





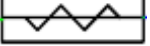
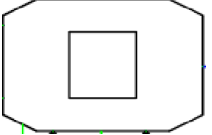
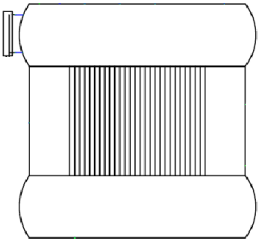








н/п	Зображення елемента	Опис
1		Вентилятор осьового типу
2		Асинхронний двигун змінного струму
3		Керуючий клапан з електроприводом
4		Теплообмінник
5		Захисний підривний клапан
6		Вентиль
7		Шнековий транспортер
8		Спалювач (пальник)
9		Котлоагрегат
10		Ділянка трубопроводу загального призначення
11		Контрольний сигнал
12		Давач температури спалювача зони 1
13		Давач температури спалювача зони 2
14		Давач температури спалювача зони 3
15		Давач температури димових газів «після» економайзера

16		Давач температури води «після» економайзера
17		Давач температури води «до» економайзера
18		Давач температури димових газів «до» економайзера
19		Давач температури димових газів в димоході
20		Давач надлишкового тиску пари в котлоагрегаті
21		Давач диференційного тиску рівня води в верхньому барабані
22		Давач диференційного тиску розрідження в топці котлоагрегату
23		Частотний перетворювач регулювання обертами двигуна асинхронного струму шнеку подачі 1
24		Частотний перетворювач регулювання обертами двигуна асинхронного струму шнеку подачі 2
25		Частотний перетворювач регулювання обертами двигуна асинхронного струму вентилятора подачі повітря в топку 1
26		Частотний перетворювач регулювання обертами двигуна асинхронного струму вентилятора подачі повітря в топку 2
27		Частотний перетворювач для регулювання обертами двигуна асинхронного струму димососа
28		Програмований логічний контролер закритого доступу
29		Людино-машинний інтерфейс відкритого типу

Додаток А

Умовні графічні позначення структурної схеми котла

Додаток Б
Програмний код ПІД-регулятора

FUNCTION_BLOCK PID_C

VAR_INPUT

(*System section*)

iNum:WORD; (*Номер регулятора*)

(*Block section*)

bRegDirect: BOOL; (*Напряв роботи регулятора*)

Q_Min:REAL; (*Обмеження виходу*)

Q_Max:REAL;

END_VAR

VAR_IN_OUT

(*System section*)

COM_SA:DWORD; (*commArea Start pointer*)

DB:pidStruct;

(*Block section*)

PV:REAL;(*Physical input*)

Q:REAL; (*Physical output*)

END_VAR

VAR_OUTPUT

END_VAR

VAR

(*System section*)

COM_OpSA:DWORD;(*Start adress*)

COM_OpShA:DWORD;(*Shifted Address*)

DB_OpSA: DWORD; (*start element in global data base*)

DB_OpShA:DWORD;(*database shifted address*)

StructSize:DWORD; (*Struct size*)

(*Block section*)

error: REAL := 0.0;

PID_Sum: REAL := 0.0;

Saturation: REAL := 0.0;

derivSum: REAL := 0.0; (**)

derivValue: REAL := 0.0; (**)

Integrator: REAL := 0.0; (**)

A:REAL;(*Safety*)

R_FRONT: R_TRIG;

F_FRONT: F_TRIG;

plcManOutFN: F_TRIG;

(*TEMP*)

tDw1:DWORD;

tDw2:DWORD;

tDw3:DWORD;

END_VAR

(*Секція копіювання*)

(*Розрахунок розміру блоку даних регуляторів*)

StructSize := SIZEOF(DB.Op);

(*Розрахунок комунікаційних адрес поточного регулятора*)

COM_OpSA := COM_SA; (*ADR(COM_SE);*)

COM_OpShA := COM_OpSA + StructSize * iNum;

DB_OpSA := ADR(DB.Op);

(**)

