Analyzing Data Streams by Online DFT

Alexander Hinneburg[1], Dirk Habich[2], and Marcel Karnstedt[3]

[1] Martin-Luther University of Halle-Wittenberg, Germany
hinneburg@informatik.uni-halle.de
[2] Dresden University of Technology, Germany
dirk.habich@tu-dresden.de
[3] Technical University of Ilmenau, Germany
marcel.karnstedt@tu-ilmenau.de

Abstract. Sensor data have become very huge and single measures are produced at high rates, resulting in *streaming* sensor data. In this paper, we present a new mining tool called Online DFT, which is particularly powerful for estimating the spectrum of a data stream. Unique features of our new method include low update complexity with high-accuracy estimations for very long periods, and the ability of long-range forecasting based on our Online DFT. Furthermore, we describe some applications of our Online DFT.

1 Motivation

Physical sensor environments can be found all over the world, and they are involved in a wide range of applications. In recent time, sensor data have become very huge and single measures are produced at high rates, resulting in *streaming* sensor data. Examples of this can be found in financial applications dealing with time series for various stocks, in applications monitoring sensors to detect failures and problems in large buildings at an early stage, or in real-time market analyses of supermarket customer data.

Aside from application-related issues, interesting general questions in data stream environments are: What are the dominant frequencies of a streaming signal? Which of them are currently most active in the stream? Is the stream very volatile (mostly high frequencies are active) or more stable (low frequencies are active)? How does the stream look like in the near future? To analyze the frequencies in a data stream, the discrete Fourier transform (DFT) in its original form is an unqualified tool, because the Fourier coefficients do not contain any information about the temporal location of the frequencies. To overcome this disadvantage, the Short Time Fourier transform (STFT) can be used, which chops the stream into overlapping pieces and estimates frequencies from these pieces. The STFT represents a compromise between the resolution in the frequency domain and the time domain. But, due to the processing of pieces, it is difficult to use the frequency information to derive a forecast of the stream. Wavelets have been used to analyze streams as well. Aside from many advantages of wavelets, such as fast online decomposition, there are also some disadvantages. First, there is no one-to-one relation between wavelet coefficients and frequency intervals.

This complicates the task of interpreting the results of a wavelet transform. Second, the wavelet transform is not shift-invariant; that means, the wavelet coefficients of a stationary signal (i.e., frequencies do not change) also depend on the starting point of the transformation. This is not the case for the DFT, and therefore, we decided not to use a wavelet model as starting point.

Our main contribution is a one-pass method with sublinear runtime for estimating the frequencies of a data stream. Our Online DFT approach partitions the frequency range into subintervals of increasing size. Low frequencies are estimated with higher resolution, while high frequencies are estimated more coarsely. This is justified by the observation that many data streams are driven mainly by low frequencies. Frequencies within the subintervals are estimated separately, which allows using different sample rates for different frequency subintervals. This leads to enormous savings in runtime, because the sample rates can be adjusted to the estimated frequencies (low frequencies need only low sample rates). The second feature of the new Online DFT is the adaptive choice of sample points, which increases the power of our approach. The properties of the Online DFT lead to a number of interesting applications, e.g., forecasting, cleaning, and monitoring of streams, which are described in detail in the experiment section.

Related Work Stream mining is related to time series analysis and forecasting (e.g., [5]), which is based on discovering patterns and periodicity [3, 6]. Traditional generative time series models include auto-regressive (AR) models and their generalizations (ARMA etc.). They can be used for forecasting, but usually they fail in streaming environments and often have a number of limitations (see [11] for more details on this). There are also approaches for nonlinear forecasting, but they always require human intervention [12]. Papadimitriou et al. [11] utilize incremental DWT [4] in order to find periods in streams and to do long-range forecasting. In their AWSOM method, the authors model the wavelet coefficients by an online linear auto-regression model [2]. This work focuses on forecasting and thus, it is related to the proposed Online DFT, but it is based on completely different methods. In this work, we focus on the DFT approach and its characteristics only.

The Fourier transform has successfully been applied to numerous tasks. Recent work deals with the approximation of large Fourier coefficients [9] by sampling only a part of a time series. [5] discover representative trends using sketches and they utilize FFT to compute distances between stream sections. Based on this, [1] defines approximate periodicity and applies efficient sampling techniques in order to find potential periods. Most relevant to our work is indexing of time series (e.g., [8]), which would benefit from the availability of a fast Online DFT. An approach closely related to the presented Online DFT is the incremental algorithm introduced by Zhu and Shasha [13]. They use DFT to summarize streams within a finite window. In contrast to our work, they focus on finding correlations between different streams, instead of forecasting. Moreover, they uniformly discretize the frequency domain, whereas we represent lower frequencies with higher accuracy.

The remainder of this paper is organized as follows: After presenting necessary prerequisites in Section 2, we introduce our novel Online DFT approach in Section 3. In Section 4 and 5 we discuss some complexity aspects and present some data stream experiments. The paper ends with conclusion in Section 6.

2 Prerequisites

Discrete Fourier Transform The n-point discrete Fourier transform of a time sequence $x = x_0, \ldots, x_{n-1}$ is a sequence $X = X_0, \ldots, X_{n-1}$ of complex numbers given by

$$X_f = {}^1\!/\!\sqrt{n} \sum_{t=0}^{n-1} x_t \cdot e^{-j2\pi f \cdot t/n}, \quad f = 0, \ldots, n-1$$

where j is the imaginary unit $j = \sqrt{-1}$. The original signal x can be recovered by the inverse Fourier transform. The runtime complexity of the discrete Fourier transform is $O(n^2)$. The fast Fourier transform (FFT) can compute the DFT coefficients in $O(n \log n)$.

Filtering When estimating a frequency spectrum from a finite sample, aliasing becomes an issue. Aliasing happens if the stream contains frequencies larger than half the sample rate. Those frequencies distort the spectrum in an unpredictable way and cannot be removed afterwards. To be on the save side, high frequencies are removed from the stream by a low pass filter before applying the DFT. Filtering can be done in the frequency as well as in the time domain. The advantage of a time-domain filter is that the output values have the same rate as the input data. Linear filters in the time domain take a sequence x_t of input values and produce a sequence y_t of output values by the formula: $y_t = \sum_{k=0}^{M} c_k \cdot x_{t-k} + \sum_{j=1}^{N} d_j \cdot y_{t-j}$. The $M+1$ coefficients c_k and the N coefficients d_j are fixed and define the filter response. There exists an elaborate theory on how to determine the coefficients c_k and d_j for a given pass band. In the later sections we use Chebyshev filters, but any other filter can be used as well. We observed that the setting $M = 5$ and $N = 5$ gives good results in the experiments. However, the results are not sensitive to the particular setting.

Any filter delays the signal. Note that the delay is in general not a constant, but a function depending on the frequency. The delay for specific frequencies can be determined from the filter coefficients. We denote the phase delay by $\tau(f)$, where f is the normalized frequency. Details on how to determine the filter coefficients and the delay can be found in [7].

3 Online DFT

Now, we propose our novel data analyzing approach for streams based on Online DFT. We introduce a model for the stream which is based on Fourier coefficients. The approach is well-suited for streaming scenarios as it has sublinear runtime complexity.

The method slides a window over the stream, which includes the last T elements. The straight-forward estimation of the frequencies from the window using the fast Fourier transform (FFT) runs in $O(T \log T)$. For large T, online processing is not feasible, since the FFT has to be re-applied when new data arrives. Our solution to the problem is to break up the frequency domain into smaller intervals. The frequency intensities in each subinterval are estimated from a small sample.

3.1 Online DFT Streaming Model

The stream elements $S = x_0, x_1, \dots, x_{t-1}, x_t$ are measured at an even rate. The window of the last T stream elements at the current time index t is denoted by $W_t = x_{t-T-1}, \dots, x_{t-1}, x_t$. We assume that (i) the duration between the arrival of two consecutive data stream elements is one standard time unit, and (ii) the stream does not contain frequencies above the critical Nyquist frequency[4] $f_c = 1/2$.

The goal is to estimate all frequencies between zero and $1/2$ of the window W_t to model the stream. Our idea is to partition the frequency range $(0, 1/2)$ into L subintervals. The l-th interval is $(1/2^{l+1}, 1/2^{l})$ for $1 \leq l < L$ and $(0, 1/2^{l})$ for $l = L$. So, the numbering goes from right to left as shown below:

$$(0, 1/2^{L}),\ (1/2^{L}, 1/2^{L-1}),\ \dots, (1/2^{2+1}, 1/2^{2}),\ (1/2^{1+1}, 1/2^{1})$$

The intervals can be estimated separately. We use short sliding windows for high frequencies and long sliding windows for low frequencies. Using filters, the input for the estimation of the l-th frequency interval $(1/2^{l+1}, 1/2^{l})$ is bandwidth-limited to $1/2^{l}$. Due to the Nyquist theorem, the sample rate for the input can be much lower without loss of information. We use a bandpass filter with a pass band $(1/2^{l+1}, 1/2^{l})$ for $1 \leq l < L$ and a lowpass filter $(0, 1/2^{l})$ when $l = L$, respectively. The filter used for the l-th interval is denoted by H_l and the filtered stream by $H_l(S) = y_1^{(l)}, y_2^{(l)}, \dots, y_{t-1}^{(l)}, y_t^{(l)}$.

The distance between sample points, δ_l, increases with l. Thus, for high frequencies, the sample points are close, while for low frequencies, they lie more apart: $\delta_l = 2^{l-1},\ 1 \leq l \leq L$. To determine the frequencies in the l-th interval, we consider only the filtered data stream sample points with a distance of δ_l. This is an enormous saving, e.g., when $l = 8$, only every $\delta_l = 2^{8-1} = 128$th point has to be used as sample point, instead of having to use all points. As explained before, this saving comes without loss of information.

The number of sample points for each frequency interval is upper-bounded by $N_{\max}$, where $N_{\max} \ll T$ and $\delta_L \cdot N_{\max} \leq T$. The sample points are the points that arrived last in the stream. The actual number of sample points, N_l, for each frequency subinterval l is adaptively chosen with respect to the data stream in order to avoid artifacts. The adaptive choice of N_l is crucial for the power of our approach. Details on how to choose N_l will be explained in subsection 3.2.

[4] Filtering can be used to guarantee this assumption.

The number of intervals, L, depends on the maximal number of sample points, $N_{\max}$, and the size of the sliding window, T: $L = \max\{i \in \mathbb{N}\colon 2^{i-1} \cdot N_{\max} \leq T\} = \lfloor \log {}^{T-1}/_{N_{\max}-1} \rfloor + 1$. The input for the frequency estimation in the l-th subinterval is the sequence of sample points taken from the output of the filter H_l: $S^{(l)} = y^{(l)}_{t-(N_l-1)\delta_l}, y^{(l)}_{t-(N_l-2)\delta_l}, \ldots, y^{(l)}_{t-\delta_l}, y^{(l)}_{t}$, where $N_l \leq N_{\max}$. The frequencies in the l-th subinterval are estimated from $S^{(l)}$ by DFT. The output consists of N_l Fourier coefficients, from which those between $\lfloor {}^{N_l}/_4 \rfloor$ and $\lceil {}^{N_l}/_2 \rceil$ estimate frequencies in the desired interval. The first quarter of coefficients is zero due to the band pass filter. An exception is the estimate of the L-th interval, from which the Fourier coefficients between 1 and $\lceil {}^{N_l}/_2 \rceil$ are used. Figure 1 illustrates the interplay of the frequency estimation for different subintervals.

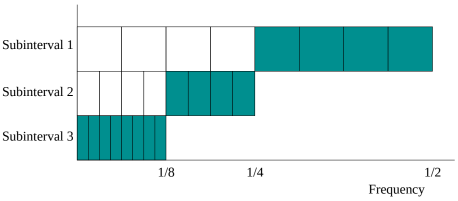
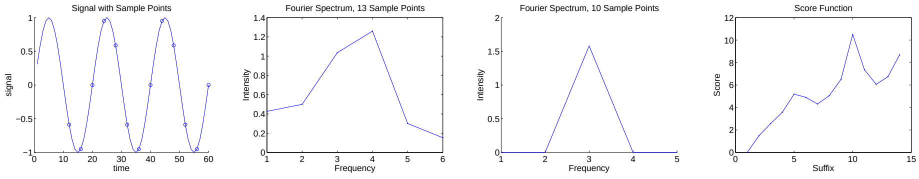


Fig. 1. Results of the DFT of $S^{(1)}, S^{(2)}, S^{(3)}$ in the frequency domain, ($L = 3, N_{\max} = 16$). Each rectangle represents a Fourier coefficient, the shaded coefficients are non-zero.

After this overview of the Online DFT model, the next sections discuss some further details and explain how the model is used for several example applications on data streams. In subsection 3.2, we explain how to choose the number of sample points per level. In subsection 3.3, we introduce the inverse Online DFT.

3.2 The Difficult Choice of Sample Points

The estimation of dominant frequencies by the DFT from a finite set of sample points is very sensitive to the choice of actual sample points. As the DFT needs infinite support of the signal, it must be possible to cyclically continue to sample points. Otherwise, the Fourier spectrum would include some artificial frequencies. For example, let the signal be a pure sinus $s(t) = sin({}^{2\pi}/_{20} \cdot t)$, and the current time point $t = 60$. If 13 sample points are used with $\delta = 4$ (Figure 2(a)), the spectrum based on this choice of sample points includes more than one non-zero component (Figure 2(b)). Ideally, the spectrum should include exactly one non-zero component, because the signal is a pure tone. In case the right-most

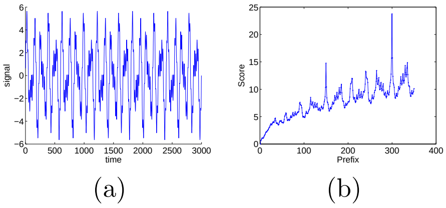


(a) (b) (c) (d)

Fig. 2. (a) Signal with sample points, (b) spectrum when all 13 points are used, (c) spectrum when only the last 10 points are used, (d) score function.

three sample points are discarded, the DFT gives the desired spectrum (shown in Figure 2(c)).

When $N_{\max}$ sample points of the filtered signal $y_{t-(N_{\max}-1)\delta_l}$, $y_{t-(N_{\max}-2)\delta_l}$, $\ldots$, $y_{t-\delta_l}$, y_t are available, it remains to be decided which suffix of that sequence is the best choice to determine the spectrum by the DFT. The first observation is: if the sample points cannot be cyclically continued, some extra effort has to be applied to compensate the jumps at the borders. We call the sum of the absolute Fourier coefficients the energy E of the DFT, $E = \sum_{i=1}^{N_{\max}} |Y_i|$. If some compensations at the borders are necessary, this is reflected by increased energy. Thus, a criterion for the choice of a good suffix is to take one with low energy. As a second observation, the suffix needs a certain length to include information about all relevant vibrations. Very short suffixes may have low energy, but do not reveal much about the signal's spectrum. The length of a suffix is denoted by $N_i \leq N_{\max}$ and its energy by E_i, which is computed by a DFT of that particular suffix of sample points. So, we are looking for a long suffix with low energy. We combine both requirements to a scoring function



(a) (b)

Fig. 3. (a) Complex signal and (b) the score function for the suffixes.

$$sc(N_i) = \frac{N_i \cdot \log N_i}{E_i} = \frac{N_i \cdot \log N_i}{\sum_{i=1}^{N_i} |Y_i|}$$

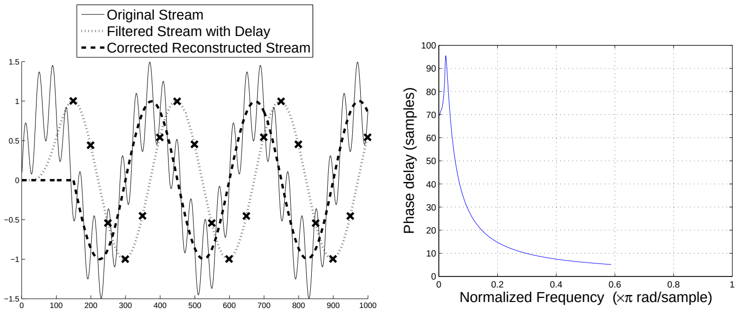
The scoring function of our example (Figure 2(d)) has a local maximum at suffix length 10, which gives the ideal spectrum, as shown in Figure 2(c). The figure shows that also small, non-informative suffixes can get a high score. To exclude those from further considerations, we set a lower bound $N_{\min}$ for the minimum length of a suffix.

A more complex signal is shown in Figure 3(a). Our proposed scoring function (Figure 3(b)) has two clear peaks for the suffixes of length 150 and 300, which are exactly the positions at which the true spectrum of the signal is revealed.

3.3 Inverse Online DFT

The main obstacle for the inverse Online DFT is the delay from the used filters. We only gain from the splitting strategy if the bandpass filters can be applied at little runtime costs. Therefore, we use fast linear filters, e.g., Chebyshev filters, where filter delay is unavoidable. The example in Figure 4(a) shows the original stream (solid curve), which consists of two sinuses $x = \sin(2\pi/300) + 1/2 \sin(2\pi/40)$. The output of the filter is the dotted curve, which is delayed with respect to the original curve. The phase delay is not a constant, but a function $\tau(f)$ depending on the normalized frequency (see Figure 4b).

Our idea is to correct the phase delay during the inverse DFT. In the example, the corrected result is the dashed curve, which is reconstructed also between the sample points. The reconstructed curve (dashed) matches almost perfectly with the original $\sin({}^{2\pi}/_{300})$.



(a) (b)

Fig. 4. The left plot shows the original stream consisting of two sinuses, the filtered stream from which sample points (x) are taken, and the reconstructed stream with phase delay correction. The right plot shows the phase delay depending on the frequency.

To explain our example formally, we assume n sample points $y_1, \ldots, y_n$ with sample distance δ. From the sample points, we get n complex Fourier coefficients $Y_1, \ldots, Y_n$ by DFT. The reconstruction of the n sample points with corrected phase delay works as follows:

$$\tilde{y}_i = re\left(\frac{2}{\sqrt{n}} \cdot \sum_{k=0}^{\lceil \frac{n}{2} \rceil - 1} Y_k \cdot \exp\big(j \cdot 2\pi \cdot k \cdot (i + \tau({}^{2\pi}/_{kn\delta}))/n\big)\right)$$

with $i = 0, \ldots, n-1$. The function $re()$ returns the real part of a complex number. In general, an inverse DFT of a real-valued stream does not need to use the $re()$ function, since – due to the symmetry property – each coefficient in the lower half has a counterpart in the upper half with the same real part, but with a negative imaginary part. Thus, the imaginary parts would be canceled out, while the real part is doubled. Here, we are additionally dealing with the phase delay correction. It is simpler to double the real part and to cut off the imaginary part explicitly.

When i is incremented in steps smaller than one, intermediate points between the sample points are computed. By that strategy, the reconstructed stream can be refined to the original resolution, or when $i > n - 1$, even future stream values can be predicted. For the final inverse Online DFT, the inverse DFTs with phase delay correction from different levels are added up at the streams original resolution.

4 Complexity Consideration

In this section, we conduct some runtime and space complexity considerations. As presented previously, when a new stream element arrives the L subintervals are completely recomputed and the sliding window is shifted. This is the most simple implementation. The main runtime factor is formed by choosing the best of all $N_{max} - N_{min}$ suffixes for each subinterval. When the DFT is implemented by FFT, this step takes time $\sum_{i=N_{\min}}^{N_{\max}} N_i \cdot \log N_i = O(N_{max}^2 \log N_{max})$ in each subinterval l , where N_i is the length of each possible suffix of sample points. Thus, the update runtime complexity of the core algorithm is $O(L \cdot N_{max}^2 \log N_{max})$, which can be simplified to $O(\lfloor \log {}^{T-1}/_{N_{\max}-1} \rfloor \cdot N_{max}^2 \log N_{max})$. This shows that the update runtime complexity is sublinear in the size of the sliding window.

Figure 5 illustrates these results on an empirical basis. We figure the runtime of the core algorithm (for $N_{\max} = 50$ and $N_{\max} = 100$) in comparison to the FFT applied to the entire sliding window, where T is varied and $N_{\min} = 1$. In both cases, the runtime grows sublinearly with the size T of the sliding window. The runtime of the Online DFT is a stair-case function: each step indicates the introduction of an additional subinterval. A better implementation is to investigate the best suffix only when the score for that suffix drops. We conducted an experiment on the sunspot data stream (see Figure 6) and computed the length of the best suffixes for each subinterval, where the window slided from time 1000 till 1500. The percentage of how often the score dropped and the suffix length changed with the next stream element is $l = 1 : 20\%, l = 2 : 76\%, l = 3 : 35\%, l = 4 : 19\%$. This shows that there is still room for optimization.

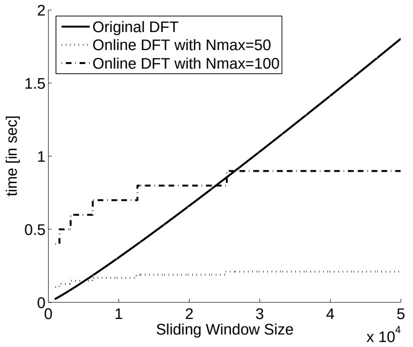
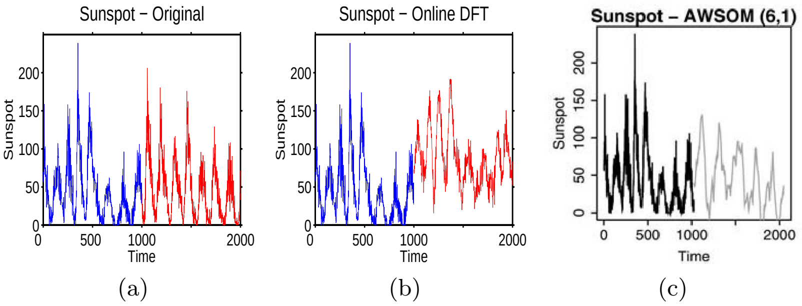


Fig. 5. Runtime Comparison of our Core Algorithm against the DFT applied on the entire sliding window.

The space complexity of our algorithm depends on the size of the sliding window. In order to be able to shift the window by one element, all elements have to be stored. We have to store $\delta_l N_{max}$ elements per subinterval, where the sum over all subintervals is in $O(T \cdot N_{max})$. The space needed can be reduced by applying time series compression techniques, e.g., piecewise linear approximation [10]. More optimization issues are focus of our ongoing work.

5 Experiments

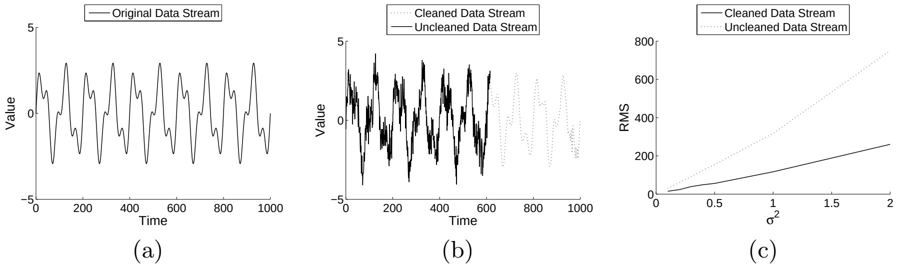
In this section, we present some applications of our Online DFT. Prediction of future behaviors of data streams and time series is important for various applications. With the Online DFT, any time forecasting is possible using the inverse transformation. Figure 6 shows the original sunspot data stream, the forecasts based on our Online DFT, and the AWSOM forecast. As code of AWSOM is not available, we took the best plot for AWSOM for that experiment from [11]. The part to be forecasted (starting at time point 1000) shows 7 peaks in the original stream. AWSOMs forecast contains 5 peaks, while the forecast of our Online DFT includes 8 peaks. Both forecasts find the average peak width quite well, even they are based on different approaches. Using the Online DFT, the average peak height is predicted more accurate.



(a) (b) (c)

Fig. 6. Experiment for forecasting data streams; (a) Original Sunspot Stream; (b) Forecasting based on Online DFT; (c) Forecasting based on AWSOM.

