```matlab
%% input and output variables
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [data]=plot_stationarity(data,save_path,save_name)
% This function plots the mean, standard deviation, skewness and kurtosis of sections of a length of
% 5\% of the data to check the stationarity of the data. In the title of the figure the number of
% nan's in the dataset and the turbulence intensity is printed.
%
% Arguments IN
% data = 1D array of data
% save_name = name for saving files
%
% Arguments OUT
% data = 1D array of data without nan's
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function plot_pdf(data,increment_bin,save_path,save_name)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This function plots the probability density function (PDF) of the data with the specified number
% of bins. It also plots the Gaussian distribution which has the same standard deviation and mean as
% of the data. In the title of the figure the range of the data (difference between the maximum and
% minimum values of sample data), the skewness and flatness of the data is printed.
% COMMENT: hist is a standard matlab function==>Look in to standard MATLAB documentation
%
% Arguments IN
% data  = 1D array for which you would like to plot the probability density function(PDF)
% increment_bin = The number of bins in which you would like to divide your data
% save_path = path for saving figures and files
% save_name = name for saving files
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function  [f, E_f_no_filter, f_avg_no_filter, E_avg_no_filter, P_avg_no_filter,K41_index]=spectrum(data,Fs,↙
increment_bin)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This function calculates the energy spectral density (ESD) of the hot wire signal using
% $\textbf{\textit{fft}}$ function of MATLAB. Also, the ESD with and without averaging
% (moving average with equally spaced frequency interval in log-space) as a function of frequency
% will be plotted.
%
% Arguments IN
% data = 1D array for which you would like to plot power spectram density(PSD)
% Fs= Sampling frequency in Hz
% increment_bin = The number of bins in which you would like to divide your data
%
% Arguments OUT
% f = Frequency without smoothing
% E_f_no_filter = Energy spectral density(ESD) without smoothing
% f_avg_no_filter = Frequency with smoothing
% E_avg_no_filter = Energy spectral density(ESD) with smoothing
% P_avg_no_filter = Power spectral density(ESD) with smoothing
% K41_index = index to the frequency which will be used to fit $f^{(-5/3)}$
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function    [data_filter,f_avg_filter,E_f_avg_filter,k_avg_filter,Ek_avg_filter,r_avg_filter,Dr_avg_filter,↙
Dk_avg_filter] = frequency_filter(data,Fs,low_freq,K41_index,kin_vis,filter,save_path,save_name);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This function returns the filtered data using the low pass filter at the previously set frequency
% (using $\textbf{\textit{butter}}$ and $\textbf{\textit{filtfilt}}$ function of MATLAB). The
% filtered data named as $\textbf{data\_filter}$ for all the further data post-processing. If the
% filtering was negated in the previous step, $\textbf{data\_filter}$ and $\textbf{data}$ are equal.
% In addition, different representation/normalization of the energy spectrum density with respect to
% frequency $f$, scale $r$, wave number $k$ will also be plotted.
%
% Arguments IN
% data = 1D array of data to which you would like to apply the low pass filter
% Fs = Acquisition/Sampling Frequency in Hz
% low_freq = Frequency at which you would like to use the low-pass filter in Hz
% K41_index = index to the frequency which will be used to fit $f^{(-5/3)}$
% kin_vis = Kinematic viscosity of fluid in m^2/sec
% filter = logical data type represents true and false states using the
% numbers 1 and 0 if data is filtered (Low-pass filter)
% save_path = path for saving figures and files
% save_name = name for saving files
%
% Arguments OUT
% data_filter = this is the filtered data which will be returned by this function
% f_avg_filter = Frequency after filtering
% E_f_avg_filter = Frequency Spectrum after filtering
% k_avg_filter = k waven umbers after filtering
% Ek_avg_filter = Wavenumber Spectrum (m^3/sec^2) after filtering
% r_avg_filter = r Spektrum; Scales,r (m)
% Dr_avg_filter = Dissipation Spectrum; scale domain; after filtering
```

```matlab
% Dk_avg_filter = Dissipation Spectrum; wave number domain; after filtering
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [int_L,taylor_L,int_L_calc, taylor_L_calc,epsi,epsi_calc,diss_scale,Ce, Ce_calc, Re, Re_lambda ] =
length_scales(data_filter,Fs,low_freq,kin_vis,C2,increment_bin,m_data,save_path,save_name);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% In this function the integral length scale $L$, Taylor length scale $\lambda$ and Kolmogorov
% length scale are estimated using different methods of calculation. Within this function, pop-up
% dialog boxes will be generated to enter the values of the integral, Taylor length scale in $m$
% and energy dissipation rate in $m^2/s^3$ on which the further processing of data (solving the
% FPE and extracting cascade trajectories) should be referred. The entered length scales will be
% round towards the nearest integer sample. The proposed value in the pop-up dialog box is the
% median length scale for all methods.
% COMMENT: All velocity increments in this script are taken as right hand increment i.e.
% dV=V(t2)-V(t1) where t2>t1; t1,t2 in seconds
%
% Arguments IN
% data_filter = it is the filtered data
% Fs = Acquisition/Sampling Frequency in Hz
% low_freq = Frequency at which you would like to use the low-pass filter in Hz
% kin_vis = Kinematic viscosity of fluid in m^2/sec
% C2 = constant, C2 associated with second order structure function(in between 2.0 to 2.2)
% increment_bin = The number of bins in which you would like to divide your data
% save_path = path for saving figures and files
% save_name = name for saving files
%
% Arguments OUT
% int_L = Integral length scale in meters
% taylor_L = Taylors length scale in meters
% int_L_calc = 1D array of integral length scale calculated by different methods/formula
% taylor_L_calc = 1D array of Taylor's length scale calculated by different methods/formula
% epsi = mean energy dissipation rate
% epsi_calc = 1D array of mean energy dissipation rate calculated by different methods/formula
% diss_scale = Kolmogorv lenght scale in meters
% Ce = is the non-dimentional / normalized energy dissipation rate
% Ce_calc = 1D array of Ce calculated by different methods/formula
% Re = Reynolds Number  based on integral length scale
% Re_lambda= Reynolds Number based on Taylor's length scale
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function Struc_flip_test(Fs,data_filter,int_L,taylor_L,save_path,save_name)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This function tests whether the data have to be flipped or not. The decision of flipping of data
% depends on a simple relation of $3^{rd}$ order structure function ($S^{3}$) with the dissipation
% based on the assumption of homogeneous isotropic turbulence (HIT). The thumb rule is that the
% quantity ($S^{3}$) must be negative. In the literature, the keyword that goes with this picture
% is vortex stretching. To verify this, a plot of ($S^{3}$) as a function of the scale $r$ is
% plotted, from which it is possible to decide whether it is essential to flip the data or not.
% The condition is S3 should always be negative! If we see S3 to be positve===> we flip the data
% Actual flipping will be done in next function called 'normalization'
%
% Arguments IN
% data_filter = it is the filtered data
% Fs = Acquisition/Sampling Frequency in Hz
% int_L = Integral length scale in meters
% taylor_L = Taylors length scale in meters
% save_path = path for saving figures and files
% save_name = name for saving files
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [data_filter,siginf,m_data] = normalization(data_filter,askYesno('Do you want to flip hotwire data?',
'Yes'),askYesno('Do you want to perform the normalization of the data using $\sigma_\infty$?', 'Yes'));
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This function is mainly to perform the normalization of the data. Before doing so, it generates
% the pop-up dialog box which asks the user whether to flip the data or not (based on the previous
% investigation). After that, this function normalizes the entire data with the quantity of
% $\sigma_{\infty} = \sqrt{2}\sigma$, where $\sigma$ is the standard deviation of the
% \textbf{data\_filter} (this method is proposed by \cite{renner2001}). This function returns the
% filtered and normalized data as \textbf{data\_filter}, \mbox{\textbf{siginf} = $\sigma_{\infty}$}
% and \mbox{\textbf{m\_data} = mean of the data} before normalization. In addition scale $r$ is
% given in units of Taylor length scale $\lambda$. We use this normalization to compare the results
% of different data sets.
%
% Arguments IN
% data_filter = it is the filtered data

% Arguments OUT
% data_filter= flipped or non-flipped filterd data based on your input as 'tmp_flip'
% siginf=Quantity used to non-dimentionalized the velocity time series
% m_data=mean of the filtered data
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
function plot_increment_pdf(data_filter,increment_bin,save_path,save_name,int_L,taylor_L,m_data,Fs,norm_ur,↙
diss_scale)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This function plots the probability density function (PDF) of the velocity increments
% at the scale $r=L$, $r=\lambda$ and  $r=\eta$. The colored dashed line correspond to Castaing fits
% (form factor $\Lambda_r$ \cite{Castaing_1990}) and grey dashed line to Gaussian fits.
