UPDATING FORMULAE AND A PAIRWISE ALGORITHM FOR COMPUTING SAMPLE VARIANCES

by

Tony F. Chan
Gene H. Golub
Randall J. LeVeque

STAN-CS-79-773
November 1979

DEPARTMENT OF COMPUTER SCIENCE
School of Humanities and Sciences
STANFORD UNIVERSITY

Updating Formulae and a Pairwise Algorithm for Computing Sample Variances

Tony F. Chan*
Gene H. Golub**
Randall J. LeVeque**

Abstract. A general formula is presented for computing the sample variance for a sample of size $m+n$ given the means and variances for two subsamples of sizes m and n. This formula is used in the construction of a pairwise algorithm for computing the variance. Other applications are discussed as well, including the use of updating formulae in a parallel computing enviornment. We present numerical results and rounding error analyses for several numerical schemes.

*Department of Computer Science, Yale University, New Haven, CT 06520
**Department of Computer Science, Stanford University, Stanford, CA 94305.

This work was supported in part by Army contract No. DAAG29-78-G-0179 and by a National Science Foundation graduate fellowship. The paper was produced using TEX, a computer typesetting system created by Donald Knuth at Stanford.

1. Introduction.

In computing the variance of a sample of N data points $\{x_i\}$, the fundamental calculation consists of computing the sum of squares of deviations from the mean. This quantity, which for brevity will be referred to as "the sum of squares" or simply as "S", is defined as

$$S = \sum_{i=1}^{N}(x_i - \bar{x})^2, \tag{1.1a}$$

where

$$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i. \tag{1.1b}$$

This computation can be easily performed directly by the *two-pass algorithm* (1.1) provided that (a) N is small compared to the amount of core memory available and (b) the variance is not too small relative to the norm of the data, $\|x\|_2 = (\sum_{i=1}^{N} x_i^2)^{1/2}$. If either of these conditions is violated, however, the situation changes. If N is large, this approach to computing S may be unsatisfactory since it requires passing through the data twice: once to compute $\bar{x}$ and then again to compute S. This problem is sometimes avoided by use of the following *textbook algorithm*, so called because, unfortunately, it is often suggested in statistical textbooks:

$$S = \sum_{i=1}^{N} x_i^2 - \frac{1}{N}\left(\sum_{i=1}^{N} x_i\right)^2. \tag{2}$$

This rearrangement allows S to be computed with only one pass through the data, but the computation may be numerically unstable and should almost never be used in practice. This instability is particularly troublesome when S is very small compared to $\|x\|_2$, in which case even the two-pass algorithm can be unstable.

In discussing the stability of a numerical scheme for computing S, a useful concept is that of the *condition number* κ of the data. This quantity was first introduced by Chan and Lewis[2] to give a thorough discussion. Briefly, κ is a measure of the sensitivity of S to changes in the data. The quantity κu is an upper bound for the relative perturbation which would occur in the exactly computed sum of squares if the input data contained relative errors of size u. If the true sum of squares is S, then κ is given by

$$\kappa = \frac{\|x\|_2}{\sqrt{S}}. \tag{1.3}$$

It is easy to see that $\kappa \geq 1$ and that in general κ grows as the variance decreases.

An error analysis of the textbook algorithm[2] shows that the relative error in S can be bounded by something on the order of

$$3N\kappa^2 u,$$

where u is the machine roundoff unit (see section 6). This algorithm is therefore seldom useful, as confirmed by the experimental results of Table 1.

The error analysis of the two-pass algorithm found in section 6 shows that the relative error in the sum of squares computed using that algorithm can be bounded by

$$Nu + N^2\kappa^2 u^2.$$

The second term in this bound has traditionally been ignored in error analyses of the two-pass algorithm as being of second order. But in the case we are interested in here, when N and κ are both large, this term can easily dominate. Table 2 shows this happening in practice.

During the preparation of this manuscript, a simple modification of the two-pass algorithm was found by Professor Åke Björck which reduces this bound. Based on the error analysis of section 6 for the standard two-pass algorithm, he suggested computing S by

$$S = \sum_{i=1}^{N}(x_i - \bar{x})^2 - \frac{1}{N}\left(\sum_{i=1}^{N}(x_i - \bar{x})\right)^2. \tag{1.4}$$

In exact arithmetic the second term is zero, but computationally it is a good approximation to the error in the two-pass algorithm. Note that (1.4) can also be viewed as the textbook algorithm applied to the data $\{(x_i - \bar{x})\}$. The error analysis of section 6 shows that the relative error in S computed by (1.4) can be bounded by

$$Nu + 4N^2\kappa u^2.$$

This modification adds only N additions and 2 multiplications to the cost of the two-pass algorithm (already 3N $-$ 2 additions and N $+$ 1 multiplications) and can be very useful when the data is poorly conditioned. See table 3 for some numerical results.

