BAB II

BAB II

LANDASAN TEORI

2.1. Jaringan Saraf Tiruan

Dasar jaringan saraf tiruan adalah jaringan saraf manusia. Beberapa hal yang mendasari kerja jaringan saraf manusia diantaranya mengenai penyimpanan informasi dan daya ingat, akson dan dendrit yang bercabang-cabang sedemikian banyaknya, dan proses pengolahan informasi yang terdapat dalam jaringan saraf manusia. Pada jaringan saraf manusia, bila suatu sinyal tertentu melalui sinapsis secara berulang-ulang, maka sinapsis tersebut menjadi lebih mampu menghantarkan sinyal pada kesempatan berikutnya, hal ini lebih dikenal dengan istilah penyimpanan informasi dan daya ingat. Hal ini mendasari adanya proses belajar dalam jaringan saraf tiruan.

Akson dan dendrit yang bercabang-cabang sedemikian banyaknya pada jaringan saraf tiruan menunjukan adanya sistem paralel dan terdistribusi dalam jaringan saraf tiruan. Bedanya adalah akson dan dendrit bercabang-cabang dengan pola yang tak teratur sedangkan pada jaringan saraf tiruan membentuk pola tertentu. Jaringan saraf tiruan dapat belajar dari pengalaman, melakukan generalisasi dari contoh-contoh yang diperolehnya.

Jaringan saraf tiruan merupakan suatu sistem pengolah informasi yang mempunyai karakteristik menyerupai jaringan saraf biologis tubuh manusia.

Jaringan saraf tiruan telah dikembangkan dengan menggunakan model matematis untuk menirukan cara kerja jaringan saraf biologis, dengan berdasarkan asumsi :

1. Pengolah informasi terdiri dari elemen-elemen sederhana yang disebut neuron.
2. Sinyal dilewatkan dari satu neuron ke neuron yang lain melalui hubungan koneksi.
3. Tiap hubungan koneksi mempunyai nilai bobot tersendiri.
4. Tiap neuron mempergunakan fungsi aktivasi (biasanya tidak linear) terhadap masukan yang diterimanya untuk menentukan sinyal keluarannya.

Karakteristik jaringan saraf tiruan ditentukan oleh pola hubungan antar neuron (arsitektur), metode untuk menentukan nilai bobot tiap hubungan (*training*) dan ditentukan oleh fungsi aktivasi.

Jaringan saraf tiruan terdiri dari sejumlah elemen pengolah sederhana yang disebut neuron, unit atau sel. Tiap neuron terhubung dengan neuron yang lain melalui sambungan komunikasi dimana tiap sambungan mempunyai nilai bobot tersendiri. Nilai bobot ini menyediakan informasi yang akan digunakan oleh jaringan untuk memecahkan masalah. Tiap neuron mempunyai keadaan internal yang disebut level aktivasi yang terdiri atas fungsi masukan yang diterima. Suatu neuron mengirimkan nilai aktivasinya ke beberapa neuron yang lain. Sebuah neuron hanya dapat mengirim satu sinyal pada satu saat dan sinyal itu disebarkan ke beberapa neuron yang lain.

Jaringan saraf tiruan memiliki beberapa kelebihan yaitu :

1. Kemampuan mengakuisi pengetahuan walaupun dalam kondisi adanya gangguan dan ketidakpastian. Hal ini disebabkan jaringan saraf tiruan mampu melakukan generalisasi, abstraksi, terhadap karakteristik statistik dari data.
2. Kemampuan mempresentasikan pengetahuan secara fleksibel.
3. Kemampuan mentolerir suatu distorsi (*error/fault*), dimana gangguan kecil pada data dapat dianggap hanya *noise* (guncangan) belaka.
4. Kemampuan memproses pengetahuan secara efisien.

Meskipun memiliki banyak kelebihan, jaringan saraf tiruan juga memiliki sejumlah keterbatasan antara lain kekurangmampuannya dalam melakukan operasi-operasi numerik dengan presisi tinggi, operasi algoritma aritmatik, operasi logika, dan operasi simbolis serta lamanya proses pelatihan yang kadang-kadang membutuhkan waktu berhari-hari untuk jumlah data yang besar. Hal ini terjadi karena sulitnya mengukur *performance* sebenarnya dari jaringan saraf tiruan.

2.1.1 Jaringan Saraf Tiruan Propagasi Balik

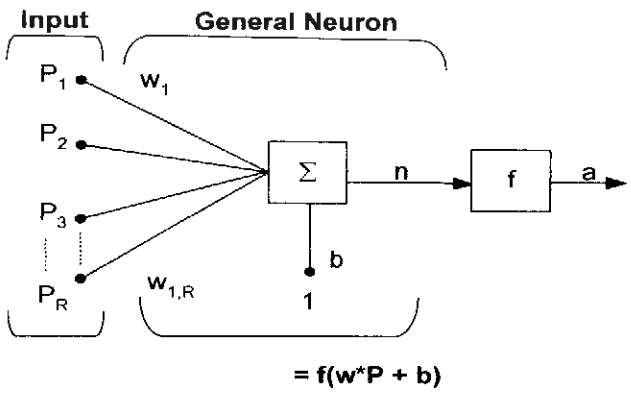
Jaringan saraf tiruan propagasi balik merupakan jaringan yang dikembangkan dengan algoritma Widrow-Hoff dalam jaringan *multilayer* dan fungsi-fungsi transfer tak linear yang diturunkan. Vektor-vektor input dan vektor-vektor target yang berhubungan digunakan untuk melatih jaringan sampai jaringan tersebut dapat memperkirakan sebuah fungsi, suatu vektor-vektor input dengan

vektor-vektor spesifik, atau mengklasifikasikan vektor-vektor sesuai yang kita inginkan.

Jaringan propagasi balik yang dilatih secara tepat biasanya akan menghasilkan output yang tepat ketika dicoba dengan input yang belum pernah dilatihkan padanya. Suatu input baru akan menyebabkan jaringan mengeluarkan output sama dengan output yang muncul ketika dilatih vektor-vektor input yang mirip dengan input baru tersebut. Dengan sifat generalisasi ini dimungkinkan untuk melatih sebuah jaringan hanya dengan suatu representasi data (input dan output) saja tanpa perlu mencobakan semua pasang vektor-vektor input dan output yang ada.

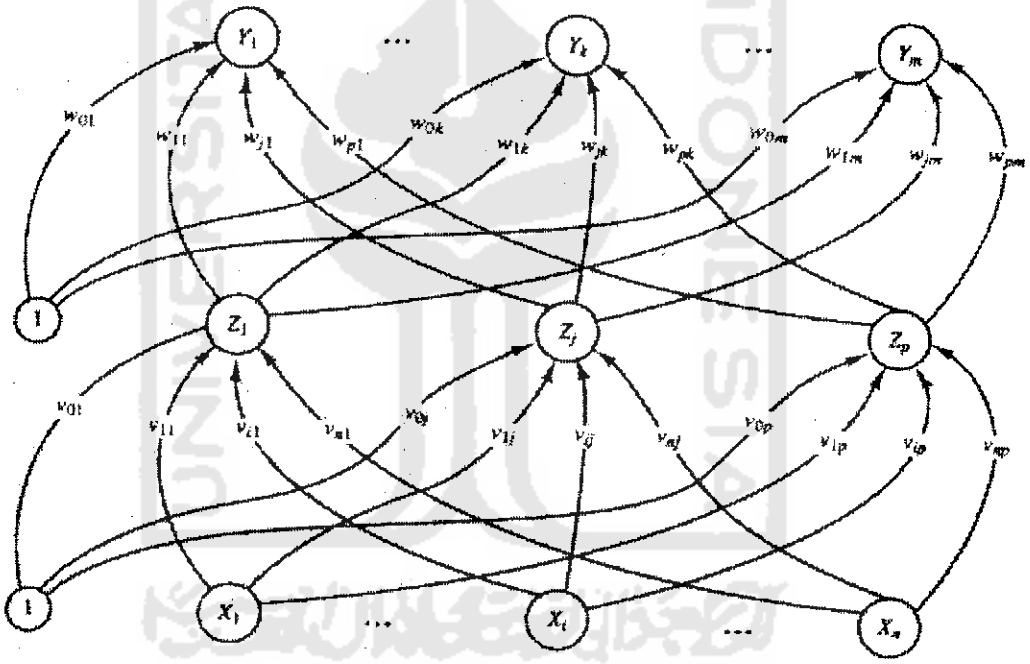
2.1.2. Arsitektur Jaringan Saraf Tiruan Propagasi Balik

Sebuah neuron dengan jumlah input R dan bobot w pada tiap koneksi input ditunjukkan pada gambar II.1. Jumlah input-input berbobot dan bias menghasilkan input total ke fungsi aktivasi *f*. Neuron-neoron dapat menggunakan sembarang fungsi aktivasi *f* yang ada diferensialnya untuk menghasilkan output.



