

Table of Contents

| | |
|--|-----|
| ACKNOWLEDGMENTS | III |
| ABSTRACT..... | IV |
| 1. SPEECH RECOGNITION | 1 |
| 1.1 Introduction..... | 1 |
| 1.2 ASR Model | 1 |
| 1.2.1 Signal Processing and Feature Extraction | 3 |
| 1.2.2 Time Alignment and Pattern Matching | 3 |
| 1.2.3 Language Processing | 4 |
| 2. STATEMENT OF THE PROJECT | 5 |
| 2.1 Theoretical Background..... | 5 |
| 2.2 Programming Part | 7 |
| 3. DEVELOPMENT ENVIRONMENT OF THE PROJECT | 8 |
| 3.1 Matlab | 8 |
| 3.2 Windows 95 | 8 |
| 3.3 Computer System Used | 9 |
| 4. WAVE EDITOR PROGRAM | 10 |
| 4.1 Main Screen | 10 |
| 4.1.1 Time Intervals | 11 |
| 4.1.2 Frame Length..... | 16 |
| 4.1.3 FFT Operations | 17 |
| 4.2 FFT Display Screen | 17 |

| | |
|---------------------|----|
| 4.2.1 FFT..... | 18 |
| 4.2.2 Contour | 18 |
| 4.2.3 Mesh..... | 21 |
| 4.2.4 Pcolor | 23 |
| APPENDIX..... | 26 |
| REFERENCES | 45 |

ACKNOWLEDGMENTS

I would like to thank to my project advisor Dr.Fikret GÜRGEN and teaching assistant Ali Taylan CEMGIL for their guidance, encouragement and patience throughout my studies. Without their valuable advice and ideas, it would be difficult for me to complete this study.

Special acknowledgments is also made to my roommates in the dormitory; their patience, understanding and sincere help made this study easier.

ABSTRACT

Project Name : A Speech Coding Application (Signal Processing and Feature Extraction Parts).

Term : 1995/1996 II. Semester.

Summary :

The objective of this project is to develop signal processing and feature extraction part of the Automated Speech Recognition System. This report includes speech recognition knowledge and analysis of signal processing environment.

The report also includes the user manual of the application program which is developed to give analysis of signal processing and feature extraction of speech signals which were digitized before. The user of the program can work in an user friendly environment where he/she can zoom in/out signals and their analysis also change certain parameters of the signal processing algorithm and view results in diagrams.

1. SPEECH RECOGNITION

1.1 Introduction

Speech recognition, perception and understanding have been active research fields since 1950's. However, in the mid 1960's introduction of general purpose digital computer led to the discipline we now call Automatic Speech Recognition (ASR). Initially speech recognition focused on recognizing a few words from a single user or speaker dependent recognition. However, as computational power and storage space is increased, the goals of ASR system have become more ambitious, extending to large vocabulary speaker independent systems. To achieve robust recognition over large vocabularies and multiple talkers, techniques from other disciplines, including signal processing, pattern matching and information theory have been employed.

1.2 ASR Model

General Automatic Speech Recognition Model (Figure 1.1), is the primary model by which most researchers in speech recognition have structured their solutions.

The major components of this model include :

- 1 - A signal processing module for obtainig a representation of the speech signal.
- 2 - Feature extraction module for identifiying key components of this representation
and eleminating redundant information.

- 3 - Time alignment and pattern matching algorithms for performing word detection.
- 4 - Language processing for selecting a final word string.

This model emphasizes specific tasks to be performed. There is a consensus on specific signal representation to be used in recognition process. This model's modular partitioning permits to able to focus on specific components of the Automatic Speech Recognition Model.

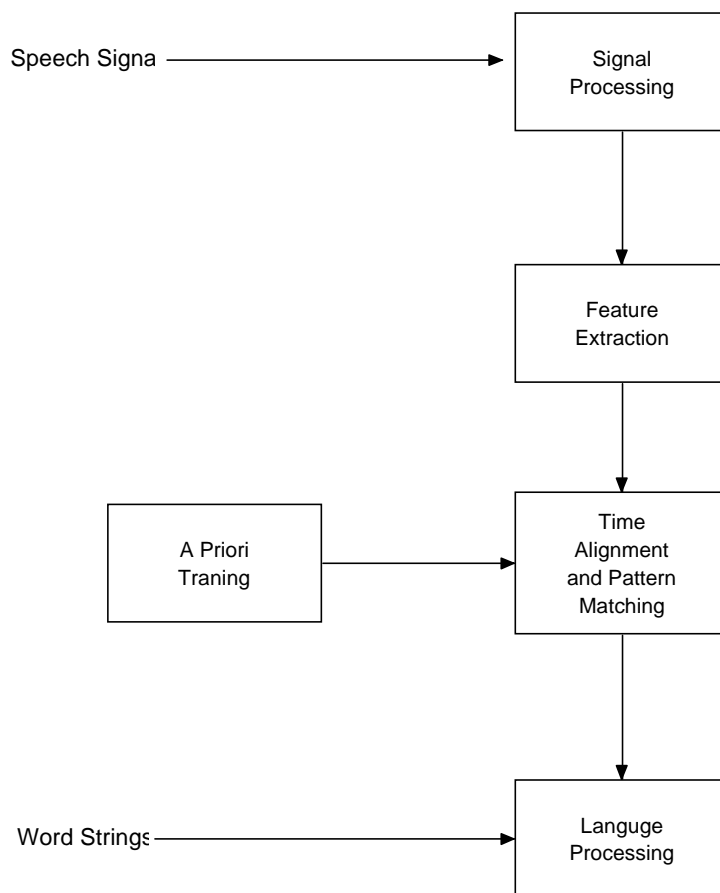


Figure 1.1 - ASR Model

1.2.1 Signal Processing and Feature Extraction

The goal of the signal processing module is to process the sampled speech signal and produce a representation which is independent of amplitude variations, talker stress, or noise introduced from the transmission medium, or channel. In general, the feature extraction module computes a set of parameters which capture transitions in the signal and robust enough to represent any phoneme. These parameters are often called the features and are generally computed at fixed - time intervals. Transitions in the signal are important cues which may indicate the encoding of the phonetic information in the speech signal.

1.2.2 Time Alignment and Pattern Matching

Time alignment and pattern matching algorithms attempt to match a spoken word against a given representation of that word. Speech signal has been transformed into some feature space where a pattern matching technique can be employed. Time alignment refers to alignment of acoustic or phonetic events which have been modeled, to those which are time wrapped in the utterance due to changes in speaking rate. This time warping or temporal variation occurs naturally and variations in speaking rate affect the duration of vowels. Time alignment and pattern matching is usually dependent on the training paradigm used to formulate a word model. The word model consists of a set of parameters extracted during the training phase.

1.2.3 Language Processing

The final stage of the recognition process consists of a language processing module which attempts to resolve the possible word selections using language specific constraints or knowledge.

2. STATEMENT OF THE PROJECT

Project is about signal processing and feature extraction part of the Automated Speech Recognition system. Digitized speech signals which can be processed by a computer will be used in the project. Standard MS Windows wave files are suitable for this project.

2.1 Theoretical Background

Once the speech signal has been digitized, the discrete time representation $s(n)$, is analyzed with short term intervals. Depending upon the application a time interval is selected assuming that speech signal is time variant. Short time Fourier Transform (STFT) is used to evaluate the signal over a time interval.

$$F(u) = \int_{-\infty}^{\infty} f(x) \exp[-j2\pi ux] dx \quad (\text{STFT})$$

A frequency domain representation of signal permits the resonant frequencies of the vocal tract transfer function to be identified.

Discrete Fourier Transform (DFT) of the signal will be computed using Fast Fourier Transform (FFT).

$$F(u) = \frac{1}{N} \sum_{x=0}^{N-1} f(x) \exp[-j2\pi ux / N] \quad (\text{DFT})$$

For finite length sequences which are periodic STFT can be computed by the DFT. FFT is an algorithm which computes the DFT in $O(N \log N)$ operations.

$$W_N = \exp[-j2\pi / N]$$

so DFT becomes

$$F(u) = \frac{1}{N} \sum_{x=0}^{N-1} f(x)W_N^{ux}$$

and N is assumed to be of the form

$$N = 2^n$$

where n is a positive integer and N can be expressed as

$$N = 2M$$

So we can express the equation as

$$F(u) = \frac{1}{2} \left[\frac{1}{M} \sum_{x=0}^{M-1} f(2x)W_M^{ux} + \frac{1}{M} \sum_{x=0}^{M-1} f(2x+1)W_M^{ux}W_{2M}^u \right]$$

Implementation of this algorithm constitutes the FFT algorithm.

2.2 Programming Part

Program will read wave file and will draw the wave file in a amplitude-wave diagram. Secondly a time interval and frame length will be selected. Then for each frame in the time interval FFT of wave file will be calculated.

After taking FFT, values of FFT will be multiplied with conjugate values of it to avoid complex values of it. They will be stored in a array to able to get spectrum energies.

Magnitude spectrum energies of the speech signal will be calculated by Mesh, Pcolor, Contour functions of the array. These all operations will be done on Matlab.

3. DEVELOPMENT ENVIRONMENT OF THE PROJECT

3.1 Matlab

Matlab is an application development environment supporting rapid development of highly efficient mathematical based applications with a minimum of coding. Many of the traditional mathematical requirements are handled for the programmer within the Matlab library, shielding him/her from complicated, or merely repetitive programming tasks.

We haven chosen Matlab to avoid working on mathematical function representations. Main part of the project is reading digitized wave files and taking their FFT algorithms. These operations can be done in Matlab automatically by its own functions which are “Waveread” and “FFT”. Another factor able to represent analysis diagrams. Matlab has “Mesh”, “Pcolor” and “Contour” functions, by them you can easily show your data and compare results.

These summarize the reasons which made us chose Matlab as our software development kit.

3.2 Windows 95

Windows 95 or version 4.0 is an advanced, 32-bit operating system that runs on 4MB machines and provides excellent response time to both 16 and 32-bit applications. It has the advantage of real multitasking which enables many programs to run at a time. This version has many features beyond the limit of version 3.1. In developing wave editor, we have used Windows 95 Version 4.00.950 Release 3.

3.3 Computer System Used

In this project, we used a rather fast 486DX2 / 66 machine which has a hard disk of 420 MB and a double speed CD-ROM drive. The amount of main memory, used in developing this project, is 8MB. With this configuration, we have encountered no serious problems in the design phase.

4. WAVE EDITOR PROGRAM

Program runs on Matlab version 4.0 or later under MS Windows version 3.1 or later. To start the program the name of the wave file should be entered. Program can be called as seen below.

```
Project('start', 'Name of the wave file');
```

4.1 Main Screen

Wave editor brings the wave file automatically on its original time intervals. It displays file in (-128 128) amplitude value by subtracting 128 from its original amplitude value. Also little sample of the wave file was shown at bottom part of the main screen. (Figure 4.1).

