T.C.
ANKARA ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

BLM2536 Bulanık Mantık – Birinci Proje Raporu

Alp Ertunga Elgün 19290238
Efe Alptekin 19290211

2021

İçindekiler

Bölüm 1
Özet

Bu rapor BLM2536 Bulanık Mantık dersinin birinci projesi için hazırlanmıştır.Programla dili olarak buradan ulaşabileceğiniz Lua dili kullanılmıştır. Kendiniz compile edip farklı inputlar için denemek isterseniz buradan ulaşabileceğiniz LuaJIT compiler'ı kullanmanızı tavsiye ederiz.

Kaynak kodlarına bu adresteki Github sayfasından ulaşabilirsiniz. Comment'lerle birlikte gayet anlaşılabilir bir program olsa da bu raporda daha detaylı bir şekilde nasıl çalıştığı anlatılmıştır. Kullanılan LuaFuzzy kütühanesinin Github sayfasına ise buradan ulaşabilirsiniz.

Program geliştirilirken kaynak olarak Roberto Lerusalimschy'nin yazdığı Programming in Lua kitabı ve LuaFuzzy kütüphanesi kullanılmıştır. Fuzzy algoritmaları geliştirirken kaynak olarak Murat Osmanoğlu'nun dersleri ve Kwang Hyung Lee'nin yazdığı First Course on Fuzzy Theory and Applications kitabı kullanılmıştır.

2. Bölümde kontrol sisteminin bileşenleri ve programda kullanılan algoritmaların çıkarımları anlatılmış, 3. bölümde programın nasıl çalştığı anlatılmış, 4. bölümde ise bir input'un programdan geçerken hani değerlere dönüşerek sonucu oluşturduğu açıklanmıştır.

Bölüm 2
Giriş

2.1 Bulanık Kontrol Sistemi

Bir bulanık control sistemi 4 farklı bileşenden oluşur. İlk bileşen olan Fuzzification aşamasında sisteme verilen crisp inputlar fuzzy hale getirilir. Dağa sonra knowledge base bileşeninde bulunan datalar ve kurallar kullanılarak Inference bileşeninde karar verme mekanizmaları uygulanır. En son aşama olan defuzzification bileşeninde ise fuzzy outputlar tekrar crisp hale getirilir ve sistemden çıkartılır.

Bu bileşenlerin projemizde nasıl çalıştığını sonraki bölümlerde bulabilirsiniz.

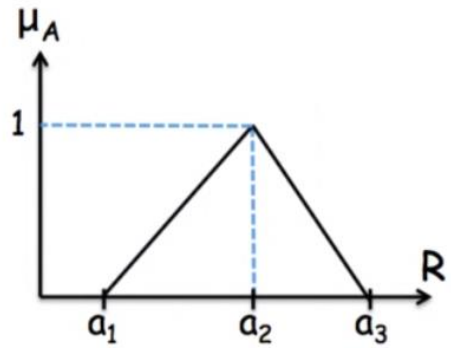
2.2 Fuzzification

Fuzzification aşamasında crisp input değerlerinin, üyelik fonksiyonları kullanarak, fuzzy kümelerdeki üyelik derecelerini buluruz.

Geliştirdiğimiz control sisteminde verilen sıcaklık ve basınç inputlarını kural matrix'ini kullarak karbondiyoksit output'una dönüştürmek için önce bu 2 değişkenin hangi fuzzy kümelerinde olduklarını ve bu kümelerdeki üyelik derecelerini bulmamız gereklidir.

2.2.1 Triangular Fuzzy Number

Geliştirdiğimiz control sisteminde verilen 3 değişkeni ve üyelik fonksiyonlarını incelersek, 3 değerin de üçgensel olduğunu kolayca farkedebiliriz. Üçgensel fuzzy sayılarda membership fonksiyonu doğrulardan oluştuğu için doğru denklemini kullanarak aşağıdaki formulü elde edebiliriz.



- $A = (a_1, a_2, a_3)$

$$\mu_A(x) = \begin{cases} 0, & \text{if } x < a_1 \\ (x - a_1)/(a_2 - a_1), & \text{if } a_1 \le x \le a_2 \\ (a_3 - x)/(a_3 - a_2), & \text{if } a_2 \le x \le a_3 \\ 0, & \text{if } x > a_3 \end{cases}$$

Bu formulü sıçaklık ve basınç değişkenlerine uygulayarak onları fuzzy hale getirebiliriz.

