The Olsen Filter for Data in Finance

ULRICH A. MÜLLER

UAM.1999-04-27

March 29, 2001

A consulting document by the O&A Research Group

Contents

Abstract

This document contains a description of the filtering technology developed at the Olsen Group.

Filtering of financial quotes has a history of more than 10 years at Olsen. Since 1997, Olsen has been refining a new, adaptive filtering algorithm. Adaptivity means the ability to apply the filter to many new financial instruments with just a minimum of preparation rather than a laborious reconfiguration. It also implies that the filter adapts to structural breaks in a time series, with no need to recalibrate any filter parameters by hand.

Olsen's filtering technology has passed the test of practice well. Many tests on a wide basis of financial instruments have led to a number of improvements; many special analysis steps for special data error types have been added.

1 Introduction

This document contains a description of the filtering technology developed at the Olsen Group.

Data filtering has a history of more than 10 years at Olsen. The problem of bad quotes and their damaging consequences in applications was recognized very early. First filtering algorithms were developed and applied, as shown in [Müller, 1989] and the appendix of [Dacorogna et al., 1993]. Although filtering as a field of research and development was underestimated by many people at that time, Olsen continued to improve the filtering technology.

In 1997, a major effort was made to raise Olsen's filtering capabilities to a new level. The new algorithm was specified by the document [Müller and Zumbach, 1997] and implemented in C++. The new Olsen filter was tested and improved for a widely enlarged set of financial instruments. New, specific data error types were detected; tests for these new errors were added to the filter algorithm. The new Olsen filter has reached a high degree of reliability.

The filter is

- applicable to new financial instruments nearly automatically (*adaptive* algorithm instead of hand-calibration),
- specified in a way suitable to programming (clear concepts, structures, hierarchies),
- profiting from 12 years of Olsen filtering experiences and a rich set of error examples.

Filtering of high-frequency time-series data is a demanding, often underestimated task. It is complicated because of

- the variety of possible errors and their causes;
- the variety of statistical properties of the filtered variables (distribution functions, conditional behavior, non-stationarity and structural breaks);
- the variety of data sources and contributors of different reliability;
- the irregularity of time intervals (sparse/dense data, sometimes long data gaps over time);
- the complexity and variety of the quoted information: transaction prices, indicative prices, FX forward premia (where negative values are allowed), interest rates, prices and other variables from derivative markets, transaction volumes, ... ; bid/ask quotes vs. single-valued quotes;

- the necessity of real-time filtering: producing instant filter results before seeing any successor quote.

There are different possible approaches to filtering. Some guidelines determine the approach of the Olsen filter:

- Plausibility: we do not know the real cause of data errors with rare exceptions (e. g. the decimal error). Therefore we judge the validity or credibility of a quote according to its *plausibility*, given the statistical properties of the series.
- We need a whole neighborhood of quotes for judging the credibility of a quote: a *filtering window*. A comparison to only the "last valid" quote of the series is not enough. The filtering window can grow and shrink with data quality and the requirements for arriving at a good filtering decision.
- The *statistical properties* of the series needed to measure the plausibility of a quote are determined inside the filtering algorithm rather than being hand-configured. The filter is thus *adaptive.*
- Quotes with complex structures (i. e. bid/ask or open/high/low/close) are split into scalar variables to be filtered separately. These filtered variables may be derived from the raw variables, e. g. the logarithm of a bid price or the bid-ask spread. Quote splitting is motivated by keeping the algorithm modular and overseeable. Some special error types may also be analyzed for full quotes before splitting.
- Numerical methods with convergence problems (such as non-linear minimization) are not used. Such methods would probably lead to problems as the filter is exposed to very different situations. The chosen algorithm produces unambiguous results.
- The filter needs a high execution speed; computing all filtering results from scratch with every new quote would not be efficient. The chosen algorithm is *iterative*: when a new quote is considered, the filtering information obtained from the previous quotes is re-used; only a minimal number of computations concerning the new quote is added.
- The filter has two modes: *real-time* and *historical.* Thanks to the filtering window technique, both modes can be supported by the same filter run. In historical filtering, the final validation of a quote is delayed to a time after having seen some successor quotes.

The paper is organized as follows. In section 2, the properties and typical errors of the filtered data is discussed. Section 3 provides an overview of the filter, its application and its hierarchical structure. Some basic concepts and operations are explained in section 4. At the lowest hierarchy level, scalar filtering windows (section 5) use these basic operations; the next level is the full-quote window (section 6); the highest level is the univariate filter with its operations (section 7). Some readers may prefer a top-down approach and therefore read these sections in reverse sequence: section 7, 6, ... Some filtering elements dealing with special data problems are discussed in the separate section 8, but they nicely fit into the main filter structure. An adaptive filter needs to learn from the data before being operational; this is organized by following the initialization and checkpointing guidelines given in section 9. Section 10 covers some further aspects of filtering such as testing. The filter has many parameters, so the list of parameters given in section 11 will be helpful. A conclusion is made in section 12.

2 The data to be filtered

Before discussing the filtering algorithm, we have to specify the object to be filtered: a time series of quotes. Here we regard only one time series at a time. The filtering of several time series together

is discussed in section 8.4. A time series is a time-ordered sequence of quotes for a given financial instrument. This data should be homogeneous as explained in the following section.

2.1 Data homogeneity: a requirement for filtering

Homogeneity means that the quotes of a series are of the same type and the same market; they may differ in the origins of the contributors, but should not differ in important parameters such as the maturity (of interest rates, ...) or the moneyness (of options or their implied volatilities).

The filter user is responsible for ensuring data homogeneity. There are special cases that pose a challenge to data homogeneity. The LIFFE futures exchange, for example, sometimes produces quotes that are intended as corrections of old quotes. In this case, the intention is not only to use this new quote, but also to erase an old quote that is replaced be the new one. The correction message may arrive quite late and thus replace an old quote in the more distant past rather than the most recent one.

Another case of inhomogeneous data occurs in data feeds that provide bid or ask quotes (or transaction quotes) alternatively in random sequence. Here we advise splitting the data stream into independent bid and ask streams. The same feed may also contains volume or open interest figures, which can also be filtered as special time series. Normal bid-ask pairs, however, are appropriately handled inside the filter.

Another violation of homogeneity happens in the case of benchmark bonds, when the maturity of the underlying bond changes. Strictly speaking, a benchmark bond is no longer the same instrument after this change, but the convention is to keep it in the same time series. This case is discussed in section 8.3.2.

2.2 The nature of the data: bid, ask, transaction quotes, ...

All quotes have the following general structure (which is in line with the "datum" structure of Olsen's ORLA software, Olsen Research LAboratory).

1. A *time stamp.* This is the time of the data collection, the arrival time of the real-time quote in the collector's environment. Time stamps are monotonically increasing over the time series. We might also have other time stamps (e. g. the reported time of the original production of the quote). Such a secondary time stamp would however be considered as side information (in category 4) rather than a primary time stamp.

2. Information on the quote *level.* There are different types of level information as markets and sources are different in nature and also differently organized. Some level information can be termed "price", some other information such as transaction volume figures cannot. Some non-prices such as implied volatility quotes can be treated as prices with bid and ask quotes. A neutral term such as "level" or "filtered variable" is therefore preferred to "price". In the case of options, the price might be first converted to an implied volatility which is then the filtered variable. Different quoting types require different filtering approaches; this is discussed below.

3. Information on the *origin* of the quote: information provider, name of exchange or bank, city, country, time zone, ... In the filtering algorithm, we only need one function to *compare* two origins. This will be used when judging the independence and credibility of quotes as explained in further sections. A further analysis of bank names or other IDs is not really needed.

4. Side information: everything that does not fall into one of the three aforementioned categories, e. g. a second time stamp. This is ignored by filtering.

This list summarizes the general data organization from the perspective of the filtering requirements.

The information on the quote levels is organized in different structures depending on the market and the source. Some important cases are listed here:

- *single-valued quotes*: each quote has only one value describing its level. Example: stock indices.
- *bid-ask quotes*: each quote has a bid value *and* an ask value. Example: foreign exchange (FX) spot rates.
- *bid or ask quotes*: each quote has a bid *or* an ask value, often in unpredictable sequence. This can be regarded as two different single-valued time series. Example: quotes on some exchanges.
- *bid or ask or transaction quotes*: each quote has a bid value *or* an ask value *or* a transaction value. Again, this can be regarded as three different single-valued time series. Example: the data stream from the major short-term interest rate futures exchanges also includes transaction data.
- *middle* quotes: in certain cases, we only obtain a time series of middle quotes with are treated as single-valued quotes. The case of getting only *transaction* quotes (no bid, no ask) is technically identical. Also transaction volume figures are treated as single-valued quotes, for example.
- *OHLC* quotes: open/high/low/close. There is no Olsen filter for OHLC quotes as there is no such input data stream in Olsen's data collection. However, the filter architecture is flexible: a special OHLC filter can be made in analogy to the bid-ask filter, also with some tests of the whole quote followed by quote splitting as to be explained.
- *other quote types*: we can never be sure to have a comprehensive list of level quoting types.

The chosen filter structure is flexible enough to treat all these different quote structures. If a new, unexpected quote structure is encountered, it is straightforward to derive a new class for it, derived from the base class of univariate filters (in the terminology of object-oriented programming).

2.3 Data error types

Our definition of the term *data error* is as follows.

> *A data error is present if a piece of quoted data does not conform to the real situation of the market.*

We have to identify a price quote as being a data error if it is neither a correctly reported transaction price nor a possible transaction price at the reported time. In the case of indicative prices, we have to tolerate a certain transmission time delay, though.

There are many causes for data errors. The errors can vaguely be separated in two classes:

1. human errors: errors directly caused by human data contributors, for different reasons:
   (a) unintentional errors, e. g. typing errors;
   (b) intentional errors, e. g. dummy quotes produced just for technical testing;
2. system errors: errors caused by computer systems, their interactions and their failures.

Strictly speaking, system errors are also human errors because human operators have the ultimate responsibility for the correct operation of computer systems. However, the distance between the data error and the responsible person is much larger for system errors.

In many cases, it is impossible to find the exact reason for the data error even if the quote is very aberrant. The task of the filter is to identify such outliers, whatever the reason.

Sometimes the cause of the error can be guessed from the particular behavior of the bad quotes. This knowledge of the error mechanism can help to improve filtering and, in some cases, correct the bad quotes.

The Olsen filter has some algorithmic elements to deal with special errors of different types:

1. **Decimal errors**: Failure to change a "big" decimal digit of the quote. Example: a bid price of 1.3498 is followed by a true quote 1.3505, but the published, bad quote is 1.3405. This error is most damaging if the quoting software is using a *cache* memory somewhere. The wrong decimal digit may stay in the cache and cause a long series of bad quotes. For Reuters page data, this was a dominant error type around 1988! Nowadays, this error type seems to be rare.

2. **"Test" quotes**: Some data contributors sometimes send test quotes to the system, usually at times when the market is not liquid. These test quotes can cause a lot of damage because they may look plausible to the filter, at least initially. Two important examples:

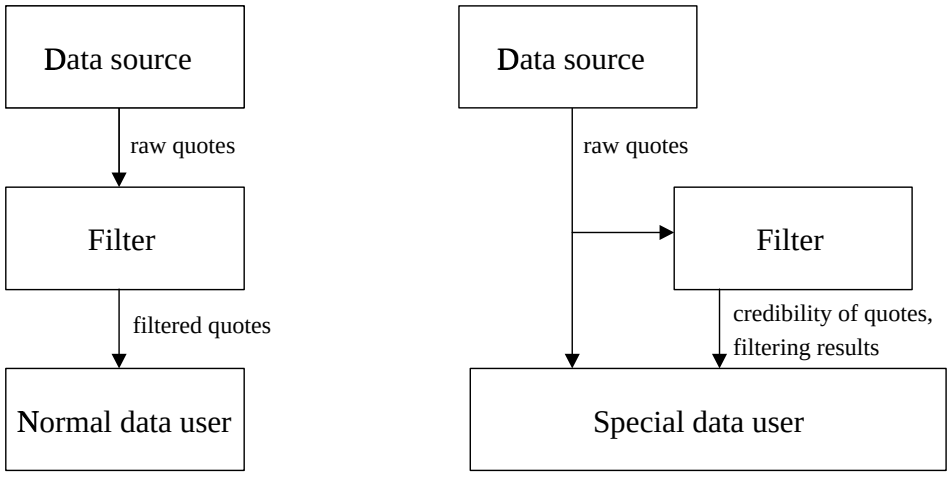
   - **"Early morning test"**: A contributor sends a bad quote very early in the morning, in order to test whether the connection to the data distributor (e. g. Reuters) is operational. If the market is inactive overnight, no trader would take this test quote seriously. For the filter, such a quote may be a major challenge. The filter has to be very critical to first quotes after a data gap.
   - **Monotonic series**: Some contributors test the performance and the time delay of their data connection by sending a long series of linearly increasing quotes at inactive times such as overnight or during a weekend. For the filter, this is hard to detect because quote-to-quote changes look plausible. Only the monotonic behavior in the long run can be used to identify the fake nature of this data.

3. **Repeated quotes**: Some contributors let their computers repeat the last quote in more or less regular time intervals. This is harmless if it happens in a moderate way. In some markets with high granularity of quoting (such as Eurofutures), repeated quote values are quite natural. However, there are contributors that repeat old quotes thousands of times with high frequency, thereby obstructing the filtering of the few good quotes produced by other, more reasonable contributors.

4. **Quote copying**: Some contributors employ computers to copy and re-send the quotes of other contributors, just to show a strong presence on the data feed. Thus they decrease the data quality, but there is no reason for a filter to remove copied quotes that are on a correct level. Some contributors run programs to produce slightly modified copied quotes by adding a small random correction to the quote. Such slightly varying copied quotes are damaging because they obstruct the clear identification of fake monotonic or repeated series made by other contributors.

5. **Scaling problem**: Quoting conventions may differ or be officially redefined in some markets. Some contributors may quote the value of 100 units, others the value of 1 unit. The filter may run into this problem "by surprise" unless a very active filter user anticipates all scale changes in advance and preprocesses the data accordingly.

The filter has means to treat all these different error types, as explained in further sections, notably section 8.



Normal users just want to eliminate bad quotes from the application (left chart). In special cases, users want to know filtering results such as the credibility or the reason for filtering a quote (right chart). In both cases, the filter may compute corrected quote values if the reason of the quote error is well known.

Figure 1: Flow charts of the filter application

3 General overview of the filter

Before explaining the many details of the filter, some overview is given. The functionality of the filter from the user's perspective as well as the basic internal structure is described.

3.1 The functionality of the filter

The functionality of the filter is first described on a general level. The flowcharts of Figure 1 show some typical applications of the filter in a larger context. Normal users just want to eliminate "invalid" data from the stream, but the chart on the right-hand side shows that the filter can also deliver more information on the quotes and their quality.

The filter has some configuration parameters depending on the type of the instrument, as to be shown later. Once it is created, it performs the following operations:

- It is fed by financial quotes in the ordered sequence of their time stamps.
- It delivers the filtering results of the same quotes in the same ordered sequence; for each quote:
  - the credibility of the quote between 0 (totally invalid) and 1 (totally valid), also for individual elements such as bid or ask prices or the bid-ask spread;
  - the value(s) of the quote, whose errors can possibly be *corrected* in some cases where the error mechanism is well known;
  - the filtering reason, explaining why the filter has rejected (or corrected) the quote.

Special filter users as on the right-hand side of Figure 1 may want to use all these filtering results, e. g. for filter testing purposes. Normal users use only those (possibly corrected) quotes with a credibility exceeding a threshold value (which is often chosen to be 0.5). They ignore all invalid quotes and all side

results of the filter such as the filtering reason. By doing so, they are processing the data as in the left flowchart of Figure 1.

The timing of the filter operations is non-trivial. In *real-time* operation, a per-quote result is produced right after the corresponding quote has entered the filter. In *historical* operation, the user can see a per-quote result only after the filter has seen a few newer quotes and adapted the credibility of older quotes accordingly.

The filter needs a *build-up period* as specified by section 9.1. This is natural for an adaptive filter. If the filtering session starts at the first available quote (database start), the build-up means to run the filter for a few weeks from this start, storing a set of statistical variables in preparation for restarting the filter from the first available quote. The filter will then be well adapted because it can use the previously stored statistical variables. If the filtering session starts at some later point in the time series, the natural build-up period is the period immediately preceding the first quote of the session.

The described functionality refers to the univariate Olsen filter, where a financial instrument is filtered independently from other instruments. This concept has to be upgraded in the case of a multivariate filtering algorithm yet to be developed, see section 8.4.

The filtering algorithm can be seen as one whole block that can be used several times in a data flow, also in series. Examples:

- Mixing already filtered data streams from several sources where the mixing result is again filtered. The danger is that the combined filters reject too many quotes, especially in the real-time filtering of fast moves (or price jumps).
- Filtering combined with computational blocks: raw data $\rightarrow$ filter $\rightarrow$ computational block $\rightarrow$ filter $\rightarrow$ application. Some computational blocks such as cross rate or yield curve computations require filtered input and produce an output that the user may again want to filter.

Repeated filtering in series is rather dangerous because it may lead to too many rejections of quotes. If it cannot be avoided, only one of the filters in the chain should be of the standard type. The other filter(s) should be configured to be weak, i. e. they should eliminate not more than the totally aberrant outliers.

3.2 Overview of the filtering algorithm and its structure

The filtering algorithm is structured in a hierarchical scheme of sub-algorithms. Table 1 gives an overview of this structure. The Olsen filter is *univariate*, it treats only one financial instrument at a time. Of course, we can create many filter objects for many instruments, but these filters do not interact with each other.

However, we can add a higher hierarchy level at the top of Table 1 for multivariate filtering. A multivariate filter could coordinate several univariate filters and enhance the filtering of sparse time series by using information from well-covered instruments. This is discussed in section 8.4.

Details of the different algorithmic levels are explained in the next sections. The sequence of these sections follows Table 1, *from bottom to top.*

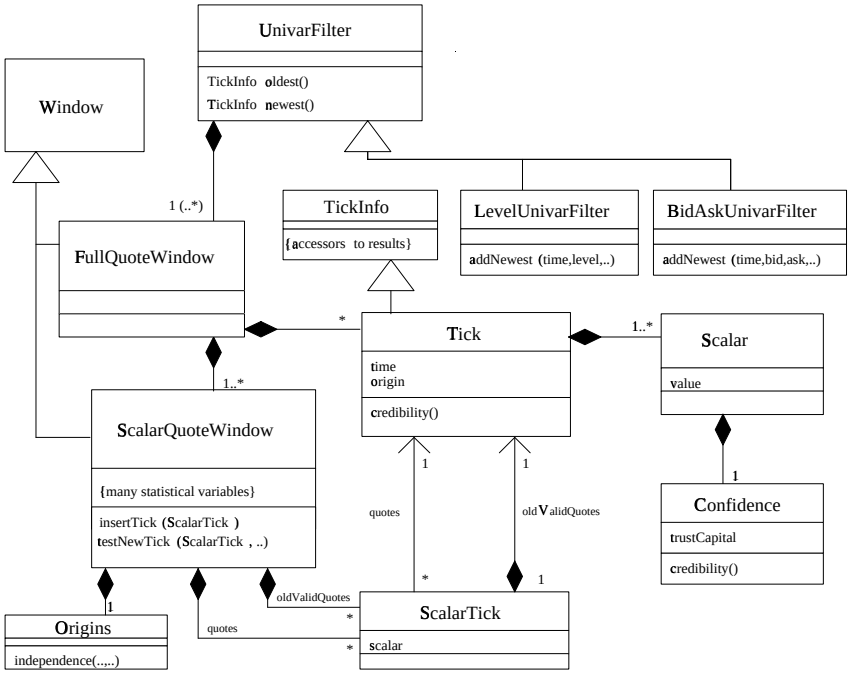
In Figure 2, the structure of the filter is also shown in the form of a UML class diagram. UML diagrams (the standard in object-oriented software development) are explained in [Fowler and Scott, 1997], for example. The same filter technology might also be implemented slightly different from Figure 2.

The three hierarchy levels of Table 1 can be found again in Figure 2: (1) the univariate filter (UnivarFilter), (2) the full-quote filtering window (FullQuoteWindow) and (3) the scalar filtering window (ScalarQuoteWindow). Note that the word “tick” is just a synonym of the term “quote”. The filter as

hier-archy level	name of the level	purpose, description
1	univariate filter	The complete filtering of one time series: • passing incoming quotes to the analysis of the lower hierarchy levels; • managing the filter results of the lower hierarchy levels and packaging these results into the right output format of the filter; • supporting real-time and historical filtering; • supporting one or more filtering hypotheses, each with its own full-quote filtering window.
2	full-quote filtering window	A sequence of recent full quotes, some of them possibly corrected according to a general filtering hypothesis. Tasks: • *quote splitting* (the most impostant task): splitting full quotes (such as bid/ask) into scalar quotes to be filtered individually in their own scalar filtering windows; • a basic validity test (e. g. whether prices are in the positive domain); • a possible mathematical transformation (e. g.: logarithm); • all those filtering steps that require full quotes (not just bid or ask quotes alone) are done here.
3	scalar filtering window	A sequence of recent scalar quotes whose credibilities are still in the process of being modified. Tasks: • testing new, incoming scalar quotes; • comparing a new scalar quote to all older quotes of the window (using a special business time scale and a dependence analysis of quote origins); • computing a first (real-time) credibility of the new scalar quote; • modifying the credibilities of older quotes based on the information gained from the new quote; • dismissing the oldest scalar quote when its credibility is finally settled; • updating the statistics with sufficiently credible scalar quotes when they are dismissed from the window.

Table 1: The basic structure of the filtering algorithm.

explained in the whole document is much richer than the class structure shown in Figure 2. The filter also contains some special elements such as a filters for monotonic fake quotes or scaled quotes. The description of these special filter elements has been moved to the separate section 8, following the main filter description. However, everything fits into the main structure given by Table 1 and Figure 2. We recommend that the reader repeatedly consult this table and this figure in order to regain an overview of the whole algorithm while reading the next sections.



A possible UML diagram of the filter implementation. The main classes and relations are shown; many more elements have to be locally added according to the descriptions of the document.

Figure 2: UML class diagram of the filter

4 Basic filtering elements and operations

The first element to be discussed in a bottom-to-top specification is the scalar filtering window. Its position in the algorithm is shown in Figure 2 (class ScalarQuoteWindow). Window filtering relies on a number of concepts and operations that are presented even before discussing the management of the window.

The basic filtering operations see the quotes in the simplified form of *scalar quotes* consisting of:

1. the time stamp,
2. one scalar variable value to be filtered (e. g. the logarithm of a bid price), here denoted by x,
3. the origin of the quote (as in the full quote of section 2.2).

The basic operations can be divided into two types:

1. Filtering of single scalar quotes: considering the credibility of one scalar quote alone. An important part is the *level filter* where the level of the filtered variable is the criterion.
2. Pair filtering: comparing two scalar quotes. The most important part is the *change filter* that considers the change of the filtered variable from one quote to another one. Filtering depends on the time interval between the two quotes and the time scale on which this is measured. Pair filtering also includes a comparison of quote origins.

The basic filtering operations and another basic concept of filtering, credibility, are presented in the following sections. Their actual application in the larger algorithm is explained later, starting from section 5.

C_{total}	C_1 =				
	0	0.25	0.5	0.75	1
C_2 =					
1	(0.5)	1	1	1	1
0.75	0	0.5	0.75	0.878	1
0.5	0	0.25	0.5	0.75	1
0.25	0	0.122	0.25	0.5	1
0	0	0	0	0	(0.5)

Table 2: The total credibility C_{total} resulting from two independent credibility values C_1 and C_2. The function $C_{\text{total}} = C[T(C_1) + T(C_2)]$ defines an addition operator for credibilities. Eqs. 4.1 and 4.2 are applied. The values in brackets, (0.5), are in fact indefinite limit values; C_{total} may converge to any value between 0 and 1.

4.1 Credibility and trust capital

Credibility is a central concept of the filtering algorithm. It is expressed by a variable C taking values between 0 and 1, where 1 indicates certain validity and 0 certain invalidity. This number can be interpreted as the probability of a quote being valid according to a certain arbitrary criterion. For two reasons, we avoid the formal introduction of the term "probability". First, the validity of a quote is a fuzzy concept; e. g. slightly deviating quotes of an over-the-counter spot market can perhaps be termed valid even if they are very unlikely to lead to a real transaction. Second, we have no model of probability even if validity could be exactly defined. Credibility can be understood as a "possibility" in the sense of fuzzy logic [Zimmermann, 1985].

Credibility is not additive: the credibility of a scalar quote gained from two tests is not the sum of the credibilities gained from the individual tests. This follows from the definition of credibility between 0 and 1. The sum of two credibilities of, say, 0.75 would be outside the allowed domain.

For internal credibility computations, it is useful to define an additive variable, the *trust capital* T which is unlimited in value. There is no theoretical limit for gathering evidence in favor of accepting or rejecting the validity hypothesis. Full validity corresponds to a trust capital of $T = \infty$, full invalidity to $T = -\infty$. We impose a fixed, monotonic relation between the credibility C and the trust capital T of a certain object:

$$C(T) = \frac{1}{2} + \frac{T}{2\sqrt{1+T^2}} \tag{4.1}$$

and the inverse relation

$$T(C) = \frac{C - \frac{1}{2}}{\sqrt{C\,(1-C)}} \tag{4.2}$$

There are possible alternatives to this functional relationship. The chosen solution has some advantages in the formulation of the algorithm that will be shown later.

The additivity of trust capitals and eqs. 4.1 and 4.2 imply the definition of an addition operator for credibilities. Table 2 shows the total credibility resulting from two independent credibility values.

4.2 Filtering of single scalar quotes: the level filter

In the Olsen filter, there is only one analysis of a single quote, the level filter. Comparisons between quotes (done for a pair of quotes, treated in section 4.3) are often more important in filtering than the analysis of a single quote.

The level filter computes a first credibility of the value of the filtered variable. This is useful only for those volatile but mean-reverting time series where the levels as such have a certain credibility in the absolute sense – not only the level *changes*. Moreover, the timing of the mean reversion should be relatively fast. Interest rates or IR futures prices, for example, are mean-reverting only after time intervals of years; they appear to be freely floating within smaller intervals [Ballocchi, 1996]. For those rates and for other prices, level filtering is not suitable.

The obvious example for fast mean reversion and thus for using a level filter is the bid-ask spread which can be rather volatile from quote to quote but tends to stay within a fixed range of values that varies only very slowly over time. For spreads, an adaptive level filter is at least as important as a pair filter that considers the spread change between two quotes.

The level filter first puts the filtered variable value x (possibly transformed as described in section 6.3) into the perspective of its own statistical mean and standard deviation. Following the notation of [Zumbach and Müller, 2001], the standardized variable $\hat{x}$ is defined:

$$\hat{x} = \frac{x - \bar{x}}{\mathrm{MSD}[\,\Delta\vartheta_r, 2;\, x\,]} = \tag{4.3}$$

$$= \frac{x - \bar{x}}{\sqrt{\mathrm{EMA}[\,\Delta\vartheta_r;\, (x-\bar{x})^2\,]}}$$

where the mean value of x is also a moving average:

$$\bar{x} = \mathrm{EMA}[\,\Delta\vartheta_r;\, x\,] \tag{4.4}$$

The ϑ-time scale [Dacorogna et al., 1993] to be used is discussed in section 4.4. The variable $\Delta\vartheta_r$ denotes the configurable range of the kernel of the moving averages and should cover the time frame of the mean reversion of the filtered variable; a reasonable value for bid-ask spreads has to be chosen. The iterative computation of moving averages is explained in [Müller, 1991] and [Zumbach and Müller, 2001]. Here and for all the moving averages of the filtering algorithm, a simple exponentially weighted moving average (EMA) is used by the Olsen filter for efficiency reasons.

A small $|\hat{x}|$ value deserves high trust; an extreme $|\hat{x}|$ value indicates an outlier with low credibility and negative trust capital. Before arriving at a formula for the trust capital as a function of $\hat{x}$, the *distribution function* of $\hat{x}$ has to be discussed. A symmetric form of the distribution function is assumed at least in coarse approximation. Positive definite variables such as the bid-ask spread are quite asymmetrically distributed; this is why they must be mathematically transformed. This means that x is already a transformed variable, e. g. the logarithm of the spread as explained in section 6.3.

The amount of negative trust capital for outliers depends on the tails of the distribution at extreme (positive and negative) $\hat{x}$ values. A reasonable assumption is that the credibility of outliers is approximately the probability of exceeding the outlier value, given the distribution function. This probability is proportional to $\hat{x}^{-\alpha}$ where α is called the tail index of the fat-tailed distribution. We know that distribution functions of level-filtered variables such as bid-ask spreads [Müller and Sgier, 1992] are indeed fat-tailed. Determining the distribution function and α in a moving sample would be a considerable task, certainly too heavy for filtering software. Therefore, we choose an approximate assumption on α that was found

acceptable across many rates, filtered variable types and financial instruments: $\alpha = 4$. This value is also used in the analogous pair filtering tool, e. g. for price changes, and discussed in section 4.3.1.

For extreme events, the relation between credibility and trust capital, eq. 4.1, can be asymptotically expanded as follows:

$$C \;=\; \frac{1}{4\,T^2} \quad \text{for } T \;\ll\; -1 \tag{4.5}$$

Terms of order higher than $(1/T)^2$ are neglected here. Defining a credibility proportional to $\hat{x}^{-\alpha}$ is thus identical to defining a trust capital proportional to $\hat{x}^{\alpha/2}$. Assuming $\alpha = 4$, we obtain a trust capital proportional to $\hat{x}^2$. For outliers, this trust capital is negative, but for small $\hat{x}$, the trust capital is positive up to a maximum value we define to be 1.

Now, we have the ingredients to come up with a formula that gives the resulting trust capital of the *i*th quote according to the level filter:

$$T_{i0} \;=\; 1 - \xi_i^2 \tag{4.6}$$

where the index 0 of T_{i0} indicates that this is a result of the level filter only. The variable ξ_i is x in a scaled and standardized form:

$$\xi_i \;=\; \frac{\hat{x}_i}{\xi_0} \tag{4.7}$$

with a constant ξ_0. Eq. 4.6 together with eq. 4.7 is the simplest possible way to obtain the desired maximum and asymptotic behavior. For certain rapidly mean-reverting variables such as hourly or daily trading volumes, this may be enough.

However, the actual implementation in the Olsen filter is for bid-ask spreads which have some special properties. Filter tests have shown that these properties have to be taken into account in order to attain satisfactory spread filter results:

- Quoted bid-ask spreads tend to cluster at "even" values, e. g. 10 basis points, while the real spread may be an odd value oscillating in a range below the quoted value. A series of formal, constant spreads can therefore hide some substantial volatility that is not covered by the statistically determined denominator of eq. 4.3. We need an offset $\Delta x^2_{\min}$ to account for the typical hidden volatility in that denominator. A suitable choice is $\Delta x^2_{\min} = [\text{constant}_1(\bar{x} + \text{constant}_2)]^2$.
- High values of bid-ask spreads are worse in usability and plausibility than low spreads, by nature. Thus the quote deviations from the mean as defined by eq. 4.3 are judged in a biased way. Deviations to the high side ($\hat{x}_i > 0$) are penalized by a factor p_{high} whereas no such penalty is applied against low spreads.
- For some (minor) financial instruments, many quotes are posted with zero spreads, i. e. bid quote = ask quote. This is discussed in section 6.1 (and its subsections). In some cases, zero spreads have to be accepted, but we set a penalty against them as in the case of positive $\hat{x}_i$.

We obtain the following refined definition of ξ_i:

$$\xi_i \;=\; \begin{cases} \frac{\hat{x}_i}{\xi_0} & \text{if } \hat{x}_i \;\leq\; 0 \text{ and no zero-spread case} \\ p_{high}\,\frac{\hat{x}_i}{\xi_0} & \text{if } \hat{x}_i \;>\; 0 \text{ or in a zero-spread case} \end{cases} \tag{4.8}$$

where $\hat{x}_i$ comes from a modified version of eq. 4.3,

$$\hat{x} \;=\; \frac{x-\bar{x}}{\sqrt{\mathrm{EMA}[\,\Delta\vartheta_r;\;(x-\bar{x})^2\,]+\Delta x^2_{\min}}} \tag{4.9}$$

The constant ξ_0 determines the size of an $\hat{x}$ that is just large enough to neither increase nor decrease the credibility.

Eq. 4.8 is general enough for all mean-reverting filterable variables. The Olsen filter has a level filter only for bid-ask spreads. If we introduced other mean-reverting variables, a good value for $\Delta x^2_{\min}$ would probably be much smaller or even 0, p_{high} around one and ξ_0 larger (to tolerate volatility increases in absence of a basic volatility level $\Delta x^2_{\min}$).

4.3 Pair filtering

The pairwise comparison of scalar quotes is a central basic filtering operation. Simple filtering algorithms, such as the old Olsen filter as it was around 1990, indeed consisted of a simple sequence of pair filtering operations: each new quote was judged only in relation to the "last valid" quote. The current Olsen filter makes more pairwise comparisons also for quotes that are not neighbors in the series as explained in section 5.

Pair filtering contains several ingredients, the most important one being the filter of variable changes. The time difference between the two quotes plays a role, so the time scale on which it is measured has to be specified. The criterion is *adaptive* to the statistically expected volatility estimate and therefore uses some results from a moving statistical analysis.

Another basic pair filtering operation is the comparison of the origins of the two quotes. Some data sources provide rich information about contributors, some other sources hide this information or have few contributors (or just one). The comparison of quote origins has to be seen in the perspective of the observed diversity of quote origins. Measuring this diversity (which may change over time) adds another aspect of adaptivity to the filter.

4.3.1 The change filter

The change filter is a very important filtering element. Its task is to judge the credibility of a variable change according to experience, which implies the use of on-line statistics and thus adaptivity. The change of the filtered variable from the jth to the ith quote is

$$\Delta x_{ij} \;=\; x_i - x_j \tag{4.10}$$

The variable x may be the result of a transformation in the sense of section 6.3. The time difference of the quotes is $\Delta\vartheta_{ij}$, measured on a time scale to be discussed in section 4.4.

The variable change Δx is put into a relation to a volatility measure: the expected variance $V(\Delta\vartheta)$ about zero. V is determined by the on-line statistics as described in section 4.3.2. The relative change is defined as follows:

$$\xi_{ij} \;=\; \frac{\Delta x_{ij}}{\xi_0\,\sqrt{V(\Delta\vartheta_{ij})}} \tag{4.11}$$

with a positive constant ξ_0 which has a value of 5.5 in the Olsen filter and is further discussed below. Low $|\xi|$ values deserve high trust, extreme $|\xi|$ values indicate low credibility and negative trust capital: at least one of the two compared quotes must be an outlier.

The further algorithm is similar to that of the level filter as described in section 4.2, using the relative change ξ_{ij} instead of the scaled standardized variable ξ_i.

The amount of negative trust capital for outliers depends on the distribution function of changes Δx, especially the tail of the distribution at extreme Δx or ξ values. A reasonable assumption is that the credibility of outliers is approximately the probability of exceeding the outlier value, given the distribution function. This probability is proportional to $\xi^{-\alpha}$ where α is the tail index of a fat-tailed distribution. We know that distribution functions of high-frequency price changes are indeed fat-tailed [Dacorogna et al., 1998]. Determining the distribution function and α in a moving sample would be a considerable task beyond the scope of filtering software. Therefore, we make a rough assumption on α that is good enough across many rates, filtered variable types and financial instruments. For many price changes, a good value is around $\alpha \approx 3.5$, according to [Dacorogna et al., 1998, Müller et al., 1998]. As in section 4.2, we generally use $\alpha = 4$ as a realistic, general approximation.

As in section 4.2 and with the help of eq. 4.5, we argue that the trust capital should asymptotically be proportional to ξ^2 and arrive at a formula that gives the trust capital as a function of ξ:

$$U_{ij} \;=\; U(\xi_{ij}^2) \;=\; 1 - \xi_{ij}^2 \tag{4.12}$$

which is analogous to eq. 4.6. This trust capital depending only on ξ is called U to distinguish it from the final trust capital T that is based on more criteria. At $\xi = 1$, eq. 4.12 yields a zero trust capital, neither increasing nor decreasing the credibility. Intuitively, a variable change of few standard deviations might correspond to this undecided situation; smaller variable changes lead to positive trust capital, larger ones to negative trust capital. In fact, the parameter ξ_0 of eq. 4.11 should be configured to a high value, leading to a rather tolerant behavior even if the volatility V is slightly underestimated.

The trust capital U_{ij} from eq. 4.12 is a sufficient concept under the best circumstances: independent quotes separated by a small time interval. In the general case, a modified formula is needed to solve the following three special pair filtering problems.

1. Filtering should stay a local concept on the time axis. However, a quote has few close neighbors and many more distant neighbors. When the additive trust capital of a quote is determined by pairwise comparisons to other quotes as explained in section 5.2.1, the results from distant quotes must not dominate those from the close neighbors; the interaction range should be limited. This is achieved by defining the trust capital proportional to $(\Delta\vartheta)^{-3}$ (assuming a constant ξ) for asymptotically large quote intervals $\Delta\vartheta$.

2. For large $\Delta\vartheta$, even moderately aberrant quotes would be too easily accepted by eq. 4.12. Therefore, the aforementioned decline of trust capital with growing $\Delta\vartheta$ is particularly important in the case of positive trust capital. Negative trust capital, on the other hand, should stay strongly negative even if $\Delta\vartheta$ is rather large. The new filter needs a selective decline of trust capital with increasing $\Delta\vartheta$: fast for small ξ (positive trust capital), slow for large ξ (negative trust capital). This treatment is essential for *data holes* or gaps, where there are no (or few) close neighbor quotes.

3. Dependent quotes: if two quotes originate from the same source, their comparison can hardly increase the credibility (but it can reinforce negative trust in the case of a large ξ). In section 4.3.3, we introduce an independence variable I_{ij} between 0 (totally dependent) and 1 (totally independent).

The two last points imply a certain asymmetry in the trust capital: gathering evidence in favor of accepting a quote is more delicate than evidence in favor of rejecting it.

T	$d\Delta\vartheta/\nu$ =				
	0	0.5	1	2	4
$\lvert\xi\rvert$ =					
4	-15.0	-14.9	-14.2	-10.2	-3.2
2	-3.0	-2.9	-2.5	-1.2	-0.22
1	0	0	0	0	0
0.5	0.75	0.68	0.42	0.10	0.014
0	1	0.89	0.50	0.11	0.015

Table 3: The trust capital T resulting from a comparison of two independent ($I^\star = 1$) scalar quotes, depending on the relative variable change ξ and the time interval $\Delta\vartheta$ between the quotes. ξ is defined by eq. 4.11, and d and ν are explained in the text.

All these concerns can be taken into account in an extended version of eq. 4.12. This is the final formula for the trust capital from a change filter:

$$T_{ij} \;=\; T(\xi_{ij}^2, \Delta\vartheta_{ij}, I_{ij}) \;=\; I_{ij}^\star \,\frac{1-\xi_{ij}^4}{1+\xi_{ij}^2+(\frac{d\;\Delta\vartheta_{ij}}{\nu})^3} \tag{4.13}$$

where

$$I_{ij}^\star \;=\; \begin{cases} I_{ij} & \text{if } \xi_{ij}^2 < 1 \\ 1 & \text{if } \xi_{ij}^2 \geq 1 \end{cases} \tag{4.14}$$

The independence I_{ij} is always between 0 and 1 and is computed by eq. 4.23. The variable d is a quote density explained in section 4.3.2. The configurable constant ν determines a sort of filtering interaction range in units of the typical quote interval ($\approx 1/d$).

Table 3 shows the behavior of the trust capital according to eq. 4.13. The trust capital converges to zero with an increasing quote interval $\Delta\vartheta$ much more rapidly for small variable changes $\lvert\xi\rvert$ than for large ones. For small $\Delta\vartheta_{ij}$ and $I_{ij} = 1$, eq. 4.13 converges to eq. 4.12.

4.3.2 Computing the expected volatility

The expected volatility is a function of the size of the time interval between the quotes and thus requires a larger computation effort than other statistical variables. Only credible scalar quotes should be used in the computation. The updates of all statistics are therefore managed by another part of the algorithm that knows about final credibilities as explained in section 5.4.2.

Choosing an appropriate time scale for measuring the time intervals between quotes is also important. A scale like ϑ-time [Dacorogna et al., 1993] is good because it leads to reasonable volatility estimates without seasonal disturbances. This is further discussed in section 4.4.

The expected volatility computation can be implemented with more or less sophistication. Here, a rather simple solution is taken. The first required statistical variable is the quote density:

$$d \;=\; \mathrm{EMA}[\,\Delta\vartheta_r;\, \frac{c_d}{\delta\vartheta}\,] \tag{4.15}$$

This is a moving average in the notation of [Zumbach and Müller, 2001]; $\delta\vartheta$ is the time interval between two "valid" (as defined on a higher level) neighbor quotes on the chosen time scale, which is ϑ-time, as in all these computations. $\Delta\vartheta_r$ is the configurable range of the kernel of the moving average. The variable c_d is the weight of the quote which has a value of $c_d = 1$ or lower in case of repeated quote values. The iterative computation of moving averages is explained in [Müller, 1991] and [Zumbach and Müller, 2001]. The value $1/\delta\vartheta$ has to be assumed for the whole quote interval which implies using the "next point" interpolation as explained by the same documents. It can be shown that a zero value of $\delta\vartheta$ does not lead to a singularity of the EMA (but must be handled correctly in a software program).

An annualized squared "micro"-volatility is defined as a variance, again in form of a moving average:

$$v \;=\; \mathrm{EMA}[\,\Delta\vartheta_r;\; \frac{(\delta x)^2}{\delta\vartheta + \delta\vartheta_0}\,] \tag{4.16}$$

where the notation (also the δ operator) is again defined by [Zumbach and Müller, 2001] and the range $\Delta\vartheta_r$ is the same as in eq. 4.15. δx is the change of the filtered variable between (sufficiently credible) neighbor quotes. There is a small time interval offset

$$\delta\vartheta_0 \;=\; \max(\,\frac{d_0}{d},\, \delta\vartheta_{\min}\,) \tag{4.17}$$

The small positive term $\delta\vartheta_0$ accounts for some known short-term behaviors of markets: (1) certain asynchronicities in the quote transmissions, (2) some temporary market level inconsistencies that need time to be arbitraged out, (3) a negative autocorrelation of many market prices over short time lags [Guillaume et al., 1997]. However, $\delta\vartheta_0$ is not needed to avoid singularities of v; even a zero value of both $\delta\vartheta$ and $\delta\vartheta_0$ would not lead to a singularity of the EMA. The "next point" interpolation is again appropriate in the EMA computation.

Strictly speaking, v can be called annualized only if ϑ is measured in years, but the choice of this unit does not matter in our algorithm. The exponent of the annualization (here: assuming a Gaussian drift) is not too important because the different values of $\delta\vartheta$ share the same order of magnitude.

Experience shows that the volatility measure of the filter should not only rely on one variance v as defined above. In the Olsen filter, we use three such volatilities: v_{fast}, v and v_{slow}. All of them are computed by eq. 4.16, but they differ in their ranges $\Delta\vartheta_r$: v_{fast} has a short range, v a medium-sized range and v_{slow} a long range. The expected volatility is assumed to be the *maximum* of the three:

$$v_{\text{exp}} \;=\; \max(v_{\text{fast}}, v, v_{\text{slow}}) \tag{4.18}$$

This is superior to taking only v. In case of a market shock, the rapid growth of v_{fast} allows for a quick adaptation of the filter, whereas the inertia of v_{slow} prevents the filter from forgetting volatile events too rapidly in a quiet market phase.

From the annualized v_{exp}, we obtain the expected squared change as a function of the time interval $\Delta\vartheta$ between two quotes. At this point, the Olsen filter has a special element to prevent the filter from easily accepting price changes over large data gaps, time periods with no quotes. Data gaps are characterized by a large value of $\Delta\vartheta$ but very few quotes within this interval. In case of data gaps, an upper limit of $\Delta\vartheta$ is enforced:

$$\Delta\vartheta_{\text{corr}} \;=\; \min[\,\frac{2.5Q}{d},\, \max(\frac{0.1Q}{d}, \Delta\vartheta)\,] \tag{4.19}$$

where d is taken from eq. 4.15 and Q is the number of valid quotes in the interval between the two quotes; this is explained in section 5.2. Eq. 4.19 also sets a lower limit of $\Delta\vartheta_{\text{corr}}$ in case of a very high frequency of valid quotes; this is important to validate fast trends with many quotes.

The corrected quote interval $\Delta\vartheta_{\text{corr}}$ is now used to compute the expected squared change V:

$$V \;=\; V(\Delta\vartheta_{\text{corr}}) \;=\; (\Delta\vartheta_{\text{corr}} + \delta\vartheta_0)\, v_{\text{exp}} + V_0 \tag{4.20}$$

This function $V(\Delta\vartheta_{\text{corr}})$ is needed in the trust capital calculation of section 4.3.1 and inserted in eq. 4.11. The positive offset V_0 is small and could be omitted in many cases with no loss of filter quality. However, a small $V_0 > 0$ is useful. Some quotes are quoted in coarse granularity, i. e. the minimum step between two possible quote values is rather large as compared to the volatility. This is the case in some interest rate futures and also for bid-ask spreads (in FX markets) which often have a rounded size of 5, 10 or 20 basis points with rarely a value in between. Quotes with coarse granularity have a *hidden* volatility: a series of identical quotes may hide a movement of a size smaller than the typical granule. The term V_0 thus represents the hidden volatility:

$$V_0 \;=\; 0.25 g^2 + \varepsilon_0^2 \tag{4.21}$$

where the granule size g is determined by eq. 8.65. The whole granularity analysis is explained in section 8.2 where it plays a more central role. There is yet another term ε_0^2 which is extremely small in normal cases. This ε_0^2 is not related to economics; it has the purely numerical task to keep $V_0 > 0$.

The term ε_0^2 however plays a special role if the scalar variable to be filtered is a (mathematically transformed) bid-ask spread. The spread filter is the least important filter, but leads to the highest number of rejections of FX quotes if it is configured similar to the filter of other scalars. This fact is not accepted by typical filter users: they want a more tolerant spread filter. A closer look shows that different contributors of bid-ask quotes often have different spread quoting policies. They are often interested only in the bid or ask side of the quote and tend to push the other side off the real market by choosing a too large spread. This results in the so-called bid-ask bouncing and in spreads of different sizes between neighbor quotes even in quiet markets. In some minor FX markets, some contributors even mix retail quotes with very large spreads into the stream of interbank quotes. In order not to reject too many quotes for spread reasons, we have to raise the tolerance for fast spread changes and reject only extreme jumps in spreads. This means raising ε_0^2. The Olsen filter has $\varepsilon_0 = \text{constant}_1(\bar{x} + \text{constant}_2)$, where $\bar{x}$ is defined by eq. 4.4. This choice of ε_0 can be understood and appreciated if the mapping of the bid-ask spread, eq. 6.45, is taken into account.

In a filter run starting from scratch, we set $V_0 = \varepsilon_0^2$ and replace this by eq. 4.21 as soon as the granule estimate g is available, based on real statistics from valid quotes (as explained in section 8.2).

4.3.3 Comparing quote origins

Pair filtering results can add some credibility to the two quotes only if these are independent. Two identical quotes from the same contributor do not add a lot of confidence to the quoted level – the fact that an automated quoting system sends the same quote twice does not make this quote more reliable. Two non-identical quotes from the same contributor may imply that the second quote has been produced to correct a bad first one; another interpretation might be that an automated quoting system has a random generator to send a sequence of slightly varying quotes to mark presence on the information system. (This is why a third quote from one contributor in a rapid sequence should be given much less confidence than a second one, but this subtle rule has not yet been implemented). Different quotes from entirely different contributors are the most reliable case for pair filtering.

The basic tool is a function to compare the origins of the two quotes, considering the main source (the information provider), the contributor ID (bank name) and the location information. This implies that available information on contributors has a value in filtering and should be collected rather than ignored.

An "unknown" origin is treated just like another origin name. The resulting independence measure I'_{ij} is confined between 0 for identical origins and 1 for clearly different origins. In some cases (e. g. same bank but different subsidiary), a broken value between 0 and 1 can be chosen.

I'_{ij} is not yet the final result; it has to be put into relation with the general origin *diversity* of the time series. An analysis of data from only one or very few origins must be different from that of data with a rich variety of origins. The general diversity D can be defined as a moving average of the $I'_{i\,i-1}$ of valid neighbor quotes:

$$D \;=\; \mathrm{EMA}[\,\text{tick-time},\, r;\, I'_{i\,i-1}\,] \tag{4.22}$$

where the range r (center of gravity of the kernel) is configured to about 9.5. The "tick-time" is a time scale that is incremented by 1 at each new quote used; the notation of [Zumbach and Müller, 2001] is applied. The "next point" interpolation is again appropriate in the EMA computation. Only "valid" quotes are used; this is possible on a higher level of the algorithm, see section 5.4.2. By doing so, we prevent D from being lowered by bad mass quotes from a single computerized source overnight or weekend. Thus we are protected against a tough filtering problem: the high number of bad mass quotes from a single contributor will not force the filter to accept the bad level.

The use of D makes the independence variable I_{ij} adaptive through the following formula:

$$I_{ij} \;=\; I'_{ij} + f(D)\,(1 - I'_{ij}) \tag{4.23}$$

with

$$f(D) \;=\; \frac{0.0005 + (1-D)^8}{2.001} \tag{4.24}$$

If the diversity is very low (e. g. in a single-contributor source), this formula (reluctantly) raises the independence estimate I_{ij} in order to allow for some positive trust capital to build up. For a strictly uniform source ($I' = D = 0$), I_{ij} will reach 0.5, that is one half of the I_{ij} value of truly independent quotes in a multicontributor series.

The output variable I_{ij} resulting from eq. 4.14 is always confined between 0 and 1 and is generally used in eq. 4.14. Some special cases need a special discussion:

- Repeated quotes. Rarely, the raw data contains long series of repeated quotes from the same contributor, and the obtained value of I_{ij} may still be too high. Solution: in the Olsen filter, this case is handled by a special filtering element described in section 8.2.
- High-quality data. In Olsen's database, we have completed and merged our collected data with some old, historical, commercially available daily data that was of distinctly higher quality than the data from a single, average-quality contributor. When comparing two quotes from this historical daily data, we forced $I'_{ij} = 1$ although these quotes came from the same "contributor". This special filtering element is necessary only if there are huge, proven quality differences between contributors; otherwise we can omit this.
- Only in multivariate filtering (which is not included in the current Olsen filter, see section 8.4): Artificial quotes that might be injected by a multivariate covariance analysis should have $I'_{ij} = 1$ when compared to each other or to any other quote.

market	k	$t_{\text{start},k}$	$t_{\text{end},k}$
(East) Asia	1	21:00	7:00
Europe	2	6:00	16:00
America	3	11:00	21:00

Table 4: Daytimes limiting the active periods of three generic, continent-wide markets; in Greenwich Mean Time (GMT). The scheme is coarse, modeling just the main structure of worldwide financial markets. The active periods differ according to local time zones and business hours; the Asian market already starts on the day before (from the viewpoint of the GMT time zone).

4.4 A time scale for filtering

Time plays a role in the adaptive elements of the level filter as well as in almost all parts of the change filter. Variable changes are tolerated more easily when separated by a large time interval between the time stamps. When using the term "time interval", we need to specify the time scale to be used.

The algorithm works with any time scale, but some are more suitable than others. If we tolerate quote level changes of the same size over weekend hours than over working hours, we have to accept almost any bad quote from the few weekend contributors. These weekend quotes are sometimes test quotes or other outliers in the absence of a liquid market. The same danger arises during holidays (but holidays may be confined to individual countries).

The choice of the time scale is important. Accounting for the low weekend activity is vital, but the exact treatment of typical volatility patterns during working days is less important. Therefore, we cannot accept using only physical time (= calendar/clock time), but the following solutions are possible:

1. A very simple business time with two states: active (working days) and inactive (weekend from Friday 21:00:00 GMT to Sunday 21:00:00 GMT, plus the most important and general holidays); the speed of this business time as compared to physical time would be either 1.4 (in active state) or 0.01 (in inactive state);
2. An adaptively weighted mean of three simple, generic business time scales ϑ: smoothly varying weights according to built-in statistics. This is the solution recommended for a new filter development independent of Olsen's complex ϑ technology.
3. An adaptively weighted mean of three generic business time scales ϑ as defined by [Dacorogna et al., 1993]; This is the solution of the filter running at Olsen, which requires a rather complicated implementation of ϑ-time.

For a new filter implementation outside Olsen, the second solution can be taken. It has a good adaptivity without depending on the complex ϑ technology of Olsen. The second solution differs from the third one only in the definition of the basic ϑ-time scales. Their use and the adaptivity mechanism are the same for both solutions.

Three generic ϑ-times are used, based on typical volatility patterns of three main markets: Asia, Europe and America. In the second solution, these theta times are simply defined as follows:

$$\frac{\mathrm{d}\vartheta_k}{\mathrm{d}t} = \begin{cases} 3.4 & \text{if } t_{\text{start},k} \le t_d < t_{\text{end},k} \text{ on a working day} \\ 0.01 & \text{otherwise (inactive times, weekends, holidays)} \end{cases} \tag{4.25}$$

where t_d is the daytime in Greenwich Mean Time (GMT) and the generic start and end times of the working-daily activity periods are given by Table 4; they correspond to typical observations in several

markets. The active periods of exchange-traded instruments are subsets of the active periods of Table 4. The time scales ϑ_k are time integrals of $d\vartheta_k/dt$ from eq. 4.25. Thus the time ϑ_k flows either rapidly in active market times or very slowly in inactive times; its long-term average speed is similar to physical time. The implementation of eq. 4.25 requires some knowledge on holidays. The database of holidays to be applied may be rudimentary (e. g. just Christmas) or more elaborate to cover all main holidays of the financial centers on the three continents. The effect of daylight saving time is neglected here as the market activity model is anyway coarse.

In Olsen's in-house solution, the three ϑ_k-times are defined according to [Dacorogna et al., 1993]; effects like daylight saving time and local holidays (i. e. characteristic for one continent) are covered. The activity in the morning of the geographical markets is higher than that in the afternoon – a typical behavior of FX rates and, even more so, interest rates, interest rate futures and other exchange-traded markets. In both solutions, the sharply defined opening hours of exchange-traded markets cannot fully be modeled, but the approximation turns out to be satisfactory in filter tests. Olsen is currently testing an improved version of ϑ-time where sudden changes of market activity over daytime can be modeled.

Once the three scales ϑ_k are defined (by the integrals of eq. 4.25 in our suggestion), their adaptively weighted mean is constructed and used as the time scale ϑ for filtering. This ϑ-time is able to approximately capture the daily and weekly seasonality and the holiday lows of volatility. Absolute precision is not required as ϑ is only one among many ingredients of the filtering algorithm, many of which are based on rather coarse approximations. This is the definition of ϑ-time:

$$\vartheta \;=\; \sum_{\text{all } k} w_k \, \vartheta_k \tag{4.26}$$

with

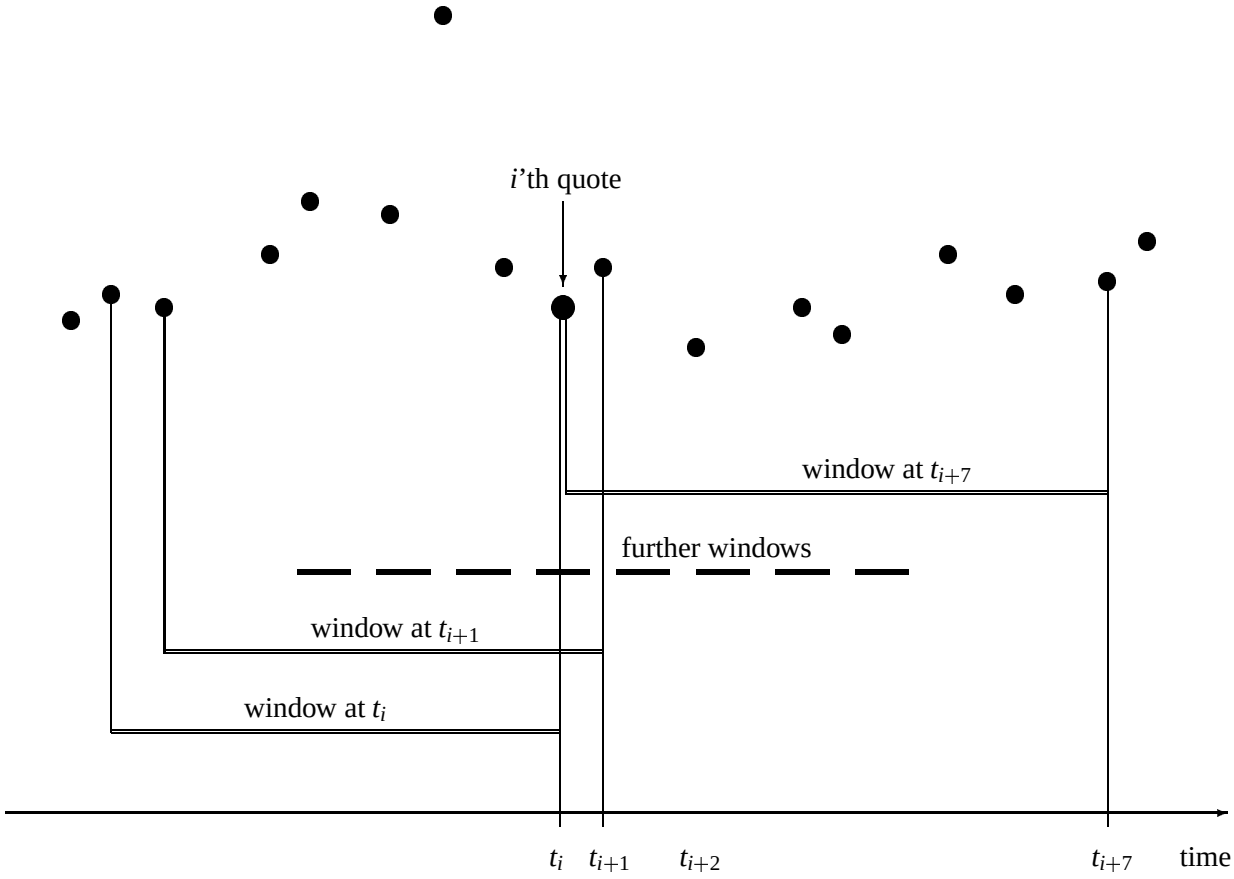
$$\sum_{\text{all } k} w_k \;=\; 1 \tag{4.27}$$

where "all k" means "all markets". This is 3 in our case, but the algorithm also works for any other number of generic markets. The weights w_k are adaptive to the actual behavior of the volatility. A high w_k reflects a high fitness of ϑ_k, which implies that the volatility measured in ϑ_k has low seasonal variations.

The determination of the w_k might be done with complicated methods such as maximum likelihood fitting of a volatility model. However, this would be inappropriate, given the convergence problems of fitting and the anyway existing modeling limitations of eq. 4.26. The heuristic method of the Olsen filter always returns an unambiguous solution. The volatility of changes of the filtered variable is measured on all ϑ_k-scales in terms of a variance similar to eq. 4.16:

$$\sigma_k \;=\; \sqrt{\mathrm{EMA}[\,\Delta\vartheta_{\text{smooth}};\; \frac{(\delta x)^2}{\delta\vartheta_k + \delta\vartheta_0}\,]} \tag{4.28}$$

where $\delta\vartheta_k$ is the interval between validated neighbor quotes in ϑ_k-time, δx is the corresponding change of the filtered variable, $\delta\vartheta_0$ is defined by eq. 4.17 and the time scale of the EMA is ϑ_k-time. The notation of [Zumbach and Müller, 2001] is again used. Smoothing with a short range $\Delta\vartheta_{\text{smooth}}$ is necessary to diminish the influence of quote-to-quote noise. The EMA computation assumes a constant value of $(\delta x)^2/(\delta\vartheta_k + \delta\vartheta_0)$ for the whole quote interval, this means the "next point" interpolation [Müller, 1991, Zumbach and Müller, 2001].



The scalar filtering window moves forward in time by integrating new scalar quotes and dismissing old ones.

Figure 3: Schematic scalar filtering window

The fluctuations of the variable σ_k indicate the badness of the ϑ_k model. In the case of a bad fit, σ_k is often very low (when the ϑ_k-scale expands time) and sometimes very high (when the ϑ_k-scale compresses time). The fluctuations are quantified in terms of the variance F_k:

$$F_k \;=\; \mathrm{EMA}[\,\Delta\vartheta_r;\,(\,\sigma_k - \mathrm{EMA}[\,\Delta\vartheta_r;\,\sigma_k\,]\,)^2\,] \;= \tag{4.29}$$

$$\;=\; \mathrm{MVar}[\,\Delta\vartheta_r,\,2;\,\sigma_k\,]$$

in the notation of [Zumbach and Müller, 2001], where the time scale is again ϑ_k-time. The range $\Delta\vartheta_r$ has to be suitably chosen. In our heuristic approximation, the fluctuations directly define the weight of the k'th market:

$$w_k = \frac{1}{F_k \sum_{\text{all } k'} \frac{1}{F_{k'}}} \tag{4.30}$$

which satisfies eq. 4.27 and can be inserted in eq. 4.26.

5 The scalar filtering window

The filter is using a whole neighborhood of quotes for judging the credibilities of scalar quotes: the scalar filtering window. This window covers the set of all quotes of a time series that are contained in a time interval. In the course of the analysis, new quotes are integrated and old quotes are dismissed at the back end of the window following a certain rule. Thus the window is moving forward in time. This mechanism is illustrated by Figure 3.

All the scalar quotes within the window have a provisional credibility value which is modified with new incoming quotes. When the quotes leave the window, their credibilities are regarded as finally determined. Sufficiently credible quotes are then used to update the statistics needed for adaptivity.

At the initialization of a filter from scratch, the window is empty. When the first scalar quote enters, it cannot be filtered by pair filtering yet, only the level filter applies.

5.1 Entering a new quote in the scalar filtering window

Whenever a new scalar quote enters the window, an analysis is made based on earlier results and the new quote.

There are two possible ways in which a new quote enters the scalar filtering window:

1. The normal update: A new scalar quote from the data source enters, is analyzed and finally becomes the newest member of the scalar filtering window. The window variables are updated accordingly. These operations are described by sections 5.2 and 5.3.

2. A filter test: A new scalar quote from any source is merely tested. It is analyzed as in a normal update, but it does *not* become a member of the window. No window variable is changed by this test. Thus we execute the steps of section 5.2 and avoid those of section 5.3. The resulting trust capital of the new scalar quote is returned.

5.2 Analyzing a new quote in the scalar filtering window

5.2.1 Computing the trust capital of a new quote

The algorithm of the filtering window is organized in an iterative way. Whenever a new quote enters the window, an update is made based on earlier results and an analysis of the new quote.

When the new, *i*th scalar quote arrives, it already satisfies certain basic validity criteria (e. g. a price is not negative) and has possibly been transformed to a logarithmic value. This is ensured by the higher-level quote splitting algorithm explained in section 6. The following filtering operations are done with the incoming *i*th scalar quote:

1. The base trust capital T_{i0} is computed as the result of the level filter, eq. 4.6, if the scalar quote is a bid-ask spread. Otherwise, $T_{i0} = 0$. The resulting T_{i0} of eq. 4.6 is multiplied here by a configured constant c_{level} that determines the importance of level filtering.

2. The new quote is compared to *all* old quotes of the window through pair filtering steps as described in section 4.3. The trust capitals T_{ij} resulting from eq. 4.13 determine the trust capital T_i of the new quote and also affect the trust capitals T_j of the old quotes.

For computing T_{ij}, we need the expected squared value change V from eq. 4.20 and $\Delta\vartheta_{\text{corr}}$ from eq. 4.19 and therefore the number Q of valid quotes in the time interval from quote j to quote i. For this, we use the valid-quote age Q_j of the old quotes:

$$Q \;=\; Q_j + 1; \tag{5.31}$$

The increment by 1 stands for the new quote which is not yet integrated in the window. The computation of Q_j is explained at the end of section 5.3. The resulting value of Q is inserted in eq. 4.19.

The trust capital of the new, ith quote is computed in an additive way as follows:

$$T_i' \;=\; c_{\text{level}}\, T_{i0} + \sum_{j=i-n}^{i-1} C_j\, T_{ij} \tag{5.32}$$

T' is not termed T because it is not yet the final resulting trust capital in some cases. Eq. 5.32 is a weighted sum with weights $C_j = C(T_j)$ from eq. 4.1 which are the current credibilities of the n other quotes of the window.

The number n of quotes used for comparison to the ith quote has an influence on the trust capital and thus the credibility. The higher the value of n, the higher the trust capital according to eq. 5.32 (provided that we are in a series of good data). This effect reflects reality: the more comparisons to other quotes, the more certain our judgment on credibility. However, the trust capital cannot be extended infinitely by increasing n, as to be explained. If some more distant quotes are used, the trust capital gains T_{ij} are rapidly decreasing in absolute value, thanks to a term proportional to $(\Delta\vartheta_{ij})^3$ in the denominator of eq. 4.13. The choice of n is further discussed in section 5.4.

Eq. 5.32 is a conservative concept insofar as it judges the credibility of a new quote in light of the previously obtained credibilities C_j of the earlier quotes. In the case of an unusually large real move or price jump, new quotes on a new level might be rejected for a too long time.

5.2.2 The trust capital in an “after-jump” situation

The Olsen filter has a special mechanism to deal with “after-jump” situations. This may lead to a re-evaluation of the situation, a correction of the resulting trust capital T_i and a quicker acceptance of a new level after a jump.

The first step of the after-jump algorithm is to identify the location of a possible real jump within the scalar filtering window. This is done during the computation of eq. 5.32. At every j, we test whether the incomplete sum of that equation,

$$T'_{i,\text{at}\,j} \;=\; T_{i0} + \sum_{j'=i-n}^{j-1} C_{j'}\, T_{ij'} \tag{5.33}$$

is less than the critical value T_{crit}:

$$T_{\text{crit}} \;=\; \mu c_{\text{level}}\, T_{i0} - 1 \tag{5.34}$$

(where μ is defined below). At the same time, we test $T_{ij} > 0$ (this indicates having reached a new, stable level after the jump rather than an outlier). At the first j where *both* conditions are satisfied, we conclude that a value jump must have taken place somewhere before quote $j-1$. Although this jump certainly happened before quote j, we define $j_{\text{jump}} = j$ because this is the index of the first quote where we have a

reason to believe that the jump was real. In order to validate this possibly real value jump, we initialize an *alternative* trust capital T_i'':

$$T''_{i,\text{at } j_{\text{jump}}} \;=\; T_{\text{crit}} - 0.5 + \mu\,(T'_{i,\text{at } j} - T_{\text{crit}}) \tag{5.35}$$

We *dilute* the normal trust capital $T'_{i,\text{at } j}$ by a small dilution factor μ. When initializing the filter from scratch (before having seen some 10 acceptable quotes), we choose a slightly larger μ value in order to prevent the filter from being trapped by an initial outlier. The offset term -0.5 in eq. 5.35 prevents the alternative hypothesis from being too easily accepted. For all values of $j \geq j_{\text{jump}}$, we set

$$T''_j \;=\; \mu\, T_j \tag{5.36}$$

and insert these diluted trust capitals T''_j of old quotes in eq. 4.1; the resulting credibilities C''_j are used to complete the computation of the alternative trust capital T''_i:

$$T''_i \;=\; T''_{i,\text{at } j_{\text{jump}}} + \sum_{j=j_{\text{jump}}}^{i-1} C''_j\, T_{ij} \tag{5.37}$$

analogously to eq. 5.32. Thanks to the dilution effect, T''_i is less determined by old credibilities than T'_i.

Now, we decide whether to take the normal, conservative trust capital T'_i or the alternative T''_i. The resulting, final trust capital is

$$T_i \;=\; \begin{cases} T''_i & \text{if } T''_i > T'_i \text{ and } T''_i > 0 \\ T'_i & \text{otherwise} \end{cases} \tag{5.38}$$

The alternative solution wins if its trust capital exceeds 0 and the trust capital of the conservative solution.

The trust capital T_i of the new quote is the end result of a pure filter test. In the case of a normal update, the window has to be updated now.

5.3 Updating the scalar filtering window with a new quote

A new quote affects the trust capitals of the old quotes of the window. The most dramatic change happens in the case of accepting the alternative hypothesis according to eq. 5.38. In this case, a real value jump is acknowledged; this leads to a major re-assessment of the old quotes. First, the pairwise trust capital of quote comparisons across the jump is diluted:

$$T_{\text{corr},ij} \;=\; \begin{cases} \mu T_{ij} & \text{for } j < j_{\text{jump}} \\ T_{ij} & \text{otherwise} \end{cases} \tag{5.39}$$

In the normal case with no jump, $T_{\text{corr},ij} = T_{ij}$. Then, the quotes after the newly detected jump get a new chance:

$$T_{j,\text{new}} \;=\; \begin{cases} \mu T_j & \text{if } j \geq j_{\text{jump}} \text{ and } T_j < 0 \\ T_j & \text{otherwise} \end{cases} \tag{5.40}$$

In the case of a jump, this new value $T_{j,\text{new}}$ replaces T_j.

In every case, whether there is a jump or not, the trust capitals of all quotes are finally updated in additive way, in the spirit of eq. 5.32:

$$T_{j,\text{new}} \;=\; T_j + C_i\, T_{\text{corr},ij} \;\;, \quad \text{for } j \;=\; i-n \,\ldots\, i-1 \tag{5.41}$$

where $C_i = C(T_i)$ follows from eq. 4.1, inserting the newly obtained T_i (from eq. 5.38). The result $T_{j,\text{new}}$ of eq. 5.41 is replacing the old value T_j. It should also be clarified that the diluted values T_j'' from eq. 5.36 are never directly used to modify the trust capitals T_j.

In historical filtering, eqs. 5.39 - 5.41 may lead to the rehabilitation of an initially rejected old quote. Even in real-time filtering, the corrected trust capital of an old quote indirectly contributes to the filtering of new quotes through eq. 5.32 and through the use of only sufficiently credible old quotes in the statistics of adaptive filtering.

The valid-quote age Q_j of all the old quotes is also updated:

$$Q_{j,\text{new}} \;=\; Q_j + C_i \;, \quad \text{for } \; j \;=\; i-n \;\ldots\; i-1 \tag{5.42}$$

where $C_i = C(T_i)$. The more credible the new quote, the higher the increment of the valid-quote age Q_j.

After all these updates, the new quote with index i and with its newly computed trust capital T_i is inserted in the window as its newest member, with the valid-quote age Q_i initialized to zero.

5.4 Dismissing quotes from the window and updating the statistics

5.4.1 The quote dismissal rules

The window does not grow infinitely. There is a rule for dismissing scalar quotes at the end of every normal update as described in section 5.3. There are three criteria for a proper size of the window: (1) a sufficient time interval, (2) a sufficient number of quotes, (3) a sufficient *overall* credibility of all scalar quotes. These criteria are listed here in the sequence of increasing importance.

In our general quote dismissal rule, we use the product of the criteria. At the end of an update with a new quote, the following condition for dismissing the oldest quote (with index $i-n$) is evaluated:

$$(\vartheta_i - \vartheta_{i-n+1})\; n^2\; (\sum_{j=0}^{n-1} C_{i-j})^6 \;\;\geq\;\; W \tag{5.43}$$

The sum of credibility, the overall credibility, is the most important criterion and is therefore raised to the 6th power. The configuration parameter W defines the sufficient size of the window and has the dimension of a time. The parameter W is somehow related to the parameter ν of eq. 4.13 which determines a sort of a filtering range. Choosing a very large W when ν is limited does not add a lot of value because the distant quotes have a negligible weight in this case.

A few considerations may illustrate the behavior of eq. 5.43. If the data in the window is of good quality, the window is of small size. As soon as a cluster of low-quality or doubtful data enters the window, it will grow (sometimes to a very large size) until the situation becomes clearer and most old quotes can be dismissed again. In the case of a sparse time series, the window may contain few quotes but these quotes will extend further in time than for a dense time series.

After dismissing the oldest quote when eq. 5.43 is fulfilled, the whole quote dismissal procedure is repeated as long as the remaining window still satisfies eq. 5.43.

In very rare cases, the window grows to a very large size and the filtering algorithm becomes slow. This problem and Olsen's solution are discussed in section 5.4.3. Aside from this, an other safety measure is taken by the Olsen filter: an oldest quote older than 300 days is dismissed from the window even if eq. 5.43 is not fulfilled, as long as the remaining window still has at least two quotes.

Dismissed scalar quotes are also reported to the higher level of the filtering algorithm. This is necessary in the case of historical filtering for producing the final filtering results.

5.4.2 Updating the statistics

When a scalar quote is dismissed from the window, its credibility C_i has reached a final value that will no more be changed: $C_i = C(T_i)$ as resulting from eq. 4.1. This is the right moment to update all the statistics needed for the adaptivity of the filter.

Invalid quotes are excluded from these statistics, they are simply ignored when updating the statistical variables. We set a critical credibility level C_{crit}; only quotes with credibility values above C_{crit} are used for updating the statistics. However, we should not be too rigid when excluding quotes. The filter has to adapt to unexpected events such as sudden volatility increases, but this requires including also some mildly rejected quotes. In fact, tests have shown that only the totally invalid quotes should be excluded here. We choose a low critical credibility level. Only in the initial phase right after a filter starts from scratch (before having seen some 10 acceptable quotes), we take a larger, more cautious value.

If a dismissed quote has a credibility $C_i > C_{\text{crit}}$, we update all the statistics. These updates typically imply the computation of moving average iteration formulas; the statistics are explained in sections 4.2 - 4.4.

5.4.3 A second scalar filtering window for old valid quotes

The quote dismissal rule of eq. 5.43 makes sure that the scalar window stays reasonably small – except in case of a very long series of bad quotes. Such long series rarely occur, usually generated by computerized quoting, e. g. repeated or monotonic quotes. The filtering window technology as described so far is well able to handle this case, but the computation time of the filter grows very much in case of a very large window. In real time, this does not really matter, but historical filtering becomes slow.

For efficiency reasons, the Olsen filter therefore supports a second queue of old valid quotes. The normal scalar window size is strictly limited to a maximum number of quotes, but an old quote dismissed from the normal window is stored in a second scalar window if its credibility exceeds a low threshold value. Otherwise, the dismissed quote is treated as any dismissed quote, as explained before, including the updating of statistics and the final reporting of its credibility.

This second scalar window of old valid quotes is normally empty. As soon as one or more dismissed quotes are in this window, it is treated as a prepended part of the normal scalar window in all computations. The trust capital computation of eq. 5.32, for example, has a sum over both scalar windows, starting at the window of old valid quotes. The window of old valid quotes stays and possibly grows as long as the quote dismissal condition of eq. 5.43 for *both* scalar windows together is not fulfilled. When it is fulfilled, the oldest quote of the scalar window of old valid quotes is deleted; after deleting all of its quotes, the second window is again empty and filtering is back to the normal mode.

The proposed scalar window of old valid quotes is also presented as an element of the UML diagram of Figure 2. Note that a scalar quote of the “oldValidQuotes” window *owns* the pointer to a full tick (because the “FullQuoteWindow” no longer supports this full tick at this time), whereas the scalar quotes of the normal scalar window merely have a pointer to the corresponding full tick with no ownership.

The concept of a second scalar filtering window for old valid quotes adds quite some complexity to the filter but it is motivated only by efficiency reasons.

6 The full-quote filtering window

The full-quote filtering window is managed on hierarchy level 2 of Table 1. It is basically a sequence of recent full quotes plus a set of algorithmic methods of managing and processing this sequence. Its position in the algorithm is also shown in Figure 2 (class FullQuoteWindow).

The full-quote filtering window has the following tasks:

- Splitting the quotes into scalar quotes that can be used in the filtering operations of section 4.
- A first basic validity test for the filtered variables. This is usually a domain test, e. g. rejecting negative prices. Rejected scalar quotes are marked as invalid ($C_i = 0$) and eliminated from all further tests. They do not enter a scalar filtering window.
- In many cases, a mathematical transformation of the quoted level. Example: taking logarithms of prices instead of raw price values.
- Creating independent filtering environments for all types of scalar quotes, each with its own scalar filtering window.
- Storing the credibility of dismissed scalar quotes until all the other scalar quotes belonging to the same full quote have also been dismissed. (The spread filter may dismiss quotes before the bid price filter, for example).
- Storing the full quotes as long as two or more filtering hypotheses coexist, until one of them wins. This is decided by the next higher hierarchy level, see section 7; in the Olsen filter, the decision between filtering hypotheses is made fast enough to make this point superfluous.
- When a full quote is finally dismissed, reporting it, together with its filtering results, to the higher level (needed only in historical filtering).

In principle, the full-quote filtering window also offers the opportunity of analyzing those data errors that affect *full* quotes in a way that cannot be analyzed when just looking at scalar quotes after splitting. In the Olsen filter tests, we have never found a good reason to implement this (aside from the filtering hypotheses discussed in section 7); therefore there is no further discussion.

The full quotes may enter a full-quote filtering window in a form already corrected by a filtering hypothesis. This fact plays no role here: the algorithm of the full-quote window does not care about quote corrections. This is managed on a higher level.

The most important task of the full-quote filtering window is *quote splitting.*

6.1 Quote splitting depending on the instrument type

Quotes can have complex structures as explained in section 2.2. The Olsen filter follows the guideline of quote splitting which is motivated by the goals of modularity and transparency. Instead of trying to formulate complex algorithms for complex data structures, we split the quotes into scalar quotes that are individually filtered, wherever possible. Some filtering operations, however, are done on a higher level before splitting, as explained in section 7.

The quote splitting unit has the task of splitting the stream of full quotes into streams of different filtered variables, each with its scalar quotes that are used in the filtering operations of section 4.

Some quotes, such as bid-ask or open/high/low/close quotes, are splittable by nature. In Olsen's data collection, only the bid-ask case matters. Many instruments come in the form of bid-ask quotes. Other instruments have single-valued quotes. Bid-ask quotes are split into three scalar quotes:

1. bid quote,
2. ask quote,
3. bid-ask spread.

Other instruments have single-valued quotes which are "split" into one scalar quote:

1. the "level" quote.

This is not as trivial as it looks because quote splitting is coupled with two other operations: basic validity testing and mathematical transformations, as explained below.

The user of the filter has to know whether an instrument has single-valued or bid-ask quotes and has to select or configure the filter accordingly.

6.2 The basic validity test

Many quotes have a natural lower limit of the allowed *domain*. This instrument-dependent information has an impact on quote splitting and needs to be configured by the user. The lower limit of the allowed domain is called $p_{\min}$. For some instruments, there is no limit (or $p_{\min} = -\infty$). The choice of the lower limit is rather obvious for most instruments. Here is a list of important examples:

Prices: Genuine asset prices of whatever kind, including FX and equity prices, are never negative. This means: $p_{\min} = 0$.

FX forward premia/discounts: Unlike outright FX forward prices, the "forward points" (or FX swap rates) are not prices but rather values to be added to prices. Therefore, they can be negative or positive. There is no lower limit ($p_{\min} = -\infty$).

Interest rates: These can be *slightly* negative in practice, e. g. in the case of JPY interest rates towards the end of 1998, depending on the credit rating of the quoting bank (but these interest rates were above -1%). There are some theories that rely on interest rates staying always positive, but a filter is not allowed to reject slightly negative interest rates if these are posted by reasonable contributors. The Olsen filter is using a slightly negative value of $p_{\min}$ here.

Short-term interest-rate futures: These are often handled as normal prices (where $p_{\min} = 0$), but are defined as $100\% - 0.25f$ (by LIFFE, where f is the forward interest rate for the maturity period), so there is no lower limit (but an upper one). In practice, the futures quotes are so far from 0 that it does not matter whether we assume a lower limit of 0 or none.

The choice of the lower limit is important for the further treatment.

The following errors lead to complete invalidity:

- Quotes that violate the monotonic sequence of time stamps, i. e. quotes with a time stamp before the previously treated quote. (In some software environments, this is an impossible error).

- A domain error: an *illegal* level p of the filtered variable, i. e. $p < p_{\min}$ (as opposed to a merely implausible level).

Invalid scalar quotes with an error of this kind do not enter a scalar filtering window and are completely ignored in all further filtering steps. We mark them by setting $C_i = 0$. This is a fundamentally stronger statement than merely giving a very low credibility as a result of the scalar filtering window.

In the case of bid-ask quotes, the three resulting scalar quotes are tested individually:

1. bid quote: domain error if bid quote $p_{bid} < p_{\min}$.
2. ask quote: domain error if ask quote $p_{ask} < p_{\min}$.
3. bid-ask spread: domain error if $p_{ask} < p_{bid}$.

Thus it is possible that the same quote leads to a valid bid quote passed to the scalar filtering window of bid prices and an invalid ask quote that is rejected.

The domain test of bid-ask spreads needs to be further discussed. First, we might interpret bad values ($p_{ask} < p_{bid}$) as the result of a sequence error, i. e. the contributor typed ask/bid instead of bid/ask, an error that could be corrected by the filter. This interpretation, although being true in many cases, is dangerous as a general rule. We prefer to reject all ask quotes that are less than the bid quote.

On the other hand, a more rigid test might also reject *zero* spreads. In this case the domain error condition would be $p_{ask} \leq p_{bid}$. This is a good condition in many cases, well founded by theory. However, there are quite some quote contributors to minor markets interested only in either bid or ask or middle quotes. These contributors often produce formal quotes with $p_{bid} = p_{ask}$. In some markets, such quotes are the rule rather than the exception. A filter that rejects all those quotes is throwing away some valuable and irreplaceable information.

The Olsen filter solves this problem as follows. First, there is a filtering option of generally rejecting zero spreads, i. e. the case $p_{bid} = p_{ask}$. If the user chooses this option, the quote splitting algorithm will act accordingly. Otherwise, zero spreads can be accepted, but they have low credibilities in a market dominated by positive spreads. This is further explained in the next section.

6.3 Transforming the filtered variable

The filtered variable is mathematically transformed in order to reach two goals:

1. A simpler (e. g. more symmetric) distribution function. The basic filtering operations, e. g. eq. 4.6, assume a roughly symmetric distribution function of the scalar quote values (and their changes). Some variables, mainly the bid-ask spread, have a skewed distribution function. The filtering method contains no full-fledged analysis to determine the exact nature of the distribution function; that would be too much for an efficient filter algorithm. The idea of the transformation is that the mathematically transformed variable has a more symmetric distribution function than the raw form. For the logarithm of bid-ask spreads, this has been demonstrated [Müller and Sgier, 1992].
2. Stationarity: making the behavior of the variable closer to stationary. A nominal price is less stationary than its logarithm; this is why most researchers in quantitative finance formulate their models for logarithmic prices. Stationarity is a strong requirement only for very volatile instruments. If the time series is less volatile, the filter can also cope with non-stationary raw data thanks to its adaptivity.

The Olsen filter has simple rules for the mathematical transformation closely related to the validity tests of section 6.2. The mathematical transformation never fails because all illegal quotes have already been removed by the domain tests. The transformed quote value is denoted by x and used in many formulas of section 4.

For single-valued quotes, bid quotes and ask quotes, the following transformation is made:

$$x = \begin{cases} \log(p - p_{\min}) & \text{if } p_{\min} > -\infty \text{ exists} \\ p & \text{otherwise} \end{cases} \tag{6.44}$$

For bid-ask spreads, the transformation is

$$x = x_{\text{spread}} = 45.564\sqrt{x_{\text{ask}} - x_{\text{bid}}} \tag{6.45}$$

where x_{bid} and x_{ask} are results from eq. 6.44. Eq. 6.45 has been chosen to return a value similar to $\log(x_{\text{ask}} - x_{\text{bid}}) + \text{constant}$ for a wide range of arguments $x_{\text{ask}} - x_{\text{bid}}$ of typically occurring sizes. Indeed, a logarithmic transformation of spread values would be a natural choice. There is a reason to use eq. 6.45 rather than a logarithmic transformation: the treatment of zero spreads as already discussed at the end of section 6.2. A logarithmic transformation would make zero spreads impossible (as $\log(0) = -\infty$). When inserting a zero spread in eq. 6.45, we obtain the legal result $x = 0$. This value is quite far away from typical ranges of values obtained for positive spreads, so its credibility is likely to be low in normal situations. When zero spreads become a usual event, the filter will accept them after a while.

7 Univariate filtering

Univariate filtering is the top level of the Olsen filter. All the main filtering functions are managed here. This is also shown in Figure 2 (class UnivarFilter).

The full-quote filtering window with its quote splitting algorithm of section 6.1 is on a lower hierarchy level (see Table 1). Thus the univariate filter sees full quotes before they are split; it has access to all components of a full quote in their raw form (i. e. not mathematically transformed as in section 6.3). Its position on top of the filtering algorithm is also shown in Figure 2 (class UnivarFilter).

The tasks of univariate filtering are:

- Main configuration of a filter.
- Analyzing those data errors that affect not only individual quotes but a whole continuous sequence of quotes. The presence (or absence) of such a general error defines the *filtering hypothesis.* Two such cases were found in financial data and are therefore covered by the Olsen filter:

  1. Decimal errors: a wrong decimal digit of the quote, corresponding to a constant offset from the true quote.
  2. Scaling factor: the quote deviates from the true level by a constant factor, often a power of 10.

  Both cases are further discussed below.
- Creating a new full-quote filtering window for a newly detected filtering hypothesis.
- Managing filtering hypotheses and their full-quote filtering windows during their lifetimes, selecting the winning hypothesis.

- In the case of an error hypothesis: correcting the error of new incoming quotes according to the hypothesis and passing the corrected quotes to the full-quote filtering window.
- Packaging the filtering results to be accessed by the user. The timing of the filtering output is different in historical and real-time filtering as to be explained.
- Recommending a suitable build-up period of the filter prior to the desired start date of the filtering result production, based on the filter configuration. A discussion of this task can be found in section 9.1.

The errors affecting a continuous sequence of quotes cannot be sufficiently filtered by the means described in the previous sections; they pose a special challenge to filtering. The danger is that the continuous stream of false quotes is accepted to be valid after a while because this false series appears *internally* consistent.

A filtering hypothesis is characterized by one general assumption on an error affecting all its quotes. This can lead to another unusual property. Sometimes the cause of the error is so clear and the size of the error so obvious that quotes can be *corrected.* In these cases, the filter produces not only credibilities and filtering reasons but also corrected quotes that can be used in further applications. This is discussed further below.

The errors leading to a filter hypothesis are rare. Before discussing the details, we should evaluate the relevance of this filtering element in general. Such an evaluation may lead to the conclusion that the filtering hypothesis algorithm is not necessary in a new implementation of the filter.

Decimal errors have been the dominant error type in the page-based data feed from Reuters in 1987-1989. In later years, they have become rare; they hardly exist in modern data feeds. The few remaining decimal errors in the 1990s often were of short duration so they could successfully be filtered also through the standard data filter. Thus there is no convincing case for adding a decimal error filter algorithm to a filter of modern data. The Olsen filter nevertheless has a decimal error filter because it is also used for filtering old, historical data.

The scaling filter is also superfluous if the user of the filter has a good organization of raw data. If a currency is re-scaled (e. g. 1000 old units = 1 new unit as in the case of the Russian Ruble), a company with good data handling rules will not need the *filter* to detect this; this re-scaling will be appropriately handled before the filter sees any scaling problem. At Olsen, re-scaled currencies (or equity quotes after a stock split) are treated as a *new* time series. However, the transition between the two definitions may not be abrupt, and there may be a mixture of quotes of both scaling types for a while. A scaling analysis within the filter can serve as an additional element of safety to treat this case and detect unexpected scale changes. This is the purpose of the scaling analysis in Olsen's filter.

There is the possibility of having coexisting hypotheses, for example the hypothesis of having a decimal error and the hypothesis of having none. This feature of the filter architecture is not used by the Olsen filter. In Olsen's filter, an immediate decision in favor of one hypothesis is always made, so there is no need to store two of them simultaneously.

Although the filtering hypothesis algorithms may not be needed in a new filter development, this document contains a description of these algorithms. Note that all these algorithms are executed for each new quote before quote splitting.

7.1 Decimal error filter

The decimal error [Müller, 1989, Rukkers, 1991] is the only well-known, relevant type of a general error affecting whole quote sequences. (Scale changes cannot be called errors as they are usually made for

good reasons).

The decimal error as a persistent phenomenon is caused by data processing technologies using a *cache* memory. This means intermediary storage of data by the data providers or their software installed at customers' sites. The data customers sees a quote extracted from this cache memory rather than the original quote or its direct copy. The error happens when cache memories are updated by partial updates rather than full refreshments of cached quotes. The use of partial updates is motivated by minimization of message sizes. If one of these update messages gets lost and the following updates succeed, the result in the cache memory can be wrong, e. g. a decimal error. Unfortunately, this error may stay for a long time once parts of the cache are wrong.

Let us now look at a practical example: the decimal errors typical for page-based data feeds of Reuters in the late 1980s. Assume an original, correct quote, 1.5205/15, in the cache memory (format: bid price / ask price, last two digits). This price is updated by a partial update message, "198/08", placed at the end of the initial quote location, with the intention to store the new quote 1.5198/08. This update is lost now, the quote in the cache stays 1.5205/15. The next update message is "95/05", placed at the end of the initial quote, with the intention to store the new quote 1.5195/05. The resulting bad quote in the cache is 1.5295/05 because the "big" decimal digit "2" has survived in the cache. This is a decimal error: a shift by 0.01 which is a power of 10.

It seems that decimal errors are much less likely to occur in record-based data feeds than in page-based feeds. However, there may be hidden cache mechanisms also in the information processing of record-based data feeds; this should be investigated. If there is no cache or similar mechanism in the whole environment of data to be filtered, the decimal error filter can be omitted.

The Olsen filter contains a decimal error filter. At the very beginning, it always assumes the default state of no decimal error. Every new, incoming quote can be distorted by a decimal error and is tested accordingly. The test for entering a decimal error focuses on some characteristic features of the error generation mechanism as described above. The following criteria are tested in sequence:

- Size of the jump: the value change d between the previous quote and the new quote is computed. If $|d|$ is close to a power of 10 (preferably slightly below), the decimal error hypothesis is supported. The test is $|d|/p > 0.6999$, where p ($> |d|$) is the next larger power of 10, also with negative exponent. In our example, the bid price change is from 1.5205 to 1.5295, so $d = 0.0090$, $|d|/p = 0.9$, and the decimal error hypothesis is supported.

- Size of the time interval: support the decimal error hypothesis only if the interval from the previous quote to the new quote is less than 70 minutes. In Olsen's practical experience, decimal errors do not happen over long time gaps.

- Validity of the corrected quote: the corrected quote is computed, assuming there was a decimal error with shift of size p. This corrected value must pass the validity test of section 6.2.

- Credibility advantage: the decimally corrected value should be much more credible than the raw value. If this is not the case, the decimal error is rejected.

- Critical digit: in the lifetime of a decimal error generated by the mechanism described above, the bad decimal digit ("2" in our example) stays the same. This is also tested. However, this test may fail for incomplete ask quotes of a true decimal error, as in the typical quoting style used in page-based feeds, like 1.5295/05. This quote is expanded to 1.5295/1.5305. Therefore, the Olsen filter requires the critical digit test to succeed just for the bid quote, not for ask.

A decimal error is accepted only if it is confirmed by all these tests. The credibility advantage test needs a further explanation. Decimal errors are rare, and we should not accept spurious decimal errors, so the

credibility advantage in their favor must be overwhelming in order to confirm them. The following test is used:

$$T_{\text{raw}} \quad < \quad -0.1 - [4 - \min(T_{\text{corr}}, 4)]^2 \tag{7.46}$$

This means that acceptable uncorrected quotes with $T_{\text{raw}} > -0.1$ will never be rejected in favor of decimally corrected quotes. The two trust capitals T_{raw} (of the raw, uncorrected quote) and T_{corr} (of the corrected quote) result from eq. 5.38 after a mathematical transformation of the raw or corrected quote by eq. 6.44. Here we have a case where the trust capital of new quotes is computed just for testing, *without updating* any variable of the scalar or full-quote windows. This is explained at the beginning of section 5.1.

The case of bid-ask quotes is more complicated. The bid and ask quotes are tested independently, but the critical digit test is omitted for the ask quote, as explained above. If both have a decimal error, we test whether they have the same error, i. e. whether their value changes d have the same sign and the same power of 10, p. Then the error is a *common* decimal error. Other results are also possible: decimal error for bid quote, for ask quote alone or independent decimal errors for bid and ask. (In the last two cases, the critical digit test must succeed also for ask quotes). Olsen's tests have shown that the common decimal error is the only decimal error type of bid-ask quotes that really needs to be tested. All the other types are of short duration so they can be covered also by the normal filter (or they are spurious). Therefore, we recommend testing only for common decimal errors.

Once a decimal error is found, the decimal error hypothesis is accepted and the new quotes enter the full-quote filtering window in corrected form. This is also true for all successor quotes until the decimal error hypothesis is terminated. This is the place where the coexistence of two full-quote filtering windows could start: we might have one window with corrected quotes and another, parallel window with uncorrected quotes. This would be the best solution in doubtful cases: there, we could keep both hypotheses alive until some later quotes would give clear evidence in favor of one of them. Thanks to the low importance of decimal errors, the Olsen filter does not need to use this possible best solution. It always takes an immediate decision, either against the decimal error or, if all the conditions are fulfilled, for the decimal error. Thus it does not need to conduct parallel full-quote filtering windows.

The decimal error hypothesis, once accepted, can be terminated only if some termination conditions are fulfilled. For every new quote, the following tests are made in sequence:

- Time-out: a decimal error hypothesis older than 2 full days is terminated.
- Back-jump: the value change d between the previous quote and the new quote is computed. If d has the opposite sign of the d at the start of the decimal error and $|d|/p > 0.4999$, we probably see the back-jump from the bad level to the true level. This leads to a termination of the decimal error only if the next test is also passed.
- Credibility advantage: this test is only executed if the previous test finds a probable back-jump. The decimally corrected value should be much less credible than the raw value. If this is the case, the decimal error is terminated.
- Critical digit: if the critical digit changes, the decimal error is immediately terminated. In the case of bid-ask data with a common decimal error, this test is made for the bid quote only. The ask quote, if expanded from an incomplete quoting format such as 1.5295/05, may change the critical digit also while staying in a decimal error.

The credibility advantage test for the back-jump is less rigid than the test for jumping into a decimal error, eq. 7.46. The following test for terminating a decimal error is used:

$$T_{\text{corr}} \quad < \quad 0.3 - 0.5\,[4 - \min(T_{\text{raw}}, 4)]^2 \tag{7.47}$$

The two trust capitals T_{raw} (of the raw, uncorrected quote) and T_{corr} (of the corrected quote) again result from eq. 5.38 after a mathematical transformation of the raw or corrected quote by eq. 6.44. These computations are done just for testing and do not lead to any updates of the scalar or full-quote windows.

After the termination of a decimal error, the filter will continue its normal operations using the raw, uncorrected form of the quote. This is also true for the following quotes, but each one will again be tested by the entry conditions of a new decimal error.

7.2 Scaling analysis in the filter

Some quotes are scaled by a constant factor as compared to the level of the earlier quotes. There is often a good reason for this scale factor: changed quoting habits (e. g. quoting the value of 1 DEM in NOK, Norwegian Crowns, instead of the value of 100 DEM) or re-definitions of currencies (e. g. 1 new Russian Ruble = 1000 old Rubles). If this data is sent to the filter with no intervention, the standard filter just sees a huge price jump which it has to reject.

The Olsen filter analyzes and possibly corrects those scaling factors that are powers of 10. Other factors are also possible (see section 8.3.1) but not easily tractable.

The scaling analysis always starts from the previous scale factor S_{prev}. At initialization from scratch, this is of course $S_{\text{prev}} = 1$. The new quote p is checked in comparison to the old quote p_{prev}. (At initialization from scratch, p_{prev} is initialized to the first incoming quote value). Normally, $p/p_{\text{prev}} \geq \sqrt{0.1}$ and $p/p_{\text{prev}} < \sqrt{10}$. (Square roots of 10 are used in these tests as natural separators between factors of 10). In rare cases, one of these conditions is violated. If p and p_{prev} have different signs, the whole analysis is stopped and the old factor S_{prev} is kept. Otherwise the power of 10, 10^n is determined that satisfies $10^n p/p_{\text{prev}} \geq \sqrt{0.1}$ and $10^n p/p_{\text{prev}} < \sqrt{10}$. The newly proposed scaling factor is then $10^n S_{\text{prev}}$.

This new scaling factor must be confirmed by further tests to be accepted. In some cases, a quote level change of a factor 10 or more can be natural for a time series. Examples: an FX forward premium can easily and quickly shrink from 10 to 1 basis point; Japanese interest rates were so low that they could easily change from 0.06% to 0.6% between two quotes (especially if these two quotes came from banks with different credit ratings). In these cases, a new scale factor would be spurious; correcting by it would be wrong.

Thus we first test whether the new quote scaled by the previous factor S_{prev} would be rejected by the filter:

$$T_{\text{oldFactor}} \quad < \quad \begin{cases} -17 & \text{if } 10^n = 10 \text{ or } 10^n = 0.1 \\ -5 & \text{otherwise} \end{cases} \tag{7.48}$$

where $T_{\text{oldFactor}}$ is obtained from eq. 5.38. As in the decimal error filter, T is computed just for testing, without updating any variable of the scalar or full-quote windows. The condition of eq. 7.48 must be fulfilled in order to give the new scaling factor factor a chance. It is particularly strong if the scale change is by a factor 10 or 0.1, for two reasons. A scale change by a factor 10 or 0.1 is

1. extremely rare (there is no known case in the FX, IR and many other markets);
2. more likely than any other scale change to be a true move of the quote value, as in the example of the Japanese interest rates mentioned above.

If the new quote scaled by the previous scaling factor is rejected, we need yet another test: does the new quote scaled by the *new* factor $10^n S_{\text{prev}}$ have a superior credibility? We compute $T_{\text{newFactor}}$ in the same

way as $T_{\text{oldFactor}}$ and test the following condition:

$$T_{\text{newFactor}} \;>\; \begin{cases} T_{\text{oldFactor}} + 20 & \text{if } 10^n = 10 \text{ or } 10^n = 0.1 \\ T_{\text{oldFactor}} + 8 & \text{otherwise} \end{cases} \tag{7.49}$$

Again, the condition is particularly strong if the scale change is by a factor 10 or 0.1. If it is fulfilled, the new scaling factor $10^n S_{\text{prev}}$ of the quoted value is accepted.

For bid-ask quotes, this scaling analysis is made for both the bid and ask quotes. A scale change is accepted only if both have the same new scale factor. Otherwise, the new scale factor is rejected and the old scaling factor S_{prev} is kept.

The value of an *accepted* new scaling factor, $10^n S_{\text{prev}}$, is finally assigned to the variable S_{prev}.

7.3 The results of univariate filtering

The output of the univariate filter consists of several parts. For every quote entered, the following filtering results are available:

1. The credibility of the quote.
2. The value(s) of the quote, possibly corrected according to a filtering hypothesis as explained in sections 7.1 and 7.2.
3. The filtering reason, explaining why the filter has rejected a quote.
4. Individual credibilities of scalar quotes (bid, ask, spread)

Users may only want a minimum of results, perhaps just a yes/no decision on using or not using the quote. This can be obtained by simply checking whether the credibility of the quote is above or below a threshold value which is usually chosen to be 0.5.

In the case of bid-ask data, the credibility C of the full quote has to be determined from the credibilities of the scalar quotes. The Olsen filter uses the following formula:

$$C \;=\; \min(C_{\text{bid}}, C_{\text{ask}}, C_{\text{spread}}) \tag{7.50}$$

This formula is conservative and safe: valid quotes are meant to be valid in every respect.

The timing of the univariate filtering output depends on its timing mode: historical or real-time.

7.4 The production of filtering results – in historical and real-time mode

The terms “historical” and “real-time” are defined from the perspective of filtering here. The final application may have another perspective. A filter in real-time mode may be applied in a historical test, for example.

The two modes differ in their timing:

- In the *real-time* mode, the credibilities of a newly integrated quote as resulting from eq. 5.38 (inserted in eq. 4.1) are immediately passed to the univariate filtering unit. If there is only one filtering hypothesis as in the Olsen filter, these credibilities are directly accessible to the user. If there are several hypotheses, the hypothesis with the highest overall credibility will be chosen (but the pure fact of the existence of several hypotheses leads to a reduced credibility of all of them).

- In the case of *historical* filtering, the initially produced credibilities are modified by the advent of new quotes. Only those quotes are output whose credibilities are finally "cooked". At that time, the quotes leave the full-quote filtering window; this implies that their components have also left the corresponding scalar filtering windows. If several filtering hypotheses coexist (never in the Olsen filter), their full-quote windows do not dismiss any quotes, so we get filtering results only when conflicts between filtering hypotheses are finally resolved in favor of one winning hypothesis.

Although these modes are different, their implementation and selection is very easy. In the historical mode, we retrieve the oldest member of the full-quote window only after a test on whether this oldest quote and its results are ready. In the real-time mode, we pick the newest member of the same full-quote window. Thus it is possible to get both modes from the same filter run.

A special option of historical filtering should be available: obtaining the last quotes and their results when the analysis reaches the most recent available quote. It should be possible to "flush" the full-quote window (of the dominant filtering hypothesis) for that purpose, even if the credibilities of its newest quotes are not finally corrected.

This leads to another timing mode that might frequently occur in practice. A real-time filter might be started from historical data. In this case, we start the filter in historical mode, flush the full-quote window as soon as the filter time reaches real time and then continue in real-time mode. This can be implemented as a special, combined mode if such applications are likely.

8 Special filter elements

In the previous sections, all the standard Olsen filter have been explained. The standard algorithm is good for covering a large set of typically occurring outliers. However, some data errors are of a very special nature, so we need specialized filter elements for efficiently catching these errors. A list of special errors is already given in section 2.3.

The special filter elements are using the same filtering structure as the standard filter algorithm, e. g. the scalar filtering window as defined by section 5. The special filter elements can thus be well integrated in the standard Olsen filter. Descriptions of the individual filter elements follow.

8.1 Filtering monotonic series of quotes

In several time series, mostly FX, the following phenomenon can be observed overnight or mostly during weekends. A contributor posts a monotonically increasing (or decreasing) series of quotes, starting at an acceptable quote level, often at a rather high frequency, e. g. 1.3020/25, 1.3021/26, 1.3022/27, 1.3023/28, ... The first few quotes look acceptable, but the quote level will be very far from the true level if the monotonic series is long. The motivation for this casually occurring quoting behavior can only be guessed; it is probably testing the performance and the time delay of the connection to the data service.

The standard filter is not very suited to properly analyzing this bad quoting behavior because the individual quote-to-quote changes look harmless, just 1 basis point in the given example. Even the value change over several quotes is not large enough to ensure proper action of the filter. After a long monotonic series of fake quotes, the filter may even give low credibility to the following *correct* values as these are far away from the end values of the monotonic series.

If the contributor of the monotonic series stands alone, it is not hard to detect the unnatural linearity of these quotes. Unfortunately, a bad habit has become very widespread among contributors to multi-

contributor quote feeds: *copying* of quotes as mentioned in section 2.3. The copied quotes are often modified by simple post-processing steps such as averaging of a few recent quotes, a small random value change and/or a small random time delay. The frequent posting of copied quotes seems to be motivated by the desire of the issuers to advertise their market presence. Together, the many copied and modified quotes make the monotonic series appear valid because they accompany it in a slightly irregular, less monotonic way, seemingly confirmed by many "independent" contributors.

The following algorithm is used to identify and filter monotonic series of fake quotes under the described, difficult circumstances. The framework of this algorithm is the scalar filtering window as explained by section 5; the algorithm is simply added to the other computations of the scalar filtering window. First the weight of the positive value changes from old quotes to the new quote is computed:

$$S_+ \;=\; \sum_{j=i-n}^{i-1} w_{ij}\,\delta_{x_i>x_j} \tag{8.51}$$

where $\delta_{x_i>x_j} = 1$ in case of an up move, otherwise 0. We also do this for the negative value changes,

$$S_- \;=\; \sum_{j=i-n}^{i-1} w_{ij}\,\delta_{x_i<x_j} \tag{8.52}$$

and the sum of weights:

$$S \;=\; \sum_{j=i-n}^{i-1} w_{ij} \tag{8.53}$$

The weight is defined to emphasize the comparison to the most *recent* quotes from the **same** contributor:

$$w_{ij} \;=\; \frac{1.1 - I_{ij}}{4 + (i-j)^2} \tag{8.54}$$

where I_{ij} is given by eq. 4.23.

The *share* of positive movements is given by the following formula which is a simple EMA iteration as defined by [Müller, 1991]:

$$S'_{+,i} \;=\; \frac{2.5}{2.5 + S_+ + S_-}\,S'_{+,i-1} + \frac{S_+ + S_-}{2.5 + S_+ + S_-}\,S_+ \tag{8.55}$$

The negative movements have the complementary share because zero changes are ignored here:

$$S'_{-,i} \;=\; 1 - S'_{+,i} \tag{8.56}$$

At the very beginning, $S'_{+,0}$ and $S'_{-,i}$ are initialized to 0.5. $S'_{+,0}$ and $S'_{-,i}$ lead to the direction balance B_i:

$$B_i \;=\; \frac{f^4 \min(S'_{-,i}, S'_{+,i}) + 1}{f^4 + 2} \tag{8.57}$$

In case of low-frequency data, e. g. with a "crawling peg" (a case of a "natural" monotonic move), the direction balance is made less extreme in eq. 8.57 using the quote frequency f:

$$f_i \;=\; 0.1 A_i^2 \tag{8.58}$$

where A_i^2 is measure of the recent mean distance between two quotes with a non-zero value change, computed by an EMA iteration similar to eq. 8.55:

$$A_i \;=\; \frac{2.5}{2.5+S_+ +S_-}\,S'_{+,i-1} + \frac{S_+ + S_-}{2.5+S_+ +S_-}\,\sqrt{\Delta\vartheta_{\text{nonZero}}} \tag{8.59}$$

$\Delta\vartheta_{\text{nonZero}}$ is the age of the last quote value different from x_i, seen from time ϑ_i, measured in days of ϑ-time.

If the obtained direction balance is very poor (low value of B_i), we suspect monotonic quoting. The following maximum value of the trust capital T_i of the new quote is defined:

$$T_{i,\text{monoton}} \;=\; 25 - \frac{0.001\,B_i^4\,\sqrt{f_{\text{rel}}}}{a^2} \tag{8.60}$$

with the recent relative quote frequency f_{rel}

$$f_{\text{rel}} \;=\; f_i/d_i \tag{8.61}$$

where d_i is the current quote density as computed by eq. 4.15. The expected activity of the market is also needed:

$$a \;=\; \frac{\Delta\vartheta_i}{\Delta t_i} \tag{8.62}$$

This activity, the ratio between two different measures of the last quote interval (once in ϑ-time and once in physical time), is very low on holidays and weekends, often leading to a strongly reduced trust in eq. 8.60. This is intentional and successful in tests as monotonic quoting is a problem mainly in market periods of low liquidity. According to eq. 4.25, a has a value of 0.01 on holidays and weekends; otherwise the value can go up to 3.4 depending on the activity and weight of different markets.

Under normal circumstances, $T_{i,\text{monoton}}$ has a high value. In the special situation of $T_{i,\text{monoton}}$ being less than the trust capital T_i obtained by eq. 5.38, T_i is replaced by $T_{i,\text{monoton}}$ and "monotonic quoting" is established as the filtering reason of the new quote.

8.2 Filtering long series of repeated quotes

As explained in section 2.3, some contributors let their computers repeat a quote many times. If this happens hundreds or thousands of times, the repeated quotes may obstruct the filtering of good quotes from other contributors. Very long series of repeated quotes cannot be tolerated.

Filtering repeated quotes by one contributor seems to be easy. However, there are three difficulties:

1. contributor IDs may not be available;
2. repeated quotes may be mixed with other quotes;
3. repeated quotes may be "natural" in a certain market.

The latter case is found in markets where quote values are denominated with coarse granularity, e. g. in Eurofutures where the lowest resolution (= granule) is of one basis point. Repeated Eurofutures quote values, also from different contributors, are quite normal.

If the filtered scalar is a bid-ask spread, repeated quotes are also natural. We do not conduct any repeated-quote analysis for bid-ask spreads; the following explanation applies to bid, ask or single-valued quotes only.

The repeated-quote analysis is done for scalar quotes in the framework of the scalar quote window as presented in section 5. This analysis is thus located at the same place as that of monotonic quotes as discussed in section 8.1. These two algorithms share a part of their variables and computations.

As a first step, the length of the sequence of repeated quotes is measured by the following counting formula:

$$n_{\text{zero},i} \;=\; n_{\text{zero},i-1} + \frac{S - S_+ - S_-}{S} \tag{8.63}$$

This is not a simple counter but rather a cumulator of the variable $(S - S_+ - S_-)/S$, based on eqs. 8.51 - 8.53. This variable is a weighted measure of very recent zero changes, emphasizing quotes from the same contributor. By using this variable, the analysis can detect repeated quotes also if they are mixed with others. At the very beginning, $n_{\text{zero},0}$ is initialized to 0. Whenever $(S - S_+ - S_-)/S$ is less than a threshold value, the sequence of repeated quotes is regarded as terminated and $n_{\text{zero},0}$ is reset to 0.

The obtained length $n_{\text{zero},0}$ of the sequence has to be put into the context of the quoting *granularity*. If the granule (= minimum size of a usual value change) is high as compared to the mean value change from quote to quote, high values of $n_{\text{zero},0}$ are natural, and the filter should accept this. The granule size of each market may be configured as a nominal value. However, we prefer to *determine* the typical granule size in an adaptive way. This has the advantage of measuring the actual practice rather than formal conventions; changes of granularity are followed in an adaptive way.

The granularity is measured by an interesting algorithm that would deserve a detailed discussion. However, this is not a central issue of filtering, so we just give the resulting formulas. The computation is based on moments of the value change distribution function where the exponent of the moments is *negative*. All zero changes thus have to be excluded from this computation. Negative exponents emphasize small value changes, i. e. those that are close to the typical granule size. The moment with exponent $-k$ is computed with the help of the following moving average iteration:

$$m_{-k,i} \;=\; \mu_m\, m_{-k,i-1} + (1 - \mu_m)\, |\Delta x_i|^{-k} \tag{8.64}$$

where Δx_i is the value change from the last quote and the coefficient μ_m determines the memory of $m_{-k,i}$. Eq. 8.64 can also be seen as an EMA iteration according to [Zumbach and Müller, 2001], on a time scale that ticks with every quote with a non-zero value change. This iteration is applied to only valid quotes with a non-zero value change as they are dismissed from the scalar quote window, see section 5.4.2. At the first quote that brings a non-zero value change Δx_i, $m_{-k,i}$ is initialized to $|\Delta x_i|^{-k}$. Before this initialization (when starting from scratch), $m_{-k,i}$ cannot be used and the repeated-quote analysis is not yet operational. Numerous tests with granular (discrete) model distributions such as binomial distributions (also some more fat-tailed distributions) have shown that the following formula gives a good estimate of the granule g:

$$g \;=\; \frac{m^2_{-1/2,i}}{m_{-2,i}} \tag{8.65}$$

Thus we use the moments with exponents -1/2 and -2. When starting a filter run from scratch, $m_{-1/2,i}$ and $m_{-2,i}$ are not yet available for a short period, so g is undefined. The repeated-quote analysis can be done only after this period.

Now we need the natural probability of a zero change, given a certain distribution function and the granule size g. The same tests that led to eq. 8.65 also lead to another heuristic approximation formula of the probability of zero changes:

$$p_{\text{zeroChange}} = \mathrm{e}^{4.2\,(g\,m^2_{-1/2}-1)} \tag{8.66}$$

This formula (which is not yet our end result) is conservative: it overestimates $p_{\text{zeroChange}}$ mainly for small granules (where $p_{\text{zeroChange}}$ goes to 0). This is what we want here.

In fact, we have $n_{\text{zero},i}$ zero changes in a row, not just one. The probability of this is much smaller:

$$p_{\text{nZeroChanges}} = p^{n_{\text{zero},i}}_{\text{zeroChange}} = \mathrm{e}^{4.2\,n_{\text{zero},i}\,(g\,m^2_{-1/2}-1)} \tag{8.67}$$

Now we formulate a maximum trust capital for repeated quotes whose value, when converted to a credibility according to eq. 4.1, corresponds to $p_{\text{nZeroChanges}}$ in the limit of low credibility:

$$T_{i,\text{repeated}} = 25 - 0.5\,p^{-0.32}_{\text{nZeroChanges}} \approx 25 - 0.5\,\mathrm{e}^{1.35\,n_{\text{zero},i}\,(g\,m^2_{-1/2}-1)} \tag{8.68}$$

This maximum trust capital is much higher than the theoretical value postulated before: there is a large positive offset of 25, and the exponent -0.32 (instead of -0.5) dampens the effect of a high number $n_{\text{zero},i}$ of repeated quotes. $T_{i,\text{repeated}}$ is a conservative estimate; if we get low trust, we can be sure that the series of repeated quotes is unnatural in the given market. It must be the result of thoughtless repetition, probably by a computer.

Eventually, we compare $T_{i,\text{repeated}}$ to the normal trust capital T_i of the quote. T_i results from eq. 5.38 and is assumed to be already corrected after a comparison to $T_{i,\text{monoton}}$ (from eq. 8.60). If $T_{i,\text{repeated}}$ is less than T_i, we replace T_i by $T_{i,\text{repeated}}$ and set “long repeated series” as the filtering reason of the quote.

8.3 Disruptive events and human intervention

The filter as described so far is adaptive to a wide range of different financial instruments and also adapts to changes in the behavior of these instruments. Adaptation needs time - it means adapting to statistical results which have a certain inertia in their behavior by definition. The volatility measure of eq. 4.16, for example, has a range $\Delta\vartheta_r$ of 7 days. Thus the filter cannot immediately adapt to a disruptive event that leads to an extreme, unprecedented behavior in few hours.

Disruptive events are rare events. There are some additional filtering elements to cover them. Disruptive events can be sorted in three types:

1. **Scaling change**: a financial instrument quoted with a changed scale factor. Examples: a stock split or the re-definition of units of an instrument. These changes result in a sudden change of quote values by a constant factor, often but not always by a power of 10. This does not imply any market shock or change in volatility.

2. **Change in the instrument definition**: a financial instrument undergoes a substantial change of its definition, not only in scaling. Example: Benchmark bonds are quoted as the underlying instrument of certain bond futures. When they approach maturity, they are replaced by another bond, but the quotes appear in the same, uninterrupted time series. This implies a sudden jump in value, but the volatility of the new bond on the new level is not fundamentally different from the old one.

3. **Regime change**: the regulation of financial instruments can be changed. Example: the floating regime of the Brazilian Real after a long period of a fixed rate to the USD. Regime changes of this kind have two effects:

(a) a sudden move to a new level (e. g. devaluation of the Brazilian Real),

(b) very high volatility of oscillations around the new level.

These two effects together make filtering difficult. The most extreme, unexpected *market shocks* can behave similarly to a regime change. Inverse regime changes - from floating to fixed - are no real problem for the filter. These changes are usually anticipated by a quiet market behavior, so the normal filter can well adapt.

The filter *user* has the following means to deal with disruptive events:

- Anticipation by opening a new time series: many disruptive events such as definition and regime changes are announced in advance. A newly defined Russian Ruble, for example, is a new time series and should be treated as such: it should get its own, independent filter. Thus each filter sees data either only following the old definition or the new one.
- Human intervention: re-starting the filter immediately after the disruptive event. This is a viable, though not elegant solution, given the rarity of disruptive events and the difficulty of implementing a satisfying on-line solution.
- Anticipation by on-line warning message: the user can send a warning message to the filter, informing it on an imminent change of scaling factor or volatility shock.
- Human on-line correction: the user can tell the filter to accept a certain quote or approximate quote level after the disruptive event.

The last two features are not yet implemented by the Olsen filter. Olsen relies on off-line human intervention in the rare case of a disruptive event.

The filter should treat disruptive events as well as possible even without human intervention. It has the following means:

- Reacting to human messages: in case of a warning message, the filter can be prepared to the imminent event, e. g. by lowering the dilution factor μ as discussed in section 5.2.2. A human correction can be used to modify the stored trust capitals of the scalar filtering queue. This is not yet implemented by the Olsen filter as explained above.
- Immediate detection and adaptation: some disruptive events follow from the analysis of a side variable of the quote. This is implemented in the Olsen filter as explained in section 8.3.2.
- Delayed detection: a disruptive event may initially result in an unusually long series of rejected quotes after the change. This fact can be used to inform the user on possible problem. The filter may either send a special warning to the user, or an independent monitoring tool may produce this warning (this is Olsen's solution).
- Delayed adaptation: eventually, the filter will adapt to the new regime after the disruptive event. This is ensured by the adaptivity and the time-out of the scalar filtering window.

Disruptive events are not a problem of the filter alone. Their treatment also depends on the application, the environment and the goals of the filter user.

The three types of disruptive events are further discussed in the following sections.

8.3.1 Scaling change

Scaling changes are often caused by real economic facts. If these can be anticipated, the filter user should prepare the filtering of newly scaled quotes in advance. However, some scale changes may come by surprise.

Here are some examples:

- Changing quoting habits: Norwegian banks used to quote the value of 100 DEM in NOK. Later, they preferred quoting the value of 1 DEM in NOK.
- Re-definition of units: one new Russian Ruble is re-defined to be 1000 old Rubles.
- The French stock index (CAC40) is quoted in Euros instead of French Francs.
- Stock splits: one old share is split into three new shares of the same company.

The first examples usually cause a scale factor that is a power of 10 with an integer exponent. The filter deals with such factors as described in section 7.2. The last examples usually have odd scale factors different from a power of 10. In this case, the user has to intervene, e. g. by opening a new time series and a new filter for the newly scaled quotes.

8.3.2 Change in the instrument definition

Changes in the instrument definition are rare, but there is one instrument where they regularly occur: benchmark bonds. Benchmark bonds are quoted as the underlying instrument of certain bond futures. When they approach maturity, they are replaced by another bond, but the quotes appear in the same, uninterrupted time series. The change of the maturity results in a value jump.

Fortunately, the maturity of a benchmark bond is available from the data source. This information can be used for an immediate detection of and adaptation to the new definition. The program used for filtering benchmark bonds is continuously checking the maturity before passing the corresponding quote to the filter. As soon as it detects a change of the maturity, it calls a special function of the Olsen filter.

This new-definition function acts on all scalar filtering windows. For all scalar quotes in all scalar filtering windows, the stored age is incremented by $1+5/d$ days, where d is the quote density per day as computed by eq. 4.15; this leads to a higher expected volatility. At the same time, the valid-quote age Q_j of all these scalar quotes is incremented by 15; this can also lead to a higher expected volatility through eq. 4.19. By virtually pushing all old quotes into the past, the filter becomes more tolerant to the sudden value change to follow, without accepting bad outliers.

The new-definition function can be used for any new definition of any instrument, it is not restricted to benchmark bonds.

8.3.3 Regime changes and extreme market shocks

Regime changes and extreme market shocks imply sudden outbursts of quoted values and their volatility. The standard filter already has a means to deal with this phenomenon: the dilution mechanism explained in section 5.2.2. An alternative trust capital T_i'' of a scalar quote is computed in case of a value jump. If T_i'' fulfills the condition of eq. 5.38, the value jump is quickly accepted.

Unfortunately, this mechanism stops working if the market shock is extreme, i. e. the volatility becomes several orders of magnitude higher than the historical value, and if the volatility stays high also after the

initial jump. One remedy would be to choose a lower dilution factor μ. This cannot be a general solution as it would lead to accepting too many ordinary outliers.

Optimal filtering of regime changes and extreme market shocks is a subject of on-going research. Meanwhile, the filtering of regime changes and extreme market shocks may require human intervention.

8.4 Multivariate filtering – an idea for filtering sparse quotes

Multivariate filtering is *not* a part of the Olsen filter but rather a subject of unfinished research. In the Olsen filter, univariate filtering as described by section 7 is the highest algorithmic level.

Multivariate filtering requires a more complex and less modular software than univariate filtering – but it seems the only way to filter very sparse time series with very unreliable quotes. Some ideas of a possible implementation (based on research studies yet to be completed) are presented here.

In the financial markets, there is a quite stable structure of only slowly varying correlations between financial instruments. In risk management software packages, a large, regularly updated covariance matrix is used to keep track of these correlations.

Covariance matrices between financial instruments can also be applied in the filtering of sparse quotes. Univariate filtering methods are working well for dense quotes. If the density of quotes is low, the methods lose a large part of their power. When a new quote of a sparse series comes in, we have only few quotes to compare; these quotes can be quite old and thus not ideal for filtering. This is the place where some additional information from the covariance matrix becomes useful. This can technically be done in several ways.

The only method outlined here is the *artificial quote* method. If the sparse rate (e. g. in form of a middle price) is included in a covariance matrix that also covers some denser rates, we can generate some artificial quotes of the sparse series by exploiting the most recent quotes of the denser series and the covariance matrix. The expectation maximization (EM) algorithm [Morgan Guaranty, 1996] is a method to produce such artificial quotes; there are also some alternative methods. The best results can be expected if all the series included in the generation of an artificial quote are highly correlated or anticorrelated to the sparse series.

Artificial quotes may suffer from three uncertainties: (1) they have a stochastic error in the value because they are estimated, (2) there is an uncertainty in time due to asynchronicities in the quotes of the different financial instruments [Low et al., 1996], (3) only a part of the full quote is estimated from the covariance matrix (e. g. the middle price, whereas the bid-ask spread has to be vaguely estimated as an average of past values). Therefore, an additional rule may be helpful: using artificial quotes only if they are not too close to good quotes of the sparse series.

In some cases such as FX cross rates, we can simply use arbitrage conditions to construct an artificial quote (e. g. triangular currency arbitrage); a covariance matrix is not necessary there.

The following algorithmic steps are done in the artificial quote method:

- define a basket of time series which are denser than the sparse series and fairly well correlated or anticorrelated to it (radical version: a basket of *all* financial instruments);
- generate artificial quotes from the correlation matrix and mix them with the normal quotes of the sparse series, thus reinforcing the power of the univariate filtering algorithm;
- eliminate the artificial quotes from the *final* output of the filter (because a filter is not a gap-filler).

This algorithm has the advantage of leaving the univariate filtering algorithm almost unchanged; the multivariate element enters only in the technical form of quotes which already exists in univariate filtering.

The disadvantage lies in the fact that the artificial quotes may be of varying, uncertain quality (but better than none at all).

9 Initialization, termination, checkpointing

9.1 Filter initialization

Usually, there is a requested start time of the filtered data sequence. The filter needs to see some data before this start in order to work reliably: it needs a *build-up period.* A build-up is technically identical to a normal filter run; the only difference is that the filtering results of the individual quotes are not accessible to the end user.

The adaptive filter contains some elements based on iterations, mostly EMAs (exponential moving averages). These variables must be properly initialized. At the very beginning of a filtering session, the initial values are probably rather bad, so the filter has to run for a while to gradually improve them. Moreover, the windows are initialized to be empty, so the first quotes will also have shaky credibility values due to the insufficient window size.

Thus the filter initialization over a build-up period is absolutely necessary; it is not just an option.

The build-up period must have a sufficient size, the longer the better. A size of 2 weeks is a minimum; Olsen's recommendation is a a build-up period of at least three months. Extremely sparse data needs an even longer build-up, for obvious reasons.

Thus the filter is a tool for a sequential analysis, *not* for *random access.* If random access to filtered data is an issue, we recommend storing sequentially filtered data and its filtered results in a database and then retrieving data from there.

The timing of the build-up period is also essential. The best choice of the build-up period is the time interval immediately preceding the starting point from which the filtering results are needed. In this normal case,

1. the filter is started from scratch at start time *minus the build-up period*;
2. filtering results are ignored until the analysis reaches the start time;
3. normal filter operation starts at this start time.

This is not possible if the start time of the filter is close to the beginning of available data. In this exceptional case, we take the second best choice, a more complicated start-up procedure:

1. start the filter from the beginning of available data;
2. run it over a full build-up period;
3. ignore filtering results until the analysis reaches the end of build-up;
4. keep the values of all statistical variables (this is what the adaptive filter has learned from the data), either in memory or in a checkpoint file;
5. re-start the filter, again from the beginning of available data, but profiting from the statistical variables already built up;

6. ignore filtering results until the analysis reaches the desired start time of the filtering session;

7. reach the state of normal filter operation at this start time.

The statistical variables at the end of a build-up can be stored in a *checkpoint file*. This is explained in section 9.3. A filter reading from a well-chosen checkpoint file needs no build-up.

The initialization of a *new* financial instrument, such as a futures contract with a new expiry time, can be improved by starting its filter from a checkpoint file generated by the filter of a *similar* instrument such as a corresponding futures contract with an earlier expiry. This is important in real time when the new instrument does not yet have sufficient data for a regular build-up.

9.2 Filter termination

The termination of a filter run poses no problems. If the filter run is a build-up or could be a build-up for another run, a last checkpoint file should be written.

In the case of historical filtering, some quotes and their provisional results are still buffered in the filtering windows. A user may want to either continue filtering for a while to get the last quotes before termination time or simply flush the buffers.

9.3 Checkpointing

A checkpoint is a set of variable values that determine the state of a filter. All the statistical variables needed for adaptivity belong to this set. Checkpoints are written to files, in some time intervals, e. g. one checkpoint a day. The user program may also want to write additional checkpoints on request. Several checkpoints are kept for safety. Checkpoint files have to be managed by the user or the program using the filter. Their name or header should contain date and time of the last entered quote.

The main reason to introduce checkpointing is a gain of time. A checkpoint is a starting point for a quick start. Checkpointing is related to the build-up. A checkpoint written at a quote time exactly at or slightly before the desired starting point of the filter run can replace a new build-up. The filter simply reads the checkpoint file and initializes the variables accordingly. On the other hand, the build-up procedure can replace checkpointing and make it superfluous. This however implies that *every* filter run, also in a real-time application, must be preceded by a lengthy build-up. Only few users will accept such an amount of additional computation time. For most users, checkpointing is compulsory.

In filtering, there are two possible concepts of checkpointing:

1. **Strong checkpointing**: Storing all the variables that determine the state of the filter: not only the statistical variables but also the full-quote and scalar filtering windows with all their quotes and relevant results of these quotes. Strong checkpoints thus take a lot of storage space.

2. **Weak checkpointing**: storing only the statistical variables needed for adaptivity. (Olsen's weak checkpointing additionally includes some condensed information on a "last valid" quote).

Both are implemented in Olsen's filter. The Olsen data collector is using weak checkpointing for storage space reasons: the strong checkpoints of all the very many collected instruments would take an unwieldy amount of space. The storage size also depends on the format: human-readable ASCII or binary (which is more compact).

The filter is immediately and fully operational after starting from a strong checkpoint. When starting from a weak checkpoint, the filtering windows are empty, so the filter is not immediately operational. We need an additional "mini"-build-up to fill the windows with a sufficient amount of good quotes before the true filtering start, in order to prevent a bad start from outlier quotes. This is a disadvantage of weak checkpointing. Olsen's weak checkpoints therefore include a "last valid" quote which is the most recent among the good quotes from the full-quote filtering window at the time when the checkpoint was written. At start from a weak checkpoint, this last valid quote is used to initialize the filtering windows. More precisely, this quote (i. e. its scalar quotes) enters a special window of old valid quotes that was initially introduced for another purpose as explained in section 5.4.3. This is an acceptable, pragmatic solution not as good as strong checkpointing.

Another reason to introduce checkpointing is related to the information content of the checkpoint files. Human readers can obtain some insight into the filter status by looking at checkpoint files. To make this possible, checkpoints should be human-readable, either directly or through a tool in the case of encoded (binary) checkpoint files. In this sense, checkpoints are also a *diagnostic* tool, see section 10.1.

In summary, checkpointing is an almost absolute necessity. It can be avoided only under two conditions: (1) always tolerating slow initialization due to long build-up periods and (2) introducing diagnostic tools other than checkpointing.

10 Miscellaneous features of the filter

10.1 Diagnostics

Some users may want to know the current state of the filter while it is running in real time. Three possible solutions are proposed: (1) looking at the most recent regularly produced checkpoint file; (2) adding a feature to produce checkpoint files on demand; (3) a feature to produce diagnostic messages in text form on demand.

Olsen has implemented solution (1) with a tool to convert the weak checkpoints from binary to text format.

10.2 Statistics

Statistical information is useful to characterize the behavior of a filter. Even a user who does not understand the filter in its full complexity will still want to know simple statistical properties such as the *rejection rate*, the percentage of rejected quotes. The required statistical analysis is not necessarily identical to that internally used by the adaptive filter algorithm. Typical rejection rates are below 1% for major financial instruments and can be above 10% for some minor instruments with bad coverage and bad quoting discipline.

A tool to produce some standard, user-friendly statistical filtering information should be implemented as a part of the of the wider filtering environment. Such a tool cannot be part of the filter because it also has to manage the access to the data.

10.3 Testing

Testing plays a key role in the filter development. The correct operation has to be tested as for every piece of software. The role of testing extends far beyond this. Testing is also needed to fine-tune the many parameters of the filter. This has been done extensively by Olsen. The values recommended for all the many parameters of the filter in this document are results of our tests. A new filter developed accordingly will inherit our testing experience.

For the initial filter developer (as opposed to a user), filter testing is not trivial. A large part of it means comparing the results of tested filters to *human judgment.* This is hand work that can hardly be automated. The behavior of the filter has to be tested for very different data and error types. Examples of all the special errors of section 2.3 have to tested. Few automatic testing tools can help. From filtering statistics, we can identify instruments and periods with high rejection rates and uninterrupted series of rejected quotes. These often (but not always) indicate remaining problems of a certain filter version. Finally, we can compare two filter versions by looking at differences in the results, highlighting and studying only those quotes where a large difference in credibility was found. These methods have been used to fine-tune the Olsen filter.

Another part of testing was done in the framework of Olsen's own data collection which can be seen as a huge real-life testing environment of the filter. Olsen is using the collected data for research and its value-added information system. Remaining problems of Olsen's early filter prototypes quickly became apparent and could be solved by further research. This is the way some special features such as the filter of monotonic sequences (section 8.1) were added.

The world of financial data is almost infinite, evolving over time and changing its behavior, so we cannot exclude the possible occurrence of new, yet undetected filter problems. Olsen's research environment is a guarantee for finding solutions of new problems quickly. We recommend all filter users to report possible problems to Olsen.

10.4 User-defined filtering

Users have several ways to influence the behavior of the filter, if they wish so. They can choose the credibility threshold for accepted quotes different from the recommended value 0.499. They can change the recommended values of the many parameters of the filter. A higher value of ξ_0 in eq. 4.11, for example, will lead to a more tolerant filter.

In section 8.3, we have proposed some means to influence filtering through direct intervention.

11 Summary of filter parameters

The filter algorithm as a whole is complicated and depends on many configuration parameters. The definitions and explanations of all these parameters are scattered in the text. A central list of all parameters is therefore helpful; Table 5 is such a list.

The parameters are listed in the sequence of their appearance in the document. Some less important parameters have no symbol and appear directly as numbers in the text; nevertheless they have been included in Table 5. The important parameters (with their own symbols, of course) determine the character of the filter. Filter users may choose the parameter values in order to obtain a filter with properties suited to their needs.

Description of parameter	Symbol	Equation number
Range of mean x	$\Delta\vartheta_r$	4.3, 4.4
Parameters of $\Delta x^2_{\min}$ used in the level filter		(after eq. 4.7)
Critical deviation from mean x	ξ_0	4.8
Critical size of value change	ξ_0	4.11
Interaction range in change filter (normal value, special value for bid-ask spread)	ν	4.13
Range of quote density	$\Delta\vartheta_r$	4.15
Weight of new quote in quote density (normal value, special value for repeated quotes)	c_d	4.15
Range of short-term, standard and long-term volatility (v_{fast}, v, v_{slow})	$\Delta\vartheta_r$	4.16
Relative time interval offset for volatility	d_0	4.17
Absolute time interval offset for volatility	$\delta\vartheta_{\min}$	4.17
Relative limits of quote interval $\Delta\vartheta$ (upper, lower)		4.19
Weight of squared granule in volatility offset		4.21
Parameters used for volatility offset ε_0 for bid-ask spreads		(after eq. 4.21)
Exponent of $1 - D$ in the formula for the impact of quote diversity		4.24
Other parameters of the impact of quote diversity		4.24
Activity of active periods, for ϑ_k		4.25
Activity of inactive periods, for ϑ_k		4.25
Range of short-term volatility used for ϑ	$\Delta\vartheta_{\text{smooth}}$	4.28
Range of the variance of volatility fluctuations used for ϑ	$\Delta\vartheta_r$	4.29
Weight of the level filter	c_{level}	5.32
Trust capital dilution factor (normal value, special value at initialization from scratch)	μ	5.34 - 5.36
Window size parameter	W	5.43
Critical credibility for statistics update (normal value, special value at initialization from scratch)	C_{crit}	(section 5.4.2)
Lower limit of allowed domain (prices, FX forwards, interest rates)	$p_{\min}$	6.44 (+ section 6.2)
Factor in transformation of bid-ask spreads		6.45
Maximum quote interval size in decimal error		(section 7.1)
Parameters of trust capital test for accepting a decimal error		7.46
Parameters of trust capital test for terminating a decimal error		7.47
Trust capital limits of old scale factor (factor 10 or 0.1, other factor)		7.48
Trust capital advantage of new scale factor (factor 10 or 0.1, other factor)		7.49
Standard credibility threshold for accepting a quote		(sections 7.3, 10.4)
Many parameters of the filter of monotonic quotes		8.54 - 8.60
Memory (related to sample range) of moments	μ_m	8.64
Parameters of the maximum trust capital of repeated quotes		8.68

Table 5: List of the filter parameters.

12 Conclusion

The Olsen filter described in this document is a powerful tool to analyze and clean up data in financial time series. The credibility of quotes is analyzed; bad quotes and outliers can be removed. The filter works for all data frequencies, from high-frequency to sparse data.

The described filtering algorithm is adaptive to the filtered financial instrument: it learns from the data. While the basic structure of the algorithm is clear and simple (Table 1), the rich and diverse nature of financial data, its quoting styles and errors requires the addition of some more complex filtering elements.

The Olsen filter has been checked and fine-tuned in many tests. It is running in practice, in Olsen's own

data collection which is the basis of research as well as commercial services. In this real-life test, the Olsen filter has shown its good performance.

There are few remaining problems. Most are related to disruptive events: events that rapidly and radically change the behavior of a time series. In this field, filter research at Olsen is continued and will lead to improved future versions of the filter.

References

[Ballocchi, 1996] **Ballocchi G.**, 1996, *Interbank money market rate statistics: Results for the forecaster optimization,* Internal document GBA.1996-03-01, Olsen & Associates, Seefeldstrasse 233, 8008 Zürich, Switzerland.

[Dacorogna et al., 1993] **Dacorogna M. M., Müller U. A., Nagler R. J., Olsen R. B., and Pictet O. V.**, 1993, *A geographical model for the daily and weekly seasonal volatility in the FX market,* Journal of International Money and Finance, **12**(4), 413–438.

[Dacorogna et al., 1998] **Dacorogna M. M., Pictet O. V., Müller U. A., and de Vries C. G.**, 1998, *The distribution of extremal foreign exchange rate returns in extremely large data sets,* Internal document UAM.1992-10-22, Olsen & Associates, Seefeldstrasse 233, 8008 Zürich, Switzerland.

[Fowler and Scott, 1997] **Fowler M. and Scott K.**, 1997, *UML Distilled: Applying the Standard Object Modeling Language,* Addison-Wesley.

[Guillaume et al., 1997] **Guillaume D. M., Dacorogna M. M., Davé R. D., Müller U. A., Olsen R. B., and Pictet O. V.**, 1997, *From the bird's eye to the microscope: A survey of new stylized facts of the intra-daily foreign exchange markets,* Finance and Stochastics, **1**, 95–129.

[Low et al., 1996] **Low A., Muthuswamy J., and Sarkar S.**, 1996, *Time variation in the correlation structure of exchange rates: high frequency analyses,* Proceedings of the Third International Conference on Forecasting Financial Markets, London, England, March 27-29, 1996, **1**, 1–24.

[Morgan Guaranty, 1996] **Morgan Guaranty**, 1996, *RiskMetrics^{TM} – Technical Document,* Morgan Guaranty Trust Company of New York, New York, 4th edition.

[Müller, 1989] **Müller U. A.**, 1989, *Price filtering – a survey of the filters implemented and used by O&A,* Internal document UAM.1989-05-08, Olsen & Associates, Seefeldstrasse 233, 8008 Zürich, Switzerland.

[Müller, 1991] **Müller U. A.**, 1991, *Specially weighted moving averages with repeated application of the EMA operator,* Internal document UAM.1991-10-14, Olsen & Associates, Seefeldstrasse 233, 8008 Zürich, Switzerland.

[Müller et al., 1998] **Müller U. A., Dacorogna M. M., and Pictet O. V.**, 1998, *Heavy tails in high-frequency financial data,* published in the book "A practical guide to heavy tails: Statistical Techniques for Analysing Heavy Tailed Distributions", edited by Robert J. Adler, Raisa E. Feldman and Murad S. Taqqu and published by Birkhäuser, Boston 1998, 55–77.

[Müller and Sgier, 1992] **Müller U. A. and Sgier R. G.**, 1992, *Statistical analysis of intra-day bid-ask spreads in the foreign exchange market,* Internal document UAM.1992-04-10, Olsen & Associates, Seefeldstrasse 233, 8008 Zürich, Switzerland.

[Müller and Zumbach, 1997] **Müller U. A. and Zumbach G. O.**, 1997, *Specification of an adaptive filter for time series of financial quotes*, Internal document UAM.1997-01-28, Olsen & Associates, Seefeldstrasse 233, 8008 Zürich, Switzerland.

[Rukkers, 1991] **Rukkers J. M.**, 1991, *Decimal error filtering*, Internal document JMR.1991-05-07, Olsen & Associates, Seefeldstrasse 233, 8008 Zürich, Switzerland.

[Zimmermann, 1985] **Zimmermann H. J.**, 1985, *Fuzzy Set Theory – and Its Applications*, Kluwer - Nijhoff Publishing, Boston - Dordrecht - Lancaster.

[Zumbach and Müller, 2001] **Zumbach G. O. and Müller U. A.**, 2001, *Operators on inhomogeneous time series*, International Journal of Theoretical and Applied Finance, **4**(1), 147–178.