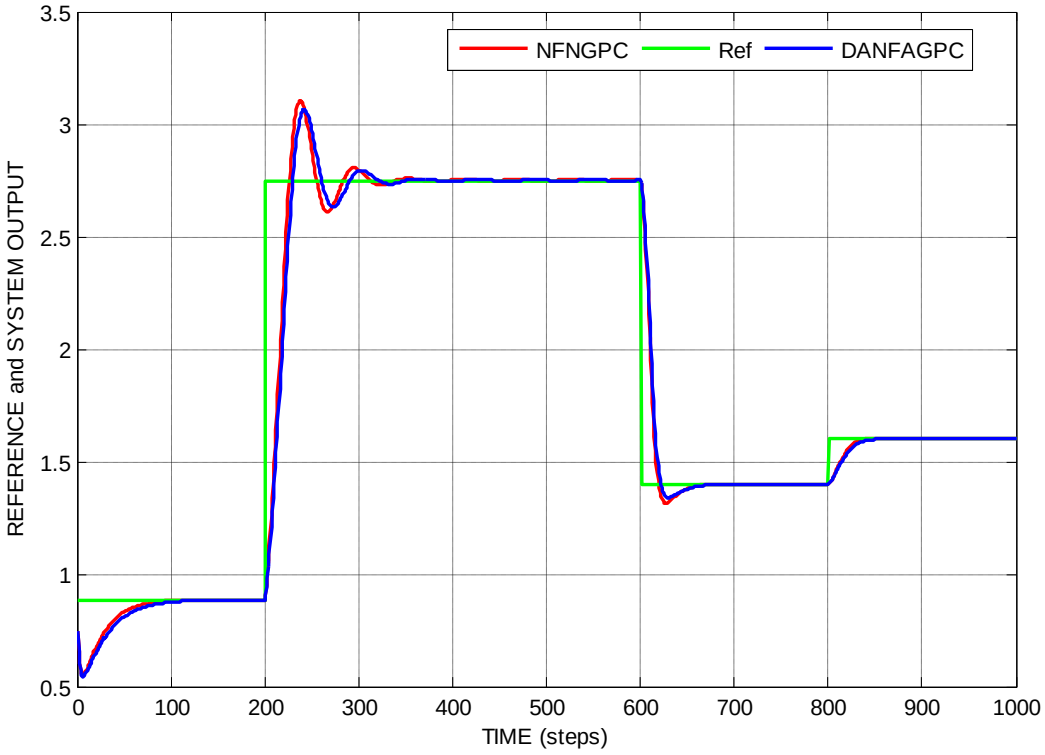


Фиг. 4.2.2.4 Преходен процес при променливо задание, промяна в параметрите на CSTR и добавяне на смущение в стъпка 400

Подобни резултати се получават при управлението на химичния реактор с обобщен предсказващ регулатор, при който за модел са използвани и другите предложени модели - SFNN в трите му варианта и нео-размит, реализиран с Тип 2 и Интуиционистка размита логика. Поради това е счетено за излишно представянето им на отделни графики. В подкрепа на тази теза едно примерно сравнение на обобщен предсказващ регулатор с DANFA модел и с нео-размит модел е представено на фиг.4.2.2.5. Разликата в получените преходни процеси може да се счита за пренебрежима и се дължи основно на факта, че в регулатора се използвани различни по топология невронно-размити модели (т.е. с различен брой размити правила и параметри). Следователно може да се направи извода, че структурата на използвания невронно-размит модел не оказва влияние върху качеството на управление.

Друг интересен извод може да се направи по отношение на времето t, необходимо за реализиране на всички изчисления в оптимизатора за един такт. Стойностите на t при обобщен предсказващ регулатор с използване

на различни невронно-размити модели са представени в Таблица 4.2.2.1. Става ясно, че времето за работа на обобщен предсказващ регулатор с невронно-размит модел описан в т.2.1 е приблизително два пъти по-голямо в сравнение с това на регулатор базиран на някой от предложените в настоящата дисертация модели. Трябва да се отбележи, че това време t нараства с увеличаване на хоризонтите на предсказване N_p и на управление N_u.

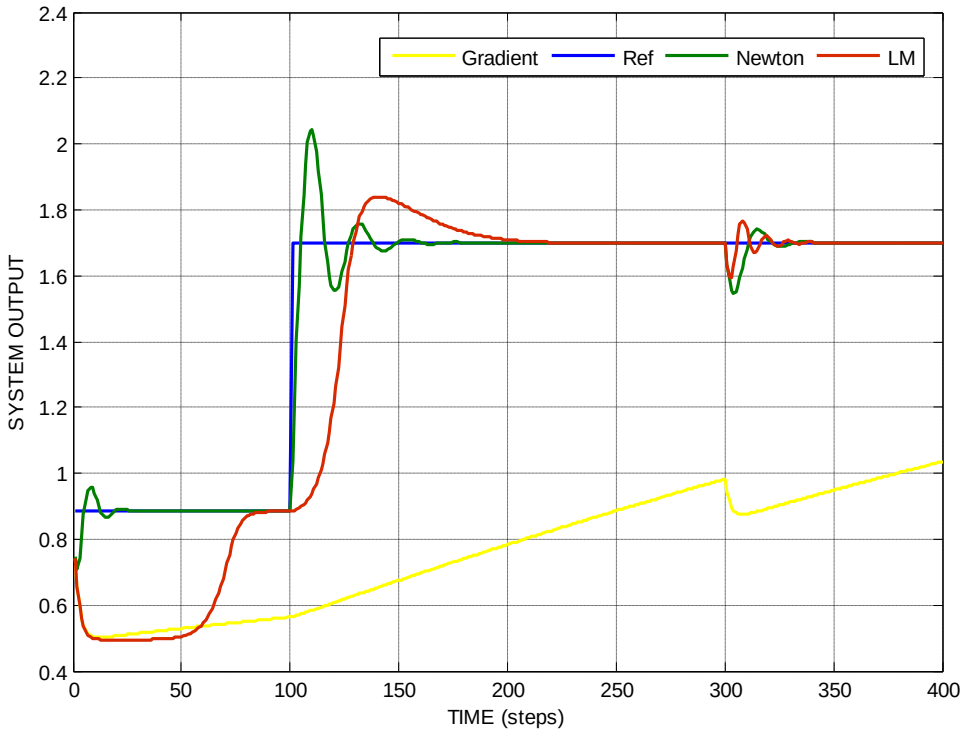


Фиг. 4.2.2.5 Преходен процес при променливо задание, промяна в параметрите на обекта и добавяне на смущение в стъпка 400

Таблица 4.2.2.1

Регулатор	N_1=1; N_u=3; N_p=5	N_1=1; N_u=3; N_p=10	N_1=1; N_u=7; N_p=10
	t [s]	t [s]	t [s]
DANFAGPC	0.0010	0.0020	0.0045
SFNNGPC	0.0012	0.0022	0.0037
NFNGPC	0.0008	0.0016	0.0038
T2NFNGPC	0.0010	0.0021	0.0039
INFNGPC	0.0009	0.0017	0.0035
NFMGPC	0.0019	0.0038	0.0063

Добре известен факт е, че методът за оптимизация оказва съществено влияние върху изчислителната ефективност на оптимизатора. В настоящата дисертация са разработени алгоритми от втори ред, а именно тези на Нютон и на Левенберг-Маркгуард, тъй като при тях е намален броят на итерациите, свързани със сходимостта на алгоритъма, а това от своя страна води до по-голямо бързодействие на предсказващия регулатор. Последното е видно от фиг.4.2.2.6, на която е направено сравнение между обобщен предсказващ регулатор с SFNN модел, използващ различни методи за оптимизация – градиентен от първи ред, Нютон и Левенберг-Маркгуард. При едни и същи стойности на хоризонтите на управление и предсказване и на тегловният коефициент ρ (N_1=1; N_u=5; N_p=5; ρ=0.002) в рамките на 100 стъпки регулаторът с градиентен метод не може да достигне заданието. В същото време за регулатор с метод на Нютон са необходими около 25 стъпки, а на регулатор с метод на Левенберг-Маркгуард около 70 стъпки. На стъпка 300 е подадено смущение, което много бързо е отработено от регулаторите с Нютон и Левенберг-Маркгуард оптимизация, докато регулаторът с градиентен метод изпитва затруднение.



Фиг. 4.2.2.6 Сравнение на обобщено предсказващо управление, реализирано с различни оптимизационни алгоритми

Основен проблем при градиентните методи от втори ред е необходимостта на всеки такт да се намира обратната матрица на Хесе. Това е процедура, която изисква допълнителни изчисления и следователно ще доведе до увеличаване на времето t, необходимо за реализиране на изчисленията в оптимизатора за един такт. Данни за това са представени в Таблица 4.2.2.2, като на хоризонтите на управление и на предсказване са еднакви (N_1=1; N_u=N_p=5; ρ=0.002).

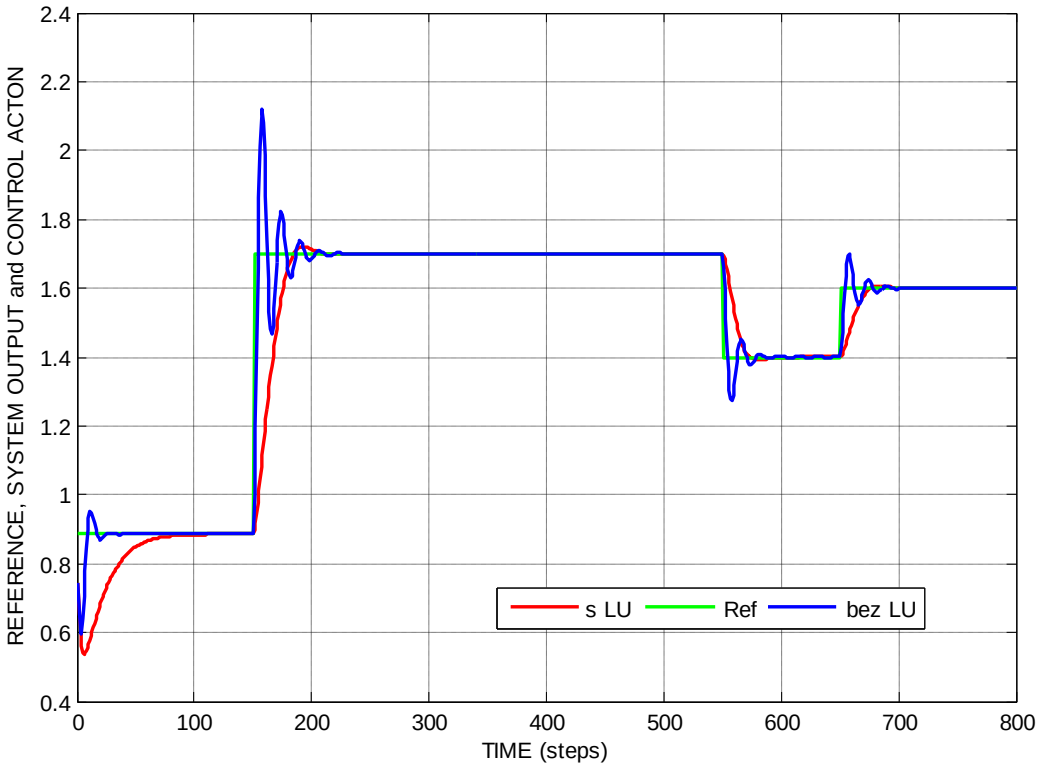
Таблица 4.2.2.2

Регулатор	Метод на Нютон	Метод на Левенберг-Маркгуард	Метод на Нютон с LU декомпозиция	Метод на Левенберг-Маркгуард с LU декомпозиция
	t [s]	t [s]	t [s]	t [s]
DANFAGPC	0.0029	0.0029	0.0026	0.0026
SFNNGPC	0.0026	0.0026	0.0022	0.0022
NFNGPC	0.0023	0.0023	0.0019	0.0019
T2NFNGPC	0.0027	0.0027	0.0021	0.0021
INFNGPC	0.0022	0.0022	0.0018	0.0018
NFMGPC	0.0055	0.0055	0.0047	0.0047

Резултатите от табл. 4.2.2.2 показват, че в изчислителен план разлика между метода на Нютон и този на Левенберг-Маркгуард на практика не съществува. Що се отнася до качеството на управление метода на Нютон осигурява по-бързо достигане до заданието, но и преходен процес с пререгулиране.

За да се избегне изчислително тежката процедура по определянето на обратната матрица на Хесе на всеки такт при градиентните методи от втори ред е използван методът на LU декомпозицията, вид Гаусова елиминация, която в програмната среда *Matlab* се осъществява с операцията дясно деление. На фиг.4.2.2.7 са представени преходните процеси при управление на химичния реактор с обобщен предсказващ регулатор с Нютон метод за оптимизация с изчисляване на обратната матрица на Хесе и с LU декомпозиция. Вижда се, че с въвеждането на LU декомпозиция колебанията в преходния процес се „изглаждат” и

характерът му се свежда до апериодичен. Освен това, от данните в табл.4.2.2.2 става ясно, че при модифицираните алгоритми на Нютон и на Левенберг-Маркгуард времето t за реализиране на изчисленията в оптимизатора за един такт намалява.

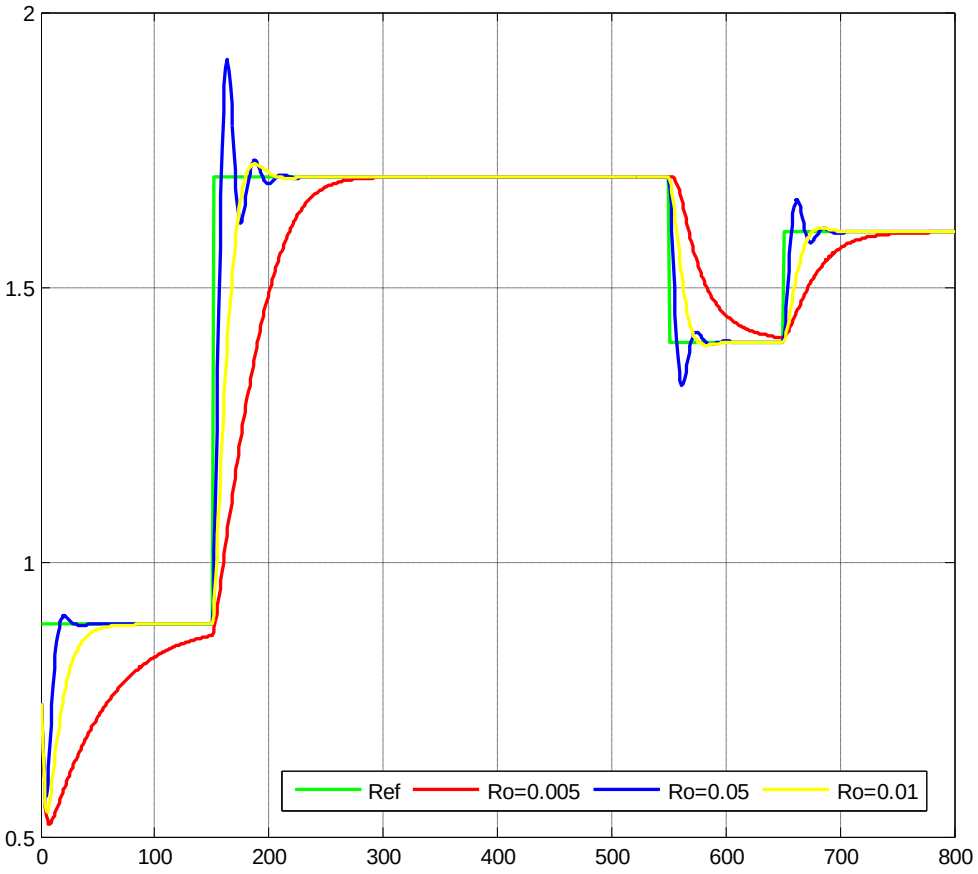


Фиг. 4.2.2.7 Преходен процес при променливо задание, промяна в параметрите на обекта и добавяне на смущение в стъпка 400

За да се демонстрира влиянието на тегловния коефициент ρ от целевия критерий (1.4.1) върху качеството на управление, на фиг.4.2.2.8 са показани преходните процеси при управление на химичния реактор с обобщен предсказващ регулатор с SFNN модел и Нютон метод за оптимизация с изчисляване на обратната матрица на Хесе при различни стойности на ρ (ρ_1=0.005; ρ_2=0.001; ρ_3=0.05). Изхождайки от графиката може да се заключи, че при малки стойности на ρ преходният процес е апериодичен, но времето за достигане на заданието е по-голямо. С увеличаване на стойността на ρ преходният процес придобива колебателен характер, но значително се съкращава времето за достигане на заданието. Или с други думи влиянието на тегловния коефициент ρ върху качеството на управление е съществено и подобно на това на пропорционалната

съставка на ПИД регулатора. Ето защо е реализиран алгоритъм на обобщен предсказващ регулатор, при който се осъществява адаптивна супервайзорна донастройка на този коефициент. Алгоритъмът, който е изграден на базата на невронно-размитията модел, описан в т.2.1 и Нютонова оптимизационна процедура, е тестван в реални условия за управление на топлинна система. Експериментите са направени при следните стойности на параметрите:

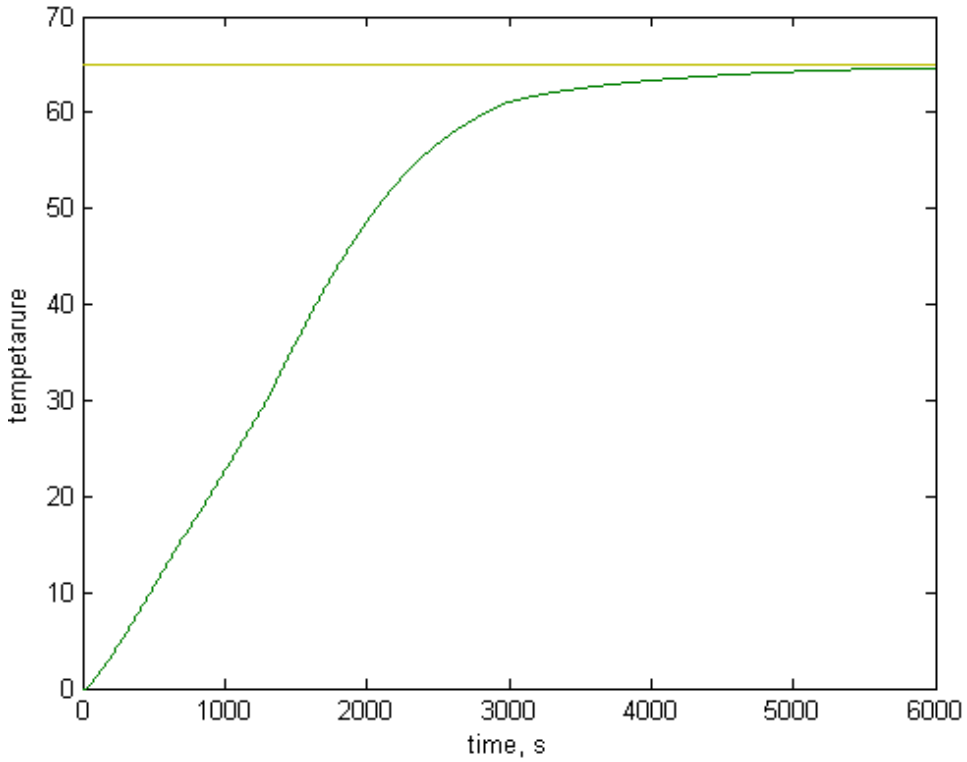
- Параметри на регулатора: $N_1=1$ $N_u=3$ $N_2=5$
- Постоянно задание: $r=65$ °C
- Променливо задание: *r=65 °C при t=[0:2000]s; r=70 °C при t=[2000:3000]s; r=55 °C при t=[3000:5000]s*
- Скорост на обучение на невронно-размитията модел: η=0.09
- Скорост на обучение на невронно-размитията модел в супервайзора: η_{sup}=0.07.



