# Audio Signal Generation

Young Won Lim
2/2/18

Please send corrections (or suggestions) to youngwlim@hotmail.com.

This document was produced by using OpenOffice.

# Based on

Signal Processing with Free Software : Practical Experiments
F. Auger

# SOX information

http://sox.sourceforge.net/Docs/Documentation

http://www.thegeekstuff.com/2009/05/sound-exchange-sox-15-examples-to-manipulate-audio-files/

http://billposer.org/Linguistics/Computation/SoxTutorial.html

# Audacity information

http://www.library.kent.edu/files/SMS_Audacity_Basics.pdf

http://manual.audacityteam.org/man/tutorials.html

https://multimedia.journalism.berkeley.edu/tutorials/audacity/

http://ctlt.jhsph.edu/help/views/tutorials/audacity/GuideToUsingAudacity.pdf

# Visual Analyser

http://www.sillanumsoft.org/

# TimeFreqPlot

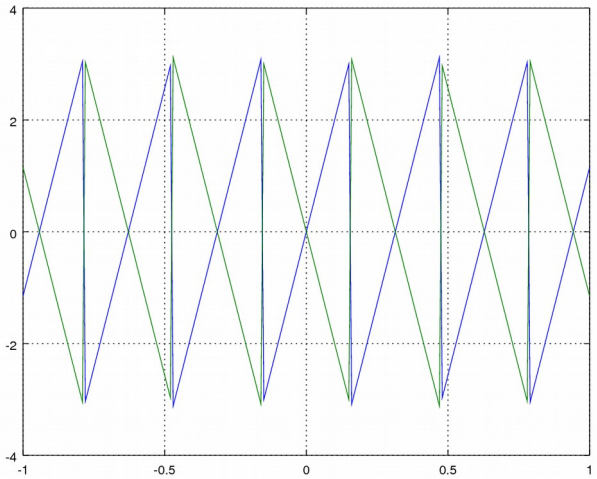
```
# Prevent Octave from thinking that this
# is a function file:
1;

#---------------------------------------
# n : time index vector
# x : time domain signal vector
# k : frequency index vector
# X : frequency domain signal vector
#---------------------------------------
function TimeFreqPlot(n, x, k, X, mode)

NF = length(n);

if (strmatch(typeinfo(mode), "range"))
  trange = mode; frange = mode;
else
  if (strmatch(mode, "half"))
    trange = 1:NF; frange = 1:NF/2;
  elseif (strmatch(mode, "full"))
    trange = 1:NF; frange = 1:NF;
    fs = k(2)+k(NF)
    X = fftshift(X); k = fftshift(k);
    k(1:NF/2) = k(1:NF/2) - fs;
  else
    trange = 1:NF; frange = 1:NF;
  endif
endif

subplot(3,1,1);
p = stem(n(trange), x(trange), 'k'); grid on;
xlabel('Time, t (s)'); ylabel('x(t)');

subplot(3,1,2);
p = stem(k(frange), abs(X(frange)), 'k');
set(p,'LineWidth',2,'MarkerSize',4); grid on;
xlabel('Harmonic number, k');
ylabel('|X(k)|');

subplot(3,1,3);
p = stem(k(frange), angle(X(frange)), 'k');
set(p,'LineWidth',2,'MarkerSize',4); grid on;
xlabel('Harmonic Number, k');
ylabel('Ang{X[k]}');

endfunction
```

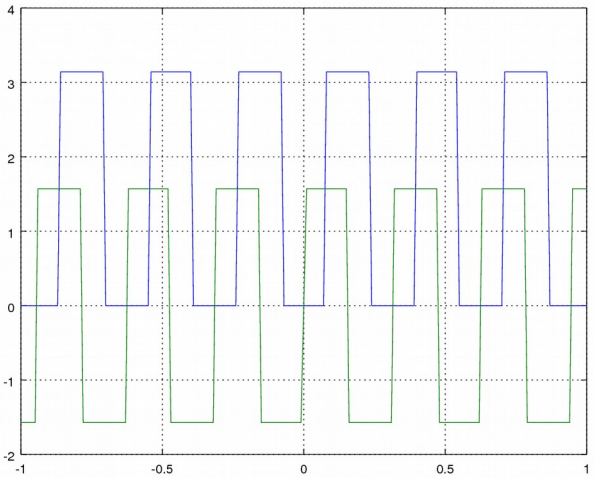
# wavplot

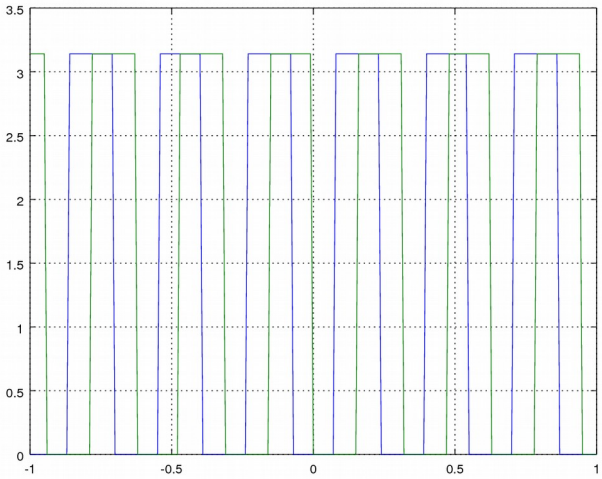
```
function wavplot(fname)

[x, fs, nbits] = wavread(fname);
nn = length(x);
printf('fs = %f \n', fs);
printf('nn = %d \n', nn);

n = (0 : nn-1) / fs ;
k = fs * (0 : nn-1) / nn ;
X = fft( x );

source "../0.util.octave/util.m"
#TimeFreqPlot(n, x, k, X, "half")
#TimeFreqPlot(n, x, k, X, (1:1024)+1024*0.5)
#TimeFreqPlot(n, x, k, X, (1*1024:2*1024))
#TimeFreqPlot(n, x, k, X, (35200-0.5*1024:35200+0.5*1024))
TimeFreqPlot(n, x, k, X, (2*1024:3*1024))


endfunction
```

# Arg of cos(ωt) and sin(ωt)

```
t = -1:0.01:1;
wt = 20 * t;

y1 = e.^(i*wt);
y2 = e.^(-i*wt);
y3 = y1+y2;
y4 = y1-y2;
y5 = y3/2;
y6 = y4/(2*i);

plot(t, arg(y1), t, arg(y2));
plot(t, arg(y3), t, arg(y4));
plot(t, arg(y5), t, arg(y6));
```



