Accurate and Fast Evaluation of Elementary Symmetric Functions

Stef Graillat

LIP6/PEQUAN - Université Pierre et Marie Curie (Paris 6) - CNRS

Joint work with Hao Jiang and Roberto Barrio

21st IEEE International Symposium on Computer Arithmetic
Austin, Texas, USA, April 7-10, 2013
Motivations (1/2)

- **Polynomials** play a central role in computational and applied mathematics

- The determination of the *zeros of polynomials* is a classical problem of computational mathematics

- **Inverse problem**: given the zeros, determine the *coefficients of the polynomial*
Motivations (2/2)

Characteristic polynomial of a $n \times n$ matrix $A$

\[
\det(\lambda I - A) = \lambda^n + c_1 \lambda^{n-1} + \cdots + c_{n-1} \lambda + c_n
\]

\[
c_1 = \text{trace}(A) \quad \quad \quad c_n = \det(A)
\]

Eigenvalues: $(\lambda_i)$ for $i = 1, \ldots, n$

\[
c_1 = \sum_{i=1}^{n} \lambda_i \quad \quad \quad c_n = \prod_{i=1}^{n} \lambda_i
\]

→ the $c_i$ are elementary symmetric functions of the $\lambda_i$
Outline of the talk

- Motivations
- Classical Summation Algorithm
- Error-free transformations
- Compensated Summation Algorithm
- Conclusion and future work
Elementary Symmetric Functions (ESF)

**Definition 1**

The *k*-th Elementary Symmetric Function (ESF) associated with a vector of *n* numbers \(X = (x_1, \ldots, x_n)\) is defined by

\[
S^{(n)}_k (X) = \sum_{1 \leq \pi_1 < \cdots < \pi_k \leq n} x_{\pi_1} x_{\pi_2} \cdots x_{\pi_k} \quad \text{with} \quad 1 \leq k \leq n
\]

For \(k = 0\), \(S^{(n)}_0 = 1\)

The *k*-th function \(S^{(n)}_k (X)\) consists of \(\binom{n}{k}\) summands

→ straightforward computation is very expensive
Applications of computing ESF

The ESFs appear when expanding a linear factorization of a polynomial

\[
\prod_{i=1}^{n} (x - x_i) = \sum_{i=0}^{n} c_i x^i = \sum_{i=0}^{n} (-1)^{n-i} S^{(n)}_{n-i}(x_1, \ldots, x_n) x^i
\]

One can evaluate polynomial’s coefficients \( \{c_i\}_{i=0}^{n} \) from its zeros \( \{x_i\}_{i=1}^{n} \), specially compute characteristic polynomials from eigenvalues

Part of conditional maximum likelihood estimation (CMLE) of item parameters under the Rasch model in psychological measurement. Accurate evaluation allows much more items to be calibrated

Thermodynamic properties of systems of fermions
Condition number of ESF

Condition numbers measure the sensitivity of the solution of a problem to perturbation in the data.

**Definition 2 (Condition number of the \( k \)-th ESF)**

\[
\text{cond}(S_{k}^{(n)}(X)) = \lim_{\epsilon \to 0} \sup \left\{ \frac{|S_{k}^{(n)}(X + \Delta X) - S_{k}^{(n)}(X)|}{\epsilon |S_{k}^{(n)}(X)|} : |\Delta X| < \epsilon |X| \right\}
\]

A direct calculation yields

\[
\text{cond}(S_{k}^{(n)}(X)) = \frac{kS_{k}^{(n)}(|X|)}{|S_{k}^{(n)}(X)|}
\]

In particular,

\[
\text{cond}(S_{n}^{(n)}(X)) = \text{cond}(\prod_{i=1}^{n} x_{i}) = n \quad \text{and} \quad \text{cond}(S_{1}^{(n)}(X)) = \text{cond}(\sum_{i=1}^{n} x_{i}) = \frac{\sum_{i=1}^{n} |x_{i}|}{|\sum_{i=1}^{n} x_{i}|}.
\]
Algorithm 1

**Input:** \( X = (x_1, \ldots, x_n) \) and \( k \)

**Output:** \( k \)-th ESF \( S_k^{(n)}(X) = S_k^{(n)} \)

**function** \( S_k^{(n)} \)=SumESF\((X, k)\)

\[
S_0^{(i)} = 1, \ 1 \leq i \leq n-1; \quad S_j^{(i)} = 0, \ j > i; \quad S_1^{(1)} = x_1;
\]

for \( i = 2 : n \)
  
  for \( j = \max\{1, i + k - n\} : \min\{i, k\} \)
    
    \[
    S_j^{(i)} = S_j^{(i-1)} + x_i S_{j-1}^{(i-1)};
    \]
  
end

end

\[
S_j^{(i)} = S_j^{(i)}(x_1, \ldots, x_i) = \sum_{1 \leq \pi_1 < \ldots < \pi_j \leq i} x_{\pi_1} x_{\pi_2} \ldots x_{\pi_j}
\]