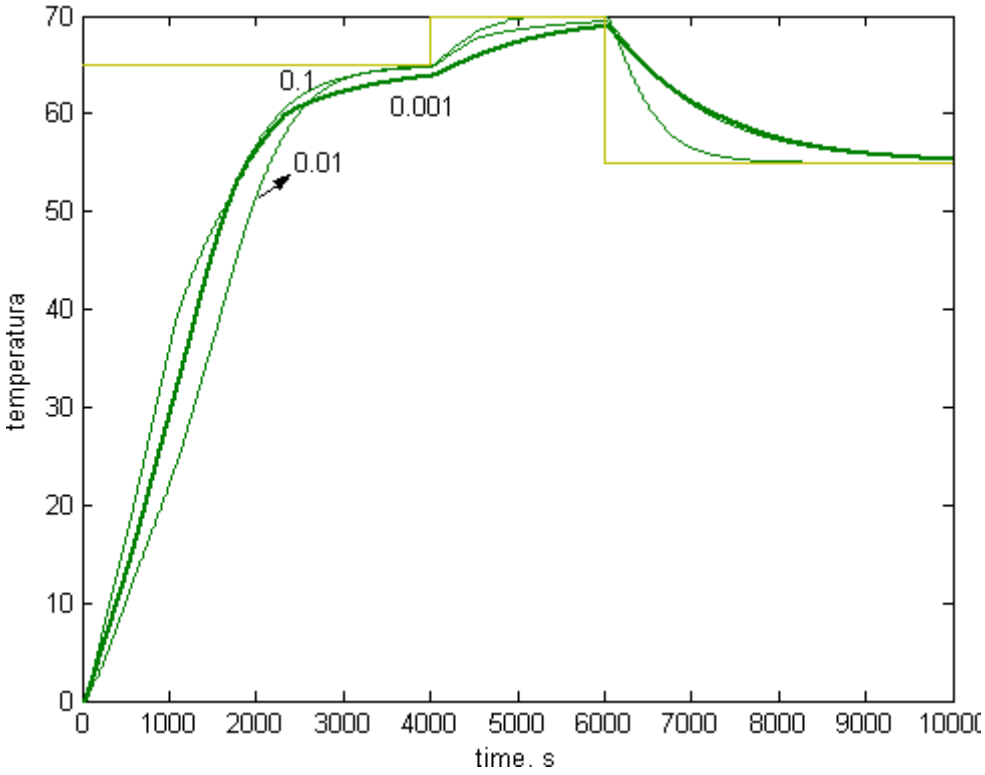
Фиг. 4.2.2.8 Преходен процес при променливо задание, промяна в параметрите на обекта и добавяне на смущение в стъпка 400

Супервайзорният елемент е реализиран в два варианта - с механизъм на извеждане Мамдани и с механизъм на извеждане Такаги-Сугено. Резултатите, получени в първия случай са показани на фиг.4.2.2.9 и фиг.4.2.2.10. Поведението на системата при еднакъв брой размити правила в структурата на супревайзора и различен начален коефициент ρ показват проява на сходна динамика на процесите както коментираната по-горе. Вижда се, че изборът на начална стойност на този параметър е определящ за динамиката на преходния процес. Увеличаването му води до по-големи стойности на пререгулирането и по-бързото достигане на желаната стойност.

Също така трябва да се отбележи и фактът, че наличието на модификации в схемата на предсказващия регулатор свързани с допълнителни изчисления води до значително влияние върху времето на преходния процес. От друга страна това забавяне получено от наличието на супервайзора в управляващия контур може да се компенсира с по-голям начален коефициент ρ. Това запазва динамиката на процеса, като влияе благоприятно върху намаляване на времето на преходния процес и максималното динамично отклонение.

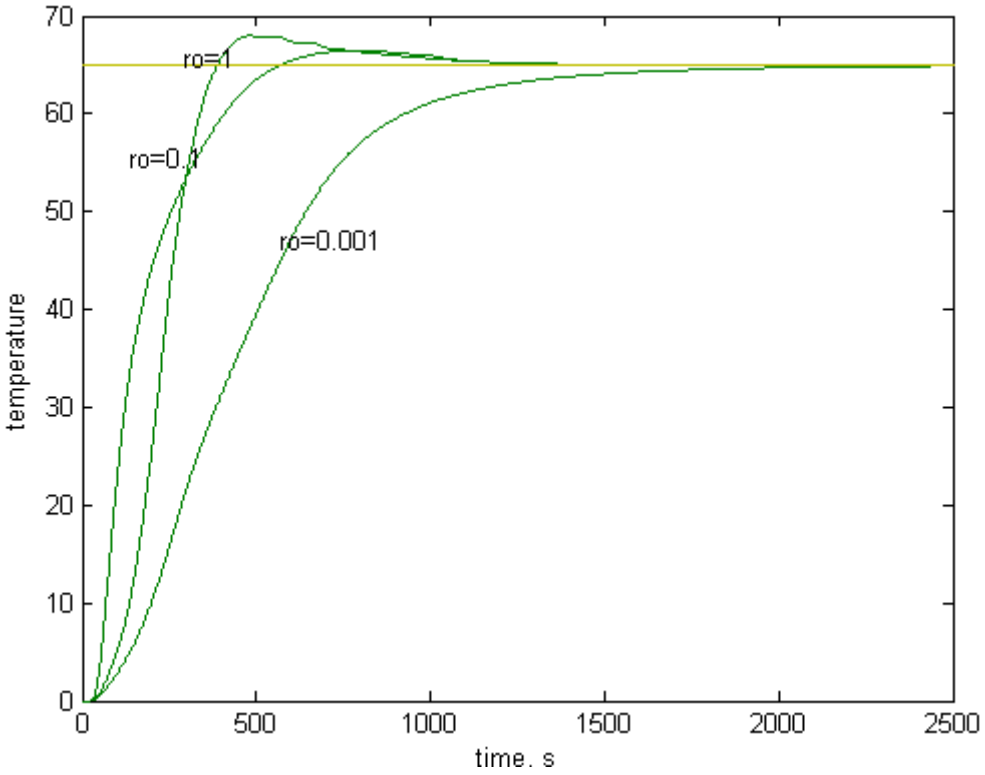


Фиг.4.2.2.9 Преходен процеси при постоянно задание и при начално ρ=0.001

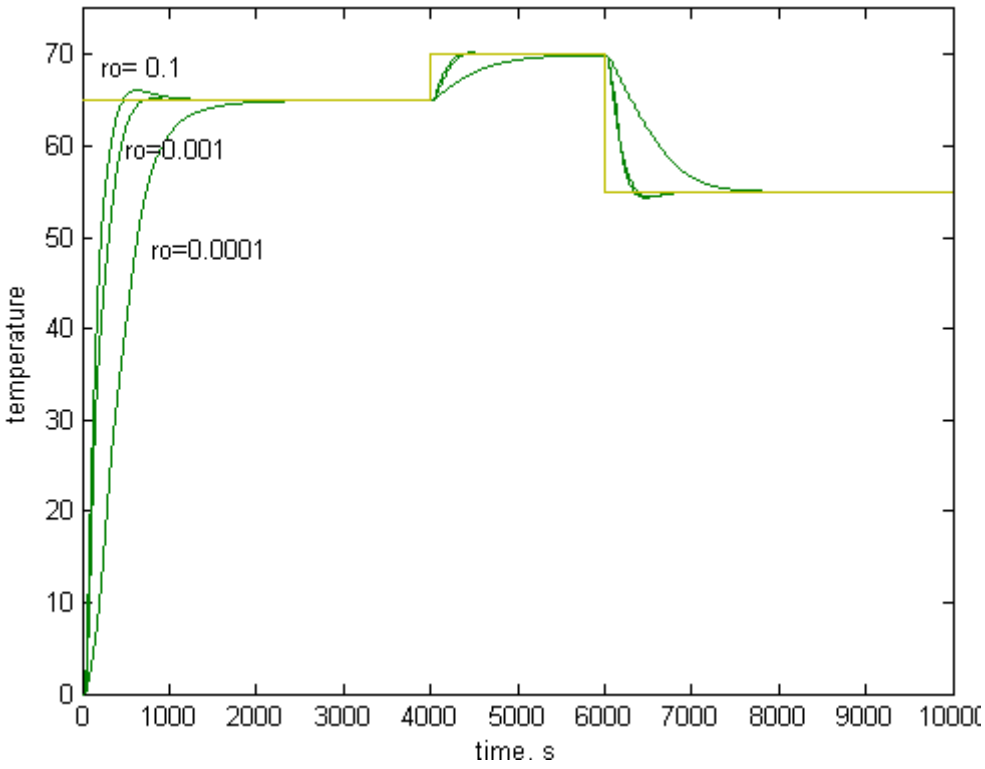


Фиг.4.2.2.10 Преходни процеси при променливо задание и различни начални стойности на ρ

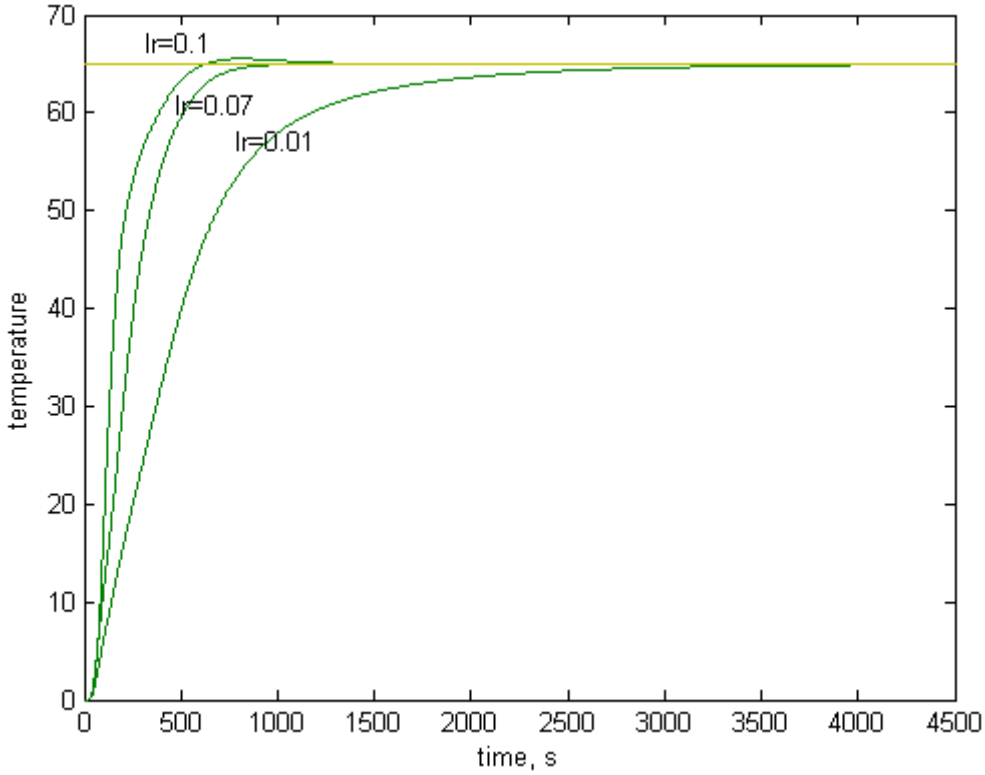
По-долу са представени резултатите с използване на супервайзорен елемент с механизъм на извеждане Такаги-Сугено. На фиг.4.2.2.11 са дадени преходните процеси при постоянно задание и при различни начални стойности на ρ, а на фиг.4.2.2.12 при променливо задание и различни начални стойности на ρ. Използваните различни начални стойности на тегловния коефициент са както следва: ρ=1; ρ =0.1; ρ=0.001 при постоянно задание и при променливо: ρ=0.1; ρ=0.001; ρ=0.0001. Скоростта на обучение на тегловния коефициент в случая е избрана да бъде постоянна η_{sup}=0.07. На фиг.4.2.2.13 са представени резултати при различни стойности на скоростта на обучение на невронно-размития модел в супервайзора. Началният коефициент е приет да бъде ρ=0.001 съответно при η_{sup} =0.1; η_{sup} =0.07; η_{sup} =0.01. Изменението на ρ при различни начални стойности е показано по-долу на фиг.4.2.2.14, където е прието: η_{sup} =0.07 и ρ=0.1; ρ=0.001; ρ=0.0001.



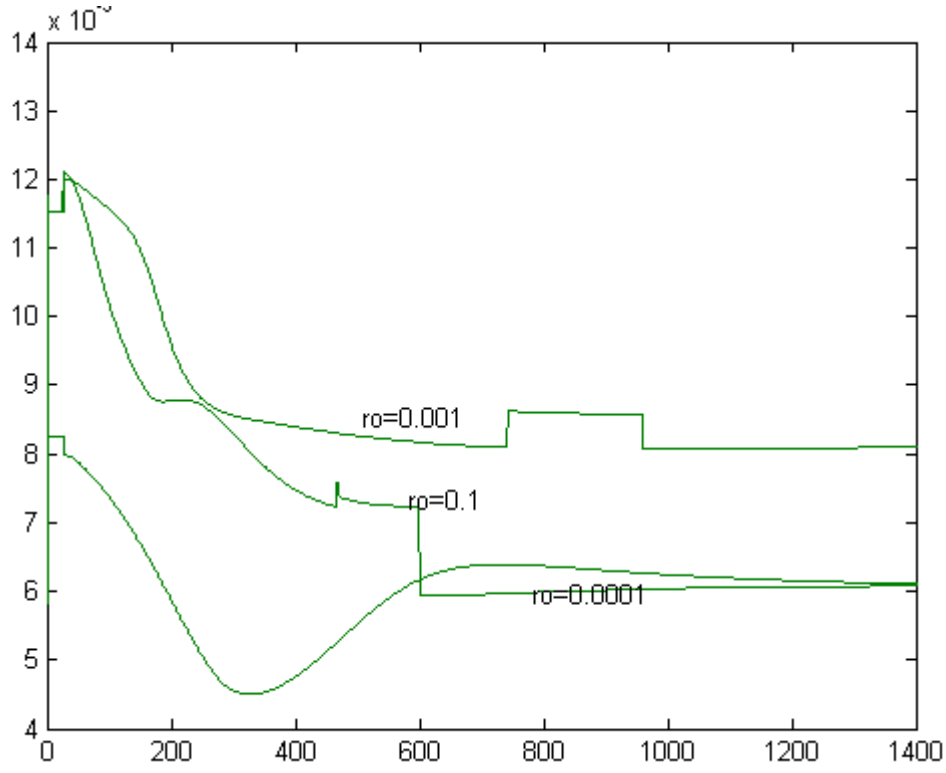
Фиг.4.2.2.11 Преходни процеси при постоянно задание и при различни начални стойности на ρ



Фиг.4.2.2.12 Преходни процеси при променливо задание и при различни начални стойности на ρ

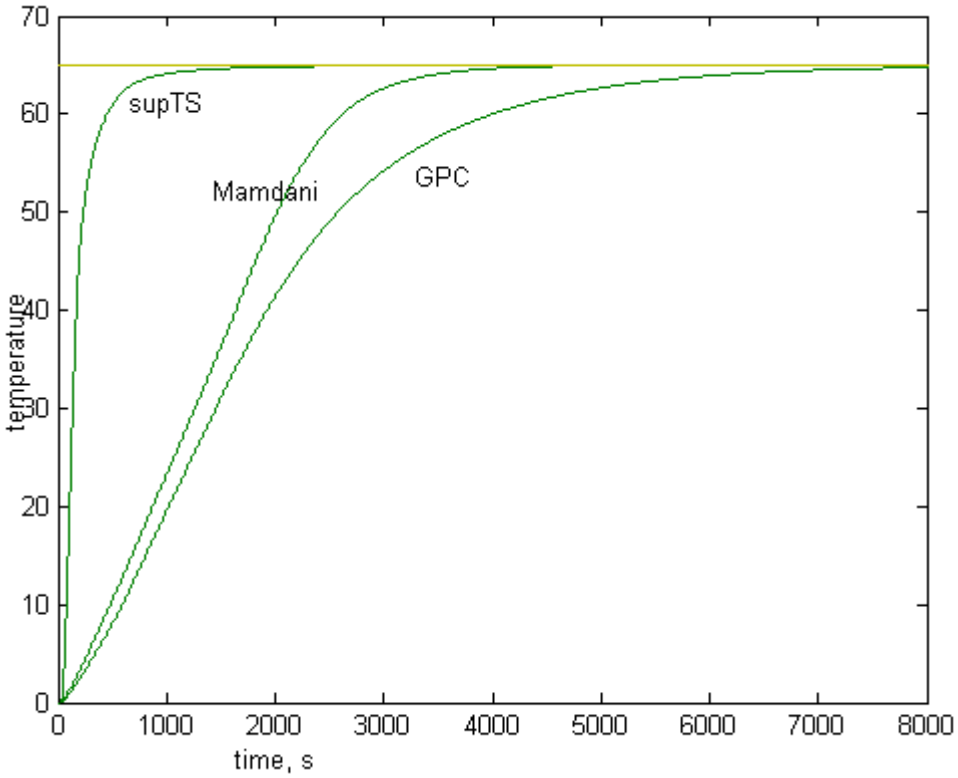


Фиг.4.2.2.13 Преходни процеси при различни стойности на скоростта на обучение



Фиг.4.2.2.14 Изменението на ρ при различни начални стойности

Сравнението между различните типове схеми на управление е направено при еднакви условия (фиг.4.2.2.15). Стойността на тегловния коефициент при *GPC* управлението е приет да бъде $\rho=0.001$, съответно и за началните стойности при двата метода със супервайзор.



Фиг.4.2.2.15 Сравнение между различните методи на управление

От показаните преходни процеси се вижда, че методът с допълнително супервайзорно ниво използващ размитите структури на Такаги-Сугено дава най-добри резултати. Адаптивната донастройка на тегловния коефициент осигурява бързодействие и качество на процеса. Въпреки фактът, че наличието на модификации в схемата на предсказващия регулатор свързани с допълнителни изчисления води до значително влияние върху времето на преходния процес, в случая не се отразява на бързодействието за достигане на желаната стойност. При метода използващ подхода на Мамдани за донастройката на ρ това обаче не е така (фиг.4.2.2.10). Ако се сравни със метода използващ Newton*GPC* управление, лесно може да се забележи забавянето получено от наличността на допълнителното супервайзарно ниво. То обаче може да се компенсира със задаването на по-висока начална стойност на тегловния коефициент, което намалява времето на преходния процес. Така от

направеното сравнение може да се стигне до заключението че, системата със супервайзорно донастройващо ниво използващо Такаги-Сугено модел е един по-добър вариант за управление на изследваната система, който осигурява по-добро качество и ефективност на процеса.

