A4 RECORDING – JULY 7TH 2022

5 files:

1. A4_Before_Packing.txt → raw byte data from audio library.

2. A4_After_Packing.txt → byte data repacked into proper format (repacked using C#).

3. A4_Autocorrelated_Frequency → applying autocorrelation on "A4_After_Packing.txt"

4. corrected.mat → provided by Professor Boyd for Comparison

5. corrected2.txt → Inserting "A4_Before_Packing.txt" into an online MatLab IDE, copying the first ~1000 values. (repacked using MatLab)

NAUDIO

NAudio is the library being used to record audio in C# - we use C# so that we can directly import the library into Unity for the game.

https://github.com/naudio/NAudio/blob/master/Docs/RecordWavFileWinFormsWaveIn.md ← Helpful example

https://github.com/naudio/NAudio/tree/master/Docs <-- Master Documentation.

NAUDIO - WAVEIN

```
//Making an Naudio wave in event!
WaveInEvent wave = new WaveInEvent(){
    DeviceNumber = deviceNum,
    WaveFormat = new WaveFormat(sampleRate, bitDepth, channelCount),
    BufferMilliseconds = BufferMiliseconds
};
```

Device number → recording device to use

WaveFormat → Shown Below

BufferMilliseconds → The speed at which we collect data, currently set to 20ms such that we get 50 new buffers/chunks every second.

NAUDIO - WAVEFORMAT

Docs → https://github.com/SjB/NAudio/blob/master/NAudio/Wave/WaveFormats/WaveFormat.cs

This is the way we setup the format of the input audio.

Wave format takes three parameters:

1. sampleRate → 44100hz
2. bitDepth → 16
3. channelCount → 1

ONLINE MATLAB SETUP (FOR REPACKING)

Used an online IDE: https://www.jdoodle.com/execute-octave-matlab-online/

```
% xRaw has 8-bit values that must be packed into 16-bit 2s comp
xRaw = load( '/uploads/A4_Before_Packing.txt' );
% pack 8 bit values into 16 bit
xLo = xRaw(1:2:end);
xHi = xRaw(2:2:end);
x = xHi * 256 + xLo;
% adjust for 2s complement
x( x >= 32768 ) = x( x >= 32768 ) - 65536;
x = x / 32768;

disp(x)
```

C# SETUP (FOR REPACKING)

```
for(int i = 0; i < xRaw.Length/2; i++){
    //val = xhigh * 256 + xlo
    //values[0] = xRaw[1] * 256 + xRaw[0] --> i = 0
    //values[1] = xRaw[3] * 256 + xRaw[2] --> i = 1
    //values[2] = xRaw[5] * 256 + xRaw[4] --> i = 2 // this has type Int32
    values[i] = xRaw[(i*2) + 1] * 256 + xRaw[i*2];
}

for(int i = 0; i < values.Length; i++){
    if(values[i] >= 32768){
        values[i] = values[i] - 65536;
    }
}
for(int i = 0; i < values.Length; i++){

    values[i] = values[i] / 32768;

}

// for(int i = 0; i < values.Length; i++){
//     values[i] = values[i] / target_freq;
// }
```

- xRaw is a direct copy of e.Buffer, the raw byte data from the audio input library
- The commented out for loop was an attempt at replicating the t = ((1:length(x)) – 1) / fSample;
- Not exactly sure what fSample represents.
- This method is a replication of the matlab method provided by Professor Boyd.