Substitution of \( j = 1 : i \) for \( j = \max\{1, i + k - n\} : \min\{i, k\} \) makes it possible to compute all ESF simultaneously

→ **Algorithm used in MATLAB poly function**
Standard model of floating-point arithmetic

Assume floating point arithmetic adhering IEEE 754 with rounding to nearest with rounding unit $u$ (no underflow nor overflow)

Let $x, y \in \mathbb{F}$ and $\circ \in \{+, -, \cdot, /\}$.

The result $x \circ y$ is not in general a floating-point number

$$\text{fl}(x \circ y) = (x \circ y)(1 + \delta), \quad |\delta| \leq u$$

**IEEE 754 standard (2008)**

<table>
<thead>
<tr>
<th>Type</th>
<th>Size</th>
<th>Mantissa</th>
<th>Exponent</th>
<th>Unit rounding</th>
<th>Interval</th>
</tr>
</thead>
<tbody>
<tr>
<td>binary32</td>
<td>32 bits</td>
<td>23+1 bits</td>
<td>8 bits</td>
<td>$u = 2^{1-24} \approx 1.92 \times 10^{-7}$</td>
<td>$\approx 10^{\pm 38}$</td>
</tr>
<tr>
<td>binary64</td>
<td>64 bits</td>
<td>52+1 bits</td>
<td>11 bits</td>
<td>$u = 2^{1-53} \approx 2.22 \times 10^{-16}$</td>
<td>$\approx 10^{\pm 308}$</td>
</tr>
</tbody>
</table>

We denote

$$\gamma_n := \frac{nu}{1 - nu}$$
Theorem 1 (Rehman, Ipsen (2011))

If $X = (x_1, \ldots, x_n)$ is a vector of floating-point numbers, the computed $k$-th elementary symmetric function $\hat{S}_k^{(n)} = \hat{S}_k^{(n)}(X)$ by Algorithm 1 in floating-point arithmetic verifies

$$
\left| \frac{\hat{S}_k^{(n)} - S_k^{(n)}}{S_k^{(n)}} \right| \leq \frac{1}{k} \gamma_2 (n-1) \text{cond}(S_k^{(n)}), \ 2 \leq k \leq n - 1,
$$

$$
\left| \frac{\hat{S}_1^{(n)} - S_1^{(n)}}{S_1^{(n)}} \right| \leq \gamma_{n-1} \text{cond}(S_1^{(n)}) = \gamma_{n-1} \frac{\sum_{i=1}^{n} |x_i|}{|\sum_{i=1}^{n} x_i|}, \ k = 1,
$$

$$
\left| \frac{\hat{S}_n^{(n)} - S_n^{(n)}}{S_n^{(n)}} \right| \leq \frac{1}{n} \gamma_{n-1} \text{cond}(S_n^{(n)}) = \gamma_{n-1}, \ k = n.
$$

S. Graillat (Univ. Paris 6)
Getting more accuracy with compensated algorithms

Error-free transformations are properties and algorithms to compute the generated elementary rounding errors,

\[ a, b \text{ entries } \in \mathbb{F}, \quad a \circ b = \text{fl}(a \circ b) + e, \text{ with } e \in \mathbb{F} \]

Key tools for accurate computation

- fixed length expansions libraries: double-double (Briggs, Bailey, Hida, Li), quad-double (Bailey, Hida, Li)
- arbitrary length expansions libraries: Priest, Shewchuk
- compensated algorithms (Kahan, Priest, Ogita-Rump-Oishi, Graillat-Langlois-Louvet, etc.)
EFT for the summation

\[ x = \text{fl}(a \pm b) \Rightarrow a \pm b = x + y \quad \text{with} \quad y \in \mathbb{F}, \]


**Algorithm 2** (EFT of the sum of 2 floating point numbers with \(|a| \geq |b|\))

function \([x, y] = \text{FastTwoSum}(a, b)\)

\[
x = a \oplus b \\
y = (a \ominus x) \oplus b
\]