Yine bu formulü uygularken bir crisp değerin birden fazla fuzzy değere karşılık gelebileceğini unutmamamız gerekir.

2.3 Knowledge Base

Knowledge Base'de kontrol sisteminin kullanacağı datalar ve kurallar bulunur. Geliştirdiğimiz kontrol sisteminde 3 farklı değişken ve bu değişkenlerin üyelik fonksiyonu tanımlanmıştır. Ayrıca kural matrix'i IF-THEN olarak kullanılmak üzere bize verilmiştir.

2.4 Fuzzy Inference

Bu aşamada önceden belirlenmiş kurallarla inputları birleştirip fuzzy bir output elde edilir. Geliştirdiğimiz kontrol sisteminde bizden Mamdani çıkarım yöntemi kullanmamız isteniyor.

2.4.1 IF-THEN RULE

IF-THEN çıkarımlar genel olarak modus ponens ile yapılır. Bizim geliştirdiğimiz kontrol sisteminde de bu yöntemi uygulamak kolay olduğu için bu yöntemi uygulayacağız. Modus ponens genel olarak şu şekilde formule edilebilir.

- modus ponens

fact : x is A' : R(x)

rule : if x is A, then y is B : R(x, y)

result : y is B' : R(y) = R(x) o R(x, y)

Bu formulü fuzzy logic olarak göstermek istersek aşağıdaki formulü elde ederiz.

$$R(y) = R(x) \circ R(x, y)$$

$$\mu_R(y) = \max_x (\min (\mu_R(x), \mu_R(x, y)))$$

2.4.2 Mamdani Method

Mamdani method'unda implication kısmında fuzzy logic olarak and kullanılır. Geliştirdiğimiz kontrol sisteminde bizden mamdani method kullanılması istendiği için bu method kullanıcağız.

Mamdani

$$f(\mu_A(x), \mu_B(y)) = \mu_A(x) \wedge \mu_B(y)$$

2.4.3 Decomposition of Rules

Birden çok input ve output olduğu durumları, birden çok input ama bir output'dan oluşan fuzzy systemlerin bir koleksiyonu olarak kabul edip sonra bu outputları birleştirebiliriz. Bu işlemi yaparken aşağıdaki çıkarımları kullanabiliriz.

2.4.3.1 Singleton Input

Bir input'u, bir kural varken output'a çevirmek için aşağıdaki çıkarımı kullanabiliriz.

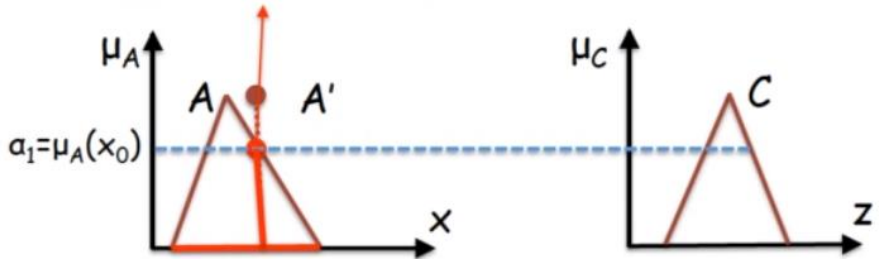
Singleton Input

- the fact is : x is x_0
  the rule is : If x is A, then z is C
  the result is : z is C'
- $C' = x_0 \circ (A \rightarrow C) = A' \circ R$

$$\mu_{C'}(z) = \mu_0 \circ (\mu_A(x) \rightarrow \mu_C(z))$$

$$\mu_{C'}(z) = \max_x \{\mu_0 \wedge \mu_R(x,y)\} = \max_x \{\mu_0 \wedge (\mu_A(x) \wedge \mu_C(z))\}$$

$$= \max_x \{\mu_0 \wedge \mu_A(x)\} \wedge \mu_C(z) = \alpha_1 \wedge \mu_C(z)$$



2.4.3.2 Two Input Single Output

Birden çok inputu, tek kural varken output'a çevirmek için aşağıdaki çıkarımı kullanabiliriz.