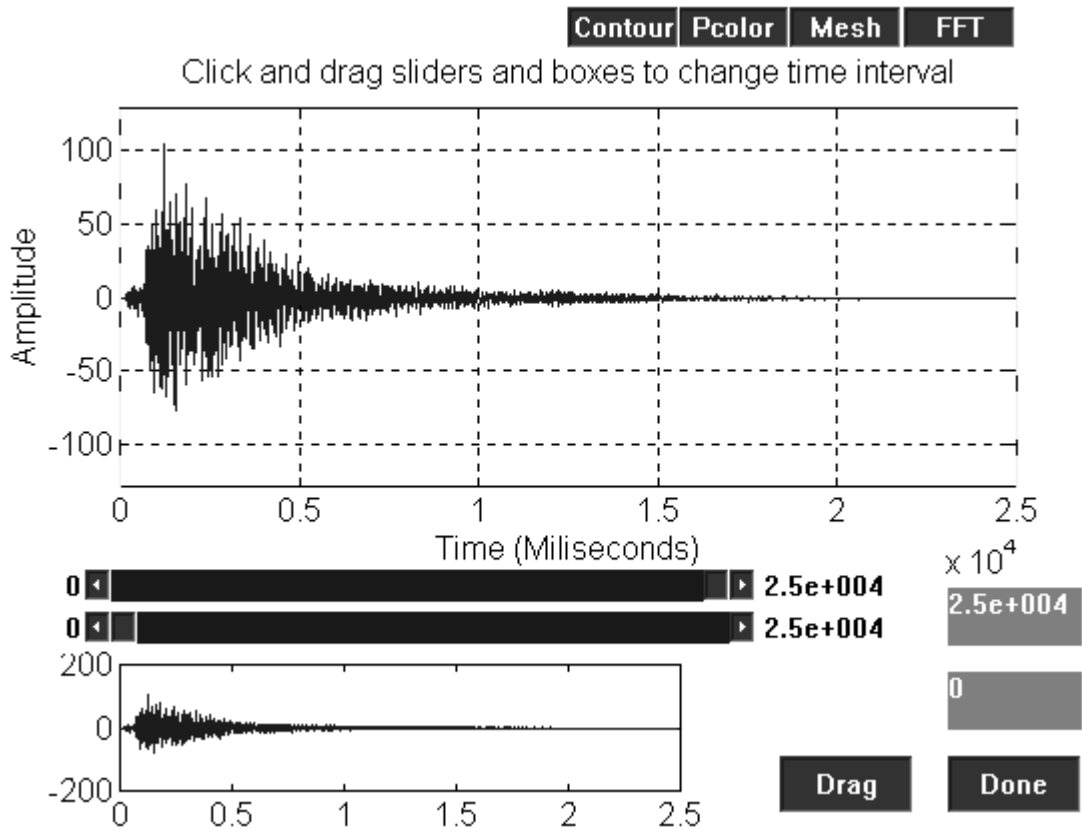


Figure 4.1 Main Screen

4.1.1 Time Intervals

You can change time intervals as you want in three different ways (Figure 4.2):

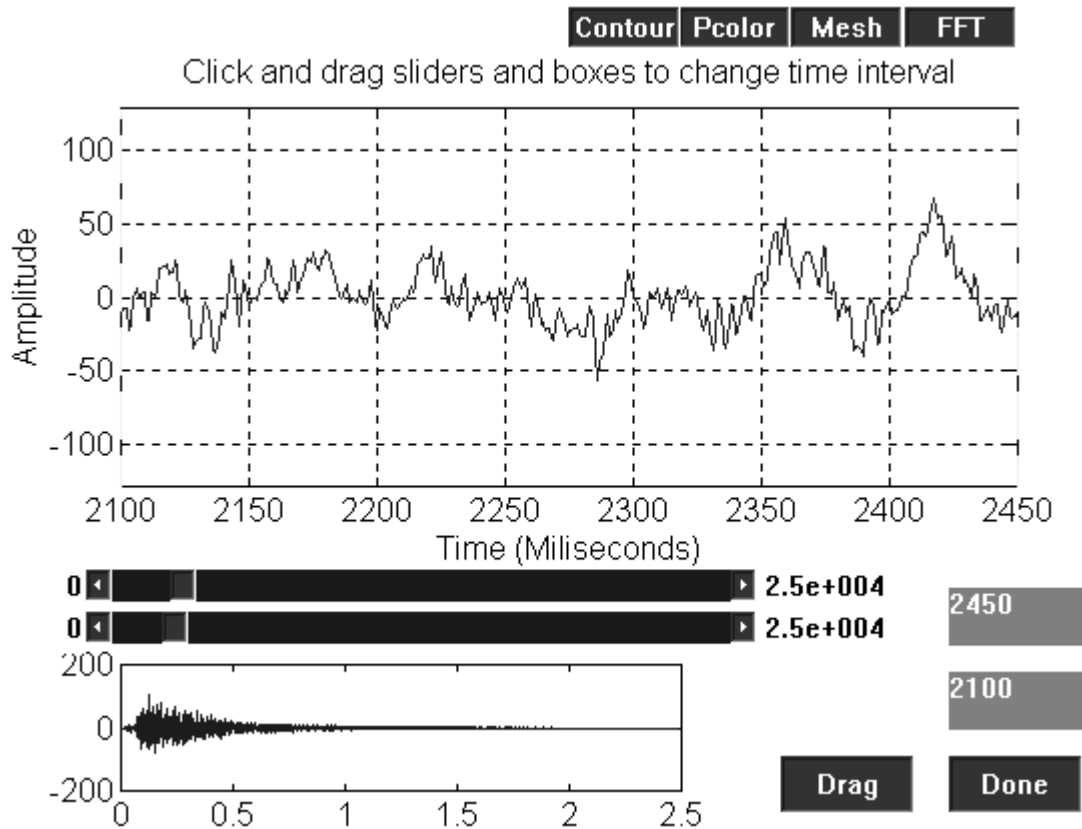


Figure 4.2 Main screen with selected time interval

1. Sliders: There are two different sliders. They both can have value between minimum and maximum time values. The slider value which is higher will be end value and the other will be start value. The values of the sliders can be seen in the edit boxes next to them. Program will automatically redraw the figure in new values.(Figure 4.3)

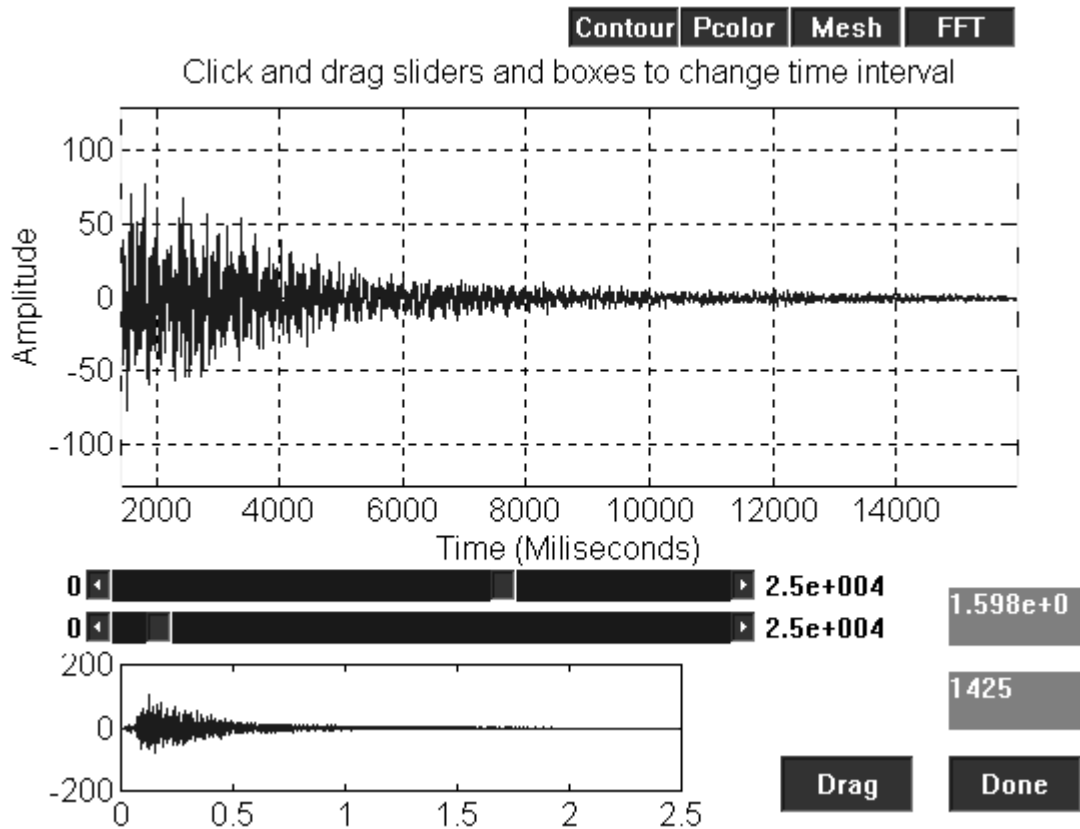


Figure 4.3 Main screen with new slider value

2. Edit Boxes : There are two different edit boxes they will initially present the values of the sliders next to them. You can change their values by clicking on them and writing new values to them. The edit box value which is higher will be end value and other will be start value. The values of edit boxes can be seen in sliders. Program will automatically redraw the figure in new values.(Figure 4.4)

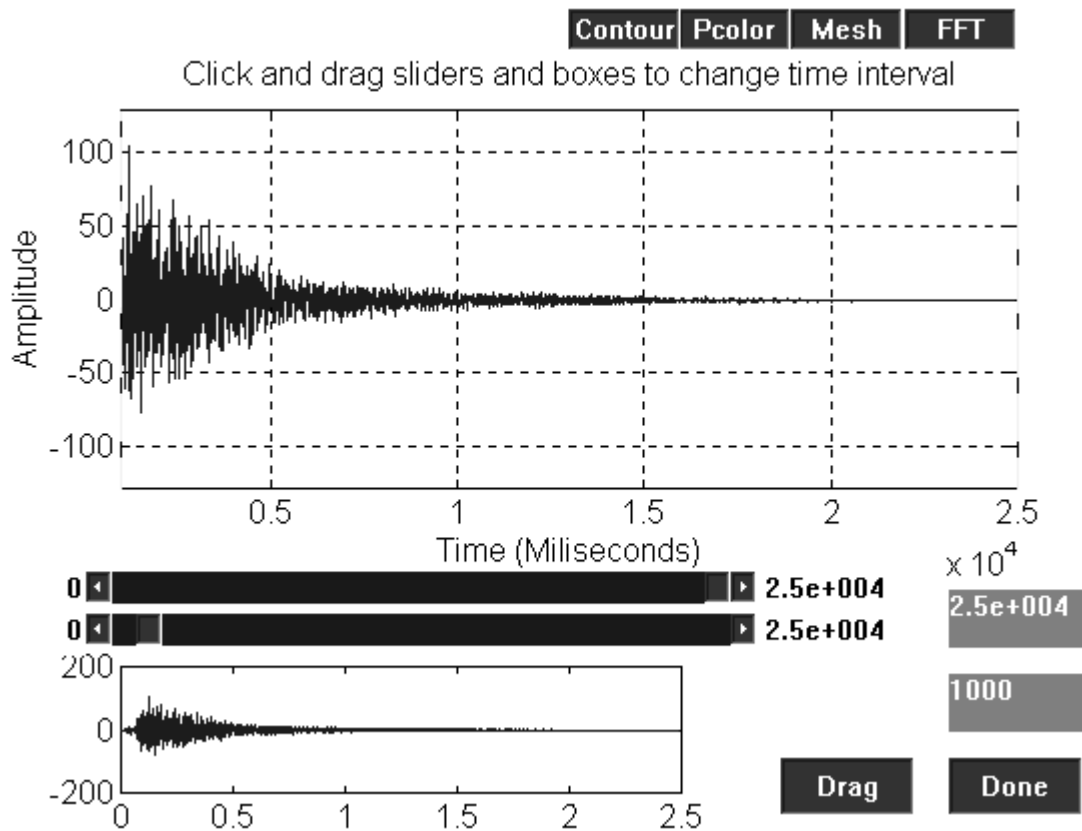


Figure 4.4 Main screen with new edit box value

3. Mouse : You can change values of time interval by clicking the lines on start and end values of time interval. You can bring lines to the interval you want then you should drag "Drag" button to redraw the figure in new values and to see these values in edit boxes and on sliders.(Figure 4.5, Figure 4.6)

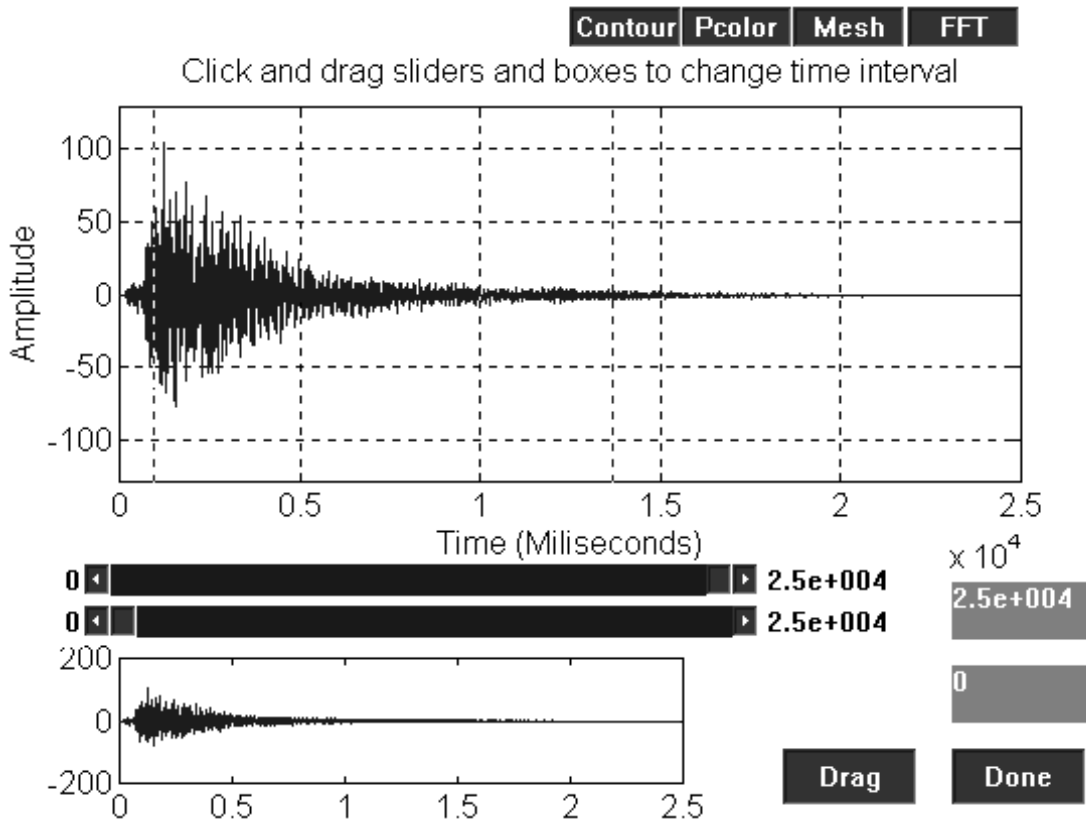


Figure 4.5 Main screen with new mouse lines

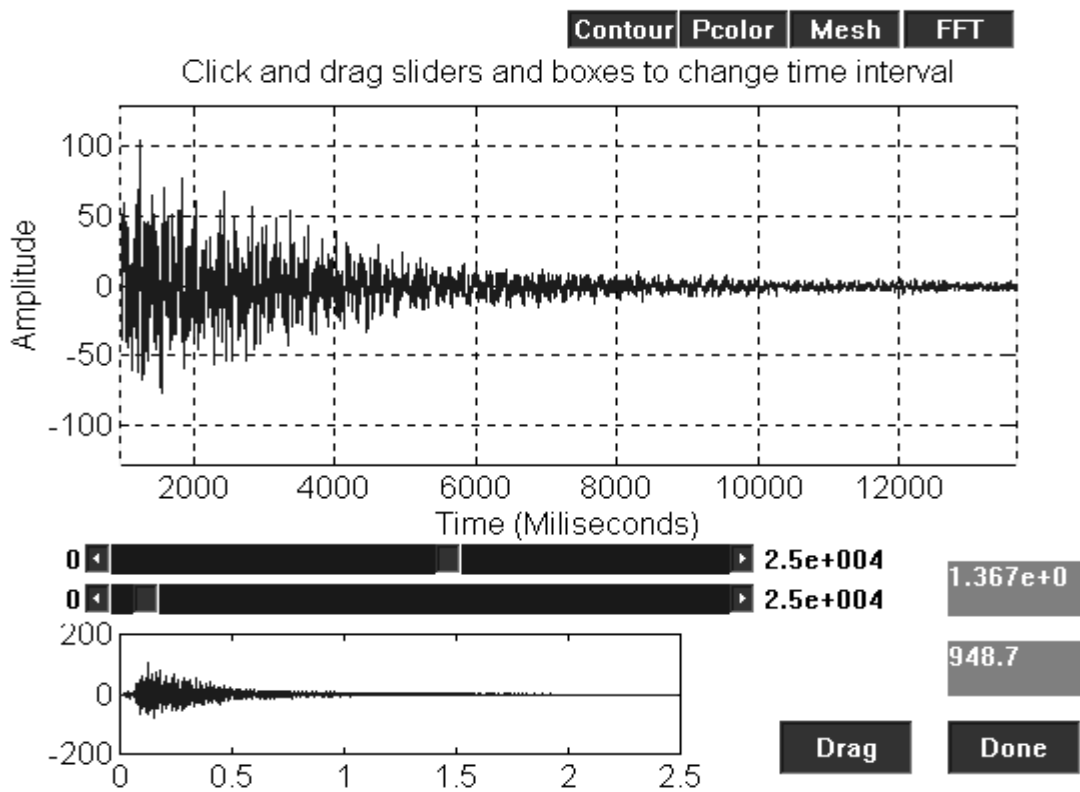


Figure 4.6 Main screen with new time interval

4.1.2 Frame Length

Wave editor does FFT operations and their representations in selected frame lengths. These frame lengths can be in an interval between 128 to 2048. You can select a frame length from the “Frame Length” menu over the main screen. (Figure 4.7) If no frame length is selected program will do FFT operations on a frame length of “512”.

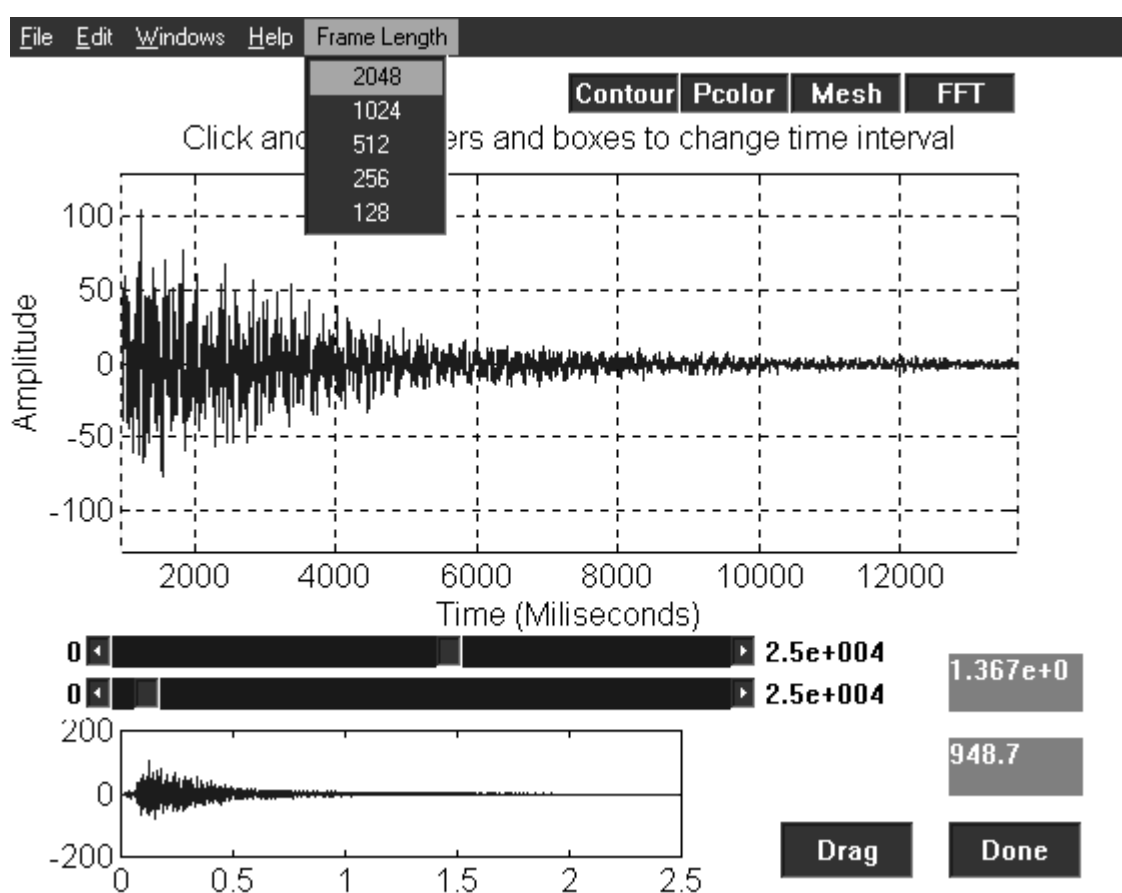


Figure 4.7 Frame Length Menu

4.1.3 FFT Operations

When the time interval and frame length is selected you should drag to the buttons listed on the top of the editor which are “Contour”, “Pcolor”, “Mesh” and “FFT”. (Figure 4.8) By dragging any of them program starts its Fast Fourier Transform operations.