Of course formula (1.4) is still a two-pass algorithm. For large N it may be desirable to compute S with only one pass through the data. A number of papers have appeared recently on "updating" algorithms for computing S. These are algorithms which are based on formulae for adding one new data point to a sample and computing the value of S for the combined sample by updating the (presumably known) value of S for the original sample. By starting with a sample of size 1 and applying this formula repeatedly, we get a one-pass algorithm for computing S for a sample of arbitrary size. Youngs and Cramer[6] have investigated several such algorithms and have found the following algorithm to be the best:

```
$S := 0$
$T := x_1$
for j := $2, 3, \ldots, N$ do
    $T := T + x_j$
    $S := S + \frac{1}{j(j-1)}(jx_j - T)^2.$
```
(1.5)

This is based on the updating formula

$$S_{1,j} = S_{1,j-1} + \frac{1}{j(j-1)}(jx_j - T_{1,j})^2 \tag{1.6}$$

where $S_{i,j}$ stands for the sum of squares for the data points x_i through x_j and $T_{i,j}$ is the sum of x_i through x_j. This notation will be used throughout.

One imporant characteristic of this updating formula is that $S_{1,j}$ is formed from $S_{1,j-1}$ by adding to it a nonnegative quantitiy. In the textbook algorithm **(1.2)**, on the other hand, S is formed by a subtraction which can lead to gross cancellation and even to negative values of S being computed.

In practice the method (1.5) generally performs on a level comparable to the two-pass algorithm (1.1). Chan and Lewis[2] present detailed error analyses of some similar updating methods.

In the next section we present a generalization of the updating formula (1.6) for combining two samples of arbitrary size. Then in section 3 we describe a **pairwise** algorithm for computing S which is essentially still a one-pass algorithm but which numerically is often more stable than the standard two-pass algorithm.

2. A General Updating Formula.

The method of Youngs and Cramer depends on an updating formula which allows one to compute S for j+ 1 points when given the value of S for j points and one new point. In other words, we can combine a sample of size j with a sample of size 1 and determine the value of S for the combined sample.

This formula can be easily generalized to allow us to combine two samples of arbitrary size. Suppose we have two samples $\{x_i\}_{i=1}^{m}, \{x_i\}_{i=m+1}^{m+n}$ and we know

$$T_{1,m} = \sum_{i=1}^{m} x_i, \qquad T_{m+1,m+n} = \sum_{i=m+1}^{m+n} x_i,$$

$$S_{1,m} = \sum_{i=1}^{m}(x_i - \frac{1}{m}T_{1,m})^2, \qquad S_{m+1,m+n} = \sum_{i=m+1}^{m+n} (x_i - \frac{1}{n}T_{m+1,m+n})^2.$$

Then, if we combine all of the data into a sample of size $m+n$, it can be shown that

$$T_{1,m+n} = T_{1,m} + T_{1,m+n} \tag{2.1a}$$

$$S_{1,m+n} = S_{1,m} + S_{m+1,m+n} + \frac{m}{n(m+n)}\left(\frac{n}{m}T_{1,m} - T_{m+1,m+n}\right)^2. \tag{2.1b}$$

If we rewrite the latter formula as

$$S_{1,m+n} = S_{1,m} + S_{m+1,m+n} + \frac{m}{n(m+n)}\left(\frac{m+n}{m}T_{1,m} - T_{1,n+m}\right)^2,$$

then we see that for $m = 1$, $n = j - 1$, this reduces to the formula of Youngs and Cramer, since S = 0 for any single data point. The form (2.1b) is more stable numerically, however.

Regardless of what method is used to compute S, the formulae (2.1) may be useful in their own right whenever two samples must be combined. One possible application is to parallel processing. If one has two or more processors available, the sample can be split up into smaller subsamples, and the sum of squares computed -for each subsample independently using any algorithm desired. The sum of squares for the original sample can then be calculated using the updating formulae.

However, even in the case of a single yroccssor, it is very desirable to compute S using (2.1). The method (1.5) may be generalized to compute S by processing the-data in groups of m elements: cornpute the sum of squares for each group using the two-pass algorithm and then update the global S accordingly. Traditional updating algorithms such as that of Youngs and Cramer have used $m = 1$.

