Time Averages and the Totals in the Billing of Natural Gas

Mark van den Bosch[1], Alessandro Di Bucchianico[2], Erwin Luesink[3], Julie Merten[4], Vivi Rottschäfer[1] and Jelle van der Voort[1]

Abstract

We develop a model that allows us to make a quantitative statement about the effect and accuracy of calculating the total energy E in the billing of natural gas, making use of time-averaged data for the volume V_i and calorific value H_i. This is achieved for the current approach performed by VSL and the energy producer, in which now carefully the dependencies between successive measurements are taken into account. Furthermore, we introduce a new methodology that aims to solve the drawbacks of the current averaging procedure. This method concerns filtering, coarse-graining, and segmentation.

Keywords: metrology, gas flow metering, time series, averaging over time, statistically stationary, filter, change detection, segmentation.

1 Introduction

At the Study Group Mathematics with Industry held at Vrije Universiteit of Amsterdam in the week of 29 January – 2 February, a problem was posed by VSL. VSL is the Dutch Metrology Institute, and it makes measurement results directly traceable to international standards. The fiscal metering of natural gas is often performed using a flow meter and a gas chromatograph. The flow meter measures the volume flow rate of the gas passing the metering station. The gas chromatograph measures the composition from which the calorific value is determined. Both are measured at regular intervals in time, but not necessarily at the same intervals. It can be that the gas chromatograph produces a result every 5 minutes, whereas the flow meter submits a result every minute.

From this, the energy (for Standardization, 2018) can be determined as $E = HV$, where H denotes the calorific value of the gas (for Standardization, 2016) and V the volume at reference conditions (e.g., 0 degrees Celsius and 101.325 kPa). The uncertainty of the total

[1]Leiden University, the Netherlands
[2]Eindhoven Unversity of Technology, the Netherlands
[3]Twente Unversity of Technology, the Netherlands
[4]University of Groningen, the Netherlands

energy delivered over a certain time is an important quantity for operating the gas grid. This will ensure energy producers can feed the grid and users can draw gas from the grid.

The calorific value H and volume V are often averages over time (e.g., hourly average). These averages are usually calculated as the arithmetic mean of a series of measurement results. However, either or both quantities may change during this time period. These changes do not follow any particular pattern; they can be a steady increase/decrease, but depending on supply and demand, they can also have any other pattern. More importantly, these measurement results are not independent since these are subsequent time points.

VSL wanted to know the best averages for aggregating the measurement data and to evaluate the uncertainty of the mean. This problem is connected to WP 4 of the Euramet project Meth4H2. The overall objective of the project is to develop further and integrate the metrology necessary to support the entire supply chain of hydrogen, from production to storage and end-use. The project will disseminate metrological traceability to the field so that measurement results become fit-for-purpose with respect to health, safety, environmental, and fiscal purposes.

Section 2 gives a more detailed description of the problem under study. In particular, we provide details of the current averaging approach performed by VSL and the energy producer. Then, in Section 3, we give a mathematical description of the current approach and answer the question of how to correctly compute the total uncertainty of the total energy. Subsequently, in Section 4, we discuss the effect and drawbacks of the arithmetic mean and introduce a more generalized framework of averaging, which we refer to as filtering and coarse-graining. In Section 5, we show how autoregressive models can be used after one has partitioned a time series into stationary and transitional segments. Ultimately, we conclude and discuss in Section 6.

2 Problem statement and approach

In this section, we first describe the current approach to fiscal metering taken by the energy producer, which is based on taking averages. Based on this, we formulate our research questions. We conclude this section with an improved approach that addresses issues with the current methodology.

2.1 Overview of the current approach

The energy producer measures both the calorific value and volume at a metering station, see Figure 1 (left). The calorific value and the volume are measured at certain, not necessarily the same, time intervals. As illustrated in Figure 1 (right), the energy producer calculates averages and standard deviations of these measurements over fixed intervals, e.g., 15 minutes. The aggregated data is then sent to VSL. We denote these averaged values of calorific value and volume by H_i and V_i, respectively. From this, the total energy can then be determined by

$$E = \sum_i H_i V_i. \tag{1}$$

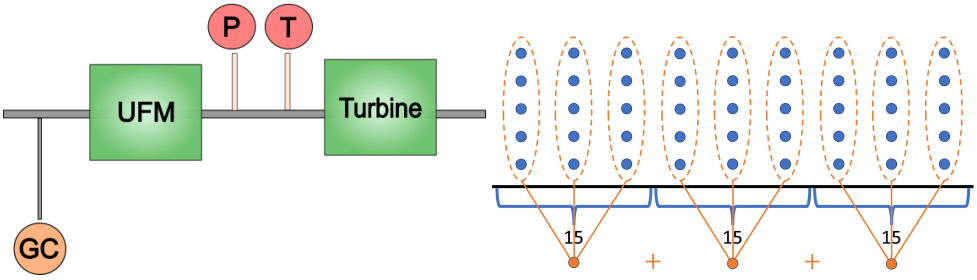

Figure 1: Schematic representation of the measurements at a metering station of an energy producer (left). A simplified schematic overview of the current approach to averaging time-series data (right), and we refer to Section 3.1 for more information.

In the past, the total uncertainty of total energy was computed by assuming that all data points were uncorrelated. This assumption is false, of course, as it describes continuous dynamics. Apart from the loss of information caused by aggregating raw measurement values into averages and standard deviations, a significant problem is that this method *a priori* lacks a clear way to accurately assess the total uncertainty of the computed total energy. This is because we must not ignore the dependencies between successive measurements. The solution to this issue is nontrivial and subject to Section 3.

2.2 Research questions

The issues indicated above led us to the following research questions:

1. What is the effect of the averaging on the uncertainty in fiscal metering?
2. How can the uncertainty be correctly calculated, taking into account dependencies?

In the next subsection, we outline our approach to tackle these research questions.

2.3 Overview and new methodology

One of our main goals is to outline a model in which we can quantify the effect of averaging and deal with in-time correlated data. The basic step in our approach is to describe both the measurements at the metering stations done by the energy producer, i.e., the raw data, as well as the subsequent aggregation of the data by a model. The model for the raw measurements has to take into account the dependencies between the successive measurements. This is done in Section 3.

One way to approach the above is to use time series models from the Box-Jenkins approach. In particular, we look at AR(1) and AR(2) processes because, on the one hand, these models (especially AR(2)) work surprisingly well in several industrial contacts, and on the other hand, they allow for explicit calculations (see, e.g., textbooks like Chatfield and Xing (2019), Cowpertwait and Metcalfe (2009) and Shumway and Stoffer (2000)). Explicit calculations

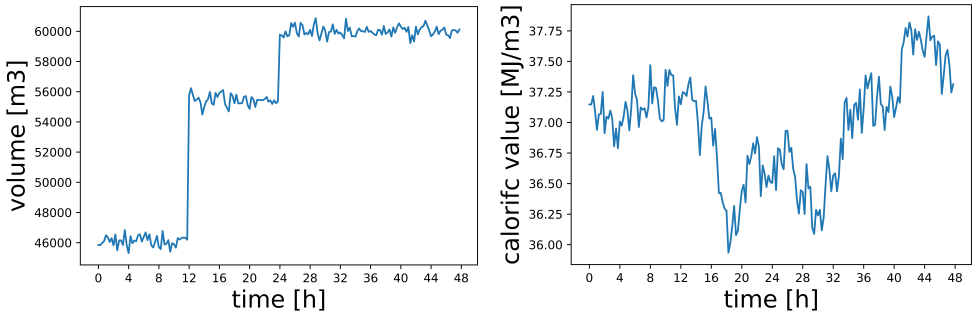

Figure 2: Synthetic data of the calculated averages of volume V_i (left) and calorific value H_i (right) over fixed intervals of 15 minutes. Plots are produced by `SWI_VSL_main.py` and are an improvement of the synthetic data set provided (see Appendix E) in the sense that successive measurements are correlated. Note that the time series of the volume is constructed by means of three different AR(1) processes and the calorific value by only one AR(1) process. See Section 3.3 for more information.

are important because even if the models are too simple for reality, they allow us to use these models as test benches to compare the current approach with new approaches. Using mathematical models not only allows us to incorporate dependencies but also to explore refined ways of averaging based on ideas of signal processing. This is subject to Section 3.3 and Section 4.

It must be pointed out that a data set like in Figure 2 (left) cannot be modeled with only one autoregressive model. The time series must be partitioned into several segments before we can argue that the data can be seen as a realization of an AR process. This is explained in Section 5. Note that in most sections, we assume without loss of generality that we are considering a period of time in which the time series can be seen as a single AR process.

3 Modelling the current approach

In this section, we formulate a mathematical model corresponding to the current averaging method employed by VSL and the energy produced. In Section 3.1, we provide the current averaging method and derive an upper bound for the uncertainty in the total energy. We then extend the work in Section 3.2 by explicitly considering the correlations present in the data. This results in a valid approximation for the desired uncertainty. At last, in Section 3.3, we consider the case where more information on the underlying processes is available. Under the assumption that volume and the calorific value can be modeled by means of AR processes, which is likely to hold (recall Section 2.3), we obtain an analytical expression for the total uncertainty in terms of knowns without the need for approximations.

We refer to Crowder et al. (2020) and Pavese and Forbes (2008) for suitable textbooks on

statistics and modeling in metrology from a mathematical perspective.

3.1 A model corresponding to the current approach and an upper bound for the total uncertainty

In this section, we derive a model which explains the current approach in mathematical terms step by step. Also, we obtain an upper bound for the total uncertainty. As illustrated in Figure 1, we need to distinguish between different layers to model the measurement operation and the averaging process by the energy producer accurately.

In the first layer, K measurements are repeated in a very short time frame to obtain estimates of the precision of the raw measurements (in virtue of repeatability). These repeated measurements are averaged, and the standard deviation is computed. We assume a fixed time Δ between the subsequent averaged measurements. In the second and third layers, we consider two additional processes of averaging. We introduce a clear notation before we elaborate on the specific operations within these layers.

We denote the individual measurements by $W_{i,j,k}$ (volume) and $D_{i,j,k}$ (calorific value), respectively. Here, i is the number of the time interval under consideration, j denotes the index of the averaged measurements in one of the time intervals of length Δ, and k is the index of the repeated measurement at a time instance. Therefore, if we start at $t = 0$, then the repeated measurements $W_{i,j,k}$ and $D_{i,j,k}$ for $k = 1, \ldots, K$ take place at time $(im + j)\Delta$, where m is the fixed amount of averaged measurement instances within an arbitrary time interval. We assume additive measurement errors $\varepsilon_{i,j,k}$ (volume) and $\delta_{i,j,k}$ (calorific value).

