

Nyola Sound Library  
1.0

Generated by Doxygen 1.8.3.1

Mon Oct 7 2013 23:23:16



# Contents

<b>1</b>	<b>File Index</b>	<b>1</b>
1.1	File List	1
<b>2</b>	<b>File Documentation</b>	<b>3</b>
2.1	cosft.cpp File Reference	3
2.1.1	Function Documentation	3
2.1.1.1	complete_arr	3
2.1.1.2	cosft	4
2.1.1.3	get_array_lenght	4
2.2	fft.cpp File Reference	4
2.2.1	Macro Definition Documentation	4
2.2.1.1	BACKWARD	4
2.2.1.2	FORWARD	4
2.2.1.3	PI	4
2.2.2	Function Documentation	4
2.2.2.1	bitrev	5
2.2.2.2	bits_length	5
2.2.2.3	fft	5
2.2.2.4	next_pow_2	5
2.3	fftwest.cpp File Reference	5
2.3.1	Function Documentation	5
2.3.1.1	fft_west	6
2.3.1.2	ifft_west	6
2.4	freqfilters.cpp File Reference	6
2.4.1	Macro Definition Documentation	6
2.4.1.1	PI	6
2.4.2	Function Documentation	6
2.4.2.1	bandpass	6
2.4.2.2	highpass	6
2.4.2.3	lowpass	7
2.5	noise.cpp File Reference	7

2.5.1	Function Documentation	7
2.5.1.1	whitenoise	7
2.6	ogg_vorbis.cpp File Reference	7
2.6.1	Macro Definition Documentation	8
2.6.1.1	READ	8
2.6.2	Function Documentation	8
2.6.2.1	decode	8
2.6.2.2	encoder	8
2.6.2.3	samples_number	9
2.6.2.4	samples_rate	9
2.6.2.5	total_time	9
2.7	pitch_filter.cpp File Reference	9
2.7.1	Macro Definition Documentation	10
2.7.1.1	PI	10
2.7.1.2	PITCH	10
2.7.1.3	UNPITCH	10
2.7.2	Function Documentation	10
2.7.2.1	pitch_filter	10
2.8	plot.cpp File Reference	10
2.8.1	Function Documentation	11
2.8.1.1	plot_freq	11
2.8.1.2	plot_time	11
2.9	sinft.cpp File Reference	11
2.9.1	Function Documentation	11
2.9.1.1	complete_arr	11
2.9.1.2	get_array_lenght	11
2.9.1.3	sinft	11

# Chapter 1

## File Index

### 1.1 File List

Here is a list of all files with brief descriptions:

<a href="#">cosft.cpp</a>	3
<a href="#">fft.cpp</a>	4
<a href="#">fftwest.cpp</a>	5
<a href="#">freqfilters.cpp</a>	6
<a href="#">noise.cpp</a>	7
<a href="#">ogg_vorbis.cpp</a>	7
<a href="#">pitch_filter.cpp</a>	9
<a href="#">plot.cpp</a>	10
<a href="#">sinft.cpp</a>	11



## Chapter 2

# File Documentation

### 2.1 cosft.cpp File Reference

```
#include <stdlib.h>
#include <iostream>
#include <math.h>
#include <complex>
#include <fftwest.hh>
```

#### Functions

- int [get\\_array\\_lenght](#) (const int size)
- void [complete\\_arr](#) (double \*arr\_in, double \*arr\_out, const int size)  
*This function fill an array power of two with other smaller or equal array and with zeros in extra spaces.*
- void [cosft](#) (double \*y, complex< double > \*rbuffer, const int size)  
*Is the function in charge to realize the cosine transform the one that you can obtain with equation  $f_j = \{k=0\}^{\wedge}\{n-1\}$   $x_k [\{n\} j (k+\{1\}\{2\}) ]\$$ .*

#### 2.1.1 Function Documentation

##### 2.1.1.1 complete\_arr ( double \* arr\_in, double \* arr\_out, const int size )

This function fill an array power of two with other smaller or equal array and with zeros in extra spaces.

##### Parameters

<i>*arr_in,:</i>	is a pointer towards the smallest array with which will fill the biggest array.
<i>*arr_out,:</i>	is a pointer towards the biggest array where is goint to get out the smaller array one time that is complete with the zeros.
<i>size,:</i>	is the size of the smaller array .

It is equivalent to the imaginary parts of a DFT of roughly twice the length, is a linear and invertible function

##### Parameters

<i>*arr_in,:</i>	is a pointer towards the smallest array with which will fill the biggest array.
<i>*arr_out,:</i>	is a pointer towards the biggest array where is goint to get out the smaller array one time that is complete with the zeros.
<i>size,:</i>	is the size of the smaller array .

### 2.1.1.2 `cosft ( double * y, complex< double > * rbuffer, const int size )`

Is the function in charge to realize the cosine transform the one that you can obtain with equation  $f_j = \{k=0\}^{n-1} x_k \{ \{n\} j (k+\{1\}{2}) \}$ .

this function is variation of the fast fourier transform, this gives the sum of cosine functions oscillating at different frequencies from lossy compression of audio and images

#### Parameters

<code>size,:</code>	is the size of the array.
<code>*y,:</code>	is a pointer towards an array that has to be of a size power of two.
<code>*rbuffer,:</code>	is a pointer towards an array where it is going to return the cosine transform into other array of complex numbers.

### 2.1.1.3 `int get_array_lenght ( const int size )`

## 2.2 `fft.cpp` File Reference

```
#include <stdlib.h>
#include <math.h>
#include <algorithm>
#include <complex>
```

### Macros

- `#define PI 3.141592654`
- `#define FORWARD 1`
- `#define BACKWARD -1;`

### Functions

- `int bits_length (int n)`  
*Calculates the necessary bits to store the array indexes.*
- `int bitrev (int num, int bitslength)`  
*The function attenuates only the highest frequencies to a determined frequency in a wave.*
- `int next_pow_2 (int size)`  
*calculates the next power of two .*
- `void fft (int smpls_read, double *buffer, complex< double > *transformed)`  
*The function calculated the fast fourier transform.*

### 2.2.1 Macro Definition Documentation

#### 2.2.1.1 `#define BACKWARD -1;`

#### 2.2.1.2 `#define FORWARD 1`

#### 2.2.1.3 `#define PI 3.141592654`

### 2.2.2 Function Documentation



2.2.2.1 `bitrev ( int num, int bitslength )`

The function attenuates only the highest frequencies to a determinated frequency in a wave.

## Parameters

<i>bitslength,:</i>	the number of bits.
<i>num:is</i>	the number you are looking for the bit reverse.

2.2.2.2 `bits_length ( int n )`

Calculates the necessary bits to store the array indexes.

## Parameters

<i>n,:</i>	is number tells how many bits (binary digits) are required to store that number in decimal
------------	--

2.2.2.3 `fft ( int smpls_read, double * buffer, complex< double > * transformed )`

The function calculated the fast fourier transform.

FFT requires the array length to be a power of two. The input array contains N complex samples, with real and imaginary part alternating, so the number of samples must be multiplied by two

## Parameters

<i>*buffer,:</i>	is a pointer towards the array whit the decoded song .
<i>smpls_num,:</i>	is the number of total samples read.
<i>*transformed,:</i>	is a pointer towards the array complex where send the result of the transform.

2.2.2.4 `next_pow_2 ( int size )`

calculates the next power of two .

## Parameters

<i>size</i>	: is the large of the array .
-------------	-------------------------------

2.3 `fftwest.cpp` File Reference

```
#include <stdlib.h>
#include <iostream>
#include <complex>
#include <fftw3.h>
```

## Functions

- void `fft_west` (int smpls\_num, double \*buffer, complex< double > \*retbuffer)
- void `ifft_west` (int smpls\_num, complex< double > \*buffer, double \*retbuffer)

## 2.3.1 Function Documentation

2.3.1.1 void `fft_west` ( int *smpls\_num*, double \* *buffer*, complex< double > \* *retbuffer* )

2.3.1.2 void `ifft_west` ( int *smpls\_num*, complex< double > \* *buffer*, double \* *retbuffer* )

## 2.4 freqfilters.cpp File Reference

### Macros

- #define `PI` 3.141592654

### Functions

- void `lowpass` (int *smpls\_num*, int *smpls\_rate*, double \**buffer*, double \**lowbuff*, double *freq*)  
The function attenuates only the highest frequencies to a determinated frequency in a wave.
- void `highpass` (int *smpls\_num*, int *smpls\_rate*, double \**buffer*, double \**highbuff*, double *freq*)  
The function attenuates only the lowest frequencies to a determinated frequency in a wave.
- void `bandpass` (int *smpls\_num*, int *smpls\_rate*, double \**buffer*, double \**bandbuff*, double *lowfreq*, double *highfreq*)  
The function in charge of attenuated frequencies outside of a given rack in a wave.

### 2.4.1 Macro Definition Documentation

2.4.1.1 #define `PI` 3.141592654

### 2.4.2 Function Documentation

2.4.2.1 `bandpass` ( int *smpls\_num*, int *smpls\_rate*, double \* *buffer*, double \* *bandbuff*, double *lowfreq*, double *highfreq* )

The function in charge of attenuated frequencies outside of a given rack in a wave.

#### Parameters

<i>smpls_num</i> ,:	is the number of total samples.
<i>smpls_rate</i> ,:	is the number of samples per second.
* <i>buffer</i> ,:	is a pointer towards the array whit the decoded song .
* <i>bandbuff</i> ,:	is a pointer towards an array where is going to send the attenuated frequencies.
<i>lowfreq</i> :	: is the is the minimun unattenuated frequency.
<i>highfreq</i> :	: is the is the maximun unattenuated frequency.

2.4.2.2 `highpass` ( int *smpls\_num*, int *smpls\_rate*, double \* *buffer*, double \* *highbuff*, double *freq* )

The function attenuates only the lowest frequencies to a determinated frequency in a wave.

#### Parameters

* <i>buffer</i> ,:	is a pointer towards the array whit the decoded song .
* <i>highbuff</i> ,:	is a pointer towards an array where is going to send the attenuated frequencies.
<i>freq</i> ,:	is the is the minimun unattenuated frequency.
<i>smpls_num</i> ,:	es el tamaño del array buffer
<i>smpls_rate</i> ,:	is the number of samples per second.

### 2.4.2.3 lowpass ( int *smpls\_num*, int *smpls\_rate*, double \* *buffer*, double \* *lowbuff*, double *freq* )

The function attenuates only the highest frequencies to a determinated frequency in a wave.

#### Parameters

<i>*buffer,:</i>	is a pointer towards the array whit the decoded song .
<i>*lowbuff,:</i>	is a pointer towards an array where is going to send the attenuated frequencies.
<i>freq,:</i>	is the is the maximun unattenuated frequency.
<i>smpls_num,:</i>	is the number of total samples.
<i>smpls_rate,:</i>	is the number of samples per second.

## 2.5 noise.cpp File Reference

```
#include <math.h>
#include <stdlib.h>
#include <time.h>
#include <iostream>
```

### Functions

- void [whitenoise](#) (int *smpls\_num*, double \**buffer*)

*The function fill a pointer with aleatoriis random numbers which when they join the song they can be seen like noisy.*

### 2.5.1 Function Documentation

#### 2.5.1.1 whitenoise ( int *smpls\_num*, double \* *buffer* )

The function fill a pointer with aleatoriis random numbers which when they join the song they can be seen like noisy.

#### Parameters

<i>*buffer,:</i>	is a pointer towards an array where is going to return the ramdon numbers.
<i>smpls_num,:</i>	is the number of total samples and is equal to size of buffer.

## 2.6 ogg\_vorbis.cpp File Reference

```
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include <time.h>
#include <iostream>
#include <vorbis/vorbisenc.h>
#include <vorbis/vorbisfile.h>
```

### Macros

- #define [READ](#) 1000000

## Functions

- int [samples\\_number](#) (const char \*fname)  
The function receives a pointer towards to the name of the archive thar has to be .ogg and return the number of tatal samples.
- double [total\\_time](#) (const char \*fname)  
The function receives a pointer towards to the name of the archive thar has to be .ogg and return the time of the song in seconds.
- double [samples\\_rate](#) (const char \*fname)  
The function receives a pointer towards to the name of the archive thar has to be .ogg and return the samples for second.
- long [decode](#) (const char \*fname, double \*buffer, int smpls\_num)  
The functiondecodes the song in a real array and send it to buffer this need the file are monochannel sound and be a .ogg.
- void [encoder](#) (const char \*outfilename, double \*inbuffer, int smpls\_num, double smpls\_rate)  
The function is the inverce to decoder this convert a array in a file .ogg using pointers.

