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

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

$$c_1 = \mathrm{trace}(A) \qquad\qquad c_n = \det(A)$$

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

$$c_1 = \sum_{i=1}^{n} \lambda_i \qquad\qquad c_n = \prod_{i=1}^{n} \lambda_i$$

$\rightarrow$ the c_i are elementary symmetric functions of the λ_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*

$$S_k^{(n)}(X) = \sum_{1 \le \pi_1 < \ldots < \pi_k \le n} x_{\pi_1} x_{\pi_2} \ldots x_{\pi_k} \quad \textit{with} \quad 1 \le k \le n$$

For $k = 0$, $S_0^{(n)} = 1$

The k-th function $S_k^{(n)}(X)$ 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

$$\prod_{i=1}^{n}(x-x_i)=\sum_{i=0}^{n}c_i x^i=\sum_{i=0}^{n}(-1)^{n-i}S_{n-i}^{(n)}(x_1,\dots,x_n)x^i$$

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)

$$\mathtt{cond}(S_k^{(n)}(X)) = \lim_{\varepsilon\to 0}\sup\left\{\frac{|S_k^{(n)}(X+\triangle X) - S_k^{(n)}(X)|}{\varepsilon |S_k^{(n)}(X)|} : |\triangle X| < \varepsilon |X|\right\}$$

A direct calculation yields

$$\mathtt{cond}(S_k^{(n)}(X)) = \frac{kS_k^{(n)}(|X|)}{|S_k^{(n)}(X)|}$$

In particular, $\mathtt{cond}(S_n^{(n)}(X)) = \mathtt{cond}(\prod_{i=1}^n x_i) = n$ and $\mathtt{cond}(S_1^{(n)}(X)) = \mathtt{cond}(\sum_{i=1}^n x_i) = \frac{\sum_{i=1}^n |x_i|}{|\sum_{i=1}^n x_i|}$.

Classic Summation Algorithm

Algorithm 1

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

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

`function` $S_k^{(n)}$=`SumESF`(X, k)

  $S_0^{(i)} = 1,\ 1 \le i \le n-1;$ $S_j^{(i)} = 0,\ j > i;$ $S_1^{(1)} = x_1;$

  `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`

$$S_j^{(i)} = S_j^{(i)}(x_1, \dots, x_i) = \sum_{1 \le \pi_1 < \dots < \pi_j \le i} x_{\pi_1} x_{\pi_2} \dots x_{\pi_j}$$

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 $\mathbf{u}$ (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

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

IEEE 754 standard (2008)

Type	Size	Mantissa	Exponent	Unit rounding	Interval
binary32	32 bits	23+1 bits	8 bits	$\mathbf{u} = 2^{1-24} \approx 1,92 \times 10^{-7}$	$\approx 10^{\pm 38}$
binary64	64 bits	52+1 bits	11 bits	$\mathbf{u} = 2^{1-53} \approx 2,22 \times 10^{-16}$	$\approx 10^{\pm 308}$

We denote

$$\gamma_n := \frac{n\mathbf{u}}{1 - n\mathbf{u}}$$

Rounding error analysis

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 $\widehat{S}_k^{(n)} = \widehat{S}_k^{(n)}(X)$ by Algorithm 1 in floating-point arithmetic verifies

$$\left|\frac{\widehat{S}_k^{(n)} - S_k^{(n)}}{S_k^{(n)}}\right| \leq \frac{1}{k}\gamma_{2(n-1)}\mathtt{cond}(S_k^{(n)}),\ 2 \leq k \leq n-1,$$

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

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

Getting more accuracy with compensated algorithms

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

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

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

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

Algorithms of Dekker (1971) and Knuth (1974)

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

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

$x = a \oplus b$

$y = (a \ominus x) \oplus b$

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

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

$x = a \oplus b$

$z = x \ominus a$

$y = (a \ominus (x \ominus z)) \oplus (b \ominus z)$

EFT for the product (1/3)

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

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)

function $[x, y] = \mathtt{Split}(a)$
  factor $= 2^s + 1$ % $\mathbf{u} = 2^{-p}$, $s = \lceil p/2 \rceil$
  $c = \mathtt{factor} \otimes a$
  $x = c \ominus (c \ominus a)$
  $y = a \ominus x$

EFT for the product (2/3)

Algorithm 5 (EFT of the product of 2 floating point numbers)

function $[x, y] = \mathtt{TwoProduct}(a, b)$

$x = a \otimes b$

$[a_1, a_2] = \mathtt{Split}(a)$

$[b_1, b_2] = \mathtt{Split}(b)$

$y = a_2 \otimes b_2 \ominus (((x \ominus a_1 \otimes b_1) \ominus a_2 \otimes b_1) \ominus a_1 \otimes b_2)$