In mathematical terms, we consider

$$D_{i,j,k} = \eta_{i,j} + \delta_{i,j,k}, \quad \delta_{i,j,k} \overset{\text{iid}}{\sim} \mathcal{N}(0, \tau_{i,j}^2), \quad W_{i,j,k} = \mu_{i,j} + \varepsilon_{i,j,k}, \quad \varepsilon_{i,j,k} \overset{\text{iid}}{\sim} \mathcal{N}(0, \sigma_{i,j}^2). \tag{2}$$

Here, the values $\mu_{i,j}$ and $\eta_{i,j}$ denote the real values of the volume and the calorific value, respectively, at time $(im+j)\Delta$. We like to point out that the normal distribution can be replaced by any distribution with mean zero and finite variance.

The statistics obtained from the individual measurements at time points $(im + j)\Delta$ are the means $Z_{i,j}$ (volume) and $Y_{i,j}$ (calorific value) and the sample variances $S^2_{Z;i,j}$ and $S^2_{Y;i,j}$. That is,

$$\begin{aligned} Y_{i,j} &= \frac{1}{K}\sum_{k=1}^{K} D_{i,j,k}, \quad S^2_{Y;i,j} = \frac{1}{K}\sum_{k=1}^{K}(D_{i,j,k} - Y_{i,j})^2, \\ Z_{i,j} &= \frac{1}{K}\sum_{k=1}^{K} W_{i,j,k}, \quad S^2_{Z;i,j} = \frac{1}{K}\sum_{k=1}^{K}(W_{i,j,k} - Z_{i,j})^2. \end{aligned} \tag{3}$$

These variables represent the data that the energy producer has available.

Subsequently, the energy producer aggregates m of such computed statistics into averaged values V_i and H_i of the volume and calorific value, respectively. We capture this operation

within the second layer. We represent the uncertainty by the associated sample variances:

$$\begin{aligned} H_i &= \frac{1}{m}\sum_{j=1}^{m} Y_{ij}, \quad S^2_{H;i} = \frac{1}{mK}\sum_{j=1}^{m}\sum_{k=1}^{K}(D_{i,j,k} - H_i)^2, \\ V_i &= \frac{1}{m}\sum_{j=1}^{m} Z_{ij}, \quad S^2_{V;i} = \frac{1}{mK}\sum_{j=1}^{m}\sum_{k=1}^{K}(W_{i,j,k} - V_i)^2. \end{aligned} \tag{4}$$

These sample variances can be calculated by adding the sample variances of the averaged repeated measurements and the sample variance of this layer relative to the previous layer, see Appendix A. More specifically, we have the decompositions

$$\begin{aligned} S^2_{H;i} &= \underbrace{\frac{1}{mK}\sum_{j=1}^{m}\sum_{k=1}^{K}(D_{i,j,k} - H_i)^2}_{\substack{\text{overall uncertainty in } i\text{-th} \\ \text{interval of length } m\Delta}} = \underbrace{\frac{1}{m}\sum_{j=1}^{m} S^2_{Y;i,j}}_{\substack{\text{uncertainty in} \\ \text{measurements}}} + \underbrace{\frac{1}{m}\sum_{j=1}^{m}(Y_{i,j} - H_i)^2}_{\text{fluctuations in time}}, \\ S^2_{V;i} &= \overbrace{\frac{1}{mK}\sum_{j=1}^{m}\sum_{k=1}^{K}(W_{i,j,k} - V_i)^2}^{} = \overbrace{\frac{1}{m}\sum_{j=1}^{m} S^2_{Z;i,j}}^{} + \overbrace{\frac{1}{m}\sum_{j=1}^{m}(Z_{i,j} - V_i)^2}^{}. \end{aligned} \tag{5}$$

The energy (product of calorific value and volume) within time interval i is denoted by

$$E_i = H_i V_i. \tag{6}$$

An appropriate estimator Γ_i for the variance of E_i is obtained by exploiting identity (37) in Appendix A, which yields

$$\Gamma_i = S^2_{H;i} S^2_{V;i} + H_i^2 S^2_{V;i} + V_i^2 S^2_{H;i} \approx \mathrm{Var}(E_i). \tag{7}$$

Alternatively, the product of the calorific value and volume could also have been considered one layer earlier, thus by defining $E_{i,j} = Y_{i,j} Z_{i,j}$. This choice would result in a smaller variance, which illustrates that our chosen approach is less optimal in that sense; we simply obtain an upper bound for the uncertainty. However, an earlier product results in non-explicit expressions for the uncertainty terms, i.e., there is no equivalent to the useful decompositions as in (5).

The total energy, as determined in the third and last layer simply reads

$$T = m\Delta \sum_{i=1}^{n} E_i, \tag{8}$$

together with its variance, i.e., the total uncertainty squared,

$$\mathrm{Var}(T) = m^2\Delta^2 \left[\sum_{i=1}^{n} \mathrm{Var}(E_i) + 2 \sum_{1 \le i < i' \le n} \mathrm{Cov}(E_i, E_{i'})\right]. \tag{9}$$

Now, we can use the Cauchy-Schwarz inequality $|\mathrm{Cov}(E_i, E_{i'})| \leq \sqrt{\mathrm{Var}(E_i)}\,\sqrt{\mathrm{Var}(E_{i'})}$ to conclude that

$$\mathrm{Var}(T) \leq m^2\Delta^2 \left[\sum_{i=1}^{n} \sqrt{\mathrm{Var}(E_i)}\right]^2. \tag{10}$$

As a last step, we note that Γ_i in (7) is an accurate estimator of $\mathrm{Var}(E_i)$, which yields

$$\text{Total uncertainty} \lesssim m\Delta \left[\sum_{i=1}^{n} \sqrt{\Gamma_i}\right], \tag{11}$$

where $\lesssim$ is an approximate inequality since the right-hand side is an estimator. The inequality in (11) gives the desired upper bound for the uncertainty in the total energy.

Discussion Under the assumption that $\mu_{i,j}$ and $\eta_{i,j}$ are uncorrelated, the expression in (9) simplifies and we obtain

$$\mathrm{Var}(T) = m^2\Delta^2 \sum_{i=1}^{n} \mathrm{Var}(E_i). \tag{12}$$

However, from a physical point of view, this assumption seems to be false. Indeed, it is clear that the volume of the gas at a certain point in time is influenced by its past, and likewise for the calorific value. We shall continue in the next section with the understanding that the data is related in order to approximate (9) accurately.

Furthermore, suppose that we have $\mathrm{Var}(E_i) \approx N$ for all $1 \leq i \leq n$, for some $N > 0$. Then, upon ignoring the covariances, we obtain

$$\text{Total uncertainty} \lesssim m\Delta\sqrt{nN}, \tag{13}$$

which is generally speaking incorrect. Using the upper bound (11) results into

$$\text{Total uncertainty} \lesssim m\Delta n\sqrt{N} = \text{Total time} \cdot \sqrt{N}, \tag{14}$$

where Total time $= nm\Delta$. The increase on the right-hand side is due to the covariances, and the truth is expected to be somewhere in the middle. We study this in future sections.

3.2 A more accurate approximation of the total uncertainty using the fact that the data is correlated

In this section, we no longer assume the data to be uncorrelated. The volume measurements are assumed to be correlated, and the same holds for the calorific value measurements. This highlights the importance of the inclusion of the second term within (9) as a contribution of these correlations to the uncertainty in the total energy. Therefore, from this point onwards, we consider $\mu_{i,j}$ and $\eta_{i,j}$ to be realisations of the correlated stochastic processes $(\eta(t))_{t\geq 0}$ and $(\mu(t))_{t\geq 0}$, respectively.

By invoking $E_i = H_i V_i$ we can rewrite the covariance-term in (9) as follows

$$\begin{aligned}\mathrm{Cov}(E_i, E_{i'}) = \mathrm{Cov}(H_i, H_{i'})\mathrm{Cov}(V_i, V_{i'}) + \mathbb{E}[H_i]\mathbb{E}[H_{i'}]\mathrm{Cov}(V_i, V_{i'}) \\ + \mathbb{E}[V_i]\mathbb{E}[V_{i'}]\mathrm{Cov}(H_i, H_{i'}).\end{aligned} \tag{15}$$

We can approximate the means by the single observations in the same manner as done in Appendix A. Regarding the covariance terms, we have

$$\mathrm{Cov}(H_i, H_{i'}) = \frac{1}{m^2}\sum_{\ell=1}^{m}\sum_{\ell'=1}^{m}\mathrm{Cov}(Y_{i\ell}, Y_{i'\ell'}) = \frac{1}{m^2}\sum_{\ell=1}^{m}\sum_{\ell'=1}^{m}\mathrm{Cov}(\eta_{i\ell}, \eta_{i'\ell'}), \tag{16}$$

which follows from the bi-linearity and the independence of the measurement errors. Likewise, the following holds

$$\mathrm{Cov}(V_i, V_{i'}) = \frac{1}{m^2}\sum_{\ell=1}^{m}\sum_{\ell'=1}^{m}\mathrm{Cov}(Z_{i\ell}, Z_{i'\ell'}) = \frac{1}{m^2}\sum_{\ell=1}^{m}\sum_{\ell'=1}^{m}\mathrm{Cov}(\mu_{i\ell}, \mu_{i'\ell'}). \tag{17}$$

Without more knowledge of the statistical properties of the processes η_t and μ_t it is difficult to find a meaningful, analytical expression in terms of knowns for $\mathrm{Cov}(H_i, H_{i'})$ and $\mathrm{Cov}(V_i, V_{i'})$. However, we can approximate the covariances $\mathrm{Cov}(Y_{i\ell}, Y_{i'\ell'})$ and $\mathrm{Cov}(Z_{i\ell}, Z_{i'\ell'})$ by introducing the sample covariances

$$\begin{aligned}S_{H;i,i'} &= \frac{1}{m}\sum_{j=1}^{m}\sum_{j'=1}^{m}(Y_{ij} - H_i)(Y_{i'j'} - H_{i'}), \\ S_{V;i,i'} &= \frac{1}{m}\sum_{j=1}^{m}\sum_{j'=1}^{m}(Z_{ij} - V_i)(Z_{i'j'} - V_{i'}).\end{aligned} \tag{18}$$

These terms can be directly approximated by evaluation using the available data. To sum up, we can approximate $\mathrm{Cov}(E_i, E_{i'})$ by

$$\Gamma_{ij} = S_{H;i,i'}S_{V;i,i'} + H_i H_{i'} S_{V;i,i'} + V_i V_{i'} S_{H;i,i'} \approx \mathrm{Cov}(E_i, E_{i'}). \tag{19}$$

Together with the estimators Γ_i of $\mathrm{Var}(E_i)$, this gives us an approximation of the uncertainty of the total energy. Importantly, note that $\Gamma_{ii} \neq \Gamma_i$, for all i, because $S_{*;i,i} \neq S^2_{*;i}$ with $* \in \{H, V\}$.

In conclusion, we have determined a valid approximation of the total uncertainty by direct implementation of the data. If more statistical properties on the underlying processes η_t and μ_t are assumed to be known, we can say more about the uncertainty without the need for appropriate approximations. We will discuss this rationale in more detail in the next section.

3.3 An underlying stationary process

Our approach is based on the idea that we can segment the time series so that all observations within a segment behave similarly in a statistical sense. This is called stationarity in the statistical literature. Please note that unlike in other mathematical communities where

stationarity basically means constant behavior, it is meant here that the statistical behavior (means and covariances) does not change over time. In other words, the mean is constant, and the covariance between the two observations depends only on the time difference between them. Since covariances are related to second-order moments, this type of stationarity is often called second-order stationarity.

In this section, we consider $\mu_{i,j}$ and $\eta_{i,j}$ to be realisations of specific stationary stochastic processes $(\eta(t))_{t\geq 0}$ and $(\mu(t))_{t\geq 0}$, respectively. Concerning the real measurement structure, an AR(1) or AR(2) process could, for example, be suitable, as more commonly done in practice.

Under these assumptions, we have expressions for the mean, variance, and covariances of the underlying processes $(\eta(t))_{t\geq 0}$ and $(\mu(t))_{t\geq 0}$ in terms of one or more unknown model parameters. We can find accurate estimations of these key parameters by implementation of the measurement data available

Consequently, we are able to evaluate the uncertainty in the total energy directly. Indeed, the expressions for $\mathrm{Var}(E_i)$ and $\mathrm{Cov}(E_i, E_{i'})$ within (9) can now be found analytically. For example, for the latter, this can be done by employing equations (15)-(17) and filling in the covariances corresponding to $(\eta(t))_{t\geq 0}$ and $(\mu(t))_{t\geq 0}$. See Appendix B for such statistics for AR(1) and AR(2) processes.

Although analytically advantageous, we have to be able to confidently 1) identify the underlying process and 2) estimate the key parameters.

3.4 Simulations

Let us introduce the notion of relative error, which is given by

$$\text{Relative error} = \frac{\text{Absolute error}}{\text{Total value}} \times 100\% \approx \frac{\text{Total uncertainty}}{\text{Total energy}} \times 100\%. \tag{20}$$

Exploiting the upper bound in (11) yields

$$\text{Relative error} \lesssim \frac{m\Delta}{E} \sum_{i=1}^{n} \sqrt{\Gamma_i} \times 100\%. \tag{21}$$

For the synthetic data as in Figure 2, it turns out that the upper bound on the total uncertainty of the total energy results in a suitable value, namely: Relative error $\lesssim 3.2\%$.

Note that if one assumes that the covariance structure can be ignored entirely (which is false!), and subsequently exploit formula (9) with $\mathrm{Cov}(E_i, E_{i'}) = 0$ for all $1 \leq i < i' \leq n$, one would obtain a relative error which is at least a factor 10 smaller for this example. This motivates that, indeed, there was a need for a better understanding of the total uncertainty. One could improve the approximation of the relative error by the considerations in Section 3.2 and Section 3.3, and see also Section 6 for further discussion.

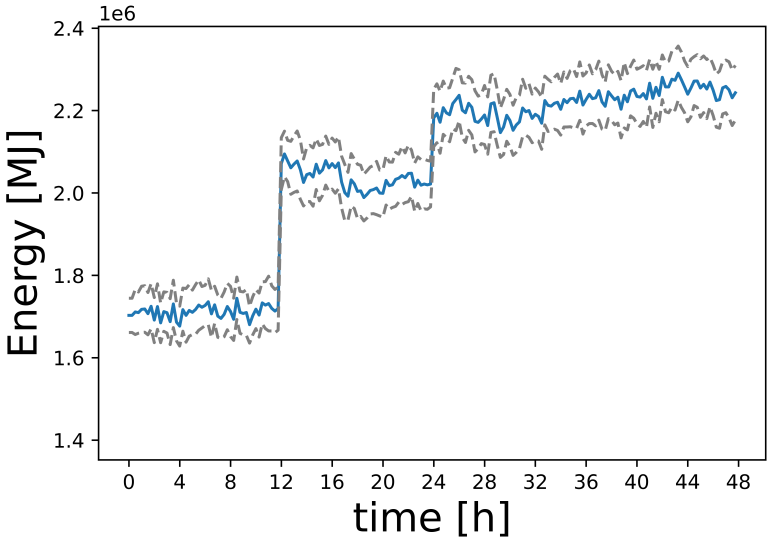

Figure 3: An illustration of $E_i = H_i V_i$ with respect to time instant i (blue), where H_i and V_i are as in Figure 2. The gray dashed lines are obtained by determining $E_i \pm \sqrt{\Gamma_i}$ for all time instants i. On account of formula (21), we obtain the following: Relative error $\lesssim 3.2\%$.

4 The effect of averaging and towards a general approach

The goal of this section is to study the implications of averaging, as performed by the energy producer (recall Section 3.1), on the total uncertainty and on how to improve it. We set out a more general approach than just taking the average, which is achieved by means of exploiting existing theory on filtering and coarse-graining for time series. In Section 4.1, we focus on the drawbacks that come with simple averaging. In Section 4.2 we introduce the concepts of filtering and coarse-graining (also referred to as downsampling), respectively, and find several useful analytical expressions. It is assumed throughout that we are dealing with a statistically stationary process, as justified in Section 3.3.

4.1 Drawbacks of the arithmetic mean

In this section, we describe the drawbacks of the arithmetic mean as a method to average the data. Averaging is a mathematical procedure that involves taking an integral (in a certain sense, see also Section 4.2). In probability theory, the computation of means and variances also involves integration. The integral is mathematically defined via a limiting procedure that involves Riemann sums and partitions. Different choices of sequences of partitions lead to different ways of converging towards the true integral, see Appendix C.1. Furthermore, only specific choices of Riemann sums and partitions respect important properties of the true integral at the continuous level, such as the chain rule. We are interested in seeing how the

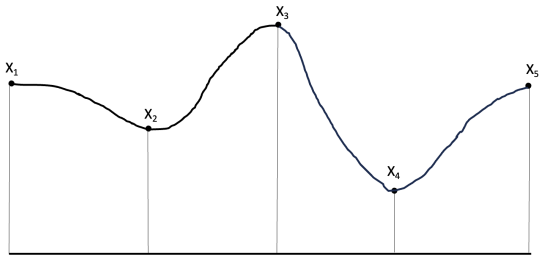

Figure 4: An illustration of a smooth curve that goes through the five points X_1, X_2, X_3, X_4, X_5.

mean and the variance are affected by the choices of the number of elements in the partition and the choice of Riemann sums.

Let us consider a simple example to get a feeling for the effects of averaging on means and variances. Suppose X_1, X_2, X_3, X_4, X_5 i.i.d. with $\mathrm{Var}(X_i) = \sigma^2$. We compare approximating the area under the curve by first dividing the area into two halves and then by performing the same computation while dividing the area into four parts (see Figure 4). Observe that

$$\mathrm{Var}\left(2 \cdot \frac{X_1 + X_2 + X_3}{3} + 2 \cdot \frac{X_3 + X_4 + X_5}{3}\right) = \frac{64}{18}\sigma^2 \approx 3.56 \cdot \sigma^2 \tag{22}$$

and

$$\mathrm{Var}\left(\frac{X_1 + X_2}{2} + \frac{X_2 + X_3}{2} + \frac{X_3 + X_4}{2} + \frac{X_4 + X_5}{2}\right) = \frac{63}{18}\sigma^2 \approx 3.5 \cdot \sigma^2. \tag{23}$$

The major conclusion is that although averaging on a finer scale improves accuracy, it also has a negative effect on precision (i.e., higher variance).

4.2 Generalizing the averaging method: filtering and coarse-graining

We now describe some generalizations of the arithmetic mean. The generalization uses methods of signal processing, specifically filters. In signal processing, a filter is some device or process that removes unwanted features from a signal, see Vetterli et al. (2014), de Cheveigné and Nelken (2019), and references therein. In the context of the problem here, we want to reduce the uncertainty in the value of the energy. By taking into account more measurements over a longer time period, it is expected that the uncertainty will be reduced. The filtering theory will quantify exactly by how much.

Note that the below can be used to replace the following quantities: sample mean (4), sample variance (5), and sample covariance (18). Any other formula in Section 3 remains valid. In particular, one could now invoke `SWI_VSL_main.py` and `SWI_VSL_functions.py` to do the same numerical analysis as in Section 3.4, but now for any possible filter.

4.2.1 Filtering and its characteristics

In this section, we describe how filtering works. We consider a discrete filtering process. Let X_t be a stationary time series with mean μ_X and autocovariance function $\gamma_X(h)$. Furthermore, we have a transfer function $\psi(j)$ corresponding to the filtering process under consideration. The filtered process can be defined as

$$Y_t = \sum_{j=-\infty}^{\infty} \psi(j) X_{t-j}, \tag{24}$$

where ψ only takes finitely many non-zero values. We refer to Appendix C.2 for its continuous analog. The following link provides an extensive list of possible filters suitable for a Python implementation:

https://docs.scipy.org/doc/scipy/reference/signal.windows.html

Regarding the mean of the filtered process, we observe the identity

$$\mu_Y := \mathbb{E}(Y_t) = \sum_{j=-\infty}^{\infty} \psi(j) \mathbb{E}\left(X_{t-j}\right) = \mu_X \sum_{i=-\infty}^{\infty} \psi(j) =: \mu_X. \tag{25}$$

Here, we used the stationarity of the original time series X_t and one of the basic properties of the transfer function ψ.

We calculate the autocovariance function of the filtered process as follows

$$\begin{aligned} \gamma_Y(h) &= \mathrm{Cov}(Y_t, Y_{t+h}) = \mathrm{Cov}\left(\sum_{i=-\infty}^{\infty} \psi(i) X_{t-i}, \sum_{j=-\infty}^{\infty} \psi(j) X_{t+h-j} \right) \\ &= \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} \psi(i)\psi(j) \mathrm{Cov}\left(X_{t-i}, X_{t+h-j}\right) = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} \psi(i)\psi(j) \mathrm{Cov}\left(X_t, X_{t+h+i-j}\right) \\ &= \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} \psi(i)\psi(j)\gamma_X(h+i-j). \end{aligned} \tag{26}$$

From the expressions (25) and (26), we can conclude that the stationarity of the original time series X_t transfers to the filtered process Y_t.

Without further assumptions, the analysis is limited. In Section 5, we exploit the strength of numerics and compare the effect of some different filters. Below, we illustrate the power of the analytical expressions above.

Example. We consider a simple averaging filter. For each point, the $2n$ nearest neighboring points are taken into account to determine the mean. This corresponds to the following filter function:

$$\psi(j) = \begin{cases} \frac{1}{2n+1} & -n \le j \le n, \\ 0 & \text{otherwise.} \end{cases} \tag{27}$$

This gives the filtered process

$$Y_t = \sum_{j=-n}^{n} \frac{1}{2n+1} X_{t-j} = \frac{1}{2n+1} \left(X_{t-n} + X_{t-n+1} + \ldots + X_{t+n-1} + X_{t+n} \right). \tag{28}$$

If we assume X_t to be an AR(1) process, we can derive the autocovariance function of Y_t as follows

$$\begin{aligned} \gamma_Y(h) &= \sum_{i=-n}^{n} \sum_{j=-n}^{n} \frac{1}{(2n+1)^2} \gamma_X(h+i-j) = \frac{1}{(2n+1)^2} \sum_{i=-n}^{n} \sum_{j=-n}^{n} \frac{a^{|h+i-j|}}{1-a^2} \sigma_\varepsilon^2 \\ &= \frac{\sigma_\varepsilon^2}{(1-a^2)(2n+1)^2} \sum_{j=-2n}^{2n} a^{|h+j|} \cdot (2n+1-|j|). \end{aligned} \tag{29}$$

Note that only for $h = 0$, we can eliminate the sum. △

Discussion. Filtering can also be used to transform a statistically nonstationary sequence into a stationary one (Box et al., 2015). Many time series in real life are not stationary, yet a sequence is often transformed into a time series that is (assumed to be) stationary (van der Vaart, 2021). For instance, an AR(1) process with linear growth, as in Appendix B, is not a (statistically) stationary process. Nevertheless, taking the difference filter

$$\nabla X_k = X_k - X_{k-1}, \tag{30}$$

that is, $\psi(0) = 1$ and $\psi(-1) = -1$, yields a stationary process.

4.2.2 Coarse-graining and its characteristics

In this section, we describe the coarse-graining procedure in the context of signal processing. Let X_t be a stationary time series and let w be a positive integer larger than or equal to 2. We can coarse-grain the process X_t by introducing the coarse-grained process Y_t as follows:

$$Y_t = X_{w \cdot t}. \tag{31}$$

The mean of the coarse-grained process Y_t is equal to the mean of the original process X_t:

$$\mu_X := \mathbb{E}(Y_t) = \mathbb{E}(X_{w \cdot t}) = \mathbb{E}(X_t) =: \mu_X. \tag{32}$$

For the autocovariance function, we calculate:

$$\gamma_Y(h) = \mathrm{Cov}(Y_t, Y_{t+h}) = \mathrm{Cov}(X_{w \cdot t}, X_{w \cdot (t+h)}) = \mathrm{Cov}(X_t, X_{t+w \cdot h}) = \gamma_X(w \cdot h). \tag{33}$$

Similarly to the filtering process, we find that the stationarity of the original time series X_t transfers to the coarse-grained process Y_t.

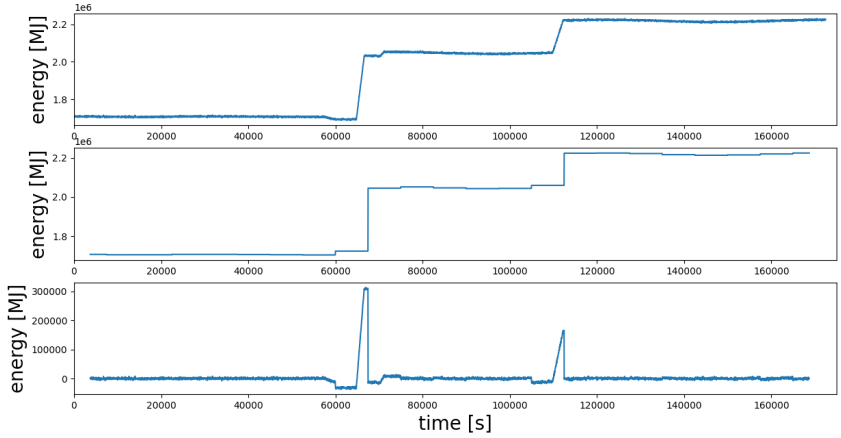

Figure 5: The top plot shows the typical features in the time series data for energy. The middle plot shows the coarse-grained data and the bottom plot shows the difference between the top and middle data, illustrating the large error in regions of abrupt changes.

4.2.3 Product of stationary processes

The product of stationary processes is again stationary, see Wecker (1978) for more details. On a restricted time interval (see Section 5 on how to quantify this), this enables us to interpret the energy over time as a statistically stationary process since $E = HV$, where the calorific value H and volume V can be seen as statistically stationary processes.

5 Quantifying the energy uncertainty for correlated data

In this section, we describe how to quantify the energy uncertainty for correlated data when the dataset is not stationary as a whole. We shall decompose the dataset into stationary parts and "jump" parts by means of segmentation algorithms. These "jump" parts, which we may also refer to as transition segments, can be approximated roughly because the time interval on which it happens is relatively small. Therefore, the stationary parts are mainly of interest to us.

5.1 Segmenting the data

Change point detection detects abrupt changes in the trend of time series data and can be used to segment a time series into stationary chunks. This is necessary because abrupt changes in the trend of a time series can cause large errors in the filtering and coarse-graining procedure, as can be seen in Figure 5.

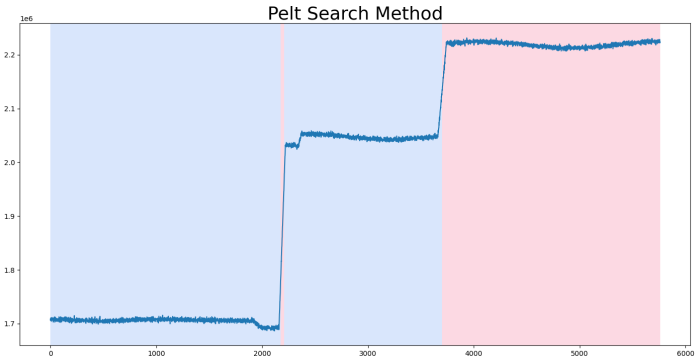

Figure 6: The PELT search method for offline change point detection.

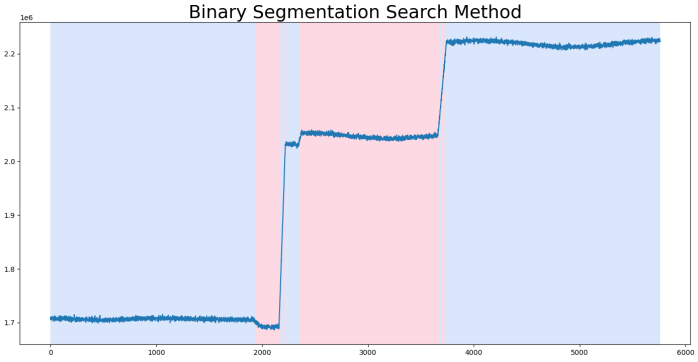

Figure 7: The binary search method for offline change point detection.

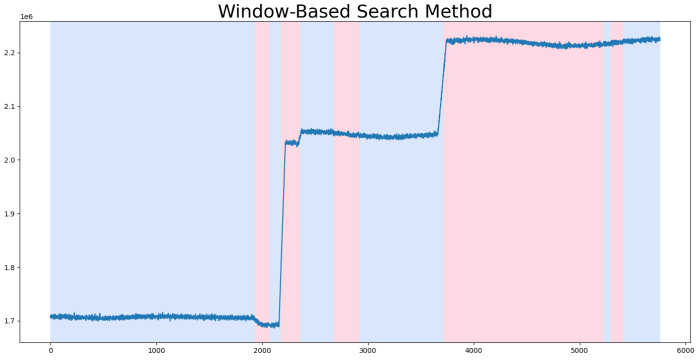

Figure 8: The window-based search method for offline change point detection.

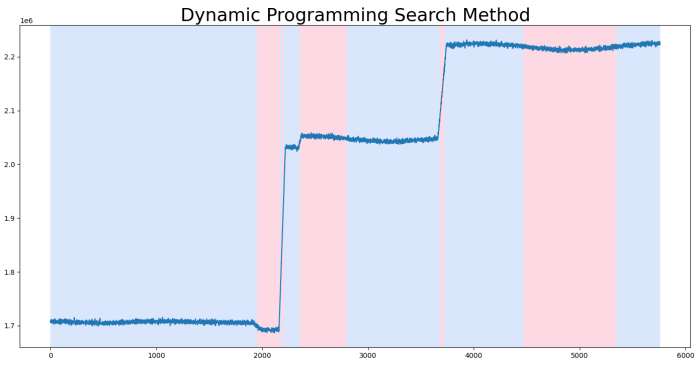

Figure 9: The dynamic programming search method for offline change point detection.

Abrupt changes in the trend of a time series are generally easy to detect with the human eye, but it is more involved in detecting change points algorithmically. In the setting of the present problem, the time series data describes the behavior of the volume and calorific value over a fixed amount of time in the past. In other words, all data is received and processed simultaneously. This means that offline change point detection algorithms can be used, which are methods that detect all change points in the time series dataset. Offline change point detection is different from online change point detection, where the time series data arrives in a live-steamed fashion. Online change point detection is used for continuous monitoring or immediate anomaly detection, and only the most recent change point is of interest.

While change point detection algorithms are best developed in the changepoint package in R, python also offers several options. A particularly convenient Python package for offline changepoint detection is the ruptures package because it is well-documented. For a review and details on offline changepoint detection methods, see Truong et al. (2020) and references therein. We explain and illustrate four change point detection methods below.

Pruned exact linear time (PELT) search method. The PELT search method is an exact method with a computational cost of $O(n)$, where n is the number of points in the time series. The method finds change points based on cost minimization.

Binary segmentation search method. The binary segmentation search method is an approximate method with a computational cost of $O(n \log n)$, where n is the number of points in the time series. The method finds change points based on iteratively applying a method that finds a single change point in a dataset. Having found one such point, the dataset is then divided into two parts, and the procedure is repeated.

Window-based search method. The window-based search method is an approximate method that uses two adjacent windows that move along the time series. It computes a discrepancy

between the windows, which is large when they are highly dissimilar. Based on the discrepancy curve created from discrepancy over the time series, the algorithm locates optimal change point indices in the time series.

Dynamic programming search method. The dynamic programming search method is an exact method with a computational cost of $O(kn^2)$, where n is the number of points in the time series and k is the number of change points. It is the most expensive of the four methods but also produces the best results.

The dynamic programming method is the more computationally expensive method but also yields the best results in terms of change point detection.

5.2 Stationarity tests

As explained in Section 3.3, our methods depend on (statistical) stationarity within segments. It is therefore important to check whether the data within the segments obtained from one of the methods described in Section 5.1 exhibit stationarity. We cannot test stationarity by fitting models like the AR(1) and AR(2) and checking whether the estimated model parameter values correspond to values for which the fitted model is stationary (see Appendix B). The reason is that by fitting such models, we bias ourselves towards these models. There is thus a need to check stationarity without assuming any specific model. Fortunately, such stationarity tests exist. A standard test is to test stationarity in an AR model such as the Dickey-Fuller test (see Dickey and Fuller (1979)) and the Kwiatkowskiâ€“Phillipsâ€“Schmidtâ€“Shin (KPSS) test (see Kwiatkowski et al. (1992)).

6 Conclusions and Discussion

Let us recall the research questions:

1. What is the effect of the averaging on the uncertainty in fiscal metering?
2. How can the uncertainty be correctly calculated, taking into account dependencies?

In Section 3, and in particular Section 3.1, we have answered the second question partly by deriving an upper bound on the total uncertainty with regards to the total energy, which is given by

$$\text{Total uncertainty} \lesssim m\Delta \left[\sum_{i=1}^{n} \sqrt{\Gamma_i} \right], \tag{34}$$

where Δ is a fixed time between the subsequent averaged measurements, where m is the fixed amount of averaged measurement instances within an arbitrary time interval, where n is the number of time intervals, and where

$$\Gamma_i = S_{H;i}^2 S_{V;i}^2 + H_i^2 S_{V;i}^2 + V_i^2 S_{H;i}^2, \quad 1 \le i \le n. \tag{35}$$

Recall that H_i and V_i are the averaged values of calorific value and volume, respectively, and $S^2_{V;i}$ and $S^2_{H;i}$ are the corresponding sample variances; this is the minimal data that has to and will be provided by the energy producer. This answer is partial because we are now giving a sufficiently high enough bound so that there is no need to understand the dependency structure.

We like to recall that the sample variances $S^2_{V;i}$ and $S^2_{H;i}$ can be computed with the help of the decomposition provided in (5). We caution the reader that obtaining $S^2_{V;i}$ and $S^2_{H;i}$ can easily be done incorrectly because one has to take both the uncertainty in measurements into account and the fluctuations in time.

A full answer to question 2 is that

$$\text{Total uncertainty} \approx m\Delta \sqrt{\sum_{i=1}^{n} \Gamma_i + 2 \sum_{1\leq i<i'\leq n} \Gamma_{ij}}, \tag{36}$$

where $\Gamma_{ij} = S_{H;i,i'} S_{V;i,i'} + H_i H_{i'} S_{V;i,i'} + V_i V_{i'} S_{H;i,i'}$ and with $S_{H;i,i'}$ and $S_{V;i,i'}$ the sample covariances of calorific value and volume, respectively. A crucial observation is that within the expression of the sample covariances, the "uncertainty in measurements"-term is not present because measurement errors are assumed to be independent. The total uncertainty can be computed numerically with the help of (36) for any data set, and we have performed some simulations in Section 3.4 for an artificial data set.

In Section 4, we have considered the first question in more depth. From the results above, we deduce that the total uncertainty growths at least as $O(m\sqrt{n})$, where m is the amount of time instants averaged upon and n the amount of time intervals, and is bounded above by $O(mn)$. Since the truth is somewhere in the middle, so to speak, it is better to average as little as possible. This conclusion, however, can be improved upon by taking the dependencies between measurements into account in a smarter way. Having introduced the framework of filtering and coarse-graining, we provide analytical results on how the covariances are affected by the filter considered. The filtering and coarse-graining steps come with explicit formulas. These analytical expressions can be used numerically to find the optimal filter for the data sets in question. We believe the Gaussian filter would perform better than the current moving average / boxcar filter in the sense of total uncertainty while simultaneously smoothing out the aggregated data set and containing less information than the full data set. The latter results in several interesting follow-up research questions.

In Section 5 we have shown how to segment a data set. It is appropriate for most segments to assume it can be modelled as an AR process. An alternative approach to the direct statistical computations performed as in Section 3.2 is to fit an autoregressive model to the stationary data segments. The benefit of this method is that explicit characterizations of the covariance in the data set can be obtained, as illustrated in Section 3.3 and Appendix B. Recall that the filtering and coarse-graining steps come with explicit formulas as well, which can be exploited to find the covariance changes for autoregressive models either analytically or numerically.

A Fundamental statistics

In this section, we collect some important properties of variances.

Property I. The variance of a product of independent variables X and Y is given by

$$\begin{aligned}\mathrm{Var}(XY) &= \mathrm{E}\left(XY)^2\right) - (\mathrm{E}(XY))^2 \\ &= \mathrm{E}(X^2)\,\mathrm{E}(Y^2) - (\mathrm{E}(X))^2\,(\mathrm{E}(Y))^2 \\ &= (\mathrm{Var}(X) + (\mathrm{E}(X))^2\,(\mathrm{Var}(Y) + (\mathrm{E}(Y))^2 - (\mathrm{E}(X))^2\,(\mathrm{E}(Y))^2 \\ &= ((\mathrm{Var}(X)(\mathrm{Var}(Y)) + (\mathrm{E}(X))^2\,\mathrm{Var}(Y) + (\mathrm{E}(Y))^2\,\mathrm{Var}(X).\end{aligned} \tag{37}$$

Property II. For any two random variables X and Y, not necessarily independent, we have the Cauchy Schwarz inequality

$$\mathrm{Cov}(X, Y) \leq \sqrt{\mathrm{Var}(X)}\,\sqrt{\mathrm{Var}(Y)}. \tag{38}$$

Property III. As pointed out in Rudmin (2010), the total variance of a data set equals the sample mean of the sample variances of the pooled sets (i.e., pool variance) plus the sample variance of the sample means of the pooled sets. This result is achieved from the result below.

Result. Assume $(Y_{j,k})_{1\leq j\leq m, 1\leq k\leq K}$ to be a collection of data points with $m, K \geq 1$. Consider the quantities

$$X_j := \frac{1}{K}\sum_{k=1}^{K} Y_{j,k}, \quad S_j^2 := \frac{1}{K}\sum_{k=1}^{K}(Y_{j,k} - X_j)^2, \tag{39}$$

for any $1 \leq j \leq m$, and let $A := \frac{1}{mK}\sum_{j=1}^{m}\sum_{k=1}^{K} Y_{j,k}$. The following decomposition holds:

$$S^2 := \frac{1}{mK}\sum_{j=1}^{m}\sum_{k=1}^{K}(Y_{j,k} - A)^2 = \frac{1}{m}\sum_{j=1}^{m} S_j^2 + \frac{1}{m}\sum_{j=1}^{m}(X_j - A)^2. \tag{40}$$

Proof. Fix $1 \leq i \leq n$. Let $D = (Y_{j,k})_{1\leq j\leq m, 1\leq k\leq K}$ be a pooled data set assembled from data sets $D_1 = (Y_{1,k})_{1\leq k\leq K}$, $D_2 = (Y_{2,k})_{1\leq k\leq K}$, …, $D_m = (Y_{m,k})_{1\leq k\leq K}$. Note that the number of measurements in the pooled set is mK.

The sample mean of data set D is

$$A = \frac{1}{mK}\sum_{j=1}^{m}\sum_{i=1}^{K} Y_{j,k} = \frac{1}{m}\sum_{j=1}^{m}\frac{1}{K}\sum_{k=1}^{K} Y_{j,k} = \frac{1}{m}\sum_{j=1}^{m} X_j, \tag{41}$$

and it thus equals the mean of means X_{ij}, i.e., the means of the smaller data sets $D_{i1}, \ldots, D_{im}$.

A direct computation shows that the sample variance of data set D is given by

$$S^2 = \frac{1}{mK}\sum_{j=1}^{m}\sum_{k=1}^{K}(Y_{j,k} - A)^2 = \frac{1}{mK}\sum_{j=1}^{m}\sum_{k=1}^{K} Y_{j,k}^2 - A^2 = \frac{1}{mK}Z^2 - A^2, \tag{42}$$

where $Z^2 = \sum_{j=1}^{m} \sum_{k=1}^{K} Y_{j,k}^2$ is the sum of squares. Note that

$$S_j^2 = \frac{1}{K} \sum_{k=1}^{K} (Y_{j,k} - X_j)^2 = \frac{1}{K} \sum_{k=1}^{K} Y_{j,k}^2 - X_j^2 = \frac{1}{K} Z_j^2 - X_j^2, \tag{43}$$

is the sample variance of the data set D_j, where $Z_j^2 = \sum_{k=1}^{K} Y_{j,k}^2$. Also, note that $Z^2 = \sum_{j=1}^{m} Z_j^2$ holds. Our goal is to compute S^2 via the means and variances of the smaller datasets. Indeed,

$$\begin{aligned} mKS^2 &= \sum_{j=1}^{m} Z_j^2 - mKA^2 \\ &= K \sum_{j=1}^{m} S_j^2 + K \sum_{j=1}^{m} X_j^2 - mKA^2 \\ &= K \sum_{j=1}^{m} S_j^2 + K \sum_{j=1}^{m} (X_j - A)^2, \end{aligned} \tag{44}$$

which completes the proof. □

B Autoregressive models

In this appendix, we give a brief description of AR(1) and AR(2) processes. In addition, we provide the concept of an AR(1) process with linear growth.

B.1 The AR(1) process

The AR(1) process is given by the following time series:

$$X_t = c + aX_{t-1} + \varepsilon_t, \tag{45}$$

where $c \in \mathbb{R}$ is refered to as the drift term and $a \in \mathbb{R}$ the intensity of the process. Further, it is assumed that $\mathbb{E}(\varepsilon_t) = 0$ and $\mathbb{V}(\varepsilon_t) = \sigma_\varepsilon^2 > 0$ hold. Furthermore, we assume that the ε_t are independently and identically distributed.

For stationary processes (which is the case for $|a| < 1$ (van der Vaart (2021)), an elementary calculation shows that

$$\mu_X := \mathbb{E}(X_t) = \frac{c}{1-a}. \tag{46}$$

Regarding the autocovariance function, we have

$$\gamma_X(h) = \mathrm{Cov}(X_t, X_{t+h}) = \frac{a^{|h|}}{1-a^2} \sigma_\varepsilon^2. \tag{47}$$

This model is stationary if and only if $|a| < 1$. If $|a| < 1$, then the autocovariance function exponentially decreases in h. Therefore, the process is still considered to exhibit short-range dependence.

Now, let us consider an AR(1) process with linear growth. That is,

$$\begin{cases} X_t &= c + aX_{t-1} + \varepsilon_t \\ Y_t &= X_t + \delta t, \end{cases} \tag{48}$$

with $X_0 = 0$ and $\delta \neq 0$ the linear growth term. This yields the following properties

$$\mathbb{E}Y_n = \mu_X + \delta t, \quad \mathrm{Var}(Y_n) = \mathrm{Var}(X_n), \quad \mathrm{Cov}(Y_t, Y_{t+h}) = \mathrm{Cov}(X_t, X_{t+h}). \tag{49}$$

B.2 The AR(2) process

The AR(2) process is given by the following time series

$$X_t = c + a_1 X_{t-1} + a_2 X_{t-2} + \varepsilon_t. \tag{50}$$

As for the AR(1) process, it is assumed that $\mathbb{E}(\varepsilon_t) = 0$ and $\mathbb{V}(\varepsilon_t) = \sigma_\varepsilon^2$ hold, and the ε_t are assumed to be independently and identically distributed.

The corresponding characteristic equation is given by $x^2 - a_1 x - a_2 = 0$ (van der Vaart (2021)). The AR(2) process is stationary if and only if the roots of the characteristic equation are outside of the unit circle $S^1 = \{z \in \mathbb{C} : |z| = 1\}$. In case of stationarity, the mean equals

$$\mu_X := \mathbb{E}(X_t) = \frac{c}{1 - a_1 - a_2}. \tag{51}$$

We compute the autocovariance function for a stationary AR(2) process as follows:

$$\begin{aligned} \gamma_X(h) &= \mathrm{Cov}(X_t, X_{t-h}) = \mathbb{E}(X_t X_{t-h}) - \mu_X^2 = \mathbb{E}\left((c + a_1 X_{t-1} + a_2 X_{t-2} + \varepsilon_t) X_{t-h}\right) - \mu_X^2 \\ &= c\mu_X + (a_1 + a_2 - 1)\mu_X^2 + a_1 \gamma_X(h-1) + a_2 \gamma_X(h-2) + \mathbb{E}(\varepsilon_t X_{t-h}) \\ &= a_1 \gamma_X(h-1) + a_2 \gamma_X(h-2) + \mathbb{E}(\varepsilon_t X_{t-h}). \end{aligned} \tag{52}$$

Here,

$$\mathbb{E}(\varepsilon_t X_{t-h}) = \begin{cases} 0 & h \neq 0 \\ \sigma_\varepsilon^2 & h = 0. \end{cases} \tag{53}$$

So, for $h \neq 0$ we find

$$\gamma_X(h) = a_1 \gamma_X(h-1) + a_2 \gamma_X(h-2). \tag{54}$$

To solve this recurrence relation, we determine $\gamma_X(0)$ and $\gamma_X(1)$. Using $\gamma_X(h) = \gamma_X(-h)$ for all $h \in \mathbb{Z}$, we find

$$\begin{aligned} \gamma_X(0) &= a_1 \gamma_X(1) + a_2 \gamma_X(2) + \sigma_\varepsilon^2, \\ \gamma_X(1) &= a_1 \gamma_X(0) + a_2 \gamma_X(1), \\ \gamma_X(2) &= a_1 \gamma_X(1) + a_2 \gamma_X(0). \end{aligned} \tag{55}$$

By elementary algebraic manipulations, the following expressions follow

$$\gamma_X(0) = \frac{1-a_2}{1+a_2}\frac{1}{(1-a_2)^2-a_1^2}\sigma_\varepsilon^2, \tag{56}$$

$$\gamma_X(1) = \frac{a_1}{1+a_2}\frac{1}{(1-a_2)^2-a_1^2}\sigma_\varepsilon^2. \tag{57}$$

C Theoretical results

In this appendix, we summarize the continuous analogs of the theory presented in this paper.

C.1 Details of averaging

Let g be the cumulative distribution function of a random variable X. The derivative of g with respect to X is the probability density function g'. If f is any function for which the expression

$$\mathbb{E}[f(X)] = \int_{-\infty}^{\infty} f(x)g'(x)dx \tag{58}$$

is finite, then we call $\mathbb{E}[f(X)]$ the mean of $f(X)$. When the integrand $f(x)$ and the integrator $g'(x)dx$ are both smooth enough, the Riemann-Stieltjes integral is the unambiguous method to compute the average. The Riemann-Stieltjes integral for an α-Hölder continuous function f and a β-Hölder continuous function g is given by

$$\int_a^b f(x)dg(x), \tag{59}$$

where it is necessary that $\alpha + \beta > 1$ for the integral to be well-defined. This was shown by Young (1936). However, when this condition fails, a more general notion of integration is required, leading to the development of the theory of rough paths, see Friz and Hairer (2020). In either case, unless the integral can be computed analytically, one requires approximations to the integral. A natural way of approximating the integral is by returning to its definition. Let P_n be an element of a sequence of partitions of the domain of integration (in this case, $[a, b]$), given by

$$P_n = \{a = x_0 < x_1 < \ldots < x_n = b\}. \tag{60}$$

The integral is defined to be the limit of the approximating sum S as the mesh size (the length of the largest subinterval) tends to zero

$$S(P_n, f, g) = \sum_{i=0}^{n-1} f(c_i)[g(x_{i+1}) - g(x_i)], \tag{61}$$

where $c_i \in [x_i, x_{i+1}]$ is in the ith subinterval. This sum is an approximation to the integral and can be evaluated independently of the regularity of f and g. In summary, the integral is given

by

$$\int_a^b f(x)dg(x) = \lim_{n\to\infty} S(P_n, f, g), \tag{62}$$

where the regularity of f and g is required to evaluate the limit. While the definition of the integral using this limit procedure is unambiguous, the sequence $S(P_n, f, g)$ depends on the partition and on the choice of c_i.

C.2 Continuous version of filtering

In general, the variance of a stochastic process after convolution with a filter can be calculated by using the formula below

$$\mathrm{Cov}(Y_s, Y_t) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(s-u)f(t-v)\mathrm{Cov}(X_u, X_v)dudv, \tag{63}$$

with Y_t denoting the convoluted process, i.e.,

$$Y_t = \int_{-\infty}^{\infty} f(t-s)X_s ds, \tag{64}$$

where X_t is the original process, $f(t)$ is the filter function (also known as the impulse response function), and $\mathrm{Cov}(X_s, X_t)$ is the covariance function of the original process. We refer to Faris (2001) for more information on the latter.

As an example, take $f(s) = \mathbf{1}_{[0,\Delta t]}(s)/\Delta t$. This is a continuous version of the boxcar filter and yields

$$\mathrm{Var}(Y_t) = \frac{1}{\Delta t}\int_t^{t+\Delta t}\int_t^{t+\Delta t} \mathrm{Cov}(X_u, X_v)dudv. \tag{65}$$

C.3 On the effect of coarse-graining

One common type of coarse-graining is temporal coarse-graining, which involves dividing the time axis into intervals of length Δt and replacing the original process $X(t)$ with a new process $Y(t)$ that takes the value of $X(t)$ at the beginning or the end of each interval. For example, if $X(t)$ is a Brownian motion, then $Y(t)$ is a random walk with step size $\sqrt{\Delta t}$. Temporal coarse-graining can reduce the variance of a stochastic process by smoothing out some of the fluctuations and noise. However, the exact effect of temporal coarse-graining on the variance depends on the autocorrelation function of the original process and the length of the time interval.

D Python Code

For the implementation of the methods that we have described in Section 3 and Section 4, we used Python. In particular, we use pandas (McKinney and Team (2015)) for data analysis, numpy for basic array manipulation, scipy (Virtanen et al. (2020)) for its time series capabilities, matplotlib (Hunter (2007)) for visualization and ruptures (Truong et al. (2020)) for offline changepoint detection algorithms.

SWI_VSL_main.py This code is used to create Figures 2 and 3.

```
import matplotlib.pyplot as plt
from SWI_VSL_functions import *

start_time = 0  # starting time in seconds [s]
end_time = 2*24*3600  # final elapsed time in seconds [s]
num_samples = 25  # number of measurements per time instance
uncorrelated = False

dt_mins = 60  # number of seconds in a minute
dt_quarters = 900  # number of seconds in a quarter
dt_hours = 3600  # number of seconds in an hour
dt_days = 24*3600  # number of seconds in a day

num_mins = int(np.floor(end_time/dt_mins))  # number of minutes in the
    dataset
num_quarters = int(np.floor(end_time/dt_quarters))  # number of quarters in
    the dataset
num_hours = int(np.floor(end_time/dt_hours))  # number of hours in the
    dataset
num_days = int(np.floor(end_time/dt_days))  # number of days in the dataset

time = np.linspace(start=start_time, stop=end_time, num=num_mins, endpoint=
    False)  # time in seconds [s]
fw = int(dt_hours/dt_mins)  # filter width used for time-averaging
win = sp.signal.windows.bartlett(fw)

np.random.seed(1)
epsi1ijk = np.random.normal(loc=0, scale=1, size=(num_mins, num_samples))
np.random.seed(2)
epsi2ijk = np.random.normal(loc=0, scale=1, size=(num_mins, num_samples))

if uncorrelated:
    muij = 100+50*np.sin(np.sqrt(2)*np.pi*time / (5*dt_hours))
    etaij = 100+50*np.cos(np.sqrt(2)*np.pi*time / (5*dt_hours))
else:
    muij = ar1(a=.5, b=25, ym=100, Nt=num_mins, R=1)
    etaij = ar1(a=.9, b=2, ym=100, Nt=num_mins, R=1)

Dijk = np.kron(muij, np.ones(num_samples)).reshape((num_mins, num_samples)) +
     epsi1ijk
Wijk = np.kron(etaij, np.ones(num_samples)).reshape((num_mins, num_samples))
    + epsi2ijk

EDij = 1/num_samples * np.sum(Dijk, axis=1)
EWij = 1/num_samples * np.sum(Wijk, axis=1)

VarDij = 1/num_samples * np.sum((Dijk - np.kron(EDij, np.ones(num_samples)).
    reshape((num_mins, num_samples)))**2, axis=1)
VarWij = 1/num_samples * np.sum((Wijk - np.kron(EWij, np.ones(num_samples)).
    reshape((num_mins, num_samples)))**2, axis=1)

ctime = coarse_grain(time, fw)
Hi = coarse_grain(filter_data(EDij, win, fw), fw)
Vi = coarse_grain(filter_data(EWij, win, fw), fw)
```

```

VarHi = coarse_grain(filter_data(VarDij, win, fw) + filter_data((EDij - np.
    kron(Hi, np.ones(fw)))**2, win, fw), fw)
VarVi = coarse_grain(filter_data(VarWij, win, fw) + filter_data((EWij - np.
    kron(Vi, np.ones(fw)))**2, win, fw), fw)

Ei = Hi*Vi
VarEi = VarHi*VarVi + (Hi**2)*VarVi + (Vi**2)*VarHi

TE = len(Ei)*fw*np.sum(Ei)
VarTE = (len(Ei)**2 * fw**2) * np.sum(VarEi)

print(TE, np.sqrt(VarTE))

plt.figure()
plt.plot(Ei)
plt.plot(np.sqrt(VarEi))
plt.show()
```

SWI_VSL_segmentation.py This code is used to create the figures in Section 5.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import scipy as sp
import ruptures as rpt
from SWI_VSL_functions import *

"""Synthetic Data"""

T = 175000  # Final time in [s]
dtf = 5*2*30  # Finest time resolution
dtc = 15*2*dtf  # 15min resolution
Nf = int(T/dtf)  # Number of steps
Nc = int(T/dtc)
tf = np.linspace(0, T, Nf+1)
tc = np.linspace(0, T, Nc+1)

R = 5  # Number of realisations of AR(1) process

xf = np.zeros((Nf+1, R))
vol = np.zeros((Nf+1, R))
cal = np.zeros((Nf+1, R))
def mu_V(t, R):
    return [46000 if t < int(Nf/3) else 55000 if t < int(2*Nf/3) else 60000]*
    np.ones(R)

def mu_C(t, R):
    return 37.1*np.ones(R)

# Parameters of the AR(1) process
a_vol = .5
var_vol = 25
measure_err_vol = 1
a_cal = .9
```

```
var_cal = .2
measure_err_cal = 1
xf[0, :] = 0
""" Type A errors are measurement errors due to calibration of measurement devices
    Type B errors are measurement errors due to other sources and can be correlated"""
e_vol = np.random.normal(0, var_vol, size=(Nf+1, R))  # Type B error for volume measurements
em_vol = np.random.normal(0, measure_err_vol, size=(Nf+1, R))  # Type A error for volume measurements
e_cal = np.random.normal(0, var_cal, size=(Nf+1, R))  # Type B error for calorific measurements
em_cal = np.random.normal(0, measure_err_cal, size=(Nf+1, R))  # Type A error for calorific measurements

# AR(1) process with parameter a and variance v_AR
for k in range(Nf):
    xf[k+1, :] = a_vol*xf[k, :] + e_vol[k+1, :]
    vol[k+1, :] = mu_V(k, R) + xf[k+1, :]

    xf[k+1, :] = a_cal*xf[k, :] + e_cal[k+1, :]
    cal[k+1, :] = mu_C(k, R) + xf[k+1, :]

vol = vol + em_vol
cal = cal + em_cal

sample_mean_vol = np.mean(vol, axis=1)
sample_mean_cal = np.mean(cal, axis=1)
sample_var_vol = np.var(vol, axis=1)
sample_var_cal = np.var(cal, axis=1)

ene = vol * cal  # Energy
sample_mean_ene = np.mean(ene, axis=1)
sample_var_ene = np.var(ene, axis=1)
ene_mean = sample_mean_vol * sample_mean_cal
ene_var = sample_var_cal * sample_var_vol + sample_var_cal * (sample_mean_vol)**2 + sample_var_vol * (sample_mean_cal)**2

fig, ax = plt.subplots(1, 2)
ax[0].plot(tf[1:], vol[1:, :], '.')
ax[0].set_xlabel('time [s]', fontsize=20)
ax[0].set_ylabel('volume [m3]', fontsize=20)
ax[0].tick_params(axis='both', labelsize=12)
ax[1].plot(tf[1:], cal[1:, :], '.')
ax[1].set_xlabel('time [s]', fontsize=20)
ax[1].set_ylabel('calorific value [MJ/m3]', fontsize=20)
ax[1].tick_params(axis='both', labelsize=12)

fig, ax = plt.subplots(1, 2)
ax[0].errorbar(tf[1:], sample_mean_vol[1:], sample_var_vol[1:], linestyle='None', marker='o')
ax[0].set_xlabel('time [s]', fontsize=20)
ax[0].set_ylabel('volume [m3]', fontsize=20)
ax[0].tick_params(axis='both', labelsize=12)
```

```
ax[1].errorbar(tf[1:], sample_mean_cal[1:], sample_var_cal[1:], linestyle='
    None', marker='o')
ax[1].set_xlabel('time [s]', fontsize=20)
ax[1].set_ylabel('calorific value [MJ/m3]', fontsize=20)
ax[1].tick_params(axis='both', labelsize=12)

fig, ax = plt.subplots(1, 3)
ax[0].errorbar(tf[1:], sample_mean_ene[1:], sample_var_ene[1:], linestyle='
    None', marker='o')
ax[0].set_xlabel('time [s]', fontsize=20)
ax[0].set_ylabel('energy [MJ]', fontsize=20)
ax[0].tick_params(axis='both', labelsize=12)
ax[1].errorbar(tf[1:], ene_mean[1:], ene_var[1:], linestyle='None', marker='o
    ')
ax[1].set_xlabel('time [s]', fontsize=20)
ax[1].set_ylabel('energy [MJ]', fontsize=20)
ax[1].tick_params(axis='both', labelsize=12)
ax[2].errorbar(tf[1:], np.abs(sample_mean_ene[1:] - ene_mean[1:]), np.abs(
    sample_var_ene[1:] - ene_var[1:]), linestyle='None', marker='o')
ax[2].set_xlabel('time [s]', fontsize=20)
ax[2].set_ylabel('energy [MJ]', fontsize=20)
ax[2].tick_params(axis='both', labelsize=12)

"""Real Synthetic Data"""

df = pd.read_csv('test_data.csv')
df['energy [MJ]'] = df.loc[:, 'volume [m3]'] * df.loc[:, 'calorific_value [MJ
    /m3]']
FW = int(250)  # filter width
win = sp.signal.windows.hann(FW)

# Changepoint detection
CPdetect = False
if CPdetect:
    # Changepoint detection with the Pelt search method
    model="rbf"
    points = np.array(df.loc[:, 'energy [MJ]'])
    algo = rpt.Pelt(model=model).fit(points)
    result = algo.predict(pen=10)
    rpt.display(points, result, figsize=(10, 6))
    plt.title('Pelt Search Method', fontsize=26)

    # Changepoint detection with the Binary Segmentation search method
    model = "l2"
    algo = rpt.Binseg(model=model).fit(points)
    my_bkps = algo.predict(n_bkps=10)
    # show results
    rpt.show.display(points, my_bkps, figsize=(10, 6))
    plt.title('Binary Segmentation Search Method', fontsize=26)

    # Changepoint detection with window-based search method
    model = "l2"
    algo = rpt.Window(width=40, model=model).fit(points)
    my_bkps = algo.predict(n_bkps=10)
    rpt.show.display(points, my_bkps, figsize=(10, 6))
```

```
    plt.title('Window-Based Search Method', fontsize=26)

    # Changepoint detection with dynamic programming search method
    model = "l1"
    algo = rpt.Dynp(model=model, min_size=3, jump=5).fit(points)
    my_bkps = algo.predict(n_bkps=10)
    rpt.show.display(points, my_bkps, figsize=(10, 6))
    plt.title('Dynamic Programming Search Method', fontsize=26)
    plt.show()


mean_vol, data_mean_vol, diff = mean_data_test(df.loc[:, 'volume [m3]'], FW)
fig, ax = plt.subplots(1,2)
ax[0].plot(mean_vol[:-1])
ax[0].plot(data_mean_vol[1:-1])
ax[0].set_title('Moving average')
ax[1].plot(diff[1:-1])
ax[1].set_title('Difference between Filter+CG and ordinary averaging')

# Convolution with filter
volume_filt = filter_data(df.loc[:, 'volume [m3]'], win, FW)
calorific_filt = filter_data(df.loc[:, 'calorific_value [MJ/m3]'], win, FW)
energy_filt = filter_data(df.loc[:, 'energy [MJ]'], win, FW)
time_filt = filter_data(df.loc[:, 'time [s]'], win, FW)

# Remove filter boundary
volume_filt = remove_filter_boundary(volume_filt, FW)
calorific_filt = remove_filter_boundary(calorific_filt, FW)
energy_filt = remove_filter_boundary(energy_filt, FW)
time_filt = remove_filter_boundary(time_filt, FW)

# Coarse graining data
volume_filt_cg = coarse_grain(volume_filt, FW)
calorific_filt_cg = coarse_grain(calorific_filt, FW)
energy_filt_cg = coarse_grain(energy_filt, FW)
time_filt_cg = coarse_grain(time_filt, FW)

# Plotting of the original data and the filtered data
fig, ax = plt.subplots(1, 2)
ax[0].plot(df.loc[:, 'time [s]'], df.loc[:, 'volume [m3]'])
ax[0].plot(time_filt, volume_filt)
ax[0].set_xlabel('time [s]', fontsize=22)
ax[0].set_ylabel('volume [m3]', fontsize=22)
ax[0].tick_params(axis='both', labelsize=14)
ax[1].plot(df.loc[:, 'time [s]'], df.loc[:, 'calorific_value [MJ/m3]'])
ax[1].plot(time_filt, calorific_filt)
ax[1].set_xlabel('time [s]', fontsize=22)
ax[1].set_ylabel('calorific value [MJ/m3]', fontsize=22)
ax[1].tick_params(axis='both', labelsize=14)

# Plotting the original and the filtered data
fig, ax = plt.subplots(3, 1)
ax[0].plot(df.loc[:, 'time [s]'], df.loc[:, 'energy [MJ]'])
ax[0].set_ylabel('energy [MJ]', fontsize=20)

```

```
ax[1].set_ylabel('energy [MJ]', fontsize=20)
ax[1].plot(time_filt, energy_filt)

ax[2].set_xlabel('time [s]', fontsize=20)
ax[2].set_ylabel('energy [MJ]', fontsize=20)
ax[2].plot(time_filt, energy_filt)

# Plotting data for the same filter at different filter widths
fig, ax = plt.subplots(1, 1)

FW1, FW2, FW3, FW4 = 20, 100, 500, 2500
win1 = sp.signal.windows.boxcar(FW1)
win2 = sp.signal.windows.boxcar(FW2)
win3 = sp.signal.windows.boxcar(FW3)
win4 = sp.signal.windows.boxcar(FW4)
t1 = remove_filter_boundary(sp.signal.convolve(df.loc[:, 'time [s]'], win1,
    mode='same') / sum(win1), FW1)
t2 = remove_filter_boundary(sp.signal.convolve(df.loc[:, 'time [s]'], win2,
    mode='same') / sum(win2), FW2)
t3 = remove_filter_boundary(sp.signal.convolve(df.loc[:, 'time [s]'], win3,
    mode='same') / sum(win3), FW3)
t4 = remove_filter_boundary(sp.signal.convolve(df.loc[:, 'time [s]'], win4,
    mode='same') / sum(win4), FW4)
e1 = remove_filter_boundary(sp.signal.convolve(df.loc[:, 'energy [MJ]'], win1
    , mode='same') / sum(win1), FW1)
e2 = remove_filter_boundary(sp.signal.convolve(df.loc[:, 'energy [MJ]'], win2
    , mode='same') / sum(win2), FW2)
e3 = remove_filter_boundary(sp.signal.convolve(df.loc[:, 'energy [MJ]'], win3
    , mode='same') / sum(win3), FW3)
e4 = remove_filter_boundary(sp.signal.convolve(df.loc[:, 'energy [MJ]'], win4
    , mode='same') / sum(win4), FW4)
ax.plot(df.loc[:, 'time [s]'], df.loc[:, 'energy [MJ]'], label='Truth')
ax.set_ylabel('energy [MJ]', fontsize=20)
ax.plot(t1, e1, label='Boxcar 20')
ax.plot(t2, e2, label='Boxcar 100')
ax.plot(t3, e3, label='Boxcar 500')
ax.plot(t4, e4, label='Boxcar 2500')
ax.legend(fontsize=18)
ax.set_title('Fixed filter averaging with different filter widths', fontsize
    =24)

# Plotting data for different filters at the same width
fig, ax = plt.subplots(1, 1)
FWt = 100
win1 = sp.signal.windows.boxcar(FWt)
win2 = sp.signal.windows.gaussian(FWt, 7)
win3 = sp.signal.windows.bartlett(FWt)
win4 = sp.signal.windows.exponential(FWt)
t1 = remove_filter_boundary(sp.signal.convolve(df.loc[:, 'time [s]'], win1,
    mode='same') / sum(win1), FWt)
t2 = remove_filter_boundary(sp.signal.convolve(df.loc[:, 'time [s]'], win2,
    mode='same') / sum(win2), FWt)
t3 = remove_filter_boundary(sp.signal.convolve(df.loc[:, 'time [s]'], win3,
    mode='same') / sum(win3), FWt)
```

```
t4 = remove_filter_boundary(sp.signal.convolve(df.loc[:, 'time [s]'], win4, mode='same') / sum(win4), FWt)
e1 = remove_filter_boundary(sp.signal.convolve(df.loc[:, 'energy [MJ]'], win1, mode='same') / sum(win1), FWt)
e2 = remove_filter_boundary(sp.signal.convolve(df.loc[:, 'energy [MJ]'], win2, mode='same') / sum(win2), FWt)
e3 = remove_filter_boundary(sp.signal.convolve(df.loc[:, 'energy [MJ]'], win3, mode='same') / sum(win3), FWt)
e4 = remove_filter_boundary(sp.signal.convolve(df.loc[:, 'energy [MJ]'], win4, mode='same') / sum(win4), FWt)
ax.plot(df.loc[:, 'time [s]'][400:500], df.loc[:, 'energy [MJ]'][400:500], label='Truth')
ax.set_ylabel('energy [MJ]', fontsize=20)
ax.plot(t1[350:450], e1[350:450], label='Boxcar 100')
ax.plot(t2[350:450], e2[350:450], label='Gaussian 100')
ax.plot(t3[350:450], e3[350:450], label='Bartlett 100')
ax.plot(t4[350:450], e4[350:450], label='Exponential 100')
ax.legend(fontsize=18)
ax.set_title('Fixed filter width averaging with different filters', fontsize=24)


# Plotting original data, mean and variance
fig, ax = plt.subplots(3, 1)
ax[0].plot(usable_data(df.loc[:, 'time [s]'], FW), usable_data(df.loc[:, 'energy [MJ]'], FW))
ax[0].set_xlim(0, T)
ax[0].set_ylabel('energy [MJ]', fontsize=20)
ax[1].plot(time_filt, remove_filter_boundary(pointwise_mean_data(df.loc[:, 'energy [MJ]'], win, FW), FW))
ax[1].set_xlim(0, T)
ax[1].set_ylabel('energy [MJ]', fontsize=20)
ax[2].plot(time_filt, remove_filter_boundary(usable_data(df.loc[:, 'energy [MJ]'], FW) - pointwise_mean_data(df.loc[:, 'energy [MJ]'], win, FW), FW))
ax[2].set_xlim(0, T)
ax[2].set_ylabel('energy [MJ]', fontsize=20)
ax[2].set_xlabel('time [s]', fontsize=20)

plt.show()
```

SWI_VSL_functions.py This code contains general functions which are called upon in the codes `SWI_VSL_main.py` and `SWI_VSL_segmentation.py`.

```
import scipy as sp
import numpy as np

"""Functions"""
def usable_data(data, filterwidth):
    N_data_points = len(data)
    N_intervals = int(N_data_points/filterwidth)
    return data[:N_intervals*filterwidth]
def filter_data(data, window, filterwidth):
    data = usable_data(data, filterwidth)
```

```
    data_filt = sp.signal.convolve(data, window, mode='same') / sum(window)
    return data_filt
def coarse_grain(data, filterwidth):
    return data[int(filterwidth/2)::filterwidth]
def remove_filter_boundary(data, filterwidth):
    return data[int(filterwidth/2):-int(filterwidth/2)]
def pointwise_mean_data(data, window, filterwidth):
    data = usable_data(data, filterwidth)
    data_filt = filter_data(data, window, filterwidth)
    data_filt_cg = coarse_grain(data_filt, filterwidth)
    pointwise_mean = np.kron(data_filt_cg, np.ones(filterwidth))
    return pointwise_mean
def mean_data_test(data, filterwidth):
    data = usable_data(data, filterwidth)
    mean_dat = [1/filterwidth * sum(data[i:i+filterwidth]) for i in range(0,
    len(data), filterwidth)]
    boxcar = sp.signal.windows.boxcar(filterwidth)
    data_filt = sp.signal.convolve(data, boxcar, mode='same') / sum(boxcar)
    data_mean = data_filt[::filterwidth]
    difference = mean_dat - data_mean
    return mean_dat, data_mean, difference
def auto_corr(data):
    result = sp.signal.correlate(data, data, mode='full')
    return result[result.size/2:]

def ar1(a, b, ym, Nt, R=1):
    # a is the coefficient of the AR(1) process
    # b is the variance of the normal distributed errors
    # Nt is the number of time points
    # R is the number of realisations
    if R == 1:
        e = np.random.normal(loc=0, scale=b, size=Nt+1)
        x = np.zeros(Nt+1)
        y = np.zeros(Nt+1)
        for k in range(Nt):
            x[k+1] = a*x[k] + e[k]
            y[k+1] = ym + x[k+1]
    else:
        e = np.random.normal(loc=0, scale=b, size=(Nt+1, R))
        x = np.zeros((Nt+1, R))
        y = np.zeros((Nt+1, R))
        for k in range(Nt):
            x[k+1, :] = a*x[k, :] + e[k+1, :]
            y[k+1, :] = ym*np.ones(R) + x[k+1, :]
    return y[:-1]
```

E Synthetic data set

In this appendix, we show a figure of the synthetic data provided by VSL. This data set is used for the energy time series in Section 5.

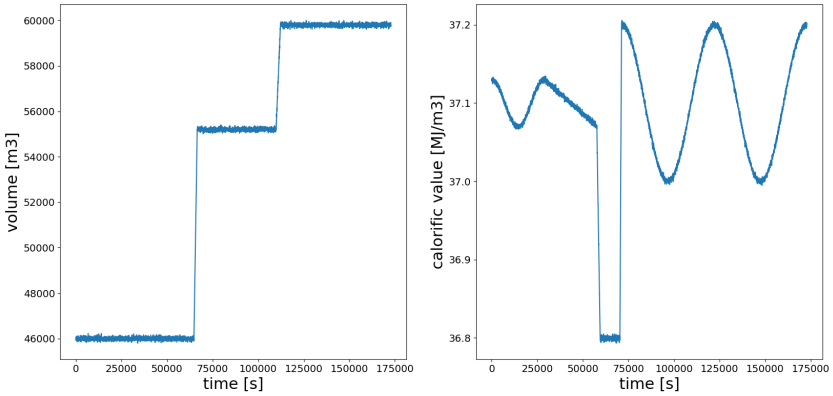

Figure 10: Synthetic data set provided by VSL with a time series of volume (left) and a time series of calorific value (right) over a time span of two days. See Figure 2 for a time series of volume and caloric value which are closer to reality.

References

George EP Box, Gwilym M Jenkins, Gregory C Reinsel, and Greta M Ljung. *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

C. Chatfield and H. Xing. *The Analysis of Time Series: An Introduction with R*. CRC Press, Boca Raton, Florida, 2nd edition, 2019.

P.S.P. Cowpertwait and A.V. Metcalfe. Introductory time series with R, 2009.

S. Crowder, C. Delker, E. Forrest, and N. Martin. *Introduction to statistics in metrology*. Springer, Cahm, Switzerland, 2020.

Alain de Cheveigné and Israel Nelken. Filters: when, why, and how (not) to use them. *Neuron*, 102(2):280–293, 2019.

D.A. Dickey and W.A. Fuller. Distribution of the estimators for autoregressive time series with a unit root. *Journal of the American Statistical Association*, 74(366a):427–431, 1979.

William G Faris. Lectures on stochastic processes, 2001.

International Organization for Standardization. ISO 6976 natural gas — calculation of calorific values, density, relative density, and Wobbe indices from composition. ISO, 2016.

International Organization for Standardization. ISO 15112 Natural gas — Energy determination. ISO, 2018.

Peter K Friz and Martin Hairer. *A course on rough paths*. Springer, 2020.

John D Hunter. Matplotlib: A 2D graphics environment. *Computing in science & engineering*, 9(03):90–95, 2007.

D. Kwiatkowski, P.C.B. Phillips, P. Schmidt, and Y. Shin. Testing the null hypothesis of stationarity against the alternative of a unit root: How sure are we that economic time series have a unit root? *Journal of econometrics*, 54(1-3):159–178, 1992.

Wes McKinney and PD Team. Pandas–Powerful python data analysis toolkit. *Pandas–Powerful Python Data Analysis Toolkit*, 1625, 2015.

F Pavese and A.B. Forbes. *Data modeling for metrology and testing in measurement science*. Birkhäuser, Boston, 2008.

Joseph W Rudmin. Calculating the exact pooled variance. *arXiv preprint arXiv:1007.1012*, 2010.

R.H. Shumway and D.S. Stoffer. *Time Series analysis and its applications*. Springer, Cham, Switzerland, fourth edition, 2000.

Charles Truong, Laurent Oudre, and Nicolas Vayatis. Selective review of offline change point detection methods. *Signal Processing*, 167:107299, 2020.

Aart van der Vaart. *Statistical time series*. Lecture Notes, 2021.

Martin Vetterli, Jelena Kovačević, and Vivek K Goyal. *Foundations of signal processing*. Cambridge University Press, 2014.

Pauli Virtanen, Ralf Gommers, Travis E Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, et al. SciPy 1.0: fundamental algorithms for scientific computing in Python. *Nature methods*, 17(3): 261–272, 2020.

William E Wecker. A note on the time series which is the product of two stationary time series. *Stochastic Processes and their Applications*, 8(2):153–157, 1978.

Laurence C Young. An inequality of the Hölder type, connected with Stieltjes integration. *Acta Math.*, 67(1), 1936.