Scilab Manual for Audio and Speech Processing Part 1 by Prof Muralikrishna H Electronics Engineering Manipal Institute of Technology[1]

Solutions provided by
Prof Muralikrishna H
Electronics Engineering
Manipal Institute of Technology

June 3, 2026

[1]Funded by a grant from the National Mission on Education through ICT, http://spoken-tutorial.org/NMEICT-Intro. This Scilab Manual and Scilab codes written in it can be downloaded from the "Migrated Labs" section at the website http://scilab.in

Contents

List of Experiments

Experiment: 1

Basic Operations on Signals

Scilab code Solution 1.1 Generating a sin wave

```
1 //Generating a sine wave
2 //
3 //OS: windows 7
4 //Scilab: 5.5.2
5 //
6 clc;
7 close;
8 clear;
9 f=100;
10 Fs=44000; //sampling frequency
11 t=0:1/Fs:.02; // 2 cycles only
12 y=sin(2*%pi*f*t);
13 plot(y);
14 title('Sine wave')
15 xlabel('sample number');
16 ylabel('amplitude');
```

Scilab code Solution 1.2 Program to compare two sine waves with different frequencies

```
1  //Program to compare two sine waves with different
       frequencies.
2  //
3  //OS: windows 7
4  //Scilab: 5.5.2
5  //
6
7  clc;
8  close;
9  clear;
10 f1=100;
11 F=4400; //sampling frequency
12 t=0:1/F:.02;
13 y=sin(2*%pi*f1*t);
14
15 f2=200;
16 z=2*sin(2*%pi*f2*t);
17
18
19 subplot(211)
20 plot(y);
21 xlabel('sample number');
22 ylabel('amplitude');
23
24 subplot(212)
25 plot(z)
26 xlabel('sample number');
27 ylabel('amplitude');
```

Scilab code Solution 1.3 Program to generate squarewave

```
1  //Program to generate squarewave
2  //
3  //OS: windows 7
4  //Scilab: 5.5.2
```

```
5 //
6 f=100;
7 Fs=44000; //sampling frequency
8 t=0:1/Fs:.02; // 2 cycles only
9 y=squarewave(2*%pi*f*t);
10 plot(y);
11 xlabel('sample number');
12 ylabel('amplitude');
```

Scilab code Solution 1.4 Program to add sin and squarewave

```
1 //Program to add sin and squarewave
2 //
3 //OS: windows 7
4 //Scilab: 5.5.2
5 //
6 clc;
7 close;
8 clear;
9 f=100;
10 Fs=44000; //sampling frequency
11 t=0:1/Fs:.02; // 2 cycles only
12 y=squarewave(2*%pi*f*t);
13 z=sin(2*%pi*f*t);
14 subplot(311)
15 plot(y)
16 title('Suqre wave')
17 subplot(312)
18 plot(z)
19 title('Sine wave')
20 zz=y+z;  // Adding two signals
21 subplot(313)
22 plot(zz);
23 title('Result of adding sin and square wave')
```

Experiment: 2

Concatenation of Speech Signals

Scilab code Solution 2.1 Program to concatenate speech signals

```
1  //This code Reads 2 audio files and coccatenates
       them.
2  //
3  //OS: windows 7
4  //Scilab: 5.5.2
5  //
6  clc;
7  close;
8  clear;
9  [y,Fs] = wavread('C:\Users\ACER\Desktop\Number_1.wav
       '); // reading the audiofile 1
10 [x,Fs]= wavread('C:\Users\ACER\Desktop\Number_2.wav'
       );// reading the audiofile 2
11 z=[y,x];
12 sound(z,Fs)// playing concatenated file.
13 t=(0:length(z)-1)*1/Fs;
14 plot(t,z)
15 title('Concatenated Speech signal waveform')
16 xlabel('Time in seconds')
```

```
17 ylabel('Amplitude')
```

Scilab code Solution 2.2 Concatenating into Stereo file

```
1 //This Program Reads 2 audio files and coccatenates
    them into a stereo file.
2 //
3 //OS: windows 7
4 //Scilab: 5.5.2
5 //
6 clc;
7 close;
8 clear;
9 [y,Fs] = wavread('C:\Users\ACER\Desktop\Number_1.wav
    '); // reading the audiofile 1
10 [x,Fs]= wavread('C:\Users\ACER\Desktop\Number_2.wav'
    );// reading the audiofile 2
11 z=[x;y]; // Concatenating. //Two files must be of
    same length
12 sound(z,Fs)// playing concatenated file. // Observe
    two seperate signals played together.
```