Figure 4.8 FFT menu

Program divides the wave in chosen time intervals to frames which were selected frame length long. Secondly it gets FFT values of these frames one by one. Thirdly it multiplies FFT value of each frame by its conjugate to rescue from complex values. Then program cuts half of the FFT^2 matrix values to avoid displaying the same value two times. Finally it puts FFT^2 matrix values into an array one by one. Then it displays results as chosen before as FFT operations.

4.2 FFT Display Screen

Results of FFT operations are displayed on another screen. These operations are as shown before (Figure 4.8) are “FFT”, “Contour”, “Pcolor” and “Mesh”. The operation chosen from FFT menu will be displayed on this screen.

4.2.1 FFT

It displays FFT² matrix for each frame one by one (Figure 4.9). Also it gives information about start time of the frame, end time of the frame, total number of the frames and which frame was displaying on the screen. By pressing “Enter” you can see next frame. After sawing all frames you can drag “Done” button and return to main screen.

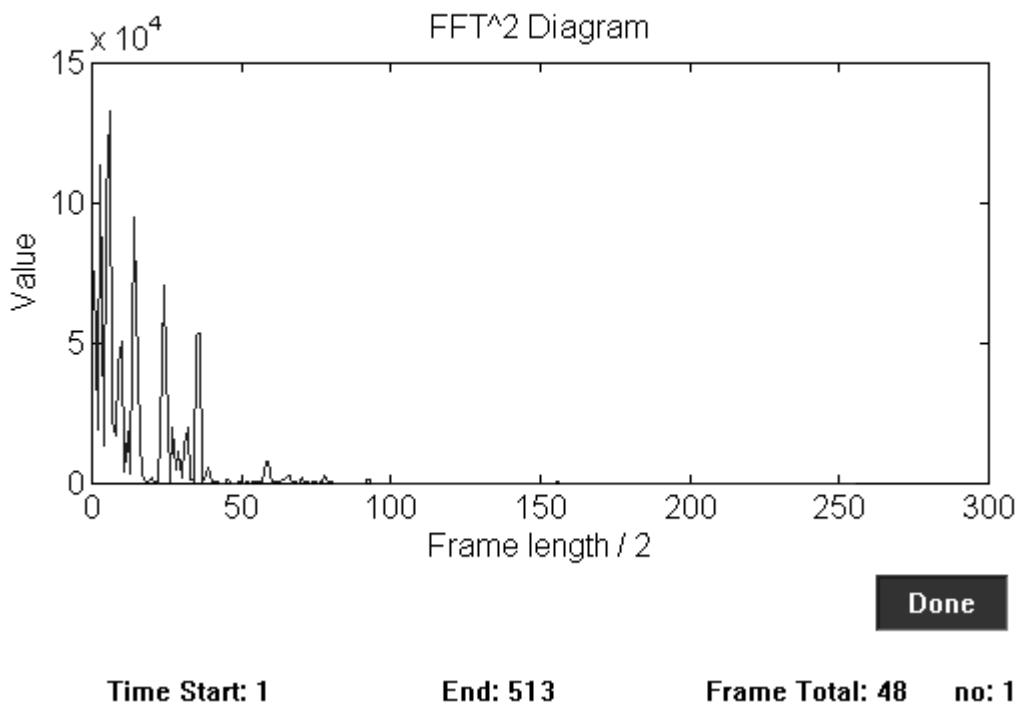


Figure 4.9 FFT frames

4.2.2 Contour

It displays Contour function of the array of FFT² matrices also it displays the original wave in chosen time interval under the Contour diagram (Figure 4.10).

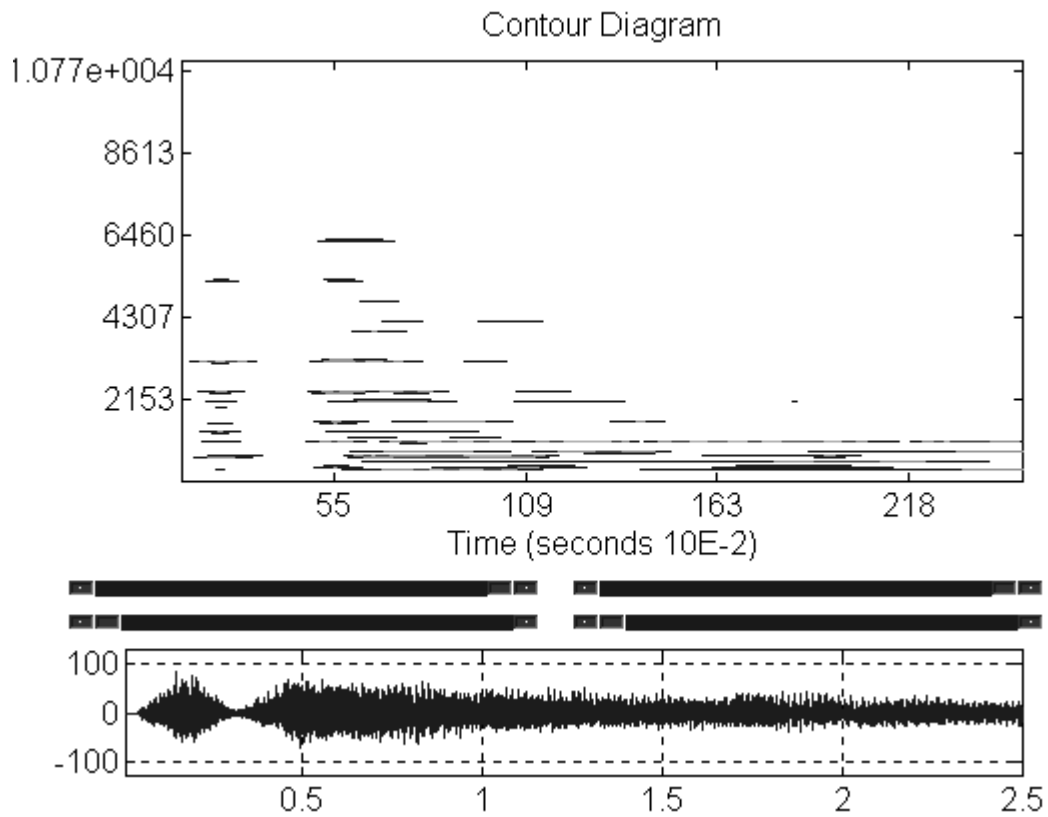


Figure 4.10 Contour diagram

There are four sliders under the Contour diagram. First two sliders are for X axis and other two sliders are for Y axis. By using these sliders user can zoom in or zoom out the diagram (Figure 4.11).

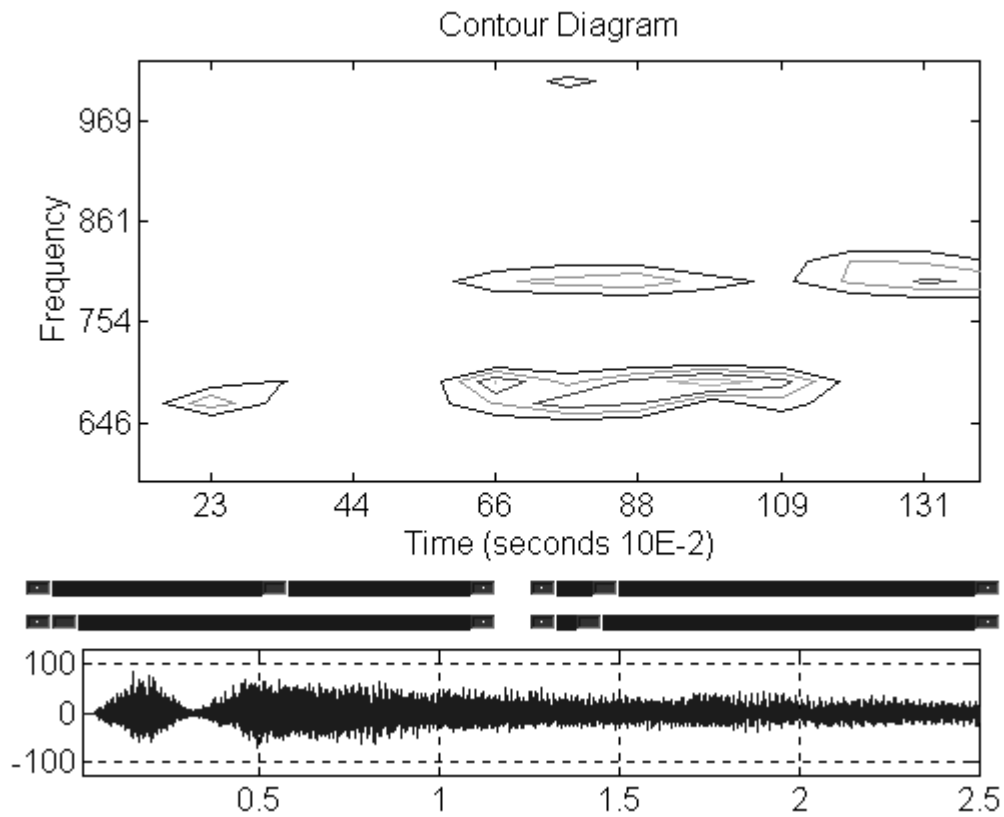


Figure 4.11 Counter diagram as zoomed

X axis of the diagram gives time values of the diagram and Y axis values gives frequency values of the diagram. There is a “Color” menu at top of the screen. (Figure 4.12) which consists of “hsv”, ”gray”, ”hot”, ”cool”, ”bone”, ”copper”, ”pink”, ”prism”, ”jet” and “flag” options. By this menu user can change color by dragging on it.

