CSCI 4118/6105 Lab 4:

Floating Point Precision in Geometric Algorithms

Winter 2022

Objective

1. Understand the cause for floating-point rounding errors in summation and mitigate the error.
2. Validate the implication of rounding errors in geometric algorithms and mitigate and compare different solutions' performance.

Pre-read/Terminologies

1. Floating Point Precision.
2. Geometric Algorithms (Convex Hull Graham Scan).

Resources

- For submission: https://git.cs.dal.ca/courses/2022-winter/csci-4118-6105/lab4/???? where ???? is your CSID

Procedure

Note: Undergraduate students are encouraged to attempt additional questions meant for graduate students and get bonus points.

Precision Summation Algorithms (Part 1):

1. Download and install the dependencies to run locally or use an online Jupyter Notebook Editor (Refer to the installation section in `README.md`).
2. The default code computes Pi using Nilakantha's infinite series. Try the different summation algorithms in `summation.ipynb/summation.py` to validate the extent of precision with other inputs and compare the time taken [5].

Robustness Problems in Planar Convex Hull Computation (Part 2):

1. Follow the Part 2 installation instructions in `README.md` to build and run the robust and non-robust convex hull algorithms.
2. Try more examples in the repository inside the `data` folder and compare the differences between robust and non-robust convex hull algorithms.

Questions

1. **All Students:** Follow the procedure and report your findings for both part-1 and part-2. Show your work by summarizing in `output.md`
    a. Try with at least three other inputs other than the given examples.
    b. Show testing evidence by including the inputs and respective outputs.
    c. Finally, summarize your findings, look for patterns and compare the performance of algorithms.

2. **Graduate students:**
   Write a program to find the convex hull for a given set of points using floating-point value with arbitrary precision [1] (with the same input and output contract as robust and non-robust convex hull algorithms in part-2).
   a. Suggestion: Use `MP_Float` in the CAGL library [2].
   b. Compare the time taken by the robust convex hull algorithm and the algorithm you have written using arbitrary precision float (include the inputs, respective outputs and time taken).

Submission

Note:

- For submission - git add, commit and push the answers to the lab4/???? repository and verify the submission in the GitLab web interface.
- To keep it consistent, highly recommend using C++ for part-2

1. **All Students:** Answer Q1 (in `output.md`)
2. **Graduate Students:** Answer Q2 (implement and submit a new file `exact_convex_hull.cpp` and summarize findings in `output.md`)

Grading

Task	10 Points	5 Points	0 Points
1	Q1 Thoroughly tested and reported findings in `output.md`.	Q1 tested partially and reported findings in `output.md`.	No evidence of testing.

Task	5 Points	3 Points	0 Points
2	Q2 program implemented, tested and compared with given convex hull algorithms.	Algorithm implemented, logically correct (pseudo-code), and/or evidence for testing and explanation.	Incorrect answer or not answered.

Grading for Under-graduate Students:

	Task 1	**Task 2**	**Total**
Points	10	-	10

Grading for Graduate Students:

	Task 1	**Task 2**	**Total**
Points	10	5	15

Under-graduate students can get an additional +1% for attempting graduate questions.

References

[1] “The Exact Computation Paradigm,” *www.cgal.org*. https://www.cgal.org/exact.html (accessed Mar. 11, 2022).

[2] “CGAL 5.4 - Manual: Main Page,” *doc.cgal.org*. https://doc.cgal.org/latest/Manual/index.html (accessed Mar. 11, 2022).

[3] "Mörig, M. (2015, November). Another Classroom Example of Robustness Problems in Planar Convex Hull Computation. In *International Conference on Mathematical Aspects of Computer and Information Sciences* (pp. 446-450). Springer, Cham. http://wwwisg.cs.uni-magdeburg.de/ag/ClassroomExample/ "

[4] L. Kettner, K. Mehlhorn, S. Pion, S. Schirra, and C. Yap, “Classroom examples of robustness problems in geometric computations,” Computational Geometry, vol. 40, no. 1, pp. 61–78, May 2008, doi: 10.1016/j.comgeo.2007.06.003.

[5] “Pi,” *Wikipedia*, Feb. 27, 2022. https://en.wikipedia.org/wiki/Pi#Rate_of_convergence (accessed Mar. 14, 2022).