**Algorithm 3** (EFT of the sum of 2 floating point numbers)

function \([x, y] = \text{TwoSum}(a, b)\)

\[
x = a \oplus b \\
z = x \ominus a \\
y = (a \ominus (x \ominus z)) \oplus (b \ominus z)
\]
EFT for the product (1/3)

\[ x = \text{fl}(a \cdot b) \quad \Rightarrow \quad a \cdot b = x + y \quad \text{with } y \in \mathbb{F}, \]

Algorithm TwoProduct by Veltkamp and Dekker (1971)

\[ a = x + y \quad \text{and} \quad x \text{ and } y \text{ non overlapping with } |y| \leq |x|. \]

Algorithm 4 (Error-free split of a floating point number into two parts)

```plaintext
function \([x, y] = \text{Split}(a)\)
    factor = \(2^s + 1\) \quad \% \quad u = 2^{-p}, s = \lfloor p/2 \rfloor
    c = factor \otimes a
    x = c \ominus (c \ominus a)
    y = a \ominus x
```

S. Graillat (Univ. Paris 6) Accurate and Fast Evaluation of ESF 13 / 28
Algorithm 5 (EFT of the product of 2 floating point numbers)

function \([x, y] = \text{TwoProduct}(a, b)\)
\[
\begin{align*}
  x &= a \otimes b \\
  [a_1, a_2] &= \text{Split}(a) \\
  [b_1, b_2] &= \text{Split}(b) \\
  y &= a_2 \otimes b_2 \ominus (((x \ominus a_1 \otimes b_1) \ominus a_2 \otimes b_1) \ominus a_1 \otimes b_2)
\end{align*}
\]

Theorem 2

Let \(a, b \in \mathbb{F}\) and let \(x, y \in \mathbb{F}\) such that \([x, y] = \text{TwoProduct}(a, b)\). Then,
\[
  a \cdot b = x + y, \quad x = \text{fl}(a \cdot b), \quad |y| \leq u|x|, \quad |y| \leq u|a \cdot b|,
\]

The algorithm \text{TwoProduct} requires 17 flops.
Given $a, b, c \in \mathbb{F}$,

- $\text{FMA}(a, b, c)$ is the nearest floating point number $a \cdot b + c \in \mathbb{F}$

**Algorithm 6 (EFT of the product of 2 floating point numbers)**

```plaintext
function $[x, y] = \text{TwoProductFMA}(a, b)$
    $x = a \otimes b$
    $y = \text{FMA}(a, b, -x)$
```

The FMA is available for example on PowerPC, Itanium, Cell, Xeon Phi processors.
Compensated Summation Algorithm

**Algorithm 7**

**Input:** \( X = (x_1, \ldots, x_n) \) and \( k \)

**Output:** \( k \)-th ESF \( \overline{S}_k^{(n)}(X) = \overline{S}_k^{(n)} \)

function \( \overline{S}_k^{(n)} = \text{CompSumESF}(X, k) \)

\[
\begin{align*}
\hat{S}_0^{(i)} &= 1, \ 1 \leq i \leq n-1; \\
\hat{S}_j^{(i)} &= 0, \ j > i; \\
\hat{S}_1^{(1)} &= x_1; \\
\hat{\epsilon}_j^{(i)} &= 0, \ \forall \ i, j
\end{align*}
\]

for \( i = 2 : n \)

\[
\begin{align*}
\text{for } j &= \max\{1, i + k - n\} : \min\{i, k\} \\
[p, \beta_j^{(i)}] &= \text{TwoProd}(x_i, \hat{S}_{j-1}^{(i-1)}); \quad \% \ S_j^{(i)} = S_j^{(i-1)} + x_i S_{j-1}^{(i-1)} \\
[\hat{S}_j^{(i)}, \sigma_j^{(i)}] &= \text{TwoSum}(\hat{S}_j^{(i-1)}, p); \\
\hat{\epsilon}_j^{(i)} &= \hat{\epsilon}_j^{(i-1)} \oplus (\beta_j^{(i)} \oplus \sigma_j^{(i)}) \oplus x_i \otimes \hat{\epsilon}_{j-1}^{(i-1)} \\
\end{align*}
\]

done

end

\[
\overline{S}_k^{(n)} = \hat{S}_k^{(n)} \oplus \hat{\epsilon}_k^{(n)}
\]
Error bound on the Compensated Summation Algorithm

