

**JARINGAN SYARAF TIRUAN JENIS AMN
(Associative Memory Networks):
CMAC, B-SPLINE dan RBF UNTUK
APLIKASI PEMODELAN DAN
PENGONTROLAN**

(BY IWAN SETIAWAN, ST, MT)

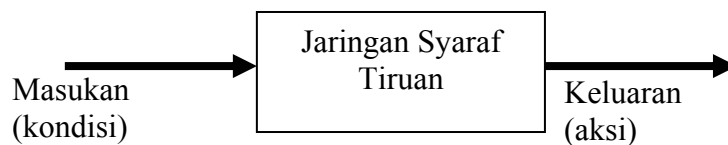
setiaone.iwan@gmail.com

Office: Automatic Control Laboratory, Electrical Engineering of UNDIP

Bab I. TINJAUAN JARINGAN SYARAF TIRUAN SECARA UMUM

Jaringan Syaraf Tiruan (JST) dibangun pada awalnya dengan tujuan untuk mengemulasikan (meniru) secara fungsional mekanisme kerja otak manusia dalam menyimpan, belajar, dan mengambil kembali pengetahuan yang tersimpan dalam sel saraf atau neuron.

Secara teknis Jaringan syaraf tiruan ini dapat dipandang sebagai fungsi pemetaan masukan keluaran sistem yang bebas model matematis (estimator bebas model), sistem ini memetakan kondisi ke aksi, seperti yang diperlihatkan gambar 1.1 dibawah:



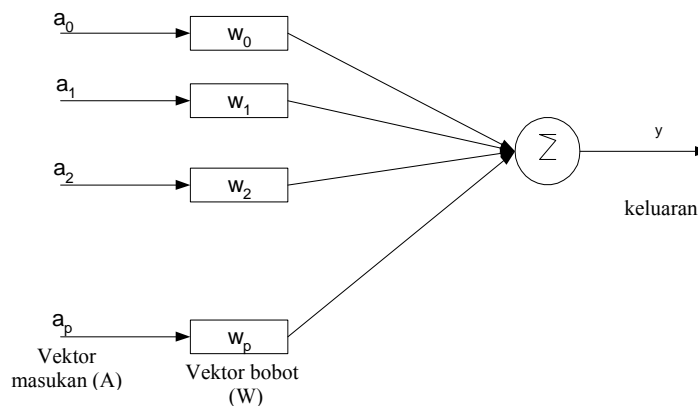
Gambar 1.1. Jaringan Syaraf Tiruan sebagai Fungsi Pemetaan

Ada banyak jenis JST yang telah diusulkan dalam literatur-literatur, masing-masing dengan kelebihan dan kekurangannya tergantung dari struktur dan metode pembelajarannya. Dalam aplikasinya pada bidang pemodelan dan pengontrolan, jenis jaringan syaraf yang cocok digunakan secara *off line* maupun *on line* umumnya adalah JST yang tergolong pada kelas AMN (*Associative Memory Networks*).

Dalam tulisan ini akan dibahas beberapa jenis jaringan syaraf yang tergolong pada kelas AMN tersebut secara khusus, yaitu antara lain: CMAC, B-Spline dan RBF.

I. 1. Adaptive Linear Combiner (ALC)

Salah satu komponen dasar pembangun jaringan syaraf tiruan jenis AMN adalah apa yang dikenal dengan nama *Adaptive Linear Combiner* (ALC) , seperti digambarkan dibawah ini:



Gambar I.2. Adaptive Linear Combiner

Seperti yang terlihat pada gambar 1.2, keluaran dari ALC (y) merupakan kombinasi linear dari vektor masukan (A) yang masing-masing diboboti oleh vektor pembobot (W), atau secara matematis dapat ditulis:

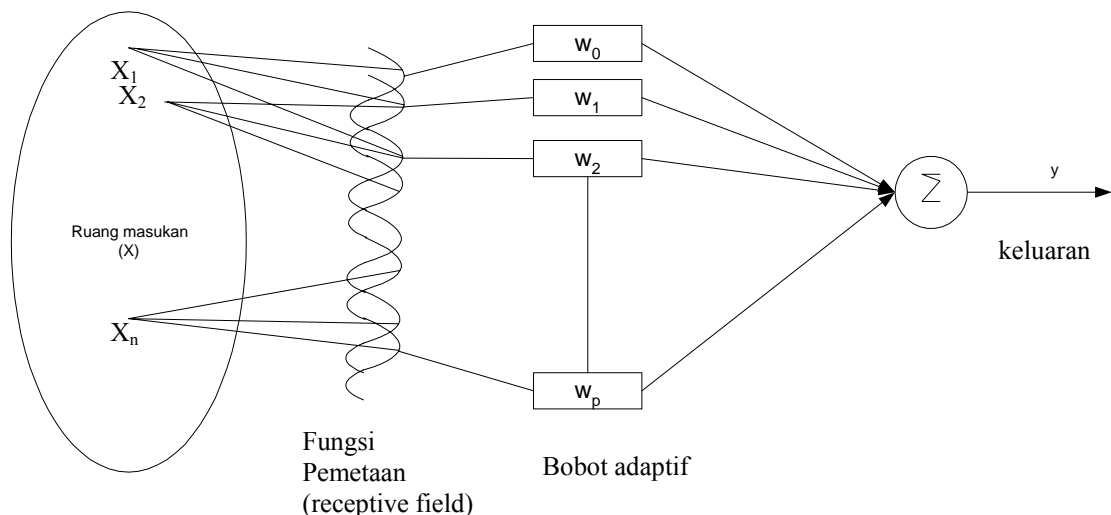
$$y = \sum_{i=0}^p a_i w_i \tag{1.1}$$

Dalam hal ini vektor pembobot (W) bersifat adaptif dalam pengertian nilai yang tersimpan dalam vektor ini dimungkinkan berubah dengan algoritma pembaharuan bobot tertentu.

I.2 Associative Memory Networks (AMN)

AMN adalah salah satu jenis jaringan syaraf tiruan yang menyimpan informasi (pengetahuan) dan pembelajaran secara lokal atau dikenal dengan istilah generalisasi lokal, hal ini menyebabkan laju pembelajaran pada AMN relatif lebih cepat dibandingkan dengan jenis jaringan syaraf tiruan lainnya. Berdasarkan sifatnya tersebut diatas penyimpanan informasi pada AMN juga bersifat lebih transparan, serta memiliki representasi fungsional dan dalam beberapa hal memiliki interpretasi logika fuzzy.

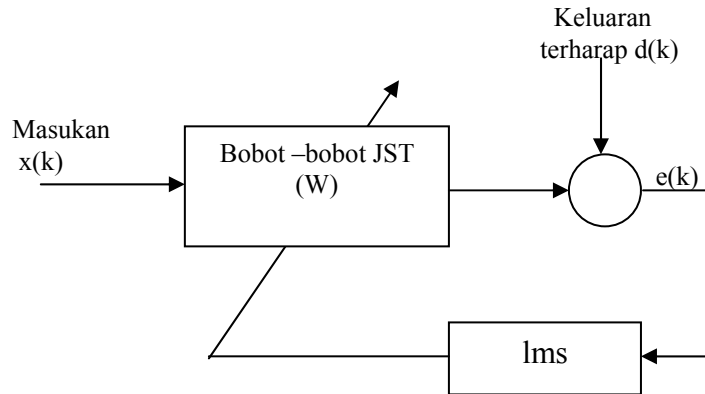
Gambar 1.3 memperlihatkan blok fungsional JST jenis AMN. Dari gambar terlihat bahwa untuk setiap nilai masukan tertentu, akan diaktifkan sejumlah fungsi nonlinear atau basis fungsi (jumlah fungsi yang diaktifkan tergantung jenis AMN dan perancangan awal), sedangkan keluaran (y) dihitung berdasarkan hasil pemetaan (keluaran) setiap basis fungsi yang diboboti oleh sejumlah bobot adaptif yang terasosiasi dengan basis fungsi tersebut.



Gambar 1.3. Diagram kotak AMN

I.2 Pembelajaran (*learning*)

Dalam kaitannya dengan AMN, pembelajaran pada dasarnya adalah metode untuk memperbaharui bobot-bobot adaptif secara iteratif. Blok Pembelajaran JST ini secara umum diperlihatkan pada gambar 1.4.



