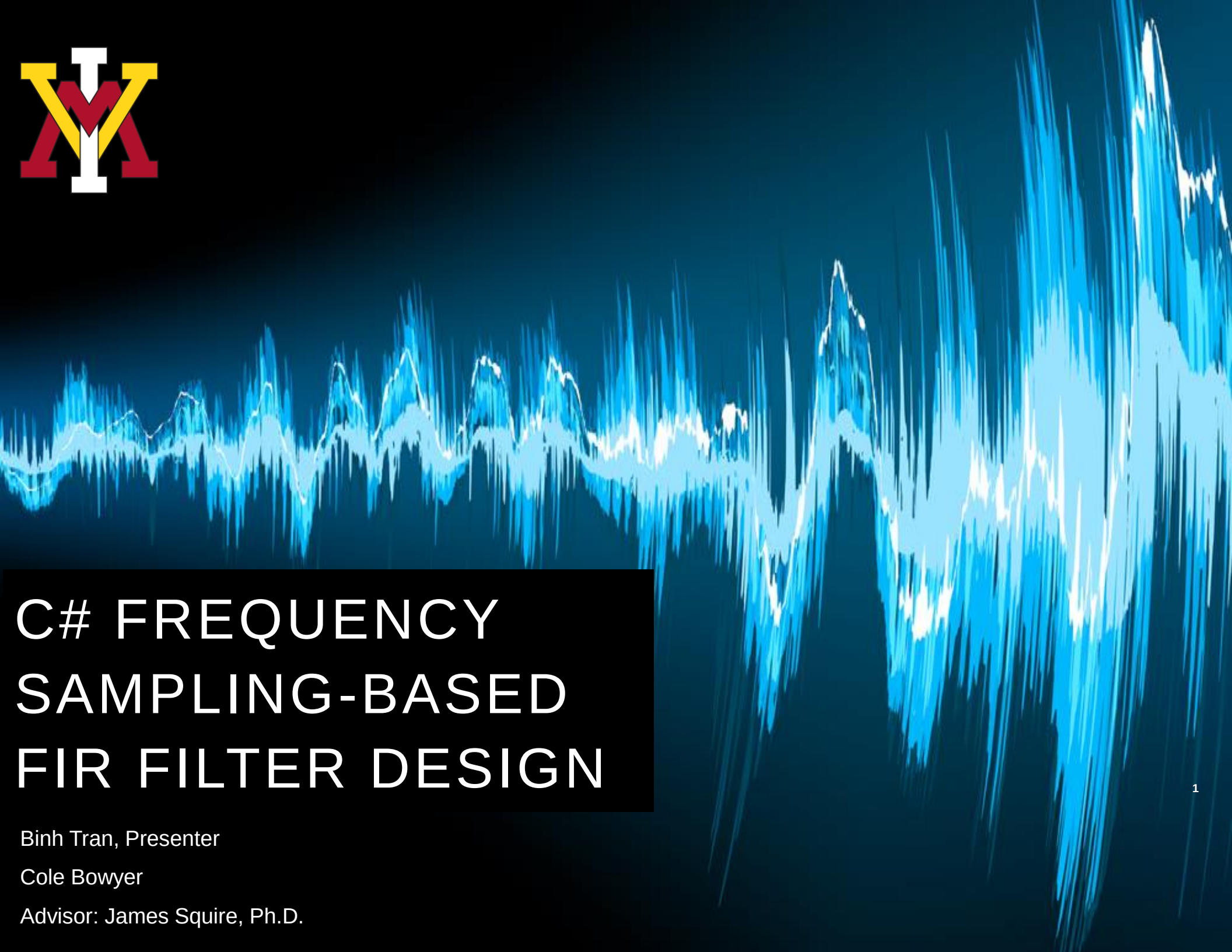


C# FREQUENCY SAMPLING-BASED FIR FILTER DESIGN

Binh Tran, Presenter

Cole Bowyer

Advisor: James Squire, Ph.D.

OVERVIEW

- Introduction
- Impact
- Background
- FIR Filter Design Method
- Result
- Conclusion
- Reference

INTRODUCTION

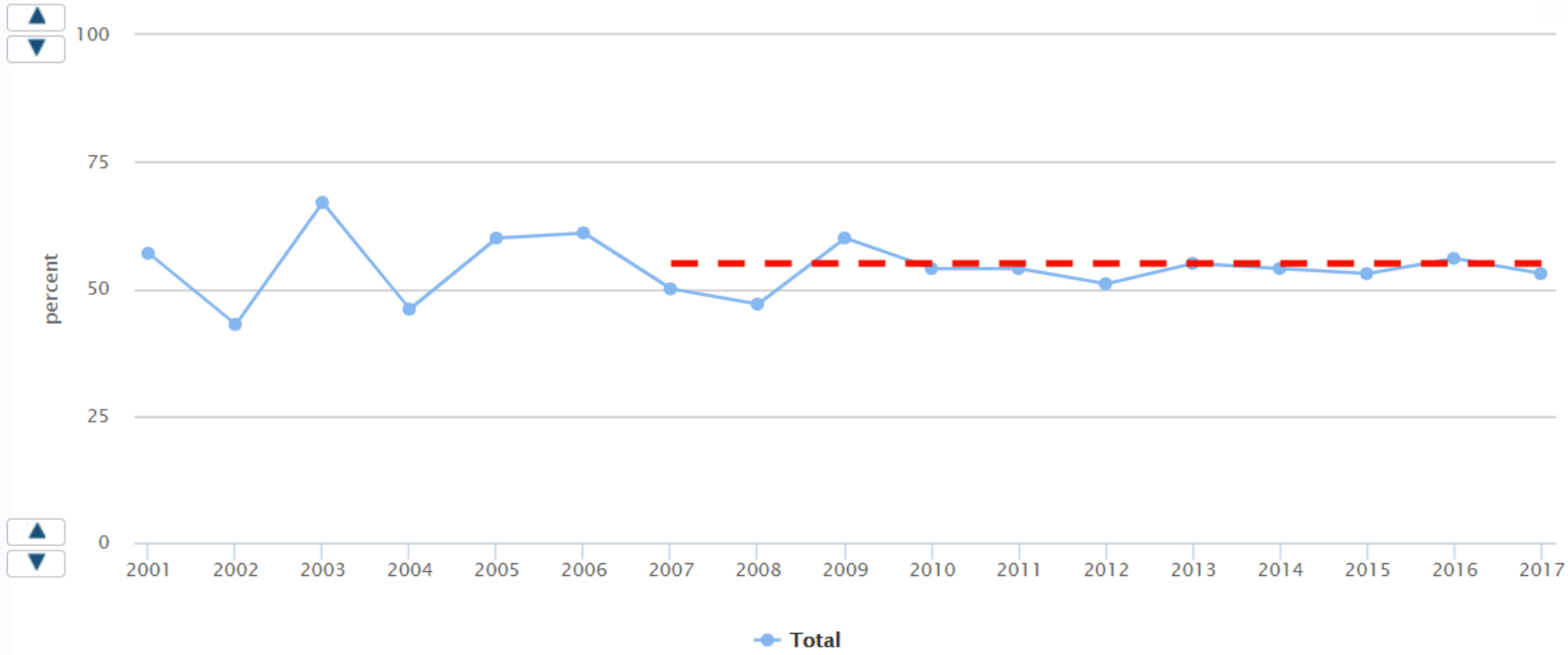
- Frequency Sampling-Based FIR Filter Design
- Open-source filter
- C#

