POLITECNICO DI MILANO

School of Industrial and Information Engineering
Department of Energy
Master of Science in Nuclear Engineering

POLITECNICO
MILANO 1863

Development and first applications of a statistical analysis toolbox for edge magnetic turbulence studies in the TCV tokamak

Supervisors:
Prof. Matteo Passoni,
Politecnico di Milano, Micro and Nanostructured Materials Lab
Prof. Duccio Testa,
École Polytechnique Fédérale de Lausanne, Swiss Plasma Center

Author:
Alessandro Tolio
837641

Academic Year 2016-2017

A mio padre e mia madre,
che non potrò mai ringraziare abbastanza
per avermi sempre sostenuto.

Contents

List of Figures

List of Tables

Abstract

The path towards economically viable controlled thermonuclear fusion is very long, even after decades of cutting edge research. Fusion power might not be deployed in time to contribute to the solution of the ongoing climate change crisis. Nevertheless, a clean, almost limitless and non-intermittent energy source will be a revolutionary achievement for human civilization.

Multiple reactor designs have been proposed in the past seven decades. Among these, the tokamak is the most studied one because of its superior performance. Its successful development allowed the achievement of stable operation at near-reactor conditions. This means that tokamaks have almost met all requirements for burning plasmas sustain. ITER is the next iteration of this technology. Its aim is to demonstrate the viability of tokamak reactors by achieving the physics power gain $Q = 10$. In order to do so, it is designed to be almost an order of magnitude larger than the largest currently operating tokamak in the world, JET. ITER's first plasma operation is planned for 2025.

One of the current major issues regards transport phenomena. Neoclassical transport theory underestimates experimental parameters by one order of magnitude. This enhanced behavior is due to anomalous transport, which is turbulence driven. To deal with this discrepancy, empirical scaling laws are used to obtain new parameters from the available databases. The validity of the procedure is weakened as it is performed very far away from the available data. This is a problem for DEMO's design, the next iteration after ITER.

To improve transport models, a better understanding of turbulence is needed. The aim of this work is the development of a statistical analysis toolbox for the study of edge magnetic field turbulence inside TCV. This device is the best choice to perform this study, since its fast magnetic acquisition system was recently upgraded to handle more than 200 signals. The code is validated with real experimental data. The main results of this work are: a different scaling behavior from theoretical references, the detection of dissipation during the energy cascade, and the fact that turbulence follows fBm functions.

Sommario

Nonostante più di mezzo secolo di ricerca, il traguardo della fusione termonucleare controllata ed economicamente sfruttabile è ancora lontano. L'energia da fusione nucleare, infatti, potrebbe non essere parte della soluzione all'attuale crisi climatica. Ciononostante, ottenere accesso a una fonte energetica pulita, quasi inesauribile e non intermittente sarebbe una rivoluzione per il genere umano.

Molte tipologie di reattore a fusione sono state proposte nel tempo. Fra questi, il tokamak è quello più studiato, grazie alle sue migliori prestazioni. Infatti, condizioni di operatività a livello reattore sono raggiunte con questa configurazione. ITER, progetto che rappresenta la prossima iterazione di questa tecnologia, dovrà dimostrare la fattibilità di utilizzo di un tokamak come reattore, producendo il guadagno fisico $Q = 10$. Per soddisfare questo obiettivo ITER è dieci volte la dimensione del più grande tokamak ora in funzione, JET. Il primo plasma di ITER è pianificato per il 2025.

Uno dei problemi dei tokamak riguarda il trasporto nel plasma. La teoria del trasporto neoclassico sottostima di un ordine di grandezza i coefficienti di trasporto del plasma. Ciò è dovuto al cosiddetto trasporto anomalo, generato dalla turbolenza. A causa di ciò, per ottenere nuovi parametri si utilizzano delle leggi semi-empiriche. La validità dell'estrapolazione è indebolita quando è effettuata a diversi ordini di grandezza dai dati a disposizione. Questo è il caso della progettazione di DEMO, il successore di ITER.

Una conoscenza più profonda dei fenomeni turbolenti è necessaria per migliorare gli attuali modelli di trasporto. Lo scopo di questa tesi è lo sviluppo di un codice in grado di effettuare un'analisi statistica del campo magnetico di bordo in TCV. Vista la recente installazione di oltre 200 sonde magnetiche dentro tale macchina, TCV è la migliore scelta per effettuare questo tipo di analisi. Il codice è validato con dati sperimentali reali e i principali risultati ottenuti sono: osservazione di una legge di scala differente da quelle di riferimento, presenza di dissipazione durante la cascata energetica e descrizione dei segnali acquisiti come fBm.

Estratto

In questo lavoro di tesi viene affrontato uno studio sperimentale della statistica della turbolenza magnetica di bordo in un plasma in configurazione tokamak. In particolare, lo scopo è quello di sviluppare un pacchetto software in grado di effettuare tale tipo di studio in maniera semi-automatizzata e affidabile. Il codice sviluppato servirà come base per lo sviluppo di algoritmi più avanzati, in grado di caratterizzare statisticamente la turbolenza magnetica di bordo all'interno del *Tokamak à Configuration Variable*, il tokamak dello *Swiss Plasma Center*, laboratorio ospitato dall'*École Polytechnique Fédérale de Lausanne* in Svizzera.

Il problema del cambiamento climatico necessita lo sviluppo di soluzioni energetiche innovative, in modo da creare un vantaggio economico diretto nell'investire su energie pulite rispetto a energie da fonti fossili. La transizione verso fonti energetiche pulite accelererà drammaticamente solo quando queste saranno più economiche di quelle tradizionali. Nonostante ciò, un'economia di mercato avanzata non potrà essere totalmente supportata da fonti rinnovabili intermittenti. Con questa qualifica si intendono quelle fonti che non sono in grado di fornire alla rete un livello di potenza constante lungo intervalli temporali giornalieri e stagionali; tra di esse si annoverano l'energia solare e quella eolica.

L'energia nucleare è in grado di fornire un profilo di carico di base affidabile senza produrre gas serra. Sopra questo profilo si potrebbero aggiungere quelli forniti dalle rinnovabili, così da ottenere un generazione elettrica totalmente pulita. Anche se il nucleare non gode, generalmente, del favore dell'opinione pubblica, i progetti della cosiddetta Generazione IV si profilano come soluzioni nucleari innovative, in grado di sopperire ad alcuni dei punti deboli dei reattori ad acqua tradizionali.

La soluzione dell'energia da fusione termonucleare controllata è auspicabile, ma ancora lontana. Infatti, il fronte della ricerca sulla fusione è rappresentato da ITER, un reattore sperimentale attualmente in costruzione nel sud della Francia. L'obiettivo di tale progetto è quello di dimostrare che la tipologia di reattore tokamak è in grado di produrre un guadagno fisico $Q = 10$, cioè che la macchina è in grado di produrre una potenza termica netta dieci volte superiore a quella fornita al sistema. Il primo plasma di ITER è pianificato per il 2025, ma questo progetto non rappresenta ancora lo sviluppo di un vero e proprio impianto di potenza basato su energia da fusione termonucleare. Questo obiettivo dovrà essere raggiunto da progetti successivi a ITER, come potrebbe essere DEMO, di ideazione europea. Le prime operazioni di tale progetto, comunque, non sono previste prima del 2050. In questo senso l'energia da fusione potrebbe non essere in grado di far parte di un buon mix energetico strategico per la risoluzione del problema del riscaldamento globale. Ciononostante, lo sviluppo di una fonte energetica pulita, quasi inesauribile

e non intermittente sarà potenzialmente una rivoluzione per il genere umano.

La reazione di fusione termonucleare avviene quando due nuclei leggeri si urtano con energia cinetica sufficiente a superare la barriera coulombiana che è presente tra di loro. Quando ciò avviene, la reazione dà vita a uno o più nuclei la cui massa totale è in difetto rispetto a quella di partenza. Questo difetto è l'origine del rilascio di energia, sotto forma di energia cinetica dei prodotti di reazione. Questa energia può essere parzialmente trasformata in energia elettrica attraverso un ciclo termodinamico, o direttamente nel caso di alcune reazioni. La reazione di maggiore interesse scientifico è quella tra due isotopi dell'idrogeno: deuterio (D) e trizio (T). Questo perché è quella che offre la più alta probabilità di avvenire a temperature considerate accessibili. Infatti, per poter superare la barriera coulombiana la temperatura del sistema deve essere molto alta, dell'ordine di $T \sim 10\,\mathrm{keV}$. A queste temperature la materia sopravvive solo allo stato di plasma. Dato che non esistono materiali che possano sopravvivere a tali condizioni, un plasma termonucleare deve essere confinato. Due metodi di confinamento sono possibili: magnetico e inerziale. Entrambi prevedono l'utilizzo di una pressione esterna al plasma, infatti nel primo caso questa è di origine magnetica, mentre nel secondo è di origine cinetica.

Un tokamak è una macchina di forma toroidale in grado di confinare magneticamente un plasma. Le caratteristiche principali di un tokamak sono la presenza di un intenso campo magnetico e di una corrente, entrambi toroidali. Tale tipologia di macchina è storicamente quella più studiata in quanto è quella che permette di raggiungere le più elevate prestazioni. Per un plasma a confinamento magnetico le prestazioni sono misurate dal prodotto tra pressione cinetica e tempo di confinamento dell'energia, $p\tau_\mathrm{E}$. La condizione di ignizione, cioè di auto-sostentamento della temperatura del plasma, è data in termini del minimo prodotto $p\tau_\mathrm{E}$ al variare della temperatura del sistema.

τ_E è un parametro sperimentale utilizzato per evitare il problema di dover trovare una descrizione analitica dei flussi di calore alla superficie esterna del plasma. Infatti, le teorie di trasporto classica e neoclassica, la quale tiene conto degli effetti della geometria toroidale del sistema, sottostimano da uno a tre ordini di grandezza i coefficienti di trasporto delle tre grandezze considerate: densità di particelle, pressione cinetica, campo magnetico. Ciò è dovuto al cosiddetto trasporto anomalo di origine turbolenta. Infatti, la turbolenza ha la caratteristica di aumentare la diffusività del sistema. Per calcolare il tempo di confinamento dell'energia di un nuovo reattore si procedere con l'utilizzo di una formula semi-empirica basata sui dati raccolti da esperimenti attuali. Questa estrapolazione è valida nell'intorno dei dati presenti nei database, ma è indebolita a ordini di grandezza di distanza. Questo problema è tipico della progettazione di macchine tipo DEMO, la cui dimensioni saranno due ordini di grandezza superiori a quelle di JET, il tokamak più grande attualmente in funzione al mondo.

Per poter migliorare i modelli di trasporto attuali è necessario approfondire le conoscenza sulla turbolenza del plasma. Generalmente gli studi sui flussi turbolenti sono di tipo statistico, infatti questa è la strategia seguita anche per questo lavoro. Molti strumenti statistici posso essere utilizzati per l'analisi della turbolenza, per questo lavoro i seguenti sono stati scelti: calcolo del tempo di de-correlazione, calcolo dello spettro di densità di potenza, calcolo della funzione di densità di probabilità degli incrementi temporali, calcolo delle funzioni di struttura temporale, calcolo

degli esponenti di *Hurst* e calcolo della caratteristiche di entropia e complessità delle permutazioni del segnale.

Il segnale analizzato è l'intensità del campo magnetico turbolento di bordo del plasma dentro TCV. Questo segnale è acquisito grazie a un sistema di oltre 200 solenoidi ad alta frequenza installati sulle pareti interne della camera di vuoto di TCV. I solenoidi sono orientati in modo da acquisire la componente poloidale del segnale, ritenuta quella di maggior intensità fuori dal plasma. Le sonde sono organizzate in gruppi, chiamati array, in modo tale da poter acquisire il segnale secondo schemi toroidali e poloidali, o entrambi contemporaneamente.

Il primo strumento prevede il calcolo della funzione di auto-correlazione dei segnali analizzati e il successivo calcolo del tempo di de-correlazione. Questo parametro indica dopo quanto tempo il segnale "perde memoria di sé stesso" e indica una scala temporale importante per i fenomeni di turbolenza.

Lo spettro di densità di potenza indica il contenuto energetico delle varie armoniche che compongono il segnale. Un andamento lineare decrescente dello spettro indica la presenza di un intervallo inerziale della turbolenza, durante il quale l'energia delle strutture turbolente è trasmessa a quelle di minore dimensione. Al termine di questo intervallo iniziano i fenomeni di dissipazione della turbolenza. La pendenza dell'intervallo inerziale è prevista teoricamente da alcuni modelli, i quali sono presi come riferimento per questo lavoro: teoria *K41*, teoria *IK* ed esponente di *Hall*.