Gambar 1.4. Diagram kotak Pembelajaran JST

Sedangkan Salah satu metode pembelajaran yang populer karena keunggulan-keunggulannya adalah LMS :

$$W(k+1) = W(k) + 2 \cdot \alpha \cdot e(k) \cdot x(k) \quad 1.2$$

Dengan:

- $W(k)$: bobot –bobot JST pada cacah ke-k
- α : laju konvergensi ($0 < \alpha < 1$)
- $e(k)$: error pada cacah ke-k
- $x(k)$: masukan pada cacah ke-k

Bab II. PEMODELAN DAN PENGONTROLAN DENGAN JST

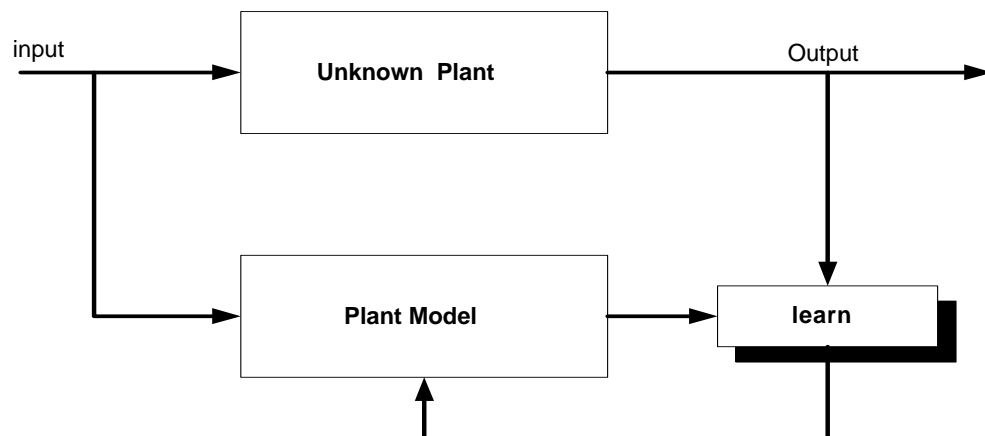
2.1 Pemodelan

Dalam kaitannya dengan bidang pengontrolan, pengetahuan mengenai model sistem atau plant yang akan dikontrol merupakan salah satu faktor penentu pemilihan strategi kontrol yang akan dirancang.

Secara umum ada 3 buah skema pemodelan yang penting untuk diketahui, masing-masing adalah: *direct modelling*, *invers modelling* dan *operator modelling*.

Direct Modelling

Tujuan dari direct modelling adalah mendapatkan model sedemikian sehingga karakteristik atau hubungan masukan keluaran dari model akan sama dengan karakteristik masukan keluaran plant atau sistem yang dimodelkan, skemanya dapat dilihat pada gambar 2.1 dibawah ini:



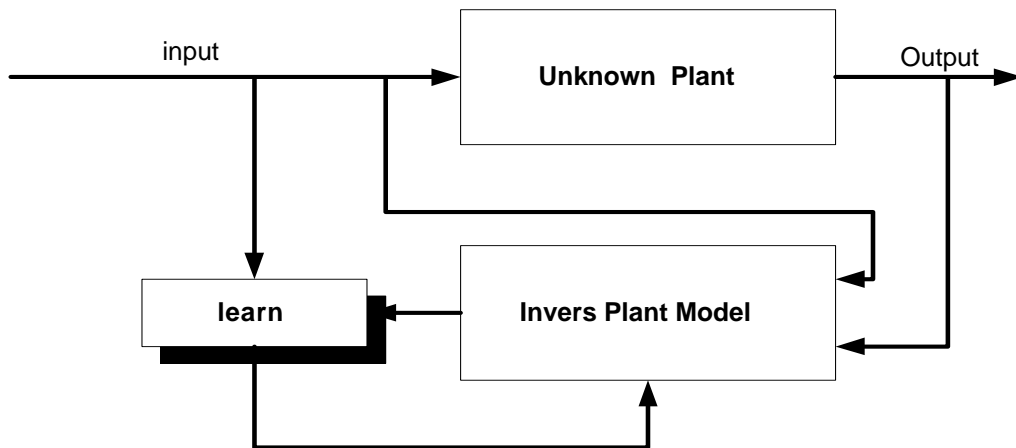
Gambar 2.1. Skema pemodelan Direct Modelling

Invers Modelling

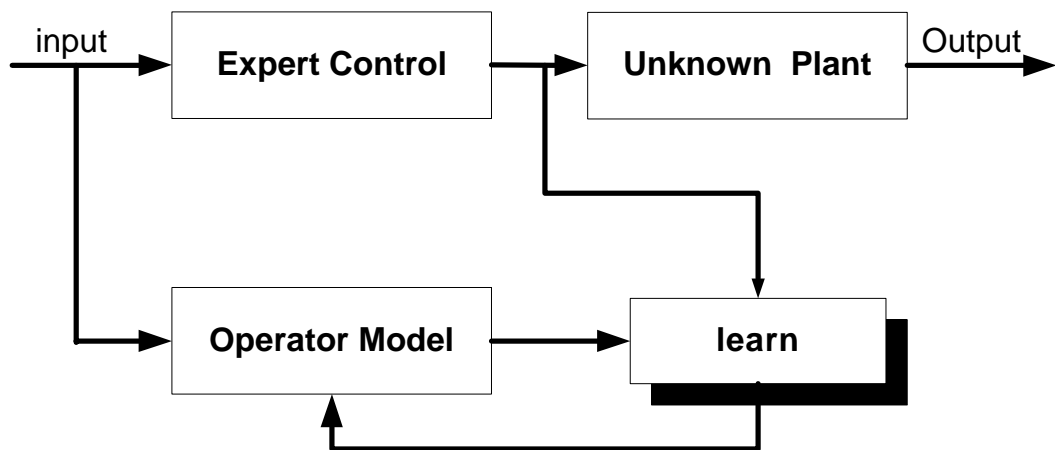
Tujuan dari pemodelan plant invers adalah mendapatkan formulasi pengendali sedemikian sehingga struktur plant/kontrol keseluruhan memiliki fungsi alih satuan, seperti terlihat pada gambar 2.2.

Operator Modelling

Tujuan dari pemodelan operator adalah mensintesis pengendali melalui pembelajaran dari operator (pakar). Algoritma pembelajaran berjalan secara parallel dengan operator, hal ini diperlihatkan gambar 3.3



Gambar 2.2. Skema pemodelan Invers Modelling



Gambar 2.3. Skema pemodelan Operator Modelling

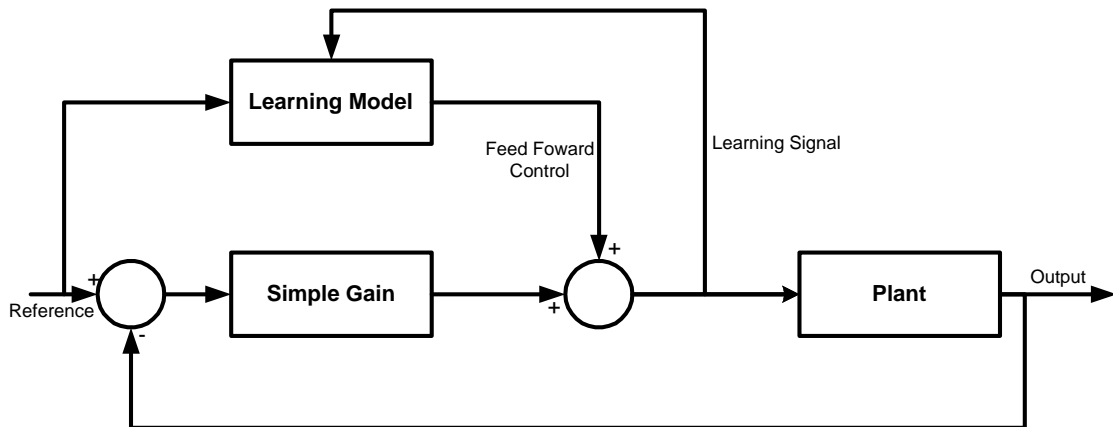
2.1 Pengontrolan

Masalah utama pengontrolan *plant* dengan jaringan syaraf tiruan terutama secara *on line* adalah tidak tersedianya sinyal kontrol terharap (*desired control signal*) dan umumnya hanya keluaran *plant* terharap saja yang dapat digunakan untuk melatih sistem kontrol tersebut.

Ada dua buah pendekatan yang berbeda yang telah diformulasikan dalam bidang kontrol adaptif : sistem kontrol adaptif langsung (*direct*) dan sistem kontrol adaptif tidak langsung (*indirect*). Sistem adaptif langsung dibangun berdasarkan model kontrol terharap secara eksplisit, sedangkan sistem kontrol adaptif tidak langsung membutuhkan formulasi model *plant* dan mensintesis hukum kontrol dengan menggunakan metode optimasi tertentu.

Fix Stabilising Controller

Salah satu arsitektur kontrol dengan menggunakan komponen jaringan syaraf tiruan secara *on line* diusulkan oleh kraft G. pada tahun 1990. Arsitektur ini dikenal dengan istilah *fixed stabilising controller* yang diagram kotak selengkapnya diperlihatkan pada gambar dibawah.



