Let's do the Time Warp again!
Revisiting Dynamic Time Warping – A practical tutorial in Python on North Sea field data

Jesper Sören Dramsch, Anders Nymark Christensen, Mikael Lüthje

ABSTRACT

This tutorial revisits Dynamic Time Warping for geoscientific applications. This algorithm can be used to match arbitrary time-series, which is applicable to 4D time shifts, seismic-well ties, well-to-well ties, and seismic pre- and post-stack migration. DTW is notorious to be computationally slow and expensive, while underperforming on seismic field data. We show that a choice of similarity measures, optimization, and constraints can both speed up calculation and significantly improve results. We show a full implementation in Python code on 4D seismic traces recorded over 10 years apart. Moreover, we explore recent developments in DTW and the significance to machine learning metrics.

INTRODUCTION

Seismic data analysis often relies on the comparison of two or more seismic traces. In quantitative 4D seismic analysis particularly, significant effort is invested in aligning traces that were acquired in the same location at different times. These comparisons tend to be more complicated than simple differencing of the traces, due to slight location, and imaging variations as well as physical changes in the subsurface.

Major problems in the alignment of traces are mismatches in trace and source location, new acquisition equipment, and changes in the subsurface. Location misalignment can stem from streamer feathering or obstructions due to newly built structures, which the acquisition vessel has to avoid. Differences in acquisition technology and azimuth play a role in misalignment. While technology often is matched as closely as possible, it tends to be a tradeoff between improved imaging and matched acquisition. Changes in the subsurface stem from multiple effects that include geomechanics, pressure changes, temperature fluctuations, and fluid movement. These combined sources of misalignment cause changes in the ray path, mismatched wavelets, cycle-skipping (time-shifts by one cycle) and amplitude variations.

Conventional methods include the following methods with its most prominent properties:

- Windowed correlation – quick and reliable, given an appropriate window
- Optical Flow – Prone to amplitude changes and cycle-skipping
- Inversion-based methods – Expensive, sometimes model-dependent but reliable

Dynamic Time Warping (DTW) is a time series analysis tool that can be used for these comparisons of traces from the same location, acquired at different times. Within the methods to measure time-shifts, DTW forms its own distinct class, different from the aforementioned methods. DTW was introduced for seismic analysis in Hale (2013) in the context of 4D seismic image warping. In Luo and Hale (2014) they extended it to improve Least Squares Migration. Moreover, while we focus on 4D seismic applications, DTW can be used for seismic-well ties, post- and pre-stack seismic data analysis, well-log analysis, and time series classification.

In this tutorial paper we investigate DTW in Python, where we will use two traces from the Danish North Sea, courtesy of Total E&P Denmark. These are from the same location in a field, recorded over ten years apart. In terms of seismic acquisition this is the difference of oil-filled to solid state streamers, air gun design and differential steering of streamers.

We will focus on exploring different loss functions to align the trace data. Moreover, we investigate the use of

constraints in DTW to improve the results of the time warp results.

DATA PREPARATION

DTW does not need special preparation of the data. However, it can benefit from upsampling the traces to obtain a denser match of the warping result. This increases the computational cost of the algorithm. While the computational cost of standard DTW is $\mathcal{O}(n^2)$ it can be significantly reduced (Ratanamahatana and Keogh, 2004). Rakthanmanon et al. (2012) show a diverse suite of optimizations to search a trillion data points in under 120 seconds.

In Listing 1 we load the seismic traces into memory using segyio (Kvalsvik and Contributors, 2019), seen in Figure 1. We go on to slice the first 100 samples, upsample to 200 samples and cut away the first 25 samples due to edge effects (ringing). Just by visual inspection of the close up in Figure 2 we can see that comparing these traces will be complicated as the waveform is noticeably different and amplitudes vary strongly. These strong differences due to acquisition make it an excellent case for exploring the robustness of DTW.

```
import segyio
from segyio import tools
from scipy.signal import resample

f_old = 'new_trace.sgy'
f_new = 'old_trace.sgy'

with segyio.open(f_old, strict=False) as f:
    new_trace_ = tools.collect(f.trace[:]).T
with segyio.open(f_new, strict=False) as f:
    old_trace_ = tools.collect(f.trace[:]).T

new_trace = resample(new_trace_[:101],200)[25:]
old_trace = resample(old_trace_[:101],200)[25:]
sz = len(old_trace)
```

Listing 1: Load, slice and resample Seismic Traces. Output: Figure 1 and Figure 2

DYNAMIC TIME WARPING

In its essence, dynamic time warping is taking a similarity measure of every sample between two time series and then finding the optimal warp path to align these traces. By taking a step to the left or right through this matrix of similarity values the algorithm aligns these traces dynamically.

For the results to be sensible, several constraints are added:

- Every sample in both series must be matched.
- Each sample can have several matches
- The mappings must be monotonically increasing. I.e. we cannot have crossed matches between the two series

Dynamic time warping does not use a windowed approach. However, some constraints that can improve the results may be interpreted as a form of windowing. These windows constrain globally or locally, how far the algorithm searches for best matches in the warp path, further discussed in the section Constraints. In a geophysical sense this is limiting the allowable time shifts between traces, where a global constraint applies a uniform maximum and local constraint can vary the amount of time shifts in different sections of the subsurface.