Mamdani Fuzzy Inference

Two Input Single Output

- input : x is A' and y is B'

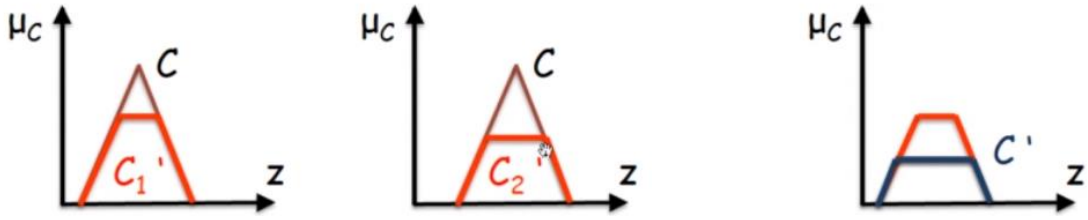
  R : if x is A and y is B, then z is C : (A and B) → C

  output : z is C '

- $C' = A' \circ R = A' \circ [(A \text{ and } B) \rightarrow C] = A' \circ [(A \rightarrow C) \cap (B \rightarrow C)]$

  $C' = [A' \circ (A \rightarrow C)] \cap [A' \circ (B \rightarrow C)] = C_1' \cap C_2'$

  $\mu_{C'}(z) = \min \{ \mu_{C_1'}(z), \mu_{C_2'}(z) \}$



Sizin de farkettiğiniz gibi bu iki çıkarımı birleştirirken, üyelik derecesi küçük olan input her zaman baskın çıkacakdır. Bu sebeple üyelik derecelerinin minimumunu alıp singleton yöntemini uygularsak istediğimiz sonuça kolayca ulaşabiliriz.

2.4.3.3 Single Input Single Output

Bir input ve yine bir input varsa fakat birden çok kural varsa aşağıdaki çıkarımı kullanabiliriz.

Mamdani Fuzzy Inference

Single Input Single Output

- input : x is A'

  R_1 : if x is A_1, then z is C_1 : $A_1 \rightarrow C_1$

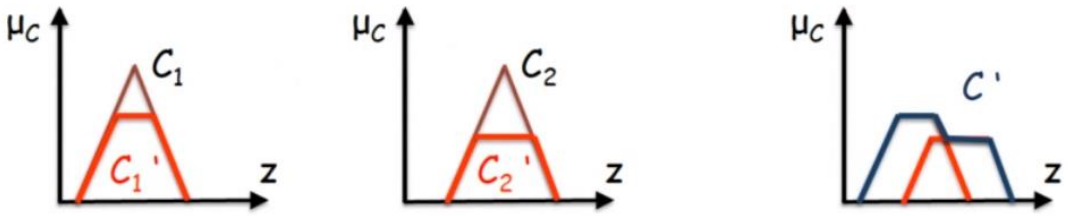
  R_2 : if x is A_2, then z is C_2 : $A_2 \rightarrow C_2$

  output : z is C'

- $C' = A' \circ (R_1 \cup R_2) = A' \circ [(A_1 \rightarrow C_1) \cup (A_2 \rightarrow C_2)]$

  $C' = [A' \circ (A_1 \rightarrow C_1)] \cup [A' \circ (A_2 \rightarrow C_2)] = C_1' \cup C_2'$

  $\mu_{C'}(z) = \max \{ \mu_{C\_1'}(z) , \mu_{C\_2'}(z) \}$



2.4.3.4 Sonuç

Sonuç olarak bu tüm çıkarımları birleştirerek geliştirdiğimiz kontrol sisteminde input değerlerini fuzzy değerlere dönüştürebiliriz.

İlk yapmamız gereken iki input'un üyelik değerlerine AND uygulamak yani küçük olanını bulmak, sonra bulduğumuz bu üyelik değerlerinden küçük üyelik değerine sahip olan output'daki üyelik değerlerini bulmak ve en sonda da farklı kurallardan bulduğumuz bu tüm outputların bileşkesini almak yani max'ını bulmak. Böylece bulmak istediğimiz output fuzzyset'ini elde etmiş oluruz.

2.5 Defuzzification

Kontrol sisteminin en sonunda elde ettiğimiz fuzzy değeri, crisp bir değere dönüştürüp çıktı olarak vermemiz gerekmektedir. Bu işlemi yapmak için birden fazla yöntem olsa da geliştirdiğimiz kontrol sisteminde bizden Center of Area yöntemi kullanılması istenildiği için biz bu yöntemi kullanacağız.