4.3. Изводи

Резултатите от симулационни изследвания и реални експерименти на проектираните алгоритми за регулатори с невронно-размито предсказващо управление са представени в тази част от дисертационния труд. Първоначално е изследвана способността на DANFA, SFNN и нео-размития модел да предсказват хаотични времеви серии. За обучението им е използван двустъпков градиентен алгоритъм с постоянна скорост на обучение $\eta=0.05$. Направено е сравнение на тези модели с невронно-размития модел, описан в т.2.1, при което за най-добър модел в съответствие със заложените критерии е определен модифицирания нео-размит модел. Въпреки, че работи с по-голям брой размити правила в сравнение с SFNN Type III, модифицираният нео-размит модел има по-малък брой параметри за актуализиране в процеса на обучение (дори и в случая с обучаване параметрите на размитите множества), по-точен е и по-бърз. Ето защо именно модифицираният нео-размит модел е реализиран във вариант с Тип 2 размита логика и с Интуиционистка размита логика в присъствието на шум, за да се види как се справя в условия на неопределености. От получените резултати става ясно, че Интуиционисткият нео-размит модел работи с по-малка грешка в сравнение с Тип 2 нео-размития модел. Времето за осъществяване на необходимите изчисления за една стъпка от обучението при Интуиционисткия нео-размит модел е съответно $1.48e^{-4}$ за случая без шум и $1.50e^{-4}$ с шум. При Тип 2 нео-размит модела това време е по-голямо – съответно $1.93e^{-4}$ и $1.96e^{-4}$ за случаите без и с шум. Следователно Интуиционисткият нео-размит модел е по-точен и по-бърз и това го прави подходящ за включване в предсказващ регулатор. Модифицираният нео-размит модел е разработен и в MIMO вариант, тъй като едно от предимствата на предказващите регулатори е способността им да се справят с многомерни обекти.

Всеки един от предложените невронно-размити модели е използван за реализиране на обобщен предсказващ регулатор с итеративен оптимизационен алгоритъм от първи и от втори ред. От получените резултати може да се заключи, че структурата на използвания невронно-размит модел не оказва влияние върху качеството на управление. С използването на предложените невронно-размити модели се съкращава приблизително два пъти времето t, необходимо за реализиране на всички изчисления в оптимизатора за един такт. Това време зависи също и от хоризонтите на предсказване и на управление и нараства с увеличаването им.

Бързодействието на невронно-размит обобщен предсказващ регулатор значително се увеличава когато се използва итеративен оптимизационен алгоритъм от втори ред. Основната трудност при тези алгоритми - изчисляването на обратната матрица на Хесе на всеки такт – е преодоляна с въвеждането на LU декомпозиция. От получените резултати се вижда, че с въвеждането на LU декомпозиция колебанията в преходния процес се „изглаждат” и характерът му се свежда до апериодичен. Освен това, от данните в табл.4.2.2.2 става ясно, че при модифицираните с помощта на Гаусовата елиминация алгоритми на Нютон и на Левенберг-Маркгуард времето t за реализиране на изчисленията в оптимизатора за един такт намалява.

Що се отнася до настройката на предсказващите регулатори един съществен извод, които може да се направи е, че тегловният коефициент ρ оказва съществено влияние върху качеството на управление, но не и върху времето за работа на оптимизатора. Действието му може да бъде сравнено с това на пропорционалната съставка на ПИД регулатора.

Глава V

Софтуерна реализация на предложените невронно-размити модели и алгоритми за регулатори с невронно-размито предсказващо управление

Софтуерната реализация на предложените в Глава III на настоящата дисертация невронно-размити модели с редуциран брой размити правила и алгоритми за предсказващи регулатори е осъществена в програмната среда *Matlab.* Тя е изключително популярна сред инженери, изследователи, научни работници и специалисти във всички области на науката и техниката. *Matlab* предлага големи възможности за извършване на аналитични преобразувания, изчисления и висококачествена визуализация на резултатите. *Matlab* има вграден програмен език от високо ниво, което дава възможност на всеки потребител да създава свои собствени програми и функции, решаващи определени задачи от областта на научните му интереси.

5.1. Структура и действие на програмния код на основен алгоритъм на невронно-размит модел

Програмите, реализиращи отделните невронно-размити модели с оптимизиран брой правила от Глава III, като цяло имат сходна структура и работят по един и същ начин. Те могат да бъдат условно разделени на следните части: инициализация, невронно-размит модел, обучение и визуализация на графичните резултати. В софтуерната реализация на отделните модели различия има главно във втора и трета част. Те са свързани с типа на програмираната невронно-размитата структура и с избрания метод за обучението й. В същото време инициализацията и представянето на резултатите се осъществяват по идентичен начин.

При инициализацията се избира броя на Гаусовите функции на принадлежност и се фиксират техните център и ширина; определя се броя на използваните в конкретният модел размити правила, който зависи от броя на входните сигнали и от броя на Гаусовите функции; задават се начални стойности на параметрите във функцията на Сугено. В настоящата дисертация е избрано да се работи с 3 Гаусови функции, чиито центрове са разпределени в интервала (0, 1). Това налага входните сигнали да бъдат нормализирани. Параметрите във функцията на Сугено се явяват тегла на връзките и като такива могат да приемат стойности в интервала (0, 1). Техните начални стойности се избират на случаен принцип като за целта се използва вградената в *Matlab* функция ***rand***, която генерира равномерно разпредалени числа в споменатия интервал.

Софтуерната реализация на конкретен невронно-размит модел се осъществява във втората част. Операциите, изпълнявани в нея, следват механизма на работа на модела и в най-общ вид имат следната последователност: размиване, генериране на база от размити правила, импликация и определяне на стойността на изхода на модела. Размиването се изпълнява от отделна подпрограма, към която се обръща основната програма. Всъщност са разработени три варианта на подпрограмата в зависимост дали се реализира размиване с размита логика Тип 1, Тип 2 или Интуиционистка размита логика. Пълният код на трите варианта на подпрограмата за размиване е представен в Приложението. Генерирането на размити правила се осъществява посредством цикъл, който има следният обобщен вид:

```
for i=1:(Nrules)
    foutn1(i)=b1(i)*x1n+b2(i)*x2n+b3(i)*x3n+b4(i)*x4n+b01(i);
end
```

Трябва да се отбележи, че при реализирането на DANFA, SFNN или модифициран нео-размит модел с този цикъл се генерират различен брой размити правила. Освен това самите правила също имат различна структура, т.е включват различен брой параметри. Например, за всеки

отделен подмодел на DANFA модела се генерират правила в следната форма:

```
for i=1:(Nrules)
        foutn1(i)=b1(i)*x1n+b2(i)*x2n+b01(i);
 end
```

Операцията импликация се осъществява посредством три вложени един в друг цикъла:

```
%Fuzzy Implication
 iii_1=0;
for i=1:a
        for j=1:a
                iii_1=iii_1+1;
                 if (cx1(i)==i) & (cx2(j)==j)
                my1(iii_1)=mx1(i)*mx2(j);
                else
                 my1(iii_1)=0;
                end
        end
end
```

Определянето на стойността на изхода се прави в съответствие математическия израз за конкретния невронно-размит модел.

Алгоритъмът за обучение на моделите се изпълнява от третата част на програмния код. Той се основава на минимизирането на текущата стойност на моделната грешка и включва актуализирането на две групи параметри - линейни, които са във функцията на Сугено и нелинейни, определящи центъра и ширината на Гаусовите функции. Разработени са няколко алгоритъма за обучение като основния е двустъпков градиентен алгоритъм, който представлява модификация на метода с обратно

разпространение на грешката. При него и двете групи параметри се актуализират посредством градиентен метод от първи ред. Този метод е представен в Приложението и с цел да се избегне излишно повторение на програмен код не е описан тук.

Останалите разработени алгоритми за обучение по същество са хибридни. При тях линейните параметри се актуализират съответно по РМНК, метода на Нютон и метода на Левенберг-Маркгуард, докато за нелинейните софтуерът е същия като този при двустъпковия градиентен алгоритъм. Това е причината, поради която в настоящата част от дисертационния труд е представен само кодът за изменение на линейните параметри.

- програмен код за обучение на параметрите във функцията на Сугено по РМНК:

```
%Training premise parameters
q=Pinit*fi;
K=q*inv(II+fi'*Pinit*fi);
tita=tita_1+K*err;
P=Pinit-K*fi'*Pinit;
```

- програмен код за обучение на параметрите във функцията на Сугено по метода на Нютон:

```
%Training premise parameters
d2=lr*err;
db0=d2*inv(xx'*xx)*xx';
b0=b0+db0;
db1=x1n*db0; b1=b1+db1;
db2=x2n*db0; b2=b2+db2;
db3=x3n*db0; b3=b3+db3;
db4=x4n*db0; b4=b4+db4;
```

- програмен код за обучение на параметрите във функцията на Сугено по метода на Левенберг-Маркгуард:

```
%Training premise parameters
d2=lr*err;
db0=d2*inv(xx'*xx+λ*I)*xx';
b0=b0+db0;
db1=x1n*db0; b1=b1+db1;
```

```
db2=x2n*db0; b2=b2+db2;
db3=x3n*db0; b3=b3+db3;
db4=x4n*db0; b4=b4+db4;
```

Последната част от софтуерната програма реализира визуализацията на получените резултати. За целта са използвани графичните функции на Matlab.

5.2. Структура и действие на програмния код на основен алгоритъм на регулатор с невронно-размито предсказващо управление

Предсказващият регулатор представлява съвкупност от модел и оптимизатор и този факт е използван като ръководен при разработването на софтуера за невронно-размито предсказващо управление. Структурата и работата на програмния код на невронно-размит модел вече бяха описани в предходната точка. Ето защо тук ще бъде представен само кода на оптимизатора, който получава като вход стойностите от предсказващия модел. Оптимизаторът е реализиран в няколко варианта: с градиентен алгоритъм от I ред, с алгоритъм на Нютон и с алгоритъм на Левенберг-Маркгуард. Последните два принадлежат към групата градиентни алгоритми от II ред. Характерно за тях е необходимостта от изчисляването на обратна матрица на Хесе на всеки такт на дискретизация, което внася в алгоритъма допълнително изчислително натоварване. За да се преодолее този недостатък тези два алгоритъма са модифицирани чрез въвеждане на LU декомпозиция.

- Програмен код на оптимизатор с използване на градиентен алгоритъм от I ред;

```
PredErr(ky)=Ref-PredOut; %predicted error
% computations of first derivative dy/du
if  ky==ku dydu_first(ky,ku)= b3*xx; end
if (ky-ku)==1 dydu_first(ky,ku)=(b1*dydu_first(ky,ku)+b4)*xx; end
if (ky-ku)>=2 dydu_first(ky,ku)=(b1*dydu_first(ky,ku)+b2*dydu_first(ky-1,ku))*xx; end
```

```
% computations of control actions
for ku=Nu:-1:1
  if ku==Nu deltau(ku)=(Ro*PredErr(N1:N2)*dydu_first(N1:N2,ku));end
  if ku<Nu  deltau(ku)=deltau(ku+1)+(Ro*PredErr(N1:N2)*dydu_first(N1:N2,ku));end
end
for ku=1:Nu
  if ku==1
    PredCon(ku)=deltau(ku)+un_1;
    if PredCon(ku)<-4 PredCon(ku)=-4; end
    if PredCon(ku)>4 PredCon(ku)=4; end
    un=PredCon(ku);
  end
  if ku>1  PredCon(ku)=deltau(ku)+PredCon(ku-1);
    if PredCon(ku)<-4 PredCon(ku)=-4; end
    if PredCon(ku)>4 PredCon(ku)=4; end
  end
```

- Програмен код на оптимизатор с използване на алгоритъм Нютон:

```
PredErr(ky)=Ref-PredOut; %predicted error
% computations of first derivative dy/du
if  ky==ku dydu_first(ky,ku)= b3*xx; end
if (ky-ku)==1 dydu_first(ky,ku)=(b1*dydu_first(ky,ku)+b4)*xx; end
if (ky-ku)>=2
dydu_first(ky,ku)=(b1*dydu_first(ky,ku)+b2*dydu_first(ky-1,ku))*xx;
end
% computations of second derivative dy/du
dydu_second = dydu_first*dydu_first';
Hes=inv(dydu_second);
rrr=Hes*dydu_first;
% computations of control actions
for ku=Nu:-1:1
  if ku==Nu deltau(ku)=(Ro*PredErr(N1:N2)*rrr(N1:N2,ku));end
  if ku<Nu  deltau(ku)=deltau(ku+1)+(Ro*PredErr(N1:N2)*rrr(N1:N2,ku));end
end
for ku=1:Nu
  if ku==1
    PredCon(ku)=deltau(ku)+un_1;
    if PredCon(ku)<-4 PredCon(ku)=-4; end
```

```
    if PredCon(ku)>4 PredCon(ku)=4; end
    un=PredCon(ku);
  end
  if ku>1  PredCon(ku)=deltau(ku)+PredCon(ku-1);
    if PredCon(ku)<-4 PredCon(ku)=-4; end
    if PredCon(ku)>4 PredCon(ku)=4; end
  end
end
```

- Програмен код на оптимизатор с използване на алгоритъм на Левенберг-Маркгуард:

```
PredErr(ky)=Ref-PredOut; %predicted error
% computations of first derivative dy/du
if  ky==ku dydu_first(ky,ku)= b3*xx; end
if (ky-ku)==1 dydu_first(ky,ku)=(b1*dydu_first(ky,ku)+b4+d*dydu_first(ky,ku))*xx; end
if (ky-ku)>=2
dydu_first(ky,ku)=(b1*dydu_first(ky,ku)+b2*dydu_first(ky-1,ku)+d*dydu_first(ky,ku))*xx;
end
% computations of second derivative dy/du
dydu_second = dydu_first*dydu_first';
LM=dydu_second+lambda*ed_matrix;
x=inv(LM);
rrr=x*dydu_first;
% computations of control actions
for ku=Nu:-1:1
  if ku==Nu deltau(ku)=(Ro*PredErr(N1:N2)*rrr(N1:N2,ku));end
  if ku<Nu  deltau(ku)=deltau(ku+1)+(Ro*PredErr(N1:N2)*rrr(N1:N2,ku));end
end
for ku=1:Nu
  if ku==1
    PredCon(ku)=deltau(ku)+un_1;
    if PredCon(ku)<-4 PredCon(ku)=-4; end
    if PredCon(ku)>4 PredCon(ku)=4; end
    un=PredCon(ku);
  end
  if ku>1  PredCon(ku)=deltau(ku)+PredCon(ku-1);
    if PredCon(ku)<-4 PredCon(ku)=-4; end
```

```
    if PredCon(ku)>4 PredCon(ku)=4; end
  end
end
```

- Програмен код на оптимизатор с използване на алгоритъм Нютон с LU декомпозиция:

```
PredErr(ky)=Ref-PredOut; %predicted error
% computations of first derivative dy/du
if  ky==ku dydu_first(ky,ku)= b3*xx; end
if (ky-ku)==1 dydu_first(ky,ku)=(b1*dydu_first(ky,ku)+b4+d*dydu_first(ky,ku))*xx; end
if (ky-ku)>=2
dydu_first(ky,ku)=(b1*dydu_first(ky,ku)+b2*dydu_first(ky-1,ku)+d*dydu_first(ky,ku))*xx;
end
% computations of second derivative dy/du
dydu_second = dydu_first*dydu_first';
if dydu_second(1,1)==0
   dydu_second(1,1)=dydu_second(1,1)+0.000001;
end
[L,U] = lu(dydu_second);
x=U\(L\dydu_first);
% computations of control actions
for ku=Nu:-1:1
  if ku==Nu deltau(ku)=(Ro*PredErr(N1:N2)*x(N1:N2,ku));end
  if ku<Nu  deltau(ku)=deltau(ku+1)+(Ro*PredErr(N1:N2)*x(N1:N2,ku));end
end
for ku=1:Nu
  if ku==1
    PredCon(ku)=deltau(ku)+un_1;
    if PredCon(ku)<-4 PredCon(ku)=-4; end
    if PredCon(ku)>4 PredCon(ku)=4; end
    un=PredCon(ku);
  end
  if ku>1  PredCon(ku)=deltau(ku)+PredCon(ku-1);
    if PredCon(ku)<-4 PredCon(ku)=-4; end
    if PredCon(ku)>4 PredCon(ku)=4; end
  end
end
```

- Програмен код на оптимизатор с използване на алгоритъм на Левенберг-Маркгуард с LU декомпозиция:

```
PredErr(ky)=Ref-PredOut; %predicted error
% computations of first derivative dy/du
if  ky==ku dydu_first(ky,ku)= b3*xx; end
if (ky-ku)==1 dydu_first(ky,ku)=(b1*dydu_first(ky,ku)+b4+d*dydu_first(ky,ku))*xx; end
if (ky-ku)>=2
dydu_first(ky,ku)=(b1*dydu_first(ky,ku)+b2*dydu_first(ky-1,ku)+d*dydu_first(ky,ku))*xx;
end
% computations of second derivative dy/du
dydu_second = dydu_first*dydu_first';
LM=dydu_second+lambda*ed_matrix;
if LM(1,1)==0
   LM(1,1)=LM(1,1)+0.000001;
end
[L,U] = lu(LM);
x=U\(L\dydu_first);
% computations of control actions
for ku=Nu:-1:1
  if ku==Nu deltau(ku)=(Ro*PredErr(N1:N2)*x(N1:N2,ku));end
  if ku<Nu  deltau(ku)=deltau(ku+1)+(Ro*PredErr(N1:N2)*x(N1:N2,ku));end
end
for ku=1:Nu
  if ku==1
     PredCon(ku)=deltau(ku)+un_1;
     if PredCon(ku)<-4 PredCon(ku)=-4; end
     if PredCon(ku)>4 PredCon(ku)=4; end
     un=PredCon(ku);
  end
  if ku>1  PredCon(ku)=deltau(ku)+PredCon(ku-1);
     if PredCon(ku)<-4 PredCon(ku)=-4; end
     if PredCon(ku)>4 PredCon(ku)=4; end
  end
end
```

5.3. Структура и действие на програмния код на алгоритъм за супервайзорна адаптивна донастройка на регулатор с невронно-размито предсказващо управление

Програмният код на алгоритъм за супервайзорна донастройка на регулатор с невронно-размито предсказващо управление до голяма степен се припокрива с този на невронно-размит предсказващ регулатор. В него се включва конкретен невронно-размит модел и оптимизационен алгоритъм, чиято софтуерна реализация вече беше описана. Съществено различие се явява добавянето на супервайзорния елемент. В ролята на супервайзор би могъл да се използва всеки един от предложените невронно-размити модели. Особеното в случая е, че като входове се използват стойностите на системната грешка, а изход е стойността на тегловния коефициент ρ, който се подава към оптимизатора.

