

CPSC 501 Assignment 4

OPTIMIZING PROGRAM PERFORMANCE

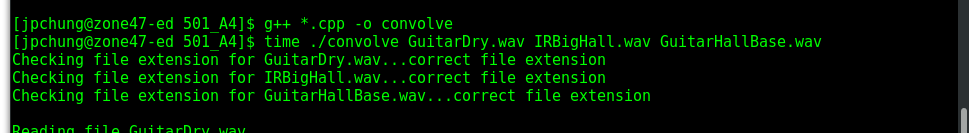
Johnny Phuong Chung | 10036448 | Dec 8 2017

# BaseLine Program

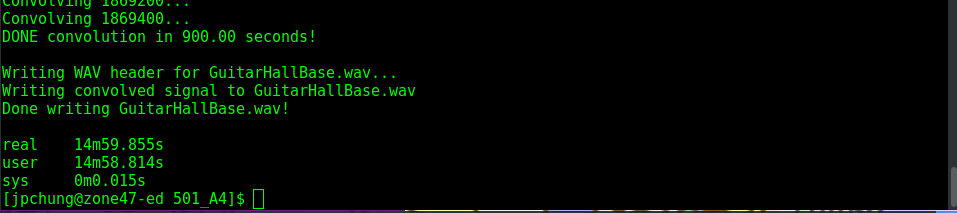
The initial version of the convolution reverb program (**convolve.cpp**) implemented convolution of input and impulse signals directly in the time domain, using an input-side algorithm, to produce the convolved signal:

* inputFile: **GuitarDry.wav**
* impulseFile: **IRBigHall.wav**
* outputFile: **GuitarHallBase.wav**

|  |  |  |
| --- | --- | --- |
| **Program version** | **Time (s)** | **Reduction (%)** |
| Baseline (convolve.cpp) | 899.855 | N/A |

The runtime performance was measured using the Linux **time** command:

Specifically, the “real” time value was used for the measured time values, with minutes converted back to seconds. For consistency, all subsequent time measurements in this report will also use the same **time** command and “real” value:



A Profiler report (**Profile\_convolveBase.txt)** was also generated using **gprof** profiler when compiling with **g++**:

* g++ -p \*.cpp –o convolve
* time ./convolve GuitarDry.wav IRBigHall.wav GuitarHallBase.wav
* gprof convolve

For consistency, subsequent optimizations will also have corresponding Profiler reports generated in the same manner with their respective object files (NOTE: time fluctuations may occur due to remote access operations).

# Algorithm-Based optimization

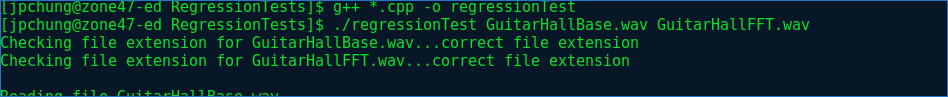
The subsequent version of the program (**convolveFFT.cpp**) was a re-implementation of the convolution using a frequency-domain convolution algorithm, specifically a Fast Fourier Transform (FFT). The *four1* FFT algorithm, more formally known as the *Danielson-Lanczos* algorithm, was chosen for the frequency-domain convolution optimization. Given that the input-side algorithm is O(n2) algorithm and the *four1* FFT algorithm is an O(n) algorithm, it is not surprising to see the following increase in runtime performance:

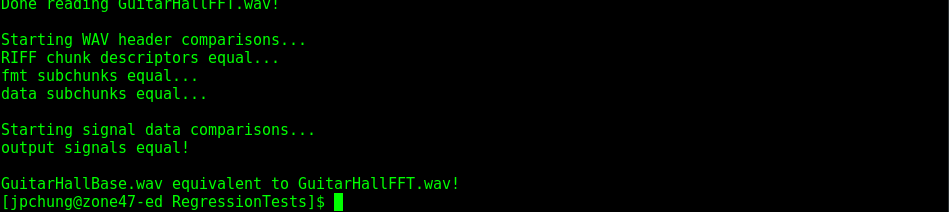
* inputFile: **GuitarDry.wav**
* impulseFile: **IRBigHall.wav**
* outputFile(s): **GuitarHallBase.wav, GuitarHallFFT.wav**

|  |  |  |
| --- | --- | --- |
| **Program version** | **Time (s)** | **Reduction (%)** |
| Baseline (convolve.cpp) | 899.855 | N/A |
| FFT (convolveFFT.cpp) | 8.745 | 99.028 |