FastDTW

Several attempts have been made to optimize the time to calculate DTW. One scheme to improve the warp speed is to start the search on a severely downsampled signal and progressively upsample the signal with iterative DTW searches. This approach has been named FastDTW. In our early experiments this approach does not perform well on seismic data, therefore, we will only mention the algorithm for completeness. In the section Constraints we show several global constraints that significantly improve the time shift results and lower the computational cost.

Soft DTW

Technically, DTW is not a metric as it does not satisfy the triangular inequality

$$D(k,l) \leq D(k,m) + D(l,m) \tag{1}$$

for $D()$ being the distance between two points, and k, l, and l being arbitrary points. Moreover, as it's a dynamic programming problem it is not differentiable, which is a desirable property in machine learning. Cuturi and Blondel (2017) redefine DTW as a differentiable loss-function. The DTW similarity measure - i.e. the length of the warped path between two time series - has been shown to outperform the euclidean distance for time-series classification, which makes it a valuable extension to the machine learning toolbox. With the following definition of the minimum

$$min^{\gamma}\{a_1, ..., a_n\} := \begin{cases} min_{i \leq n} a_i, & \text{for } \gamma = 0 \\ -\gamma log \sum_{i=1}^{n} \exp -\frac{a_i}{\gamma}, & \text{for } \gamma > 0 \end{cases}$$

where a_i are points along the path in a cost matrix, we can define the γ-soft-DTW

$$DTW_{\gamma}(x,y) := min^{\gamma}\{\langle A, \Delta(x,y)\rangle, A \in A_{n,m}\}$$

with $\Delta(x,y)$ being the distance matrix and A an alignment matrix containing the path.

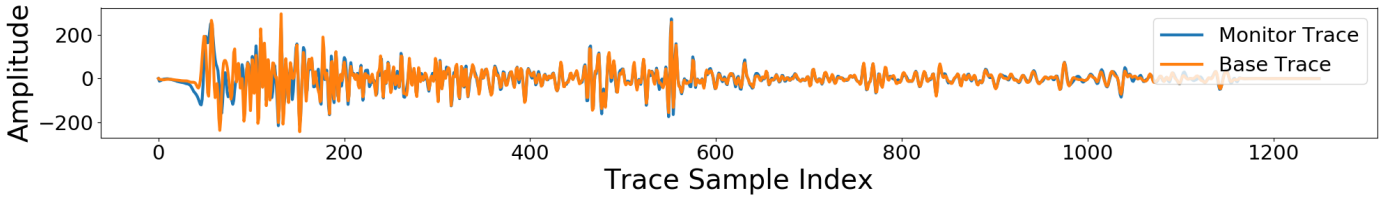


Figure 1: Comparison of Monitor and Base Trace

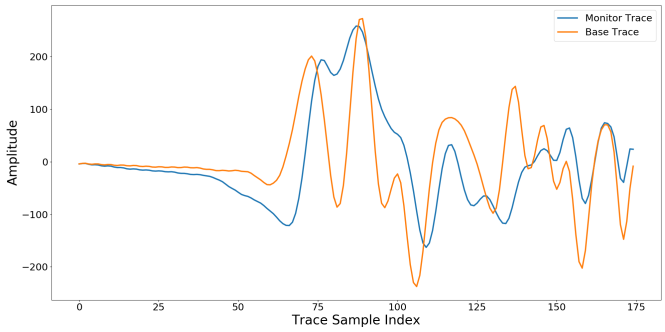


Figure 2: Close-Up of Traces in Figure 1

Dynamic Image Warping (2D/3D DTW)

Dynamic Image Warping (DIW) is the extension of DTW to 2D and 3D datasets. Hale (2013) introduced DIW for seismic data by applying the DTW algorithm in z-direction along the time-series and smoothing adjacent time-shifts to obtain a consistent results. This process can be done iteratively with progressively smaller smoothing windows to obtain x-y consistent DIW results. It is important to note that DIW does not increase the computational cost of the DTW algorithm itself. Contrary to the intuition, the distance matrixes and cumulative cost we present in the section Finding the warp, are calculated in the same way resulting in a 2D cost matrix for each pair of 1D time series. The conclusions and optimizations we present in this paper are therefore directly applicable to DIW and especially the Least-Squares Migration with DTW (Luo and Hale, 2014). Considering the speed-up to linear computational cost, DIW becomes feasible in the pre-stack domain.

DISTANCE METRICS AND SIMILARITY

A metric – or distance function – is a measure of similarity between two sets. We will consider L_1, L_2 and Huber Loss.

The warp path will be influenced by the choice of distance metric. Distances are notorious for having many different names. The easiest to calculate is the L_1 distance, also known as Manhattan distance or the (mean) absolute error.

The L_2 norm is calculated as $(a-b)^2$. It is also known as least squares or Euclidean distance.

One can see that L_1 and L_2 have some different benefitial properties and trade-offs. L_1 is linear, but non-differentiable at 0. L_2 is convex and differentiable, but the error explodes for outliers due to the square operation. Hence, the introduction of the Huber loss, which is L_2 for small values of a and L_1 for large values of a.