Gambar II.1 Satu *Layer* Jaringan Sebagai Penyusun *Multilayer*

Arsitektur jaringan yang paling umum digunakan untuk jaringan saraf tiruan propagasi balik adalah jaringan lapisan banyak. Lapisan ini dapat dikelompokkan kedalam tiga bagian yaitu lapisan input, lapisan tersembunyi dan lapisan output. Pada lapisan tersembunyi, dimungkinkan untuk menggunakan lebih dari satu lapisan. Gambar II.2 menunjukan jaringan saraf tiruan propagasi balik dengan satu lapisan tersembunyi.



Gambar II.2 Struktur Jaringan Saraf Tiruan Propagasi Balik Dengan Satu Lapisan Tersembunyi

Pada gambar, lapisan jaringan saraf tiruan propagasi balik terdiri dari tiga lapisan jaringan yaitu :

Unit masukan (Input: X_1, X_i, X_n) : *Nodes* yang terdapat dalam lapisan ini disebut sebagai unit input yang mengubah suatu input yang dimasukan ke dalam

bentuk sinyal yang dapat dimengerti sistem dan diteruskan ke dalam jaringan untuk diproses.

Unit tersembunyi (*Hidden* : Z_1, Z_j, Z_p) : *Nodes* yang berada pada lapisan ini disebut sebagai *hidden unit*, yaitu unit-unit yang tidak berhubungan secara langsung dengan dunia luar (misal : informasi input). Lapisan inilah yang membuat jaringan memiliki sifat non *linear* karena terjadi proses komputasi.

Unit keluaran (Output : Y_1, Y_k, Y_m) : Unit ini merupakan sebutan untuk *nodes* yang berada di dalam lapisan ini, yang keluar dari proses sehingga dapat ditafsirkan sesuai dengan kasus yang dikehendaki.

Pada jaringan propagasi balik, fungsi aktivasi yang digunakan haruslah fungsi yang dapat dicari turunannya, kontinyu (*continue*) dan menurun secara tajam.

2.1.3. Fungsi Aktivasi Jaringan Saraf Tiruan Propagasi Balik

Suatu fungsi aktivasi untuk sebuah jaringan saraf tiruan propagasi balik harus memiliki beberapa karakteristik penting yaitu harus kontinyu, dapat dideferensialkan dan menurun secara tajam. Bentuk fungsi aktivasi tersebut adalah sebagai berikut :

***Sigmoid* (*Logsig*) :**

$$f(x) = \frac{1}{1 + e^{-x}} \qquad (2.1)$$

dengan : $f'(x) = f(x)\,[1 - f(x)]$

***Tansig* (*Tansigmoid*) :**

$$y = f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \qquad (2.2)$$

dengan : $f'(x) = [1 + f(x)]\,[1 - f(x)]$

Fungsi *Linear* (*Purelin*) :

$$y = f(x) = x \qquad (2.3)$$

dengan : $f'(x) = 1$

Fungsi aktivasi *sigmoid* (*logsig*) biner mempunyai nilai output berkisar antara 0 dan 1, *tansigmoid* (*tansig*) nilai outputnya berkisar antara -1 dan +1. Sedangkan *purelin* nilai outputnya sama dengan nilai inputnya. Ketiga fungsi aktivasi tersebut biasanya digunakan pada jaringan saraf tiruan propagasi balik.

2.1.4. Pembelajaran Jaringan Saraf Tiruan

Pembelajaran pada jaringan saraf tiruan dibedakan menjadi 2 bagian yaitu:

1. Pembelajaran terawasi (*supervised learning*) yaitu metode pembelajaran dengan output yang diharapkan telah diketahui sebelumnya. Salah satu contoh pembelajaran dengan menggunakan operasi AND. Pada proses pembelajarannya, satu pola input akan diberikan ke satu neuron pada lapisan input. Pola ini akan dirambatkan disepanjang jaringan saraf hingga sampai ke neuron pada lapisan output yang nantinya akan dicocokkan dengan pola output targetnya. Apabila terjadi perbedaan antara pola output hasil pembelajaran dengan pola target, maka disini akan muncul *error*. Apabila

nilai *error* masih cukup besar, mengindikasikan bahwa masih perlu dilakukan lebih banyak pembelajaran.

2. Pembelajaran tidak terawasi (*unsupervised learning*) yaitu metode pembelajaran yang tidak memerlukan target output. Pada metode ini, tidak dapat ditentukan hasil yang seperti apakah yang diharapkan selama proses pembelajaran. Selama proses pembelajaran, nilai bobot disusun dalam suatu *range* tertentu tergantung pada nilai input yang diberikan. Tujuan pembelajaran ini adalah mengelompokan unit-unit yang hampir sama dalam suatu area tertentu. Pembelajaran ini biasanya sangat cocok untuk pengelompokan (klasifikasi) pola.

2.1.5. Algoritma Belajar Jaringan Saraf Tiruan Propagasi Balik

Algoritma pelatihan pada jaringan saraf tiruan propagasi balik adalah sebagai berikut :

Langkah 0 : Inisialisasi nilai bobot (diatur pada nilai acak yang kecil).

Langkah 1 : Selama kondisi berhenti masih tidak terpenuhi, lakukan langkah 2-9.

Langkah 2 : Untuk setiap pasangan pelatihan, lakukan langkah 3-8.

Perambatan maju (*Feedforward*) :

Langkah 3 : Tiap unit masukan (X_i, i=1,...,n) menerima sinyal masukan x_i dan menyebarkan sinyal itu ke semua unit pada lapisan di atasnya (lapisan tersembunyi).

Langkah 4 : Setiap unit lapisan tersembunyi (Z_j, j=1,...,p) dihitung nilai masukan dengan menggunakan nilai bobotnya :

$$z\_in_j = \upsilon_{0j} + \sum_{i=1}^{n} x_i \upsilon_{ij} \qquad (2.4)$$

V_{0j} = bias pada unit tersembunyi j, kemudian dihitung nilai keluaran dengan menggunakan fungsi aktivasi yang dipilih :

$$z_j = f(z\_in_j) \qquad (2.5)$$

Hasil fungsi tersebut dikirim ke semua unit pada lapisan di atasnya (unit keluaran).

Langkah 5 : Tiap unit keluaran (Y_k, k=1,..,m) dihitung nilai masukan dengan menggunakan nilai bobotnya :

$$y\_in_k = w_{0k} + \sum_{j=1}^{p} z_j w_{jk} \qquad (2.6)$$

w_{0k} = bias pada unit keluaran k, kemudian dihitung nilai keluaran dengan menggunakan fungsi aktivasinya :

$$y_k = f(y\_in_k) \qquad (2.7)$$

Perambatan-balik kesalahan :

Langkah 6 : Tiap unit keluaran (Y_k, k=1,..,m) menerima pola target yang berhubungan dengan pola masukan pelatihan, dan kemudian dihitung kesalahan informasinya :

$$\delta_k = (t_k - y_k) f'(y\_in_k) \qquad (2.8)$$

Kemudian dihitung koreksi nilai bobotnya yang kemudian akan digunakan untuk memperbaharui nilai bobot w_{jk} :

$$\Delta w_{jk} = \alpha \delta_k z_j \tag{2.9}$$

Hitung koreksi nilai biasnya yang kemudian akan digunakan untuk memperbaharui nilai w_{0k} :

$$\Delta w_{0k} = \alpha \delta_k \tag{2.10}$$

dan kemudian nilai δ_k dikirim ke unit pada lapisan di bawahnya.