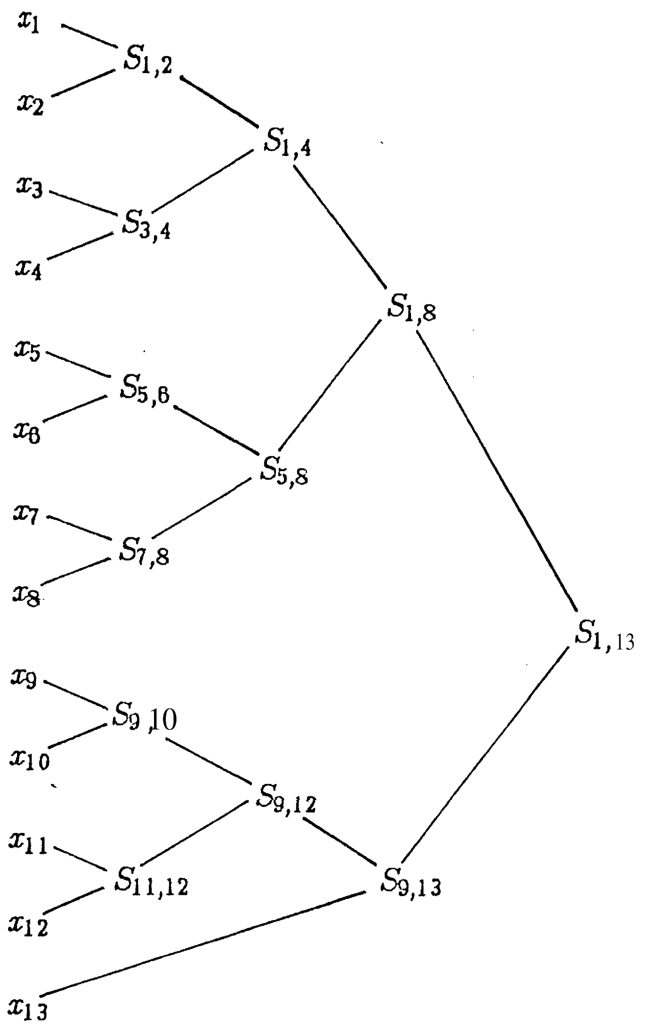
We have found, however, that the stability of the algorithm is increased by taking $m > 1$. One can easily see that the total number of arithmetic operations performed on the data is minimized by taking $m = \sqrt{N}$. We might expect that this choice of m will minimize the resulting error. Although we do not have a satisfactory error analysis of this algorithm, the experimental results of table 4 do tend to confirm this prediction. Strictly speaking, with $m > 1$ this is no longer a one-pass algorithm, but we see that only m data values at a time need to be kept in core, and m can be as small as necessary.

3. The Pairwise Algorithm.

Table 4 shows that choosing $m > 1$ not only gives more accuracy than using $m = 1$, but can actually give significantly more accuracy than the two-pass algorithm. This suggests that when computing the sum of squares for the subsamples of size m, we should not use the two-pass algorithm when S is small. Rather, we should split the subsample into yet smaller groups. Taking this idea to the limit yields a pairwise algorithm analogous to the well-known pairwise algorithm for computing the sum of N numbers. Let $S_{i,j}$ stand for the sum of squares of elements x_i through x_j, and let $m = \lfloor N/2 \rfloor$, the largest integer not exceeding N/2. Then the method consists of computing $S_{1,N}$ by first computing $S_{1,m}$ and $S_{m+1,N}$ and then combining these by means of (2.1). Each of these latter quantities has been computed by a-similar combination of still smaller subsamples.

The algorithm can be implemented as just described, but for reasons which we will explain shortly it is actually best to perform the pairwise algorithm in a somewhat modified manner. Consider the following example with $N = 13$. Schematically, we compute from left to right in the tableau (3.1). The intermediate $T_{i,j}$ are also computed in a similar tableau for use in updating the $S_{i,j}$. The final value $T_{1,N}$ will be the sum of all the data points as computed by the pairwise summation algorithm. In practice we can compute from top to bottom in these tableaux requiring only one pass through the data and using only $O(\log_2 N)$ storage locations for intermediate results. We require one such location for each column in each tableau. The computation for the tableau (3.1) would proceed as follows:

(a) Compute $S_{1,2}$ and store in S[1].
(b) Compute $S_{3,4}$, combine with S[1] to get $S_{1,4}$, and store this in S[2].
(c) Compute $S_{5,6}$ and store in S[1].
(d) Compute $S_{7,8}$, combine with S[1] to get $S_{5,8}$ and then combine this with S[2] to get $S_{1,8}$, which is then stored in S[3].
(e) Compute $S_{9,10}$ and store in S[1].
(f) Compute $S_{11,12}$, combine with S[1] to get $S_{9,12}$, which is then stored in S[2].
(g) Clean-up (necessary when N is not a power of 2):
Combine x_{13} with S[2] to get $S_{9,13}$. Combine this with S[3] to get $S_{1,13}$.



(3.1)

Alternatively, we can use a stack structure for the temporary locations as is done in the sample FORTRAN routine given in section 9.