C# SETUP (FOR AUTOCORRELATION)

```
float sum_old;
float sum = 0.0f;
float thresh = 0f;

int pd_state = 0;
int period = 0;
//Autocorrelation
for(int i = 0; i < values.Length; i++){ //Needs to change..

    sum_old = sum;
    sum = 0.0f;

    for(int k = 0; k < values.Length-i; k++){
        sum+= (float)((values[k]) * (values[k+i]));
    }

    if(pd_state == 2 && (sum-sum_old) <=0){
        period = i-1;
        pd_state = 3;
    }
    if(pd_state == 1 && (sum > thresh) && (sum-sum_old) > 0 ){
        pd_state = 2;
    }
    if(pd_state == 0){

        thresh = sum * 0.5f;
        pd_state = 1;
    }
}
freq_per = sample_freq/period; //Current Freq in Hertz
```

- Values contains the repacked values:
- E.buffer → xRaw → values
- This method was created using the article on autocorrelation shown by Professor Boyd.
- Uses a state machine to determine peaks and obtain frequency.
- Sample_Frequency = 44100hz

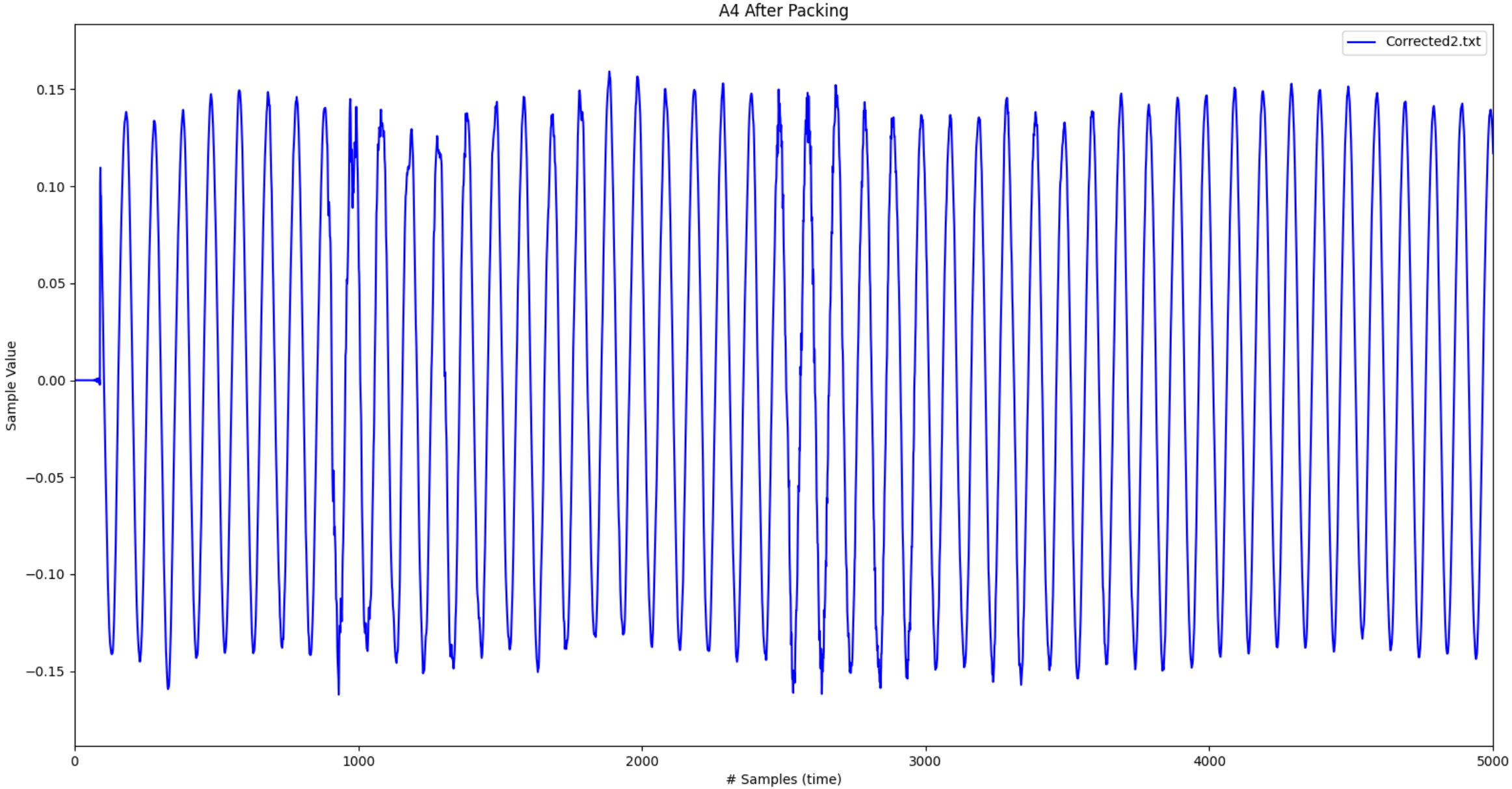
A4 AFTER PACKING GRAPH

This data was obtained by feeding “A4_Before_Packing.txt” into the previously mentioned C# Setup For Repacking.

The red line is the mean / average line of all y values.

Seems to very closely represent a sin wave.