Gambar 2.4 Skema Kontrol Fix Stabilising Controller

Berdasarkan gambar tersebut terlihat bahwa pengontrolan dengan metode ini memiliki dua buah kalang, kalang pertama adalah kalang umpan balik biasa dengan kontrol proporsional, sedangkan kalang kedua adalah kalang kontrol dengan jaringan syaraf tiruan.

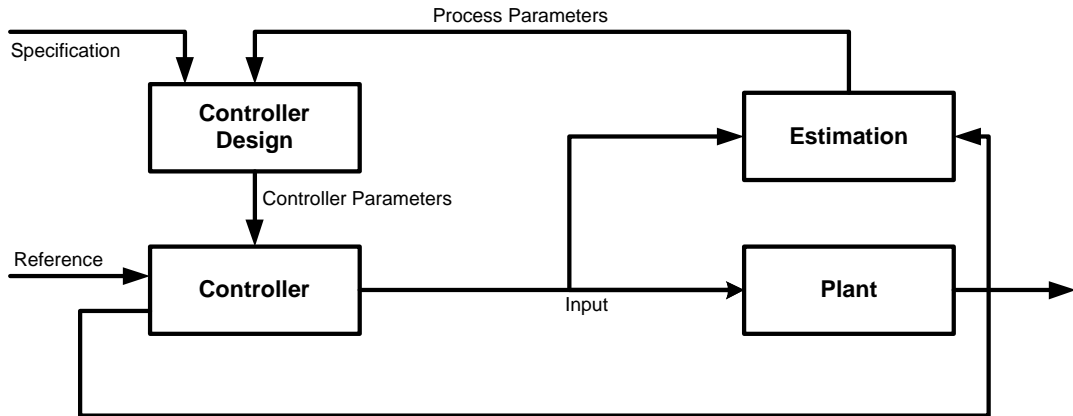
Sesungguhnya arsitektur kontrol gambar 2.4 tersebut adalah merupakan sistem kontrol adaptif langsung dengan keluaran gain proporsional digunakan untuk melatih modul jaringan syaraf tiruan. Gain proporsional ini dirancang sedemikian sehingga sistem kontrol keseluruhan stabil. Sinyal kontrol juga memberikan sinyal latih untuk jaringan syaraf tiruan tersebut. Unjuk kerja sistem kontrol ini tergantung pada titik operasi pengontrolan, walaupun demikian pelatihan iterative pada modul jaringan syaraf tiruan akan menyebabkan peningkatan unjuk kerja secara *on line*.

Secara fungsional, modul jaringan syaraf tiruan ini dapat dipandang sebagai model inverse yang langsung digunakan sebagai komponen pengendali secara *on line*. Sebagaimana diketahui tujuan utama pemodelan *plant* inverse adalah untuk memformulasikan sebuah pengendali, sedemikian sehingga arsitektur kontrol *plant* keseluruhan memiliki fungsi alih satuan.

Karena digunakan secara *on line* pada proses pengontrolan, jaringan syaraf tiruan yang dipilih sebagai komponen pengendali (yang secara fungsional juga merupakan model *plant* inverse) harus memiliki laju konvergensi yang relatif cepat, sehingga tidak semua jaringan syaraf tiruan cocok diimplementasikan sebagai komponen pengendali *plant* secara *on line*.

Self Tuning Regulator

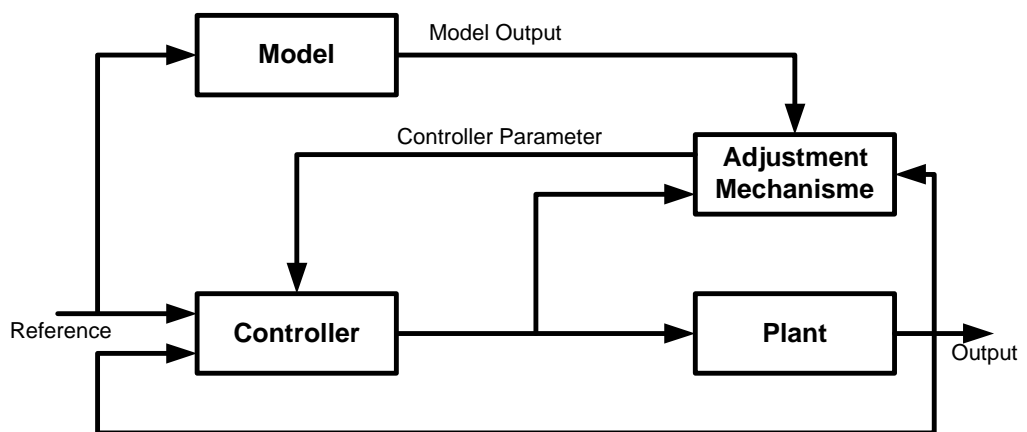
Self Tuning Regulator adalah skema kontrol adaptif indirect, dimana untuk mensintesis pengendali secara on line diperlukan model plant secara eksplisit, model ini umumnya diperoleh dengan cara pengidentifikasian plant atau menggunakan estimator bebas model. Struktur kontrolnya diperlihatkan pada gambar 2.5.



Gambar 2.5. Skema Kontrol Self Tuning Regulator

Model Reference Adaptive Control

MRAC adalah salah satu skema kontrol adaptif direct, dimana keluaran plant untuk sebuah setting point tertentu mengikuti keluaran dari model referensinya, seperti terlihat pada gambar 2.6



Gambar 2.6. Skema Kontrol Model Reference Adaptive Control

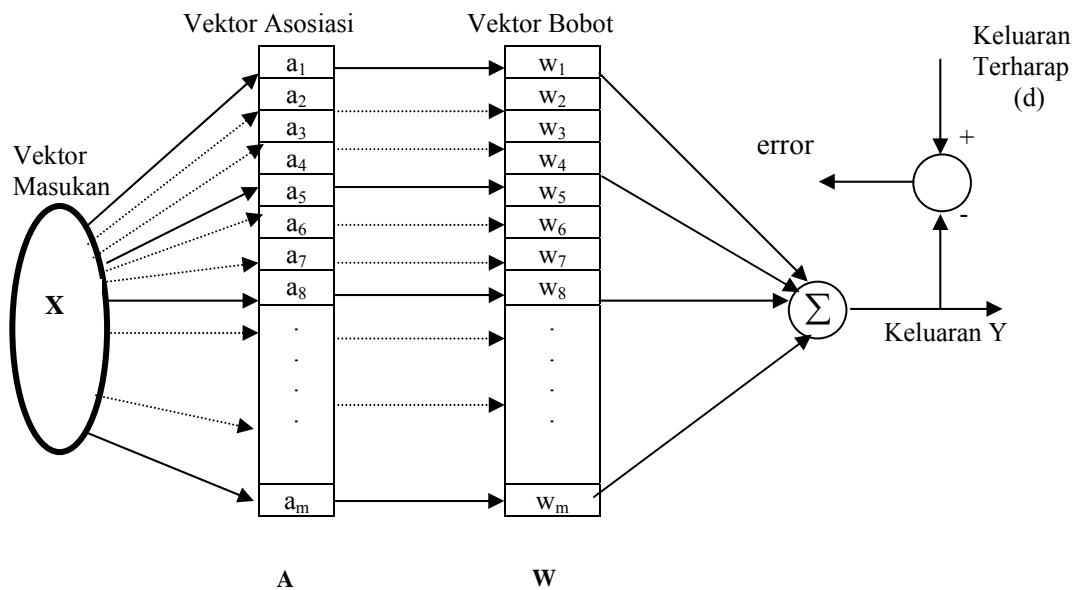
Bab III.
JARINGAN SYARAF TIRUAN :
CMAC

CMAC (*Cerebellum model articulation controller*) adalah salah satu jenis jaringan syaraf tiruan yang berusaha meniru pola kerja *cerebellum* (otak belakang) manusia. Setiap rangsangan yang diterima oleh otak belakang ini dipercayai hanya akan mengaktifkan sekitar satu prosen dari total sel otak belakang yang mungkin jutaan bahkan milyaran jumlahnya. Ditinjau dari jenis arsitektur atau strukturnya, CMAC dapat dimasukkan kedalam kelas AMN (*Associative Memory Network*).

Pada dasarnya struktur CMAC standar ini memiliki banyak kemiripan dengan jaringan perceptron yang diusulkan oleh Rosenblat. Gambar 3.1 memperlihatkan struktur dasar CMAC. Operasi CMAC ini dapat direpresentasikan kedalam dua buah pemetaan.

$$f: X \longrightarrow A$$

$$g: A \longrightarrow Y$$



Gambar 3.1. Model jaringan syaraf tiruan CMAC standar

dengan X adalah vektor ruang masukan malar berdimensi n, A adalah vektor asosiasi (memori konseptual) berdimensi m sedangkan Y adalah keluaran berdimensi satu (untuk keluaran lebih dari satu strukturnya dapat diubah secara langsung dengan menambah vektor asosiasi dan vektor bobot sebanyak tambahan keluaran yang diinginkan).

