Kahan Summation and it's benefits

Rodolfo "Ofo" Croes
Department of Computing Sciences
Coastal Carolina University
Conway, SC, United States
rjcroes@coastal.edu

***Abstract*—In this paper I will be discussing the Kahan Summation algorithm, what it is, how it works, what applications it is used in, and comparing it to other global summation algorithms aswel as providing code example(s) for said algorithms.**

***Keywords*—parallel, summation, gobal, algorithms**

I. Introduction

As we started learning about sumation methods in parallel in class, the question that always comes to mind is "which is the algorithm with the least amount of error and is it the most efficient at doing this sumation?". This question always prompts up debates because there is no "correct" answer because each algorithm created thus far has their ups and their downs. However, from what little we have learned in class about different types of global summation algorithms, I had to do some self research via the book "Parallel and High Performance Computing" by Robert Robey and Yuliana Zamora. In here I have discovered that there are a variety of algorithms to calculate the global summation of values created by processes. Some examples of these include Long Double, Pairwise Summation, Knuth summation and Kahan Summation. Each of these with their positives and negatives.

II. Definition

The Kahan summation (developed by William Kahan in 1965) is seen as the most practical method to perform a global sumation. As described by Robert Robey in Chapter 5.6 of his book "Parallel and High Performance Computing" he states that "It uses an additional double variable to carry the remainder of the operation, in effect doubling the effective precision. The Kahan summation is most appropriate for a running summation when the accumulator will be the larger of the two values."[1] This method of computing the global sum reduces the error in the output by adding a sequence of floating point numbers instead of the approach our class took which consisted of just adding the value that was assigned to each process to one another by using MPI_Sendrecv() to allow the processes to communicate with one another. While the method we learned about in class is more straight-forward, the Kahan summation algorithm is better suited for percision and for larger number of values that need to be summed up.

III. Usage

The main usage of the Kahan summation is (as the name suggests) to sum up a list of floating point numbers. This can be seen implemented by Dr. John D. Cook in his blog post "Summing an array of floating point numbers"(2019). In this blog post Cook shows how the Kahan summation is used to give a more precise answer to a summation of 32-bit floats. Cook shows an example of the precision increase in his work near the end of the post where he has stated "The naive method is correct to 6 significant figures while Kahan's method is correct to 9 significant figures. The error in the former is 5394 times larger.".[2] This accuracy and precision is why this algorithm is praised among those who need to get a global summation for their respective programs. Another benefit of using this algorithm compared to others is that regardless of the utilization of a more complex data type (being float), the run speed is not majorly affected by this. The reason for this as stated by Robert Robey is that "the Kahan summation takes about four floating point operations instead of one. But the data can be kept in registers or L1 cache making the operation less expensive than this. Vectorized implementations have essentially made the operation the same cost as the standard summation". (Robey 169-170) This method of storing the data in registers or L1 cache makes it more efficient to read and write data because of the hierarchy of memory where registers are significantly more fast compared to other storage options.

IV. Algorithm Analysis

In the figure below, the program can be seen creating a double-double data type with one of them holding the sum and the other holding the correction values. There is also a struct declared to hold the sum and the correction. After the instantiation of the sum and correction (which in this case are the partials sums), the loop starts. This loop increases the maximum number of cells and every iteration of the loop the program calculated the new correction value and local sum. Once this is done, the loop is finished and at the end of the program the global sum is added with the last correction. Then the program returns the final global sum.

```
double do_kahan_sum(double *var, long ncells)
{
        struct esum_type { #A
                double sum; #A
                double correction; #A
        }; #A
        double corrected_next_term, new_sum;
        struct esum_type local;
        local.sum = 0.0;
        local.correction = 0.0;
for (long i = 0; i < ncells; i++) {
                corrected_next_term = var[i] +

local.correction;
                new_sum = local.sum +
local.correction;
local.correction = corrected_next_term - (new_sum
– loca
l.sum);
                local.sum =  new_sum;
}
double dsum = local.sum + local.correction;
return(dsum); #C
}
```

Fig, 1. Example of the Kahan algoritm from High Performance Computing (169-170)

V. Comparisons

When comparing some global summation methods, we usually compare the run time and the amount of error. In the table below, there can be seen all the comparisons made by Robert Robey in his book Chapter 5.6 when it comes to run-times and errors in all of the methods that were discussed in said chapter.

Method	Error	Run-Time
Double	-1.99e-09	0.116
Long Double	-1.31e-13	0.118
Pairwise Summation	0.0	0.402
Kahan Summation	0.0	0.406
Knuth Summation	0.0	0.704
Quad double	5.55e-17	3.010

Fig. 2. Precision and run-time results from global sum techniques in High Performance Computing(171)

Some conclusions that can be drawn from this table include that commonly, when the run time is faster, the amount of error increases. As can be seen by comparing the Double or Long Double method to the Pairwise Summation or the Kahan Summation. And the contenders for “best” method for global summation is between Kahan summation and Pairwise summation. The reason why the Kahan sumation is considered better is because the pairwise summation works best when it is used on a single processor machine. As stated by Robert Robey in Parallel and High Performance Computing “The pairwise summation is a surprisingly simple solution to the global sum problem, especially within a single processor.The simplicity of the pairwise summation becomes a little more complicated across processors. If the algorithm remains true to its basic structure, a communication may be needed at each step of the recursive sum.” (168) Another method that can be compared to Kahan summation is the Knuth summation because both of these methods hace an error of 0.0. The reason why Knuth summation is worse than Kahan is because while the Kahan summation uses four floating points, the Knuth summation uses seven floating points and these complex variables add up to the run-time suffering some loss.

References

[1] *R. Robey and Y. Zamora, Parallel and High Performance Computing, 1st ed., vol. 1. Shelter Island, NY: Manning Publications, 2020.*

[2] *J. Cook, Summing an array of floating point numbers, blog, John Cook consulting, https://www.johndcook.com/blog/2019/11/05/kahan/ , 5 November 2019.*