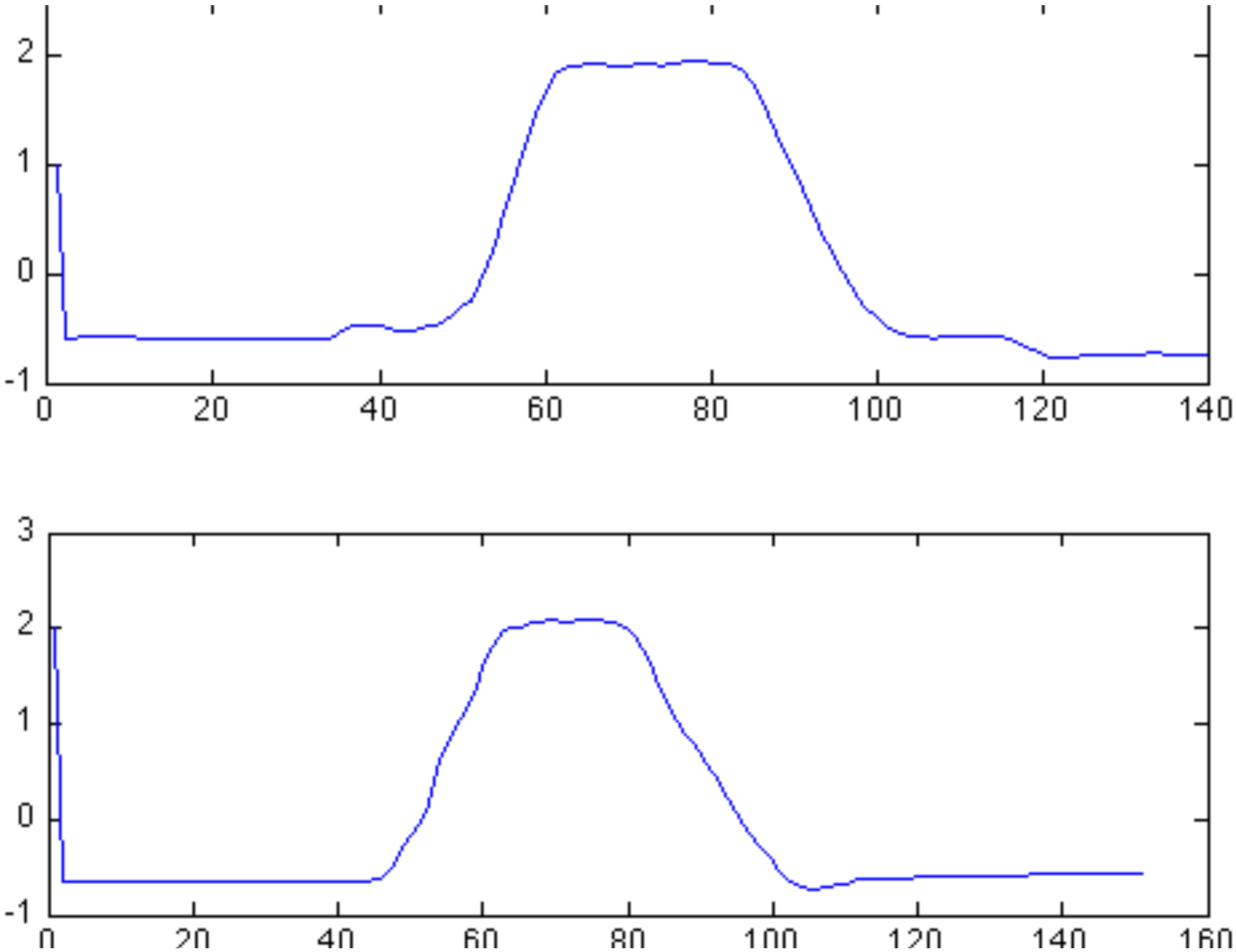
Dynamic Time Warping

Quim Llimona Torras
Journal Club 2011. MTG - UPF

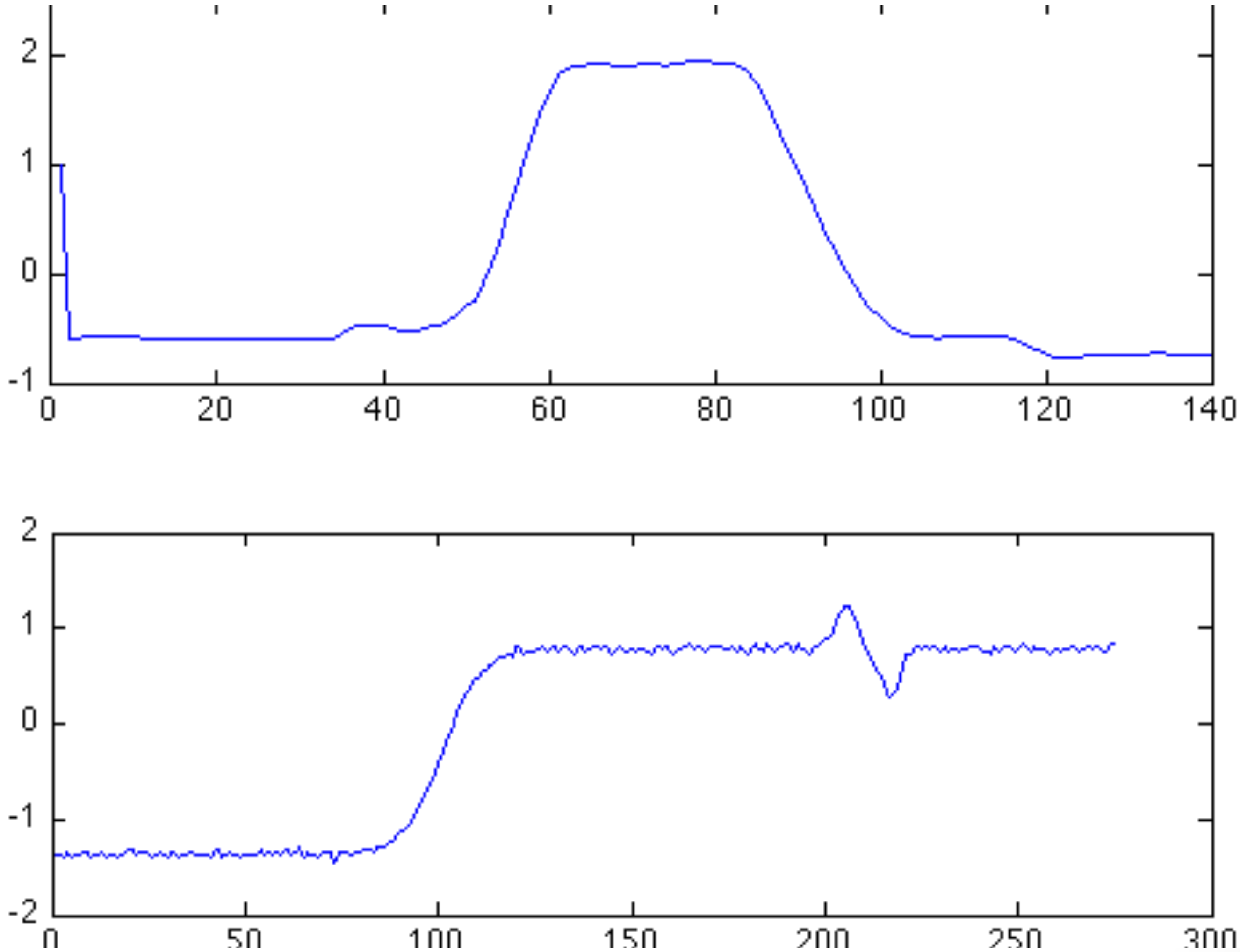
Time series similarity

WHERE DID IT COME FROM?

