

**Sztuczna inteligencja**

**Laboratorium**

**Sterownik klimatyzatora z wykorzystaniem logiki rozmytej**

**Autor: Krzysztof Wiśniewski 10458**

**Prowadzący: Bartosz Popławski**

**Rok akademicki 2019/20**

**Semestr V**

1. **Logika rozmyta**

Logika rozmyta pozwala nam na opisywanie zjawisk zachodzących w naturze w bardziej naturalny dla człowieka sposób i równocześnie możliwe do odczytania przez układu sterowania maszynami. Jest ona coraz częściej wykorzystywana w automatyce przemysłowej, gdzie służy ona do kontroli złożonych procesów i sterowania różnymi układami.

W codziennym życiu rzadko spotykamy się w opisie różnych zjawisk poprzez podanie ich konkretnych parametrów. Nikt nie mówi raczej „Dziś jest 23,5 stopnia Celsjusza” , każdy powie raczej „Dziś jest ciepły dzień”. Taki sposób rozumowania pozwala ludziom na postrzeganie świata w sposób bardziej abstrakcyjny. Między innymi dzięki temu człowiek jest mądrzejszy niż maszyny. Właśnie tutaj w grę wchodzi logika rozmyta, która pozwala na zatarcie tych różnic.

Logika rozmyta bazuje na pojęciu zbiorów rozmytych, czyli zbiorów bez jasno określonych granic. O tym, w jakim stopniu dany element można

zaliczyć do konkretnego zbioru, decyduje tzw. funkcja przynależności.

Przypisuje ona poszczególnym argumentom wartość z zakresu od 0 do 1.

Przetwarzanie danych przy udziale logiki rozmytej dzieli się na kilka etapów. Najważniejsze z nich to fuzyfikacja i defuzyfikacja

Proces fuzyfikacji, jest etapem przypisywania danych wejściowych do

określonych zbiorów rozmytych. Dane wejściowe są analizowane na podstawie

zdefiniowanych wcześniej funkcji przynależności i zostają im na ich podstawie

przypisane wartości.

Kolejnym elementem systemu jest blok odpowiedzialny za przetwarzanie

danych uzyskanych w procesie fuzyfikacji. Na tym etapie, określanym często

jako wnioskowanie, na podstawie przyjętego zestawu kryteriów ustala się

wypadkową wartość wyjściową dla każdego szczebla w schemacie sterowania.

Wynika stąd, że podstawową sprawą w tworzeniu systemu sterowania w oparciu o logikę rozmytą, jest stworzenie bazy zasad. Powstaje ona zazwyczaj na podstawie doświadczeń związanych z konkretnym zjawiskiem, a praca całego układu sterowania zależy od tego, w jakim stopniu jego twórcy odwzorują rzeczywiste zależności między poszczególnymi parametrami. Zbiór kryteriów stanowią zazwyczaj instrukcje warunkowe, dla których tablice

prawdy są wyznaczane na bazie spostrzeżeń ekspertów zajmujących się danym

zjawiskiem.

Ostatnim etapem logiki rozmytej jest proces wyostrzania, czyli defuzyfikacji. Polega on na wyznaczaniu wartości wypadkowej zmiennej wyjściowej na podstawie danych uzyskanych podczas procesu fuzyfikacji, oraz bazy reguł. Jednym ze sposobów stosowanych na tym etapie jest metoda średniej ważonej. W tym celu niezbędne jest przypisanie wag poszczególnym składnikom równania. Jest to jedno z możliwych rozwiązań i istnieje jeszcze wiele innych metod wyznaczania wyjściowej wartości wypadkowej.

1. **Opis zadania**

Projekt polega na implementacji sterownika rozmytego w języku C#, z wykorzystaniem bibliotek AForge.