2.5.1 Center of Area

Yaygın olarak kullanılan Center of Area yöntemi, bir fuzzy kümenin ağırlık merkezinin crisp bir değer olarak kabul edebileceğimizi söyler. Ağırlık merkezini hesaplamak için …

Discrete bir sette aşağıdaki formula:

$$z_0 = \frac{\sum_{j=1}^{n} \mu_C(z_j) \cdot z_j}{\sum_{j=1}^{n} \mu_C(z_j)}$$

Continuous bir sette ise aşağıdaki formulü kullanabiliriz:

$$CoA = \frac{\int_{x_{min}}^{x_{max}} f(x) * x \, dx}{\int_{x_{min}}^{x_{max}} f(x) dx}$$

2.5.1.1 Discrete ve Continuos Farkı

Geliştirdiğimiz kontrol sistemindeki output fonksiyonu continuous olmasına rağmen programda discrete bir fonksiyon olarak kabul edip defuzzification işlemini gerçekleştirdik.

Bunu yapmamızdaki sebep continuos setteki formülü programlamanın zor olması ve discrete versiyonunda aralıkları çok küçük yaparsak sonuçun neredeyse hiç değişmemesi.

Şuanki bilgisiyarların hızları ve hafızaları, aralıkları sonucu neredeyse etkilemeyecek kadar küçültebilmemize izin veriyor. 4. Bölümde aralıkları çok küçültsek bile sonucun neredeyse hiç değişmediğine dair örnekler verdik. Ayrıca 3. bölümde programın tüm kontrol sistemini nasıl çalıştırdığını anlattık.

Bölüm 3
Yapı

3.1 LuaFuzzy Kütüphanesi

LuaFuzzy kütüphanesi Mathlab'un fuzzy logic toolbox'ına benzer olarak fuzzy logic kullanarak problem çözmeye yardımcı olmak amacıyla geliştirilmiş basit bir kütüphanedir. 2008 yılında Lucas Lorensi tarafından sadece Lua dili kullanılarak geliştirilmiştir.

Kütüphane bir kontrol sistemi geliştirmek için gereken neredeyse tüm fonksiyon ve class'ları hazır olarak barındırdığı halde bölümün devamında bu fonksiyon ve class'ların neredeyse hepsinin nasıl çalıştığını detaylı olarak bulabilirsiniz.

Bazı class'lar sadece OOP kurallarına uyulmak için yazıldıkları ve fuzzy logic algoritmalarıyla alakaları olmadıkları için detaylıca anlatılmamıştır.

LuaFuzzy Github sayfasına buraya tıklayarak ulaşabilirsiniz.

3.1.1 Fuzzy System

458-507. Satırlar arasında bulunan luafuzzy fonksiyonu yeni bir fuzzy sistemi oluşturur. Bu sistemindeki step variable'ı Center of Area'yı bulurken her discrete öğe arasında bıraktığımız boşluğu ifade eder. 4. Bölümde bu variable ile sonucun ilişkisini gözlemleyeceğiz.

3.1.2 Inputs

3.1.2.1 Triangular Membership Function

Bu fonksiton değişkenler üçgensel membership fonksiyonuna sahip ise verilen crisp değerin karşılığı olan üyelik değerini bulmayı sağlar.

```
111  -- pertinency
112  --    ^
113  --    |
114  --    |  /2\
115  --    | /   \
116  --  -|1-----3-->x
117  ----------------------------------------------------
118  function trimf( x, params )
119    if x > params[1] and x < params[2] then
120      return (x-params[1])/(params[2]-params[1])
121    elseif x >= params[2] and x < params[3] then
122      return  (params[3]-x)/(params[3]-params[2])
123    else
124      return 0
125    end
126  end
```

"params" Parametresi üç öğeli bir table alır. 113. satırdan 116. satıra kadar olan şekilde de görebileceğiniz gibi table'ın 1. index'ine en düşük x değeri, 2. index'ine tepedeki x değeri, 3. index'ine de

en büyük x değeri atanır. Bu table ile 2.2.1 nolu bölümdeki formül oluşturulur ve bu formule gore x'in üyelik değeri bulunur.

3.1.3 Deffuzification Functions

3.1.3.1 Center of Area

```
134  ----------------------------------------------------------------------------
135  -- Centroid deffuzyfication method
136  -- @param fs A fuzzyset with pairs of x and y like: fs[1].x, fs[1].y
137  -- @return Return the geometric center value
138  ----------------------------------------------------------------------------
139  function centroid( fs )
140    if #fs <= 1 then
141      error('invalid number of entries in the fuzzyset')
142    end
143    local accxy = 0
144    local accy = 0
145    for i,v in ipairs(fs) do
146      if v.y > 0 then
147        accxy = accxy + v.x*v.y
148        accy = accy + v.y
149      end
150    end
151    if accy > 0 then
152      return accxy/accy
153    else
154      return (fs[1]-fs[#fs])/2
155    end
156  end
```

En son aşamada elde ettiğimiz sonuç fuzzy kümesi kodda parameter olan "fs" dir. fs'nin her indexinde 1 sonraki stepdeki x ve ona karşılık gelen y vardır.