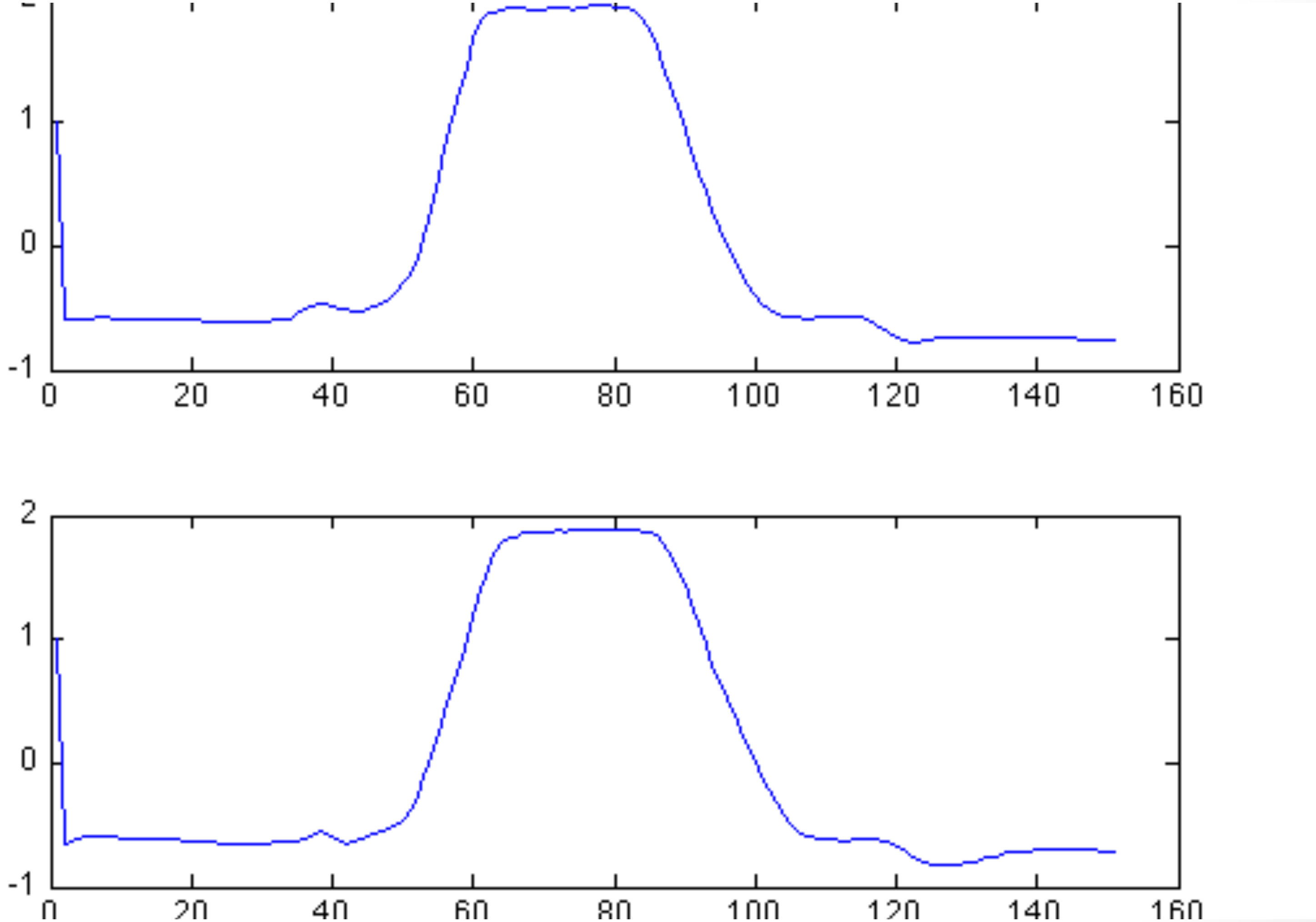
How similar are 2 time series?



How similar are 2 time series?



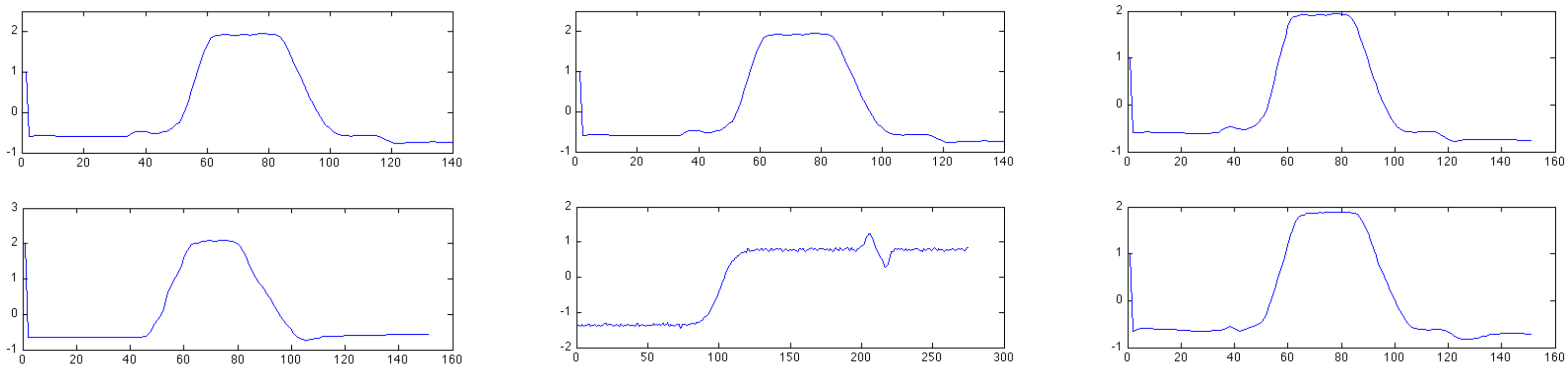
How similar are 2 time series?



How similar are 2 time series?

How similar are 2 time series?

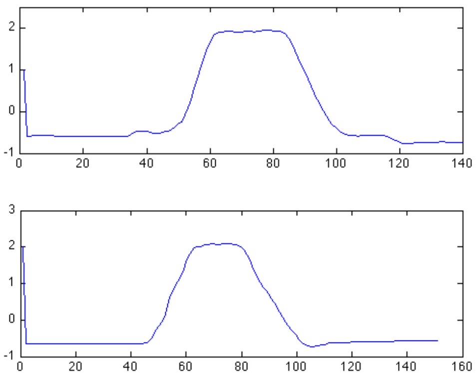
Euclidean Distance: $\sqrt{(a_1-b_1)^2+(a_2-b_2)^2+\cdots+(a_n-b_n)^2}=\sqrt{\sum_{i=1}^{n}(a_i-b_i)^2}.$



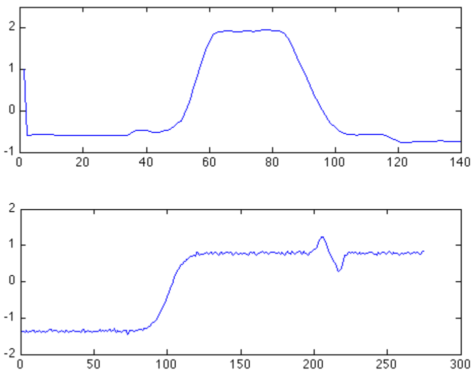
How similar are 2 time series?

Euclidean Distance: $\sqrt{(a_1-b_1)^2+(a_2-b_2)^2+\cdots+(a_n-b_n)^2}=\sqrt{\sum_{i=1}^{n}(a_i-b_i)^2}.$

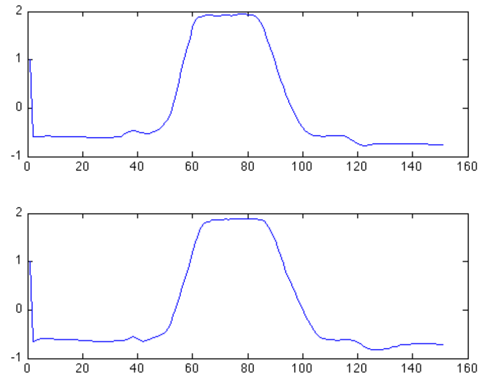
2.0895



18.7087

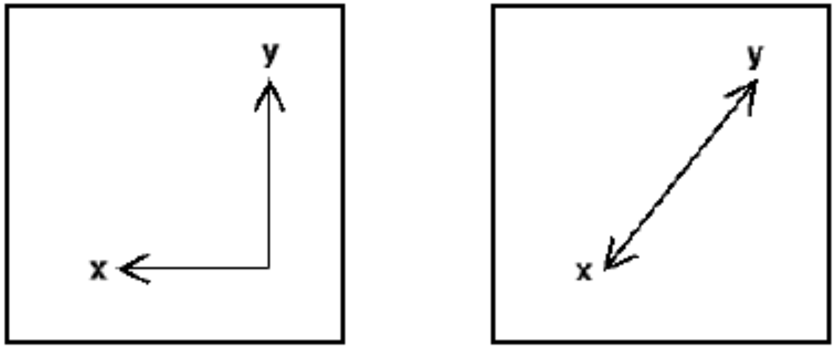


2.3226



They didn't even have the same length!