Zoomed in

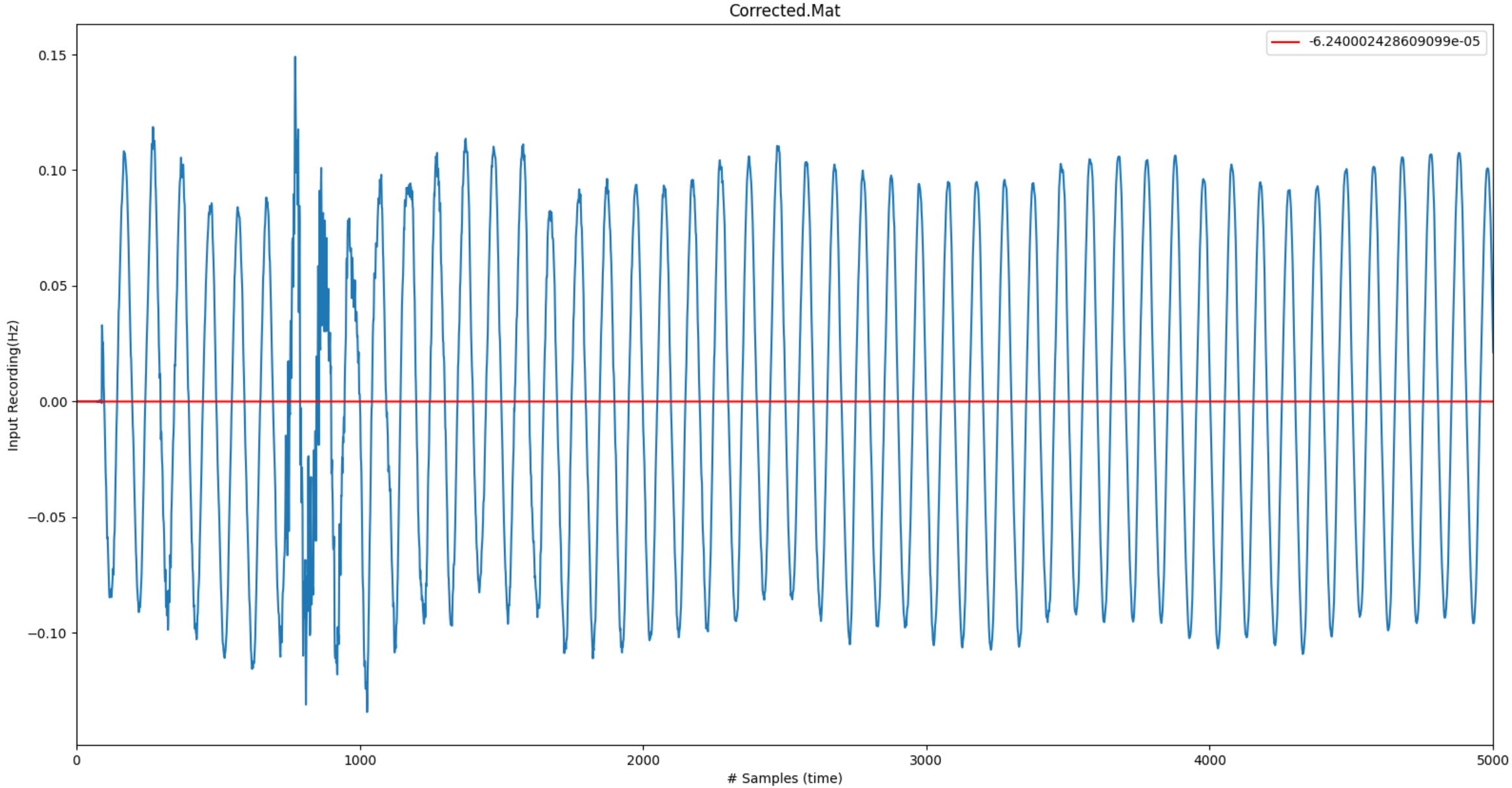

A4 After Packing
Corrected2.txt
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15
Sample Value
0
1000
2000
3000
4000
5000
Samples (time)

CORRECTED.MAT GRAPH

Y-axis label is incorrect, it is not in frequency. It is just the value of each sample.

This data was obtained from corrected.mat (from Professor Boyd).

