Department of Informatics, University of Zürich

Facharbeit

Centroid Decomposition Based Recovery for Segmented Time Series

Jonathan Nagel

Matrikelnummer: 08-737-421

Bülach, Zürich, CH

Email: jonathan.nagel@uzh.ch

September 5, 2013

supervised by Prof. Dr. M. Böhlen and M. Khayati

University of Zurich[UZH]

Department of Informatics

Abstract

The application of the Centroid Decomposition technique has been used for the recovery of missing values in time series. This technique uses the entire time series in order to recover a block of missing values. However, no work was proposed to evaluate the recovery accuracy using segments of time series and to compare it against the recovery using the entire time series. This is main goal of this work. In fact, we propose to combine two segmentation techniques i.e., sliding windows and bottom up and we apply applied them for the recovery in different types of time series. The results show that in some cases the recovery performs better when only a segment of the entire time series is used.

Contents

List of Figures

1 Introduction

1.1 Context and Motivation

Matrix decomposition on based recovery technique is able to accurately recover missing values in time series using a set of time series [KB12]. The Centroid decomposition in a matrix decomposition technique that has been applied for the recovery of missing values using entire data sets [Bö13]. The application of the previous technique on synthetic data showed that in some cases the recovered values imitate the shape of the time series used for the recovery. For such cases the recovery technique is dependent on the shifts between the time series with the missing values and the other time series.

The main task of this work is to evaluate the impact of using segmentation of the time series the accuracy of the recovery. In order to achieve this goal, first a combination of two segmentation techniques, i.e., sliding windows and bottom up techniques are applied to segment time series. Then, the obtained segments are used for the recovery of missing blocks in time series. We present an empirical evaluation with different setups of time series.

1.2 Contributions

The main contributions of this thesis are the following:

- Implement the Centroid Decomposition and apply it for the recovery of missing values in time series
- Implement the segmentation techniques sliding windows and bottom up to create segments that can be used as input of the recovery process.
- apply the implemented algorithms for real world hydrological dataset
- compare the accuracy of recovery using segmented time series against the entire time series

1.3 Structure of Thesis

In Chapter 2, we introduce the main concepts that we be used throughout this report. In Chapter 3, we introduce the different segmentation techniques of time series. Chapter 4, we experimentally evaluate the accuracy of the recovery based on segmented time series. In chapter 5, we summarize the main contributions and we discuss future works.

2 Background

2.1 Time Series

A time series S_i is a sequence of n observations x_j, such that $S_i = \{x_1, \ldots, x_n\}$. Each value x_j has a corresponding timestamp t_i, such that every observation is a pair of $\{x_i, t_i\}$. We use the following notation throughout this report:

- S_{ref}: The time series that contains the missing values that shall be recovered
- S_{inf}: The series containing the information (e.g. their shape) used to recover the S_{ref}
- S_{org}: The time series S_{ref} before some values are skipped out and are declared as missing values.
- S_{seg}: the S_{ref} recovered by using only a segment of the whole data set
- S_{noSeg}: the S_{ref} recovered by using the whole data set

Missing values occur, when for a specific time series S_1, no value is recorded for an existing timestamp t_i. Then, the time series contains a missing value or block compared to another time series S_2. The aim is to accurately recover the missing values in time series. In the following M_i denotes the set of missing values

$$M_i = [a_{v1}, \ldots, a_{w1}]$$

where v is the first and w the last timestamp containing a missing value in the first column of A, which is the S_{ref}.

2.2 Pearson Correlation

In this report, the Pearson correlation ρ is used as the method to compare any time series with each other. This correlation describes the relation between two time series S_1 and S_2 such that ρ = 1 is a fully positive linear relation, r = -1 is a fully negative linear relation and ρ = 0 is no linear relation at all. The closer the absolute value of ρ to 1 is, the better one series can be used in the recovery process. In this report the Pearson correlation is used to compare different time series or segments with each other.

In order to compare different time series with each other, we use the correlation value between them. In case of the comparison of different segments with each other, the average of all the correlations between each series off S_{inf} and the S_{ref} is calculated in each segment.

2.3 Centroid Decomposition

The centroid decomposition is a technique to decompose a matrix $A^{n\times m}$ into two matrices $L^{n\times m}$ and $R^{m\times m}$ such that $A = LR^T$ where R^T is the transpose of R.

Each value a_{ij} denotes the value of the variable i for the entity j. R is called the factor matrix with the column vectors $[r_1, \ldots, r_m]$ L is called the loading matrix with the column vectors $[l_1, \ldots, l_m]$.

$$A = \begin{bmatrix} a_{11} & \ldots & a_{1j} & \ldots & a_{1m} \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ a_{i1} & \ldots & \ldots & \ldots & \ldots \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ a_{n1} & \ldots & \ldots & \ldots & a_{nm} \end{bmatrix}$$

For more detailed explanations about the algorithm of the centroid decomposition one may read [CRF02]. Algorithm 1 describes the Centroid Decomposition process of an input matrix A.

Algorithm 1: CD()

Data: A^{nxm}
Result: L, R

```
i = 1;
A_i = A;
m = A.numberOfColumns;
while m > 0 do
    z = FindSignVector(A_i);
    c_i = A_i^T z;
    r_i = c_i / ||c_i||;
    l_i = A_i r_i;
    A_i = A_i − b_i v_i^T;
    R = append(R, r_i);
    L = append(L, l_i);
    i++;
    m−;
return L, R;
```

2.3.1 Frobenius Norm

The Frobenius norm is defined as the difference between A_i and A_{i+1}. This difference is defined as follows:

$$\|A\|_F := \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n} a_{ij}^2}.$$

This norm will be used in the recovery process of missing values in time series.

2.3.2 Recovery Process

To be able to recover missing values in a time series using the centroid decomposition based recovery technique, these missing values have to be initialized. So for each pair $\{x_i, t_i\}$ in a time series S_i where no x_i exists, a value is created.

For the recovery, the input matrix A_i is decomposed into L_i and R_i. Next the matrix LR^T is calculated out of l_i and r_i. l_i is the first column of L_i and r_i is the first row of R_i. Algorithm 2 describes the recovery process.

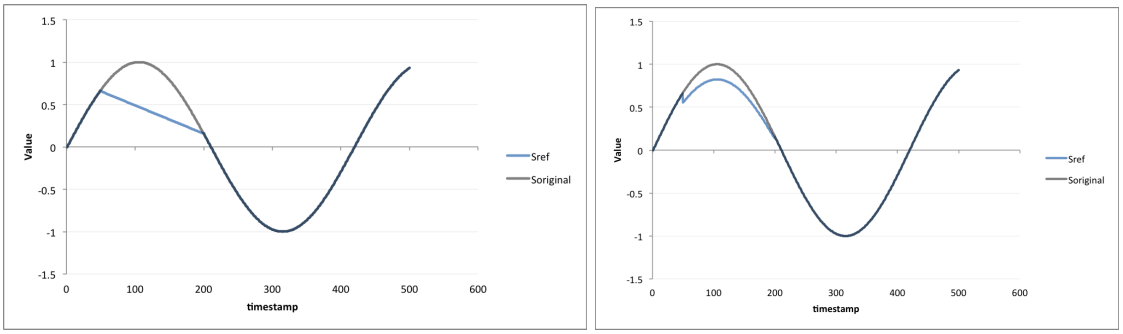
Algorithm 2: RecoveryProcess

Data: A: with S_{ref} in the first column and S_{inf}
Data: M: set of timestamps of the missing values
Result: A: A with recovered missing values

```
1  i = 1;
2  j = 1;
3  A_i = A;
4  A_{i+1} = A;
5  repeat
6      A_i = A_{i-1};
7      L_i, R_i = centroidDecomposition(A);
8      l_i = L_i.getFirstColumn();
9      r_i = R.getFirstColumn();
10     x = l_i r_i;
11     for j=1; j<=M.getSize() do
12         A_i[M_j, 1] = x[M_j, 1];
13     i++;
14 until ||A_{i-1}||_F − ||A_i||_F < ε_F;
```

In each iteration of Algorithm 2 the values in A_i at the timestamps a_{v1} to a_{w1} are replaced with the values with the same timestamps from LR^T with the same timestamps. Therefore, the obtained matrix A_{i+1} is then used as the new A_i in the next loop. This procedure is repeated until a stopping criteria is met.

In order to perform an accurate recovery, a precise stopping criteria has to be set. The obtained matrix A_i should be different from the matrix A_i, because the values on a_{v1} to a_{w1} are meant to have changed and by that recover the data differently than the initialized values. If these values did not change strongly, further iterations of the loop will not change them any stronger. So it would not make sense to continue the recovery process.



(a) Example for a non recovered time series

(b) Example for a recovered time series

Figure 2.1: S_{ref} before and after recovery using the S_inf

A value of the Frobenius norm between A_i and A_{i+1} that is below a predefined threshold value, denoted as ε_F, will terminate the loop of the recovery process.

3 Segmentation

The main task of this report is the comparison of the recovery accuracy when a segmentation technique is used and compare it with the recovery using the entire time series. For this segmentation a combination of the sliding windows and the bottom up method is used [KCHP01].