Per garantire che il segnale non abbia un contenuto puramente casule, sia cioè rumore, il calcolo della funzione di densità di probabilità degli incrementi temporali è effettuato. Grazie a questa analisi la distribuzione degli incrementi temporali del segnale è confrontata qualitativamente e quantitativamente con quella del segnale rumore.

La funzione di struttura temporale è utile per indicare la presenza di dissipazioni durante il supposto intervallo inerziale. Inoltre è utilizzata per il calcolo dell'esponente di *Hurst* del segnale. Questo parametro è importante perché qualifica la correlazione a lunghi tempi del segnale. Un segnale, infatti, può essere persistente, anti-correlato, stazionario o ad incrementi indipendenti.

L'ultima analisi permette di disegnare il grafico CH del segnale. Grazie a questo grafico è possibile comparare le caratteristiche di entropia e complessità delle possibili permutazioni degli schemi d'ordine del segnale. Questi analizzano le posizioni relative tra i punti di un segnale. Questa analisi è molto importante perché permette di associare una funzione matematica a un segnale nel caso le due avessero le stesse caratteristiche di entropia e complessità.

Il codice è stato testato effettuando l'analisi su 27 esperimenti effettuati in TCV. Questi sono stati scelti da due missioni sperimentali i cui obiettivi erano molto diversi, ma che hanno dato la possibilità a questo codice di essere provato su plasmi con caratteristiche differenti, come diversi modi di confinamento e triangolarità.

I risultati delle analisi su singoli segnali sono molto sparpagliati. Ciò è probabilmente dovuto al basso rapporto segnale-rumore dei segnali stessi. Per sopperire a questo problema la strategia delle medie globali è stata adottata. Questa prevede il calcolo delle medie dei risultati su tutte le sonde utilizzate, andando ad escludere eventuali valori considerati anomali.

I risultati principali delle analisi sono tre. Il primo riguarda lo spettro di densità

di potenza del segnale. È stato trovato che nessuno dei segnali sembra seguire chiaramente uno degli esponenti prescritti dalle teorie sulla turbolenza nell'intervallo di frequenze considerato. Il secondo è che sembrano essere presenti delle iniezioni (o dissipazioni) di energia appena prima della cascata inerziale, dovute a strutture di cui non è stata studiata la natura. Il risultato più sorprendente, però, è ricavato dai grafici CH. Grazie a questa analisi, infatti, è stato osservato che i segnali analizzati seguono le proprietà statistiche delle funzioni fBm. Questo significa che i segnali possono essere descritti matematicamente da fBm aventi il corretto esponente di *Hurst*.

Questo lavoro di tesi è stato effettuato durante un periodo di otto mesi presso lo *Swiss Plasma Center*, Svizzera.

Chapter 1

The energy problem and the quest for fusion power

Human civilization is requested to tackle the most difficult issue it has probably ever faced, that is the climate change crisis. The drivers of the global rise in temperatures, which is the cause of climate change, are greenhouse gasses, among which the main contributor is CO_2. The production of this air pollutant is mostly due to electricity production, transportation and human industrial activities.

There are multiple options that need to be considered in order to deal with this problem, but the most effective strategy would have to comprehend the best mix of solutions. Energy efficiency is quite important and, being the easiest option to apply in developed Countries, it has already extensively been explored and implemented.[1] However, efficiency will give a little contribution to the solution, also considering that energy demands are globally increasing, as shown in Fig. 1.1. This is mostly due to developing Countries, where the economy is largely expanding and the population is rapidly rising its living standards. Consider that the Chinese and Indian populations alone make up more than 2 billions people and, taking into account all of Asia and Africa, half of the world's population is expected to gain access to higher living standards in the next decades. Even though poverty decline is a good achievement from a social point of view, the problem of supplying energy to these power-hungry populations remains. To sustain this growth the Governments of such Countries are seeking cheap means of energy production, that are heavily relying on fossil fuels; see Fig. 1.2. Subsidies to renewable energies are a widespread solution in the developed Countries, and these helped increase the adoption rate of cleaner energy sources. However, this approach may not be feasible in those Nations that are now leaving poverty, since huge debts are related to subsidies. As a matter of fact, this solution is debatable even in the case of rich Countries since relevant economic resources are given to subsidize the deployment of expensive solutions, whereas these might have been spent on research.

In fact, the largest contribution to the solution needs to come form innovation. Developing new energy sources, or refining the clean existing ones, would play a huge role in shaping worldwide energy policies. The adoption of clean energy production technologies would be faster if the related generation costs were lower

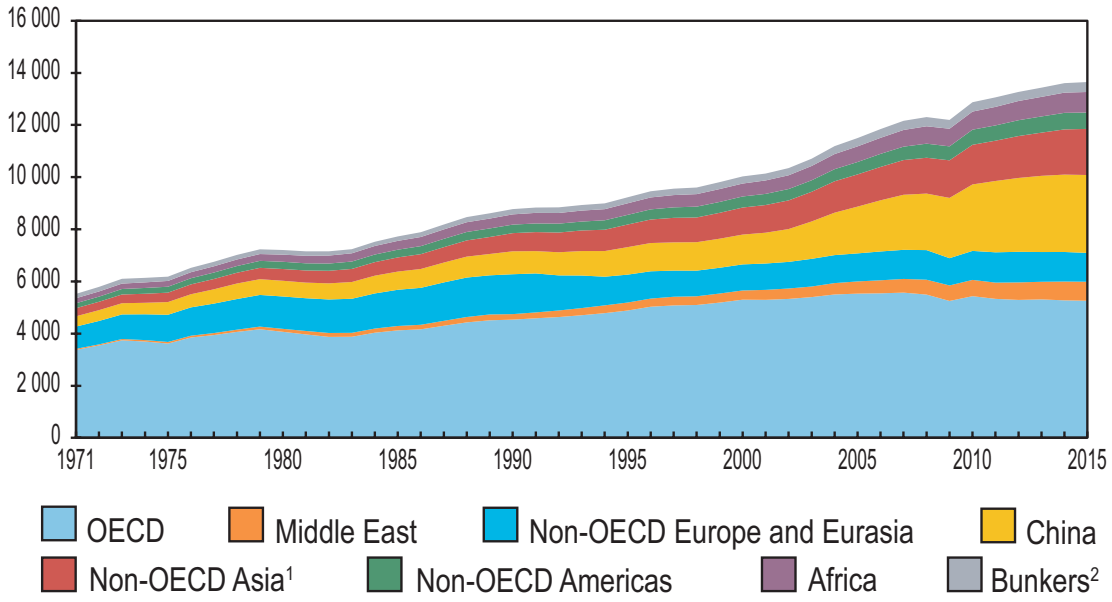

Figure 1.1: World total primary energy supply by region in Mtoe.[(1)]
[1] Does not include China. [2] Includes international aviation and international marine bunkers.

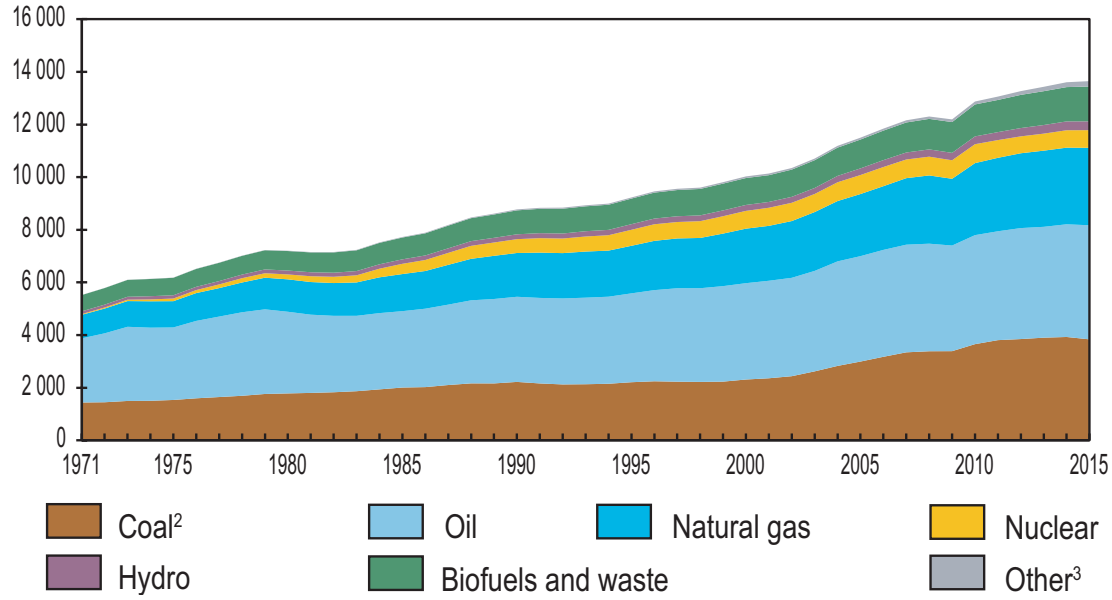

Figure 1.2: World total primary energy supply by fuel in Mtoe, including international aviation and international marine bunkers.[(1)]
[2] Peat and oil shale are aggregated with coal. [3] Includes geothermal, solar, wind, tide/wave/ocean, heat and other.

than those of fossil fuels sources. In order to do so expenditure for energy research needs to get higher, also considering that the reduction of CO_2 production levels has to take place in the least amount of time possible.[2]

A complete transition to intermittent energy resources is, however, not feasible with current technologies.[3] These types of energy resources are those that are not capable of offering both daily and seasonal reliable power output. Among these, solar and wind energy are two examples. The argument of poor energy storage technologies is usually provided to prove this conclusion.[3] Nevertheless, a maximization of intermittent energy production is foreseeable, but even in this scenario the need of an energy source capable of providing a stable and reliable base load has to be fulfilled. This role can be taken by nuclear power, which is a mean of producing greenhouse-gasses-free base load electricity.

Even though in the near future many innovations are going to be achieved by new nuclear fission technologies, in the frame of the Generation IV projects[4], the commercialization of nuclear fusion reactors is quite appealing, at least from the public opinion's point of view. This is because fusion reactors are inherently safe by design since the nuclear fuel is not stored inside the device, but it is continuously added to the reaction chamber. In case of an accident the fuel supply stops and the reaction dies out on its own. Moreover, as a first approximation, fusion energy does not produce nuclear waste. In fact, since neutrons are produced during the chosen reaction (see Sec. 1.1), the structure of the machine is going to be activated, therefore becoming the only waste to be disposed of. However, plant's decommissioning is also a problem that conventional nuclear facilities have to deal with.

In the course of the second part of the 20^{th} century fusion research has produced important advancements, as it is shown in Fig. 1.3, but a commercial technology still seems distant into the future, at least to join the solution to the climate change crisis. At the end of an arduous negotiation and design path, in November 2006 the ITER project finally kicked off.[5] On the 21^{st} day of that month the final commitment of China, the European Union, India, Japan, Russia, South Korea and the United States gave start to the construction of the largest fusion experimental reactor that has ever been built. The ITER acronym stays for International Thermonuclear Experimental Reactor and it is also the Latin word for 'the way'. In fact, this device is going to demonstrate the feasibility of the most studied nuclear fusion reactor design, namely the tokamak.

This project represents the forefront of fusion research and its facilities are under construction in southern France.[6] Its first plasma operations are planned to begin in 2025.[6] However, a commercial tokamak-based power plant is expected to be constructed only after ITER will reach an advanced operational status, meaning that such energy resource is still a long way in the future. The European ITER's next iteration is expected to be a power plant demonstrator called DEMO.[7]

Nevertheless, the characteristics of clean, almost inexhaustible and non-intermittent source qualify nuclear fusion energy as a possibly revolutionizing technology for humankind. In the next Section an overview of fusion reactor basics is given.

