MODUL 6
TOPIK LANJUTAN PENGOLAH WICARA

I. TUJUAN

- Mahasiswa mampu memanfaatkan contoh-contoh program perangkat lunak untuk menyusun sebuah modul praktikum lanjutan secara bebas.

II. DASAR TEORI

Berbeda dengan modul sebelumnya yang selalu menampilkan teori tentang sinyal wicara secara terstruktur, pada modul ini akan diberikan gambaran sepintas tentang fungsi-fungsi dan contoh program dalam perangkat lunak (dalam hal ini Matlab) yang berkaitan dengan topik pilihan yang harus disusun oleh mahasiswa yang melakukan praktikum.

2.1. *Linear Predictive Coding*

Linear predictive analysis pada sinyal wicara akan diberikan contohnya disini. Metode yang digunakan adalah metode auto korelasi. Metode autokorelasi mengasumsikan bahwa sinyal bahwa sinyal memiliki nilai sama dengan nol untuk interval di luar daerah yang dianalisa (0<=m<=N-1).

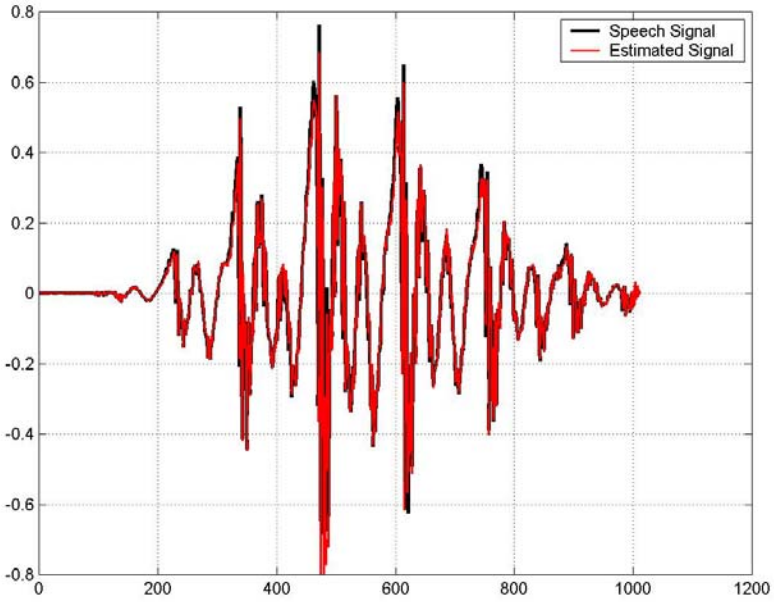
Selanjutnya disini akan dicoba untuk meminimisasikan *prediction error* ketika posisinya *nonzero*, yaitu sinyal yang berada di dalam interval 0<=m<=N-1+p. Dalam hal ini *p* adalah order pada model yang digunakan. Nilai *error* menunjukkan angka besar pada saat awal dan pada bagian akhir interval. Ini memberikan alasan kenapa segmen sinyal wicara yang dianalisa di-*tap* dengan menggunakan sebuah *window*, misalnya *Hamming*. Untuk pemilihan panjang window untuk sementara sebaiknya disamakan dengan panjang sinyal wicara yang akan dianalisa . Satu keuntungan metode ini adalah hasil yang diperoleh menunjukkan kestabilan nilai. Nilai *error autocorrelation* dan *spectrum* dihitung sebagai ukuran kepresisian dalam prediksi.