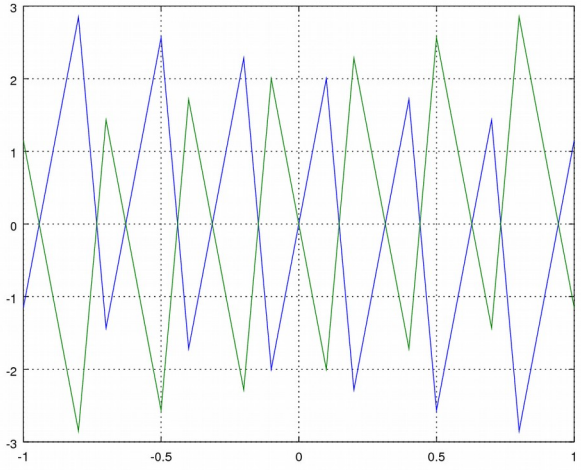
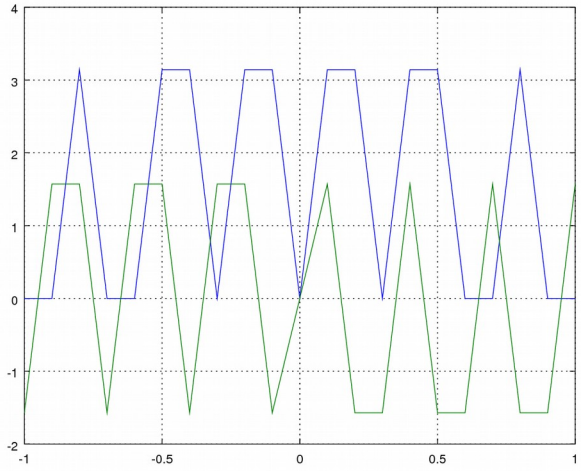
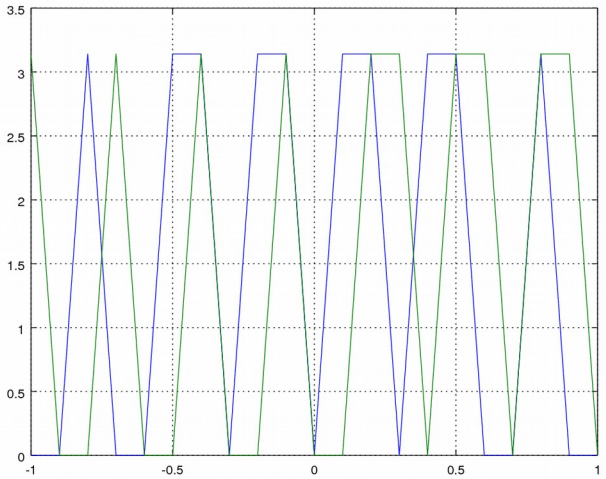
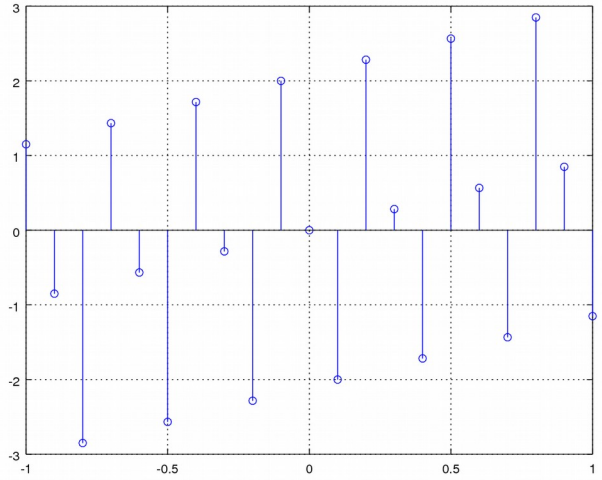
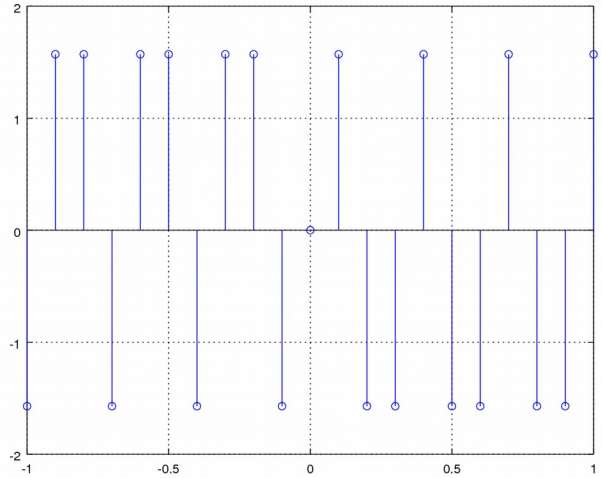
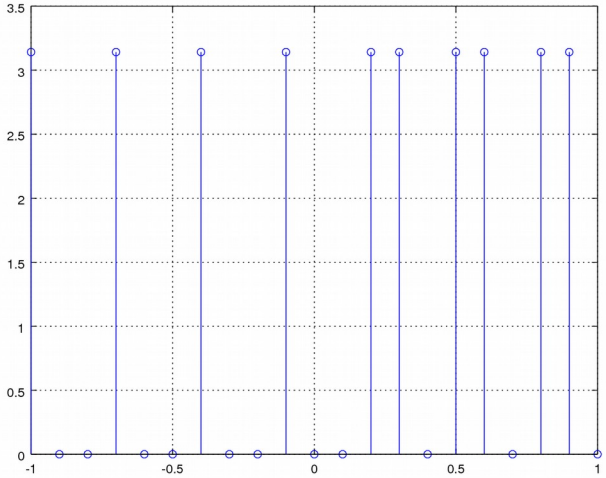



# Plots with a low resolution

t = -10:0.1:10;