IMPACT

Infants with hearing loss receiving intervention services before age 6 months (percent)
By Total

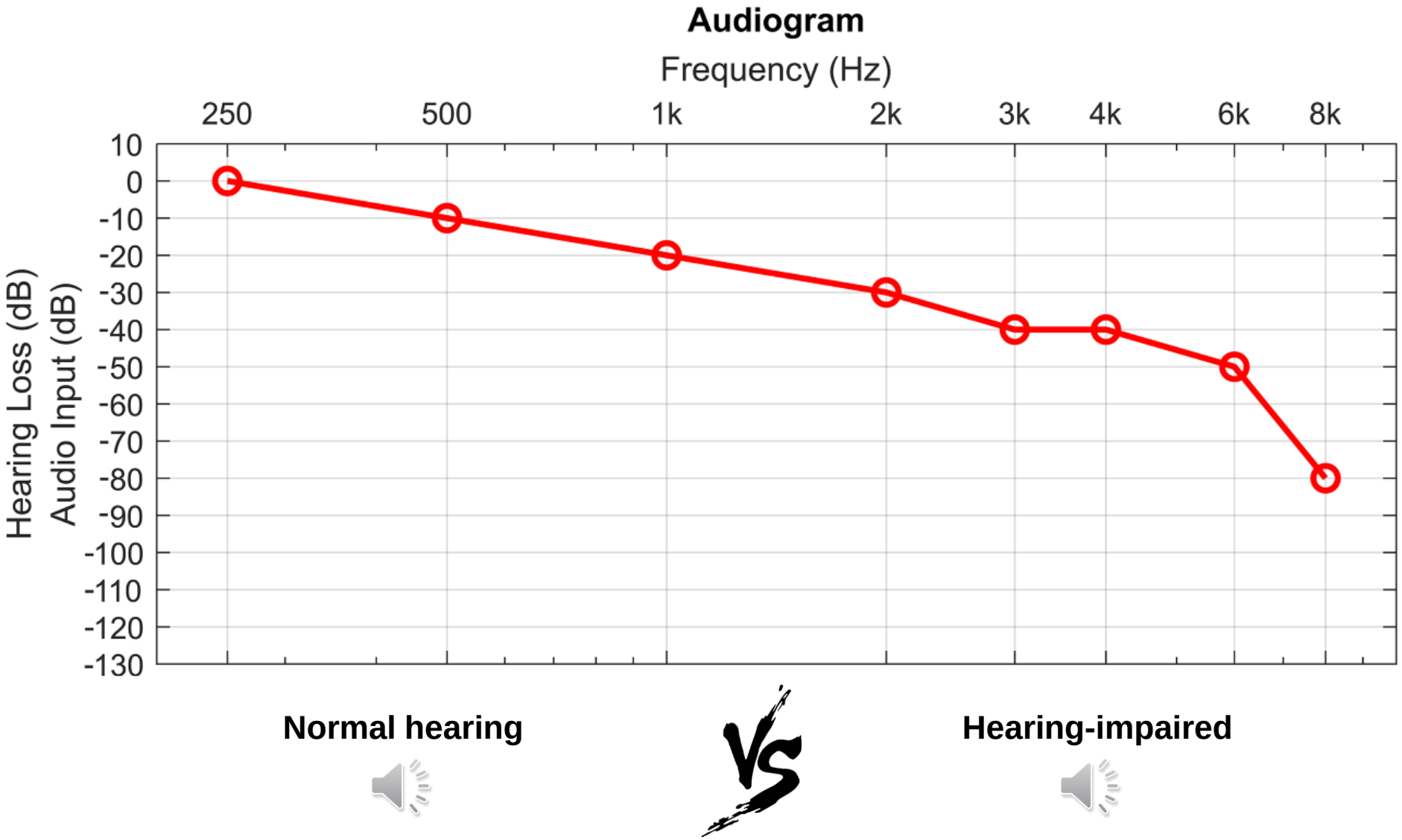
2020 Baseline (year): 50.0 (2007) **--- 2020 Target:** 55.0 **Desired Direction:** ↑ Increase desired