The final step (g) of our algorithm requires the combination of samples of quite disparate sizes. Such a calculation would be avoided if we adopted the algorithm as described in the first paragraph of this section. For the pairwise summation algorithm, Tsao[4] points out that the corresponding method gives a decreased "average error complexi ty" and presents an implementation based on the binary expansion of N. That strategy could be adopted in the present context as well, but its usefulness here is questionable. We feel that the small increase in accuracy which might result would be more than offset by the increased work which we would thus incur. For the updating formula (2.1b), it is desirable to have $n = m$ whenever possible, since that formula then becomes simply

$$S_{1,2m} = S_{1,m} + S_{m+1,2m} + \frac{1}{2m}(T_{1,m} - T_{m+1,2m})^2.$$

In this respect, the tableau (3.1) gives the preferable computational scheme. In fact, the amount of work required to perform the pairwise algorithm as described here is not significantly more than that required for the two-pass algorithm. An operation count shows that roughly $2N$ additions and $5N/2$ multiplications are required, as opposed to 3N additions and N multiplications for the two-pass algorithm. In addition, some bookkeeping operations are required to manage the pairwise algorithm.

Although we are not able to provide any error bounds proving the superiority of the pairwise algorithm, our experimental results have been quite satisfactory. Some of these results are shown in table 5 of section 8.

5. Extensions.

Often one wants to compute a weighted sum of squares of deviations from the mean,

$$S_{1,N}^{(W)} = \sum_{i=1}^{N} w_i(x_i - \bar{x})^2. \tag{5.1}$$

The updating formulae (2.1) still hold with only a few minor modifications. Let $W_{j,k} = \sum_{i=j}^{k} w_i$. Then (2.1) is replaced by

$$\begin{aligned} T_{1,m+n} &= T_{1,m} + T_{m+1,m+n} \\ W_{1,m+n} &= W_{1,m} + W_{m+1,m+n} \\ S_{1,m+n}^{(W)} &= S_{1,m}^{(W)} + \frac{W_{1,m}}{W_{m+1,m+n}(W_{1,m} + W_{m+1,m+n})} \\ &\quad \times \left(\frac{W_{m+1,m+n}}{W_{1,m}} T_{1,m} - T_{m+1,m+n} \right)^2. \end{aligned} \tag{5.2}$$

Another quantity which is often of interest is the covariance of two samples $\{x_i\}$ and $\{y_i\}$. For this it is necessary to compute

$$C_{1,N} = \sum_{i=1}^{N} (x_i - \bar{x})(y_i - \bar{y}).$$

If we let $T^{(x)}_{j,k} = \sum_{i=j}^{k} x_i$, $T^{(y)}_{j,l} = \sum_{i=j}^{k} y_i$, then the updating formula for C is

$$C_{1,m+n} = C_{1,m} + C_{m+1,m+n} + \frac{m}{n(m+n)} \times \left(\frac{n}{m} T^{(x)}_{1,m} - T^{(x)}_{m+1,m+n}\right)\left(\frac{n}{m} T^{(y)}_{1,m} - T^{(y)}_{m+1,m+n}\right). \tag{5.3}$$

6. Error analysis *of* the two-pass olgorithms.

We assume throughout our error analyses that we are dealing with a machine with a guard digit and relative precision u. On a base β machine with a. t digit mantissa and proper unbiased rounding, u = $\frac{1}{2}\beta^{1-t}$.

Roman letters with tildas over them will be used to denote quantities actually computed numerically. The same letter without a tilda will indicate the corresponding exact--quantity.

In this section we present error analyses for both the standard two-pass algorithm (1.1) and Björck's modification (1.4). Let

$$S_1 = \sum_{i=1}^{N} (x_i - \bar{x})^2,$$

$$S_2 = \frac{1}{N}\left(\sum_{i=1}^{N} (x_i - \bar{x})\right)^2,$$

$$S = S_1 - S_2.$$

The standard two-pass algorithm is S_1. We first compute a value $\tilde{\bar{x}}$ for the mean of $\{x_i\}$. If this is computed in the standard manner we have

$$\tilde{\bar{x}} = \frac{1}{N}\sum_{i=1}^{N} x_i(1+\xi_i), \qquad \text{with } |\xi_i| < Nu + O(u^2), \tag{6.1}$$
$$= \bar{x} + \frac{1}{N}\sum x_i \xi_i.$$

The computed value $\tilde{S}_1$ is then given by

$$
\begin{aligned}
\tilde{S}_1 &= \sum (x_i - \tilde{\bar{x}})^2(1+\eta_i), \qquad |\eta_i| < (N+2)u + O(u^2) \\
&= \sum \Big((x_i - \bar{x}) + (\bar{x} - \tilde{\bar{x}})\Big)^2 (1+\eta_i) \\
&= \sum \Big((x_i - \bar{x})^2 + 2(x_i - \bar{x})(\bar{x} - \tilde{\bar{x}}) + (\bar{x} - \tilde{\bar{x}})^2\Big)(1+\eta_i) \\
&= S + \sum (x_i - \bar{x})^2 \eta_i - \frac{2}{N}\Big(\sum x_i\xi_i\Big) \sum (x_i - \bar{x})(1+\eta_i) \\
&\quad + \Big(\frac{1}{N}\sum x_i\xi_i\Big)^2 \Big(N + \sum \eta_i\Big).
\end{aligned} \tag{6.2}
$$

