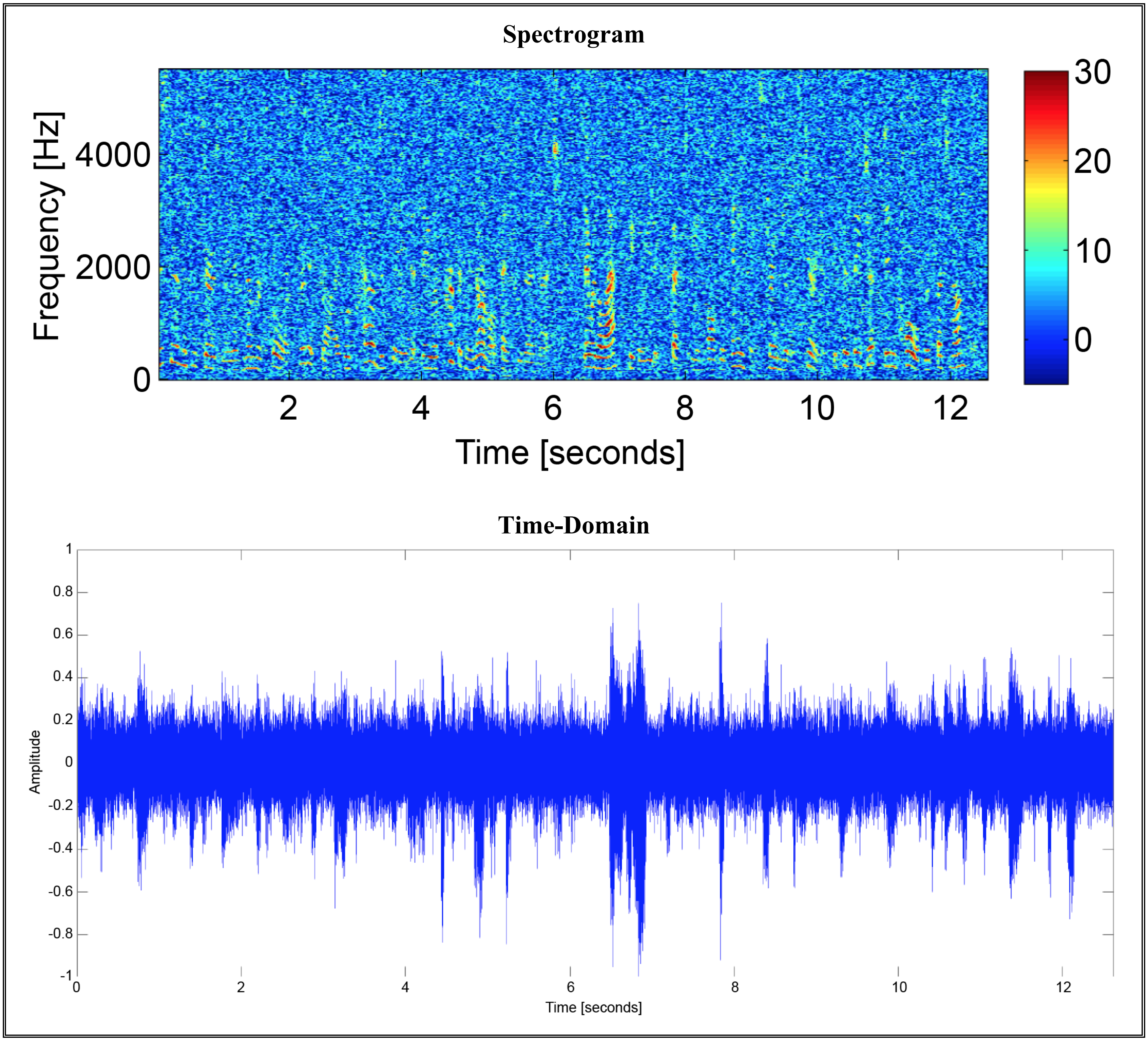
Lab 5 – Spectral Subtraction

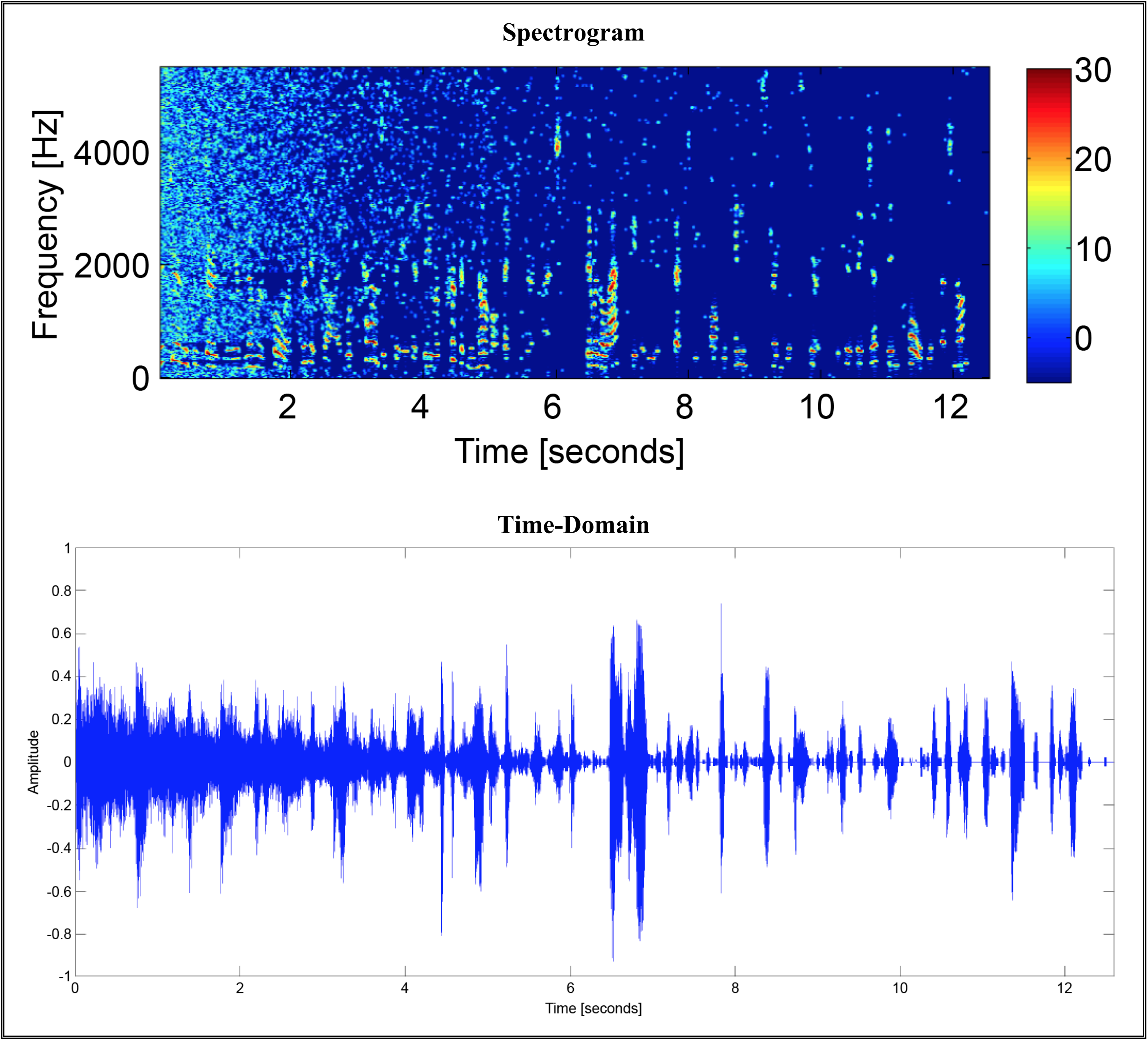
C Program

```c
#include <stdio.h>
#include <stdlib.h>
#include <fftw3.h>

// This code uses single precision floating point arrays.
// FFTW must be configured to use floats by typing:
// ./configure --enable-float && make
// sudo make install
//
// To compile at the command line:
// gcc -o fftwf_test fftwf_test.c -lfftw3 -I/usr/local/include -L/usr/local/lib

typedef struct
{
        int ndim;
        // signal (1)
        // image  (2)
        // video  (3)
        int nchan;
        // signal (1=mono, 2=stereo, etc.)
        // grayscale image/video (1)
        // color image/video (3)
        int dim0;
        // signal => length
        // image or video => number rows
        int dim1;
        // signal => 0
        // image or video => number columns
        int dim2;
        // signal => 0
        // image  => 0
        // video  => number_frames
} dsp_file_header;

int main(int argc, char *argv[])
{
        // FFTW variables
        fftwf_plan p_fft, p_ifft;
        int nfft = 256;
        float *x, *y;
        fftwf_complex *X, *Y;
        x = (float*)calloc(sizeof(float),nfft);
        y = (float*)calloc(sizeof(float),nfft);
        X = (fftwf_complex*)fftwf_malloc(sizeof(fftwf_complex)*nfft);
        Y = (fftwf_complex*)fftwf_malloc(sizeof(fftwf_complex)*nfft);
        p_fft  = fftwf_plan_dft_r2c_1d(nfft, x, X, FFTW_MEASURE);
        p_ifft = fftwf_plan_dft_c2r_1d(nfft, Y, y, FFTW_MEASURE);

        // alogorithm variables
        int step = nfft/2;
        float lam1 = 0.999;
        float lam2 = 1.0 - lam1;
        float *Px = (float*)calloc(sizeof(float),nfft);
        float *out = (float*)calloc(sizeof(float),nfft);

        // files and I/O buffers
        dsp_file_header h;