Poziom temperatury i wilgotności w pomieszczeniu sterowany jest przez sztuczną inteligencje w postaci sterownika rozmytego. Efektem działania programu jest sterowanie dwoma parametrami wyjściowymi:

* Temperatura – temperatura powietrza wydmuchiwanego przez klimatyzator
* Moc wentylatora – moc z jaką klimatyzator wydmuchuje powietrze

Danymi wejściowymi dla sytemu są:

* Temperatura zewnętrzna
* Temperatura wewnętrzna
* Wilgotność wewnętrzna
* Wilgotność zewnętrzna
* Poziom temperatury (Chłodno, Optymalnie, Ciepło) – definiuje czy użytkownik oczekuje że w pomieszczeniu ma być chłodno, optymalnie, czy ciepło. Wejście te jest opcjonalne i nie było ujęte w wymaganiach projektu.

Danym podane na wejściu , oraz tym które będą na wyjściu, należy zdefiniować funkcję przynależności. Następnie należy utworzyć bazę reguł wykorzystywaną w procesie wyostrzania, a na samym końcu należy użyć funkcji środka ciężkości w celu wyznaczenia danych wyjściowych.

1. **Cel ćwiczenia**

Celem wykonanego ćwiczenia jest zaznajomienie się z działaniem logiki rozmytej oraz nauka implementacji logiki rozmytej w aplikacji – sterownika do klimatyzatora. Sterownik ma przyjmować różne wartości wyjściowe wyliczone na podstawie logiki rozmytej, zależne od wartości wejściowych.

1. **Główne klasy programu**

Projekt działa w oparciu o 4 klasy:

* Form1 – klasa która ma za zadanie pobrać od użytkownika dane wejściowe, oraz zwrócić dane wyjściowe
* FormFuzzyChart – klasa która ma za zadanie wizualnie pokazać przedziały przynależności na diagramach.
* InputData – klasa przechowująca dane wejściowe pobrane od użytkownika
* FuzzyControl – najważniejsza klasa w projekcie, to ona odpowiada za cały system logiki rozmytej użyty w programie i to na jej opisie skupię się najbardziej w dalszym opisie działania. Swoje działanie opiera we wielu przypadkach na klasie AForge.Fuzzy

1. **Główne metody klasy FuzzyControl**

**SetFuzzySet**

Metoda która ma za zadanie utworzyć odpowiednie funkcje przynależności dla danych wejściowych i wyjściowych

private void SetFuzySet() //ustawia wartość przedziałów

{

//ustawia wartości przedziałów dla temperatury zewnętrzej

TrapezoidalFunction function = new TrapezoidalFunction(-10,30 , TrapezoidalFunction.EdgeType.Right);

tmpColdOut = new FuzzySet("ColdOut", function);

function = new TrapezoidalFunction(-10, 30, TrapezoidalFunction.EdgeType.Left);

tmpHotOut = new FuzzySet("HotOut", function);

//ustawia wartości przedziałów dla temperatury

function = new TrapezoidalFunction(1, 10, TrapezoidalFunction.EdgeType.Right);

tmpVeryCold = new FuzzySet("VeryCold", function);

function = new TrapezoidalFunction(5, 10, 13, 16);

tmpCold = new FuzzySet("Cold", function);

function = new TrapezoidalFunction(13, 16, 21, 26);

tmpMedium = new FuzzySet("Medium", function);

function = new TrapezoidalFunction(21, 25, 30, 35);

tmpHot = new FuzzySet("Hot", function);

function = new TrapezoidalFunction(30, 35, TrapezoidalFunction.EdgeType.Left);

tmpVeryHot = new FuzzySet("VeryHot", function);

//ustawia wartości przedziałów dla wilgotności

function = new TrapezoidalFunction(0, 40, TrapezoidalFunction.EdgeType.Right);

smallHumidity = new FuzzySet("Small", function);

function = new TrapezoidalFunction(20, 40, 60, 80);

goodHumidity = new FuzzySet("Good", function);

function = new TrapezoidalFunction(60, 100, TrapezoidalFunction.EdgeType.Left);

bigHumidity = new FuzzySet("Big", function);