Dalam CMAC fungsi $f(X)$ memetakan setiap titik ruang masukan X kedalam sebuah vektor asosiasi A_p (sel-sel asosiasi yang aktif untuk sebuah titik ruang masukan tertentu), sedangkan $Y=g(A_p)$ besarnya tergantung pada nilai bobot W yang mungkin nilainya berubah selama proses pembelajaran (*learning*), dan dapat dirumuskan :

$$Y=g(A_p) = \sum_{i=1}^{\rho} W_{\rho i} \quad (3.1)$$

Dalam perancangan CMAC, langkah awal yang harus dilakukan adalah menentukan jangkauan nilai ruang masukan, nilai tersebut selanjutnya dikuantisasi. Setelah nilai kuantisasi masukan didapat maka langkah selanjutnya adalah memetakan nilai-nilai terkuantisasi ini pada sel-sel asosiasi CMAC berdasarkan parameter generalisasi lokal yang diinginkan.

Generalisasi lokal ini berarti untuk setiap nilai masukan CMAC yang berdekatan maka keluarannya akan berdekatan juga, sedangkan jika jarak nilai masukannya berjauhan, keluarannya masing-masing akan independen. Secara teknis, penentuan parameter generalisasi ini dilakukan dengan menetapkan jumlah sel asosiasi aktif untuk setiap cacah titik masukan, misal jika parameter generalisasi sama dengan 4 maka untuk tiap titik masukan akan mengaktifkan sejumlah 4 sel asosiasi. Sel-sel asosiasi tertentu dapat diaktifkan oleh titik masukan yang berbeda (jumlah titik masukan yang dapat mengaktifkan satu sel asosiasi ini tergantung dari nilai parameter generalisasi yang dipilih dan jarak tiap titik masukan).

Salah satu permasalahan utama pada CMAC adalah pemetaan titik ruang masukan pada vektor asosiasi A_p . Permasalahan pemetaan ini secara langsung akan menentukan unjukkerja kecepatan pengaktifan alamat sel-sel asosiasi CMAC. Dalam hal ini diperlukan sebuah algoritma yang secara efisien dapat secara langsung memetakan titik-titik ruang masukan pada sel-sel asosiasi tertentu (misal tanpa proses pencarian). Persamaan dibawah ini dapat digunakan sebagai generator alamat:

$$A(q,l) = 1 + \left\lceil \frac{q-l+d-1}{\rho} \right\rceil + (l-1) \left(\left\lceil \frac{s-l}{\rho} \right\rceil + 1 \right) \quad (2.2)$$

Sebagai bahan ilustrasi tinjau permasalahan berikut : Misal masukan terkuantisasi CMAC memiliki jangkauan nilai antara $q_{\min}=0$ sampai $q_{\max} = 9$ (jumlah elemen ruang masukannya $s = 10$) sedangkan parameter generalisasi yang dipilih $\rho = 3$, maka berdasarkan evaluasi generator alamat, bobot-bobot yang diaktifkan oleh masukan tertentu dapat dilihat pada gambar 3.2

Dalam kasus ruang masukan dua dimensi, generator alamat dapat diperluas menjadi :

$$A_p(q,l) = 1 + \left\lceil \frac{q_1-1-d_1+2}{\rho} \right\rceil + (l-1) \left(\left\lceil \frac{s_1-1}{\rho} \right\rceil + 1 \right) \left(\left\lceil \frac{s_2-1}{\rho} \right\rceil + 1 \right) + \left(\left\lceil \frac{s_1-1}{\rho} \right\rceil + 1 \right) \left(\left\lceil \frac{q_2-1-d_2+2}{\rho} \right\rceil \right) \quad (3.3)$$

0	1	2	3	4	5	6	7	8	9	q	
1	2		3		4						layer 1
5		6		7		8					layer 2
9			10		11		12				layer 3

Gambar 3.2. Hasil evaluasi generator alamat pemetaan

Untuk kasus ruang masukan n dimensi perumusan generator alamat dapat diperluas menjadi

$$A\rho(q,l) = 1 + \left\lceil \frac{q^l - l - d_l + 2}{\rho} \right\rceil + (l-1) \prod_{k=1}^n \left(\left\lceil \frac{s_k - 1}{\rho} \right\rceil + 1 \right) + \sum_{i=2}^n \left(\left\lceil \frac{q_i - l - d_i + 2}{\rho} \right\rceil \prod_{j=1}^{i-1} \left(\left\lceil \frac{s_j - 1}{\rho} \right\rceil + 1 \right) \right) \quad (3.4)$$

Beberapa Contoh Program Realisasi dan Aplikasi

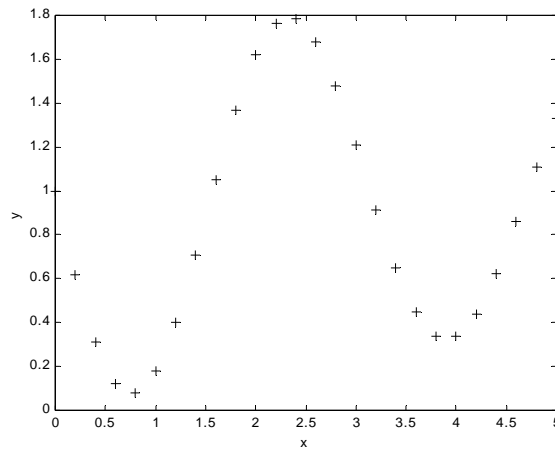
Contoh 1. Listing Program MATLAB : Diskretisasi masukan kontinyu sepanjang 8 bit (model ADC jenis SAC)

```
function d=adc8bitfs5(va);
%(Created By I'one)
% adc 8 bit dengan metoda sac (succesive approximation adc)
% full scale = 5 volt
% masukan 0 - 5 volt
% jumlah step =2^8 - 1 = 255
% resolusi =1/jumlah step = 1/255 =0.00392
% step size = full scale/jumlah step=5/255 = 0.0196 volt
%-----
%initial
if va>5
    va=5;
end
sz=0.0196; %step size
d=0;
n=8;
while n>=1
    d=bitset(d,n);vaaksen=d*sz;
    if vaaksen>va
        d=bitset(d,n,0);
    end
    n=n-1;
end
```

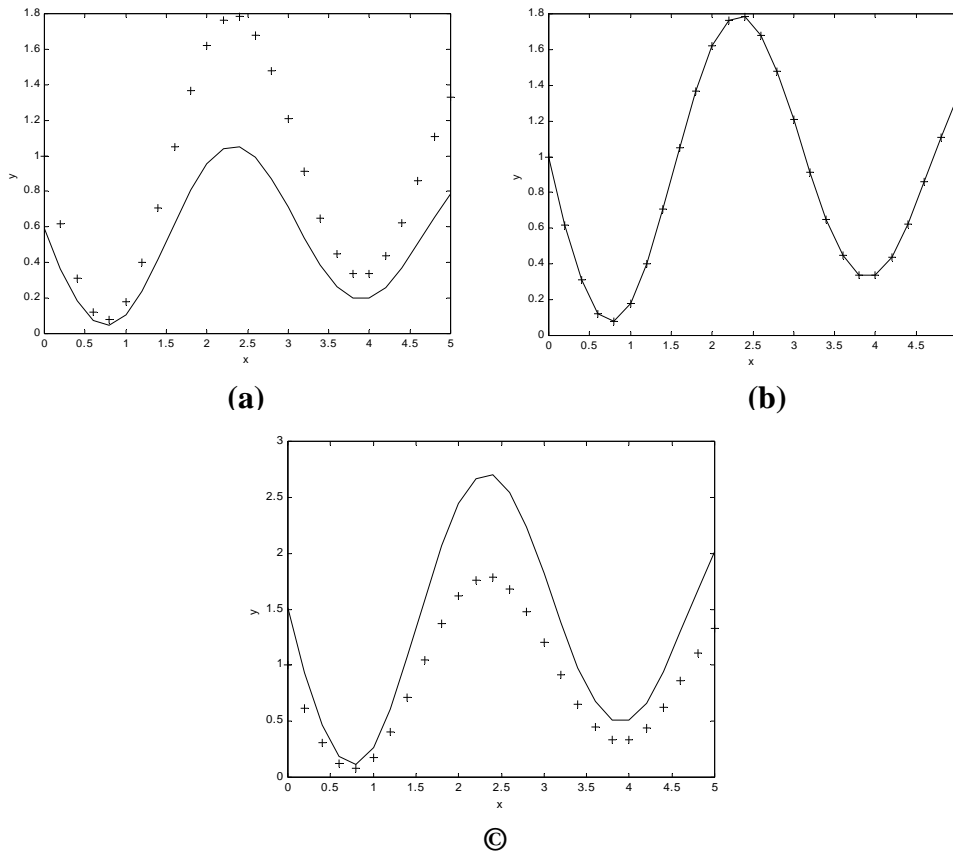
Contoh 2. Listing Program MATLAB : Realisasi Generator Alamat CMAC

```
function a=map1dim8bit(q);  
% (Created By I'one)  
% conceptual mapping one dimension  
% a : conceptual address  
% d : displacement vektor  
% q : quantized input  
% l : number of layer  
% rho : generalization parameter  
% Jumlah ruang Masukan: 256 -->0:255  
% note q = 0:s-1  
%-----  
rho=7;  
s=256; % jmlh ruang masukan  
l=[1:rho];  
d1=1;  
a=1+ceil((q-l-d1+2)/rho)+(l-1)*(ceil((s-1)/rho)+1);%a=alamat aktif
```