"accxy" accumulatedxy ve "accy" de accumulated y olarak kullanılmışlardır. Alanın ortasını bulmak için tüm x'ler ağırlıkları olan y ile çarpılmış ve en son toplam ağırlığa bölünmüştür. 2.5.1 nolu bölümdeki discrete formula kullanılmıştır.

3.1.4 Solve Fuzzy

Fuzzy kontrol sisteminin çözülüp sonuçun basıldığı fonksiyon. Vararg olan 2. Argumanına sırasıyla input değerleri atanır.

Error kısımlarını anlaşılması zor olmadığı için anlatmadık. Diğer her kısmın açıklamasını aşağıda bulabilirsiniz.

3.1.4.1 Input to Membership

Bu kısımda 2.2.1'de açıkladığımız Triangular Membership fonksiyonunu kullanarak, her bir inputun her bir linguistic değişkeninin üyelik değerini hesaplıyoruz.

```
179    -- solve each input
180    for i,inp in ipairs(fuzzy.inps) do
181
182      -- current input value
183      local x = inpvals[i]
184
185      -- error check
186      if x > inp.mx or x < inp.mn then
187        error('value ' .. x .. ' out of range for input \'' .. inp.name .. '\'')
188      end
189
190      -- save the current value of the function for each lingvar
191      for _,lg in pairs(inp.lingvars) do
192        lg.value = lg.mf( x, lg.params )
193      end
194    end
```

Adım adım açıklarsak:

1. Fuzzy sistemdeki her input için, i şuan işlenen input'un index'idir:
   1.1. X, fonksiyonun 2. Argumani olan vararg listesindeki i'ninci sayıdır.
   1.2. Inputtaki her linguistic değişken için:
      1.2.1. O linguistic değişkenin üyelik fonksiyonu (Geliştirdiğimiz sistemde 3.1.2'deki trimf fonksiyonu) kullanılarak üyelik değeri hesaplanır ve kuralı triggerlayacak değeri bulmak için lg.value'de saklanır.

3.1.4.2 Rule Trigger Value

Bu kısımda kuralları çalıştıracak olan üyelik değerini hesaplıyoruz. Bunu yaparken 2.4.3.2'de açıkladığımız çıkarımı kullanacağız.

Bu işlemi her rule'a uyguladığımız için bundan sonra anlatılan kısımlar, loop'un sona erdiği belirtilmediği sürece, fuzzy.rules table'ını itere eden bir loop içindedir.

```
196    -- for each rule...
197    for ri,r in ipairs(fuzzy.rules) do
```

```
236     -- retrive the connection function
237     local conn = fuzzy[r.connmethod]
238
239     -- result value of premises
240     r.value = fuzzy.inps[r.pres[1].ifpart].lingvars[r.pres[1].ispart].value
241
242     -- for each premise...
243     for pi=2,#r.pres do
244       local v = fuzzy.inps[r.pres[pi].ifpart].lingvars[r.pres[pi].ispart].value
245       r.value = conn(v, r.value)
246     end
```

Adım adım açıklarsak:

1. conn şuan incelenen rule'un connection methoduna (Bizim geliştirdiğimiz kontrol sisteminde her rule için bu method AND'dir. tmin fonksiyonu AND işlemini gerçekleştiren fonksiyondur) eşittir.
2. 1. Öncülün değeri, karşılaştırma yapılabilmesi için,
   r.value değerine atanır.
3. İncelenen ruledaki her bir öncül için, pi öncülün index olarak:
   3.1. Şuan işlenen öncülün değeri, v değerine atanır.
   3.2. r.value v eve değerleri conn(tmin) fonksiyonunda kıyaslanır ve r.value'de saklanır.