In order to ensure fidelity between the baseline and FFT algorithm versions, and to compare future optimizations, regression tests (**ConvolveRegressionTests.**cpp) were written so as to compare equivalence between the respectively produced output files:

* baseline output: **GuitarHallBase.wav**
* FFT output: **GuitarHallFFT.wav**



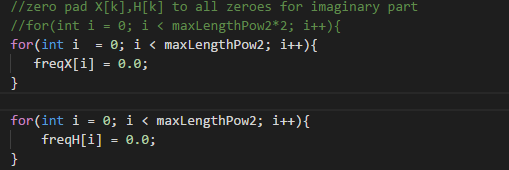


# CODE-Tuning optimization #1 – Code JAMMING

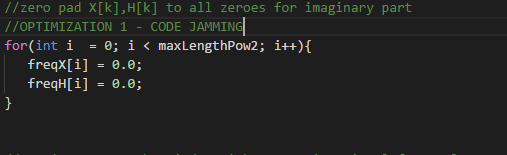
In order to further optimize the FFT convolution, the following code-tuning was performed:

* **Code Jamming**
  + combined two loops into one due to iteration over same counter

**BEFORE:**



**AFTER:**



The optimized version was then compared with original FFT for improvements in runtime performance:

* inputFile: **GuitarDry.wav**
* impulseFile: **IRBigHall.wav**
* outputFile: **GuitarHallFFT.wav, GuitarHallFFT\_Opt1.wav**

|  |  |  |
| --- | --- | --- |
| **Program version** | **Time (s)** | **Reduction (%)** |
| FFT (convolveFFT.cpp) | 8.745 | N/A |
| Optimization 1 (convolveFFT\_Opt1.cpp) | 8.489 | 1.256 |

The output file of the optimized version was also checked for equivalence with regression testing:

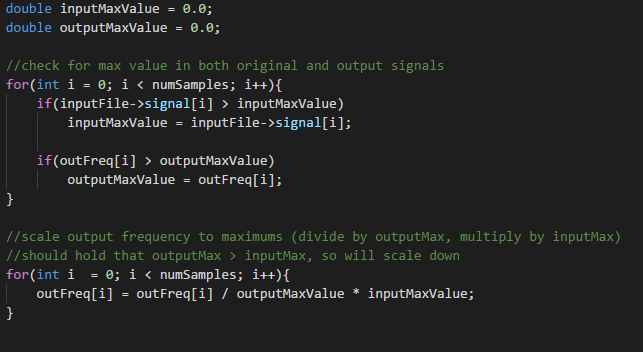


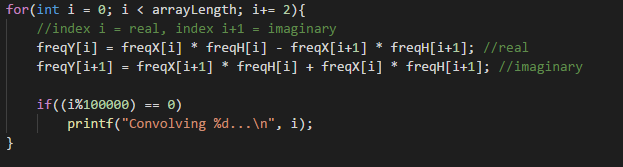
# CODE-Tuning optimization #2 – MINIMIZING WORK INSIDE LOOPS

The next optimization following Optimization #1 for the FFT convolution is as follows:

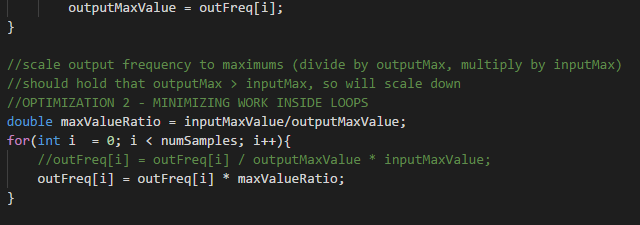
* **Minimizing work inside loops** 
  + put calculation that results in a constant before the loop
  + removed print statement from inside loop

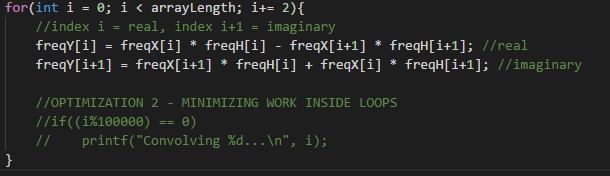
**BEFORE:**





**AFTER:**



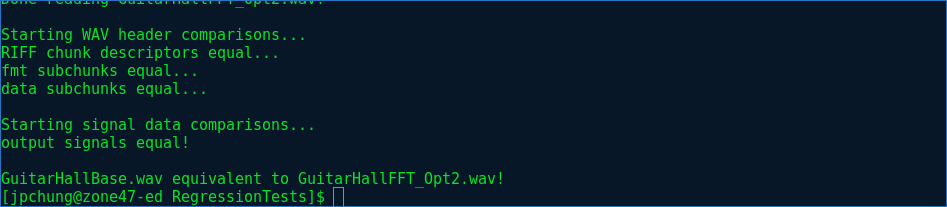


This next optimized version was then compared the first optimization for improvements in runtime performance:

* inputFile: **GuitarDry.wav**
* impulseFile: **IRBigHall.wav**
* outputFile: **GuitarHallFFT\_Opt1.wav, GuitarHallFFT\_Opt2.wav**

|  |  |  |
| --- | --- | --- |
| **Program version** | **Time (s)** | **Reduction (%)** |
| Optimization 1 (convolveFFT\_Opt1.cpp) | 8.489 | N/A |
| Optimization 2 (convolveFFT\_Opt2.cpp) | 8.366 | 1.449 |

The output file of the optimized version was also checked for equivalence with regression testing:

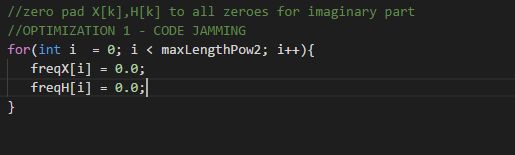
****

# CODE-Tuning optimization #3 – UNROLLING (PARTIAL)

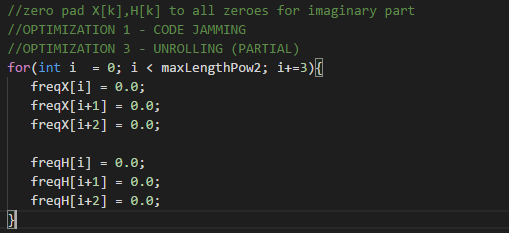
Following the second optimization to the FFT convolution, the following code-tuning was performed:

* **Partial Unrolling**
  + Multiple cases handled inside the loop instead of just one

**BEFORE:**



**AFTER:**

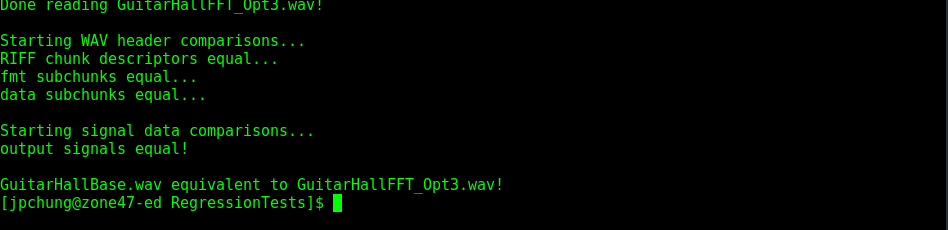


This third optimized version was then compared the previous optimization for improvements in runtime performance:

* inputFile: **GuitarDry.wav**
* impulseFile: **IRBigHall.wav**
* outputFile: **GuitarHallFFT\_Opt2.wav, GuitarHallFFT\_Opt3.wav**

|  |  |  |
| --- | --- | --- |
| **Program version** | **Time (s)** | **Reduction (%)** |
| Optimization 2 (convolveFFT\_Opt2.cpp) | 8.366 | N/A |
| Optimization 3 (convolveFFT\_Opt3.cpp) | 8.262 | 1.243 |

The output file of the optimized version was also checked for equivalence with regression testing:

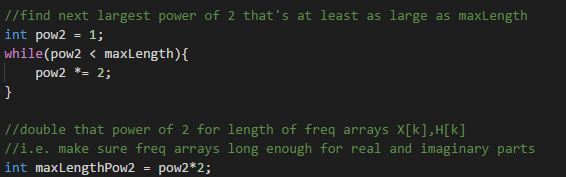


# CODE-Tuning optimization #4 – STRENGTH REDUCTION

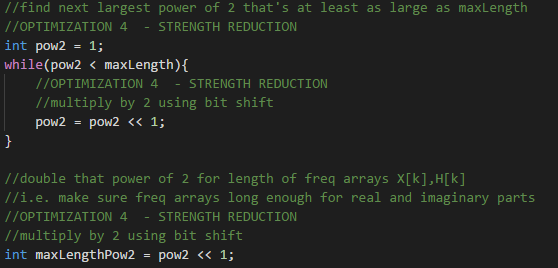
Following the third optimization to the FFT convolution, the following code-tuning was performed:

* **Strength Reduction**
  + Replace an expensive operation (multiplication) with a cheaper operation (bitshift)

**BEFORE:**



**AFTER:**



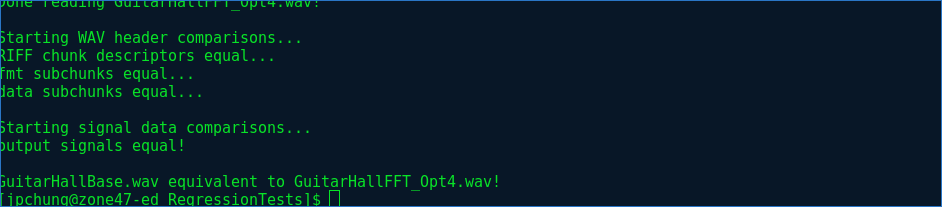
This optimization was implemented for all multiplications by 2 in the FFT convolution program, including parts *not shown* in the **AFTER**.

This fourth optimized version was then compared the previous optimization for improvements in runtime performance:

* inputFile: **GuitarDry.wav**
* impulseFile: **IRBigHall.wav**
* outputFile: **GuitarHallFFT\_Opt3.wav, GuitarHallFFT\_Opt4.wav**

|  |  |  |
| --- | --- | --- |
| **Program version** | **Time (s)** | **Reduction (%)** |
| Optimization 3 (convolveFFT\_Opt3.cpp) | 8.262 | N/A |
| Optimization 4 (convolveFFT\_Opt4.cpp) | 8.059 | 2.457 |

The output file of the optimized version was also checked for equivalence with regression testing:

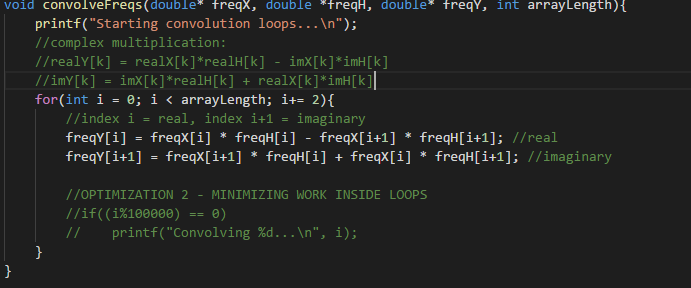


# CODE-Tuning optimization #5 – MINIMIZE ARRAY REFERENCES

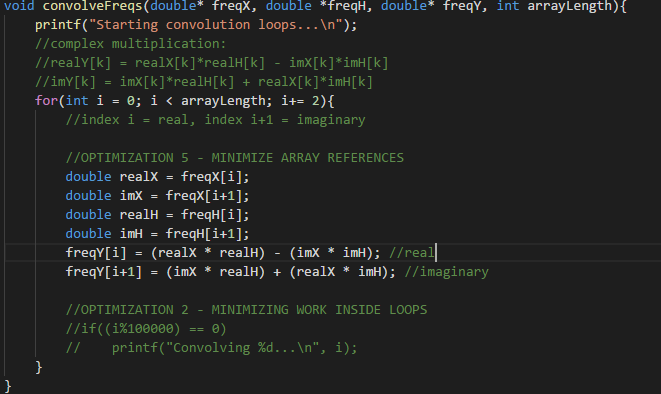
To optimize the FFT convolution even further, one last code-tuning was performed:

* **Minimize Array References**
  + Store common array elements as values instead of continuously referencing

**BEFORE:**



**AFTER:**



The fifth optimized version was then compared the previous optimization for improvements in runtime performance:

* inputFile: **GuitarDry.wav**
* impulseFile: **IRBigHall.wav**
* outputFile: **GuitarHallFFT\_Opt3.wav, GuitarHallFFT\_Opt4.wav**

|  |  |  |
| --- | --- | --- |
| **Program version** | **Time (s)** | **Reduction (%)** |
| Optimization 4 (convolveFFT\_Opt4.cpp) | 8.059 | N/A |
| Optimization 5 (convolveFFT\_Opt5.cpp) | 7.958 | 1.253 |

The output file of the optimized version was also checked for equivalence with regression testing:

