KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
KOMPIUTERIŲ KATEDRA

Vaidas Jukavičius

Triukšmo slopinimo sistema

Magistro darbas

Darbo vadovas:
prof. dr. E. Kazanavičius

Kaunas, 2008

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
KOMPIUTERIŲ KATEDRA

Vaidas Jukavičius

Triukšmo slopinimo sistema

Magistro darbas

Recenzentas
doc. dr. P. Kanapeckas
2008-05-26

Vadovas
prof. dr. E. Kazanavičius
2008-05-26

Atliko
IFM–2/1 gr. stud.
Vaidas Jukavičius
2008-05-26

Kaunas, 2008

Santrauka

Šiame darbe sudaroma triukšmo slopinimo sistema (triukšmą slopinančios ausinės), kurios funkcionalumas paremtas aktyviais triukšmo slopinimo principais. Pagrindinis darbo tikslas sudaryti triukšmo slopinimo filtrą, kurio veikimas paremtas miglotojų išvadų sistemų ir dirbtinių neuroninių tinklų veikimo principais.

Darbe atliekama rekurentinių miglotojų neuroninių filtrų analizė apžvelgiant pagrindinius miglotojų išvadų sistemų ir dirbtinių neuroninių tinklų modelius. Triukšmo slopinimo filtru pasirenkamas dinaminis miglotasis neuroninis tinklas.

Pasiūloma ir įvertinama pasirinkto triukšmo slopinimo filtro struktūrinio identifikavimo metodika, leidžianti automatiškai nustatyti modeliuojamo filtro struktūrą turint sistemos įėjimo ir išėjimo duomenis.

Dinaminis miglotasis neuroninis filtras realizuojamas MatLab programinio paketo pagalba. Atliekamas eksperimentinis tyrimas su realiai įrašytais signalais, leidžiantis įvertinti gaunamus rezultatus sistemai funkcionuojant realioje aplinkoje.

Summary

Presented work covers a noise reduction system development (noise cancellation headphones). The functionality of the system is based on active noise cancellation. Main objective is to develop noise reduction filter, which is based on fuzzy inference systems and artificial neural networks functionality.

The analysis part of presented work goes through fuzzy inference systems and artificial neural networks analysis, recurrent fuzzy neural filter analysis and selection of most suitable recurrent fuzzy neural filter according to analysis results. The dynamic fuzzy neural network is selected as noise reduction filter.

The structural identification technique of selected noise reduction filter is developed. The presented technique represents a systematic approach to generating fuzzy rules from a given input-output data set.

The dynamic fuzzy neural network was implemented and simulated using MatLab program package. The experimental research was made using real measurements from laboratory of acoustics to evaluate the results when system is functioning in real environment.

Turinys

1 Įvadas 7
2 Rekurentinių miglotojų neuroninių filtrų analizė 9
2.1 Triukšmo slopinimo problema 9
2.2 Miglotojų išvadų sistemų modeliai 10
2.3 Rekurentinių miglotojų neuroninių filtrų modeliai 12
2.4 Dinaminis miglotasis neuroninis tinklas 13
2.4.1 TSK modelio apibendrinta forma 13
2.4.2 Dinaminio miglotojo neuroninio tinklo modelio aprašymas 14
2.4.3 Dinaminis miglotasis neuroninis apriboto optimizavimo apmokymo metodas 16
3 Triukšmo slopinimo sistemos modelis 20
3.1 Dinaminio miglotojo neuroninio tinklo modelio identifikavimas 20
3.1.1 Duomenų grupavimo metodas 20
3.1.2 Grupių centrų optimizavimas miglotuoju c-vidurkių metodu 22
3.1.3 Dinaminio miglotojo neuroninio tinklo prielaidų dalies identifikavimas 23
3.1.4 Dinaminio miglotojo neuroninio tinklo išvadų dalies identifikavimas 23
3.2 Triukšmo slopinimo sistemos struktūrinis modelio aprašymas 24
3.3 Triukšmo slopinimo sistemos modelio elgsenos aprašymas 26
3.4 Dinaminio miglotojo neuroninio tinklo identifikavimo metodikos įvertinimas 29
4 Eksperimentinis tyrimas 33
4.1 Eksperimentinio tyrimo aplinka 33
4.2 Eksperimentinio tyrimo atlikimo metodika 34
4.3 Eksperimentinio tyrimo rezultatų įvertinimas 34
4.4 Rekomendacijos 37
5 Išvados 38
6 Literatūra 40
7 Priedai 42
7.1 Priedas Nr. 1 DFNN struktūros identifikavimo metodikos programinė realizacija. 42
7.2 Priedas Nr. 2 Duomenų grupavimo metodo programinė realizacija 44
7.3 Priedas Nr. 3 C-vidurkių optimizavimo metodo programinė realizacija 45
7.4 Priedas Nr. 4 DFNN filtro programinė realizacija 46
7.5 Priedas Nr. 5 Apmokymo metodo programinė realizacija 49

Paveikslėliai

1 pav. Triukšmo slopinimo sistemos struktūra. 10
2 pav. Miglotosios išvadų sistemos struktūra. 11
3 pav. DFNN modelio architektūra. 14
4 pav. Rekurentinio neuroninio tinklo struktūra 15
5 pav. Apibendrintas Frasconi-Gori-Soda (G-FGS) neuronas su vienu įėjimu. 15
6 pav. Triukšmo slopinimo sistemos struktūrinė schema. 24
7 pav. DFNN struktūrinė schema 25
8 pav. Grupių centrų optimizavimo metodo blokinė schema. 26
9 pav. Duomenų grupavimo metodo blokinė schema. 27
10 pav. Apsimokymo metodo blokinė schema. 28
11 pav. Modelio įėjimo duomenys. 29
12 pav. Modelio išėjimo duomenys. 29
13 pav. Duomenų grupių centrų lokalizacija po duomenų grupavimo. 30
14 pav. Gaunamos narystės funkcijos po duomenų grupavimo. 30
15 pav. Duomenų grupių centrų lokalizacija po optimizavimo procedūros. 31
16 pav. Gaunamos narystės funkcijos po optimizavimo procedūros. 31
17 pav. Įrašančios įrangos išdėstymas. 33
18 pav. Akustinė atspindžio problema. 34
19 pav. Narystės funkcijos. 35
20 pav. Apmokymo įėjimo duomenys. 35
21 pav. Tikrieji ir apmokymo metu gautieji išėjimo duomenys. 36
22 pav. Tikrieji ir filtravimo metu gautieji išėjimo duomenys. 36
23 pav. Tikrieji ir apmokymo metu gautieji išėjimo duomenys. 37

1 Įvadas

Vienas iš penkių svarbiausių žmogaus jutimų yra klausa. Garsas, kuris sukelia žmogui nekomfortabilumo jausmą ar yra tiesiog erzinantis, identifikuojamas kaip triukšmas. Nuolatinis triukšmas žmogui trukdo jaustis komfortabiliai ar netgi gali smarkiai įtakoti jo sveikatą. Ši problema tampa vis aktualesnė mūsų gyvenime: vis didesnis triukšmas supantis mus gatvėse, darbe, lėktuve ir t.t.

Minėtą problemą galime išspręsti naudodami triukšmo slopinimo ausines. Egzistuoja du triukšmo slopinimo metodai: pasyvus ir aktyvus. Pasyvaus triukšmo slopinimo idėja remiasi specialių garso izoliacinių medžiagų naudojimu. Minėtosios medžiagos pasižymi gera garso izoliacija aukštų dažnių diapazone, kita vertus žemo dažnio diapazone garso izoliavimas tampa labai brangus ir nepraktiškas. Aktyvus triukšmo slopinimas paremtas garso bangų interferencija, t.y. dvi vienodos, bet skirtingų fazių garso bangos susitikusios erdvėje viena kitą neutralizuoja – bangų energija virsta šiluma. Priešingų fazių garso bangoms generuoti naudojamas skaitmeninis signalų apdorojimas ir specialūs adaptyvūs filtrai.

Praktikoje šie abu metodai naudojami kartu. Triukšmo slopinimo ausinių gaubteliai dažniausiai gaminami iš medžiagų gebančių gerai absorbuoti aukštus dažnius, o žemi dažniai ausinių viduje slopinami aktyviu būdu.

Pagrindinė aktyvaus slopinimo problema yra ta, kad triukšmas perėjęs per ausinių gaubtelius (nežinomą sistemą) pasikeičia ir bendru atveju negalima tiksliai nustatyti, kaip jis keičiasi, nes sistema nuolatos kinta. Be to realiose situacijose sistemos aplinka yra ant tiek sudėtinga, kad sistema iš tikrųjų charakterizuojama netiesine funkcija. Sistemos identifikavimui per eilę metų buvo pasiūlyta gana daug ir įvairių adaptavimosi algoritmų tiesiniams filtrams, deja netiesiškose dinaminėse sistemose šie filtrai nepasižymėjo geromis sistemos identifikavimo savybėm.

Pastaraisiais metais dėmesys skiriamas rekurentiniams miglotiesiems neuroniniams filtrams, kurie pasižymi geromis sistemų aproksimavimo, dinaminėmis ir prognozavimo savybėmis netiesiškose dinaminėse sistemose.

Šio darbo tikslas sudaryti triukšmo slopinimo filtrą realizuojant tokius uždavinius:

- Išanalizuoti rekurentinių miglotojų neuroninių filtrų veikimo principus;
- Pasirinkti ir išanalizuoti rekurentinį miglotąjį neuroninį filtrą tinkamą triukšmo slopinimo problemai spręsti;
- Sudaryti metodiką pasirinkto rekurentinio miglotojo neuroninio filtro architektūriniam identifikavimui;
- Atlikti rekurentinio miglotojo filtro tyrimą ir įvertinti gautus rezultatus;

Analizės metu atliekama rekurentinių miglotojų neuroninių filtrų analizė, kurios metu pateikiami argumentai, dėl ko pastarieji tinklai tinka triukšmo slopinimo sistemai spręsti, analizuojami miglotojų sistemų ir dirbtinių neuroninių tinklų modeliai, parenkamas ir išanalizuojamas tam tikras rekurentinis miglotasis filtras. Vėliau sudaroma pasirinkto rekurentinio miglotojo filtro struktūros identifikavimo metodika ir aprašomas triukšmo slopinimo sistemos modelis. Pagal sudarytą filtro struktūros identifikavimo metodiką nustatoma filtro struktūra ir atliekamas eksperimentinis tyrimas, įvertinami gautieji rezultatai ir pateikiamos rekomendacijos triukšmo slopinimo sistemos vystymui.

2 Rekurentinių miglotojų neuroninių filtrų analizė

Per eilę metų triukšmo slopinimo problema tapo dar aktualesnė, o adaptyvus filtravimas paplito daugumoje sričių, tokių kaip sistemų valdymo, vaizdų apdorojimo ir komunikacijų. Labiausiai paplitę adaptyvūs tiesiniai filtrai [14] buvo pritaikyti ir sėkmingai naudojami daugelyje sričių, tačiau kuomet buvo taikomi netiesinėms sistemoms [15], nepasižymėjo pakankamai geromis filtravimo savybėmis. Dėl šios priežasties adaptyviam filtravimui pradėta taikyti dirbtinių neuroninių tinklų ir miglotojų išvadų sistemų modeliai.

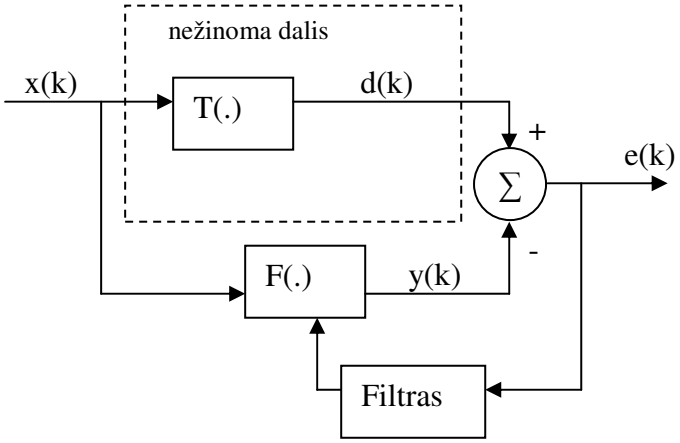
Statiniai dirbtinių neuroninių tinklų modeliai pradėti plačiai taikyti sistemų procesų modeliavime ir valdyme [12, 13]. Statinių neuroninių tinklų išėjimas priklauso tik nuo įėjimo, t.y. neturi grįžtamojo ryšio, todėl tinklas modeliuoja sistemas neįtraukdamas dinaminių sistemos savybių. Tačiau norint identifikuoti dinamines sistemas reiktų naudoti rekurentinius neuroninius tinklus su grįžtamuoju ryšiu, kurio pagalba į neuroninio tinklo modelį įtraukiamos dinaminės savybės. Rekurentinių neuroninių tinklų pranašumas dinaminėse sistemose prieš statinius neuroninius tinklus pateiktas [16].

Miglotojų išvadų sistemų modelių panaudojimas filtravime, skirtingai nuo neuroniniais tinklais paremtų modelių, leidžia išskaidyti įėjimo duomenis į grupes ir atskirai modeliuoti modelio elgseną kiekvienos grupės atžvilgiu. Modelio elgsena kiekvienoje grupėje aprašoma lokaliu, paprastesnę struktūrą turinčiu sub-modeliu, kuris gali būti aprašomas, priklausomai nuo išvadų sistemos tipo, neuroninio tinklo modeliu. Tokia kombinacija turi miglotosios teorijos ir neuroninių tinklų privalumus, todėl analizės objektu pasirinktas rekurentinis miglotasis neuroninis filtras.

Toliau šiame skyriuje supažindinama su triukšmo slopinimo problema ir analizuojami rekurentinių miglotojų neuroninių filtrų architektūriniai sprendimai, bei veikimo principai.

2.1 Triukšmo slopinimo problema

Tipinė triukšmo slopinimo sistemos struktūra [11] pavaizduota 1 paveikslėlyje. Triukšmas *x(k)*, perėjęs per triukšmo slopinimo sistemos ausines, pasikeičia dėl pasyvaus triukšmo slopinimo ir taip gaunamas triukšmo signalas *d(k)*. Ausinės charakterizuojamos nuolatos kintančia netiesine perdavimo funkcija *T(.)*. Ausinių viduje signalas *d(k)* susiduria su jam priešingos fazės signalu *y(k)* ir pastarasis efektyviai nuslopina signalą *d(k)*. Nuslopinimo efektyvumas matuojamas signalu *e(k)*.



1 pav. Triukšmo slopinimo sistemos struktūra.

Jeigu laikysime *F(.)* taip pat sistemos perdavimo funkcija, tuomet idealiu atveju:

$$y(k) = F(x(k)) = d(k) = T(x(k)) \tag{1}$$

ir

$$e(k) = d(k) - y(k) \to 0 \tag{2}$$

Iš (2) gauname klaidos signalo *e(k)* įvertį:

$$e^2(k) = (d(k) - y(k))^2 \to E\{e^2(k)\} = E\{(d(k) - y(k))^2\} \tag{3}$$

Taikant optimizavimo metodus adaptyvių filtrų parametrai turi būti randami tokie, kad būtų minimizuojama klaidos signalo galia $E\{e^2(k)\}$, kitaip tariant turi būti randama kuo panašesnė perdavimo funkcija *F(.)* į *T(.)*. Iš pastarojo teiginio visa problema susiveda į sistemos perdavimo funkcijos radimą.

2.2 Miglotojų išvadų sistemų modeliai

Praktiškai miglotoji teorija paremta žmogaus smegenų informacijos interpretavimo koncepcija. Žmogaus smegenys informaciją gaunamą iš jutimo organų interpretuoja netiksliai ir nevisiškai pilnai. Klasikinėje matematikoje matematinė skaičių aibė apibrėžiama konkrečiomis nelygybių ar funkcijų apribojimais. Tokiu atveju skaičius gali priklausyti aibei tik jei jis tenkina griežtus apribojimus. Tokios skaičių aibės iš esmės negali būti naudojamos aprašyti biologinės smegenų veiklos. Todėl, kaip priešingybė klasikinei skaičių aibei, įtraukiama miglotosios aibės sąvoka, kuri neturi griežtų apribojimų.

