City University of New York (CUNY)
CUNY Academic Works

Computer Science Technical Reports | CUNY Academic Works

2007

TR-2007013: Error-free Computations via Floating-Point Operations: The Code

V. Y. Pan

B. Murphy

G. Qian

R. E. Rosholt

How does access to this work benefit you? Let us know!

More information about this work at: https://academicworks.cuny.edu/gc_cs_tr/293
Discover additional works at: https://academicworks.cuny.edu

This work is made publicly available by the City University of New York (CUNY).
Contact: AcademicWorks@cuny.edu

Error-free Computations via Floating-Point Operations: The Code

V. Y. Pan[1,3], B. Murphy[1], G. Qian[2], R. E. Rosholt[1]
[1] Department of Mathematics and Computer Science,
Lehman College, The City University of New York,
Bronx, NY 10468 USA
victor.pan/brian.murphy/rhys.rosholt@lehman.cuny.edu
[2] Ph.D. Program in Computer Science,
The City University of New York,
New York, NY 10036 USA
gqian@gc.cuny.edu
[3] http://comet.lehman.cuny.edu/vpan/

Abstract

Division-free arithmetic computations can be boiled down to summation due to Dekker/Veltkamp's algorithm of 1971. The known double-precision numerical algorithms for summation are highly effective but limited by rounding errors. Our new summation algorithms relax this limitation although they still almost entirely amount to double-precision additions. The efficiency of the algorithms is confirmed by our analysis and extensive tests.

1 Introduction

This report is an addendum to our previous report, **Error-free Computations via Floating-Point Operations**. In this report, we reproduce our code for the basic Algorithm 5.1 of the previous report, which succeeded in all our extensive experimental computations of nearly vanishing sums.

2 Implementation and testing

The presented summation algorithms have been implemented in MATLAB and have been extensively tested for computing nearly vanishing sums. The algorithms have never failed and have produced correct answers in all our tests.

Below is our code for Algorithm 5.1 which depends on the `Split` function (Algorithm 4.1) and two sorted summations. The function `sumHi2Lo` adds the summands

in descending order of their absolute values. It is applied to the computation of s^* and x in order to provide accuracy in the highest order bits where cancellations will occur. The function `sumLo2Hi` adds the summands in ascending order of their absolute values. It is applied to the computation of s^*_+, an estimate of the sum of the absolute values of the summands, to achieve high accuracy within double precision.

```
function [s, delta_plus] = leadingBitsOfSum(summands);

p = 52;
c = 1.12;

while ( 1 )
     h          = length(summands);
     gamma      = h / (pow2(p + 1) - h);
     e_         = (1 + c * gamma) * c * gamma;
     s          = sumHi2Lo(summands);
     s_plus     = sumLo2Hi(abs(summands));
     delta_plus = e_ * s_plus;
     s_tilda    = abs(s) + delta_plus * 4;

     if ( delta_plus > abs(s) / 2 )
          [significand, d] = log2(s_tilda);
          d                = d + 2;
          for i = 1 : h
               [significand, e] = log2(summands(i));
               g                = p + 1 - (e - d);
               [x(i), y(i)]     = split(summands(i), g);
          end
          x        = sumHi2Lo(x);
          summands = [y, x];
     else
          break;
     end
end
```