Other similarity measures



Manhattan

Euclidean

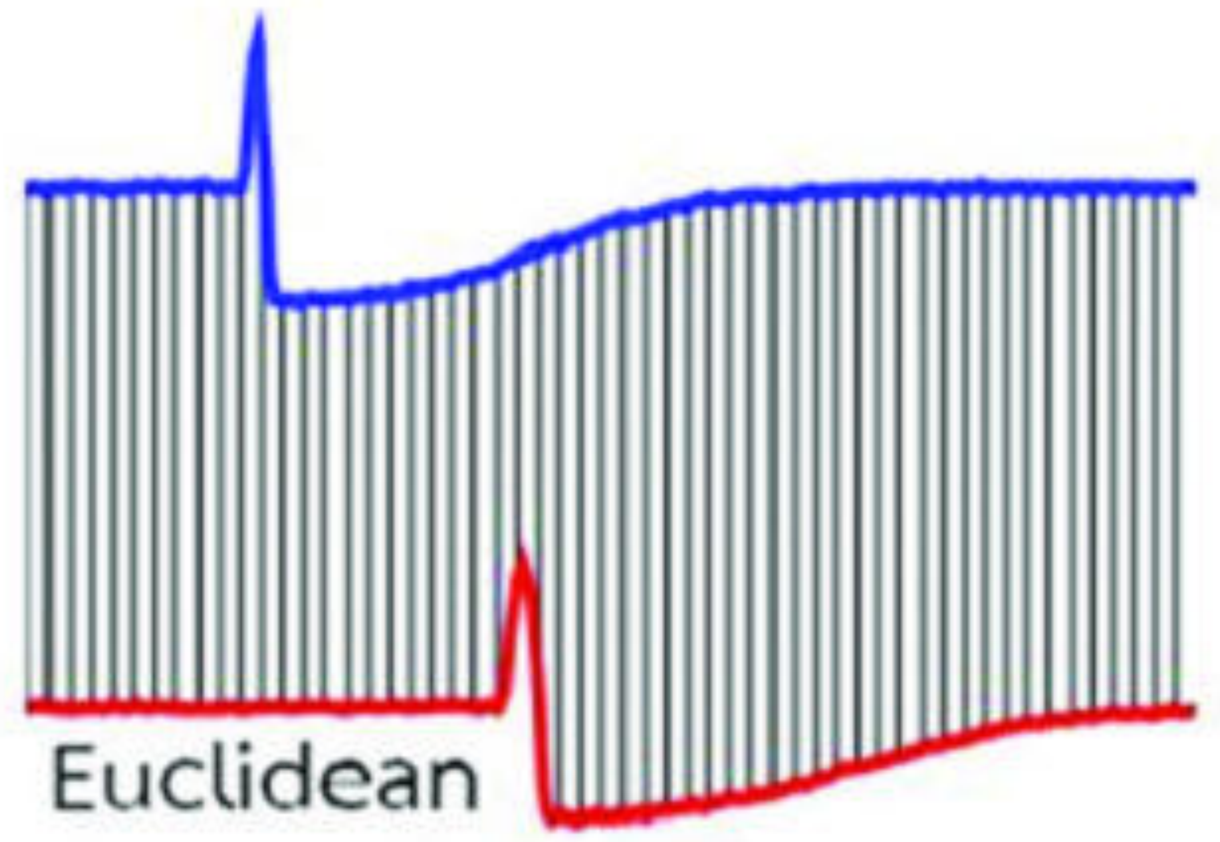
Other similarity measures

Euclidean (or Cartesian) distance	$D_{[2]}(\mathbf{x},\mathbf{y}) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$
Chebyshev distance	$D_{[\infty]}(\mathbf{x},\mathbf{y}) = \max_{i=1}^{n} \lvert x_i - y_i \rvert$
Manhattan (city-block) distance	$D_{[1]}(\mathbf{x},\mathbf{y}) = \sum_{i=1}^{n}\lvert x_i - y_i \rvert$
Minkowsky distance	$D_{[p]}(\mathbf{x},\mathbf{y}) = \left[\sum_{i=1}^{n}\lvert x_i - y_i \rvert^p\right]^{\frac{1}{p}}$
Weighted Minkowsky distance	$D_{[p,\mathbf{w}]}(\mathbf{x},\mathbf{y}) = \left[\sum_{i=1}^{n} w_i \lvert x_i - y_i \rvert^p\right]^{\frac{1}{p}}$
Mahalanobis distance	$D(\mathbf{x},\mathbf{y}) = \lvert \det \mathbf{C} \rvert^{1/n} (\mathbf{x}-\mathbf{y})^{\mathrm{T}} \mathbf{C}^{-1} (\mathbf{x}-\mathbf{y})$
Generalised Euclidean (quadratic) distance	$D(\mathbf{x},\mathbf{y}) = (\mathbf{x}-\mathbf{y})^{\mathrm{T}} \mathbf{K} (\mathbf{x}-\mathbf{y})$
Correlation coefficient	$\rho(\mathbf{x},\mathbf{y}) = \frac{\sum_{i=1}^{n}(x_i - \bar{x}_i)(y_i - \bar{y}_i)}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x}_i)^2 \sum_{i=1}^{n}(y_i - \bar{y}_i)^2}}$
Relative entropy (Kullback-Leibler divergence)	$D(\mathbf{x} \Vert \mathbf{y}) = \sum_{i=1}^{n} x_i \log \frac{x_i}{y_i}$ when $\sum_{i=1}^{n} x_i = \sum_{i=1}^{n} y_i = 1$
χ^2-Distance	$D_{\chi^2}(\mathbf{x},\mathbf{y}) = \sum_{i=1}^{n} \frac{(x_i - y_i)^2}{y_i}$ when $\sum_{i=1}^{n} x_i = \sum_{i=1}^{n} y_i = 1$

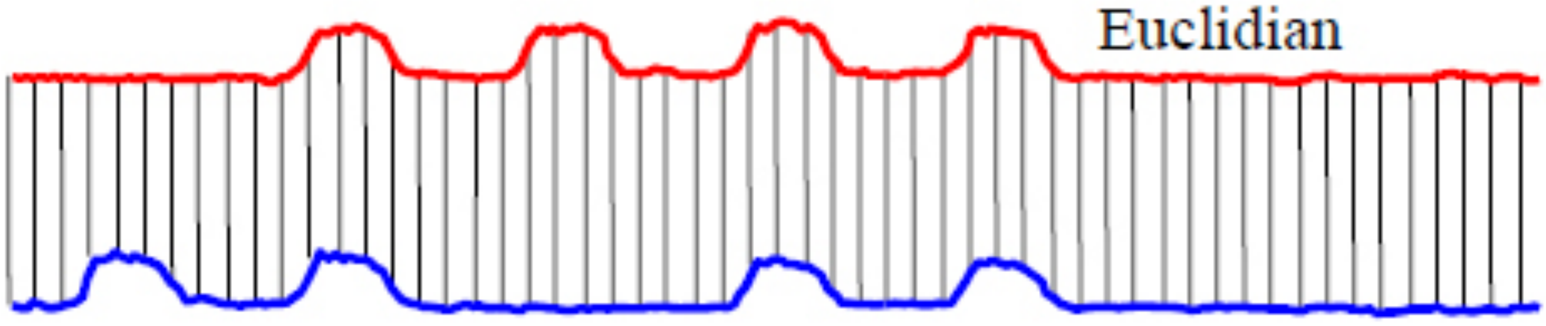
Forcing correlation

NON-SYNCED SOURCES

Our measures no longer work!

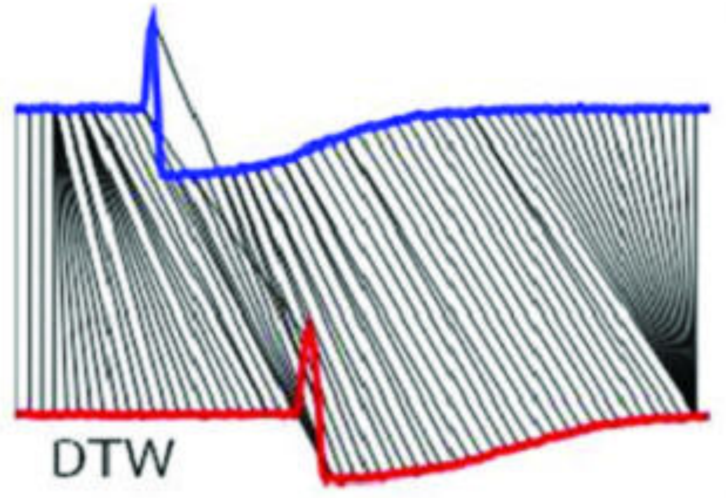


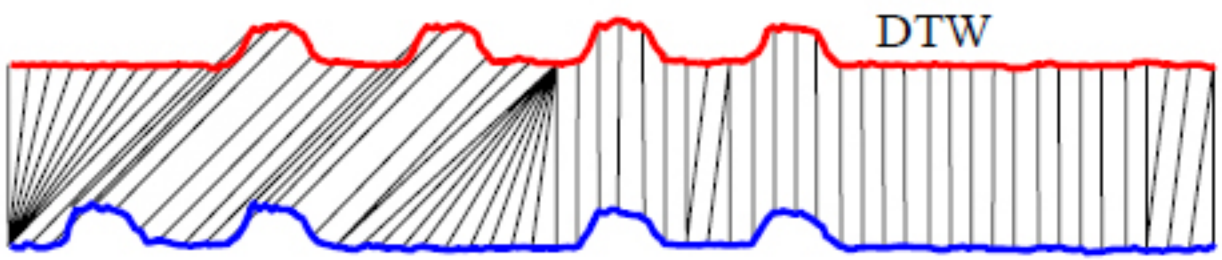
It can get even worse…



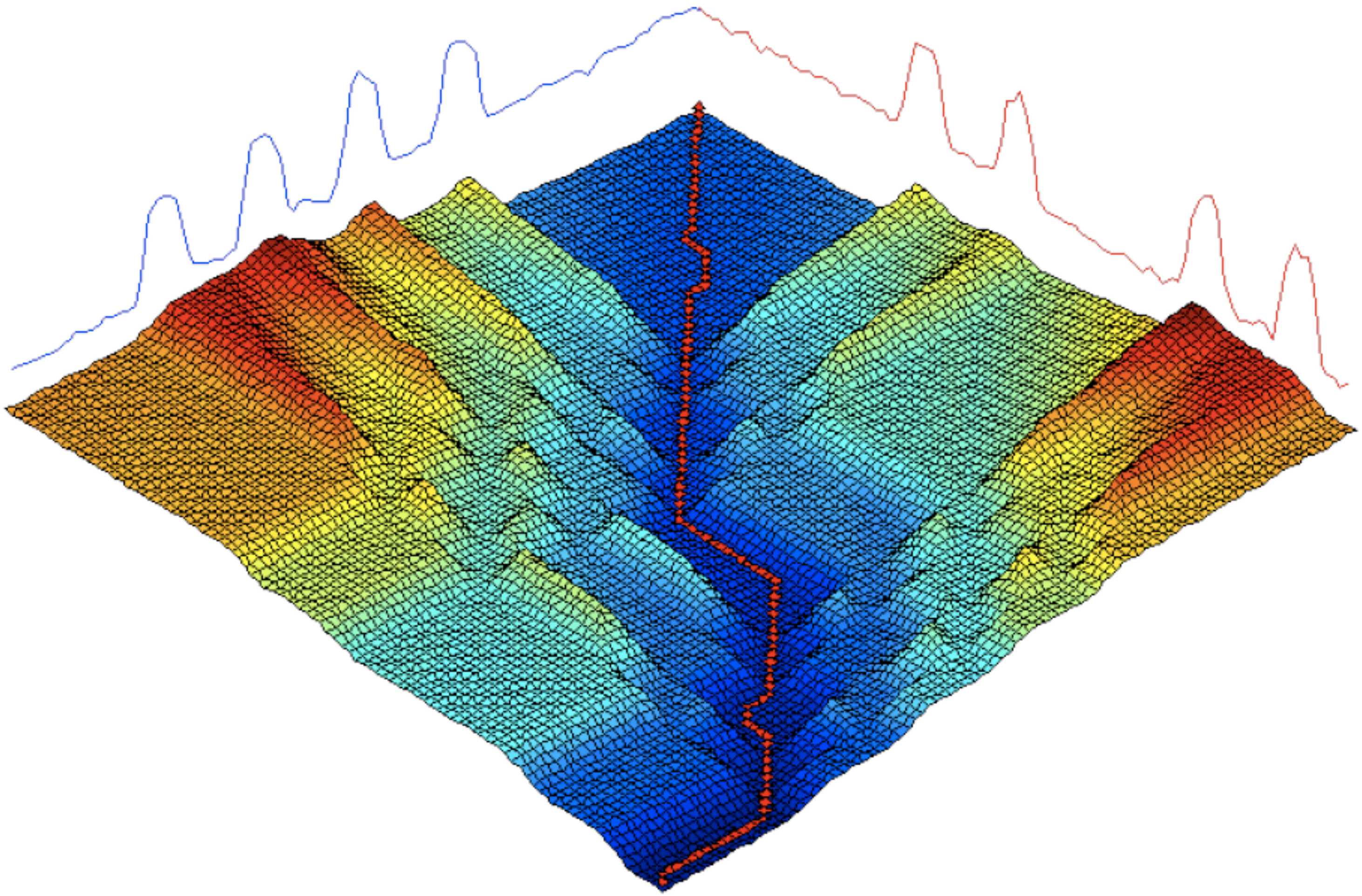
The Dynamic Time Warping approach

It's an alignment algorithm





How does it work?



Cost functions

Audio: choose a good descriptor first (Chroma, MFCC)!

2 options: mapping and multidimensional distance

Why not raw audio?

Euclidean (or Cartesian) distance	$D_{[2]}(\mathbf{x},\mathbf{y}) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$
Chebyshev distance	$D_{[\infty]}(\mathbf{x},\mathbf{y}) = \max_{i=1}^{n} \lvert x_i - y_i \rvert$
Manhattan (city-block) distance	$D_{[1]}(\mathbf{x},\mathbf{y}) = \sum_{i=1}^{n} \lvert x_i - y_i \rvert$
Minkowsky distance	$D_{[p]}(\mathbf{x},\mathbf{y}) = \left[\sum_{i=1}^{n} \lvert x_i - y_i \rvert^p\right]^{\frac{1}{p}}$
Weighted Minkowsky distance	$D_{[p,\mathbf{w}]}(\mathbf{x},\mathbf{y}) = \left[\sum_{i=1}^{n} w_i \lvert x_i - y_i \rvert^p\right]^{\frac{1}{p}}$
Mahalanobis distance	$D(\mathbf{x},\mathbf{y}) = \lvert \det \mathbf{C} \rvert^{1/n} (\mathbf{x}-\mathbf{y})^{\mathrm{T}} \mathbf{C}^{-1} (\mathbf{x}-\mathbf{y})$
Generalised Euclidean (quadratic) distance	$D(\mathbf{x},\mathbf{y}) = (\mathbf{x}-\mathbf{y})^{\mathrm{T}} \mathbf{K} (\mathbf{x}-\mathbf{y})$
Correlation coefficient	$\rho(\mathbf{x},\mathbf{y}) = \dfrac{\sum_{i=1}^{n}(x_i - \bar{x}_i)(y_i - \bar{y}_i)}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x}_i)^2 \sum_{i=1}^{n}(y_i - \bar{y}_i)^2}}$
Relative entropy (Kullback-Leibler divergence)	$D(\mathbf{x} \Vert \mathbf{y}) = \sum_{i=1}^{n} x_i \log \frac{x_i}{y_i}$ when $\sum_{i=1}^{n} x_i = \sum_{i=1}^{n} y_i = 1$
χ^2-Distance	$D_{\chi^2}(\mathbf{x},\mathbf{y}) = \sum_{i=1}^{n} \frac{(x_i - y_i)^2}{y_i}$ when $\sum_{i=1}^{n} x_i = \sum_{i=1}^{n} y_i = 1$

Finding the cheapest/best path

DYNAMIC PROGRAMMING

Example: Fibonacci sequence

1 1 2 3 5 8 13 21

$$F_n = \begin{cases} 0 & \text{if } n = 0; \\ 1 & \text{if } n = 1; \\ F_{n-1} + F_{n-2} & \text{if } n > 1. \end{cases}$$

Example: Fibonacci sequence

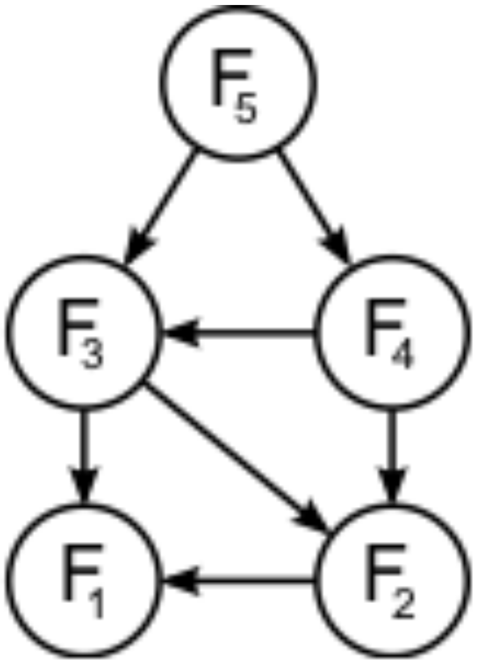
```
Algorithm Fibonacci(n)
if n <= 1, then:
  return 1
else:
  return Fibonacci(n - 1) + Fibonacci(n - 2)
end of if
```

A call tree:

```
        5
      /   \
    4       3
   / \     / \
  3   2    2 1
 / \ / \  / \
 2 1 1 0  1 0
/ \
1 0
```

This takes $O(1.618^n)$ time!

```
 to compute      takes
Fibonacci(40)  75.22 seconds
Fibonacci(70)   4.43 years
```



Example: Fibonacci sequence

Dynamic Programming: Bottom-to-up approach

```
int fib(int n)
    {
    int f[n+1];
    f[1] = f[2] = 1;
    for (int i = 3; i <= n; i++)
        f[i] = f[i-1] + f[i-2];
    return f[n];
    }
```

Example: Fibonacci sequence

Dynamic Programming: Bottom-to-up approach

```
int fib(int n)
    {
    int f[n+1];
    f[1] = f[2] = 1;
    for (int i = 3; i <= n; i++)
        f[i] = f[i-1] + f[i-2];
    return f[n];
    }
```

Dynamic Time Warping

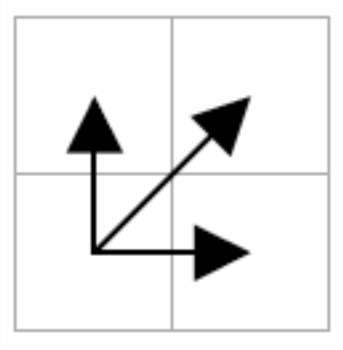
```
int DTWDistance(char s[1..n], char t[1..m]) {
    declare int DTW[0..n, 0..m]
    declare int i, j, cost

    for i := 1 to m
        DTW[0, i] := infinity
    for i := 1 to n
        DTW[i, 0] := infinity
    DTW[0, 0] := 0

    for i := 1 to n
        for j := 1 to m
            cost:= d(s[i], t[j])
            DTW[i, j] := cost + minimum(DTW[i-1, j  ],    // insertion
                                        DTW[i  , j-1],    // deletion
                                        DTW[i-1, j-1])    // match

    return DTW[n, m]
}
```