SysMemCpy (DB_OpSA, COM_OpShA, StructSize);

(*BLOCK STARTS HERE*)

(*#####*)

(*Секція роботи над статусами регулятора*)

(*Статус ручна робота*)

DB.Op.Sta.sta08.0 := DB.Op.Cmd.cmd08.0 (*manual =1,auto=0*);

(*Статус локальне завдання*)

DB.Op.Sta.sta08.1 := DB.Op.Cmd.cmd08.1;

(*Значення входу регулятора*)

DB.Op.Sta.PV := PV;

(*Опрацювання локальних завдань регулятора*)

IF DB.Op.Cmd.cmd08.1 (*SPLoc =1,SPAuto=0*) THEN

 DB.Op.Sta.SP := DB.Op.Cmd.LocSp;

ELSE

 DB.Op.Sta.SP := DB.PlcCmd.AutoSp;

END_IF

(*Визначення напрямку роботи регулятора прямий чи зворотній*)

IF bRegDirect THEN

 error := DB.Op.Sta.SP - DB.Op.Sta.PV ;

ELSE

 error := DB.Op.Sta.PV - DB.Op.Sta.SP ;

END_IF

(*Реалізація алгоритму без стресового переходу автоматичний-ручний і навпаки*)

R_FRONT (CLK := DB.Op.Cmd.cmd08.0 (*manual =1,auto=0*));

F_FRONT (CLK := DB.Op.Cmd.cmd08.0 (*manual =1,auto=0*));

plcManOutFN(CLK := DB.PlcCmd.cmd01.1(*Plc man LMN*));

IF R_FRONT.Q THEN

 DB.Op.Cmd.ManLMN := DB.Op.Sta.LMN;

ELSIF F_FRONT.Q THEN

 Integrator := DB.Op.Cmd.ManLMN - (error * DB.Op.Cmd.K);

ELSIF plcManOutFN.Q THEN

 Integrator := DB.PlcCmd.manLMN - (error * DB.Op.Cmd.K);

END_IF

IF DB.PlcCmd.cmd01.1 (*PLC manual =1,auto=0*) AND DB.Op.Cmd.cmd08.0
 (*manual =1,auto=0*) OR DB.Op.Cmd.cmd08.0 (*manual =1,auto=0*) THEN
 (*Ручна робота з панелі оператора*)

 PID_Sum := DB.Op.Cmd.ManLMN;

ELSIF DB.PlcCmd.cmd01.1 (*PLC manual =1,auto=0*) THEN

 (*Ручна робота регулятора*)

 PID_Sum := DB.PlcCmd.manLMN;

ELSIF NOT DB.PlcCmd.cmd01.1 AND NOT DB.Op.Cmd.cmd08.0 AND
 DB.PlcCmd.cmd01.0 THEN

 (*Визначення диференціалу і опрацювання суми регулятора*)

 derivValue := ((error * DB.Op.Cmd.D) - derivSum);

 derivSum := derivSum + derivValue;

 Integrator := Integrator + error * DB.Op.Cmd.I + (DB.Op.Sta.LMN -
 PID_Sum);

 PID_Sum := ((error * DB.Op.Cmd.K) + Integrator) + derivValue;

ELSE

 PID_Sum := 0;

END_IF

(*Обробка обмежень виходів*)

```
IF PID_Sum <= DB.Op.Cmd.LMN_Min THEN
    DB.Op.Sta.LMN := DB.Op.Cmd.LMN_Min;
ELSIF PID_Sum >= DB.Op.Cmd.LMN_Max THEN
    DB.Op.Sta.LMN := DB.Op.Cmd.LMN_Max;
ELSE
    DB.Op.Sta.LMN := PID_Sum;
END_IF
```

(*Масштабування фізичного виходу*)

```
A := DB.Op.Cmd.LMN_Max - DB.Op.Cmd.LMN_Min;
IF A <> 0 THEN
    Q := DB.Op.Sta.LMN * ((Q_Max - Q_Min) / A) + Q_Min;
ELSE
    Q:=0;
END_IF
```