Contoh 3. Pemodelan data (fungsi Statis) dengan menggunakan CMAC: Data yang dimodelkan berupa data pengamatan masukan-keluaran (atau sebuah fungsi) seperti gambar dibawah



Keluaran model CMAC dengan generalisasi 7 dan jumlah pelatihan (iterasi) 4 kali untuk data Pengamatan tersebut dengan beberapa nilai laju konvergensi yang dipilih dapat dilihat dari gambar-gambar berikut:

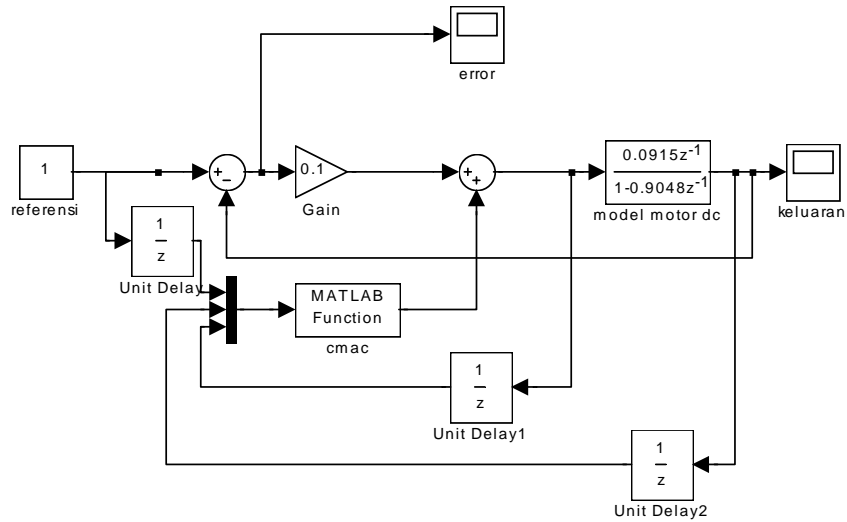


Gambar Contoh 3. Keluaran Model CMAC untuk laju konvergensi masing-masing (a) nilai laju konvergensi 0.1 (b) nilai laju konvergensi 0.5 (c) nilai laju konvergensi 0.9

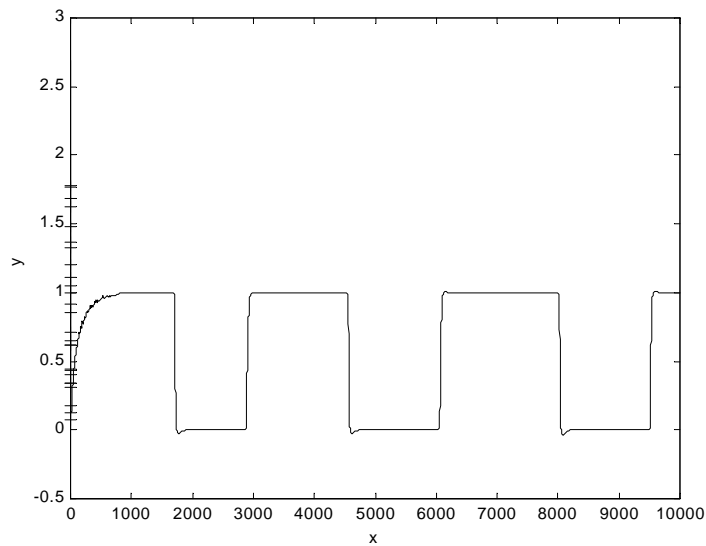
Listing Program selengkapnya:

```
% (Created By I'one)
clf
x=[0:0.2:5];
y=1-exp(-x/10).*sin(x/0.5);%fungsi yang akan dimodelkan
W(1:300)=0; %inisialisasi bobot CMAC
global W;
gen=7; %generalisasi (jumlah bobot aktif tiap cuplikan)
alpha=0.5; %laju konvergensi
for j=1:4 %jumlah iterasi
    for i=1:length(x)
        xx=adc8bitfs5(x(i)) ; %kuantisasi masukan sepanjang 8 bit
        a=map1dim8bit(xx);
        outcmac(i)=sum(W(a));
        error=y(i)-outcmac(i);
        W(a)=W(a)+2*alpha*error*ones(1,7)/gen;
    end
end
plot(x,y,'+');
xlabel('x');
ylabel('y');
hold on
plot(x,outcmac);
```

Contoh 4. Aplikasi CMAC pada Pengontrolan plant motor dc dengan menggunakan struktur Fix Stabilising Controller (dengan simulink)



Gambar Contoh 4. a Diagram Pengontrolan Fix Stabilising Controller dengan CMAC pada Simulink



Gambar Contoh 4.b Hasil keluaran Pengontrolan

Bab IV. JARINGAN SYARAF TIRUAN: B-SPLINE

B-spline adalah jenis JST yang dapat digolongkan dalam kelas AMN. Secara historis B-spline ini digunakan secara umum sebagai sebuah algoritma pencocokan fungsi (*surface fitting*).

Seperti halnya JST jenis AMN lainnya, keluaran B-spline merupakan kombinasi bobot-bobot adaptif dari sejumlah fungsi basis yang diaktifkan oleh masukan tertentu.

Hal menarik dari AMN jenis ini adalah adanya hubungan langsung antara JST dengan sistem fuzzy. Dari sudut pandang Fuzzy logic, fungsi-fungsi basis B-spline *univariate* merepresentasikan statemen-statement linguistik fuzzy, seperti 'error positif kecil', 'error besar', dan sebagainya. Hal ini menyebabkan JST B-spline dapat diinterpretasikan sebagai himpunan aturan fuzzy

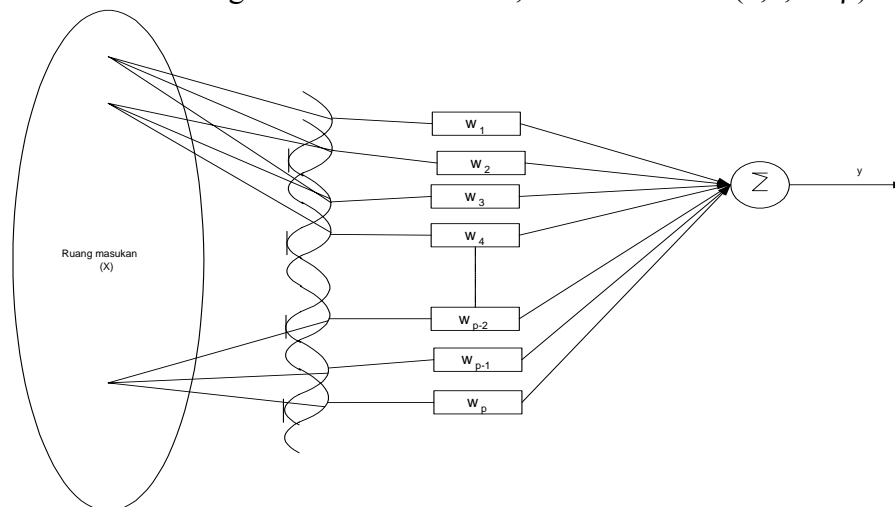
4.1 Notasi

Jumlah fungsi basis yang memberi kontribusi pada keluaran B-spline adalah konstan yaitu sebanyak ρ . Dalam hal ini ada kaitan langsung antara jumlah basis fungsi yang diaktifkan oleh masukan tertentu dengan orde basis B-spline yang dipilih. Untuk masukan X dengan dimensi n dan keluaran skalar y seperti diperlihatkan oleh gambar 4.1

Maka keluaran B-spline adalah:

$$y(k) = \sum_{i=1}^{\rho} a_i(k)w_i(k) \quad (4.1)$$

Dengan $w_i(k)$ adalah bobot yang terasosiasi dengan fungsi basis ke- i dan a_i adalah keluaran fungsi basis non zero ke- i , dalam hal ini $i=(1,2,\dots, \rho)$