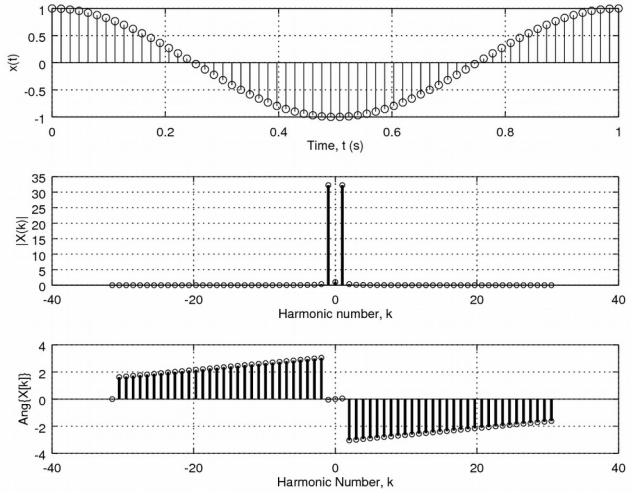
graphical illusions

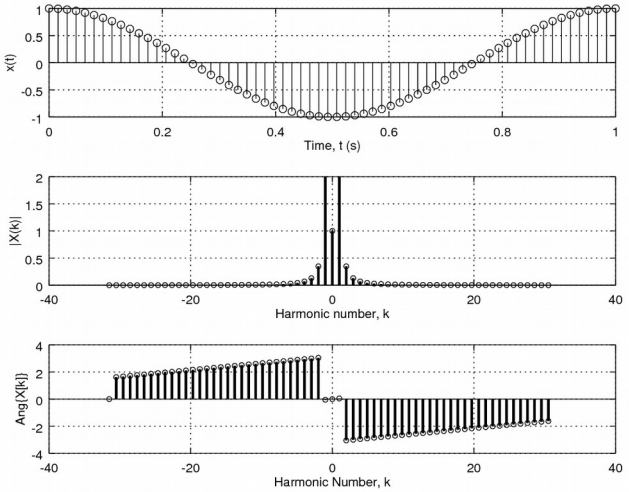







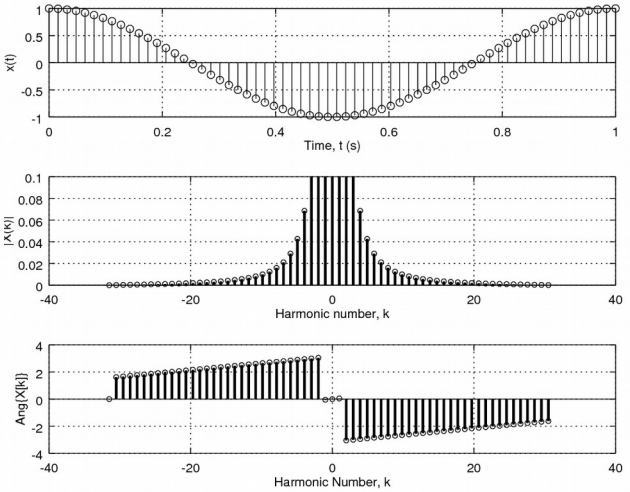
# wavplot

```
fs = 2^6-1;
n = linspace(0, 1, fs+1);
nn = length(n);
k = fs *(0:nn-1)/nn;
x = cos(2*pi*440*n);
X = fft(x);

source "../0.util.octave/util.m"
TimeFreqPlot(n, x, k, X, "full")
```

Small magnitude overrides phase



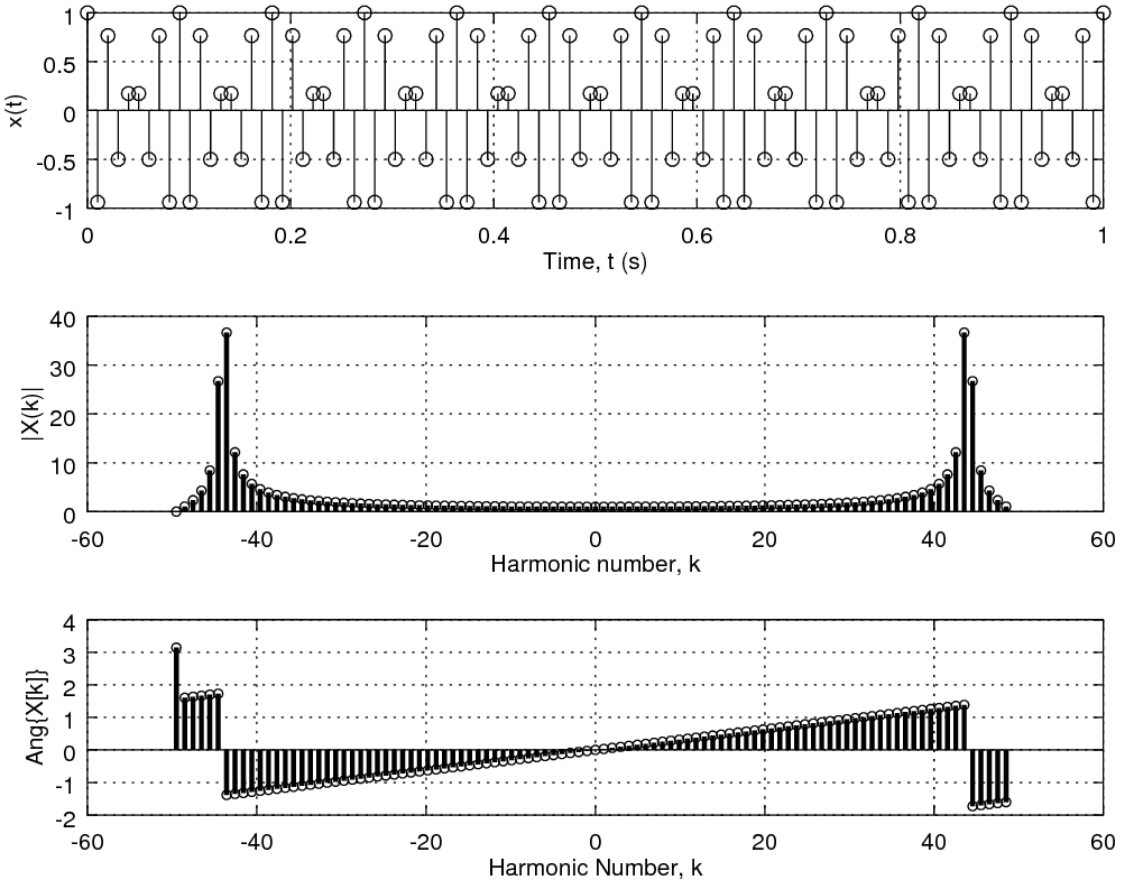



# wavplot

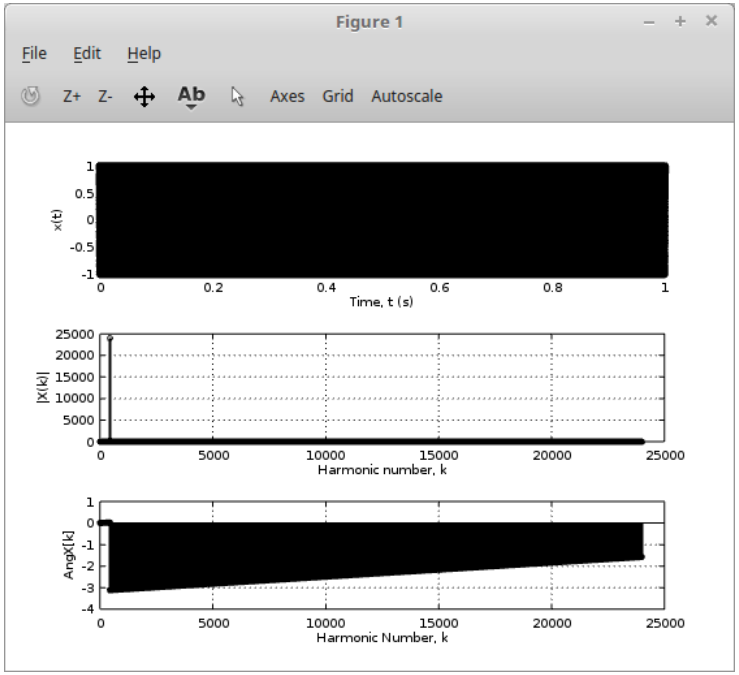
```
fs = 100-1;
n = linspace(0, 1, fs+1);
nn = length(n);
k = fs *(0:nn-1)/nn;
x = cos(2*pi*440*n);
X = fft(x);

source "../0.util.octave/util.m"
TimeFreqPlot(n, x, k, X, "full")
```

Not exact integer multiple cycles

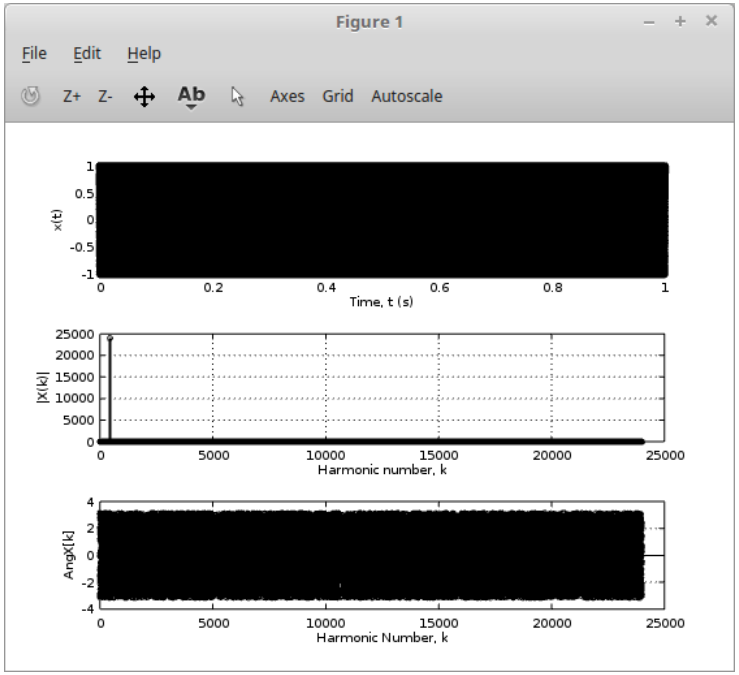


# wavplot

```
fs = 48000;
n = linspace(0, 1, fs+1);
nn = length(n);
k = fs *(0:nn-1)/nn;
x = cos(2*pi*440*n);
X = fft(x);
TimeFreqPlot(n, x, k, X, "half")
```




```
fs = 48000;
n = linspace(0, 1, fs+1);
nn = length(n);
n = n(1:nn-1);
nn = length(n);
k = fs *(0:nn-1)/nn;
x = cos(2*pi*440*n);
X = fft(x);
TimeFreqPlot(n, x, k, X, "half")
```



# Generating signals using sox

| | | | |
|---|---|---|---|
| sox -n s1.wav synth | 3.5 | sine | 440 |
| sox -n s2.wav synth | 90000s | sine | 660:1000 |
| sox -n s3.wav synth | 1:20 | triangle | 440 |
| sox -n s4.wav synth | 1:20 | trapezium | 440 |
| sox -n s5.wav synth | 6 | square | 440 0 0 40 |
| sox -n s6.wav synth | 5 | noise | |

time duration

frequency

-V0, -V1, -V2, -V3, -V4 : verbosity levels

-n (null) : absence of an input signal

# Generating signals using sox

$f_s = 1/48000 = 2.0833\text{e-}05 \ (sample/sec)$

$T_s = 0.0208 \ msec/sample$

| | | | | |
|---|---|---|---|---|
| s1.wav | 3.5 *sec* | $T/T_s = T f_s = 3.5 \cdot 48000 =$ | 168000 *samples* | $N =$ 168000 *pt FFT* |
| s2.wav | 90000 *s* | | 90000 *samples* | $N =$ 90000 *pt FFT* |
| s3.wav | 80 *sec* | $T/T_s = T f_s = 80 \cdot 48000 =$ | 3840000 *samples* | $N =$ 3840000 *pt FFT* |
| s4.wav | 80 *sec* | $T/T_s = T f_s = 80 \cdot 48000 =$ | 3840000 *samples* | $N =$ 3840000 *pt FFT* |
| s5.wav | 6 *sec* | $T/T_s = T f_s = 6 \cdot 48000 =$ | 288000 *samples* | $N =$ 288000 *pt FFT* |
| s6.wav | 5 *sec* | $T/T_s = T f_s = 5 \cdot 48000 =$ | 240000 *samples* | $N =$ 240000 *pt FFT* |