//ustawia wartość przedziału przedział siły wentylatora (od zera do 100)

function = new TrapezoidalFunction(20, 30, TrapezoidalFunction.EdgeType.Right);

veryLowStrenght = new FuzzySet("VeryLow", function);

function = new TrapezoidalFunction(20, 30, 40, 50);

lowStrenght = new FuzzySet("Low", function);

function = new TrapezoidalFunction(40, 50, 60, 70);

mediumStrenght = new FuzzySet("Medium", function);

function = new TrapezoidalFunction(60, 70, 80, 90);

hightStrenght = new FuzzySet("Hight", function);

function = new TrapezoidalFunction(80, 90, TrapezoidalFunction.EdgeType.Left);

veryHightStrenght = new FuzzySet("VeryHight", function);

}

**SetLinguisticVariable**

Metoda odpowiedzialna za utworzenie tzw. zmiennych lingwistycznych odpowiadającym danym wejściowym i wyjściowym, oraz na przypisaniu do nich odpowiednich funkcji przynależności

private void SetLinguisticVariable() //ustawia zmienne lingwistyczne

{

airOutTemperature = new LinguisticVariable("AirOutTemperature", -20, 40);

airOutTemperature.AddLabel(tmpColdOut);

airOutTemperature.AddLabel(tmpHotOut);

airTemperature = new LinguisticVariable("AirTemperature", 0, 40);

airTemperature.AddLabel(tmpVeryCold);

airTemperature.AddLabel(tmpCold);

airTemperature.AddLabel(tmpMedium);

airTemperature.AddLabel(tmpHot);

airTemperature.AddLabel(tmpVeryHot);

Humidity = new LinguisticVariable("Humidity", 0, 100);

Humidity.AddLabel(smallHumidity);

Humidity.AddLabel(goodHumidity);

Humidity.AddLabel(bigHumidity);

HumidityOut= new LinguisticVariable("HumidityOut", 0, 100);

HumidityOut.AddLabel(smallHumidity);

HumidityOut.AddLabel(goodHumidity);

HumidityOut.AddLabel(bigHumidity);

desiredTemperature = new LinguisticVariable("DesiredTemperature", 0, 40);

desiredTemperature.AddLabel(tmpVeryCold);

desiredTemperature.AddLabel(tmpCold);

desiredTemperature.AddLabel(tmpMedium);

desiredTemperature.AddLabel(tmpHot);

desiredTemperature.AddLabel(tmpVeryHot);

FunTemperature = new LinguisticVariable("FunTemperature", 0, 40);

FunTemperature.AddLabel(tmpVeryCold);

FunTemperature.AddLabel(tmpCold);

FunTemperature.AddLabel(tmpMedium);

FunTemperature.AddLabel(tmpHot);

FunTemperature.AddLabel(tmpVeryHot);

FunStrenght = new LinguisticVariable("FunStrenght", 0, 100);

FunStrenght.AddLabel(veryLowStrenght);

FunStrenght.AddLabel(lowStrenght);

FunStrenght.AddLabel(mediumStrenght);

FunStrenght.AddLabel(hightStrenght);

FunStrenght.AddLabel(veryHightStrenght);

}

**SetDatabase**

Tworzy bazę danych która zawiera w sobie zdefiniowane wcześniej zmienne lingwistyczne

private void SetDatabase() //tworzy baze danych wyrazeń lingwistycznych

{

db = new Database();

db.AddVariable(airTemperature);

db.AddVariable(Humidity);

db.AddVariable(FunTemperature);

db.AddVariable(FunStrenght);

db.AddVariable(desiredTemperature);

db.AddVariable(airOutTemperature);

db.AddVariable(HumidityOut);