Глава V

Заключения и изводи

Настоящият дисертационен труд е посветен на разработването на невронно-размити модели за целите на предсказващото управление, т.е такива, които гарантират висока точност на моделиране и намалена изчислителна тежест. Направеният в Глава I обзор разкрива изключително голямо разнообразие на хибридни невронно-размити структури, използвани в системи с НМПУ в ролята на предсказващи модели. Многобройните изследвания и разработки в това направление говорят както за засилен интерес и стремеж към усъвършенстване на предсказващите регулатори с невронно-размит модел, така и за все още нерешени проблеми в тази област. Един съществен недостатък, който трябва да бъде преодолян, е намаляване на изчислителната тежест на предсказващите регулатори. Това би могло да се постигне от една страна чрез използване на невронно-размити структури, работещи с редуциран брой размити правила, а от друга с оптимизационни алгоритми, осигуряващи бързодействие на предсказващите регулатори. Друг важен аспект е точността на предсказване, тъй като от нея зависи качеството на управление. Едно подходящо решение в това отношение е включването на Тип 2 или Интуиционстка размита логика в хибридните навронно-размити модели. Прави впечатление също така, че в областта на настройката на предсказващите регулатори не е работено задълбочено. Не е проучена и възможността за използването на интелигентни структури за тази цел.

Отправна точка в процеса на търсене на решения на дискутираните вече проблеми се явява алгоритъмът на нелинейния невронно-размит обобщен предсказващ регулатор, описан в Глава II. При него в ролята на предсказващ модел е използван невронно-размит авторегресионен модел (NNFARX), като е прието той да има четири входа (две минали стойности на изхода, миналата и текуща стойност на управлението) и всеки от тях да се размива с по три Гаусови множества. Оттук произтича и един от основните недостатъци на този регулатор, а именно големият брой размити правила, с които работи предсказващият модел. Вече беше споменато, че

техният брой се определя съгласно израза $N=m^p$ и следователно в конкретния случай броят на правилата е 81. Това от своя страна означава, че на всеки такт по време на обучението на този модел трябва да бъдат определени стойностите на 1053 параметъра! Това го прави практически неприложим за работа в реално време, особено за процеси с бързо изменяща се динамика. Поради това е необходимо да се търсят нови невронно-размити модели, работещи с редуциран брой размити правила и същевременно осигуряващи висока точност на предсказване. Освен по отношение на модела, работата на този регулатор би могла да се подобри и с включването на градиентни алгоритми от II ред в оптимизатора. Върху качеството на управление на нелинейния невронно-размит обобщен предсказващ регулатор, описан в Глава II, би могло да се повлияе и чрез подходящата му настройка.

В търсене на конкретни решения на поставените проблеми в Глава III от дисертационния труд са описани проектираните алгоритми за регулатори с невронно-размито предсказващо управление. Предложени са три невронно-размити модела, работещи с редуциран брой размити правила и подходящи за включване в системи с НМПУ. Това са *Разпределен невронно-размит модел, Невронно-размит модел с частично размиване на входните сигнали и Модифициран нео-размит модел*. В сравнение с невронно-размития модел от т.2.1, и трите предложени структури имат значително по-малък брой параметри за актуализиране по време на обучението.

За да могат да се справят успешно в условията на неопределености DANFA, SFNN и Модифицираният нео-размит модел са реализирани във вариант с Тип 2 размита логика и с Интуиционистка размита логика. Това повишава тяхната точност на предсказване, но е за сметка на завишен брой параметри, които трябва да се актуализират по време на обучението. Дори и във вариант с Тип 2 размита логика и с Интуиционистка размита логика предложените DANFA, SFNN и Модифицираният нео-размит модел имат значително по-малко параметри в сравнение с модела от т.2.1.

Едно от основните предимства, на които предсказващите регулатори дължат широкото си приложение, е възможността им да се справят с многомерни обекти с неизвестно или променливо времезакъснение. За да бъде възможно това е нужно да се разработят многомерни невронно-размити модели, за които също е в сила изискването да осигуряват висока

точност на предсказване и да са с малък брой параметри за настройка. Това е постигнато с реализирания в многомерен вариант модифициран нео-размит модел.

Реализирани са и различни алгоритми за обучение на предложените невронно-размити структури. Наред с превърналия се в класически метод с обратно разпространение на грешката, са разработени също и хибридни алгоритми, комбиниращи рекурентен метод на най-малките квадрати, метод на Нютон и метод на Левенберг-Маркгуард за обучение на параметрите във функцията на Сугено с градиентен метод от I ред за обучение на параметрите на размитите множества.

При системите с НМПУ методът за оптимизация е от особено значение, тъй като оказва съществено влияние върху изчислителната ефективност на оптимизатора. С оглед на това в настоящата дисертация са предпочетени алгоритми от втори ред, а именно тези на Нютон и на Левенберг-Маркгуард, тъй като при тях е намален броят на итерациите, свързани със сходимостта на алгоритъма, а това от своя страна води до по-голямо бързодействие на предсказващия регулатор.

Изследвана е възможността за използване на интелигентни невронно-размити структури за настройката на предсказващите регулатори. За целта е реализиран алгоритъм за супервайзорна адаптивна донастройка на тегловния коефициент в целевия критерий (1.4.1), като се запазват постоянни стойностите на хоризонтите на управление и на предсказване. Една интересна идея за бъдещи изследвания е използване на многомерния модифициран нео-размит модел за цялостна настройка на предсказващите регулатори, т.е. за определянето както на тегловния коефициент, така и на хоризонтите на управление N_u и на предсказване N_p.

Резултатите от симулационни изследвания и реални експерименти на проектираните алгоритми за регулатори с невронно-размито предсказващо управление са представени в Глава IV от дисертационния труд. Първоначално е изследвана способността на DANFA, SFNN и модифицирания нео-размит модел да предсказват хаотични времеви серии. За обучението им е използван двустъпков градиентен алгоритъм с постоянна скорост на обучение $\eta=0.05$. Направено е сравнение на тези модели с невронно-размития модел, описан в т.2.1, при което за най-добър модел в съответствие със заложените критерии е определен модифицирания

нео-размит модел. Въпреки, че работи с по-голям брой размити правила в сравнение с SFNN Type III, модифицираният нео-размит модел има по-малък брой параметри за актуализиране в процеса на обучение (дори и в случая с обучаване параметрите на размитите множества), по-точен е и по-бърз. Ето защо именно модифицираният нео-размит модела е реализиран във вариант с Тип 2 размита логика и с Интуиционистка размита логика в присъствието на шум, за да се види как се справя в условия на неопределености. От получените резултати става ясно, че Интуиционисткият нео-размит модел работи с по-малка грешка в сравнение с Тип 2 нео-размития модел. Времето за осъществяване на необходимите изчисления за една стъпка от обучението при Интуиционисткия нео-размит модел е съответно $1.48e^{-4}$ за случая без шум и $1.50e^{-4}$ с шум. При Тип 2 нео-размития модела това време е по-голямо – съответно $1.93e^{-4}$ и $1.96e^{-4}$ за случаите без и с шум. Следователно Интуиционисткият нео-размит модел е по-точен и по-бърз и това го прави подходящ за включване в предсказващ регулатор. Модифицираният нео-размит модел е разработен и в MIMO вариант, тъй като едно от предимствата на предказващите регулатори е способността им да се справят с многомерни обекти.

Всеки един от предложените невронно-размити модели е използван за реализиране на обобщен предсказващ регулатор с итеративен оптимизационен алгоритъм от първи и от втори ред. От получените резултати може да се заключи, че структурата на използвания невронно-размит модел не оказва влияние върху качеството на управление. С използването на предложените невронно-размити модели се съкращава приблизително два пъти времето t, необходимо за реализиране на всички изчисления в оптимизатора за един такт. Това време зависи също и от хоризонтите на предсказване и на управление и нараства с увеличаването им.

Бързодействието на невронно-размит обобщен предсказващ регулатор значително се увеличава когато се използва итеративен оптимизационен алгоритъм от втори ред. Основната трудност при тези алгоритми - изчисляването на обратната матрица на Хесе на всеки такт – е преодоляна с въвеждането на LU декомпозиция. От получените резултати се вижда, че с въвеждането на LU декомпозиция колебанията в преходния процес се „изглаждат” и характерът му се свежда до апериодичен. Освен

това, от данните в табл.4.2.2.2 става ясно, че при модифицираните с помощта на Гаусовата елиминация алгоритми на Нютон и на Левенберг-Маркгуард времето t за реализиране на изчисленията в оптимизатора за един такт намалява.

Що се отнася до настройката на предсказващите регулатори един съществен извод, които може да се направи е, че тегловният коефициент ρ оказва съществено влияние върху качеството на управление, но не и върху времето за работа на оптимизатора. Действието му може да бъде сравнено с това на пропорционалната съставка на ПИД регулатора.

В заключение може да се каже, че предложените DANFA, SFNN, Модифициран нео-размит модел и техните варианти с Тип 2 размита логика и с Интуиционистка размита логика удовлетворяват напълно поставените в дисертационния труд цел и задачи. От представените резултати е видно, че те осигуряват предсказване с висока точност като същевременно са с намалена изчислителна тежест. Включването им в обобщен предсказващ регулатор води до намаляване на времето за изчисления приблизително два пъти. В същото време обаче приложението на DANFA, SFNN и Модифицирания нео-размит модел не трябва да се ограничава само до предсказващите регулатори. Те биха могли успешно да се използват при разпознаване на образи и обработка на изображения, в ролята на класификатори, за обработка на данни и извличане на знания от тях, в системите за сигурност и др.

Приноси от изследването

Приносите на настоящия дисертационен труд са ориентирани към обогатяване на съществуваща научна област с нови знания, модели и алгоритми и като цяло имат научно-приложен и приложен характер.

1. Предложен и софтуерно реализиран е DANFA модел, който работи с редуциран брой размити правила и има намален брой параметри.
2. Предложен е втори изчислително ефективен невронно-размит модел, а именно SFNN модел, който е програмно реализиран в три варианта.
3. Разработен е програмен код за модифициран нео-размит модел, работещ с минимален брой размити правила и осигуряващ висока точност на моделиране.
4. Софтуерно е реализиран модифициран нео-размит модел във вариант с Тип 2 размита логика и с Интуиционистка размита логика, с оглед постигане на по-добро управление в условия на неопределеност.
5. Реализирани са програмно итеративни градиентни оптимизационни алгоритми от втори ред, осигуряващи бързодействие на нелинеен невронно-размит обобщен предсказващ регулатор.
6. Оптимизационните алгоритми от II ред са реализирани софтуерно във вариант с LU декомпозиция – изчислително ефективен метод, който позволява да се избегне намирането на обратната матрица на Хесе на всеки такт.
7. Разработен е програмно алгоритъм, за супервайзорна адаптивна донастройка на тегловния коефициент в целевия критерий за оптимизация.

Декларация за оригиналност на резултатите

Декларирам, че настоящата дисертация съдържа оригинални резултати, получени при проведени от мен научни изследвания (с подкрепата и съдействието на научните ми ръководители). Резултатите, които са получени, описани и/или публикувани от други учени, са надлежно и подробно цитирани в библиографията.

Настоящата дисертация не е представяна в процедури за придобиване на научна степен вдруго висше училище, университет или научен институт.

Подпис:

Маргарита Терзийска

Приложение

В настоящото Приложение са дадени листингите на основните програми, използвани в дисертационния труд:

- danfa.m - Разпределен невронно-размит модел.
- sfnn.m - Невронно-размит модел с частично размиване на входните сигнали. С цел да не се увеличи прекомерно обема на дисертационния труд е представен само листинга на един от разработените варианти, а именно SFNN Type I. От него с минимални корекции биха могли да се получат SFNN Type II и SFNN Type III.
- neo_fuzzy_model.m - Модифициран нео-размит модел.
- mimo_NFN.m - Многомерен модифициран нео-размит модел.
- SFNN_III_Newton_GPC.m - Алгоритъм на нелинеен обобщен предсказващ регулатор, базиран на невронно-размит модел с частично размиване на входните сигнали от тип III (с кръстосани връзки) и оптимизационен метод на Нютон.
- LM_LU_GPC.m - Алгоритъм на нелинеен обобщен предсказващ регулатор, базиран на невронно-размит модел и Левенберг-Маркгуард оптимизационен метод, реализиран във вариант с LU декомпозиция.

Тези програми използват следните подпрограми:

- fuzzyg.m - Реализира операцията размиване с Гаусови функции като използва размита логика Тип 1.
- gauss_t2.m - Тази функция се явява алтернатива на fuzzyg.m. Ако вместо последната в кода се използва gauss_t2.m ще се реализира операцията размиване с Гаусови функции като се използва размита логика Тип 2.
- ifs.m - Тази функция също се явява алтернатива на fuzzyg.m. Ако вместо последната в кода се използва ifs.m ще се реализира операцията размиване с Гаусови функции като се използва Интуиционистка размита логика.

- MG.m, lorentz.m, rossler.m - Тези функции генерират съответно Макгий-Глас, Лоренц и Рослер хаотични времеви серии. Те не са разработени в рамките на настоящия дисертационен труд, поради което кодът им не е даден тук.

danfa.m

```
%Initialization
%Gaussian membership function parameters
fuz=[1 0.2123 0
    1 0.2123 0.5
    1 0.2123 1];
 as=size(fuz); a=as(1,1); Nrules= a^2;
 lr=0.04;
 %Variable parameters
 serr=0; serr_1=0; Nerr=0;
    w2=zeros(1, Nrules);
    b1=0.1*rand(1,Nrules);
    b2=0.1*rand(1,Nrules);
    b3=0.1*rand(1,Nrules);
    b4=0.1*rand(1,Nrules);
    b01=0.1*rand(1,Nrules);
    b02=0.1*rand(1,Nrules);
    d2=0; da2=w2; my=w2; fout=w2;
    err=0; err_1=0; foutn=w2;
    fuzx1=fuz; fuzx2=fuz; fuzx3=fuz; fuzx4=fuz;
    ymn_2=0; ymn_1=0; ymn=0;
    %Start
    for t=1:500
       tstart = tic;
       %DANFA MODEL
       un=MG(t);
       x1n=y2n_1;
       x2n=y2n_2;
       x3n=un;
```

```
x4n=un_1;
%Fuzzyfication FNN1
[mx1,cx1]=Fuzzyg(fuzx1,x1n,a);
[mx2,cx2]=Fuzzyg(fuzx2,x2n,a);
%Fuzzy Implication
iii_1=0;
for i=1:a
   for j=1:a
       iii_1=iii_1+1;
        if (cx1(i)==i) & (cx2(j)==j)
              my1(iii_1)=mx1(i)*mx2(j);
           else
              my1(iii_1)=0;
           end
         end
      end
%Defuzzyfication
xx1=[my1]'; summy1=0;
for i=1:(Nrules)
   foutn1(i)=b1(i)*x1n+b2(i)*x2n+b01(i);
   summy1=summy1+xx1(i);
end
xx1=xx1/(summy1+0.001);
ymn1=foutn1*xx1;
%Fuzzyfication FNN2
[mx3,cx3]=Fuzzyg(fuzx3,x3n,a);
[mx4,cx4]=Fuzzyg(fuzx4,x4n,a);
%Fuzzy Implication
iii_2=0;
 for k=1:a
   for l=1:a
     iii_2=iii_2+1;
       if (cx3(k)==k) & (cx4(l)==l)
          my2(iii_2)=mx3(k)*mx4(l);
            else
```

```
                my2(iii_2)=0;
            end
        end
    end
%Defuzzyfication
xx2=[my2]'; summy2=0;
for i=1:(Nrules)
   foutn2(i)=b3(i)*x3n+b4(i)*x4n+b02(i);
   summy2=summy2+xx2(i);
end
xx2=xx2/(summy2+0.001);
ymn2=foutn2*xx2;
ymn=ymn1+ymn2;
%Model error calculation
 Nerr=Nerr+1; %number of calculations
 err=un-ymn;
 serr=err^2;
 SSE=serr+serr_1;  %Sum of squared errors
 RSE=sqrt(SSE);
 MSE=SSE/Nerr; %Mean squared errors
 RMSE=sqrt(MSE); %Root mean squared errors
 %Training process
 %premise parameters
   d2=lr*err;
   db01=d2*xx1'; b01=b01+db01;
   db1=x1n*db01; b1=b1+db1;
   db2=x2n*db01; b2=b2+db2;

   db02=d2*xx2'; b02=b02+db02;
   db3=x3n*db02; b3=b3+db3;
   db4=x4n*db02; b4=b4+db4;
 %consequent parameters
   iii=0;
   for i=1:a
      for j=1:a
```

```
            iii=iii+1;
                if (cx1(i)==i) & (cx2(j)==j)
                    da2_1(iii)=d2*(foutn1(iii)-ymn1)*xx1(iii)';

                    fuzx1(i,3)=fuzx1(i,3)+da2_1(iii)*((x1n-fuzx1(i,3))/(fuzx1(i,2))^2);
                    fuzx1(i,2)=fuzx1(i,2)+da2_1(iii)*((x1n-fuzx1(i,3))^2/(fuzx1(i,2))^3);
                    if fuzx1(i,3)>fuz(a,3) fuzx1(i,3)=fuz(a,3); end
                    if fuzx1(i,3)<fuz(a,3) fuzx1(i,3)=fuz(1,3); end
                    if fuzx1(i,2)<0 fuzx1(i,2)=-fuzx1(i,2); end
                    if fuzx1(i,2)>2*fuz(i,3) fuzx1(i,2)=2*fuz(i,2); end

                    fuzx2(j,3)=fuzx2(j,3)+da2_1(iii)*((x2n-fuzx2(j,3))/(fuzx2(j,2))^2);
                    fuzx2(j,2)=fuzx2(j,2)+da2_1(iii)*((x2n-fuzx2(j,3))^2/(fuzx2(j,2))^3);
                    if fuzx2(j,3)>fuz(a,3) fuzx2(j,3)=fuz(a,3); end
                    if fuzx2(j,3)<fuz(a,3) fuzx2(j,3)=fuz(1,3); end
                    if fuzx2(j,2)<0 fuzx2(j,2)=-fuzx2(j,2); end
                    if fuzx2(j,2)>2*fuz(j,3) fuzx2(j,2)=2*fuz(j,2); end
            end
          end
        end

      iii=0;
          for k=1:a
            for l=1:a
                iii=iii+1;
                if (cx3(k)==k) & (cx4(l)==l)
                    da2_2(iii)=d2*(foutn2(iii)-ymn2/6)*xx2(iii)';
                    fuzx3(k,3)=fuzx3(k,3)+da2_2(iii)*((x3n-fuzx3(k,3))/(fuzx3(k,2))^2);
                    fuzx3(k,2)=fuzx3(k,2)+da2_2(iii)*((x3n-
fuzx3(k,3))^2/(fuzx3(k,2))^3);
                    if fuzx3(k,3)>fuz(a,3) fuzx3(k,3)=fuz(a,3); end
                    if fuzx3(k,3)<fuz(a,3) fuzx3(k,3)=fuz(1,3); end
                    if fuzx3(k,2)<0 fuzx3(j,2)=-fuzx3(k,2); end
                    if fuzx3(k,2)>2*fuz(k,3) fuzx3(k,2)=2*fuz(k,2); end
```

```
                        fuzx4(l,3)=fuzx4(l,3)+da2_2(iii)*((x4n-fuzx4(l,3))/(fuzx4(l,2))^2);
                        fuzx4(l,2)=fuzx4(l,2)+da2_2(iii)*((x4n-fuzx4(l,3))^2/(fuzx4(l,2))^3);
                        if fuzx4(l,3)>fuz(a,3) fuzx4(l,3)=fuz(a,3); end
                        if fuzx4(l,3)<fuz(a,3) fuzx4(l,3)=fuz(1,3); end
                        if fuzx4(l,2)<0 fuzx4(l,2)=-fuzx4(l,2); end
                        if fuzx4(l,2)>2*fuz(l,3) fuzx4(l,2)=2*fuz(l,2); end
                    end
                end
            end
        un_2=un_1;un_1=un; telapsed = toc(tstart);
        RMSEplot(t)=RMSE; MSEplot(t)=MSE;
        pred_err(t)=err; serr_1=SSE;
%Plot
yplot(t)=yn;
ymplot(t)=ymn;
figure(1); grid on;
plot(yplot,'b');hold on;
plot(ymplot,'r');hold on;
legend('yobj','ymod');
figure(2);
plot(MSEplot,'b'); hold on;
plot(RMSEplot,'r');grid on; legend('MSE','RMSE');
 figure(3); grid on;
 plot(pred_err); grid on; legend('pred error');
end %training process
```