Contoh Program

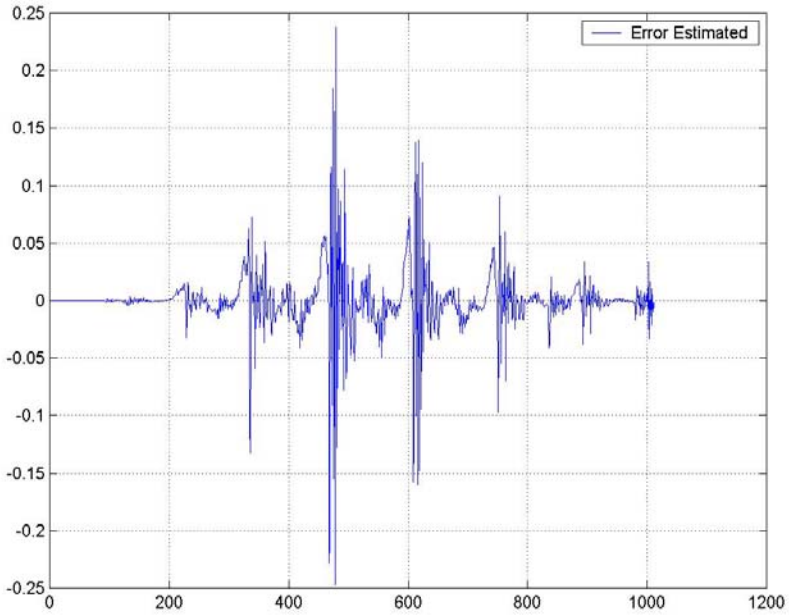
Anda coba pelajari program berikut ini.

```
[x,fs] = wavread('File_e.wav');
wavplay(x,fs)
len_x = length(x);

% The signal is windowed
w = hamming(len_x);
wx = w.*x;

% LPC autocorrelation method
order = 12;
% LPC function of MATLAB is used
[lpcoefs, errorPow] = lpc(x, order);

% The estimated signal is calculated as the output of linearly filtering
% the speech signal with the coefficients estimated above
estx = filter([0 -lpcoefs(2:end)], 1, [wx; zeros(order,1)]);

% Display results
figure(1)
plot([wx; zeros(order,1)],'g');
title('Linear Predictive Analysis, Autocorrelation Method');
hold on;
plot(estx,'black');
hold off;
%xlim([0 length(er)])
legend('Speech Signal','Estimated Signal');

% The prediction error is estimated in the interval 0<=m<=N-1+p
er = [wx; zeros(order,1)] - estx;

figure(2)
plot(abs(er));
legend('Error Signal');

%Prediction error energy in the same interva
erEn = sum(er.^2);
```

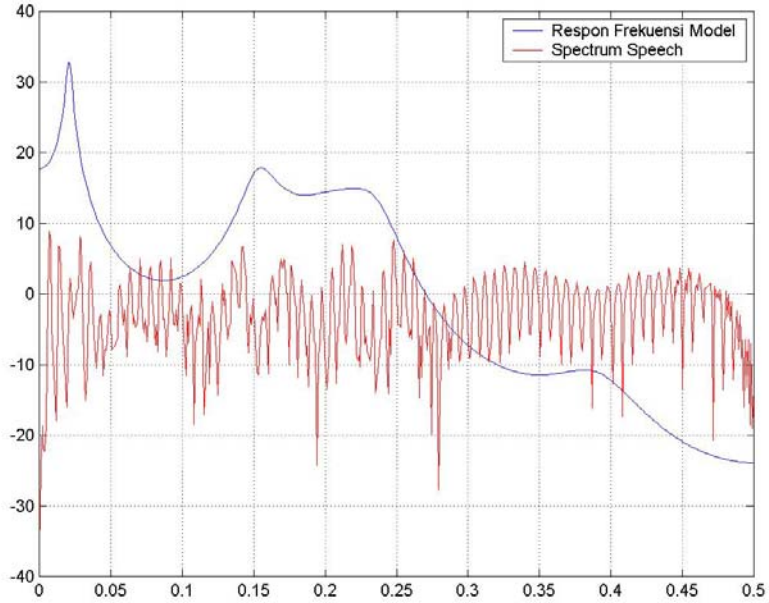
```
% Calculate the frequency response of the linear prediction model
[H, W] = freqz(sqrt(erEn), lpcoefs(1:end), 513);

% Calculate the spectrum of the windowed signal
S = abs(fft(wx,1024));

figure(3)
plot(linspace(0,0.5,513), 20*log10(abs(H)),'black');
hold on;
plot(linspace(0,0.5,513), 20*log10(S(1:513)));
legend('Model Frequency Response','Speech Spectrum')
hold off;

% Autocorrelation of the prediction error
[acs,lags] = xcorr(er);

figure(4)
plot(lags, acs);
legend('Prediction Error Autocorrelation')
% Calculate the spectrum of the error signal
eS = abs(fft(er,1024));

figure(5)
plot(linspace(0,0.5,513), 20*log10(eS(1:513)));
legend('Prediction Error Spectrum')
```