# Generating signals using sox

$f_s = 1/48000 = 2.0833\text{e-}05\ (sample/sec)$

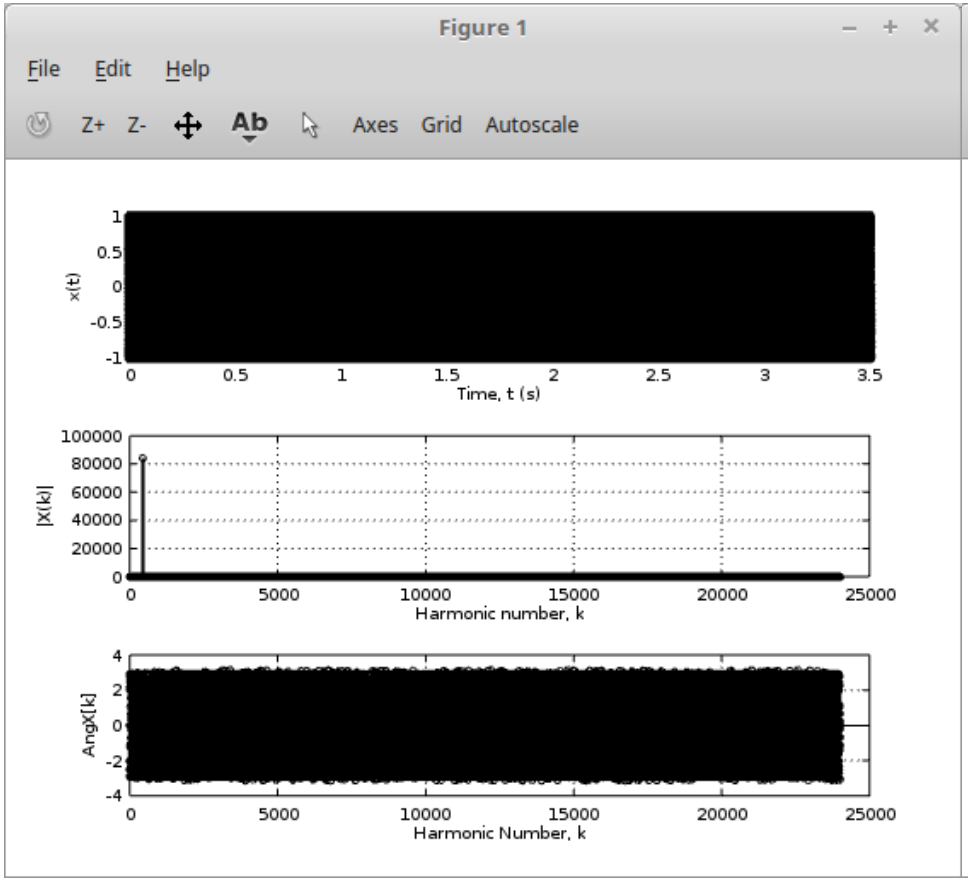
$T_s = 0.0208\ msec/sample$

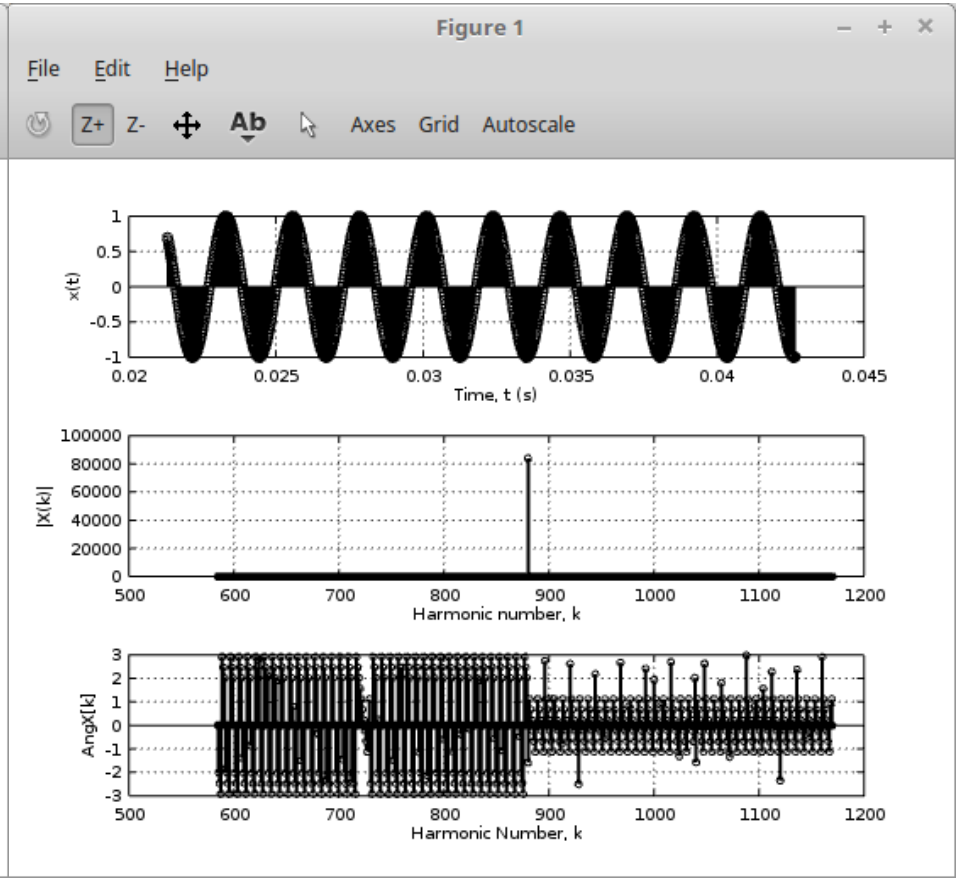
| | | | | |
|---|---|---|---|---|
| s1.wav | $k \cdot f_s / N =$ | $k \cdot 48000/168000$ | $440 \cdot 168000/48000$ | $440 \cdot 3.5 = 1540$ |
| s2.wav | $k \cdot f_s / N =$ | $k \cdot 48000/90000$ | $660 \cdot 90000/48000$ | $660 \cdot 1.875 = 1237.5$ |
| | | | $1000 \cdot 90000/48000$ | $1000 \cdot 1.875 = 1875$ |
| s3.wav | $k \cdot f_s / N =$ | $k \cdot 48000/3840000$ | $440 \cdot 3840000/48000$ | $440 \cdot 80 = 35200$ |
| s4.wav | $k \cdot f_s / N =$ | $k \cdot 48000/3840000$ | $440 \cdot 3840000/48000$ | $440 \cdot 80 = 35200$ |
| s5.wav | $k \cdot f_s / N =$ | $k \cdot 48000/288000$ | $440 \cdot 288000/48000$ | $440 \cdot 6 = 2640$ |
| s6.wav | $k \cdot f_s / N =$ | $k \cdot 48000/240000$ | $440 \cdot 240000/48000$ | $440 \cdot 5 = 2200$ |