**Theorem 3**

For a vector of $n$ floating-point numbers $X = (x_1, \ldots, x_n)$, the relative forward error bound in Algorithm satisfies

\[
\left| \frac{\tilde{S}_k^{(n)} - S_k^{(n)}}{S_k^{(n)}} \right| \leq u + \frac{1}{k} \gamma^2 \frac{1}{2(n-1)} \text{cond}(S_k^{(n)}(X)),
\]

\[
\left| \frac{\tilde{S}_1^{(n)} - S_1^{(n)}}{S_1^{(n)}} \right| \leq u + \gamma^2 \frac{1}{n-1} \text{cond}(S_1^{(n)}),
\]

\[
\left| \frac{\tilde{S}_n^{(n)} - S_n^{(n)}}{S_n^{(n)}} \right| \leq u + \frac{1}{n} \gamma n \gamma^2 n \text{cond}(S_n^{(n)}),
\]

with $2 \leq k \leq n-1$, $k = 1$, $k = n$, respectively.
Validated Running Error bound on the Compensated Summation Algorithm (1/2)

Algorithm 8

**Input:** $X = (x_1, \ldots, x_n)$ and $k$

**Output:** $k$-th ESF $\overline{S}_k^{(n)}(X) = \overline{S}_k^{(n)}$ and Running Error Bound $\mu$

function $[\overline{S}_k^{(n)}, \mu] = \text{CompSumESFwErr}(X, k)$

$\hat{S}_0^{(i)} = 1, 1 \leq i \leq n - 1$; 
$\hat{S}_j^{(i)} = 0, j > i$; 
$\hat{S}_1^{(1)} = x_1$; 
$\hat{\epsilon}_S_j^{(i)} = 0, \hat{E}_S_j^{(i)} = 0, \forall \ i, j$

for $i = 2 : n$

for $j = \max\{1, i + k - n\} : \min\{i, k\}$

$[p, \beta_j^{(i)}] = \text{TwoProd}(x_i, \hat{S}_{j-1}^{(i-1)});$ 
$[\hat{S}_j^{(i)}, \sigma_j^{(i)}] = \text{TwoSum}(\hat{S}_{j-1}^{(i-1)}, p);$ 
$\hat{\epsilon}_S_j^{(i)} = \hat{\epsilon}_S_j^{(i-1)} \oplus (\beta_j^{(i)} \oplus \sigma_j^{(i)}) \oplus x_i \otimes \hat{\epsilon}_S_{j-1}^{(i-1)}$ 
$\hat{E}_S_j^{(i)} = \hat{E}_S_j^{(i-1)} \oplus |\beta_j^{(i)} \oplus \sigma_j^{(i)}| \oplus |x_i| \otimes \hat{E}_S_{j-1}^{(i-1)}$

end

end

$[\overline{S}_k^{(n)}, c] = \text{FastTwoSum}(\hat{S}_k^{(n)}, \hat{\epsilon}_S_k^{(n)})$

$\hat{\alpha} = (\hat{\gamma}_{2(n-1)} \otimes \hat{E}_S_k^{(n)}) \otimes (1 - 3nu);$ 
$\mu = (|c| \oplus \hat{\alpha}) \otimes (1 - 2u)$
Theorem 4

Assume $3nu < 1$, then a running error bound of Algorithm 8 is given by

$$|\overline{S}_k^{(n)} - S_k^{(n)}| \leq \text{fl}\left(\frac{|c| \oplus \hat{\alpha}}{1 - 2u}\right) := \mu,$$

where $\hat{\alpha}$ is the “error bound” on the rounding errors and $c$ is obtained by $[\overline{S}_k^{(n)}, c] = \text{FastTwoSum}(\overline{S}_k^{(n)}, \hat{\epsilon} S_k^{(n)})$. 
Library double-double

A double-double number $a$ is the pair $(a_h, a_l)$ of IEEE-754 floating-point numbers with $a = a_h + a_l$ and $|a_l| \leq u |a_h|$.

Algorithm 9 (Product of a d-d $(a_h, a_l)$ by a d $b$)

function $[c_h, c_l] = \text{prod}_\text{dd}_\text{d}(a_h, a_l, b)$

$[s_h, s_l] = \text{TwoProduct}(a_h, b)$

$[t_h, t_l] = \text{FastTwoSum}(s_h, (a_l \odot b))$

$[c_h, c_l] = \text{FastTwoSum}(t_h, (t_l \oplus s_l))$

Algorithm 10 (Addition of a d $b$ and a d-d $(a_h, a_l)$)

function $[c_h, c_l] = \text{add}_\text{dd}_\text{d}(a_h, a_l, b)$

$[t_h, t_l] = \text{TwoSum}(a_h, b)$

$[c_h, c_l] = \text{FastTwoSum}(t_h, (t_l \oplus a_l))$
Accurate Summation Algorithm with double-double