The corrected.mat file was obtained from a different audio recording, the one previously sent to Professor Boyd.


Corrected.Mat
-6.240002428609099e-05
0.15
0.10
0.05
0.00
−0.05
−0.10
Input Recording(Hz)
0
1000
2000
3000
4000
5000
Samples (time)

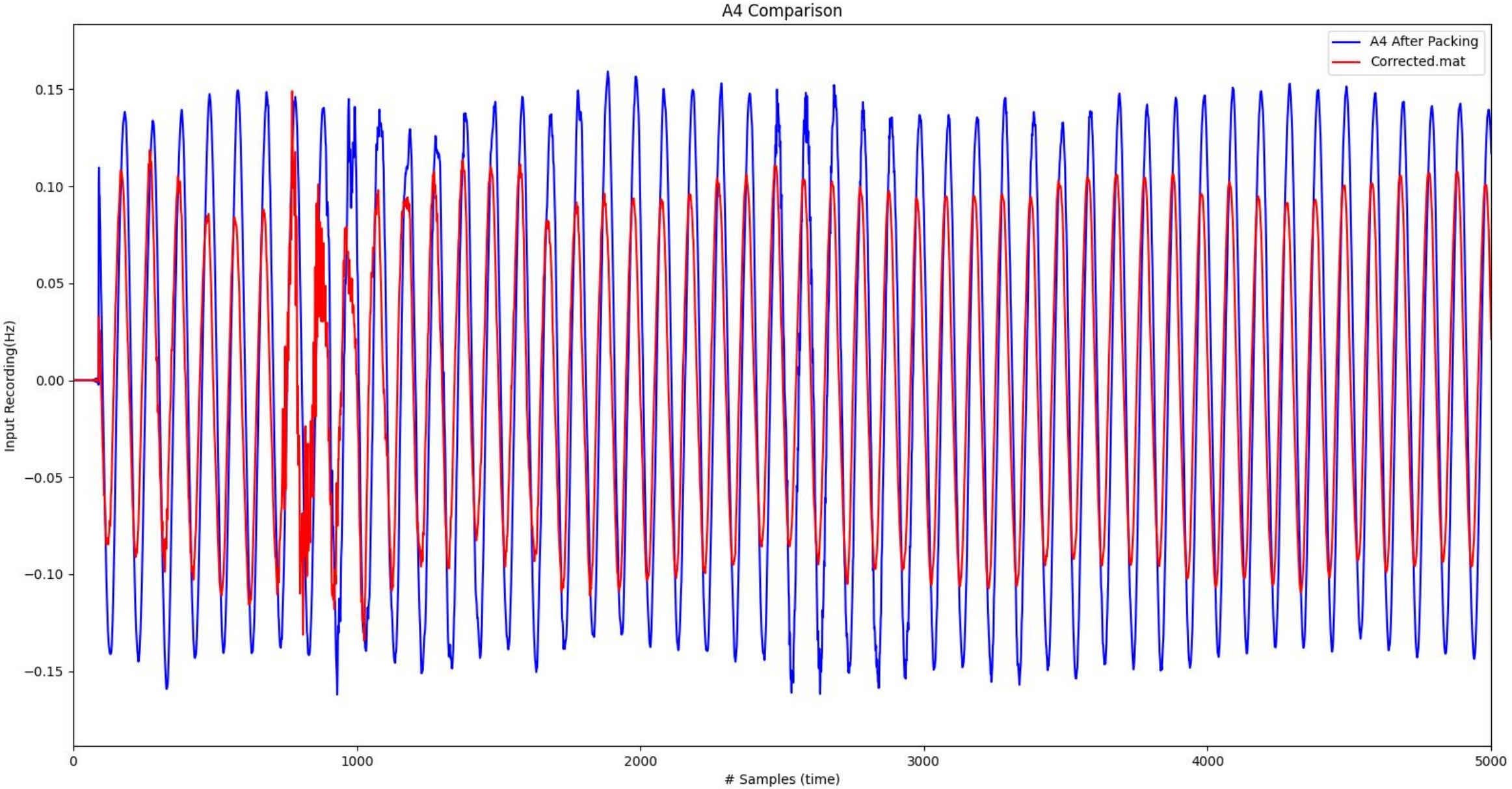
COMPARING CORRECTED.MAT AND A4_AFTER_PACKING.TXT