}

**CreateInferenceSystem**

Bardzo ważna metoda. Tutaj zostaję utworzony cały system wnioskowania i baza reguł na podstawie której następuje wnioskowanie

private void CreateInferenceSystem() //tworzy system wnioskowania, wraz z gotową bazą reguł

{

IS = new InferenceSystem(db, new CentroidDefuzzifier(1000));

IS.NewRule("Rule 1", "IF AirTemperature IS VeryCold AND Humidity IS Small AND DesiredTemperature IS Cold THEN FunTemperature IS Hot");

IS.NewRule("Rule 2", "IF AirTemperature IS VeryCold AND Humidity IS Good AND DesiredTemperature IS Cold THEN FunTemperature IS Hot");

IS.NewRule("Rule 3", "IF AirTemperature IS VeryCold AND Humidity IS Big AND DesiredTemperature IS Cold THEN FunTemperature IS Medium");

.

.

.

IS.NewRule("Rule100", "IF AirOutTemperature IS HotOut AND HumidityOut IS Good THEN FunStrenght IS Low");

IS.NewRule("Rule101", "IF AirOutTemperature IS ColdOut AND HumidityOut IS Big THEN FunStrenght IS Hight");

IS.NewRule("Rule102", "IF AirOutTemperature IS HotOut AND HumidityOut IS Big THEN FunStrenght IS VeryLow");

}

**GetFunTemperature**

Metoda która zwraca obliczoną wartość temperatury na wyjściu. Wymaga ona podania argumetu „input” , który to jest obiektem typu InputData. Input zawiera w sobie dane wejściowe przekazane przez użytkownika które zostają przypisane do odpowiednich zmiennych lingwistycznych jako ich wartości. Następnie przy pomocy wbudowanej w bibliotekę AForge.Fuzyy metody obiektu IS typu InferenceSystem (utworzony w metodzie CreateInferenceSystem) -„Evaluate” zwraca wyliczoną wartość temperatury klimatyzatora na wyjściu.

public int GetFunTemperature(InputData input) //zmiena zwracająca finalną wartość temperatury wiatraka

{

airTemperature.NumericInput = input.Temperature;

Humidity.NumericInput = input.Humidity;

desiredTemperature.NumericInput = input.DesiredTemperature;

airOutTemperature.NumericInput = input.TemperatureOut;

HumidityOut.NumericInput = input.HumidityOut;

int result;

int tmp = Convert.ToInt32(IS.Evaluate("FunTemperature"));

result = tmp;

return result;

}

**GetFunStrenght**

Metoda która zwraca obliczoną wartość mocy wentylatora na wyjściu. Wymaga ona podania argumetu „input” , który to jest obiektem typu InputData. Input zawiera w sobie dane wejściowe przekazane przez użytkownika które zostają przypisane do odpowiednich zmiennych lingwistycznych jako ich wartości. Następnie przy pomocy wbudowanej w bibliotekę AForge.Fuzyy metody obiektu IS typu InferenceSystem (utworzony w metodzie CreateInferenceSystem) -„Evaluate” zwraca wyliczoną wartość mocy wentylatora na wyjściu.

public int GetFunStrenght(InputData input) //zmiena zwracająca finalną wartość siły wiatraka

{

airOutTemperature.NumericInput = input.TemperatureOut;

HumidityOut.NumericInput = input.HumidityOut;

airTemperature.NumericInput = input.Temperature;

Humidity.NumericInput = input.Humidity;

desiredTemperature.NumericInput = input.DesiredTemperature;

int result;

int tmp = Convert.ToInt32(IS.Evaluate("FunStrenght"));

result = tmp;

return result;

}

**Funkcje zwracające wartość przynależności do przedziału**

Poniże funkcję zwracają stopień przynależności do przedziału dla konkretnej wartości. Nie są one opisywane osobno ponieważ są nieistotne z punktu działania programu. Zostały utworzone jedynie w celu umożliwienia wizualizacji przedziałów na diagramach w klasie FormFuzzyChart