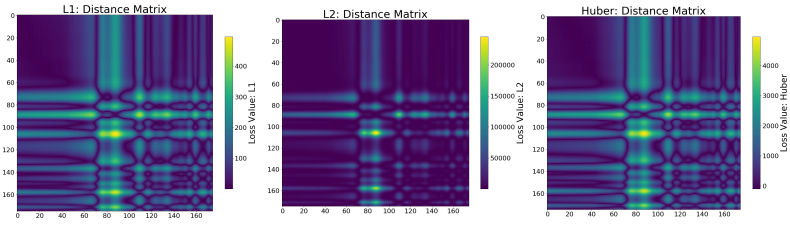
$$L_\delta(a,b) = \begin{cases} \frac{1}{2}(a-b)^2 & \text{for } |a-b| \le \delta, \\ \delta(|a-b| - \frac{1}{2}\delta), & \text{otherwise.} \end{cases} \quad (2)$$

For convenience the smooth Pseudo-Huber loss is defined as $L_\delta(a,b) = \delta^2(\sqrt{1+((a-b)/\delta)^2} - 1)$. The parameter δ is introduced to match the slope at the change from L_2 to the L_1 loss. The choice of δ can change the results for the Huber loss significantly. Experimenting on subsets of the data to obtain a good value for δ is feasible. However, looking at the variance of errors has given good results and setting δ to the standard deviation σ.

FINDING THE WARP

First, we start by building a distance or similarity matrix, with all possible combinations between the two traces, as shown in listing 2. We provide a convenient implementation of L_1, L_2, and the Pseudo-Huber loss.

In index [0,0] (using Pythonic 0-indexing) we have the distance between sample 1 in trace 1 and sample 1 in trace 2. In [2,6] we have the distance between sample 3 in trace 1 and sample 7 in trace 2, etc. Depending on the distance metric we will get different results shown in Figure 3.



(a) L_1 Distance (b) L_2 Distance (c) Huber Distance

Figure 3: Distance Matrices.

Then, we calculate the cumulative cost matrix from the distance matrix. This is achieved by first calculating the cumulative cost around the edge of the cost matrix. Then we loop over each sample of the cost matrix and fill in the values by adding the minimal value of the adjacent samples to the current sample saved in the underlying distance

matrix. This creates a cumulative cost that ideally is lowest at sample [0, 0] and increases to the last sample at [sz, sz]. The value in this last index, can beused to cheack in an optimal warp has been found. In that case the DTW similairy will match the value in [sz, sz].

We can now maximise the similarity between the traces. We search the minimum path by backtracking from [sz, sz] the bottom right corner, where we have the last points from both traces, to [0, 0] the top left corner. The minimum path will be the maximum similarity, i.e. the combination, where the traces differ the least.

An easy way to find the path is by taking the cumulative values of the similarity matrix and the use steepest descent from the lower right corner, where we simply select the direction by taking the neighbouring point with the least value. The code can be found in Appendix A and a comparison of the similarity matrix, and the obtained paths for the three different metrics can be found in Figure 5.

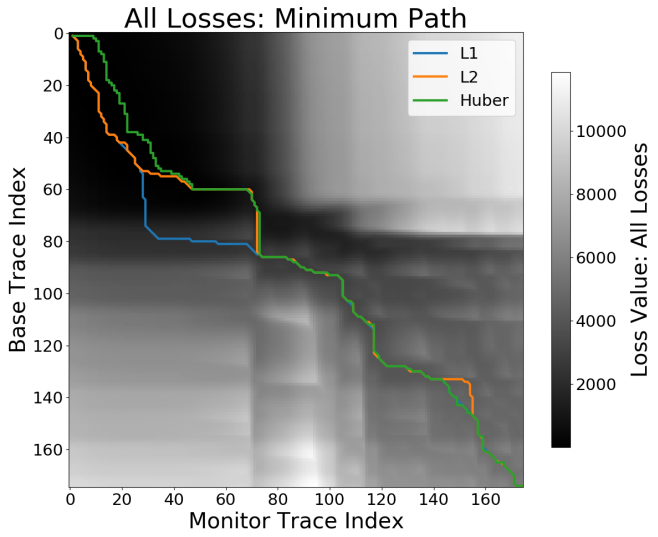


Figure 5: All Paths on Cumulative Cost.

There are several mathematical superior ways to steepest descent for finding the minimum path in a matrix. However, steepest descent is fast and easy to implement, and understand for this demonstration, as well as, many other use-cases. We show the result of the L_2 warp in Figure 7b. Mathematically, the Dijkstra algorithm is optimal for traversing a directed graph, which would be a suitable reformulation of this problem, but would exceed the scope of this tutorial (Cormen et al., 2009).