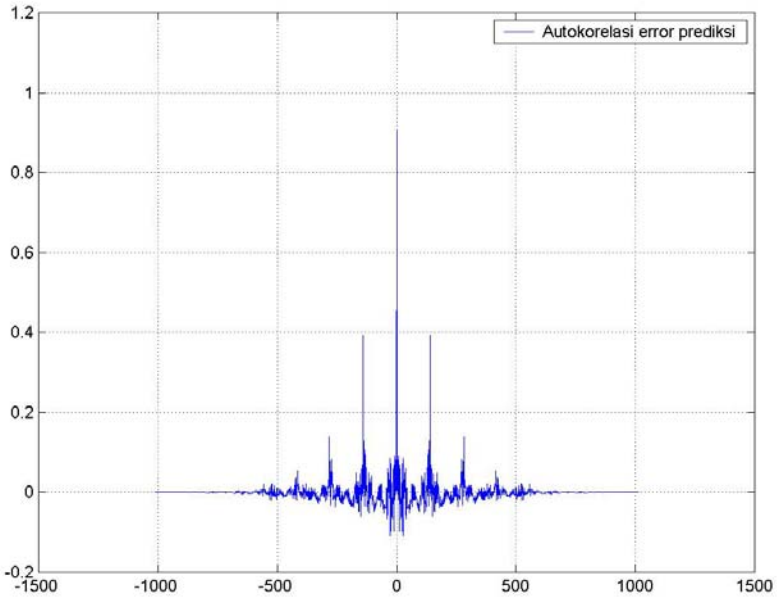
Gambar 1. Sinyal wicara terwindow



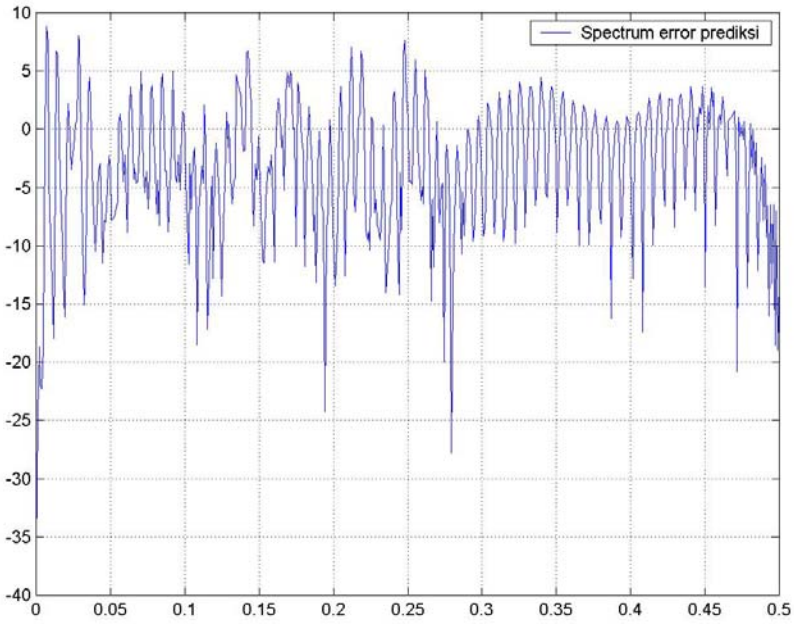
Gambar 2. Error estimasi



Gambar 3. Spectrum



Gambar 4. Auto korelasi



Gambar 5. Error Spectral

2.2.Pengaruh Pemilihan Niai Orde pada LPC

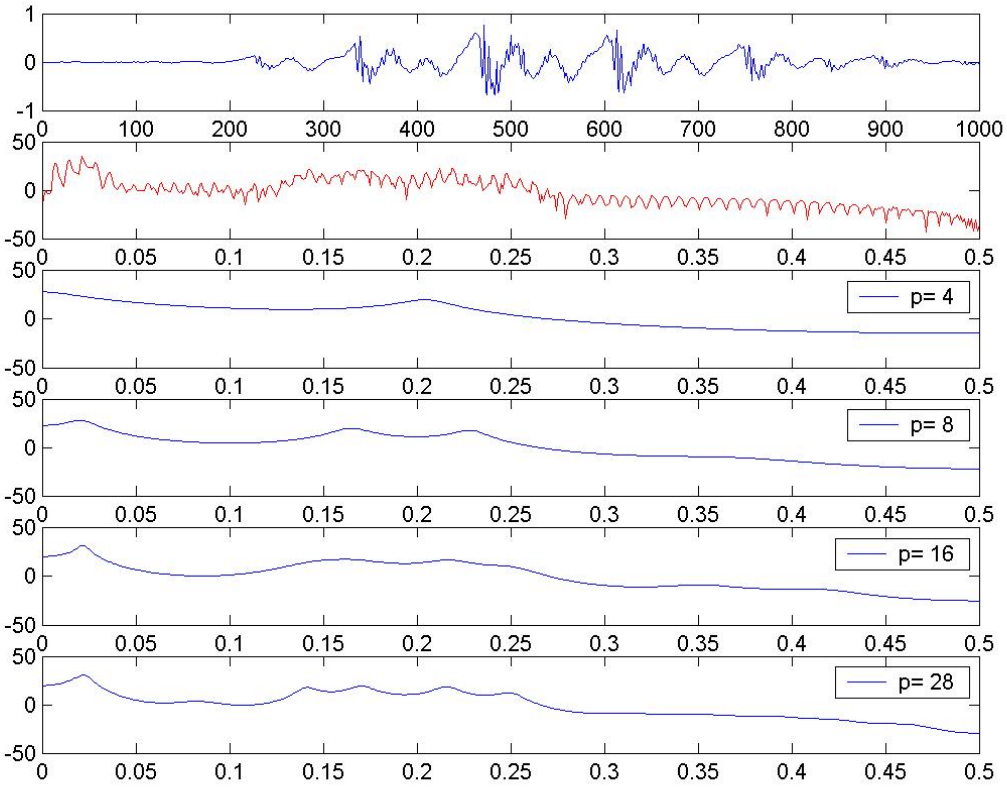
Disini akan dicoba menggunakan berbagai nilai order LPC yang berbeda. Dengan cara ini diharapkan akan dapat diketahui respon frekuensi yang akan dihasilkan secara lebih mendetail jika nilai orde LPC kita rubah-rubah. Nilai *prediction error* seharusnya mengalami penurunan nilai jika ordernya kita naikkan. Pemilihan orde LPC pada umumnya tidak tergantung pada metode yang digunakan, tetapi lebih berhubungan dengan frekuensi *sampling* yang digunakan. Umumnya kita mamilih model yang menghasilkan satu pole pada setiap 1 kHz *spectrum* frekuensi, sehingga dengan suara manusia kita akan mendapatkan 3 sampai 4 pole yang merepresentasikan *source excitation spectrum* dan radiasinya. Untuk model *speech* yang dirancang denagn frekuensi sampling 16 kHz secara umum pemilihan orde senilai 20 sudah sangat mencukupi.