Bu işlemin sonunda r.value en küçük öncül değerine eşittir ve kuralı triggerlayayacağı için saklanır.

```
25  function tmin(a1,a2)
26    if a2 < a1 then
27      return a2
28    else
29      return a1
30    end
31  end
```

3.1.4.3 Output Fuzzy Sets

```
249     -- Calculate the resulting fuzzyset for each implication...
250     for _,imp in ipairs(r.imps) do
251
252       -- retrive the out's table
253       local out = fuzzy.outs[imp.thenpart]
254
255       -- retrive the lingvar from the out table
256       local lg = out.lingvars[imp.ispart]
257
258       -- save the fuzzyset for each implication
259       imp.fuzzyset = {}
260
261       -- compute the step
262       local step = (out.mx - out.mn)*fuzzy.step
263
264       -- calculate the resulting fuzzyset
265       for i=out.mn,out.mx,step do
266
267         -- computes the mf value for i
268         local lgval = lg.mf( i, lg.params )
269
270         -- compute the implication result
271         local v = fuzzy.implicmethod( lgval , r.value )
272
273         -- add the result to the fuzzy set
274         table.insert( imp.fuzzyset, { x=i, y=v } )
275       end
276     end -- end implications
277   end -- end rules
```

Bu kısımda kuralların sonucunda elde edeceğimiz fuzzy setleri hesaplıyoruz. Bunu yaparken 2.4.3.1'de açıkladığımız çıkarımı kullanacağız.

Bu kısımların da rule loop'unun içinde olduğunu unutmadan adım adım açıklarsak :

1. İncelenen ruledaki her bir gerektirme için, imp şuan incelenen gerektirme olarak:
   1.1. out, imp'in ismindeki output'dur.
   1.2. lg, out'un imp'de kullanılan linguistic değişkenidir.
   1.3. Fuzzyset table'ını yaratırız.
   1.4. step, continuous olan output fonksiyonunu discrete olarak incelerken kullanacağımız aralık uzunluğudur.
   1.5. Output'un x ekseninde başından sonuna kadar her bir step için, i şuan incelenen x olak üzere:
      1.5.1. 3.5.1'e benzer şekilde, output'un i deki üyelik değeri bulunur ve lgval a atanır.
      1.5.2. Fuzzy.implicmethod ana class'da tanımlandığı üzere tmin'e eşittir. Çıkarımdan görüleceği üzere sadece rule'u triggerlayan değerden küçük kısımları alıcağımız için bu fonksiyonu kullanırız ve sonucu v ye atarız. Sonuç olarak v i'ninci x'e denk gelen y'ye eşit olur.
      1.5.3. X=i, y=v olmak üzere değeri fuzzyset table'ına ekleriz.

Bu işlemin sonucunda imp.fuzzyset çıkarımda kullanılacak fuzzyset'e eşittir ve sonraki aşamalarda kullanılmak üzere saklanır. En son olarak artık rule loop'unu kapatırız.

3.1.4.4 Aggregation of Fuzzy Sets

```lua
--
-- Computes output...
--

-- result table
local outvals = {}

-- solve each output
for i,out in ipairs(fuzzy.outs) do

  -- clear previews fuzzyset
  out.fuzzyset = nil

  -- for each rule...
  for _,r in ipairs(fuzzy.rules) do

    --
    -- Aggregation
    --

    -- for each implication
    for _,imp in ipairs(r.imps) do

      if out.name == imp.thenpart then

        if not out.fuzzyset then
          out.fuzzyset = {}
          -- copy the first implication fuzzyset
          for k,v in ipairs(imp.fuzzyset) do
            out.fuzzyset[k] = { x=v.x, y=v.y }
          end
        else
          -- calculate the resulting fuzzyset with aggregation method
          for i,v in ipairs(out.fuzzyset) do
            -- computes the new value
            out.fuzzyset[i] = { x=v.x, y = fuzzy.aggregmethod( v.y, imp.fuzzyset[i].y ) }
          end
        end
      end
    end
  end
```

```lua
----------------------
-- Tmax
----------------------
function tmax(a1,a2)
  if a2 < a1 then
    return a1
  else
    return a2
  end
end
```

Bu kısımda 2.4.3.3'de açıkladığımız çıkarımı kullanarak output fuzzy kümelerini birleştiriyoruz.

Adım adım açıklarsak:

1. outvals, output'ların son halini depolayacak table'dır.
2. Her bir output için, i incelenen output'un index'i, out incelenen output olmak üzere:
   2.1. Her bir kural için, r incelenen kural olmak üzere:
      2.1.1. Her bir gerektirme için, imp incelenen gerektirme olarak:
         2.1.1.1. Eğer out'un ismi ve imp'in ismi aynı ise:
            2.1.1.1.1. Eğer out.fuzzyset yoksa:
               2.1.1.1.1.1. imp.fuzzyset'I out.fuzzyset' kopyala
            2.1.1.1.2. Eğer out.fuzzyset varsa:
               2.1.1.1.2.1. imp.fuzzyset'i aggregmethod'u kullanarak out.fuzzyset'e kopyala. aggregmethod ana class'da tanımlandığı üzere tmax'e eşittir. Çıkarımdan görüleceği üzere sadece y'si büyük olan değeri alacağız.