The $O(u^2)$ terms in the bounds for $|\eta_i|$ and $|\xi_i|$ turn out to be unimportant in the present error analysis and will be dropped below. Note that $\sum(x_i - \bar{x}) = 0$ and that the following inequalities hold:

$$
\begin{aligned}
&\Big|\sum (x_i - \bar{x})^2 \eta_i\Big| \le S\|\eta\|_\infty < S(N+2)u, \\
&\Big|\sum x_i\xi_i\Big| \le \|x\|_2\|\xi\|_2 \le N^{1/2}\|x\|_2\|\xi\|_\infty < N^{3/2}\|x\|_2 u, \\
&\Big|\sum (x_i - \bar{x})\eta_i\Big| \le S^{1/2}\|\eta\|_2 \le S^{1/2}N^{1/2}\|\eta\|_\infty < S^{1/2}N^{1/2}(N+2)u.
\end{aligned}
$$

So from (6.2) we obtain the bound

$$
\|\tilde{S}_1 - S\| \le S(N+2)u + 2N(N+2)S^{1/2}\|x\|_2 u^2 + N^2\|x\|_2^2 u^2(1 + (N+2)u).
$$

Recalling the definition (1.3) of κ, we see that

$$
\begin{aligned}
\left|\frac{\tilde{S}_1 - S}{S}\right| &< (N+2)u + 2N(N+2)\kappa u^2 + N^2\kappa^2u^2(1 + (N+2)u) \\
&\sim Nu + N^2\kappa^2u^2 + 2N^2\kappa u^2.
\end{aligned} \tag{6.3}
$$

When $N >> 1$, $\kappa >> 1$, the term $N^2\kappa^2u^2$ may cause problems as was seen in table 2. Note that this term results from the term $N(\frac{1}{N}\sum x_i\xi_i)^2$ in (6.2). We will now show that the computed value $\tilde{S}_2$ is a good approximation to this error. We have that

$$\begin{aligned}
\tilde{S}_2 &= \frac{1}{N}\Big(\sum(x_i-\tilde{\bar{x}})(1+\gamma_i)\Big)^2 \qquad \text{with } |\gamma_i| < (N+2)u + O(u^2). \\
&= \frac{1}{N}\Big(\sum((x_i-\bar{x})+(\bar{x}-\tilde{\bar{x}}))(1+\gamma_i)\Big)^2 \\
&= \frac{1}{N}\Big(\sum(x_i-\bar{x})\gamma_i - \frac{1}{N}\Big(\sum x_i\xi_i\Big)\Big(N+\sum\gamma_i\Big)\Big)^2 \\
&= \frac{1}{N}\Big(\sum(x_i-\bar{x})\gamma_i\Big)^2 - \frac{2}{N^2}\Big(\sum(x_i-\bar{x})\gamma_i\Big)\Big(\sum x_i\xi_i\Big)\Big(N+\sum\gamma_i\Big) \\
&\quad + \frac{1}{N^3}\Big(\sum x_i\xi_i\Big)^2\Big(N^2+2N\sum\gamma_i+\Big(\sum\gamma_i\Big)^2\Big)
\end{aligned}$$

Note that $\tilde{S}_2$ contains a term $\frac{1}{N}(\sum x_i\xi_i)^2$, so, using Björck's modification of the two-pass algorithm we compute $\tilde{S}$ as

$$\begin{aligned}
\tilde{S} &= (\tilde{S}_1 - \tilde{S}_2)(1+\delta), \qquad \text{with } |\delta| < u \\
&= \Big(S + \sum(x_i-\bar{x})^2\eta_i - \frac{2}{N}\Big(\sum x_i\xi_i\Big)\sum(x_i-\bar{x})(1+\eta_i) \\
&\quad + \frac{1}{N^2}\Big(\sum x_i\xi_i\Big)^2\sum\eta_i - \frac{1}{N}\Big(\sum(x_i-\bar{x})\gamma_i\Big)^2 \\
&\quad - \frac{2}{N^2}\Big(\sum(x_i-\bar{x})\gamma_i\Big)\Big(\sum x_i\xi_i\Big)\Big(N+\sum\gamma_i\Big) \\
&\quad - \frac{1}{N^3}\Big(\sum x_i\xi_i\Big)^2\Big(2N\sum\gamma_i+\Big(\sum\gamma_i\Big)^2\Big)\Big)(1+\delta).
\end{aligned}$$

Bounding these quantities as before gives the following bound for the relative error:

$$\begin{aligned}
\left|\frac{\tilde{S}-S}{S}\right| &< (N+2)u + (N+2)(4N\kappa + 2(N+2))u^2 \\
&\quad + (N+2)(3N^2\kappa^2 + (6N+4)\kappa + (N+2))u^3 + O(u^4) \\
&\sim Nu + 4N^2\kappa u^2 + 3N^3\kappa^2u^3.
\end{aligned}$$

The modification has thus reduced the "second order term" by roughly a factor of κ.