Jeigu *X* yra objektų (skaičių) rinkinys aprašomas konkrečiais elementais *x*, tuomet miglotoji aibė *A* rinkinyje *X* aprašoma kaip aibė porų [2]:

$$A = \{(x, \mu_A(x)) \mid x \in X\} \tag{4}$$

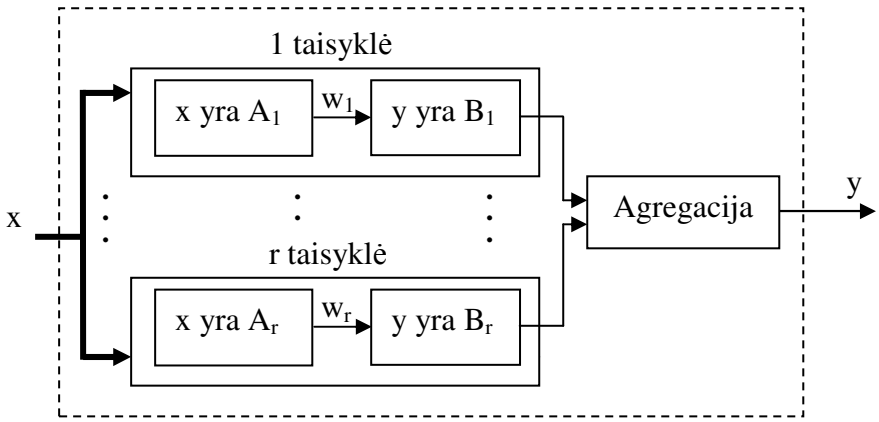
čia $\mu_A(x)$ yra narystės funkcija (NF) miglotajai aibei *A*. NF priskiria kiekvienam *X* elementui narystės laipsnį. Narystės laipsnis nusako, kaip smarkiai elementas priklauso (nepriklauso) miglotajai aibei. Naudojant įvairias NF [2] galima aprašyti kur kas racionaliau lingvistines išraiškas, tokias pavyzdžiui, kaip "senas žmogus".

Panaudojant miglotąsias aibes formuojamos jeigu-tai taisyklės aprašomos tokia forma [2]:

$$jeigu\ x\ yra\ A\ \ tuomet\ y\ yra\ B \tag{5}$$

čia *A* ir *B* yra miglotojų aibių reikšmės objektų rinkiniams *X* ir *Y*. Dažniausiai „*x yra A*" dalis vadinama prielaida, o „*y yra B*" dalis išvada. Tokių taisyklių rinkiniais galima aprašyti netgi ir labai sudėtingus procesus. Procesas kuomet iš taisyklių rinkinio ir žinomų faktų daroma išvada vadinama miglotuoju mąstymu.

Miglotoji išvadų sistema paremta anksčiau minėta miglotojų aibių, miglotojų jeigu-tai taisyklių ir miglotojo mąstymo teorija. Sistema susideda iš trijų pagrindinių komponentų: taisyklių rinkinio, duomenų bazės, kurioje saugomos narystės funkcijos kiekvienai taisyklei, ir išvadų darymo mechanizmo, kuris realizuoja išvadų darymo procedūrą. Tipinė tokios sistemos struktūra pavaizduota 2 paveikslėlyje [2].



2 pav. Miglotosios išvadų sistemos struktūra.

Miglotojų išvadų sistemų yra keletas tipų [2], kurios dažniausiai skiriasi išvadų ir miglotumo naikinimo dalimis. Plačiausiai paplitę miglotojų išvadų sistemų modeliai yra Mamdani ir Sugeno.

Mamdani modelis pasiūlytas E. H. Mamdani [17] kaip modelis geriausiai tinkantis sistemų valdymui ir valdiklių modeliavimui. Šio tipo sistemos išvada yra miglota, kurios miglotas aibes daro iš įėjimo miglotų aibių agregavimo operatorius. Pastaroji išvada yra agregavimo būdu gauta miglotoji aibė, pagal kurią naudojant miglotumo naikinimo metodus, gaunama tiksli išėjimo reikšmė. Dažniausiai naudojami miglotumo naikinimo metodai yra

arba svorio centro arba maksimumo vidurio metodai [2]. Mamdani tipo sistema yra labiau intuityvi, geriau tinka žmogaus teikiamiems duomenims apdoroti, plačiai taikoma sistemų valdyme, tačiau yra skaičiavimų ir optimizavimo prasme neefektyvi.

Sugeno miglotasis modelis [18, 19], kuris dar žinomas TSK modelio pavadinimu, buvo pasiūlytas Takagi, Sugeno ir Kang tam, kad turint įėjimo-išėjimo sistemos duomenis, būtų galima sistematiškai generuoti miglotąsias taisykles, kurios gebėtų atkartoti sistemos veikimą. Tipinė miglotoji taisyklė TSK modelyje aprašoma tokia forma:

$$\textit{jeigu } x \textit{ yra } A \textit{ ir } y \textit{ yra } B \textit{ tuomet } z=f(x, y), \tag{6}$$

čia A ir B yra miglotieji rinkiniai, o z yra išvadų funkcija. Sugeno tipo sistemos išvada taip pat yra miglota, kurioje kiekvienos taisyklės išvada yra visų įėjimų darinys, o išėjimas yra visų taisyklių išvadų tiesinis darinys, dažniausiai apskaičiuojamas svorinių vidurkių metodo pagalba.

Pagrindinis skirtumas tarp Sugeno ir Mamdani modelių yra tas, kad Sugeno modelio išvadų dalis pakeista išvadų funkcija, ko pasėkoje sistema tampa kompaktiškesnė ir skaičiavimų, bei optimizavimo prasme efektyvesnė už Mamdani sistemos modelį. Todėl Sugeno modelis gerai tinka adaptyviems modeliams sudaryti ir netiesinėms sistemoms modeliuoti [18, 19].

2.3 Rekurentinių miglotojų neuroninių filtrų modeliai

Rekurentinių miglotojų neuroninių filtrų (RMNF) veikimas kaip jau buvo minėta paremtas miglotojų išvadų sistemų ir neuroninių tinklų veikimu. Pagal 2.2 skyrių RMNF modeliai paremti dažniausiai TSK išvadų sistemos modeliu, tačiau gali būti klasifikuojami pagal naudojamų neuroninių tinklų rekurencijos tipą: globalų ir lokalų.

Globalus rekurencijos tipas apibrėžia klasę tų neuroninių tinklų modelių, kurių modelių išėjimai yra su vėlinimais paduodami atgal į modelio įėjimą. Tokio tipo modeliams priklauso Williams and Zipser [21] pasiūlytas globaliai rekurentinis neuroninis tinklas ar laiko-vėlinimo neuroninis tinklas [20]. Tačiau tokio tipo tinklai turi problemų apsimokymo procese, nes nėra pakankamai stabili ir greita apsimokymo procedūra.

Lokalus rekurencijos tipas apibrėžia klasę tų neuroninių tinklų modelių, kurių rekurentiniai ryšiai yra realizuoti tinklo struktūros viduje, dažniausiai naudojant rekurentinius neuronus [22, 4, 5]. Šie modeliai gali įtraukti sistemų dinamines savybes naudojant savo vidines būsenas [4], tokiu būdu išvengiant rekurentinių ryšių išvadų sistemų modelių įėjimuose ir taip supaprastinant RMNF architektūrą. Pagal [4, 5] dėmesys turėtų būti skiriamas Frasconi-Gori-Soda (FGS) dinaminiams (rekurentiniams) neuronams, kuomet FGS

neurono išėjimas yra vėlinamas ir paduodamas atgal į neurono įėjimą. Pagal [5] straipsnyje nagrinėtų dinaminių neuronų modelių charakteristikas FGS neurono modelis yra pranašesnis prieš kitus dinaminių neuronų modelius. Taipogi [23] straipsnyje parodyta, kad naudojant lokalios rekurencijos neuroninį tinklą su FGS neuronais, sėkmingai galima aproksimuoti netiesinės sistemos dinamines savybes.

Apibendrinus 2.2 ir šiame skyriuje išdėstytus faktus ir išvadas, bei remiantis [1, 3] straipsniais galima padaryti išvadą, kad vienas tinkamiausių RMNF triukšmo slopinimo problemai spręsti yra dinaminis miglotasis neuroninis tinklas [3], kuris paremtas TSK miglotojų išvadų sistemos modeliu ir lokaliai rekurentiniais neuroniniais tinklais su FGS neuronais, nes pasižymi geromis netiesinių sistemų dinamikos aproksimavimo savybėmis.

2.4 Dinaminis miglotasis neuroninis tinklas

Dinaminis miglotasis neuroninis tinklas (DFNN), kuris paprastai priklauso rekurentinių miglotojų neuroninių filtrų klasei, paremtas TSK modelio architektūra ir rekurentinėmis TSK taisyklėmis. Apmokymas atliekamas naudojant dinaminį miglotąjį neuroninį apriboto optimizavimo metodą (D-FUNCOM).

2.4.1 TSK modelio apibendrinta forma

TSK modelis sudaromas iš miglotojų taisyklių kurios apibendrinta forma gali būti užrašytos taip [3]:

$$R^{(l)}: \text{Jeigu } u_1(k) \text{ yra } A_1^l \text{ ir } u_2(k) \text{ yra } A_2^l \text{ ir .... ir } u_m(k) \text{ yra } A_m^l$$
$$\text{tuomet } \hat{y}_l(k) = g_l(u(k)), \quad l=1, ..., r \tag{7}$$

čia $R^{(l)}$ yra l-oji miglotoji taisyklė, r taisyklių skaičius ir $u(k)$ įėjimo vektorius laiku k. $g_l(u(k))$ yra funkcija aprašanti $R^{(l)}$ taisyklės išvadų dalį, pagal kurią paskaičiuojamas ir taisyklės išėjimas $\hat{y}_l(k)$ laiku k vektoriui $u(k)$. A_i^l yra miglotoji aibė aprašoma Gauso [3] narystės funkcija:

$$\mu_{A_i^l}(u_i) = \exp\left\{-\frac{1}{2}\frac{(u_i - m_i^l)^2}{(\sigma_i^l)^2}\right\} \tag{8}$$

čia m_i^l ir σ_i^l yra atitinkamai vidurkis ir standartinis nuokrypis. Nustačius skaičių ir parametrus šių funkcijų kiekvienai taisyklei prielaidų dalis yra suskaidoma ir rinkinys aibių $A^{(l)} = (A_1^l, ..., A_m^l)$ suformuoja multi-dimensinę miglotąją aibę, kurios narystės funkcija gali būti aprašoma taip:

$$\mu_{A^{(l)}}(u(k)) = \mu_l(k) = \prod_{i=1}^{m} \mu_{A_i^l}(u_i(k)) \tag{9}$$

Kitaip tariant (9) išraiška aprašo kaip stipriai konkretus vektorius $u(k)$ priklauso multi-dimensiniai miglotajai aibei $A^{(l)}$ arba kiek svarbi konkrečios taisyklės išvada laiku *k* visos sistemos atžvilgiu. TSK modelio išėjimas laiku *k* apskaičiuojamas naudojant svorinį vidurkio metodą:

$$\hat{y}(k) = \frac{\sum_{l=1}^{r} \mu_l(k) \cdot \hat{y}_l(k)}{\sum_{l=1}^{r} \mu_l(k)} \tag{10}$$

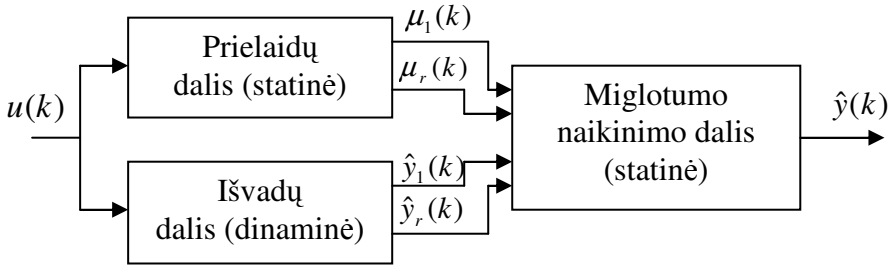
TSK modelis yra svarbus tuo, kad lengvai galima optimizuoti išvadų dalies funkcijas $g_l(u(k))$, taip, kad kuo tiksliau būtų galima atkartoti sistemos veikimą.

2.4.2 Dinaminio miglotojo neuroninio tinklo modelio aprašymas

Dinaminio miglotojo neuroninio tinklo modelis susideda iš tokių TSK miglotojų taisyklių [3]:

$$R^{(l)}: \text{Jeigu } u(k) \text{ yra } A^l \text{ tuomet } \hat{y}_l(k) = g_l(k) = RNN_l(u(k)), \quad l=1, ..., r$$

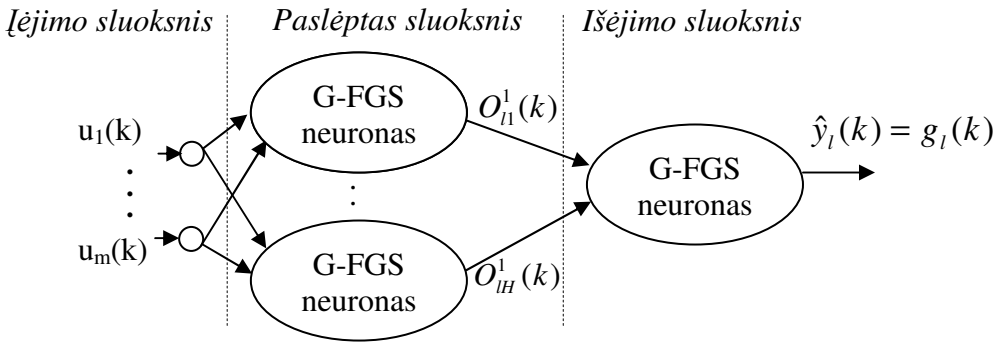
čia A^l yra multi-dimensinė miglotoji aibė prielaidų dalyje, o RNN_l yra rekurentinis neuroninis tinklas, kuris išvadų dalyje realizuoja *l*-tąją taisyklę $R^{(l)}$. Kadangi modelis paremtas TSK modelio architektūra [2], tai DFNN modelį galima suskaidyti į tokias tris pagrindines dalis: prielaidų, išvadų ir miglotumo naikinimo (3 paveikslėlis).



3 pav. DFNN modelio architektūra.

Prielaidų dalis realizuoja (9) išraišką ir paskaičiuoja konkrečių taisyklių svarbumus $\mu_l(k)$ ir yra suskaidoma arba eksperto, kurio sprendimai paremti pirmine informacija apie sistemą, arba naudojant duomenų grupavimo metodą [7]. Miglotumo naikinimo dalis realizuoja (10) išraišką kiekvienu laiko momentu *k* paskaičiuodama bendrą išėjimą $\hat{y}(k)$. Išvadų dalis sudaryta iš *r* taisyklių sub-modelių (RNN_l). Kiekvienas sub-modelis yra lokalus rekurentinis

neuroninis tinklas [5], kuris yra *m-H-1* formos (4. paveikslėlis), kur *m* ir *H* yra neuronų skaičius įėjimo ir paslėptam tinklo sluoksniuose, o išėjimo sluoksnis turi vieną neuroną, kuris išduoda taisyklės išėjimą $\hat{y}_l(k)$.

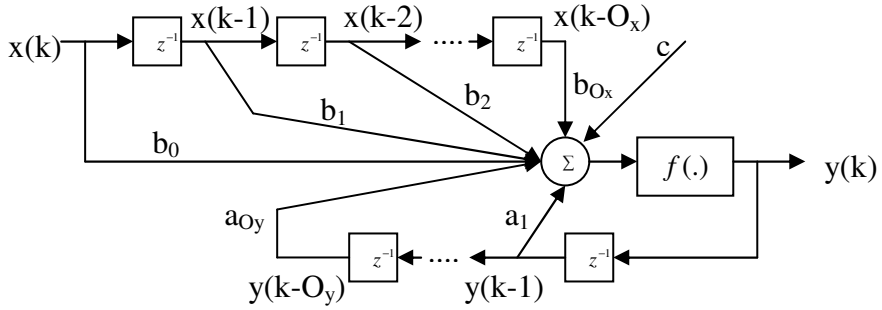


4 pav. Rekurentinio neuroninio tinklo struktūra

Įėjimo sluoksnis yra tiesinis, kai tuo tarpu paslėptas ir išėjimo sluoksnis sudarytas iš apibendrintų Fransconi-Gori-Soda (G-FGS) neuronų (5. Paveikslėlis) [4], kur $x(k)$ yra įėjimo signalas iš ankstesnio sluoksnio neurono, o $y(k)$ neurono išėjimas aprašomas išraiška:

$$y(k) = f\left(\sum_{j=0}^{O_x} b_j x(k-j) + \sum_{i=1}^{O_y} a_i y(k-i) + c \right) \tag{11}$$

čia b_j, $j = 0,...,O_x$, yra FIR jungiamieji svoriai ir a_i, $i = 0,...,O_y$, yra IIR jungiamieji svoriai. $f(.)$ neurono aktyvacijos funkcija, *c* paklaidos konstanta, o O_x ir O_y vėlinimo laipsnis įėjimo ir išėjimo signalų.



5 pav. Apibendrintas Frasconi-Gori-Soda (G-FGS) neuronas su vienu įėjimu.

Taigi pagal (11) išraišką RNN_l funkcionalumas aprašomas tokiomis išraiškomis:

$$O^1_{li}(k) = f_1\left(\sum_{j=1}^{m} \sum_{q=0}^{O_u} w^{1(l)}_{ijq} u_j(k-q) + \sum_{j=1}^{O_{y_1}} w^{2(l)}_{ij} O^1_{li}(k-j) + w^{3(l)}_i \right) \tag{12a}$$
$$l = 1,...,r, \;\; i = 1,...,H$$

$$\hat{y}_l(k) = f_2\left(\sum_{j=1}^{H}\sum_{q=0}^{O_{y3}} w_{jq}^{4(l)} O_{lj}^1(k-q) + \sum_{j=1}^{O_{y2}} w_j^{5(l)} \hat{y}_l(k-j) + w_i^{6(l)}\right) \quad (12b)$$
$$l = 1,...,r$$

čia f_1 ir f_2 yra neuronų aktyvacijos funkcijos, m įėjimų skaičius, $O_{li}^1(k)$ išėjimas *i*-tojo paslėpto sluoksnio neurono, $\hat{y}_l(k)$ išėjimas išėjimo sluoksnio neurono, O_u, O_{y_1}, O_{y_2}, O_{y_3} vėlinimo reikšmės, $w_{ijq}^{1(l)}, w_{ij}^{2(l)}, w_{jq}^{4(l)}, w_j^{5(l)}$ FIR ir IIR jungiamieji svoriai ir $w_i^{3(l)}, w_i^{6(l)}$ paklaidos konstantos.