Langkah 7 : Setiap unit lapisan tersembunyi (Z_j, j=1,...,p) dihitung perubahan masukan yang dari unit-unit pada lapisan di atasnya :

$$\delta\_in_j = \sum_{k=1}^{m} \delta_k w_{jk} \tag{2.11}$$

Kemudian nilai tersebut dikalikan dengan nilai turunan dari fungsi aktivasinya untuk menghitung informasi kesalahannya :

$$\delta_j = \delta\_in_j f'(z\_in_j) \tag{2.12}$$

Hitung koreksi nilai bobot yang kemudian digunakan untuk memperbaharui nilai v_{ij} :

$$\Delta vij = \alpha \delta_j x_i \tag{2.13}$$

dan hitung nilai koreksi bias yang kemudian digunakan untuk memperbaharui v_{oj} :

$$\Delta v_{oj} = \alpha \delta_j \tag{2.14}$$

Memperbaharui nilai bobot dan bias :

Langkah 8 : Tiap unit (Y_k, k=1,…,m) keluaran diperbaharui nilai bias dan bobotnya (j=0,…,p):

$$w_{jk}(baru) = w_{jk}(lama) + \Delta w_{jk} \quad (2.15)$$

dan pada tiap unit lapisan tersembunyi (Zj, j = 1,…,p) diperbaharui bias dan bobotnya (i = 1, …,n):

$$v_{ij}(baru) = v_{ij}\ (lama) + \Delta v_{ij} \quad (2.16)$$

Langkah 9 : Menguji apakah kondisi berhenti sudah terpenuhi. Kondisi berhenti ini terpenuhi jika nilai kesalahan yang dihasilkan lebih kecil dari nilai kesalahan referensi.

Suatu jangka waktu (*epoch*) adalah satu set putaran vektor-vektor pelatihan. Beberapa *epoch* diperlukan untuk pelatihan sebuah jaringan saraf tiruan propagasi balik. Dalam algoritma ini dilakukan perbaikan bobot setelah masing-masing pola pelatihan dimasukan. Setelah pelatihan selesai bobot-bobot yang telah diperbaiki disimpan.

2.1.6 Prosedur Pengujian Jaringan Saraf Tiruan Propagasi Balik

Setelah pelatihan, sebuah jaringan saraf propagasi balik diaplikasikan hanya pada fase umpan maju (*feedforward*). Prosedur aplikasinya adalah sebagai berikut:

Langkah 0 : Inisialisasi bobot awal (hasil dari pelatihan).

Langkah 1 : Untuk tiap vektor masukan, lakukan langkah 2-4.

Langkah 2: Untuk setiap unit masukan, distribusikan masukan *xi* kesetiap unit diatasnya (unit dalam) .

Langkah 3 : Untuk j = 1,...,p ; gunakan persamaan berikut :

$$z\_in_j = \upsilon_{0j} + \sum_{i=1}^{n} x_i \upsilon_{ij} \tag{2.17}$$

$$z_j - f(z\ in_j) \tag{2.18}$$

Langkah 4 : Untuk k = 1,...,m ; gunakan persamaan sebagai berikut :

$$y\_in_k = w_{0k} + \sum_{j=1}^{p} z_j w_{jk} \tag{2.19}$$

$$y_k = f(y\_in_k) \tag{2.20}$$

2.1.7 Perbaikan Bobot Dengan Momentum dan Pembelajaran Adaptif (*Adaptive Learning Rate*)

Salah satu pengembangan dalam jaringan saraf tiruan propagasi balik untuk memperbaiki bobot digunakan parameter *momentum* (β). JST PB dengan momentum mempunyai sifat konvergensi lebih cepat dari JST PB standar. Persamaan yang digunakan untuk pengubahan bobot pada JST PB dengan *momentum* adalah :

$$w_{jk}(t+1) = w_{jk}(t) + \alpha\, \delta_k z_j + \beta[w_{jk}(t) - w_{jk}(t-1)] \tag{2.21}$$

dan

$$v_{ij}(t+1) = v_{ij}(t) + \alpha\, \delta_j x_i + \beta[v_{ij}(t) - v_{ij}(t-1)] \tag{2.22}$$

Dalam proses belajar ini, ada satu hal yang perlu diperhatikan berapakah nilai optimal untuk laju belajar (*learning rate*) α dan *momentum* β ? sebagian besar jaringan saraf tiruan propagasi balik menggunakan $\alpha = 0.25$ dan $\beta = 0.9$ dapat memberikan hasil terbaik pada banyak masalah. Tetapi belum ada ketetapan besar nilai α dan β yang harus digunakan dalam proses belajar. Sebab dalam kenyataannya nilai α dan β tergantung pada permasalahan yang terjadi.

Salah satu metode pelatihan jaringan saraf tiruan propagasi balik yang menggunakan *momentum* dan *learning rate* yaitu *traingdx* yang merupakan metode pelatihan cepat. *Traingdx* merupakan suatu fungsi pelatihan jaringan yang memperbaharui nilai bobot dan bias berdasarkan *gradient descent momentum* dan *learning rate*.

2.2 Sistem Pengenal Sinyal Tutur

Sinyal analog dari penutur perlu dilakukan parameterisasi sinyal. Parameter sinyal merupakan proses konversi sinyal tutur yang telah dicuplik dengan menggunakan algoritma yang handal ke suatu bentuk yang lebih efektif. Hasilnya adalah suatu bentuk vektor multidimensi. Tujuan dari parameterisasi adalah untuk menemukan ciri (*feature*) yang mampu menaikkan probabilitas pengenalan sinyal tutur tersebut.

Sinyal tutur yang akan dikenali masih berupa sinyal analog. Sinyal analog ini dikenai operasi pencuplikan, sehingga menjadi sinyal digital yang selanjutnya diproses oleh prosesor digital yang ada pada PC (*Personal Computer*). Sinyal

digital hasil pencuplikan diproses oleh bagian pemroses sinyal yang kemudian dihasilkan himpunan ciri-ciri atas sinyal tutur tersebut.

Salah satu metode yang dapat digunakan untuk menghasilkan himpunan ciri-ciri ini adalah analisis spektral dengan penyandian prediksi *linear* (LPC). Karena pada tahap ini dihasilkan runtun ciri-ciri, pemrosesan ini juga biasa disebut sebagai ekstraksi ciri. Ekstraksi ciri berfungsi untuk mengambil fitur-fitur penting yang terdapat pada sinyal tutur. Hal ini penting dilakukan karena pada sinyal tutur banyak sekali informasi dan sangat kompleks, sehingga tidak mungkin untuk dilakukan pengenalan. Metode ini menganggap bahwa sinyal masukan adalah berupa sinyal *quasi-stationer*. Akan tetapi pada kenyataannya sinyal tutur adalah sinyal yang tak *stationer* terutama berkaitan dengan kata letup.

Sinyal tutur tidak dapat langsung dimasukan sebagai masukan sistem pengenal tutur karena sinyal tutur adalah sinyal yang sangat kompleks dan mempunyai dimensi yang sangat besar. Mereka tidak pernah muncul sama dua kali walaupun tutur yang diucapkan sama. Banyak sekali Variabel yang menentukan variasi suatu tutur yaitu :

1. Variabel yang bersumber dari pembicara itu sendiri, seperti kecepatan bicara, emosi, nada bicara.
2. Variabel yang bersumber dari orang yang berbeda seperti jenis kelamin umur.
3. Variabel yang berasal dari lingkungan seperti derau.