Contoh Program

Anda coba pelajari program berikut ini.

```
%File Name: LPC_02.m
clear all;
fs=8000;
%membaca file
[phons,fs]=wavread('File_e');
x=phons(1:1000); wavplay(x)
x_len=length(x);

%Proses windowing
w=hamming(x_len);
wx=w.*x;
%menentukan orde LPC
orders=[4, 8, 16, 28];
l_ord=length(orders);

%Hitung spectrum windowed signal
S=abs(fft(wx,1024));
%title('Analisis dengan beragam nilai LPC')
subplot(l_ord + 2,1,1)
plot(wx);
subplot(l_ord + 2,1,2)
plot(linspace(0,0.5,513),20*log10(S(1:513)),'r');
```

```
for o=orders
   [lpcoefs,e]=lpc(wx,o);
   %Estimasi Sinyal
   estx=filter([0 -lpcoefs(2:end)], 1, [wx;zeros(o,1)]);

   %Error prediksi
   er=[wx;zeros(o,1)] - estx;
   erEn=sum(er.^2);

   %Respon Frekuensi
   [H,W]=freqz(sqrt(erEn),lpcoefs(1:end), 513);

   subplot(l_ord + 2,1,find(orders==o) + 2);
   plot(linspace(0,0.5,513), 20*log10(abs(H)));
   legend(['p= ',int2str(o)]);
end
```



Gambar 6. Pengaruh pemilihan nilai order pada LPC

2.3. *Spectral Warping* pada LPC

Standard LPC analysis memisahkan sebuah sinyal menjadi suatu representasi *spectral* yang lebih halus resonansi pada suatu *time-varying all-pole filter* dan suatu pendekatan *white excitation* yang mana ketika dilewatkan melalui sebuah *time-varying filter*, akan menghasilkan sinyal seperti aslinya. Dekomposisi ini merupakan dasar bagi semua jenis kompresi dan teknik modifikasi pada sinyal wicara.

Satu hal yang dapat kita lakukan adalah menggeser secara sistematis frekuensi resonansi pada model LPC dengan mengggunakan sebuah *warping transformation* menjadi IIR filter (secara esensial, sustitusi pada suatu *all-pole system* untuk semua elemen *delay*). Efek perceptual pada kasus ini terhadap suara manusia adalah terjadinya perubahan warna suara, yang dalam gambaran *spectrogram*-nya. *Pitch* yang direpresentasikan dengan sebuah excitasi tidak mengalami perubahan.

Kode Program Matlab

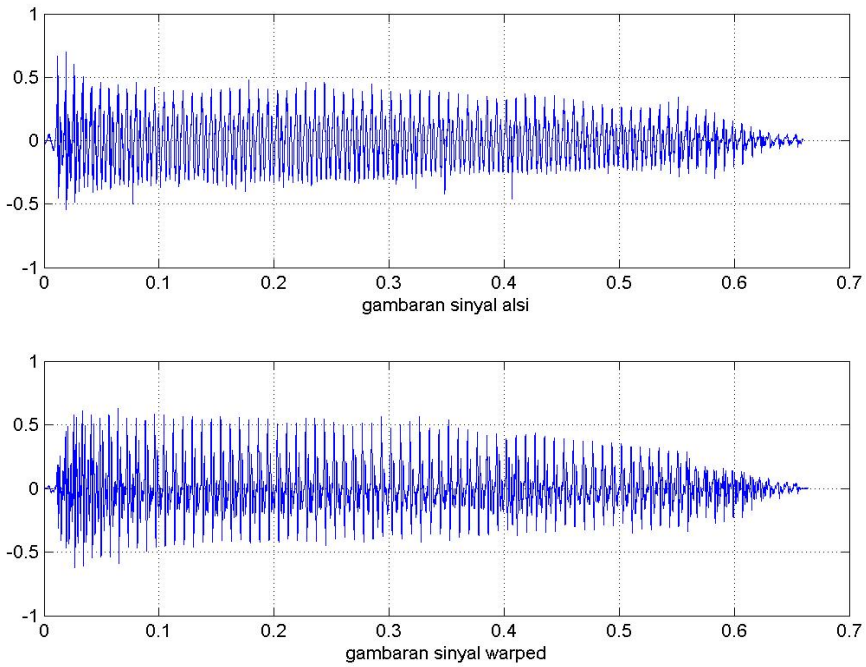
Untuk penjalankan program ini anda memerlukan beberapa fungsi berikut:

- **[B,A] = warppoles(a, alpha)** *warps all-pole filter* didefinisikan dengan nilai koefisien-koefisien numerator (pembilang) menggunakan sebuah *first-order allpass substitution* dengan parameter *alpha* untuk membangkitkan sebuah filter baru (dengan *pole* dan *zero*) yang didefinisikan oleh *polynomial* B dan A. Alpha dihasilkan dalam pergeseran pole-pole yang menaikkan nilai frekuensi .
- **[A,G,E] = lpcfit(D,P,H,W,O)** menetapkan *P-th order LPC* (*all-pole, autoregressive*) model untuk suara gelombang sinyal D, menggunakan *W-point windows* dikembangkan dengan H sampel. Baris pada A tersusun dari koefisien *all-pole filter* [*1 a1 a2 .. aP*], yang berkaitan dengan element-elemen pada G dan memberikan penguatan frame (*residual RMS*). E merupakan *excitation residual actual*. Spesifikasi O sebagai zero mencegah terjadinya *overlap-add* pada *residual*, untuk penyempurnaan reconstruksi tetapi kurang berguna pada E.
- **D = lpcsynth(A,G,E,H,OV)** digunakan untuk *resynthesize* dari perameter LPC kembali ke **lpcfit**, atau bisa juga menggunakan *noise excitation* jika E dihilangkan.

Contoh Program

Anda coba pelajari program berikut ini.

```
%File Name: freq_warping_LPC_01.m
clear all;

% Load sebuah file wicara
fs=8000;
[d,fs] = wavread('FILE_I.wav');

% Tetapkan  model LPC original (high-order)
[a,g,e] = lpcfit(d,20);

% Warp pole-pole
% (warppoles memodifikasi setiap frame - baris pada a - pada waktu sama)
alpha = -0.2;
[bhat, ahat]  = warppoles(a, alpha);

%  Resynthesize denagn LPC yang baru
% (untungnya, bhat bernilai sama untuk semua frame)
dw = filter(bhat(1,:), 1, lpcsynth(ahat, g, e));

% Plot dan dengarkan suara asli original...
tt=length(d);
t=1/fs:1/fs:tt/fs;
subplot(2,1,1)
plot(t,d)
xlabel('gambaran sinyal alsi')
wavplay(d,fs);
grid

% Plot dan dengarkan warped version
tt=length(dw);
t=1/fs:1/fs:tt/fs;
subplot(2,1,2)
plot(t,dw)
xlabel('gambaran sinyal warped')
wavplay(dw,fs);
grid
```



Gambar 7. Gambaran sinyal warping dengan LPC

2.4. *Dynamic Time Warp* (DTW)

Satu masalah yang cukup rumit dalam *speech recognition* (pengenalan wicara) adalah proses perekaman yang terjadi seringkali berbeda durasinya, biarpun kata atau kalimat yang diucapkan sama. Bahkan untuk satu suku kata yang sama atau vocal yang sama seringkali proses perekaman terjadi dalam durasi yang berbeda. Sebagai akibatnya proses matching antara sinyal uji dengan sinyal referensi (*template*) seringkali tidak menghasilkan nilai yang optimal.

Sebuah teknik yang cukup popular di awal perkembangan teknologi pengolahan sinyal wicara adalah dengan memanfaatkan sebuah teknik *dynamic-programming* yang juga lebih dikenal sebagai *Dynamic Time Warping* (DTW). Teknik ini ditujukan utuk mengakomodasi perbedaan waktu antara proses perekaman saat pengujian dengan yang tersedia pada *template* sinyal referensi. Prinsip dasarnya adalah dengan memberikan sebuah rentang '*steps*' dalam ruang (dalam hal ini sebuah frame-frame waktu dalam sample, frame-frame waktu dalam template) dan digunakan untuk mempertemukan lintasan yang menunjukkan *local match* terbesar (kemiripan) antara time frame yang lurus. Total `*similarity cost*' yang diperoleh dengan algorithm ini merupakan sebuah indikasi seberapa bagus *sample* dan *template* ini memiliki kesamaan, yang selanjutnya akan dipilih *best-matching template.*

Dengan tampilan kode program dan contoh hasilnya berikut ini diharapkan akan membantu anda dalam memahaminya.

Kode Program Matlab

Untuk penjalankan program ini anda memerlukan beberapa fungsi berikut:

- **simmx.m** – berguna untuk menghitung *full local-match matrix*, yaitu penghitungan jarak setiap pasangan *frame* dari sinyal sampel dengan sinyal *template*.
- **dp.m** – digunakan untuk menjalankan algorithma *dynamic programming* yang mengikuti langkah-langkah berikut **- (1,1), (0,1)** dan **(1,0)** – dengan bobot yang sama .
- **dp2.m** – merupakan versi alternatif sehingga memungkinkan bagi anda untuk melakukan 5 langkah **- (1,1), (0,1), (1,0), (1,2),** dan **(2,1)** – dengan bobot yang berbeda.