The Code Listing 2 shows the full calculation of multiple distance matrixes and calculating the cost matrix from it. On line 1 to 8 we present the three distance metrics we evaluate in this paper. On line 10 we assign the metric to a common name, therefore we don't have to rewrite the entire script. On line 12 we allocate the cost matrix with zeros in memory.

```
1  l1_distance = abs(new_trace.T-old_trace) # L1
2  l2_distance = (new_trace.T-old_trace)**2 # L2
3
4  delta = 10
5  hu_distance = ( np.sqrt( ( 1 + ( \
6                      new_trace.T  - old_trace) \
7                      / delta)**2) - 1) \
8                  * delta**2 #Huber
9
10 distance = hu_distance # Choose a Distance
11
12 cost_mat = np.zeros((sz,sz))
13
14 cost_matrix[0, :] = np.cumsum(distance[0, :])
15 cost_matrix[:, 0] = np.cumsum(distance[:, 0])
16
17 for old in range(1, sz):
18     for new in range(1, sz):
19         cost_matrix[old,new] = distance[old, new] \
20             + min(cost_matrix[old-1, new-1],
21                   cost_matrix[old-1, new],
22                   cost_matrix[old, new-1])
23
24 p,q = backtrack(cost_matrix)
```

Listing 2: Calculate unconstrained cumulative cost matrix. Output: Figure 4

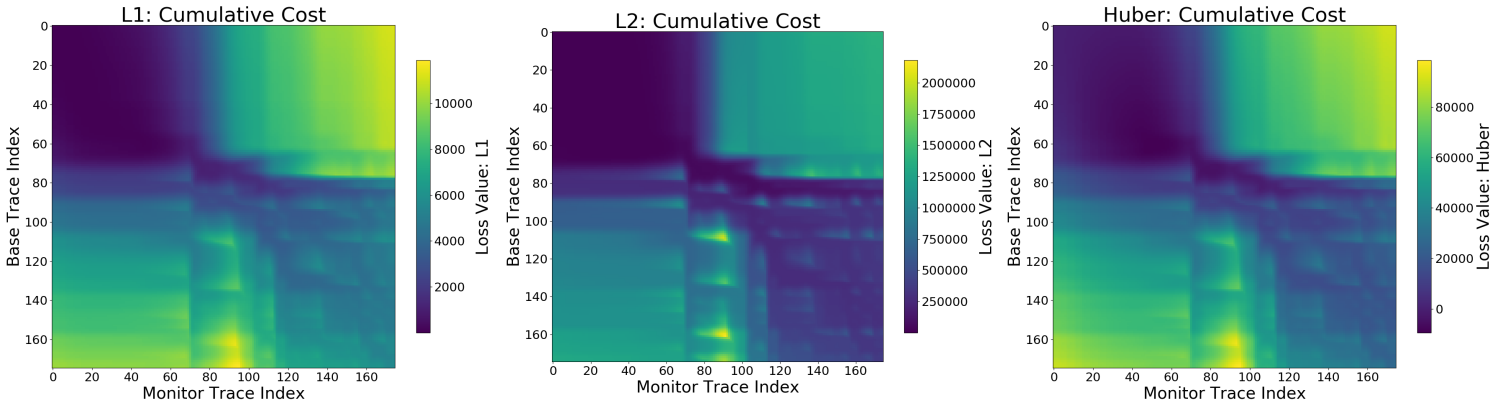
On line 14 and 15 we fill the left and upper edges of the cost matrix with the cumulative sum. Each value is the sum of the previous cumulative value and the current distance matrix. Using numpy instead of loops offloads the computation to optimized C-code in the background.

On line 17 to 22 we populate the cost matrix iteratively with the cumulative sum of the minimal value between the diagonal upper left, left and upper value in the matrix. Unfortunately, due to the nature of this dynamic programming problem, we have to implement this with two for-loops. However, significant speed-ups can be obtained by using just-in-time (JIT) compilation with Numba (Lam et al., 2015).

CONSTRAINTS

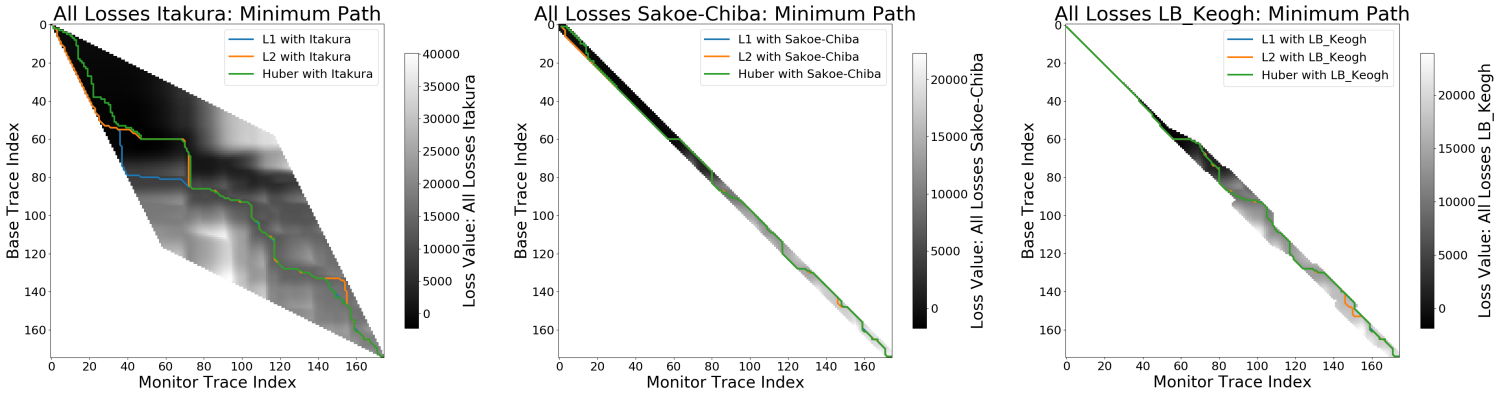
Dynamic time warping in its original form allows warping of any sample to any location within the constraints mentioned earlier. Realistically, boundary conditions improve the result, by constraining the maximum warp distance. In the physical realm of 4D this is equivalent to setting a maximum time shift. This would require some knowledge of the reservoir, but we can make some conservative estimates for 4D seismic anyways.

Several implementations were suggested to limit warping to central values of the distance matrix. Namely, the most wide-spread global constraints are the Itakura parallelogram (Itakura, 1975) and the Sakoe-Chiba disk



(a) L_1 Cost (b) L_2 Cost (c) Huber Cost

Figure 4: Cumulative Cost Matrices.



(a) Itakura (1975) Parallelogram (b) Sakoe and Chiba (1978) Disc (c) LB_Envelope (Keogh and Ratanamahatana, 2005)

Figure 6: Minimum path for constraint masks for cumulative cost in DTW.

(Sakoe and Chiba, 1978). Furthermore, we will explore the LB_Keogh lower bound for timeseries (Keogh and Ratanamahatana, 2005).

Itakura Global Constraint

The Itakura parallelogram does not take any input parameters in its original form. The general shape constrains the warp path stronger towards the ends of the time series and gives more liberties in the central region of the time series. The change of the minimum path for the three distance metrics are shown in Figure 6a.

The Code Listing 3 shows the calculation of the Itakura parallelogram constraint. On line 1 we allocate the full mask matrix with ∞ as the default value. On line 3 to 4 we define the calculate $2 * l - k$, with l and k being the indices of the matrix. On line 5 to 9 we exploit the symmetry of the parallelogram and set every sample within the parallelogram to 0.

```
1 itakura = np.full((sz,sz), np.inf)
2
3 parallel = (np.subtract.outer( \
4                 range(0,sz*2,2), range(sz)))
5 itakura[(parallel   > 0)  * \
6         (parallel.T > 0)  * \
7         (parallel   < sz) * \
8         (parallel.T < sz)   \
9         ] = 0
```

Listing 3: Itakura Parallelogram Global Constraint. (Itakura, 1975) Output: Figure 6a

Sakoe-Chiba Global Constraint

The Sakoe-Chiba Disc is a uniform constraint that limits the maximum time-shifts uniformly. This leaves the path to be constrained within the center following band. E.g if we want a constraint on 5 samples, which is equivalent to

a maximum time-shift of 10 ms at a sampling interval of 2 ms, we will use a Sakoe Chiba radius of 3.

```
1 sa_radius = 4
2 sakoe = np.full((sz,sz), np.inf)
3 sakoe[np.abs( \
4          np.subtract.outer(range(sz),range(sz)) \
5       ) < sa_radius] = 0
```

Listing 4: Sakoe-Chiba Disc Global Constraint. (Sakoe and Chiba, 1978) Output: Figure 6b

The Code Listing 4 shows the calculation of the Sakoe-Chiba disc constraint. On line 1 we define the disc radius. On line 2 we allocate the full array for the mask with ∞ as the non-included cost. On line 3 to 5 we subtract the row vector of indices from the column vector of indices and apply the absolute to that difference. Then we set every sample in the mask matrix to 0, if it's within the disc radius.

Lower Bound Keogh Envelope Global Constraint

The Lower Bound Keogh constraint is based on the Ratanamahatana-Keogh constraint (Niennattrakul and Ratanamahatana, 2009), which introduced arbitrary global constraints. While these give a possibility for an application of machine learning to refine the algorithmic accuracy, we will first explore the applicability of lower bounds for time series. Rath and Manmatha (2002) prove that the LB_Keogh lower bound holds for multivariate time series.

For this to hold, $LB\_Keogh(Q,C) \leq DTW(Q,C)$ was shown to be true. The LB_Keogh lower bound is defined as follows:

$$LB\_Keogh(Q,C) = \sum_{i=1}^{n} \begin{cases} (q_i - U_i)^2, & \text{for } q_i > U_i \\ (q_i - L_i)^2, & \text{for } q_i < L_i \\ 0, & otherwise \end{cases} \tag{3}$$

with Q the Query time series with q_i being the elements of Q. C the candidate sequence, with U_i and L_i being the upper and lower envelope of C respectively. In the 4D seismic domain the query string corresponds to tho monitor trace and the candidate string corresponds to the base trace.

We calculate U and L as follows:

$$U_i = max(q_{i-r}, ..., q_{i+r}) \tag{4}$$

$$L_i = min(q_{i-r}, ..., q_{i+r}) \tag{5}$$

with r being analogous to the radius of the Sakoe-Chiba Disc, defining the perimeter for the time-series envelope.

This definition of LB_Keogh focuses on the L_2 norm. However, it can be shown that LB_Keogh holds for arbitrary distance metrics (Minkowski distances of order (p-), incuding L_1 and Huber). This allows for an efficient constraint of DTW on arbitrary metrics and arbitrary time series. Lower bounded DTW can reduce the computation from $\mathcal{O}(n^2)$, including applications to segment search Smith and Craven (2008). In geoscience this would allow for target-oriented migration results to be aligned within a larger cube as well as sequential scanning.