Bu işlemin sonunda out.fuzzyset'ler tüm kurallar uygulanmış, kontrolller yapılmış bir şekilde depolanır. Sonraki bölüme en yukardaki output loop'unu kapatmadan devam edeceğiz.

3.1.4.5 Defuzzification

En son kısımda output.fuzzyset'leri 2.5'deki çıkarım ile defuzzificated hale getirip elde ettiğimiz sonuçları döndürüyoruz.

```
321     --
322     -- Defuzzification...
323     --
324
325     -- call deffuzyfication method
326     out.value = fuzzy.defuzzmethod( out.fuzzyset )
327     -- add output value to result table
328     table.insert( outvals, out.value )
329   end
330
331   return unpack( outvals )
332 end
```

Bu kısımların da output loop'unun içinde olduğunu unutmadan adım adım açıklarsak :

1.1.1. defuzzmethod ana class'da tanımlandığı üzere 3.1.3'deki centroid fonksiyonuna eşittir. Bu fonksiyondan dönen crisp sayıyı out.value'ye atarız.
1.1.2. out.value'yi outvals table'ına ekleriz.
1.1.3. En baştaki output loop'unu kapatırız.
1.1.4. En son olarak da outvals table'ındaki sayıları unpackleyerek fonksiyondan döndürürüz.

3.2 Main Program

Burada LuaFuzzy'de tanımlı olan fonksiyonları cağırarak sistemimizi oluşturuyor ve sonra da çözüp ekrana basıyoruz.

İlk başta LuaFuzzy sistemini yüklüyoruz. Ondan sonra da fuzzy sistemi global bir değişken olan fuzzy değişkenine atıyoruz.

```
--
-- Load Fuzzy Lua Framework
--
require 'luafuzzy'

-----------------------------------
-- Fuzzy System
-----------------------------------

-- fuzzy is global
fuzzy = luafuzzy()
```

Sonra input değişkenlerini tanımlıyacağız. Bu değişkenler fuzzy sistemin bir parçası olduğu için tanımlarken "self" argümanına fuzzy sistemin kendisini atamalıyız. 1. Argüman değişkenin ismi, 2. argüman min value, 3. argüman ise max value olacak şekilde inputlarımızı tanımlıyoruz.

Sonra bu değişkenlerin linguistic kısımlarını tanımlamalıyız. Bu değişkenler input'un bir parçası olduğu için tanımlarken "self" argümanına input'un kendisini atamalıyız. 1. Argümana linguistic ismi, 2. argümana membership fonksiyonu, 3. argümana ise table içinde sırasıyla triangular sayının min, tepe, max değerlerini atıyoruz.

```
-------------------------------
-- Fuzzy Inputs
-------------------------------

--
-- Heat
--
local heat = fuzzy:addinp( 'heat', 7, 16 )
heat:addlingvar( 'verycold', trimf, { 7, 7, 9 } )
heat:addlingvar( 'cold', trimf, { 7, 9, 11 } )
heat:addlingvar( 'normal', trimf, { 10, 12, 14 } )
heat:addlingvar( 'hot', trimf, { 12, 14, 16 } )
heat:addlingvar( 'veryhot', trimf, { 13, 16, 16 } )

--
-- Pressure
--
local pressure = fuzzy:addinp( 'pressure', 1.75, 4.00 )
pressure:addlingvar( 'verybad', trimf, { 1.75, 1.75, 2.25 } )
pressure:addlingvar( 'bad', trimf, { 1.75, 2.25, 2.50 } )
pressure:addlingvar( 'normal', trimf, { 2.25, 2.75, 3.25 } )
pressure:addlingvar( 'good', trimf, { 2.50, 3.25, 3.50 } )
pressure:addlingvar( 'verygood', trimf, { 2.75, 4.00, 4.00 } )
```

