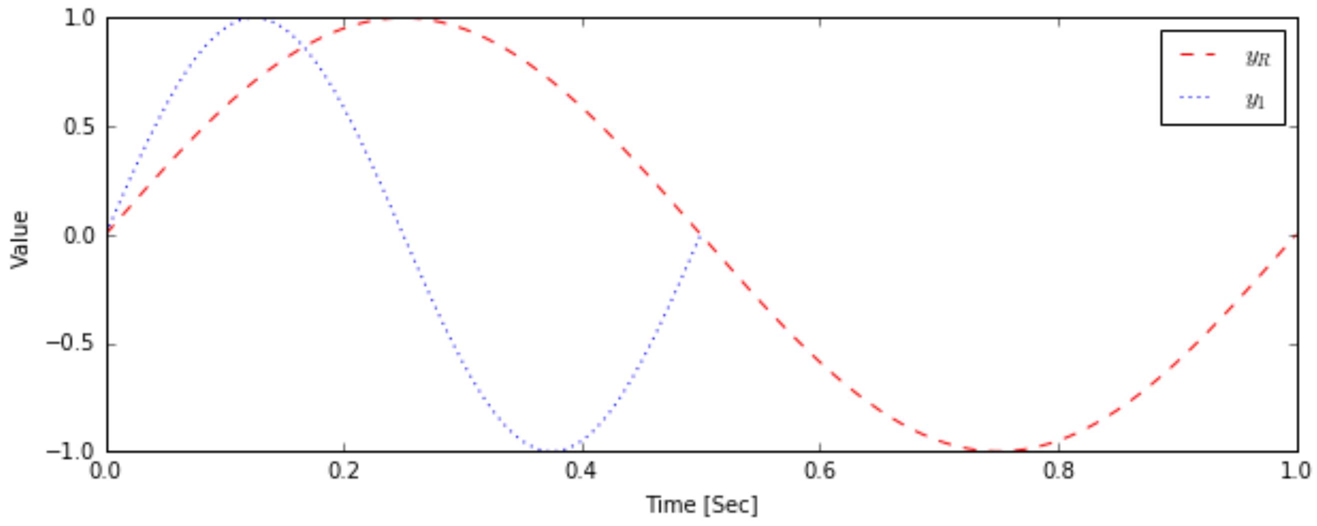
Dynamic Time Warping

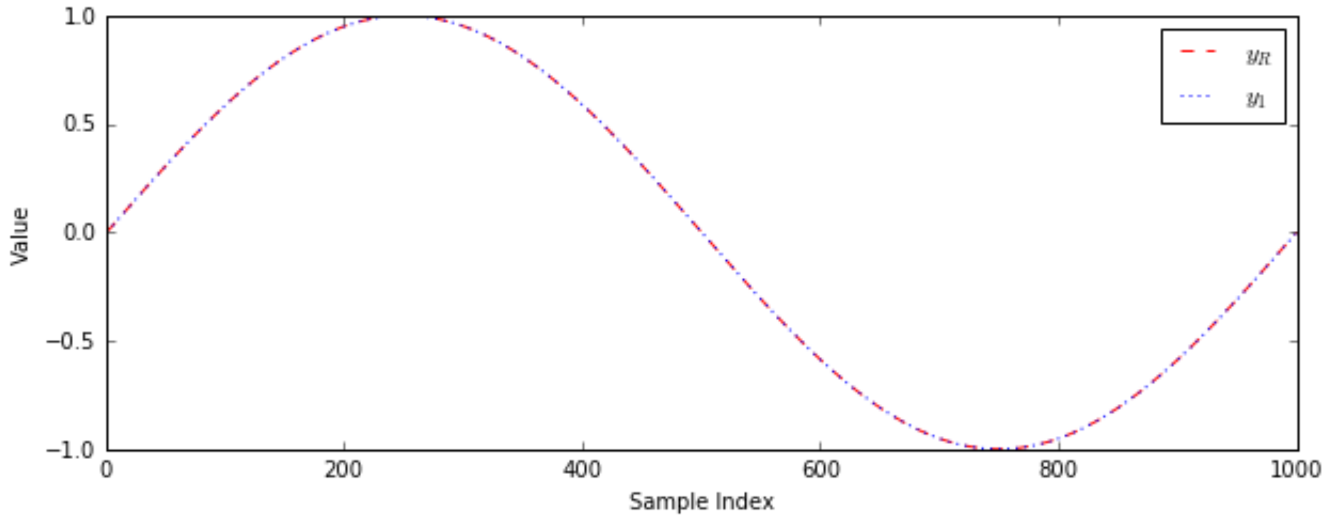
The Problem

Assume having a reference signal $y_R = f(t)$.
Given signals $\left\{ y_i = f(s_i(t)\,t) \right\}$, namely temporally scaled versions of the same signal.
How could on measure the similarity between the signals with no sensitivity to the "Tempo"?

Example

$$y_R = \sin(2\pi t),\ \ t \in [0, 1]$$
$$y_1 = \sin(2\pi 2t),\ \ t \in [0, 0.5]$$





Intuition Think of a curve in 1D.
Assume different people walking the path on different pace while the output is the height of ground the walk on (Or their head assuming they have the same height).

Solution

> In Time Series Analysis, Dynamic Time Warping (DTW) is an algorithm for measuring similarity between two temporal sequences which may vary in speed.

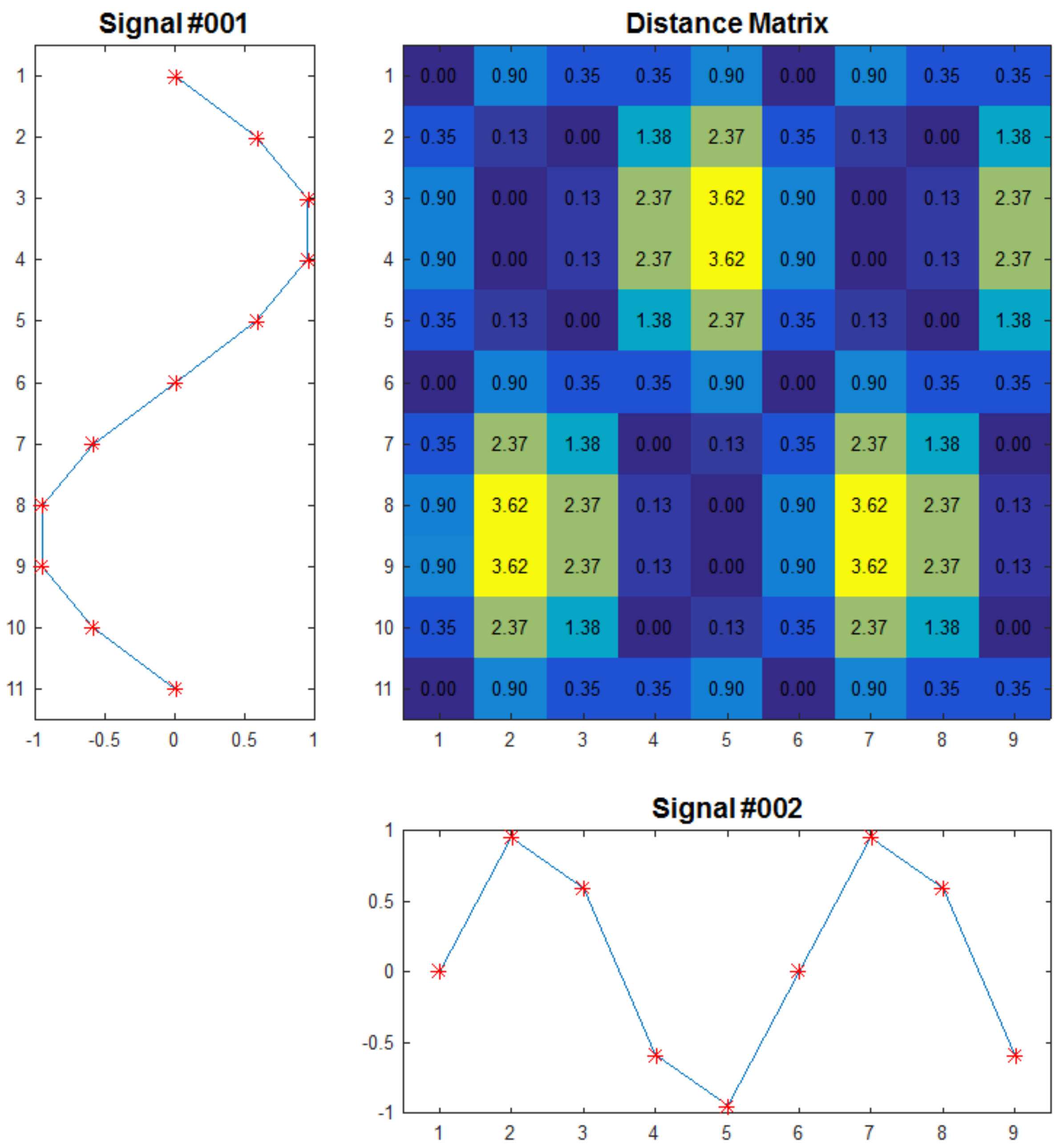
Namely, the DTW provide us a way to measure similarity between signals with no sensitivity to their Time Scale.

Where Can it Be Used?

- Speech Recognition.
- Audio Signature.
- Shape Matching (Font Recognition).

How Can It Be Done?

The answer is by creating a **Distance Matrix** and looking for the Optimal Match.



	1	2	3	4	5	6	7	8	9
1	0.00	0.90	0.35	0.35	0.90	0.00	0.90	0.35	0.35
2	0.35	0.13	0.00	1.38	2.37	0.35	0.13	0.00	1.38
3	0.90	0.00	0.13	2.37	3.62	0.90	0.00	0.13	2.37
4	0.90	0.00	0.13	2.37	3.62	0.90	0.00	0.13	2.37
5	0.35	0.13	0.00	1.38	2.37	0.35	0.13	0.00	1.38
6	0.00	0.90	0.35	0.35	0.90	0.00	0.90	0.35	0.35
7	0.35	2.37	1.38	0.00	0.13	0.35	2.37	1.38	0.00
8	0.90	3.62	2.37	0.13	0.00	0.90	3.62	2.37	0.13
9	0.90	3.62	2.37	0.13	0.00	0.90	3.62	2.37	0.13
10	0.35	2.37	1.38	0.00	0.13	0.35	2.37	1.38	0.00
11	0.00	0.90	0.35	0.35	0.90	0.00	0.90	0.35	0.35

```
vTimeSupport1 = [0:0.1:1].';
vTimeSupport2 = [0:0.2:1.6].';

vY1 = sin(2 * pi * vTimeSupport1);
vY2 = sin(2 * pi * vTimeSupport2);

mDistMtx = bsxfun(@minus, vY1, vY2.') .^ 2;
```

In the case above the Distnace Matrix is given by $D\left(i,j\right) = \left(y_1\left(i\right) - y_2\left(j\right)\right)^2$.
Yet one could use any Distance Measure.

Optimal Match

Given the Distance Matrix the DTW is given by enforcing prior knowledge and extracting optimal path. Paths are called **Warping Path** and defined as a sequence of coordinates $p = \left(p_1, p_2, \cdots, p_L\right)$. The path cost is given by $pathCost = D\left(p_1\right) + D\left(p_2\right) + \cdots + D\left(p_L\right)$. The Optimal (Minimum Cost Path) problem (Generally) can be solved by iterating all the possible paths.

Classic Dynamic Time Warping

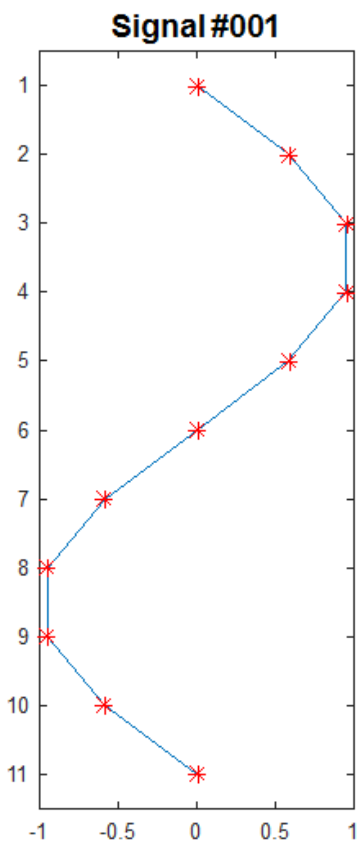
The **Warping Path** p satisfying: 1. Boundary Condition - $p_1 = (1, 1)$ and $p_L = (M, N)$. 2. Monotonicity Condition - $n_1 \leq n_2 \leq \cdots \leq n_L$ and $m_1 \leq m_2 \leq \cdots \leq m_L$. 3. Step Size Condition - $p_{l+1} - p_l \in \left\{\left(1, 0\right), \left(0, 1\right), \left(1, 1\right)\right\}$.

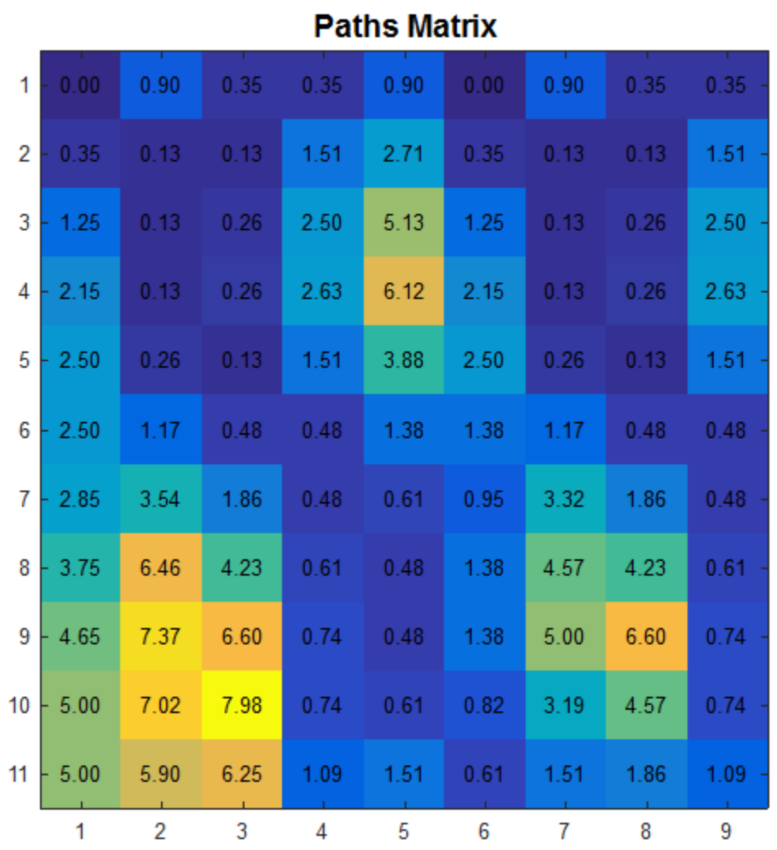
Intuitively, assuming the signals start and end at the same time.
Using the Walking Man metaphor, they start at the same point and end at the same point and each man must stand or walk forward, never go back.

The Optimal (Minimum Cost Path) problem under those constraints is a Dynamic Programming Problem given by:

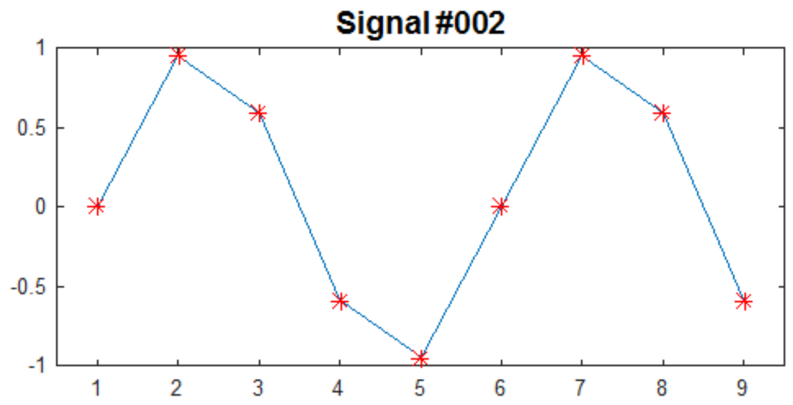
$$\gamma\left(i,j\right) = D\left(i,j\right) + \begin{cases} D\left(1,1\right) + D\left(1,2\right) + \cdots D\left(1,j-1\right) & \text{if } i = 1 \\ D\left(1,1\right) + D\left(2,1\right) + \cdots D\left(i-1,1\right) & \text{if } j = 1 \\ \min\left\{\gamma\left(i-1,j\right), \gamma\left(i,j-1\right), \gamma\left(i-1,j-1\right)\right\} & \text{if } i > 1, j > 1 \end{cases}$$