%
% Arguments IN
% data_filter = 1D array of data to which you would like to apply the low pass filter
% increment_bin = The number of bins in which you would like to divide your data
% save_path = path for saving figures and files
% save_name = name for saving files
% m_data = mean of the data
% Fs = Acquisition/Sampling Frequency in Hz
% int_L = Integral length scale in meters
% taylor_L = Taylors length scale in meters
% norm_ur = normalization of the data using $\sigma_\infty$? data 1=Yes, 0=No
% diss_scale = Kolmogorv lenght scale in meters
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [r_struc,S_exp_2,S_exp_3,S_exp_4,S_exp_5,S_exp_6,S_exp_7]=plot_struc_function(r_n,Fs,data_filter,↙
m_data,int_L,taylor_L, mu, D,norm_r,save_path,save_name);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This function plots the $k$-th order structure function $S^{k}(r)$ with $k={1-7}$ for scales
% $\lambda \leq r \leq L$. Dashed line represents -4/5 law. The 4/5 law states that 3rd-order
% structure functions of longitudinal velocity increments should scale linearly with their
% displacement distances. The 3rd-order structure function is one of the most fundamental of all
% Navier-Stokes equation results; it is exact, with no adjustable constants, so any model that is
% expected to produce accurate turbulence results must reasonably well duplicate this.
%
% Arguments IN
% r_n = Number of seperated scales between Integral and Taylor (calculation of structure functions)
% Fs = Acquisition/Sampling Frequency in Hz
% data_filter = it is the filtered data
% m_data = mean of the data
% int_L = Integral length scale in meters
% taylor_L = Taylors length scale in meters
% mu = intermittency coefficient mu
% D = Beta Model Coefficient D
% norm_r = normalization of the scale using $\lambda$? data 1=Yes, 0=No
% save_path = path for saving figures and files
% save_name = name for saving files
%
% Arguments OUT
% r = scales between Integral and Taylor (calculation of structure functions)
% S_exp = $k$-th order structure function $S^{k}(r)$ with $k={1-7}$
% Lam_sq_L_1 = shape parameter
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function markov=wilcoxon_test(data_filter,Fs,m_data,int_L,taylor_L,diss_scale,increment_bin,save_path,save_name,↙
f_avg_filter, E_f_avg_filter);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This function determines the Einstein-Markov length $\Delta_{EM}$\cite{renner2001}. Above this
% length scale, the Markov properties hold and below this length scale, the Markov properties cease
% to hold. The Wilcoxon test is a parameter-free procedure to compare two empirically determined
% probability distributions (two data sets of velocity increments) (see \cite{Lueck2006markov} for
% details). It is a quantitative test that determines the $\Delta_{EM}$. A sufficient resolution
% in measurement below Taylor's length scale is expected to perform this test. Also, a vertical
% dashed line at the Taylor length scale $\lambda$ will be added to the plot.
% COMMENT: for mulitpoint include Increment_point(tau1,tau2,d,condition,tol)
%
% Arguments IN
% data_filter = it is the filtered data
% Fs =Acquisition/Sampling Frequency in Hz
% m_data = mean of the filtered data
% int_L = Integral length scale in meters
% taylor_L = Taylor length scale in meters
% diss_scale = Kolmogorv lenght scale in meters
% increment_bin = The number of bins in which you would like to divide your data
% save_path = path for saving figures and files
% save_name = name for saving files
% f_avg_filter = Frequency with smoothing;
% E_f_avg_filter = Energy spectral density(ESD) with smoothing
%
% Arguments OUT
% markov = Einstein-Markov length scale in samples
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function conditional_PDF_markov(data_filter,Fs,m_data,markov,second_condi,increment_bin,min_events,norm_ur,↙
save_path,save_name)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% The main work of this function is to plot the single conditioned and double conditioned PDF's and
% check visually if the markovian property is fulfilled or not. This is a qualitative check.