Dynamic Time Warping

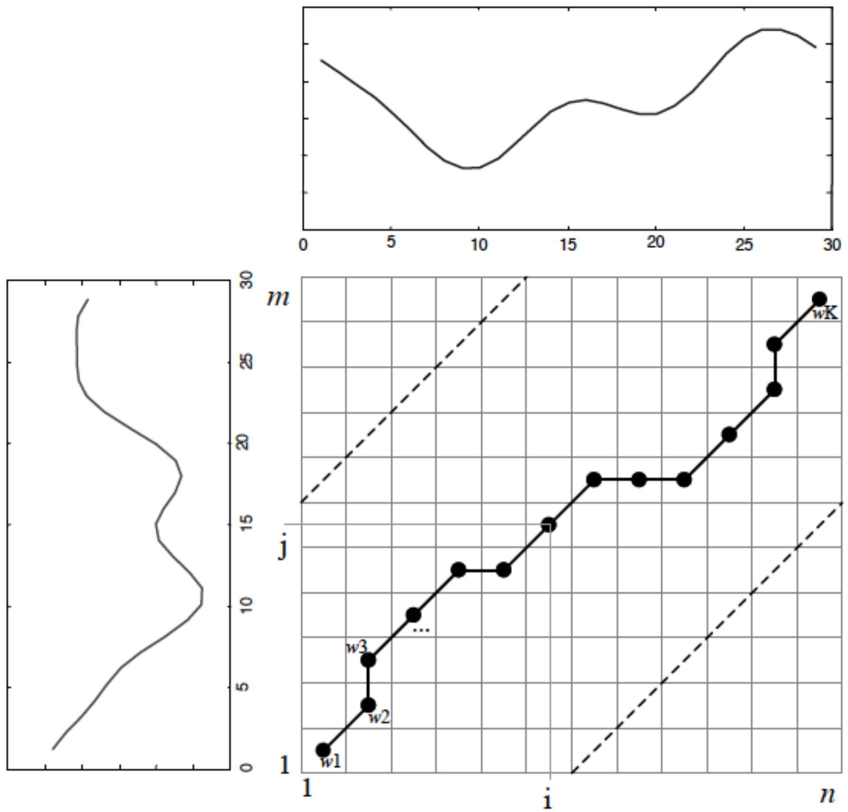
```
int DTWDistance(char s[1..n], char t[1..m]) {
    declare int DTW[0..n, 0..m]
    declare int i, j, cost

    for i := 1 to m
        DTW[0, i] := infinity
    for i := 1 to n
        DTW[i, 0] := infinity
    DTW[0, 0] := 0

    for i := 1 to n
        for j := 1 to m
            cost:= d(s[i], t[j])
            DTW[i, j] := cost + minimum(DTW[i-1, j  ],    // insertion
                                        DTW[i  , j-1],    // deletion
                                        DTW[i-1, j-1])    // match

    return DTW[n, m]
}
```

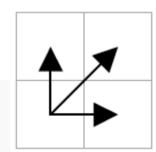
Dynamic Time Warping

```
int DTWDistance(char s[1..n], char t[1..m]) {
    declare int DTW[0..n, 0..m]
    declare int i, j, cost

    for i := 1 to m
        DTW[0, i] := infinity
    for i := 1 to n
        DTW[i, 0] := infinity
    DTW[0, 0] := 0

    for i := 1 to n
        for j := 1 to m
            cost:= d(s[i], t[j])
            DTW[i, j] := cost + minimum(DTW[i-1, j  ],     // insertion
                                        DTW[i  , j-1],     // deletion
                                        DTW[i-1, j-1])     // match

    return DTW[n, m]
}
```



Dynamic Time Warping

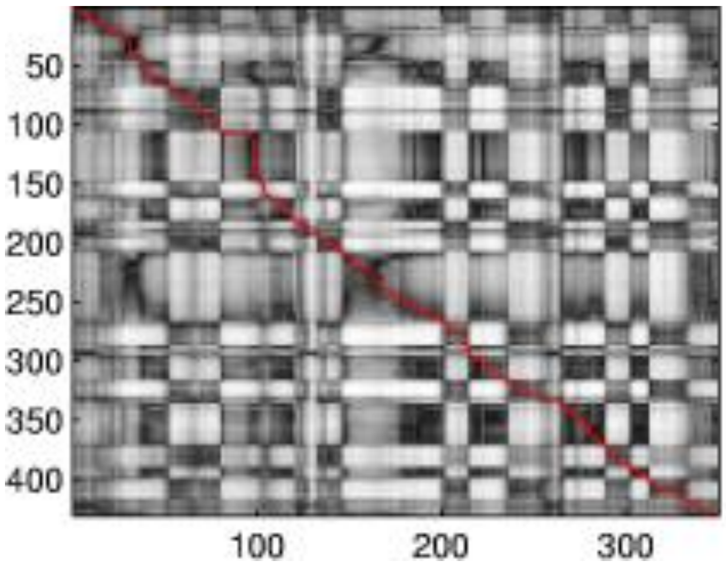




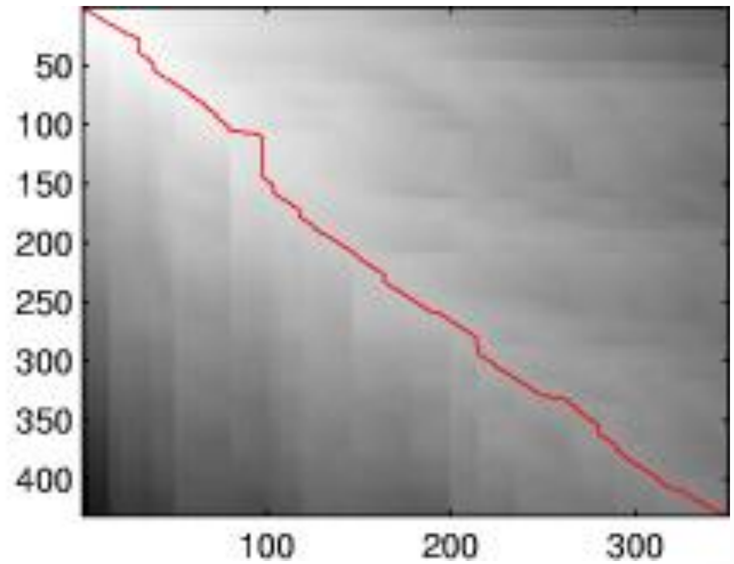
```
for i := 1 to n
    for j := 1 to m
        cost:= d(s[i], t[j])
        DTW[i, j] := cost + minimum(DTW[i-1, j  ],    // insertion
                                    DTW[i  , j-1],    // deletion
                                    DTW[i-1, j-1])    // match
```

Dynamic Time Warping

Distance Matrix



Cost-to-x Matrix

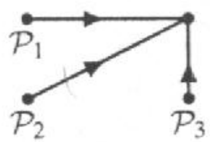
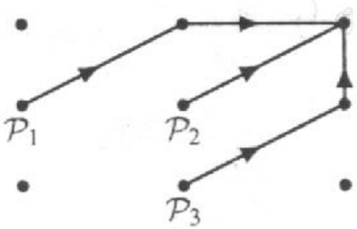
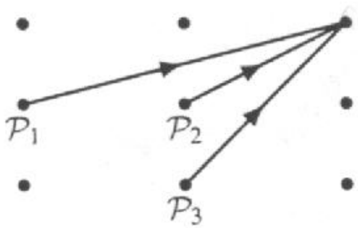
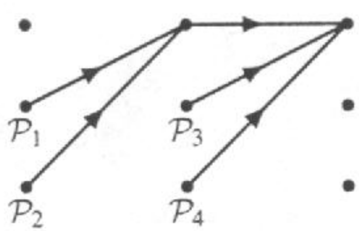



```
for i := 1 to n
    for j := 1 to m
        cost:= d(s[i], t[j])
        DTW[i, j] := cost + minimum(DTW[i-1, j  ],    // insertion
                                    DTW[i  , j-1],    // deletion
                                    DTW[i-1, j-1])    // match
```