# sox -n **s1.wav** synth 3.5 sine 440

$440 \cdot 3.5 = 1540$

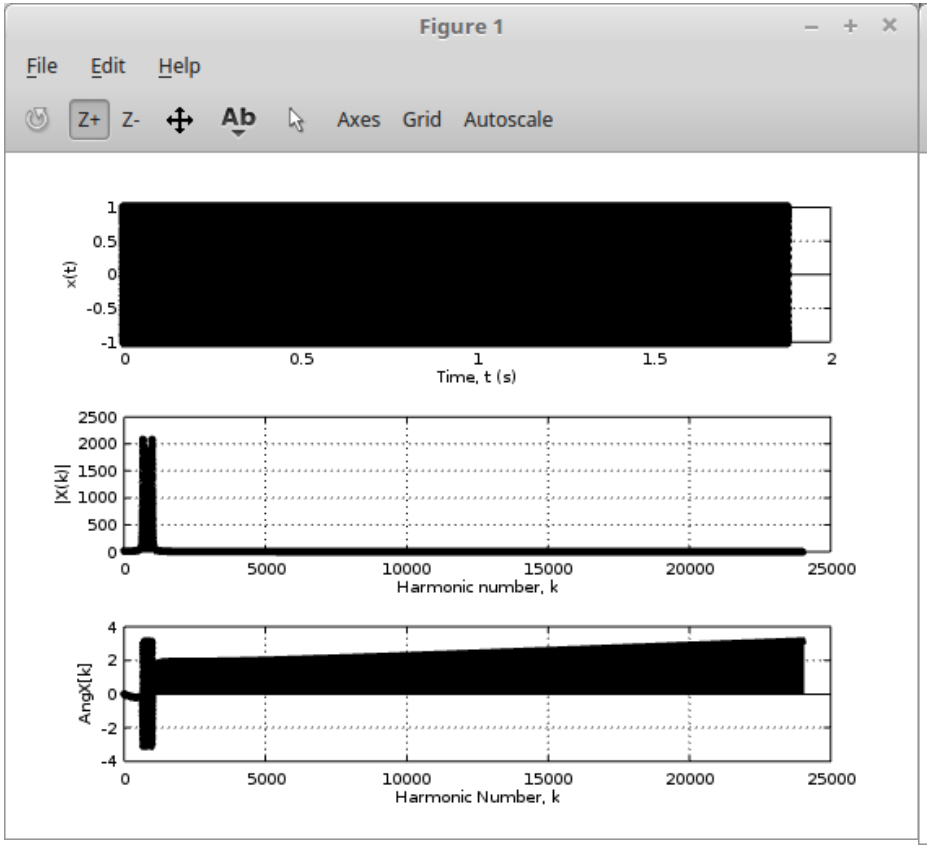
$N = 168000\ pt\ \ FFT$

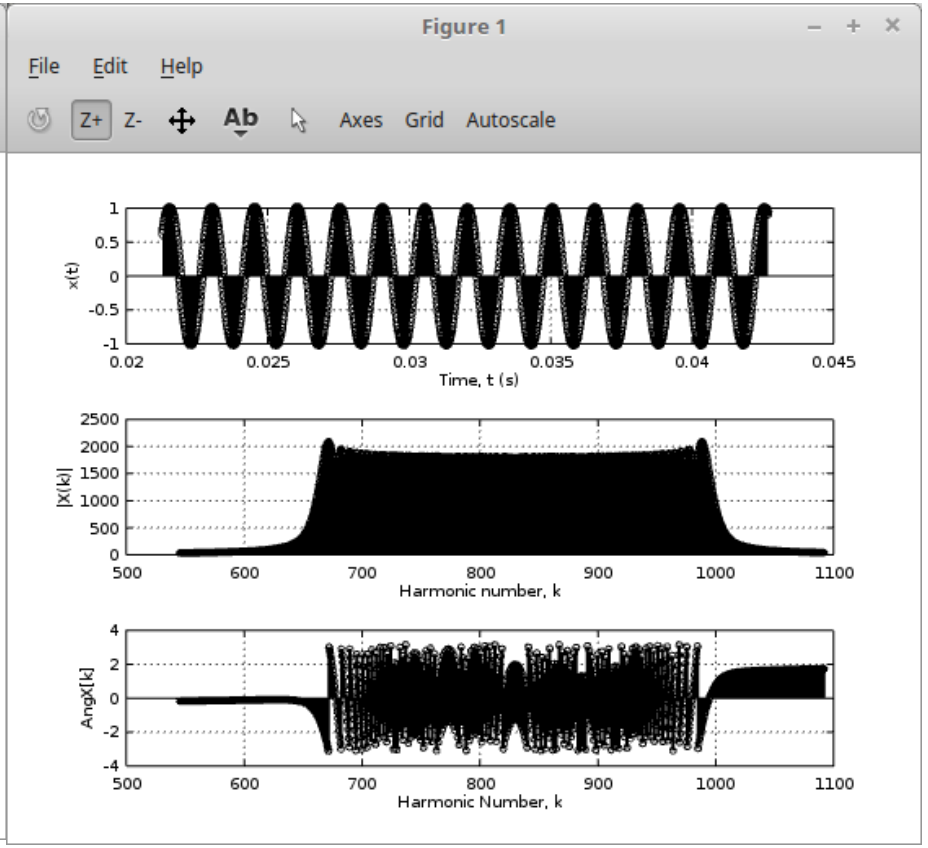


# sox -n **s2.wav** synth 90000s sine 660:1000

$660 \cdot 1.875 = 1237.5$ $1000 \cdot 1.875 = 1875$ $N = 90000\ pt\ FFT$

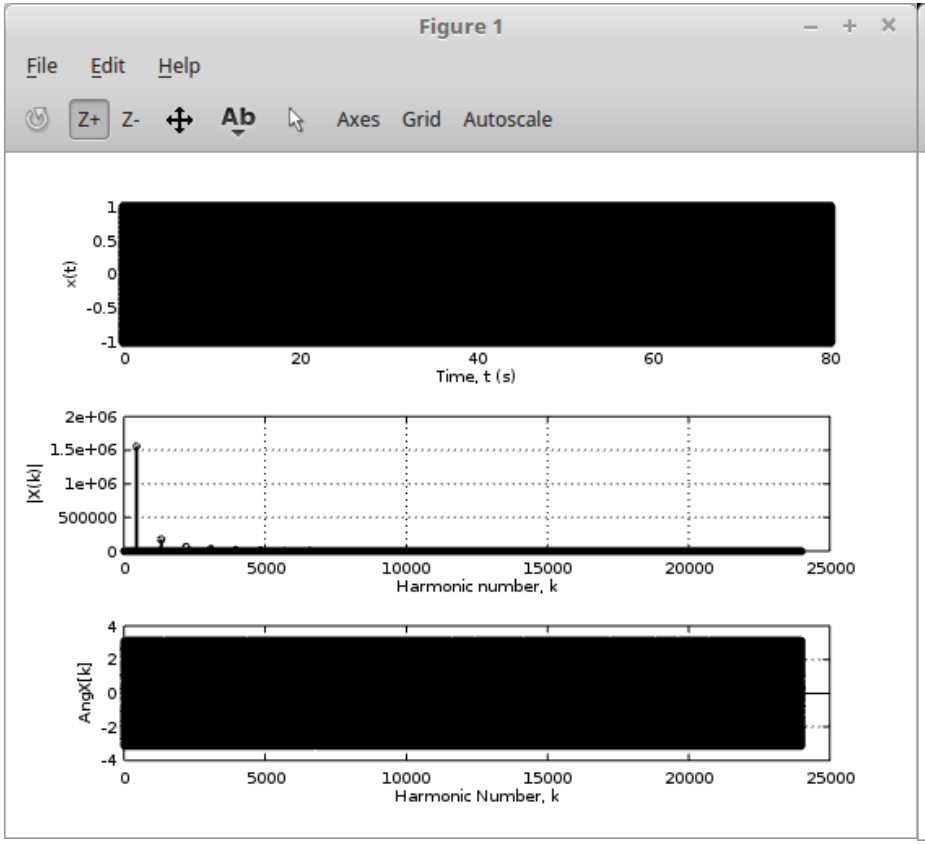


