PRÁCTICA 3

TRANSFORMADA DISCRETA DE FOURIER

Teoría de comunicaciones y señales

Profesor: Gutiérrez Aldana Eduardo

Alumno:

Amador Nava Miguel Ángel

Grupo: 3CM8

Índice

Introducción

Se implementará la Transformada Discreta de Fourier (DFT) en lenguaje C, con la cual se pasará del dominio del tiempo a la frecuencia, haciendo el análisis de esta.

Objetivos

Generales: Implementación de la DFT

Específicos:

- Identificar el efecto alias.
- Comprobar el teorema del muestreo.
- Identificar la frecuencia de una señal dado la duración de la misma y su frecuencia de muestreo.

Software y equipo utilizado

- Ubuntu 17.10
- Wine 1.7
- GoldWave 4.26
- Sublime Text 3

Marco teórico

Teorema de Nyquist-Shannon

El teorema de muestreo de Nyquist-Shannon establece que la frecuencia mínima a la que se debe muestrear una señal debe ser al menos el doble de la frecuencia más grande de la señal.

Aliasing

Es el efecto que causa que señales continuas distintas se tornen indistinguibles cuando se muestrean digitalmente**.**

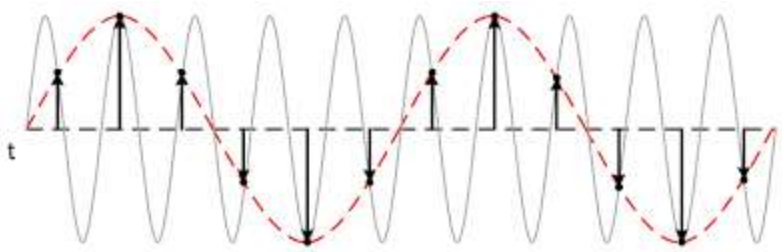

Ilustración 1 Efecto Alias

Este efecto se evita cumpliendo el teorema de Nyquist o también llamado teorema de muestreo.

Transformada discreta de Fourier

La transformada discreta de Fourier o DFT (del inglés, discrete Fourier transform) es un tipo de transformada discreta utilizada en el análisis de Fourier. Transforma una función matemática en otra, obteniendo una representación en el dominio de la frecuencia, siendo la función original una función en el dominio del tiempo. Pero la DFT requiere que la función de entrada sea una secuencia discreta y de duración finita. Dichas secuencias se suelen generar a partir del muestreo de una función continua, como puede ser la voz humana**.**

$$\boldsymbol{X_k = \sum_{n=0}^{N-1} X_n e^{-\frac{2\pi i}{N} kn} \quad k = 0, \dots, N-1}$$

Ecuación 1

En la ecuación 1 la potencia de una frecuencia w, se puede evaluar en un fragmento de señal de longitud L, de igual manera se puede notar que lo valores de X(jw), son números complejos.

$$\boldsymbol{X_k = \sum_{n=0}^{N-1} X_n \left[cos\left(\frac{2\pi}{N} kn\right) - jsen\left(\frac{2\pi}{N} kn\right)\right]}$$

Ecuación 2

Preparación del entorno para el desarrollo de la práctica

GoldWave 4.26 fue creado para Windows por lo cual para poder ejecutarlos en Ubuntu 17.10 necesitaremos instalar wine.

Introduciremos los siguientes comandos:

sudo add-apt-repository ppa:ubuntu-wine/ppa

sudo apt-get update

sudo apt-get install wine1.7 winetricks

Una vez instalado wine para ejecutar GoldWave lo haremos de la siguiente manera:

WINEPREFIX~=/.wine32 wine "nombre del ejecutable"

Planteamiento del problema

Se pide realizar la Transformada Discreta de Fourier y guardar la parte real en el canal 1 y la parte imaginaria en el canal 2.

Para probar el programa se crearan los siguientes archivos:

1. Cosenos
   - Archivos monoaurales de 2000 muestras por segundo de 0.016s de duración.
   - $Cos(2 * \Pi * \mathrm{f} * \mathrm{t} * \mathrm{n})$
   - $f = 62.5$
   - $n = 0,1,2, \dots ,32$

2. Senos
   - Archivos monoaurales de 2000 muestras por segundo de 0.016s de duración.
   - $Sin(2 * \Pi * \mathrm{f} * \mathrm{t} * \mathrm{n})$
   - $f = 62.5$
   - $n = 0,1,2, \dots ,32$

3. Suma de senos y cosenos
   - Archivo monoaural de 2000 muestras por segundo de 0.016s de duración.
   - $(Cos(2 * \Pi * \mathrm{f} * \mathrm{t} * 2) + Sin(2 * \Pi * \mathrm{f} * \mathrm{t} * 4))/2$
   - $f = 62.5$

Análisis Teórico

Cabecera de un archivo wav

Bytes	Campo	Descripción
4	ChunkID	Contiene las letras “RIFF” en ASCII
4	ChunkSize	36 bytes más SubChunk2Size. Este es el tamaño entero del archivo en bytes menos 8
4	Format	Contiene las letras “WAVE”
4	Subchunk1ID	Contiene las letras “fmt”
4	Subchunk1Size	Este es el tamaño del resto del Subchunk
2	AudioFormat	Los valores distintos de 1 indican alguna forma de compresión.
2	NumChannels	Numero d canales: 1= mono, 2=Stereo
4	SampleRate	Número de muestras por segundo
4	ByteRate	SampleRate * NumChannels * BitsPerSample/8
2	BlockAlign	NumChannels * BitsPerSample/8. El número de bytes para una muestra incluyen todos los canales
2	BitsPerSample	Bits por muesta. 8,16,etc.
4	Subchunk2ID	Contiene las letras “data”
4	Subchunk2Size	NumSamples * NumChannels * BitsPerSample/8. Este es el número de bytes en la data
*	Data	Datos del sonido

Tabla 1 Información de la cabecera de un archivo wav

Transformada Discreta de Fourier

Dependiendo del autor donde se consultó la DFT también podemos encontrarla definida por:

$$X_k = \frac{1}{N}\sum_{n=0}^{N-1} X_n e^{-\frac{2\pi i}{N}kn} \quad k = 0, \dots, N-1$$

Ecuación 3 DFT con cálculo del promedio

$$X_k = \frac{1}{\sqrt{N}}\sum_{n=0}^{N-1} X_n e^{-\frac{2\pi i}{N}kn} \quad k = 0, \dots, N-1$$

Ecuación 4 DFT con cálculo de la raíz del promedio

Por simplicidad usaremos la ecuación número 3, pues recordemos que estamos limitados al tamaño de un short, y si no sacamos el promedio de la DFT, se desbordaran nuestros datos y para que no suceda esto podemos hacer una regla de tres en la cual nuestro valor máximo de entrada será proporcional a nuestro valor máximo de salida, pero se requerirían más calculos por lo cuál escogímos la ecuación número tres para esta práctica.

Como se pide que en el canal 1 se envíen los valores reales y en el canal 2 los valores imaginarios, de la ecuación 2 deducimos:

Canal 1:

$$X_k = \frac{1}{N}\sum_{n=0}^{N-1} X_n cos\left(\frac{2\pi}{N} kn\right) \quad k = 0, \dots, N-1$$

Canal 2:

$$X_k = -\frac{1}{N}\sum_{n=0}^{N-1} X_n sen\left(\frac{2\pi}{N} kn\right) \quad k = 0, \dots, N-1$$

```
//DFT
for(int k=0;k<numMuestrasAudio;k++){
	muestrasAudio2[2*k]=0;
	muestrasAudio2[2*k+1]=0;
	for(int n=0;n<numMuestrasAudio;n++){

	muestrasAudio2[2*k]+=muestrasAudio[n]*cos(2*PI*k*n/numMuestrasAudio);
	}
	muestrasAudio2[2*k]/=numMuestrasAudio;
	for(int n=0;n<numMuestrasAudio;n++){
	muestrasAudio2[2*k+1]+=muestrasAudio[n]*sin(2*PI*k*n/numMuestrasAudio);
	}
	muestrasAudio2[2*k+1]/=numMuestrasAudio;
	muestrasAudio2[2*k+1]*=-1;
}
```