% This function calculates the single conditioned and double conditioned PDF of velocity increments
% for a pair of two scales each of which is separated by the $\Delta_{EM}$. To do this, a pop-up
% dialog box is generated to enter the conditioned value for large scale increment $u_{r_{3}}$, for
% example $u_{r_{3}}=\pm 1$. Note, the condition $u_{r_{3}}=0$ corresponds to the maximum number of
% statistics. This function also plots various representations of the single and double conditioned
% PDF's. This is a qualitative/visual check for the validation of Markov property based on the
% alignment or misalignment of these single and double conditioned PDF's. If there is not a good
% agreement between the single conditioned and double conditioned PDF of velocity increments, it is
% possible to modify the Einstein-Markov length and/or the minimum number of events and repeat this
% qualitative check.
%
% Arguments IN
% data_filter = it is the filtered data
% Fs = Acquisition/Sampling Frequency in Hz
% m_data = mean of the filtered data
% markov = markov length in samples/steps NOT in meters
% second_condi = increment of the second condition
% increment_bin = Number of bins
% min_events = minimum number of events to consider in a single bin
% norm_ur = normalization of the scale using $\lambda$? data 1=Yes, 0=No
% save_path = path for saving figures and files
% save_name = name for saving files
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [condition,tol] = multi_point_condition(data_filter);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Arguments IN
% data_filter = filtered data
%
% Arguments OUT
% condition = This input is for multipoint statistics
% tol = This input is for multipoint statistics
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [evaluated,step_con_moment] = conditional_moment(low_freq,Fs,markov,m_data,int_L,taylor_L,scale_steps,↙
increment_bin,data_filter,multi_point,condition,tol);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This function estimates the $k$-order conditional moment M^{(k)}\left(u_r,r,\Delta r\right)
% with $k={1,2,3,4}$ for all scales $2\Delta_{EM} <r \leq L$ (specified in the function
% \textbf{\textit{scale\_steps}}) and for each bin (specified in the function
% \textbf{\textit{increment\_bin}}) for all values of longitudinal velocity increments $u_r$. For a
% fixed scale $r$ the conditional moments are calculated for 5 different scales separations
% $\Delta r=r-r'$ within the range of $\Delta_{EM}\leq \Delta r\leq 2\Delta_{EM}$.
% The condition $r'<r$ is always fulfilled.
%
% Arguments IN
% low_freq = Frequency at which you would like to use the low-pass filter in Hz
% Fs = Acquisition/Sampling Frequency in Hz
% markov = markov length in number of samples
% m_data = mean of the data
% int_L = Integral length scale in meters
% taylor_L = Taylor length scale in meters
% scale_steps = Number of seperated scales between Integral and Taylor length
% increment_bin = number of bins
% data_filter = filtered data
% multi_point = Multipoint condition 1=YES or 2=NO
% condition = This input is for multipoint statistics
% tol = This input is for multipoint statistics
%
% Arguments OUT
% evaluated = struct array containing all the information about conditional moments for each scale and for each↙
bin
% step_con_moment = the steps at which the value of conditional moments are calculated
% Enclosed in a "evaluated":
% r is the scale in meters at which moments will be calculated and hence this r will
% be the same at which D1 & D2 will be calculated===>r2
% r_samp is the r in number of samples==>r2
% r_short_sample is r1 ===> (r2>r1)
% M11 = First order conditional moment
% M21 = Second order conditional moment
% M31 = Third order conditional moment
% M41 = Fourth order conditional moment
% eM1 = Error associated with M11
% eM2 = Error associated with M21
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function  plot_conditional_moment(scal,bin_num,evaluated,step_con_moment,markov,multi_point,increment_bin,↙
condition,tol,data_filter,save_path,save_name);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% This function plots the first and second conditional moments as a function of the scale
% separation. For this purpose, a scale and the number of a bin (value of the increment) condition
% must be specified. In addition, a linear extrapolation in $\Delta r$ (solid black line) of the
% first and second order conditional moments is plotted (see Chapter: Estimation of Kramers-Moyal
% coefficients).