### 2.6.1 Macro Definition Documentation

#### 2.6.1.1 #define READ 1000000

### 2.6.2 Function Documentation

#### 2.6.2.1 `decode ( const char * fname, double * buffer, int smpls_num )`

The functiondecodes the song in a real array and send it to buffer this need the file are monochannel sound and be a .ogg.

##### Parameters

<i>*buffer</i>	: is a pointer towards array where de decoder song are sent.
<i>*fname</i>	: is a pointer towards the name and direction of the song.
<i>smpls_num</i>	: is the number of total samples and is equal to size of buffer.

##### Returns

the number of samples read

#### 2.6.2.2 `encoder ( const char * outfilename, double * inbuffer, int smpls_num, double smpls_rate )`

The function is the inverce to decoder this convert a array in a file .ogg using pointers.

##### Parameters

<i>*outfilename,:</i>	is pointer toward the first letter to file where send the array once encoder
<i>*inbuffer,:</i>	is a pointer to the array whit the song decoder or the file you want to convert.
<i>*fname</i>	: is a pointer towards the fist letter to the archive song.
<i>smpls_num</i>	: is the number of total samples and is equal to size of buffer.
<i>smpls_rate</i>	: is the number of samples per second.

The function the function choose a centroid and does window as a bell of Hann,whit this one values all the points ariund the centroid and does the point if greatest amplitude pitch and the point of less aplitude unpitc,with this made groups around,start to calculate the news centroids using f2 and f3,repeat the proces until the centroids aren't moving or until 100 iterations, chages the size of the window exponentially dependin of the increment. is chosen the size of the window with more distance between the centroids

## Parameters

<i>*buffer,:</i>	is a pointer towards array where de decoder song is saves.
<i>percent,:</i>	is percent od samples per second.
<i>increment,:</i>	is the increment and is given exponentially.
<i>smpls_num,:</i>	is the number of total samples and is equal to size of buffer.
<i>smpls_rate,:</i>	is the number of samples per second.

## 2.6.2.3 samples\_number ( const char \* fname )

The function receives a pointer towards to the name of the archive thar has to be .ogg and return the number of tatal samples.

## Parameters

<i>*fname,:</i>	is a pointer towards the name and direction of the song.
-----------------	--

## Returns

The number of samples in total song.

## 2.6.2.4 samples\_rate ( const char \* fname )

The function receives a pointer towards to the name of the archive thar has to be .ogg and return the samples for second.

## Parameters

<i>*fname</i>	: is a pointer towards the name and direction of the song.
---------------	--

## Returns

The samples for second of the song.

## 2.6.2.5 total\_time ( const char \* fname )

The function receives a pointer towards to the name of the archive thar has to be .ogg and return the time of the song in seconds.

## Parameters

<i>*fname</i>	: is a pointer towards the name and direction of the song.
---------------	--

## Returns

The total time in seconds song.

## 2.7 pitch\_filter.cpp File Reference

```
#include <stdlib.h>
#include <math.h>
#include <stdio.h>
#include <cmath>
#include <iostream>
```

## Macros

- #define `PI` 3.141592654
- #define `PITCH` 1
- #define `UNPITCH` 0

## Functions

- void `pitch_filter` (int `smpls_num`, int `smpls_rate`, double `*inbuffer`, double `percent`, double `increment`)  
*The function receives a pointer to real array and with their generates a graphyc respect to time.*

### 2.7.1 Macro Definition Documentation

2.7.1.1 #define `PI` 3.141592654

2.7.1.2 #define `PITCH` 1

2.7.1.3 #define `UNPITCH` 0

### 2.7.2 Function Documentation

2.7.2.1 `pitch_filter` ( int *smpls\_num*, int *smpls\_rate*, double \* *inbuffer*, double *percent*, double *increment* )

The function receives a pointer to real array and with their generates a graphyc respect to time.

The function receives a pointer towards an array of complex numbers and with their generates a graphyc respect to the frequency.

#### Parameters

<i>*buffer</i> ,:	is a pointer towards the array whit the decoded song .
<i>smpls_read</i>	is the number of samples read .
<i>smpls_rate</i>	is the number of samples per second.
<i>*buffer</i>	is a pointer towards the array whit the decoded song .
<i>smpls_read</i>	is the number of samples read .
<i>smpls_rate</i>	is the number of samples per second.