Resultados

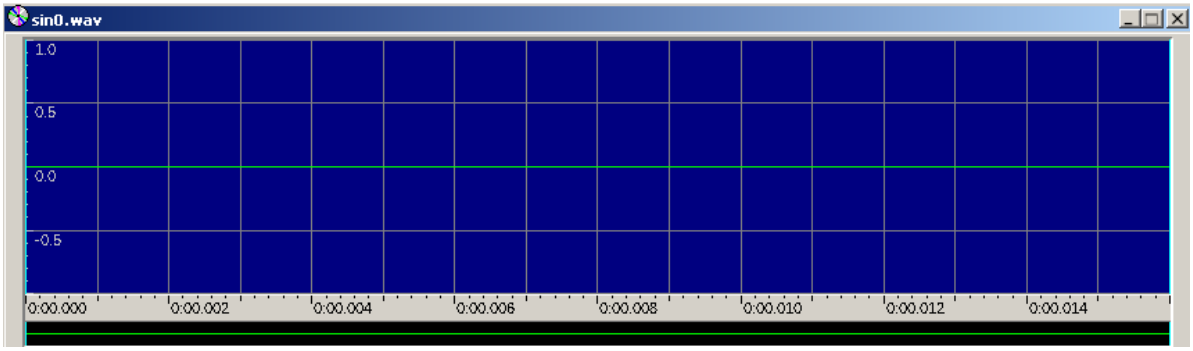

Ilustración 2 Seno de 0 Hz

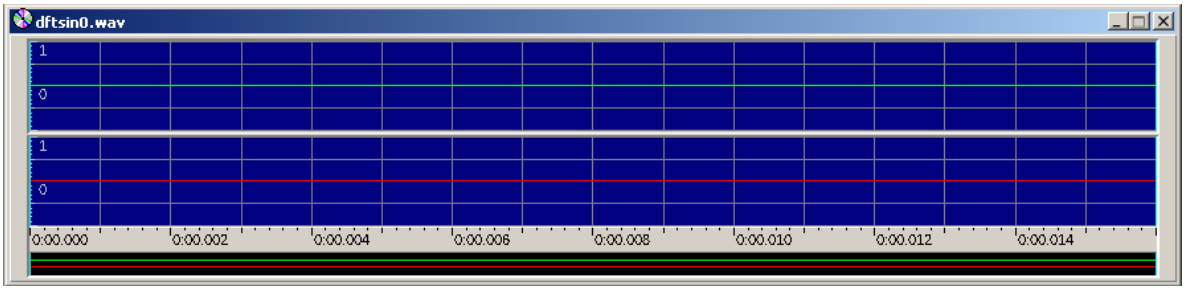

Ilustración 3 DFT del seno de 0 Hz

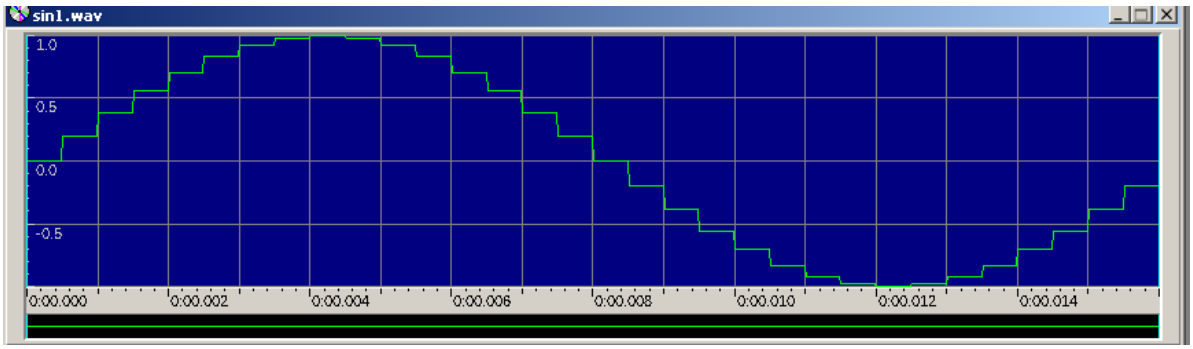

Ilustración 4 Seno de 62.5 Hz

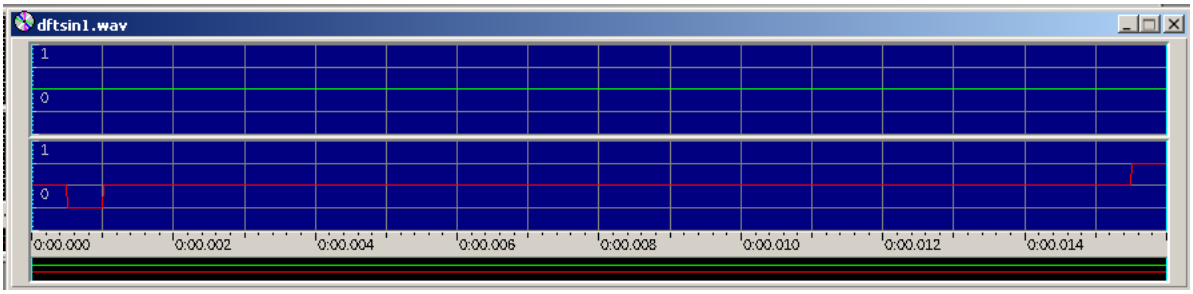

Ilustración 5 DFT del Seno de 62.5 Hz

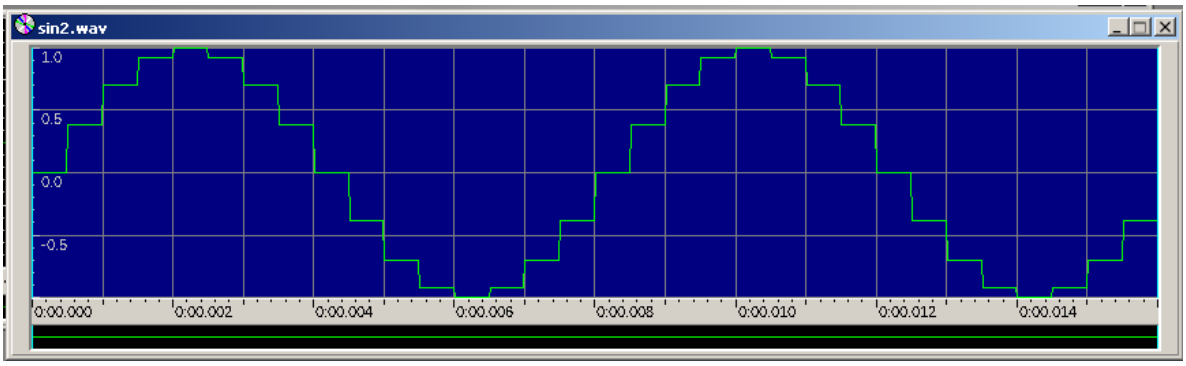

Ilustración 6 Seno de 125 Hz

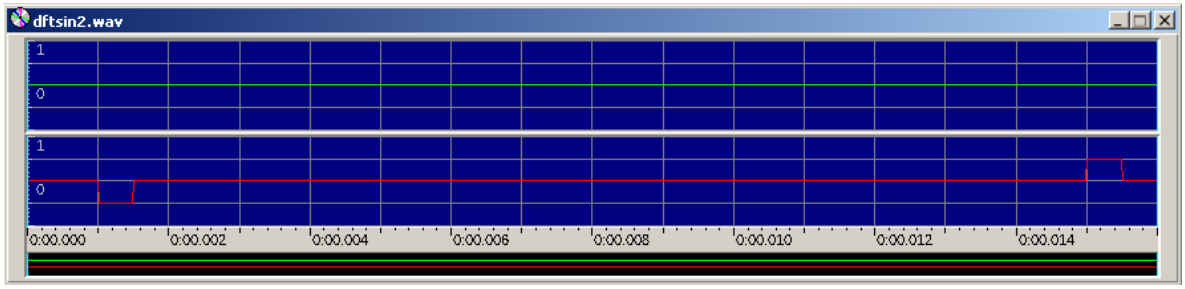

Ilustración 7 DFT del seno de 125 Hz

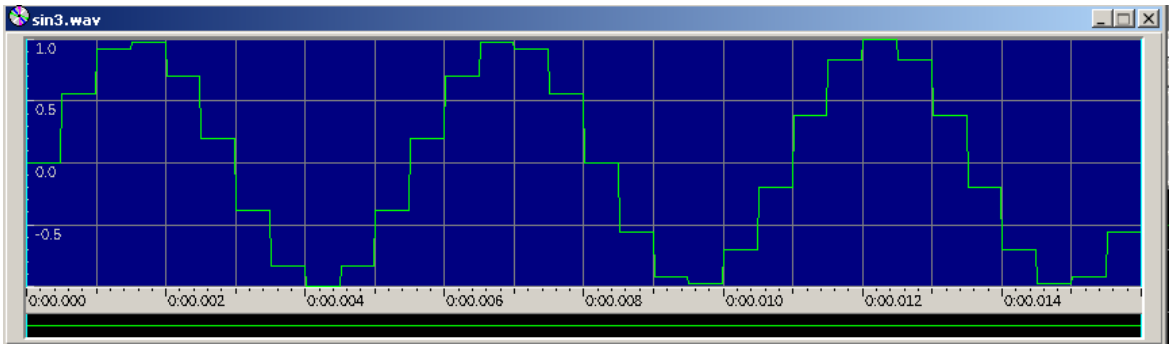

Ilustración 8 Seno de 187.5 Hz

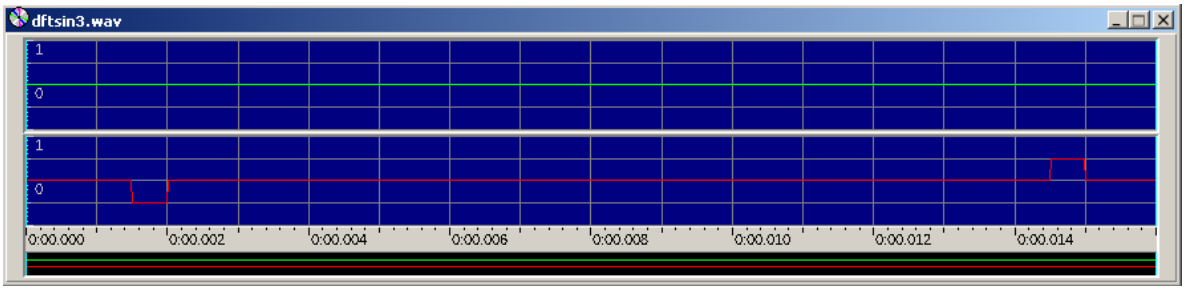

Ilustración 9 DFT del Seno de 187.5 Hz

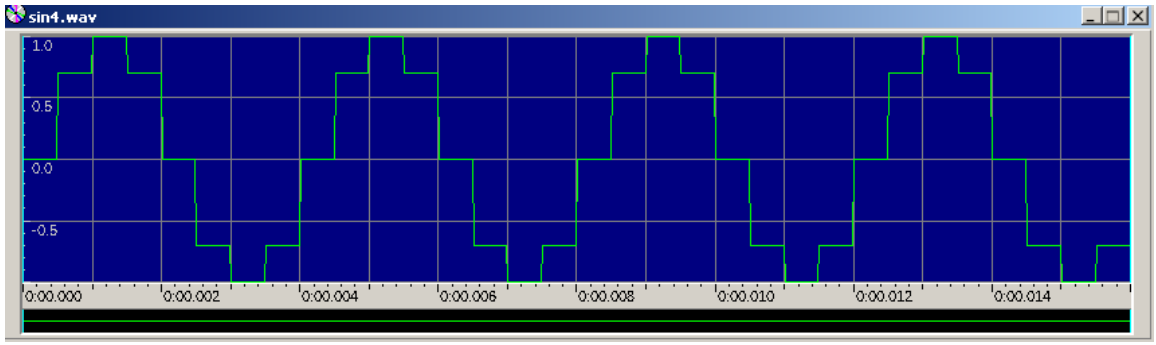

Ilustración 10 Seno de 250 Hz

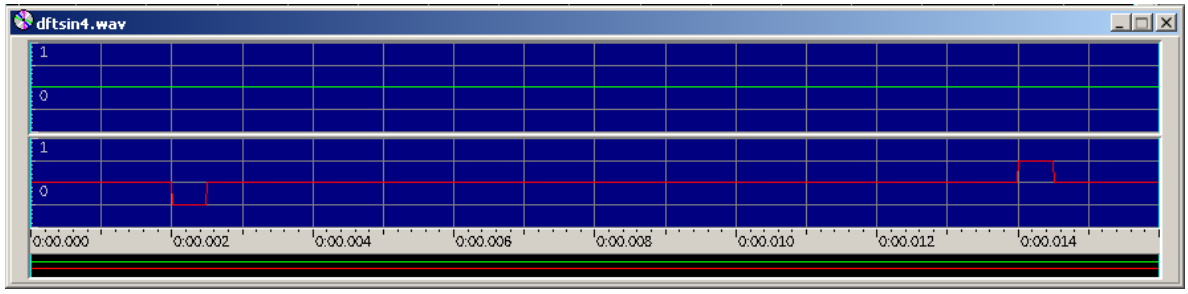

Ilustración 11 DFT del Seno de 250Hz

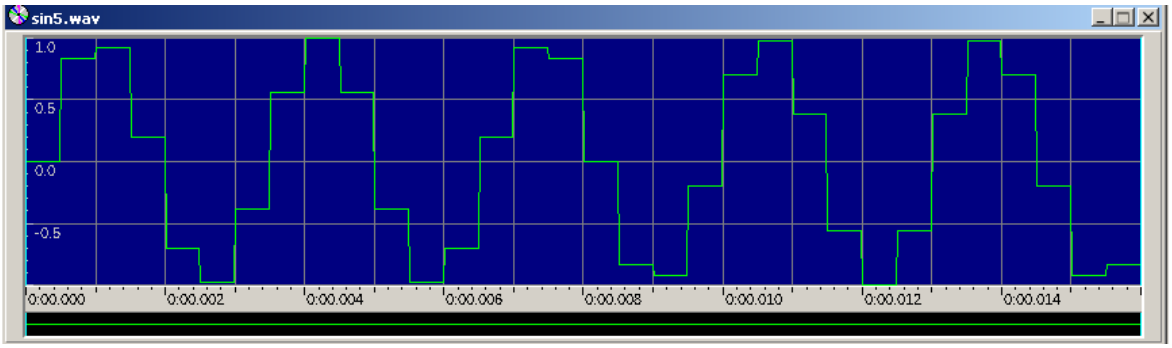

Ilustración 12 Seno de 312.5 Hz

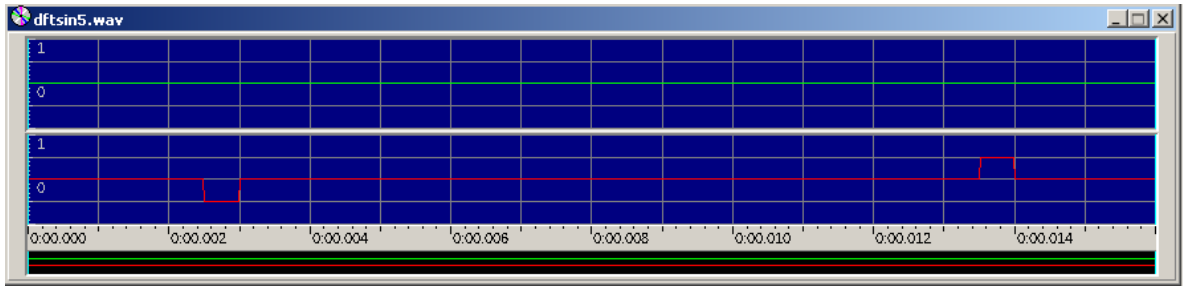

Ilustración 13 DFT del Seno 312.5 Hz

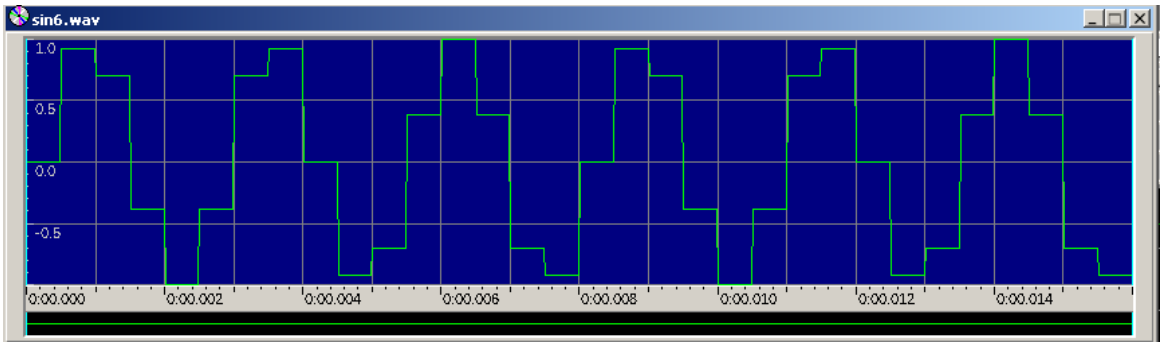

Ilustración 14 Seno de 375 Hz

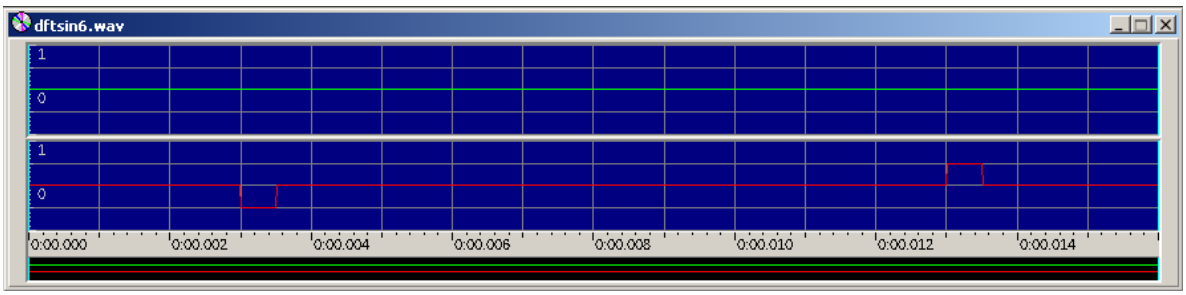

Ilustración 15 DFT del Seno de 375 Hz

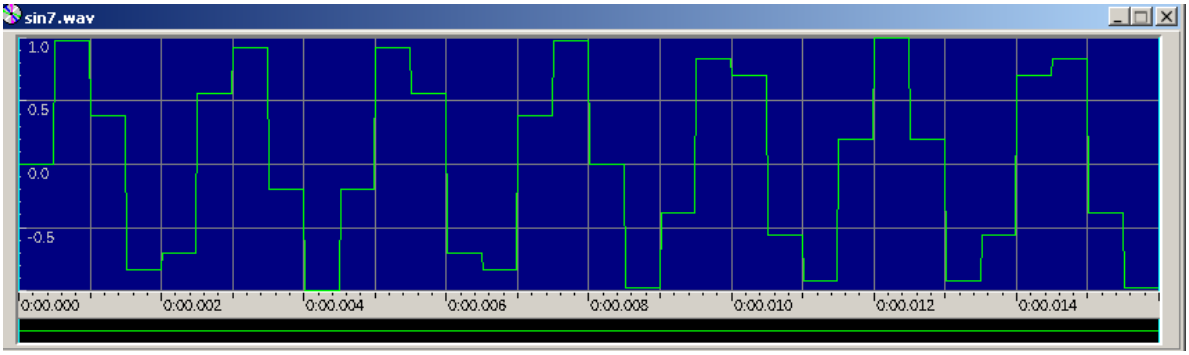

Ilustración 16 Seno de 437.5 Hz

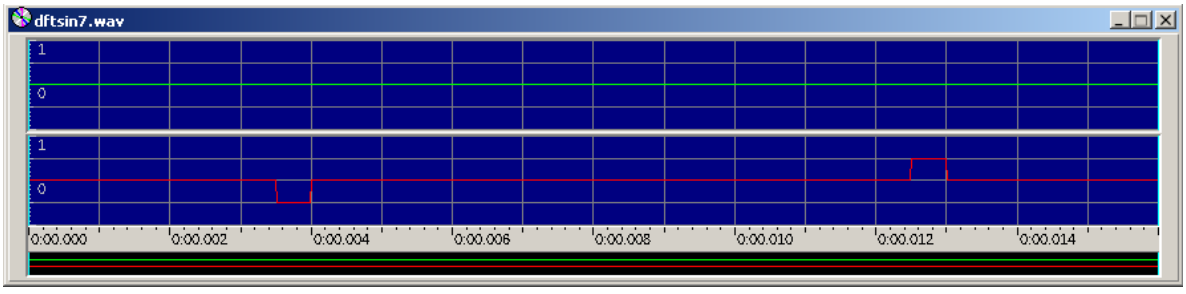

Ilustración 17 DFT del Seno de 437.5 Hz

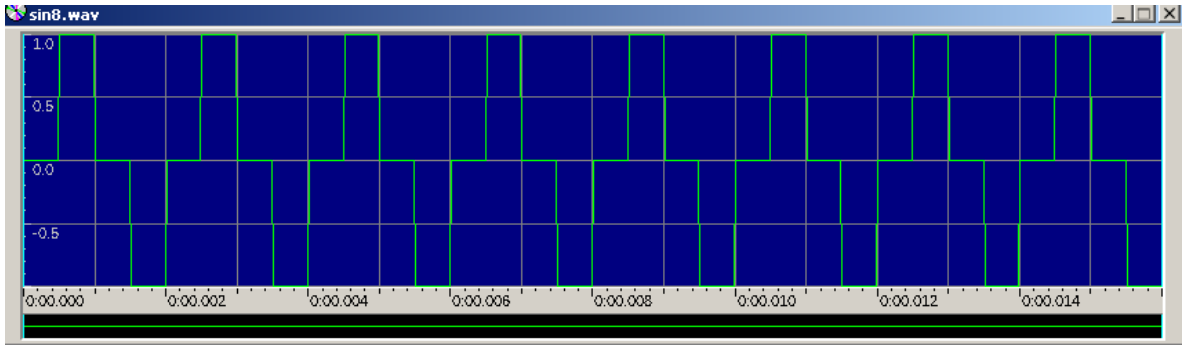

Ilustración 18 Seno de 500 Hz

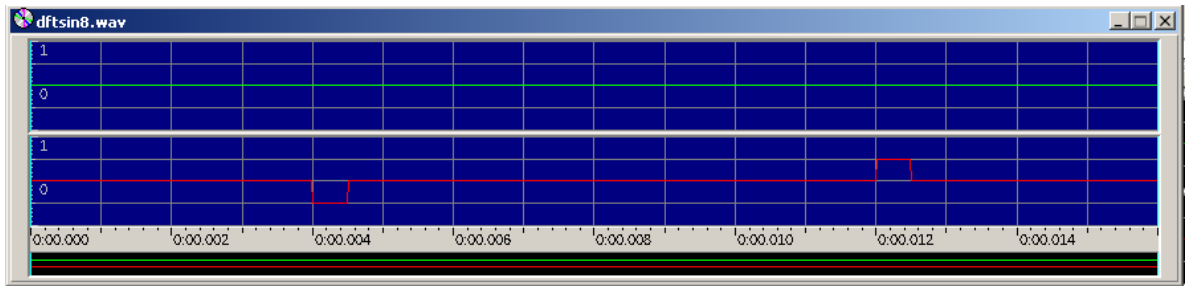

Ilustración 19 DFT del Seno de 500 Hz

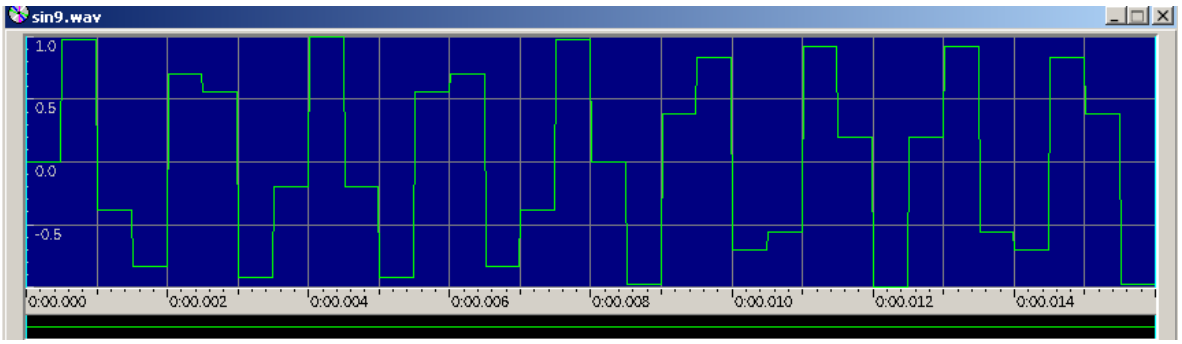

Ilustración 20 seno de 562.5 Hz

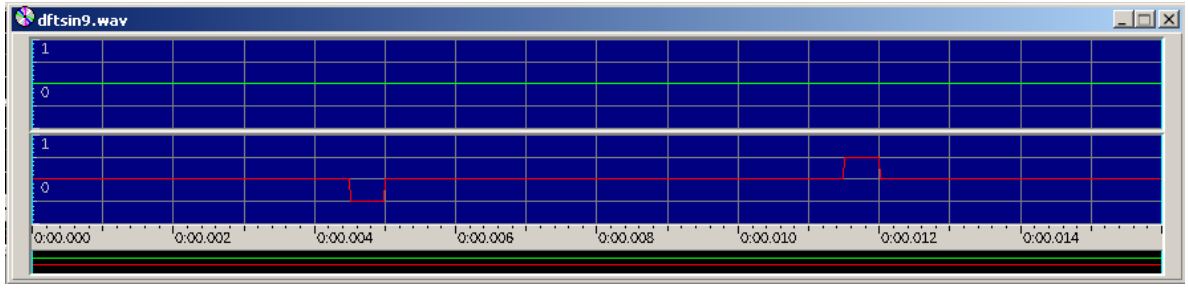

Ilustración 21 DFT del Seno de 562.5 Hz

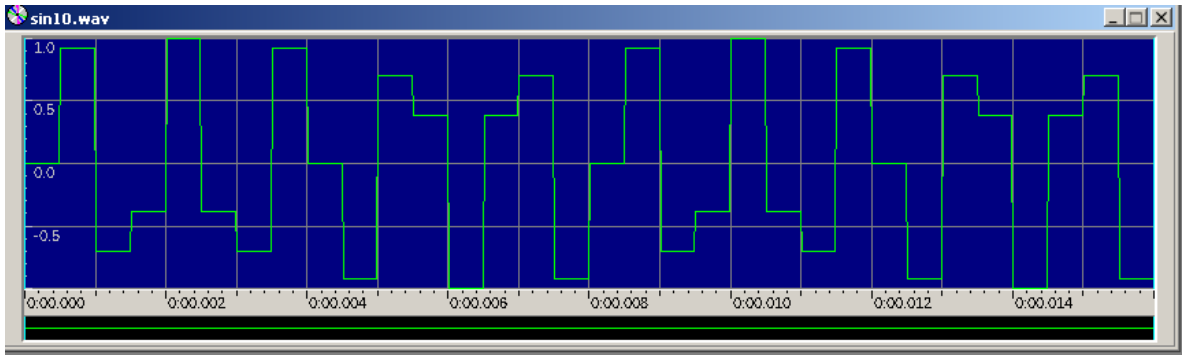

Ilustración 22 Seno de 650 Hz

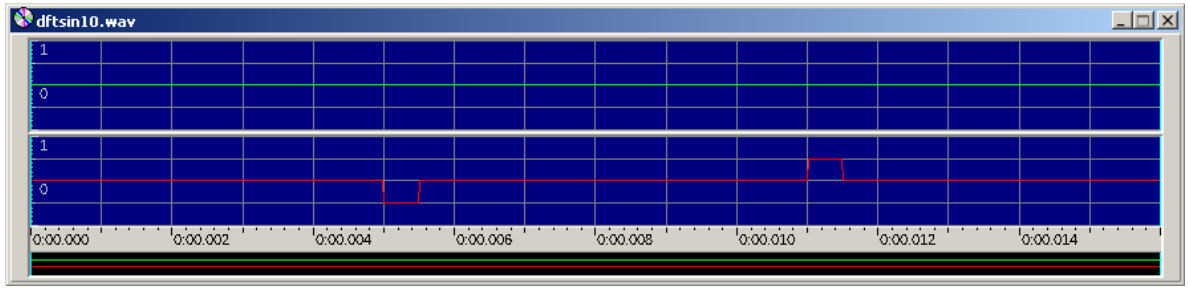

Ilustración 23 DFT del Seno de 650 Hz

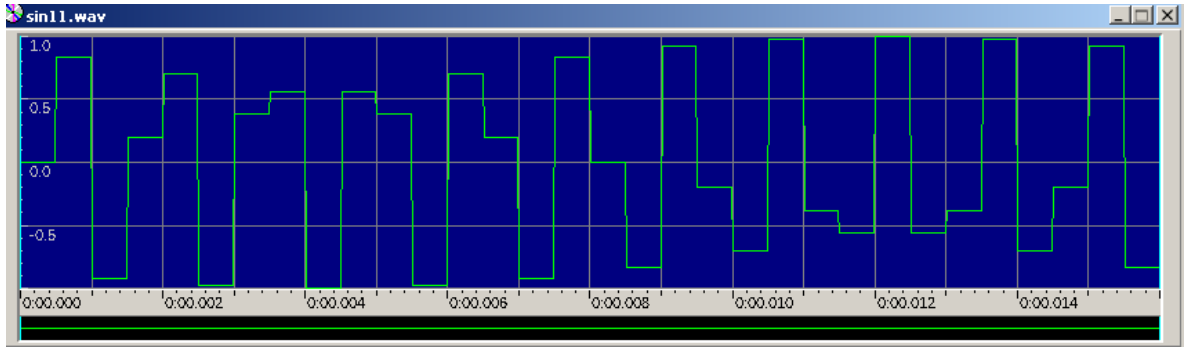

Ilustración 24 Seno de 687.5 Hz

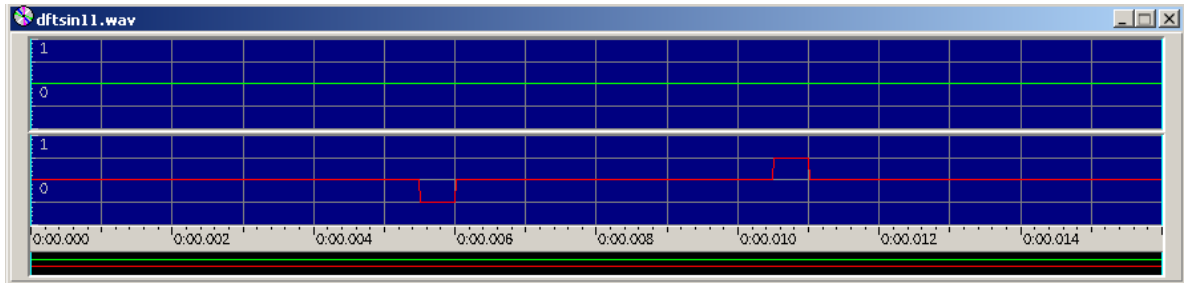

Ilustración 25 DFT del Seno de 687.5 Hz

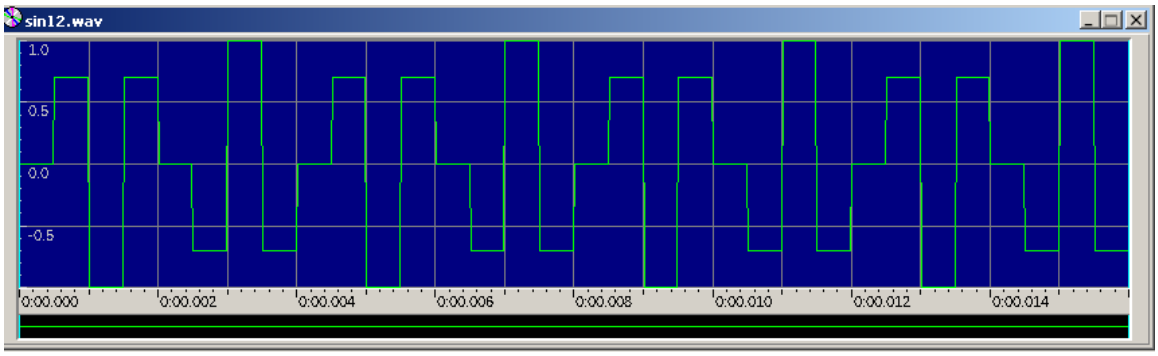

Ilustración 26 Seno de 750 Hz

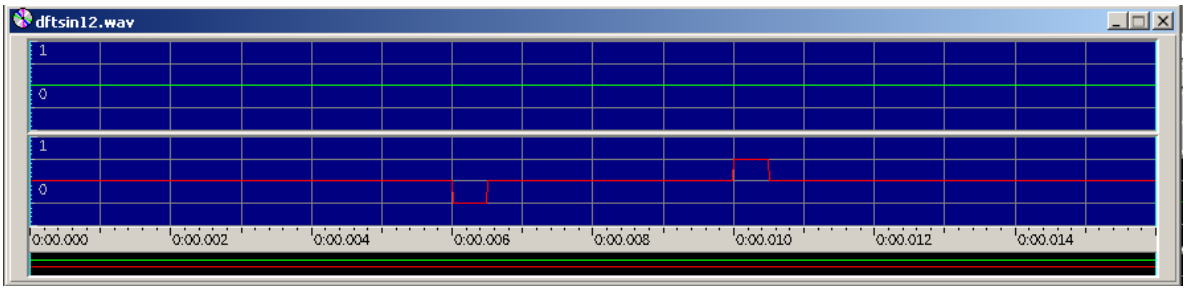

Ilustración 27 DFT del Seno de 750 Hz

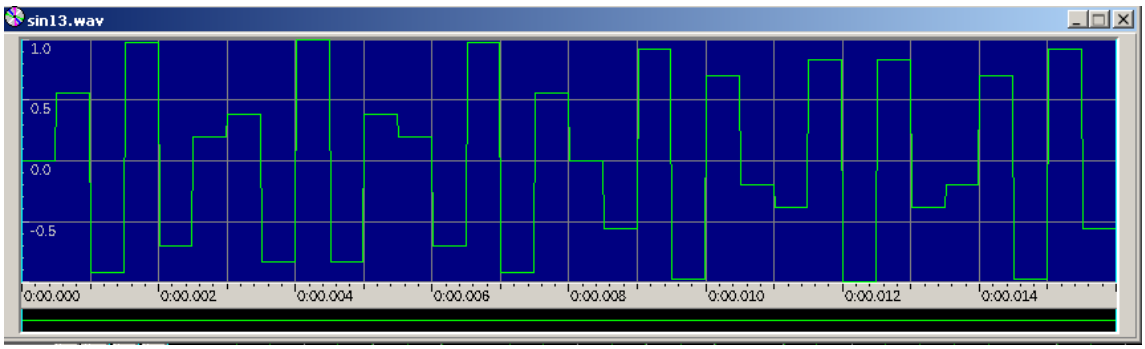

Ilustración 28 Seno de 812.5 Hz

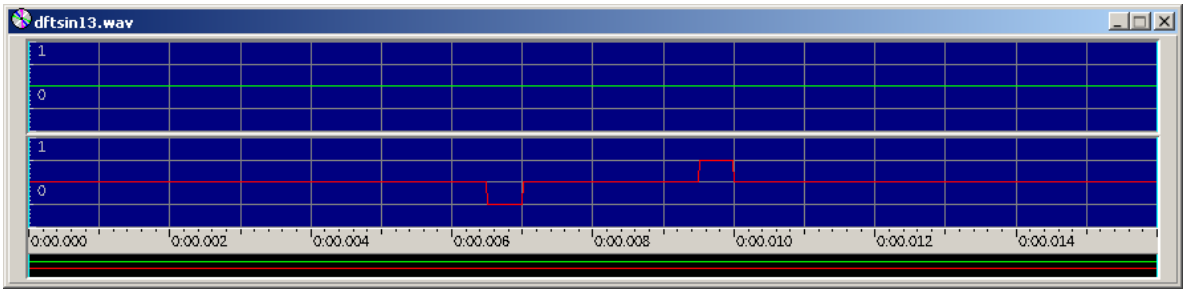

Ilustración 29 DFT del Seno de 812.5 Hz

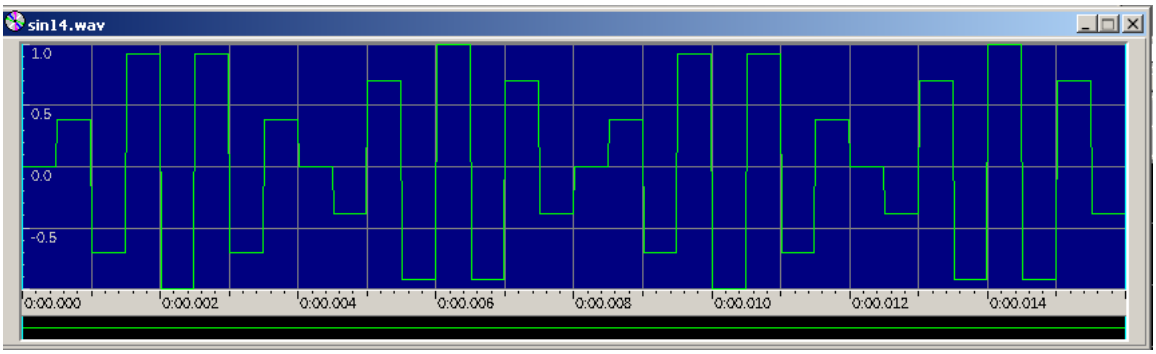

Ilustración 30 Seno de 875 Hz

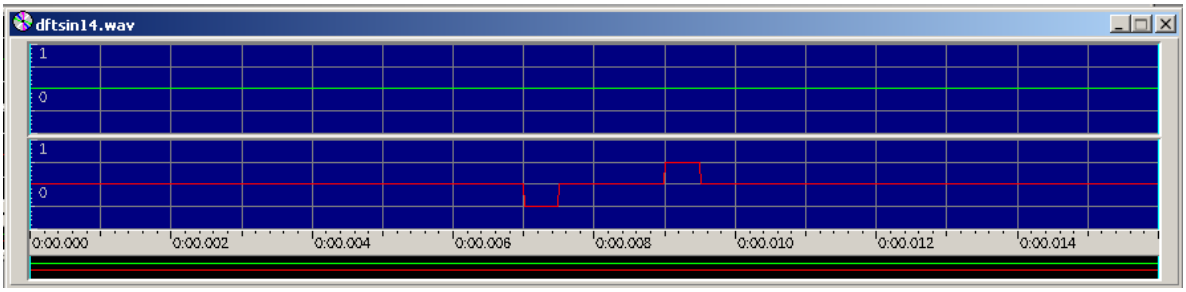

Ilustración 31 DFT del Seno de 875 Hz

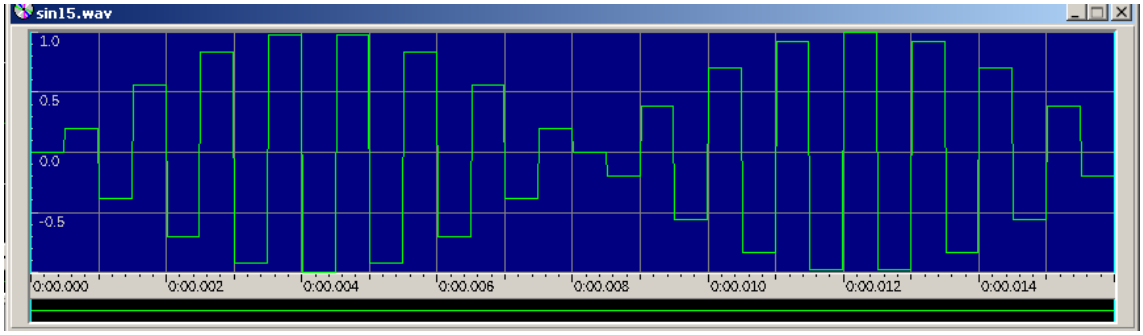

Ilustración 32 Seno de 937.5

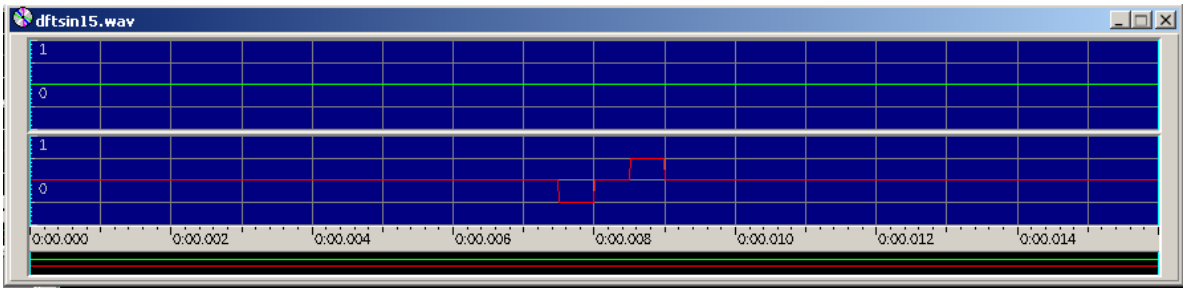

Ilustración 33 DFT del Seno de 937.5

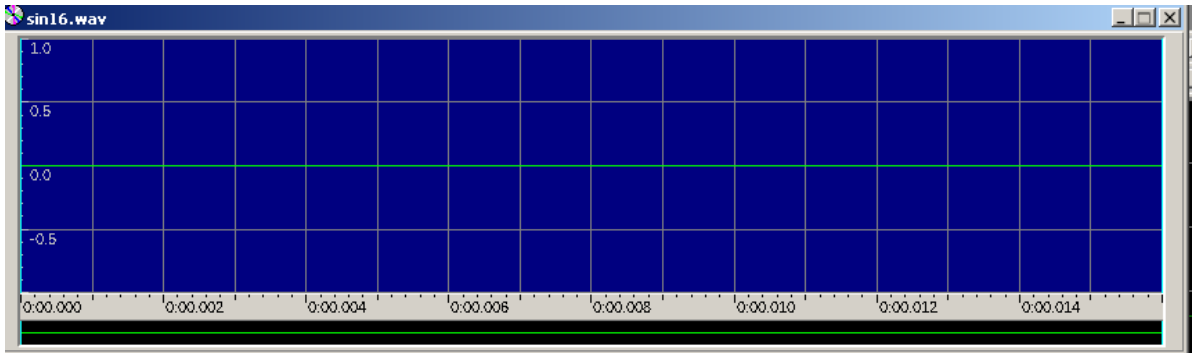

Ilustración 34 Seno de 1000 Hz

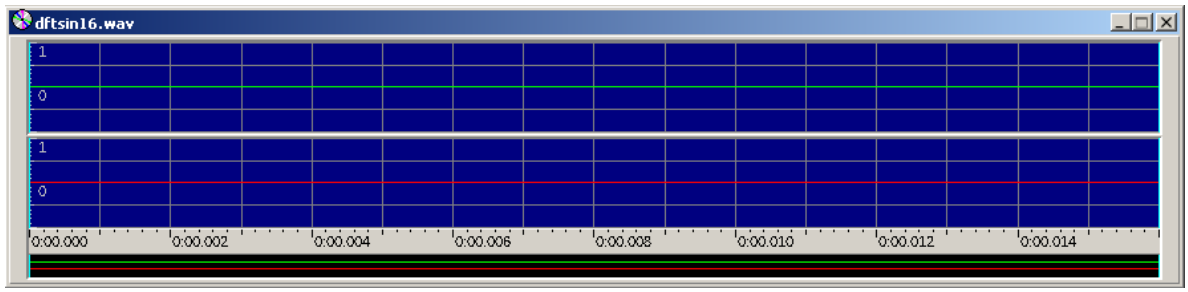

Ilustración 35 DFT del Seno de 1000 Hz

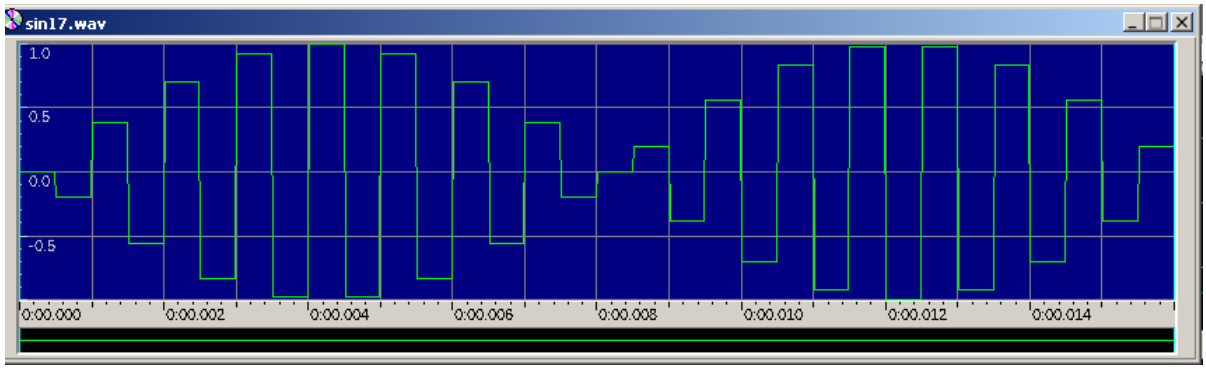

Ilustración 36 Seno de 1062.5 Hz