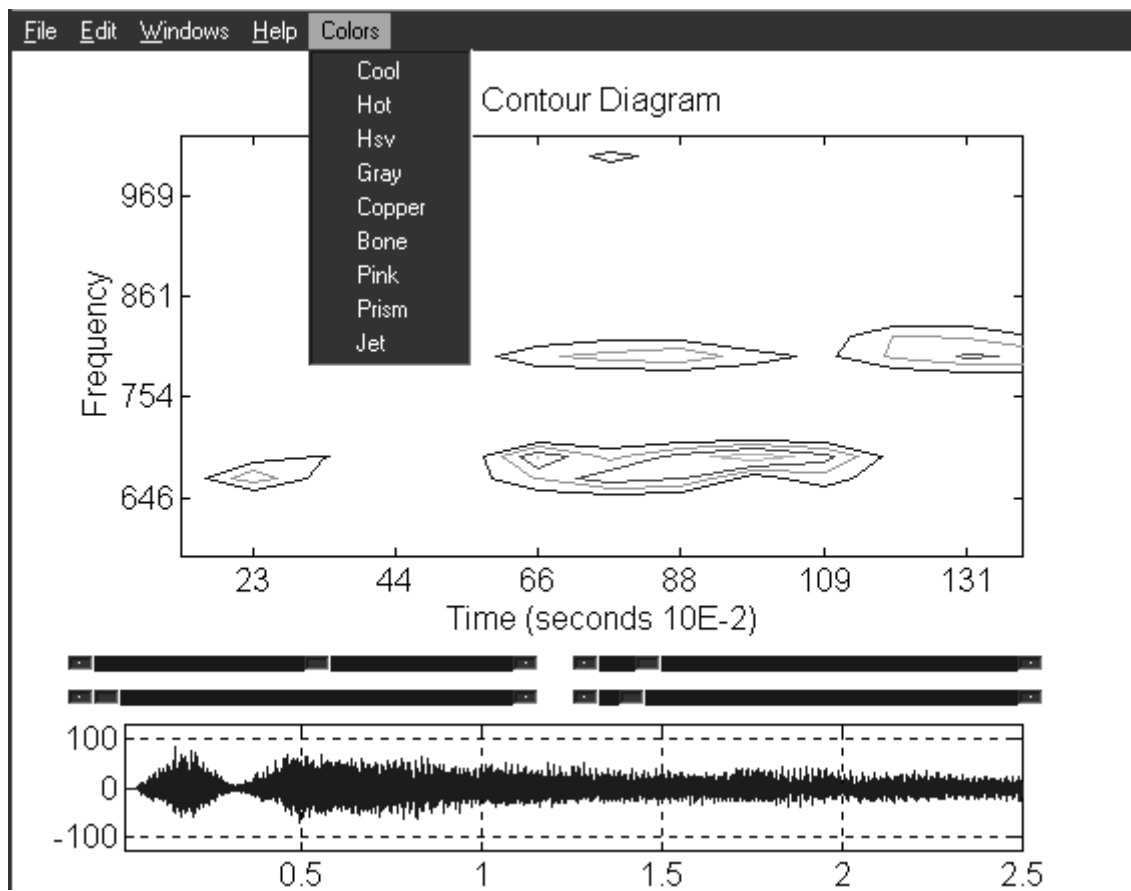


Figure 4.12 Color Menu

4.2.3 Mesh

It displays Mesh function of the array of FFT² matrices also it displays the original wave in chosen time interval under the Mesh function (Figure 4.13).

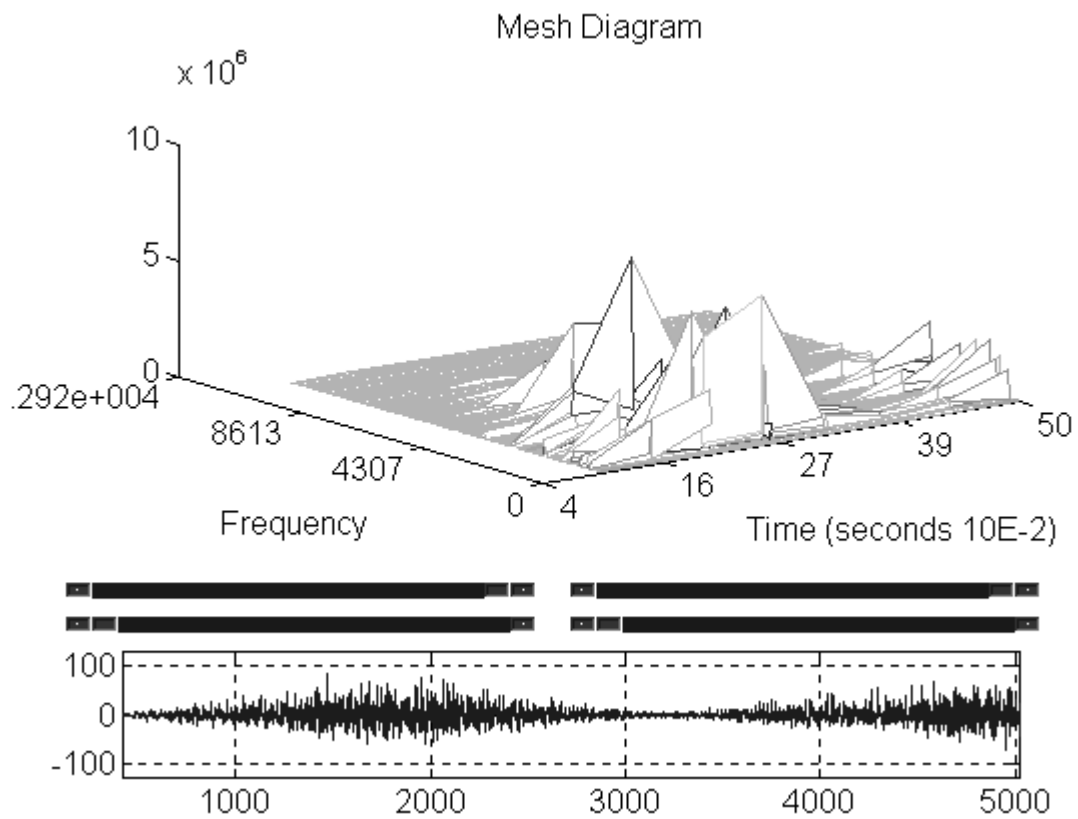


Figure 4.13 Mesh Diagram

There are four sliders under the Mesh diagram. First two sliders are for X axis and other two sliders are for Y axis. By using these sliders user can zoom in or zoom out the diagram (Figure 4.11). User can change color of the diagram by using the “Color” menu.

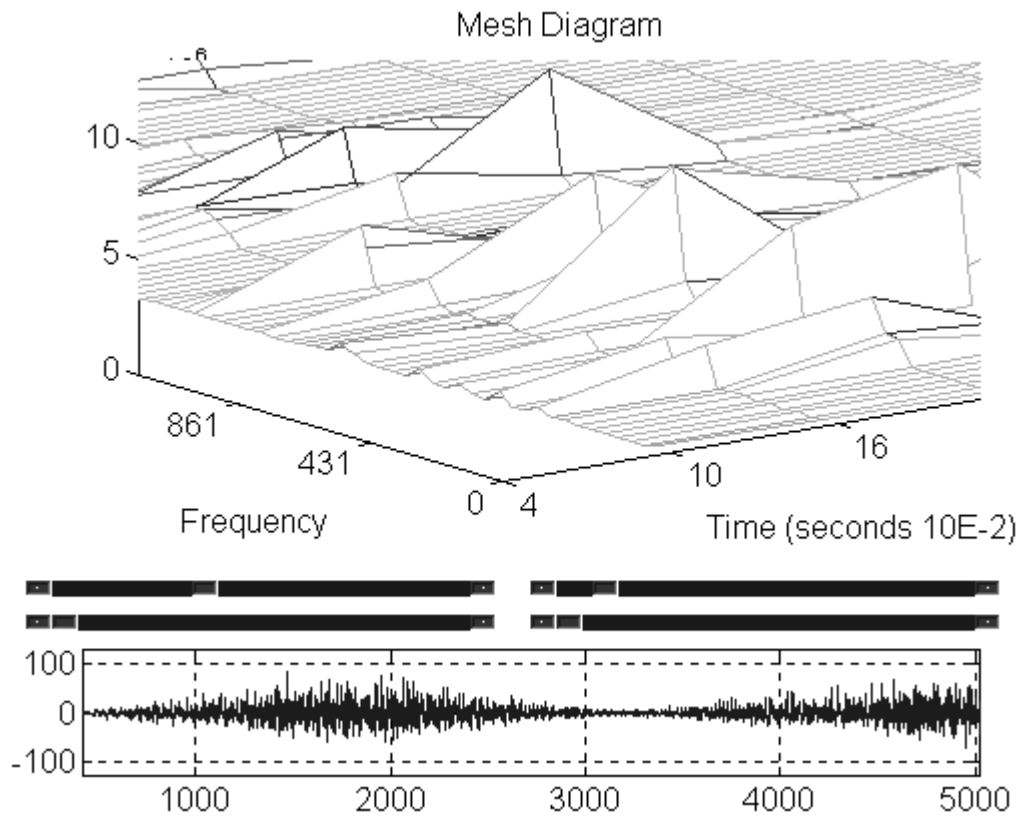


Figure 4.14 Mesh diagram zoomed

4.2.4 Pcolor

It displays Pcolor function of the array of FFT² matrices also it displays the original wave in chosen time interval under the Pcolor function (Figure 4.15).