Based on such prediction property, we are also able to detect outliers and other kinds of deviations by comparing the current element with the prediction for that element. In general, outlier detection in data streams is a further important issue, e.g., to quickly detect situations when something unexpected happens. Furthermore, knowledge about the frequency distribution in a data stream can be used for cleaning tasks. In particular, several anomalies are easier to correct in the frequency domain than in the time domain. There are many possible reasons for dirty sensor data. This includes missing and/or incorrect values (often forming noise). Reasons for those errors may be sensors low on power, network or communication problems, inaccurate sensors, etc. Therefore, streaming sensor data must be appropriately cleaned with respect to errors.



(a) (b) (c)

Fig. 7. Experiment for noise reduction; (a) original data stream, (b) data stream after cleaning, and (c) data stream after cleaning.

To demonstrate the utilization of our Online DFT for cleaning tasks, we generated a synthetic data stream consisting of two sinuses. The original data stream is depicted in Figure 7(a). We simulated noise by adding a Gaussian distributed random number to each stream element. We investigated this noisy data stream with our proposed Online DFT approach. Figure 7(b) depicts the result of the noise reduction starting at time point 610. The noise cleaning task is processed by neglecting high frequencies and frequencies with small intensities. Using the inverse Online DFT, we are able to reconstruct a data stream with reduced noise. As the figures show, the cleaned data stream is close to the original generated data stream without noise.

To highlight the quality of this cleaning process, we determined the root mean square values (RMS) as error measures of (i) the cleaned data stream, and (ii) the data stream with noise, against the pure data stream without noise. Figure 7(c) shows the RMS depending on different variances. The higher the variance, the more noise is included in the generated data stream. With our proposed Online DFT approach, we are able to significantly reduce noise in data streams.

6 Conclusion

In this paper, we proposed a novel online model for data streams called Online DFT for estimating the frequencies of a data stream. Characteristics of our Online DFT are low update complexity with high-accuracy estimations for periods and the ability of long-range forecasting. Furthermore, we described some applications of our approach. Beside forecasting of data streams, Online DFT can be applied to detect outliers by comparing the current arriving element with the prediction. We demonstrated how to use our Online DFT for data stream cleaning tasks. Open issues for future research are specialized implementations for sensor applications as well as cost- and quality-based optimizations of the used algorithms. Additionally, we plan to do performance comparisons to related methods based on more real-world data sets.

References

1. F. Ergün, S. Muthukrishnan, and S. C. Sahinalp. Sublinear methods for detecting periodic trends in data streams. In *LATIN*, 2004.
2. Box et al. *Time Series Analysis: Forecasting and control.* Prentice Hall, 1994.
3. Elfeky et al. Periodicity detection in time series databases. *IEEE Trans. Knowl. Data Eng.*, 17(7), 2005.
4. Gilbert et al. Surfing wavelets on streams: One-pass summaries for approximate aggregate queries. In *VLDB*, 2001.
5. Indyk et al. Identifying representative trends in massive time series data sets using sketches. In *VLDB*, 2000.
6. Keogh et al. Finding surprising patterns in a time series database in linear time and space. In *Proc. of the ACM SIGKDD*, 2002.
7. Oppenheim et al. *Discrete-time signal processing.* Prentice-Hall, Inc., 1989.
8. Ch. Faloutsos, M. Ranganathan, and Y. Manolopoulos. Fast subsequence matching in time-series databases. In *SIGMOD*, 1994.
9. A. C. Gilbert, S. Guha, P. Indyk, S. Muthukrishnan, and M. Strauss. Near-optimal sparse fourier representations via sampling. In *STOC*, 2002.
10. D. P. Kacso. Approximation by means of piecewise linear functions. *Results in Mathematics*, 35, 1999.
11. S. Papadimitriou, A. Brockwell, and Ch. Faloutsos. Adaptive, unsupervised stream mining. *The VLDB Journal*, 13(3), 2004.
12. A. S. Weigend and N. A. Gerschenfeld. *Time Series Prediction: Forecasting the Future and Understanding the Past.* Addison-Wesley, 1994.
13. Y. Zhu and D. Shasha. StatStream: Statistical Monitoring of Thousands of Data Streams in Real Time. In *VLDB*, 2002.