Paul Avery
CBX 95–55
June 30, 1996

# Combining Measurements with Correlated Errors

**Abstract**

I discuss the problem of combining measurements whose errors are correlated because of common uncertainties such as tracking and shower efficiencies. It turns out that the optimal (in the sense of giving the smallest error in the average) weights for the measurements are influenced by these correlations in a way that is trivial to calculate. Several routines which aid the calculation of quantities composed of variables with correlated systematic errors are documented. I also discuss how the tracking systematic errors in the recent CLEO measurements of the $D^0 \to K^-\pi^+$ and $D^+ \to K^-\pi^+\pi^+$ branching fractions are anti-correlated with $B \to D^{(*)}n\pi$ tracking errors and thus serve to *reduce* the overall systematic error in the calculation of these branching fractions. The results shown here do not invalidate the Big Bang theory.

## 1 Working with Independent Measurements

The problem of averaging together branching fractions with different errors is one steeped in tradition and mystery at CLEO. Anyone who has ever served time as a graduate student in experimental physics has had to deal with a version of this problem, which can be stated quite simply. Suppose one is trying to determine the value of a quantity $x$, and there are $n$ independent measurements $x_i$ of it with corresponding errors $\sigma_i$. Any statistics textbook will explain that the best linear estimator for $x$ is $\overline{x}$, defined by

$$\overline{x} = \frac{\sum_i x_i/\sigma_i^2}{\sum_i 1/\sigma_i^2}. \tag{1}$$

The astute reader [1] will note that this result depends on a precise understanding of the words *independent* and *best*. For our purposes, measurements $i$ and $j$ are independent if they are uncorrelated, i.e. if

$$\langle \delta x_i \delta x_j \rangle = 0 \qquad i \neq j, \tag{2}$$

where $\delta x_i = x_i - \langle x_i \rangle$.

A "best" estimator in statistical theory also has a precise meaning. In the problem above, for example, one could construct an infinite number of linear estimators from the measurements $x_i$. However, the one shown above is the best one because (1) it is *unbiased* and (2) it is *efficient*. Let's look at what unbiased means first. Suppose one conducts an experiment where several sample measurements are taken and he constructs an estimator for the quantity in question using this sample. If the experiment is repeated an infinite number of times, we say that the estimator is unbiased if its average over all experiments is equal to the true value of the quantity.

However, our real world experience tells us that unbiasedness is not enough. For example, if we made 10 measurements with different errors, it is perfectly permissible to take the first measurement as the estimator. This estimator is certainly unbiased, but is not what a smart individual [2] would choose. A better choice would be the unweighted average of the 10 measurements, but it suffers from the problem of not discriminating between "good" measurements (those with small errors) and "bad" ones (those with large errors). What we are looking for is an *efficient* estimator of the quantity, which is defined in statistical theory as the estimator which on average has the smallest standard deviation from the true value. The estimator shown in equation (1) is efficient in that sense.

# 2 Correlated Measurements

In CLEO the problem of averaging branching fractions is complicated by the fact that different measurements have correlated errors from tracking efficiency uncertainties, $\pi^0$ efficiencies and common branching fraction uncertainties. Although I laid out in a previous note [3] the general solution for fitting $m$ parameters to $n$ measurements, including the case where the measurements are correlated, it is useful here to consider specifically the case where we are averaging together measurements of the same quantity with errors that might be correlated.

Again, suppose we have $n$ measurements $x_i$ of a particular branching fraction. Since the measurements are correlated we need more information than just the individual errors. Accordingly, we define the *covariance matrix* $V_x$, whose terms are given by

$$(V_x)_{ij} = \langle \delta x_i \delta x_j \rangle \tag{3}$$

The diagonal members of $V_x$ are the variances, e.g., $(V_x)_{ii} = \sigma_i^2$. The *correlation coefficient* $r_{ij}$ is also useful. It is given by

$$r_{ij} = \frac{(V_x)_{ij}}{\sigma_i \sigma_j} \tag{4}$$

It can be proved that the $r_{ij}$ satisfy the inequality $-1 \leq r_{ij} \leq 1$.