Aynı işlemi output değeri için yaptıktan sonra sıra kuralları tanımlamaya geliyor. 25 tane kural olduğu ve hepsini ana programda tanımlarsak karmaşa yaratacağı için kuralları başka bir file'da tanımlayıp ana file'a yüklüyoruz.

```
-------------------------------
-- Fuzzy Outputs
-------------------------------

--
-- CO2
--
local co2 = fuzzy:addout( 'co2', 2, 6 )
co2:addlingvar( 'verybad', trimf, { 2, 2, 3 } )
co2:addlingvar( 'bad', trimf, { 2, 3, 4 } )
co2:addlingvar( 'normal', trimf, { 3, 4, 5 } )
co2:addlingvar( 'good', trimf, { 4, 5, 6 } )
co2:addlingvar( 'verygood', trimf, { 5, 6, 6 } )
```

```
---------------------------------
-- Rules
---------------------------------

local rules = require ("rulematrix")
```

Kuralları tanımlarken, kurallar fuzzy sistemin bir parçası olduğu için “self” argümanı yerine fuzzy sistemin kendisini atamalıyız. Fuzzy değişkenini global tanımladığımız için farklı bir file’da kullanabiliyoruz. 1. Argümana kuralın ağırlığını, 2. Argümana ise kuralda kullandığımız connection methodu atıyoruz. Geliştirdiğimiz kontrol sistemiminde kuralların ağırlığı eşit olduğu için hepsinin ağırlığına 1, hepsinin connection method’u AND olduğu için 2. argümanlarına “andmethod” değerini atıyoruz.

```
1  local M = {}
2
3  -------------------------------
4  -- Rules
5  -------------------------------
6
7  --
8  -- Rule 1
9  --
10 M.r1 = fuzzy:addrule( 1, 'andmethod' )
11 M.r1:addpremise( false, 'heat', 'verycold' )
12 M.r1:addpremise( false, 'pressure', 'verybad' )
13 M.r1:addimplic( false, 'co2', 'normal' )
```

Sonra bu kurallara öncülleri ve bu öncüller doğru ise çıkacak sonucu ekliyoruz. Öncüller ve gerektirmeler kuralın bir parcası olduğu için “self” argümanına kuralın kendisini atıyoruz. 1. argümana not kullanıp kullanmadığımızı, 2. argümana değişkenin ismini, 3. argümana da deişkenin alması gereken linguistic değişkeni atıyoruz.

```
57 -----------------------------------
58 -- Evaluate...
59 -----------------------------------
60 print( fuzzy:solve( 14, 3.25 ) )
61
```

En son olarak da ilk argüman sıcaklık değeri, 2. argüman ise basınç değeri olarak sistemi çözdürüp sonucu bastırıyoruz.

Bölüm 4
Uygulama

Bu bölümde bir input setinin program içinde nasıl değiştiğini inceleyeceğiz. Ayrıca step değişkenine göre sonucun ne kadar değiştiğini inceleyeceğiz.

4.1 Program

İlk olarak sıcaklığın 14, basınçın 3.25 olduğu giriş setini inceleyelim.

İlk başta program 3.1.4.1'de açıklandığı gibi her linguistic değişkenin üyelik değerini hesaplar. Sıcaklık'da veryhot üyelik değeri 0.33333, hot üyelik değeri 1, diğer değişkenlerin üyelik değerleri 0 olacaktır. Basınç'da ise good 1, verygood 0.4 ve diğerleri 0 olacaktır.

Sonraki aşamada program 3.1.4.2'de açıkladığı gibi kuralları triggerlayan değeri hesaplar. Bu değerler 19. kural için 1, 20. kural için 0.4, 24 ve 25. kurallar için 0.33333 diğer kurallar için ise 0 olacaktır.

Sonraki aşamada program 3.1.4.3'de açıklandığı gibi kuralların sonucunda elde edeceğimiz fuzzy set'leri hesaplar. 19, 20, 24 ve 25. kurallar dışında bu setler Y = 0 fonksiyonunu verir.

Sonraki aşamalarda bu setlerin bileşkesini alır ve defuzzification fonksiyonu uygulayıp sonucu bulur. Bizim inputlarımız için sonuç 4.3714285714286 olacaktır.

4.2 Step

Aynı inputlar için step değişkeni 0.001'iken sonuç 4.3714285714286'dir. Step değişkenini 0.0001 yaparsak sonuç 4.3714285714283 olur. Aradaki fark 0.0000000000003 olduğu ve çok küçük bir fark olduğu için step değişkeni 0'a olabilecek en yakın sayıyı olsa bile fark neredeyse hiç değişmeyecektir. Bu yüzden continuous bir evrende çalışsak bile discrete olarak kabul edip sonucu üretmemiz, sonucu neredeyse hiç etkilemeyecektir.