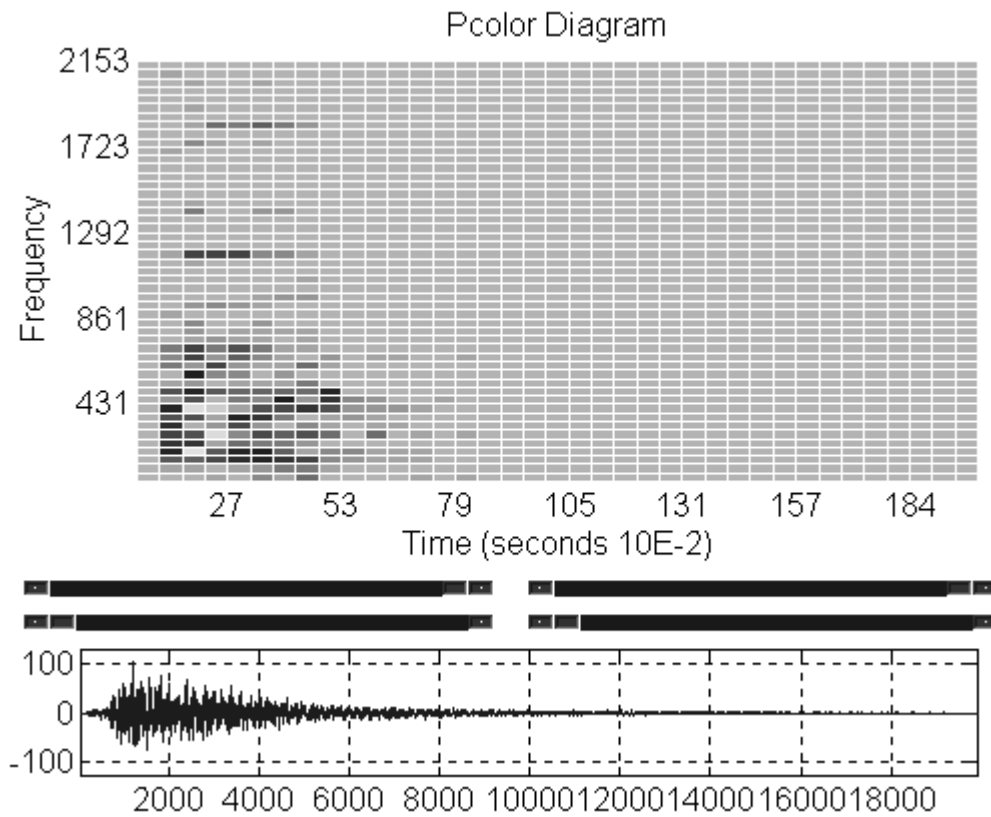


Figure 4.15 Pcolor Diagram

There are four sliders under the Pcolor diagram. First two sliders are for X axis and other two sliders are for Y axis. By using these sliders user can zoom in or zoom out the diagram (Figure 4.16). User can change color of the diagram by using the “Color” menu.

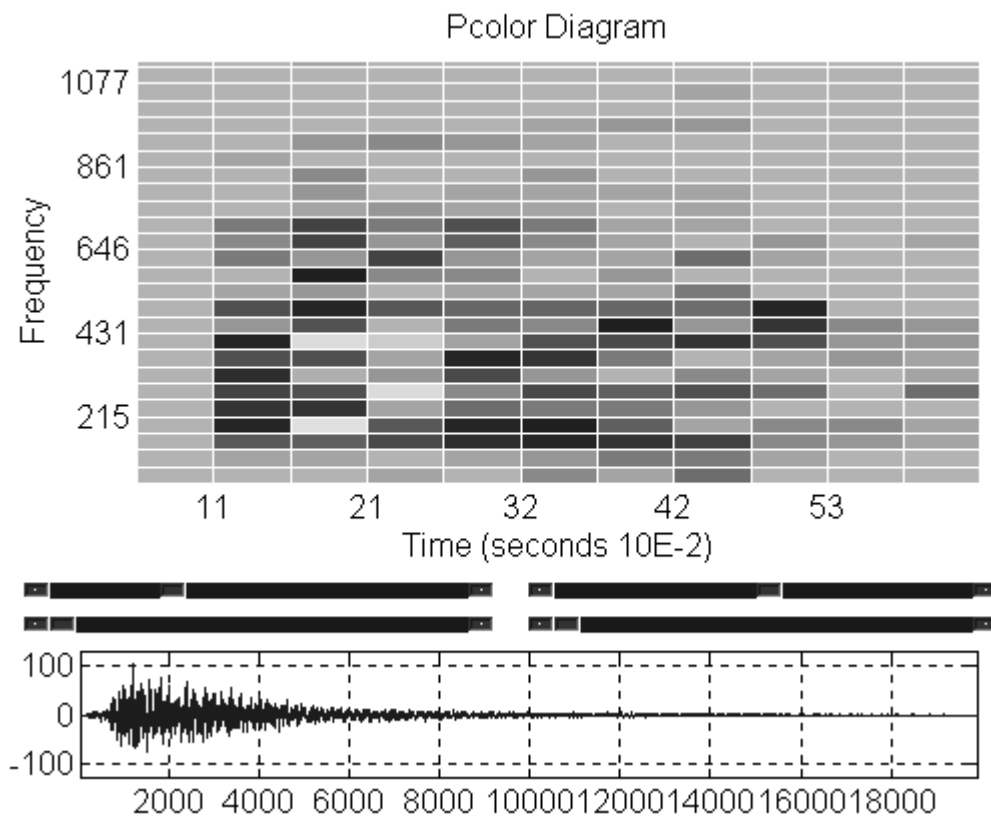


Figure 4.16 Pcolor diagram zoomed

APPENDIX

Matlab code of the wave editor program.

```
function project(action,type1);
%
%     TANKUT CIZMECI CMPE 492 PROJECT
% This project reads a wave and gives spectrum analysis of that wave
% in a user friendly type.
% Program starts as typing "project('start','... .Wav')"
%
if nargin<1,
    action='start';
end;

global DATAS LN1 LN2 W1 FRLEN FS
% DATAS for storing all values in a array
% LN1, LN2 for lines with mouse control
% W1 for storing wave
% FRLEN is storing the frame length for FFT operations
% FS for frequency of the wave

if strcmp(action,'start'),

    [wav1,FS] = loadwave(type1);
% Wave reader of Matlab, it also gets frequency of the wave
    wav1=wav1-128;          % for initializing the wave
    W1=wav1;
    FRLEN = 512;  % starting frame length

% graphics initialization
    figure(1)
    clf reset;
    set(gcf,'Units','normalized','backingstore','off')
```

```

% plotting full wave at bottom part of the screen
    axes('Position',[.10 .05 .50 .15]);
    plot(wav1);
    v=get(gca,'xlim');
    % getting start and finish times of the wave
    min_time = v(1);
    max_time = v(2);
    timebeg = min_time;
    timeend = max_time;
    time1 = timebeg;
    time2 = timeend;

% time slider 1 min time text
uicontrol('style','text','pos',[.05 .21 .02 .05],...
    'Units','normalized','BackgroundColor','black',...
    'ForegroundColor','white','String',num2str(min_time));

% time slider 1 max time text
uicontrol('style','text','pos',[.67 .21 .12 .05 ],...
    'Units','normalized','BackgroundColor','black',...
    'ForegroundColor','white','String',num2str(max_time));

% time slider 1
tslider1 =uicontrol('Style','slider','Position',[.07 .22 .6 .04],...
    'Units','normalized','Value',time1,'Max',max_time,'Min',min_time,...
    'Callback','project("settime",1); project("redraw");');

% time slider 2 min time text
uicontrol('style','text','pos',[.05 .26 .02 .05],...
    'Units','normalized','BackgroundColor','black',...
    'ForegroundColor','white','String',num2str(min_time));

% time slider 2 max time text
uicontrol('style','text','pos',[.67 .26 .12 .05 ],...
    'Units','normalized','BackgroundColor','black',...
    'ForegroundColor','white','String',num2str(max_time));

% time slider 2
tslider2 = uicontrol('Style','slider','Position',[.07 .27 .6 .04],...

```

```

    'Units','normalized','Value',time2,'Max',max_time,'Min',min_time,...
    'Callback','project("settime",1); project("redraw");');

% time box 1
tfield1 = uicontrol('Style','edit','Position',[.84 .12 .12 .07],...
    'Units','normalized','String',num2str(time1),...
    'Callback','project("settime",2); project("redraw");');

% time box 2
tfield2 = uicontrol('Style','edit','Position',[.84 .22 .12 .07],...
    'Units','normalized','String',num2str(time2),...
    'Callback','project("settime",2); project("redraw");');

% close button
close_button=uicontrol('Style','Pushbutton','Position',[.84 .02 .12 .07],...
    'Units','normalized','Callback','project("done")','String','Done');

% drag button (for activating the time interval between mouse lines)
d_button=uicontrol('Style','Pushbutton','Position',[.69 .02 .12 .07],...
    'Units','normalized','Callback','project("redraw")','String','Drag');

% fft button (for displaying FFT^2 diagrams of each frame one by one)
f_button=uicontrol('Style','Pushbutton','Position',[.80 .93 .10 .05],...
    'Units','normalized','Callback','project("stfft",1)','String','FFT');

% mesh button (for displaying mesh diagram of the wave)
m_button=uicontrol('Style','Pushbutton','Position',[.70 .93 .10 .05],...
    'Units','normalized','Callback','project("stfft",2)','String','Mesh');

% pcolor button (for displaying pcolor diagram of the wave)
p_button=uicontrol('Style','Pushbutton','Position',[.60 .93 .10 .05],...
    'Units','normalized','Callback','project("stfft",3)','String','Pcolor');

% contour button (for displaying contour diagram of the wave)
c_button=uicontrol('Style','Pushbutton','Position',[.50 .93 .10 .05],...
    'Units','normalized','Callback','project("stfft",4)','String','Contour');

% frame length menu
    frmenu = uimenu(gcf, 'Label', 'Frame Length');

```

```

        uimenu (frmenu, 'Label','2048','Callback','project("setframe",2048)');
        uimenu (frmenu, 'Label','1024','Callback','project("setframe",1024)');
        uimenu (frmenu, 'Label','512','Callback','project("setframe",512)');
        uimenu (frmenu, 'Label','256','Callback','project("setframe",256)');
        uimenu (frmenu, 'Label','128','Callback','project("setframe",128)');

% wave-time figure
ax_time=axes('Position',[.10 .41 .80 .45],'XLim',[timebeg timeend],'YLim',[0 200]);
    time_line=plot(wav1);
    axis([timebeg timeend -128 128]);
    grid;
    ylabel('Amplitude');
    xlabel('Time (Milliseconds)');
    title('Click and drag sliders and boxes to change time interval');

hold on
% lines under mouse control
LN1 = plot([time1 time1],[-128 128],'Erasemode','xor','LineStyle','--');
LN2 = plot([time2 time2],[-128 128],'Erasemode','xor','LineStyle','--');
hold off

% mouse operations
set(LN1,'ButtonDownFcn','project("line",1)');
set(LN2,'ButtonDownFcn','project("line",2)');

% data array for storing values
DATAS = [tslider1; tslider2; tfield1; tfield2; timebeg; timeend;...
        ax_time; time_line; time1; time2; max_time; min_time];

elseif strcmp(action,'settime'),
% time interval set operations

if (type1==1),
    time1=get(DATAS(1),'value');
    time2=get(DATAS(2),'value');

else    min_time=DATAS(12);
        max_time=DATAS(11);
        time1=str2num(get(DATAS(3),'string'));