Our goal is to construct the best linear estimator for the branching fraction which accounts for all the measurements, including their errors and correlations. The estimator can be defined generally as

$$\overline{x} = \sum_i w_i x_i \tag{5}$$

where the weights $w_i$ must be found. Assuming that the $x_i$ each provide an unbiased estimate of the branching fraction and requiring that the estimator $\overline{x}$ be unbiased gives the condition $\sum_i w_i = 1$. We now force the standard deviation of $\overline{x}$ to be as small as possible. Writing $\delta\overline{x} = \sum_i w_i \delta x_i$ and using the definition of the standard deviation, we get

$$\sigma_{\overline{x}}^2 = \sum_{ij} w_i w_j (V_x)_{ij} \tag{6}$$

To minimize $\sigma_{\overline{x}}^2$ subject to the condition $\sum_i w_i = 1$ we use the Lagrange multiplier technique. We write

$$\sigma_{\overline{x}}^2 = \sum_{ij} w_i w_j (V_x)_{ij} + \lambda(\sum_i w_i - 1) \tag{7}$$

and set the derivative of $\sigma_{\overline{x}}^2$ with respect to the $w_i$ and $\lambda$ to zero. This gives the solution [1]

$$w_i = \frac{\sum_k (V_x^{-1})_{ik}}{\sum_{jk} (V_x^{-1})_{jk}} \tag{8}$$

Note that this expression for $w_i$ is identical to equation (1) when the errors are independent, since in that case $(V_x^{-1})_{ij} = \delta_{ij}/\sigma_i^2$.

Some software routines which permit simple handling of correlated errors are described in the Appendix.

# 3 Counterintuitive Results Using Correlated Errors

## 3.1 Non-positive weights

Some of the more perspicacious among you [4] will have already noticed that it is sometimes possible for the weights calculated in equation (8) to be negative. This behavior turns out to be quite reasonable in certain circumstances, as seen from the following example.

Suppose there are two measurements of $x$, $1.0 \pm 0.1$ and $1.0 \pm 0.2$, and that the measurements are 100% correlated with a correlation coefficient of $+1$. It is easy to see using equation (8) that the weights which give the "best" estimate of $x$ are 2.0 and $-1.0$, e.g., $\overline{x} = 2x_1 - x_2$. This makes sense, because if $x_1$ fluctuates upward by 1 standard deviation (+0.1), then by virtue of the correlation coefficient being 1.0, $x_2$ will fluctuate up also by 1 standard deviation (+0.2) and the estimator $\overline{x} = 2x_1 - x_2$ gives 1.0 for the measurement. In fact, it is easy to see for this special case that this estimator is exact because all fluctuations are precisely canceled. Thus negative weights per se are not a problem.

Since independent errors give positive weights and positively correlated errors pull the weights negative, it is not surprising that special cases can arise where some of the weights are zero. Consider the two measurements, $x_1 = 1.0 \pm 0.1 \pm 0.1$ and $x_2 = 1.0 \pm 0.1 \pm 0.2$, where the first error is statistical (uncorrelated) and the second is systematic and 100% correlated with the other one. The covariance matrix is

$$V_x = \begin{pmatrix} 0.02 & 0.02 \\ 0.02 & 0.05 \end{pmatrix} \tag{9}$$

Application of equation (8) yields weights of 1.0 and 0.0, respectively. Thus the best estimator is obtained by ignoring the second measurement altogether! What has happened is that the second measurement does not contribute any more information about the desired quantity. Other cases of this type can be constructed with more measurements. Try working out the $n = 3$ case using the values $x_1 = 1.0 \pm 0.1 \pm 0.1$, $x_2 = 1.0 \pm 0.1 \pm 0.2$ and $x_3 = 1.0 \pm 0.1 \pm 0.2$ and show that the weights are 1, 0 and 0.

[1] Actually, $V_x^{-1}$ does not have to exist for the weights to be determined. Such singular covariance matrices arise when only a fully correlated error is present. The simplest way to deal with this situation numerically is to add a small term $\epsilon$ to the diagonal elements of $V_x$.

### 3.2 Sensitivity of weights to correlations

For measurements with substantial correlated errors (i.e., comparable to uncorrelated errors) the values of the optimal weights are sensitive to the precise values of the errors, so it might be wise not to put too much trust in the derived weights. A simple example illustrates this situation. Consider the two measurements

$$\begin{aligned} x_1 &= 0.95 \pm 0.08 \pm 0.08, \\ x_2 &= 1.08 \pm 0.08 \pm 0.08, \end{aligned} \tag{10}$$

where the first error is statistical and the second is systematic and 100% positively correlated with the other systematic error. Plugging the numbers into the routines described in the Appendix, we get for the combined measurement

$$\begin{aligned} \overline{x} &= 1.015 \pm 0.057 \pm 0.080, \\ \overline{x} &= 1.015 \pm 0.057 \pm 0.057, \end{aligned} \tag{11}$$

with the errors in the same order. The two results assume that the systematic errors are correlated and uncorrelated, respectively. The corresponding optimal weights are $(0.50, 0.50)$ for both cases.