(*Скид команди запуску ПІД*)

```
DB.PlcCmd.cmd01 := 0;
```

(*#####*)

(*Копіювання даних в область збереження даних регуляторів*)

```
SysMemCpy (COM_OpShA, DB_OpSA, StructSize);
```

Додаток В

Програмний код нейро-адаптивного блоку

```
%% Очищення змінних, підготування середовища
warning off;
clc;
clear all;
close all;

%% Конфігурація, параметри
dilD = 0;
dilK = 0.1;
dilI = 0.1;

%% Підготування моделі нейро-емулятора
open ('steamboiler_NN_3layers_01');
sim ('steamboiler_NN_3layers_01');
outputs = [0,0,0];

%% Конфігурація параметрів нейро-мережі
nbrOfNeuronsInEachHiddenLayer = [50 50 50];
nbrOfOutUnits = 3;
unipolarBipolarSelector = 0;

learningRate = 0.15;
nbrOfEpochs_max = 500000;

enable_resilient_gradient_descent = 1; %1 for enable, 0 for disable
learningRate_plus = 0.1;
learningRate_negative = 0.5;
deltas_start = 0.1;
deltas_min = 10^-6;
deltas_max = 50;

enable_decrease_learningRate = 0;
learningRate_decreaseValue = 0.01;
min_learningRate = 0.05;

enable_learningRate_momentum = 1;
momentum_alpha = 0.05;

draw_each_nbrOfEpochs = 500;

%% Зчитування даних, нормалізація даних
```



```

inDataSetMin = repmat(min(NN_Adapt_IN),length(NN_Adapt_IN(:,1)),1);
inDataSetMax = repmat(max(NN_Adapt_IN),length(NN_Adapt_IN(:,1)),1);

inDataSetNorm = (NN_Adapt_IN-inDataSetMin)./(inDataSetMax-inDataSetMin);

Samples = inDataSetNorm(:,[1 2 3]);

%% Розрахунок кількості вхідних і вихідних нодів
nbrOfInputNodes = length(Samples(1,:));
nbrOfLayers = 2 + length(nbrOfNeuronsInEachHiddenLayer);
nbrOfNodesPerLayer = [nbrOfInputNodes nbrOfNeuronsInEachHiddenLayer
nbrOfOutUnits];

%% Додання зміщення як нодів із 1
nbrOfNodesPerLayer(1:end-1) = nbrOfNodesPerLayer(1:end-1) + 1;
Samples = [ones(length(Samples(:,1)),1) Samples];

%% Цільові виходи
TargetOutputs = TargetClasses;

%% Ініціалізація важиків рандомом.
Weights = cell(1, nbrOfLayers);
Delta_Weights = cell(1, nbrOfLayers);
ResilientDeltas = Delta_Weights;

for i = 1:length(Weights)-1
    Weights{i} = 2*rand(nbrOfNodesPerLayer(i), nbrOfNodesPerLayer(i+1))-1;
    Weights{i}(:,1) = 0;    Delta_Weights{i} = zeros(nbrOfNodesPerLayer(i),
nbrOfNodesPerLayer(i+1));
    ResilientDeltas{i} = deltas_start*ones(nbrOfNodesPerLayer(i),
nbrOfNodesPerLayer(i+1));
end
Weights{end} = ones(nbrOfNodesPerLayer(end), 1);
Old_Delta_Weights_for_Momentum = Delta_Weights;
Old_Delta_Weights_for_Resilient = Delta_Weights;

NodesActivations = cell(1,nbrOfLayers);
for i = 1:length(NodesActivations)
    NodesActivations{i} = zeros(1,nbrOfNodesPerLayer(i));
end
NodesBackPropagatedErrors = NodesActivations;
zeroRMSReached = 0;
nbrOfEpochs_done = 0;