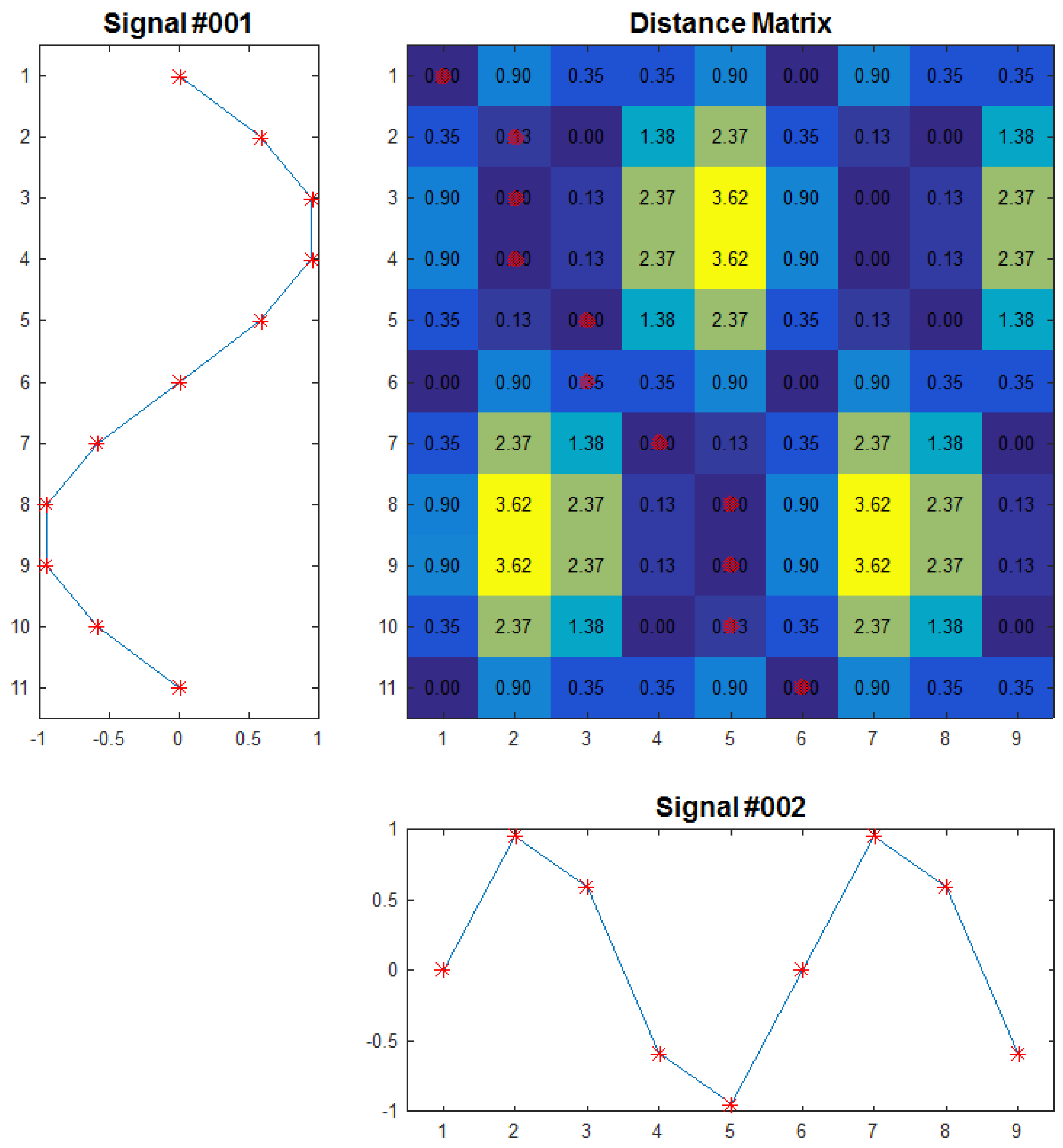
Where $\gamma\left(i,j\right)$ stands for the minimum cost to reach to point $\left(i,j\right)$.





	1	2	3	4	5	6	7	8	9
1	0.00	0.90	0.35	0.35	0.90	0.00	0.90	0.35	0.35
2	0.35	0.13	0.13	1.51	2.71	0.35	0.13	0.13	1.51
3	1.25	0.13	0.26	2.50	5.13	1.25	0.13	0.26	2.50
4	2.15	0.13	0.26	2.63	6.12	2.15	0.13	0.26	2.63
5	2.50	0.26	0.13	1.51	3.88	2.50	0.26	0.13	1.51
6	2.50	1.17	0.48	0.48	1.38	1.38	1.17	0.48	0.48
7	2.85	3.54	1.86	0.48	0.61	0.95	3.32	1.86	0.48
8	3.75	6.46	4.23	0.61	0.48	1.38	4.57	4.23	0.61
9	4.65	7.37	6.60	0.74	0.48	1.38	5.00	6.60	0.74
10	5.00	7.02	7.98	0.74	0.61	0.82	3.19	4.57	0.74
11	5.00	5.90	6.25	1.09	1.51	0.61	1.51	1.86	1.09