Scilab code Solution 2.3 Stereo to Mono conversion

```
1 //This Program Reads a stereo file and converts it
    in to individual files.
2 //
3 //OS: windows 7
4 //Scilab: 5.5.2
5 //
6 clc;
7 close;
8 clear;
```

```
9  [y,Fs] = wavread('C:\Users\ACER\Desktop\stereo.wav')
       ; // Reading a stereo file
10 //sound(y,Fs);  // Uncomment to play stereo file
11 left=y(1,:);    //Extracts the first row
12 sound(left,Fs); //Uncomment to get left side
       component
13 t=(0:length(left)-1)*1/Fs;
14 subplot(211)
15 plot(t,left)
16 title('Left side audio component')
17 xlabel('Time in seconds')
18 ylabel('Amplitude')
19
20 right=y(2,:);
21 sound(right,Fs);  // Uncomment to get right side
       componet.
22 subplot(212)
23 plot(t,left)
24 title('Right side audio component')
25 xlabel('Time in seconds')
26 ylabel('Amplitude')
```

Experiment: 3

Finding the Resonating Frequency of a Tuning Fork

Scilab code Solution 3.1 Resonating Freq using spectrum

```
1  // Program to find the Resonating frequency of a
       Tuning fork
2  // using spectrum of the signal
3  //
4  //OS: windows 7
5  //Scilab: 5.5.2
6  //
7  clc;
8  close;
9  clear;
10 [y,Fs] = wavread('C:\Users\ACER\Desktop\Freq512Hz.
       wav');
11 y=y(200:1900); // considering only a segment to
       reduce the amount of computation
12 t=(0:length(y)-1)/Fs;
13 subplot(211)
14 plot(t,y)
15 xlabel('Time in seconds')
16 title('Original signal')
```

```
17 Y=abs(fft(y));  //find the fourier transform
18
19 l = length(Y)/2;
20 f = (0:(l-1))*Fs/(2*l); //modify the x axis to
     represent frequency instead of samples
21
22 abs_fft = abs(Y(1:l));
23 subplot(212)
24 plot(f,abs_fft);   //plot magnitude of fourier
     transform
25 title('Fourier Transform of signal')
26 xlabel('Frequency')
27 ylabel('Amplitude')
28
29 peak = max(abs_fft); //find the first maxima of the
     spectrum
30 peakfreq = [f(abs_fft == peak)]
31 disp('Resonating frequency of given tuning fork (in
     Hz): ');
32 disp(peakfreq); //display the contained frequencies
```

Scilab code Solution 3.2 Resonating Freq in mixed frequency signal

```
1 // This program finds the frequency components in a
     given mixed frequency signal
2 // Mixed frequency signal was recorded by playing 2
     Tunng forks simultaneously
3 //
4 //OS: windows 7
5 //Scilab: 5.5.2
6 //
7 clc;
8 close;
9 clear;
10
```

```
11 [y,Fs] = wavread('C:\Users\ACER\Desktop\
      Mixed512and384.wav'); //read the audio file
12
13 Y=abs(fft(y));  //find the fourier transform
14
15 l = length(Y)/2;
16 f = (0:(l-1))*Fs/(2*l); //modify the x axis to
      represent frequency instead of samples
17
18 abs_fft = abs(Y(1:l));
19 plot(f,abs_fft);   //plot magnitude of fourier
      transform
20 title('Fourier Transform of signal')
21 xlabel('Frequency')
22 ylabel('Amplitude')
23
24 peak = max(abs_fft); //find the first maxima of the
      spectrum
25 secpeak = max(abs_fft(abs_fft<max(abs_fft)));  //
      find the second maxima of the spectrum
26 peakfreq = [f(abs_fft == peak), f(abs_fft == secpeak
      )];   //find the frequency corresponding to the
      peaks
27 peakfreq = gsort(peakfreq); //sort the detected
      frequencies
28 peakfreq(abs(max(peakfreq) - max(peakfreq(peakfreq<
      max(peakfreq)))) < 10) = []; //remove frequencies
       that are very close to one another
29
30 disp('Given signal has following frequencies (in Hz)
      : ');
31 disp(peakfreq); //display the contained frequencies
```

Scilab code Solution 3.3 Resonating Freq using Autocorrelation

```
1  // Finding the Resonating frequency of a Tuning fork
       using
2  // Autocorrelation method.
3
4  //
5  //OS: windows 7
6  //Scilab: 5.5.2
7  //
8  clc;
9  close;
10 clear;
11 [y,Fs] = wavread('C:\Users\ACER\Desktop\Freq512Hz.
       wav');
12 y=y(200:1900); // considering only a segment to
       reduce the amount of computation
13 t=(0:length(y)-1)/Fs;
14 subplot(211)
15 plot(t,y)
16 xlabel('Time in seconds')
17 title('Original signal');
18 c1=xcorr(y);
19 subplot(212)
20 lag=(1:length(c1))-ceil(length(c1)/2);
21 plot(lag,c1)
22 title('Autocorrelation if the given segment')
23
24 c1 = -c1; //flip the ACF to get two maxima peaks
25
26 [a,b] = max(c1); //find the first maxima
27 [c,d] = max(c1(b+1:$));  //find the second maxima
28
29 numsamples = d+1;   //get the number of samples
       between the peaks
30
31 freqncy = Fs/numsamples;   //calculate the frequency
       of the signal
32
33 disp('Resonating frequency in Hz: ');
```

```
34 disp(freqncy);     //display the frequency
```