Source code documentation for:

[1] Malvestio I, Kreuz T, Andrzejak RG:
“Robustness and versatility of a nonlinear interdependence method for directional coupling detection from spike trains.”
Physical Review E, 96 (2017) 022203.

Relevant literature:

[2] Andrzejak RG, Kreuz T: “Characterizing unidirectional couplings between point processes and flows.” EPL, 96 (2011) 50012.

[3] Satuvuori E, Mulansky M, Bozanic N, Malvestio I, Zeldenrust F, Lenk K, Kreuz T: "Measures of spike train synchrony for data with multiple time scales." Journal of Neuroscience Methods 287 (2017): 25-38.

[4] http://wwwold.fi.isc.cnr.it/users/thomas.kreuz/Source-Code/cSPIKE.html.

The source code allows you to calculate the measure L first introduced in [2] with the improvements introduced in [1]. Please cite Ref. [1] if you use this code.

In order to understand the following documentation you should at first read Ref. [1]. In particular, we here use the same notation for mathematical symbols as in this paper. Additional comments can be found throughout the source code. In case you have any questions, please contact the authors at malvestio.irene@gmail.com.

To get started, copy the folder `cSPIKEmex` and the files: `MalvestioPRE2017Example.m`, `MalvestioPRE2017HpointProcess.m`, `MalvestioPRE2017FromHtoL.m`, `MalvestioPRE2017Example024.mat`, `InitializecSPIKE.m`, `SpikeTrainSet.m` into some directory.

To run the program, call the function `MalvestioPRE2017Example(signal1, signal2, Q, s_div, N_wo, dchoice, k)`.
If you do not insert any input parameters, then the simulated example stored in `MalvestioPRE2017Example024.mat` will be loaded (see below for more details). This file includes `signal1` and `signal2`. Furthermore, the other input parameters will be set to default values.

`signal1`: Times of the spikes in the first signal, vector of dimension `(1, N1)`, where `N1` is the number of spikes in the first signal.
`signal2`: Analogous to `signal1` but for the second spike train.

`Q`: total recording time in the same time unit as the spike times.
`s_div`: Factor of overlap between windows (s_div = q/s).
`N_wo`: Number of windows without overlap (N_wo = Q/q).
`dchoice`: Allows you to select the spike train distance
1 : ISI-distance
2 : SPIKE-distance
3 : Adaptive ISI-distance
4 : Adaptive SPIKE-distance

`k`: Number of nearest neighbors

The example we provide in the file `MalvestioPRE2017Example024.mat` is derived from coupled Hindmarsh-Rose dynamics studied in Ref. [1] to illustrate the code. It includes two example signals:

signal1: A point process derived from dynamics X.
signal2: A point process derived from dynamics Y.

The dynamics X is driving dynamics Y with coupling strength epsilon = 0.24.

The matrix of the ranks, derived from the distance matrix, is calculated using:

```
function
HP = MalvestioPRE2017Hpointprocess(spiketimes,Q,q,s,W,W_ex,dchoice)
```

Here `spiketimes` is a vector containing the times of the spikes. The length of the vector is determined by the number of spikes. The other parameters are automatically derived from the ones described above in the function `MalvestioPRE2017Example`.

In Ref. [1] the performance of the adaptive ISI- and the adaptive SPIKE-distance are compared (so `dchoice = 3` and `dchoice = 4`). In this release of the source code also the original versions of the ISI- and the SPIKE-distance are implemented.
In contrast, the Victor-Purpura distance and the van Rossum distance are not included here. If you are interested in the code, they are available here:
http://www.fi.isc.cnr.it/users/thomas.kreuz/sourcecode.html.

The output `HP` is a matrix of dimension (`N_w, N_w`) with elements HP(i,j) = g_{i,j} as introduced in Ref. [1] just before Eq. 1.
Compared to previous versions of this code (see source code documentation for [2]), this function gives as output the rank derived from the distance matrix (g_{i,j}), instead of the distance matrix (d_{i,j}). This choice makes the code considerably faster.

The folder cSPIKEmex, and the files `SpikeTrainSet.m` and `InitializecSPIKE.m` are needed to calculate the distance matrix with the recently introduced implementation of the adaptive versions introduced in [3]. It uses MEX files with C++ backend in order to increase the speed. For a detailed description please refer to [4].

The main function is:

```
function [Lj, L] = MalvestioPRE2017FromHtoL(Hranks,k,W)
```

`Hranks` is a 3-D array of size `(N_w, N_w, N_node)`. `N_node` is the number of signals for which L is computed. In the case of the paper `N_node = 2`.
For each signal `w = 1: N_node`, `Hranks(i, j, w)` is g_{i,j} for the signal w. The individual rank matrix for each signal can be obtained with the function `MalvestioPRE2017Hpointprocess` described above.

`L` is a 2-D array of size (`N_node, N_node`). For example, L(1,2) contains L(signal1| signal2) = L(spike train from the driving dynamics X| spike train from the response dynamics Y). The diagonal is set to zero. For the present example, since X is driving Y, we get a higher value for L(1|2) than for L(2|1).

`Lj` is a 3-D array of size (`N_node, N_node, N_w`). It corresponds to the values of L for separate individual windows, before the final average is taken. In the paper these values are not used, but we provide them here because they can be useful for checking the meaningfulness of L in different settings.