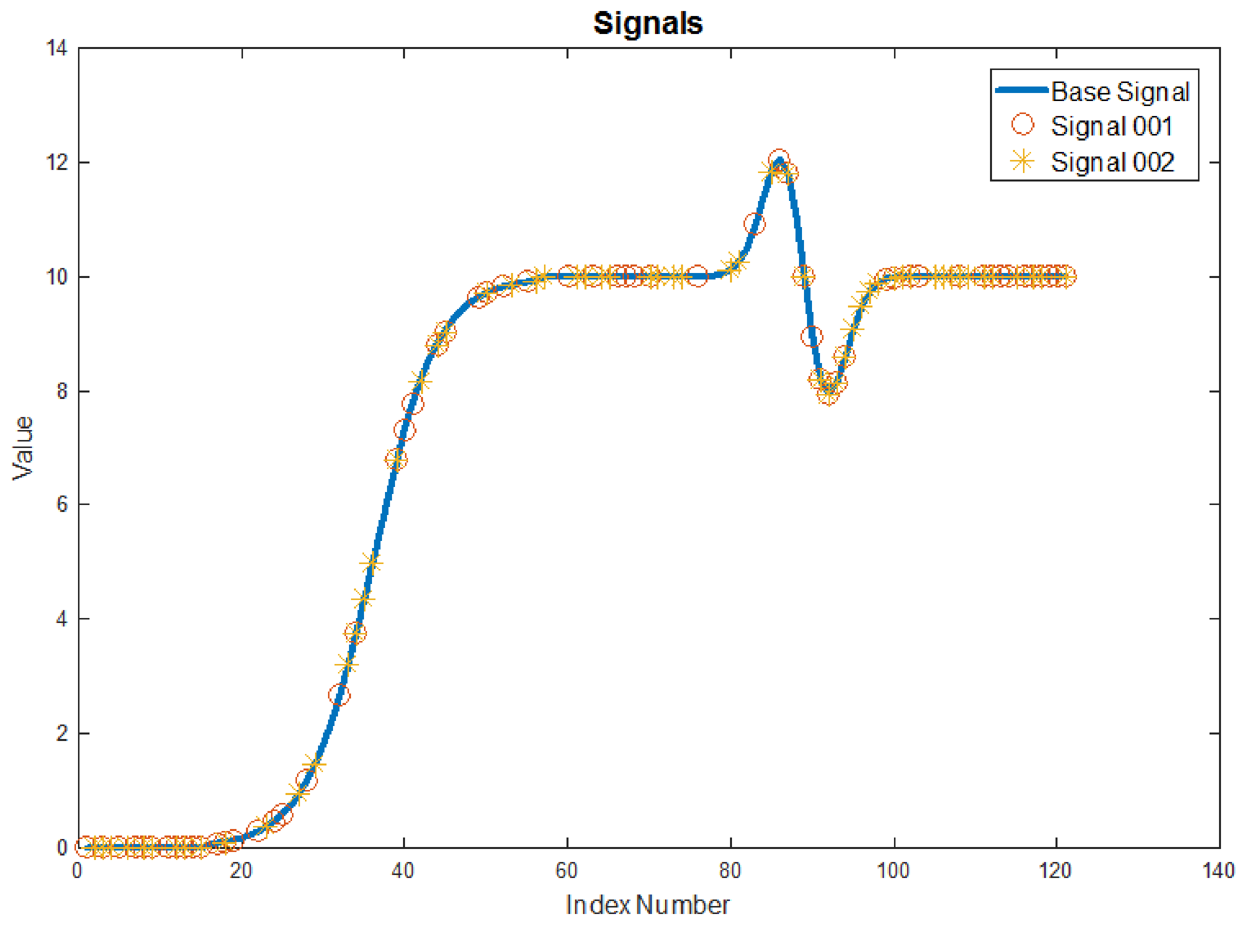


Different Flavors

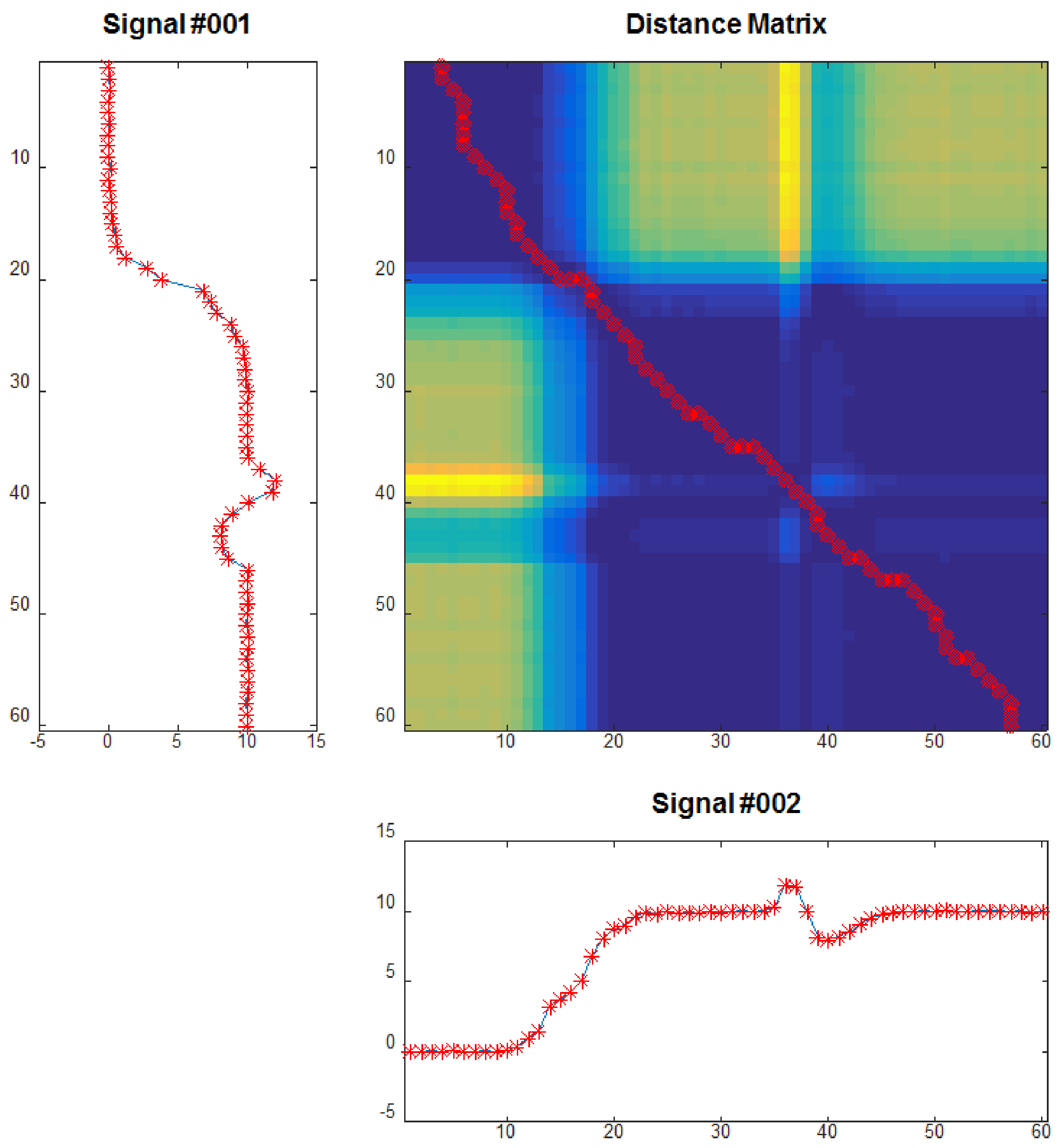
- Allowing shifted start / end point (Removing Constrain #001).
  Update the "Initial Conditions" in the algorithm.
- Enforcing maximum distance between samples.
- Favoring Horizontal / Vertical / Diagonal Movement. For cases with prior knowledge on the path. Add weights for the "Base Points" for the current location.