Y-axis label is incorrect, it is not in frequency. It is just the value of each sample.

In blue we have “A4_After_Packing.txt”, in red we have “corrected.mat”.

The corrected.mat file was obtained from a different audio recording, the one previously sent to Professor Boyd.

Note: They do not match perfectly, but they were in fact from other recordings.


A4 Comparison
A4 After Packing
Corrected.mat
Input Recording(Hz)
Samples (time)
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15
0
1000
2000
3000
4000
5000

COMPARING CORRECTED.MAT AND CORRECTED2.MAT

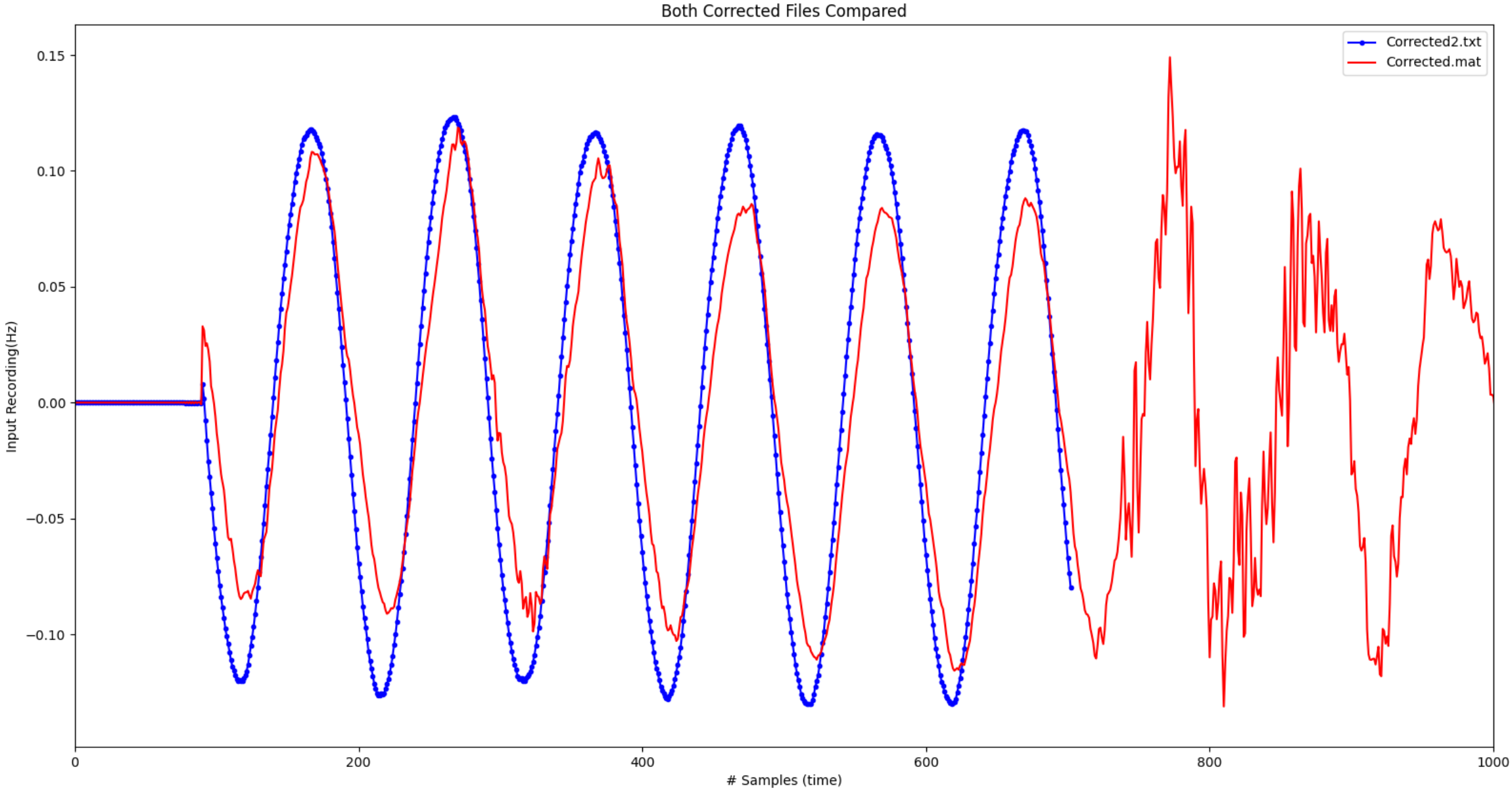
Y-axis label is incorrect, it is not in frequency. It is just the value of each sample.

In blue we have “corrected2.txt”, in red we have “corrected.mat”.

As noted above, corrected2.txt was obtained by inserting “A4_Before_Packing.txt” into the online Matlab Editor.

Seems to be a very close match, leads me to believe that my coding method for repacking works. Or would it be the complete opposite, since its not exact match, it cannot be right?

Unfortunately the online IDE has limited output, so I could only get < 1000 samples.