Now change the first systematic error from 0.08 to 0.10 and compute the average again. This yields the averages

$$\begin{aligned} \overline{x} &= 1.033 \pm 0.059 \pm 0.087, \\ \overline{x} &= 1.023 \pm 0.057 \pm 0.063, \end{aligned} \tag{12}$$

again including and not including the correlation. The new weights are

$$\begin{aligned} \mathbf{w} &= (0.364, 0.636), \\ \mathbf{w} &= (0.438, 0.562), \end{aligned} \tag{13}$$

for these cases. Note that when correlations are included the weights change much more than they would if the correlations are ignored. The average is not affected much, however, while the statistical error of the combined measurement is somewhat larger for the correlated case, mostly because of the greater weighting of the second measurement.

## 4 Examples of Correlated Errors in CLEO

### 4.1 Exclusive $B$ decay modes to $D^{(*)}X$

Nowhere do correlated errors crop up more than in the Big $B$ paper published by CLEO in 1994 [5]. For example, to calculate the branching fraction for $B^- \to D^0\pi^-$, three different $D^0$ submodes needed to be averaged together: $D^0 \to K^-\pi^+$, $D^0 \to K^-\pi^+\pi^0$ and $D^0 \to K^-\pi^+\pi^+\pi^-$. Each of these modes had an assumed tracking uncertainty of 2% per track. The branching fraction for the $K^-\pi^+$ submode has calculated to be

$$\mathcal{B}(B^- \to D^0\pi^-) = \frac{N_{\rm obs}}{N_{B^-}\epsilon_1 \mathcal{B}_{K\pi}} \tag{14}$$

where $N_{\rm obs}$ was the number of observed candidates, $N_{B^-}$ was the number of $B^-$ mesons, $\epsilon_1$ was the efficiency and $\mathcal{B}_{K\pi}$ was the $D^0 \to K^-\pi^+$ branching fraction.

All of these quantities have statistical and systematic errors. However $\mathcal{B}_{K\pi} = 3.91 \pm 0.08 \pm 0.17\%$ itself included a 2% error per track systematic uncertainty which, because it was calculated from an efficiency which appears in the denominator, is *negatively* correlated with the tracking uncertainty in $\epsilon_1$. Unfortunately, assessing the degree of cancelation is somewhat difficult because the momentum spectrum of tracks used in determining $\mathcal{B}(D^0 \to K^-\pi^+)$ was different (stiffer) than that for the tracks used in this analysis. To do a better job we would need to look for systematic momentum dependencies in the tracking efficiencies. The solution that was adopted in the paper was to remove the tracking error from the $D^0 \to K^-\pi^+$ branching fraction and include it when forming the systematic error on the measurement. This ensures that the error is not double counted, but it is a very conservative course of action since the errors probably cancel to a significant degree.

## 4.2 $\tau \to h^-h^+h^-\nu_\tau$

Two different analyses of tau decays to three hadrons [6, 7, 8] were performed which had common systematic tracking and shower finding errors. The combined branching fraction having the smallest errors was found by a technique similar to that derived here but limited to two measurements. They authors noted that the weights tended to be somewhat sensitive to the assumed size of the systematic errors, although the final errors did not change much.

# Appendix

## Callable Routines Which Use Correlated Errors

I have written a set of routines which implement error handling in a way that is reasonably simple to use. The routines can be found in the library `libcorrel.a`. All variables marked with an asterisk (*) refer to output variables. Arrays are shown together with their maximum range in parenthesis, e.g., `xin(nxin)`.

The meaning of the input errors in these routines must be defined. The input quantities are denoted `xin(nxin)`, where `nxin` is the number of input variables. The input errors are specified by the array `xin_err(merr,nxin)`, where `merr` must be at least as large as the number of errors. For example, every quantity might have 4 errors defined as follows:

| | | |
|---|---|---|
| 1 | Statistical | ($\texttt{err\_type} = 0 \Longrightarrow \texttt{err1}$) |
| 2 | Uncorrelated systematic | ($\texttt{err\_type} = 0 \Longrightarrow \texttt{err2}$) |
| 3 | Track systematic | ($\texttt{err\_type} = 1 \Longrightarrow \texttt{err3}$) |
| 4 | pi0 systematic | ($\texttt{err\_type} = 1 \Longrightarrow \texttt{err4}$) |

The measurements and the errors can be written mathematically as

$$\begin{array}{l} \texttt{x}_1 \pm \texttt{err1}_1 \pm \texttt{err2}_1 \pm \texttt{err3}_1 \pm \texttt{err4}_1 \\ \texttt{x}_2 \pm \texttt{err1}_2 \pm \texttt{err2}_2 \pm \texttt{err3}_2 \pm \texttt{err4}_2 \\ \texttt{x}_3 \pm \texttt{err1}_3 \pm \texttt{err2}_3 \pm \texttt{err3}_3 \pm \texttt{err4}_3 \end{array}$$