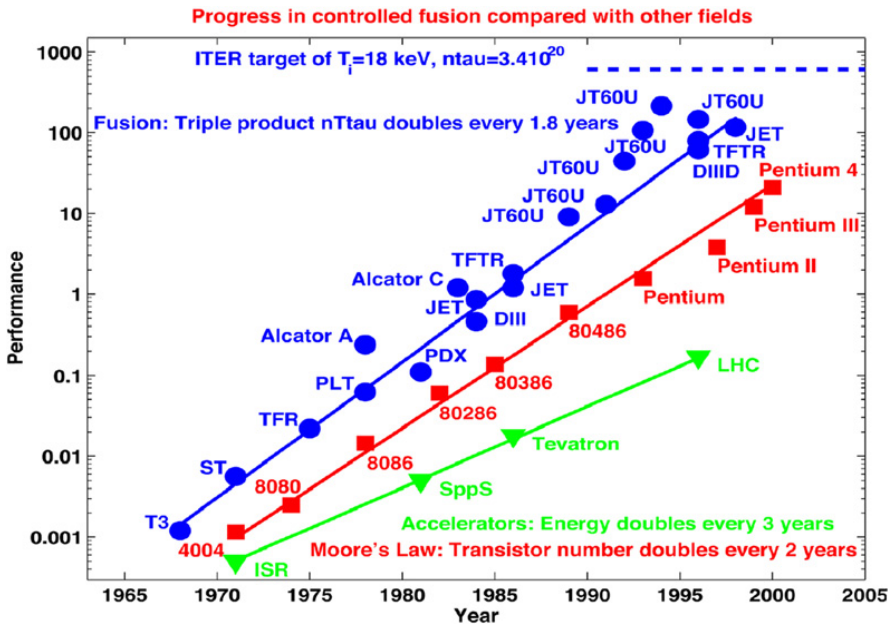

Figure 1.3: Comparison of performances among different technology fields. Notice how fusion performance has been steeper than *Moore*'s law.[5]

1.1 Nuclear fusion reactor basics

The idea of extracting useful energy from a nuclear reaction arises from the study of the plot of binding energy per nucleon as function of A, the nuclear mass number. This is shown in Fig. 1.4, where it is possible to see that the most stable nuclei are those having largest binding energy per nucleon, that is at $A \sim 60$. Any reaction that increases the binding energy of a system is exoenergetic and, therefore, those that are able to move A towards the maximum of the curve will release net energy. At the two extremes of the curve two types of such reactions are possible: nuclear fission and fusion. The former relies on the splitting of a heavy nucleus following the absorption of a neutron. The latter is obtained through the reaction of two light nuclei forming an heavier one. The major drawback of nuclear fusion, with respect to fission, is that the Coulomb repulsive potential has to be overcome since both light nuclei necessarily have the same net charge sign. In order to do so a large amount of kinetic energy is supplied to the nuclei via the increase of internal energy of the system. This translates into increasing the temperature of the reacting medium, hence the name thermonuclear fusion. At the temperature levels needed for the reaction to occur the only surviving state of matter is plasma. This is why plasma physics is strictly involved in the study of nuclear fusion.

The main fusion fuels are deuterium (D or ^{2}H), tritium (T or ^{3}H) and helium.[8] These are involved in the following fusion reactions:

$$\begin{aligned}
\mathrm{D} + \mathrm{D} &\longrightarrow {}^3\mathrm{He} + \mathrm{n} + 3.27\,\mathrm{MeV},\\
\mathrm{D} + \mathrm{D} &\longrightarrow \mathrm{T} + \mathrm{p} + 4.03\,\mathrm{MeV},\\
\mathrm{D} + {}^3\mathrm{He} &\longrightarrow {}^4\mathrm{He} + \mathrm{p} + 18.3\,\mathrm{MeV},\\
\mathrm{D} + \mathrm{T} &\longrightarrow {}^4\mathrm{He} + \mathrm{n} + 17.6\,\mathrm{MeV}.
\end{aligned}$$

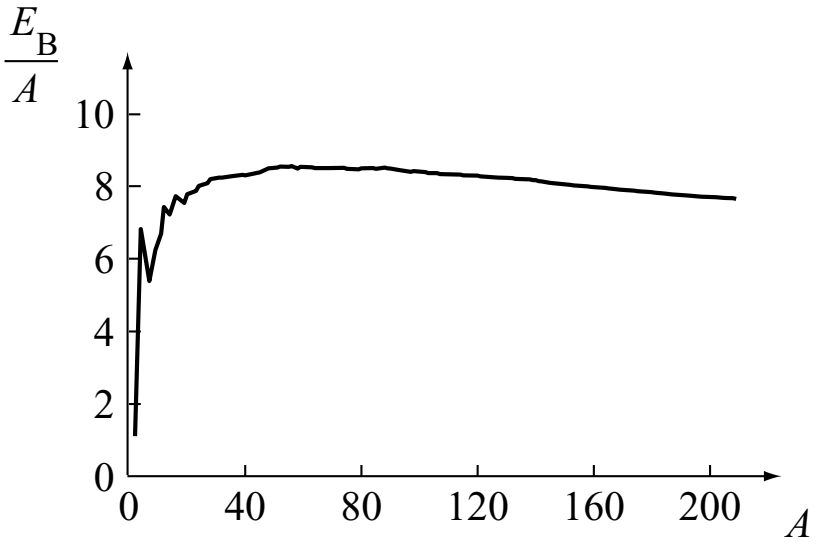

Figure 1.4: Binding energy per nucleus as function of A.[8]

The total fusion power generated by one of these reactions, S_f is function of both the energy released during a single event, E_f, and the reaction rate, R_{12}:

$$S_f = E_f R_{12}, \tag{1.1}$$

where the reaction rate is given by the following:

$$R_{12} = n_1 n_2 \langle \sigma v \rangle . \tag{1.2}$$

In the last equation the two particles densities are the ones of the reacting species, σ is the cross section of the reaction and v is the relative speed between two reagents. The product σv is averaged over the spectrum of possible relative speeds, since the cross section is actually function of v. In Fig. 1.5 the velocity-averaged cross section is shown as function of the system's temperature.

From Fig. 1.5 the interest for the D-T reaction is easily understood as it is the one with the highest $\langle \sigma v \rangle$ at all temperatures. This is, in fact, the reaction chosen for the first iteration of a fusion reactor. Notice that the velocity-averaged cross section peaks at $T \approx 70\,\mathrm{keV}$ for this reaction. The D-T fuel mixture that maximizes S_f is the 50%-50% one, for which the total fusion power output is:

$$S_f = \frac{1}{4} E_f {n_e}^2 \langle \sigma v \rangle , \tag{1.3}$$

where $n_D + n_T = n_e$ is exploited and n_e is the total electrons density of the system. S_f is divided between the neutron and the *alpha* particle, i.e. the ^{4}He nucleus, according to their mass. In fact $E_\alpha = E_f/5$, for which:

$$S_\alpha = \frac{1}{4} E_\alpha {n_e}^2 \langle \sigma v \rangle . \tag{1.4}$$

The steady state zero-dimensional power balance of a plasma in a generic fusion reactor is given by the following:[8]

$$S_\alpha + S_h = S_B + S_\kappa, \tag{1.5}$$

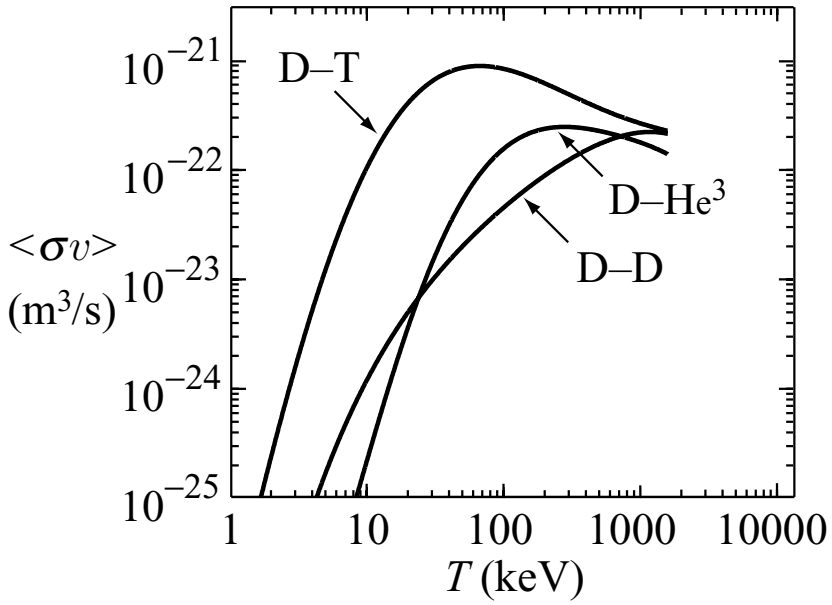

Figure 1.5: Velocity-averaged cross section as function of the system's temperature for the fusion reactions of interest.[8]

where S_{h} is the external heating, S_{B} is the power lost because of Bremsstrahlung, and S_κ is the power loss due to heat fluxes. The first term is the fraction of fusion power generated in the plasma that remains in the system. In fact, fusion neutrons, which carry most of the generated energy, are lost from the plasma because they cannot be magnetically confined. Nevertheless, the energy carried away by the neutrons is the one that can be converted to electrical output.

The external heating is supplied through different means. The first is ohmic heating, which is due to the resistivity of the plasma, that generates heat through the *Joule* effect. The second major contribution to external heating can be supplied through microwave heating, which exploits electromagnetic waves to excite characteristic plasma modes. The last major contribution can be given by a neutrals beam injection system. This involves the injection of highly energetic neutral hydrogen atoms inside the plasma. Thanks to collisions the injected particles loose their electron, increasing the temperature of the ion population.

Bremsstrahlung losses are caused by charged particles that accelerate in an electromagnetic field. This means of energy loss is unavoidable in plasmas, and needs to be taken into account for the power balance in Eq. 1.5.

Heat fluxes give the last contribution to the 0-D power balance. These are difficult to model since the plasma thermal conductivity is complicated to express analytically. S_κ is given by the following:

$$S_\kappa = \frac{1}{V}\int_A \mathbf{q}\cdot d\mathbf{A}, \tag{1.6}$$

where V and A are, respectively, the volume and external surface of the plasma, and $\mathbf{q}$ is the heat flux. In the case of a cylinder, and assuming a constant thermal

conductivity, the following is obtained:

$$\frac{1}{V}\int_A \mathbf{q}\cdot d\mathbf{A} = -2\frac{\kappa}{r}\left.\frac{\partial T}{\partial r}\right|_{r=a}, \tag{1.7}$$

where a is the radius of the cylinder. However, in order to circumvent the problem of transport coefficients, in plasma physics heat fluxes are generally expressed as follows:

$$S_\kappa = \frac{3}{2}\frac{p}{\tau_\mathrm{E}}, \tag{1.8}$$

where p is the volume-averaged plasma kinetic pressure and τ_E is the energy confinement time. The latter is defined as the plasma energy relaxation time due to heat conduction.

Eq. 1.5 for the D-T optimal mixture can be expressed in terms of the volume-averaged p and T, as follows:[8]

$$\begin{gathered} S_\alpha + S_\mathrm{h} = S_\mathrm{B} + S_\kappa \\ \frac{E_\alpha}{16}p^2\frac{\langle\sigma v\rangle}{T^2} + S_\mathrm{h} = \frac{C_\mathrm{B}}{4}\frac{p^2}{T^{3/2}} + \frac{3}{3}\frac{p}{\tau_\mathrm{E}}, \end{gathered} \tag{1.9}$$

where C_B is a constant factor. This simple balance is very useful since it allows to set constraints on p, T and τ_E.

The condition of ideal ignition is achieved when the α heating of the plasma is capable of sustaining radiation losses when no heat fluxes are established. This means that:

$$S_\alpha = S_\mathrm{B}, \tag{1.10}$$

from which a requirement on the minimum plasma temperature is set. For the optimal D-T mixture this is:

$$T \geq 4.4\,\mathrm{keV}.$$

The ignition condition assumes that α heating sustains both radiation losses and heat fluxes, that is:

$$S_\alpha = S_\mathrm{B} + S_\kappa, \tag{1.11}$$

from which a minimum on the $p\tau_\mathrm{E}$ product is set as follows:

$$p\tau_\mathrm{E} \geq \frac{K_\kappa T^2}{K_\alpha\langle\sigma v\rangle - K_\mathrm{B}T^{1/2}} \approx \frac{K_\kappa T^2}{K_\alpha\langle\sigma v\rangle}. \tag{1.12}$$

The last term is the approximation for negligible Bremsstrahlung effects. In Fig. 1.6 both cases are shown. For the 50%-50% D-T mixture the minimum of the curve is at the following coordinates:

$$\begin{aligned} T &= 15\,\mathrm{keV}, \\ p\tau_\mathrm{E} &= 8.3\,\mathrm{atm\ s}. \end{aligned}$$

To take into account external heating the α fraction is introduced and defined by the following:

$$f_\alpha = \frac{S_\alpha}{S_\alpha + S_\mathrm{h}}. \tag{1.13}$$

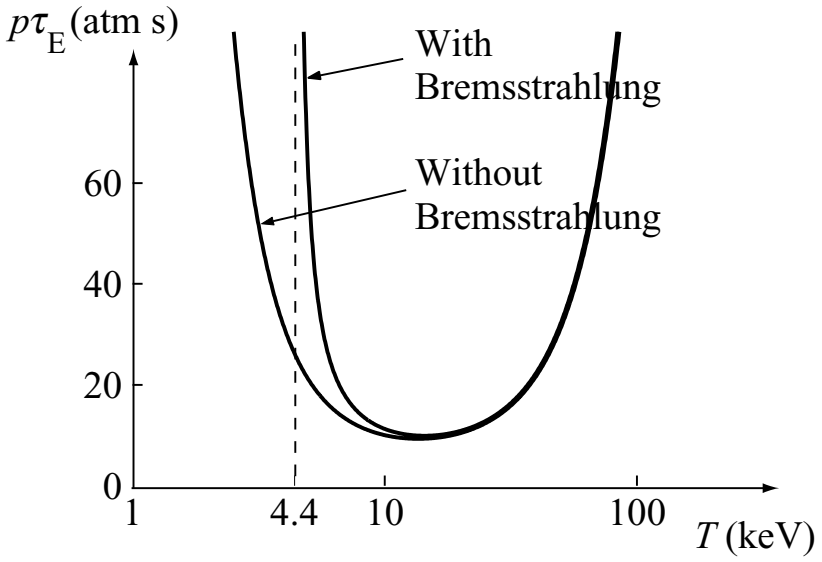

Figure 1.6: Minimum $p\tau_{\mathrm{E}}$ to establish ignition as function of temperature.[8]