Akibat sifatnya yang begitu kompleks, maka pada sinyal tersebut perlu diambil fitur-fitur yang dapat mempresentasikan sinyal tersebut dalam bentuk

yang lebih sedehana. Dengan memakai metode seperti LPC maka 1 kerangka (*frame*) sinyal yang berisi ratusan data sampel data tercuplik akan direpresentasikan dalam satu vektor fitur yang terdiri dari puluhan data saja.

2.2.1 LPC (*Linear Predictive Coding*)

Beberapa keuntungan metode LPC (*Linear Predictive Coding*) untuk pengenalan suara manusia yaitu :

1. LPC menyediakan pemodelan yang baik untuk sinyal suara (*speech signal*), hal ini terutama untuk bagian *voiced* (suara tutur berbunyi) dan *unvoiced* (suara tutur tak berbunyi) meskipun pemodelan pada bagian *unvoiced* tidak seefektif pada bagian *voiced* tetapi masih dapat digunakan untuk pengenalan suara.
2. LPC dapat dengan mudah diterapkan baik secara peangkat lunak maupun perangkat keras, sebab perhitungan matematis yang dilibatkan *relative* lebih singkat.
3. Hasil pengenalan suara yang didapat dengan menerapkan LPC cukup baik dan bahkan lebih baik dibanding metode-metode yang dikenal sebelumnya seperti *filter bank*.

2.2.1.1 Pemodelan LPC

Suatu sampel sinyal tutur pada waktu ke-n dapat didekati dengan kombinasi *linear* dari p sampel sinyal tutur yang lalu. Prediksi nilai cuplikan ini yang selanjutnya disebut LPC (Rabiner dan Schafer, 1978)

$$s(n) \approx a_1 s(n-1) + a_2 s(n-2) + \ldots + a_k s(n-p) \quad (2.23)$$

Persamaan (2.23) dapat dikonversi ke dalam suatu persamaan dengan memasukkan suatu faktor eksitasi, $G\,u(n)$ yaitu :

$$s(n) = \sum_{k=1}^{p} a_k s(n-k) + Gu(n) \quad (2.24)$$

Keterangan : - u(n) = eksitasi ternormalisasi

- G = *gain* eksitasi,
- a_k = koefisien ,dianggap konstan pada kerangka sinyal tutur.

Transformasi-Z persamaan (2.24) menghasilkan :

$$S(z) = \sum_{k=1}^{p} a_k z^{\,k} S(z) + GU(z) \quad (2.25)$$

Dari persamaan (2.25) , maka fungsi alih model :

$$H(z) = \frac{S(z)}{GU(z)} = \frac{1}{1 - \sum_{k=1}^{p} a_k z^{-k}} = \frac{1}{A(z)} \quad (2.26)$$

Fungsi eksitasi untuk sinyal tutur merupakan deretan pulsa hampir periodis untuk suara tutur berbunyi (*voiced*), atau sumber derau acak untuk suara tutur tak berbunyi (*unvoiced*) (Rabiner dan Juang ,1993).

2.2.1.2 Analisis Persamaan LPC

Prediktor linear s(n) dengan koefisien-koefisien prediksi a_k mempunyai keluaran:

$$\tilde{s}(n) = \sum_{k=1}^{p} a_k s(n-k) \quad (2.27)$$

error prediksi e(n) adalah:

$$e(n) = s(n) - \tilde{s}(n) = s(n) - \sum_{k=1}^{p} a_k s(n-k) \tag{2.28}$$

dengan fungsi alih *error* adalah:

$$A(z) = \frac{E(z)}{S(z)} = 1 - \sum_{k=1}^{p} a_k z^{-k} \tag{2.29}$$

Jelas, jika s(n) secara aktual dibangkitkan seperti persamaan (2.24), maka *error* prediksi e(n) akan sama dengan eksitasi terskala Gu(n).

Permasalahannya adalah penentuan himpunan koefisien prediktor a_k secara langsung dari sinyal tutur dengan cara sedemikian sehingga diperoleh perkiraan yang baik terhadap sifat-sifat spektral sinyal tutur melalui penggunaan persamaan (2.26). Sinyal tutur bersifat berubah terhadap waktu karena itu koefisien prediktor harus diperkirakan dari segmen-segmen singkat sinyal tutur. Pendekatan dasarnya adalah mencari himpunan koefisien prediktor yang akan meminimalkan rerata *error* prediksi kuadrat pada bagian sinyal tutur yang singkat.

Didefinisikan bagian tutur singkat dan *error* singkat pada waktu n sebagai:

$$s_n(m) = s(n+m) \tag{2.30}$$

$$e_n(m) = e(n+m) \tag{2.31}$$

Rerata *error* kuadrat pada waktu n yang akan diminimalkan adalah:

$$E_n = \sum_m e_n^2(m) \tag{2.32}$$

$$E_n = \sum_m \left[s_n(m) - \sum_{k=1}^{p} a_k s_n(m-k) \right]^2 \tag{2.33}$$

koefisien-koefisien prediktor diperoleh dengan menurunkan E_n terhadap masing-masing a_k dan menyamakannya dengan nol,

$$\frac{\partial E_n}{\partial a_k} = 0, \qquad k = 1, 2, \ldots p \tag{2.34}$$

schingga diperoleh:

$$\sum_m s_n(m-i)s_n(m) = \sum_{k=1}^{p} \hat{a}_k \sum_m s_n(m-i)s_n(m-k) \tag{2.35}$$

Jika didefinisikan

$$\phi_n(i,k) = \sum_m s_n(m-i)s_n(m-k) \tag{3.36}$$

persamaan (2.35) dapat dinyatakan dalam bentuk notasi sebagai berikut:

$$\phi_n(i,0) = \sum_{k=1}^{p} \hat{a}_k \phi_n(i,k) \qquad i = 1, 2, \ldots p \tag{2.37}$$

yang menyatakan himpunan p persamaan. Rerata *error* kuadrat minimum dapat dinyatakan sebagai:

$$\hat{E}_n = \sum_m s_n^2(m) - \sum_{k=1}^{p} \hat{a}_k \sum_m s_n(m)s_n(m-k) \tag{2.38}$$

$$= \phi_n(0,0) - \sum_{k=1}^{p} \hat{a}_k \phi_n(0,k) \tag{2.39}$$

Pada persamaan (2.39) terlihat bahwa rerata *error* kuadrat minimum terdiri atas bagian yang tetap dan bagian yang tergantung pada koefisien prediktor.

Besaran $\phi_n(i,k)$ harus dihitung lebih dahulu, agar koefisien prediktor optimum diperoleh dengan menyelesaikan persamaan (2.37). Dalam praktek metode penyelesaian tersebut sangat tergantung pada rentang m yang digunakan

dalam mendefinisikan bagian-bagian tutur untuk analisis, dan daerah untuk penghitungan rerata *error* kuadrat.

2.2.1.3 Metode Autokorelasi

Salah satu metode penyelesaian yang paling sederhana, yaitu metode autokorelasi. Metode ini mengasumsikan bahwa bagian tutur $s_n(m)$ secara identik bernilai nol diluar interval $0 \le m \le N-1$. Hal ini diperoleh dengan melakukan penjendelaan terhadap sinyal tutur s(n+m):

$$s_n(m) = s(n+m)w(m) \tag{2.40}$$

w(n) merupakan jendela dengan panjang berhingga (misalnya jendela Hamming) yang bernilai nol diluar interval $0 \le m \le N-1$. Sehingga *error* prediksi yang bernilai tidak nol berada pada interval $0 \le m \le N-1+p$. Rerata *error* kuadrat menjadi:

$$E_n = \sum_{m=0}^{N-1+p} e_n^2(m) \tag{2.41}$$

dan besaran $\phi_n(i,k)$ dapat dinyatakan sebagai:

$$\phi_n(i,k) = \sum_{m=0}^{N-1+p} s_n(m-i)s_n(m-k), \qquad \begin{matrix} 1 \le i \le p \\ 0 \le k \le p \end{matrix} \tag{2.42}$$

atau

$$\phi_n(i,k) = \sum_{m=0}^{N-1+(i-k)} s_n(m)s_n(m+i-k), \qquad \begin{matrix} 1 \le i \le p \\ 0 \le k \le p \end{matrix}. \tag{2.43}$$