%
% Arguments IN
% scal = Plot conditional moment number of Scale
% bin_num = Number of a bin (value of the velocity increment u_r)
% evaluated = struct array containing all the information about conditional moments for each scale and for each ↙
bin
% step_con_moment = the steps at which the value of conditional moments are calculated
% markov = markov length in number of samples
% multi_point = Multipoint condition 1=YES or 2=NO
% increment_bin = number of bins
% condition = This input is for multipoint statistics
% tol = This input is for multipoint statistics
% data_filter = filtered data
% save_path = path for saving figures and files
% save_name = name for saving files
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [evaluated] = KM_Calculation(increment_bin,min_events,evaluated,step_con_moment,Fs,taylor_L,m_data,↙
multi_point,condition,norm_r);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This function calculates the Kramers-Moyal coefficients $D^{(k)}\left(u_r,r\right)$ with $k={1,2,3,4}$
% for all scales (specified in \textbf{scale\_steps}) and for each bin (specified in \textbf{increment\_bin})↙
for
% all values of longitudinal velocity increments by a linear extrapolation in $\Delta r$ of the
% $k$-order conditional moments $M^{k}\left(u_r,r\right)$ (see Fig. \ref{fig:con_mom}) and the
% function \textbf{KM\_plot\_raw} plots them accordingly. With $r'<r$:
%
% This limit approximation leads to uncertainties in the absolute values of the Kramers-Moyal
% coefficients, whereas the functional forms of $D^{(k)}\left(u_r,r\right)$ are commonly well estimated.
% In order to overcome this problem, the optimization algorithm described below is performed.
%
% Estimation of Kramers-Moyal coefficients (D1 and D2) for each scale and for each bin are given
% by derivatives of the corresponding conditional moments % For each bin, a linear fit is computed
% for all steps in steps.
%
% Arguments IN
% increment_bin = number of bins
% min_events = minimum number of events
% evaluated = struct array calculated in the function 'conditional_moment'
% step_con_moment = the steps at which the value of conditional moments are calculated
% Fs = Acquisition/Sampling Frequency in Hz
% taylor_L = Taylor length scale in meters
% m_data = mean of the data
% multi_point = Multipoint condition 1=YES or 2=NO
% condition = This input is for multipoint statistics
% norm_r = normalization of the scale using $\lambda$? data 1=Yes, 0=No
%
% Arguments OUT
% evaluated = a modified/updated struct 'evaluated' array
% Enclosed in a "evaluated":
% r is the scale in meters at which moments will be calculated and hence this r will
% be the same at which D1 & D2 will be calculated===>r2
% r_samp is nothing but the r in number of samples==>r2
% r_short_sample is nothing but r1 ===> (r2>r1)
% D1 = Drift coefficient
% eD1 = error associated with drift coefficient
% D2 = Diffusion coefficient
% eD2 = error associated with diffusion coefficient
% D3 = Third order Kramers-Moyal coefficient
% D4 = Fourth order Kramers-Moyal coefficient
% D1_opti = Optimised D1
% D2_opti = Optimised D2
% M11 = First order conditional moment
% M21 = Second order conditional moment
% M31 = Third order conditional moment
% M41 = Fourth order conditional moment
% eM1 = Error associated with M11
% eM2 = Error associated with M21
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function KM_plot_raw(evaluated,Fs,taylor_L,multi_point,condition,norm_ur,norm_r,save_path,save_name)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This function plots the non-optimized $D^{(1,2,4)}\left(u_r,r\right)$ with respect to scale $r$
% and velocity increment $u_r$.