3.1 Sliding Windows

In the sliding windows method, an initial segment of size $minS$ is defined and than grown by adding neighboring values. After each update, the segment is evaluated and while a predefined stopping criteria is not yet reached, the growing of the segment is continued. This way a stream of data can be segmented. The segments created by the sliding windows method are saved in a sequence denoted Seg as triplets seg_i containing the lower bound lb_i and upper bound ub_i and the corresponding correlation ρ_{segi} of the segment seg_i as follows: $Seg = [seg_1, \ldots, seg_n]$

One of the main advantages of the sliding windows method is the online Applicability. Because it parses through the data it is not necessary to have the whole data set from the beginning. This makes it online applicable.

3.1.1 Application of Sliding Windows

The aim of applying the sliding windows is to separate those segments that have more similarities with the S_{org} from those that have less. As described above, a high correlation ρ indicates, whether one series is similar to another one and though useful to recover another. So the stopping criteria is defined using the Pearson correlation. Normally a constant error ε is chosen and used to decide, whether more data is added to a segment or not ([KCHP01]). This way segments are created with $\rho >= \varepsilon$. For the function the sliding windows shall have in this report, a constant ε does not lead to the desired result.

To separate the highly correlated parts from the lowly correlated ones, a variable error ε_{sw} has to be defined. The aim is, to find the pair of timestamps t_i, t_{i+1} between which the local correlation, e.g. the correlation of a subsegment of A, changes from high to low or vice-versa. If such a pair is found, t_i is set as the upper bound of the actual segment and t_{i+1} is the lower bound of the next segment. To find such a pair of timestamps the error ε_{sw} is defined as a relative value of the initial correlation ρ_{init} of the actual segment as follows: $\varepsilon_{sw} = \rho_{init}/d$

The variable d is a predefined constant which defines, how sensible the stopping criteria for the sliding windows is. The higher it is, the earlier the growing of the segment stops and though the segments will be smaller. The ρ_{init} is calculated from the initial segment before it is grown. Algorithm 3 introduces the application of the sliding windows technique.

Algorithm 3: slidingWindows

Data: Matrix A
Data: ε_{sw}
Data: minS
Result: seg: a sequence saving the segments and their ρ_{segi}

```
1  lb = 1;
2  anchor = minSize-1;
3  i = 1;
4  repeat
5      ρ_init = getCorrelation(A, lb, anchor);
6      repeat
7          anchor++;
8          ρ_next = getCorrelation(A, lb, anchor);
9      until r_next > ρ_init − ε_sw ∧ ρ_next < ρ_init + ε_sw;
10     seg_i1 = lb;
11     seg_i2 = anchor;
12     seg_i3 = ρ_next;
13     lb = anchor+1;
14     anchor = lb+minSize-1;
15 until anchor < A.getNumberOfTimestamps;
16 return seg;
```

where getCorrelation(X, y, z) returns the average correlation between the S_{ref} and each of the S_{inf} of the subsegment A(y, z).

3.2 Bottom Up

The bottom up method does what one could guess: It segments a whole data set into the smallest possible segments and merges them until a stopping criteria is met. More precisely: For each segment the cost of merging is calculated and then the merge with the lowest cost is done. This is repeated until the cost for the next merge exceeds a threshold.

3.2.1 Application of Bottom Up

First applying the sliding windows to create segments and then apply the bottom up method using these segments instead of the smallest possible ones, has already been proposed in the context of representing data [KCHP01]. In this report the same combination is used in to recover missing values in time series.

The aim of the bottom up method is to find a segment seg_i that contains a block of missing values and has ρ as high as possible. So the idea of the bottom up method is only applied to the segments containing missing values and not to all the segments in seg. The criteria whether the merging is done or not is as well different to the above described criteria that uses

a maximum cost threshold. Because the aim is, to find a highly correlated segment, merging a segment seg_i is not done if the correlation of the new segment would be smaller then ρ_{segi}. If merging a segment seg_i with one of it's neighboring segments seg_{i+1} and seg_{i-1} would increase the correlation, the merge that increases the correlation more is made.

The obtained segment is then the input matrix A in the in Algorithm 2. If there is another seg_i in seg with missing values, the bottom up is applied for it using the non-merged segments obtained from the sliding windows. So, the merged segment obtained through the bottom up method is not saved. In case where low correlated segment seg_i is merged with a segment seg_{i+1} also containing missing values, only the values of seg_i are recovered. The aim of this is to avoid that segments that contain missing values and already have a high correlation are merged with preceding segments, that would decrease the correlation.

The effect of the bottom up application in this report should though be, that the highly correlated segments seg_i are directly used for the recovery and that the lowl correlated seg_i are merged before applying the recovery.