sfnn.m

```
%Initialization
%Gaussian membership function parameters
fuz3=[1 0.2123 0
    1 0.2123 0.5
    1 0.2123 1];
  fuz=fuz3;
  as=size(fuz); a=as(1,1); Nrules= a^2;
```

```
lr=0.05;
%Variable parameters
serr=0; serr_1=0; Nerr=0; %RSE_1=0; %SSE=0;  MSE=0; RSE=0;
   w2=zeros(1, Nrules);
   b1=0.001*rand(1,Nrules);
   b2=0.001*rand(1,Nrules);
   b3=0.001*rand(1,Nrules);
   b4=0.001*rand(1,Nrules);
   b01=0.001*rand(1,Nrules);
   b02=0.001*rand(1,Nrules);
   d2=0; da2=w2; my=w2; fout=w2;
   err=0; err_1=0; foutn=w2;
   fuzx1=fuz; fuzx2=fuz; fuzx3=fuz; fuzx4=fuz;
   u1=0; ek=0; ek_1=0; ek_2=0; se_1=0; se=0;  un_1=0;
   ymn_2=0; ymn_1=0; ymn=0;
   %Start
      for t=1:500
      tstart = tic;
      %Semi Fuzzy Neural Model
      un=MG(t)
      x1n=ymn_1;
      x2n=ymn_2;
      x3n=un;
      x4n=un_1;
      %Fuzzyfication
      [mx1,cx1]=fuzzyg(fuzx1,x1n,a);
      [mx2,cx2]=fuzzyg(fuzx2,x2n,a);
      %Fuzzy Implication
      iii_1=0;
      for i=1:a
         for j=1:a
            iii_1=iii_1+1;
             if (cx1(i)==i) & (cx2(j)==j)
                    my1(iii_1)=mx1(i)*mx2(j);
                else
```

```
                my1(iii_1)=0;
            end
        end
    end
%Defuzzyfication
xx1=[my1]'; summy1=0;
for i=1:(Nrules)
   foutn1(i)=b1(i)*x1n+b2(i)*x2n+b3(i)*x3n+b4(i)*x4n+b01(i);
   summy1=summy1+xx1(i);
end
xx1=xx1/(summy1+0.001);
ymn=15*foutn1*xx1;
 %Model error calculation
 Nerr=Nerr+1; %number of calculations
 err=un-ymn;
 serr=err^2;
 SSE=serr+serr_1;  %Sum of squared errors
 RSE=sqrt(SSE);
 MSE=SSE/Nerr; %Mean squared errors
 RMSE=sqrt(MSE); %Root mean squared errors
 %Training premise parameters
   d2=lr*err;
   db01=d2*xx1'; b01=b01+db01;
   db1=x1n*db01; b1=b1+db1;
   db2=x2n*db01; b2=b2+db2;
   db3=x3n*db01; b3=b3+db3;
   db4=x4n*db01; b4=b4+db4;
 %consequent parameters
   iii=0;
   for i=1:a
      for j=1:a
         iii=iii+1;
            if (cx1(i)==i) & (cx2(j)==j)
                da2_1(iii)=d2*(foutn1(iii)-ymn)*xx1(iii)';
```

```
                    fuzx1(i,3)=fuzx1(i,3)+da2_1(iii)*((x1n-fuzx1(i,3))/(fuzx1(i,2))^2);
                    fuzx1(i,2)=fuzx1(i,2)+da2_1(iii)*((x1n-fuzx1(i,3))^2/(fuzx1(i,2))^3);
                    if fuzx1(i,3)>fuz(a,3) fuzx1(i,3)=fuz(a,3); end
                    if fuzx1(i,3)<fuz(a,3) fuzx1(i,3)=fuz(1,3); end
                    if fuzx1(i,2)<0 fuzx1(i,2)=-fuzx1(i,2); end
                    if fuzx1(i,2)>2*fuz(i,3) fuzx1(i,2)=2*fuz(i,2); end

                    fuzx2(j,3)=fuzx2(j,3)+da2_1(iii)*((x2n-fuzx2(j,3))/(fuzx2(j,2))^2);
                    fuzx2(j,2)=fuzx2(j,2)+da2_1(iii)*((x2n-fuzx2(j,3))^2/(fuzx2(j,2))^3);
                    if fuzx2(j,3)>fuz(a,3) fuzx2(j,3)=fuz(a,3); end
                    if fuzx2(j,3)<fuz(a,3) fuzx2(j,3)=fuz(1,3); end
                    if fuzx2(j,2)<0 fuzx2(j,2)=-fuzx2(j,2); end
                    if fuzx2(j,2)>2*fuz(j,3) fuzx2(j,2)=2*fuz(j,2); end
                end
            end
        end
%update previous values
 ymn_2=ymn_1; ymn_1=ymn;
un_2=un_1;un_1=un; telapsed = toc(tstart);
        RMSEplot(t)=RMSE; MSEplot(t)=MSE;
        pred_err(t)=err; serr_1=SSE; lrplot(t)=lr;
yplot(t)=un;
ymplot(t)=ymn;
figure(1); grid on;
plot(yplot,'b');hold on;
plot(ymplot,'r');hold on;
legend('yobj','ymod');
figure(2);
plot(MSEplot,'b'); hold on;
plot(RMSEplot,'r');grid on; legend('MSE','RMSE');
figure(3); grid on;
plot(pred_err); grid on; legend('pred error');
end %training process
```

neo_fuzzy_model.m

```
%Initialization
%Gaussian membership function parameters
fuz=[1 0.2123 0
     1 0.2123 0.5
     1 0.2123 1];
  as=size(fuz); a=as(1,1);
  lr=0.05;
  %Variable parameters
  serr=0; serr_1=0; Nerr=0;
     b1=0.001*rand(1,a);
     b2=0.001*rand(1,a);
     b3=0.001*rand(1,a);
     b4=0.001*rand(1,a);
     d2=0;
     err=0; err_1=0;
     fuzx1=fuz; fuzx2=fuz; fuzx3=fuz; fuzx4=fuz;
        u1=0; ek=0; ek_1=0; ek_2=0; se_1=0; se=0;
        y1n=0.144; y2n=0.866; y3n=0; yn=0;
        y1n_1=0.144; y2n_1=0.866; y3n_1=0; un_1=0;
        y1n_2=0.144; y2n_2=0.866; y3n_2=0; un_2=0;
        ymn_2=0; ymn_1=0; ymn=0;
       Ref=0.886; x=0;
     %Start
        for t=1:500
        tstart = tic;
        un=ans(t)/10;
        x1n=y2n_1/10;
        x2n=y2n_2;
        x3n=un;
        x4n=un_1;
        %Fuzzyfication
        [mx1,cx1]=fuzzyg(fuzx1,x1n,a);
        y_mod1=mx1'*b1';
        [mx2,cx2]=fuzzyg(fuzx2,x2n,a);
```

```
        y_mod2=mx2'*b2';
        [mx3,cx3]=fuzzyg(fuzx3,x3n,a);
        y_mod3=mx3'*b3';
        [mx4,cx4]=fuzzyg(fuzx4,x4n,a);
        y_mod4=mx4'*b4';
        ymn=(y_mod1+y_mod2+y_mod3+y_mod4)*10;
        %Model error calculation
         Nerr=Nerr+1; %number of calculations
         err=un-ymn;
         serr=err^2;
         SSE=serr+serr_1;  %Sum of squared errors
         RSE=sqrt(SSE);
         MSE=SSE/Nerr; %Mean squared errors
         RMSE=sqrt(MSE); %Root mean squared errors
         %Training premise parameters
           for i=1:a
              b1(i)=b1(i)+lr*err*mx1(i);
              b2(i)=b2(i)+lr*err*mx2(i);
              b3(i)=b3(i)+lr*err*mx3(i);
              b4(i)=b4(i)+lr*err*mx4(i);
           end

un_2=un_1;un_1=un;telapsed = toc(tstart);
        RMSEplot(t)=RMSE; MSEplot(t)=MSE;
        pred_err(t)=err; serr_1=SSE; lrplot(t)=lr;
yplot(t)=un;
ymplot(t)=ymn;
figure(1); grid on;
plot(yplot,'b');hold on;
plot(ymplot,'r');hold on;
legend('yobj','ymod');
figure(2);
plot(MSEplot,'b'); hold on;
plot(RMSEplot,'r');grid on; legend('MSE','RMSE');
figure(3); grid on;
```

```
plot(pred_err); grid on; legend('pred error');
end %training process
```

mimo_NFN.m

```
%Initialization
%Gaussian membership function parameters
fuz=[1 0.2123 0
     1 0.2123 0.5
     1 0.2123 1];
  as=size(fuz); a=as(1,1);
  lr=0.05;
  %Variable parameters
  serr1=0; serr1_1=0; Nerr=0;
  serr2=0; serr2_1=0;
  B=0.001*rand(12,a);
  C=0.001*rand(12,a);
  d2=0;
  err=0; err_1=0;
   fuzx1=fuz; fuzx2=fuz; fuzx3=fuz; fuzx4=fuz; fuzx5=fuz; fuzx6=fuz;
   fuzx7=fuz; fuzx8=fuz; fuzx9=fuz; fuzx10=fuz; fuzx11=fuz; fuzx12=fuz;
         u1=0; ek=0; ek_1=0; ek_2=0; se_1=0; se=0;
         y1n=0; y2n=0; y3n=0; y4n=0;
         y1n_1=0; y2n_1=0; y3n_1=0; y4n_1=0; un_1=0; u1n_1=0;
         y1n_2=0; y2n_2=0; y3n_2=0; y4n_2=0; un_2=0; u1n_2=0;
          ymn1_2=0; ymn1_1=0; ymn1=0;
          ymn2_2=0; ymn2_1=0; ymn2=0;
     %Start

     for t=1:300
         un= MG(t);
         u1n=MG(t);
         x1n=y1n_1;
         x2n=y1n_2;
         x3n=y2n_1;
```

```
x4n=y2n_2;
x5n=y3n_1;
x6n=y3n_2;
x7n=y4n_1;
x8n=y4n_2;
x9n=un;
x10n=un_1;
x11n=u1n;
x12n=u1n_1;
%Fuzzyfication
[mx1,cx1]=fuzzyg(fuzx1,x1n,a);
y1_mod1=mx1'*B(1,:)'; y2_mod1=mx1'*C(1,:)';
[mx2,cx2]=fuzzyg(fuzx2,x2n,a);
y1_mod2=mx2'*B(2,:)'; y2_mod2=mx2'*C(2,:)';
[mx3,cx3]=fuzzyg(fuzx3,x3n,a);
y1_mod3=mx3'*B(3,:)'; y2_mod3=mx3'*C(3,:)';
[mx4,cx4]=fuzzyg(fuzx4,x4n,a);
y1_mod4=mx4'*B(4,:)'; y2_mod4=mx4'*C(4,:)';
[mx5,cx5]=fuzzyg(fuzx5,x5n,a);
y1_mod5=mx5'*B(5,:)'; y2_mod5=mx5'*C(5,:)';
[mx6,cx6]=fuzzyg(fuzx6,x6n,a);
y1_mod6=mx6'*B(6,:)'; y2_mod6=mx6'*C(6,:)';
[mx7,cx7]=fuzzyg(fuzx7,x7n,a);
y1_mod7=mx7'*B(7,:)'; y2_mod7=mx7'*C(7,:)';
[mx8,cx8]=fuzzyg(fuzx8,x8n,a);
y1_mod8=mx8'*B(8,:)'; y2_mod8=mx8'*C(8,:)';
[mx9,cx9]=fuzzyg(fuzx9,x9n,a);
y1_mod9=mx9'*B(9,:)'; y2_mod9=mx9'*C(9,:)';
[mx10,cx10]=fuzzyg(fuzx10,x10n,a);
y1_mod10=mx10'*B(10,:)'; y2_mod10=mx10'*C(10,:)';
[mx11,cx11]=fuzzyg(fuzx11,x11n,a);
y1_mod11=mx11'*B(11,:)'; y2_mod11=mx11'*C(11,:)';
[mx12,cx12]=fuzzyg(fuzx12,x12n,a);
y1_mod12=mx12'*B(12,:)'; y2_mod12=mx12'*C(12,:)';
```

```
ymn1=y1_mod1+y1_mod2+y1_mod3+y1_mod4+y1_mod5+y1_mod6+y1_mod7+y1_
mod8+y1_mod9+y1_mod10+y1_mod11+y1_mod12;

ymn2=y2_mod1+y2_mod2+y2_mod3+y2_mod4+y2_mod5+y2_mod6+y2_mod7+y2_
mod8+y2_mod9+y2_mod10+y2_mod11+y2_mod12;


   %Model error calculation
    Nerr=Nerr+1;
    err1=y1n-ymn1;
    err2=y3n-ymn2;
    serr1=err1^2;
    serr2=err2^2;
    SSE1=serr1+serr1_1;  %Sum of squared errors
    SSE2=serr2+serr2_1;  %Sum of squared errors
    RSE1=sqrt(SSE1);
    RSE2=sqrt(SSE2);
    MSE1=SSE1/Nerr; %Mean squared errors
    MSE2=SSE2/Nerr; %Mean squared errors
    RMSE1=sqrt(MSE1); %Root mean squared errors
    RMSE2=sqrt(MSE2); %Root mean squared errors
    %Training process
    %premise parameters
        lr1=0.1*sum(mx1)^2;
        lr2=0.1*sum(mx2)^2;
        lr3=0.1*sum(mx3)^2;
        lr4=0.1*sum(mx4)^2;
        lr5=0.1*sum(mx5)^2;
        lr6=0.1*sum(mx6)^2;
        lr7=0.1*sum(mx7)^2;
        lr8=0.1*sum(mx8)^2;
        lr9=0.1*sum(mx9)^2;
        lr10=0.1*sum(mx10)^2;
        lr11=0.1*sum(mx11)^2;
        lr12=0.1*sum(mx12)^2;
```

```
        for i=1:a
           B(1,i)=B(1,i)+lr1*err1*mx1(i);
           B(2,i)=B(2,i)+lr2*err1*mx2(i);
           B(3,i)=B(3,i)+lr3*err1*mx3(i);
           B(4,i)=B(4,i)+lr4*err1*mx4(i);
           B(5,i)=B(5,i)+lr5*err1*mx5(i);
           B(6,i)=B(6,i)+lr6*err1*mx6(i);
           B(7,i)=B(7,i)+lr7*err1*mx7(i);
           B(8,i)=B(8,i)+lr8*err1*mx8(i);
           B(9,i)=B(9,i)+lr9*err1*mx9(i);
           B(10,i)=B(10,i)+lr10*err1*mx10(i);
           B(11,i)=B(11,i)+lr11*err1*mx11(i);
           B(12,i)=B(12,i)+lr12*err1*mx12(i);

           C(1,i)=C(1,i)+lr1*err2*mx1(i);
           C(2,i)=C(2,i)+lr2*err2*mx2(i);
           C(3,i)=C(3,i)+lr3*err2*mx3(i);
           C(4,i)=C(4,i)+lr4*err2*mx4(i);
           C(5,i)=C(5,i)+lr5*err2*mx5(i);
           C(6,i)=C(6,i)+lr6*err2*mx6(i);
           C(7,i)=C(7,i)+lr7*err2*mx7(i);
           C(8,i)=C(8,i)+lr8*err2*mx8(i);
           C(9,i)=C(9,i)+lr9*err2*mx9(i);
           C(10,i)=C(10,i)+lr10*err2*mx10(i);
           C(11,i)=C(11,i)+lr11*err2*mx11(i);
           C(12,i)=C(12,i)+lr12*err2*mx12(i);
        end
%MIMO plant
y1n=(y1n_1)^2/((y1n_1)^2+1)+0.5*y2n_1;
y2n=(y1n_1)^2/((y2n_1)^2+(y3n_1)^2+(y4n_1)^2+1)+un_1;
y3n=(y3n_1)^2/((y3n_1)^2+1)+0.3*y4n_1;
y4n=(y3n_1)^2/((y1n_1)^2+(y2n_1)^2+(y4n_1)^2+1)+0.5*u1n_1;
%update previous values
ymn1_2=ymn1_1; ymn1_1=ymn1;
ymn2_2=ymn2_1; ymn2_1=ymn2;
```

```
y1n_2=y1n_1; y1n_1=y1n;
y2n_2=y2n_1; y2n_1=y2n;
y3n_2=y3n_1; y3n_1=y3n;
y4n_2=y4n_1; y4n_1=y4n;
un_2=un_1;un_1=un;
u1n_2=u1n_1;u1n_1=u1n;
        RMSE1plot(t)=RMSE1; MSE1plot(t)=MSE1;
        RMSE2plot(t)=RMSE2; MSE2plot(t)=MSE2;
        pred_err1(t)=err1; pred_err2(t)=err2;
yplot1(t)=y1n;
ymplot(t)=ymn1;
yplot3(t)=y3n;
ym2plot(t)=ymn2;
figure(1); grid on;
plot(yplot1,'b');hold on;
plot(ymplot,'r');hold on;
plot(yplot3,'g');hold on;
plot(ym2plot,'y');hold on;
legend('y1','ymod1','y2','ymod2');
 figure(2);
 plot(MSE1plot,'b'); hold on;
 plot(RMSE1plot,'r');grid on;
 plot(MSE2plot,'g'); hold on;
 plot(RMSE2plot,'y');grid on;
 figure(3); grid on;
 plot(pred_err1,'b'); hold on; plot(pred_err2,'r'); grid on;
end %training process
```

fuzzyg.m

```
function [m,c]=fuzzyg(fuz,e,a)
m=zeros(a,1);c=m;
   if(e>=fuz(1,3))&e<=(fuz(3,3))
       for i=1:a
           if fuz(i,1)==0
```

```
            i=i+1;
          end
            m(i)=exp(-(e-fuz(i,3))^2/(2*(fuz(i,2))^2));
            if m(i)~=0 c(i)=i; end
      end
end  % range of  error
if e<fuz(1,3)% out of range
   i=1;
   m(i)=1;
   c(i)=i;
end
if e>fuz(a,3)% out of range
  i=a;
  m(i)=1;
  c(i)=i;
end
```

gauss_t2.m

```
function [m_low,m_up,c_low,c_up]=gauss_t2(fuz_up,fuz_low,e,a)
m_low=zeros(a,1);c_low=m_low;
m_up=zeros(a,1);c_up=m_up;
if(e>=fuz_low(1,3))&e<=(fuz_low(3,3))
      for i=1:a
          if fuz_low(i,1)==0
            i=i+1;
          end
            m_low(i)=exp(-(e-fuz_low(i,3))^2/(2*(fuz_low(i,2))^2));
            if m_low(i)~=0 c_low(i)=i; end
      end
end  % range of  error
if(e>=fuz_up(1,3))&e<=(fuz_up(3,3))
      for i=1:a
          if fuz_up(i,1)==0
            i=i+1;
```

```
            end
               m_up(i)=exp(-(e-fuz_up(i,3))^2/(2*(fuz_up(i,2))^2));
               if m_up(i)~=0 c_up(i)=i; end
        end
 end  % range of  error
if e<fuz_up(1,3)% out of range
    i=1;
    m_up(i)=1;
    c_up(i)=i;
 end
 if e>fuz_up(a,3)% out of range
   i=a;
   m_up(i)=1;
   c_up(i)=i;
 end
```

ifs.m

```
function [m,c,v,vc]=ifs(fuz,e,a)
m=zeros(a,1); c=m; v=m; vc=m;
  if(e>=fuz(1,3))&e<=(fuz(3,3))
       for i=1:a
           if fuz(i,1)==0
             i=i+1;
           end
              m(i)=exp(-(e-fuz(i,3))^2/(2*(fuz(i,2))^2));
              v(i)=(1-exp(-(e-fuz(i,3))^2/(2*(fuz(i,2))^2)))^2;
              if m(i)~=0 c(i)=i; end
       end
 end  % range of  error
 if e<fuz(1,3)% out of range
    i=1;
    m(i)=1;
    c(i)=i;
 end
```

```
 if e>fuz(a,3)% out of range
    i=a;
    m(i)=1;
    c(i)=i;
 end
```