Auto Scale

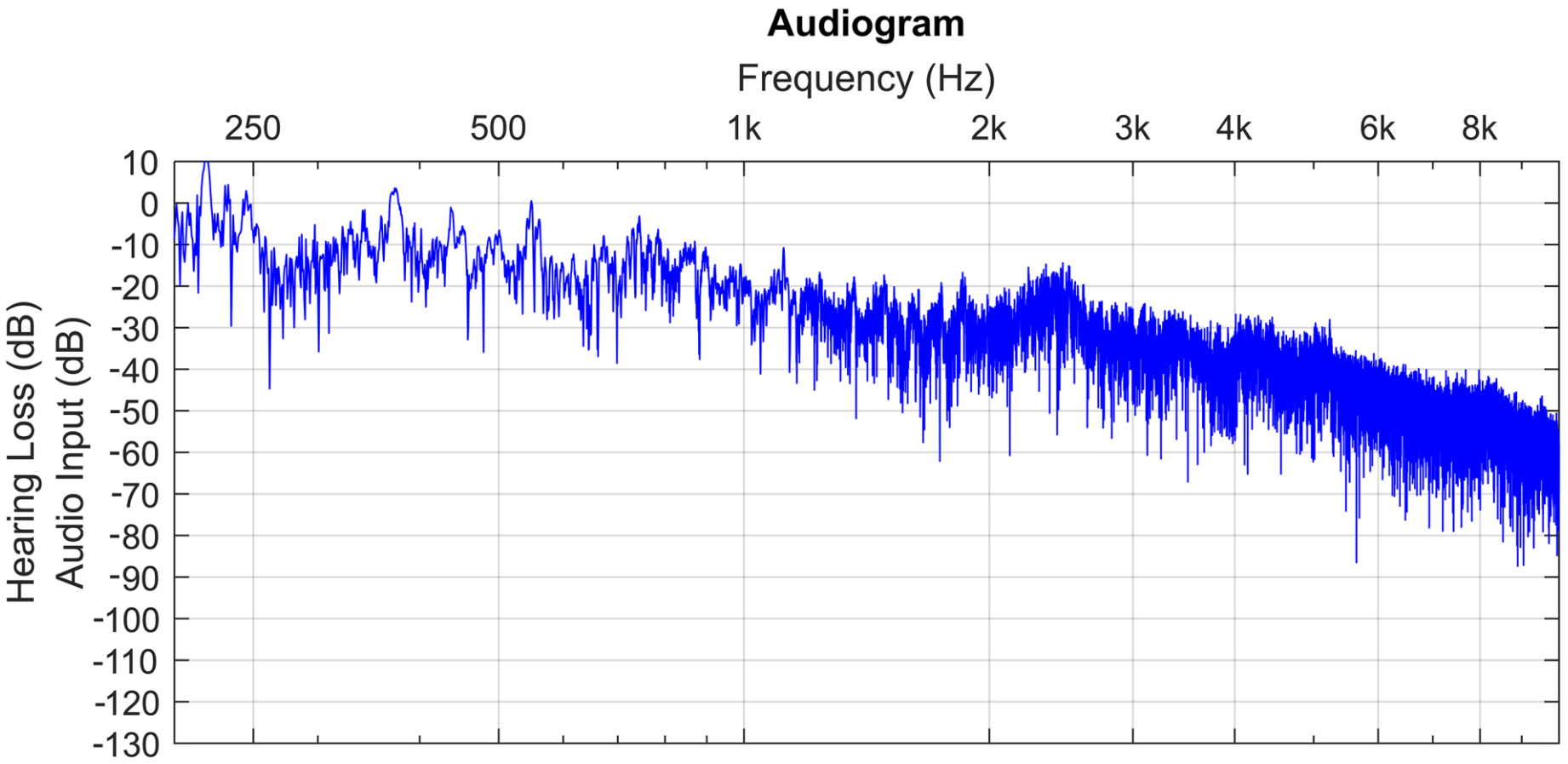


Data Source: State-based Early Hearing Detection and Intervention Program Network (EHDI); Centers for Disease Control and Prevention, National Center on Birth Defects and Developmental Disabilities (CDC/NCBDDD)
Additional footnotes may apply to these data. Please refer to footnotes below the data table for further information.

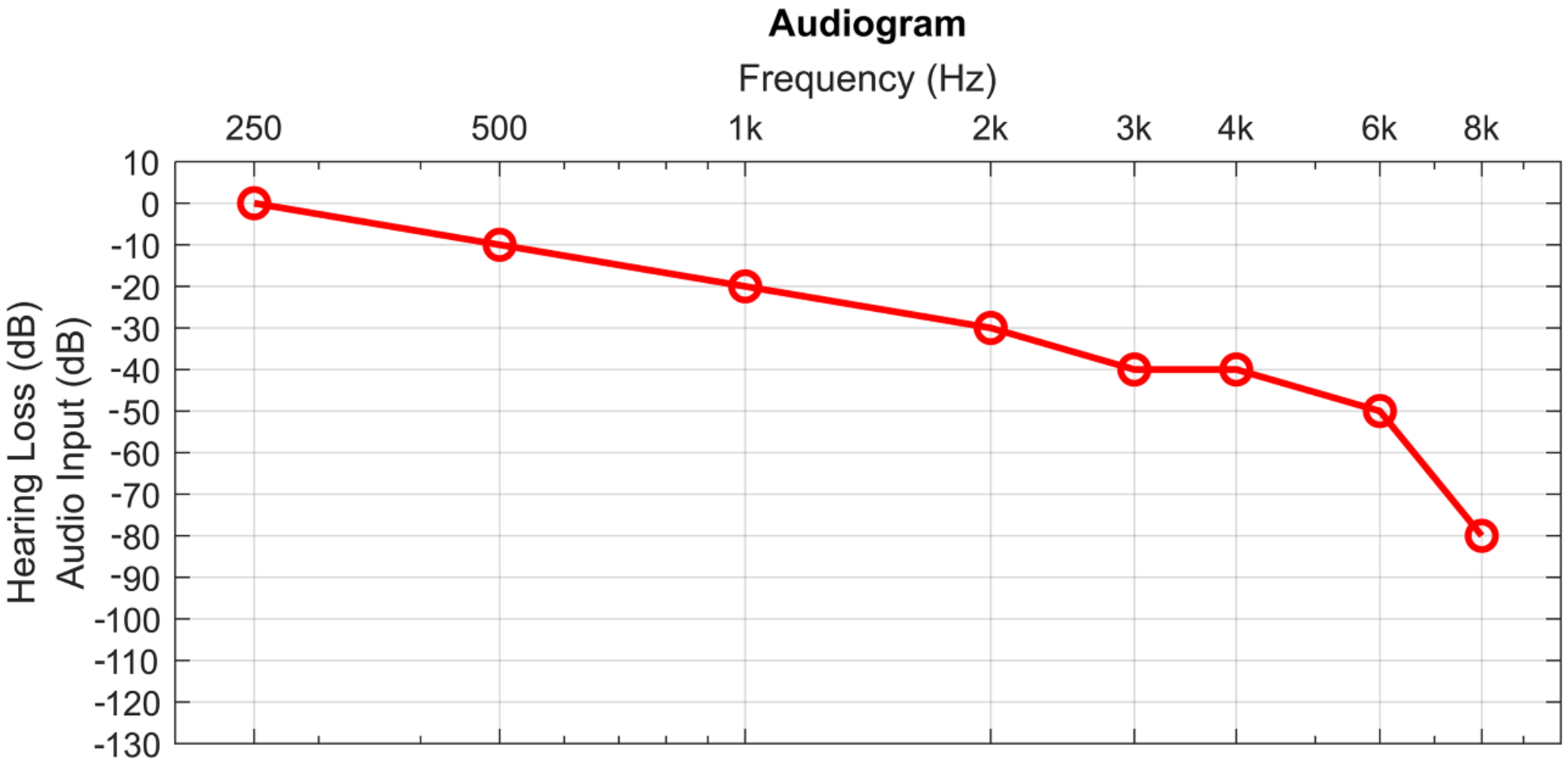
IMPACT (CONT)


Audiogram
Frequency (Hz)
250
500
1k
2k
3k
4k
6k
8k
10
0
-10
-20
-30
-40
-50
-60
-70
-80
-90
-100
-110
-120
-130
Hearing Loss (dB)
Audio Input (dB)
Normal hearing
VS
Hearing-impaired