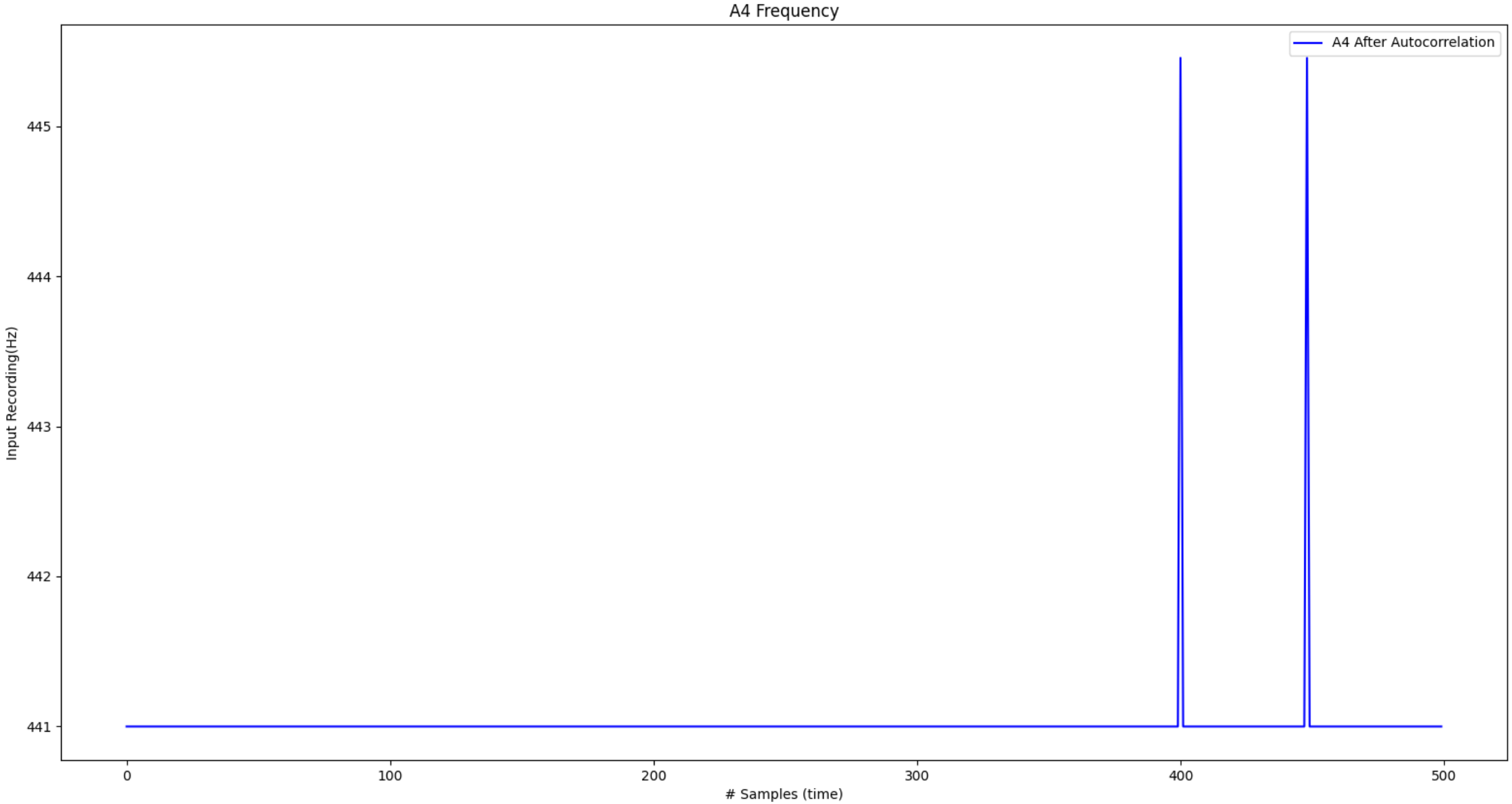

Both Corrected Files Compared
Corrected2.txt
Corrected.mat
Input Recording(Hz)
Samples (time)
0.15
0.10
0.05
0.00
−0.05
−0.10
0
200
400
600
800
1000

PLOTTING THE FREQUENCY

This is plotting the “A4_Autocorrelated_Frequency.txt”.

This data was obtained by using the Autocorrelation method provided by professor Boyd.

The line is hovering at directly 441hz (except for the spikes).


A4 Frequency
A4 After Autocorrelation
445
444
443
442
441
Input Recording(Hz)
0
100
200
300
400
500
Samples (time)

FINAL QUESTIONS AND THOUGHTS

How can I determine whether the before/after packing data is good or bad?

Is there a visual way that the Autocorrelation method can be related to the “After Packing Before Autocorrelation” Graph?

What does the t line do in the Octave example?

How does the loudness/distance from the microphone relate to the noise and or amplitude of the Sin Graph?

If the Before Autocorrelation but After Packing values are OK, then why are we not bang on at 440hz? Is there an error with how I am doing the Auto Correlation?