Gambar 4.1. Diagram Blok JST B-spline

4.2. Univariate Basis Function

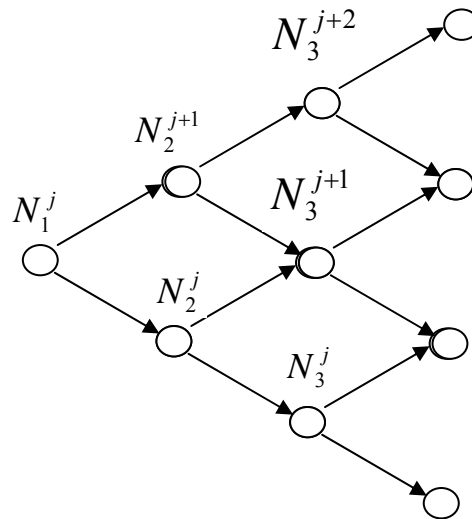
Keluaran basis fungsi yang diaktifkan oleh masukan tertentu (X) dapat dihitung dengan menggunakan hubungan recurrence dibawah ini:

$$N_k^j(x) = \left(\frac{x - \lambda_{j-k}}{\lambda_{j-1} - \lambda_{j-k}} \right) N_{k-1}^{j-1}(x) + \left(\frac{\lambda_j - x}{\lambda_j - \lambda_{j-k+1}} \right) N_{k-1}^j(x)$$

$$N_1^j(x) = 1 \text{ jika } x \in I_j (\lambda_{j-1}, \lambda_j)$$

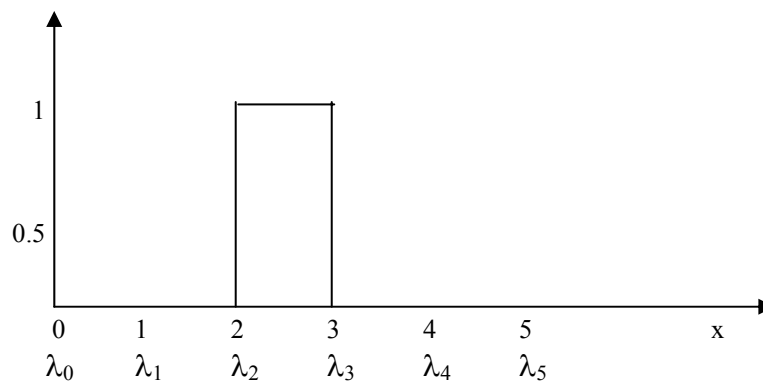
$$= 0, \text{ lainnya} \tag{4.2}$$

dengan λ_j adalah knot (posisi) ke-j dan $I_j = (\lambda_{j-1}, \lambda_j)$ adalah interval ke-j sedangkan k adalah orde dari basis fungsi tersebut, hubungan recurrence tersebut diilustrasikan oleh gambar 4.2 berikut:



Gambar 4.2. Hubungan recurrence

Fungsi Basis Orde 1 (konstan sebagian-sebagian)



Gambar 4.3. Fungsi Basis orde 1

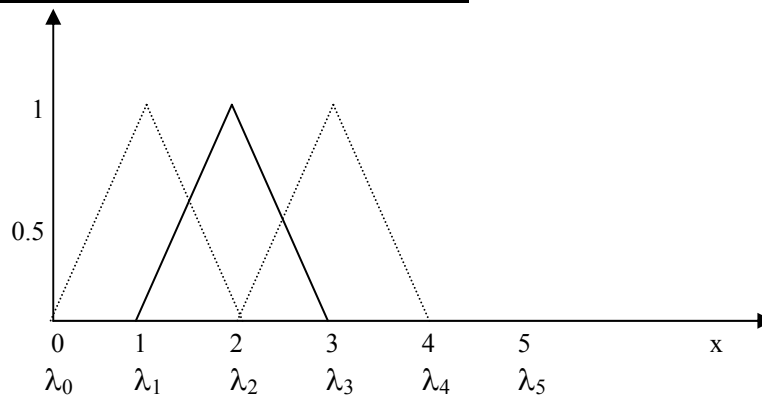
Misal N_1^j adalah fungsi basis ke-j dan I_j adalah interval ke-j (λ_{j-1}, λ_j) seperti terlihat pada gambar 4.3. Maka keluaran fungsi basisnya untuk masukan x adalah:

$$\begin{aligned} N_1^j(x) &= 1 \text{ jika } x \in I_j (\lambda_{j-1}, \lambda_j) \\ &= 0, \text{ lainnya} \end{aligned} \quad (4.3)$$

Secara matematis persamaan 4.3 diatas dapat diimplementasikan oleh fungsi berikut:

$$\begin{aligned} j &= \lceil x \rceil \\ N(j) &= 1 \end{aligned} \quad (4.4)$$

Fungsi Basis Orde 2 (linear sebagian-sebagain)



Gambar 4.4. Fungsi basis Orde 2

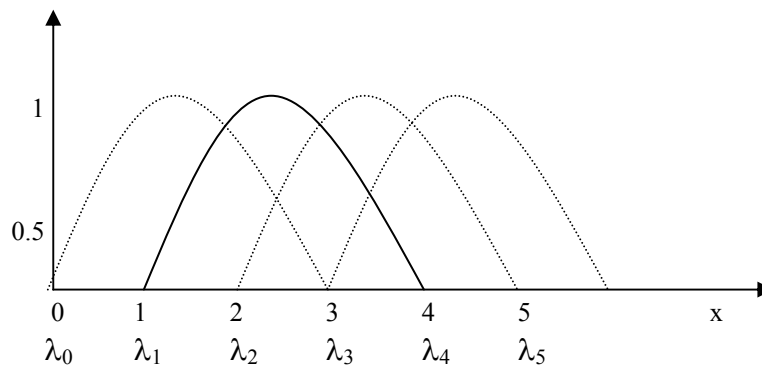
Keluaran basis untuk masukan x dihitung:

$$N_2^j(x) = \left(\frac{x - \lambda_{j-2}}{\lambda_{j-1} - \lambda_{j-2}} \right) N_1^{j-1}(x) + \left(\frac{\lambda_j - x}{\lambda_j - \lambda_{j-1}} \right) N_1^j(x) \quad (4.5)$$

dalam hal ini $N_1^{j-1}(x)$ dan $N_1^j(x)$ dihitung dengan menggunakan fungsi basis orde 1 berikut:

$$\begin{aligned} N_1^j(x) &= 1 \text{ jika } x \in I_j (\lambda_{j-1}, \lambda_j) \\ &= 0, \text{ lainnya} \end{aligned}$$

Fungsi Basis Orde 3 (kuadratik sebagian-sebagian)



Gambar 4.3. Fungsi basis Orde 3

Keluaran basis untuk masukan x dihitung:

$$N_3^j(x) = \left(\frac{x - \lambda_{j-3}}{\lambda_{j-1} - \lambda_{j-3}} \right) N_2^{j-2}(x) + \left(\frac{\lambda_j - x}{\lambda_j - \lambda_{j-2}} \right) N_2^j(x) \quad (4.6)$$

dalam hal ini $N_2^{j-1}(x)$ dan $N_2^j(x)$ dihitung dengan menggunakan fungsi basis orde 2 berikut:

$$N_2^j(x) = \left(\frac{x - \lambda_{j-2}}{\lambda_{j-1} - \lambda_{j-2}} \right) N_1^{j-1}(x) + \left(\frac{\lambda_j - x}{\lambda_j - \lambda_{j-1}} \right) N_1^j(x)$$

Beberapa Contoh Program Realisasi dan Aplikasi

Contoh 1. Listing Program Spline untuk beberapa fungsi Basis

```
function [j,N]=spline1(x)
%(created by i'one)
%j:interval ke-j
%N1:nilai fungsi pada interval ke-j
%ket : untuk x<=0<=7 ->ada 8 interval(bobot): 0-7
j=ceil(x);
N=1;

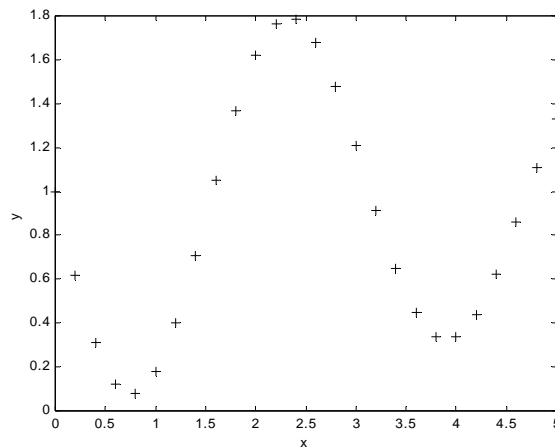
function [j,jj,N2j,N2jj]=spline2(x)
%(created by i'one)
%j menunjukkan interval
%N2 menunjukkan nilai fungsi keluarannya
%ket : untuk x<=0<=7 ->ada 8 interval(bobot): 0-7
[j,N1]=spline1(x);
jj=j+1;
N2j=((j-x)/1)*N1;
N2jj=((x-(j-1))/1);

function [j,jj,jjj,N3j,N3jj,N3jjj]=spline3(x)
%(created by i'one)
%j menunjukkan interval
%N3 menunjukkan nilai fungsi keluarannya
%ket : untuk x<=0<=7 ->ada 10 interval(bobot)
[j,jj,N2j,N2jj]=spline2(x);
jjj=jj+1;
N3j=((j-x)/2)*N2j;
N3jj=((x-(j-2))/2)*N2j+((j+1-x)/2)*N2jj;
N3jjj=((x-(j-1))/2)*N2jj;
```