Listing 5 contains the envelope (U, L) of C as a hard boundary as opposed to the penalty proposed in equation 3. We combine the envelopes of the query and candidate time-series to artificially increase the search window. This is not necessary but brings possible benefits in cases of cycle-skipping.

```
1  keogh_radius=10
2  keogh = np.full((sz,sz), np.inf)
3
4  L_old, U_old = metrics.lb_envelope(old_trace, \
5                          radius=keogh_radius)
6  L_new, U_new = metrics.lb_envelope(new_trace, \
7                          radius=keogh_radius)
8
9  L = np.min((L_old, L_new), axis=0)
10 U = np.max((U_old, U_new), axis=0)
11
12 L /= np.max((np.abs(U),np.abs(L))) \
13             / (keogh_radius)
14 U /= np.max((np.abs(U),np.abs(L))) \
15             / (keogh_radius)
16
17 for Q in range(sz):
18     for C in range(sz):
19         if (L[k] < C - Q < U[k]):
20             keogh[Q,C] = 0
```

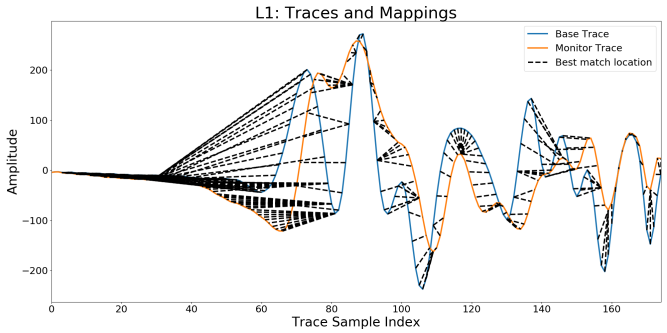
Listing 5: LB_Envelope Local Constraint. (Keogh and Ratanamahatana, 2005) Output: Figure 6c

The Code In Listing 5 we calculate the envelope-based constraint on DTW. On line 1 we define the radius for the envelope. On line 2 we allocate the mask matrix with ∞. On line 4 to 7 we calculate the envelope of both time series using the `tslearn` library, according to equation 4 and equation 5. On line 9 to 15 we combine them and scale them to the index matrix. On line 17 to 20 we iterate through both time series. If the indices are between the lower and upper bound, we set the sample in the mask matrix to 0.

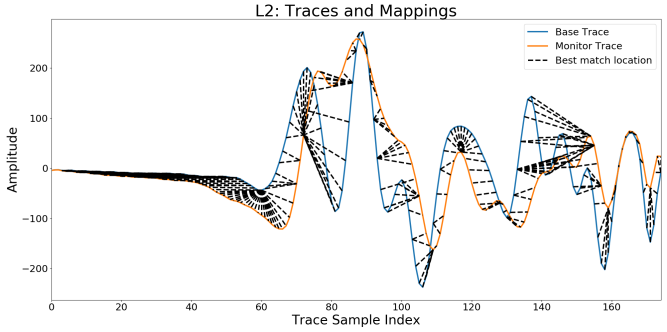
DO THE TIME WARP

The DTW algorithm may find one-to-many maps in its warp path considering that everywhere, where the path is vertical or horizontal in Figure 5 one value of the base trace is mapped to several points in the monitor trace

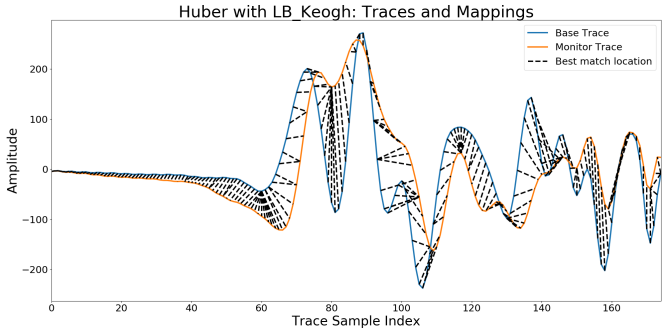
and vice versa. While technically this may be the optimal mapping between traces, this leads to non-physical interpolation results, when aligning the traces. We will handle this by taking the average value of the indices then filling the missing values. This can be achieved by using a moving average after removing duplicate values, which is shown in listing 6 and illustrated in Figure 8. By correcting this at the indexing stage, we can achieve smooth alignment of traces without deteriorating the result.



(a) Unconstrained Mapping of Traces, $L1$



(b) Unconstrained Mapping of Traces, $L2$



(c) Constrained Mapping of Traces, Huber

Figure 7: Mapping of Traces using DTW, showing benefits of constraints and distance measures.

The Code In Listing 6 we calculate the minimum path, adjust the path, and warp the monitor trace to match the base trace. On line 1 we import the 1D univariate spline interpolation. On line 3 we calculate the minimum path for the cumulative cost matrix from Listing 4 and constrain it with the keogh envelope from Listing 5. On line 5 to 7 we replace the one-to-many mappings visible in Figure 7c with the mean value. On line 9 to 12 we smooth indices to condition the indexing for the interpolation. On line 14 to 16 we interpolate the monitor trace to the smoothed indices, which results in Figure 8.

```
1  from scipy.interpolate import \
2  UnivariateSpline as interp1d
3
4  re_old, re_new = backtrack(cost_matrix + keogh)
5
6  out = [(np.mean(np.array(re_new) \
7          [np.array(re_old) == q])) \
8          for q in range(len(new_trace))]
9
10 N = 7 #Smoothing Factor
11 out_smooth = np.convolve(out,
12                           np.ones((N,))/N,
13                           mode='same')
14
15 trace_interp = interp1d(np.arange(new_trace.size),
16                          new_trace.T,
17                          k=4)
```