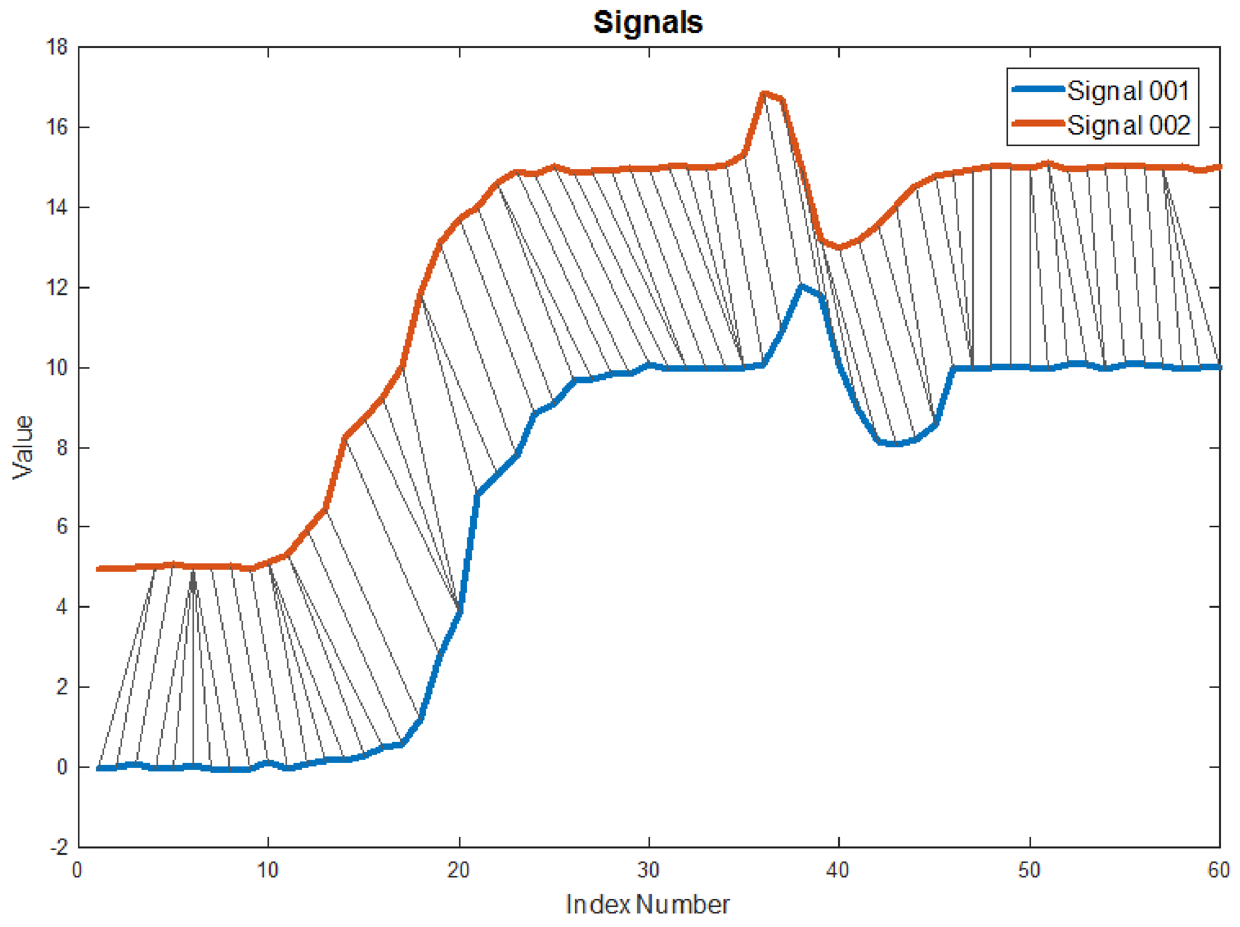
- Different Step Size (Removing Constrain #003). Add "Base Points" to get to the current location.
- Multi Scale approach To find similar Sub Sequences.
-

Example



```
numSamples   = 60;
noiseStd     = 0.05;
shiftVal     = 5;

vX                    = [-1:0.05:1];
vGaussianDerivative = CalcGaussianGradient(3, 4);
vSigmoidSignal        = 1 ./ (1 + exp(-5 * vX));
vBaseSignal           = [zeros([1, 15]), vSigmoidSignal, ones([1, 20]), (1 + vGaussianDer
vBaseSignal           = 10 * vBaseSignal(:);
numSamplesBase        = size(vBaseSignal, 1);
vXBase                = [1:numSamplesBase].';

vY1Idx = sort(randperm(numSamplesBase, numSamples));
vY2Idx = sort(randperm(numSamplesBase, numSamples));

vY1 = vBaseSignal(vY1Idx);
vY2 = vBaseSignal(vY2Idx);

vY1 = vY1 + (noiseStd * randn([numSamples, 1]));
vY2 = vY2 + (noiseStd * randn([numSamples, 1]));
```


Signal #001
Distance Matrix
Signal #002


Signals
Signal 001
Signal 002
Value
Index Number
18
16
14
12
10
8
6
4
2
0
-2
0
10
20
30
40
50
60