Note the following

1. Errors in different columns are always independent.

2. Errors in the same column are independent only if `err_type` = 0, otherwise they are 100% correlated.

3. Negative correlations are obtained by setting `err_type` = 1 and making the corresponding error negative.

In the above example, the last two errors are 100% correlated with the other errors in that column. The first two errors are independent of other errors in the same columns. When the output quantity `xout` is calculated, it will have 4 output errors `xout_err`, corresponding one-to-one with the input errors.

```
integer function err_simulate(num, nxin, xin, nerr, err_type, xin_err,
                              merr, xout_mean*, xout_err*, func)
```

Using a small Monte Carlo generator and an array of input variables `xin` with correlated errors `xin_err`, calculate an output variable `xout` and its errors `xout_err`. This is done by throwing `num` $\times$ `nerr` random values of `xin` (using the errors and their correlations appropriately) and calling the user provided function `func(nxin, xin)` once per iteration to establish the variation of the output variable with each type of input error. The output errors `xout_err` are in one-to-one correspondence with the input errors `xin_err` as discussed above.

| | | |
|---|---|---|
| `num` | integer | Number of iterations |
| `nxin` | integer | Number of input variables |
| `nerr` | integer | Number of errors for each input variable |
| `err_type(nerr)` | int. array | Array of error types<br>0 $\rightarrow$ uncorrelated error<br>1 $\rightarrow$ correlated error |
| `xin_err(nerr,nxin)` | real array | Array of errors per input variable |
| `merr` | integer | First dimension of `xin_err` |
| `xout_mean` | real | Output value corresponding to initial inputs |
| `xout_err(nerr)` | real array | Output errors |
| `func` | real func. | Name of user supplied function (declared external in calling program). `func` is called once during each iteration with two arguments, (1) the number of input variables (`nxin`) and (2) the array of input variables (`xin`). |

```
integer function err_average_correl(nxin, xin, nerr, err_type, xin_err,
                                    merr, xout_mean*, xout_err*, weights*,
                                    chisq*)
```

Assuming input variables `xin` with *correlated* errors `xin_err`, calculate analytically the weighted average `xout` and its errors `xout_err`. The errors in `xin_err` and in `xout_err` are assumed to be in the same order as described previously. As a bonus, the (double precision) weights used in the calculation of the average and the $\chi^2$ of the fit are also returned.

| | | |
|---|---|---|
| `nxin` | integer | Number of input variables |
| `xin(nerr)` | real array | Array of input variables |
| `nerr` | integer | Number of errors for each input variable |
| `err_type(nerr)` | int. array | Array of error types<br>0 $\rightarrow$ uncorrelated error<br>1 $\rightarrow$ correlated error |
| `xin_err(nerr,nxin)` | real array | Array of errors per input variable |
| `merr` | integer | First dimension of `xin_err` |
| `xout_mean` | real | Output value corresponding to initial inputs |
| `xout_err(nerr)` | real array | Output errors |
| `weights(nerr)` | double array | Weights used to compute average |
| `chisq` | real | $\chi^2$ of fit to average |

```
integer function err_average_indep(nxin, xin, nerr, xin_err, merr,
                                   xout_mean*, xout_err*, weights*, chisq*)
```

Assuming a set of input variables `xin` with *independent* errors `xin_err`, calculate analytically the optimal weighted average `xout` and its errors `xout_err`. The errors in `xin_err` and in `xout_err` are assumed to be in the same order. As a bonus, the (double precision) weights used in the calculation of the average and the $\chi^2$ of the fit are also returned.

| | | |
|---|---|---|
| `nxin` | integer | Number of input variables |
| `xin(nerr)` | real array | Array of input variables |
| `nerr` | integer | Number of errors for each input variable |
| `xin_err(nerr,nxin)` | real array | Array of errors per input variable |
| `merr` | integer | First dimension of `xin_err` |
| `xout_mean` | real | Output value corresponding to initial inputs |
| `xout_err(nerr)` | real array | Output errors |
| `weights(nerr)` | double array | Weights used to compute average |
| `chisq` | real | $\chi^2$ of fit to average |

```
integer function err_average_weight(nxin, xin, nerr, err_type, xin_err,
                                    merr, weights, xout_mean*, xout_err*)
```

Assuming input variables `xin` with *correlated* errors `xin_err`, calculate analytically the weighted average `xout` and its errors `xout_err` using the weights (double precision) in the array `weights`. The errors in `xin_err` and in `xout_err` are assumed to be in the same order.