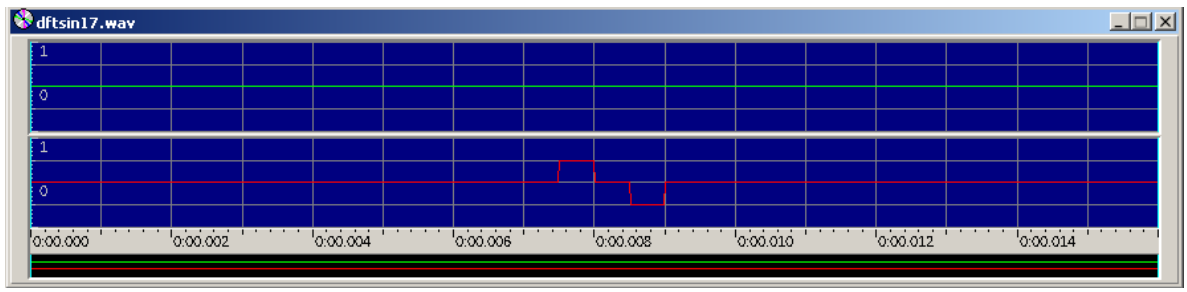

Ilustración 37 DFT del Seno de 1062.5 Hz

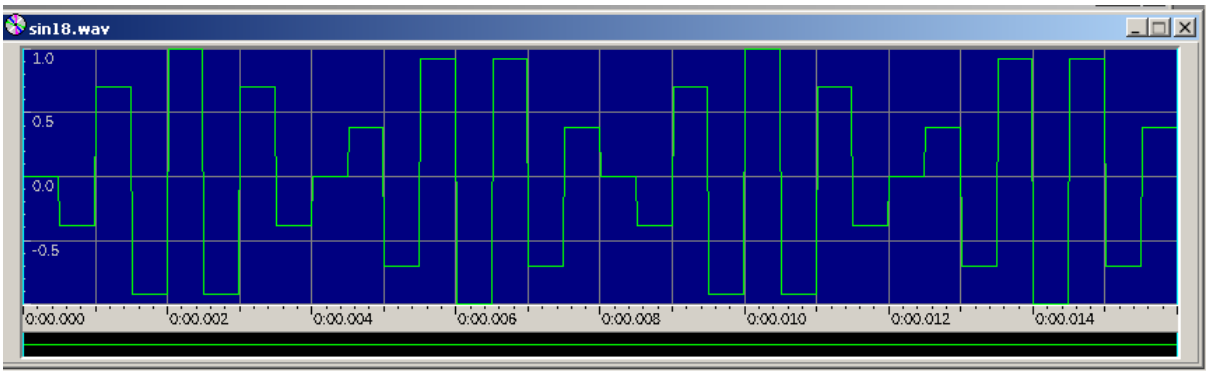

Ilustración 38 Seno de 1125 Hz

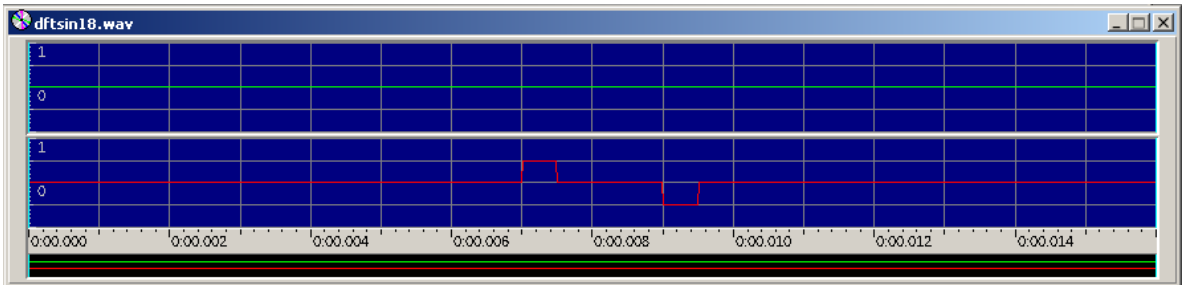

Ilustración 39 DFT del Seno de 1125 Hz

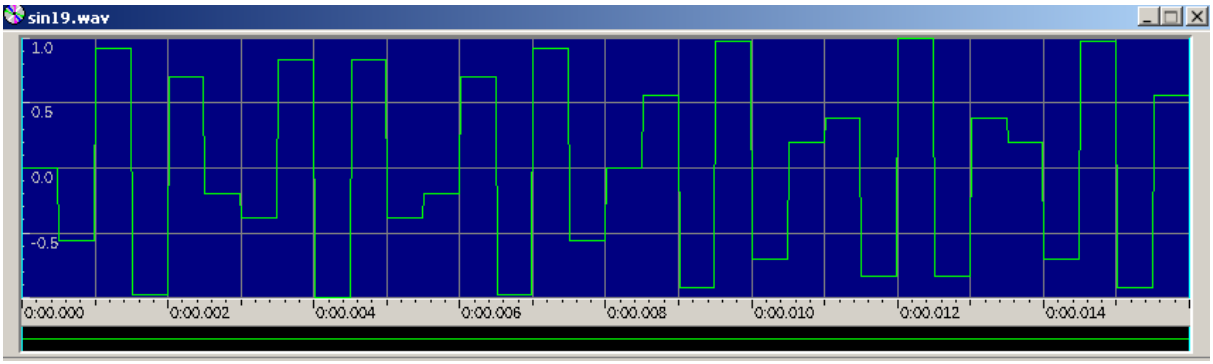

Ilustración 40 Seno de 1187.5 Hz

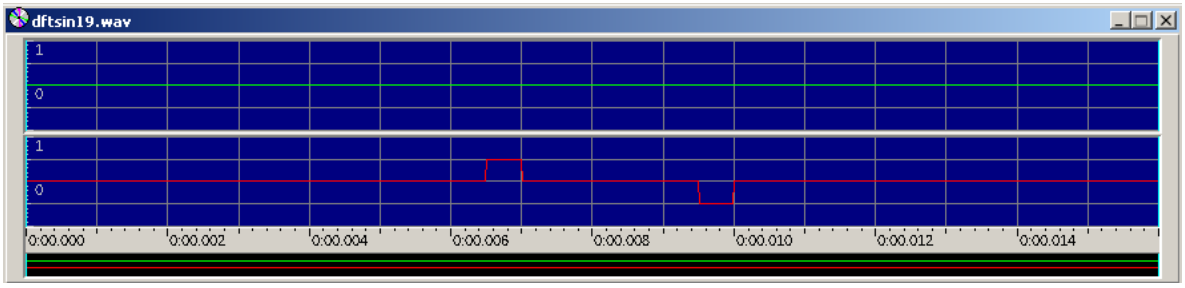

Ilustración 41 DFT Seno de 1187.5 Hz

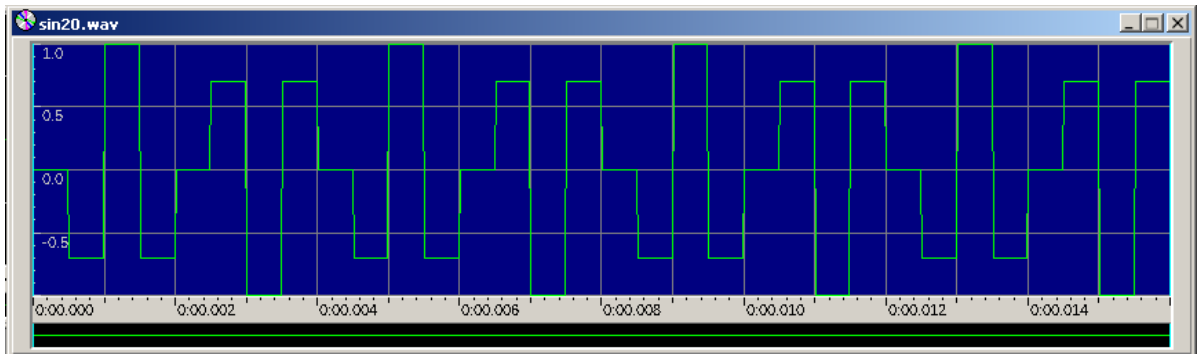

Ilustración 42 Seno de 1250 Hz

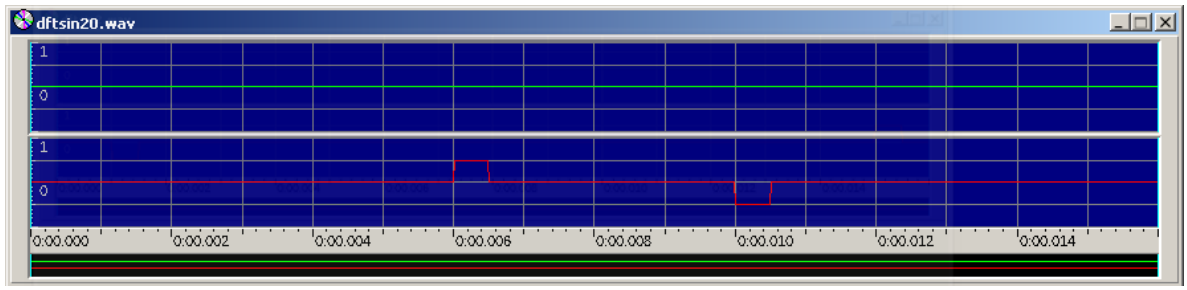

Ilustración 43 DFT del Seno de 1250 Hz

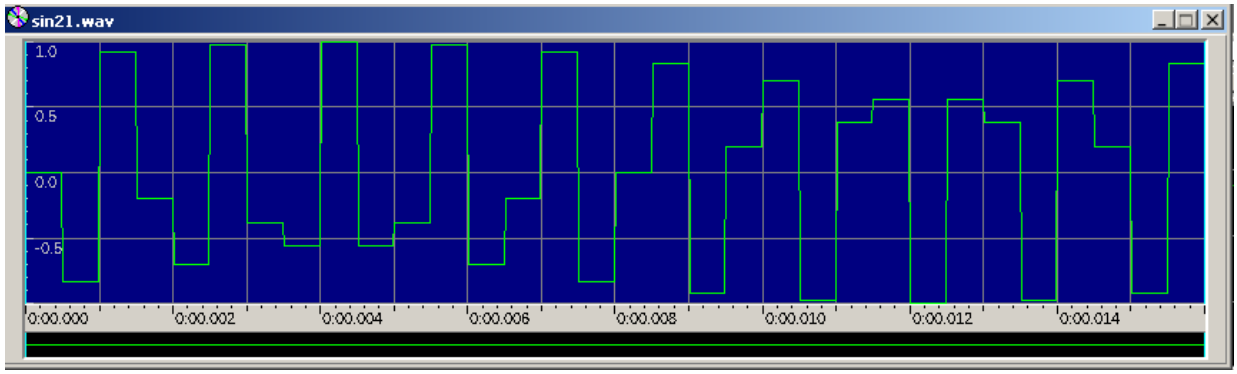

Ilustración 44 Seno de 1312.5 Hz

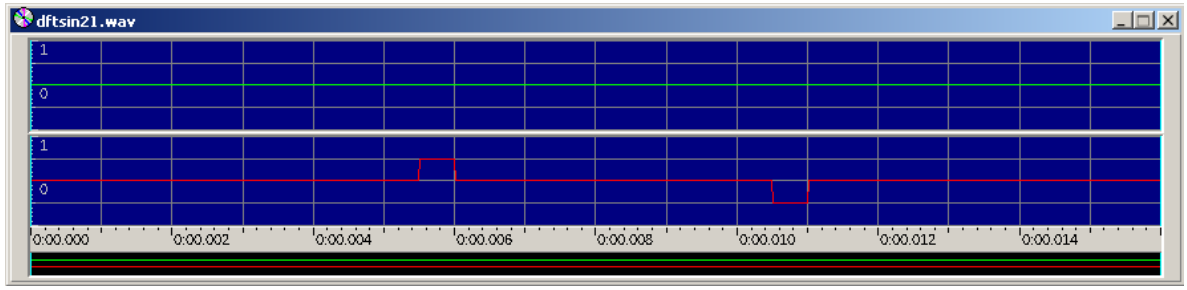

Ilustración 45 DFT del Seno de 1312.5 Hz

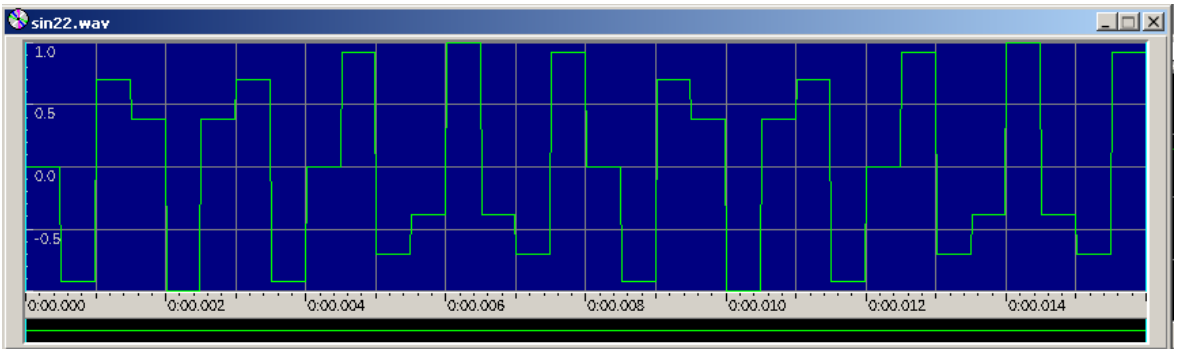

Ilustración 46 Seno de 1375 Hz

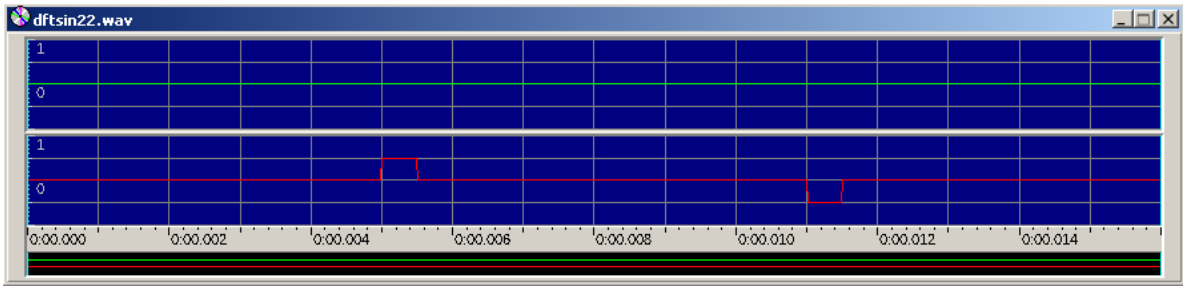

Ilustración 47 DFT del Seno de 1375 Hz

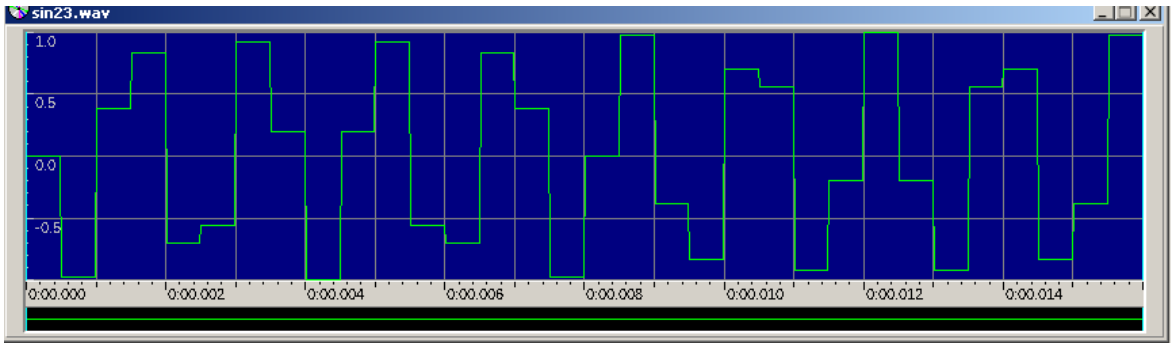

Ilustración 48 Seno de 1437.5 Hz

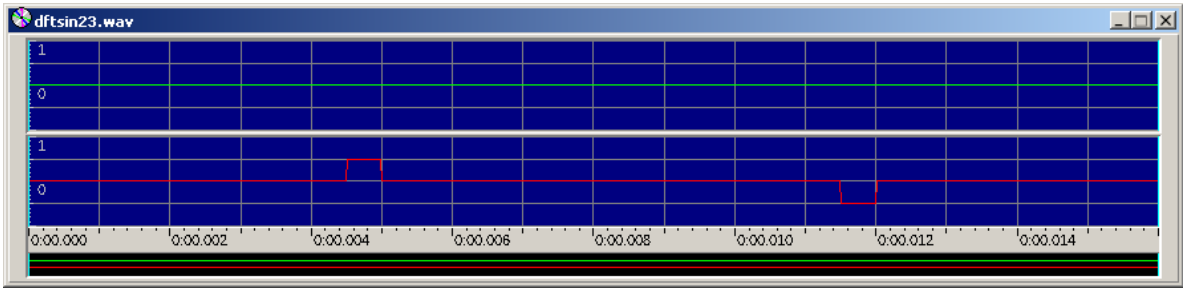

Ilustración 49 DFT del Seno de 1437.5 Hz

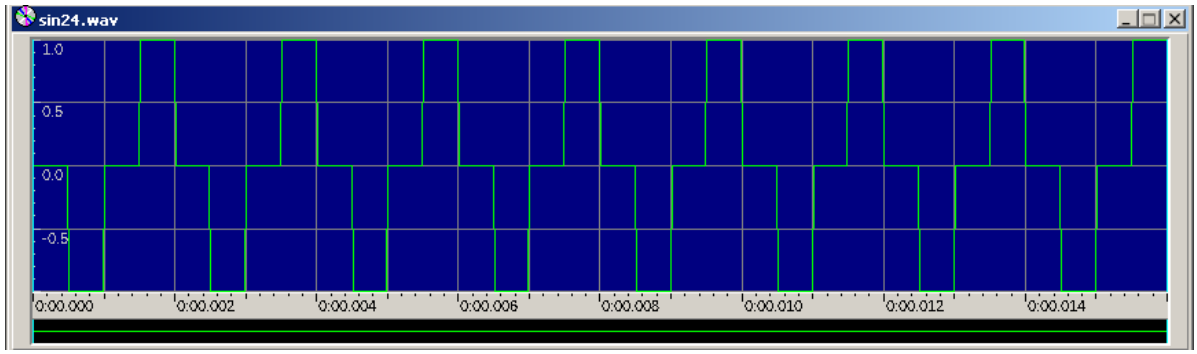

Ilustración 50 Seno de 1500 Hz

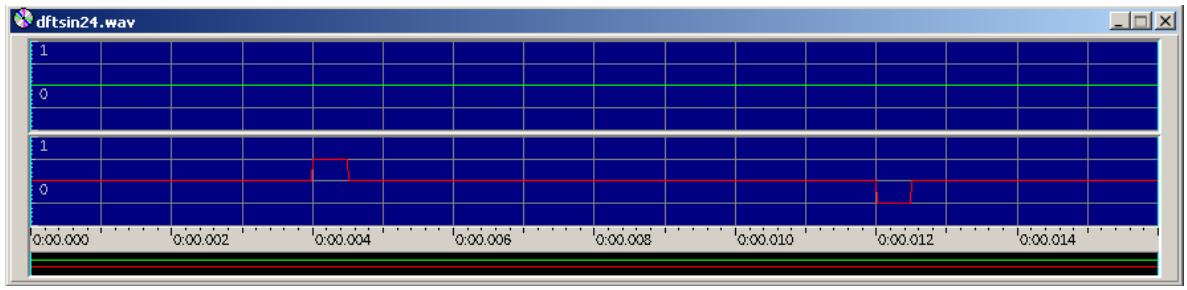

Ilustración 51 DFT del Seno de 1500 Hz

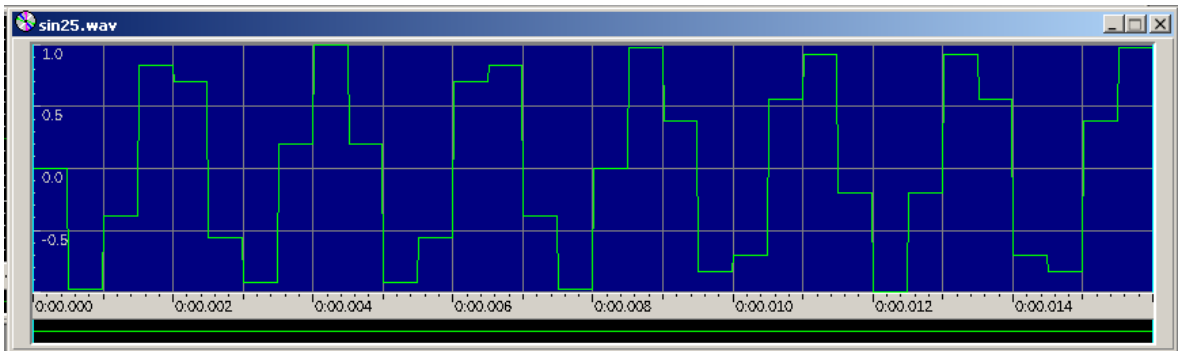

Ilustración 52 Seno de 1562.5 Hz

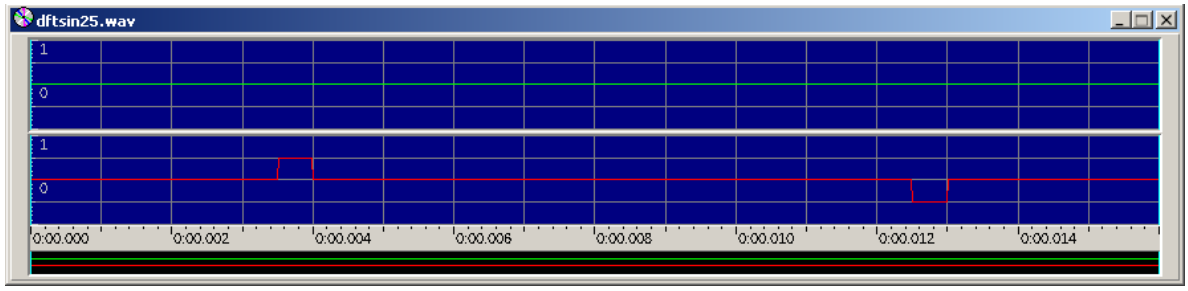

Ilustración 53 DFT del Seno de 1562.5 Hz

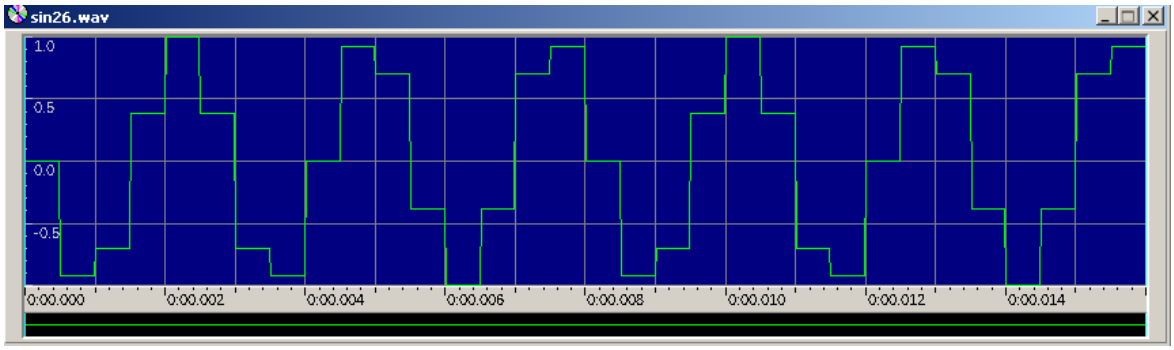

Ilustración 54 Seno de 1625 Hz

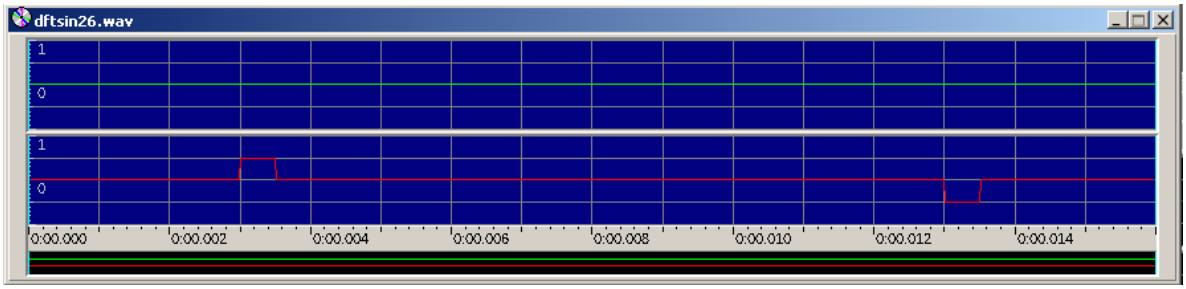

Ilustración 55 DFT del Seno de 1625 Hz

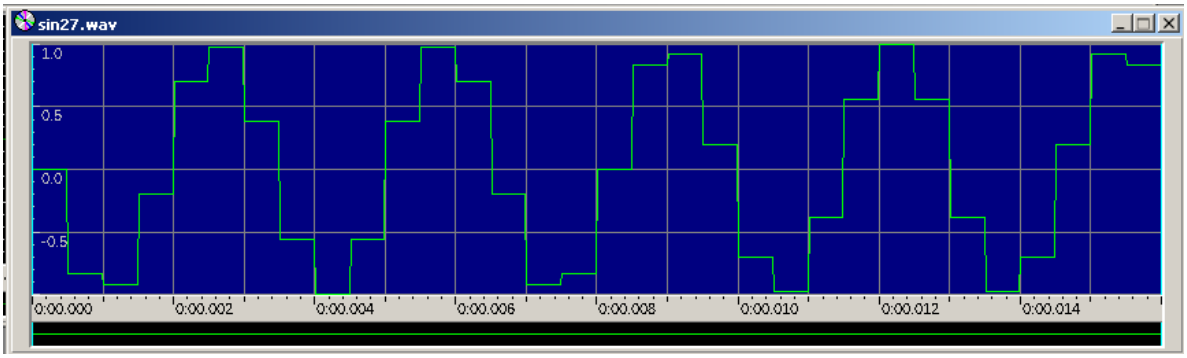

Ilustración 56 Seno de 1687.5 Hz

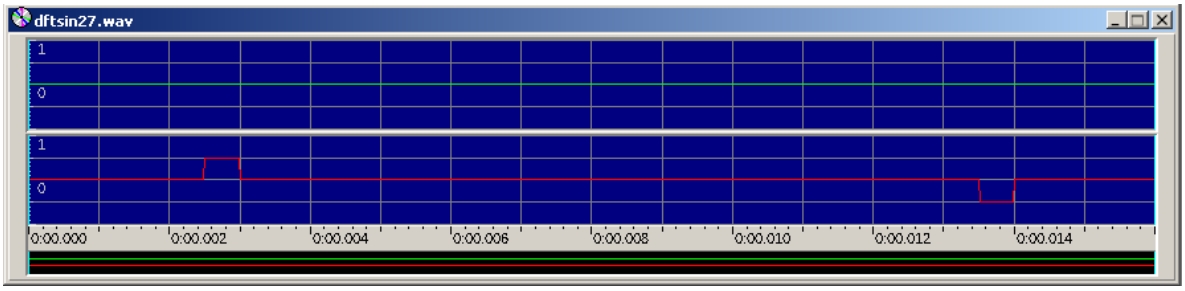

Ilustración 57 DFT del Seno de 1687.5 Hz

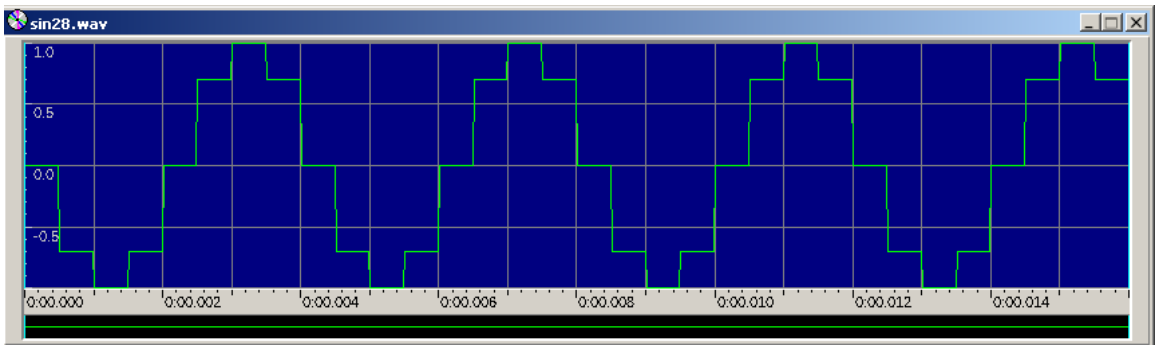

Ilustración 58 Seno de 1750 Hz

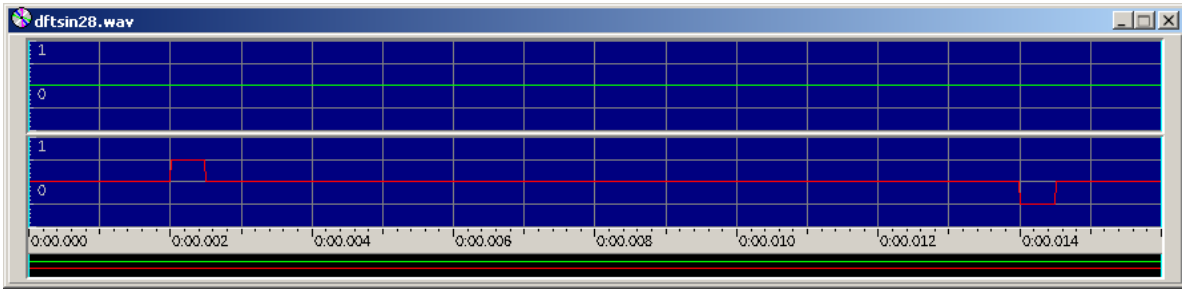

Ilustración 59 DFT del Seno de 1750 Hz

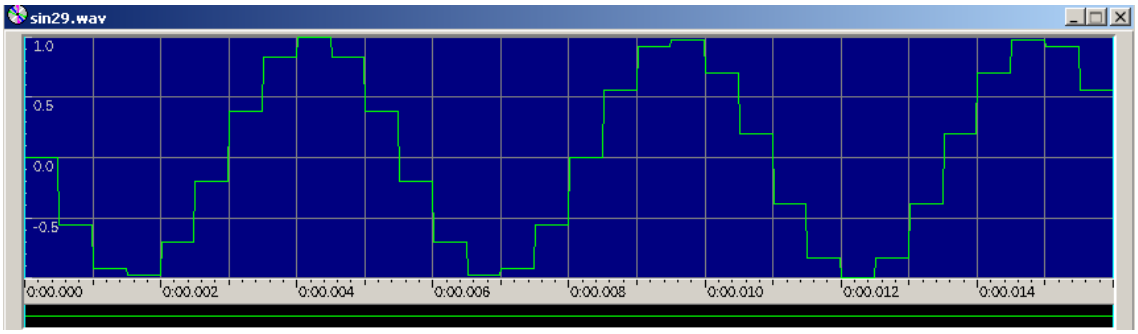

Ilustración 60 Seno de 1812.5 Hz

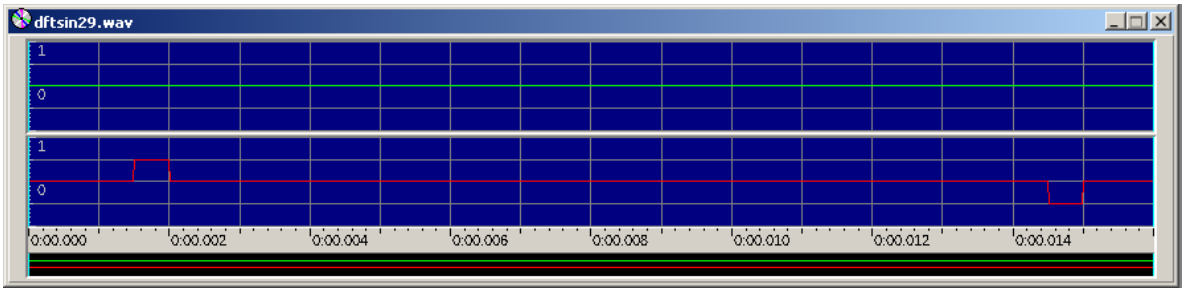

Ilustración 61 DFT Seno de 1812.5 Hz

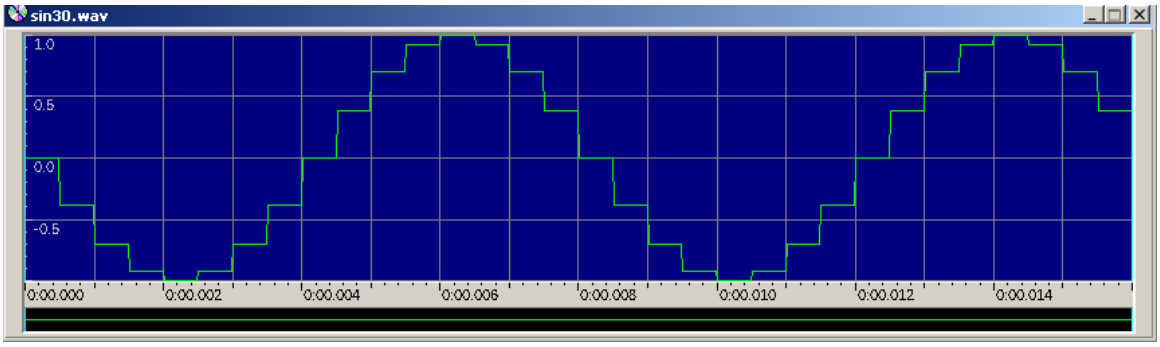

Ilustración 62 Seno de 1875 Hz

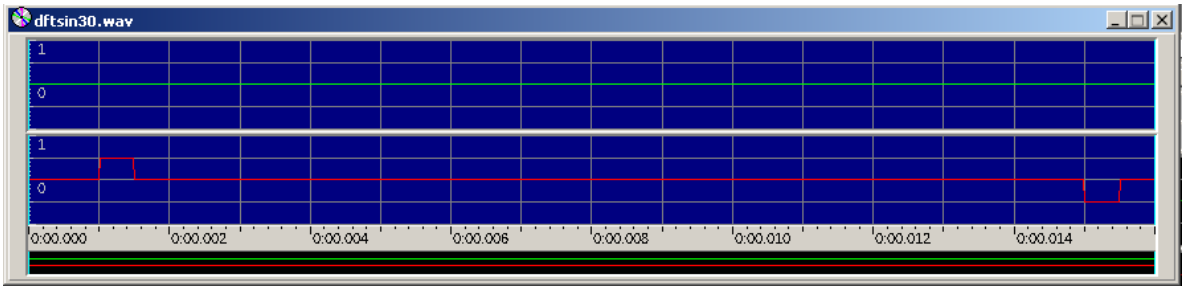

Ilustración 63 DFT del Seno de 1875 Hz

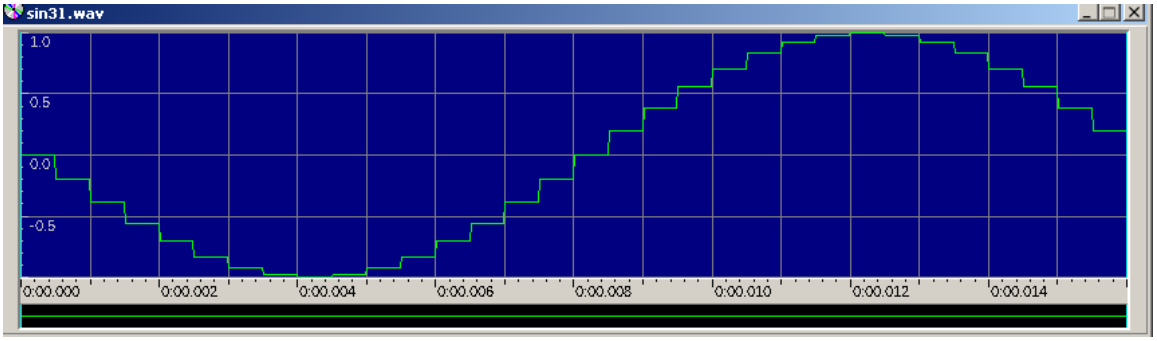

Ilustración 64 Seno de 1937.5 Hz

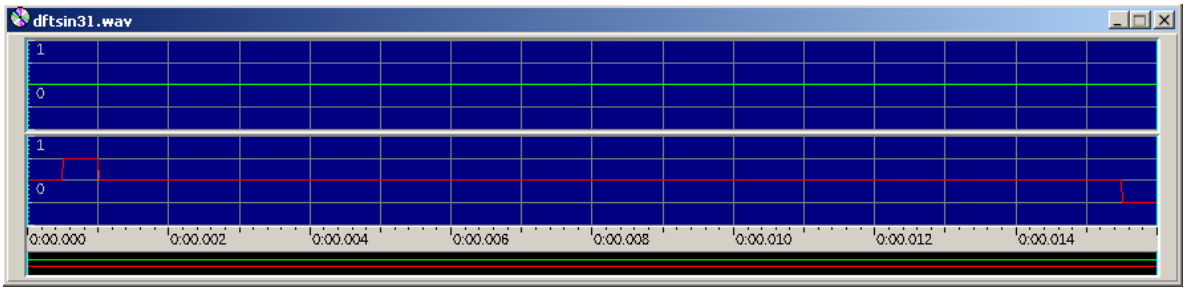

Ilustración 65 DFT del Seno de 1937.5 Hz

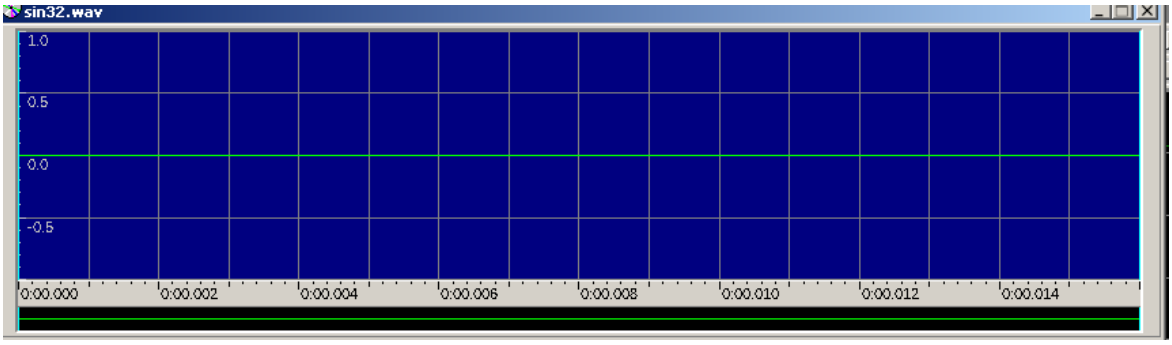

Ilustración 66 Seno de 2000 Hz

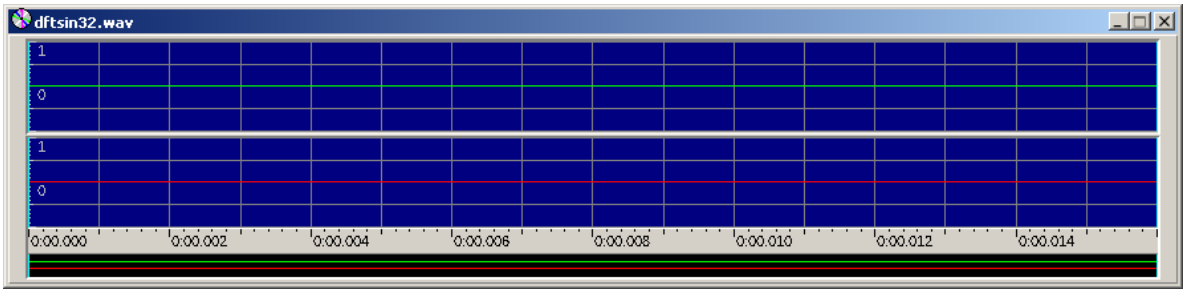

Ilustración 67 DFT del Seno de 2000 Hz

Cosenos

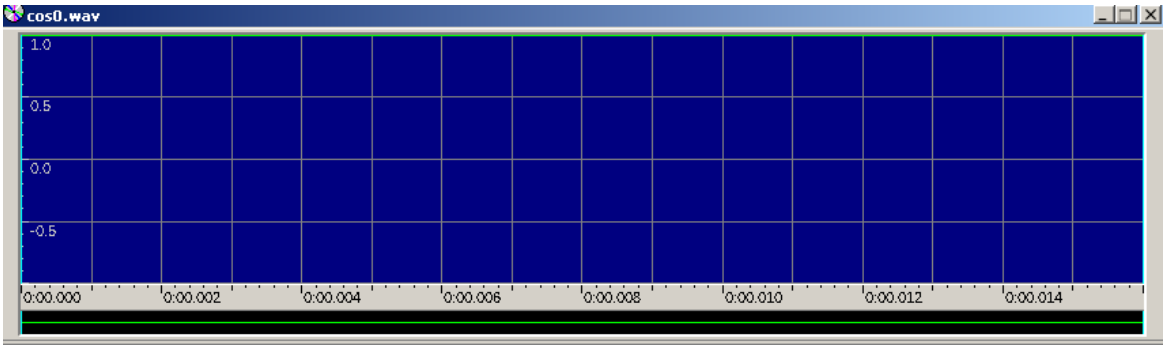

Ilustración 68 Coseno de 0 Hz

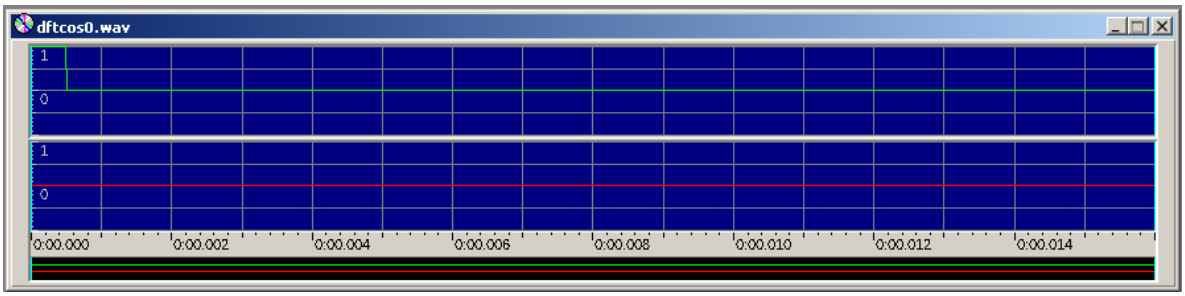

Ilustración 69 DFT del Coseno de 0 Hz

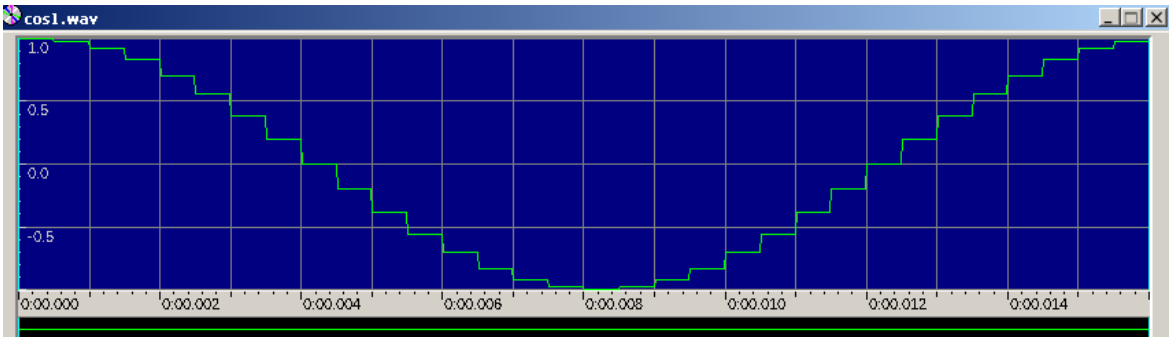

Ilustración 70 Seno de 62.5 Hz

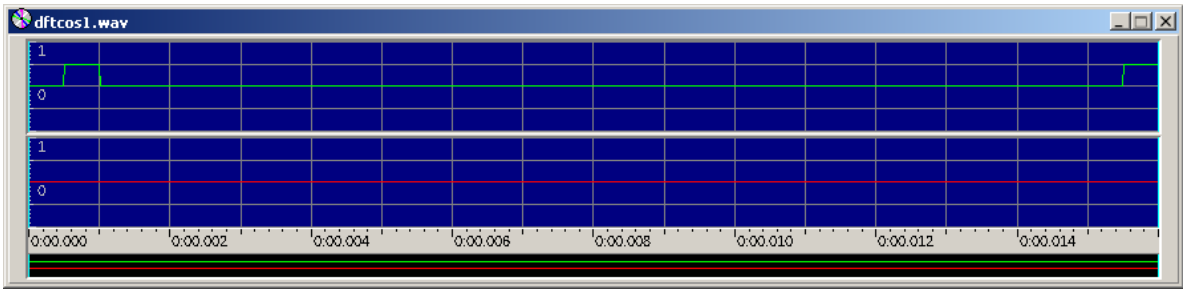