Algorithm 4: bottomUp

Data: A
Data: seg
Data: mv: indicates which seg_i contains the missing values and shall be processed
Result: A_{sub}: Subsegment of A

```
1  A_sub= getSubsegment(A, seg.getLowerBound(mv), seg.getUpperBound(mv);
2  ρ = seg.getCorrelation(mv);
3  ρ_right = seg.getCorrelation(mv+1);
4  ρ_left = seg.getCorrelation(mv-1);
5  while ρ < ρ_left ∨ ρ < ρ_right do
6      if (ρ_left > ρ_right) then
7          A_sub = mergeLeft(A, seg, mv)
8      else
9          A_sub = mergeRight(A, seg, mv);
10         mv = mv − 1;
```

Where getSubsegment(X, y, z) returns the subsegment of X(y, z), mergeLeft and mergeRight return the subsegment of A with a new lowerBound or a new upperBound respectively and getCorrelation(x) returns the correlation of the segment seg_x.

4 Experiments

4.1 Evaluation Strategy

In order to compare the accuracy of recovery produced based on segmentation strategy, we compare the resulting recovered time series S_{seg} and S_{noSeg}. We compute the mean square error MSE between each of the recovered time series and the S_{org}.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2.$$

where $\hat{Y}_i$ is the original value of S_{org} and Y_i is the recovered value.

Secondly the two $MSEs$ are subtracted, such that the improvement gets positive, when the MSE_{seg} between the S_{seg} and the S_{org} is smaller than the MSE_{noSeg} between S_{noSeg} and the S_{org}. The improvement is computed as follows

$$\text{impr} = MSE_{noSeg} - MSE_{seg}.$$

Further the correlations will be compared with the corresponding ***MSEs***. So the MSE_{seg} will be opposed to the ρ_{segi} and the MSE_{noSeg} will be compared to the correlation ρ_{whole}, which is calculated as the average of all the correlations between S_{ref} and each S_{inf}.

4.2 Input Matrices

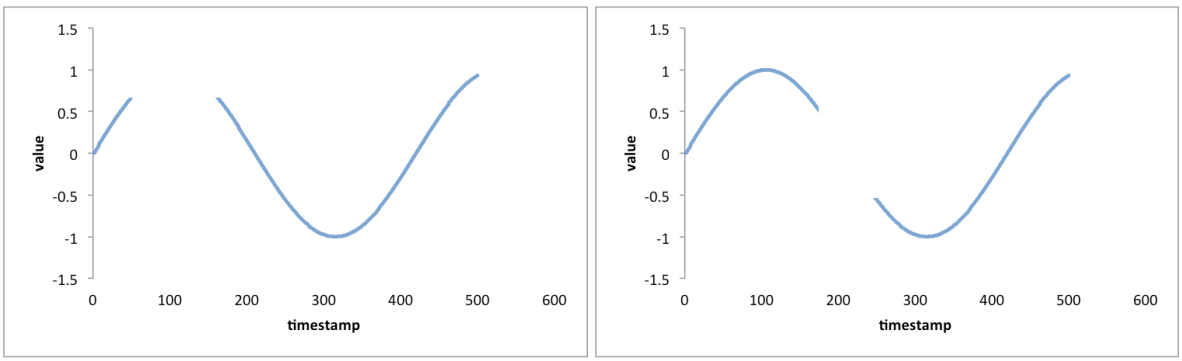
To test the differences between no segmented and segmented time series, a sine-curve is created and used as the S_{org}. It's frequency f_{org} is used for the calibration and is though set to 1. The Offset is 0.

For the S_{ref} in *A*, the frequencies are multiplied by a variable ***f-multiplier*** and the ***phase*** in degrees is used to denote the shift.

The missing values in S_{ref} are created in blocks either on a peak of the S_{org} or between peaks, as shown in figures 4.1(a) and 4.1(b). For this the values in S_{org} are skipped out and replaced with interpolated values.

4.3 Pretesting

First the recovery is made with S_{inf} = S_{org} to test, what the minimum MSE_{org} is. The MSE_{noSeg} and the MSE_{seg} are identically in this case, because the segmentation methods



(a) missing values block at a peak

(b) missing values block between two peaks

Figure 4.1: Positioning of the missing values blocks

used creates only one segment containing the entire data set. The obtained value is an orientation for the other results. The MSE_{org} is set to -0.0025.