SFNN_III_Newton_GPC.m

```
% nonlinear plant - CSTR
da=0.072; phi=20; b=8; delta=0.69;
%PREDICTIVE CONTROLLER PARAMETERS
Nu=5; N2=5; Ro=0.001; N1=1; fc=0.75;%fc=0.15;fu=0.25;
for ky=1:N2
   x1(ky)=0; x2(ky)=0; x3(ky)=0; x4(ky)=0;
   PredCon(ky)=0; PredErr(ky)=0;
end
dydu_first=zeros(N2,Nu);
epsf=0; epsf_1=0; PredOut=0; PredOut_1=0; PredOut_2=0;unf_1=0;
%Gaussian membership functions parameters
 fuz3=[1    0.2123     0
       1    0.2123    0.5
       1    0.2123     1];
     fuz=fuz3;
     as=size(fuz); a=as(1,1); Nrules=a^2;
     nref=0;nt=0;
    % Variable parameters
   w2=zeros(1,Nrules);
      b1=0.1*rand(1,Nrules);
      b2=0.1*rand(1,Nrules);
      b3=0.1*rand(1,Nrules);
      b4=0.1*rand(1,Nrules);
      b01=0.1*rand(1,Nrules);
      b02=0.1*rand(1,Nrules);
   d2=0;da2=w2;my=w2;fout=w2;ym=0;err=0;err_1=0;foutn=w2;
   fuzx1=fuz;fuzx2=fuz;fuzx3=fuz;fuzx4=fuz;fuzx5=fuz;
```

```
u1=0;ek=0;ek_1=0;ek_2=0;se_1=0;se=0;
y1n=0.144;y2n=0.866;y3n=0;un=0;yn=0;
y1n_1=0.144;y2n_1=0.866;y3n_1=0;un_1=0;
y1n_2=0;y2n_2=0;y3n_2=0;un_2=0;
Ref=0.886;
%=========START==========
for t=1:1000 %900
    tstart = tic;
  refplot(t)=Ref;%n(t)=noise;
e=Ref-yn;err_t(t)=e;
%========PREDICTIVE CONTROLLER==================
        x1(1)=y2n_1/6;
        x2(1)=y2n_2/6;
        x3(1)=(un+4)/8;
        x4(1)=(un_1+4)/8;
      epsf_1=0;
  for ku=1:Nu
  for ky=1:N2
%========FUZZYFICATION FNN1==================
        [mx1,cx1]=fuzzyg(fuzx1,x1(ky),a);
        [mx2,cx2]=fuzzyg(fuzx2,x3(ky),a);
        %Fuzzy Implication
        iii=0;
        for i=1:a
           for j=1:a
               iii=iii+1;
                if (cx1(i)==i) & (cx2(j)==j)
                       my(iii)=mx1(i)*mx2(j);
                   else
                       my(iii)=0;
                   end
               end
             end
        %====DEFUZZYFICATION============
        xx=[my]'; summy=0;
```

```
        for i=1:(Nrules)
           foutn(i)=b1(i)*x1(ky)+b2(i)*x2(ky)+b3(i)*x3(ky)+b4(i)*x4(ky)+b01(i);
           summy=summy+xx(i);
        end
        xx=xx/(summy+0.001);
        ymn=15*foutn*xx;
        PredOut=ymn; %{1} Predicted model output
        PredOut=6*foutn*xx; %{1} Predicted model output
%Filtration of predicted model error
eps=y2n-PredOut; epsf=(1-fc)*epsf_1+fc*eps; epsf_1=epsf; PredOut=PredOut+epsf_1;
%update input values for the next computation
if ky < N2
 x1(ky+1)=PredOut/6; x2(ky+1)=x1(ky);
 x3(ky+1)=(PredCon(ky+1)+4)/8; x4(ky+1)=x3(ky);
if (ky+1)>Nu  x3(ky+1)=x3(ky); x4(ky+1)=x4(ky); end
end
PredErr(ky)=Ref-PredOut; %predicted error

% computations of first derivative dy/du
if  ky==ku dydu_first(ky,ku)= b3*xx; end
if (ky-ku)==1 dydu_first(ky,ku)=(b1*dydu_first(ky,ku)+b4)*xx; end
if (ky-ku)>=2 dydu_first(ky,ku)=(b1*dydu_first(ky,ku)+b2*dydu_first(ky-1,ku))*xx; end

% computations of second derivative dy/du
dydu_second = dydu_first^2;
Hes=inv(dydu_second);
rrr=Hes*dydu_first;

end %for ky=1:N2
end %for ku=1:Nu
% computations of control actions
for ku=Nu:-1:1
  if ku==Nu deltau(ku)=(Ro*PredErr(N1:N2)*rrr(N1:N2,ku));end
  if ku<Nu  deltau(ku)=deltau(ku+1)+(Ro*PredErr(N1:N2)*rrr(N1:N2,ku));end
end
```

```
for ku=1:Nu
  if ku==1
    PredCon(ku)=deltau(ku)+un_1;
    if PredCon(ku)<-4 PredCon(ku)=-4; end
    if PredCon(ku)>4 PredCon(ku)=4; end
    un=PredCon(ku);
  end
  if ku>1  PredCon(ku)=deltau(ku)+PredCon(ku-1);
    if PredCon(ku)<-4 PredCon(ku)=-4; end
    if PredCon(ku)>4 PredCon(ku)=4; end
  end
end
%============ PLANT ================
% discrete equations of the nonlinear CSTR
expo=da*exp(y2n_1/(1+y2n_1/phi));
    y1n=0.5*(y1n_1+expo*(1-y1n_1));
    y2n=(y2n_1+b*expo*(1-y1n)+delta*un)/(2+delta);
if y2n<0 y2n=0; end

%=====SET-POINT CHANGE========
if t>=200 Ref=2.75;  end
if t>=600 Ref=1.4;  end
if t>=800 Ref=1.6;  end

%update previous values
y1n_2=y1n_1;y1n_1=y1n;
y2n_2=y2n_1;y2n_1=y2n;
un_2=un_1;un_1=un;
PredOut_2=PredOut_1; PredOut_1=PredOut;telapsed = toc(tstart);
yn=y2n;
y1plot(t)=y1n;
y2plot(t)=y2n;
uplot(t)=un;
figure(1);
plot(y2plot,'r');hold on;
```

```
plot(refplot,'g');
%plot(y1plot,'b');
%plot(uplot,'y');
grid;
xlabel('TIME (steps)');
ylabel('REFERENCE and SYSTEM OUTPUT');
hold off;
end % epochs
```

LM_LU_GPC.m

```
% ====== INITIAL PLANT PARAMETERS =========
% nonlinear plant - CSTR
da=0.072; phi=20; b=8; delta=0.69;
%PREDICTIVE CONTROLLER PARAMETERS
Nu=5; N2=5; Ro=0.01; N1=1; fc=0.75;
lambda=10^4; ed_matrix=eye(5,5);
for ky=1:N2
  x1(ky)=0; x2(ky)=0; x3(ky)=0; x4(ky)=0;
  PredCon(ky)=0; PredErr(ky)=0; PredErr_1(ky)=0;
end
dydu_first=zeros(N2,Nu); dydu_second=zeros(N2,Nu);
epsf=0; epsf_1=0; PredOut=0; unf_1=0;
%Gaussian membership functions parameters
 fuz3=[1    0.2123     0
       1    0.2123    0.5
       1    0.2123     1];
    fuz=fuz3;
    as=size(fuz); a=as(1,1); Nrules=a^4;
   lr=0.002;
   nref=0;nt=0;
   % Variable parameters
  w2=zeros(1,Nrules);
  b1=0.1*rand(1,Nrules);b2=0.1*rand(1,Nrules);b3=0.1*rand(1,Nrules);
```

```
  b4=0.1*rand(1,Nrules);b5=0.1*rand(1,Nrules);b0=0.1*rand(1,Nrules);
  d2=0;da2=w2;my=w2;fout=w2;ym=0;err=0;err_1=0;foutn=w2;
  fuzx1=fuz;fuzx2=fuz;fuzx3=fuz;fuzx4=fuz;fuzx5=fuz;
u1=0;ek=0;ek_1=0;ek_2=0;se_1=0;se=0;
y1n=0.144;y2n=0.866;y3n=0;un=0;yn=0;
y1n_1=0.144;y2n_1=0.866;y3n_1=0;un_1=0;
y1n_2=0;y2n_2=0;y3n_2=0;un_2=0;
Ref=0.886;
%=========START==========
for t=1:800 %900
   tstart = tic;
  refplot(t)=Ref;%n(t)=noise;
e=Ref-yn;err_t(t)=e;
%========PREDICTIVE CONTROLLER==================
  x3(1)=(un+4)/8; x4(1)=(un_1+4)/8; x1(1)=y2n_1/6; x2(1)=y2n_2/6; epsf_1=0;
  for ku=1:Nu
  for ky=1:N2

%======FUZZYFICATION=======
[mx1,cx1]=fuzzyg(fuzx1,x1(ky),a);
[mx2,cx2]=fuzzyg(fuzx2,x2(ky),a);
[mx3,cx3]=fuzzyg(fuzx3,x3(ky),a);
[mx4,cx4]=fuzzyg(fuzx4,x4(ky),a);
% Fuzzy implication
iii=0;
for i=1:a
   for j=1:a
      for k=1:a
        for l=1:a
            iii=iii+1;
  if (cx1(i)==i) & (cx2(j)==j) & (cx3(k)==k)& (cx4(l)==l)
      my(iii)=mx1(i)*mx2(j)*mx3(k)*mx4(l);
  else
    my(iii)=0;
      end
```

```
     end
   end
  end
end
%  DEFUZZIFICATION
xx=[my]';summy=0;
for i=1:(Nrules)
   foutn(i)=b1(i)*x1(ky)+b2(i)*x2(ky)+b3(i)*x3(ky)+b4(i)*x4(ky)+b0(i); %{1}
   summy=summy+xx(i);
end
xx=xx/(summy+0.001);
%ym=fout*xx;
PredOut=foutn*xx;
%Filtration of predicted model error
eps=y2n-PredOut; epsf=(1-fc)*epsf_1+fc*eps; epsf_1=epsf; PredOut=PredOut+epsf_1;
%update input values for the next computation
if ky < N2
x1(ky+1)=PredOut/6; x2(ky+1)=x1(ky);
x3(ky+1)=(PredCon(ky+1)+4)/8; x4(ky+1)=x3(ky);
if (ky+1)>Nu  x3(ky+1)=x3(ky); x4(ky+1)=x4(ky); end
end
PredErr(ky)=Ref-PredOut; %predicted error
% computations of first derivative dy/du
if  ky==ku dydu_first(ky,ku)= b3*xx; end
if (ky-ku)==1 dydu_first(ky,ku)=(b1*dydu_first(ky,ku)+b4)*xx; end
if (ky-ku)>=2 dydu_first(ky,ku)=(b1*dydu_first(ky,ku)+b2*dydu_first(ky-1,ku))*xx; end
% computations of second derivative dy/du
dydu_second = dydu_first*dydu_first';
LM=dydu_second+lambda*ed_matrix;
if LM(1,1)==0
   LM(1,1)=LM(1,1)+0.000001;
end
[L,U] = lu(LM);
x=U\(L\dydu_first);
rrr=x*dydu_first;
```

```
end %for ky=1:N2
end %for ku=1:Nu
% computations of control actions
for ku=Nu:-1:1
  if ku==Nu deltau(ku)=(Ro*PredErr(N1:N2)*rrr(N1:N2,ku));end
  if ku<Nu  deltau(ku)=deltau(ku+1)+(Ro*PredErr(N1:N2)*rrr(N1:N2,ku));end
end
for ku=1:Nu
  if ku==1
    PredCon(ku)=deltau(ku)+un_1;
    if PredCon(ku)<-4 PredCon(ku)=-4; end
    if PredCon(ku)>4 PredCon(ku)=4; end
    un=PredCon(ku);
  end
  if ku>1  PredCon(ku)=deltau(ku)+PredCon(ku-1);
    if PredCon(ku)<-4 PredCon(ku)=-4; end
    if PredCon(ku)>4 PredCon(ku)=4; end
  end
end
if PredErr(1)>PredErr_1(1) lambda=0.005*lambda;
else lambda=1.1*lambda;
end
deltau_1=deltau;
PredErr_1=PredErr;
%============ PLANT ================
% discrete equations of the nonlinear CSTR
expo=da*exp(y2n_1/(1+y2n_1/phi));
    y1n=0.5*(y1n_1+expo*(1-y1n_1));
    y2n=(y2n_1+b*expo*(1-y1n)+delta*un)/(2+delta);
if y2n<0 y2n=0; end
%=====SET-POINT CHANGE========
if t>=150 Ref=1.7;  end
if t>=550 Ref=1.4;  end
if t>=650 Ref=1.6;  end
%update previous values
```

```
y1n_2=y1n_1;y1n_1=y1n;
y2n_2=y2n_1;y2n_1=y2n;
un_2=un_1;un_1=un;
yn=y2n; lambdaplot(t)=lambda;telapsed = toc(tstart);
y2plot(t)=y2n; uplot(t)=un;
figure(1);
plot(y2plot,'r');hold on;
plot(uplot,'b');
plot(refplot,'g');
grid;
xlabel('TIME (steps)');
ylabel('REFERENCE, SYSTEM OUTPUT and CONTROL ACTON');
hold off;
end % epochs
```

Списък с публикации по дисертационния труд

Доклади в списания

1. **Terziyska**, M., A Distributed Adaptive Neuro-Fuzzy Network for Chaotic Time Series Prediction, Cybernetics and Information Technologies. Vol. 15, Issue 1, Pages 24–33, ISSN (Online) 1314-4081, DOI: 10.1515/cait-2015-0003, March 2015.

2. **Terziyska**, M., Y. Todorov, Fuzzy-neural predictive control using fast optimization polices, International Journal of Intelligent Reasoning-Based Systems, vol. 6, issue 3/4, pp. 136-144, ISSN online: 1755-0564, ISSN print: 1755-0556, 2014. DOI: 10.1504/IJRIS.2014.066250.

3. **Terziyska**, M., Y. Todorov, M. Petrov, Real-time Supervisory tuning of Predictive Controller, International Conference TechSys'2013, Plovdiv Bulgaria, Published in Journal of Technical University-Sofia, branch Plovdiv, Bulgaria, "Fundamental Sciences and Applications" Vol. 19, pp. 155-160, ISSN 1310-8271, 2013.

4. **Terziyska**, M., Y. Todorov, L. Doukovska, Neo-fuzzy Network for Modeling of Nonlinear MIMO Dynamics, International Conference TechSys'2015, Plovdiv Bulgaria, Published in Journal of Technical University-Sofia, branch Plovdiv, Bulgaria, "Fundamental Sciences and Applications" Vol. 21, book 1, pp. 65-70, ISSN 1310-8271, 2015.

Доклади от международни конференции

5. **Terzyiska**, M., L. Doukovska, M. Petrov, Implicit Generalized Predictive Controller Based on Semi Fuzzy Neural Network Model, Proc. of the 7th IEEE International Conference Intelligent Systems - IS'14, September 24–26 2014, Warsaw, Poland, ISSN 2194-5357, ISSN 2194-5365 (electronic), ISBN 978-3-319-11312-8, ISBN 978-3-319-11313-5 (eBook), DOI 10.1007/978-3-319-11313-5, Volume 1: Mathematical Foundations, Theory, Analyses, Springer International Publishing, Switzerland, P. Angelov et al. (eds.), Advances in Intelligent Systems and Computing vol. 322, pp. 695-706, 2015.

6. Todorov, Y., M. **Terziyska**, S. Ahmed, M. Petrov, Fuzzy-Neural Predictive Control using Levenberg-Marquardt optimization approach, In Proceedings of International IEEE conference on Innovations in Intelligent Systems and

Applications, INISTA'2013, Albena, Bulgaria, pp. 1-5, ISBN 978-1-4799-0659-8, 2013.

7. Todorov, Y., M. **Terziyska**, Modeling of Chaotic time series by Interval Type-2 NEO-Fuzzy Neural Network, International Conference on Artificial Neural Networks (ICANN'2014), Hamburg, Germany, Springer Lecture Notes on Computer Science, vol. 8681, pp. 643-650, ISBN 978-3-319-11178-0, ISSN 0302-9743, 2014.

Библиография

Велев, К. (1994). Адаптивни системи. *София*.

Доневска, С., И. Трендафилов (1994). Линейна алгебра и аналитична геометрия. *Издателство „Техника”, София*.