Theorem 2

Let $a, b \in \mathbb{F}$ and let $x, y \in \mathbb{F}$ such that $[x, y] = \mathtt{TwoProduct}(a, b)$. Then,

$$a \cdot b = x + y, \quad x = \mathrm{fl}(a \cdot b), \quad |y| \leq \mathbf{u}|x|, \quad |y| \leq \mathbf{u}|a \cdot b|,$$

The algorithm `TwoProduct` *requires* 17 *flops.*

EFT for the product (3/3)

Given $a, b, c \in \mathbb{F}$,

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

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

function $[x, y] = \texttt{TwoProductFMA}(a, b)$

$x = a \otimes b$

$y = \texttt{FMA}(a, b, -x)$

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 $\overline{S}_k^{(n)}$=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 $j = \max\{1, i+k-n\} : \min\{i, k\}$

$[p, \beta_j^{(i)}] =$ TwoProd$(x_i, \widehat{S}_{j-1}^{(i-1)})$; % $S_j^{(i)} = S_j^{(i-1)} + x_i S_{j-1}^{(i-1)}$

$[\widehat{S}_j^{(i)}, \sigma_j^{(i)}] =$ 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

$\overline{S}_k^{(n)} = \widehat{S}_k^{(n)} \oplus \widehat{\epsilon S}_k^{(n)}$

Error bound on the Compensated Summation Algorithm

Theorem 3

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

$$\left|\frac{\overline{S}_k^{(n)} - S_k^{(n)}}{S_k^{(n)}}\right| \le \mathbf{u} + \frac{1}{k}\gamma_{2(n-1)}^2 \,\mathtt{cond}(S_k^{(n)}(X)),$$

$$\left|\frac{\widehat{S}_1^{(n)} - S_1^{(n)}}{S_1^{(n)}}\right| \le \mathbf{u} + \gamma_{n-1}^2 \,\mathtt{cond}(S_1^{(n)}),$$

$$\left|\frac{\widehat{S}_n^{(n)} - S_n^{(n)}}{S_n^{(n)}}\right| \le \mathbf{u} + \frac{1}{n}\gamma_n \gamma_{2n} \mathtt{cond}(S_n^{(n)}),$$

with $2 \le k \le n-1$, $k = 1$, $k = n$, 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 μ

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

$\widehat{S}_0^{(i)} = 1,\ 1 \le i \le n-1;\qquad \widehat{S}_j^{(i)} = 0,\ j > i;\qquad \widehat{S}_1^{(1)} = x_1;\qquad \widehat{\epsilon S}_j^{(i)} = 0, \widehat{ES}_j^{(i)} = 0, \forall\ i, j$

for $i = 2 : n$

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

$[p, \beta_j^{(i)}] = \mathtt{TwoProd}(x_i, \widehat{S}_{j-1}^{(i-1)});\qquad [\widehat{S}_j^{(i)}, \sigma_j^{(i)}] = \mathtt{TwoSum}(\widehat{S}_j^{(i-1)}, p);$

$\widehat{\epsilon S}_j^{(i)} = \widehat{\epsilon S}_j^{(i-1)} \oplus (\beta_j^{(i)} \oplus \sigma_j^{(i)}) \oplus x_i \otimes \widehat{\epsilon S}_{j-1}^{(i-1)}$

$\widehat{ES}_j^{(i)} = \widehat{ES}_j^{(i-1)} \oplus |\beta_j^{(i)} \oplus \sigma_j^{(i)}| \oplus |x_i| \otimes \widehat{ES}_{j-1}^{(i-1)}$

end

end

$[\overline{S}_k^{(n)}, c] = \mathtt{FastTwoSum}(\widehat{S}_k^{(n)}, \widehat{\epsilon S}_k^{(n)})$

$\hat{\alpha} = (\widehat{\gamma}_{2(n-1)} \otimes \widehat{ES}_k^{(n)}) \oslash (1 - 3nu);\qquad \mu = (|c| \oplus \hat{\alpha}) \oslash (1 - 2u)$

Validated Running Error bound on the Compensated Summation Algorithm (2/2)

Theorem 4

Assume $3n\mathbf{u} < 1$, *then a running error bound of Algorithm 8 is given by*

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

where $\widehat{\alpha}$ *is the "error bound" on the rounding errors and c is obtained by* $[\overline{S}_k^{(n)}, c] = \texttt{FastTwoSum}(\widehat{S}_k^{(n)}, \widehat{\epsilon S}_k^{(n)})$.

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 $|a_l| \leq \mathbf{u}|a_h|$.