Ilustración 71 DFT del Seno de 62.5 Hz

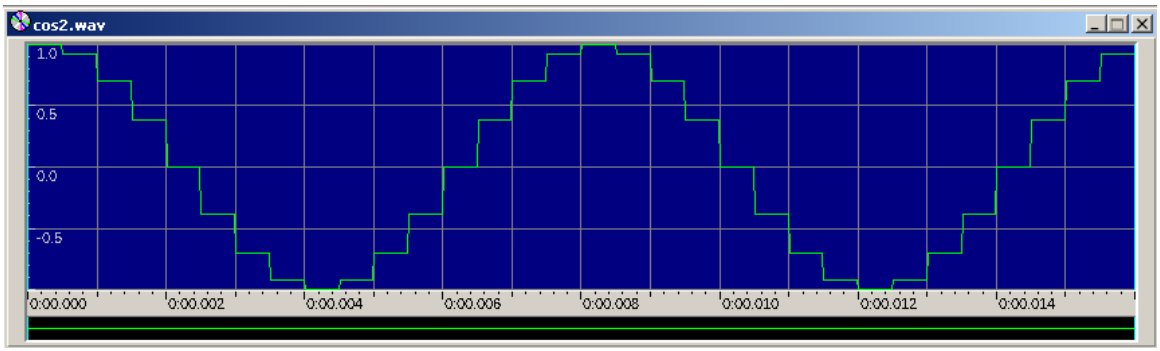

Ilustración 72 Coseno de 125 Hz

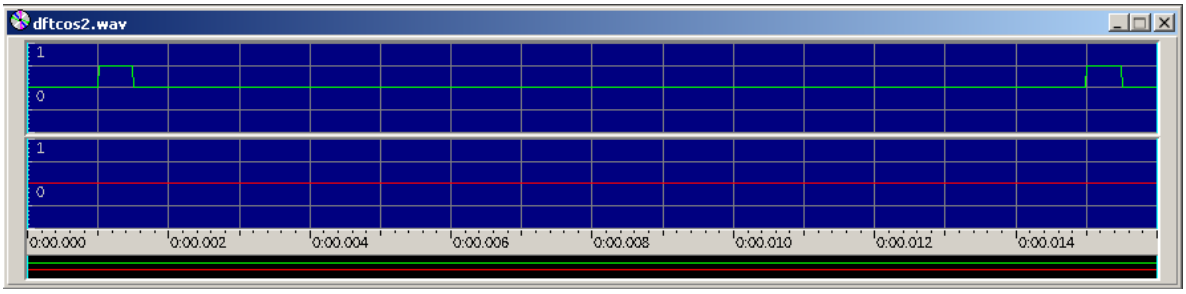

Ilustración 73 DFT del Coseno de 125 Hz

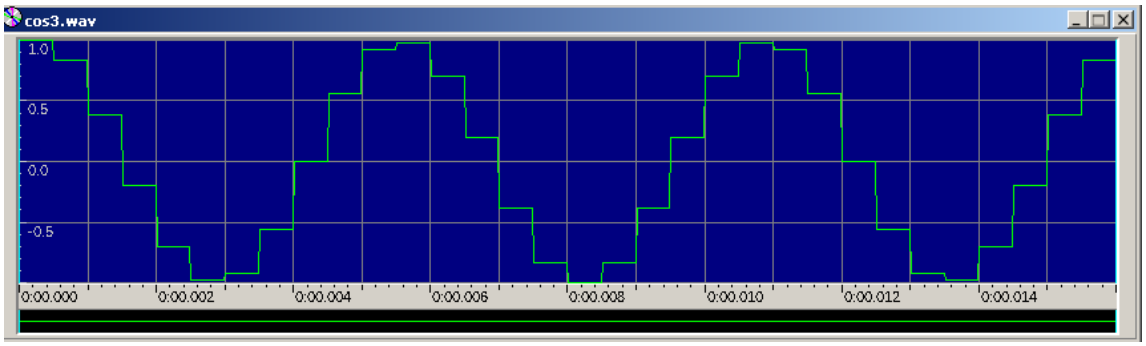

Ilustración 74 Coseno de 187.5 Hz

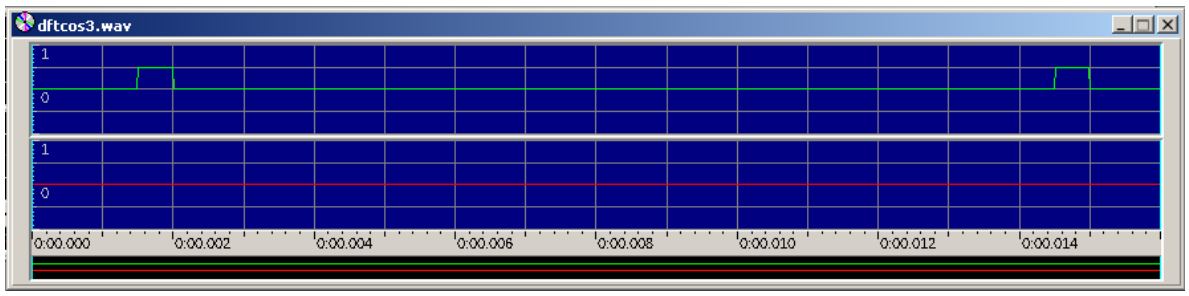

Ilustración 75 DFT del Coseno de 187.5 Hz

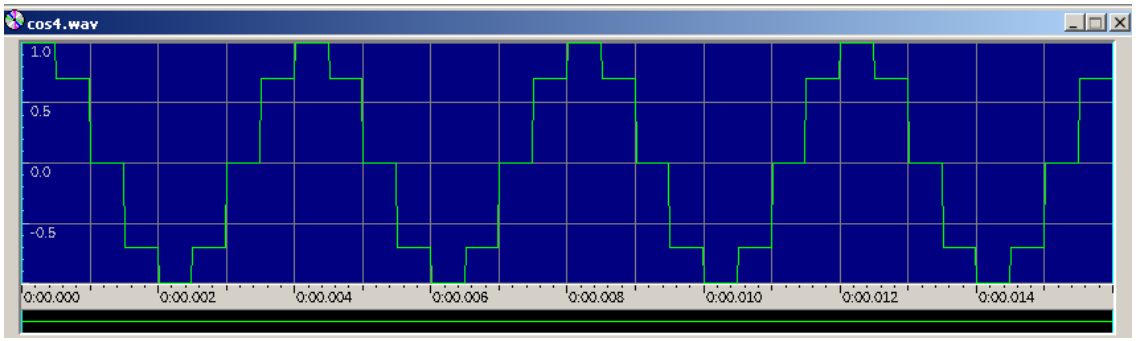

Ilustración 76 Coseno de 250 Hz

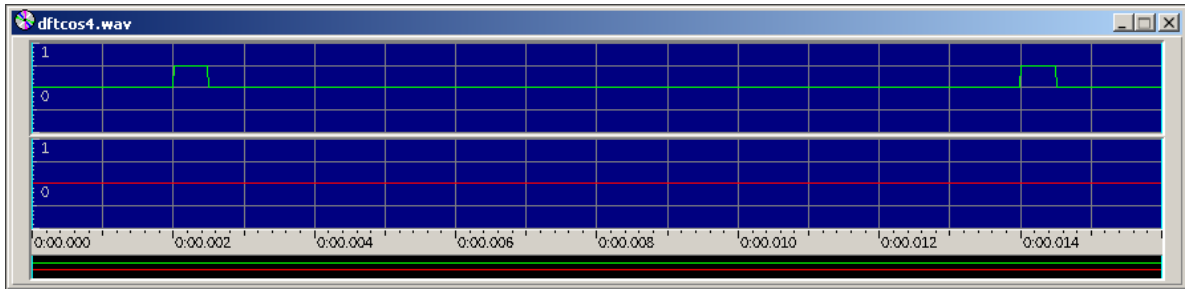

Ilustración 77 DFT del Coseno de 250 Hz

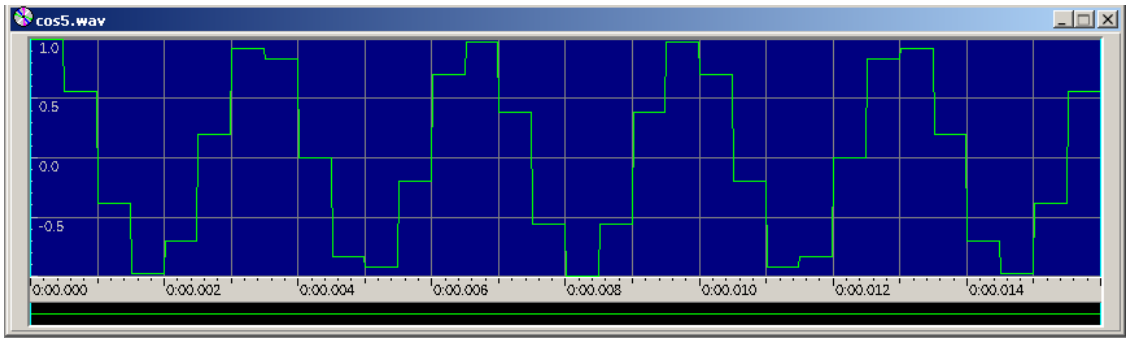

Ilustración 78 Coseno de 312.5 Hz

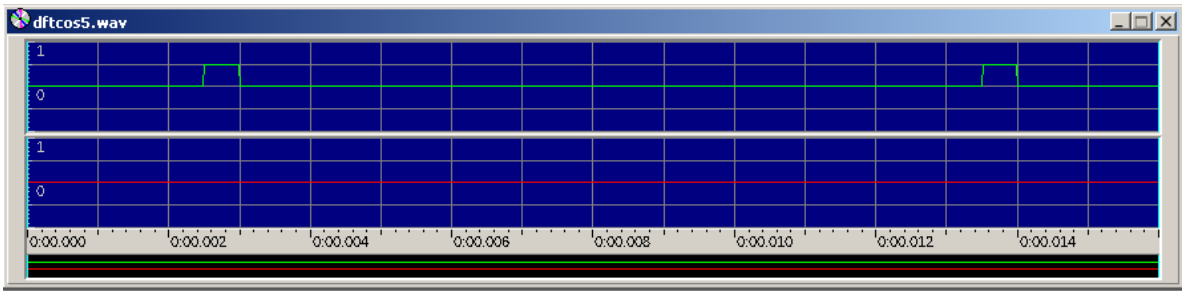

Ilustración 79 DFT del Coseno de 312.5 Hz

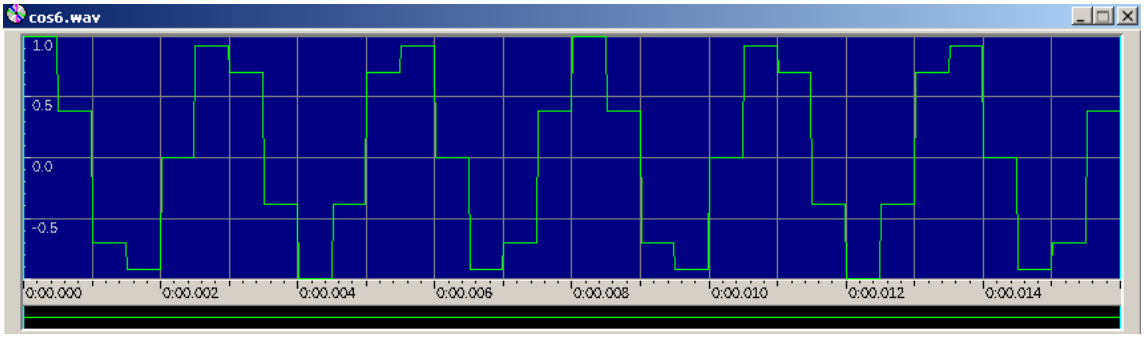

Ilustración 80 Coseno de 375 Hz

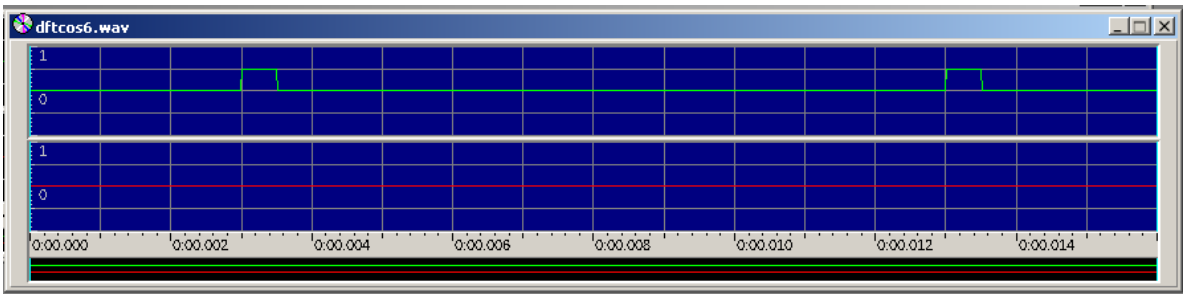

Ilustración 81 DFT Coseno de 375 Hz

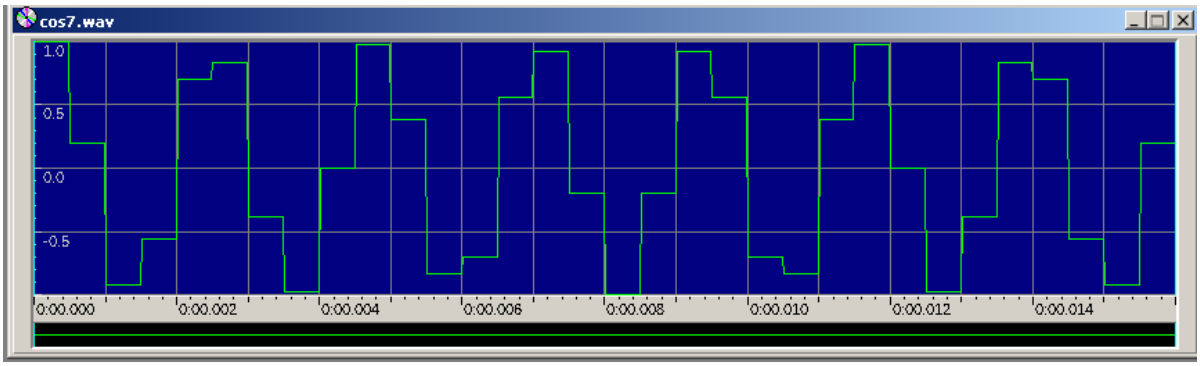

Ilustración 82 Coseno de 437.5 Hz

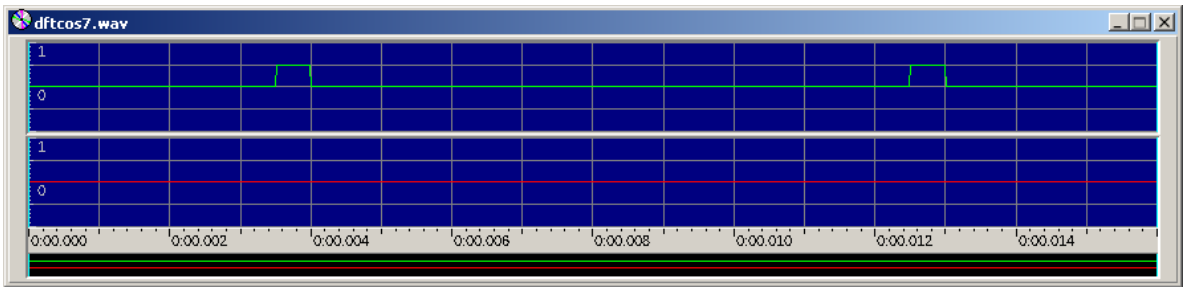

Ilustración 83 DFT del Coseno de 437.5 Hz

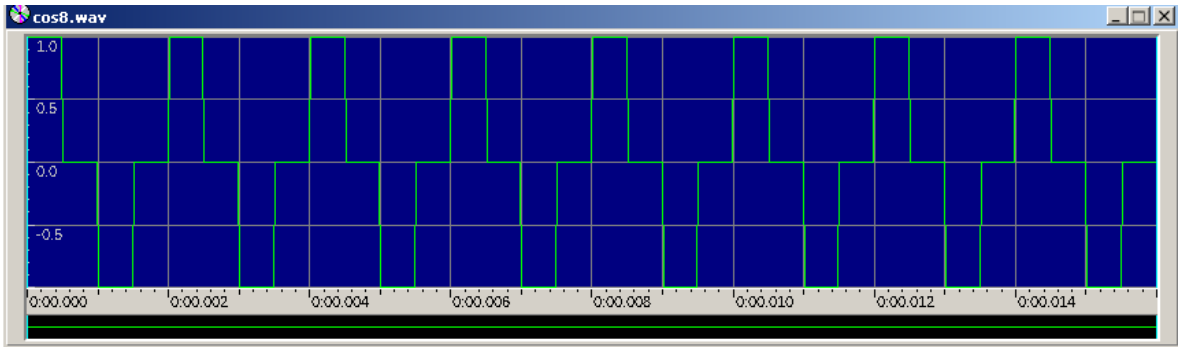

Ilustración 84 Coseno de 500 Hz

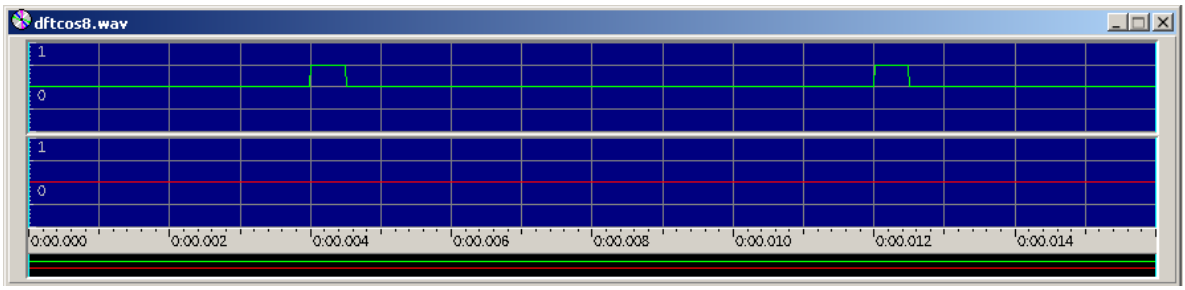

Ilustración 85 DFT del Coseno de 500 Hz

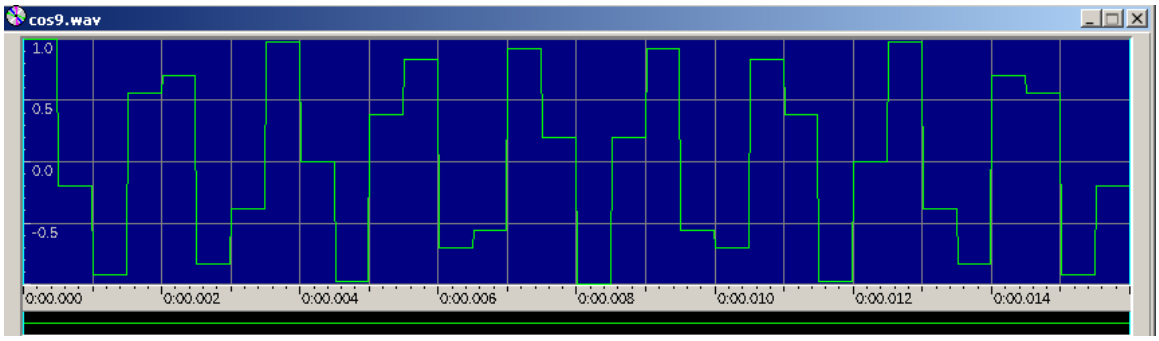

Ilustración 86 Coseno de 562.5 Hz

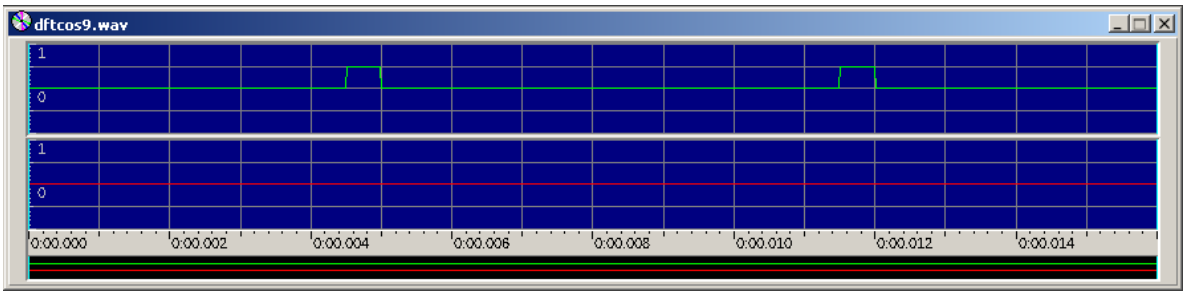

Ilustración 87 DFT del Coseno de 562.5 Hz

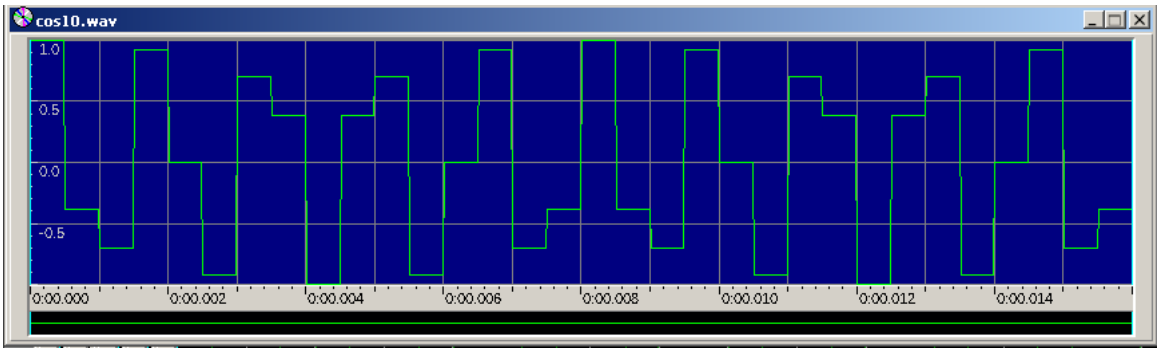

Ilustración 88 Coseno de 650 Hz

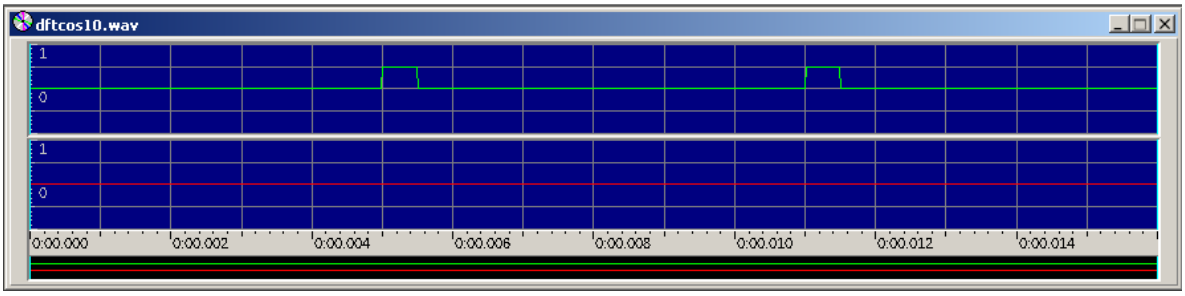

Ilustración 89 DFT del Coseno de 650 Hz

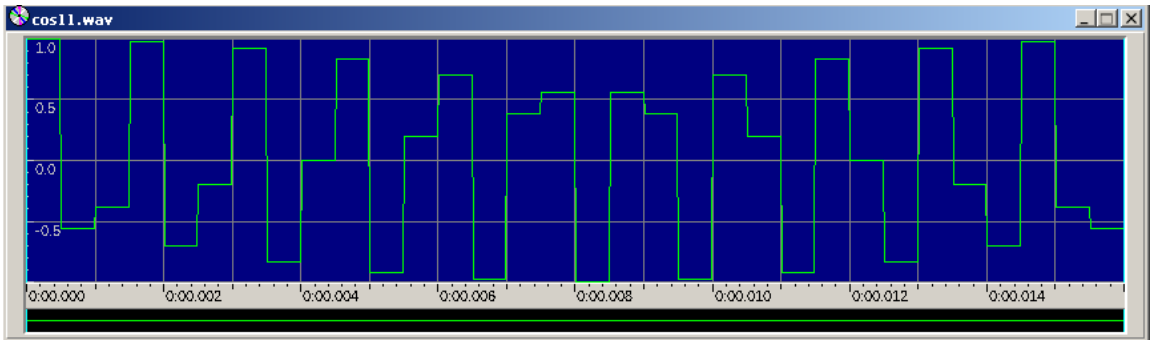

Ilustración 90 Coseno de 687.5 Hz

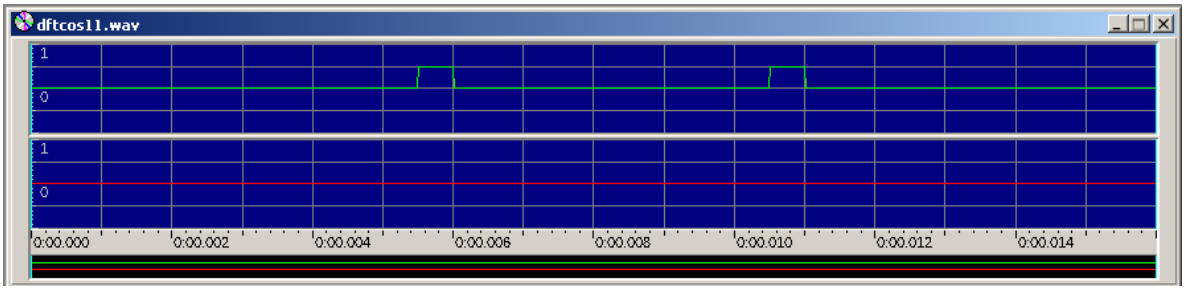

Ilustración 91 DFT del Coseno de 687.5 Hz

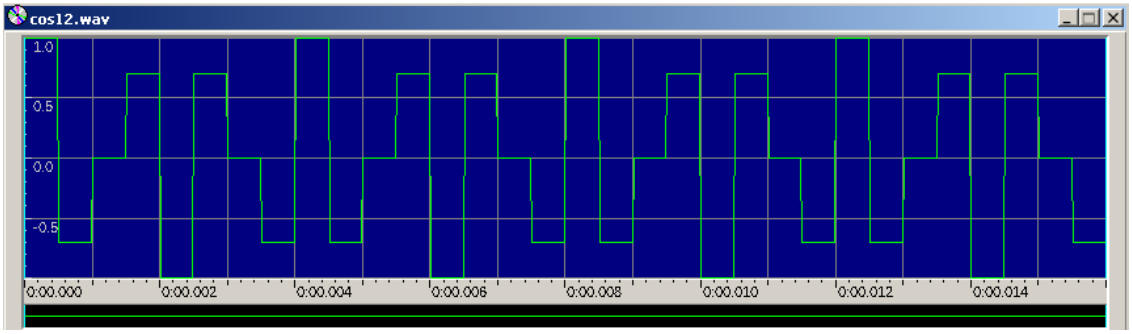

Ilustración 92 Coseno de 750 Hz

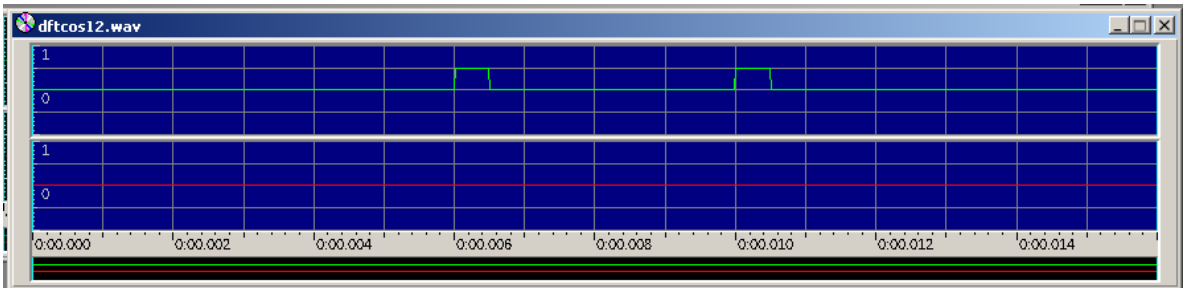

Ilustración 93 DFT del Coseno de 750 Hz

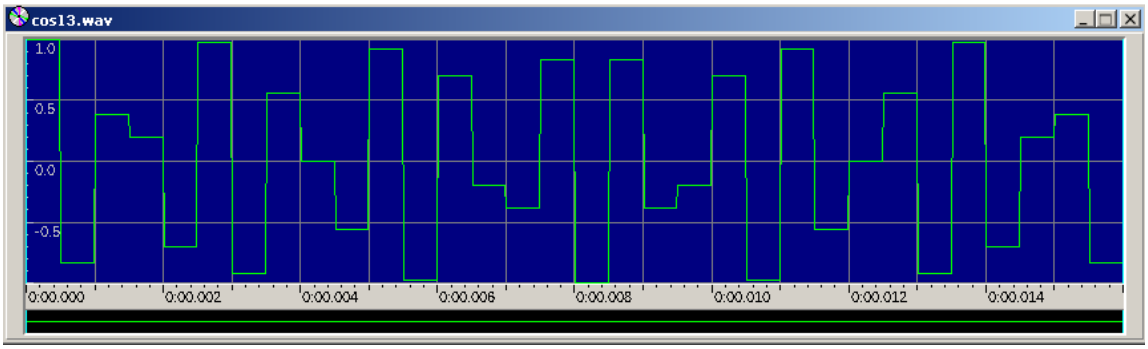

Ilustración 94 Coseno de 812.5 Hz

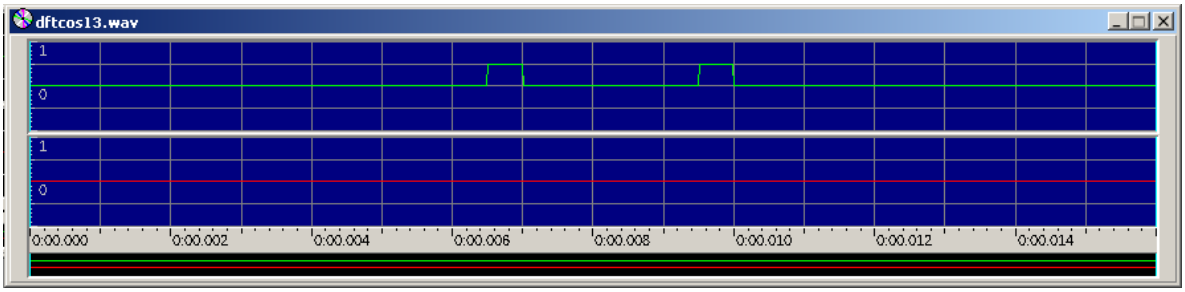

Ilustración 95 DFT del Coseno de 812.5 Hz

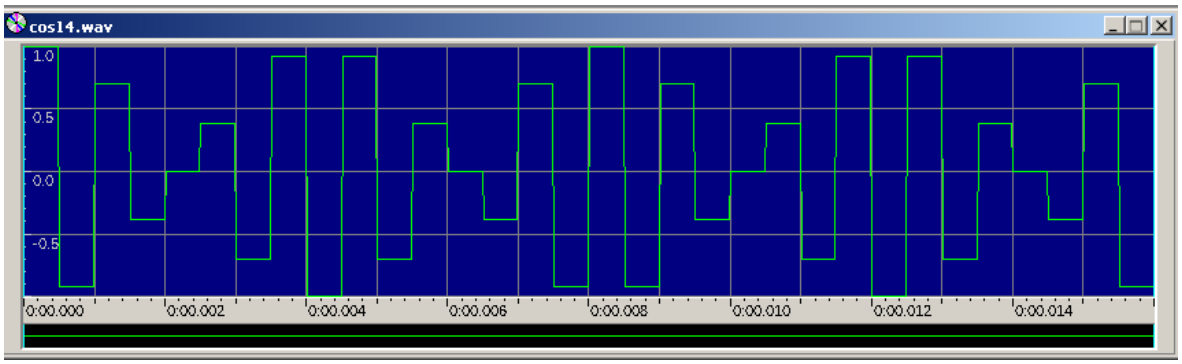

Ilustración 96 Coseno de 875 Hz

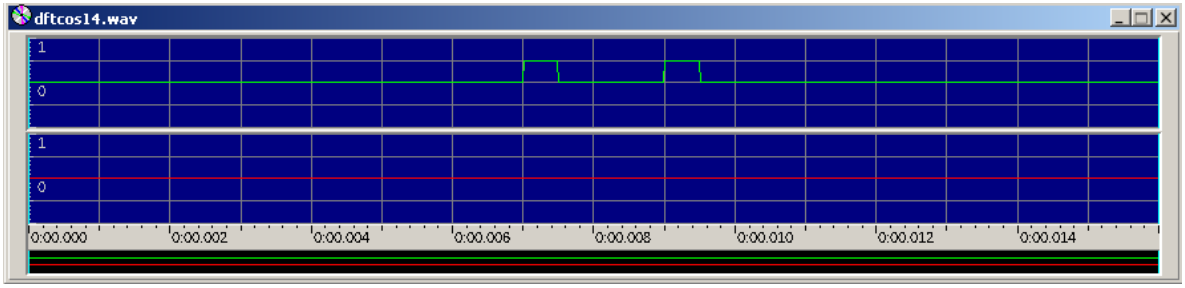

Ilustración 97 DFT del Coseno de 875 Hz

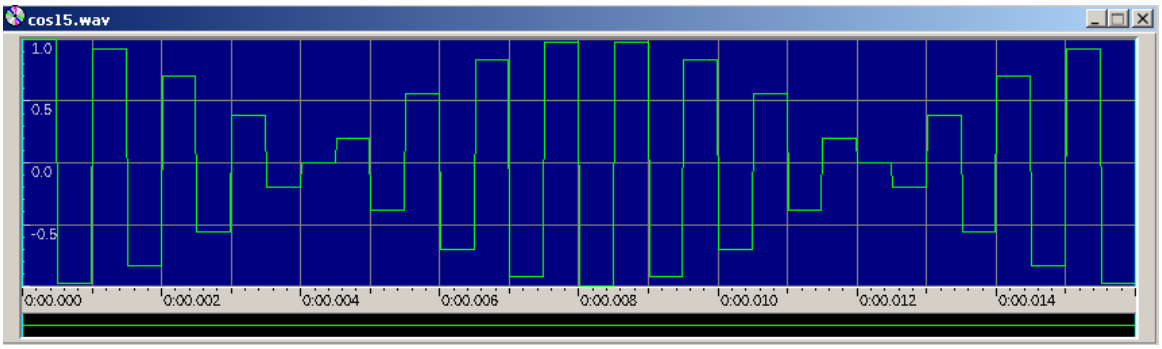

Ilustración 98 Coseno de 937.5

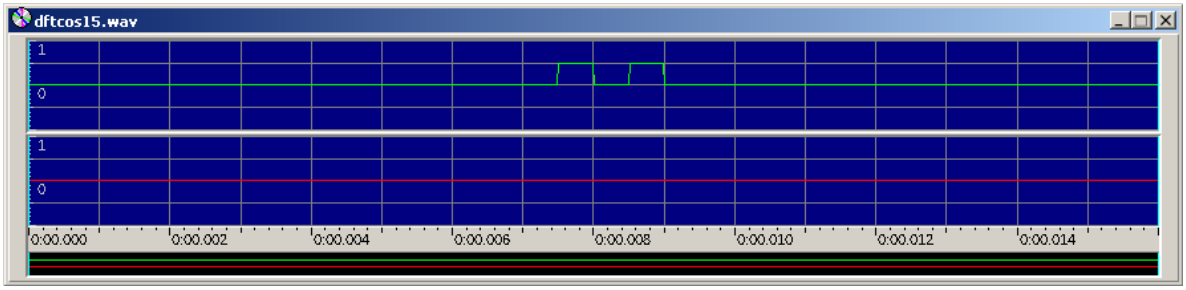

Ilustración 99 DFT del Coseno de 937.5

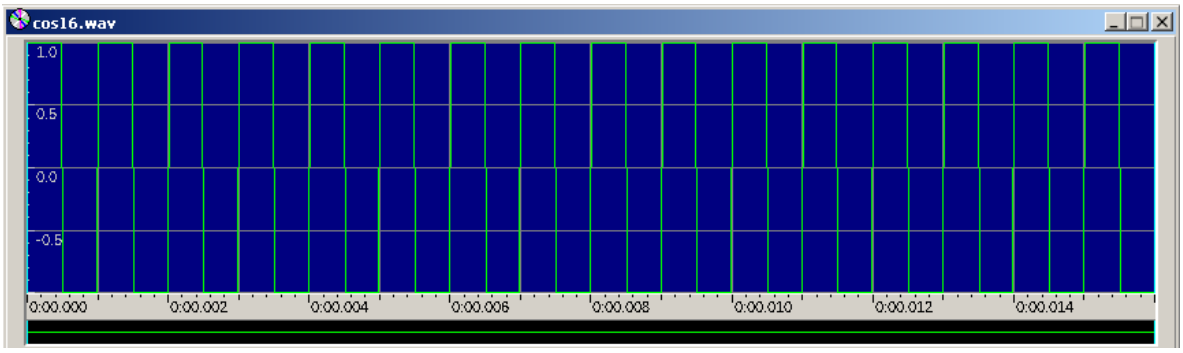

Ilustración 100 Coseno de 1000 Hz

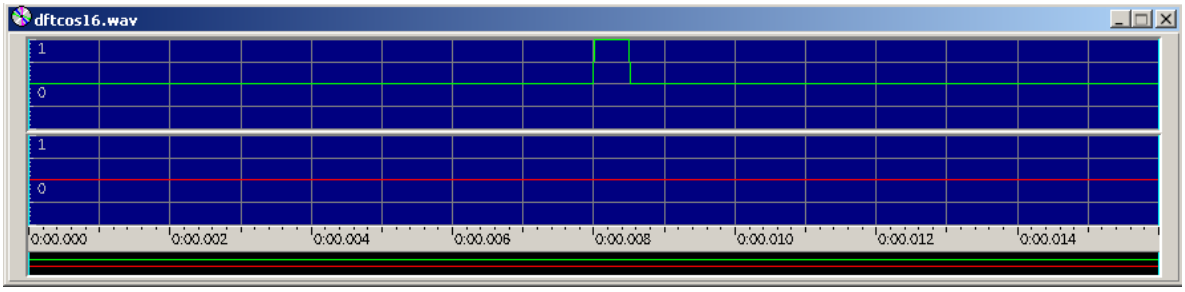

Ilustración 101 DFT del Coseno de 1000 Hz

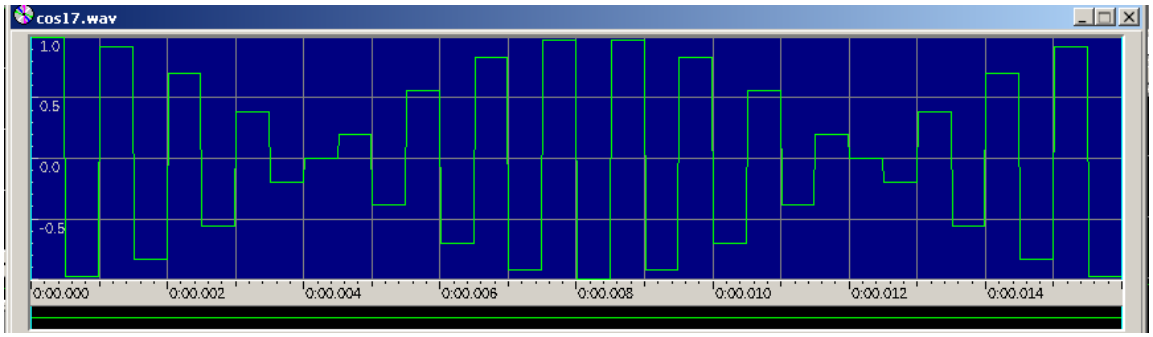

Ilustración 102 Coseno de 1062.5 Hz

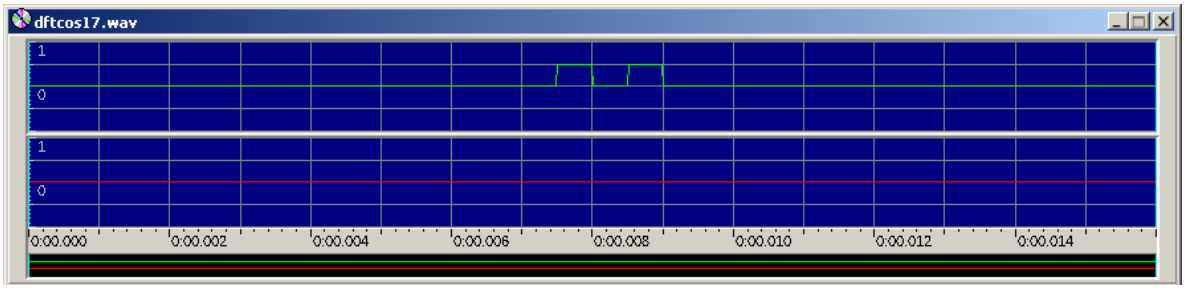

Ilustración 103 DFT del Coseno de 1062.5 Hz

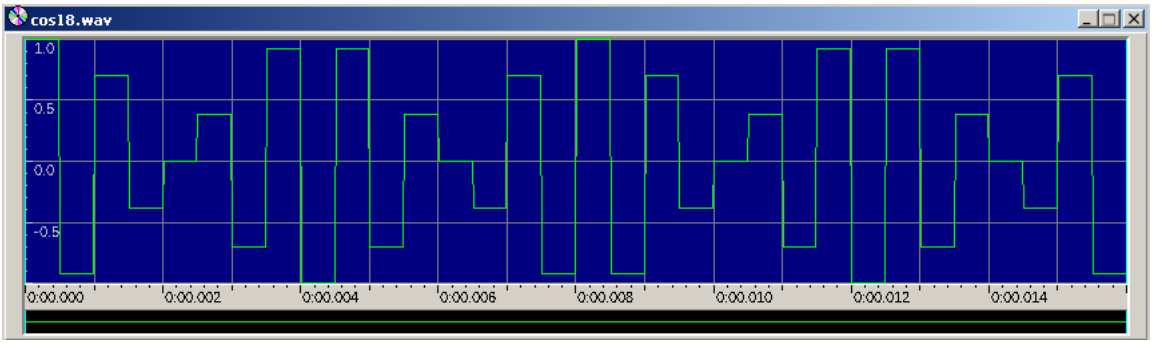

Ilustración 104 Coseno de 1125 Hz