Then Eq. 1.12 is modified as follows:

$$p\tau_{\mathrm{E}} \geq \frac{K_\kappa T^2}{\frac{K_\alpha}{f_\alpha}\langle\sigma v\rangle - K_{\mathrm{B}} T^{1/2}} \approx f_\alpha \frac{K_\kappa T^2}{K_\alpha \langle\sigma v\rangle}. \tag{1.14}$$

Once plasma ignition is achieved, it is important to compare how much power the reactor is producing with respect to the supplied one. To do so, two parameters are computed, namely: physic and engineering energy gains. The former is calculated as the ratio of the net thermal power output to the external heating power:

$$Q = \frac{P_{\mathrm{out}} - P_{\mathrm{in}}}{P_{\mathrm{in}}} = \frac{S_{\mathrm{f}}}{S_{\mathrm{h}}}. \tag{1.15}$$

Neglecting Bremsstrahlung the following is obtained[8]:

$$Q = 5\frac{p\tau_{\mathrm{E}}}{(p\tau_{\mathrm{E}})_{\mathrm{I}} - p\tau_{\mathrm{E}}}, \tag{1.16}$$

where $(p\tau_{\mathrm{E}})_{\mathrm{I}}$ is the product required for ignition.

The engineering gain factor is defined as the ratio of the net electric power output to the electric power supply:

$$Q_{\mathrm{E}} = \frac{{P^{(\mathrm{E})}}_{\mathrm{out}} - {P^{(\mathrm{E})}}_{\mathrm{in}}}{{P^{(\mathrm{E})}}_{\mathrm{in}}}, \tag{1.17}$$

which for the optimal D-T mixture is demonstrated to be equal to the following[8]:

$$Q_{\mathrm{E}} = \frac{(6.4\eta_{\mathrm{t}}\eta_{\mathrm{e}}\eta_{\mathrm{a}} + 1 - \eta_{\mathrm{t}}\eta_{\mathrm{e}})\, p\tau_{\mathrm{E}} - (1 - \eta_{\mathrm{t}}\eta_{\mathrm{e}})\,(p\tau_{\mathrm{E}})_{\mathrm{I}}}{(p\tau_{\mathrm{E}})_{\mathrm{I}} - p\tau_{\mathrm{E}}}, \tag{1.18}$$

where η_{t}, η_{e} and η_{a} are the efficiencies of the steam turbine cycle, the heating power generation and heat absorption in the plasma, respectively. In Fig. 1.7 both Q

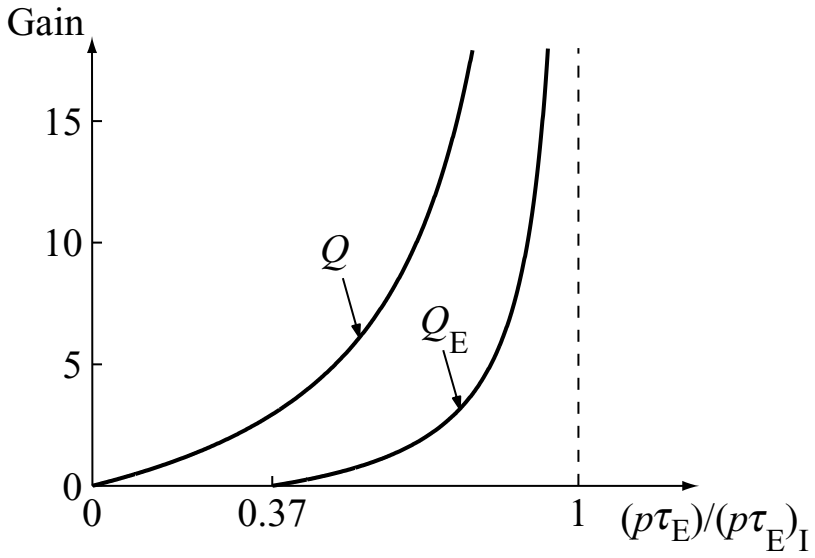

Figure 1.7: Energy gains as function of $(p\tau_E) / (p\tau_E)_I$.[8]

and Q_E are plotted as functions of $(p\tau_E) / (p\tau_E)_I$. Notice that to obtain electric power breakeven, which is $Q_E = 0$, $p\tau_E$ needs to be greater than zero. Moreover, the corresponding value of the physics gain is $Q \approx 2.9$.

ITER is going to achieve $Q = 10$, dealing $Q_E \approx 1.8$ which is quite low. Although the efficiency coefficients can be improved, it will be necessary to increase the value of $p\tau_E$ in order to achieve $Q_E \geq 10$. This goal will be achieved by projects that will follow ITER. As already mentioned, one of such projects is the Demonstration Fusion Power Plant, i.e. DEMO.[7] This project will address the following issues: resolve all power-plant-relevant physics and technological problems, demonstrate production of several hundreds MW of electric power, and achieve closed fuel-cycle operations.[7, 8] In Fig. 1.8 a comparison of size and thermal power output of JET, ITER and DEMO is visualized.

Many problems need to be resolved during the design process of a machine like a tokamak. In the case of DEMO new challenges arise as the dimensions of such a device are unexplored. This means that extrapolation of relevant plasma and reactor parameters to the size of DEMO are weakened. This is because the semi-empirical laws used to perform the extrapolation rely on databases built on data coming from tokamaks that are two orders of magnitude lower. The most relevant example of such parameters is τ_E, that is a transport-related quantity of the utmost importance.[8] In order to improve future fusion reactor designs, a better understanding of plasma transport properties is necessary. This means that further plasma turbulence studies are needed, since the discrepancy between empirical data and theoretical predictions are due to anomalous transport phenomena.

Turbulence is a phenomenon typical of fluid flows. For this reason a fluid-like description of plasmas is introduced in the next Chapter.

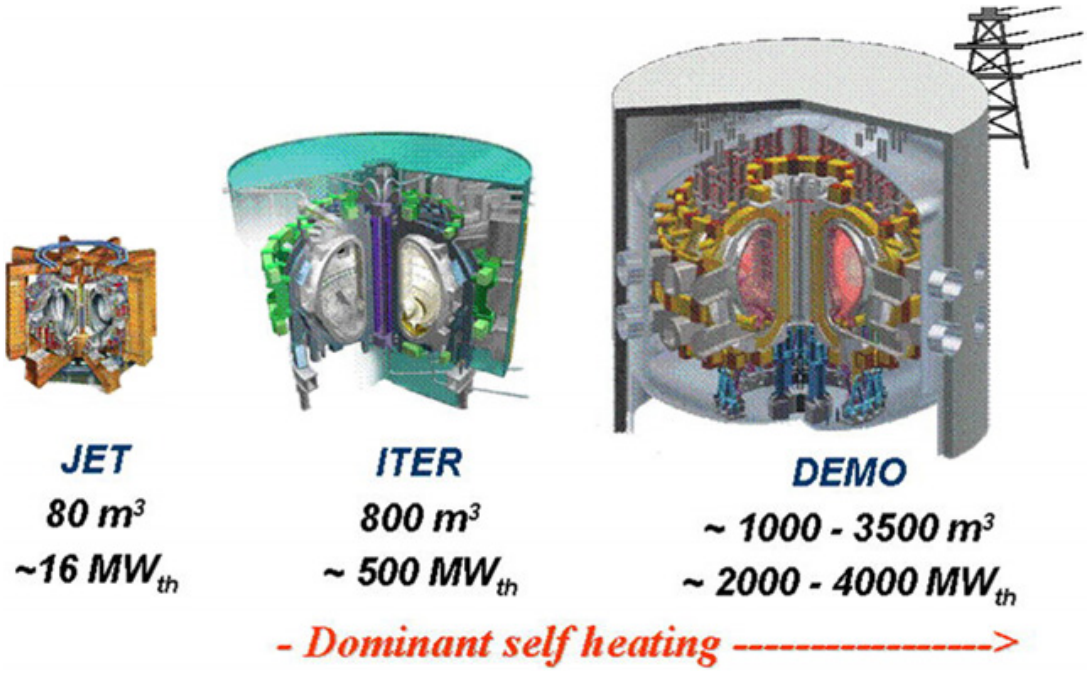

Figure 1.8: Size and thermal power output comparison among JET, ITER and DEMO.[5]

Chapter 2

Magnetized plasmas and turbulence phenomena

"Big whorls have little whorls
That feed on their velocity,
And little whorls have lesser whorls
And so on to viscosity."
*Lewis F. Richardson**

Plasmas can be described with different degrees of approximation and detail, going from an exact Newtonian model, to a single fluid description. Depending on which phenomena have to be described, a suitable model is chosen. A fluid-like model is a good option in order to describe the transport properties of a plasma. In fact, transport coefficient are derived from such a model, as well as turbulent phenomena, that are inherently present in the fluid equations.

In Sec. 2.1 the single-fluid description of a plasma, also known as magneto-hydrodynamic model, is derived as a reduction of the multi-fluids model, which is computed from the exact kinetic description of the system.

In Sec. 2.2 the issue of transport in a plasma is presented. Classical and neoclassical transport theories give analytical results that, in the latter case, take into account the toroidal geometry of a tokamak. However, both approaches fail in the prediction of transport coefficients for the particle density, energy, and magnetic fields inside the system. This is due to another contribution to transport, called anomalous, which is caused by turbulent flows in the medium.

Finally, in Sec. 2.3, an introduction to turbulence is given. The main characteristics of a turbulent behavior are presented, as well as turbulence theories that are exploited as a reference for this thesis work.

*English mathematician, physicist and meteorologist.

2.1 Derivation of the MHD description of a plasma

The aim of this section is the introduction of the system of equations that make up the so-called magneto-hydrodynamic model (MHD). This is a single-fluid-like description that is exploited for the modelization of the macroscopic behavior of conductive fluids, such as a plasma.

An exact description of any physical system is based on a Newtonian approach, meaning that such a description is given by one *Newton*'s equation for each particle that is part of the system. For a non-relativistic plasma, this is mathematically given by the following, in SI units[(9)]:

$$m_a \frac{d^2\mathbf{x}_{i,a}}{dt^2} = q_a \left(\mathbf{E}_{\text{micr}} \left(\mathbf{x}_{i,a}, t \right) + \frac{d\mathbf{x}_{i,a}}{dt} \times \mathbf{B}_{\text{micr}} \left(\mathbf{x}_{i,a}, t \right) \right) = \mathbf{F}_{a,\text{micr}}, \tag{2.1}$$

where m_a and q_a are, respectively, the mass and electric charge characteristic of the generic a-th population, $\mathbf{x}_{i,a}$ is the position vector of the i-th particle of said population and the *Lorentz*'s force that rules the dynamics is given by the microscopic exact electric and magnetic field. These are the self consistent fields that are given by both internal and external sources, i.e. internal and external charge and current densities. Such a description is easy from a theoretical point of view, however, it is quite complex from a computational one, since for macroscopic systems the number of involved particles is noticeably high and, therefore, the number of *Newton*'s equations that have to be solved is huge. Moreover, the self consistent *Maxwell*'s equations are added to the computational load. These are solved in a self consistent fashion, as shown here:

$$\begin{cases} \nabla \cdot \mathbf{E}_{\text{micr}} = \dfrac{\rho_{\text{micr}} \left(\mathbf{x}_{i,a}, t \right)}{\varepsilon_0} \\ \nabla \cdot \mathbf{B}_{\text{micr}} = 0 \\ \nabla \times \mathbf{E}_{\text{micr}} = -\dfrac{\partial \mathbf{B}_{\text{micr}}}{\partial t} \\ \nabla \times \mathbf{B}_{\text{micr}} = \mu_0 \left(\mathbf{j}_{\text{micr}} + \varepsilon_0 \dfrac{\partial \mathbf{E}_{\text{micr}}}{\partial t} \right) \end{cases} \tag{2.2}$$

The charge density is computed both in terms of the system's properties and of the external sources:

$$\rho_{\text{micr}} \left(\mathbf{x}_{i,a}, t \right) = \rho_{\text{ext}} + \sum_a \rho_{a,\text{micr}} = \rho_{\text{ext}} + \sum_a q_a \sum_i \delta \left(\mathbf{x} - \mathbf{x}_{i,a} \right), \tag{2.3}$$

where the *Dirac*'s delta distribution is exploited and the summations are performed over all particles of a given population and over all the considered populations. The current density is given in a similar way:

$$\mathbf{j}_{\text{micr}} = \mathbf{j}_{\text{ext}} + \sum_a \mathbf{j}_{a,\text{micr}} = \mathbf{j}_{\text{ext}} + \sum_a q_a \sum_i \mathbf{v}_{i,a} \delta \left(\mathbf{x} - \mathbf{x}_{i,a} \right). \tag{2.4}$$

An equivalent description, i.e. still exact, is obtained exploiting an Eulerian approach. The so-called kinetic model is based on the plasma's distribution function,

$f_{a,\mathrm{micr}}(\mathbf{x},\mathbf{v},t)$. This is defined as follows[9]:

$$f_{a,\mathrm{micr}}(\mathbf{x},\mathbf{v},t)=\sum_i \delta(\mathbf{x}-\mathbf{x}_{i,a})\,\delta(\mathbf{v}-\mathbf{v}_{i,a}), \tag{2.5}$$

where, again, the delta distribution is exploited and the summation is performed over all particles of the chosen population. The distribution function is related to the density of particles of the a-th population in phase space; in fact, the quantity

$$dn = f_{a,\mathrm{micr}}(\mathbf{x},\mathbf{v},t)\,d^3x d^3v \tag{2.6}$$

is the number of particles in the arbitrarily infinitesimal phase space volume $d^3x d^3v$. Under the assumption of no chemical, nor nuclear, reactions, the distribution function is conserved in time[9]:

$$\frac{df_{a,\mathrm{micr}}}{dt}=\frac{\partial f_{a,\mathrm{micr}}}{\partial t}+\nabla_{\mathbf{x}}\cdot(\mathbf{v}f_{a,\mathrm{micr}})+\nabla_{\mathbf{v}}\cdot\left(\frac{\mathbf{F}_{\mathrm{micr}}}{m_a}f_{a,\mathrm{micr}}\right)=0; \tag{2.7}$$

this is the *Klimontovich*'s equation. The middle term represents the change of the distribution function because of position changes in length space, while the last one describes alterations in momentum space caused by interactions among the particles. As previously stated, the system formed by one such equation for each population, and the *Maxwell*'s equations (Eq. 2.2), provides an exact description of the system. However, now the self consistency is guaranteed through another definition for the internal microscopic charge and current densities:

$$\begin{aligned}\rho_{a,\mathrm{micr}} &= q_a\int f_{a,\mathrm{micr}}d^3v,\\ \mathbf{j}_{a,\mathrm{micr}} &= q_a\int \mathbf{v}f_{a,\mathrm{micr}}d^3v.\end{aligned} \tag{2.8}$$

The two models previously presented are discrete, in the sense that they are based on the instantaneous positions in phase space of all the considered particles. To obtain a smooth distribution function, an ensemble average operation is carried out, and then $f_{a,\mathrm{micr}}$ is defined in terms of the averaged one:

$$f_{a,\mathrm{micr}}(\mathbf{x},\mathbf{v},t)=\langle f_{a,\mathrm{micr}}\rangle+\tilde{f}_{a,\mathrm{micr}}(\mathbf{x},\mathbf{v},t)=f_a(\mathbf{x},\mathbf{v},t)+\tilde{f}_a(\mathbf{x},\mathbf{v},t), \tag{2.9}$$

where the average part is smoothly changing through phase space and the other addend represents the fluctuations around the mean value. On the rightmost side the notation has been simplified for the sake of readability.
An analogous averaging procedure is carried out for the electric and magnetic fields:

$$\begin{aligned}\mathbf{E}_{\mathrm{micr}} &= \mathbf{E}+\tilde{\mathbf{E}},\\ \mathbf{B}_{\mathrm{micr}} &= \mathbf{B}+\tilde{\mathbf{B}}.\end{aligned} \tag{2.10}$$

Inserting Eq. 2.9-2.10 in Eq. 2.7 and performing a proper averaging procedure the *Boltzmann*'s equation is obtained:

$$\frac{\partial f_a}{\partial t}+\mathbf{v}\cdot\nabla_{\mathbf{x}}f_a+\frac{q_a}{m_a}(\mathbf{E}+\mathbf{v}\times\mathbf{B})\cdot\nabla_{\mathbf{v}}f_a=c_a, \tag{2.11}$$

where c_a is called collision integral, since it represents short range interactions among particles, and it is defined as follows:

$$c_a = -\frac{q_a}{m_a}\left\langle \tilde{\mathbf{E}} + \mathbf{v} \times \tilde{\mathbf{B}} \cdot \nabla_{\mathbf{v}} \tilde{f}_a \right\rangle . \tag{2.12}$$

Smooth long range interactions are represented by the last term on the left hand side in Eq. 2.11. The average is computed over the chosen phase space scales. Eq. 2.6 still holds for this model; however, the infinitesimal phase space volume cannot be arbitrarily small since now the average scale sets a lower limit.
Eq. 2.11 plus the properly averaged *Maxwell*'s equations form the so-called *Boltzmann-Maxwell* kinetic model.

From this, the first fluid-like model is derived through the computation of characteristic fluid quantities.[10] The fundamental ones are the particles' density, n_a, and the fluid velocity, $\mathbf{u}_a$. Here these quantities are computed for one population:

$$\begin{aligned} n_a &= \int f_a d^3v, \\ \mathbf{u}_a &= \frac{1}{n_a}\int \mathbf{v} f_a d^3v. \end{aligned} \tag{2.13}$$

Multiplying Eq. 2.11 by suitable scalar or vector quantities and then performing the integration over the velocity space, the system of equations for a multi-fluid description of a plasma is obtained:

$$\begin{cases} \dfrac{dn_a}{dt} + n_a \nabla \cdot \mathbf{u}_a = 0 & \text{(2.14a)} \\ m_a n_a \dfrac{d\mathbf{u}_a}{dt} = q_a n_a \left(\mathbf{E} + \mathbf{u}_a \times \mathbf{B}\right) + \nabla \cdot \underline{\underline{P}}_a + \underline{\underline{R}}_a & \text{(2.14b)} \\ \dfrac{3}{2} n_a \dfrac{dT_a}{dt} = -\nabla \cdot \mathbf{h}_a - \underline{\underline{P}}_a : \nabla \mathbf{u}_a - \mathbf{u}_a \cdot \underline{\underline{R}}_a + Q_a & \text{(2.14c)} \end{cases}$$

For each population there exists one of such a system. Eq. 2.14a is the continuity equation for the particles' density of the given population, Eq. 2.14b is its momentum balance and Eq. 2.14c is its energy balance. All total time derivative are to considered as follows:

$$\frac{d}{dt} = \frac{\partial}{\partial t} + \mathbf{u}_a \cdot \nabla. \tag{2.15}$$

$\underline{\underline{R}}_a$ is the momentum exchange rate between the for the a-th population and all the others. $\underline{\underline{P}}_a$ is the pressure tensor for the same population, for which $\underline{\underline{P}}_a = P_a \underline{\underline{I}} + \underline{\underline{\Pi}}_a$, where the second addend is the anisotropic pressure tensor. $\mathbf{h}_a$ is the heat flux and Q_a is the energy exchanged between the populations because of collisions. The second and third terms on the right hand side of the energy equation are, respectively, the power related to stresses and the dissipation caused by friction among different populations.
Now the source terms of the coupled *Maxwell*'s equations are given as functions of the fluid quantities:

$$\begin{aligned} \rho\left(\mathbf{x}, t\right) &= \rho_{\text{ext}} + \sum_a \rho_a = \rho_{\text{ext}} + \sum_a q_a n_a, \\ \mathbf{j}\left(\mathbf{x}, t\right) &= \mathbf{j}_{\text{ext}} + \sum_a \mathbf{j}_a = \mathbf{j}_{\text{ext}} + \sum_a q_a n_a \mathbf{u}_a. \end{aligned} \tag{2.16}$$

Notice that the system of equations given by the fluid relations and the *Maxwell*'s ones is not closed since pressure, heat flux and collision related quantities need to be expressed in terms of the fluid unknowns.

To simplify the description, a single fluid model is developed: the MHD model.[10] This is derived from the multi-fluid one by taking into account some approximations. For the sake of simplicity consider a two-fluid model for which the populations are given by electrons and singly charged ions.
Dropping high frequency phenomena allows to neglect macroscopic charge separation effects and displacements currents. This is done considering a characteristic MHD frequency that is lower than the lowest one present in the plasma; this is usually the ions' cyclotron frequency, therefore for MHD to be valid the following is required $\omega_{\mathrm{MHD}} \ll \Omega_i$. Furthermore, over length scales larger than the *Debye*'s one, i.e. $\lambda_{\mathrm{D}} \ll L_{\mathrm{MHD}}$, microscopic charge separation is neglected as well. Under these approximations the quasineutrality hypothesis holds true:

$$n_e = n_i = n. \tag{2.17}$$

The MHD equations are obtained as the summation of all the multi-fluids ones and are here shown:

$$\begin{cases} \dfrac{d\rho_m}{dt} + \rho_m \nabla \cdot \mathbf{u} = 0 & \text{(2.18a)} \\ \rho_m \dfrac{d\mathbf{u}}{dt} = \mathbf{j} \times \mathbf{B} - \nabla \cdot \underline{\underline{P}} & \text{(2.18b)} \\ \dfrac{3}{2} n \dfrac{dT}{dt} = (\mathbf{E} + \mathbf{u} \times \mathbf{B})\,\mathbf{j} - \underline{\underline{P}} : \nabla \mathbf{u} - \nabla \cdot \mathbf{Q} & \text{(2.18c)} \end{cases}$$

The particles' density is substituted by the mass density, ρ_m, and, in place of the multiple-fluid velocities, the single-fluid one is given, $\mathbf{u}$. Eq. 2.18c can be simplified taking into account the so-called generalized *Ohm*'s law:

$$\mathbf{E} + \mathbf{u} \times \mathbf{B} = \frac{1}{ne}\left(\mathbf{j} \times B - \nabla \cdot \underline{\underline{P}}_e\right) + \eta \mathbf{j}, \tag{2.19}$$

where the first and second terms on the right hand side represent, respectively, the *Hall*'s and *Seedback*'s effects, while the last term describes the plasma's resistivity. Under the assumptions made for MHD validity only the last term is relevant and, therefore, the resistive *Ohm*'s law is obtained. Then, the related MHD model is called resistive MHD and a *Joule*'s effect term is present in the energy balance equation $\left(\eta\,|\mathbf{j}|^2\right)$. The plasma resistivity is defined as follows:

$$\eta = \frac{m_e \overline{\nu}_{ei}}{e^2 n_e}, \tag{2.20}$$

where $\overline{\nu}_{ei}$ is the velocity-averaged electrons-ions collision frequency. When η is negligible the ideal *Ohm*'s law holds true and the related MHD model is called ideal MHD.

2.2 Transport in magnetized plasmas

The aim of this section is to give an insight on the reasons why a deeper understanding of turbulence phenomena in plasmas is crucial to the future of thermonuclear research and development. In fact, since there is no complete theory of energy transport, the computation of the confinement time, τ_E, for new devices, is based on semi-empirical scale laws obtained from data acquired from machines of the same kind.

There is a discrepancy between the theoretical energy transport coefficients and the experimental ones. The most important transport processes in a plasma are: heat conduction, particles diffusion, and magnetic field diffusion. The first one is the most severe in terms of effects on the overall energy losses from the system. Two models are considered in order to compute these coefficients in a tokamak system, namely: classical and neoclassical transport.[8] These descriptions take into account just Coulomb collisions, neglecting the so-called anomalous transport due to microscopic plasma instabilities. This fact, ultimately, leads to the discrepancy between theoretical predictions and experimental results.