        FILE *fin  = fopen("harry8noise.bin","rb");
        FILE *fout = fopen("harry8noise_canceled.bin","wb");
        fread(&h,sizeof(dsp_file_header),1,fin);
        fwrite(&h,sizeof(dsp_file_header),1,fout);
        // processing
        float Xm;
        int n, i;
        fread(x,sizeof(float),nfft,fin);
        for (n=nfft-1; n<h.dim0; n+=step)
        {
                // compute FFT
                fftwf_execute(p_fft); // X = fft(x)

                // copy FFTs
                for (i=0; i<nfft; i++)
                {
                        Y[i][0] = X[i][0];
                        Y[i][1] = X[i][1];
                }

                // compute IFFT
                fftwf_execute(p_ifft); // y = ifft(Y)

                // overlap and add in output buffer
                float s = 0.5/nfft;
                for (i=0; i<nfft; i++)
                {
                        out[i] += s*y[i];
                }

                // write out data
                fwrite(out,sizeof(float),step,fout);

                // shift input and output buffers
                for (i=0; i<step; i++)
                {
                        x[i] = x[i+step];
                        out[i] = out[i+step];
                        out[i+step] = 0.0;
                }
                // read in next chunk
                fread(x+step,sizeof(float),step,fin);
        }

        // close files and free algorithm variables
        free(Px);
        free(out);
        fclose(fin);
        fclose(fout);

        // free FFTW variables
        fftwf_destroy_plan(p_fft);
        fftwf_destroy_plan(p_ifft);
        fftwf_free(X);
        fftwf_free(Y);
        free(x);
        free(y);

        return 1;
}
```

Noisy Signal



Clean Signal


Spectrogram
Frequency [Hz]
Time [seconds]
Time-Domain
Amplitude
Time [seconds]