4.4 Irregular Data

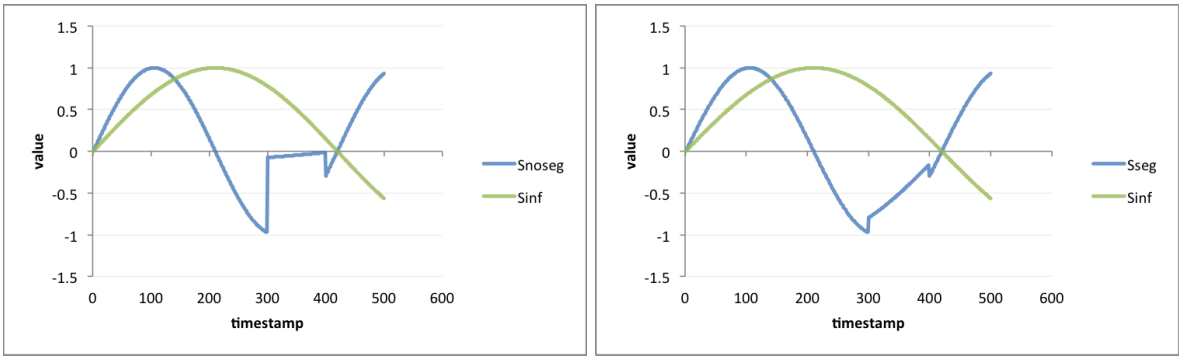
4.4.1 Recovery of Data at a Peak

When the block of missing values is set at a peak and the *f-multiplier* is set to 0.5 or 1.5 the *impr* gets positive. When the f-multiplier is set to 2 the improvement gets worse.

f-multiplier	MSE_{noSeg}	MSE_{seg}	*impr*
0.5	0.149	0.0577	0.0912
1.5	0.1413	0.0385	0.1028
2	0.1455	0.2291	-0.0835

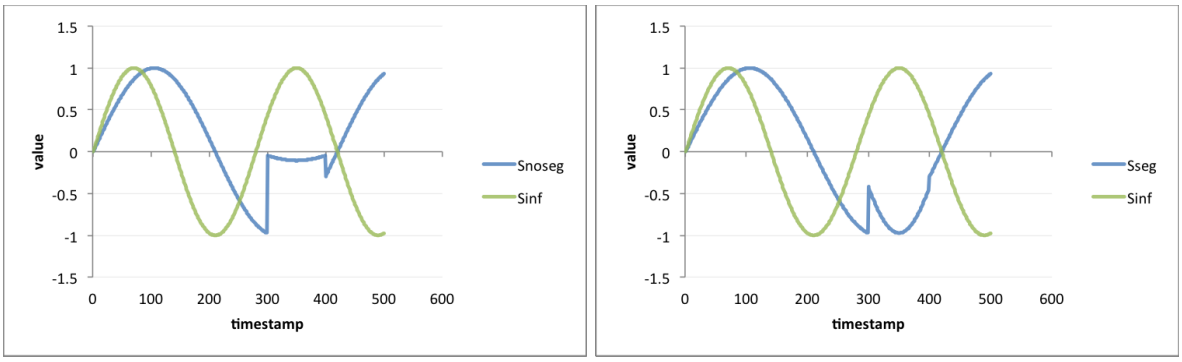
For all other tested values of *f-multiplier* no linear relation to the *improvement* or any other regularity can be found. What can clearly be seen is, that the MSE_{noSeg} varies less than the MSE_{seg}. So the low values as well as the high values of *impr* are due to the variation of the MSE_{seg}.

When comparing the figures 4.2, 4.3 and 4.4 the different reactions on different frequencies can be seen very clearly. The recovery, when using the entire dataset (S_{noSeg}), does only change little when tested with different frequencies for S_{inf}. So the small differences between the figures 4.2(a), 4.3(a) and 4.4(a) are because of the recovery using the entire data set. The values are recovered with a value close to zero, which is the same value as the average of all values in a sine-curve. The average value of a sine-curve changes, when only a segment is taken, which can be seen in the figures 4.2(b), 4.3(b) and 4.4(b) which show the results when the segmentation is used. The recovered values are very sensible to different frequencies and do though recover differently for each frequency of S_{inf}.



