Accurate Computing Elementary Symmetric Functions

Hao Jiang1, Stef Graillat2, Roberto Barrio3
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 Depto. de Matemática Aplicada and IUMA, Universidad de Zaragoza, Spain

Introduction

This work focuses on 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

\[
S_k(X) = \sum_{1 \leq i_1 < i_2 < \cdots < i_k \leq n} x_{i_1} x_{i_2} \cdots x_{i_k}, \quad 1 \leq k \leq n.
\]

We focus mainly on the case \( 2 \leq k \leq n - 1 \). 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 \text{SumESF} \( \hat{\Sigma} \), which is essentially the algorithm used by MATLAB’s \text{poly}.

Summation Algorithm

\[
\begin{align*}
\text{Input: } X &= [x_1, \ldots, x_n] \\
\text{Output: } k\text{-th ESF } S_k^n(X) &= S_k^n \\
\text{function } S_k^n &= \text{SumESF}(X, k) \\
S_k^n &= 1, 1 \leq i \leq n - 1; \quad S_k^n = 0, j > i; \quad S_k^n = x_i \\
\text{end}
\end{align*}
\]

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,

\[
\frac{|\text{SumESF}(X, k) - S_k^n(X)|}{S_k^n(X)} \leq \frac{1}{2^{2(n-k-1)}} \text{cond}(S_k^n(X))
\]

with

\[
\text{cond}(S_k^n(X)) = \frac{S_k^n(\lambda X)}{S_k^n(X)}
\]

where \( \lambda = \mu \cdot (1 - u) \) with \( u \) 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 \text{SumESF} may still be less accurate than expected due to cancellations. This is why a more accurate algorithm is required.

Compensated Summation Algorithm

\[
\begin{align*}
\text{Input: } X &= [x_1, \ldots, x_n] \\
\text{Output: } k\text{-th ESF } S_k^n(X) &= S_k^n \\
\text{function } T_k^n &= \text{CompSumESF}(X, k) \\
T_k^n &= 1, 1 \leq i \leq n - 1; \quad T_k^n = 0, j > i; \quad T_k^n = x_i \\
\text{end}
\end{align*}
\]

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,

\[
\frac{|\text{CompSumESF}(X, k) - S_k^n(X)|}{S_k^n(X)} \leq \frac{1}{2^{2(n-k-1)}} \text{cond}(S_k^n(X))
\]

with

\[
\text{cond}(S_k^n(X)) = \frac{S_k^n(\lambda X)}{S_k^n(X)}
\]

where \( \lambda = \mu \cdot (1 - u) \) with \( u \) 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 \text{CompESF} may still be less accurate than expected due to cancellations. 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 \( a + b = x + y \) with \( x = \text{fl}(a + b) \) is the usual floating-point approximation and \( y \) represents the exact rounding error. The transformation \( (a, b) \rightarrow (x, y) \) is regarded as an EFT. The EFT algorithms for the addition and product of two floating-point numbers used in \text{CompESF} are \text{TreuHe} and \text{TwoProd} algorithms, respectively. One can see the details about their properties in [2].

CompESF

\[
\begin{align*}
\text{Input: } X &= [x_1, \ldots, x_n] \\
\text{Output: } k\text{-th ESF } S_k^n(X) &= S_k^n \\
\text{function } S_k^n &= \text{CompESF}(a, b) \\
x &= a \oplus b \\
y = (a \odot (x \oplus z)) \oplus (b \odot z) \\
\text{end}
\end{align*}
\]

CompESF

\[
\begin{align*}
\text{Input: } X &= [x_1, \ldots, x_n] \\
\text{Output: } k\text{-th ESF } S_k^n(X) &= S_k^n \\
\text{function } S_k^n &= \text{CompESF}(a, b) \\
x &= a \oplus b \\
y = (a \odot (x \oplus z)) \oplus (b \odot z) \\
\text{end}
\end{align*}
\]

Application

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

\[
\prod_{i=1}^{n}(x - x_i) = \sum_{i=0}^{n} \gamma_i x^i.
\]

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 \text{ESF} is also an important part of conditional maximum likelihood estimation of form parameters under the Flach model in psychological measurement [3]. It is promising that our method, improving the numerical accuracy, can allow much more items to be calibrated.

References