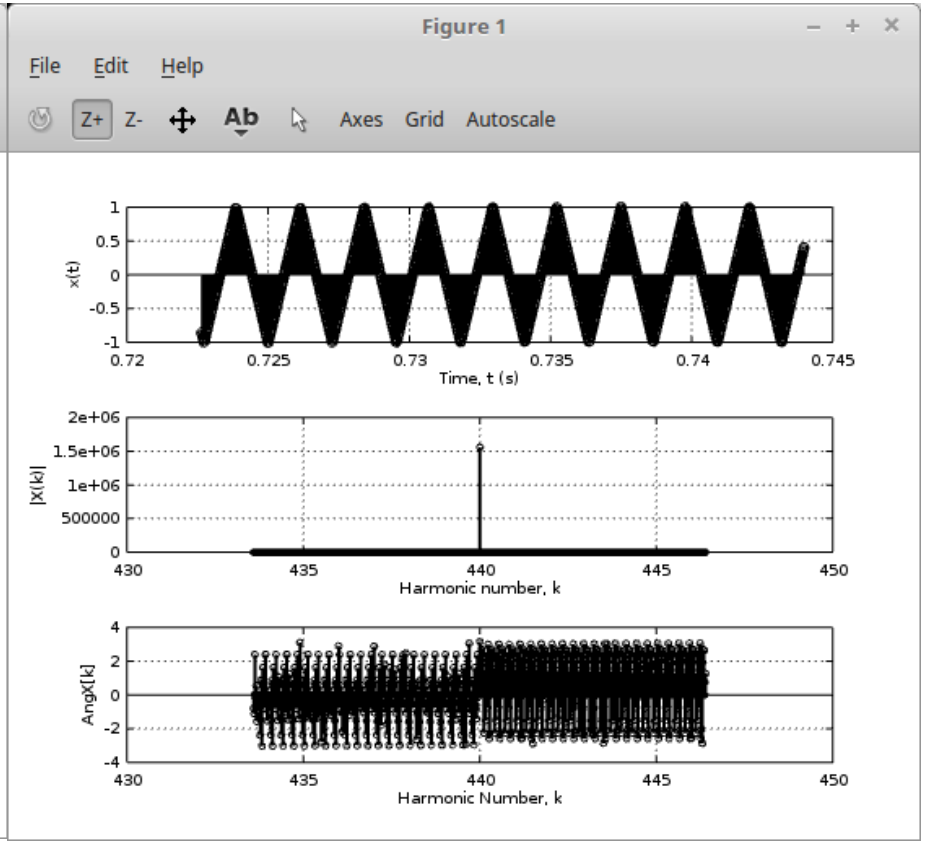



# sox -n **s3.wav** synth 1:20 triangle 440

$440 \cdot 80 = 35200$

$N = 3840000\ pt\ \ FFT$

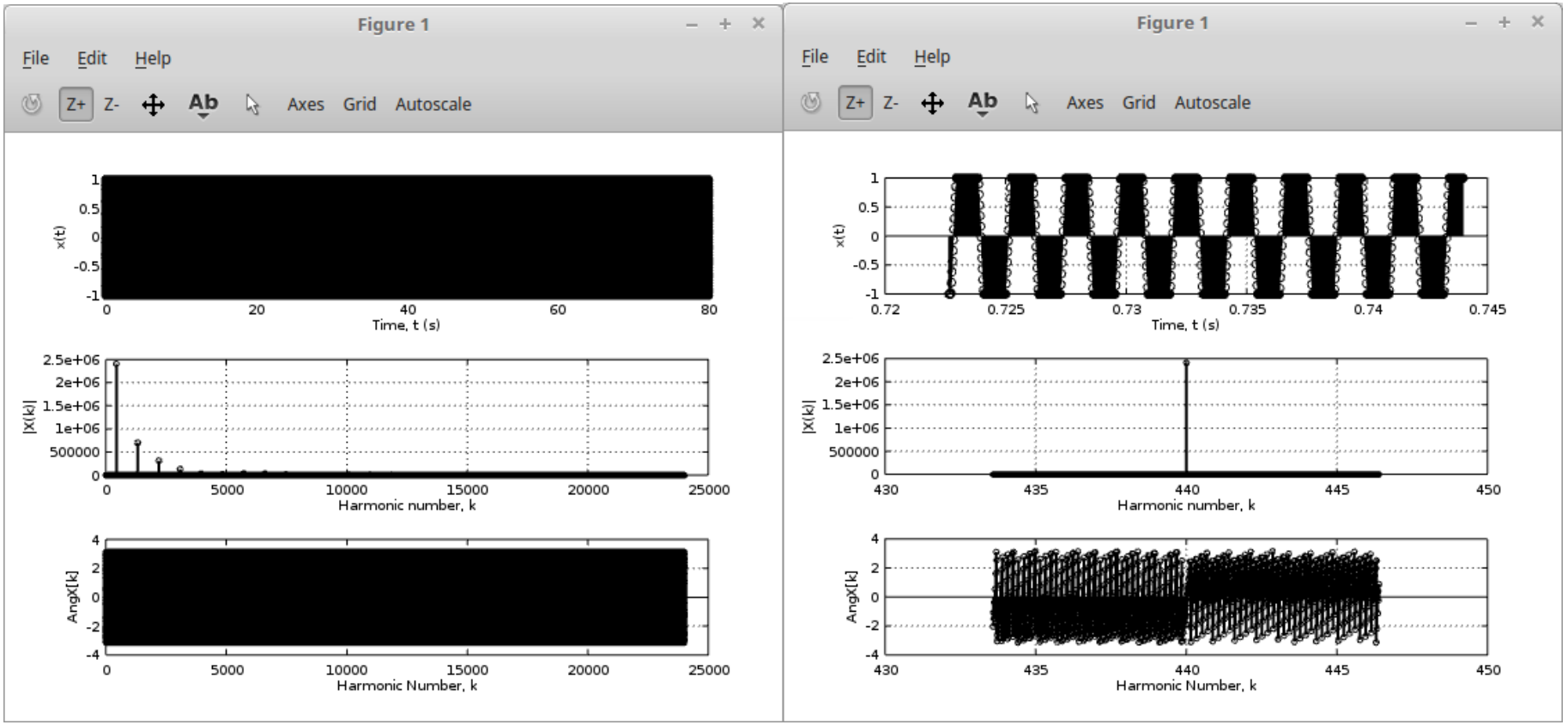


# sox -n **s4.wav** synth 1:20 trapezium 440

$440 \cdot 80 = 35200$

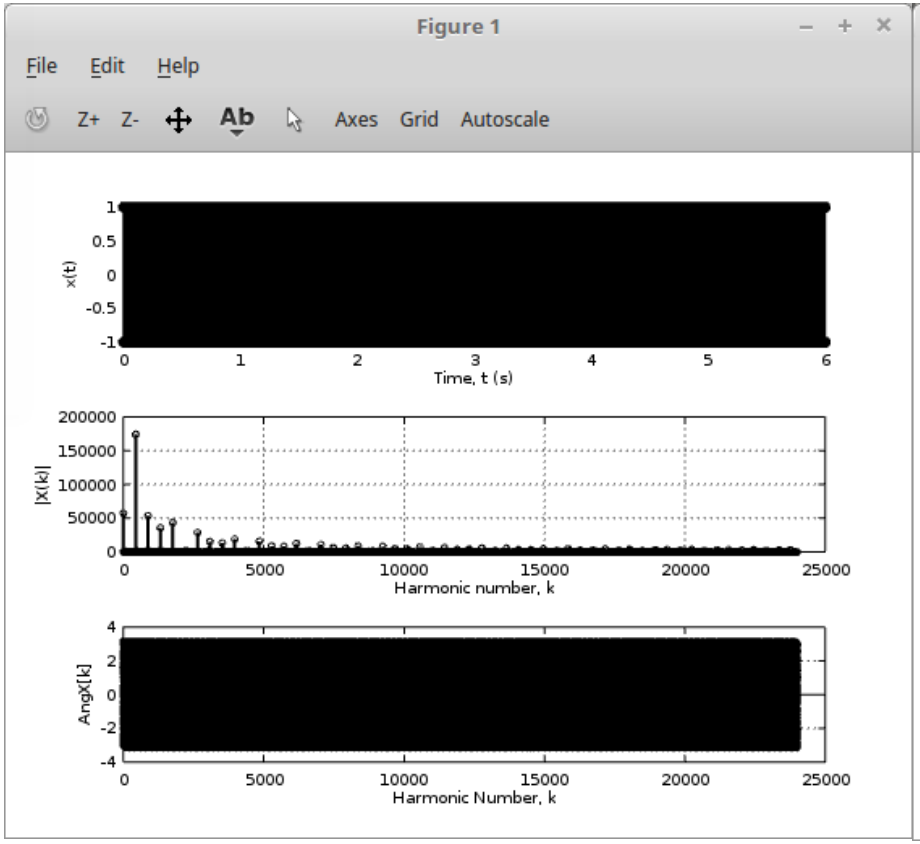
$N = 3840000\ pt\ \ FFT$