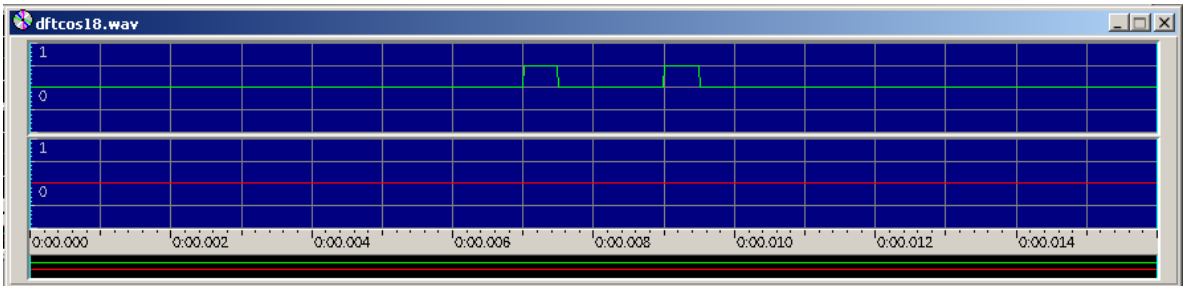

Ilustración 105 DFT del Coseno de 1125 Hz

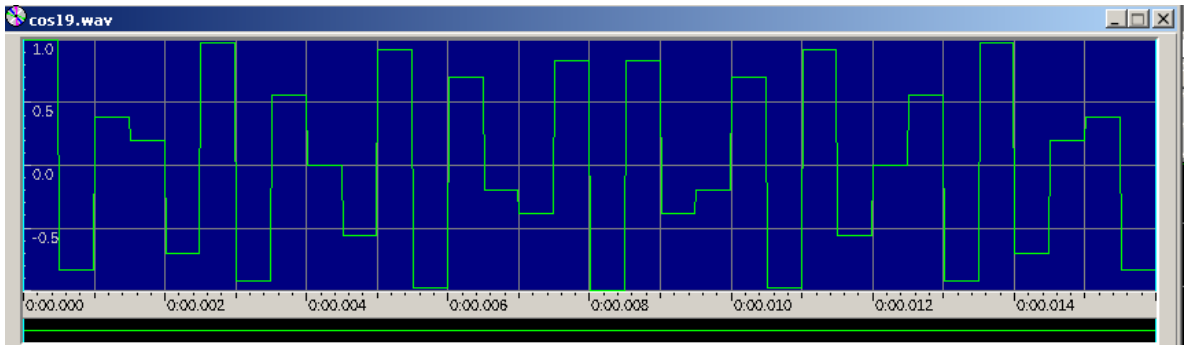

Ilustración 106 Coseno de 1187.5 Hz

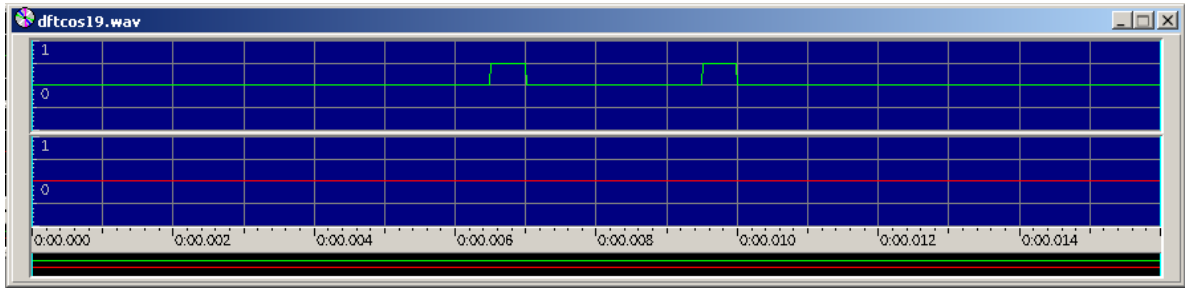

Ilustración 107 DFT del Coseno de 1187.5 Hz

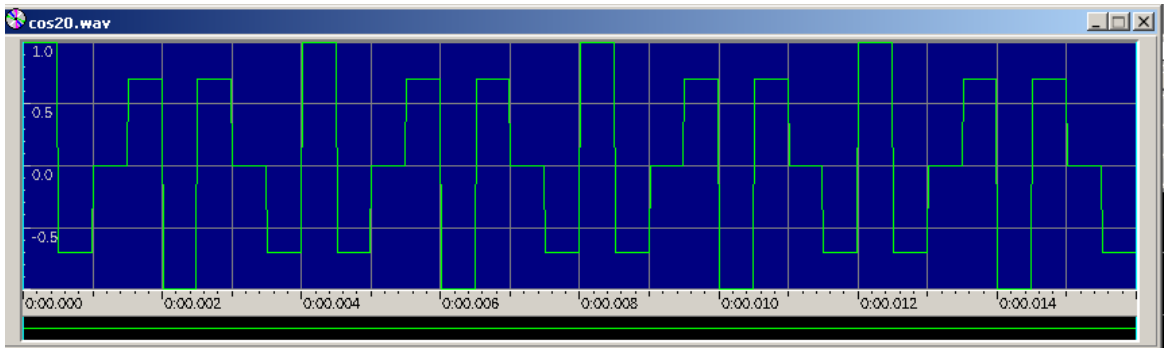

Ilustración 108 Coseno de 1250 Hz

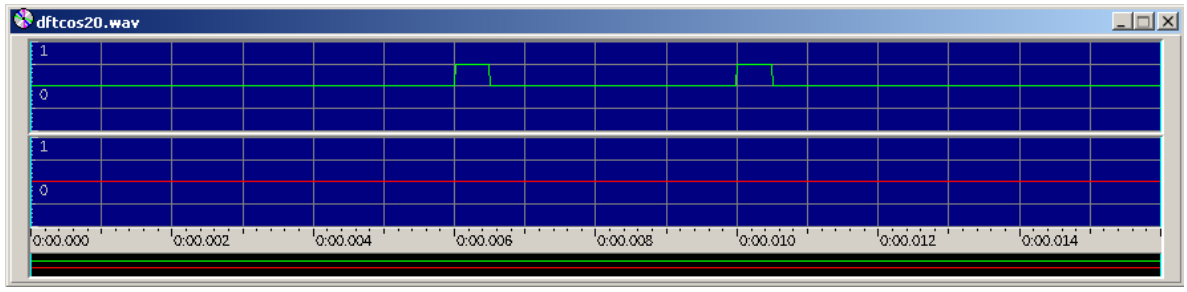

Ilustración 109 DFT del Coseno de 1250 Hz

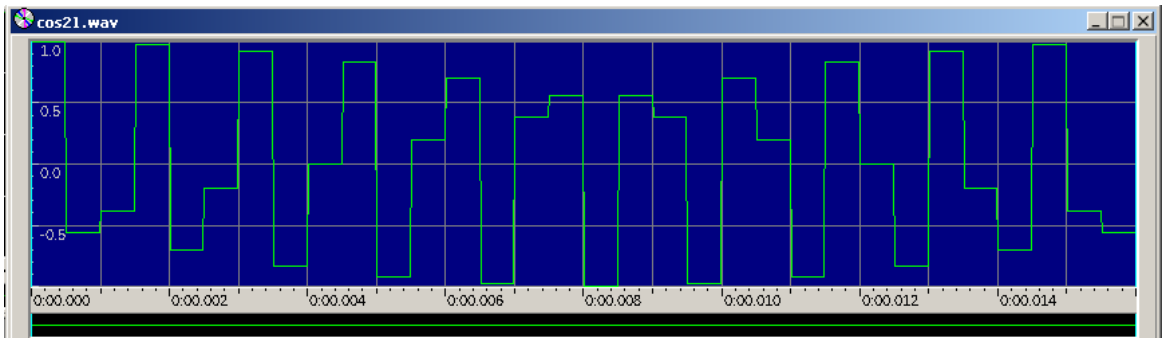

Ilustración 110 Coseno de 1312.5 Hz

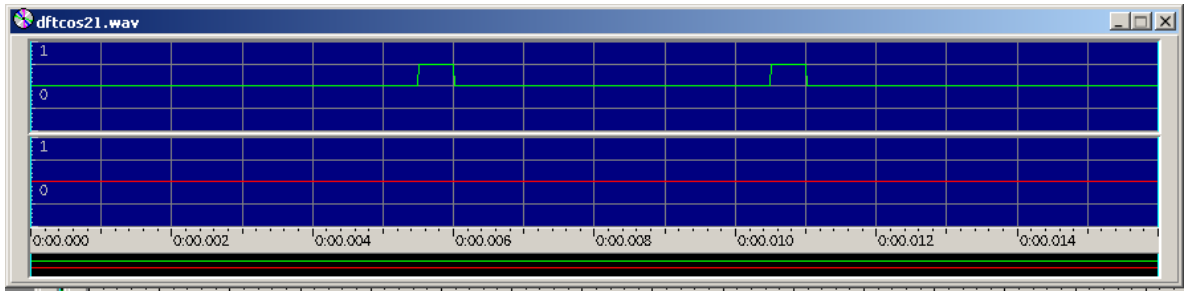

Ilustración 111 DFT del Coseno de 1312.5 Hz

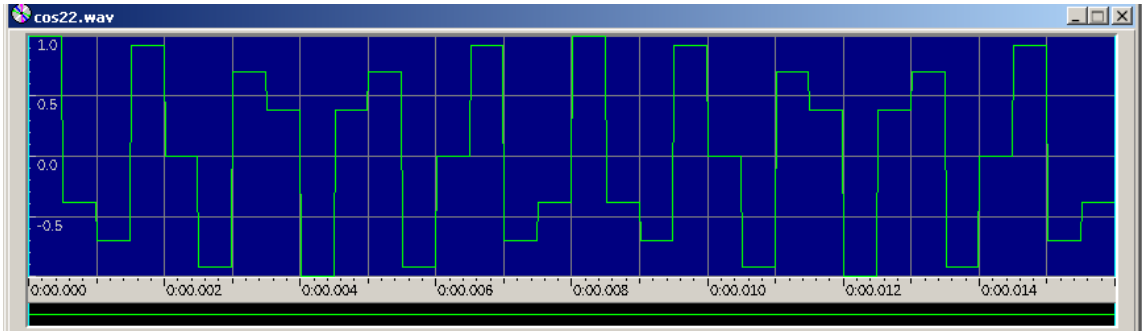

Ilustración 112 Coseno de 1375 Hz

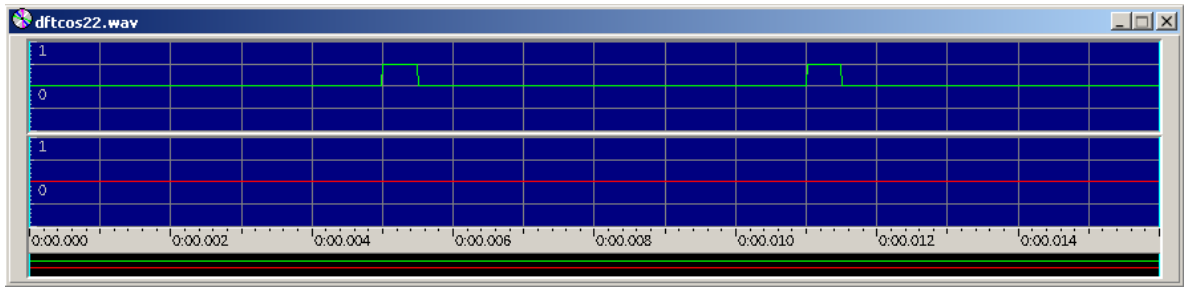

Ilustración 113 DFT del Coseno de 1375 Hz

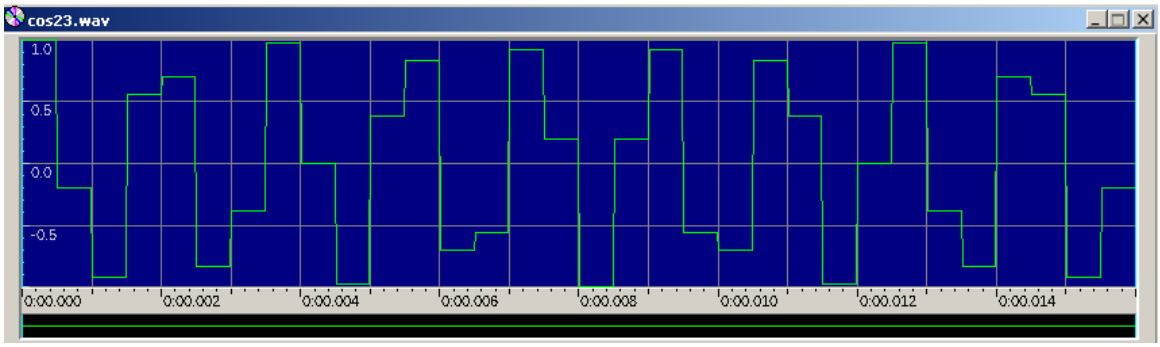

Ilustración 114 Coseno de 1437.5 Hz

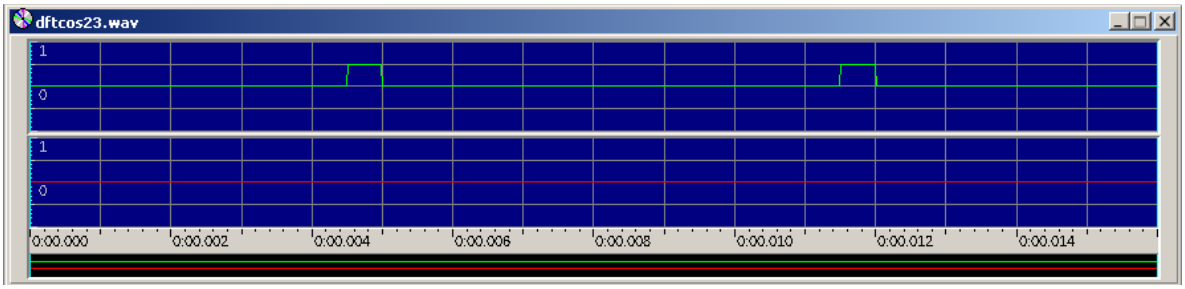

Ilustración 115 DFT del Coseno de 1437.5 Hz

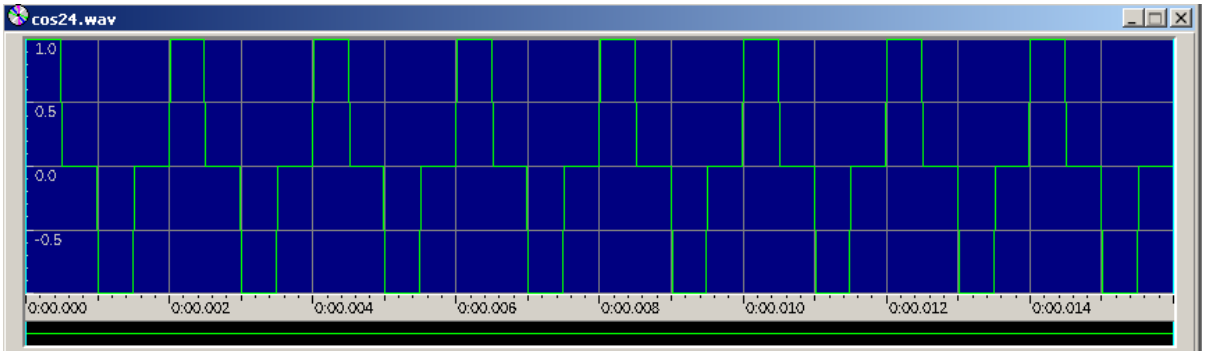

Ilustración 116 Coseno de 1500 Hz

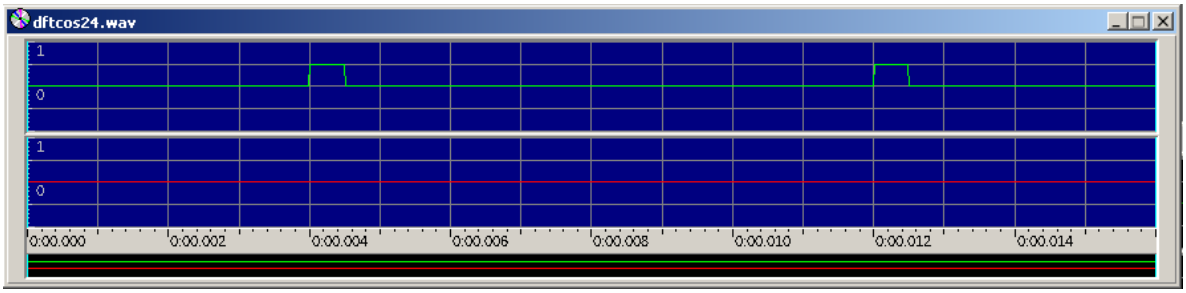

Ilustración 117 DFT del Coseno de 1500 Hz

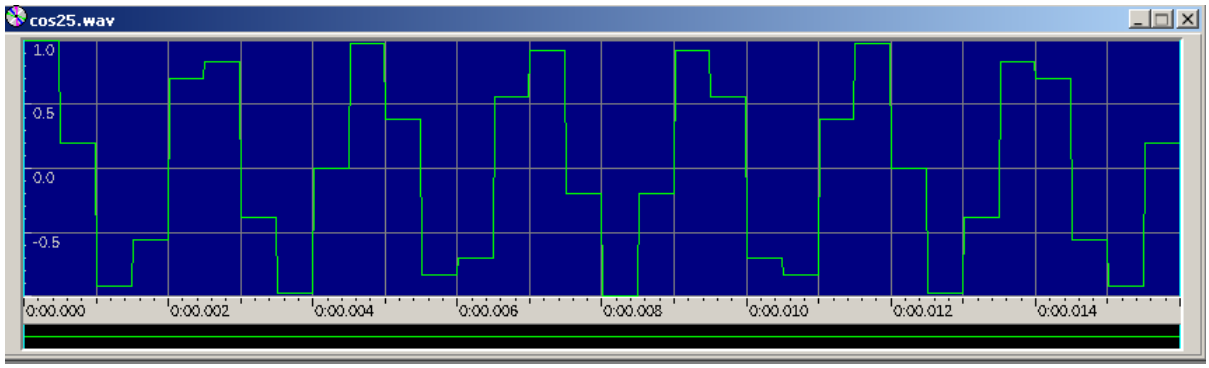

Ilustración 118 Coseno de 1562.5 Hz

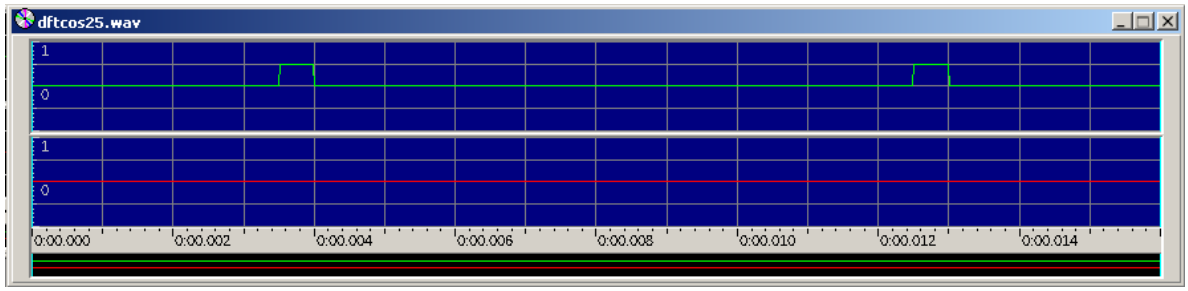

Ilustración 119 DFT del Coseno de 1562.5 Hz

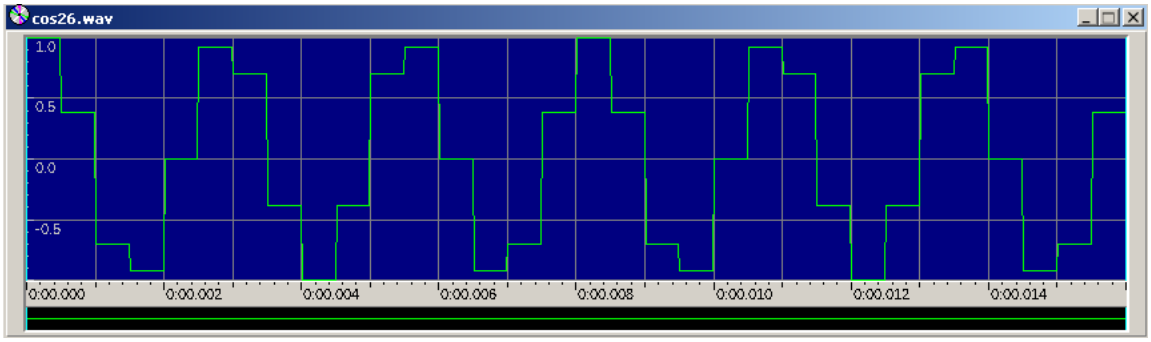

Ilustración 120 Coseno 1625 Hz

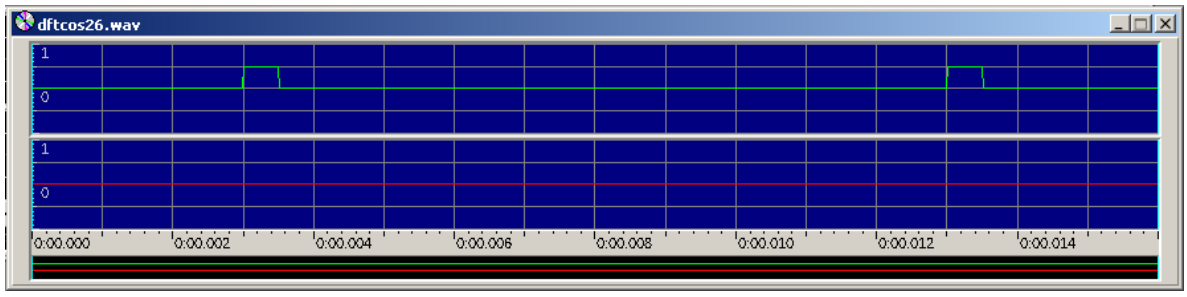

Ilustración 121 DFT del Coseno de 1625 Hz

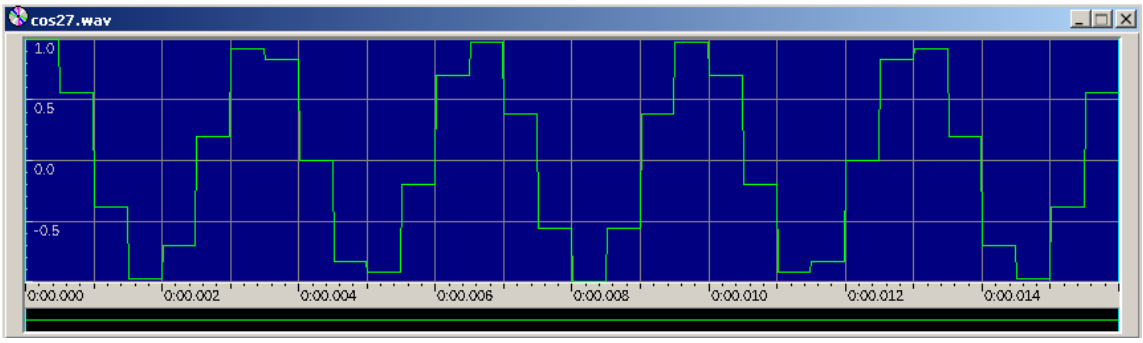

Ilustración 122 Coseno de 1687.5 Hz

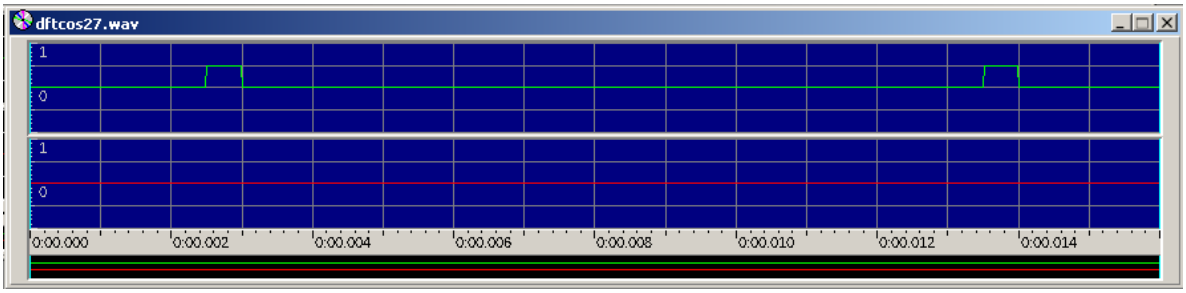

Ilustración 123 DFT del Coseno de 1687.5 Hz

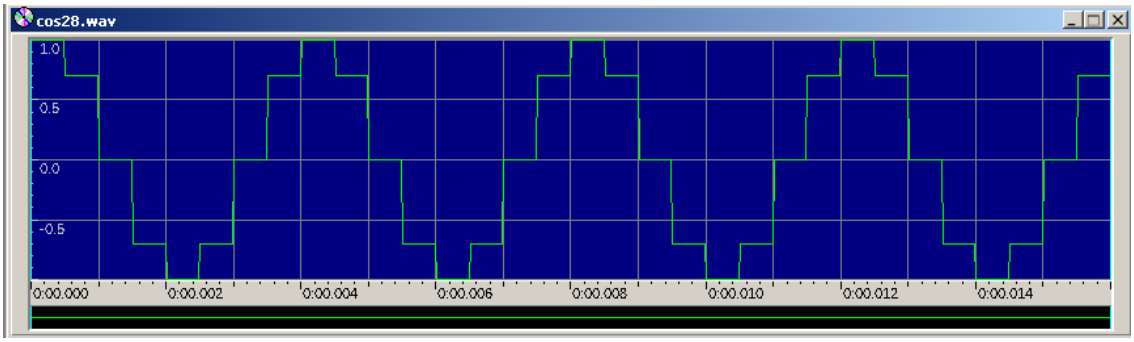

Ilustración 124 Coseno de 1750 Hz

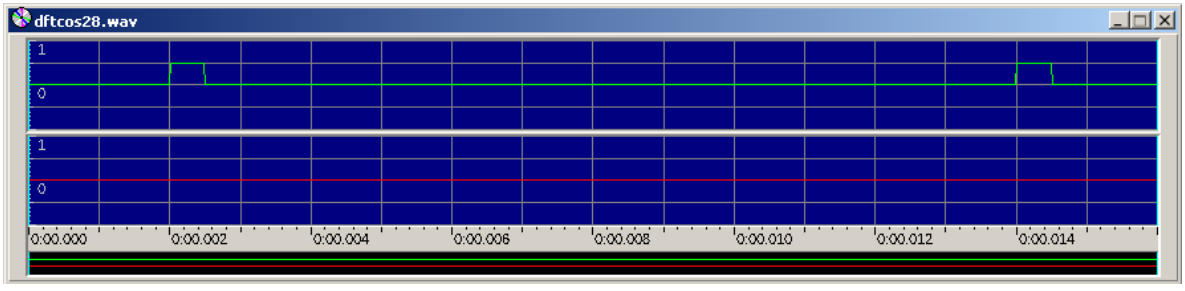

Ilustración 125 DFT del Coseno de 1750 Hz

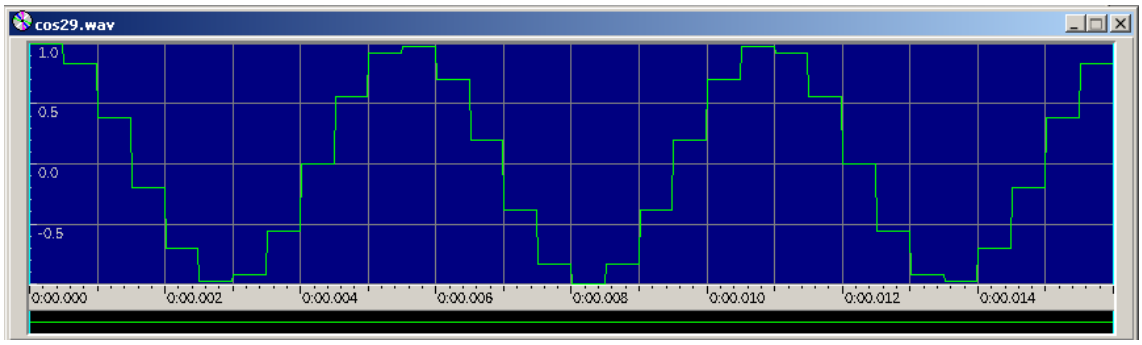

Ilustración 126 Coseno de 1812.5 Hz

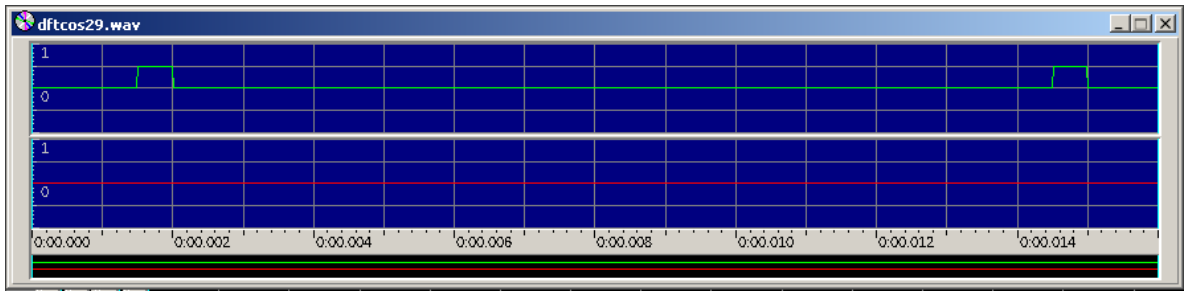

Ilustración 127 DFT del Coseno de 1812.5 Hz

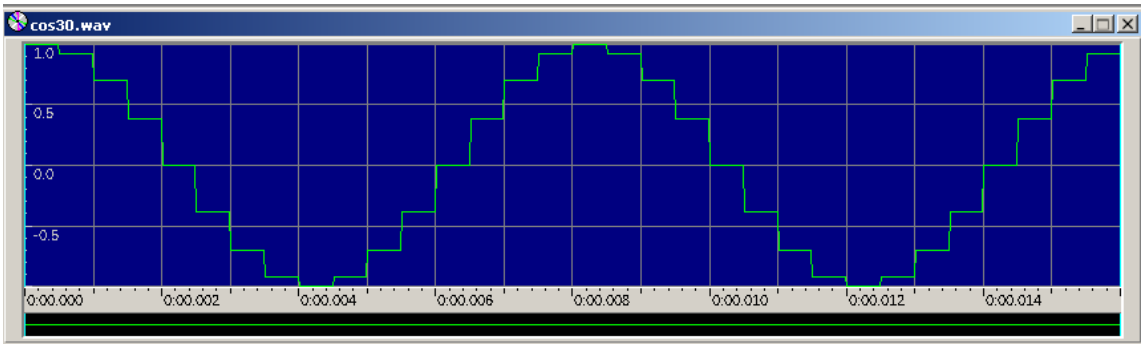

Ilustración 128 Coseno de 1875 Hz

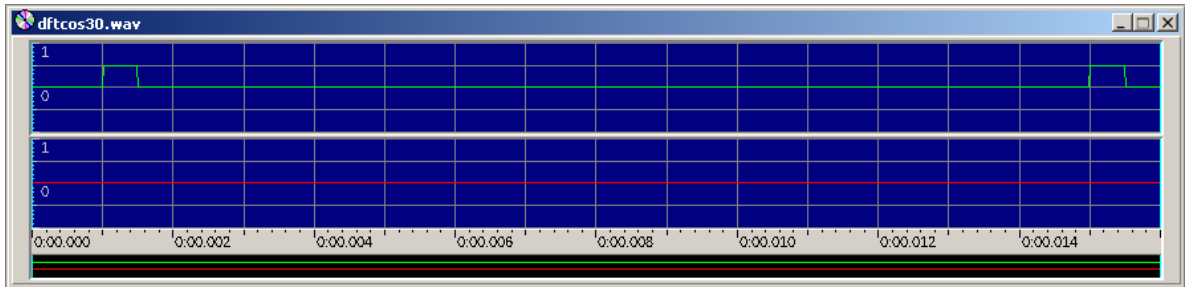

Ilustración 129 DFT del Coseno de 1875 Hz

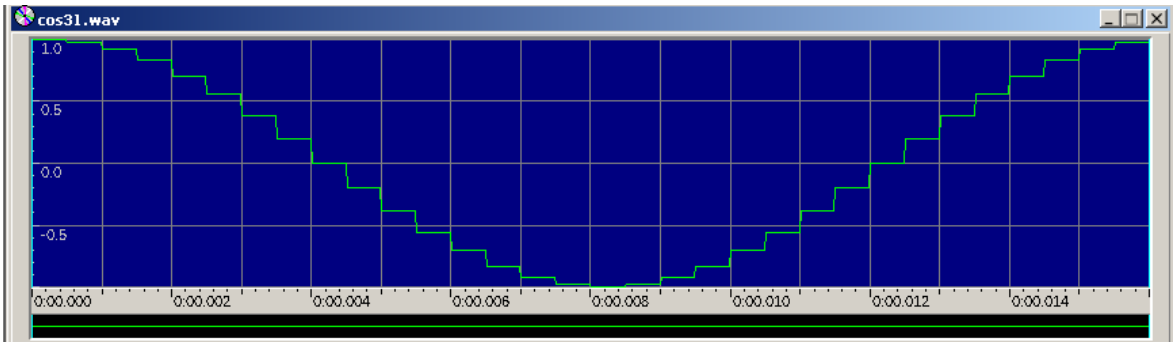

Ilustración 130 Coseno de 1937.5 Hz

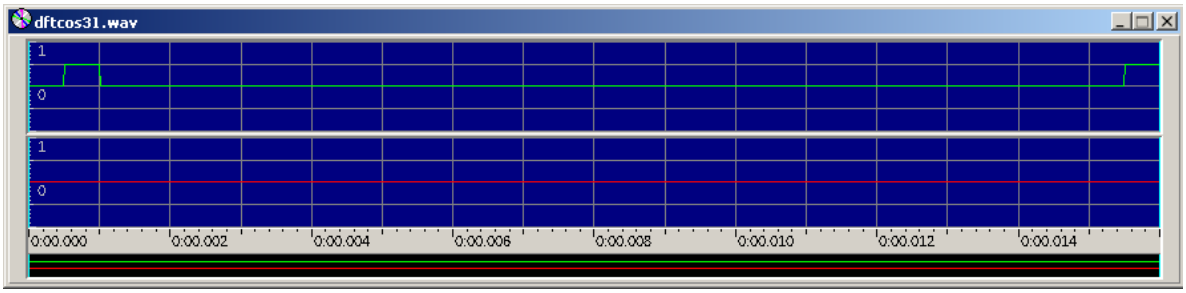

Ilustración 131 DFT del Coseno de 1937.5 Hz

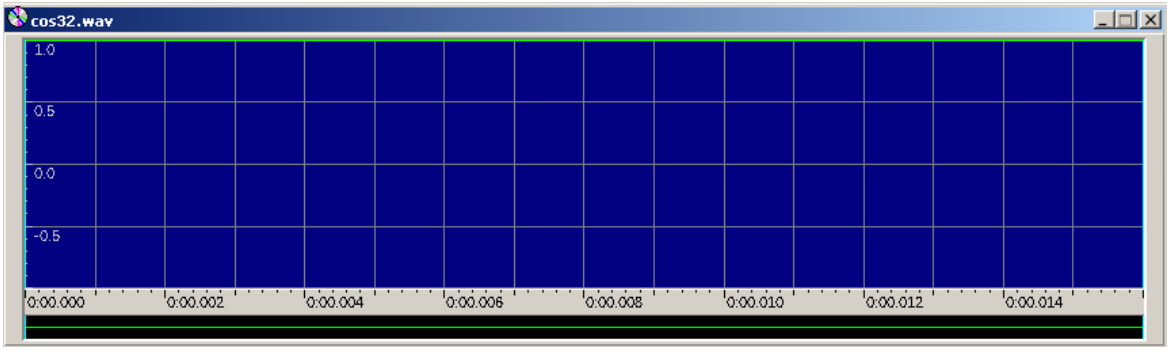

Ilustración 132 Coseno de 2000

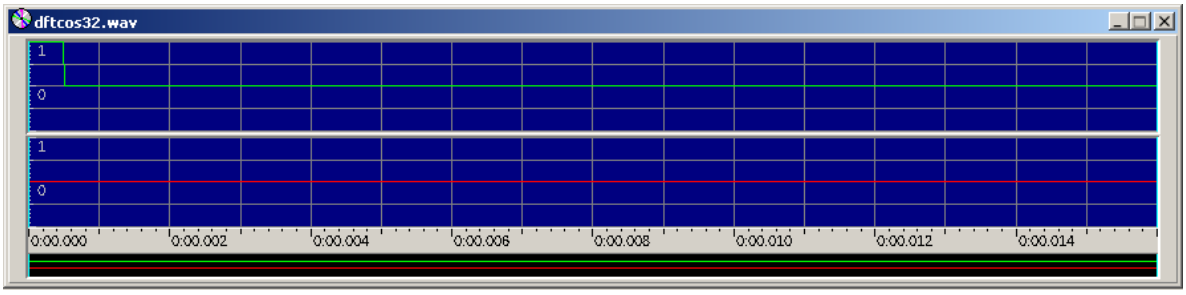

Ilustración 133 DFT del Coseno de 2000 Hz

Seno y coseno

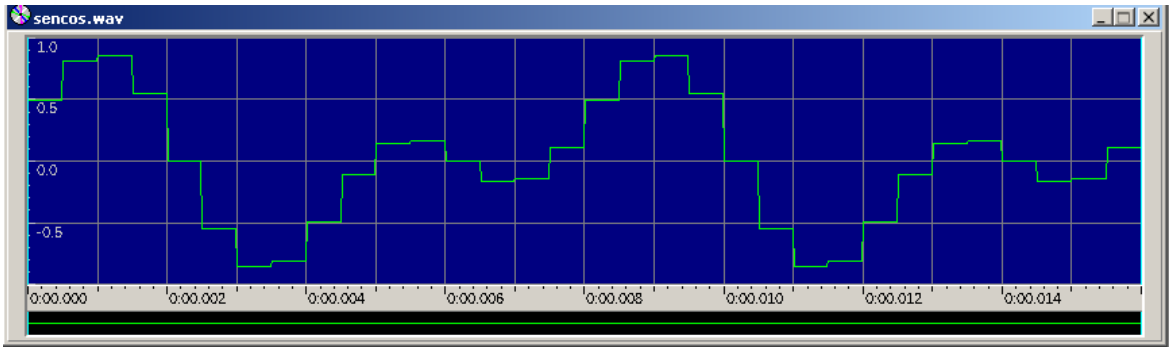

Ilustración 134 Coseno de 125 Hz más Seno de 250 Hz

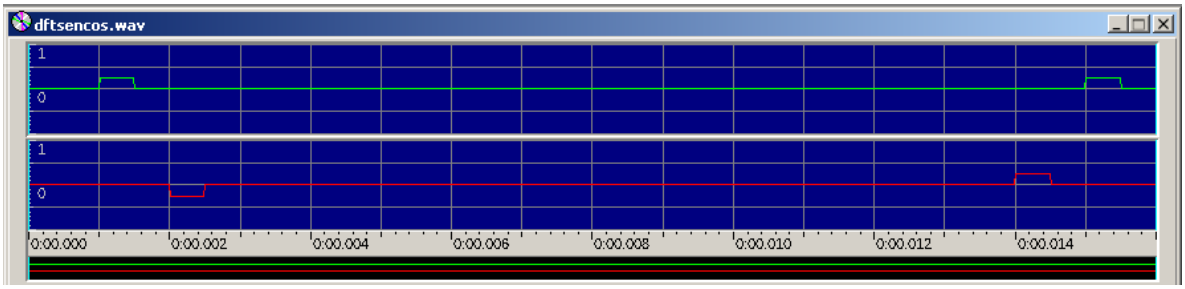

Ilustración 135 DFT del Coseno de 125 Hz más Seno de 250 Hz

Análisis Práctico

En este caso en particular se pidió un audio de 2000 muestras por segundo con una duración de 0.016 segundos. Por lo cual se tienen 32 muestras.