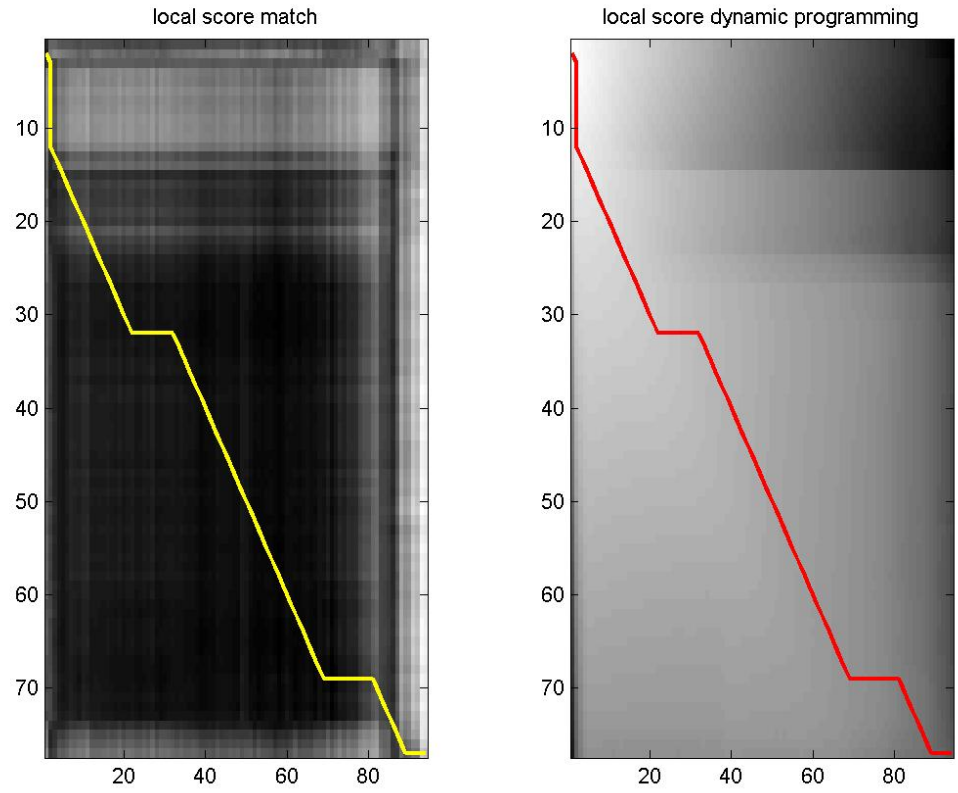
Contoh Program

Anda coba pelajari program berikut ini.

```
%File Name: DTW_01.m
%mengambil dua file speech *.wav
fs=8000;
[d1,fs] = wavread('FILE_U.wav');
[d2,fs] = wavread('FILE_E.wav');

% Mendengarkan secara bersamaan
ml = min(length(d1),length(d2));
%wavplay(d1(1:ml)+d2(1:ml),fs)

% Dalam mode stereo
wavplay([d1(1:ml),d2(1:ml)],fs)

% Menghitung STFT (short time fourier transform) features untuk kedua
%sounds (dengan 25% window overlap)
D1 = specgram(d1,512,fs,512,384);
D2 = specgram(d2,512,fs,512,384);

% Menyusun 'local match' scores matrix sebagai cosine distance antar
%magnitudo STFT
```

```
SM = simmx(abs(D1),abs(D2));

% Lihat hasilnya:
subplot(121)
imagesc(SM)
colormap(1-gray)
title('local score match')
% Anda dapat melihat sebuh strip gelap (high similarity values) mengarah
%turun secara diagonal.
% Gunakan dynamic programming untuk mendapatkan lowest-cost path antara
%pojok cost matrix yang berhadapan
% Catat bahwa kita menggunakan 1-SM karena dp akan menemukan *lowest*
%total cost
[p,q,C] = dp(1-SM);

% Overlay lintasan pada local similarity matrix
hold on;
plot(q,p,'r');
hold off

% Lintasan tampak mengikuti Path jalur dark
% Plot minimum-cost-to-this point matrix
subplot(122)
imagesc(C)
hold on;
plot(q,p,'r');
hold off
title('local score dynamic programming')

% Pojok kanan pada  C memberikan nilai minimum-cost alignment pada
C(size(C,1),size(C,2))

% Hitung frame di dalam D2 yang mengindikasikan match setiap frame
% di dalam D1, sehingga kita dapat me-resynthesize sebuah  warped, versi
% yang diluruskan
D2i1 = zeros(1, size(D1,2));
for i = 1:length(D2i1)
    D2i1(i) = q(min(find(p >= i)));
end

% Interpolasi Phase-vocoder  D2's STFT di bawah kondisi time warp
```

```
D2x = pvsample(D2, D2i1-1, 128);

% Invert kembali ke dalam time domain
d2x = istft(D2x, 512, 512, 128);

% Dengarkan hasilnya sebuah versi warped version sendiri
wavplay(d2x,fs)

% Versi warped ditambahkan ke target asli (untuk mendapatkan fine-tune
%length)
d2x = resize(d2x', length(d1),1);
wavplay(d1+d2x,fs)

% Anda dapat juga mlihat hasilnya pada mode stereo
wavplay([d1,d2x],fs)

% Bandingkan dengan pasangan unwarped:
wavplay([d1(1:ml),d2(1:ml)],fs)
```