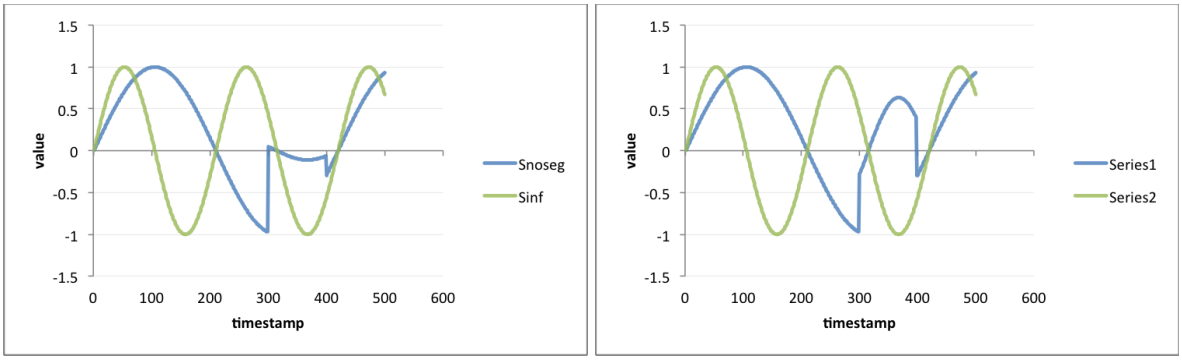
(a) Snoseg (b) Sseg

Figure 4.2: recovered series with f-multiplier = 0.5 with missing values at peak



(a) Snoseg (b) Sseg

Figure 4.3: recovered series with f-multiplier = 1.5 with missing values at peak



(a) Snoseg (b) Sseg

Figure 4.4: recovered series with f-multiplier = 2.0 with missing values at peak

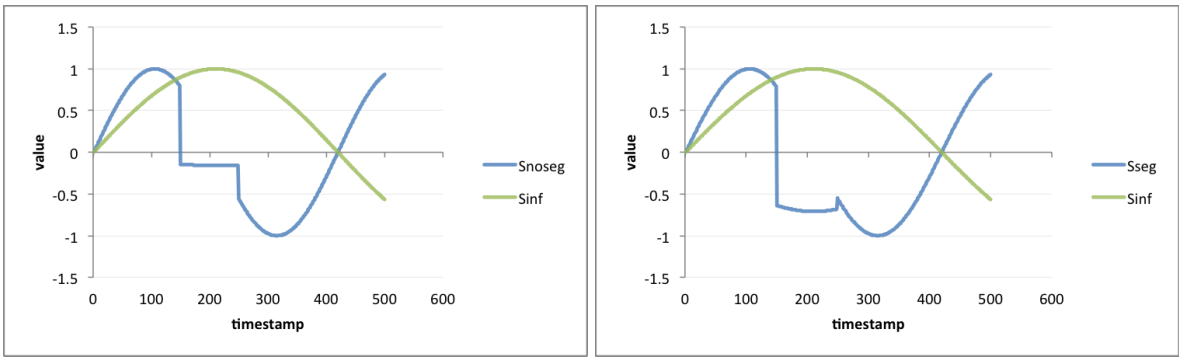
4.4.2 Recovery of Data between Peaks

When the block of missing values is set between two peaks, *impr* has the opposite results.

f-multiplier	MSE_{noSeg}	MSE_{seg}	improvement
0.5	0.084	0.1676	-0.0836
1.5	0.0745	0.1295	-0.0550
2	0.0958	0.0135	0.0822

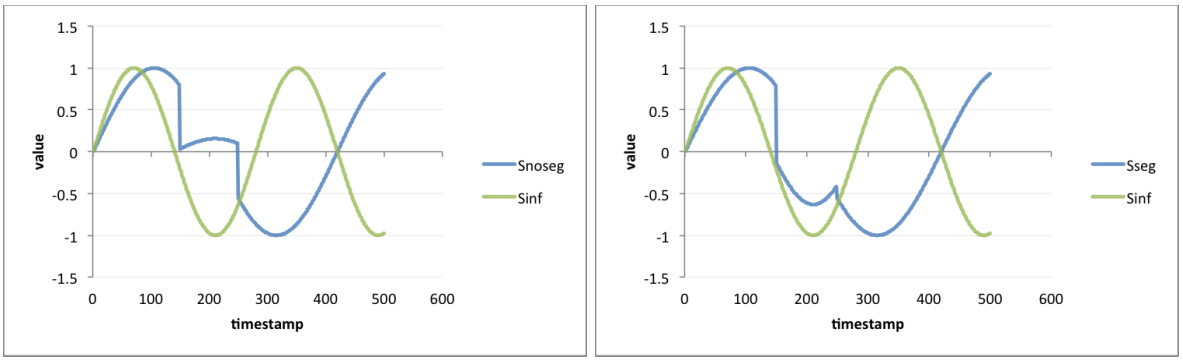
Again the MSE_{seg} has a bigger variation then the MSE_{noSeg}.

In the figures 4.5, 4.6 and 4.7 the same can be observed as when the recovered values are on a peak. When the segmentation is made, the recovery is more sensible to different frequencies of the S_{inf}.



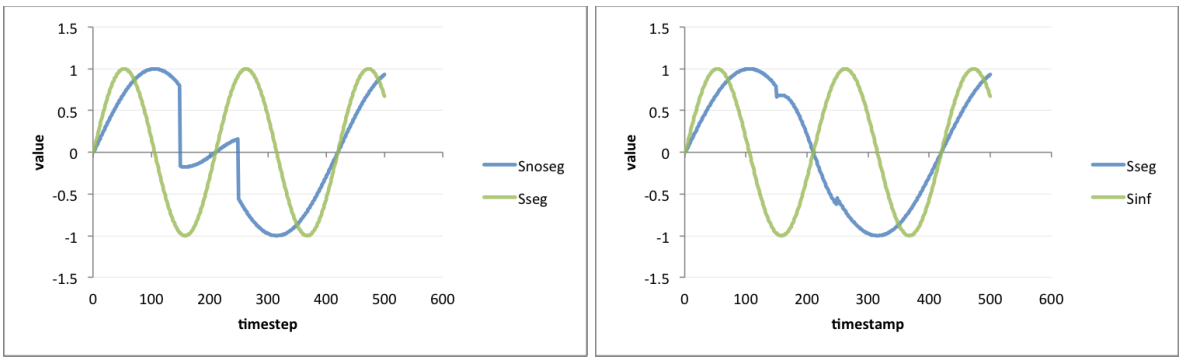
(a) Snoseg (b) Sseg

Figure 4.5: recovered series with f-multiplier = 0.5 and missing values between peaks



(a) Snoseg (b) Sseg

Figure 4.6: recovered series with f-multiplier = 1.5 and missing values between peaks



(a) Snoseg (b) Sseg

Figure 4.7: recovered series with f-multiplier = 2.0 and missing values between peaks

4.4.3 Correlations

When looking at the correlations and their corresponding *MSEs* than a big difference between the ρ_{whole} and the ρ_{segi} appears. Interestingly for the results of *no segmentation* the absolute values ρ_{whole} are always lower than those of ρ_{segi}. Still the MSE_{seg} and the MSE_{noSeg} have *impr* values that lie in a similar range. So: no clear relation between the *MSEs* and the corresponding correlations can be seen.

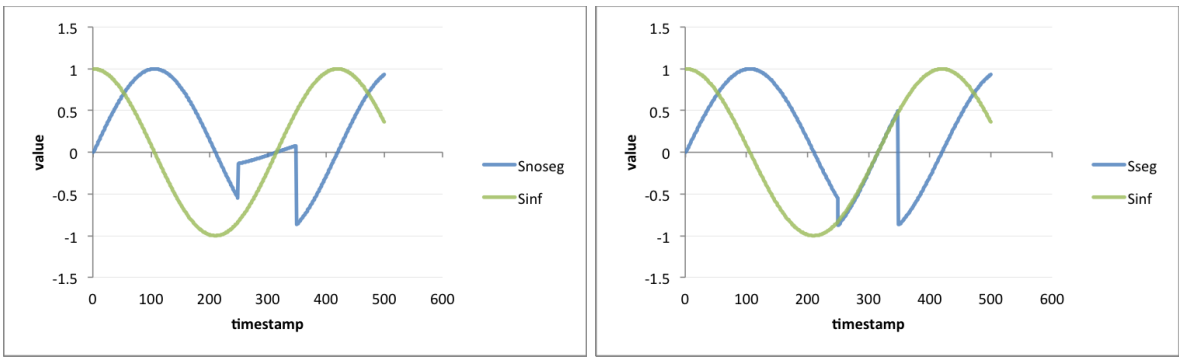
MSE_{noSeg}	ρ_{whole}	MSE_{seg}	ρ_{segi}
0.0839	-0.2529	0.1675	-0.4675
0.0745	-0.1512	0.1295	0.8971
0.0958	0.1387	0.0135	-0.9547

4.5 Shifted Data

4.5.1 Recovery of data at a peak

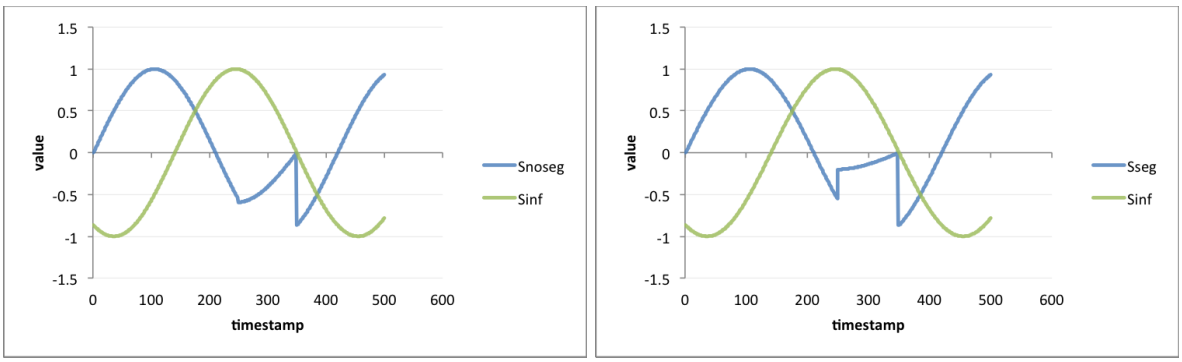
When the block of missing values is at a peak, the *impr* has no linear relation to the phase at all. Phases of 60 and 90 degrees result in a positive *improvement* and the phase 70 degrees produce a negative *improvement*. 80 degrees shift has almost the value 0 and the highest absolute value is at a shift of 240 degrees.