Note that the weights are input by the user and are not necessarily the optimal weights one would calculate from the input errors. This routine is called internally.

| | | |
|---|---|---|
| `nxin` | integer | Number of input variables |
| `xin(nerr)` | real array | Array of input variables |
| `nerr` | integer | Number of errors for each input variable |
| `err_type(nerr)` | int. array | Array of error types<br>$0 \rightarrow$ uncorrelated error<br>$1 \rightarrow$ correlated error |
| `xin_err(nerr,nxin)` | real array | Array of errors per input variable |
| `merr` | integer | First dimension of `xin_err` |
| `weights(nxin)` | double array | Weights to use in computing average |
| `xout_mean` | real | Weighted average |
| `xout_err(nerr)` | real array | Errors of weighted average |

```
integer function err_compute_weights(Vxinv, mvx, nxin, weights*)
```

Using the covariance matrix inverse `Vxinv` (computed perhaps by `err_compute_vxinv`), return the optimal weights needed for computing the average of a set of measurements. These weights can be passed to the routine `err_average_weight`.

| | | |
|---|---|---|
| `Vxinv(nxin,nxin)` | double array | Covariance matrix inverse returned to caller |
| `mvx` | integer | First dimension of `Vxinv` |
| `nxin` | integer | Number of input variables |
| `weights(nxin)` | double array | Optimal weights returned to caller |

```
integer function err_compute_vx(nxin, nerr, err_type, xin_err, merr,
                                Vx*, mvx)
```

Using the *correlated* errors `xin_err`, calculate analytically the `nxin` × `nxin` covariance matrix. Note that the returned covariance matrix is double precision.

| | | |
|---|---|---|
| `nxin` | integer | Number of input variables |
| `nerr` | integer | Number of errors for each input variable |
| `err_type(nerr)` | int. array | Array of error types<br>0 → uncorrelated error<br>1 → correlated error |
| `xin_err(nerr,nxin)` | real array | Array of errors per input variable |
| `merr` | integer | First dimension of `xin_err` |
| `Vx(nxin,nxin)` | double array | Covariance matrix returned to caller |
| `mvx` | integer | First dimension of `Vx` |

```
integer function err_compute_vxinv(nxin, nerr, err_type, xin_err, merr,
                                   Vxinv*, mvx)
```

Using the *correlated* errors `xin_err`, calculate analytically the inverse of the `nxin` × `nxin` covariance matrix. Note that the returned covariance matrix inverse is double precision.

| | | |
|---|---|---|
| `nxin` | integer | Number of input variables |
| `nerr` | integer | Number of errors for each input variable |
| `err_type(nerr)` | int. array | Array of error types<br>0 → uncorrelated error<br>1 → correlated error |
| `xin_err(nerr,nxin)` | real array | Array of errors per input variable |
| `merr` | integer | First dimension of `xin_err` |
| `Vxinv(nxin,nxin)` | double array | Covariance matrix inverse returned to caller |
| `mvx` | integer | First dimension of `Vxinv` |

```
real function err_combine(nerr, errs)
```

Combine the "nerr" errors in array "errs" into a single error. The errors are combined in quadrature.

| | | |
|---|---|---|
| `nerr` | integer | Number of errors |
| `errs(nerr` | real array | Array of errors |

```
subroutine err_print(lun, var, var_name, nerr, var_err, err_name)
```

Print to Fortran unit `lun` a summary of a quantity and its errors. This routine is useful for printing out an output variable summary in a readable format.

| | | |
|---|---|---|
| `lun` | integer | Fortran unit number |
| `var` | real | Value of variable |
| `var_name` | character | Name of variable |
| `nerr` | integer | Number of errors |
| `var_err(nerr)` | real array | Array of errors for variable |
| `err_name(nerr)` | character array | Names of errors |

# References


[1] A truly astute reader would never have looked here.

[2] What are you doing here? Go back and read the article.

[3] P. Avery, *Applied Fitting Theory I: General Least Squares Theory*, CBX 91–72.

[4] Nope, not you.

[5] M.S. Alam *et al.*, Phys. Rev. D **50**, 43 (1994).

[6] M. Lohner and J. Smith, CBX 95–32.

[7] V. Shelkov and T. Skwarnicki, CBX 95–52.

[8] M. Lohner, V. Shelkov and J. Smith, and T. Skwarnicki, CLEO Draft 95–32A. This paper draft includes a brief discussion of correlated errors. See the talk by T. Skwarnicki given at the June 27, 1995 CLEO meeting.