%
% Arguments IN
% evaluated = evaluated struct array from the function 'KM_Calculation'
% Fs = Acquisition/Sampling Frequency in Hz
% taylor_L = Taylor length scale in meters
```

```
% multi_point = Multipoint condition 1=YES or 2=NO
% condition = This input is for multipoint statistics
% norm_ur = normalization of the data using $\sigma_\infty$? data 1=Yes, 0=No
% norm_r = normalization of the scale using $\lambda$? data 1=Yes, 0=No
% save_path = path for saving figures and files
% save_name = name for saving files
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function  [evaluated] = KM_STP_optimization(evaluated,increment_bin,Fs,markov,data_filter,m_data,taylor_L,↙
test_opti,multi_point,condition,tol,min_events,tol_opti,norm_ur,norm_r,save_path,save_name);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This function performs the pointwise optimization of Kramers-Moyal coefficients
% $D^{(1,2)}\left(u_r,r\right)$ at each scale and value of velocity increment to minimize possible
% uncertainties in the absolute values of the Kramers-Moyal coefficients. The object of this
% optimization is to find the best Fokker-Planck equation to reproduce the conditional PDF's as
% these are the essential part of the Markov process. This optimization procedure is proposed in
% \cite{kleinhans2005iterative,Nawroth2007,Reinke2018} and it includes the reconstruction of the
% conditional probability density functions $p\left(u_{r'} | u_r \right)$ via the short time
% propagator \cite{Risken}
%
% Arguments IN
% evaluated = struct array calculated in the function 'conditional_moment'
% increment_bin = number of bins
% Fs = Acquisition/Sampling Frequency in Hz
% markov = markov length in number of samples
% data_filter = filtered data
% m_data = mean of the data
% taylor_L = Taylor length scale in meters
% test_opti = weather to plot or not ==> 'Plot? 1=Yes, 0=No'
% multi_point = weather to do multi-point analysis or not 1=Yes, 0=No
% condition = condition for multi-point analysis
% tol = This input is for multipoint statistics
% min_events = minimum number of events
% tol_opti = Optimization: Tolerance of the range of Kramers-Moyal coefficients in %
%            It is the percentage(For Ex: 0.1 for 10 percent or 0.2 for 20 percent)of D1 or D2
%            within these limit which you want to optimize these coeffcients D1 & D2
% norm_ur = normalization of the data using $\sigma_\infty$? data 1=Yes, 0=No
% norm_r = normalization of the scale using $\lambda$? data 1=Yes, 0=No
% save_path = path for saving figures and files
% save_name = name for saving files
%
% Arguments OUT
% evaluated = a modified/updated struct 'evaluated' array with optimized value of D1 & D2
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [co_KM_opti,co_KM_opti_no_offset,co_KM_non_opti,fitresult_D1_conf,fitresult_D2_conf] = FIT_KM↙
(evaluated,increment_bin,taylor_L,Fs,m_data,markov,multi_point,condition,norm_ur,norm_r,save_path,save_name);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This function performs the surface fit with a linear function for $D^{(1)}\left(u_r,r\right)$
% and a parabolic function for $D^{(2)}\left(u_r,r\right)$ to the optimized and non-optimized KMC's.
% Coefficients $d_{ij}(r)$ in the fits are functions of scale~$r$ of the form
% $\alpha (r/\lambda)^{\beta}+\gamma$. After fitting, this function plots the optimized $D^{(1,2)}$
% and its surface fits. This function also plots the parameters $d_{11}$, $d_{20}$, $d_{21}$ and
% $d_{22}$ as a function of $\frac{r}{\lambda}$ for optimized and non-optimized
% $D^{(1,2)}\left(u_r,r\right)$.
%
% Arguments IN
% evaluated = struct array calculated in the function 'conditional_moment'
% increment_bin = number of bins
% taylor_L = Taylor length scale in meters
% Fs = Acquisition/Sampling Frequency in Hz
% m_data = mean of the data
% markov = markov length in number of samples
% multi_point = weather to do multi-point analysis or not 1=Yes, 0=No
% condition = condition for multi-point analysis
% norm_ur = normalization of the data using $\sigma_\infty$? data 1=Yes, 0=No
% norm_r = normalization of the scale using $\lambda$? data 1=Yes, 0=No
% save_path = path for saving figures and files
% save_name = name for saving files
%
% Arguments OUT
% co_KM_opti = Coefficients $d_{ij}(r)$ of the optimized Kramers-Moyal coefficients using the surface fits
% co_KM_opti_no_offset = Coefficients $d_{ij}(r)$ of the optimized Kramers-Moyal coefficients using the
% surface fits without an offset
% co_KM_non_opti = Coefficients $d_{ij}(r)$ of the non-optimized Kramers-Moyal coefficients using the surface↙
fits
% fitresult_D1_conf = Confidence intervals for fit coefficients of D1
% fitresult_D2_conf = Confidence intervals for fit coefficients of D2
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function KM_plot(co_KM_opti,evaluated,Fs,taylor_L,int_L,multi_point,condition,norm_ur,norm_r,save_path,↙
save_name)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This function plots the optimized $D^{(1)}\left(u_r,r\right)$ and $D^{(2)}\left(u_r,r\right)$
% and the surface fit.