2.4.3 Dinaminis miglotasis neuroninis apriboto optimizavimo apmokymo metodas

Šis algoritmas paremtas miglotuoju neuroniniu apriboto optimizavimo algoritmu (FUNCOM) [6] ir pritaikytas dinaminėms sistemoms, kuris gali būti pritaikomas ir DFNN dinaminės dalies radimui.

Apmokymo procesas skirtas nustatyti tinklo parametrus taip, kad geriausiai būtų nustatyta duotų įėjimo-išėjimo duomenų sąsają. Apmokymas yra vykdomas lygiagrečiai naudojant duomenis $\{(u(k), y_d(k)), k = 1,...,k_f\}$, kur k_f yra įėjimo-išėjimo porų skaičius. Pagal (12a) (12b) algoritmas aprašomas būsenos vektoriumi [3]:

$$x(k) = [[O^1(k)]^T [g(k)]^T]^T \quad (13)$$

čia $O^1(k) = [O_{11}^1(k),...,O_{1H}^1(k),...,O_{r1}^1(k),...,O_{rH}^1(k)]$ ir $g(k) = [\hat{y}_1(k),...,\hat{y}_r(k)]^T$ yra atitinkamai paslėptų sluoksnių neuronų išėjimai ir išvadų dalies išėjimai, ir valdymo vektoriumi θ susidedančiu iš FIR ir IIR jungiamųjų svorių, bei paklaidų konstantų:

$$\theta = [w_1^T, w_2^T, w_3^T, w_4^T, w_5^T, w_6^T]^T = [\theta_i] \quad (14)$$

Kadangi apmokymas vykdomas lygiagrečiai su k_f duomenų porų, tai būsenos vektorius visam duomenų rinkiniui aprašomas:

$$f(x,\theta) = [(f_O(k))^T (f_g(k))^T]^T = 0 \quad k = 1,...,k_f \quad (15)$$

čia:

$$f_O(k) : f_1\left(\sum_{j=1}^{m}\sum_{q=0}^{O_u} w_{ijq}^{1(l)} u_j(k-q) + \sum_{j=1}^{O_{y1}} w_{ij}^{2(l)} O_{li}^1(k-j) + w_i^{3(l)}\right) - O_{li}^1(k) = 0 \quad (16a)$$

$(k_f \times r \times H)$ lygčių, $l = 1,...,r,\ i = 1,...,H$

$$f_g(k): f_2\left(\sum_{j=1}^{H}\sum_{q=0}^{O_{y3}} w_{jq}^{4(l)} O_{lj}^{1}(k-q) + \sum_{j=1}^{O_{y2}} w_j^{5(l)} \hat{y}_l(k-j) + w_i^{6(l)}\right) - \hat{y}_l(k) = 0 \qquad (16b)$$

$(k_f \times r)$ lygčių, $l = 1,...,r,\ i = 1,...,H$

Algoritmas yra iteracinė procedūra, kuria tuo pat metu bandoma pasiekti du tikslus [3]:

- Klaidos matavimo funkcija E turėtų būti minimizuota kvadratų vidurkio klaidos (MSE) prasme:

$$E = \frac{1}{k_f}\sum_{k=1}^{k_f}[\hat{y}(k) - y_d(k)]^2 = \frac{1}{k_f}\sum_{k=1}^{k_f} e^2(k) \qquad (17)$$

  kur $\hat{y}(k)$ yra modelio išėjimo reikšmė, $y_d(k)$ tikroji sistemos išėjimo reikšmė ir $e(k)$ išėjimo klaidos reikšmė.

- Netiesinė funkcija $\Phi(\theta)$, kuri susieja valdymo vektoriaus svorinius koeficientus esamos iteracijos su buvusios iteracijos svoriniais koeficientais, turi būti maksimizuojama:

  $\Phi(\theta) = (\theta - \theta_E)^T(\theta_E - \theta_B)$ (18)

  kur θ_E yra esamos ir θ_B buvusios iteracijų valdymo vektoriai. Ši funkcija svarbi tuo, kad gali įtraukti struktūrines modelio charakteristikas apmokymo procese arba pagerinti konvergavimo greitį. Kuomet prielaidų dalis į apmokymo procesą neįtraukiama ir išvadų dalies struktūra nustatyta, funkcijos maksimizavimas $\Phi(\theta)$ naudojamas pagreitinti konvergavimo greitį išvengiant zigzaginių trajektorijų ieškant optimalių valdymo koeficientų.

Kiekvienos iteracijos metu valdymo koeficientai yra keičiami mažomis reikšmėmis $d\theta$ taip, kad tenkintų sąlygą:

$$\sum_{i=1}^{n}\frac{(d\theta_i)^2}{\Delta_i^2} = 1 \qquad (19)$$

čia Δ_i yra maksimali reikšmė koeficientų keitimui (MPC) θ_i valdymo vektoriui. Valdymo vektoriaus koeficientų keitimas automatiškai keičia būsenos vektoriaus reikšmes ir funkcijų E ir Φ reikšmes. Laikant, kad MPC yra pakankamai maža, galima minėtų funkcijų reikšmes aproksimuoti diferencialais dE ir $d\Phi$. Tuomet duotai reikšmei dE reikalinga rasti tokias optimalias koeficientų keitimo reikšmes $d\theta$, kurios maksimizuotų $d\Phi$ [3]:

$$d\theta = \pm\Delta^2 \cdot \sqrt{\frac{I_{EE} - (\delta E)^2}{I_{\Phi\Phi}\cdot I_{EE} - I_{E\Phi}^2}} \cdot \left[\Lambda_\Phi^T - \Lambda_E^T \frac{I_{E\Phi}}{I_{EE}}\right] + \Delta^2 \Lambda_E^T \frac{\delta E}{I_{EE}} \qquad (20)$$

čia $\Lambda_\Phi = \frac{\partial \Phi}{\partial \theta}$, $\Lambda_E = \lambda^T \cdot \frac{\partial f}{\partial \theta}$, $I_{EE} = \Lambda_E \Delta^2 \Lambda_E^T$, $I_{\Phi\Phi} = \Lambda_\Phi \Delta^2 \Lambda_\Phi^T$, $I_{E\Phi} = \Lambda_\Phi \Delta^2 \Lambda_E^T$ ir $\delta E = -\xi \cdot \sqrt{I_{EE}}$, čia λ Lagrandžo daugiklis, o ξ konstanta iš intervalo [0,1]. Suradus $d\theta$ valdymo vektorius sekančiai iteracijai paskaičiuojamas taip:

$$\theta(t+1) = \theta(t) + d\theta \tag{21}$$

Langrandžo daugiklis aprašomas tokia forma:

$$\lambda^T = [(\lambda^1)^T (\lambda^2)^T]$$

$$\lambda_l^2 = \frac{2}{k_f} \cdot [y(k) - y_d(k)] \cdot \frac{\mu_l(k)}{\sum_{i=1}^{r} \mu_i(k)} + \sum_{\substack{j=1 \\ k+j \le k_f}}^{O_{y2}} [\lambda_l^2(k+j) \cdot f_2'(k+j) \cdot w_j^{5(l)}] \tag{22a}$$

$$\lambda_{li}^1(k) = \sum_{\substack{j=1 \\ k+j \le k_f}}^{O_{y1}} [\lambda_{li}^1(k+j) \cdot f_1'(k+j) \cdot w_{ij}^{2(l)}] + \sum_{\substack{q=1 \\ k+q \le k_f}}^{O_{y3}} [\lambda_l^2(k+q) \cdot f_2'(k+q) \cdot w_{iq}^{4(l)}] \tag{22b}$$

čia $i = 1,...,H$, $l = 1,...,r$ ir $f_2'(k+j)$, $f_1'(k+j)$ yra atitinkamai $\hat{y}_j(k+l)$ ir $O_{li}^1(k+j)$ išvestinės, o λ^1 ir λ^2 yra atitinkamai paslėpto ir išėjimo sluoksnio daugikliai. (22a) ir (22b) lygtys gali būti išsprendžiamos atbuliniu būdu kai $k = k_f, k_f - 1,...,1$, taikant tokias lygtis:

$$\lambda_l^2(k_f) = \frac{2}{k_f} \cdot [y(k_f) - y_d(k_f)] \cdot \frac{\mu_l(k_f)}{\sum_{i=1}^{r} \mu_i(k_f)} \tag{23a}$$

$$\lambda_{li}^1(k_f) = \lambda_l^2(k_f) \cdot f_2'(k_f) \cdot w_{i0}^4 \tag{23b}$$

čia $i = 1,...,H$ ir $l = 1,...,r$.

Taip pat kadangi MPC turi didelę įtaką apmokymo greičiui, logiška kiekvienos iteracijos metu MPC reikšmę adaptuoti prie besikeičiančių sąlygų. Adaptacija galima atlikti stebint klaidos dalinės išvestinės pagal valdymo vektoriaus parametrus kitimą. Jeigu laikysime $\left.\frac{\partial E}{\partial \theta}\right|_E$ esamos ir $\left.\frac{\partial E}{\partial \theta}\right|_B$ buvusios klaidos dalinėmis išvestinėmis, n^+ ir n^- MPC parametro didinimo arba mažinimo koeficientais, o $\Delta_{\min}$ minimalia MPC galima reikšme, tai adaptavimas vyksta taip:

- Pradinėje būsenoje MPC prilyginama pirminiai reikšmei $\Delta_i = \Delta_0$. Pirminis reikšmės dydis pasirenkamas laisvai.
- Jeigu $\left.\frac{\partial E}{\partial \theta}\right|_E \times \left.\frac{\partial E}{\partial \theta}\right|_B > 0$ tai $\Delta_i = n^+ \cdot \Delta_i$. (24a)

- Jeigu $\left.\frac{\partial E}{\partial \theta}\right|_E \times \left.\frac{\partial E}{\partial \theta}\right|_B < 0$ tai $\Delta_i = \max\{n^- \cdot \Delta_i, \Delta_{\min}\}$ (24b)

Dėmesys taip pat turėtų būti atkreipiamas į tai, kad stebima ne dalinės išvestinės dydžio pokytis, bet dalinės išvestinės ženklo pokytis, tokiu būdu apmokymo greitis gali būti didinamas ir paskutinėse apmokymo stadijose, kuomet išvestinės reikšmės yra gana mažos.

3 Triukšmo slopinimo sistemos modelis

Šiame skyriuje aprašoma triukšmo slopinimo sistemos modelio struktūra ir elgsena, bei metodika ir jos efektyvumo įvertinimas modelio struktūrai nustatyti.

3.1 Dinaminio miglotojo neuroninio tinklo modelio identifikavimas

Turint tik sistemos įėjimo ir išėjimo duomenis negalime įvertinti kokia turėtų būti modelio struktūra. Kad identifikuoti prielaidų bei išvadų dalis reikia žinoti kiek taisyklių sudarys modelį, kokia jų bus lokalizacija ir kiti parametrai. Tikslinga tokiu atveju naudoti duomenų grupavimo metodus. Vienas iš labiausiai paplitusių metodų yra miglotasis c-vidurkių metodas [9, 10], tačiau jis reikalauja iš anksto nustatyti būsimų grupių kiekį, todėl grupių kiekiui ir pradinei grupių centrų lokalizacijai patogu naudoti duomenų grupavimo metodą pasiūlytą straipsnyje [7].

3.1.1 Duomenų grupavimo metodas

Straipsnyje siūlomas metodas [7] remiasi pasiūlytu metodu [8], tačiau yra skaičiavimų atžvilgiu greitesnis ir tikslesnis negu pastarasis ir nereikalaujantis optimizavimo procedūrų.

Turint n duomenų taškų $\{x_1, x_2, ..., x_n\}$ M dimensijų erdvėje pirmiausia atliekamas duomenų taškų normalizavimas kiekvienoje dimensijoje, taip, kad koordinačių ribos kiekvienoje dimensijoje būtų lygios. Kadangi kiekvienas duomenų taškas laikomas potencialiu duomenų grupės centru, tikslinga skaičiuoti kiekvieno taško potencialą įvertinant atstumus iki kitų nuo jo nutolusių taškų:

$$P_i = \sum_{j=1}^{n} e^{-\alpha \left\| x_i - x_j \right\|^2} \tag{25}$$

čia $\alpha = \frac{4}{r_a^2}$ ir r_a yra teigiama konstanta. Tokių būdu taškas šalia kurio yra didžiausias taškų skaičius turės didžiausią potencialą ir bus duomenų grupės centru. Konstanta r_a nusako spindulį taško potencialui skaičiuoti, t.y. taškai esantys už šio spindulio turi mažai įtakos skaičiuojamo taško potencialui.

Po to kai paskaičiuojami potencialai kiekvienam taškui, didžiausią potencialą turintis taškas tampa pirmos taškų grupės centru. Galima padaryti išvadą, kad aplink surastą grupės centro tašką esantys taškai taip pat turi didelius potencialus. Dėl šios priežasties tam tikru spinduliu nuo grupės centro taško reiktų sumažinti taškų potencialus, taip kad nei vienas iš tų

taškų negalėtų tapti sekančių taškų grupių centru. Taigi jeigu pirmosios grupės centro taškas yra x_1^* ir P_1^* šio taško potencialas, visi taškų potencialai perskaičiuojami tokiu būdu:

$$P_i = P_i - P_1^* e^{-\beta \left\| x_i - x_1^* \right\|^2} \tag{26}$$

čia $\beta = \frac{4}{r_b^2}$ ir r_b yra teigiama konstanta. Konstanta r_b nusako spindulį nuo grupės centro taško, į kurį patenka taškai, kurių potencialai turi būti tiek sumažinti, kad nebebūtų pasirenkami naujais grupių centrais. Tikslinga konstantą r_b pasirinkti siek tiek didesnę negu, kad r_a, kad išvengti per arti vienas kito išsidėsčiusių centrų: rekomenduotina, kad $r_b = 1.5 r_a$.

Perskaičiavus pagal (26) išraišką taškų potencialus, pasirenkamas sekantis didžiausią potencialą turintis taškas, kuris tampa sekančios taškų grupės centru. Kiekvieną kartą suradus naują centro tašką x_k^* perskaičiuojame potencialus pagal išraišką:

$$P_i = P_i - P_k^* e^{-\beta \left\| x_i - x_k^* \right\|^2} \tag{27}$$

Procedūra atliekama tol, kol pirmojo ir paskutinio surasto taškų santykis yra didesnis už nustatomą konstantą ε:

$$P_k^* < \varepsilon \cdot P_1^* \tag{28}$$

Nuo konstantos ε dydžio priklauso kiek taškų grupių centrų bus surasta, todėl šią konstantą sunku parikti optimalią visiems taškų rinkiniams ir siūloma naudoti tokią grupių centrų priėmimo ir atmetimo procedūrą [7]:

- Jeigu $P_k^* > \bar{\varepsilon} \cdot P_1^*$, tai taškas x_k^* priimamas kaip taškų grupės centras ir duomenų grupavimas tęsiamas.
- Jeigu pastaroji sąlyga netenkinama, tuomet jeigu $P_k^* < \underline{\varepsilon} \cdot P_1^*$, tai taškas x_k^* atmetamas kaip taškų grupės centras ir duomenų grupavimas nutraukiamas.
- Jeigu abi pastarosios sąlygos netenkinamos, tuomet jeigu $\frac{d_{\min}}{r_a} + \frac{P_k^*}{P_1^*} \geq 1$, tai taškas x_k^* priimamas kaip taškų grupės centras, priešingai taško x_k^* potencialas prilyginamas 0, imamas sekantis taškas su aukščiausiu potencialu ir ši procedūra kartojama.

čia $\bar{\varepsilon}$ ir $\underline{\varepsilon}$ konstantos aprašančios priėmimo ir atmetimo santykius, o $d_{\min}$ trumpiausias atstumas tarp x_k^* taško ir visų iki tol rastų centrų taškų. Paskutinioji sąlyga nustato ar taškas

turi pakankamai potencialo ir ar jis nėra per arti kitų rastų centrų taškų, kai netenkinamos konkrečios dvi pirmosios sąlygos.

Šis duomenų grupavimo metodas leidžia rasti duomenų grupes ir jų centrus, tačiau gaunami grupių centrai įėjimo duomenų atžvilgiu yra per tankiai išsidėstę, kas įtakoja didelį narystės funkcijų persidengiamumą. Modelio sudarymas naudojant daug persidengiančias narystės funkcijas būtų netikslus, todėl tikslinga šiuo atveju optimizuoti grupių centrus naudojant miglotąjį c-vidurkių grupavimo metodą.

3.1.2 Grupių centrų optimizavimas miglotuoju c-vidurkių metodu

Miglotasis c-vidurkių metodas yra vienas svarbiausių ir populiariausių tarp miglotojų grupavimo metodų, tačiau turintis keletą problemų. Metodo veikimas labai priklauso nuo pirminės grupių centrų lokalizacijos, bei dažniausiai pasiekiamas optimizuojamos funkcijos lokalus minimumas.

Siekiant išvengti pernelyg tankiai išsidėsčiusių grupės centrų, laikoma, kad esant duomenų taškui ties grupės centru, taško narystės laipsnis bus maksimalus šios grupės atžvilgiu, o visų kitų grupių įėjimo narystės laipsnis bus lygus nuliui. Todėl narystės laipsniai taškams yra apriboti tokiomis sąlygomis [9]:

$$\forall_i, \sum_{j=1}^{C} \mu_{ij} = 1; \quad \forall_j, \sum_{i=1}^{n} \mu_{ij} > 0 . \tag{29}$$

čia C yra grupių skaičius, o n duomenų taškų kiekis.

Narystės laipsnis $i-tojo$ taško $j-tojoj$ grupėj aprašomas taip:

$$\mu_{ij} = \frac{1}{\sum_{k=1}^{C} \left(\frac{\left\| x_i - v_j \right\|}{\left\| x_i - v_k \right\|} \right)^{2/(m-1)}} \tag{30}$$

čia v_j, $j = 1..C$, yra grupių centrai, kurie skaičiuojami, kaip visų taškų svorinis vidurkis:

$$v_j = \frac{\sum_{i=1}^{n} {\mu_{ij}}^m \cdot x_i}{\sum_{i=1}^{n} {\mu_{ij}}^m} \tag{31}$$

Dažniausiai $m = 2$, nes tuomet taškų narystės laipsniai (30) normalizuojami taip, kad tenkintų (29) sąlygas. Miglotasis c-vidurkių metodas yra iteracinė optimizavimo procedūra, kuri minimizuoja dažniausiai tokios formos funkciją:

$$J = \sum_{i=1}^{n} \sum_{j=1}^{C} {\mu_{ij}}^m \left\| x_i - v_j \right\|^2 \tag{32}$$

Optimizavimo procesas nutraukiamas, kuomet optimizuojamos funkcijos reikšmės iteracijų metu nebesikeičia užduotu ribiniu dydžiu.

Kadangi algoritmo veikimas priklauso nuo užduodamų pirminių grupės centrų lokalizacijos, tikslinga pirminiais grupės centrais laikyti 3.1.1 skyriuje aprašytu metodu randamus centrus, kas dažniausiai sumažina iteracijų kiekį reikalingam optimizaciniam tikslumui pasiekti. Miglotoju c-vidurkių metodu gauti grupių centrai yra pakankamai toli nuo vienas kito įėjimo duomenų atžvilgiu, kas leidžia pakankamai tiksliai aprašyti dinaminio miglotojo neuroninio tinklo prielaidų dalį.

3.1.3 Dinaminio miglotojo neuroninio tinklo prielaidų dalies identifikavimas

Prielaidų dalis pagal 2.3.1 skyrių pilnai aprašoma narystės funkcijomis (8). Atlikus duomenų grupavimą, nustatoma kiek taisyklių sudarys dinaminio miglotojo neuroninio tinklo modelį, bei nustatomi kiekvienos taisyklės narystės funkcijų parametrai. Pagal (8) parametrai aprašantys narystės funkcijas yra grupės vidurkis ir standartinis nuokrypis. Grupės vidurkis šiuo atveju yra kaip parametras lokalizuojantis narystės funkciją, todėl grupių vidurkius atitinka grupių centrai, randami 3.1.1 ir 3.1.2 skyriuose aprašytais metodais. Standartinis nuokrypis yra parametras aprašantis narystės funkcijų pločius, todėl turint narystės laipsnius apskaičiuotus miglotuoju c-vidurkių metodu, nustatomos taškų priklausomybės grupėms pagal didžiausią narystės laipsnį ir paskaičiuojami kiekvienos grupės taškų standartiniai nuokrypiai. Tokiu būdu pilnai aprašoma dinaminio miglotojo neuroninio tinklo prielaidų dalis.

3.1.4 Dinaminio miglotojo neuroninio tinklo išvadų dalies identifikavimas

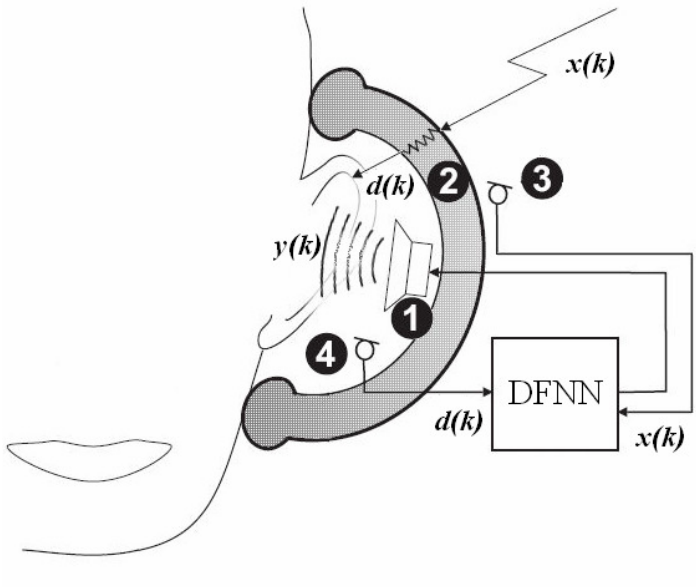
Žinant taisyklių kiekį priskiriamas kiekvienai taisyklei rekurentinis neuroninis tinklas. Pagal 2.3.2 skyriuje pateiktą medžiagą, kiekvienas lokalus rekurentinis neuroninis tinklas turi savo koeficientų aibes, kurių dydis priklauso nuo neuroninių tinklų architektūrinio sudėtingumo. Sistemos projektuotojas arba ekspertas nustato rekurentinių neuroninių tinklų naudojamų neuronų kiekį paslėptame sluoksnyje, bei pačių neuronų įėjimų, bei išėjimų vėlinimo koeficientus. Nustačius šias reikšmes tolesnis išvadų dalies identifikavimas vykdomas D-FUNCOM apsimokymo algoritmo pagalba, kuomet parenkamos optimalios lokalių rekurentinių neuroninių tinklų koeficientų reikšmės. Atlikus apmokymo procedūrą

dinaminio miglotojo neuroninio tinklo modelis tampa pilnai aprašytas ir gali būti integruotas į bendrą triukšmo slopinimo sistemos modelį.

3.2 Triukšmo slopinimo sistemos struktūrinis modelio aprašymas

Triukšmo slopinimo sistemos struktūra pavaizduota 6 paveikslėlyje. Sistema sudaryta iš tokių pagrindinių komponentų:

- Fizinės sistemos, kurios perdavimo funkcija yra nežinoma ir dinamiškai kintanti. Šiuo atveju tai yra ausinių gaubteliai pažymėti numeriu 2.
- Mikrofonų, kurie fiksuoja triukšmo signalus ausinių gaubtelių išorėje (3 numeris) ir viduje (4 numeris).
- Garsiakalbio, kuris naudojamas skleisti sugeneruotą antitriukšmo signalą ausinių viduje.
- Valdymo sistemos, kuri šiuo atveju yra dinaminis miglotasis neuroninis tinklas generuojantis antitriukšmo signalus. Dažniausiai tokia valdymo sistema įdiegiama naudojant diskretinius signalų procesorius, kurie pasižymi didele signalų apdorojimo sparta realiu laiku.

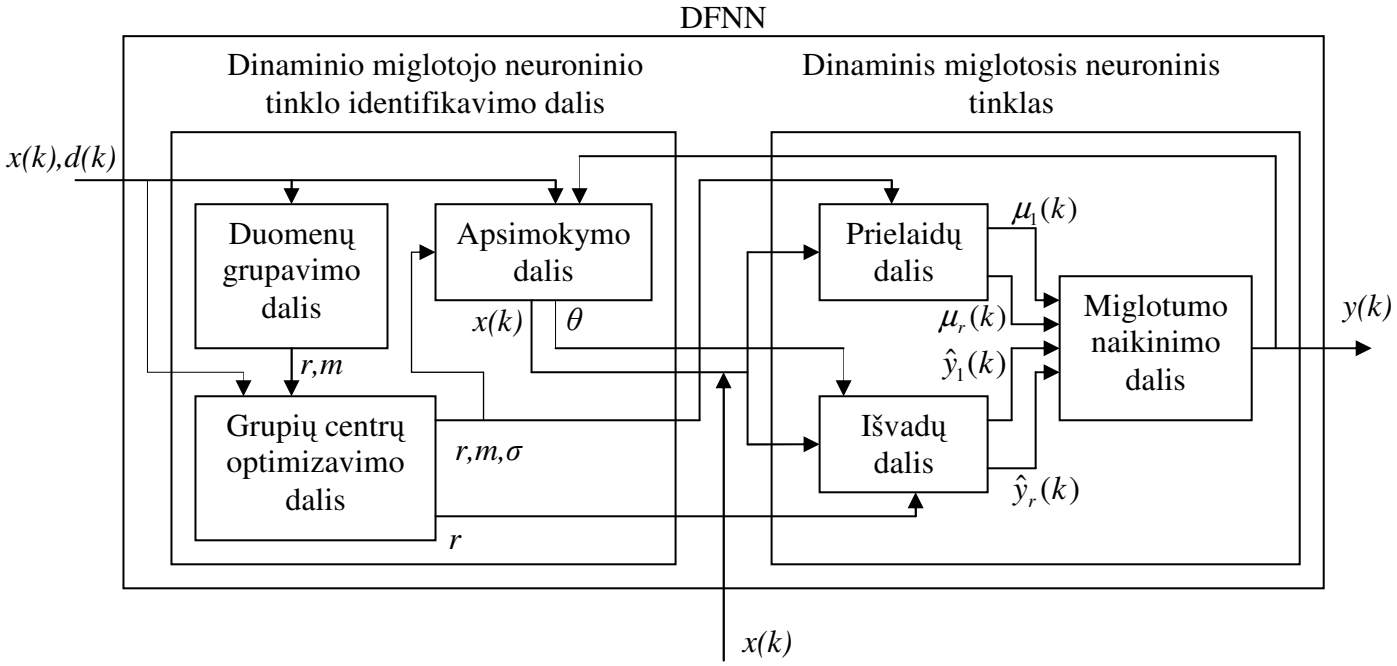


6 pav. Triukšmo slopinimo sistemos struktūrinė schema.

Dinaminio miglotojo neuroninio tinklo struktūra 7 paveikslėlyje. Tinklas šiuo atveju sudarytas iš dviejų pagrindinių dalių:

- Tinklo identifikavimo dalies, kurioje naudojant įėjimo išėjimo duomenis *x(k)* ir *d(k)* nustatoma tinklo struktūra ir valdantieji svoriniai koeficientai.

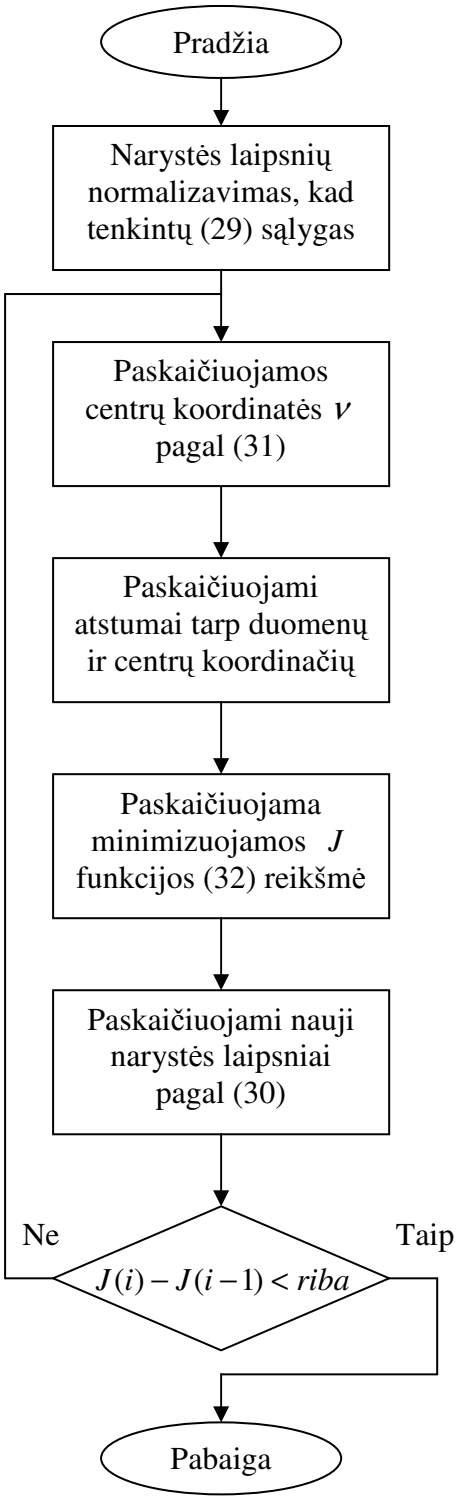
- Pačio tinklo, kuris naudojamas ne tik antitriukšmo generavimui, bet ir valdančiųjų svorinių koeficientų nustatymo metu.



