Accurate Computing Elementary Symmetric Functions

Hao Jiang[1], Stef Graillat[2],Roberto Barrio[3]

[1] School of Science and The State Key Laboratory for High Performance Computation, National University of Defense Technology, China
[2] PEQUAN, LIP6, Université Pierre et Marie Curie, France
[3] Dpto. de Matemática Aplicada and IUMA, Universidad de Zaragoza, Spain

Introduction

This work concerns with the numerical computation of the k-th elementary symmetric function (ESF) with floating-point inputs $X = (x_1, \ldots, x_n)$, which is defined as

$$S_k^{(n)}(X) = \sum_{1 \le \pi_1 < \cdots < \pi_k \le n} x_{\pi_1} x_{\pi_2} \cdots x_{\pi_k},\ 1 \le k \le n.$$

We focus mainly on the case $2 \le k \le n-1$. For $k = 1$, the problem simplifies to the computation of the sum of floating-point numbers, and for $k = n$, to the computation of floating-point product. The classic and widely-used method is the so-called *Summation Algorithm*, denoted by `SumESF`, which is essentially the algorithm used by MATLAB's `poly`.

Summation Algorithm

Input: $X = (x_1 \ldots x_n)$
Output: k-th ESF $S_k^{(n)}(X) = S_k^{(n)}$
function $S_k^{(n)}$=$\texttt{SumESF}(X,k)$
$S_0^{(i)} = 1,\ 1 \le i \le n-1;\ S_j^{(i)} = 0,\ j > i;\ S_1^{(1)} = x_1;$
For $i = 2 : n$
For $j = \texttt{Max}\{1, i+k-n\} : \texttt{Min}\{i, k\}$
$S_j^{(i)} = S_j^{(i-1)} + x_i S_{j-1}^{(i-1)};$
end
end

The error analysis has been considered in [1], and the result implies that the algorithm is forward stable. We present the relative forward error bound as follows,

$$\left| \frac{\texttt{SumESF}(X,k) - S_k^{(n)}(X)}{S_k^{(n)}(X)} \right| \le \frac{1}{k}\gamma_{2(n-1)}\texttt{cond}(S_k^{(n)}(X))$$

with

$$\texttt{cond}(S_k^{(n)}(X)) = \frac{k S_k^{(n)}(|X|)}{|S_k^{(n)}(X)|},$$

where $\gamma_n = nu/(1-nu)$ with u be the rounding error unit(in double precision $u = 2^{-53}$) and absolute value is to be understood componentwise. However, when performed in floating-point arithmetic, the computed result by `SumESF` may still be less accurate than expected due to cancelations. This is why a more accurate algorithm is required.

Error Free Transformation

For a pair of floating-point numbers $a, b \in \mathbb{F}$, when no underflow occurs, there exists a floating-point number y satisfying $a \circ b = x + y$ with $\circ \in \{+, -, \times\}$, where $x = \mathrm{fl}(a \circ b)$ is the usual floating-point approximation and y represents the exact rounding error. The transformation $(a, b) \longrightarrow (x, y)$ is regarded as an EFT. The EFT algorithms for the addition and product of two floating-point numbers used in `CompSumESF` are `TwoSum` and `TwoProd` algorithms, respectively. One can see the details about their properties in [2].

function $[x, y] = \texttt{TwoSum}(a, b)$
$x = a \oplus b$
$z = x \ominus a$
$y = (a \ominus (x \ominus z)) \oplus (b \ominus z)$

function $[x, y] = \texttt{Split}(a)$
$c = \text{factor} \otimes a$ (in double precision factor $= 2^{27} + 1$)
$x = c \ominus (c \ominus a)$
$y = a \ominus x$

function $[x, y] = \texttt{TwoProd}(a, b)$
$x = a \otimes b$
$[a1, a2] = \texttt{Split}(a)$
$[b1, b2] = \texttt{Split}(b)$
$y = a2 \otimes b2 \ominus (((x \ominus a1 \otimes b1) \ominus a2 \otimes b1) \ominus a1 \otimes b2)$

Compensated Algorithm

By introducing error-free transformation (EFT) to the traditional Summation Algorithm, we propose a fast and accurate compensated algorithm, which is denoted by `CompSumESF`.

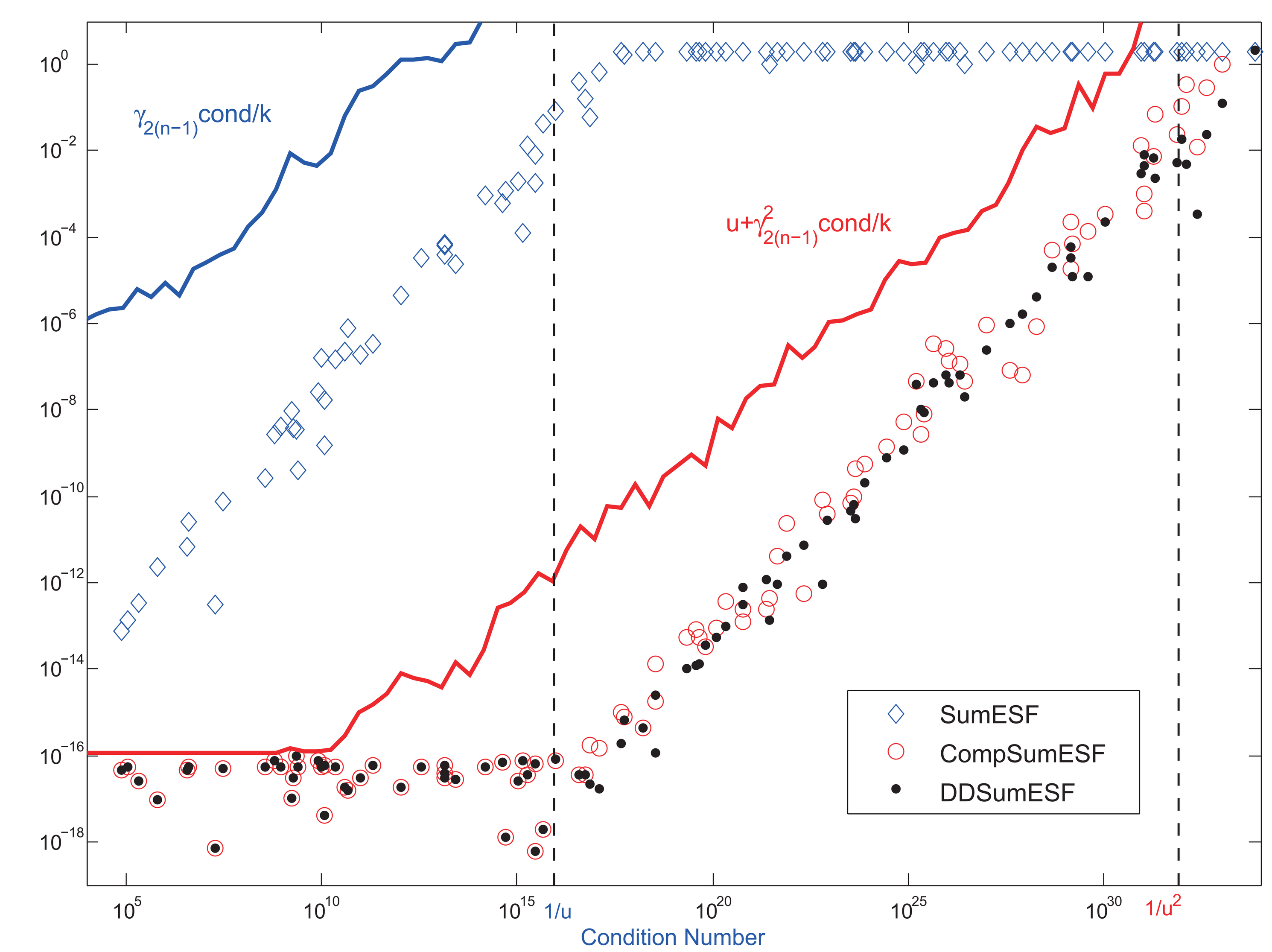
Compensated Summation Algorithm

Input: $X = (x_1 \ldots x_n)$
Output: k-th ESF $\overline{S}_k^{(n)}(X) = \overline{S}_k^{(n)}$
function $\overline{S}_k^{(n)}$=$\texttt{CompSumESF}(X,k)$
$\widehat{S}_0^{(i)} = 1,\ 1 \le i \le n-1;\ \widehat{S}_j^{(i)} = 0,\ j > i;\ \widehat{S}_1^{(1)} = x_1;$
$\widehat{\epsilon S}_j^{(i)} = 0, \forall\, i, j$
For $i = 2 : n$
For $j = \texttt{Max}\{1, i+k-n\} : \texttt{Min}\{i, k\}$
$[p, \beta_j^{(i)}] = \texttt{TwoProd}(x_i, \widehat{S}_{j-1}^{(i-1)});$
$[\widehat{S}_j^{(i)}, \sigma_j^{(i)}] = \texttt{TwoSum}(\widehat{S}_j^{(i-1)}, p);$
$\widehat{\epsilon S}_j^{(i)} = \widehat{\epsilon S}_j^{(i-1)} \oplus (\beta_j^{(i)} \oplus \sigma_j^{(i)}) \oplus x_i \otimes \widehat{\epsilon S}_{j-1}^{(i-1)}$
end
end
$\overline{S}_k^{(n)} = \widehat{S}_k^{(n)} \oplus \widehat{\epsilon S}_k^{(n)}$

Then, the forward error bound of our method is

$$\left| \frac{\texttt{CompSumESF}(X,k) - S_k^{(n)}(X)}{S_k^{(n)}(X)} \right| \le u + \frac{1}{k}\gamma_{2(n-1)}^2\texttt{cond}(S_k^{(n)}(X)),$$

It is interesting to compare our method with the approach using Bailey's double-double arithmetic denoted by `DDSumESF`. All the results about accuracy measurements are reported on Figure, which imply that the result computed by our method is as accurate as if computed in twice the working precision. When the problem is not too ill-conditioned it yields nearly full accuracy. We perform numerical tests about measured running time, using compiler VC++9.0, on a laptop with a Intel(R) Core(TM) i5-2520M processor, with two cores each at 2.50Ghz. The results show that `CompSumESF` is as accurate as `DDSumESF` but only requires on the average 57% of its measured running time. Moreover, our method only requires addition and multiplication of floating-point numbers in the same working precision as the given data. As a consequence, it seems that our method is a simple, fast and accurate algorithm to compute elementary symmetric functions.



Application

As an application, the ESFs appear when expanding a linear factorization of a polynomial

$$\prod_{i=1}^{n}(x - x_i) = \sum_{i=0}^{n} c_i x^i = \sum_{i=0}^{n} (-1)^{n-i} S_{n-i}^{(n)}(X) x^i.$$

It is an option to use our method to accurately evaluate polynomial's coefficients from zeros, specially to compute characteristic polynomials from eigenvalues. The computation of ESFs is also an important part of conditional maximum likelihood estimation of item parameters under the Rasch model in psychological measurement [3]. It is promising that our method, improving the numerical accuracy, can allow much more items to be calibrated.

References

[1] R. Rehman and I.C.F. Ipsen. Computing Characteristic Polynomials from Eigenvalues *SIAM. J. Matrix. Anal. Appl.*, 32(1):90–114, 2011.

[2] T. Ogita, S.M. Rump, and S. Oishi. Accurate sum and dot product. *SIAM J. Sci. Comput.*, 26:1955-1988, 2005.

[3] F.B. Baker, and M.R. Harwell. Computing elementary symmetric functions and their derivatives: A didactic. *Appl. Psychol. Meas.*, 20(2):169–192, 1996.