```

%% Ітерація даних

```
set_param('steamboiler_NN_3layers_01','SimulationCommand','start','SimulationCommand','pause');
```

```
for Epoch = 1:nbOfEpochs_max
```

```
set_param('steamboiler_NN_3layers_01','SimulationCommand','continue','SimulationCommand','pause');  
pause(0.01);
```

```
inDataSetNorm = (NN_Adapt_IN(end)-inDataSetMin)./(inDataSetMax-inDataSetMin);
```

```
Samples = inDataSetNorm(end,[1 2 3]);  
Samples = [ones(length(Samples(:,1)),1) Samples];  
TargetClasses = NN_Adapt_IN(end,4)*-1;  
TargetOutputs = inDataSetNorm(end,4)*-1;
```

```
for Sample = 1:length(Samples(:,1))  
    %% Зворотнє поширення помилки навчання  
    %%Прямий прохід  
    NodesActivations{1} = Samples(Sample,:);  
    for Layer = 2:nbOfLayers  
        NodesActivations{Layer} = NodesActivations{Layer-1}*Weights{Layer-1};  
        NodesActivations{Layer} = Activation_func(NodesActivations{Layer},  
unipolarBipolarSelector);  
        if (Layer ~= nbOfLayers)  
            NodesActivations{Layer}(1) = 1;  
        end  
    end  
  
    %% Зворотній прохід, збереження важиків  
    NodesBackPropagatedErrors{nbOfLayers} = TargetOutputs(Sample,:)-  
NodesActivations{nbOfLayers};  
    for Layer = nbOfLayers-1:-1:1  
        gradient = Activation_func_drev(NodesActivations{Layer+1},  
unipolarBipolarSelector);  
        for node=1:length(NodesBackPropagatedErrors{Layer})  
NodesBackPropagatedErrors{Layer}(node) =  
sum(NodesBackPropagatedErrors{Layer+1} .* gradient .* Weights{Layer}(node,:));  
        end  
    end  
end
```

```

% Зворотній прохід, розрахунок різниці важиків
for Layer = nrOfLayers:-1:2
    derivative = Activation_func_drev(NodesActivations{Layer},
unipolarBipolarSelector);
    Delta_Weights{Layer-1} = Delta_Weights{Layer-1} +
NodesActivations{Layer-1}' * (NodesBackPropagatedErrors{Layer} .* derivative);
end
end

%% Примінення градієнту до важиків.
if(enable_resilient_gradient_descent)
if(mod(Epoch,200)==0)
for Layer = 1:nrOfLayers
    ResilientDeltas{Layer} = learningRate*Delta_Weights{Layer};
end
end
for Layer = 1:nrOfLayers-1
    mult = Old_Delta_Weights_for_Resilient{Layer} .* Delta_Weights{Layer};
    ResilientDeltas{Layer}(mult > 0) = ResilientDeltas{Layer}(mult > 0) *
learningRate_plus;
    ResilientDeltas{Layer}(mult < 0) = ResilientDeltas{Layer}(mult < 0) *
learningRate_negative;
    ResilientDeltas{Layer} = max(deltas_min, ResilientDeltas{Layer});
    ResilientDeltas{Layer} = min(deltas_max, ResilientDeltas{Layer});

    Old_Delta_Weights_for_Resilient{Layer} = Delta_Weights{Layer};

    Delta_Weights{Layer} = sign(Delta_Weights{Layer}) .*
ResilientDeltas{Layer};
end
end
if(enable_learningRate_momentum)
for Layer = 1:nrOfLayers
    Delta_Weights{Layer} = learningRate*Delta_Weights{Layer} +
momentum_alpha*Old_Delta_Weights_for_Momentum{Layer};
end
Old_Delta_Weights_for_Momentum = Delta_Weights;
end
if(~enable_learningRate_momentum && ~enable_resilient_gradient_descent)
for Layer = 1:nrOfLayers
    Delta_Weights{Layer} = learningRate * Delta_Weights{Layer};
end
end
end

```

```

%% Зворотній прохід оновлення важок
for Layer = 1:nbrOfLayers-1
    Weights{Layer} = Weights{Layer} + Delta_Weights{Layer};
end

% Скид різниці між важками в 0
for Layer = 1:length(Delta_Weights)
    Delta_Weights{Layer} = 0 * Delta_Weights{Layer};
end

%% Прямий прохід

for Sample = 1:length(Samples(:,1))
    outputs = EvaluateNetwork(Samples(Sample,:), NodesActivations, Weights,
end

%% Оновлення коефіцієнтів виходами нейро-адаптивного блоку
dIlK = outputs(1);
dIlI = outputs(2);
dIlD = outputs(3);

%% Оновлення моделі нейро-емулятора новими даними
set_param('steamboiler_NN_3layers_01', 'SimulationCommand', 'update');

if (MSE(Epoch) == 0)
    zeroRMSReached = 1;
end
end

```

Додаток Г
Копія публікації

Міністерство освіти і науки України
Тернопільський національний економічний університет
Харківський національний університет радіоелектроніки
Національний університет «Львівська політехніка»
Вінницький національний технічний університет
Асоціація фахівців комп'ютерних інформаційних технологій

THEU

ФКІТ
FCIS
ASSOCIATION OF COMPUTER INFORMATION TECHNOLOGISTS

 **АСІТ'2016**
acit.tneu.edu.ua

МАТЕРІАЛИ
VI Всеукраїнської школи-семінару
молодих вчених і студентів

**СУЧАСНІ КОМП'ЮТЕРНІ
ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ**

Advanced computer information technologies

20-21 травня 2016 року

THEU
Тернопіль
2016

Міністерство освіти і науки України
Тернопільський національний економічний університет
Харківський національний університет радіоелектроніки
Національний університет «Львівська політехніка»
Вінницький національний технічний університет
Асоціація фахівців комп'ютерних інформаційних технологій

