C++ Code: (Used KissFFT with Visual Studio)

#include<iostream>

#include "dspf.hpp"

extern "C" {

#include "kiss\_fft.h"

}

kiss\_fft\_cpx\* alloc(const int n) { return (kiss\_fft\_cpx\*)KISS\_FFT\_MALLOC(n \* sizeof(kiss\_fft\_cpx)); }

void kiss\_fft\_copycpx(float\* r, kiss\_fft\_cpx\* c, const int n) { for (int i = 0; i < n; ++i) { c[i].r = r[i]; c[i].i = 0; } }

void clean\_noise(DSPFile& fin, DSPFile& fout) {

const int nfft = 256;

kiss\_fft\_cfg fft = kiss\_fft\_alloc(nfft, 0, NULL, NULL);

kiss\_fft\_cfg ifft = kiss\_fft\_alloc(nfft, 1, NULL, NULL);

kiss\_fft\_cpx \*x = alloc(nfft), \*X = alloc(nfft),

\*y = alloc(nfft), \*Y = alloc(nfft);

// Copy the header to the output file

fout.Header = fin.Header;

fout.write\_h();

// algorithm variables

int step = nfft / 2, remainder;

float lam = 0.999, lam2 = 1.0 - lam;

float

temp[nfft] = { 0 },

Px[nfft] = { 0 },

out[nfft] = { 0 };

// processing

float Xm;

fin.read\_n(temp, nfft); kiss\_fft\_copycpx(temp, x, nfft);

for (int n = nfft - 1; n < fin.Header.dim0; n += step) {

// Compute FFT

kiss\_fft(fft, x, X);

for (int i = 0; i < nfft; ++i) {

float Mag = (X[i].r \* X[i].r) + (X[i].i \* X[i].i);

Px[i] = lam \* Px[i] + (lam2 \* Mag);

Y[i] = Mag >= 10 \* Px[i] ? X[i] : kiss\_fft\_cpx{0, 0};

}

// Compute IFFT \* nfft

kiss\_fft(ifft, Y, y);

for (int i = 0; i < nfft; ++i) {

//y[i].r = x[i].r;

// overlap with output buffer

out[i] += 0.5 \* y[i].r / nfft;

}

fout.write\_d(out, step);

// Shift buffers

for (int i = 0; i < step; ++i) {

temp[i] = temp[i + step];

out[i] = out[i + step];

out[i + step] = 0;

}

remainder = fin.read\_n(temp + step, step); kiss\_fft\_copycpx(temp, x, nfft);

}

fout.write\_d(out, step + remainder); // Dr. Gunther's code is wrong...it's missing this

// Deallocate memory

free(fft);

free(ifft);

kiss\_fft\_cleanup();

}

int main() {

std::string

in = "output\\harry8noise.bin",

out = "output\\harry8.bin";

DSPFile fin(in), fout(out, DSP::Mode::Write);

clean\_noise(fin, fout);

system("pause");

return 0;

}

Matlab Code:

clear all**;**

**[**x**,** fs**]** **=** audio2bin**(**'harry8noise.wav'**);**

**[**y**]** **=** bin2audio**(**'harry8.bin'**);**

% helpers for plot axises

t **=** **(**0**:**size**(**x**,** 1**)** **-** 1**)/**fs**;**

tf **=** **(**size**(**y**,** 1**)** **-** 1**)/**fs**;**

% Plot the signals

subplot**(**2**,** 2**,** 1**);**

plot**(**t**,** x**);**

xlim**([**0 tf**]);**

title**(**'Before Spectral Subtraction'**);**

subplot**(**2**,** 2**,** 2**);**

plot**(**t**,** y**);**

xlim**([**0 tf**]);**

title**(**'After Spectral Subtraction'**);**

% Plot the spectrograms

subplot**(**2**,** 2**,** 3**);**

plot\_spectrogram**(**x**,** 10**,** fs**);**

subplot**(**2**,** 2**,** 4**);**

plot\_spectrogram**(**y**,** 10**,** fs**);**

Plots:

