1. Introduction

Triple collocation was introduced in meteorology by Stoffelen [1998] as a method to obtain error variances and linear intercalibration coefficients for three collocated data sets. It has been applied to wind components as well as other geophysical quantities as soil moisture and wave height. The method has also been extended to include more collocated measurements [*Vogelzang and Stoffelen*, 2022].

The value of the triple collocation method is that it allows to calculate absolute error variances. To do so collocated measurements from at least three independent systems are needed.

This package contains software to do a basic triple collocation analysis. It is written in Python, because that language is free, powerful, and implemented on most computer platforms.

The triple collocation method is based on a number of assumptions, the most important of which are:

1. Linear calibration is sufficient;
2. The measurement errors are uncorrelated to each other
3. The measurement errors are constant over the range of measured values.

Before applying the triple collocation method one must assure oneself that these assumptions hold by inspecting scatter plots of one measurement system against the two others. If the points in the scatter plots lie on a straight line and are evenly distributed across that line, the assumptions are likely to hold. If the scatter plots look different, the results of the triple collocation method should be considered with utmost care. Since every user has hers or his favourite graphical software, scatter plots are not included in this package.

Another important aspect in triple collocation is the notion that different measurement systems have different spatial and temporal characteristics, hence different resolution. This gives rise to so-called representativeness errors that affect the results, see section 3.3.

Chapter 2 contains a description of the triple collocation package, the format of the input file, and the use of the software. The software can be used interactively from a Python environment or from a command line environment. The package comes with a test input file, and the test results are shown.

Chapter 3 contains a simple deviation of the triple collocation method. A number of derivations has been published in the scientific literature. Nevertheless, these may differ slightly in their formulation, so an extra derivation closely related to the software implementation is included here for the sake of completeness.

Chapter 4 contains some detailed information on the structure of the software.

2. Getting started

2.1 Contents

The package consists of the following files:

- `tripcol.py` Python source code
- `triple_collocation_module.py` Python source code
- `collocations_in_u` Test file with collocations
- `triple_collocation_manual.pdf` This document

Put these files together in a directory of your choice. File `tripcol.py` contains a stand-alone Python program for use in a command line environment like a Linux terminal. File `triple_collocation_module.py` contains the main software for doing triple collocation, and is fit for use from an interactive Python environment. The software comes with a test input file and this documentation.

The software is written in Python 3, but is compatible with Python 2. It has been tested with Python 3.6.8 and Python 2.7.15 on a Linux machine, and Python 3.9 on a Windows PC.

Effort has been made to keep the code as clear as possible. As a side effect, some superfluous calculations are done. However, the example test file with 3382 collocated values is processed within a second using the default settings, so time constraints are not important. If you want to process a huge number of collocation files or extremely large collocation files you may want to consider an implementation in a language designed for numerical efficiency like Fortran.

2.2 Collocation file

The input for the software is a file with collocated measurements. This file should be in ASCII format, and each line should contain three measurement values, one by each system. See the test file `collocations_in_u` for an example. In this document the measurement systems are numbered 0, 1, and 2, following the Python convention that indices start at zero.

The triple collocation software will consider the first system, system 0, as the calibration reference. It is recommended to use the measurement system with the highest resolution as system 0 and that with the coarsest resolution as system 2, in particular if representativeness errors are used.

The collocation file contains the zonal wind component u measured by buoys, ASCAT-A scatterometer, and ECMWF IFS forecasts.

2.3 Use from command line

The command for running the triple collocation program from a Linux command line environment has the form

```
python tripcol.py <-i input> {options}
```

the first command line argument, `-i` (or `--input`) is mandatory and should contain the name and path of the file with collocations, the other arguments are optional. Without arguments the program returns an error message and instructions how to use the program with a list of the available command line options:

```
tc:
tc:  program tripcol.py - Python program for triple collocation
tc:
tc:  ERROR: no file with collocations given
tc:
tc:  usage: python tripcol.py <-i IN> [OPTIONS]
tc:  with < > mandatory arguments and [ ] free options
tc:
tc:  mandatory arguments:
tc:  -i  <IN>          : read collocations from ASCII file IN
tc:  --input <IN>      : same as -i
tc:
tc:  free arguments:
tc:  -f <F>            : set sigma test factor to F (default: 4.0)
tc:  --f_sigma <F>     : same as -f
tc:
tc:  -m <M>            : set maximum number of iterations to M (default: 20)
tc:  --maxiter <M>     : same as -m
tc:
tc:  -p <EPS           : set precision to EPS (default: 0.00001)
tc:  --precision <EPS> : same as -p
tc:
tc:  -r <R2>           : set representativeness error variance to R2 (default: 0.0)
tc:  --reprerr <R2>    : same as -r
tc:
tc:
tc:  program tripcol.py aborted
```

2.4 Interactive use

When in a Python environment, the commands

```
from triple_collocation_module import do_tc

result = do_tc(input, {options})
```

load the triple collocation module and perform the triple collocation calculation, respectively. The first argument of function `do_tc` is mandatory and should contain the name and path of the input file with collocated measurements. The remaining arguments are optional and of the usual form `keyword = value`. The keywords, their type, their default values, and their relation to the symbols in chapter 3 (when applicable) are given in table 2.1.

Keyword	Type	Default value	Symbol
input_file	string		
f_sigma	float	4.0	F_σ
repr_err	float	0.0	r_1^2
max_nr_of_iterations	integer	20	
precision	float	10^{-5}	ϵ
verbosity	integer	1	

Table 2.1 Arguments of function `do_tc`.

The relevant results are stored in a list (called `result` in the example above) as follows:

result[0] = $[a_0, a_1, a_2]$	calibration scalings
result[1] = $[b_0, b_1, b_2]$	calibration biases
result[2] = $[\sigma_0^2, \sigma_1^2, \sigma_2^2]$	error variances
result[3] = τ^2	common variance
result[4] = n_{acc}	collocations that passed the variance test
result[5] = n_{rej}	collocations that were rejected by the variance test

again using the symbols defined in chapter 3 (if applicable).

2.5 Output conventions

The output calibration coefficients are defined such that the relation between the calibrated value $\bar{x}_i$ (the overbar indicating calibrated data) and the uncalibrated one x_i reads

$$\bar{x}_i = (x_i - b_i)/a_i \tag{2.1}$$

with $i = 0,1,2$ the index of the measuring system, following the Python convention that indices start with zero. This equation can be written as

$$\bar{x}_i = \tilde{a}_i x_i + \tilde{b}_i \tag{2.2}$$

with

$$\tilde{a}_i = \frac{1}{a_i} \ , \quad \tilde{b}_i = -\frac{b_i}{a_i} \tag{2.3}$$

and with $a_0 = 1$ and $b_0 = 0$ since system 0 is considered as calibration reference. Equation (2.1) or (2.2) is simply rewritten to a form in which system 1 or system 2 is taken as calibration reference.

The error variances calculated by the program are for to the calibrated data, not for the original uncalibrated data.

2.5 Test run

A test collocation file named `collocations_in_u` is part of the package. The command

```
python tripcol.py -i collocations_in_u
```

in a Linux command line environment, or the commands

```
from triple_collocation_module import do_tc

result = do_tc("collocations_in_u")
```

in a Python environment should produce the following output:

```
tc:
tc:  settings for triple collocation
tc:  - input collocation file            : collocations_in_u
tc:  - sigma test factor                 :     4.000000
tc:  - maximum number of iterations      :           20
tc:  - precision                         :     0.000010
tc:  - representativeness error variance :     0.000000
tc:  - verbosity level                   :            1
tc:
tc:  triple collocation converged at iteration 4
tc:  final results, calibration in the form of t = (x - b)/a
tc:                                      system 0    system 1    system 2
tc:  ------------------------------------------------------------------------
tc:  - calibration scalings a      :     1.000000    1.000272    0.967527
tc:  - calibration biases b        :     0.000000    0.165876    0.030271
tc:  - error variances             :     1.367916    0.325187    2.009558
tc:  - error standard deviations   :     1.169580    0.570252    1.417589
tc:
tc:  - common variance             :    41.804757
tc:  - accepted collocations       :         3351
tc:  - rejected collocations       :           31
tc:  - total number of collocations:         3382
tc:
tc:  triple collocation completed succesfully
tc:
```

3. Triple collocation method

3.1 Triple collocation equations

The error model employed in the triple collocation software reads

$$x_i = a_i(t + \varepsilon_i) + b_i \tag{3.1}$$

where x_i stands for a collocated measurement by system i, with $i = 0,1,2$ (following the Python convention that an index starts with zero), t for the signal common to all three systems (sometimes referred to as true signal or truth), a_i for the calibration scaling, b_i for the calibration bias, and ε_i for the random error of system i. The inverse of (3.1), i.e., the relation between the calibrated measurement value $\bar{x}_i$ and the uncalibrated value x_i reads

$$\bar{x}_i = \frac{(x_i - b_i)}{a_i} \tag{3.2}$$

where the bar indicates calibrated quantities. Note that in (3.1) the common signal t is determined by the system with coarsest resolution. Any small-scale signal measured by the other systems but not measured by the system with coarsest resolution is considered as an error, see section 3.3.

The triple collocation problem can be solved under the following assumptions:

1. Linear calibration is sufficient;
2. The errors ε_i in (1) have zero average and variance σ_i^2;
3. The errors ε_i are uncorrelated to each other and to the common signal t.

Before applying the triple collocation method one must assure oneself that these assumptions hold by inspecting scatter plots of system 1 versus system 0 and system 2 versus system 0. If the points in the scatter plots lie on a straight line and are evenly distributed along that line, the assumptions are likely to hold. If the scatter plots look different, the results of the triple collocation method should be considered with utmost care.

Without loss of generality we can choose system 0 as our reference system, so $a_0 = 1$ and $b_0 = 0$, and calibrate systems 1 and 2 relative to system 0. Since the calibration is linear, it can always be rewritten to a calibration with respect to any of the two other systems. Forming averages (first moments) denoted by the brackets $\langle\ \rangle$, one finds

$$M_i = \langle x_i \rangle = a_i \langle t \rangle + b_i \tag{3.3}$$

since $\langle \varepsilon_i \rangle = 0$ by assumption 2. From this the calibration bias can be obtained as

$$b_i = M_i - a_i \langle t \rangle \tag{3.4}$$

Now $b_0 = 0$ and $a_0 = 1$, because system 0 was chosen as calibration reference, so for $i = 0$ equation (3.4) reduces to

$$\langle t \rangle = M_0 \tag{3.5}$$

Substituting (3.5) into (3.4) yields

$$b_i = M_i - a_i M_0 \tag{3.6}$$

Next, we form second moments $M_{i,j} = \langle x_i x_j \rangle$ for $i, j = 0,1,2$, and obtain

$$M_{i,j} = a_i a_j \langle t^2 \rangle + \big(a_i b_j + a_j b_i\big)\langle t \rangle + b_i b_j + \langle \varepsilon_i \varepsilon_j \rangle \tag{3.7}$$

Substituting (3.6) in (3.7) yields

$$\begin{aligned} M_{i,j} &= a_i a_j \langle t^2 \rangle + \big(a_i (M_j - a_j M_0) + a_j (M_i - a_i M_0) b_i\big) M_0 + (M_i - a_i M_0)\big(M_j - a_j M_0\big) + \langle \varepsilon_i \varepsilon_j \rangle = \\ &= a_i a_j \langle t^2 \rangle - a_i a_j M_0^2 + M_i M_j + \langle \varepsilon_i \varepsilon_j \rangle \end{aligned} \tag{3.8}$$

Introducing the covariances

$$C_{i,j} = M_{i,j} - M_i M_j \tag{3.9a}$$

$$\tau^2 = \langle t^2 \rangle - M_0^2 \tag{3.9b}$$

where τ^2 is the variance common to the three systems, the equations for the second-order moments simplify to the covariance equations

$$C_{i,j} = a_i a_j (\tau^2 + \langle \varepsilon_i \varepsilon_j \rangle) \tag{3.10}$$

The covariance equations are very general and hold for any number of measurement systems. Note that $C_{i,j}$ is symmetric in its indices, so the system (3.10) consists of six independent equations.

Putting $\langle \varepsilon_i \varepsilon_j \rangle = \delta_{i,j} \sigma_i^2$, with $\delta_{i,j}$ the Kronecker delta, the covariance equations are readily solved. Note that the errors of the three measurement systems are assumed uncorrelated so the error covariances $\langle \varepsilon_i \varepsilon_j \rangle$ for $i \neq j$ are set to zero. This is necessary to reduce the number of unknowns in order to solve the covariance equations.

The diagonal equations give the error variances as

$$\sigma_i^2 = C_{i,i} - a_i^2 \tau^2 \tag{3.11}$$

and the off-diagonal equations can be solved for a_1, a_2, and τ^2 as

$$a_1 = \frac{C_{12}}{C_{02}} \tag{3.12}$$

$$a_2 = \frac{C_{12}}{C_{01}} \tag{3.13}$$

$$\tau^2 = \frac{C_{01} C_{02}}{C_{12}} \tag{3.14}$$

3.2 Iterative solution and variance test

The triple collocation equations are solved iteratively, for reasons to become clear below. The iterations starts with assuming that the three systems are properly calibrated, so $a_i = 1$ and $b_i = 0$. The measurements are then calibrated using (3.2) – which has no effect in the first iteration step – and the first and second moments and the covariances are calculated. In the previous section these were calculated from the raw, uncalibrated data, but now they are denoted with an overbar to indicate that they are calibrated from (presumably) calibrated data. Applying (3.12), (3.13) and (3.6), the corrections to the calibration coefficients read

$$\delta a_1 = \frac{\bar{C}_{12}}{\bar{C}_{02}} \tag{3.15}$$

$$\delta a_2 = \frac{\bar{C}_{12}}{\bar{C}_{01}} \tag{3.16}$$

$$\delta b_i = \bar{M}_i - a_i \bar{M}_0 \tag{3.17}$$

If the measurements were indeed well calibrated one would find $\delta a_i = 1$ and $\delta b = 0$. This will not be the case for the first iterations, and the calibration coefficients are updated according to

$$a_i = a_i \cdot \delta a_i \tag{3.18}$$

$$b_i = b_i + \delta b_i \tag{3.19}$$

This process is continued until the iteration has converged, i.e., when both $|a_i - 1| < \epsilon$ and $|b_i| < \epsilon$ for $i = 1,2$, where ϵ is a small number with default value 10^{-5}. For the test file the iteration converges in four steps, see section 2.5.

Now it may happen that the collocated data set under consideration contains outliers. These have a detrimental effect on the second moments. The triple collocation program detects them automatically with the variance test, also referred to as sigma test. The idea is that collocations that lie too far from the calibration curve should be omitted. This can be written as

$$\left(\bar{x}_i - \bar{x}_j\right)^2 > F_\sigma^2 \bar{D}_{i,j}^2 \tag{3.20}$$

with $\bar{D}_{i,j}^2$ the variance of $\bar{x}_i - \bar{x}_j$ calculated in a separate loop over all calibrated collocations, and F_σ a factor which equals 4 by default. The effect of (3.20) is that measurements that lie at a distance of more than four-sigma from the calibration line do not contribute to the first and second moments, hence the name "four-sigma test". In the form of (3.20) it is, of course, a variance test which is fully equivalent. Note that during the iteration procedure a collocation that is rejected may be included again in the next iteration when the calibration coefficients change in such a way that the distance of the calibrated measurements to the calibration curve has diminished enough.

So each iteration step consists of three substeps: (1) calculation of the distances $\bar{D}_{i,j}^2$ from calibrated data in a first loop over all collocations, (2) application of the variance test and calculation of the first and second moments from calibrated data in a second loop over all collocations, and (3) calculation of the covariances and solution of the covariance equations.

3.3 Representativeness error

As stated earlier, the three systems often measure at different spatial and/or temporal scales and therefore have not the same resolution. The triple collocation method determines the signal common to all three systems, which is therefore determined by the system with the coarsest resolution. Suppose that system 2 is the one with coarsest resolution. Signal at smaller scales measured by systems 0 and 1 will therefore be considered as an error - the representativeness error. Its variance, denoted as r_1^2, is the variance of the common small-scale signal measured by system 0 and system 1 but not by system

2. In some cases the representativeness error can be estimated, for instance from spectral methods [*Stoffelen,* 1998; *Vogelzang et al.*, 2011] or spatial statistics [*Vogelzang et al.*, 2015].

The representativeness error can be included in the error model by changing equation (3.1) to

$$x_0 = a_0(t + t_1 + \varepsilon_0) + b_0 \tag{3.21a}$$

$$x_1 = a_1(t + t_1 + \varepsilon_1) + b_1 \tag{3.21b}$$

$$x_2 = a_2(t + \varepsilon_2) + b_2 \tag{3.21c}$$

where t_1 stands for the small-scale signal measured by systems 0 and 1 but not by system 2. It is uncorrelated with all other variables in (3.21), has zero average (because any nonzero average would be included in the calibration biases) and variance r_1^2. Now the common signal in the error model (3.21a-c) is resolved up to the resolution of system 1. The equations can be solved as before, the only difference being that $C_{i,j}$ in (3.11) – (3.14) must be replaced by $C_{i,j} - r_1^2$ for $(i,j) = (0,0),(0,1),(1,0),(1,1)$. So the covariance equations now become

$$C_{i,j} = a_i a_j(\tau^2 + \langle \varepsilon_i \varepsilon_j \rangle + R_{i,j}) \tag{3.22}$$

with

$$R_{i,j} = \begin{pmatrix} r_1^2 & r_1^2 & 0 \\ r_1^2 & r_1^2 & 0 \\ 0 & 0 & 0 \end{pmatrix} \tag{3.23}$$

If a representativeness error is given, the triple collocation software will print the corrected error variances $C_{i,j} - r_1^2$ (if the verbosity is set to a value higher than 1). Representativeness errors will mainly affect the calibration scaling and error variances of system 2; those of systems 0 and 1 are quite insensitive to them – as long as $r_1^2 \ll C_{i,j}$. This can be inferred by expressing the error variances (3.11) in terms of the $C_{i,j}$, make the substitutions $C_{i,j} \to C_{i,j} - r_1^2$ at the appropriate places, and do a first-order Taylor expansion in r_1^2.

3.4 Some advanced notes

It may be possible that also the representativeness error of system 0 with respect to system 1 is known. It can be handled in the same way as in the previous section by introducing an additional signal component t_0 with variance r_0^2 in (3.21a). In particular, the covariance equations (3.22) hold, but now with

$$R_{i,j} = \begin{pmatrix} r_0^2 + r_1^2 & r_1^2 & 0 \\ r_1^2 & r_1^2 & 0 \\ 0 & 0 & 0 \end{pmatrix} \tag{3.24}$$

The only effect of r_0^2 will be to lower the error variance of system 0. This implies that the error variance of system 0 will be overestimated when r_0^2 is important but unknown, and therefore set to zero. This is the case for the buoy-ASCAT-ECMWF collocations in the test file that is included in this

package: the high resolution buoy measurements are much more precise than the results of the triple collocation analysis suggest.

From the covariance equations (3.22) one sees that the representativeness errors are valid for calibrated data. When representativeness errors are derived from uncalibrated data an extra calibration must be applied to them.

The error variances σ_i^2 calculated by the program are for calibrated data, because of the iterative solution scheme. They are easily decalibrated by multiplying with a_i^2.

4 Program design

4.1 Module `triple_collocation_module.py`

The triple collocation module contains two functions, `do_tc` and `print_settings`, and one class named `TCProcess`. It needs the standard Python modules `__future__`, needed to print output both under Python 2 and Python 3, and `math` for calculating square roots.

Function `do_tc` performs the triple collocation calculation and is intended for interactive use, as already mentioned in chapter 2. If an optional argument is not included in the call to `do_tc` it is kept at its default value. Function `do_tc` starts with a call to `print_settings` to print the current values of the parameters and settings listed in table 4.1. Then it starts the iteration loop to solve the triple collocation problem. The actual calculations are done by functions in the `TCProcess` class (see below). If the iteration did not converge a warning message is printed. The relevant results are stored in a list for further use, see chapter 2.