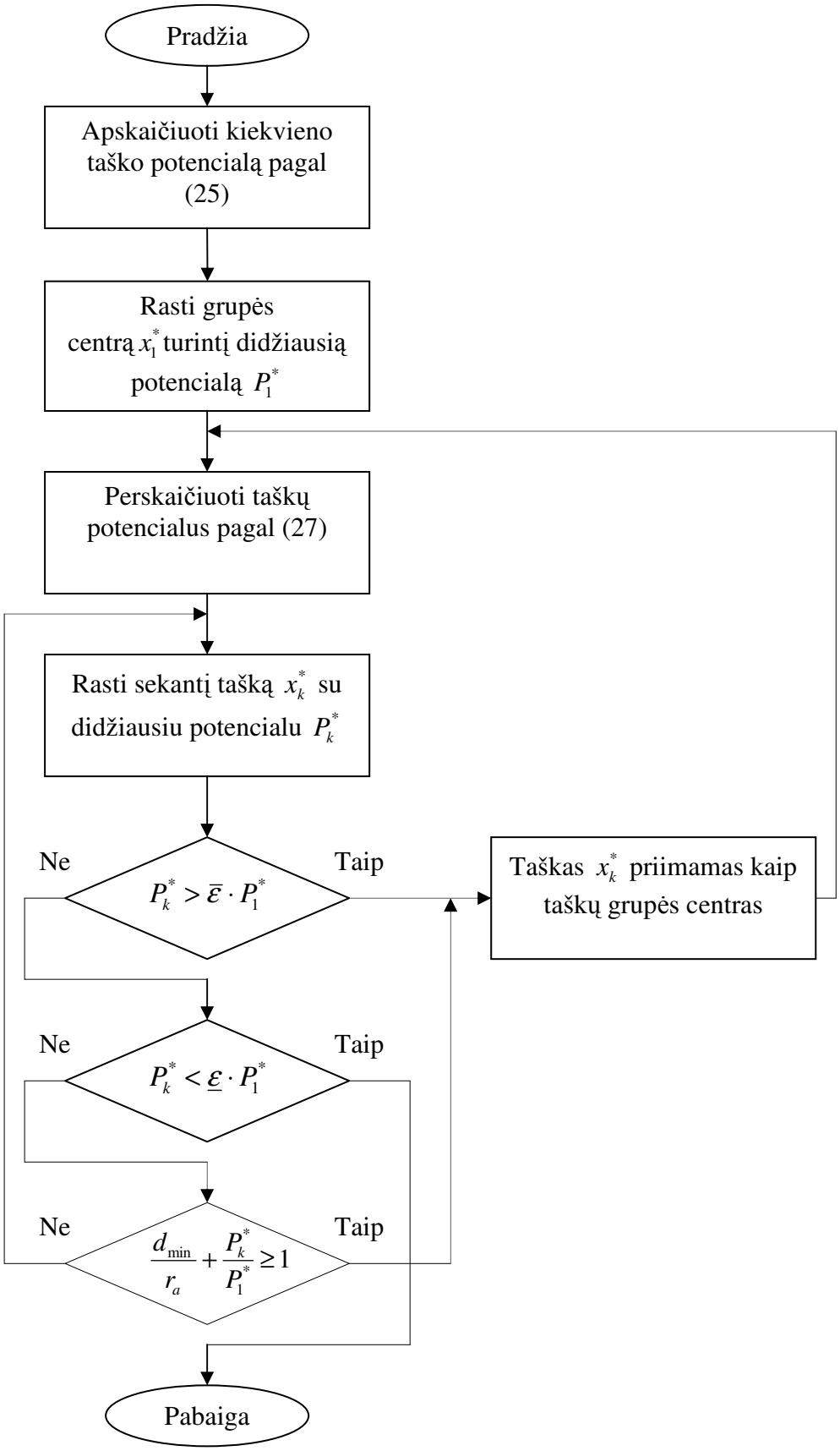
7 pav. DFNN struktūrinė schema

3.3 Triukšmo slopinimo sistemos modelio elgsenos aprašymas

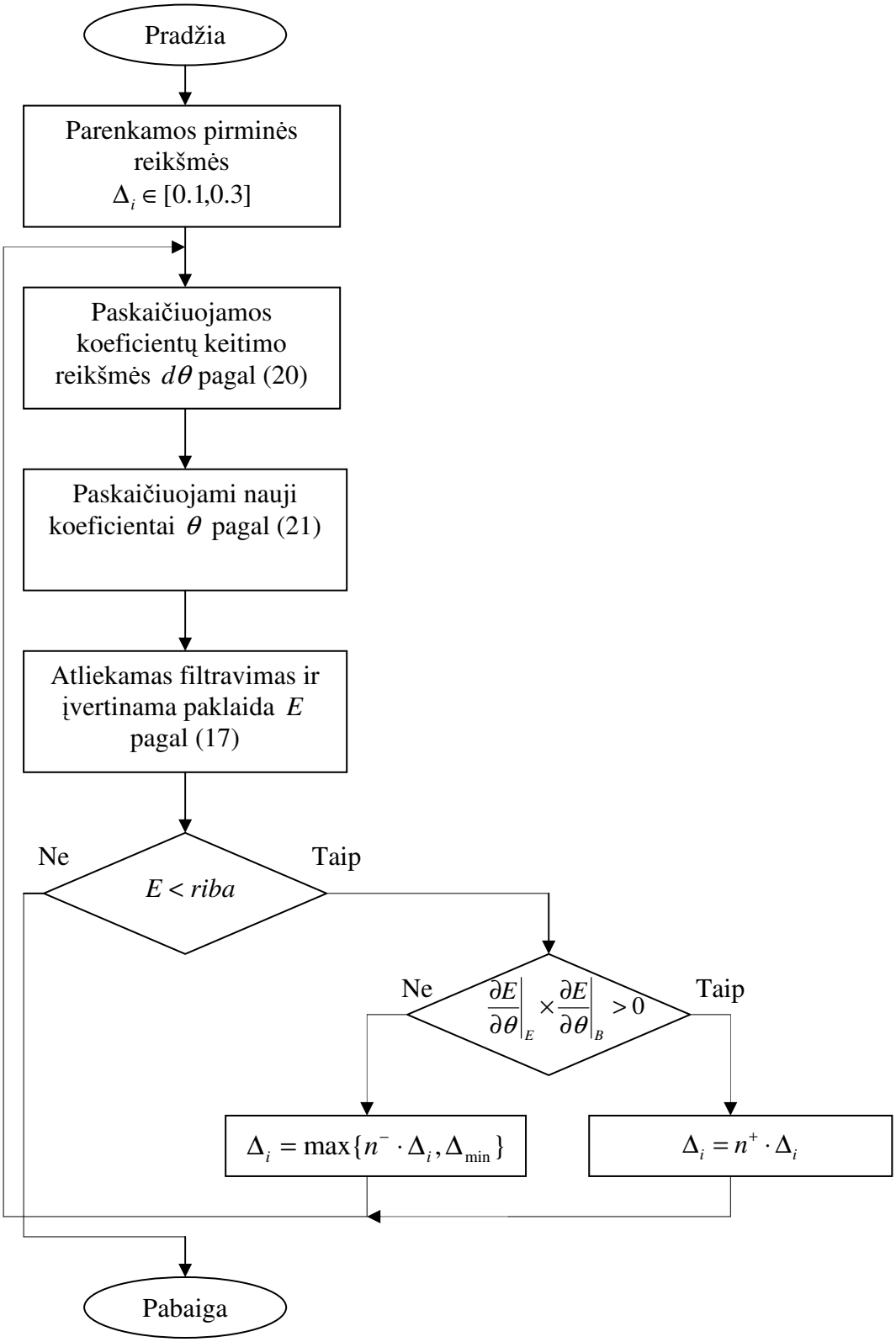
Pagal 7 paveikslėlyje pateiktą struktūrinę DFNN schemą, modelio identifikavimo dalies elgseną galima aprašyti naudojamų metodų blokinėmis schemomis pateiktomis 8, 9, 10 paveikslėliuose. Modelio DFNN tinklo funkcionalumas pilnai aprašomas (12a) ,(12b) ir 10 išraiškomis.



8 pav. Grupių centrų optimizavimo metodo blokinė schema.



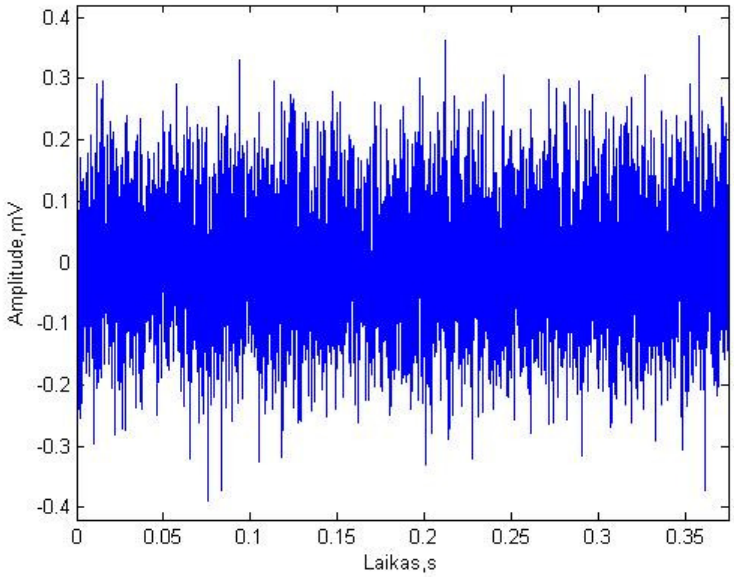
9 pav. Duomenų grupavimo metodo blokinė schema.



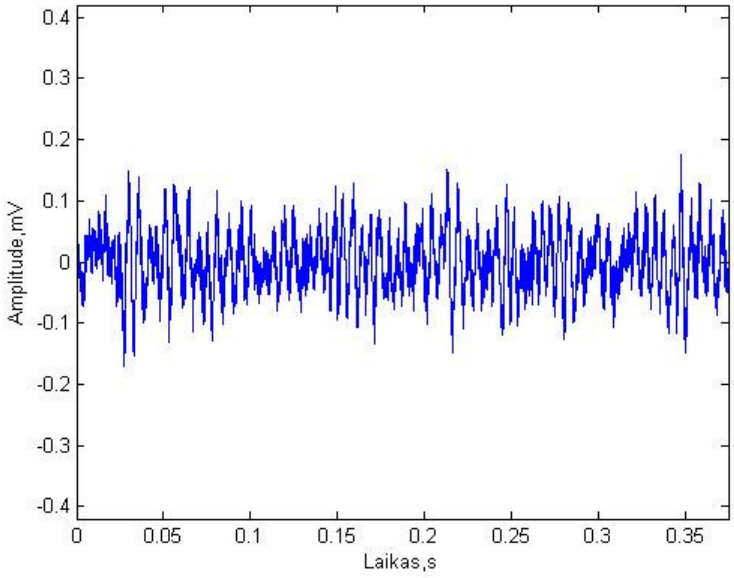
10 pav. Apsimokymo metodo blokinė schema.

3.4 Dinaminio miglotojo neuroninio tinklo identifikavimo metodikos įvertinimas

Pagal 3.1 skyriuje pateiktą metodiką, turint įėjimo ir išėjimo duomenis, galima nustatyti dinaminio miglotojo neuroninio tinklo struktūrą. Kad galima būtų įvertinti metodikos veiksmingumą konkrečiu triukšmo slopinimo problemos atveju, skaičiavimai atliekami su duomenimis gautais pagal 4.1 skyriuje aprašytą metodiką. 11 ir 12 paveikslėliuose matomi atvaizduoti įėjimo ir išėjimo duomenys. Čia įėjimo duomenimis yra sugeneruotas baltasis triukšmas.

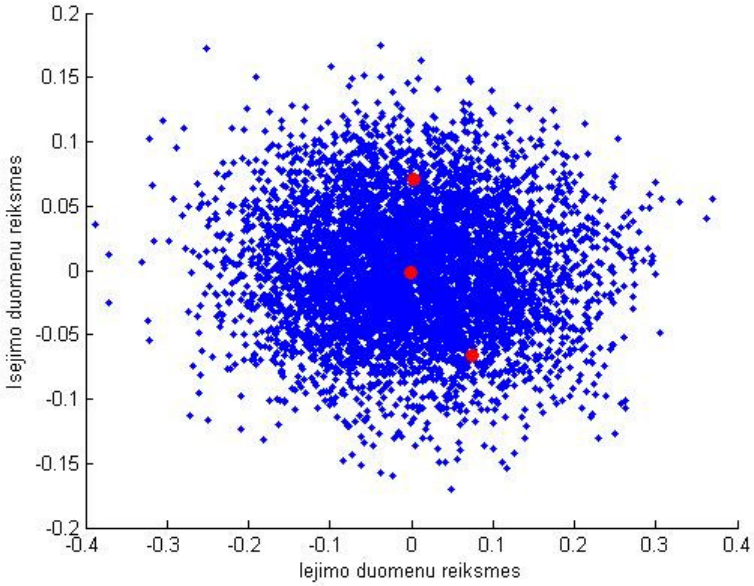


11 pav. Modelio įėjimo duomenys.

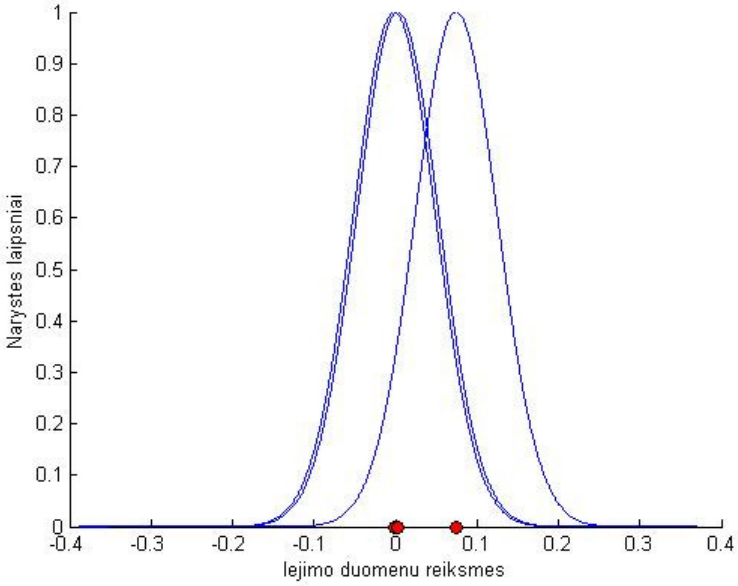


12 pav. Modelio išėjimo duomenys.

Laikantis metodikos pirmiausia atliekamas duomenų grupavimas pagal 9 paveikslėlyje pateiktą duomenų grupavimo blokinę schemą. Randami 3 duomenų grupių centrai, kurių išsidėstymas pateiktas 13 paveikslėlyje. Atkreipiamas dėmesys į tai, kad iš tikrųjų grupių centrai įėjimo duomenų atžvilgiu yra per arti vienas kito arba beveik sutampa. Dėl šios priežasties sudaromos narystės funkcijos, pateiktos 14 paveikslėlyje, yra per daug persidengiančios ir netinka sudaromo triukšmo slopinimo sistemos modelio struktūrai aprašyti.



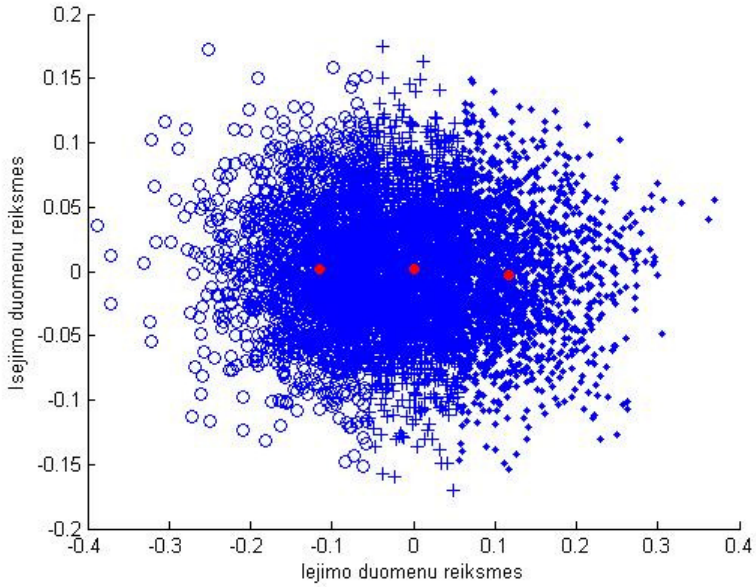
13 pav. Duomenų grupių centrų lokalizacija po duomenų grupavimo.



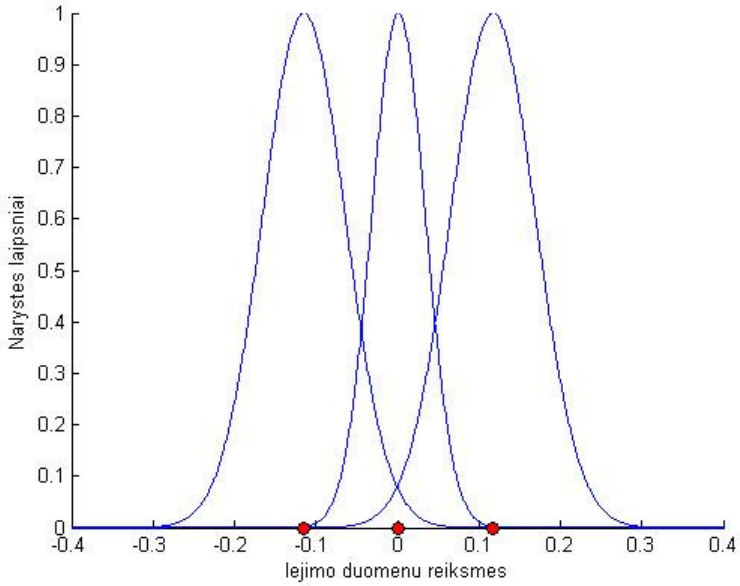
14 pav. Gaunamos narystės funkcijos po duomenų grupavimo.

Gautieji duomenų grupių centrai pagal grupių centrų optimizavimo blokinę schemą (8 paveikslėlis) optimizuojami taip, kad tenkintų 3.1.2 skyriuje pateiktas (29) sąlygas. Gaunama

duomenų grupių centrų lokalizacija, pateikta 15 paveikslėlyje, yra tokia, kad centrai yra pakankamai toli nuo vienas kito įėjimo duomenų atžvilgiu. Duomenų taškų priklausomybė tam tikrai grupei pavaizduota kaip atskirais simboliais pažymėtais taškais. Apskaičiavus kiekvienos duomenų grupės standartinį nuokrypį sudaromos patikslinto pločio narystės funkcijos pateiktos 16 paveikslėlyje.



15 pav. Duomenų grupių centrų lokalizacija po optimizavimo procedūros.



16 pav. Gaunamos narystės funkcijos po optimizavimo procedūros.

Palyginus 16 ir 14 paveikslėliuose pavaizduotas narystės funkcijas, matomas aiškus skirtumas narystės funkcijų išsidėstyme ir persidengiamume. Galima padaryti išvadą, kad naudojant 16 paveikslėlyje gaunamas narystės funkcijas žymiai tiksliau aprašomas triukšmo slopinimo

sistemos modelis, t.y. siūloma metodika pagal gautus rezultatus pasiteisino ir gali būti naudojama automatiškam triukšmo slopinimo sistemos struktūros generavimui.

4 Eksperimentinis tyrimas

Šiame skyriuje aprašomas eksperimentinis tyrimas ir jo aplinka, bei įvertinami tyrimo metu gauti rezultatai.

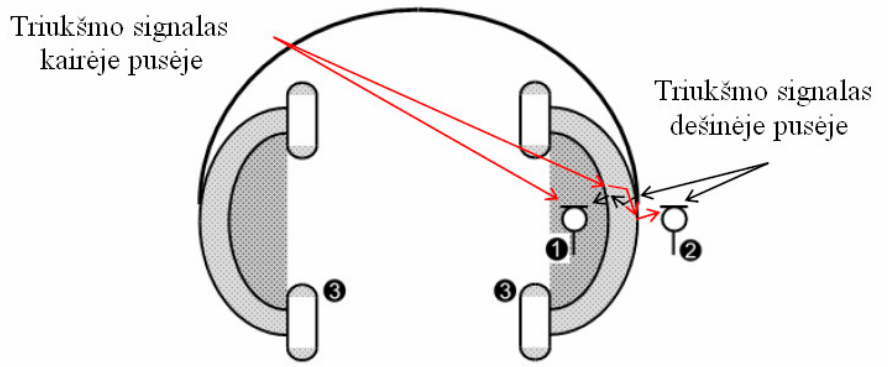
4.1 Eksperimentinio tyrimo aplinka

Siekiant kuo tiksliau sudaryti triukšmo slopinimo sistemos modelį, bei įvertinti sudaryto modelio duodamus rezultatus realioje aplinkoje, tikslinga eksperimentą atlikti su realiai įrašytais signalais, kurių pagalba į tyrimą įtraukiamos netiesinės bei dinaminės sistemos charakteristikos.