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

function $[c_h, c_l]$ = prod_dd_d(a_h, a_l, b)
  $[s_h, s_l]$ = TwoProduct(a_h, b)
  $[t_h, t_l]$ = FastTwoSum$(s_h, (a_l \otimes b))$
  $[c_h, c_l]$ = FastTwoSum$(t_h, (t_l \oplus s_l))$

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

function $[c_h, c_l]$ = add_dd_d(a_h, a_l, b)
  $[t_h, t_l]$ = TwoSum(a_h, b)
  $[c_h, c_l]$ = FastTwoSum$(t_h, (t_l \oplus a_l))$

Accurate Summation Algorithm with `double-double`

Algorithm 11

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

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

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

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

$Sl_j^{(i)} = 0, \forall\ i, j$

```
for i = 2 : n
   for j = max{1, i + k − n} : min{i, k}
      [rh, rl] = prod_dd_d(Sh_{j-1}^{(i-1)}, Sl_{j-1}^{(i-1)}, x_i);
      [Sh_j^{(i)}, Sl_j^{(i)}] = add_dd_dd(rh, rl, Sh_j^{(i-1)}, Sl_j^{(i-1)})
   end
end
```

Accuracy with `double-double` (1/2)

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

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

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

$$|\delta| \leq \mathbf{u}_{dd} \quad \mathit{for} \odot \in \{+, -\}; \qquad |\delta| \leq 2\mathbf{u}_{dd} \quad \mathit{for} \odot \in \{\times, /\}$$

where $\mathbf{u}_{dd} = 2\mathbf{u}^2 = 2^{-105}$ is the roundoff unit in double-double format.

Accuracy with `double-double` (2/2)

Theorem 5

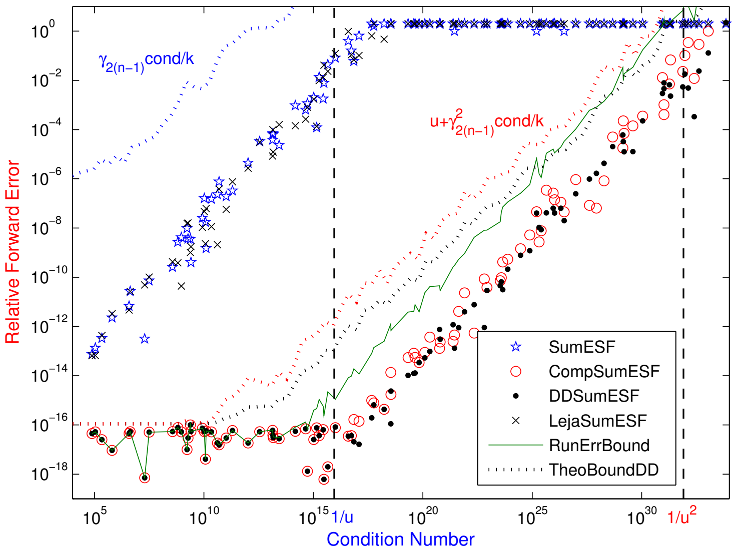
The values $\widehat{Sh}_k^{(n)}$ and $\widehat{Sl}_k^{(n)}$ returned by Algorithm 11 in floating-point arithmetic satisfy

$$\frac{|\widehat{Sh}_k^{(n)} - S_k^{(n)}|}{|S_k^{(n)}|} \leq \mathbf{u} + \frac{1}{k}(1+\mathbf{u})\overline{\gamma}_{3(n-1)}\,\mathtt{cond}(S_k^{(n)}(X)),$$

where

$$\overline{\gamma}_{3(n-1)} = \frac{3(n-1)\mathbf{u}_{dd}}{1-3(n-1)\mathbf{u}_{dd}} = \frac{6(n-1)\mathbf{u}^2}{1-6(n-1)\mathbf{u}^2}.$$

Numerical experiments (1/2)



Numerical experiments (2/2)

Time ratios of computing for k-th ESF (case 1) and for all ESF (case 2)

	$\frac{\texttt{CompSumESF}}{\texttt{SumESF}}$	$\frac{\texttt{DDSumESF}}{\texttt{SumESF}}$	$\frac{\texttt{CompSumESF}}{\texttt{DDSumESF}}$	$\frac{\texttt{CompSumESF}}{\texttt{CompSumESFwErr}}$
Case 1	3.05	5.42	57.42%	69.91%
Case 2	3.91	7.48	52.97%	68.02%

Conclusion and future work

Conclusion

- A fast algorithm to computed 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

References

Daniela Calvetti and Lothar Reichel.
On the evaluation of polynomial coefficients.
Numer. Algorithms, 33(1-4):153–161, 2003.

Takeshi Ogita, Siegfried M. Rump, and Shin'ichi Oishi.
Accurate sum and dot product.
SIAM J. Sci. Comput., 26(6):1955–1988, 2005.

Rizwana Rehman and Ilse C. F. Ipsen.
Computing characteristic polynomials from eigenvalues.
SIAM J. Matrix Anal. Appl., 32(1):90–114, 2011.

Thank you for your attention