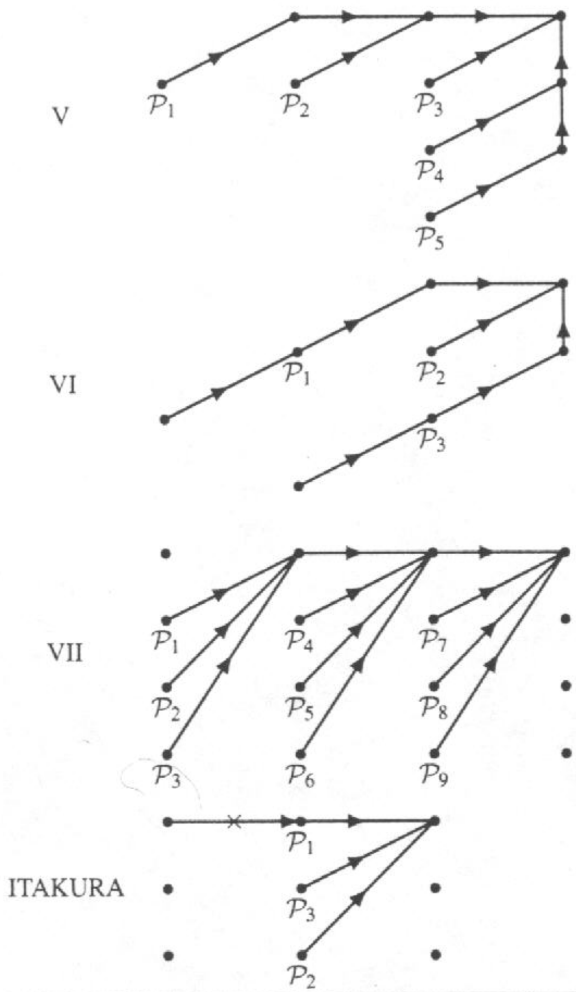
Matlab example: Gunx

Different paths are allowed

TABLE 4.5. Summary of sets of local constraints and the resulting path specifications

<table>
<tr><th>Type</th><th></th><th>Allowable Path Specification</th></tr>
<tr><td>I</td><td></td><td>$\mathcal{P}_1 \to (1,0)$
$\mathcal{P}_2 \to (1,1)$
$\mathcal{P}_3 \to (0,1)$</td></tr>
<tr><td>II</td><td></td><td>$\mathcal{P}_1 \to (1,1)(1,0)$
$\mathcal{P}_2 \to (1,1)$
$\mathcal{P}_3 \to (1,1)(0,1)$</td></tr>
<tr><td>III</td><td></td><td>$\mathcal{P}_1 \to (2,1)$
$\mathcal{P}_2 \to (1,1)$
$\mathcal{P}_3 \to (1,2)$</td></tr>
<tr><td>IV</td><td></td><td>$\mathcal{P}_1 \to (1,1)(1,0)$
$\mathcal{P}_2 \to (1,2)(1,0)$
$\mathcal{P}_3 \to (1,1)$
$\mathcal{P}_4 \to (1,2)$</td></tr>
</table>

Even more! [Step matrix]



V

$\mathcal{P}_1 \rightarrow (1,1)(1,0)(1,0)$
$\mathcal{P}_2 \rightarrow (1,1)(1,0)$
$\mathcal{P}_3 \rightarrow (1,1)$
$\mathcal{P}_4 \rightarrow (1,1)(0,1)$
$\mathcal{P}_5 \rightarrow (1,1)(0,1)(0,1)$

VI

$\mathcal{P}_1 \rightarrow (1,1)(1,1)(1,0)$
$\mathcal{P}_2 \rightarrow (1,1)$
$\mathcal{P}_3 \rightarrow (1,1)(1,1)(0,1)$

VII

$\mathcal{P}_1 \rightarrow (1,1)(1,0)(1,0)$
$\mathcal{P}_2 \rightarrow (1,2)(1,0)(1,0)$
$\mathcal{P}_3 \rightarrow (1,3)(1,0)(1,0)$
$\mathcal{P}_4 \rightarrow (1,1)(1,0)$
$\mathcal{P}_5 \rightarrow (1,2)(1,0)$
$\mathcal{P}_6 \rightarrow (1,3)(1,0)$
$\mathcal{P}_7 \rightarrow (1,1)$
$\mathcal{P}_8 \rightarrow (1,2)$
$\mathcal{P}_9 \rightarrow (1,3)$

ITAKURA

$\mathcal{P}_1 \rightarrow (1,0)$, consecutive $(1,0)(1,0)$ disallowed
$\mathcal{P}_2 \rightarrow (1,1)$
$\mathcal{P}_3 \rightarrow (1,2)$

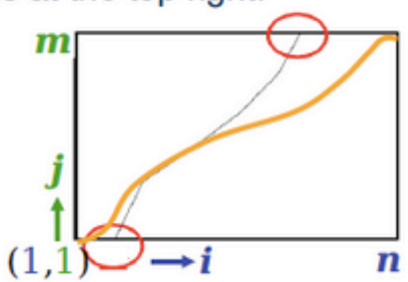
Global constraints

HEURISTICS: REDUCING COMPLEXITY & INCREASING EFFICIENCY

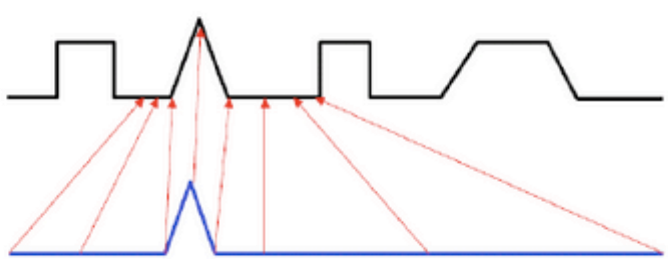
Classical DTW constraints

Boundary Conditions: $i_1 = 1, i_k = n$ and $j_1 = 1, j_k = m$.

The alignment path starts at the bottom left and ends at the top right.

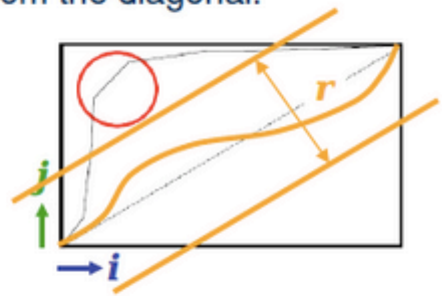


Guarantees that the alignment does not consider partially one of the sequences.

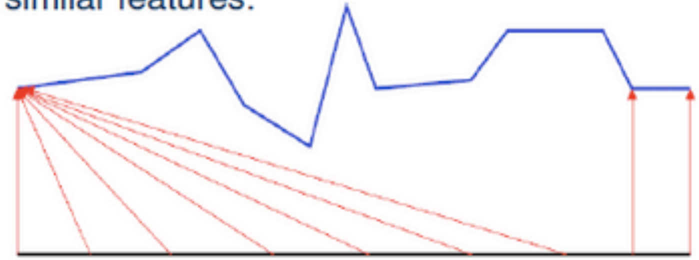


Warping Window: $|i_s - j_s| \leq r$, where $r > 0$ is the window length.

A good alignment path is unlikely to wander too far from the diagonal.



Guarantees that the alignment does not try to skip different features and gets stuck at similar features.