Стоянов, С. (1990). Методи и алгоритми за оптимизация. *Издаделство "Техника", София.*

Хаджийски, М., К. Бошнаков (2009). Нелинейно моделно предсказващо управление на процесите на пречистване на отпадъчни води с включване на логически правила. *International Conference “Automatics and Informatics’09”, Sofia, Bulgaria*. pp.IV-7 - IV-10, ISSN 1313-1850.

Abdi, J., H. Ahmadi and H. Borhanifar (2009). A novel model-based predictive wavelet network strategy for control of spark injection engines. In *International Conference on Machine Learning and Computing*, pp. 11–16.

Abiyev, R. H. and O. Kaynak (2008). Fuzzy wavelet neural networks for identificationand control of dynamic plants - a novel structure and comparative study. In *IEETransactions on Industrial Electronics,* Vol. 55, Issue 8, pp. 3133–3140.

Abiyev, R. H. and V. H. Abiyev (2012). Differential Evaluation Learning of Fuzzy Wavelet Neural Networks for Stock Price Prediction. *Journal of Information and Computing Science,* Vol. 7, No 2, pp. 121-130.

Abiyev, R., O. Kaynak and Y. Oniz (2012). Spiking neural networks for identification and control of dynamic plants. In *2012 IEEE/ASME international conference on advanced intelligent mechatronics*. AIM pp. 1030–1035.

Abonyi, J., Á. Bódizs, L. Nagy and F. Szeifert (1999). Predictor Corrector Controller using Wiener Fuzzy Convolution Model. *Hungarian Journal of Industrial Chemistry*, Vol. 27, Issue 3, pp. 227-233.

Abonyi, J., L. Nagy and F. Szeifert (2001). Fuzzy model-based predictive control by instantaneous linearization. *Fuzzy Sets and Systems*, 120(2001), pp. 109–122.

Abonyi, J., R. Babuska, F. Szeifert, L. Nagy and H. Verbruggen (2000). Design and Application of Block-Oriented Fuzzy Models - Fuzzy Hammerstein Model. *Soft Computing in Industrial Applications, Springer-London*.

Åkesson, B. M. and H. T. Toivonen (2006). A neural network model predictive controller. *Journal of Process Control*, Vol. 16, No. 3, pp. 937–946.

Alavala, C. (2008). Fuzzy Logic and Neural Networks: Basic Concepts and Applications. *New Age International Pvt Ltd.*

Al-Ghazzawi, A., E. Ali, A. Nouh and E. Zafiriou (2001). On-line tuning strategy for model predictive controllers. *Journal of Process Control*, Vol. 11, Issue 3, pp. 265– 284.

Ali, E. (2003). Heuristic on-line tuning for nonlinear model predictive controllers using fuzzy logic. *Journal of Process Control*, Vol. 13, pp. 383–396.

Ali, E. and E. Zafiriou (1993). Optimization-based tuning of non-linear model predictive control with state estimation. *Journal of Process Control,* Vol. 3, pp. 97–107.

Allende, H. et al (2008). Self-Organizing Neuro-Fuzzy Inference System. *Iberoamerican Congress on Pattern Recognition CIARP*, pp. 429-436.

Allgöwer, F., R. Findeisen and Z. K. Nagy (2004). Nonlinear Model Predictive Control: From Theory to Application. *J. Chin. Inst. Chem. Engrs.*, Vol. 35, No. 3, 299-315.

Angelov, P. (2013). Autonomous Learning Systems. *Wiley*.

Angelova, V. (2009). Investigations in the area of Soft computing. Targeted state of the art report. *Cybernetics and Information Technologies*, Vol. 9, No 1, 18-24, 2009, ISSN 1311-9702.

Astrom, K.J. and B. Wittenmark (1973). Onself tuning regulators, *Automatica*, Vol. 9, Issue 2, pp. 185-199.

Atanassov, K. (1999). Intuitionistic Fuzzy Sets. *Springern, Hielderberg*.

Azhar, A. S. S. and H. N. Al-Duwaish (2002). Identification of Wiener Model Using Radial Basis Functions Newral Networks, *J.R. Dorronsoro (Ed.): ICANN 2002, LNCS 2415*, pp. 344–350.

Babuska, R. (1998). Fuzzy Modeling for Control, *Kluwer Academic Publishers, Boston.*

Bellaaj, H., R. Ketata and M. Chtourou (2013). A new method for fuzzy rule base reduction. *Journal of Intelligent & Fuzzy Systems*, Vol. 25, pp. 605–613.

Bherenji, H. R. and P. Khedkar (1992). Learning and Tuning Fuzzy Logic Controllers through Reinforcements. *IEEE Trans. on Neural Networks*, Vol (3), pp. 724-740.

Bououden, S. et al (2009). Constrained fuzzy based model predictive control with disturbances. *IJ-STA*, Vol. 3, No. 1, pp. 986 – 995.

Brdys, M. A., M. Grochowski, T. Gminski, K. Konarczak and M. Drewa (2008). Hierarchical predictive control of integrated wastewater treatment systems. *Control Engineering Practice*, Vol. 16, Issue 6, pp. 751– 767.

Camacho, E. F. and C. Bordons (2007). Nonlinear Model Predictive Control: An Introductory Review. *Assessment and Future Directions of Nonlinear Model Predictive Control, Lecture Notes in Control and Information Sciences*, Vol. 358, pp. 1-16.

Castellano, G. (2000). A Neurofuzzy Methodology for Predictive Modeling. PhD Thesis.

Chakrabarty, A., S. Banerjee, S. Maity and A. Chatterjee (2013). Fuzzy model predictive control of non-linear processes using convolution models and foraging algorithms. *Measurement*, Vol. 46, pp. 1616–1629.

Chandana, S. and R. V. Mayorga (2006). RANFIS: Rough adaptive neuro-fuzzy inference system. *International Journal of Computational Intelligence*, Vol. 3, Issue 4, pp. 289-295.

Chang, T. S. and D. E. Seborg(1983). A linear programming approach to multivariable feedback control with inequality constraints. *International Journal of Control*, Vol. 37, pp. 583-597.

Chen, H. and F. Allgöwer (1998). A quasi-infinite horizon nonlinear model predictive control scheme with guaranteed stability. *Automatica*, Vol. 34, Issue 10, pp. 1205-1218.

Chidrawar, S. and B. Patre (2008). Generalized Predictive Control and Neural Generalized Predictive Control. *Leonardo Journal of Sciences*, Issue 13, pp. 133-152.

Ciliz, M. K. (2005). Rule base reduction for knowledge-based fuzzy controllers with application to a vacuum cleaner. *Expert Systems with Applications*, Vol. 28, pp. 175–184.

Clarke, D.W. and P. J. Gawthrop (1975). Self-tuning controller. *Proc. IEE Part D*, Vol. 122, pp. 929-934.

Clarke, D.W., C. Mohtadi and P.S. Tuffs (1987). Generalized Predictive Control. *Automatica*, Vol. 23, Issue 2, pp. 137-148.

De Almeida, G. M., J. L. F. Salles and J. D. Filho (2006). Using genetic algorithms for tuning the parameters of generalized predictive control. *VII Conferencia Internacional de Aplicações Industriais INDUSCON,* Recife.

De Almeida, G. M., J. L. F. Salles and J. D. Filho (2009). Optimal tuning parameters of the dynamic matrix predictive controller with constraints. *Lat. Am. appl. res*., Vol.39, No.1, pp. 33-40. ISSN 0327-0793.

De AraújoJúnior, J. Medeiros and F. M. U. de Araújo (2012). Identification Nonlinear by Fuzzy Wavelet Neural Networks. *ABCM Symposium Series in Mechatronics*, Vol. 5.

De Nicolao, G., L. Magni and R. Scattolini (1996). Stabilizing nonlinear receding horizon control via a nonquadratic terminal state penalty. *In Symposium on Control, Optimization and Supervision, CESA'96 IMACS Multiconference*, pp. 185-187, Lille.

Diehl, M. (2001). Real-time Optimization for Large Scale Nonlinear Processes. *PhD Thesis, University of Heidelberg*.

Dolotov, A. and Y. Bodyanskiy (2009). Analog-Digital Self-Learning Fuzzy Spiking Neural Network in Image Processing Problems. *Image Processing, Yung-Sheng Chen (Ed.)*, ISBN: 978-953-307-026-1, InTech.

Dougherty, D. and D. Cooper (2003). Tuning Guidelines of a Dynamic Matrix Controller for Integrating (Non-Self-Regulating) Processes. *Ind. Eng. Chem. Res.*, Vol. 42, pp. 1739–1752.

Esmaili, A., M. Shahbazian and B. Moslemi (2013). Nonlinear Process Identification Using Fuzzy Wavelet Neural Network Based on Particle Swarm Optimization Algorithm. *J. Basic. Appl. Sci. Res.,* Vol. 3, Issue 5, 302-309.

Falcone, P., F. Borrelli, H. E. Tseng, J. Asgari and D. Hrovat (2008). A Hierarchical Model Predictive Control Framework for Autonomous Ground Vehicles. *American Control Conference, Westin Seattle Hotel, Seattle, Washington, USA.*

Feng, J. Ch. and L. Ch. Teng (1998). An Online Self Constructing Neural Fuzzy Inference Network and its Applications. *IEEE Transactions on Fuzzy Systems*, Vol. 6, No. 1, pp. 12-32.

Ferreyra, A., J. J. Rubio(2006). A new on-line self-constructing neural fuzzy network. *Proceedings of the 45th IEEE Conference on Decision & Control, San Diego, CA, USA.*

Findeisen, R. and F. Allgöwer (2002). An Introduction to Nonlinear Model Predictive Control. *21st Benelux Meeting on Systems and Control, Veldhoven*.

Findeisen, R. and F. Allgower, L. T. Biegler (Eds.) (2007). Assessment and Future Directions of Nonlinear Model Predictive Control. *Springer-Verlag Berlin Heidelberg*.

Fink, A., S. Töpfer and R. Isermann (2003). Nonlinear model-based control with local linear neuro-fuzzy models. *Archive of Applied Mechanics, Springer-Verlag*, Vol.72, pp. 911–922.

Gawthrop, P. J., H. Demircioglu and I. Siller-Alcala (1998). Multivariable continuous-time generalised predictive control: A state-space approach to linear and nonlinear systems. *CSC Research Report, CSC-98001, Centre for Systems and Control, University of Glasgow*.

Gegov, A. (2007). Complexity Management in Fuzzy Systems. *Studies in Fuzziness and Soft Computing*, Vol. 211.

Gegov, A. and N. Gobalakrishnan (2007). Advanced Inference in Fuzzy Systems by Rule Base Compression. *Mathware& Soft Computing*, Vol. 14, pp. 201-216.

Glackin, C. et al. (2008). Classification using a fuzzy spiking neural network. *The 2008 UK workshop on computational intelligence (UKCI).*

Grosso, J.M., C. Ocampo-Martınez and V. Puig (2012). Adaptive Multilevel Neuro-Fuzzy Model Predictive Control for Drinking Water Networks. *20th*

Mediterranean Conference on Control&Automation (MED), Barcelona, Spain.

Grune, L. and J. Pannek (2011). Nonlinear Model Predictive Control Theory and Algorithms. *Springer-Verlag London Ltd*.

Gu, D. and H. Hu (2000). Wavelet neural network based predictive control for mobile robots. *Systems, Man, and Cybernetics, 2000 IEEE International Conference*, Vol. 5.

Hachino, T., K. Deguchi and H. Takata (2004). Identification of Hammerstein Model Using Radial Basis Function Networks and Genetic Algorithm. *The 5th Asian Control Conference, ASOC, Grand Hyatt-Melbourne, Australia*.

Hajimolana, S. A., M. A. Hussain, W. M. A. Wan Daud and M. H. Chakrabarti (2012). Neural Network Predictive Control of a SOFC Fuelled with Ammonia. *In J. Electrochem. Sci.*, Vol. 7, pp. 3737–3749.

Hedjar, R. (2013). Adaptive neural network model predictive control. *In Journal of Innovative Computing, Information and Control*, Vol. 9, No. 3.

Hosen, M. A., M. A. Hussain and F. S. Mjalli (2011). Control of polystyrene batch reactors using neural network based model predictive control (NNMPC): An experimental investigation. *Control Engineering Practice*, Vol. 19.

Huaguang, Z. and L. Cai (2002). Multivariable fuzzy generalized predictive control. *Cybernetics and Systems: An International Journal*, Vol. 33, pp. 69-99.

Huang, Y. L., H. H. Lou, J. P. Gong and T. F. Edgar (2000). Fuzzy Model Predictive Control. *IEEE Transactions on Fuzzy Systems*, Vol. 8, No. 6.

Ibarrola, J. J., M. Pinzolas and J. M. Cano (2006). A neurofuzzy scheme to on-line identification in an adaptive–predictive control. *Springer London, Neural Computing & Applications*, Vol. 15, No. 1.

Jamshidi, M. (1997). Fuzzy Control Systems. *Springer-Verlag, chapter of Soft Computing*, pp. 42-56.

Jang, J. S. R. (1991). ANFIS: Adaptive Network based Fuzzy Inference Systems. *IEEE Transactions, Man & Cybernetics*.

Jia, L., M. Chiu, S. S. Ge (2004). Neuro-fuzzy system based identification method for Hammerstein processes. *Control Conference, 5th Asian*, Vol. 1,

pp.104 - 111.

Jiang, A. and A. Jutan (2000). Response surface tuning methods in dynamic matrix control of a pressure tank system. *Ind. Eng. Chem. Res.*, Vol. 39, pp. 3833– 3843.

Jin, Y. (2000). Fuzzy Modeling of High-Dimensional Systems: Complexity Reduction and Interpretability Improvement. *IEEE Transactions on Fuzzy Systems*, Vol. 8, No. 2.

Johnson, C., G. K. Venayagamoorthy and P. Mitra (2009). Comparison of a spiking neural network and an MLP for robust identification of generator dynamics in a multimachine power system. *Neural Networks*, Vol. 22, Issue 5.

Karnik, N.N. and J.M.Mendel (1998). Introduction to Type-2 fuzzy logic systems. In *Proceedings of the IEEE FUZZ Conference*, Anchorage, pp. 915– 920.

Kasabov, N. (2001). Evolving Fuzzy Neural Networks for Supervised/Unsupervised On-line Knowledge-Based Learning. *IEEE Trans of Systems, Man and Cybernetics, Part B – Cybernetics*, Vol. 31, No. 6, pp. 902-918.

Kasabov, N. and Q. Song (1999). Dynamic Evolving Fuzzy Neural Networks with ‘m-out-of-n‘ Activation Nodes for On-line Adaptive Systems. *Technical Report TR99/04, Department of information science, University of Otago*.

Kasabov, N. and Q. Song (2002). DENFIS: Dynamic, evolving neural-fuzzy inference systems and its application for time-series prediction. *IEEE Trans. on Fuzzy Systems*, Vol. 10, Issue 2.

Kasabov, N. et al. (2013). Dynamic evolving spiking neural networks for on-line spatio-and spectro-temporal pattern recognition. *Neural Networks 41*, pp. 188–201.

Kaur, N., Sh. Kundra, H. Kundra and S. Singh (2012). Rule Based Reduction Using Dominant Rule Determination. *IJCTEE*, Vol. 2, Issue 3.

Kaynak, O., K.Jezernik and A.Szeghegyi (2002). Complexity reduction of rule based models: a Survey. *IEEE Tran.*, pp.1216- 1221.

Kerrigan,E.C. and J.M. Maciejowski (2002). Designing model predictive controllers with prioritised constraints and objectives. In *IEEE international symposium on computer aided control system design*, pp. 33–38.

Kim, K. J., J. B. Park and Y. H. Choi (). Wavelet Neural Network Based Generalized Predictive Control of Chaotic Systems Using EKF Training Algorithm. *ICCAS2005, KINTEX, Gyeonggi-Do, Korea*, pp. 2521-2525.

Kittisupakom, P., P. Thitiyasook, M. A. Hussein and W. Daosud (2009). Neural network based model predictive control for a steel pickling process. *Journal of Process Control*, Vol.19, No.4, pp. 579-590.

Kouvaritakis, B. and M. Cannon (2001). Nonlinear predictive control: theory and practice. *IEE Control Series*, Vol. 61.

Kulkarni, A., and A. Kumar (2012). Dynamic Recurrent Wavelet Neural Network Observer Based Tracking Control for a Class of Uncertain Nonaffine Systems. *International Journal of Intelligent Systems and Applications (IJISA)*, Vol. 4, Issue 11.

Lawrynczuk, M. (2009). Neural Networks in Model Predictive Control. *Intelligent Systems for Knowledge Management*, *Studies in Computational Intelligence*, Vol. 252, pp. 31-63.

Lawrynczuk, M. (2010). Computationally efficient nonlinear predictive control based on neural Wiener models. *Neurocomputing*. Vol. 74, pp. 401–417.

Lawrynczuk, M. and P.Tatjewski (2010). Nonlinear predictive control based on neural multi–models. *Int. J. Appl. Math. Comput. Sci.*, Vol. 20, No. 1, pp. 7–21.

Ledeneva, Y. (2006). Automatic Estimation of Parameters to Reduce Rule Base of Fuzzy Control Complex Systems. *Master thesis, INAOE Mexico*.

Lee, J. H. (2011). Model predictive control: Review of three decades of development. In *Int. J. Control, Automation and Systems*, Vol. 9, No. 3, pp. 415–424.

Lee, J. H., (2000). Modeling and Identification for Nonlinear Model Predictive Control: Requirements, Current Status and Future Research Needs. *Progress in Systems and Control Theory*, Vol. 26, Basel, Boston, Berlin: BirkhauserVerlag.

Lepetič, Marko, Igor Škrjanc, Hector G. Chiacchiarini, Drago Matko (2003). Predictive functional control based on fuzzy model: magnetic suspension system case study. *Engineering Applications of Artificial Intelligence*, Vol. 16, pp. 425–430.

Li, Zh. (2010). Intelligent Fuzzy Predictive Controller Design for Multivariable Process System. *Journal of Computational Information Systems,* Vol. 6, Issue 9, pp. 3003-3011.

Li, X., Z. Chen and Z. Yuan, (2002). Simple recurrent neural network-based adaptive predictive control for nonlinear systems. *Asian Journal of Control*, Vol. 4, No. 2, pp. 231-239.

Liao, Q., Ning Li, Shaoyuan Li (2009). Type-II T-S Fuzzy Model-based Predictive Control. *Joint 48th IEEE Conference on Decision and Control and 28th Chinese Control Conference Shanghai, P.R. China*.

Lin, C. J. and C. C. Peng (2012). System Identification Using Fuzzy Cerebellar Model Articulation Controllers. *Fuzzy Inference System - Theory and Applications, Dr. Mohammad FazleAzeem (Ed.)*, InTech.

Lin, C. J., J. H. Lee and C. Y. Lee (2008). A novel hybrid learning algorithm for parametric fuzzy CMAC networks and its classification applications. *Expert Systems with Applications,* Vol. 35, pp. 1711–1720.