## 2.8 plot.cpp File Reference

```
#include <iostream>
#include <stdlib.h>
#include <fstream>
#include <complex>
```

## Functions

- void `plot_time` (double `*buffer`, int `smpls_read`, double `smpls_rate`)
- void `plot_freq` (complex< double > `*buffer`, int `smpls_read`, double `smpls_rate`)

## 2.8.1 Function Documentation

2.8.1.1 void plot\_freq ( complex< double > \* *buffer*, int *smpls\_read*, double *smpls\_rate* )

2.8.1.2 void plot\_time ( double \* *buffer*, int *smpls\_read*, double *smpls\_rate* )

## 2.9 sinft.cpp File Reference

```
#include <stdlib.h>
#include <iostream>
#include <math.h>
#include <complex>
#include <fftwest.hh>
```

### Functions

- int [get\\_array\\_lenght](#) (const int size)
- void [complete\\_arr](#) (double \*arr\_in, double \*arr\_out, const int size)
- void [sinft](#) (double \*y, complex< double > \*rbuffer, const int size)  
*Is the function in charge to realize the sine transform the one that you can obtain with equatio  $F_k = \{j=1\}^{N-1} f_j \sin(j k/N)$*

## 2.9.1 Function Documentation

2.9.1.1 void complete\_arr ( double \* *arr\_in*, double \* *arr\_out*, const int *size* )

2.9.1.2 int get\_array\_lenght ( const int *size* )

2.9.1.3 sinft ( double \* *y*, complex< double > \* *rbuffer*, const int *size* )

Is the function in charge to realize the sine transform the one that you can obtain with equatio  $F_k = \{j=1\}^{N-1} f_j \sin(j k/N)$

### Parameters

<i>size</i> ,:	is the size of the array.
* <i>y</i> ,:	is a pointer towards an array that has to be of a size power of two.
* <i>rbuffer</i> ,:	is a pointer towards an array where is going to return the cosine transform into other array of complex numbers.

# Index

## BACKWARD

fft.cpp, 4

## bandpass

freqfilters.cpp, 6

## bitrev

fft.cpp, 4

## bits\_length

fft.cpp, 5

## complete\_arr

cosft.cpp, 3

sinft.cpp, 11

## cosft

cosft.cpp, 3

## cosft.cpp, 3

complete\_arr, 3

cosft, 3

get\_array\_lenght, 4

## decode

ogg\_vorbis.cpp, 8

## encoder

ogg\_vorbis.cpp, 8

## FORWARD

fft.cpp, 4

## fft

fft.cpp, 5

## fft.cpp, 4

BACKWARD, 4

bitrev, 4

bits\_length, 5

FORWARD, 4

fft, 5

next\_pow\_2, 5

PI, 4

## fft\_west

fftwest.cpp, 5

## fftwest.cpp, 5

fft\_west, 5

ifft\_west, 6

## freqfilters.cpp, 6

bandpass, 6

highpass, 6

lowpass, 6

PI, 6

## get\_array\_lenght

cosft.cpp, 4

sinft.cpp, 11

## highpass

freqfilters.cpp, 6

## ifft\_west

fftwest.cpp, 6

## lowpass

freqfilters.cpp, 6

## next\_pow\_2

fft.cpp, 5

## noise.cpp, 7

whitenoise, 7

## ogg\_vorbis.cpp, 7

decode, 8

encoder, 8

READ, 8

samples\_number, 9

samples\_rate, 9

total\_time, 9

## PI

fft.cpp, 4

freqfilters.cpp, 6

pitch\_filter.cpp, 10

## PITCH

pitch\_filter.cpp, 10

## pitch\_filter

pitch\_filter.cpp, 10

## pitch\_filter.cpp, 9

PI, 10

PITCH, 10

pitch\_filter, 10

UNPITCH, 10

## plot.cpp, 10

plot\_freq, 11

plot\_time, 11

## plot\_freq

plot.cpp, 11

## plot\_time

plot.cpp, 11

## READ

ogg\_vorbis.cpp, 8

## samples\_number

ogg\_vorbis.cpp, 9

## samples\_rate

ogg\_vorbis.cpp, 9

## sinft



- sinft.cpp, [11](#)
- sinft.cpp, [11](#)
  - complete\_arr, [11](#)
  - get\_array\_lenght, [11](#)
  - sinft, [11](#)
- total\_time
  - ogg\_vorbis.cpp, [9](#)
- UNPITCH
  - pitch\_filter.cpp, [10](#)
- whitenoise
  - noise.cpp, [7](#)