Warping Windows

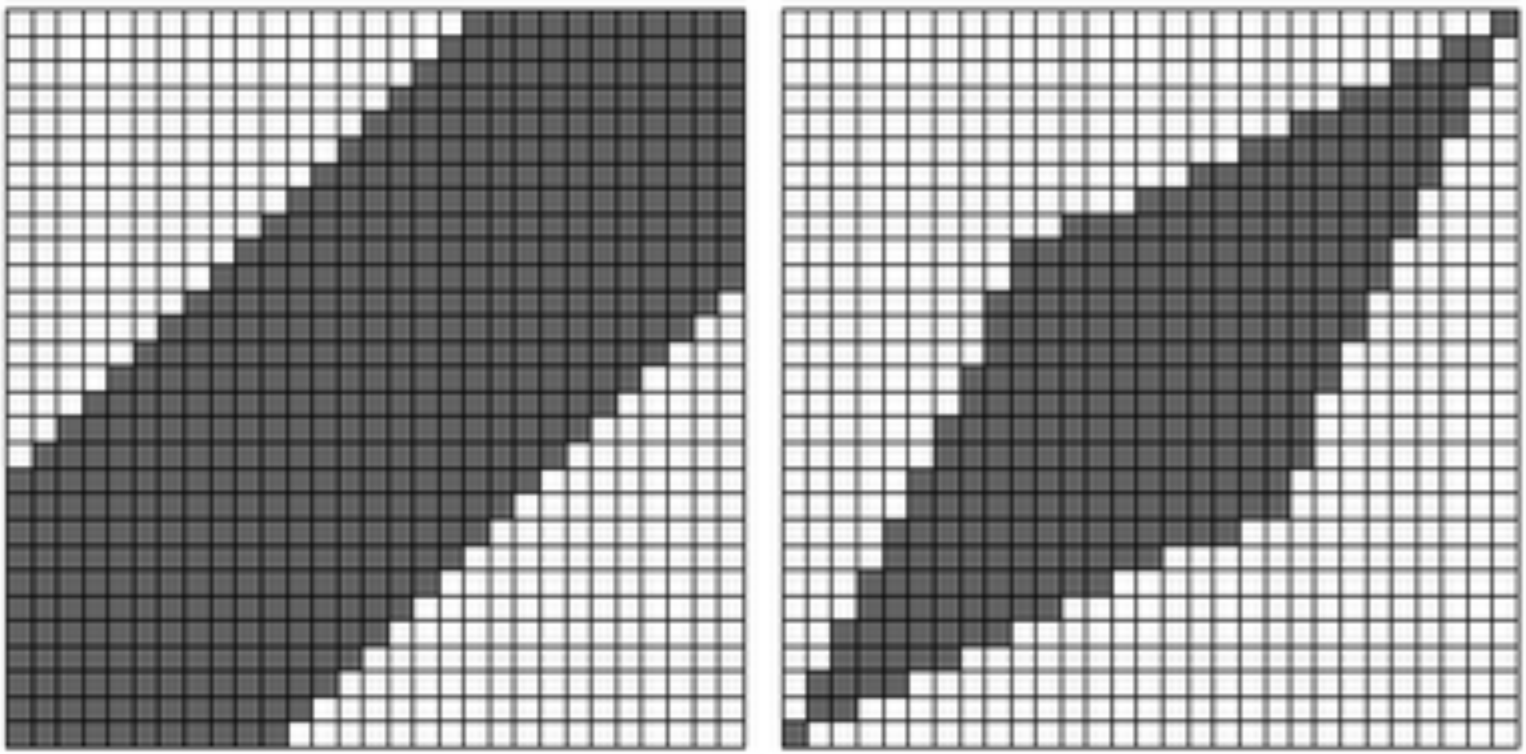
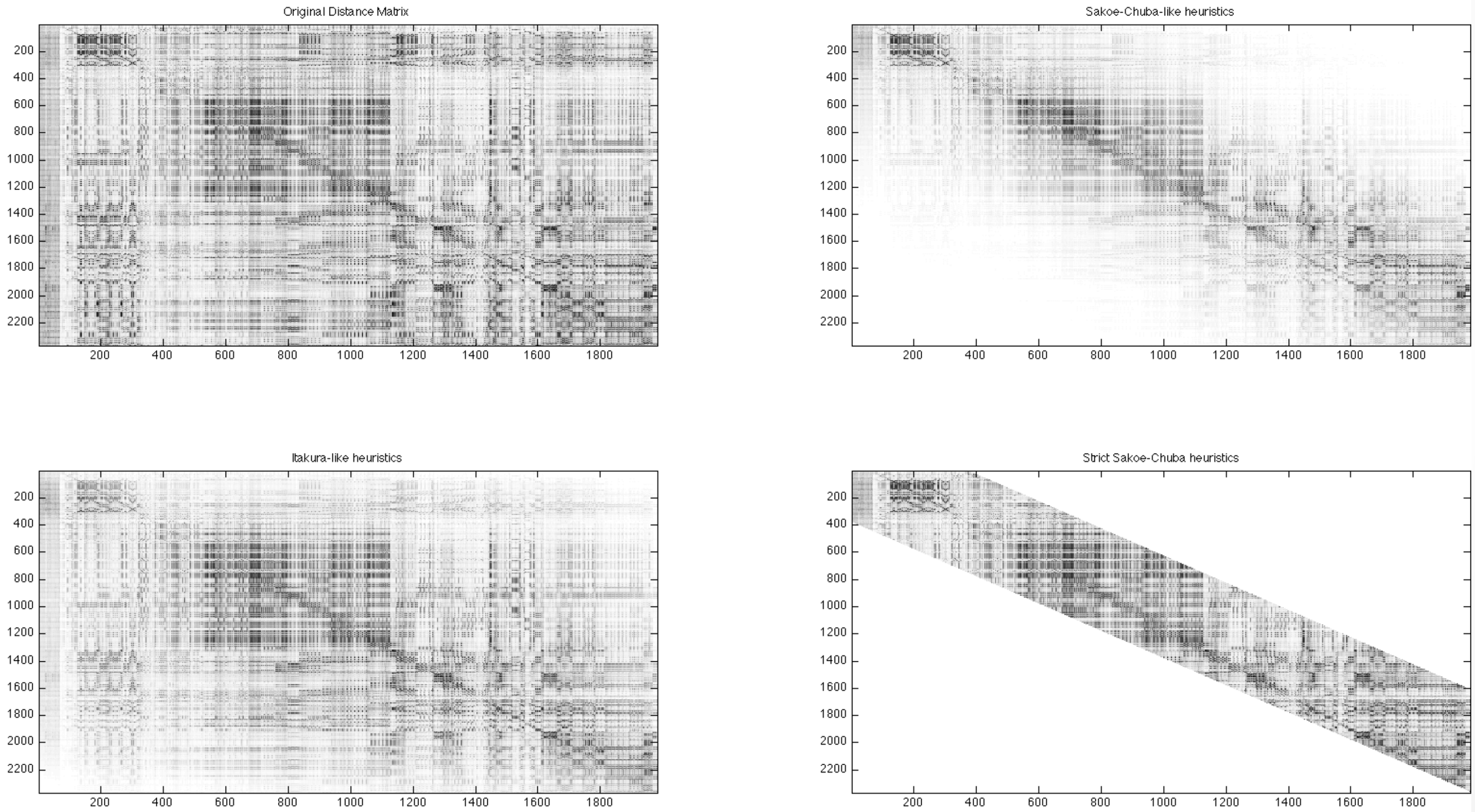


Figure 4. Two constraints: Sakoe-Chuba Band (left) and an Itakura Parallelogram (right), both have a width of 5.