4.1.1 Class `TCProcess`

The actual triple collocation calculation is done by the functions in class `TCProcess`. The vectors and matrices involved are represented as lists and nested lists, respectively, so a matrix element $A_{i,j}$ is represented as `A[i][j]`. Note that all matrices involved are symmetric, so the order of the indices does not matter. This introduces some extra calculations that are not really needed, but it keeps the code simple and the program runs fast enough.

In function `__init__` the attributes of the `TCProcess` class are defined and initialised.

The calculation is split in three parts: (1) calculation of the difference variances, (2) calculation of the moments, and (3) calculation of the covariances, solution of the covariance equations, and calculation of the calibration biases. In the first part, function `update_distances` reads the collocations from file and calculates $\overline{D}_{i,j}^2$, the variance of the differences between each pair of measurements, needed for the variance test. This is done for all collocations. In the second part, function `update_moments` rereads the collocations from file, applies the variance test, and updates the moments for the collocations that passed the variance test. The third part of the calculation is done by function `solve`. It calculates the covariances $\bar{C}_{i,j}$ from the first and second moments, $\overline{M}_i$ and $\overline{M}_{i,j}$, and solves the covariance equations (3.10) according to (3.11) - (3.14), and calculates the calibration biases b_i according to (3.6).

The moments in functions `update_distances` and `update_moments` are calculated using the formula

$$A_{n+1} = A_n + \frac{a_n - A_n}{n+1} \tag{4.1}$$

which is equivalent to

$$A_{n+1} = \frac{1}{n+1}\sum_{i=1}^{n+1} a_n \tag{4.2}$$

Equation (4.1) is a very stable algorithm for calculating moments. Note that the collocation file is opened and closed twice during each iteration. This introduces some extra computer time, but improves readability of the code.

Functions `print_moments` and `print_results` print the results from `update_moments` and `solve`, respectively to screen. The amount of output is controlled by the verbosity parameter.

Table 4.1 shows the correspondence between some variables in chapter 3 and the attributes of the `TCProcess` class.

Symbol	Attribute
x_i	`x[i]`
$\bar{x}_i$	`xcal[i]`
a_i	`a[i]`
b_i	`b[i]`
σ_i^2	`errvar[i]`
δa_i	`da[i]`
δb_i	`db[i]`
$\overline{M}_i$	`M1[i]`
$\overline{M}_{i,j}$	`M2[i][j]`
$\bar{C}_{i,j}$	`C[i][j]`
$\overline{D}_{i,j}^2$	`dvar[i][j]`

Table 4.1 Some variables in chapter 3 and the corresponding attributes of the `TCProcess` class.

4.2 Program `tripcol.py`

Program `tripcol.py` is intended for use from a command line environment like a Linux terminal. It uses the standard module `argparse` for handling command line arguments and module `triple_collocation_module` described above for the triple collocation calculation. It contains a function `usage` for printing the use and available command line arguments in case `tripcol.py` is called without valid command line arguments (or no arguments at all), see chapter 2.

The program starts with defining the parameters and settings and giving them their default value. Then the command line arguments are read, overriding the default values. Finally, function `do_tc` is called to do the triple collocation calculation.

References

Stoffelen, A., 1998.
Toward the true near-surface wind speed: error modeling and calibration using triple collocation, *J. Geophys. Res*. 103C4, 7755-7766.

Vogelzang, J., A. Stoffelen, A. Verhoef, and J. Figa-Saldaña, 2011.
On the quality of high-resolution scatterometer winds. *J. Geophys. Res*. 116, C10033. doi: 10.1029/2010JC006640.

Vogelzang, J., G.P. King, and A. Stoffelen, 2015.
Spatial variances of wind fields and their relation to second-order structure functions and spectra, *J. Geophys. Res. Oceans*, 120, 1048–1064, doi:10.1002/2014JC010239.

Vogelzang, J. and A. Stoffelen, 2022.
On the accuracy and consistency of quintuple collocation analysis of in-situ, scatterometer, and NWP winds. *Remote Sens*. 2022, 14, 4552, doi: 10.3390/rs14184552.