Karena persamaan (2.43) hanya suatu fungsi i – k (lebih daripada dua variabel bebas i dan k), maka besaran $\varnothing_n(i,k)$ berubah menjadi fungsi autokorelasi sederhana, yaitu:

$$\phi_n(i,k) = r_n(i-k) = \sum_{m=0}^{N-1+(i-k)} s_n(m)s_n(m+i-k) \tag{2.44}$$

Karena fungsi autokorelasi adalah fungsi simetris, yaitu $r_n(-k) = r_n(k)$, maka persamaan LPC dapat dinyatakan sebagai:

$$\sum_{k=1}^{p} r_n(|i-k|)\hat{a}_k = r_n(i), \qquad 1 \le i \le p \tag{2.45}$$

dan dapat dinyatakan dalam bentuk matriks sebagai:

$$\begin{bmatrix} r_n(0) & r_n(1) & r_n(2) & \dots & r_n(p-1) \\ r_n(1) & r_n(0) & r_n(1) & \dots & r_n(p-2) \\ r_n(2) & r_n(1) & r_n(0) & \dots & r_n(p-3) \\ \vdots & \vdots & \vdots & & \vdots \\ r_n(p-1) & r_n(p-2) & r_n(p-3) & \dots & r_n(0) \end{bmatrix} \begin{bmatrix} \hat{a}_1 \\ \hat{a}_2 \\ \hat{a}_3 \\ \vdots \\ \hat{a}_p \end{bmatrix} = \begin{bmatrix} r_n(1) \\ r_n(2) \\ r_n(3) \\ \vdots \\ r_n(p) \end{bmatrix} \tag{2.46}$$

Matriks p x p nilai-nilai autokorelasi adalah matriks Toeplitz, yaitu simetris dan seluruh elemen pada setiap arah diagonal bernilai sama. Dengan memanfaatkan sifat ini persamaan (2.46) dapat diselesaikan secara rekursif dengan menggunakan algoritma Durbin. Secara formal algoritma Durbin diberikan sebagai berikut (untuk menyederhanakan, notasi indeks yaitu k pada fungsi autokorelasi $r_k(m)$ dihilangkan):

$$E^{(0)} = r(0) \tag{2.47}$$

$$k_i = \left\{ r(i) - \sum_{j=1}^{i-1} \alpha_j^{(i-1)} r(|i-j|) \right\} \Big/ E^{(i-1)} \qquad 1 \le i \le p \tag{2.48}$$

$$\alpha_i^{(i)} = k_i \tag{2.49}$$

$$\alpha_j^{(i)} = \alpha_j^{(i-1)} - k_i \alpha_{i-j}^{(i-1)} \tag{2.50}$$

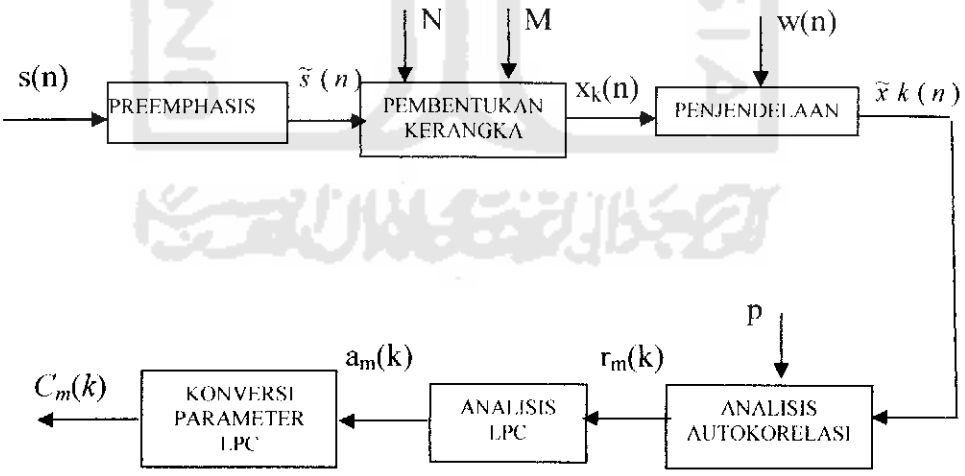
$$E^{(i)} = (1 - k_i^2) E^{(i-1)} \tag{2.51}$$

dengan menghilangkan penjumlahan dalam persamaan (2.48) untuk i = 1. Himpunan persamaan (2.47 – 2.51) diselesaikan secara rekursif untuk i = 1, 2, ., p, dan penyelesaian akhirnya diberikan sebagai:

$$a_m = koefisienLPC = \alpha_m^{(p)}, \qquad 1 \le m \le p. \tag{2.52}$$

2.2.1.4 Langkah-langkah Pemrosesan Sinyal Tutur Dengan LPC

Gambar di bawah merupakan urutan pemrosesan sinyal tutur berdasarkan analisis LPC :



Gambar II.3 Blok Diagram LPC Untuk Pengenalan Suara

Langkah-langkah pemrosesan sinyal tutur dengan LPC adalah sebagai berikut :

1. ***Preemphasis*** : sinyal tutur dilewatkan pada tapis digital *FIR* orde satu. *Preemphasis* digunakan untuk meratakan sinyal. Fungsi tapisnya adalah :

   $$H(z) = 1 - \tilde{a}z^{-1} \qquad 0{,}9 \le \tilde{a} \le 1{,}0. \qquad (2.53)$$

   Keluaran dari *preemphasis* adalah $\tilde{s}(n)$, yang dimodelkan dengan persamaan berikut:

   $$\tilde{s}(n) = s(n) - \tilde{a}s(n-1) \qquad (2.54)$$

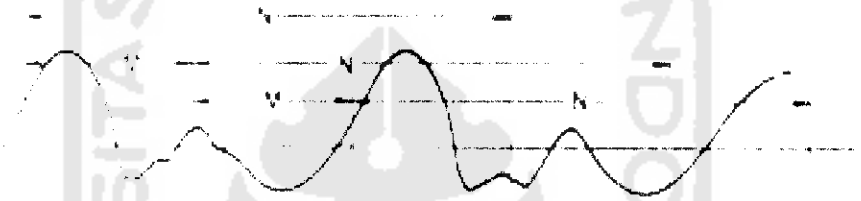
   Nilai $\tilde{a}$ yang banyak digunakan berkisar 0,95, untuk nilai tepatnya yang biasa digunakan $\tilde{a}$ = 15/16=0.9375

2. **Pembentukan kerangka (*frame blocking*) :** Keluaran dari *preemphasis*, $\tilde{s}(n)$ dibagi menjadi kerangka-kerangka analisis N sampel. Jarak antara kerangka yang berurutan dipisahkan oleh M sampel. Jadi kerangka ke-2 dimulai setelah M sampel dari awal kerangka pertama, kerangka ke-3 dimulai setelah 2M sampel setelah awal kerangka pertama, dan seterusnya. Tiap kerangka terjadi overlap sebanyak N-M sampel. Jika kerangka tutur ke-k dinyatakan sebagai $x_k(n)$ dan ada K kerangka dalam seluruh sinyal tutur maka:

   $$x_k(n) = \tilde{s}(Mk + n) \quad ,n=0,1,\ldots, N-1, k = 0,1,\ldots, K-1. \qquad (2.55)$$

   Kerangka pertama $x_0(n)$ meliputi sampel tutur $\tilde{s}(0), \tilde{s}(1), \ldots, \tilde{s}(N-1)$. Kerangka kedua $x_1(n)$ meliputi sampel tutur ke

$\tilde{s}(M), \tilde{s}(M+1),..., \tilde{s}(M+N-1)$. Dan kerangka ke L $x_{L-1}(n)$ meliputi sampel tutur $\tilde{s}(M(L-1)), \tilde{s}(M(L-1)+1),..., \tilde{s}(M(L-1)+N-1)$. Biasanya nilai N=300 dan M=100, umumnya digunakan ketika frekuensi cuplik sinyal tutur 6.67 kHz. Proses pembentukan kerangka analisis adalah ditunjukkan gambar II.4 :



