ECE 785 – Spring 2018 – Topics in Computer Design
Project 1: AHRS Performance Analysis and Optimization

Overview

An attitude and heading reference system (AHRS) uses acceleration, rotation and magnetic field sensors to determine an object's relative location and direction.

You are to profile and optimize the execution time of Madgwick's MARG filter on the Beaglebone Black Wireless. Use both source code modifications and compiler flags to improve performance. You are NOT expected to use Neon SIMD operations for this project.

Sources and Background Reading

Source code and documentation are available in a zip file on Moodle. Refer to madgwick_internal_report.pdf for details on the algorithm.

Use the MARG filter_update function in MadgwickAHRS.[c|h].

- The function's outputs are global variables: the quaternion components SEq_1, SEq_2, SEq_3 and SEq_4, and magnetic flux components b_x and b_z.
- Note that the function parameters have the same name as corresponding global variables, which may be confusing. Feed free to rename either the parameters or the globals.

Create a driver function which.

- Calculates repeatable pseudorandom input data (e.g. using the rand or random function) within the following ranges:
  - a_x, a_y, a_z: [-10.0, 10.0] g
  - w_x, w_y, w_z: [-1000.0, 1000.0] radians/sec
  - m_x, m_y, m_z: [-100.0, 100.0] microtesla
- Calls your optimized function with the input data, measuring its duration with clock_gettime.
- Compares the outputs against the expected outputs (from the original function). Note that the function changes the output variables, so you'll need to save and restore them to duplicate tests.
  The maximum allowed error for any output is 0.01%.

When evaluating function timing, don't include the validation time.

Follow this basic optimization process for this project:

1. Use perf (or gprof) to profile the application and find the dominant instructions. Use clock_gettime to measure time durations.
2. Examine the object code and compare it with the source code.
3. Tune the compiler flags and the source code to improve the execution time.
4. Repeat as necessary.

Track the amount of time taken to implement and measure each optimization successfully. After you have the first version of the code running, you are allowed a total of **four hours** to optimize the code. This does not include time writing your report.

Note: you are NOT expected to use SIMD instructions for this project. However, it is fine if the compiler generates them for your source code.

Deliverables

1. Zip file with source code and makefile.
2. A report which includes the following:
    a. Optimizations: A step-by-step explanation of your optimization process, with each step containing these sections:
        i. Initial run time, and profile data indicating hot spots
        ii. Explanation of optimization applied, with excerpts of source code
        iii. Time taken to implement optimization
        iv. Performance of optimized code
        v. Analysis
    b. Scatter-plot showing program speed-up (vertical) vs. elapsed optimization time (horizontal)