Local Symbolic Transfer Entropy

# Makefile\_lte\_mpi --- make file for compiling and creating the executable lte.exe (MPI version); To be used as: make –f Makefile\_lte\_mpi

# Makefile\_lte\_ser --- make file for compiling and creating the executable lte\_ser.exe (serial version); To be used as: make –f Makefile\_lte\_ser

# This class contains the following files:

# demo\_lte.f90

This file is initializing the LTE\_CLASS, which is the module used for calculation of local transfer entropy. It contains the subroutines/functions used to read the input parameters for computation of local transfer entropy between the time series.

# This subroutine calls one of the following functions (which are part of the LTE\_CLASS in localmte.f90 (MPI version) and localmte\_ser.f90 (serial version):

# IF (qTEMethod == 1) THEN

# CALL LSTE\_DRIVER(Nframes, Ndim, Natoms,qteShuffle, qteNorm, debug, Nshuffles, Rcut, StatP,frm) -- driver for symbolic local transfer entropy

# ELSE

# CALL LTE\_DRIVER(Nframes, Ndim, Natoms, qteShuffle, qteNorm, debug, Nshuffles, Rcut, StatP,frm) -- driver for symbolic local transfer entropy using Schreiber method

# ENDIF

# subroutine WRITE\_LTE(qTEShuffle) -- Writes on output the computed local transfer entropies

# subroutine Allocate\_LocalTransferEntropy(qTeShuffle, model) -- It allocates memory for global dynamical variables of the LTE\_CLASS

# subroutine deAllocate\_LocalTransferEntropy() -- It frees the allocated memory for the global dynamical variables of the LTE\_CLASS

# subroutine MPI\_BroadCast(qteShuffle, qteNorm, debug, Nshuffles, r0, statP, model) -- It broad casts the global variables of LTE\_CLASS to all processors (MPI version)

# subroutine getLocalSTransferEntropyNDIM\_MPIDOF(qteNorm, debug) -- It computes local symbolic transfer entropy without shuffling (MPI version) using method by Kamberaj & Van der Vaart

# subroutine getLocalSTransferEntropyNDIM\_MPISFL(qteShuffle, qteNorm, debug, Nshuffles, r0, statP) -- It computes local symbolic transfer entropy with shuffling (MPI version)

# -- using method by Kamberaj & Van der Vaart

# subroutine getLocalTransferEntropyNDIM\_MPIDOF(qteNorm, debug) -- It computes local symbolic transfer entropy without shuffling (MPI version) using Schreiber method

# subroutine getLocalTransferEntropyNDIM\_MPISFL(qteShuffle, qteNorm, debug, Nshuffles, r0, statP) -- It computes local symbolic transfer entropy with shuffling (MPI version)

# -- using Schreiber method

#

# Auxiliary functions/subroutines:

# subroutine symbolic\_LTE2\_entropies(ndata,ndim,xs,ys,m1,m2,tau1,tau2,LTxy,LTyx) -- It computes local symbolic transfer entropies (LTxy, LTyx) between two symbolic time series xs, ys

# -- using Schreiber method

# subroutine symbolic\_LTE2\_entropy(ndata,ndim,xs,ys,m1,m2,tau1,tau2,LTxy) -- It computes local symbolic transfer entropy (LTxy) between two symbolic time series xs, ys

# -- using Schreiber method

# subroutine symbolic\_LTE\_entropies(ndata,ndim,xs,ys,m1,m2,tau1,tau2,LTxy,LTyx) -- It computes local symbolic transfer entropies (LTxy, LTyx) between two symbolic time series xs, ys

# -- using method by Kamberaj & Van der Vaart

# subroutine symbolic\_LTE\_entropy(ndata,ndim,xs,ys,m1,m2,tau1,tau2,LTxy) -- It computes local symbolic transfer entropy (LTxy) between two symbolic time series xs, ys

# -- using method by Kamberaj & Van der Vaart

# The Input Parameters of the Module:

Length of time series - **Nframes**

Number of time series - **Natoms**

Dimensionality of the problem - **Ndim**

Flag for Method of TE calculation (1: Discrete (m=1); 2: Discrete (m>1); 3: Schreiber) - **qTEMethod**

Flag for Normalization of TE (0: No normalization; 1: normalization - **qTENorm**

Flag for Method of Shuffling of TE (1: Permutation; 2: Block shuffling) - **qTEShuffle**

Number of Shuffling of time series - **Nshuffles**

Cutoff for Mutual Information minimum value - **Rcut**

Confidence level for averages - **statP**

Set debugging flag value - **Debug**

Chose the format of Input/Output data – **frmt**

A bash shell script is given below:

!#/bin/bash

root=../sifm-master

nframes=10000

natoms=2

ndim=1

qTEMethod=2

qTENorm=0

qTEShuffle=1

Nshuffles=10

Rcut=0.02

statP=0.95

Debug=1

parallel=0

NP=2

frmt=’csv’

if [ parallel == 1 ]; then

mpirun -np $NP ${root}/te.exe ${nframes} ${natoms} ${ndim} ${qTEMethod} ${qTENorm} ${qTEShuffle} ${Nshuffles} ${Rcut} ${statP} ${Debug} ${frmt}

else

${root}/te\_ser.exe ${nframes} ${natoms} ${ndim} ${qTEMethod} ${qTENorm} ${qTEShuffle} ${Nshuffles} ${Rcut} ${statP} ${Debug} ${frmt}

fi