7. **Calculation of the mean** in double precision.

A greater accuracy can be achieved from any algorithm for computing the sum of squares by simply using higher precision arithmetic. It is important to note, however, that a large increase in accuracy can often be achieved by shifting only some of the calculations to double precision. From the error analysis of the two-pass algorithm, we see that computing the sample mean in double precision would replace the bound $|\xi_i| < Nu + O(u^2)$ in (6.1) by $|\xi_i| < Nu^2 + O(u^3)$. If the remainder of the calculations are still computed in single precision, the error bound (6.3) will nonetheless be replaced by the improved bound

$$\left|\frac{\tilde{S}_1 - S}{S}\right| < Nu + O(u^3).$$

The difference which this can make in practice is evident from table 6 in section 8, which gives the results of some numerical experiments.

The generalized updating algorithm and the pairwise algorithm are also improved by calculating the corresponding running sums in double precision. Numerical results for these modified algorithms are given in tables 7 and 8 respectively.

8. **Experimental** results

All of the results presented in this section were computed on an IBM 370/168 computer at the Stanford Linear Accelerator Center. The data used was provided by a random number generator with mean 1 and a variety of different variances σ^2. For this choice of the mean, $\kappa \approx 1/\sigma$. In each case the results have been averaged over 20 runs. Single precision was used in most of the tests except in the cases where the mean was computed in double precision (tables 6-8). In single precision, u $\approx 5 \times 10^{-7}$. The "correct" answer for use in computing the error was computed in quad precision. We report the number of correct digits in the calculation, defined as $-\log_{10}(E)$ where E is the relative error.

Table 1: Number of correct digits for the textbook algorithm on N data points chosen randomly from N(1.O, σ^2).

σ^2 \ N	64	256	1024	2048
1.0	5.4	4.3	4.1	4.1
10^{-1}	4.2	4.7	3.0	3.0
10^{-2}	3.2	3.2	2.0	2.0
10^{-3}	2.2	2.2	1.0	1.0
10^{-4}	1.2	1.1	0.0	0.0
10^{-5}	0.2	0.2	-1.0	-1.0
10^{-6}	—0.8	-0.8	-2.0	-2.0
10^{-7}	-1.8	-1.9	-3.0	-3.0
10^{-8}	-2.8	-2.8	-4.0	-4.0

Table 2: Number of correct digits for the two-pass algorithm on N data points chosen randomly from N(1.O, σ^2).

σ^2 \ N	64	256	1024	2048
1.0	5.2	5.1	4.0	4.0
10^{-1}	5.4	4.5	4.2	4.2
10^{-2}	5.6	4.5	4.4	3.7
10^{-3}	5.6	4.6	4.5	3.6
10^{-4}	5.2	4.8	4.4	4.0
10^{-5}	5.5	5.3	3.1	3.0
10^{-6}	4.5	4.4	2.1	1.9
10^{-7}	3.5	3.3	1.1	0.9
10^{-8}	2.5	2.3	0.1	—0.1

Table 3: Number of correct digits for Björck's two-pass algorithm on N data points chosen randomly from N(1.O, σ^2).

σ^2 \ N	64	256	1024	2048
1.0	5.2	5.1	4.0	4.0
10^{-1}	5.4	4.5	4.2	4.2
10^{-2}	5.6	4.5	4.4	3.7
10^{-3}	5.6	4.6	4.4	3.6
10^{-4}	5.2	4.8	3.9	3.7
10^{-5}	5.2	5.0	3.9	3.8
10^{-6}	5.4	5.0	4.1	4.0
10^{-7}	5.7	4.6	4.2	4.2
10^{-8}	6.2	4.6	3.8	3.3

Table 4: Number of correct digits for the generalized updating algorithm on 1024 data points chosen randomly from N(1.0, σ^2) with various values of m. (Note that $m = 1$ corresponds to algorithm (1.5) while $m = 1024$ is just the two-pass algorithm).

σ^2 \ m	1	2	4	8	16	32	64	128	256	512
1.0	4.0	4.0	4.3	4.6	4.9	5.0	5.0	5.1	5.0	4.2
10^{-1}	4.2	4.2	4.5	4.8	5.0	5.1	5.2	5.3	4.5	4.3
10^{-2}	4.5	4.4	4.7	5.0	5.1	5.3	5.4	4.9	4.5	4.4
10^{-3}	4.1	4.2	4.5	4.8	5.0	5.2	5.2	4.8	4.6	4.6
10^{-4}	3.6	3.7	4.0	4.3	4.5	4.8	5.0	4.8	4.8	4.6
10^{-5}	3.2	3.4	3.8	4.1	4.3	4.6	4.8	5.1.	5.1	3.4
10^{-6}	2.5	3.0	3.3	3.7	4.0	4.3	4.4	4.6	4.6	2.4
10^{-7}	1.4	2.1	2.5	2.9	3.6	3.6	3.5	3.4	3.3	1.4
10^{-8}	0.4	1.0	1.7	2.4	3.0	2.8	2.5	2.4	2.3	0.4

Table 5: Number of corrcct digits for thc pairwise algorithm on N data points chosen randomly from N(1.O, σ^2).