$$2000\ m/s\ x\ 0.016s = 32m$$

m: muestras

s: segundos

Cada muestra en la frecuencia representa 62.5 Hz

$$\frac{2000\ Hz}{32} = 62.5\ Hz$$

Nota: Frecuencias negativas iniciaran de la mitad de la señal resultante en frecuencia hacia adelante.

Como podemos notar el seno de 62.5 Hz (ilustración 5) está indicado en la segunda muestra, siempre la primera muestra será 0 en el seno, y la frecuencias negativas están indicadas de la mistad de la señal en adelante, en este caso en la última muestra. De igual manera en el coseno de 62.5 Hz (ilustración 71) queda indicado en la segunda muestra de la señal en frecuencia y su frecuencia negativa en la última muestra. Conforme avancemos en múltiplos de 62.5 Hz o de la frecuencia resultante de dividir la frecuencia de muestreo entre el número de muestras, avanzaremos en el número de muestra que se queda indicada en la frecuencia e iremos retrocediendo en las negativas.

Para el caso de las señales generadas con senos solo tendrán parte imaginaria en la frecuencia, y de manera análoga los cosenos solo en la parte positiva.

Como se muestra en la ilustración 133 cuando combinamos senos y cosenos, se indica en la parte real e imaginaria, en el número de muestra correspondiente, este número de muestra obedece a una regla de 3 que al simplificar obtenemos.

$$N_m = F_i * S$$

Donde:

$N_m = Número\ de\ muesta\ en\ frecuencia \qquad N_m = 0, \dots, N-1$

$$N_m < \frac{F_s}{2}$$

$F_i = Frecuencia\ de\ la\ señal\ en\ el\ tiempo$

$S = Tamaño\ de\ la\ señal\ en\ segundos$

$N = Número\ total\ de\ muestras$

$$F_s = Frecuencia\ de\ muestreo\ de\ señal\ en\ el\ tiempo$$

Conclusiones

De acuerdo a los resultados obtenidos se demuestra el teorema de muestreo, pues en el momento que las señales llegan a la mitad de la frecuencia de muestreo estas comienzan a repetirse produciendo el efecto alias.

Cuando se tienen señales múltiplos de una frecuencia obtenida de la división de Fs/N es trivial reconocerla en la frecuencia, por lo cual podemos decir que DFT es una base importante para el reconocimiento de patrones.

Código

wav.h

```
#include <stdio.h>
#include <stdlib.h>

typedef struct CabeceraWAV{
	unsigned char ChunkID[4];
	unsigned int ChunkSize;
	unsigned char Format[4];
	unsigned char Subchunk1_ID[4];
	unsigned char Subchunk1_Size[4];
	unsigned char AudioFormat[2];
	unsigned char NumChannels[2];
	unsigned int SampleRate;
	unsigned int ByteRate;
	unsigned short BlockAlign;
	unsigned short BitsPerSample;
	unsigned char Subchunk2_ID[4];
	unsigned int Subchunk2_Size;//Catidad de bytes de informacion de la
señal
}Cabecera;

void informacionAudio(Cabecera cabeceraAudio){