Gambar II.4 Pembentukan Kerangka Pada LPC

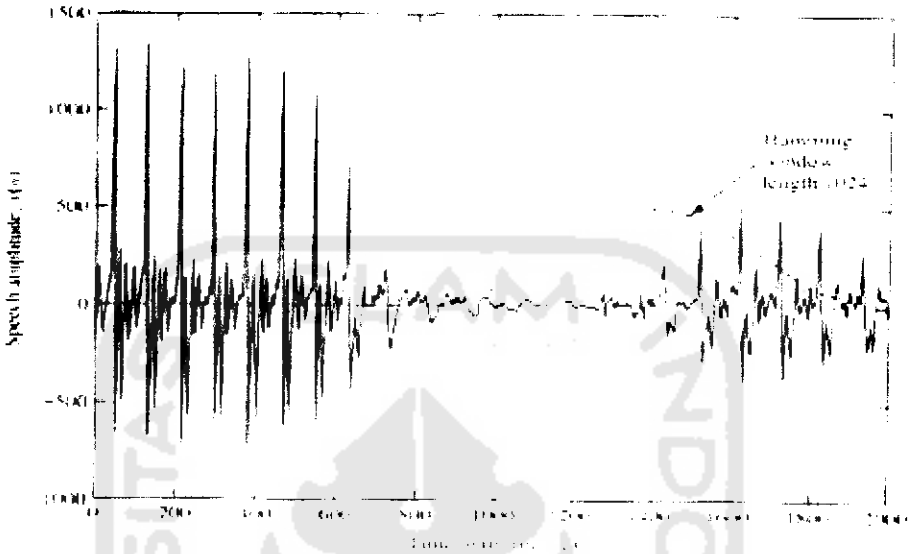
3. **Penjendelaan (*windowing*)** : Masing-masing kerangka tutur dilakukan penjendelaan, yaitu dikalikan dengan suatu model jendela tertentu (dalam hal ini jendela Hamming). Hal ini untuk meminimalkan ketakkontinyuan di awal dan akhir masing-masing kerangka. Jika didefinisikan jendela sebagai w(n), $0 \le n \le N-1$, maka hasil penjendelaan adalah:

$$\tilde{x}_k(n) = x_k(n)w(n), \qquad 0 \le n \le N-1. \qquad (2.56)$$

Jendela yang sering digunakan untuk analisis LPC metode autokorelasi adalah jendela Hamming yang berbentuk:

$$w(n) = 0{,}54 - 0{,}46\cos\left(\frac{2\pi n}{N-1}\right), \qquad 0 \le n \le N-1 \qquad (2.57)$$

Proses penjendelaan ditunjukkan oleh gambar II.5 :



Gambar II.5 Penjendelaan Sinyal Tutur Dengan Jendela Hamming

4. **Analisis autokorelasi** : Masing-masing kerangka tutur yang telah dijendela diautokorelasikan untuk memperoleh:

$$r_k(m) = \sum_{n=0}^{N-1-m} \tilde{x}_k(n)\tilde{x}_k(n+m), \qquad \text{m} = 0, 1, \ldots, \text{p}, \qquad (2.58)$$

dengan p adalah orde analisis LPC.

5. **Analisis LPC (LPC *analysis)*** : Langkah pemrosesan selanjutnya adalah analisis LPC yang akan mengkonversi p+1 autokorelasi masing-masing kerangka, menjadi vektor koefisien LPC. Salah satu metode yang digunakan adalah metode Durbin yang dinyatakan dalam algoritma berikut (untuk memudahkan penulisan kita dapat menghilangkan k pada $r_k(\text{m})$) :

$$E^{(0)} = r(0) \tag{2.59}$$

$$k_i = \left\{ r(i) - \sum_{j=1}^{i-1} \alpha_j^{(i-1)} r(|i-j|) \right\} \Big/ E^{(i-1)} \qquad 1 \le i \le p \tag{2.60}$$

$$\alpha_i^{(i)} = k_i \tag{2.61}$$

$$\alpha_j^{(i)} = \alpha_j^{(i-1)} - k_i \alpha_{i-j}^{(i-1)} \tag{2.62}$$

$$E^{(i)} = (1 - k_i^2) E^{(i-1)} \tag{2.63}$$

dengan menghilangkan penjumlahan dalam persamaan (2.60) untuk i = 1. Himpunan persamaan (2.59 – 2.60) diselesaikan secara rekursif untuk i = 1, 2, ., p, dan penyelesaian akhirnya diberikan sebagai:

$$a_m = koefisienLPC = \alpha_m^{(p)}, \qquad 1 \le m \le p. \tag{2.64}$$

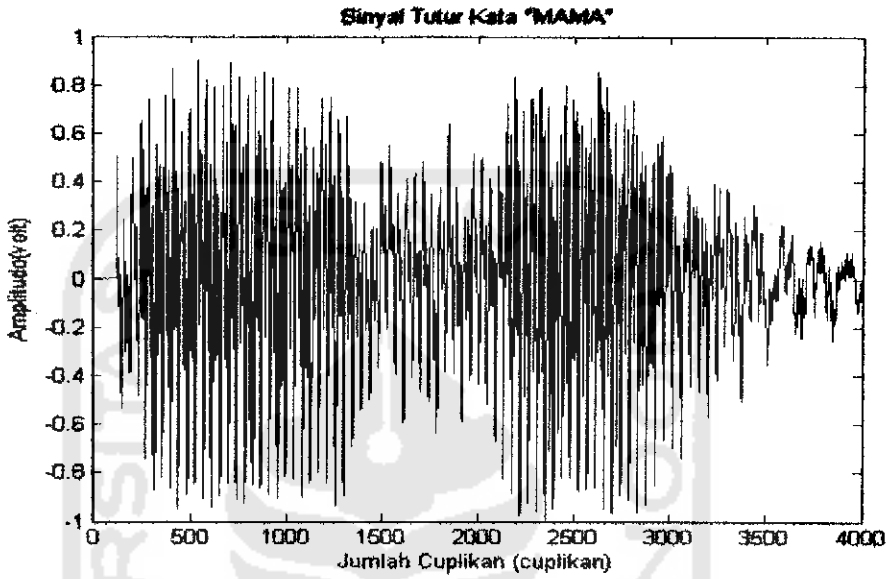
6. **Konversi parameter LPC menjadi koefisien *cepstral*.** Himpunan parameter LPC yang sangat penting yang dapat diturunkan secara langsung dari himpunan koefisien LPC adalah koefisien *cepstral* LPC, c(m). Koefisien *cepstral* LPC ditentukan secara rekursif sebagai berikut:

$$c_m = a_m + \sum_{k=1}^{m-1} \left(\frac{k}{m} \right) c_k a_{m-k}, \qquad 1 \le m \le p \tag{2.65}$$

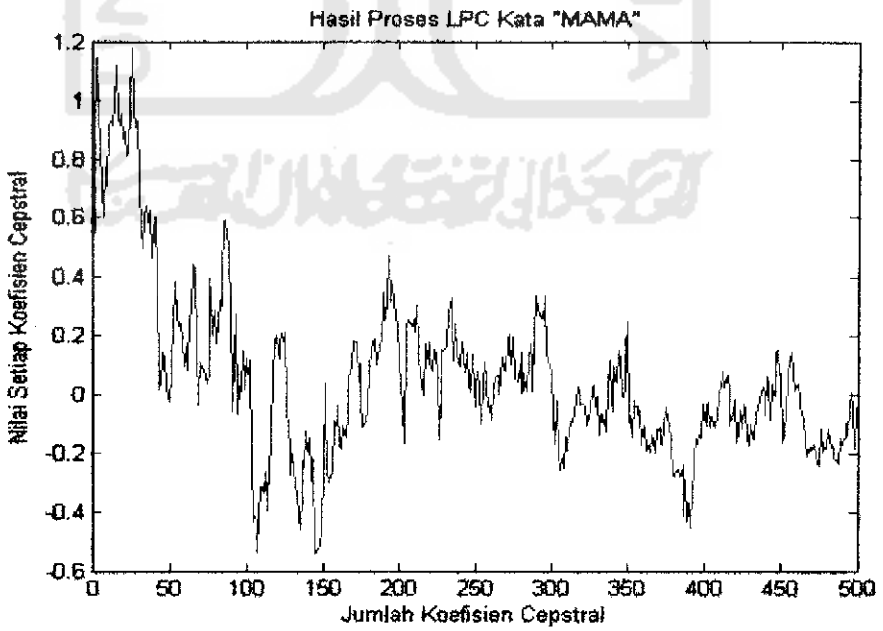
$$c_m = \sum_{k=1}^{m-1} \left(\frac{k}{m} \right) c_k a_{m-k}, \qquad m > p, \tag{2.66}$$