**Algorithm 11**

**Input:** $X = (x_1, \ldots, x_n)$ and $k$

**Output:** $k$-th ESF $S_k^{(n)}(X) = S_k^{(n)} = Sh_k^{(n)}$

function $[Sh_k^{(n)}, Sl_k^{(n)}] = \text{DDSumESF}(X, k)$

- $Sh_0^{(i)} = 1$, $1 \leq i \leq n-1$;  
- $Sh_j^{(i)} = 0$, $j > i$;  
- $Sh_1^{(1)} = x_1$;
- $Sl_i^{(i)} = 0$, $\forall \ i, j$

for $i = 2 : n$

- for $j = \max\{1, i + k - n\} : \min\{i, k\}$
  
- $[rh, rl] = \text{prod}_{dd\_d}(Sh_{j-1}^{(i-1)}, Sl_{j-1}^{(i-1)}, x_i)$;
  
- $[Sh_j^{(i)}, Sl_j^{(i)}] = \text{add}_{dd\_dd}(rh, rl, Sh_j^{(i-1)}, Sl_j^{(i-1)})$

end

end
For a standard model of floating-point arithmetic for the double-double algorithms

$$fl(a \odot b) = (a \odot b)(1 + \delta),$$

where $a, b$ are in double-double format, $\odot \in \{+, -, \times, /\}$, and $\delta$ is bounded as follows

$$|\delta| \leq u_{dd} \quad \text{for } \odot \in \{+, -\}; \quad |\delta| \leq 2u_{dd} \quad \text{for } \odot \in \{\times, /\}$$

where $u_{dd} = 2u^2 = 2^{-105}$ is the roundoff unit in double-double format.
Accuracy with double-double (2/2)

Theorem 5

The values $\hat{S}h_k^{(n)}$ and $\hat{S}l_k^{(n)}$ returned by Algorithm 11 in floating-point arithmetic satisfy

$$\frac{|\hat{S}h_k^{(n)} - S_k^{(n)}|}{|S_k^{(n)}|} \leq u + \frac{1}{k} (1 + u) \tilde{\gamma}_{3(n-1)} \text{cond}(S_k^{(n)}(X)),$$

where

$$\tilde{\gamma}_{3(n-1)} = \frac{3(n-1)u_{dd}}{1 - 3(n-1)u_{dd}} = \frac{6(n-1)u^2}{1 - 6(n-1)u^2}.$$
Numerical experiments (1/2)

The graph shows the relationship between the relative forward error and the condition number for different methods of evaluating ESF. The x-axis represents the condition number, and the y-axis represents the relative forward error. The methods compared include SumESF, CompSumESF, DDSumESF, LejaSumESF, RunErrBound, and TheoBoundDD.

Key observations:
- The error increases as the condition number increases.
- The methods show varying degrees of accuracy, with some methods having lower error than others.
- The graph also includes markers for different bounds and error measures, providing a comprehensive view of the accuracy of the evaluation methods.
### Time ratios of computing for $k$-th ESF (case 1) and for all ESF (case 2)

<table>
<thead>
<tr>
<th></th>
<th>CompSumESF</th>
<th>DDSumESF</th>
<th>CompSumESF</th>
<th>CompSumESF</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SumESF</td>
<td>SumESF</td>
<td>DDSumESF</td>
<td>DDSumESF</td>
</tr>
<tr>
<td>Case 1</td>
<td>3.05</td>
<td>5.42</td>
<td>57.42%</td>
<td>69.91%</td>
</tr>
<tr>
<td>Case 2</td>
<td>3.91</td>
<td>7.48</td>
<td>52.97%</td>
<td>68.02%</td>
</tr>
</tbody>
</table>
Conclusion and future work

Conclusion

- A fast algorithm to compute the Symmetric Elementary Functions as accurate as if computed with twice the working precision

Future work

- An algorithm making it possible to deal with complex numbers
- An algorithm to compute a faithfully rounded result and then a correctly rounded result
Daniela Calvetti and Lothar Reichel.  
On the evaluation of polynomial coefficients.  

Takeshi Ogita, Siegfried M. Rump, and Shin’ichi Oishi.  
Accurate sum and dot product.  

Rizwana Rehman and Ilse C. F. Ipsen.  
Computing characteristic polynomials from eigenvalues.  
Thank you for your attention