```

```

        time2=str2num(get(DATAS(4),'string'));
    if (time1>max_time),
        time1=max_time;
    end;
    if (time1<min_time),
        time1=min_time;
    end;

        if (time2>max_time),
            time2=max_time;
        end;
        if (time2<min_time),
            time2=min_time;
        end;
    end;

    if (time1>time2),
        timebeg=time2;
        timeend=time1;
    else
        timebeg=time1;
        timeend=time2;
    end;

    DATAS(5)=timebeg;
    DATAS(6)=timeend;
    DATAS(9)=time1;
    DATAS(10)=time2;

elseif strcmp(action,'moveline'),
% moving line for mouse operations on time intervals

    pt=get(gca,'currentpoint');
    y=pt(2,1);
    if y > 0,
        if (type1==1),
            time1=y;
            DATAS(9)=time1;
            DATAS(5)=time1;

```

```

        set(LN1,'XData',[time1 time1]);
    else
        time2=y;
        DATAS(10)=time2;
        DATAS(6)=time2;
        set(LN2,'XData',[time2 time2]);
    end;
end;

elseif strcmp(action,'redraw'),
% redrawing the wave in choosen time interval

    timebeg=DATAS(5);
    timeend=DATAS(6);
    time1=DATAS(9);
    time2=DATAS(10);
    % setting mouse lines
    set(LN1,'XData',[timebeg timebeg]);
    set(LN2,'XData',[timeend timeend]);
    % slider and text box setting
    set(DATAS(3),'string',num2str(time1));    % set text box 1
    set(DATAS(1),'value',time1);            % set slider 1
    set(DATAS(4),'string',num2str(time2));    % set text box 2
    set(DATAS(2),'value',time2);            % set slider 2
    % redrawing the wave diagram
    set(DATAS(7),'XLim',[timebeg timeend]);
    axis([timebeg timeend -128 128]);

drawnow;

elseif strcmp(action,'line'),
% setting mouse operations
    if (type1==1),
        set(LN1,'LineStyle',':')
        set(gcf,'WindowButtonMotionFcn','project("moveline",1)');
    else
        set(LN2,'LineStyle',':')
        set(gcf,'WindowButtonMotionFcn','project("moveline",2)');
    end;
end;

```

```

elseif strcmp(action,'setframe'),
% setting frame length from window menu
    FRLEN = type1;

elseif strcmp(action,'stfft'),
% starting FFT operations part
    timebeg=DATAS(5);
    timeend=DATAS(6);

    proc1('start',FRLEN,type1,W1,timebeg,timeend,FS);
    % for FFT operations calling procedure

elseif strcmp(action,'done'),
% for exiting the program
    clf reset;
    figure(2);
    clf reset;
    clear global DATAS LN1 LN2 W1 FRLEN
end

function proc1(action,frlen,in1,wave1,startt,endt,fs1);
%
% Procedure for drawing FFT frames, Mesh, Pcolor, Contour operations
%
% frlen      - frame length for FFT operations
% in1       - type of the operation
% wave1     - input wave
% startt    - start time of the wave
% endt      - finish time of the wave
% fs1      - frequency of the wave

global VARS VAR2 VAR3 WV1 ARR2 FS FRLEN FRTOT FRTOT1
% VARS - data array for storing values
% VAR2 - for storing axis
% VAR3 - for storing diagrams
% WV1 - for storing the wave
% ARR2 - for storing array of FFT values

```

```

% FS - for storing frequency of the wave
% FRLEN - for storing the frame length
% FRTOT - for storing the total number of frames
% FRTOT1 - used in frequency and time calculations

if strcmp(action,'start'),

    figure(2)
    % graphics initialization
        clf reset;
        set(gcf,'Units','normalized','backingstore','off');
    % values for standardization of the graphs
    FS = fs1;
    FRLEN = frlength;
    countval = 10;
    pcolval = 20;
    valmax = 40;
    valmin = 1;
    val1 = 1;
    valst = 1;
    val2 = 40;
    valend = 40;
    up1 = 1;
    up2 = 50;
    upmax = 50;
    upmin = 1;
    upst = 1;
    upend = 50;

% color window menu
    clmenu = uimenu(gcf,'Label','Colors');
    uimenu(clmenu,'Label','Cool','Callback','proc1("redrawfr","cool",1));
    uimenu(clmenu,'Label','Hot','Callback','proc1("redrawfr","hot",2));
    uimenu(clmenu,'Label','Hsv','Callback','proc1("redrawfr","hsv",3));
    uimenu(clmenu,'Label','Gray','Callback','proc1("redrawfr","gray",4));
    uimenu(clmenu,'Label','Copper','Callback','proc1("redrawfr","copper",5));
    uimenu(clmenu,'Label','Bone','Callback','proc1("redrawfr","bone",6));
    uimenu(clmenu,'Label','Pink','Callback','proc1("redrawfr","pink",7));
    uimenu(clmenu,'Label','Prism','Callback','proc1("redrawfr","prism",8));

```

```

uimenu (clmenu, 'Label','Jet','Callback','proc1("redrawfr","jet",9));

% slider 1
slider1 =uicontrol('Style','slider','Position',[.05 .22 .42 .02],...
    'Units','normalized','Value',val1,'Max',valmax,'Min',valmin,...
    'Callback','proc1("setx",1); proc1("redrawfr","hsv",3);');

% slider 2
slider2 = uicontrol('Style','slider','Position',[.05 .26 .42 .02],...
    'Units','normalized','Value',val2,'Max',valmax,'Min',valmin,...
    'Callback','proc1("setx",1); proc1("redrawfr","hsv",3);');

% slider 3
slider3 =uicontrol('Style','slider','Position',[.50 .22 .42 .02],...
    'Units','normalized','Value',up1,'Max',upmax,'Min',upmin,...
    'Callback','proc1("sety",1); proc1("redrawfr","hsv",3);');

% slider 4
slider4 = uicontrol('Style','slider','Position',[.50 .26 .42 .02],...
    'Units','normalized','Value',up2,'Max',upmax,'Min',upmin,...
    'Callback','proc1("sety",1); proc1("redrawfr","hsv",3);');

% calculating total number of frames
FRTOT=round((endt-startt)/FRLEN)-1;
if (startt<1),
    startt=1;
end;
st1=startt;
if ((endt-startt)<FRLEN),
    FRTOT=1;
    en1=endt;
else
    en1=st1+FRLEN;
end;

% calculating FFT values for each frame
frlen2 = FRLEN/2;
for j=1:FRTOT

```

```

x=fft(wave1(st1:en1)); % calculating fft of the wave
xs=x.*conj(x);
% multiply with its conjugate for avoiding complex values
arr1(:,j)=xs(1:frlen2);
% put values into an array for each frame
if (in1==1),
    % fft figure
    clf;
    axes('Position',[.10 .30 .80 .50]);
    plot(xs(1:frlen2));
    ylabel('Value');
    xlabel('Frame length / 2');
    title('FFT^2 Diagram');
    % frame number
    uicontrol('style','text','pos',[.84 .02 .12 .05 ],...
'Units','normalized','BackgroundColor','black',...
'ForegroundColor','white','String',['no: ' num2str(j)]);

    % total frames
    uicontrol('style','text','pos',[.64 .02 .20 .05 ],...
'Units','normalized','BackgroundColor','black',...
'ForegroundColor','white','String',['Frame Total: ' num2str(FRTOT)]);

    % finish time of the frame
    uicontrol('style','text','pos',[.34 .02 .25 .05 ],...
'Units','normalized','BackgroundColor','black',...
'ForegroundColor','white','String',['End: ' num2str(en1)]);

    % start time of the frame
    uicontrol('style','text','pos',[.04 .02 .30 .05 ],...
'Units','normalized','BackgroundColor','black',...
'ForegroundColor','white','String',['Time Start: ' num2str(st1)]);

    % close button
    close_button=uicontrol('Style','Pushbutton','Position',[.80 .12 .12 .07],...
'Units','normalized','Callback','proc1("done")','String','Done');

    pause;
end;

```

```

st1=st1+FRLEN;
en1=en1+FRLEN;
if (en1>endt),
    en1=endt;
end;
end;

if (in1>1),

    % wave-time figure at bottom part of the screen
ax_wave = axes('Position',[.10 .05 .80 .15],'XLim',[startt endt],'YLim',[0 200]);
wave_line = plot(WV1);
axis([startt endt -128 128]);
grid;

    % mesh, pcolor, contour operations
arr1=[];arr2=[]; % arrays for frequency and time values
valend=FRTOT;valmax=FRTOT;val2=FRTOT; % slider values
FRTOT1=FRTOT*100;
n = size(arr1,1);
if (in1==2),
    % mesh operation
ax_frame = axes('Position',[.15 .40 .75 .50]);
VAR3 = mesh(arr1);
ylabel('Frequency');
xlabel('Time (seconds 10E-2)');
title('Mesh Diagram');
    % getting axis values from the diagram
v=get(gca,'xlim');
val1=v(1);valmin=v(1);valst=v(1);
val2=v(2);valmax=v(2);valend=v(2);
set(slider1,'Value',v(1),'Min',v(1),'Max',v(2));
set(slider2,'Value',v(2),'Min',v(1),'Max',v(2));
v=get(gca,'ylim');
up1=v(1);upmin=v(1);upst=v(1);
up2=v(2);upmax=v(2);upend=v(2);
set(slider3,'value',v(1),'min',v(1),'max',v(2));
set(slider4,'value',v(2),'min',v(1),'max',v(2));

```

```

% calculating frequency values
yt=get(gca,'ytick');
n1=length(yt);
for j=1:n1,
    fryt=yt(j)*FS/FRLEN;
    arrt1(j)=(round(fryt));
end;
arrt2=str2mat(num2str(arrt1(1)));
for j=2:length(arrt1),
    arrt2=str2mat(arrt2,num2str(arrt1(j)));
end;
% calculating time values
set(gca,'yTickLabels',arrt2);
arrt1=[];arrt2=[];
xt=get(gca,'xtick');
n1=length(xt);
for j=1:n1
    fryt=startt/100+(xt(j)*(endt-startt)/FRTOT1);
    arrt1(j)=(round(fryt));
end;
arrt2=str2mat(num2str(arrt1(1)));
for j=2:n1,
    arrt2=str2mat(arrt2,num2str(arrt1(j)));
end;
set(gca,'xTickLabels',arrt2);
elseif (in1==3),
    % pcolor operation
ax_frame = axes('Position',[.15 .40 .75 .50],'XLim',[valst valend],'YLim',[upst
upend]);

VAR3 = pcolor(arr1);
axis([valst valend upst upend]);
ylabel('Frequency');
xlabel('Time (seconds 10E-2)');
title('Pcolor Diagram');
% calculating frequency values
yt=get(gca,'ytick');
n1=length(yt);
for j=1:n1,
    fryt=yt(j)*FS/FRLEN;