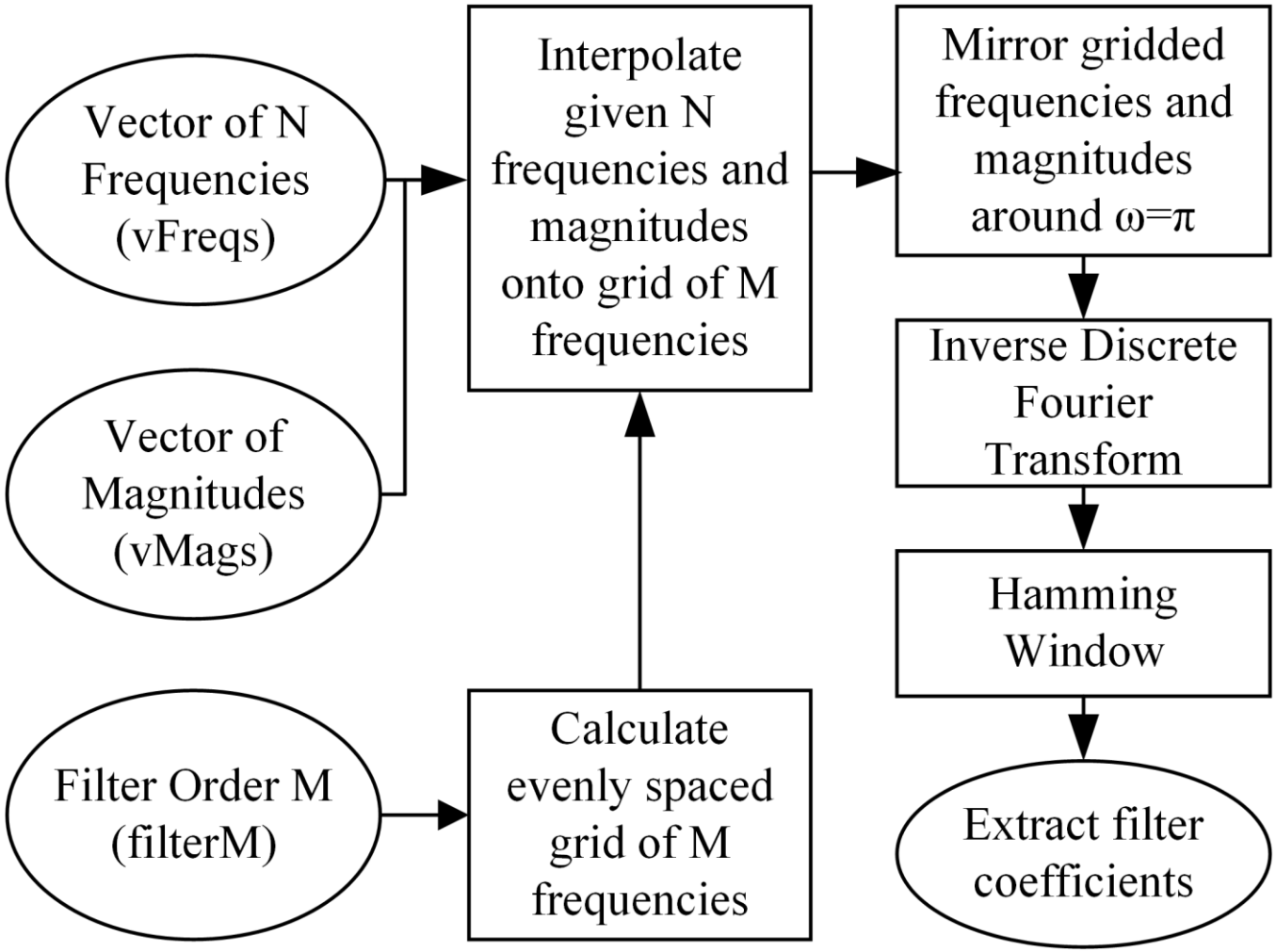
BACKGROUND


Audiogram
Frequency (Hz)
250
500
1k
2k
3k
4k
6k
8k
10
0
-10
-20
-30
-40
-50
-60
-70
-80
-90
-100
-110
-120
-130
Hearing Loss (dB)
Audio Input (dB)

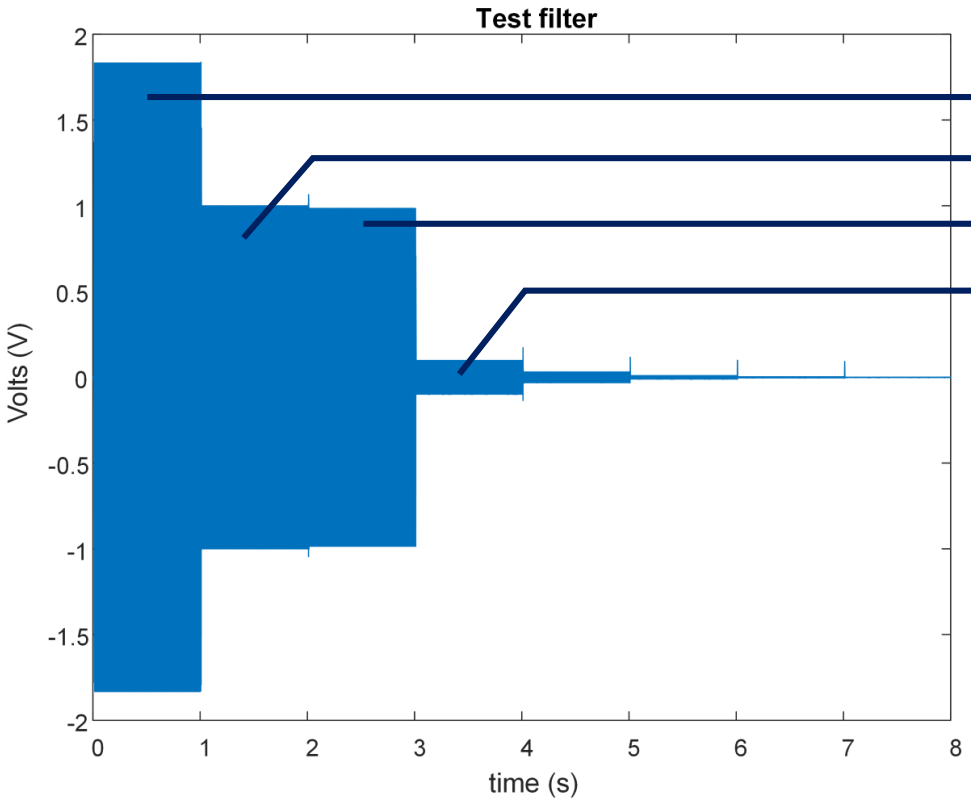
BACKGROUND


Audiogram
Frequency (Hz)
250
500
1k
2k
3k
4k
6k
8k
10
0
-10
-20
-30
-40
-50
-60
-70
-80
-90
-100
-110
-120
-130
Hearing Loss (dB)
Audio Input (dB)

