Symbolic Transfer Entropy

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

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

# This MODULE contains the following files:

# demo\_te.f90

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

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

# IF (qTEMethod == 1) THEN

# CALL STE1\_DRIVER(frm,Nframes, Ndim, Natoms,qteShuffle, qteNorm, debug, Nshuffles, Rcut, StatP)

# ELSEIF (qTEMethod == 2) THEN

# CALL STE2\_DRIVER(frm,Nframes, Ndim, Natoms,qteShuffle, qteNorm, debug, Nshuffles, Rcut, StatP)

# ELSEIF (qTEMethod == 3) THEN

# CALL STE3\_DRIVER(frm,Nframes, Ndim, Natoms,qteShuffle, qteNorm, debug, Nshuffles, Rcut, StatP)

# ELSE

# CALL TE\_DRIVER(frm,Nframes, Ndim, Natoms, qteShuffle, qteNorm, debug, Nshuffles, Rcut, StatP)

# ENDIF

# TE\_CLASS contains the following subroutines/functions:

# subroutine write\_TE(qTeShuffle, model) -- It prints out the transfer entropy

# subroutine Allocate\_TransferEntropy(qteShuffle, model) -- It allocates memory for global dynamical variables of TE\_CLASS

# subroutine deAllocate\_TransferEntropy() -- It frees allocated memory for global dynamical variables of TE\_CLASS

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

# subroutine getCrossSTransferEntropyNDIM\_MPIDOF(qteNorm, debug) -- It computes symbolic TE between two dynamical variables X(i,1,1:d), X(i,2,i:d), where

# -- i = number of pairs; d = dimensionality of the problem - MPI version

# -- No shuffling applies

# -- using method by Kamberaj & Van der Vaart

# subroutine getSTransferEntropyNDIM\_MPIDOF(qteNorm, debug) -- It computes symbolic TE between dynamical variables presented as matrix X(1:N,1:d)

# -- using method by Kamberaj & Van der Vaart

# -- MPI version

# -- No shuffling applies

# subroutine getSTransferEntropyNDIM\_MPISFL(qteShuffle, qteNorm, debug, Nshuffles, r0, statP) -- It computes symbolic TE between dynamical variables presented as matrix X(1:N,1:d)

# -- using method by Kamberaj & Van der Vaart

# -- MPI version

# -- Shuffling applies

# subroutine getTransferEntropyNDIM\_MPIDOF(qteNorm, debug) -- It computes symbolic TE between dynamical variables presented as matrix X(1:N,1:d)

# -- using Schreiber Method

# -- MPI version

# -- No shuffling applies

# subroutine getTransferEntropyNDIM\_MPISFL(qteShuffle, qteNorm, debug, Nshuffles, r0, statP) -- It computes symbolic TE between dynamical variables presented as matrix X(1:N,1:d)

# -- using Schreiber Method

# -- MPI version

# -- Shuffling applies

#

# Some auxiliary functions / subroutines:

# symbolic\_TE\_entropies1D(ndata,xs,ys,m1,m2,tau1,tau2,Txy,Tyx,Hxx,Hyy,hx,hy) -- It computes symbolic transfer entropies between two symbolic strings time series x and y

# -- using method by Kamberaj & Van der Vaart

# subroutine symbolic\_TE\_entropiesND(ndata,ndim,xs,ys,m1,m2,tau1,tau2, Txy,Tyx,Hxx,Hyy,hx,hy) --- The same as above but for Ndim symbolic time series

# --- using method by Kamberaj & Van der Vaart

# subroutine symbolic\_TE\_entropy1D(ndata,xs,ys,m1,m2,tau1,tau2,Txy) -- It computes only Txy symbolic transfer entropy between two symbolic strings time series x and y

# -- using method by Kamberaj & Van der Vaart

# subroutine symbolic\_TE\_entropyND(ndata,ndim,xs,ys,m1,m2,tau1,tau2,Txy) - The same as above but for Ndim symbolic time series using method by Kamberaj & Van der Vaart

# subroutine symbolic\_TE2\_entropies1D(ndata,xs,ys,m1,m2,tau1,tau2,Txy,Tyx,Hxx,Hyy,hx,hy) -- It computes symbolic transfer entropies between two symbolic strings time series x and y

# -- using Schreiber method

# subroutine symbolic\_TE2\_entropiesND(ndata,ndim,xs,ys,m1,m2,tau1,tau2,Txy,Tyx,Hxx,Hyy,hx,hy) - The same as above but for Ndim real variable time series

# subroutine symbolic\_TE2\_entropy1D(ndata,xs,ys,m1,m2,tau1,tau2,Txy) -- It computes only Txy symbolic transfer entropy between two symbolic strings time series x and y

# -- using Schreiber method

# subroutine symbolic\_TE2\_entropyND(ndata,ndim,xs,ys,m1,m2,tau1,tau2,Txy) - The same as above but for Ndim real variable time series

# 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}/terun.exe ${nframes} ${natoms} ${ndim} ${qTEMethod} ${qTENorm} ${qTEShuffle} ${Nshuffles} ${Rcut} ${statP} ${Debug} ${frmt}

else

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

fi