	printf("%c%c%c%c\n",cabeceraAudio.ChunkID[0],cabeceraAudio.ChunkID[
1],cabeceraAudio.ChunkID[2],cabeceraAudio.ChunkID[3]);
	printf("ChunkSize: %d\n",cabeceraAudio.ChunkSize);
	printf("%c%c%c%c\n",cabeceraAudio.Format[0],cabeceraAudio.Format[1]
,cabeceraAudio.Format[2],cabeceraAudio.Format[3]);
	printf("%c%c%c%c\n",cabeceraAudio.Subchunk1_ID[0],cabeceraAudio.Sub
chunk1_ID[1],cabeceraAudio.Subchunk1_ID[2],cabeceraAudio.Subchunk1_ID[3])
;
	printf("Subchunk1_Size:%d
%d%d%d\n",cabeceraAudio.Subchunk1_Size[0],cabeceraAudio.Subchunk1_Size[1]
,cabeceraAudio.Subchunk1_Size[2],cabeceraAudio.Subchunk1_Size[3]);
	printf("Formato del audio: %d\n",cabeceraAudio.AudioFormat[0] );
	printf("Numero de canales: %d\n",cabeceraAudio.NumChannels[0] );
	printf("Muestras por segundo %d\n",cabeceraAudio.SampleRate);
	printf("Bytes por segundo: %d\n",cabeceraAudio.ByteRate);
	printf("Numero de bytes por muestra %d\n",cabeceraAudio.BlockAlign
);
	printf("Numero de bits por muestra: %d\n",
cabeceraAudio.BitsPerSample);
	printf("%c%c%c%c\n",cabeceraAudio.Subchunk2_ID[0],cabeceraAudio.Sub
chunk2_ID[1],cabeceraAudio.Subchunk2_ID[2],cabeceraAudio.Subchunk2_ID[3])
;
	printf("Numero de bytes de la informacion
%d\n",cabeceraAudio.Subchunk2_Size);
	printf("Numero de muestras
%d\n",cabeceraAudio.Subchunk2_Size/(cabeceraAudio.NumChannels[0]*(cabecer
aAudio.BitsPerSample/8)));
	printf("Bytes de sobra (informaciondesconocida):
%d\n",cabeceraAudio.ChunkSize-36-cabeceraAudio.Subchunk2_Size);
	printf("Tam total del archivo: %d\n",cabeceraAudio.ChunkSize+8);
}
```

```
//Resegra el tamaño completo de el archivo en bytes
unsigned int getFileSize(Cabecera CW){
	return CW.ChunkSize+8;
}

//Regresa el tamaño de las muestras en bytes
char getSampleSize(Cabecera CW){
	return (CW.BitsPerSample/8);
}

//Regresa el numero de canales;
char getCannalNumber(Cabecera CW){
	return CW.NumChannels[0];
}

//Regresa el numero de muestras de audio
unsigned int getNumberAudioSamples(Cabecera CW){
	return CW.Subchunk2_Size/(CW.NumChannels[0]*(CW.BitsPerSample/8));
}

unsigned int getNumberBytesAudioInformation(Cabecera CW){
	return CW.Subchunk2_Size;
}

Cabecera setAudioStereo(Cabecera CW){
	CW.ChunkSize+=CW.Subchunk2_Size;
	CW.Subchunk2_Size*=2;
	CW.NumChannels[0]=2;
	CW.ByteRate*=2;
	CW.BlockAlign*=2;
	return CW;
}

//regresa el numero de bytes del pie del archivo wav
unsigned int getNumberBytesFoot(Cabecera CW){
	return CW.ChunkSize-36-CW.Subchunk2_Size;
}

//Regresa la cabecera del archvo wav
Cabecera getHeader(char *archivoWavEntrada){
	Cabecera CW;
	FILE *archivo=fopen(archivoWavEntrada,"r");
	fread(&CW,44,1,archivo);
	fclose(archivo);
	return CW;
}

char* getAudioSamples8bits(char *archivoWavEntrada){
	Cabecera CW;
	FILE *archivo=fopen(archivoWavEntrada,"r");