FREQUENCY SAMPLING-BASED FIR FILTER DESIGN METHOD

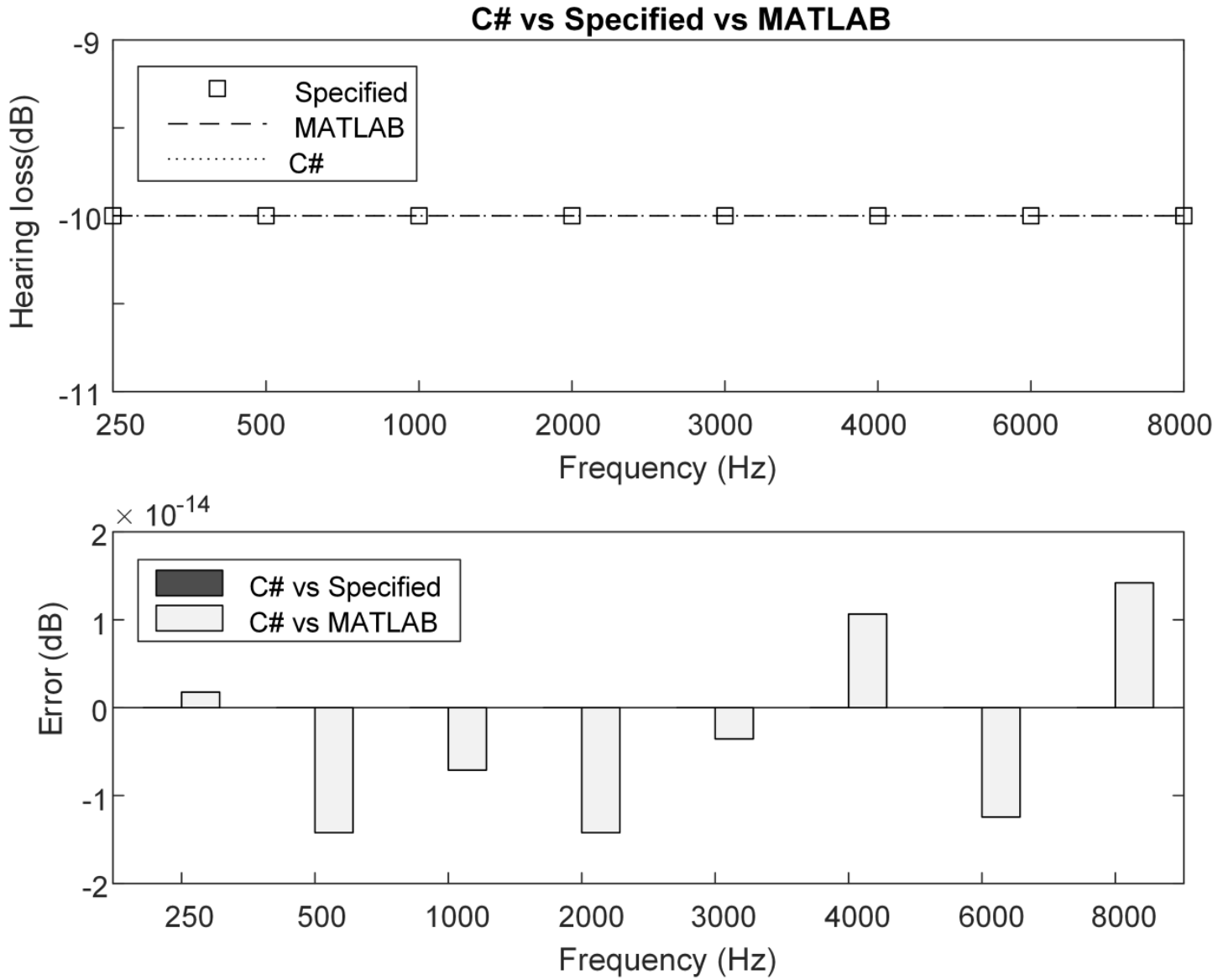


RESULT

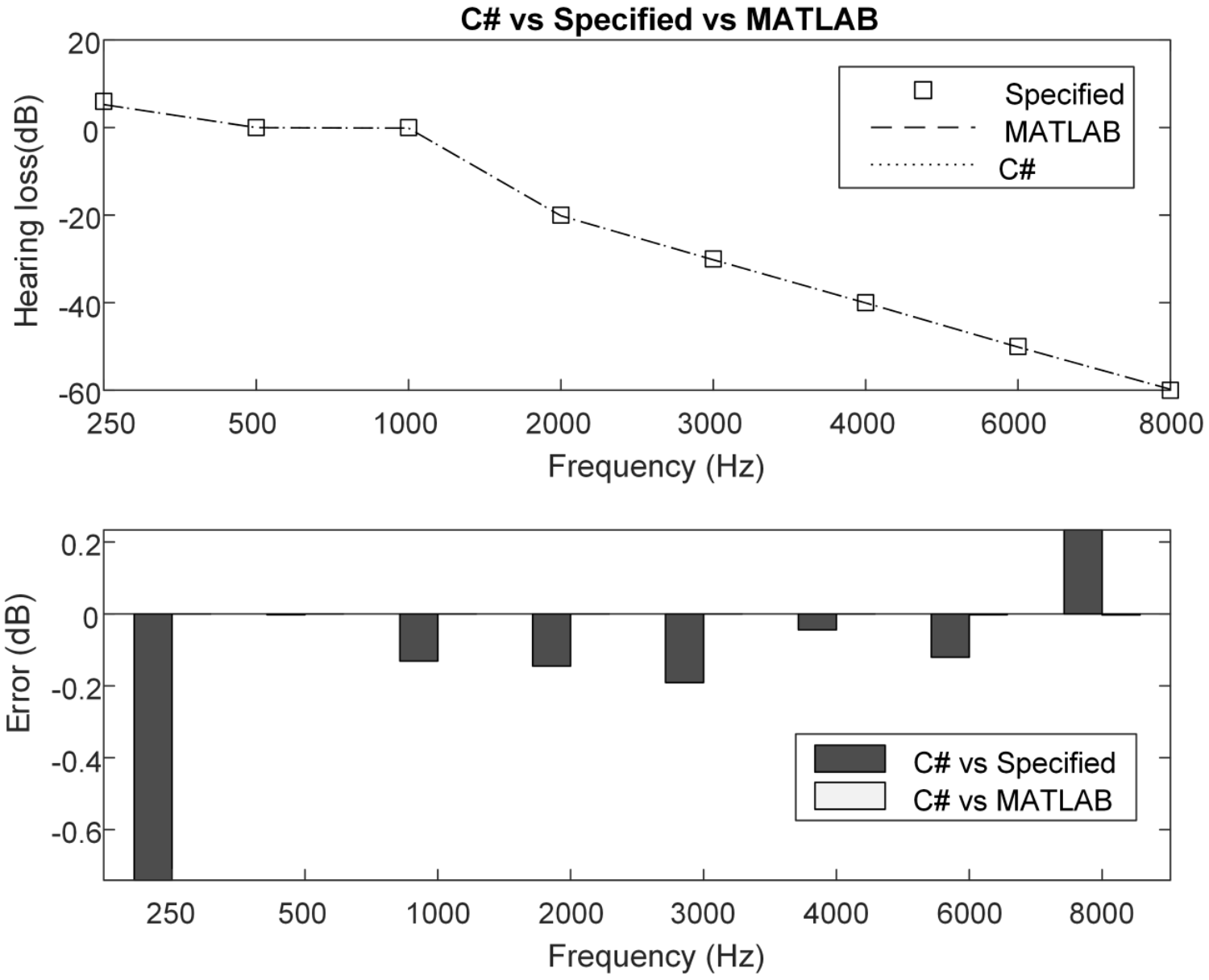


Frequency	Value
250 Hz	6 dB
500 Hz	0 dB
1 kHz	0 dB
2 kHz	-20 dB
3 kHz	-30 dB
4 kHz	-40 dB
6 kHz	-50 dB
8 kHz	-60 dB

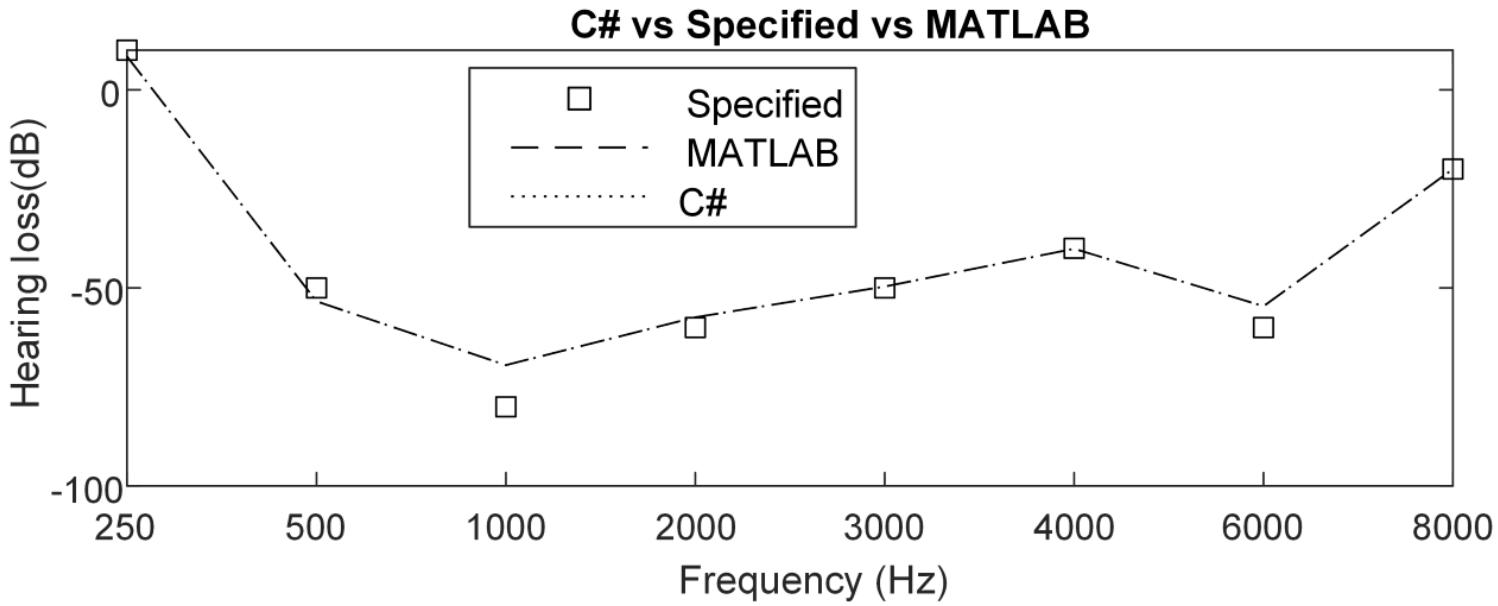