МАТЕРІАЛИ
VI Всеукраїнської школи-семінару
молодих вчених і студентів

СУЧАСНІ КОМП'ЮТЕРНІ ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ

ADVANCED COMPUTER INFORMATION TECHNOLOGIES

20-21 травня 2016 року

АСІТ'2016

Тернопіль
ТНЕУ
2016

СПЕЦІАЛІЗОВАНІ КОМП'ЮТЕРНІ СИСТЕМИ

КОРЕЛЯЦІЙНИЙ СПЕЦПРОЦЕСОР ОПРАЦЮВАННЯ ЦИФРОВИХ ПОТОКІВ ДАНИХ З РОЗПАРАЛЕЛЕНИМИ ОПЕРАЦІЯМИ ДЛЯ МОНИТОРИНГУ СИСТЕМ АВАРІЙНОГО СПОВІЩЕННЯ НАФТОПЕРЕКАЧУВАЛЬНИХ СТАНЦІЙ	
Албанський І.Б.	37
АЛГОРИТМИ АВТОМАТИЧНОГО ЗБОРУ ДАНИХ ВИТРАТ ЕНЕРГОНОСІВ "ІНТЕЛЕКТУАЛЬНОГО" МІСТА	
Борейко О.Ю., Голояд Ю.В.	39
ХАРАКТЕРИСТИКИ ТА КЛІКІСНІ ОЦІНКИ СТРУКТУРИЗОВАНОЇ ІНФОРМАЦІЇ	
Возна Н.Я.	41
РОЗРОБКА АРИФМЕТИЧНОГО МОДУЛЯ СПЕЦПРОЦЕСОРА НА ОСНОВІ ВЕРТИКАЛЬНО-ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ	
Гуменний П.В., Зоттєв С.А.	43
МЕТОД ПОБУДОВИ БАГАТОРЗЯДНОГО ОПЕРАЦІЙНОГО ПРИСТРОЮ ПІДНЕСЕННЯ ЧИСЕЛ ДО КВАДРАТУ	
Давлетова А.Я.	45
АЛГОРИТМ ЗАВАДОСТІЙКОГО КОДУВАННЯ НА ОСНОВІ ЦИКЛІЧНИХ КОДІВ	
Касячук М.М., Борис О.М., Мандебура Н.М.	47
АНАЛІЗ АЛГОРИТМІВ ВИЗНАЧЕННЯ ЕНТРОПІЇ ДЛЯ ПОБУДОВИ КЛАСТЕРНИХ МОДЕЛЕЙ КВАЗІСТАЦІОНАРНИХ ОБ'ЄКТІВ	
Николайчук Я.М., Коростіль Д.В., Слободян С.М.	48
ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ ДЛЯ ВИЗНАЧЕННЯ ВІДМІННОСТЕЙ ОЦІНОК ХАОТИЧНОСТІ ФОРМИ ЕЛЕМЕНТІВ ЕЛЕКТРОКАРДІОГРАМИ В ГРУПОВИХ ДОСЛІДЖЕННЯХ	
Оріховська К.Б.	50
АЛГОРИТМИ РЕГУЛЮВАННЯ ПАРАМЕТРІВ ТЕХНОЛОГІЧНОГО ПРОЦЕСУ В ЕНЕРГЕТИЧНИХ УСТАНОВКАХ	
Столяр О.М.	51
НЕЙРОМЕРЕЖЕВИЙ КОНТРОЛЕР ДЛЯ УПРАВЛІННЯ ТЕМПЕРАТУРОЮ В КАМЕРІ СУШІННЯ ДЕРЕВИНИ	
Трембач Р.Б., Романський А.В.	53
АПАРАТНА РЕАЛІЗАЦІЯ ОБЧИСЛЕННЯ МАКСИМАЛЬНОГО І МІНІМАЛЬНОГО ЧИСЕЛ В МАСИВІ ДАНИХ	
Цмоць І.Г., Ігнатєв І. В., Данілов П.О.	55
ФОРМУВАННЯ ВИМОГ І ВИБІР ПРИНЦИПІВ ПОБУДОВИ АПАРАТНИХ ЗАСОБІВ СОРТУВАННЯ МАСИВІВ ДАНИХ	
Цмоць І.Г., Кантелюк Ю.М.	57
АПАРАТНА РЕАЛІЗАЦІЯ НЕЙРОЕЛЕМЕНТА	
Цмоць І.Г., Кураш Я.Я.	59

СИСТЕМИ ШТУЧНОГО ІНТЕЛЕКТУ

МЕТОДИ ПОШУКУ АСОЦІАТИВНИХ ПРАВИЛ В БАЗІ ДАНИХ БІОМЕДИЧНИХ ЗОБРАЖЕНЬ	
Вербовий С.О., Зубко В.С.	61
АЛГОРИТМИ ПОБУДОВИ НЕЧІТКИХ ПРОДУКЦІЙНИХ ПРАВИЛ НА ОСНОВІ АНАЛІЗУ БІОМЕДИЧНИХ ЗОБРАЖЕНЬ	
Вербовий С.О., Мартинчук Т.О.	63
БАГАТОРІВНЕВА ПАРАЛЕЛЬНО-ІєРАРХІЧНА МЕРЕЖА ДЛЯ КЛАСИФІКАЦІЇ ЗОБРАЖЕНЬ	
Гардиш А.В.	65
МЕДИЧНІ НЕЙРОМЕРЕЖЕВІ ЕКСПЕРТНІ СИСТЕМИ В ДІАГНОСТИЦІ	
Герасімова Д.С.	66
ПРИЙНЯТТЯ РІШЕНЬ ПРИ ФОРМУВАННІ ЕКСПОЗИЦІЇ НА ОСНОВІ ВИЯВЛЕНИХ АСОЦІАТИВНИХ ЗАЛЕЖНОСТЕЙ	
Жилко І.В.	68