Listing 6: Warping of Traces constrained by LB_Keogh. Output: Figure 8

DISCUSSION

In this tutorial we revisited Dynamic Time Warping for geophysical applications. In our investigation of the method we encountered the strong dependence of DTW on the metric and constraint chosen. Moreover, we find that DTW is very difficult to use on field data without additional constraint on the warp path. The data we use in this example is field data that was recorded over 10 years apart, which introduces a multitude of complications regarding the estimation of time-shifts. We show that using an expansion of the envelope can significantly improve the warping result of seismic traces.

Distance Metrics

We find that the L_1 norm is not ideal to find correspondence in seismic traces (cf. Figure 7a). This metric does not adequately distinguish small-scale changes in the data, which is essential to obtain a good time warp result. The L_2 norm gives better results than the L_1 norm but is too prone to large scale variations in the amplitudes (cf. Figure 7b). The data we use in the example shows a change in wavelet, which makes it hard for the L_2 norm to accurately map some shifts. The Huber loss provides a good medium between the L_1 and L_2 norm. However, despite improving results, it is highly dependent on a good choice of δ, which is a drawback in comparison to the hands-off nature of the other metrics. The results of the Huber loss

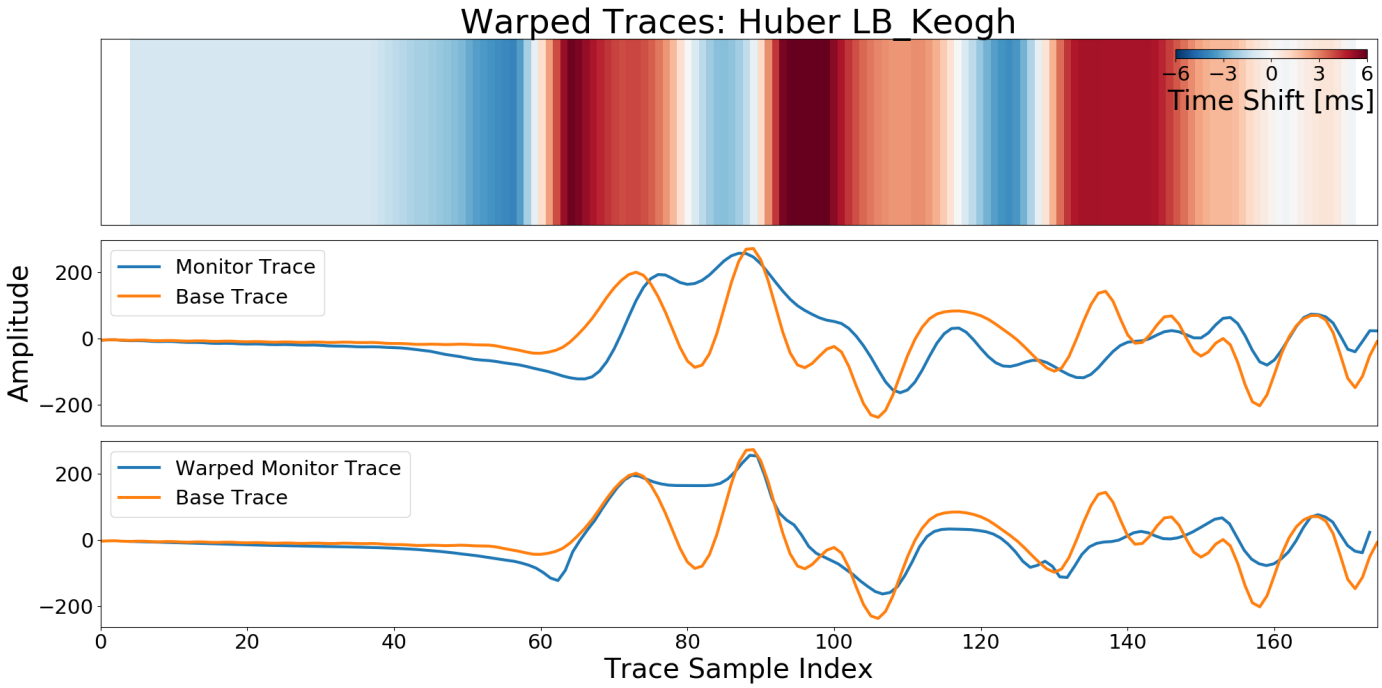


Figure 8: Time shift and warped traces for amplitude comparison using Huber loss and LB_Keogh envelope-based constraint.

itself are, however, not convincing on an unconstrained warp path. The hard nature of the particular problem we explore in this tutorial needs strong constraints to be feasible.