RESULT (CONT)

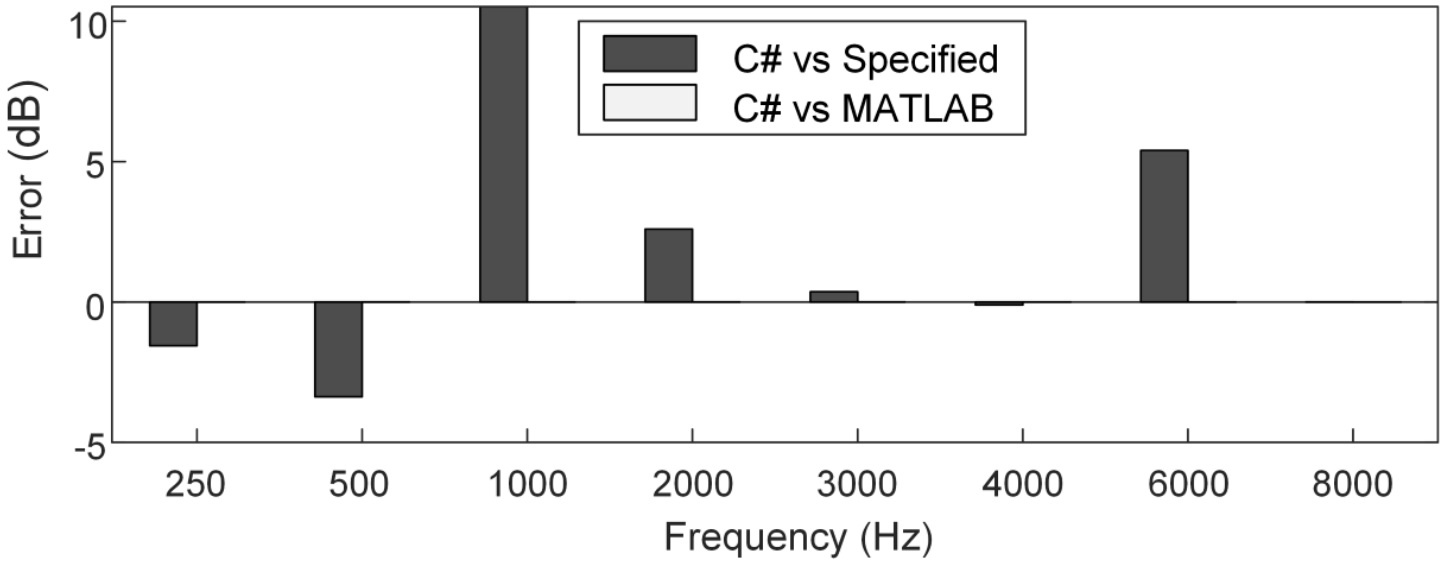

C# vs Specified vs MATLAB
Hearing loss(dB)
-9
-10
-11
Specified
MATLAB
C#
250
500
1000
2000
3000
4000
6000
8000
Frequency (Hz)
× 10-14
2
1
0
-1
-2
Error (dB)
C# vs Specified
C# vs MATLAB
250
500
1000
2000
3000
4000
6000
8000
Frequency (Hz)