phase	MSE_{noSeg}	MSE_{seg}	*impr*
60	0.1025	0.0806	0.0219
70	0.1281	0.1564	-0.0283
80	0.1525	0.1491	0.0033
90	0.17	0.1387	0.0312
240	0.1025	0.1499	-0.0473



(a) Snoseg (b) Sseg

Figure 4.8: recovered series with phase = 90 and missing values at a peak



(a) Snoseg (b) Sseg

Figure 4.9: recovered series with phase = 240 and missing values at a peak

So as well as for the irregular S_{inf} no simple linear relation can be found. It is not possible to predict the outcome of the recovery based on the shift.

The figures 4.8(a) and 4.9(a) show, that taking the whole dataset as input for the recovery does not lead to big differences, even though the shift of the S_inf are very different. As it is for different frequencies, the segmentation is more sensible to shifts too. This can be seen in the figures 4.8(b) and 4.9(b).

4.5.2 Recovery of data between peaks

When the block of missing values is set between two peaks, again no direct relation between the phase and the resulting *impr* can be seen. The most catching similarity to the results when the block of missing values is set at a peak is, that the highest absolute value of *impr* is found as well, when the phase is 240 degrees.

In contrast to the results presented above, there is a small general trend of the improvement. When calculating the average of all the resulting *impr* for shifts from 10 to 350 degrees in steps of 10 degrees, one gets the value 0.009. Even though this is a relative small value it shows, that the *segmented* method in average performs better.

4.6 Summary

The main results of the experiments are:

- The *impr* value is more sensible to irregular S_{ref} than to shifts.
- The variation of *impr* is mostly due to the segmented inputs.
- For both, phase and *f-multiplier*, no linear relation with the *impr* can be found.
- $\rho_{segi} > \rho_{whole}$ does not imply a higher *impr*.
- when the S_{inf} are shifted and the block of missing values is between peaks, the results of *impr* are slightly positive in average.

5 Conclusion

Using the segmentation for the recovery of the initialized values is more sensible to shifts and frequencies of the S_{inf}. In some cases this is an advantage. The main difference between using the entire dataset and using the segmentation is, that the recovered values are closer to zero when using the entire dataset. So if the average of the value of the S_{inf} is close to zero but the original values of S_{ref} are not, then the segmentation has a chance to recover more precisely. But at the same time it has a higher risk to recover worse.

The overall conclusion though is, that there are cases, in which the segmentation can provide a better result of the recovery, because it reacts stronger to the local values of S_{inf} around the values that are recovered. So when a dataset is highly irregular, the segmentation seems to be the better choice.

An additional conclusion has been made when the first implementation of the recovery process had been tested with real world data. The recovered values were more or less the same as the initialized had allready been. The reason for this was the idea of calculating the x not just out of the first column l_i of L_i and the first row r_i of R_i. Instead the matrices L_i and R_i had just been reduced by one column and one row respectively and were then multiplied to obtain the x. This led to a second implementation which is described in the 2.

Bibliography

[Bö13] Eszter Börzsönyi. Recovery of missing values based on centroid decomposition. *MSc Thesis*, 2013.

[CRF02] Moody T. Chu, Robert, and E. Funderlic. The centroid decomposition: Relationships between discrete variational decompositions and svds. *SIAM J. Matrix Anal. Appl*, 23:1025–1044, 2002.

[KB12] M. Khayati and M. Böhlen. Rebom: Recovery of blocks of missing values in time series. In *Proceedings of the 2012 ACM International Conference on Management of Data*, COMAD '12, pages 44–55. Computer Society of India, 2012.

[KCHP01] Eamonn J. Keogh, Selina Chu, David Hart, and Michael J. Pazzani. An online algorithm for segmenting time series. In *Proceedings of the 2001 IEEE International Conference on Data Mining*, ICDM '01, pages 289–296, Washington, DC, USA, 2001. IEEE Computer Society.