Koefisien *cepstral* ini merupakan koefisien transformasi Fourier yang lebih tahan terhadap *noise* jika digunakan pada pengenalan suara daripada koefisien LPC (Rabiner dan Juang, 1993). Gambar II.6 menunjukan sinyal

tutur dari penutur sedangkan gambar II.7 menunjukan sinyal tutur yang telah dikenai proses LPC.



Gambar II.6 Sinyal Tutur Pembicara 1 Pada File Pertama



Gambar II.7 Hasil Proses LPC Untuk Pembicara 1 Pada *File* Pertama

Bentuk nilai pada parameter analisa LPC untuk sistem pengenalan suara ditunjukan pada tabel II.1 yaitu :

Tabel II.1 Nilai Parameter Analisis LPC

Parameter	Fs = 6,67 kHz	Fs = 8 kHz	Fs = 10 kHz
N	300 (45 ms)	240 (30 ms)	300 (30 ms)
M	100 (15 ms)	80 (10 ms)	100 (10 ms)
p	8	10	10

2.3 FFT (*Fast Fourier Transform*)

Fast Fourier Transform merupakan penyederhanaan dari *Discrete Fourier Transform* (DFT). Untuk sinyal waktu diskrit *x(n)*, maka DFT dari sinyal diberikan oleh :

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-j2\pi nk/N}, k = 0,1,...,N-1 \qquad (2.67)$$

Faktor eksponensial dalam persamaan tersebut dinamakan *twiddle faktor* yang bersifat periodik dengan periode N dan dilambangkan dengan W_N^{nk}, sehingga DFT dari sinyal waktu diskrit *x(n)* dapat dituliskan sebagai :

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{nk}, k = 0,1,..., N-1 \qquad (2.68)$$

Dengan *Fast Fourier Transform*, maka jumlah titik data *N* merupakan bilangan yang dapat difaktorkan sehingga seluruh jumlah titik DFT dapat dipecah ke dalam kelompok-kelompok yang makin lama makin kecil. Pada algoritma FFT

radix 2 *Decimation In Frequency* (DIF), data sebanyak *N* dibagi menjadi dua bagian, yaitu data dengan indeks *0* sampai dengan *N/2-1* dan data dengan indeks *N/2* sampai dengan *N-1*. Dengan memanfaatkan sifat FFT yang simetri maka data dengan indeks *0* sampai *N-1* dapat dibagi dua yaitu hanya dengan menggunakan data *0* sampai *N/2-1* atau *N/2* sampai *N-1*. Dengan demikian jumlah perhitungan dapat dikurangi. Sifat simetri tersebut adalah :

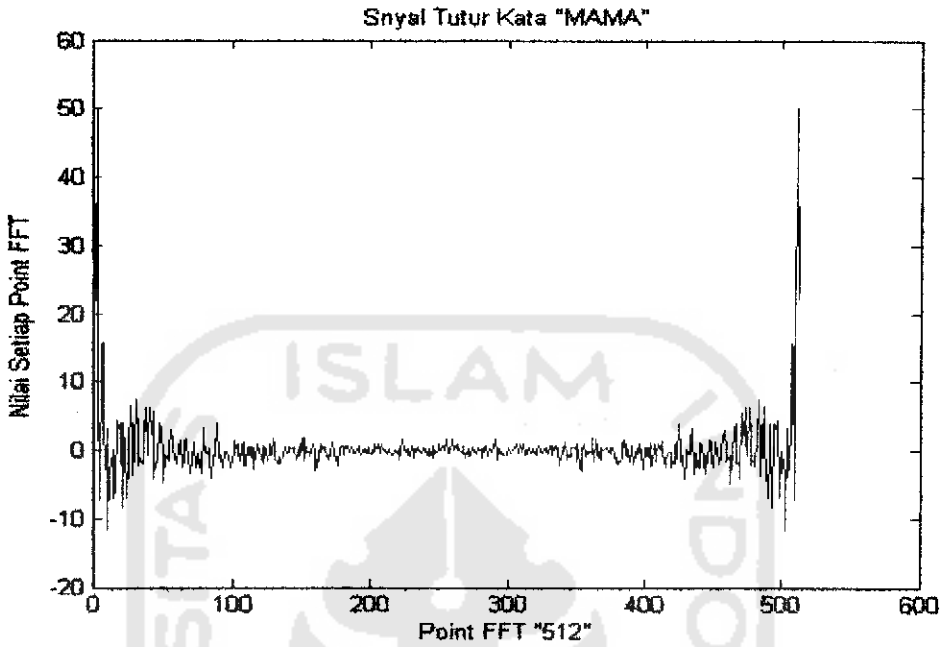
$$W_N^{-nk} = - W_N^{k} \tag{2.69}$$

dimana,

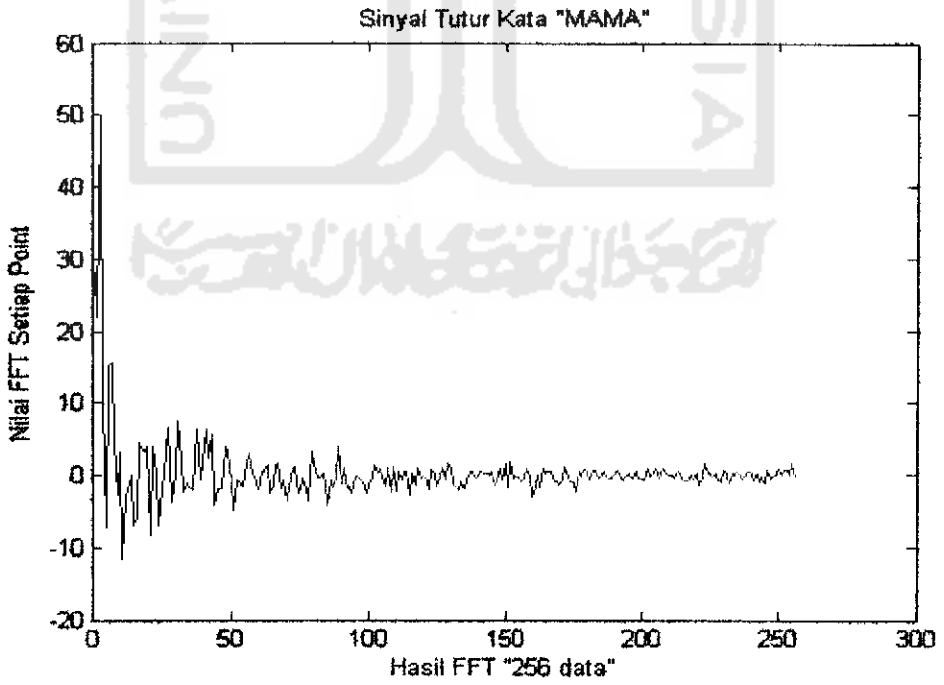
$$W_N = e^{-j2\pi/N} \tag{2.70}$$

Berikut adalah bentuk sinyal hasil proses LPC yang akan dijadikan contoh untuk pemrosesan sinyal dengan FFT. Sinyal ini merupakan sinyal tutur Pembicara 1 untuk *file* pertama dan kata " MAMA". Pemrosesan FFT dapat dilakukan dengan mengeksekusi perintah *fft(X,N)* dengan X adalah data yang akan diproses dan N adalah *point* FFT yang akan digunakan. Jika banyaknya X kurang dari N maka pada X ditambahkan nol sehingga banyaknya X = N sedangkan jika banyaknya X melebihi N, maka banyaknya X dikurangi sehingga banyaknya X = N.