σ^2 \ N	64	256	1024	2048
1.0	5.8	5.8	5.6	5.6
10^{-1}	6.0	5.7	5.7	5.7
10^{-2}	6.2	5.8	5.7	5.6
10^{-3}	5.9	6.0	5.6	5.6
10^{-4}	5.5	5.8	5.9	5.8
10^{-5}	4.7	5.2	5.4	5.4
10^{-6}	4.5	4.7	4.8	4.9
10^{-7}	3.9	4.2	4.3	4.4
10^{-8}	3.2	3.7	3.8	3.9

Table 6: Number of corrcct digits for the two-pass algorithm on N data points chosen randomly from N(1.O, σ^2). In this test the means wcre computed in double precision.

σ^2 \ N	64	25G	1024	2048
1.0	5.2	5.1	4.0	4.0
10^{-1}	5.3	4.5	4.2	4.2
10^{-2}	5.6	4.5	4.4	3.7
10^{-3}	5.6	4.6	4.4	3.6
10^{-4}	5.2	4.8	3.9	3.7
10^{-5}	5.1	5.0	3.9	3.8
10^{-6}	5.2	5.0	4.1	4.0
10^{-7}	5.4	4.5	4.2	4.2
10^{-8}	5.6	4.5	4.4	3.7

Table 7: Number of correct digits for the generalized updating algorithm on 1024 data points chosen randomly from N(1.O, σ^2) with various values of m. In this test the running sums were computed in double precision. (Note that $m = 1$ corresponds to algorithm (1.5) while $m = 1024$ is just the two-pass algorithm).

σ^2 \ m	1	2	4	8	16	32	64	128	256	512
1.0	4.0	4.0	4.3	4.6	4.9	5.0	5.1	5.1	5.0	4.2
10^{-1}	4.2	4.2	4.5	4.8	5.0	5.1	5.2	5.3	4.5	4.3
10^{-2}	4.4	4.4	4.7	4.9	5.2	5.3	5.4	4.9	4.5	4.4
10^{-3}	4.4	4.4	4.7	4.9	5.2	5.3	5.3	4.8	4.G	4.6
10^{-4}	3.9	3.9	4.2	4.5	4.8	5.0	4.9	4.8	4.8	4.8
10^{-5}	3.9	3.9	4.2	4.5	4.8	5.0	5.0	4.9	4.9	4.3
10^{-6}	4.1	4.1	4.3	4.G	4.9	5.0	5.1	5.1	4.8	4.2
10^{-7}	4.2	4.2	4.5	4.8	5.0	5.2	5.2	5.3	4.5	4.3
10^{-8}	4.4	4.4	4.7	5.0	5.2	5.3	5.4	4.9	4.5	4.4

Table 8: Number of correct digits for the pairwise algorithm on N data points chosen randomly from N(1.O, σ^2). In this test the running sums were computed in double precision.

σ^2 \ N	64	256	1024	2048
1.0	5.9	5.8	5.7	5.7
10^{-1}	5.9	5.7	5.7	5.7
10^{-2}	6.0	5.7	5.7	5.6
10^{-3}	6.0	5.7	5.6	5.5
10^{-4}	5.9	5.8	5.7	5.6
10^{-5}	5.9	5.8	5.6	5.6
10^{-6}	5.9	5.8	5.7	5.6
10^{-7}	6.0	5.8	5.7	5.7
10^{-8}	6.1	5.8	5.8	5.6