%
% Arguments IN
% co_KM_opti = Coefficients $d_{ij}(r)$ of the optimized Kramers-Moyal coefficients using the surface fits
% with a linear function for $D^{(1)}\left(u_r,r\right)$ and a parabolic function for $D^{(2)}\left(u_r,
r\right)$
% evaluated = evaluated struct array from the function 'KM_Calculation'
% Fs = Acquisition/Sampling Frequency in Hz
% taylor_L = Taylor length scale in meters
% int_L = Integral length scale in meters
% multi_point = Multipoint condition 1=YES or 2=NO
% condition = This input is for multipoint statistics
% norm_ur = normalization of the data using $\sigma_\infty$? data 1=Yes, 0=No
% norm_r = normalization of the scale using $\lambda$? data 1=Yes, 0=No
% save_path = path for saving figures and files
% save_name = name for saving files
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function CO_plot(0,co_KM_opti,co_KM_non_opti,evaluated,taylor_L,int_L,norm_ur,norm_r,siginf);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This function plots the parameters $d_{11}$, $d_{20}$, $d_{21}$ and $d_{22}$ as a function of $r$
% for optimized and non-optimized $D^{(1)}$ and $D^{(2)}$.
% Arguments IN
% co_KM_opti = Coefficients $d_{ij}(r)$ of the optimized Kramers-Moyal coefficients or
% co_KM_non_opti non optimized Kramers-Moyal coefficients
% using the surface fits with a linear function for $D^{(1)}\left(u_r,r\right)$ and a parabolic function for $D^
{(2)}\left(u_r,r\right)$
% evaluated = evaluated struct array from the function 'KM_Calculation'
% taylor_L = Taylor length scale in meters
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [Sm,Ds,DS,r,ind_trajec,rind,dr,r_s,~,~,A,~,~,~,~,~,~] = checkFT(evaluated,int_L,taylor_L,Fs,
data_filter,m_data,z,co_KM_opti,data_length,markov,trajec,dr_ind,norm_ur,norm_r);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% The calculation leading towards the integral fluctuation theorem will be done. In the spirit of
% non-equilibrium stochastic thermodynamics \cite{seifert2012stochastic} it is possible to associate
% with every individual cascade trajectory $\left[u(\cdot) \right]$ a total entropy variation
% $\Delta S_{tot}$. In this investigation it is assumed that a single cascade trajectory represents
% one realization of the turbulent cascade process and a large number of these trajectories reflect
% the statistics caused by the process.
% The set of measured cascade trajectories results in a set of total entropy variation values
% $\Delta S_{tot}$. This function calculates the system entropy, medium entropy and the total
% entropy variation for all the independent cascade trajectories.
%
% Arguments IN
% evaluated = a modified/updated struct 'evaluated' array with in function 'KM_STP_optimization'
% int_L = Integral length scale in meters
% taylor_L = Taylor length scale in meters
% Fs = Acquisition/Sampling Frequency in Hz
% data_filter = filtered data
% m_data = mean of the data
% z = This is the parameter which decides how to compute entropy using different methods using either
% overlapping or independent trajectories
% z = 1 ==> Overlapping trajectories
% z = 3 ==> Independent trajectories
% co_KM_opti = Coefficients $d_{ij}(r)$ of the optimized Kramers-Moyal coefficients, where first entry is
%      zeroth power and power laws for r-dependency with exponents co.ae and co.be, for each power
%      in D1 and D2 respectively
% data_length = A value between 0 & 1 ==> how much of data you want to consider for this calculation
% data_length = 1 for all the data i.e. data(1:end)
% data_length = 0.5 for the data i.e data(1:end/2)   <== For 50% of the data
% markov = markov length in number of samples
% trajec = 1 ==> The start/end of the cascade trajectory will be adjusted.