RESULT (CONT)


C# vs Specified vs MATLAB
Hearing loss(dB)
20
0
-20
-40
-60
250
500
1000
2000
3000
4000
6000
8000
Frequency (Hz)
Specified
MATLAB
C#
Error (dB)
0.2
0
-0.2
-0.4
-0.6
C# vs Specified
C# vs MATLAB
Frequency (Hz)

RESULT (CONT)


C# vs Specified vs MATLAB
Specified
MATLAB
C#
Hearing loss(dB)
0
-50
-100
250
500
1000
2000
3000
4000
6000
8000
Frequency (Hz)

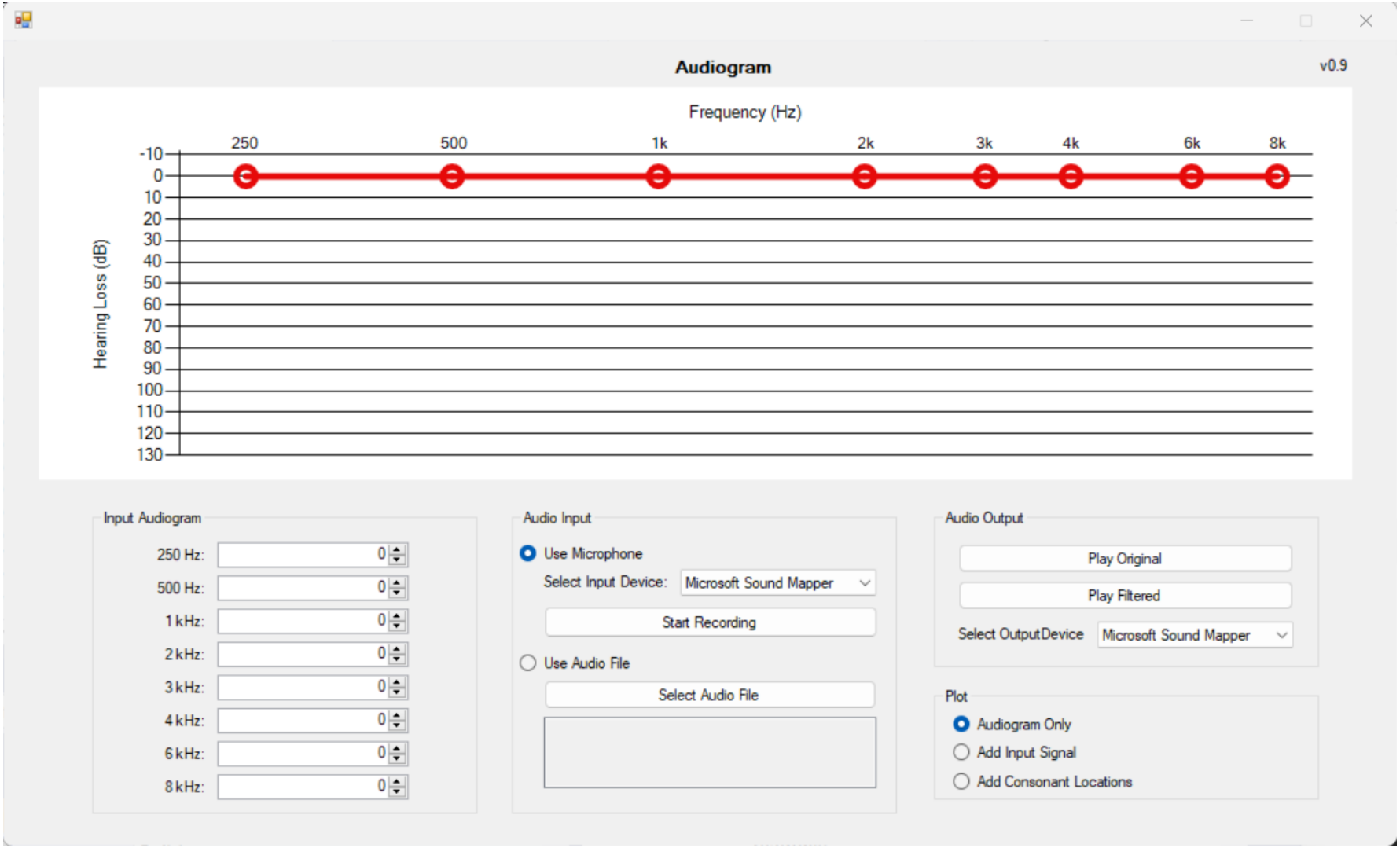

C# vs Specified
C# vs MATLAB
Error (dB)
10
5
0
-5
250
500
1000
2000
3000
4000
6000
8000
Frequency (Hz)

RESULT (CONT)