9. A **FORTRAN** implementation of the **pairwise** algorithm.

```
      SUBROUTINE UPDATE(M,N,SUM,S,X)
      INTEGER M,N
      REAL*8 S ,SUM,X(N)
C
C     GIVEN THE SUM BND SUM OF SQUARES OF DEVIATIONS FROM THE
C     MEAN FOR A SAMPLF OF M POINTS,
C
C                  M
C         SUM = SUM Y(1)
C               I=1
C
C               M                    2
C         S  = SUM  (Y(I)   - SUM/M)
C              I=1
C
C     AND GIVEN N NEW DATA POINTS X(1)...X(N), THIS ROUTINE PRODUCES
C     THE SUM AND-SUM OF SQUARES FOR THE COMBINED SAMPLE:
C
C                         N
C         SUM  := SUM + SUM X(I)
C                       I=1
C
C                M                       2     N                     2
C         S := SUM (Y (I)  - SUM/(M+N))   + SUM (X(I)  - SUM/(M+N) )
C              I= 1                           I=1
C
C     THE SUY AND SUM OF SQUARES FOR THE NEW POINTS ARE CALCULATED
C     USING THE PAIRWISE ALGORTIHM.  THE OLD SUM AND SUM OF SQUARES
C     IS THEN UPDATED.
C
C     THIS ROUTINE HRS LOCALLY DIMENSIONED ARRAYS TERMS, SUMA AND
C     SA WHICH CURRENTLY HAVE DIMENSION 21.  THIS LIMITS THE
C     NUMBER OF POINTS WHICH CAN BE HANDLED TO N <= 2**20 = 1048576.
C     TO USE WITH LAFGEF N, INCREASE THESE DIMENSIONS TO SOMETHING
C     AT LEAS';' AS LARGE AS LOG2(N) + 1.
C
      INTEGER TERMS(21),TOP,T
      REAL*8 SUMA(21),SA(21),MEAN,NSUM,NS
C
      TERMS(1) = 0
      TOP = 2
      N2 = N/2
      IF (N .LE. 0) GC TC 70
    5 IF (N .GT. 1) GO TO 6
         NSUM = X(1)
         NS = 0
         GO TO 50
C
```

```
    6 DO 20 I=1,N2
C         # COMPUTE THE SUM AND SUM OF SQUARES FOR THE NEXT TWO
C         # DATA POINTS IN X.  PUT THESE QUANTITIES ON TOP OF
C         # THE STACK.
          SUMA(TOP) = X(2*I-1) + X(2*I)
          SA(TOP) = (X(2*I) - X(2*I-1))**2 / 2.0
          TERMS(TOP) = 2
   13     IF (TERMS(TOP) .NE. TERMS(TOP-1)) GO TO 20
C             # TOP TWO ELEMENTS ON STACK CONTAIN QUANTITIES COMPUTED
C             # FROM THE SAME NUMBER OF DATA POINTS.  COMBINE THEM:
              TOP = TOP-1
              TERMS(TOP) = 2 * TERMS(TOP)
              SA(TOP) = SA(TOP) + SA(TOP+1) + (SUMA(TOP) - SUMA(TOP+1))**2
     X                    / TERMS(TOP)
              SUMA(TOP) = SUMA(TOP) + SUMA(TOP+1)
              GO TO 10
   23     TOP = TOP+1
C
      TOP = TOP-1
      IF (2*N2 .EQ. N) GO TO 30
C         # N IS ODD.  PUT LAST POINT ON STACK:
          TOP = TOP+1
          TERMS(TOP) = 1
          SUMA(TOP) = X(N)
          SA(TOP) = 0.0
   30 T = TERMS(TOP)
      NSUM = SUMA(TOP)
      NS = SA(TOP)
      IF (TOP .LT. 3) GO TO 50
C         # N IS NOT A POWER OF 2, THE STACK CONTAINS MORE THAN
C         # ONE ELEMENT. COMBINE THEM:
          DO 40 J=3,TOP
              I = TOP+2 - J
              NS = NS + SA(I) + T*(TERMS(I)*NSUM/T - SUMA(I))**2 /
     X                 (TERMS(I) * (TERMS(I)+T))
              NSUM = NSUM + SUMA(I)
   43         T = T+TERMS(I)
C
C
   50 CONTINUE
C     # COMBINE NS AND NSUM WITH S AND SUM RESPECTIVELY:
      IF (M .EQ. 0) GO TO 60
          NS = S + NS + M*(N*SUM/M - NSUM)**2 / (N*(N+M))
          NSUM = SUM + NSUM
   60 S = NS
      SUM = NSUM
C
   70 RETURN
      END
```

Acknowledgements.

We are particularly indebted to Å. Björck for his suggestions concerning the two-pass algorithm.

References.

[1] Chan, T.F.C., and Lewis, J.G. Computing standard deviations: accuracy. CACM 22,9(Sept. 1979), 526-531.

[2] Ghan, T.F.C., and Lewis, J.G. Rounding error analysis of algorithms for computing means and standard deviations. Tech. Rep. No. 284, Dcpt. of Mathematical Sciences, The Johns Hopkins University, Baltimore, Md., April 1978.

[3] Hanson, R.J. Stably updating mean and standard deviations of data. CACM 18,8(Aug. 1975), 458.

[4] Tsao, N. On "accurate" evaluation of extended sums, manuscript, Dept. of Computer Science, Wayne State University.

[5] West, D.H.D. Updating mean and variance estimates: an improved method. CACM 22,9(Sept. 1979), 532-535.

. [6] Youngs, E.A., and Cramer, E.M. Some results relevant to choice of sum and sum-of-product algorithms. Technometrics 13(Aug. 1975), 458.