Lin, C. T. and C. S. G. Lee (1991). Neural Network based Fuzzy Logic Control and Decision System. *IEEE Trans. on Comput.*, Vol. 40, Issue 12, pp. 1320-1336.

Liu, X. and J. Liu (2006). Neuro-fuzzy Generalized Predictive Control of Boiler Steam Temperature. *Springer-Verlag Berlin Heidelberg, J. Wang et al. (Eds.), LNCS 3972*, pp. 1027–1032.

Liu, X. J., L. X. Niu and J. Z. Liu (2009). Nonlinear Multivariable Supervisory Predictive Control. *American Control Conference*.

Liu, Z. (2010). Chaotic time series analysis. *Mathematical Problems in Engineering*, Vol. 2010.

Liutkevičius, R. and S. Dainys (2005). Hybrid Fuzzy Model of Nonlinear Plant. *Information Technology and Control*, Vol.34, No.1.

Lu, C. H. and C. C. Tsai (2007). Generalized predictive control using recurrent fuzzy neural networks for industrial processes. *Journal of Process Control,* Vol. 17, pp. 83–92.

Lu, C. H. and C.C. Tsai (2004). Adaptive neural predictive control for industrial multivariable processes. *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, Vol. 218, No 7.

Lu, C. H., C. M. Liu and Y. H. Charng (2011). Stable predictive control based on recurrent fuzzy neural networks. *Control Conference (ASCC), 8th Asian. IEEE*.

Maciejowski, J. M.(2002). Predictive Control with Constraints. *Prentice Hall*.

Magni, L. and R. Scattolini (2010). Automotive Model Predictive Control Lecture. *Notes in Control and Information Sciences*, Vol. 402, pp. 107-117.

Mendes, J., N. Sousa and R. Araújo (2011). Adaptive predictive control with recurrent fuzzy neural network for industrial processes. *Emerging Technologies & Factory Automation (ETFA), 16th Conference on IEEE*.

Mendes, J., R. Araujo (2012). Fuzzy Model Predictive Control for Nonlinear Processes. *17th IEEE International Conference on Emerging Technologies & Factory Automation*, *Kraków, Poland*.

Mendes, J., R. Araujo and F. Souza (2013). Adaptive fuzzy identification and predictive control for industrial processes. *Expert Systems with Applications*, Volume 40, Issue 17, pp. 6964–6975.

Mishra, D., A. Yadav and P. K. Kalra (2006). A novel multiplicative neural network architecture motivated by spiking neuron model. *J. Indian Inst. Sci*, Vol. 86, pp. 465-479.

Mjalli, F. S. (2005). Neural network model-based predictive control of liquid–liquid extraction contactors. *Chemical Engineering Science*, Vol. 60, pp. 239–253.

Mohajeri, K., M. Zakizadeh, B. Moaveni and M. Teshnehlab(2009). Fuzzy CMAC Structures. *Proceedings of IEEE International Conference on Fuzzy Systems, Korea*, pp. 2126- 2131.

Mollov, S. (2002). Fuzzy Control of Multiple-Input Multiple-Output Processes.

PhD thesis Delft University of Technology, Netherlands.

Morari, M. and J. H. Lee (1999). Model Predicitve Control: Past, Present, and Future. *Comput. Chem. Eng.*, Vol. 23, pp. 667.

Nauk, D. and R. Kruse (1992). A Neuro Fuzzy Controller Learning by Fuzzy Error Propagation. In *Proceedings of the North American Fuzzy Information Processing Society NAFIPS'92, Mexico*, pp. 388-397.

Ngia, K.S.H. and J. Sjobert (1998). Nonlinear acoustic echo cancellation using a Hammerstein model. *Acoustics, Speech, and Signal Processing, ICASSP '98. Proceedings of the 1998 IEEE International Conference*, Vol. 2, pp. 1229-1232.

Nguyen, M.N., J.F. Guo and D. Shi (2006). ESOFCMAC: Evolving Self-Organizing Fuzzy Cerebellar Model Articulation Controller. *International Joint Conference on Neural Networks, Vancouver, BC, Canada*.

Niu, L. and J. Li (2013). Adaptive Neural Network Generalized Predictive Control for Unknown Nonlinear System. *TELKOMNIKA*, Vol. 11, No. 7, pp. 3611-3617.

Nørdaard, M, O. Ravn, N. K. Poulsen and L. K. Hansen (2000). Neural Networks for Modeling and Control of Dynamic Systems: A Practitioner's Handbook. *Advanced Textbooks in Control and Signal Processing, Springer-Verlag, London*.

Oh, S., S. Joo, Ch. Jeong and H. Kim (2003). Self-Organizing Hybrid Neurofuzzy Networks. *International Conference Melbourne, Australia and St. Petersburg, Russia, Proceedings, Part IV*, pp. 877-885.

Orthogonal Transformation Methods. *IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics*, Vol. 29, No. 1.

Ortiz, F., et al. (2007). Recurrent Fuzzy CMAC for Nonlinear System Modeling. *Advances in Neural Networks, Springer Berlin Heidelberg*, pp. 487-495.

Pan, Y. and J. Wang (2012). Model predictive control of unknown nonlinear dynamical systems based on recurrent neural networks. *IEEE Trans. Ind. Electron.*, vol. 59, no. 8, pp. 3089–3101.

Patan, K. and J. Korbicz (2012). Nonlinear Model Predictive Control of a Boiler Unit: A Fault Tolerant Control Study. *Int. J. Appl. Math. Comput. Sci.*, Vol. 22, No. 1, pp. 225–237.

Paulusová J. and M. Dúbravská (2009). Hybrid Predictive Controller Based on Fuzzy-Neuro Model. *17th International Conference on Process Control 2009, StrbskePleso, Slovakia*.

Pearson, R. K., (2003). Selecting nonlinear model structures for computer control. *Journal of Process Control*, Vol. 13, Issue 1, pp. 1-26.

Peneva, V., I. Popchev (2009). Models for fuzzy multicriteria decision making based on fuzzy relations. *Compt. Rend. Acad. Bulg. Sci.*, Vol. 62, No. 5, 551-558, ISSN: 1310-1331.

Peng, H., T. Ozaki, Y. Toyoda, H. Shioya, K. Nakano, V. Haggan-Ozaki and M. Mori (2004). RBF-ARX model based nonlinear system modeling and predictive control with application to a NOx decomposition process. *Control Engineering Practice*, Vol. 12, pp. 191-203.

Ponulak, F. and A. Kasinski (2011). Introduction to spiking neural networks: Information processing, learning and applications. *Acta neurobiologia experimentalis*, Vol. 71, no. 4, pp. 409.

Primož, P. and I. Grabec (2002). Nonlinear model predictive control of a cutting process. *Neurocomputing*, Vol. 43, pp. 107–126.

Ramirez, D. R., M. R. Arahal and E. F. Camacho(2004). Min-Max Predictive Control of a Heat Exchanger Using a Neural Network Solver. *IEEE Trans. On Control Systems Technology*, Vol. 12, No 5.

Rani, K. Y. and H. Unbehauen (1997). Study of Predictive Controller Tuning Methods. *Automatica*, Vol. 33, No. 12, pp. 2243-2248.

Rankovic, V., J. Radulovic, N. Grujovic and D. Divac (2012). Neural Network Model Predictive Control of Nonlinear Systems Using Genetic Algorithms. *INT J COMPUT COMMUN*, ISSN 1841-9836, Vol.7, No. 3, pp. 540-549.

Rawlings, J. B. (2000). Tutorial Overview of Model Predictive Control. *IEEE Contr. Syst. Magazine*, Vol. 20, No. 3, pp. 38.

Reid, D., and M. Muyeba (2008). Fuzzification of Spiked Neural Networks, Computer Modeling and Simulation. *EMS'08 Second UKSIM European Symposium on IEEE*.

Ren, Q., L. Baron and M. Balazinski (2006). Type-2 Takagi-Sugeno-Kang Fuzzy Logic Modeling using Subtractive Clustering. *Fuzzy Information Processing Society*, pp. 120-125. *NAFIPS 2006.Annual meeting of the*

North American.

Rodriguez, F. O., Wen Yu, M. A. Moreno-Armendariz (2006). System Identification Using Hierarchical Fuzzy CMAC Neural Networks, *D.-S. Huang, K. Li, and G.W. Irwin (Eds.): ICIC 2006*, LNAI 4114, pp. 230–235, *Springer-Verlag Berlin Heidelberg*.

Rutkowski, L. and K. Cpalka (2003). Flexible Neuro-Fuzzy Systems. *IEEE Trans on Neural Networks*, vol. 14, No. 3.

Samek, D. and P. Dostal (2009). Artificial Neural Network with Radial Basis Function in Model Predictive Control of Chemical Reactor. *Mechanics*, Vol. 28, No. 3, 2009.

Sanchez, E. G., M.J. A. Bravo, J.M. C. Izquierdo, Y.A. Dimitriadis, J. L. Coronado and M.J. L. Nieto (1999). Control of the penicillin production using fuzzy neural networks. *Proceedings of the World Congress on System, Man and Cybernetics, SMC'99, Tokyo, Japan*, pp. 6446-6450.

Sarimveis, H. and G. Bafas (2003). Fuzzy model predictive control of non-linear processes using genetic algorithms. *Fuzzy Sets and Systems*, Vol. 139, pp. 59–80.

Scattolini, R. (2009). Architectures for distributed and hierarchical model predictive control – a review. *Journal of Process Control*, Vol. 19, pp. 723–731.

Schliebs, S. and Nikola Kasabov (2013). Evolving spiking neural network - a survey. *Evolving Systems*, pp. 1-12.

Sendrescu, D. and E. Petre (2011). Neural Network Predictive Control of a Bioprocess. *2nd International Conference on Networking and Information Technology IPCSIT*, Vol.17.

Setnes, M. and R. Babuska (2001). Rule Base Reduction: Some Comments on the Use of Orthogonal Transforms. *IEEE Transactions on Systems, Man, And Cybernetics—Part C: Applications and Reviews*, Vol. 31, No. 2.

Seyab, R., Y. Cao (2008). Differential Recurrent Neural Network based Predictive Control. *Computers and Chemical Engineering*, Vol. 32, Issue 7, pp. 1533-1545.

Shridhar, R. and D. J. Cooper (1997). A Tuning Strategy for Unconstrained SISO Model Predictive Control. *Ind. Eng. Chem. Res.*, Vol. 36, pp. 729-746.

Shridhar, R. and D. J. Cooper (1998). A Tuning Strategy for Unconstrained Multivariable Model Predictive Control. *Ind. Eng. Chem. Res.*, Vol. 37, pp. 4003-4016.

Su, C. and S. Wang (2006). Robust model predictive control for discrete uncertain nonlinear systems with time-delay via fuzzy model. *Journal of Zhejiang University SCIENCE A*, Vol. 7, No 10, pp.1723-1732.

Su, M., C. Chou, E. Lai and J. Lee (2006). A new approach to fuzzy classifier systems and its application in self-generating neuro-fuzzy systems. *Neurocomputing*, Vol. 69, pp. 586–614.

Su, Zh., P. Wang and Y. Zhang (2012). Automatic T-S fuzzy model with application to designing predictive controller. *Comput. Sci. Inf. Syst.*, Vol. 9, No. 4, pp. 1577-1601.

Sulzberger, S. M, N. N. Tschicholg-Gurman, S. J. Vestli (1993). FUN: Optimization of Fuzzy Rule Based Systems Using Neural Networks. In *Proceedings of IEEE Conference on Neural Networks, San Francisco*, pp. 312-316.

Tan, T.Z., C. Quek and G.S. Ng (2005). FALCON-AART: An Improved Complementary Learning Fuzzy Neural Network. *Proceedings of the 12th International Conference on Neural Information Processing*, pp. 628-633.

Tang, Zh. and X. Yan (2007) Fuzzy CMAC Model Predictive Control. *Fourth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD 2007)*, Vol.2, pp.566-569.

Taniguchi, T. and K. Tanaka (2000). Rule Reduction and Robust Control of Generalized Takagi-Sugeno Fuzzy Systems, *JACIII*, Vol. 4, Issue 5, pp. 373-379. Tenny, M. J., S. J. Wright and J. B. Rawlings (2002). Nonlinear Model Predictive Control via Feasibility-Perturbed Sequential Quadratic Programming. *Optimization Technical* Report 02-06, *Computer Sciences Dept, UNIV. of Wisconsin Texas-Wisconsin Modeling and Control Consortium Report TWMCC-2002-02*.

Tenny, M. (2002). Computational Strategies for Nonlinear Model Predictive

Control. *PhD Thesis, University of Wisconsin-Madison*.

Todorov, Y. and M. Petrov (2010). Model Predictive Control of a Lyophilization plant: A simplified approach using Wiener and Hammerstein systems. *International Journal of Control and Intelligent Systems, ACTA Press*, Vol. 39, 201-2183.

Tuan, T. Q. and P. X. Minh (2012). Adaptive Fuzzy Model Predictive Control for Non-minimum Phase and Uncertain Dynamical Nonlinear Systems. *Journal of computers*, Vol. 7, No. 4.

Tung, W. L. and C. Quek (2002). GenSoFNN: A Generic Self-Organizing Fuzzy Neural Network. *IEEE Trans on Neural Networks*, Vol. 13, No. 5.

Valencia-Palomo, G. and J. A. Rossiter (2009). Auto-tuned predictive control based on minimal plant information. *7th IFAC International Symposium on Advanced Control of Chemical Processes*.

Van Rullen, R., R. Guyonneau and S. J. Thorpe (2005). Spike times make sense. *Trends in neurosciences*, Vol. 28, No. 1, pp. 1-4.

Vasičkaninová, A. and M. Bakošová (2009). Neural Network Predictive Control of a Chemical Reactor. *Acta Chimica Slovaca*, Vol.2, No.2, pp. 22–36.

Vasičkaninová, A., M. Bakošová, A. Mészáros and J. J Klemeš (2011). Neural network predictive control of a heat exchanger. *Applied Thermal Engineering*, Vol. 31, No. 13.

Venkateswarlu, Ch. and K. V. Rao (2005). Dynamic recurrent radial basis function network model predictive control of unstable nonlinear processes. *Chemical Engineering Science*, Vol. 60, pp. 6718–6732.

Vieira, J., F. M. Dias and A. Mota (2004). Neuro-fuzzy systems: a survey. *5th WSEAS NNA International Conference on Neural Networks and Applications, Udine, Italia*.

Waller, J. B. and H. T. Toivonen (2002). A Neuro-Fuzzy Model Predictive Controller Applied to a pH-neutralization Process. *Proceedings of IFAC World Congress on Automatic Control, Barcelona, Spain*.

Wang, D., G. S. Ng, and C. Quek (2008). A novel hybrid intelligent system: Genetic algorithm and rough set incorporated neural fuzzy inference system. *Evolutionary Computation, IEEE World Congress on Computational Intelligence*.

Wang, L. and J. Hu,(2011). Fuzzy Predictive R2R Control to CMP Process. *IEEE International Conference on Computer Science and Automation Engineering (CSAE)*.

Wang, S. W., D.L. Yu, J.B. Gomm, G.F. Page and S.S. Douglas (2006). Adaptive neural network model based predictive control for air–fuel ratio of SI engines. *Engineering Applications of Artificial Intelligence*, Vol. 19, pp. 189–200.

Wang, X., Y. Cheng and W. Sun (2006). Multi-step predictive control with TDBP method for pneumatic position servo system. *Transactions of the Institute of Measurement and Control*, Vol. 28, Issue 1, pp. 53-68.

Wei, C. and L.X. Wang (2000). A Note on Universal Approximation by Hierarchical Fuzzy Systems. *Information Sciences*, Vol. 123, pp. 241-248.

Wen, Y. and A. Marco (2005). Moreno-Armendariz, System identification using hierarchical fuzzy neural networks with stable learning algorithms. *Proceedings of the 44th IEEE Conference on Decision and Control, and the European Control Conference, Seville, Spain*.

Xia, X., et al. (2005). Nonlinear predictive control based on wavelet neural network applied to polypropylene process. *Advances in Neural Networks, Springer Berlin Heidelberg*, pp. 131-136.

Yamakawa, T., E. Uchino, T. Miki, and H. Kusanagi (1992). A neo fuzzy neuron and its applications to system identification and prediction of the system behavior. *In Proc. 2-nd Int. Conf. on Fuzzy Logic and Neural Networks - IIZUKA-92, Japan*, pages 477–483.

Yen, J. and L. Wang (1999). Simplifying Fuzzy Rule-Based Models Using

Yoo, S. J., J. B. Park and Y. H. Choi (2005). Stable predictive control of chaotic systems using self-recurrent wavelet neural network. *International journal of control, automation, and systems*, Vol. 3, No 1, pp. 43-55.

Yordanova, S., R. Petrova and V. Mladenov (2006). Sugeno Predictive Neuro-Fuzzy Controller for Improving Dynamic Performance of Control Systems of Nonlinear Plants under Uncertainties. *Proceedings of the 10th WSEAS International Conference on Systems, Vouliagmeni, Athens, Greece*, pp. 190-197.

Yousef, A. M. (2012). Neural Network Predictive Control Based Power System

Stabilizer. *Research Journal of Applied Sciences, Engineering and Technology*, Vol. 4, No. 8, pp. 995-1003.

Yu, W., M. A. Moreno-Armendariz and F. O. Rodriguez (2007). System Identification using hierarchical fuzzy neural networks with stable learning algorithm. *Journal of Intelligent & Fuzzy Systems*, Vol. 18, pp. 171-183.

Zadeh, L.A. (1975). The concept of a linguistic variable and its applications to approximate reasoning-1. *Information Sciences*, Vol. 8, pp. 199–249.

Zamarreño, J. M., P. Vega (1999). Neural predictive control. Application to a highly non-linear system. *Engineering Applications of Artifcial Intelligence*, Vol. 12, pp. 149 - 158.

Zekri, M., S. Sadri and F. Sheikholeslam (2008). Adaptive fuzzy wavelet network control design for nonlinear systems. *Fuzzy Sets and Systems*, Vol. 159, No. 20, pp. 2668-2695.

Zeng, X., J. A. Keane, J. Y. Goulermas and P. Liatsis (2005). Approximation Capabilities of Hierarchical Neural-Fuzzy Systems for Function Approximation on Discrete Spaces. *International Journal of Computational Intelligence Research*, Vol.1, No.1, pp. 29-41.

Zhang, J. and A.J. Morris (2000). Long Range Predictive Control of Nonlinear Processes Based on Recurrent Neuro-Fuzzy Network Models. *Springer London*, *Neural Computing & Applications*, Vol. 9, No. 1.

Zhang, Y. M. and R. Kovacevic (1998). Neurofuzzy Model-Based Predictive Control of Weld Fusion Zone Geometry. *IEEE Trans on Fuzzy Systems*, Vol. 6, No. 3.

Zheng, A. (1997). A computationally Efficient Nonlinear MPC Algorithm. In *Proceedings of the American Control Conference*.

Zhou, X. et al. (2013). Intuitionistic Fuzzy Neural Networks based on Extended Kalman Filter Training algorithm. *International Workshop on Cloud Computing and Information Security (CCIS 2013)*.