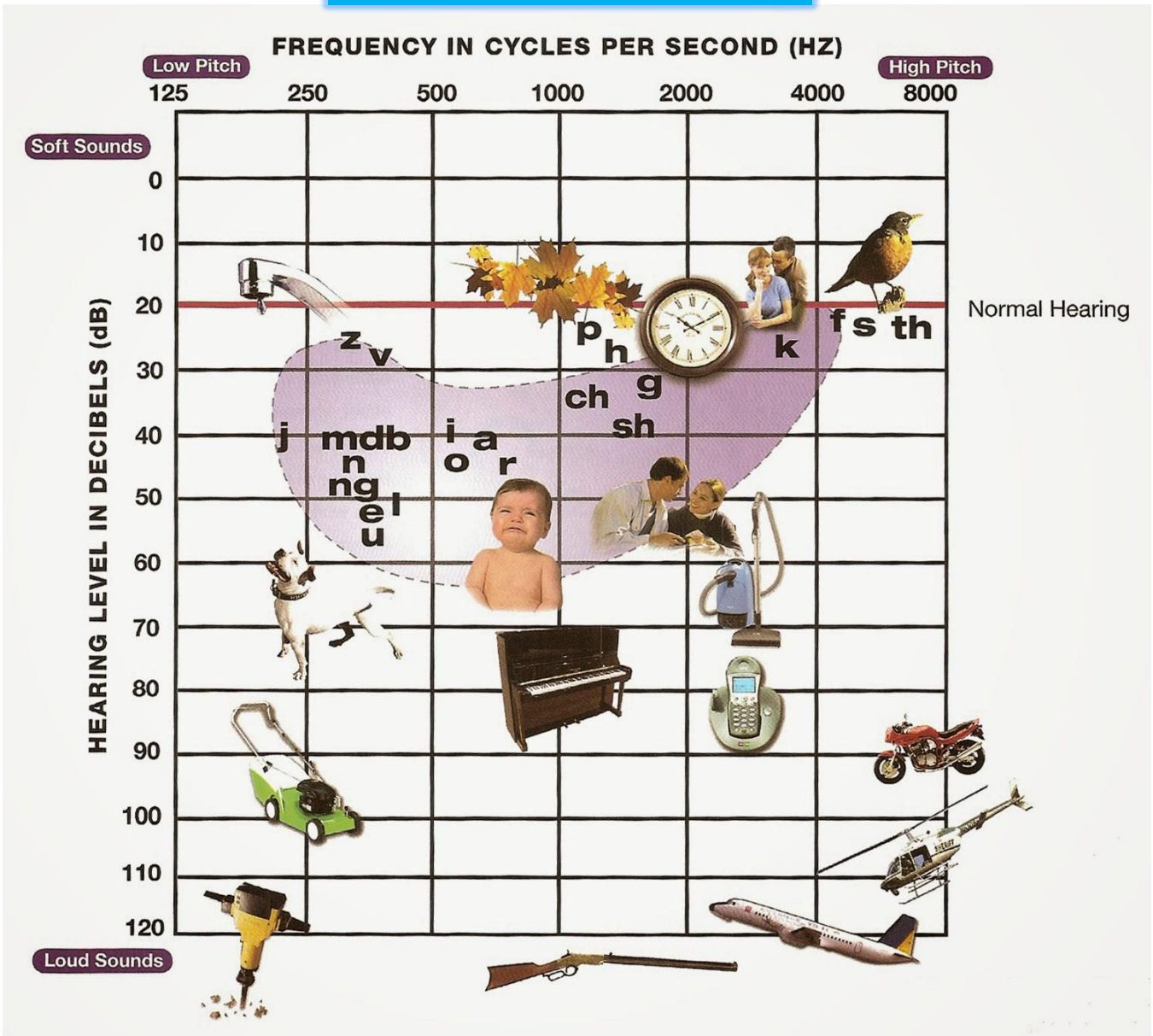
	MATLAB	C#
Speed	478 µs	200 µs
Disk Space	700 MB	120 kB
RAM	1.2 GB	2.7 MB

CONCLUSION

- GitHub: http://bit.ly/3gBNJDl
- Future Work: Audiogram Emulator



QUESTION


FREQUENCY IN CYCLES PER SECOND (HZ)
Low Pitch
High Pitch
125
250
500
1000
2000
4000
8000
Soft Sounds
Loud Sounds
HEARING LEVEL IN DECIBELS (dB)
0
10
20
30
40
50
60
70
80
90
100
110
120
Normal Hearing
z
v
j
m
d
b
n
ng
e
l
u
i
o
a
r
p
h
ch
g
sh
k
f
s
th

CITATIONS

[1] S. K. Mitra, “Digital Signal Processing: A Computer Based Approach,” McGraw-Hill Higher Education, pp. 1-15, 2001.

[2] J. W. Pitton, K. Wang, B. Juang, et al., “Time-Frequency Analysis and Auditory Monitoring for Automatic Recognition of Speech,” Proc. of the IEEE, vol. 84, pp. 1199-2004, September 1994.

[3] S. Furui, “Digital Peech Processing, Synthesis, and Recognition,” Mercel Dekker, pp. 14-20, 2000.

[4] W. M. Siebert, “Circuits, Signals, and Systems,” The MIT Press, pp. 4-5, 1986.

[5] L. R. Rabiner and B. Gold, “Theory and Application of Digital Signal Processing,” Prentice-Hall, pp. 136-140, 1975.

[6] A. V. Oppenheim, R. W. Schafer, Discrete-Time Signal Processing. Prentice Hall, pp. 309-313, 1999.

[7] “Frequency sampling-based FIR filter design - MATLAB fir2,” www.mathworks.com. https://www.mathworks.com/help/signal/ref/fir2.html (accessed Dec. 10, 2022).

[8] “Support Limitations for MATLAB for Code Generation - MATLAB & Simulink,” www.mathworks.com. https://www.mathworks.com/help/sldv/ug/support-limitations-for-matlab-for-code-generation-1.html (accessed Dec. 10, 2022).

CODE

```
// Define variables
int fs = 44100;
double[] x = new double[fs*8]; // input vector
double[] fHz = {250, 500, 1e3, 2e3, 3e3, 4e3, 6e3, 8e3};

// Create the input signal x of 1s of each frequency in fHz
double t = 0;   // current time
for (int iFreq = 0; iFreq < 8; iFreq++, t += 1.0/fs)
    for (int i = 0; i < fs; i++)
        x[i+iFreq*fs] = Math.Cos(2*Math.PI*fHz[iFreq]*t);

// Filter x using the filter h created in Fig. 5
double[] y = new double[x.Length]; // output vector
for (int i = 0; i < x.Length; i++)
{
    for (int j = 0; j < h.Length; j++)
    {
        if (i - j < 0) continue;
        y[i] += h[j] * x[i-j];
    }
}
Console.WriteLine("Filtered output Data: ");
for (int i = 0; i <= filterM; i++)
    Console.WriteLine(y[i]);
```

CODE (CONT)

```
double fs = 44100;
double[] fHz = {0,250,500,1e3,2e3, 3e3, 4e3, 6e3, 8e3, fs/2};
double[] vaudiogram = {6, 0, 0, -20, -30, -40, -50, -60};

// Filter Order M
double filterM = 1000;

// Calculate the Vector of N frequencies
double[] vFreqs = new double[fHz.Length];
for (int i = 0; i < fHz.Length; i++)
    vFreqs[i] = fHz[i] / (fs / 2.0);

// Calculate the Vector of Magnitudes
double[] vA = new double[vaudiogram.Length];
for (int i = 0; i < vaudiogram.Length; i++)
    vA[i] = Math.Pow(10, (double)vaudiogram[i] / 20.0);

double[] vMags = new double[vA.Length + 2];
vMags[0] = vA[0];
vMags[vMags.Length - 1] = vA[vA.Length - 1];

for (int i = 1; i < vMags.Length - 1; i++)
    vMags[i] = vA[i - 1];

// Calculate the Filter Coefficients
double[] h = new double[(int)filterM + 1];
Filter(filterM, vFreqs, vMags, out h);

// Print the Filter Coefficients
Console.WriteLine("Filter Coefficients: ");
for (int i = 0; i <= filterM; i++)
    Console.WriteLine(h[i]);
```