% dr_ind = Separation of scales/step increment (in samples) referred to the sequence from large to small scales
in the cascade trajectory
% norm_ur = normalization of the data using $\sigma_\infty$? data 1=Yes, 0=No
% norm_r = normalization of the scale using $\lambda$? data 1=Yes, 0=No
%
% Arguments OUT
% Sm = Entropy of the medium
% Ds = Entropy of the system or the Shanon entropy
% DS = Total entropy production=Sm+Ds
% r =  Normalized scale vector from the start to end of the cascade trajectory
% ind_trajec = index-vector of the trajectories (important vector for the recalculation of the trajectories from
data)
% rind = index-vector from the start to end of the cascade trajectory
% dr = separation of scales/step increment (in normalized scale) referred to the sequence from
%      large to small scales in the cascade trajectory
% r_s = mid-point scale vector (Stratonovich convention)
% u = used trajectoires (only z=3 ==> Independent trajectories)
% A = action functional, pathintegral of Lagrangian
% Lag,p,H,tmpvar,H1,H2,ur_s_tmp (are not yet included/used in the current version)
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function plot_entropy(Sm,Ds,DS,increment_bin,save_path,save_name)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This function plots the empirical average $\langle e^{\mathrm{-}\Delta S_{tot}} \rangle_N$ of
% $\Delta S_{tot}$ as a function of the number, $N$ (sample size), of cascade trajectories
% $\left[u(\cdot) \right]$ with errorbars. In addtion, the probability density function of the system,
% medium and total entropy will be plotted while displaying the value of $\langle\Delta S_{tot}\rangle$
% which should be $\geq$ 0. The  integral fluctuation theorem (IFT) expresses the integral balance
% between the entropy-consuming ($\Delta S_{tot}<0$) and the entropy-producing ($\Delta S_{tot}>0$)
% cascade trajectories
%
% Arguments IN
% Sm = Entropy of the medium
% Ds = Entropy of the system or the Shanon entropy
% DS = Total entropy production=Sm+Ds
% increment_bin = number of bins
% save_path = path for saving figures and files
% save_name = name for saving files
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [co_IFT_opti,history]=OPTI_IFT_dij(tol_D1,tol_D2,evaluated,co_KM_opti,fitresult_D1_conf,↙
fitresult_D2_conf,int_L,taylor_L,m_data,Fs,data_filter,3,iter,markov,trajec,dr_ind,norm_ur,norm_r,save_path,↙
save_name);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This function performs the pointwise optimization of Kramers-Moyal coefficients towards the
% integral fluctuation theorem will be done. Thereby the separation of scales/step increment
% (in samples) referred to the sequence from large to small scales in the cascade trajectory is set
% to a minimum of 1 sample and for the optimization, independent cascade trajectories (z=3) are used.
% Note, we use here a separation that is less than or equal to the Einstein-Markov length of $\Delta_{EM}$.
%
% This function performs the optimization of $D^{(1,2)}\left(u_r,r\right)$ at each scale and at each
% value of velocity increment in order to satisfy the integral fluctuation theorem with minimum
% possible error and plots the optimized $d_{ij}$ as a function of $r$. The optimization procedure
% systematically changes $D^{(1,2)}\left(u_r,r\right)$ until the error function
% \begin{eqnarray}
% \xi =|1-\langle e^{\mathrm{-}\Delta S_{tot}} \rangle_{max(N)}|
% \end{eqnarray}
% is minimized.
%
% Arguments IN
% tol_D1          = Tolerance of the range of Kramers-Moyal coefficients in %% for the Optimization
% tol_D2          = tol_D1;
% evaluated = a modified/updated struct 'evaluated' array with in function 'KM_STP_optimization'
% co_KM_opti = Coefficients $d_{ij}(r)$ of the optimized Kramers-Moyal coefficients towards the best
%              Fokker-Planck equation to reproduce the conditional PDF's using the surface fits
% fitresult_D1_conf = Confidence intervals for fit coefficients of D1 (co_KM_opti)
% fitresult_D2_conf = Confidence intervals for fit coefficients of D2 (co_KM_opti)
% int_L = Integral length scale in meters
% taylor_L = Taylor length scale in meters
% m_data = mean of the data
% Fs = Acquisition/Sampling Frequency in Hz
% data = filtered data
% z = This is the parameter which decides how to compute entropy using different methods using either
% overlapping or independent trajectories
% z = 1 ==> Overlapping trajectories
% z = 3 ==> Independent trajectories
% iter = The maximum number of iteration used for the optimization algorithms
% markov = markov length in number of samples
% trajec = 1 ==> The start/end of the cascade trajectory will be adjusted.
% dr_ind = Separation of scales/step increment (in samples) referred to the sequence from large to small scales↙
in the cascade trajectory
% norm_ur = normalization of the data using $\sigma_\infty$? data 1=Yes, 0=No
% norm_r = normalization of the scale using $\lambda$? data 1=Yes, 0=No
% save_path = path for saving figures and files
% save_name = name for saving files
%
% Arguments OUT
% co_IFT_opti = Coefficients $d_{ij}(r)$ of the optimized Kramers-Moyal coefficients towards the
%               integral fluctuation theorem using the surface fits
% history = details of the optimization for every single iteration
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```