The former approach is based on the derivation of transport equations for the three quantities stated previously, in a one dimensional system, which is the rectified torus. Under this approximation, a large aspect ratio torus is modeled as an equivalent cylinder. Thus, the MHD equations, exploited for the description of this system, are symmetric around the cylinder's axis; this means that the only relevant space coordinate is the radial one. Assuming Q as a generic physical variable, the final diffusion-like equations are of the form[8]:

$$\frac{\partial Q}{\partial t} = \frac{1}{r}\frac{\partial}{\partial r}\left(rD\frac{\partial Q}{\partial r}\right) + S\left(Q, r, t\right), \tag{2.21}$$

where D is the diffusion coefficient of interest and S is the source-sink term.
By neglecting viscosity and inertial effects, assuming a diffusive description of the heat flux, and a low-β tokamak expansion, the equations are the following[8]:

$$\begin{cases} \dfrac{\partial n}{\partial t} = \dfrac{1}{r}\dfrac{\partial}{\partial r}\left[rD_n\left(\dfrac{\partial n}{\partial r} + \dfrac{n}{T}\dfrac{\partial T}{\partial n} + \dfrac{2\eta_{\|}}{\beta_p\eta_{\perp}}\dfrac{n}{rB_\theta}\dfrac{\partial rB_\theta}{\partial r}\right)\right] \\ 3n\dfrac{\partial T}{\partial t} = \dfrac{1}{r}\dfrac{\partial}{\partial r}\left(rn\chi\dfrac{\partial T}{\partial r}\right) + S \\ \dfrac{\partial rB_\theta}{\partial t} = r\dfrac{\partial}{\partial r}\left(\dfrac{D_B}{r}\dfrac{\partial rB_\theta}{\partial r}\right) \end{cases} \tag{2.22}$$

Here, $\beta_p = 4\mu_0 nT/B_\theta^2 \sim 1$, the electric resistivity is divided in components parallel and perpendicular to the magnetic field, and χ is the thermal diffusivity of the system. In S various contributions are taken into account: ohmic and external heating, fusion alpha particle heating and radiation losses. The diffusion coefficients are given by the following:

$$\begin{aligned} D_n &= \frac{2nT\eta_{\perp}}{B_0^2}, \\ D_B &= \frac{\eta_{\|}}{\mu_0}, \end{aligned} \tag{2.23}$$

where B_0 is the on-axis magnetic field. The latter is not described by this model; in fact, in order to compute it, a single particle random walk approach is exploited.[8] According to this method, the result of a series of random collisions is described by a diffusion coefficient, which is defined in terms of the mean step size between two collisions and the average time between them. Let Δl and τ be these quantities, then:

$$D = \frac{(\Delta l)^2}{\tau}. \tag{2.24}$$

The thermal diffusivities for the ions and electrons population are computed and it is found that the former is much larger than the latter, in fact $\chi_i \sim (m_i/m_e)^{1/2} \chi_e$. Since the value used in the transport equation is the sum of the two contributions, as a first approximation the effective valued is solely given by the ions one: $\chi \approx \chi_i$. For a 50%-50% D-T plasma, the following is obtained[8]:

$$\chi_i^{\mathrm{CL}} = 0.1 \frac{n_{20}}{B_0^2 T_k^{1/2}} \quad \mathrm{m^2 s^{-1}}, \tag{2.25}$$

where n_{20} is measured in 10^{20} m^3, B_0 in T, and T_k in keV. This approach, however, largely underestimates the thermal diffusivity values, giving $\chi_i^{\mathrm{CL}} \sim 0.001\,\mathrm{m^2 s^{-1}}$, that is up to three orders of magnitude lower than the experimental one.[8]

A better description of the transport properties of a plasma in a tokamak configuration is given by an approach that actually takes into account the toroidal geometry of the system. The name of this method is neoclassical transport; this applies the classical transport theory to a toroidal geometry. For large aspect ratio tokamak, since the rectified torus approach is valid, the classical method is expected to give results in accordance to the actual toroidal computation. However, values of the thermal diffusivity up to order of magnitude greater are obtained. This is due to the fact that particles, in a toroidal plasma, are divided in two groups: passing particles or trapped particles. The second group is characterized by the fact that its particles are affected by the mirroring effect. Because of it, these particles are reflected at some point during their trajectory and are, therefore, trapped in closed orbits, which are called banana orbits thanks to the projection of their shape on the poloidal plane. Passing particles, on the other hand, can circulate around the torus. Surprisingly, both types of particles contribute to the transport phenomena. Moreover, the effects due to trapped particles dominate particle and heat losses from the system. Again, exploiting a random walk computation, the thermal diffusivity for the trapped particles is[8]:

$$\chi_i^{\mathrm{NC}} = 0.68 \cdot q^2 \epsilon^{3/2} \chi_i^{\mathrm{CL}} = 0.068 \cdot q^2 \epsilon^{3/2} \frac{n_{20}}{B_0^2 T_k^{1/2}} \quad \mathrm{m^2 s^{-1}}, \tag{2.26}$$

where q is the ions' charge and ϵ is th inverse aspect ratio. Even though the neoclassical thermal diffusivity is $\chi_i^{\mathrm{NC}} \sim 0.1\,\mathrm{m^2 s^{-1}}$ for typical tokamak parameters, it is still lower than the experimental one, which is $\chi \sim 1\,\mathrm{m^2 s^{-1}}$.[8] Moreover, it is experimentally found that both χ_e and D_n are lower than χ_i by just a factor of 3, whereas theory gives a much larger reduction, namely $(m_e/m_i)^{1/2}$.

It is obvious that even neoclassical theory is not suited for the prediction of transport parameters upon which a new machine should be designed. Anomalous

transport mechanisms are responsible for the previously presented discrepancies and their role is even more prominent in driving electrons' thermal conduction and particles' diffusion. These mechanisms are due to turbulence phenomena that are driven by plasma micro-instabilities. No complete plasma turbulence theory exists, up to the date of this work. Thus, it is possible to state that the problem of creating a theoretical model for a complete transport behavior is undoubtedly difficult.

Micro-instabilities are due to the fluctuating behavior of the fields that are present inside the plasma. Since all of the fluid and electromagnetic fields are interdependent, suitably coupled fluctuations of such fields are able to produce net transport. Consider, for example, the fluctuation of the electric field perpendicular to a magnetic field line. This leads to an $E \times B$[8] drift that produces net advection of particles when the particle density field is suitably fluctuating.[11] Multiple mechanisms contribute to the overall anomalous transport in an interactive way. Therefore, a diffusion-like approach to the description of anomalous fluxes is usually not considered valid.[12] However, if a single mode is found to be dominant, then the linked flux can be modeled as $\tilde{\Gamma} = (L^2/\tau)\,\nabla Q$, where L and τ are the the characteristic space and time scales of the considered turbulent structure and Q is the advected physical quantity.

2.3 Turbulence

The aim of this section is the introduction of the concepts of turbulence.

Even though there is no universally accepted definition of turbulence, which is also called turbulent flow, this is surely characterized by a number of peculiar features[13–15]:

1. **Irregularity**. Turbulent flows are highly irregular and chaotic. For these reasons they are usually studied with a statistical approach, instead of a deterministic one, even though these phenomena are described by the *Navier-Stokes*' equation, which is deterministic. Notice that not all chaotic flows are turbulent.

2. **Dissipation.** In a turbulent flow there are multiple motions taking place over a spectrum of different space scales, which are the sizes of the eddies that are generated in the process. The largest scales are of the order of the geometrical boundaries of the flow. Smaller eddies receive their kinetic energy from larger structures and these, in turn, receive it form even larger eddies. This energy flux is called *direct energy cascade*, since energy flows towards smaller structures. Eventually, the kinetic energy of the flow is dissipated into internal energy thanks to action of viscous shear stresses. The largest turbulent scales extract energy from the mean flow; therefore, a continuous energy supply is needed in order to sustain a turbulent behavior.

3. **Diffusivity.** Turbulence increases the transport of mass, momentum, and energy in a flow, accelerating the homogenization of fluid mixtures. This enhanced behavior is referred to as turbulent diffusivity, even though a diffusion-like description for this behavior is valid only as a first approximation.

4. **Three-Dimensional Behavior.** Turbulent flows are inherently three dimensional, even though a two-dimensional description is possible under suitable averaging of the flow.

5. **Continuum.** The flow is treated as a continuum since even the smallest turbulent scales are much larger than the molecular ones, meaning that the mean step size of the molecules' thermal motion is much shorter than the shortest turbulent space scale.

6. **Large Reynolds Numbers.** The Reynolds number is an dimensionless parameter that is defined as the ratio of inertial to viscous forces in a fluid that is subjected to relative internal motion. Mathematically, this is given by:

$$Re = \frac{uL}{\nu}, \tag{2.27}$$

where u is the mean fluid speed, L is a characteristic length scale, and ν is the kinematic viscosity of the fluid. The relative internal movements produce friction in the fluid, increasing the probability of generating a turbulent behavior. An increase of the viscosity of the system, on the other hand, inhibits turbulence generation, since more kinetic energy is dissipated through viscous mechanisms. Therefore, high values of Re are correlated to turbulent flows.

A turbulent flow is characterized by a seemingly random fluctuation of the fields describing the fluid; in the case of fluid velocity:

$$\mathbf{u} = \mathbf{U} + \tilde{\mathbf{u}}, \tag{2.28}$$

where $\mathbf{U}$ is the average flow velocity and $\tilde{\mathbf{u}}$ is the fluctuating part. The average is either taken over a given space scale, or over a given time scale. The two averaging operations are equivalent under the *Taylor*'s hypothesis of fully developed turbulence.[14, 16]

The largest turbulent scale, called injection range, I, is roughly given by the the correlation length of the turbulent field.[14] This is computed from the trace of the correlation tensor associated to the given turbulent field. Under the hypothesis of homogeneous and isotropic turbulence,the correlation tensor for the turbulent velocity is given by the following:

$$\underline{\underline{R}}\,(r) = \langle \tilde{\mathbf{u}}\,(\mathbf{x})\,\tilde{\mathbf{u}}\,(\mathbf{x}+\mathbf{r})\rangle\,; \tag{2.29}$$

notice that the tensor depends only on the relative distance between two points in the fluid. The ensemble average is exploited to obtain the results; however, a spatial or time average can be used when no ensemble is available, thanks to the ergodic theorem.[17, 18] Assuming that $R\,(r)$ is the trace of the tensor, then the correlation length is obtained:

$$\lambda_{\mathrm{c}} = \frac{1}{R\,(0)} \int_0^{+\infty} R\,(r)\,dr \approx I. \tag{2.30}$$

Kolmogorov, in his groundbreaking studies on turbulence, in addition to the hypothesis of homogeneous, isotropic and incompressible turbulence, assumed that no dissipation occurs during the energy cascade. This means that all the energy that is injected at I arrives at the so-called dissipation range, η.[19–21] Kinetic energy is rapidly dissipated at even smaller scales. The hypothesis of dissipationless energy cascade means that turbulence is self-similar over the so-called inertial range. Kolmogorov derived η purely on a dimensional analysis basis, assuming that this physical quantity solely depends on the kinematic viscosity of the fluid and on the energy dissipation rate, ϵ_d:

$$\eta = \left(\frac{\nu^3}{\epsilon_\mathrm{d}}\right)^{1/4}. \tag{2.31}$$

ϵ_d is defined from the overall energy content of the cascade. A wave-number is associated to each space scale via a *Fourier* transform; in this way the energy density of each harmonic is defined as $E(k)$, which is called kinetic energy spectrum. This quantity is rigorously defined from the *Fourier* transform of $\underline{\underline{R}}(r)$, that is the energy spectrum tensor, $\underline{\underline{\Phi}}(k)$.[14] The total turbulent kinetic energy per unit mass is computed from the following:

$$\frac{1}{2}\left\langle|\tilde{\mathbf{u}}|^2\right\rangle = \int_0^{+\infty} E(k)\,dk. \tag{2.32}$$

The energy dissipation rate is computed as follows:

$$\epsilon_\mathrm{d} = 2\nu\int_0^{+\infty} k^2 E(k)\,dk. \tag{2.33}$$

However, the most important conclusion that is derived from *Kolmogorov*'s theory, often referred to as *K41* theory, is that the spectral energy density is independent of ν over the inertial range. The Russian scientist assumed that $E \propto {\epsilon_\mathrm{d}}^a k^b$, so, since the energy dissipation rate depends on the kinematic viscosity, it is obtained that:

$$E(k) \propto {\epsilon_\mathrm{d}}^{2/3} k^{-5/3}. \tag{2.34}$$

Eq. 2.34 is the most famous result of the *K41* theory; it describes the energy density content of turbulence in the inertial range as a power law function of the wave-number. Kinetic energy is transferred in local manner, meaning that direct energy flow between modes having large k ratio is negligible.[22]

The *K41* theory is a very good approximation, or an exact description in some cases, for a very large variety of turbulent phenomena. However, since $\epsilon_\mathrm{d} \sim \langle\tilde{u}^2\rangle/\tau$, where τ is here the characteristic time scale of the turbulent structure, other power laws are obtained as this second parameter changes.[17] As an example, consider the turbulent magnetic field in a plasma and assume an expansion as the one in Eq. 2.28. Now the turbulent energy content of the magnetic field is given by Eq. 2.32, in which the turbulent velocity is changed with $\mathbf{b}$, and ϵ_d for the only magnetic part depends, again, on $\tilde{\mathbf{b}}$. If the crossing time of a typical turbulent length scale depends on the *Alfvén* speed, c_a, then:

$$\tau_\mathrm{MHD} = \frac{l}{c_\mathrm{a}} \sim \frac{1}{k\tilde{b}}. \tag{2.35}$$

Since $E_b(k) \propto \tilde{b}^2/k$, where the magnetic energy spectrum is considered, thanks to dimensional analysis Eq. 2.34 is obtained.[17] If, however, the crossing time depends on kinetic *Alfvén* waves or *Whistler* waves, then:

$$\tau_{\text{Hall}} \sim \frac{1}{k^2\tilde{b}}, \tag{2.36}$$

and then:

$$E_b(k) \propto {\epsilon_{\text{d}}}^{2/3} k^{-7/3}. \tag{2.37}$$

Another model, due to *Iroshnikov* and *Kraichnan*, predicts a different power law for the energy spectrum.[22–25] This theory, as the ones before, starts from the hypothesis of isotropic and incompressible turbulence. However, the phenomenon that initiates the turbulent energy cascade is the collision of two opposite-traveling *Alfvén* waves. The power law that results from this description is:

$$E(k) \propto k^{-3/2}. \tag{2.38}$$

It is usually referred to this model as *IK* theory.