Aplinka, kurioje realiai įrašomi signalai, turi pasižymėti:

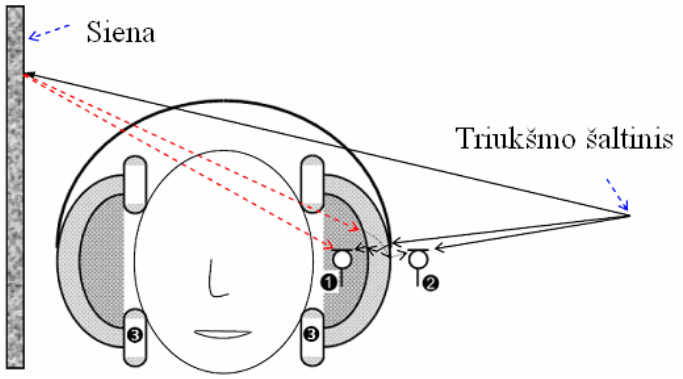
- įrašančios įrangos tam tikru išdėstymu;
- akustinėmis garsą sugeriančiomis charakteristikomis.

Įrangos išdėstymas šiuo atveju yra labai svarbus, nes naikintinas garso signalas pirmiausia turi pasiekti išorinį mikrofoną nr.2, vėliau nr.1 (17 paveikslėlis). Todėl esant tokiam mikrofonų išdėstymui garso signalų šaltinis turėtų būti dešinėje pusėje. Jeigu garso signalų šaltinis bus kairėje pusėje, gausime laike neigiamą perdavimo funkciją, t.y. neįmanoma panaikinti garso signalų, kurie anksčiau pasiekia klausos organus, negu kad išorinį mikrofoną.



17 pav. Įrašančios įrangos išdėstymas.

Aplinkos akustinės garsą sugeriančios charakteristikos taip pat labai svarbios, nes garso signalai atsispindi nuo aplinkoje esančių daiktų, sienų ir t.t. Tokiu būdu atsispindėję signalai gali pasiekti pirmiausia vėlgi mikrofoną nr.1 (18 paveikslėlis), ko pasėkoje susiduriama su anksčiau minėta problema, todėl aplinka turi pasižymėti gerai garsą sugeriančiomis akustinėmis charakteristikomis.



18 pav. Akustinė atspindžio problema.

4.2 Eksperimentinio tyrimo atlikimo metodika

Eksperimentinis tyrimas atliekamas tokiomis stadijomis, naudojant prieduose pateiktų metodų MatLab programines realizacijas:

1) Pagal 4.1 skyriuje pateiktą metodiką įrašomi signalai. Čia įėjimo duomenimis pasirenkamas baltasis triukšmas - atsitiktinis signalas, kurio energetinis spektras yra pastovus visiems dažniams.
2) Pagal 3.1 skyriuje pateiktą metodiką nustatomas taisyklių kiekis ir narystės funkcijų parametrai dinaminio miglotojo neuroninio tinklo prielaidų daliai. Metodo programinė realizacija pateikta prieduose nr. 1,2 ir 3.
3) Nustatomas neuronų kiekis paslėptame sluoksnyje ir neuronų vėlinimo koeficientai.
4) Atliekamas tinklo apmokymas naudojant 2.4.3 skyriuje pateiktą D-FUNCOM metodą (priedas nr. 5).
5) Atliekamas filtravimas (priedas nr.4).
6) Įvertinami gautieji rezultatai.

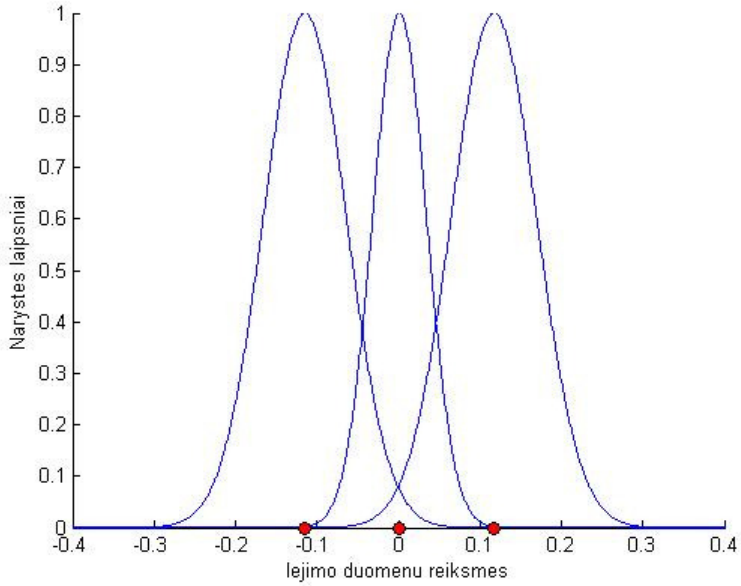
4.3 Eksperimentinio tyrimo rezultatų įvertinimas

Naudojant įrašytus duomenis, įėjimo duomenų aibė pagal 3.1 skyriuje pateiktą metodiką padalinama į tris grupes, o gaunamos narystės funkcijos pateiktos 19 paveikslėlyje. Apmokymo proceso metu duomenimis imama 20 įėjimo-išėjimo porų, parenkant paslėpto sluoksnio neuronų skaičių ir vėlinimo koeficientus, tokius, kad būtų gaunami pakankamai geri rezultatai (1 lentelė, Nr. 1). 21 paveikslėlyje pateikiami tikrieji ir apsimokymo metu gautieji išėjimo duomenys, iš kurių matyti, kad apmokymo metu gautieji duomenys mažai skiriasi nuo

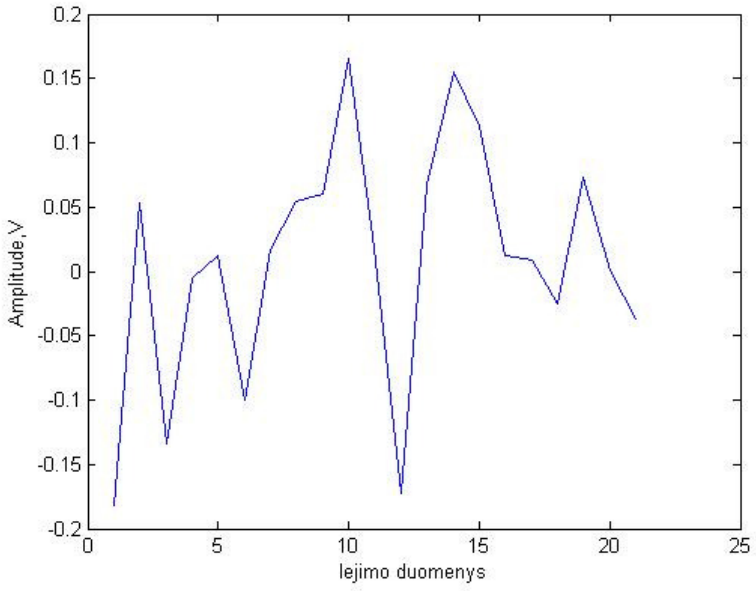
tikrųjų ir vidutinė kvadratinė paklaida yra $5.4*10^{-6}$. Tačiau atliekant filtravimą rezultatai yra gana prasti (1 lentelė, Nr. 1), nes apmokymui naudojama per mažai taškų ir dinaminis miglotasis neuroninis tinklas nesugeba aproksimuoti visų dinaminių sistemos savybių.

Nr.	Neuronų skaičius paslėptame sluoksnyje	Vėlinimų reikšmės				Vidutinė kvadratinė paklaida	
		O_u	O_{y_1}	O_{y_2}	O_{y_3}	Atlikus apmokymą	Atlikus filtravimą
1	3	2	4	5	2	$5.4*10^{-6}$	$1.8*10^{-3}$
2	3	2	5	5	2	$1.7*10^{-4}$	$1.6*10^{-3}$

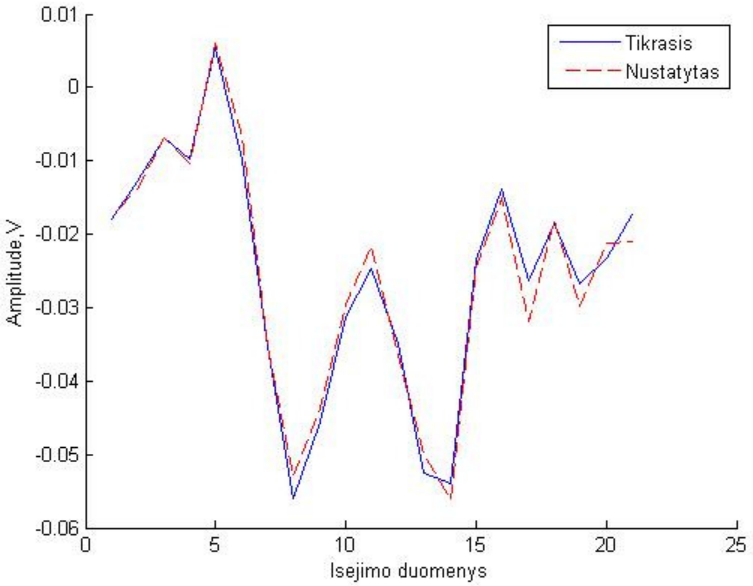
Lentelė 1. Tyrimo rezultatai.



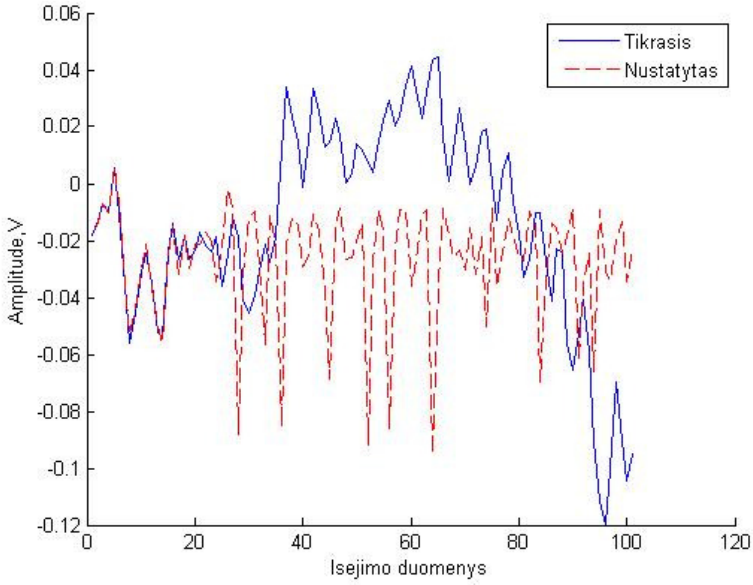
19 pav. Narystės funkcijos.



20 pav. Apmokymo įėjimo duomenys.

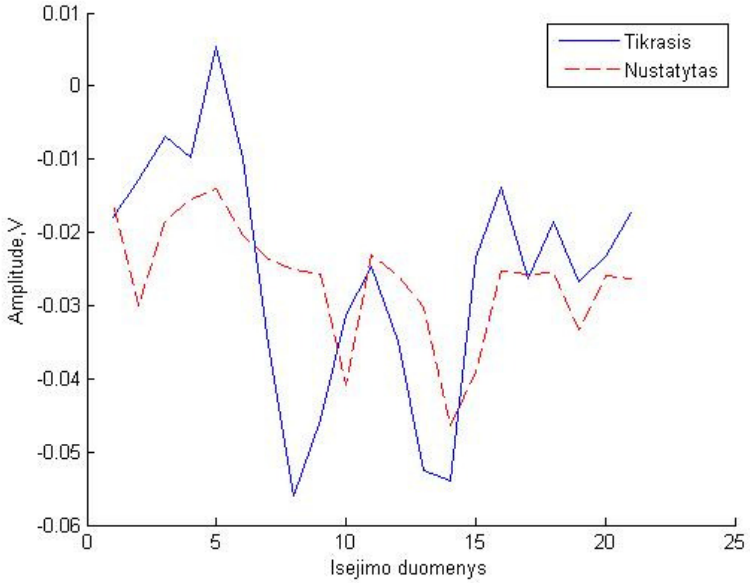


21 pav. Tikrieji ir apmokymo metu gautieji išėjimo duomenys.



22 pav. Tikrieji ir filtravimo metu gautieji išėjimo duomenys.

Pakeitus vieną iš vėlinimo koeficientų reikšmių (1 lentelė, Nr. 2), gaunami rezultatai apmokymo metu yra prastesni daugiau kaip 300 kartų (23 paveikslėlis). Tai rodo, kad dinaminis miglotasis neuroninis tinklas yra jautrus šioms reikšmėms ir nors vėlinimo koeficientas padidinamas vos vienu dydžiu, rezultatai tampa žymiai prastesni (21, 23 paveikslėliai).



23 pav. Tikrieji ir apmokymo metu gautieji išėjimo duomenys.

4.4 Rekomendacijos

Optimalių vėlinimo koeficientų ir paslėpto sluoksnio neuronų skaičiaus parinkimas kaip parodė eksperimentas yra gana problematiškas, nes šioms reikšmėms dinaminis miglotasis neuroninis tinklas yra jautrus ir nuo jų priklauso gaunamų rezultatų tikslumas. Be to galima padaryti prielaidą, kad kiekvienos taisyklės išvadų dalis nebūtinai turėtų būti realizuojama naudojant vienodą skaičių neuronų paslėptame sluoksnyje ir nebūtinai su vienodais vėlinimais. Kaip parodė eksperimentas, padidinus vieną iš vėlinimų rezultatai netgi žymiai suprastėjo. Todėl parenkant kiekvienos taisyklės išvadų daliai neuronų kiekius ir vėlinimo koeficientus galima būtų ne tik sumažinti naudojamų koeficientų kiekį ir supaprastinti tinklo struktūrą, bet tuo pačiu galbūt gauti ir geresnius rezultatus. Bet be abejonės toks koeficientų parinkimas yra gana komplikuotas ir reikalaujantis tolesnių tyrimų.

5 Išvados

- Atliekant rekurentinių miglotojų neuroninių filtrų (RMNF) analizę išanalizuoti Mamdani ir Sugeno miglotojų išvadų sistemų tipai ir dirbtinių neuroninių tinklų lokalios ir globalios rekurencijos modeliai. Parenkant modelius triukšmo slopinimo problemai spręsti buvo atsižvelgta į skaičiavimų ir optimizavimo efektyvumą, bei struktūrinį sudėtingumą.
- Apžvelgus du pagrindinius Mamdani ir Sugeno (TSK) išvadų sistemos modelių tipus, nustatyta, kad rekurentinio miglotojo neuroninio filtro architektūra turėtų būti paremta TSK miglotuoju modeliu, nes pastarojo modelio išvadų dalis yra pakeista išvadų funkcija, kurios dėka modelio išėjimas paskaičiuojamas naudojant paprasčiausią svorinį vidurkio metodą, skirtingai nuo Mamdani modelyje naudojamų integravimo procedūrų, bei šiuo atveju išvadų funkcija gali būti realizuota ir naudojant rekurentinius neuroninius tinklus.
- Rekurentinių neuroninių tinklų modelių analizės metu didžiausias dėmesys skirtas rekurencijos tipui parinkti. Geriausia naudoti lokalios rekurencijos tinklus, nes jie supaprastina viso projektuojamo RMNF architektūrą, kadangi nenaudojamas grįžtamasis ryšys į RMNF įėjimą iš išėjimo, kurio naudojimas automatiškai komplikuoja RMNF prielaidų dalies suskaidymą į miglotąsias aibes ir įtakoja taisyklių galimo kiekio padidėjimą.
- Atsižvelgus į analizės rezultatus kaip vienas tinkamiausių triukšmo slopinimo problemos sprendimui RMNF klasės filtrų pasirinktas dinaminis miglotasis neuroninis tinklas (DFNN), nes filtro architektūra yra paremta TSK modeliu, o išvadų sistema realizuojama naudojant lokalios rekurencijos tinklus.
- DFNN struktūra nėra apibrėžta ir turint tik triukšmo sistemos įėjimo ir išėjimo duomenis įvertinti DFNN struktūrą yra sudėtinga, todėl darbe pasiūlyta metodika, kuri leidžia automatiškai generuoti DFNN struktūrą turint įėjimo ir išėjimo duomenis.
- Pasiūlyta DFNN struktūros identifikavimo metodika remiasi duomenų grupavimo ir optimizavimo metodais. Vien duomenų grupavimo metodo neužtenka identifikuoti DFNN struktūrai, nes gaunami grupių centrai įėjimo duomenų atžvilgiu yra per arti vienas kito išsidėstę arba beveik sutampa įtakodami didelį narystės laipsnių persidengiamumą. Ši problema išsprendžiama optimizuojant grupių centrus c-vidurkių metodu.
- Eksperimentinis tyrimas parodė, kad DFNN geba aproksimuoti dinamines sistemos savybes, tačiau pilnam dinaminių savybių aproksimavimui reikalinga atlikti

apmokymą naudojant pakankamai daug (apie 10000) įėjimo-išėjimo duomenų reikšmių.