Gambar 8. Hasil proses dynamic time warping

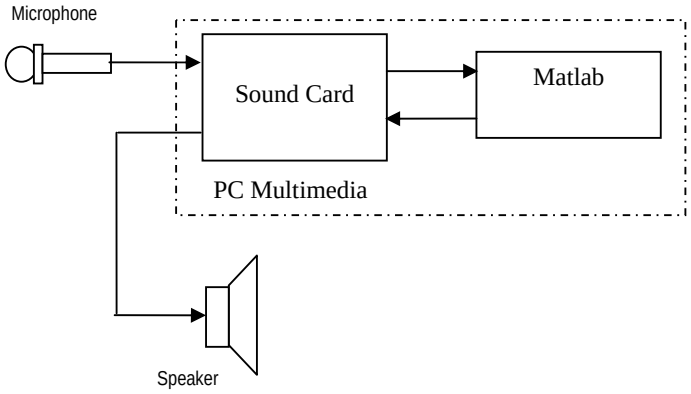
III. PERANGKAT YANG DIPERLUKAN

- 1 (satu) buah *PC Multimedia* lengkap *sound card* dan *microphone*
- Satu perangkat lunak *Matlab under windows*

IV. LANGKAH-LANGKAH PERCOBAAN

4.1. Penataan Perangkat

Sebelum melakukan percobaan harus dilakukan penataan seperti pada Gambar 9 berikut ini.



Gambar 9. Penataan perangkat percobaan pengukuran energi sinyal wicara

PC anda harus dilengkapi dengan peralatan *multimedia* seperti *sound card, speaker active* dan *microphone.* Untuk *microphone* dan *speaker active* bias juga digantikan dengan *head set* lengkap. Sebelum anda memulai praktikum, sebaiknya anda tes dulu, apakah seluruh perangkat multimedia anda sudah terintegrasi dengan PC.

4.2. Penyusunan Modul Praktikum Bebas

Pada praktikum ini anda lakukan penyusunan sebuah program, menjelaskan teori pendukungnya, menampilkan hasil, dan memberikan analisa. Semua anda lakukan dengan kelompok praktikum yang biasa anda jalankan dan dengan cara memilih topik-topik berikut ini:

1. Penggunaan metode *dynamic time warping* untuk pengenalan sinyal wicara
2. Pengkodean sinyal wicara dengan *Linear Predictive Coding*
3. Pengamatan efek frekuensi *warping* pada sinyal wicara
4. Pengaruh pemilihan orde *LPC* pada bentuk *spectral* sinyal wicara

Anda susun dengan memberikan teori pendukung, menyusun algorithma, menyusun program, menampilkan contoh hasilnya dan berikan analisa. Untuk menyelesaikan Modul 6 ini anda memiliki waktu 2 sampai 3 minggu.