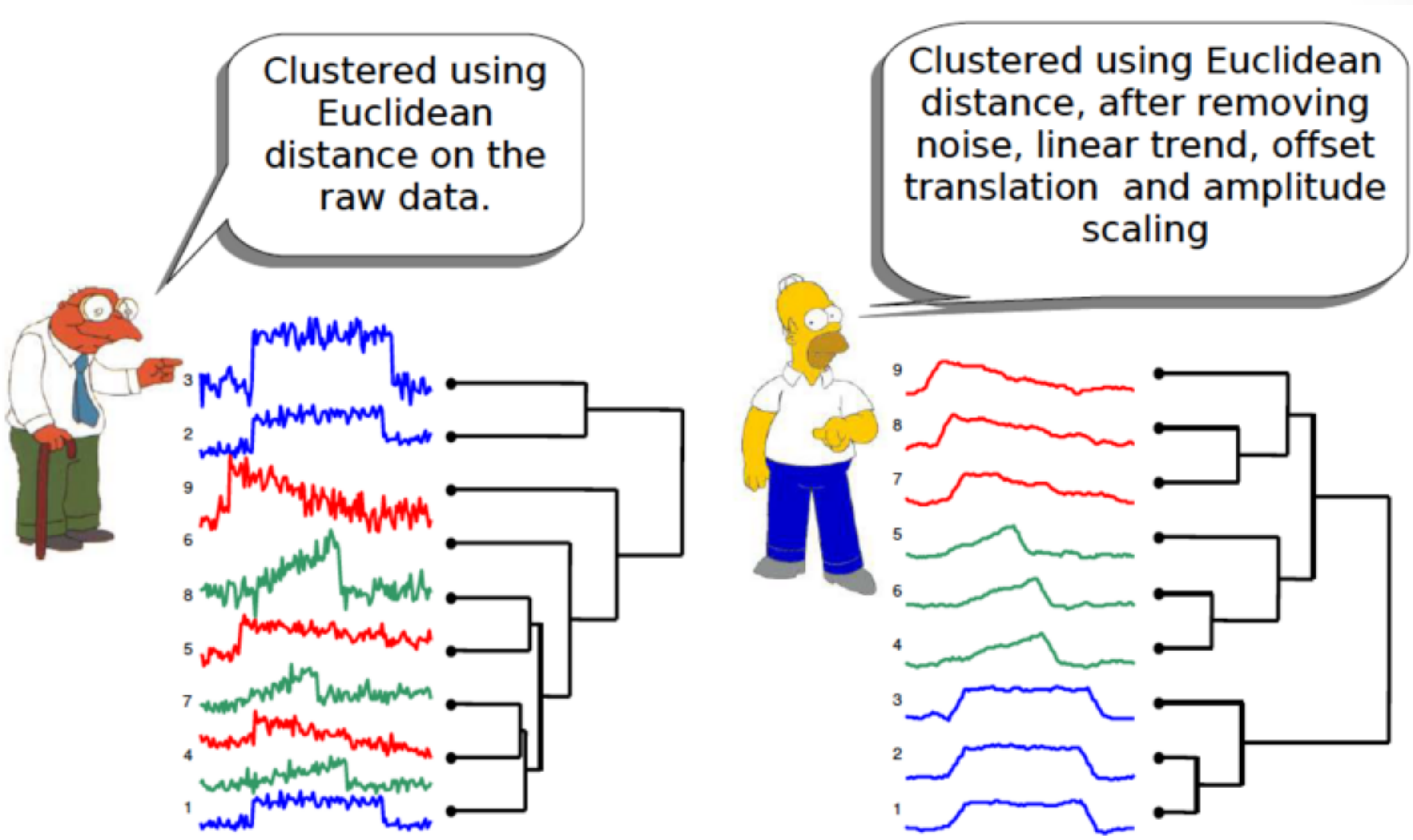
Heuristic functions



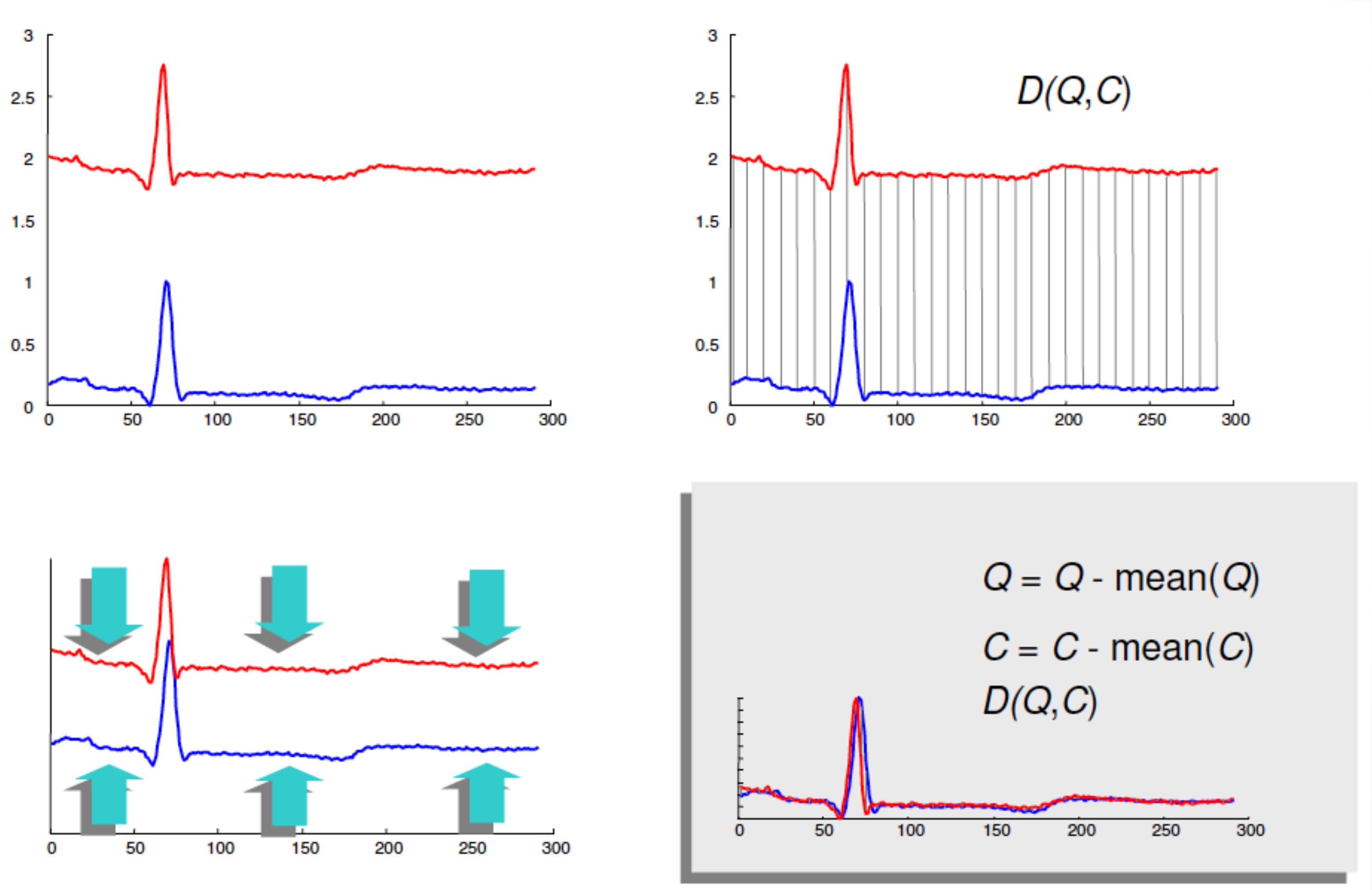
DC Offset, normalization, linear trend

PREPROCESSING: DC OFFSET, NORMALIZATION, LINEAR TREND

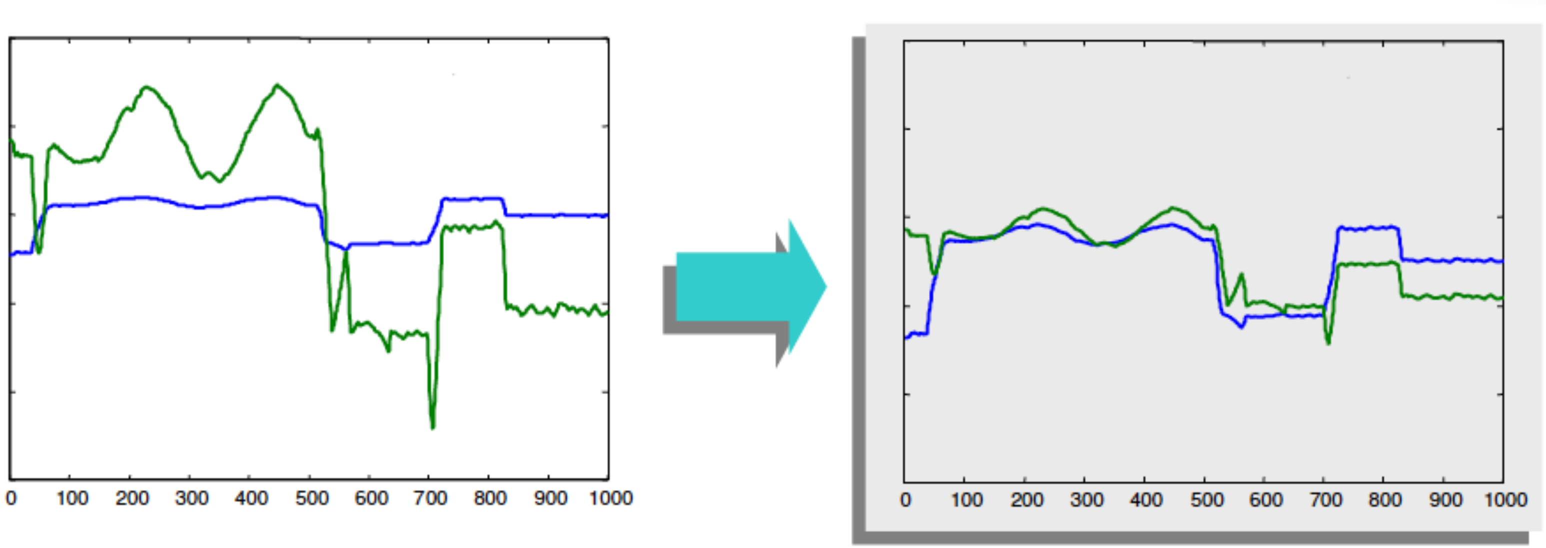
Preprocessing



Preprocessing: DC offset



Preprocessing: Amplitude

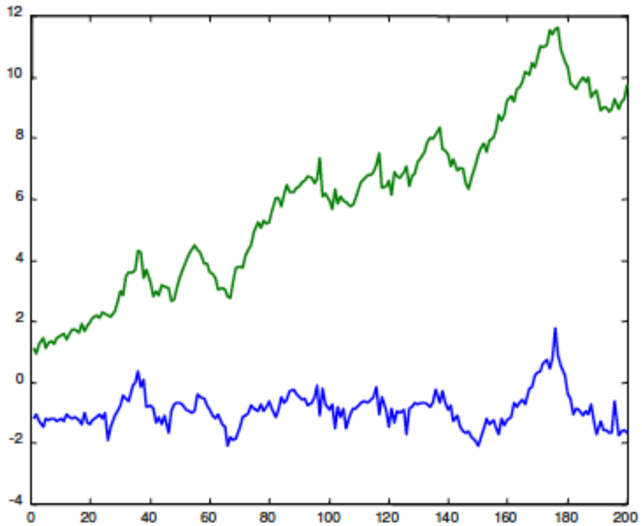


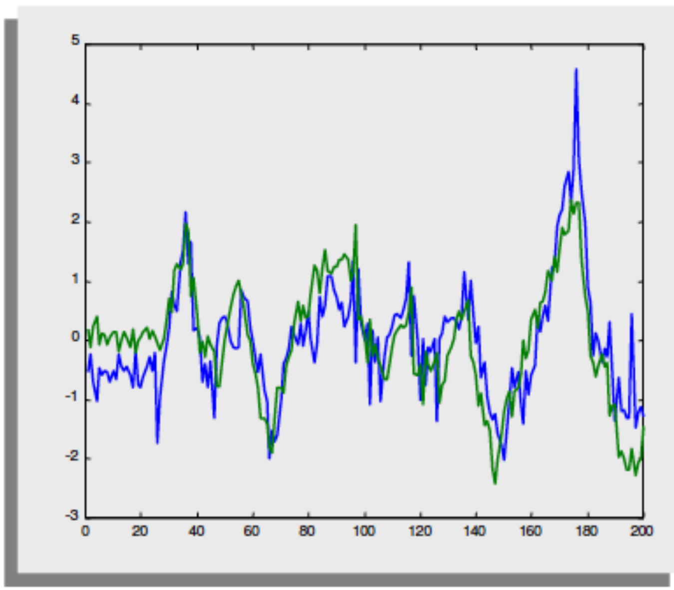
$Q = (Q - \text{mean}(Q)) / \text{std}(Q)$

$C = (C - \text{mean}(C)) / \text{std}(C)$

$D(Q,C)$

Preprocessing: Linear trend





The intuition behind removing linear trend is…

Fit the best fitting straight line to the time series, then subtract that line from the time series.

Removed **linear trend**

Removed offset translation

Removed amplitude scaling

Thank you!

That’s all for today ☺

Questions, comments…