- Tyrimo metu pastebėta, kad DFNN yra gana jautrus parinkinėjamoms neuronų vėlinimo reikšmėms ir nustatinėjamam neuronų kiekiui paslėptame tinklo sluoksnyje. Padidinus ar sumažinus vieną iš vėlinimo koeficientų viena reikšme galima gauti keliais šimtais kartų prastesnius rezultatus, todėl šių koeficientų parinkimas yra gana komplikuotas.

6 Literatūra

[1] P.Mastorocostas, D.Versamis, C.Mastorocostas, C.Hilas, „A Recurrent Fuzzy Filter for Adaptive Noise Cancellation“, 2005, IEEE.

[2] J.-S.R. Jang, C.-T. SUN, and E.Mizutani, „Neuro-Fuzzy and Soft Compiuting“, Prentice Hall, 1997.

[3] P.Mastorocostas, J.B. Theocharis, „A Recurrent Fuzzy-Neural Model for Dynamic System Identification“, 2002, IEEE.

[4] P. Frasconi, M. Gori, and G. Soda, “Local feedback multilayered networks”, Neural Comput., vol. 4, 1992.

[5] A.C. Tsoi and A. D. Back, “Locally recurrent globally feedforward networks: A critical review of architectures”, IEEE, Neural Networks, vol.5, 229-239, 1994.

[6] P.Mastorocostas, J.B. Theocharis, „FUNCOM: An Efficient Fuzzy Neural Training Algorithm“, IEEE, 1996.

[7] Stephen L. Chiu, „A Cluster Estimation Method with Extension to Fuzzy Model Identification“, IEEE, 1994.

[8] R. Yager and D. Filev, “Approximate clustering via the mountain method”, IEEE Trans. on Systems, Man & Cybernetics, 1992.

[9] Weina Wang, Yunjie Zhang, Yi Li, Xiaona Zhang, „The Global Fuzzy C-Means Clustering Algorithm“, IEEE, 2006.

[10] J.Dunn, “A fuzzy relative of the ISODATA process and its use in detecting compact, well separated clusters”, J. Cybernetics, 1974.

[11] B. Widrow. et al, “Adaptive Noise Cancellation: Principles and Applications”, IEEE, 1975.

[12] W. T. Miller, R. S. Sutton, and P. J. Werbos, “Neural Networks for control”, Cambridge, 1990.

[13] K. S. Narendra and K. Parthasarathy, “Identification and control of dynamical systems using neural networks”, IEEE, 1990.

[14] V. Jukavičius, “Construction of noise reduction system”, Kaunas University of Technology, 2006.

[15] O. Agazzi, D. G. Messerschmitt and D. A. Hodges, “Nonlinear echo cancellation of data signals”, IEEE, 1982.

[16] S.-Z. Qin, H.-T. Su and T. J. McAvoy, “Comparison of four neural net learning methods for dynamic system identification”, IEEE, 1992.

[17] E. H. Mamdani and S. Assilian, “An experiment in linguistic synthesis with a fuzzy logic controller”, International Journal of Man-Machine Studies, 1975.

[18] M. Sugeno and G. T. Kang, “Structure identification of fuzzy model”, Fuzzy Sets and Systems, 1988.

[19] M. Sugeno and T.Takagi, “Fuzzy identification of systems and its application to modeling and control”, IEEE, 1985.

[20] A. Weibel, T. Hanazawa, G. Hinton, K. Shikano and K. J. Lang, “Phoneme recognition using time-delay neural networks”, IEEE, 1989.

[21] R. J. Williams and D. Zipser, “A learning algorithm for continually running fully recurrent neural networks”, Neural Computation, 1989.

[22] A.C. Tsoi and A. D. Back, “FIR and IIR synapses, a new neural network architecture for time series modeling”, Neural Computation, 1991.

[23] M. Gori, Y. Bengio and R. D. Mori, “BDS: A learning algorithm for capturing the dynamic nature speech”, Neural Networks, 1989.

7 Priedai

7.1 Priedas Nr. 1 DFNN struktūros identifikavimo metodikos programinė realizacija

```
%**************************************************************************
%*** DFNN struktūros identifikavimo metodikos realizacija            ***
%*** IFM-2/1 Vaidas Jukavičius                                       ***
%**************************************************************************

clc;
close all;
clear all;

% Duomenys

[x, fs, bits] = wavread('Noise.wav');     % Блjimo
[d, fs, bits] = wavread('Noise_d.wav');   % Ірлjimo

t = (0:1/fs:(length(x)-1)/fs);

% Duomenys duomenų grupavimui

x1 = x(1:10000);
d1 = d(1:10000);

% Atvaizduojami duomenys

t = (0:1/fs:(length(x1)-1)/fs);

figure(1);
plot(t,x1);
xlabel('Laikas,s');
ylabel('Amplitude,mV');
axis([0 max(t) (-max(x1)-0.05) (max(x1)+0.05)]);

figure(2);
plot(t,d1);
xlabel('Laikas,s');
ylabel('Amplitude,mV');
axis([0 max(t) (-max(x1)-0.05) (max(x1)+0.05)]);

% Duomenų grupavimas

[Koord Pot Kiekis] = Clustering(x1,d1,0.25,1.5,0.5,0.15);

% Gaunamos narystės funkcijos

mu = zeros (Kiekis, length(x1));
dev = std(x1)/2;

for i = 1:Kiekis
    for j = 1:length(x1)
        mu(i,j) = exp((-1/2)*((x1(j)-Koord(1,i))^2)/(dev^2));
    end
end

figure(3);
hold on;
plot(x1,d1,'.');
plot(Koord(1,:),Koord(2,:),'ro','MarkerFaceColor','r');
xlabel('Iejimo duomenu reiksmes');
ylabel('Isejimo duomenu reiksmes');
hold off;


[xs idex] = sort(x1);
for j = 1:Kiekis
for i = 1:length(x1)
    mus(j,i) = mu(j,idex(i));
end
end

figure(4);
hold on;
```

```
for i = 1:Kiekis
    plot(xs,mus(i,:));
end
for i = 1:Kiekis
    plot(Koord(1,i),0,'blacko','MarkerFaceColor','r');
end
axis([-0.4 0.4 0 1]);
xlabel('Iejimo duomenu reiksmes');
ylabel('Narystes laipsniai');
hold off;


% Gaunamų grupių centrų optimizavimas

stulp = sum(mu);
mu = mu./stulp(ones(Kiekis, 1), :);
op = [2 1000 1e-5];

[Centrai muu iter] = fuzzycm(x1,d1,Kiekis,mu,op);

% Gaunamų grupių atvaizdavimas

figure(5);
hold on;

maxMU = max(muu);

sn = zeros(Kiekis,1);

for i = 1:Kiekis
    index = find(muu(i,:) == maxMU);
    plot(x1(index),d1(index),'o','MarkerFaceColor',[rand() rand() rand()]);
    sn(i) = std(x1(index));
end

for i = 1:Kiekis
    plot(Centrai(i,1),Centrai(i,2),'o','MarkerFaceColor','r');
end

xlabel('Iejimo duomenu reiksmes');
ylabel('Isejimo duomenu reiksmes');
hold off;

% Apskaiciuojamos galutines narystės funkcijos

xd = [-0.4:0.001:0.4];
md = zeros(Kiekis,length(xd));

for i = 1:Kiekis
    for j = 1:length(xd)
        md(i,j) = exp((-1/2)*((xd(j)-Centrai(i,1))^2)/(sn(i)^2));
    end
end

% Galutinių narystės funkcijų atvaizdavimas

figure(4);
hold on;
xlabel('Iejimo duomenu reiksmes');
ylabel('Narystes laipsniai');
for i = 1:Kiekis
    plot(xd,md(i,:));
end

for i = 1:Kiekis
    plot(Centrai(i,1),0,'blacko','MarkerFaceColor','r');
end

hold off;
```

7.2 Priedas Nr. 2 Duomenų grupavimo metodo programinė realizacija

```
%**************************************************************************
%*** Duomenų grupavimo metodo programinė realizacija                    ***
%*** IFM-2/1 Vaidas Jukavičius                                          ***
%**************************************************************************


function [Koordinates Potencialai Kiekis] = Clustering(y,z,ra,rb_konstanta,ev,ea)

% y,z - taskų koordinates
% ra - tasko potencialo skaičiavimo spindulys
% rb_konstanta - grupės taskų potencialų mažinimo konstanta
% ev - grupės centro priėmimo santykis
% ea - grupės centro atmetimo santykis


% Normalizavimas

miny = min(y);
minz = min(z);
maxy = max(y);
maxz = max(z);

y(:) = (y(:) - miny) / (maxy - miny);
z(:) = (z(:) - minz) / (maxz - minz);

% Grupavimo metodas

% Konstantos
rb = rb_konstanta * ra;
alfa = 4 / ra^2;
beta = 4 / rb^2;

Koordinates = zeros (2,1);
Potencialai = zeros (1);

P = zeros(1,length(y));

% Paskaiciuojami visu taskų potencialai

for i = 1:length(y)
    suma = 0;
    for j = 1:length(y)
        if (i~=j)
            suma = suma + exp(-alfa*(((z(i)-z(j))^2)+((y(i)-y(j))^2)));
        else
            suma = suma + 1;
        end
    end
    P(i) = suma;
end

% Randamas pirmasis centras

[k,l] = max(P);
Kiekis = 1;

Potencialai(Kiekis) = k;
Koordinates(1,Kiekis) = y(l);
Koordinates(2,Kiekis) = z(l);

% Perskaičiuojami potencialai, randami sekantys centrai

perskaiciuot = 1;
pabaiga = 1;

while pabaiga

    if perskaiciuot == 1
        for i = 1:length(y)
            P(i) = P(i) - Potencialai(Kiekis)*exp(-beta*(((z(i)-z(l))^2)+((y(i)-y(l))^2)));
            if P(i)<0
                P(i)=0;
            end
        end
    end
```

```
    [k,l] = max(P);


    if P(l) > (ev*Potencialai(1))
        Kiekis = Kiekis + 1;

        Potencialai(Kiekis) = k;
        Koordinates(1,Kiekis) = y(l);
        Koordinates(2,Kiekis) = z(l);

        perskaiciuot = 1;

    else if P(l) < (ea*Potencialai(1))

            pabaiga = 0;

        else

            d = 1000000;

            for j = 1:Kiekis
                ats = (((z(l)-Koordinates(2,j))^2)+((y(l)-Koordinates(1,j))^2));
                if ats < d
                    d = ats;
                end
            end

            if ((sqrt(d))/ra + P(l)/Potencialai(1)) >=1
                Kiekis = Kiekis + 1;

                Potencialai(Kiekis) = k;
                Koordinates(1,Kiekis) = y(l);
                Koordinates(2,Kiekis) = z(l);

                perskaiciuot = 1;

            else
                P(l) = 0;
                perskaiciuot = 0;
            end
        end
    end
end

% Perskaičiuojamos koordinates

for j = 1:Kiekis
    Koordinates(1,j) = (Koordinates(1,j) * (maxy - miny)) + miny;
    Koordinates(2,j) = (Koordinates(2,j) * (maxz - minz)) + minz;
end
```

7.3 Priedas Nr. 3 C-vidurkių optimizavimo metodo programinė realizacija

```
%*********************************************************************
%*** C-vidurkių optimizavimo metodo programinė realizacija          ***
%*** IFM-2/1 Vaidas Jukavičius                                      ***
%*********************************************************************

function [Koordinates mu i J] = fuzzycm(x,y,n,mu,options)

m = options(1);
iter = options(2);
impro = options(3);
data = [x,y];

J = zeros(iter,1);

for i = 1:iter

    % Optimizavimo žingsnis
```

```
    nf = mu.^m;
    Koordinates = nf*data./((ones(size(data, 2), 1)*sum(nf'))');

    % Skaičiuojami atstumai tarp duomenų ir centrų koordinačių

    for k = 1:n
        dist(k, :) = sqrt(sum(((data-ones(size(data, 1), 1)*Koordinates(k, :)).^2)'));
    end

    % Skaičiuojama optimizuojamos funkcijos reikšmė

    J(i) = sum(sum((dist.^m).*nf));

    % Skaičiuojami narystės laipsniai

    tmp = dist.^(-2/(m-1));

    mu = tmp./(ones(n, 1)*sum(tmp));

    % Tikrinama pabaigos sąlyga

    if i > 1
       if abs(J(i)-J(i-1)) < impro
           break;
       end
    end

end

J = J(i);
```

7.4 Priedas Nr. 4 DFNN filtro programinė realizacija

```
%**************************************************************************
%*** DFNN filtro programinė realizacija                                 ***
%*** IFM-2/1 Vaidas Jukavičius                                         ***
%**************************************************************************

clc;
close all;
clear all;

% Duomenys

[x, fs, bits] = wavread('Noise.wav');
[d, fs, bits] = wavread('Noise_d.wav');

% Apmokymo duomenys

x1 = x(1000:1020);
d1 = d(1000:1020);

figure(1)
plot(x1);
xlabel('Iejimo duomenys');
ylabel('Amplitude,V');

% Filtravimo duomenys

x2 = x(1000:1100);
y2 = d(1000:1100);

% Duomenų grupių centrai

sn = [0.0994 0.0717 0.2624];
centrai = [-0.239 0.0086 0.2624];

% Taisyklių kiekis

n = 3;
```

```
% Įėjimų skaičius

m = 1;

% Neuronų kiekis pasleptam sluoksnį

neuron_n = 2;

% Paslepto sluoksnio neuronų išėjimo masyvas

O = zeros(1,(n*neuron_n));

% Išvadų dalies išėjimai

Y = zeros(1,n);

% Laiko vėlinimai

Ov = [2 5 2 5];

% Svorių masyvai

w1 = zeros(1,(m*n*neuron_n*(Ov(1)+1)));   % Paslėptų sluoksnių neuronų
w2 = zeros(1,(n*neuron_n*Ov(2)));
w3 = zeros(1,(n*neuron_n));

w4 = zeros(1,(n*neuron_n*(Ov(3)+1)));     % Išėjimo neuronų
w5 = zeros(1,(n*Ov(4)));
w6 = zeros(1,n);

% Parametrų skaičius

parametru = length(w1)+length(w2)+length(w3)+length(w4)+length(w5)+length(w6)

W = zeros(6,length(w1)+1);

W(1,1) = length(w1);
W(1,2:length(w1)+1) = w1;

W(2,1) = length(w2);
W(2,2:length(w2)+1) = w2;

W(3,1) = length(w3);
W(3,2:length(w3)+1) = w3;

W(4,1) = length(w4);
W(4,2:length(w4)+1) = w4;

W(5,1) = length(w5);
W(5,2:length(w5)+1) = w5;

W(6,1) = length(w6);
W(6,2:length(w6)+1) = w6;

% Apmokymo procesas

[W,yf] = Apsimokymas(x1,d1,0.1,W,n,neuron_n,1,Ov,10000,sn,centrai);

ilgis = int16(W(1,1));
w1 = W(1,2:ilgis+1);
ilgis = int16(W(2,1));
w2 = W(2,2:ilgis+1);
ilgis = int16(W(3,1));
w3 = W(3,2:ilgis+1);
ilgis = int16(W(4,1));
w4 = W(4,2:ilgis+1);
ilgis = int16(W(5,1));
w5 = W(5,2:ilgis+1);
ilgis = int16(W(6,1));
w6 = W(6,2:ilgis+1);


OO = zeros((n*neuron_n),Ov(2));
OOO = zeros((n*neuron_n),(Ov(3)+1));
YY = zeros(n,Ov(4));

% Išvadų dalies apskaičiavimas

for d = 1:length(x2)
```

```
    for i = 1:n                          % per taisykles

        % per paslėptus neuronus

        for j = 1:neuron_n

            suma = 0;

            for ii = 1:m
                for jj = 1:(Ov(1)+1)
                    if (d>=jj)
                    suma = suma + w1((((Ov(1)+1)*m*neuron_n)*(i-1))+(((Ov(1)+1)*m)*(j-
1))+((Ov(1)+1)*(ii-1))+jj)*(x2(d+1-jj));
                    end
                end
            end

            for ii = 1:Ov(2)
                if (d>ii)
                suma = suma + w2(((Ov(2)*neuron_n)*(i-1))+(Ov(2)*(neuron_n-
1))+ii)*OO(((neuron_n*(i-1))+j),ii);
                end
            end

            suma = suma + w3((neuron_n*(i-1))+j);

            O((neuron_n*(i-1))+j) = tanh(suma);

            for k = Ov(2):-1:2
                OO(((neuron_n*(i-1))+j),k) = OO(((neuron_n*(i-1))+j),(k-1));
            end

            OO(((neuron_n*(i-1))+j),1) = O((neuron_n*(i-1))+j);

            for k = (Ov(3)+1):-1:2
                OOO(((neuron_n*(i-1))+j),k) = OOO(((neuron_n*(i-1))+j),(k-1));
            end

            OOO(((neuron_n*(i-1))+j),1) = O((neuron_n*(i-1))+j);

        end
    end



    % Per išėjimo neuronus

    for i = 1:n

         suma = 0;

         for ii = 1:neuron_n
                for jj = 1:(Ov(3)+1)
                    if (d>=jj)
                    suma = suma + (w4((((Ov(3)+1)*neuron_n)*(i-1))+((Ov(3)+1)*(ii-1))+jj)) *
OOO(((neuron_n*(i-1))+ii),jj);
                    end
                end
         end

         for ii = 1:Ov(4)
             if (d>ii)
                suma = suma + w5((Ov(4)*(i-1))+ii)*YY(i,ii);
             end
         end

         suma = suma + w6(i);

         Y(i) = tanh(suma);

         for k = Ov(4):-1:2
             YY(i,k) = YY(i,(k-1));
         end

         YY(i,1) = Y(i);
    end


% Išėjimo apskaičiavimas
```

```
NL = zeros(1,n);

for j = 1:n
        NL(j) = exp((-1/2)*((x2(d)-centrai(j))^2)/(sn(j)^2));
end

suma = 0;

for j = 1:n
        suma = suma + NL(j) * Y(j);
end

y_gautas(d) = suma / sum(NL);



end

s = 0;
for i =1 : length(y2)
s = s+ (y_gautas(i) - y2(i))^2;
end
Error = s/length(y2)

% Filtravimo rezultatų atvaizdavimas

figure(2)
hold on;
plot(y2,'b');
plot(y_gautas,'r--');
h = legend('Tikrasis','Nustatytas',2);
set(h,'Location','NorthEast');
set(h,'Interpreter','none');
xlabel('Isejimo duomenys');
ylabel('Amplitude,V');
hold off;
```