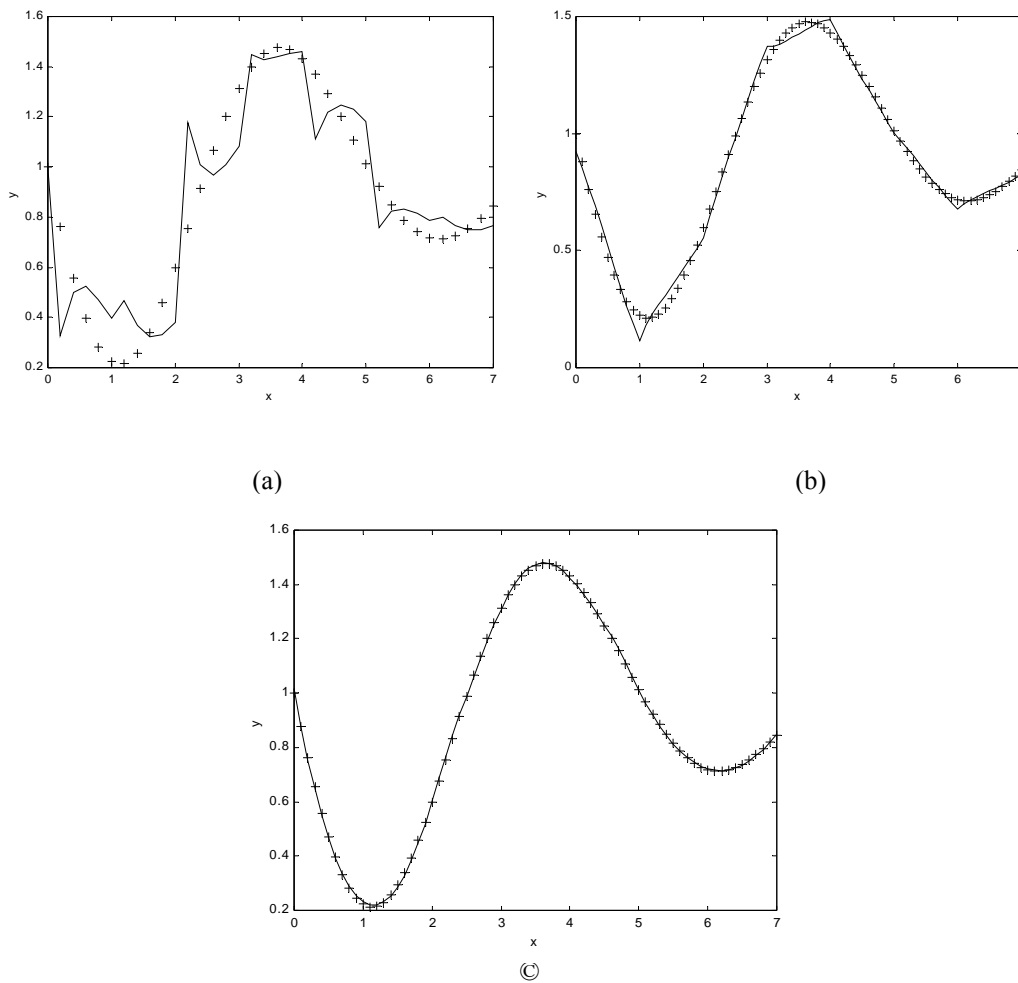
```
function [j,jj,jjj,jjjj,N4j,N4j2,N4j3,N4j4]=spline4(x)
%(created by i'one)
%j menunjukkan interval
%N3 menunjukkan nilai fungsi keluarannya
%ket : untuk x<=0<=7 ->ada 11 interval(bobot)

[j,jj,jjj,N3j,N3jj,N3jjj]=spline3(x);
jjjj=jjj+1;
N4j=((j-x)/3)*N3j;
N4j2=((x-(j-3))/3)*N3j+((j+1-x)/3)*N3jj;
N4j3=((x-(j-2))/3)*N3jj+((j+2-x)/3)*N3jjj;
N4j4=((x-(j-1))/3)*N3jjj;
```

Contoh 2. Pemodelan data (fungsi Statis) dengan menggunakan B-Spline: Data yang dimodelkan berupa data pengamatan masukan-keluaran (atau sebuah fungsi) seperti gambar dibawah



Keluaran model B-Spline dengan beberapa fungsi basis yang dipilih dan laju konvergensi 0.2 serta jumlah pelatihan (iterasi) 100 kali dapat dilihat dari gambar-gambar berikut:



Gambar Contoh 2. Keluaran Model B-spline untuk fungsi basis masing-masing (a) orde 1 (b) orde 2 (c) orde 3

Listing program untuk pemodelan dengan B-Spline orde 1 dapat dilihat dibawah (untuk pemodelan dengan B-Spline orde lebih tinggi tinggal mengganti fungsi bspline-nya saja)

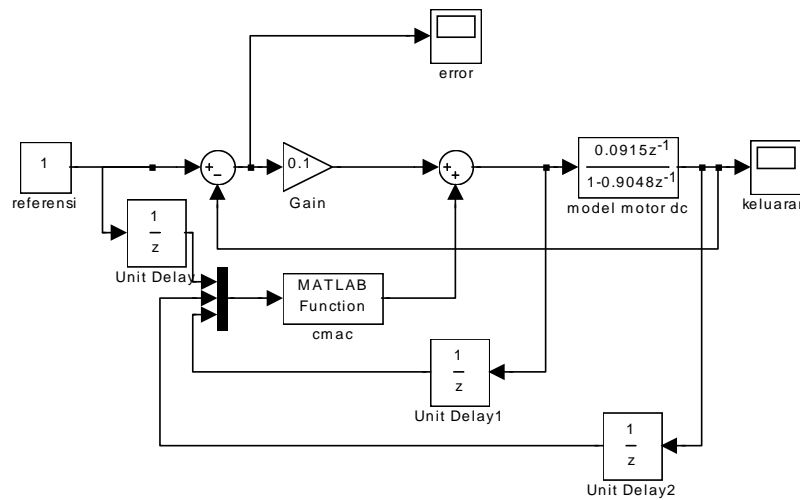
```
%pemodelan bspline orde 1
%created by i'one
clear;
x=[0:0.2:7];
y=1-exp(-x/5).*sin(x/0.8);
alpha=0.2;
%sediakan bobot sejumlah 8 (data max+1)
W(1:8)=0;
for k=1:100 %iterasi
    for i=1:length(x) % jmlh. data
        [j,N1]=splines1(x(i));
        indx=j+1;
        out(i)=W(indx)*N1;
    end
end
```

```

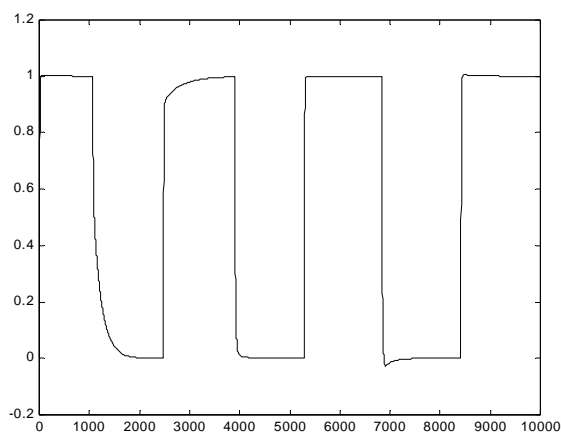
error=y(i)-out(i);
W(indx)=W(indx)+2*alpha*error*N1;
end
end
plot(x,y,'+',x,out);
xlabel('x');
ylabel('y');

```

Contoh 3. Aplikasi B-Spline pada Pengontrolan plant motor dc dengan menggunakan struktur Fix Stabilising Controller (dengan simulink)



Gambar Contoh 3. a Diagram Pengontrolan Fix Stabilising Controller dengan B-Spline pada Simulink