Constraints

Due to the particularly different nature of the problem we pose in this example, we need to better constrain the search space for the optimal warp path. The Itakura parallelogram does not perform well on the nature of geophysical problems, while the beginning of the trace varies less, the large margin for variation towards the central region of the traces is not conducive to these geophysical problems. The Sakoe-Chiba disc, which is equivalent to a global constraint of maximum time-shift gives a better overall result in the warping and time-shift result and is a good general choice for DTW problems in geophysical data. We introduce the expanded time series envelope, also used in the calculation of the Keogh lower bound, as global constraint on the search space, which provides superior results to the Sakoe-Chiba Disc. The expanded envelope allows for greater variation between regions of large amplitudes, with strong constraints on regions of low amplitudes. Due to the dynamic warping, this leads to a disentanglement of time-shifts between subsurface regions, while allowing for enough flexibility due to the expansion of the envelope to also align low amplitude regions.

Computational Cost

Implementing the constraints and subsequently calculating the cost matrix and and minimum warp path can significantly reduce the computational time. The promises of obtaining $\mathcal{O}(n)$ for DTW from other domains, are only valid for calculations of the DTW measure, they are not valid for obtaining the warp path. Nevertheless, the time cost can be reduced below $\mathcal{O}(n^2)$. These savings make DTW and its extension DIW feasible for many problems in geophysics, possibly extending to pre-stack 4D seismic warping and pre-stack migration.

CONCLUSION

In this tutorial we explore Dynamic Time Warping with reproducible code. The code presented here, is available on GitHub, where it can be copied and reproduced. However, the code examples are sufficient to run a full DTW on the data without any addition. DTW can be implemented efficiently, speeding up the calculation significantly despite being a dynamic programming problem. The constraints we introduce using expanded envelope improves the warp result significantly, but the intuitive Sakoe-Chiba Disc gives good results. These can be used to significantly speed up the calculation and reduce the problem below $\mathcal{O}(n^2)$.

Although DTW without constraints has been known to be computationally expensice and underperforming, by introducing constraints and choosing the correct metric, we can obtain convincing results even on very hard problems.

ALTERNATIVE LANGUAGES AND LIBRARIES

Other implementations in Python we recommend, are the library tslearn, pydtw, and a Soft DTW implementation. R users can refer to the excellent DTW package for R. SAS users can use this experimental implementation. The Matlab implementation is provided in the Signal Processing Toolbox.

ACKNOWLEDGMENTS

The research leading to these results has received funding from the Danish Hydrocarbon Research and Technology Centre under the Advanced Water Flooding program. We thank DTU Compute for access to the GPU Cluster.

APPENDIX A
EXAMPLE BACKTRACK FUNCTION

```python
def backtrack(cost_matrix):
    old = cost_matrix.shape[0] - 2
    new = old
    x, y = [old+1], [new+1]
    while (old > 0) | (new > 0):

        direction = np.argmin((
            cost_matrix[old, new],
            cost_matrix[old, new+1],
            cost_matrix[old+1, new]
        ))

        if direction == 0: ## Diagonal
            old -= 1
            new -= 1
        elif direction == 1: ## Up
            old -= 1
        elif direction == 2:  ## Left
            new -= 1
        x.append(old+1)
        y.append(new+1)
    return x[::-1], y[::-1]
```

REFERENCES

Cormen, T. H., C. E. Leiserson, R. L. Rivest, and C. Stein, 2009, Introduction to algorithms: MIT press.

Cuturi, M., and M. Blondel, 2017, Soft-dtw: a differentiable loss function for time-series: Proceedings of the 34th International Conference on Machine Learning-Volume 70, JMLR. org, 894–903.

Hale, D., 2013, Dynamic warping of seismic images: GEOPHYSICS, **78**, S105–S115.

Itakura, F., 1975, Minimum Prediction Residual Principle Applied to Speech Recognition: IEEE Transactions on Acoustics, Speech, and Signal Processing, **23**, 67–72.

Keogh, E., and C. A. Ratanamahatana, 2005, Exact indexing of dynamic time warping: Knowledge and information systems, **7**, 358–386.

Kvalsvik, J., and Contributors, 2019, Segyio.

Lam, S. K., A. Pitrou, and S. Seibert, 2015, Numba: A llvm-based python jit compiler: Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC, ACM, 7.

Luo, S., and D. Hale, 2014, Least-squares migration in the presence of velocity errors: SEG Technical Program Expanded Abstracts 2014, Society of Exploration Geophysicists, 3980–3984.

Niennattrakul, V., and C. A. Ratanamahatana, 2009, Learning dtw global constraint for time series classification: arXiv preprint arXiv:0903.0041.

Rakthanmanon, T., B. Campana, A. Mueen, G. Batista, B. Westover, Q. Zhu, J. Zakaria, and E. Keogh, 2012, Searching and mining trillions of time series subsequences under dynamic time warping: Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 262–270.

Ratanamahatana, C. A., and E. Keogh, 2004, Everything you know about dynamic time warping is wrong: Presented at the Third workshop on mining temporal and sequential data, Citeseer.

Rath, T. M., and R. Manmatha, 2002, Lower-bounding of dynamic time warping distances for multivariate time series: University of Massachusetts Amherst, Tech. Rep. MM-40.

Sakoe, H., and S. Chiba, 1978, Dynamic programming algorithm optimization for spoken word recognition: IEEE Transactions on Acoustics, Speech, and Signal Processing, **26**, 43–49.

Smith, A. A., and M. Craven, 2008, Fast multisegment alignments for temporal expression profiles, *in* Computational Systems Bioinformatics: (Volume 7): World Scientific, 315–326.