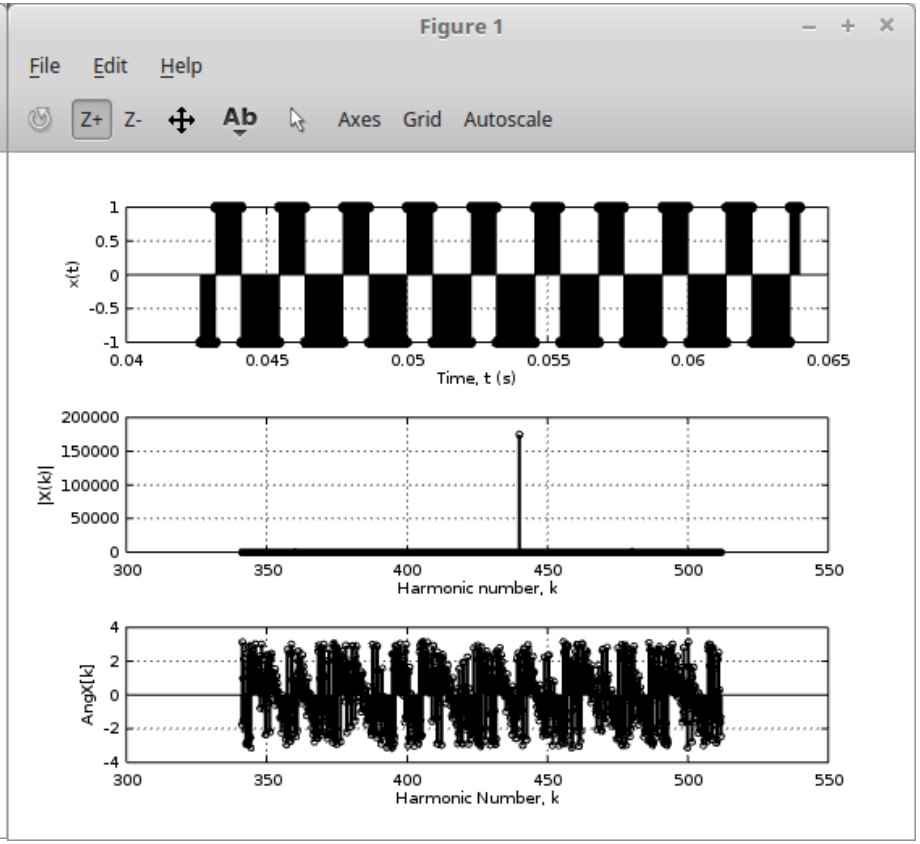


# sox -n **s5.wav** synth 6 square 440 0 0 40

$440 \cdot 6 = 2640$

$N = 288000\ pt\ FFT$

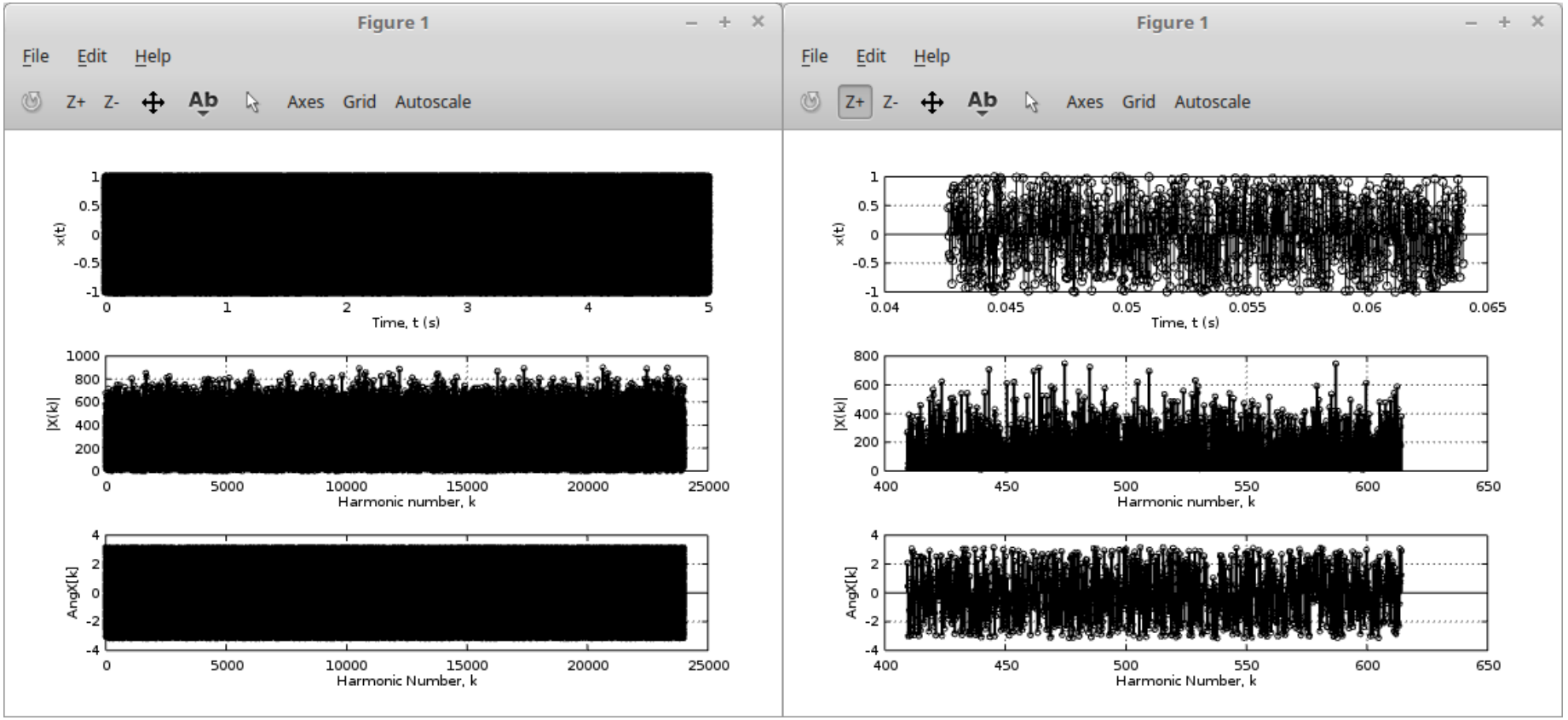


# sox -n **s6.wav** synth 5 noise

$440 \cdot 5 = 2200$

$N = 240000\ pt\ FFT$



## References

[1] F. Auger, Signal Processing with Free Software : Practical Experiments