Assignment 2: FFT Optimization (15%)

Fall 2021 – CPSC 501

Due at 23:59, Nov. 6 on D2L

Assignment policy:

- This is an **individual** assignment, so the work you hand in must be your own. Any external sources used must be properly cited (see below).
- Extensions will not be granted to individual students. Requests on behalf of the entire class will only be considered if made more than 24h before the original deadline.
- Some tips to avoid plagiarism in your programming assignments (taken from previous offerings of the course):

  1. Cite all sources of code that you hand in that are not your original work. You can put the citation into comments in your program. For example, if you find and use code found on a web site, include a comment that says, for example:

     `# the following code is from https://www.quackit.com/python/tutorial/python_hello_world.cfm.`

     Use the complete URL so that the marker can check the source.
  2. Citing sources avoids accusations of plagiarism and penalties for academic misconduct. However, you may still get a low grade if you submit code that is not primarily developed by yourself.
  3. Discuss and share ideas with other programmers as much as you like, but make sure that when you write your code that it is your own. A good rule of thumb is to wait 20 minutes after talking with somebody before writing your code. If you exchange code with another student, write code while discussing it with a fellow student, or copy code from another person's console, then this code is not yours.
  4. Collaborative coding is strictly prohibited. Your assignment submission must be strictly your code. Discussing anything beyond assignment requirements and ideas is a strictly forbidden form of collaboration. This includes sharing code, discussing code itself, or modelling code after another student's algorithm. You can not use (even with citation) another student's code.
  5. We will be looking for plagiarism in all code submissions, possibly using automated software designed for the task. Note that this still applies to the current offering.
  6. Remember, if you are having trouble with an assignment, it is always better to go to your TA and/or instructor to get help than it is to plagiarize.

Instructions:

The goal of this assignment is to optimize, in stages, the performance of a convolution reverb program. Convolution reverb is an audio digital signal processing technique where a "dry" recording of an instrument (i.e. a recording without reverberation) is convolved with the impulse response of an acoustical space such as a concert hall. The result of the convolution is a sound file where the instrument sounds as if it were playing in the actual concert hall. This is a commonly used, but computationally intensive, technique for adding natural-sounding reverberation to recorded sounds.

You are to create a command-line program that takes in a dry recording and an impulse response, and produces the convolved signal. It should be invoked from the command line as follows:

```
convolve inputfile IRfile outputfile
```

where convolve is the name of your program, inputfile is the dry recording in .wav format, IRfile is the impulse response in .wav format, and outputfile contains the convolved signal in .wav format. All .wav files should be monophonic, 16-bit, 44.1 kHz sound files (at least initially, before attempting the bonus part of the assignment). Sample audio and impulse response files are available on D2L.

As in the first assignment, you will be using GitLab to maintain version control and to share your final project with the TAs. Your assignment should be kept in a GitLab repository titled "CPSC_501_A2".

Baseline program: Create an initial version of your program where the convolution is implemented directly in the time domain. You will find this version of your program quite slow. Measure the run-time performance of your program using a dry recording that is at least thirty seconds long and an impulse response that is at least 2 seconds long. You will reuse these same inputs for timing measurements after each optimization that you do in the later stages of the assignment. There are many suitable dry recordings and impulse responses available on the Internet as well as on the course D2L site. You can find various utility programs online to convert sound files of different types (e.g. .aiff or .snd) to the .wav format.

Although the program may be implemented in any programming language, it would be best to use a language supported by the GCC compiler, since the gprof profiler is what we will be using in class to work through examples in C++ (also note the required compiler optimization). The GCC compiler is available on the CPSC servers, and can easily be installed on a personal machine.

Algorithmic optimization: Create a second version of your program where you re-implement the convolution using a frequency-domain convolution algorithm. A handout will be provided summarizing the approach discussed in lecture. Measure the run-time performance of this second version of program using the same inputs

that you used for the baseline program. Be sure to use version control, profiling, and regression testing as part of a disciplined process of optimization.

Compiler-level optimization and code-tuning: Use compiler-level optimization and manual code tuning to further optimize the performance of your program. Do your improvements step by step, measuring and testing at each stage. Be sure to test, profile, and commit your code each time you make a change. Use at least 4 different manual code-tuning techniques and at least one compiler optimization.

Report: Create a formal written report that describes how you optimized your code at each step. You must show each version of your program, and describe what changes you made at every stage of the process. Include relevant code excerpts to illustrate the changes you made. You must also quantify the improvements with your timing measurements, and describe the regression tests you performed. Use tables and/or graphs to help illustrate how you improved performance at each stage of your work. In addition, at the beginning of the written report, you need to include **directions for the TA to access your GitLab project**. This is how they will be able to access your code and commit history, so double-check this works correctly before submitting. The expected length of the report is about 2–3 pages (not including code excerpts) with standard font and margins, but there are no strict requirements. It is essential, however, that your report is clear, easy to read, thorough, and written in complete sentences.

Notes on regression testing: Rather than writing unit tests for this assignment and comparing the output to some prepared standard, you will be using **regression testing** to ensure your program remains correct after each optimization step. To do this, you will need to keep the output from your initial baseline program for a particular pair of sound/IR input files. You should be able to check whether your baseline output is correct by simply listening to the output file. Then, after each optimization, compare this baseline output to the output of your modified program on the same input files. If the two outputs are identical, you have confirmed that your optimizations have left the program function unchanged. My recommendation would be to write a bash script that performs the comparison automatically, and to include this in your version control and report.

Bonus (up to 10%): Elaborate your program so that it can handle stereo (i.e. 2-channel) impulse response files, and produce the appropriate stereo (2-channel) output file. In other words, your program will convolve a monophonic dry input sound with a stereo impulse response, and output a stereo sound file. Your program should be able to recognize automatically if the impulse response file has one or two channels. You can either implement the bonus as part of your baseline program, or after all optimizations. If you choose to implement the bonus feature, you need to indicate this clearly in your report. Sample stereo IR files are available on D2L.

Submission instructions:

Important: If you are in T05 (Chris' Monday morning tutorial), submit your assignment to **Navid**, who will be doing your grading. If you are in a different tutorial section, submit to your own TA.

Upload your written report as a **PDF file** to the Assignment 2 dropbox on D2L by 23:59 on November 6th. Make sure you **add the correct TA to your GitLab project with reporter access** using their email given on D2L. The TA will use the instructions in your report to access your GitLab project, through which they will grade your submission.

Rubric (100 pts total):

- Version control: Used Git/GitLab properly, Multiple small commits with informative messages (5 pts)
- Profiling: Tests are run after each improvement and the results are documented (5 pts)
- Regression testing: Tests are run to make sure correctness is preserved between changes (5 pts)
- Baseline program: Unoptimized program correctly performs convolution reverb in the time domain (20 pts)
- Optimizations: Evidence in version control and report of five clear and systematic optimizations. The first of these must be an algorithmic optimization, implementing the FFT. The remainder must contain a compiler optimization and at least four distinct code tuning optimizations (20 + 5 + 20 = 45 pts)
- Report: Description of each of the optimizations described above, with appropriate code excerpts shown. Report is thorough and written in full sentences. (15 pts)
- Logistics: Clear, working instructions on how to access GitLab project. Program can be run from the command line using the specified instruction (5 pts)
- Bonus: Solution can detect and handle stereo impulse response files (10 pts)

---

author: jcleahy@ucalgary.ca