//funkcje zwracające wartość przynależności do przedziału dla podanej temperatury zewnętrznej

public float valueTmpColdOut(float temperature)

{

return tmpColdOut.GetMembership(temperature);

}

public float valueTmpHotOut(float temperature)

{

return tmpHotOut.GetMembership(temperature);

}

//funkcje zwracające wartość przynależności do przedziału dla podanej temperatury

public float valueTmpVeryCold(float temperature)

{

return tmpVeryCold.GetMembership(temperature);

}

public float valueTmpCold(float temperature)

{

return tmpCold.GetMembership(temperature);

}

public float valueTmpMedium(float temperature)

{

return tmpMedium.GetMembership(temperature);

}

public float valueTmpHot(float temperature)

{

return tmpHot.GetMembership(temperature);

}

public float valueTmpVeryHot(float temperature)

{

return tmpVeryHot.GetMembership(temperature);

}

//funkcje zwracające wartość przynależności do przedziału dla podanej wilgotnosci

public float valueSmallHumidity(float temperature)

{

return smallHumidity.GetMembership(temperature);

}

public float valueGoodHumidity(float temperature)

{

return goodHumidity.GetMembership(temperature);

}

public float valueBigHumidity(float temperature)

{

return bigHumidity.GetMembership(temperature);

}

//funkcje zwracające wartość przynależności do przedziału dla podanej sily wiatraka

public float valueVeryLowStrenght(float temperature)

{

return veryLowStrenght.GetMembership(temperature);

}

public float valueLowStrenght(float temperature)

{

return lowStrenght.GetMembership(temperature);

}

public float valueMediumStrenghty(float temperature)

{

return mediumStrenght.GetMembership(temperature);

}

public float valueHightStrenght(float temperature)

{

return hightStrenght.GetMembership(temperature);

}

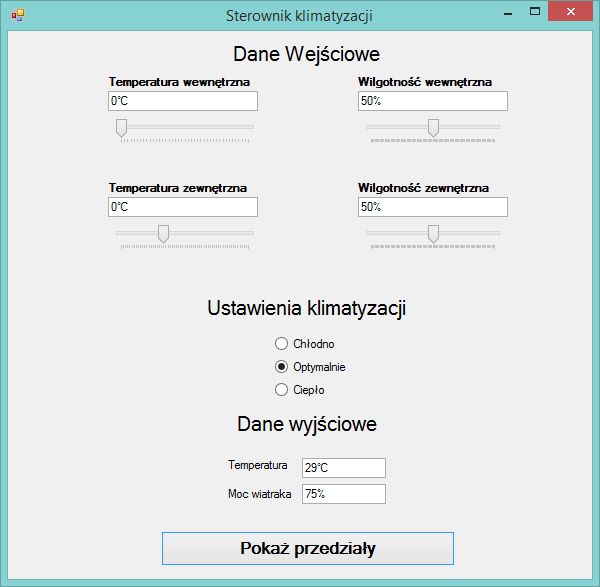
public float valueVeryHightStrenght(float temperature)

{

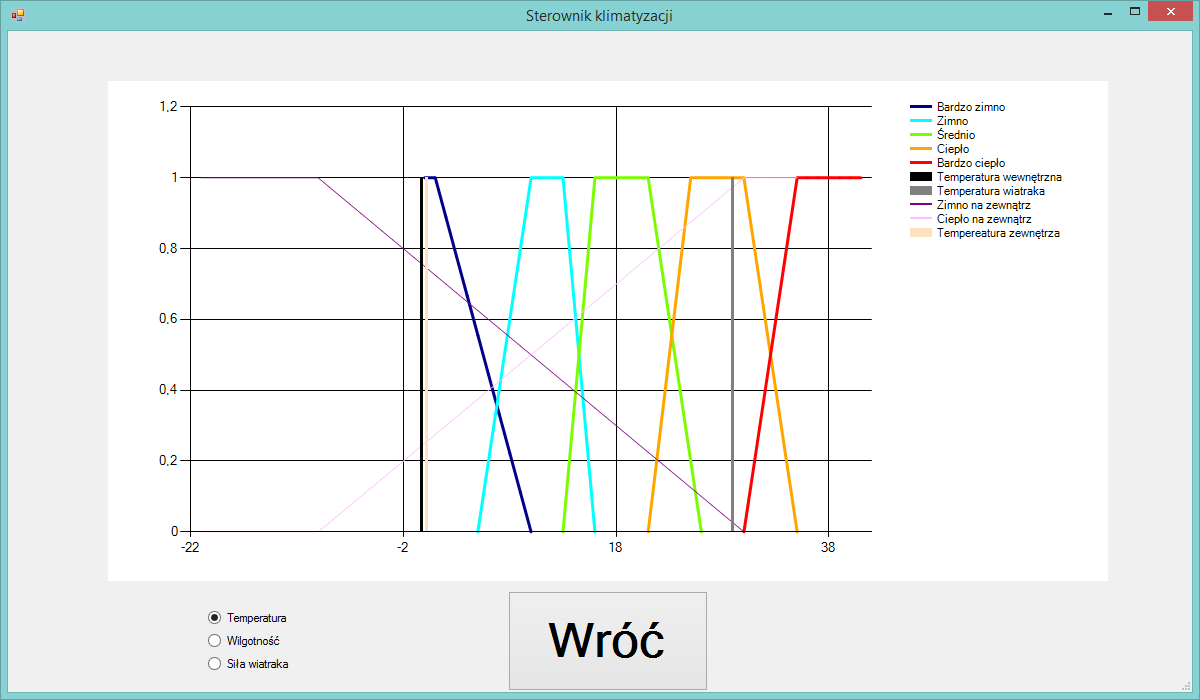
return veryHightStrenght.GetMembership(temperature);

}

1. **Wygląd programu**



Rysunek 1 Interfejs główny programu



Rysunek 2 Wygląd formy FormFuzzyChart używanej do wizualizacji przedziałów

1. **Podsumowanie**

Program spełnia swoje zadanie i potrafi sterować klimatyzacją poprzez nadawanie jej parametrów takich jak moc wentylatora i temperatura wydmuchiwanego powietrza. Program spełnia wymagania przed nim postawione, działa na danych wejściowych podanych przez użytkownika takich jak temperatura wewnętrzna, temperatura zewnętrzna, wilgotność wewnętrzna oraz wilgotność zewnętrzna. Dodatkowo został wzbogacony o piąte wejście, czyli tryb pracy (chłodny, optymalny, ciepły). Logika rozmyta, która w projekcie została użyta dzięki klasie AForge świetnie nadaje się do wykorzystywania w podobnych zadaniach. Dzięki projektowi nauczyłem się implementować logikę rozmytą do swoich aplikacji.

Aplikacja posiada potencjał do dalszego rozwoju. Przede wszystkim, dalszą drogą rozwoju projektu powinno być dążenie do podłączenie do niego fizycznych czujników temperatury i wilgotności co pozwoliło by na pobierania realnych danych wejściowych, bez konieczności wprowadzania ich przez użytkownika. Następnie można by podłączyć fizycznie klimatyzator który po otrzymaniu danych wyjściowych od programu działał by we wskazany sposób.

Kolejnym krokiem rozwoju aplikacji byłoby podpięcie bazy danych która pozwoliłaby na zapis pomiarów, co w przyszłości mogło by posłużyć do różnych analiz. Naturalnym krokiem byłoby też wprowadzenie użytkowników i systemu logowania do programu, co pozwoliłoby na zapis ulubionych ustawień.

1. **Spis rysunków wykorzystanych w sprawozdaniu**
2. Interfejs główny programu
3. Wygląd formy FormFuzzyChart używanej do wizualizacji przedziałów