	fread(&CW,44,1,archivo);
	unsigned int
numeroMuestras=CW.Subchunk2_Size/(CW.NumChannels[0]*(CW.BitsPerSample/8))
;
	char *Samples=malloc(sizeof(char)*numeroMuestras);
	fread(Samples,numeroMuestras,1,archivo);
```

```
	fclose(archivo);
	return Samples;
}

short* getAudioSamples16bits(char *archivoWavEntrada){
	Cabecera CW;
	FILE *archivo=fopen(archivoWavEntrada,"r");
	fread(&CW,44,1,archivo);
	unsigned int
numeroMuestras=CW.Subchunk2_Size/(CW.NumChannels[0]*(CW.BitsPerSample/8))
;
	short *Samples=malloc(sizeof(short)*numeroMuestras);
	fread(Samples,numeroMuestras*2,1,archivo);
	fclose(archivo);
	return Samples;
}

char* getFileFoot(char *archivoWavEntrada){
	Cabecera CW;
	FILE *archivo=fopen(archivoWavEntrada,"r");
	fread(&CW,44,1,archivo);
	unsigned int bytesSobrantes=CW.ChunkSize-CW.Subchunk2_Size-36;
	char *bytes=malloc(sizeof(char)*CW.Subchunk2_Size);
	fread(bytes,CW.Subchunk2_Size,1,archivo);
	bytes=malloc(sizeof(char)*bytesSobrantes);
	fread(bytes,bytesSobrantes,1,archivo);
	return bytes;
}
```

DFT.c

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "wav.h"

#define PI 3.141592653589793

void main(int argc, char *argv[]){

	Cabecera CW;
	FILE *archivoWavEntrada=fopen(argv[1],"r");

	if(archivoWavEntrada==NULL){
		printf("Error en la lectura del archivo de audio\n");
	}else{

		FILE *archivoWavSalida=fopen(argv[2],"w");

		if(archivoWavSalida!=NULL){
			Cabecera CW=getHeader(argv[1]);
			CW=setAudioStereo(CW);
			fwrite(&CW,44,1,archivoWavSalida);
			short *muestrasAudio=getAudioSamples16bits(argv[1]);
			CW=getHeader(argv[1]);
			int numMuestrasAudio=getNumberAudioSamples(CW);
			short
*muestrasAudio2S=malloc(sizeof(short)*numMuestrasAudio*2);
			double
*muestrasAudio2=malloc(sizeof(double)*numMuestrasAudio*2);
			//FDT

			for(int k=0;k<numMuestrasAudio;k++){
				muestrasAudio2[2*k]=0;
				muestrasAudio2[2*k+1]=0;
				for(int n=0;n<numMuestrasAudio;n++){

	muestrasAudio2[2*k]+=muestrasAudio[n]*cos(2*PI*k*n/numMuestrasAudio
);
				}
				muestrasAudio2[2*k]/=numMuestrasAudio;
				for(int n=0;n<numMuestrasAudio;n++){

	muestrasAudio2[2*k+1]+=muestrasAudio[n]*sin(2*PI*k*n/numMuestrasAud
io);
				}
				muestrasAudio2[2*k+1]/=numMuestrasAudio;
				muestrasAudio2[2*k+1]*=-1;
			}

			for(int i=0;i<numMuestrasAudio*2;i++)
				muestrasAudio2S[i]=muestrasAudio2[i];

			unsigned int
bytesAudio=getNumberBytesAudioInformation(CW);
```

```
        fwrite(muestrasAudio2S,bytesAudio*2,1,archivoWavSalida);

                    char *pie=getFileFoot(argv[1]);
                    fwrite(pie,getNumberBytesFoot(CW),1,archivoWavSalida);
                    fclose(archivoWavSalida);
                    fclose(archivoWavEntrada);
              }
        }
}
```