2.4 Thesis goal

The goal of this thesis work is the development of a set of MATLAB® tools, which allows to perform a statistical analysis of the edge magnetic turbulent field on the TCV tokamak. This toolbox is here exploited for the analysis of multiple plasma discharges. The studied data is chosen from two TCV experimental campaigns. The first one is a research mission that focuses on the study of L-H transition physics, while the second one studies core electron temperature turbulence. This work is, therefore, the foundation of a greater experimental mission, that will study the characteristics of magnetic turbulence phenomena taking place during different phases of a plasma discharge. In the frame of a longer experimental mission, this work gives a solid technical basis, from the point of view of code development and usability. Moreover, it gives a first validation based on actual experimental data.

Chapter 3

The *Tokamak à Configuration Variable* and its fast magnetic measurements system

In this chapter, the concept of tokamak is described more in detail. This is done having in mind the goal of giving a brief, but exhaustive, description of the *Tokamak à Configuration Variable*, which is the flagship device present at the Swiss Plasma Center. This is one of the major European centers for plasma physics studies and it is hosted by the École Polytechnique Fédérale de Lausanne, located in Switzerland. The TCV tokamak is a very peculiar machine. In fact, thanks to the shape of its vacuum vessel and to the advanced shaping coils system, it allows the study of a large plethora of different plasma shapes and divertor configurations. Moreover, the relatively limited dimensions of the device allow for faster maintenance and upgrade operations with respect to other tokamaks.

The TCV fast magnetic acquisition system was recently upgraded. Its main feature is the presence of 228 sensors, that are exploited for both MHD equilibrium and fluctuations analysis. Moreover, in April 2015 three additional high-frequency three-dimensional magnetic sensors have been installed. These are capable of acquiring the magnetic signal along all three toroidal coordinates. However, even if it is possible to use these signals for the analysis, the focus is set on the other ones since the goal of this work is to obtain a reliable first iteration of the analyzing code.

In Sec. 3.1, first a more detailed description of a tokamak is given, then the main features of TCV are presented.

In Sec. 3.2 the fast magnetic measurement system installed in TCV is discussed.

3.1 Tokamak concept and TCV description

The aim of this section is to present the main characteristics of the tokamak concept. In the more general frame of tokamaks, TCV is introduced as an highly flexible device.

The word tokamak is the Latin transliteration of the Cyrillic acronym that stands for *toroidal'naya kamera s magnitnymi katushkami*, which is translated as toroidal chamber with magnetic coils. A tokamak is an axisymmetric toroidal machine capable of confining a plasma with the usage of a large toroidal magnetic field. The plasma generated inside such a device is said to be in a tokamak configuration. This is characterized by a large toroidal magnetic field and a significant toroidal direct current.[8] The tokamak design is the leading candidate for the achievement of economically viable fusion reactor status. This is because of its superior performance with respect to other magnetic fusion concepts.

In Fig. 3.1 a schematic diagram of a tokamak is shown, where a poloidal slice of a typical machine is drawn. A tokamak needs four magnetic coils systems in order to operate. The first one is the toroidal field coils system, which generates the large toroidal magnetic field inside the machine exploiting a set of multiple poloidal currents. The second system is the ohmic transformer, which generates the toroidal electrical current inside the tokamak. This is needed for equilibrium and heating purposes. The third system is the vertical field one, which is required to achieve toroidal force balance. The last set is the shaping coils system, exploited to change the plasma poloidal shape. Cross sections different from the circular one are useful in order to achieve better MHD equilibrium configurations.

An example of profiles for the main physical quantities of interest are plotted in Fig. 3.2. These quantities are: toroidal and poloidal components of the magnetic field, plasma pressure, toroidal current density, and safety factor. The latter, which is denoted as $q(r)$, is a very important figure of merit for a tokamak. In fact, large values of $q(r)$ are linked to stable MHD equilibrium configurations.[8] In mathematical term this is given by the following:

$$q(r) \approx \frac{rB_\phi}{R_0 B\theta}, \tag{3.1}$$

with R_0 being the plasma major radius, which is defined as that R coordinate where $B_\theta = 0$. Specifically, $q(r) > 1$ for stability.

A typical tokamak discharge is carried out following: the toroidal magnetic field is established in the device, next the neutral gas is injected inside the vacuum chamber, and then the toroidal current is ramped up to a maximum and maintained to a flat level until the end of the experiment. However, it is often possible to change any of the said parameters during the discharge.

As anticipated, the major advantageous characteristic of a tokamak is its better physics performance. This means that, thanks to large edge safety factor values, good values of β are achievable even without a perfectly conducting first wall.[8] This is the name usually given to the wall of the vacuum vessel. Moreover, high τ_E are achieved in tokamaks, meaning that higher temperatures can be obtained with less external heating power.[8] For example, typical values of the main parameters reached in JET are: $n(0) \approx 4{\cdot}10^{19}\mathrm{m}^{-3}$, $T_\mathrm{i}(0) = 28\,\mathrm{keV}$, $T_\mathrm{e}(0) = 2T_\mathrm{i}(0)$

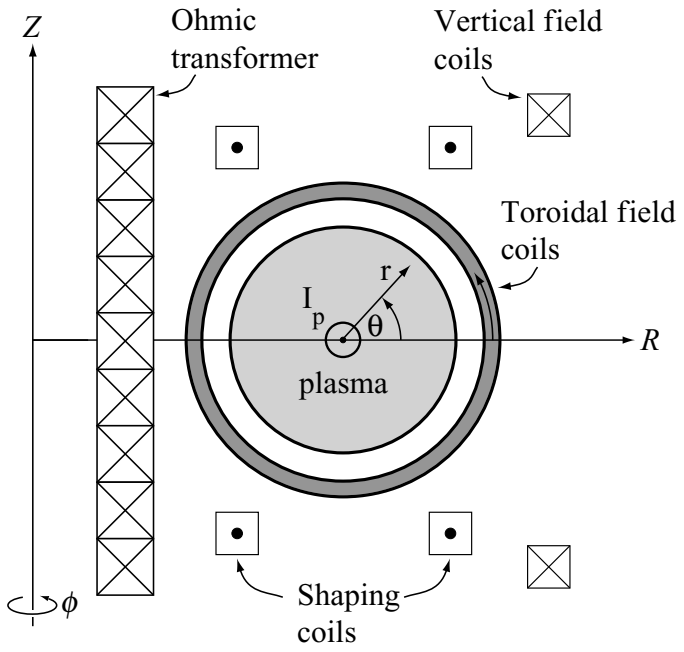

Figure 3.1: Basic diagram of a tokamak.[8]

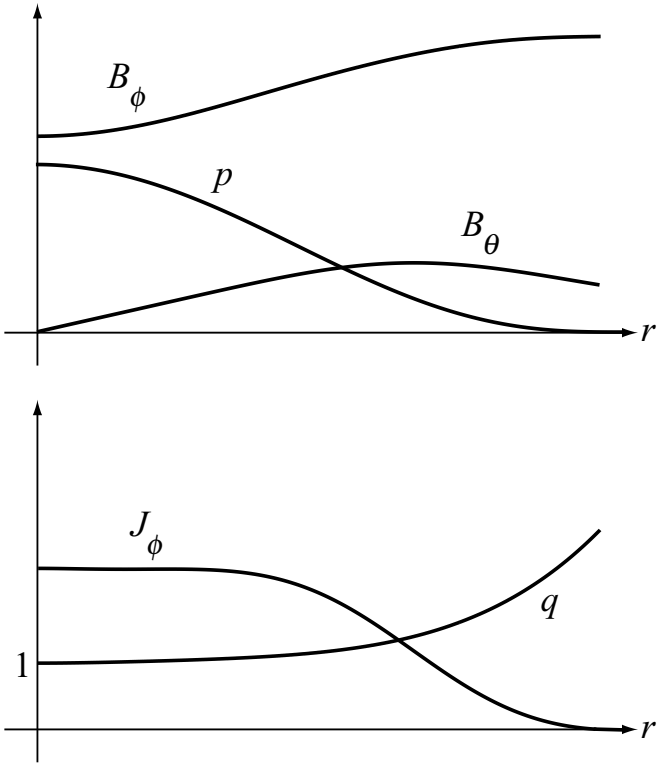

Figure 3.2: Example of profiles for the main physical quantities inside a tokamak in the large aspect ratio limit.[8]

$\langle\beta\rangle \approx 0.018$, and $\tau_{\mathrm{E}} \approx 0.9\,\mathrm{s}$.[8] These parameters give $\langle p\rangle\,\tau_{\mathrm{E}} \sim 0.84\,\mathrm{atm\ s}$, which is one order of magnitude lower than the one required for plasma ignition. See Chapter 1 for further details on the various plasma ignition conditions and power balances.

Nevertheless, tokamaks are affected by various drawbacks. The first one is that these machines would be significantly less expensive if the requirement of a large toroidal magnetic field could be lessen. This prerequisite is set by the need of achieving high values of the safety factor. In fact, other types of design trade a lower value of the toroidal field with the possibility of having much more compact and less expensive machines. This, however, poses the issue of finding intelligent solutions to deal with lower safety factor values. Nevertheless, recent advancements in the field of superconductors may allow the construction of smaller tokamaks, thanks to much higher magnetic fields.[26]

The second major problem is the achievement of steady state operations. All currently operating tokamaks require an external transformer in order to drive the toroidal current. Since an ohmic transformer cannot operate for indefinitely long periods of time, other external current drives are needed. These are either a microwave or a neutrals beam system, that are the same exploited for plasma heating. However, the current-drive efficiency is not yet satisfactory.[8] Even taking into account the phenomenon of bootstrap current, which is a neoclassical transport effect, an external drive is still needed. Bootstrap current might be able to generate up to 95% of the total toroidal current, and a required bootstrap fraction of at least of 75% is agreed upon by the research community.[8]

The TCV tokamak is a very interesting machine because of its versatility in the control of plasma shape. This flexibility is made possible by sixteen independently-powered poloidal field coils.[27] This allows to study the properties of many different plasma shapes and divertor configurations. In Fig. 3.3 it is possible to see the interiors of TCV, while in Fig. 3.4 its the cross section is shown.[28] Notice the almost rectangular shape of the TCV vacuum chamber, which is formed by sixteen toroidal sectors. Such a shape is essential in order to explore different plasma cross sections. In fact, thanks to this design choice, large intervals of elongation and triangularity are achievable in TCV, as well as somewhat exotic divertor configurations, like the snowflake one.[29] In Fig. 3.4 it is also possible to see the simulation of a snowflake configuration. Moreover, in TCV it is possible to obtain plasma doublets, which are two simultaneous plasma discharges inside the same chamber.[30]

In Table 3.1 the main TCV parameters are shown, whereas in Table 3.2 the possible plasma parameters are presented. Notice the compact dimension of the device. This is a strength, in the sense that this permits to perform relatively fast maintenance and upgrade operations. In fact, the fast magnetic measurements system of TCV was recently upgraded. This is the acquisition system exploited for this thesis work, and it is also the focus of Sec. 3.2.