Gambar Contoh 3.b Hasil keluaran Pengontrolan

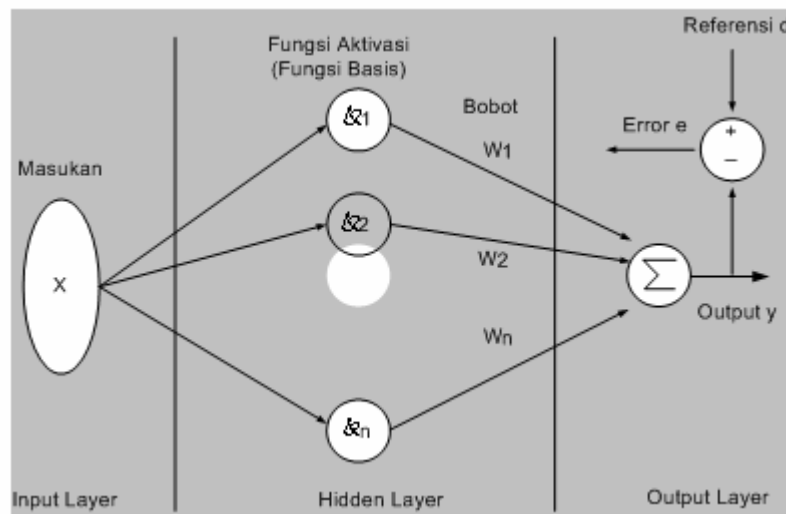
Bab V.
JARINGAN SYARAF TIRUAN:
Radial Basis Function (RBF)

RBF (ϕ) merupakan fungsi dimana keluarannya simetris terhadap *center* μ_c atau dinyatakan sebagai $\phi_c = \phi \|x - \mu_c\|$ dimana $\| \cdot \|$ merupakan vektor normal. Jaringan syaraf yang dibentuk dengan menggunakan fungsi aktivasi berupa fungsi *basis radial* dinamakan Jaringan Syaraf Tiruan RBF

5.1 Struktur Dasar RBF

Jaringan RBF terdiri atas 3 layer yaitu *layer* input, *hidden layer / kernel layer* (unit tersembunyi) dan *layer* output. Masing – masing unit tersembunyi merepresentasikan fungsi *aktivasi* yang berupa fungsi *basis radial*. Fungsi basis radial ini diasosiasikan oleh lebar dan posisi *center* dari fungsi *basis* tersebut.

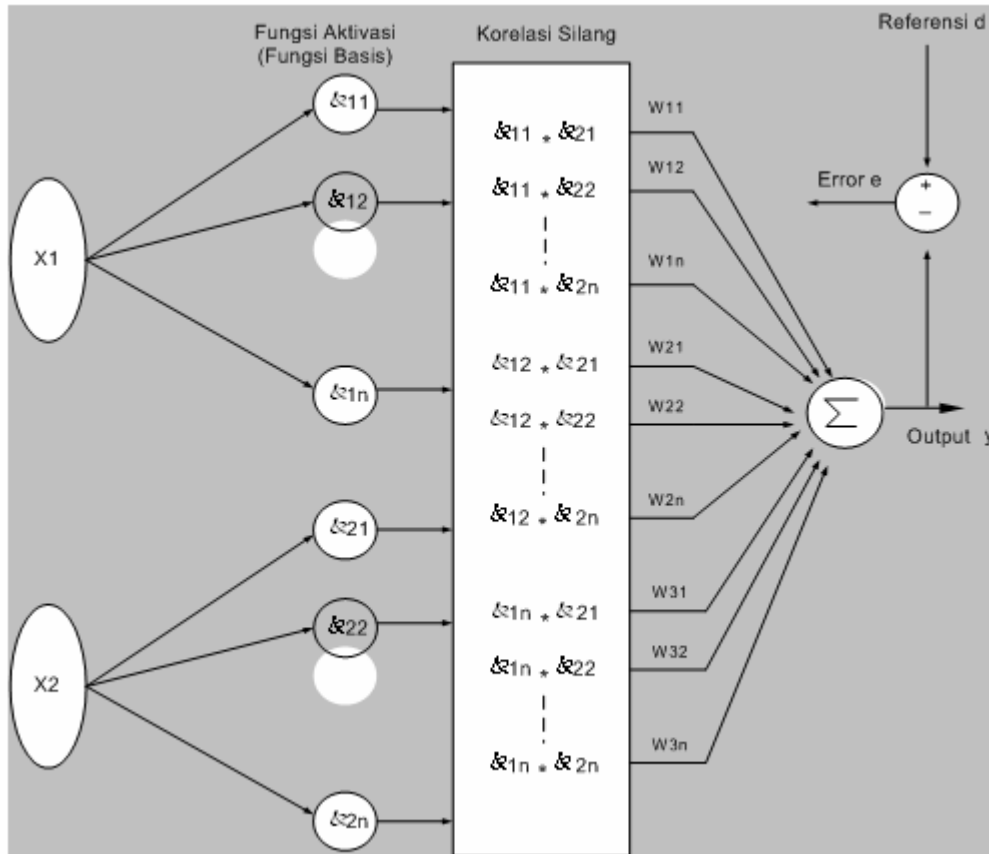
Struktur dasar jaringan RBF ditunjukkan pada Gambar 5.1.



Gambar 5.1 Struktur dasar jaringan syaraf RBF.

Jaringan syaraf RBF berbeda dengan jaringan syaraf CMAC. Setiap input dari jaringan ini akan mengaktifkan semua fungsi *aktivasi* pada *hidden layer*. Setiap unit dari *hidden layer* merupakan fungsi *aktivasi* tertentu yang disebut sebagai fungsi *basis*. Di dalam *hidden layer* terdapat sejumlah fungsi *basis* yang sejenis sesuai dengan perancangan. Setiap fungsi *basis* akan menghasilkan sebuah keluaran dengan bobot tertentu. Output jaringan ini merupakan jumlah dari seluruh output fungsi *basis* dikalikan dengan bobot masing – masing.

Untuk jaringan RBF dengan 2 masukan, proses pemetaanya ditunjukkan pada Gambar 5.2.



Gambar 5.2. Operasi jaringan syaraf RBF dengan 2 masukan

Setiap masukan akan mengaktifkan setiap fungsi *basis* pada jaringannya sendiri. Misalkan pada operasi masukan $[x_1 \ x_2]$. Masukan x_1 akan mengaktifkan setiap fungsi basis pada jaringan RBF pertama, sehingga masukan x_1 akan mengaktifkan fungsi *basis* ϕ_{11} , ϕ_{12} sampai dengan ϕ_{1n} . Masukan x_2 akan mengaktifkan setiap fungsi basis pada jaringan RBF kedua, sehingga masukan x_2 akan mengaktifkan fungsi *basis* ϕ_{21} , ϕ_{22} sampai dengan ϕ_{2n} . Langkah selanjutnya adalah melakukan korelasi silang antara setiap keluaran fungsi *basis* pada jaringan pertama dengan setiap keluaran fungsi basis pada jaringan kedua. Masing – masing hasil korelasi silang antar fungsi *basis* ini kemudian diboboti dengan bobot tertentu.

Pada jaringan RBF fungsi *basis* ini identik dengan dengan fungsi *gaussian* yang diformulasikan sebagai berikut :

$$\varphi_j = e^{-\frac{\|x-c_j\|^2}{2\sigma_j^2}} \quad (5.1)$$

Dimana :

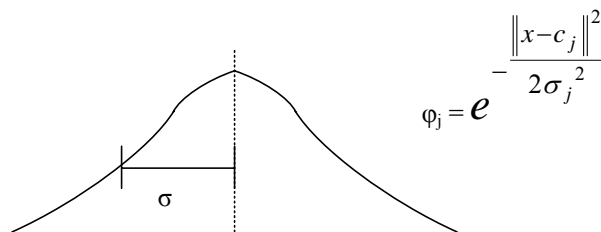
c_j = *Center* fungsi *gaussian* ke - j

σ_j = Lebar fungsi *gaussian* ke - j

x = Masukan

φ_j = Keluaran fungsi *basis* ke - j oleh masukan x

Representasi grafis fungsi gaussian (1 masukan) ditunjukkan pada Gambar 5.3 dibawah



Gambar 5.3 Fungsi *gaussian*

Setiap jaringan RBF biasanya menggunakan lebih dari 1 buah fungsi basis. Tiap – tiap fungsi *basis* mempunyai 1 *center* dan 1 bobot tertentu sehingga jumlah *center* dan bobot memori yang digunakan sama dengan jumlah fungsi *basis* yang digunakan. Untuk n buah masukan pada jaringan maka diperlukan bobot memori sebesar n x jumlah fungsi *basis* yang digunakan pada satu jaringan.

Semoga Bermanfaat ...

Referensi

Dari berbagai sumber