Hasil proses LPC pada gambar II.7 akan diproses dengan FFT 512 point, sehingga diperoleh hasil proses FFT seperti pada gambar II.8. Pada Gambar II.8 terlihat simetris yaitu kedua sisi kiri dan kanan gambar terlihat sama. Hal ini menunjukan sifat FFT yang simetris. Sehingga dengan jumlah data sebanyak 512 hanya diambil 256 dengan demikian banyaknya perhitungan dapat dikurangi. Hasil proses FFT dengan 256 data ditunjukan pada gambar II.9.



Gambar II.8 Hasil Proses FFT Untuk Pembicara 1 Pada *File* Pertama



Gambar II.9 Hasil Proses FFT 256 Data Untuk Pembicara 1 Pada *File* Pertama

Proses FFT juga dapat dilakukan dengan menggunakan persamaan 2.67 atau 2.68. Tetapi pemrosesan sinyal tutur jumlah datanya sangat banyak maka untuk memudahkan perhitungan digunakan perintah *fft(X,N)*. Contoh pemrosesan data dengan menggunakan persamaan 2.66 terhadap sebuah vektor yaitu :

Diketahui : X(n) = [0.1 0.01 0.02 0.03]

N = 8

Phi = 3,14

k = 0,1,...7

n = 0,1,...7

Ditanyakan : X(k) = ...?

Penyelesaian :

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-j2\pi nk/N}, k = 0,1,...,N-1$$

k=0, X(0)= x(0).e^(-j.2.phi.0.0)/8+x(1).e^(-j.2.phi.1.0)/8+ x(2).e^(-j.2.phi.2.0)/8
+x(3).e^(-j.2.phi.3.0)/8+ x(4).e^(-j.2.phi.4.0)/8+ x(5).e^(-j.2.phi.5.0)/8
+ x(6).e^(-j.2.phi.6.0)/8+ x(7).e^(-j.2.phi.7.0)/8

= 0,1 + 0,01 + 0,02 + 0,03 + 0 + 0 + 0 + 0

X(0) = 0,16

k=1, X(1)= x(0).e^(-j.2.phi.0.1)/8+x(1).e^(-j.2.phi.1.1)/8+ x(2).e^(-j.2.phi.2.1)/8
+x(3).e^(-j.2.phi.3.1)/8+ x(4).e^(-j.2.phi.4.1)/8+ x(5).e^(-j.2.phi.5.1)/8
+ x(6).e^(-j.2.phi.6.1)/8+ x(7).e^(-j.2.phi.7.1)/8

=0,1 + 0,01.e^(-j.phi/4) + 0,02.e^(-j.phi/2) +0,03.e^(-j.phi.3/4)+0+0+0+0

X(1) = 0,0859 – 0,0483i

k=2, X(2)= x(0).e^(-j.2.phi.0.2)/8+x(1).e^(-j.2.phi.1.2)/8+ x(2).e^(-j.2.phi.2.2)/8
+x(3).e^(-j.2.phi.3.2)/8+ x(4).e^(-j.2.phi.4.2)/8+ x(5).e^(-j.2.phi.5.2)/8
+ x(6).e^(-j.2.phi.6.2)/8+ x(7).e^(-j.2.phi.7.2)/8

=0,1 + 0,01.e^(-j.phi/2) + 0,02.e^(-j.phi) +0,03.e^(-j.phi.3/2)+0+0+0+0

X(2) = 0,0800 + 0,0200i

k=3, X(3)= x(0).e^(-j.2.phi.0.3)/8+x(1).e^(-j.2.phi.1.3)/8+ x(2).e^(-j.2.phi.2.3)/8
+x(3).e^(-j.2.phi.3.3)/8+ x(4).e^(-j.2.phi.4.3)/8+ x(5).e^(-j.2.phi.5.3)/8
+ x(6).e^(-j.2.phi.6.3)/8+ x(7).e^(-j.2.phi.7.3)/8

=0,1 + 0,01.e^(-j.3.phi/4) + 0,02.e^(-j.3.phi/2)
+0,03.e^(-j.phi.9/4)+0+0+0+0

X(3) = 0,1141 – 0,0083i

k=4, X(4)= x(0).e^(-j.2.phi.0.4)/8+x(1).e^(-j.2.phi.1.4)/8+ x(2).e^(-j.2.phi.2.4)/8
+x(3).e^(-j.2.phi.3.4)/8+ x(4).e^(-j.2.phi.4.4)/8+ x(5).e^(-j.2.phi.5.4)/8
+ x(6).e^(-j.2.phi.6.4)/8+ x(7).e^(-j.2.phi.7.4)/8

=0,1 + 0,01.e^(-j.phi) + 0,02.e^(-j.2.phi)
+0,03.e^(-j.3.phi)+0+0+0+0

X(4) = 0,0800 – 0,0000i

k=5, X(5)= x(0).e^(-j.2.phi.0.5)/8+x(1).e^(-j.2.phi.1.5)/8+ x(2).e^(-j.2.phi.2.5)/8
+x(3).e^(-j.2.phi.3.5)/8+ x(4).e^(-j.2.phi.4.5)/8+ x(5).e^(-j.2.phi.5.5)/8
+ x(6).e^(-j.2.phi.6.5)/8+ x(7).e^(-j.2.phi.7.5)/8

=0,1 + 0,01.e^(-j.5.phi/4) + 0,02.e^(-j.5.phi/2)
+0,03.e^(-j.15.phi/4)+0+0+0+0

X(5) = 0,1141 + 0,0083i

k=6, X(6)= x(0).e^(-j.2.phi.0.6)/8+x(1).e^(-j.2.phi.1.6)/8+ x(2).e^(-j.2.phi.2.6)/8

+x(3).e^(-j.2.phi.3.6)/8+ x(4).e^(-j.2.phi.4.6)/8+ x(5).e^(-j.2.phi.5.6)/8

+ x(6).e^(-j.2.phi.6.6)/8+ x(7).e^(-j.2.phi.7.6)/8

=0,1 + 0,01.e^(-j.12.phi/8) + 0,02.e^(-j.24.phi/8)

+0,03.e^(-j.36.phi/8)+0+0+0+0

X(6) = 0,0800 – 0,0200i

k=7, X(7)= x(0).e^(-j.2.phi.0.7)/8+x(1).e^(-j.2.phi.1.7)/8+ x(2).e^(-j.2.phi.2.7)/8

+x(3).e^(-j.2.phi.3.7)/8+ x(4).e^(-j.2.phi.4.7)/8+ x(5).e^(-j.2.phi.5.7)/8

+ x(6).e^(-j.2.phi.6.7)/8+ x(7).e^(-j.2.phi.7.7)/8

=0,1 + 0,01.e^(-j.14.phi/8) + 0,02.e^(-j.28.phi/8)

+0,03.e^(-j.42.phi/8)+0+0+0+0

X(7) = 0,0859 + 0,0483i

Dari hasil perhitungan akan diambil nilai *real*nya, sehingga hasil proses FFT terhadap vektor X adalah :

X(k) = [X(0) X(1) X(2) X(3) X(4) X(5) X(6) X(7)]

X(k) = [0,16 0,0859 0,0800 0,1141 0,0800 0,1141 0,0800 0,0859]

Berdasarkan hasil perhitungan yang diperoleh, data N/2 sampai N terdapat juga pada data 0 sampai N/2-1. Hal ini menunjukan adanya sifat simetri pada FFT. Dengan demikian untuk pemrosesan data selanjutnya tidak perlu semua data digunakan karena adanya sifat simetri tersebut.