7.5 Priedas Nr. 5 Apmokymo metodo programinė realizacija

```
%************************************************************************
%*** Apsimokymo metodo programinė realizacija                         ***
%*** IFM-2/1 Vaidas Jukavičius                                        ***
%************************************************************************

function [Wbu,yf] = Apsimokymas (x,y, delta_p, W, n, neuron_n,m,Ov,kiekis,sn,centrai)

% Parametrų keitimo matrica

delta = diag(ones(6,1)*delta_p,0);

% Inicializuojami koeficientai

[a b] = size(W);
Wbu = W;

for i = 1:a
    for j = 2:b
        Wes(i,j) = randn(1);
    end
end

da = max(max(abs(Wes)));
da = da/0.5;

Wes = Wes./da;

for i = 1:a
    Wes(i,1) = W(i,1);
end

lemda_fi = zeros (6,b);
lemda_e = zeros (6,b);
pok = zeros (6,b);

for i = 1:6
```

```
    lemda_fi(i,1) = W(i,1);
    lemda_e(i,1) = W(i,1);
    pok(i,1) = W(i,1);
end

[kf f] = size(x);

lemda_1 = zeros(kf, n*neuron_n);
lemda_2 = zeros(kf, n);

E_dabar = [0 0 0 0 0 0];
didinimo = 1.2;
mazinimo = 0.8;


NL = zeros(kf,n);    % Narystės laipsnių apskaičiavimas

for i = 1:kf
    for j = 1:n
        NL(i,j) = exp((-1/2)*((x(i)-centrai(j))^2)/(sn(j)^2));
    end
end

% Apmokymo pradžia

for t = 1:kiekis

% Tinklo išėjimų apskaičiavimas

IS = zeros (kf, ((neuron_n*n)+n));
ISVEST = zeros (kf, ((neuron_n*n)+n));
ISVESTT = zeros (kf, ((neuron_n*n)+n));

for k = 1:kf

    for i = 1:n

        for j = 1:neuron_n

            % Skaičiuojamas neurono išėjimas

            suma = 0;

            for ii = 1:m
                for jj = 1:(Ov(1)+1)
                    if (k>=jj)
                        suma = suma + Wes(1,1+((((Ov(1)+1)*m*neuron_n)*(i-
1))+(((Ov(1)+1)*m)*(j-1))+((Ov(1)+1)*(ii-1))+jj))*(x(k+1-jj));
                    end
                end
            end

            for ii = 1:Ov(2)
                if (k>ii)
                suma = suma + Wes(2,1+(((Ov(2)*neuron_n)*(i-1))+(Ov(2)*(j-1))+ii))*IS(k -
ii,((neuron_n*(i-1))+j));
                end
            end

            suma = suma + Wes(3,1+((neuron_n*(i-1))+j));

            % Neurono išėjimas

            IS(k,((neuron_n*(i-1))+j)) = tanh(suma);

            % Išėjimo išvestinės

            ISVEST(k,((neuron_n*(i-1))+j)) = 1 - (tanh(suma))^2;

        end
    end



    for i = 1:n

         suma = 0;

         for ii = 1:neuron_n
                for jj = 1:(Ov(3)+1)
```

```
                    if (k>=jj)
                    suma = suma + Wes(4,1+((((Ov(3)+1)*neuron_n)*(i-1))+((Ov(3)+1)*(ii-
1))+jj))*IS((k+1-jj),((neuron_n*(i-1))+ii));
                    end
                end
        end

        for ii = 1:Ov(4)
            if (k>ii)
               suma = suma + Wes(5,1+((Ov(4)*(i-1))+ii))*IS((k-ii),((neuron_n*n)+i));
            end
        end

        suma = suma + Wes(6,1+i);

        % Išėjimas

        IS(k,((neuron_n*n)+i)) = tanh(suma);
        sumaaaa = suma;

        % Išėjimo išvestinė

        ISVEST(k,((neuron_n*n)+i)) = 1 - (tanh(suma))^2;


    end
end



% Filtro išėjimų paskaičiavimas

yf = zeros(kf,1);

for i = 1:kf
    suma = 0;
    for j = 1:n
        suma = suma + (NL(i,j) * IS(i,(neuron_n*n+j)));
    end
    yf(i) = suma / sum(NL(i,:));
end

suma = 0;

for i = 1:kf
    suma = suma + (yf(i)-y(i))^2;
end

Eror = 1/kf*suma;


% MPC adaptacija
% Dalinės išvestinės skaičiavimai

E_pirmai = E_dabar;

%Pagal šeštą
summ = 0;
for i = 1:kf
    suma = 0;
    for j = 1:n
        suma = suma + (NL(i,j) * ISVEST(i,(neuron_n*n+j)));
    end
    daug = suma / sum(NL(i,:));
    summ = summ + (yf(i) - y(i))*daug;
end

E_dabar(6) = 2/kf * summ;

%Pagal penktą

summ = 0;
for i = 1:kf
    suma = 0;
    for j = 1:n
        su = 0;
        for jj = 1:Ov(4)
            if (i>jj)
                su = su + IS((i-jj),((neuron_n*n)+j)) ;
```

```
            end
        end
        suma = suma + (NL(i,j)* ISVEST(i,(neuron_n*n+j))) * su;
    end
    daug = suma / sum(NL(i,:));
    summ = summ + (yf(i) - y(i))*daug;
end

E_dabar(5) = 2/kf * summ;

%Pagal ketvirtą

summ = 0;
for i = 1:kf
    suma = 0;
    for j = 1:n
        su = 0;
        for ii = 1:neuron_n
                for jj = 1:(Ov(3)+1)
                    if (i>=jj)
                        su = su + IS((i+1-jj),((neuron_n*(j-1))+ii)) ;

                    end
                end
        end
        suma = suma + (NL(i,j)* ISVEST(i,(neuron_n*n+j)))  * su;
    end
    daug = suma / sum(NL(i,:));
    summ = summ + (yf(i) - y(i))*daug;
end

E_dabar(4) = 2/kf * summ;

%Pagal trecią

summ = 0;
for i = 1:kf
    suma = 0;
    for j = 1:n
        su = 0;
        for ii = 1:neuron_n
            for jj = 1:(Ov(3)+1)
                    if (i>=jj)
                    su = su + Wes(4,1+((((Ov(3)+1)*neuron_n)*(j-1))+((Ov(3)+1)*(ii-
1))+jj))*ISVEST((i+1-jj),((neuron_n*(j-1))+ii));

                    end
            end
        end
        suma = suma + (NL(i,j) * ISVEST(i,(neuron_n*n+j))) * su;
    end
    daug = suma / sum(NL(i,:));
    summ = summ + (yf(i) - y(i))*daug;
end

E_dabar(3) = 2/kf * summ;

%Pagal antrą

summ = 0;
for i = 1:kf
    suma = 0;
    for j = 1:n
        su = 0;
        for ii = 1:neuron_n
            for jj = 1:(Ov(3)+1)
                    if (i>=jj)
                        s = 0;
                        for iii = 1:Ov(2)
                            if (i>iii)
                                s = s + IS(i - iii,((neuron_n*(j-1))+ii));

                            end
                        end
                    su = su + Wes(4,1+((((Ov(3)+1)*neuron_n)*(j-1))+((Ov(3)+1)*(ii-
1))+jj))*ISVEST((i+1-jj),((neuron_n*(j-1))+ii)) * s;
                    end
            end
        end
        suma = suma + (NL(i,j) * ISVEST(i,(neuron_n*n+j))) * su;
```

```
    end
    daug = suma / sum(NL(i,:));
    summ = summ + (yf(i) - y(i))*daug;
end

E_dabar(2) = 2/kf * summ;

%Pagal pirmą

summ = 0;
for i = 1:kf
    suma = 0;
    for j = 1:n
        su = 0;
        for ii = 1:neuron_n
            for jj = 1:(Ov(3)+1)
                    if (i>=jj)
                        s = 0;
                        for iii = 1:Ov(1)+1
                            if (i>=iii)
                                s = s + (x(i+1-iii));

                            end
                        end
                    su = su + Wes(4,1+((((Ov(3)+1)*neuron_n)*(j-1))+((Ov(3)+1)*(ii-
1))+jj))*ISVEST((i+1-jj),((neuron_n*(j-1))+ii)) * s;
                    end
            end
        end
        suma = suma + (NL(i,j) * ISVEST(i,(neuron_n*n+j))) * su;
    end
    daug = suma / sum(NL(i,:));
    summ = summ + (yf(i) - y(i))*daug;
end

E_dabar(1) = 2/kf * summ;

for i = 1:6
if ((E_dabar(i)*E_pirmai(i))>0)
    delta(i,i) = min (delta(i,i)*didinimo, 20);
end
if ((E_dabar(i)*E_pirmai(i))<0)
    delta(i,i) = max (delta(i,i)*mazinimo, 0.0001);
end
end




% Žingsnio apskaičiavimas


% Lagranžo daugiklių apskaičiavimas

for i = 1:n
    lemda_2(kf,i) = 2/kf * (yf(kf) - y(kf)) * NL(kf,i) / sum(NL(kf,:));
end

for i = 1:n
    for j = 1:neuron_n
        lemda_1(kf,((neuron_n*(i-1))+j)) = lemda_2(kf,i) * ISVEST(kf,((neuron_n*n)+i)) *
Wes(4,1+((((Ov(3)+1)*neuron_n)*(i-1))+((Ov(3)+1)*(j-1))+1));
    end
end

% Sekancių daugiklių apskaičiavimas

for k = (kf-1):-1:1

    % Lemda_2

    for i =1:n
        suma = 0;
        for j = 1:Ov(4)
            if (k+j<=kf)
                suma = suma + lemda_2(k+j,i) * ISVEST(k+j,((neuron_n*n)+i)) *
Wes(5,1+((Ov(4)*(i-1))+j));
            end
        end
```

```
        lemda_2(k,i) = 2/kf * (yf(k) - y(k)) * NL(k,i) / sum(NL(k,:)) + suma;
    end

    % Lemda_1

    for i = 1:n
        for j = 1:neuron_n
            suma1 = 0;
            for l = 1:Ov(2)
                if (k+l<=kf)
                    suma1 = suma1 + lemda_1(k+l,(((neuron_n*(i-1))+j))) *
ISVEST(k+l,((neuron_n*(i-1))+j)) * Wes(2,1+(((Ov(2)*neuron_n)*(i-1))+(Ov(2)*(j-1))+l));
                end
            end

            suma2 = 0;
            for q = 1:Ov(3)+1
                if (k+q<=kf)
                    suma2 = suma2 + lemda_2(k+q,i) * ISVEST(k+q,i) *
Wes(4,1+((((Ov(3)+1)*neuron_n)*(i-1))+((Ov(3)+1)*(j-1))+q));
                end
            end

            lemda_1(k,((neuron_n*(i-1))+j)) = suma1 + suma2;
        end
    end

end



for i = 1:6
    lemda_fi(i,2:lemda_fi(i,1)+1) = 2 * (Wes(i,2:lemda_fi(i,1)+1) - Wbu(i,2:lemda_fi(i,1)+1));
end

I_fifi = 0;

for i = 1:6
    I_fifi = I_fifi + (lemda_fi(i,2:lemda_fi(i,1)+1) * lemda_fi(i,2:lemda_fi(i,1)+1)') *
(delta(i,:)*delta(i,:)');
end

for i = 1:n
    for j = 1:neuron_n
        for q = 1:Ov(1)+1
            suma=0;

            for k = 1:kf
                if (k>=q)
                suma = suma + lemda_1(k,(((neuron_n*(i-1))+j))) * ISVEST(k,((neuron_n*(i-
1))+j)) * x(k+1-q);
                end
            end

            lemda_e(1,1+((((Ov(1)+1)*neuron_n)*(i-1))+((Ov(1)+1)*(j-1))+q)) = suma;
        end
    end
end

for i = 1:n
    for j = 1:neuron_n
        for q = 1:Ov(2)
            suma=0;

            for k = 1:kf
                if (k>q)
                suma = suma + lemda_1(k,(((neuron_n*(i-1))+j))) * ISVEST(k,((neuron_n*(i-
1))+j)) * IS(k - q,((neuron_n*(i-1))+j));
                end
            end

            lemda_e(2,1+(((Ov(2)*neuron_n)*(i-1))+(Ov(2)*(j-1))+q)) = suma;
        end
    end
end


for i = 1:n
    for j = 1:neuron_n
            suma=0;
```

```
            for k = 1:kf
                suma = suma + lemda_1(k,(((neuron_n*(i-1))+j))) * ISVEST(k,((neuron_n*(i-
1))+j));
            end

            lemda_e(3,1+((neuron_n*(i-1))+j)) = suma;
    end
end



for i = 1:n
    for j = 1:neuron_n
        for q = 1:Ov(3)+1
            suma = 0;
            for k = 1:kf
                if (k>=q)
                    suma = suma + lemda_2(k,i) * ISVEST(k,(neuron_n*n)+i) * IS((k+1-
q),((neuron_n*(i-1))+j));
                end
            end
            lemda_e(4,1+((((Ov(3)+1)*neuron_n)*(i-1))+((Ov(3)+1)*(j-1))+q)) = suma;
        end
    end
end

for i = 1:n
        for q = 1:Ov(4)
            suma = 0;
            for k = 1:kf
                if (k>q)
                    suma = suma + lemda_2(k,i) * ISVEST(k,(neuron_n*n)+i) * IS((k-
q),((neuron_n*n)+i));
                end
            end
            lemda_e(5,1+((Ov(4)*(i-1))+q)) = suma;
        end
end

for i = 1:n
    suma = 0;
    for k = 1:kf
        suma =suma + lemda_2(k,i) * ISVEST(k,(neuron_n*n)+i);
    end
    lemda_e(6,1+i) = suma;
end


I_ee = 0;

for i = 1:6
    I_ee = I_ee + (lemda_e(i,2:lemda_e(i,1)+1) * lemda_e(i,2:lemda_e(i,1)+1)') *
(delta(i,:)*delta(i,:)');
end

if I_ee ==0
    break;

end

I_efi = 0;


for i = 1:6
    I_efi = I_efi + (lemda_e(i,2:lemda_e(i,1)+1) * lemda_fi(i,2:lemda_fi(i,1)+1)') *
(delta(i,:)*delta(i,:)');
end

dE = - 0.85 * sqrt(I_ee);

% Pokyčių paskaičiavimas

for i = 1:6
    ilgis = int16(lemda_fi(i,1));
    kvad = (delta(i,:)*delta(i,:)');
    saknis = sqrt((I_ee-(dE^2)) / (I_fifi*I_ee - (I_efi^2)));
    skirt = lemda_fi(i,2:ilgis+1) - (lemda_e(i,2:ilgis+1).*I_efi/I_ee);
```

```
    pok(i,2:ilgis+1) = (skirt.* ( -kvad * saknis )) + (kvad * lemda_e(i,2:ilgis+1).*
(dE/I_ee));

end

for i = 1:6
    ilgis = int16(Wes(i,1));
    Wbu(i,2:ilgis+1) = Wes(i,2:ilgis+1);
end

% Naujų koeficientų paskaičiavimas

for i = 1:6
    ilgis = int16(Wes(i,1));
    Wes(i,2:ilgis+1) = Wes(i,2:ilgis+1) + pok(i,2:ilgis+1);
end
end

% Atvaizduojami rezultatai

figure(1)
hold on;
plot(y,'b');
plot(yf,'r--');
h = legend('Tikrasis','Nustatytas',2);
set(h,'Location','NorthEast');
set(h,'Interpreter','none');
xlabel('Isejimo duomenys');
ylabel('Amplitude,V');
hold off;
```