```

```

        arrt1(j)=(round(fryt));
    end;
    arrt2=str2mat(num2str(arrt1(1)));
    for j=2:length(arrt1),
        arrt2=str2mat(arrt2,num2str(arrt1(j)));
    end;
    set(gca,'yTickLabels',arrt2);
    % calculating time values
    arrt1=[];arrt2=[];
    xt=get(gca,'xtick');
    n1=length(xt);
    for j=1:n1
        fryt=startt/100+(xt(j)*(endt-startt)/FRTOT1);
        arrt1(j)=(round(fryt));
    end;
    arrt2=str2mat(num2str(arrt1(1)));
    for j=2:n1,
        arrt2=str2mat(arrt2,num2str(arrt1(j)));
    end;
    set(gca,'xTickLabels',arrt2);

elseif (in1==4),
    % contour operation
    ax_frame = axes('Position',[.15 .40 .75 .50]);
    VAR3 = contour(arr1);
    ylabel('Frequency');
    xlabel('Time (seconds 10E-2)');
    title('Contour Diagram');
    % getting axis values
    v=get(gca,'xlim');
    val1=v(1);valmin=v(1);valst=v(1);
    val2=v(2);valmax=v(2);valend=v(2);
    set(slider1,'Value',v(1),'Min',v(1),'Max',v(2));
    set(slider2,'Value',v(2),'Min',v(1),'Max',v(2));
    v=get(gca,'ylim');
    up1=v(1);upmin=v(1);upst=v(1);
    up2=v(2);upmax=v(2);upend=v(2);
    set(slider3,'value',v(1),'min',v(1),'max',v(2));
    set(slider4,'value',v(2),'min',v(1),'max',v(2));

```

```

        % calculating frequency values
        yt=get(gca,'ytick');
        n1=length(yt);
        for j=1:n1,
            fryt=yt(j)*FS/FRLLEN;
            arrt1(j)=(round(fryt));
        end;
        arrt2=str2mat(num2str(arrt1(1)));
        for j=2:length(arrt1),
            arrt2=str2mat(arrt2,num2str(arrt1(j)));
        end;
        set(gca,'yTickLabels',arrt2);
        % calculating time values
        arrt1=[];arrt2=[];
        xt=get(gca,'xtick');
        n1=length(xt);
        for j=1:n1
            fryt=startt/100+(xt(j)*(endt-startt)/FRTOT1);
            arrt1(j)=(round(fryt));
        end;
        arrt2=str2mat(num2str(arrt1(1)));
        for j=2:n1,
            arrt2=str2mat(arrt2,num2str(arrt1(j)));
        end;
        set(gca,'xTickLabels',arrt2);
    end;
end;

VARS = [ startt; endt; in1; frlength; valmax; valmin; valst; valend;...
        slider1; slider2; val1; val2; countval; pcolval; up1; up2;...
        slider3; slider4; upmax; upmin; upst; upend];
VAR2 = ax_frame;
WV1 = wave1;
ARR2 = arr1;

elseif strcmp(action,'redrawfr'),
% redrawing mesh,pcolor or contour diagram

startt = VARS(1);

```

```

endt = VARS(2);
ctype = in1;
in1 = VARS(3);
valst = VARS(7);
valend = VARS(8);
val1 = VARS(11);
val2 = VARS(12);
up1 = VARS(15);
up2 = VARS(16);
upst = VARS(21);
upend = VARS(22);
% setting slider values
set(VARS(9),'value',val1);
set(VARS(10),'value',val2);
set(VARS(17),'value',up1);
set(VARS(18),'value',up2);

arrt1=[];arrt2=[];
if (in1>1),
    n = size(ARR2,1);
    if (in1==2),
        % mesh diagram redraw
        VAR3 = mesh(ARR2);
        set(gca,'XLim',[valst valend],'YLim',[up1 up2]);
        colormap(frlength);
        ylabel('Frequency');
        xlabel('Time (seconds 10E-2)');
        title('Mesh Diagram');
        % calculating frequency values
        yt=get(gca,'ytick');
        n1=length(yt);
        for j=1:n1,
            fryt=yt(j)*FS/FRLEN;
            arrt1(j)=(round(fryt));
        end;
        arrt2=str2mat(num2str(arrt1(1)));
        for j=2:length(arrt1),
            arrt2=str2mat(arrt2,num2str(arrt1(j)));
        end;
end;

```

```

set(gca,'yTickLabels',arrt2);
% calculating time values
arrt1=[];arrt2=[];
xt=get(gca,'xtick');
n1=length(xt);
for j=1:n1
    fryt=startt/100+(xt(j)*(endt-startt)/FRTOT1);
    arrt1(j)=(round(fryt));
end;
arrt2=str2mat(num2str(arrt1(1)));
for j=2:n1,
    arrt2=str2mat(arrt2,num2str(arrt1(j)));
end;
set(gca,'xTickLabels',arrt2);
drawnow;
elseif (in1==3),
    % pcolor diagram redraw
set(VAR2,'XLim',[valst valend],'YLim',[up1 up2]);
VAR3 = pcolor(ARR2);
axis([valst valend up1 up2]);
colormap(frlength);
ylabel('Frequency');
xlabel("Time (seconds 10E-2)");
title('Pcolor Diagram');
% calculating frequency values
yt=get(gca,'ytick');
n1=length(yt);
for j=1:n1,
    fryt=yt(j)*FS/FRLEN;
    arrt1(j)=(round(fryt));
end;
arrt2=str2mat(num2str(arrt1(1)));
for j=2:length(arrt1),
    arrt2=str2mat(arrt2,num2str(arrt1(j)));
end;
set(gca,'yTickLabels',arrt2);
% calculating time values
arrt1=[];arrt2=[];
xt=get(gca,'xtick');

```

```

n1=length(xt);
for j=1:n1
    fryt=startt/100+(xt(j)*(endt-startt)/FRTOT1);
    arrt1(j)=(round(fryt));
end;
arrt2=str2mat(num2str(arrt1(1)));
for j=2:n1,
    arrt2=str2mat(arrt2,num2str(arrt1(j)));
end;
set(gca,'XTickLabels',arrt2);
drawnow;
elseif (in1==4),
    % contour diagram redraw
    set(VAR2,'XLim',[valst valend],'YLim',[up1 up2]);
    % setup of colors
    if (ctype == 3),
        VAR3 = contour(ARR2);
    elseif (ctype == 4),
        VAR3 = contour(ARR2,'w');
    elseif (ctype == 5),
        VAR3 = contour(ARR2,'g');
    elseif (ctype == 1),
        VAR3 = contour(ARR2,'c');
    elseif (ctype == 2),
        VAR3 = contour(ARR2,'y');
    elseif (ctype == 6),
        VAR3 = contour(ARR2,'b');
    elseif (ctype == 7),
        VAR3 = contour(ARR2,'b');
    elseif (ctype == 8),
        VAR3 = contour(ARR2);
    elseif (ctype == 9),
        VAR3 = contour(ARR2);
    end;
    axis([valst valend up1 up2]);
    ylabel('Frequency');
    xlabel('Time (seconds 10E-2)');
    title('Contour Diagram');
    % calculating frequency values

```

```

        ytick=get(gca,'ytick');
        n1=length(ytick);
        for j=1:n1,
            fryt=ytick(j)*FS/FRLEN;
            arrt1(j)=(round(fryt));
        end;
        arrt2=str2mat(num2str(arrt1(1)));
        for j=2:length(arrt1),
            arrt2=str2mat(arrt2,num2str(arrt1(j)));
        end;
        set(gca,'yTickLabels',arrt2);
        % calculating time values
        arrt1=[];arrt2=[];
        xtick=get(gca,'xtick');
        n1=length(xtick);
        for j=1:n1
            fryt=startt/100+(xtick(j)*(endt-startt)/FRTOT1);
            arrt1(j)=(round(fryt));
        end;
        arrt2=str2mat(num2str(arrt1(1)));
        for j=2:n1,
            arrt2=str2mat(arrt2,num2str(arrt1(j)));
        end;
        set(gca,'xTickLabels',arrt2);
        drawnow;
    end;
end;

```

```
elseif strcmp(action,'setx'),
```

```
% setting values in x axis by using slider 1 and slider 2
```

```

    val1=get(VARS(9),'value');
    val2=get(VARS(10),'value');

```

```

    if (val1>val2),
        valst=val2;
        valend=val1;
    else
        valst=val1;

```

```

        valend=val2;
    end;

    VARS(7)=valst;
    VARS(8)=valend;
    VARS(11)=val1;
    VARS(12)=val2;

elseif strcmp(action,'sety'),
% setting values in y axis by using slider 3 and slider 4

    up1=get(VARS(17),'value');
    up2=get(VARS(18),'value');

    if (up1>up2),
        upst=up2;
        upend=up1;
    else
        upst=up1;
        upend=up2;
    end;

    VARS(21)=upst;
    VARS(22)=upend;
    VARS(15)=up1;
    VARS(16)=up2;

elseif strcmp(action,'done'),
% return to figure 1 in FFT^2 diagram
    clf reset;
    clear global VARS VAR2 VAR3 WV1 ARR2 FS FRLEN FRTOT FRTOT1
    figure(1);

else clf reset;
% exit from figure 2
    clear global VARS VAR2 VAR3 WV1 ARR2 FS FRLEN FRTOT FRTOT1
end;

```

REFERENCES

1. Allerhand M., Knowledge Based Speech Pattern Recognition, Kagen-Page Fifth Generation Computing Series, 1987.
2. Gonzales R.C., Woods R.E. Digital Image Processing, Addison Wesley, 1992.
3. Gougin P.T. "A Fast Spectral Estimation Algorithm Based on FFT" IEEE Transactions on Signal Processing, Vol. 42, pp. 1317-1322, 1992.
4. Harms B. "Computing Time Frequency Distributions" IEEE Transactions on Signal Processing, Vol. 39, pp. 727-729, 1994.
5. Lee Kai Fu, Automatic Speech Recognition, The Development of the SPHNX System, Kluwar Academic Publishers, 1989.
6. Morgan D.P., Scofield C.L., Neural Networks and Speech Processing, Kluwar Academic Publishers, 1991.
7. Oppenheim A.V., Willsky A.S., Young I.T., Signals and Systems, Prentice Hall, 1991.
8. Matlab User's Guide, The Mathworks Inc. 1992.
9. Matlab Release Notes Version 4.1, The Mathworks Inc. 1993.
10. Matlab New Features Guide, The Mathworks Inc. 1993.
11. Matlab External Interface Guide, The Mathworks Inc. 1993.
12. Matlab Building a Graphical User Interface, The Mathworks Inc. 1993.