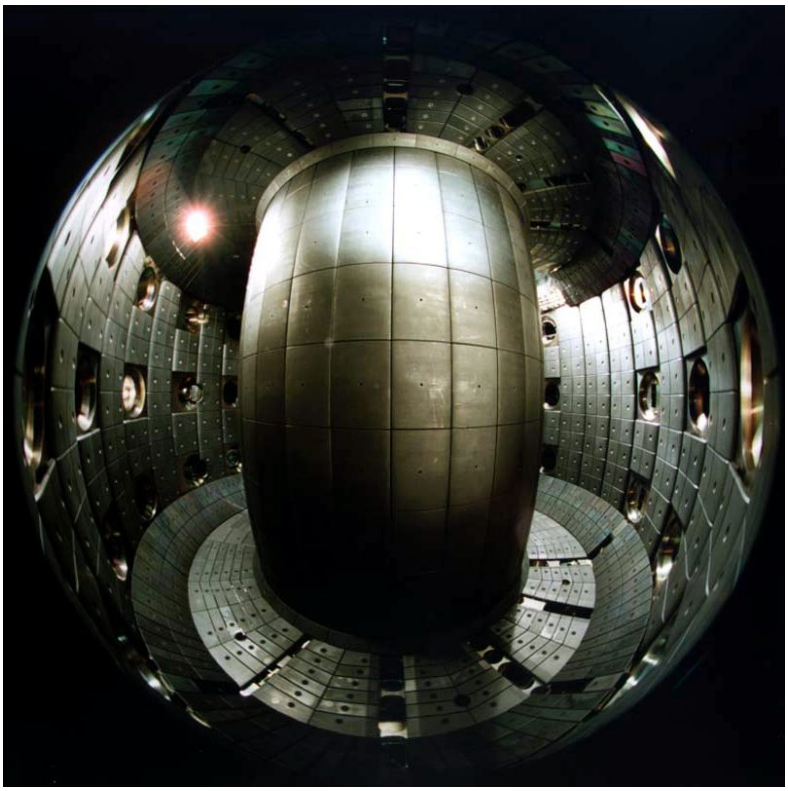

Figure 3.3: View of the interior of the TCV vacuum vessel.[28]

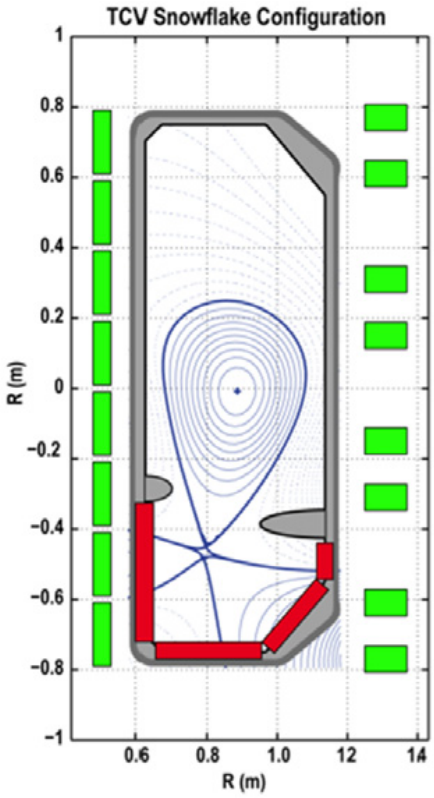

Figure 3.4: Example of snowflake divertor configuration inside TCV. Notice the sixteen poloidal shaping coils.[28]

Table 3.1: TCV tokamak main parameters.

vessel height	1.54 m	vessel width	0.56 m
vessel elongation	2.9	wall thickness	10 mm
first wall graphite coverage	>90%	number of tiles	1692
toroidal field magnets	16	on-axis field	1.54 T
ohmic coils	7	maximal ohmic current	27 kA
shaping magnets	16	maximal shaping current	7 kA
vertical force balance coils	2	maximal vertical current	2 kA
ohmic heating	<1 MW	X2 ECH	3 MW
X3 ECH	1.5 MW	neutrals beam heating	1 MW

Table 3.2: TCV plasma main parameters.

major plasma radius	0.89 m	minor plasma radius	0.25 m
aspect ratio	3.5	plasma elongation	$0.9 \div 2.8$
plasma triangularity	$-0.8 \div +0.9$	maximal I_p	1.2 MA
max discharge duration	$2.6 \div 4$ s	core n_e	$10^{19} \div 10^{20}\mathrm{m}^{-3}$
core T_e (ohmic)	<1 keV	core T_e (ECH)	<15 keV
core T_i (ohmic)	<1 keV	main ion	H, D or He

3.2 The fast magnetic acquisition system installed in TCV

The fast magnetic measurements system in TCV is quite advanced. More than two hundreds magnetic probes are currently installed inside the TCV vacuum chamber. Given the relative compactness of the device, this allows for an unprecedented spatial precision in the pick-up of edge magnetic field inside a tokamak.

The majority of these probes are *Mirnov* coils. These consist of a 1 mm THERMOCOAX mineral insulated coaxial wire that is wound in one layer around a ceramic body.[31] The layer is designed in order to minimize the effective area perpendicular to the probe axis. This is done in order to obtain coils that pick-up signals only along a single space direction. Given the absence of currents outside of the plasma, the fluctuating poloidal component of the magnetic field is expected to be much greater than the toroidal counterpart. For this reason, such *Mirnov* coils are aligned along the poloidal direction. There are currently 203 coils of this type installed inside the TCV vacuum chamber. In Fig. 3.5 a representation of two *Mirnov* coils installed on the vacuum vessel is given. These probes are grouped in toroidal and poloidal arrays. There are four poloidal arrays, having 38 probes each, located every 90° along the toroidal coordinate. The toroidal arrays are located at three different heights on both the internal and external sides of the chamber. The former is called high-field side (HFS) and the latter low-field side (LFS). The three possible heights are $z = +0.345$ m (TOP), $z = 0$ m (MID) and $z = -0.345$ m (BOT). On the HFS eight probes form an array, whereas 17 probes are part of a single LFS array. The sampling frequency of these probes can be as high as

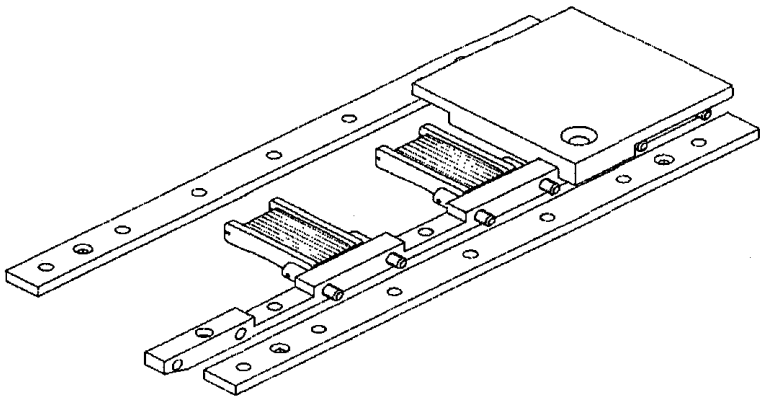

Figure 3.5: Drawing of the magnetic probe assembly on the vacuum vessel.[31]

500 MHz. However, the cutoff frequency of these probes is located at 120 kHz. In Fig. 3.6 an example of frequency response is shown.

Alongside the *Mirnov* coils, two other type of probes are installed. Three sets of eight saddle loops are located at the three standard heights. The last type of probes are the LTCC3D magnetic probes.[32, 33] This are three-dimensional probes, meaning that their pick-up is along the three toroidal directions. In fact, their aim is the detection of high frequency fluctuation in the magnetic field in the toroidal, poloidal and normal (to the last closed flux surface) directions. These probes can operate at a sampling frequency as high as 2 MHz.

A coil is exploited in order to measure a time-varying magnetic flux. When the effective area of the coil is constant, the variation of the magnetic flux is entirely due to a changing magnetic field inside the coil. The effective area along the coil's axis is measured with an *Helmholtz* coil field apparatus.[34] In Fig. 3.7 the results are shown as a function of the magnetic field frequency. In this situation the voltage output at the end of the coil is expressed by the following:

$$V(t) = NA\frac{\partial B}{\partial t}, \tag{3.2}$$

where N is the number of coil's turns and A is the surface area enclosed by a single one. Taking the *Fourier* transform of the previous equation, the output voltage is expressed as a function of the magnetic field's angular frequency:

$$V(\omega) = \omega NAB(\omega). \tag{3.3}$$

The measured voltage, however, is not the one expressed in Eq. 3.3 because the effects of the varying frequency on both the probe and the acquisition electronics needs to be taken into account. The measured quantity is given by the following:

$$V_{\text{meas}}(\omega) = B(\omega) H_{\text{probe}}(\omega) H_{\text{electr}}(\omega), \tag{3.4}$$

where $H_{\text{probe}}(\omega)$ and $H_{\text{electr}}(\omega)$ are, respectively, the probe's and electronics' transfer functions.

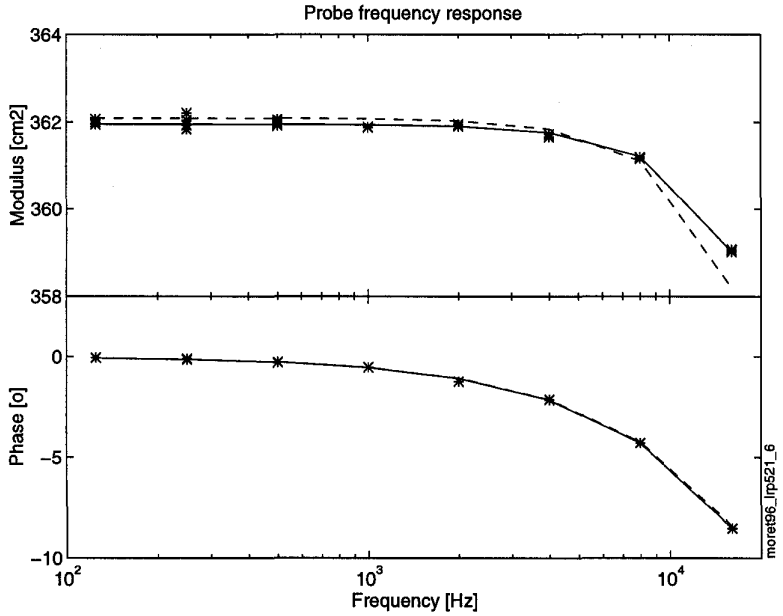

Figure 3.6: Frequency response for one the TCV *Mirnov* coils.[31] The star points are the experimental measurements, the dashed line is the first order transfer function, and the solid one is the second order transfer function.

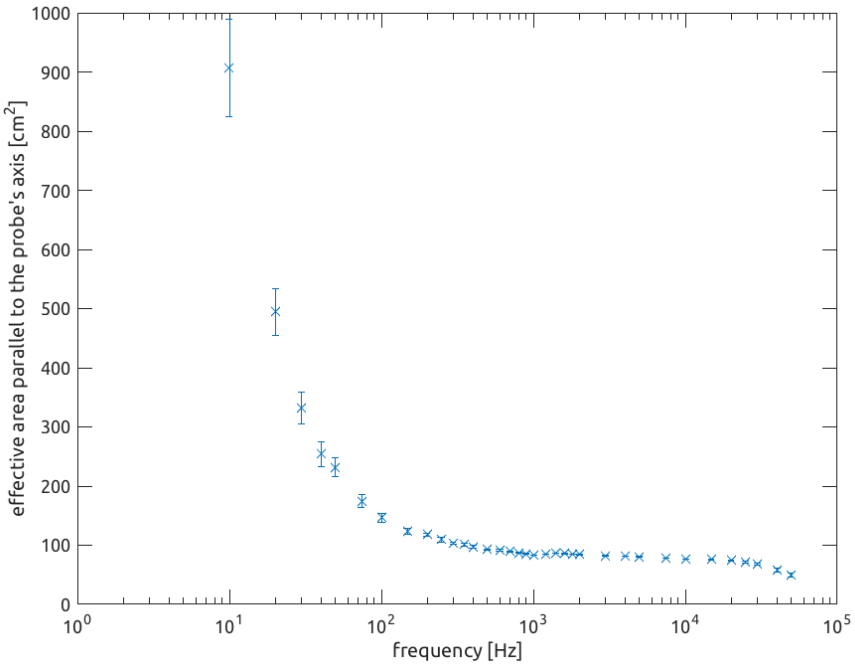

Figure 3.7: Effective area measurements along the probe's axis. Notice how the value is almost constant in the $1 \div 10$ kHz interval of frequencies.

It is a well known result that the transfer function of the probe is related to the impedance of the probe-cables side of the measurement apparatus.[35] Consider this impedance as described by the ratio of two polynomials, such as:

$$Z(\omega) = \frac{N(\omega)}{D(\omega)}, \tag{3.5}$$

then the probe's transfer function is given by:

$$H_{\text{probe}}(\omega) = \frac{1}{D(\omega)}. \tag{3.6}$$

The impedance measure is fitted with multiple rational functions of the angular frequency. Generally, the higher the orders, the better is the fit of both $|Z(\omega)|$ and $\Phi_Z(\omega)$. Typical orders are 3/2 when one resonance if found, and 5/3 for two resonances. In Fig. 3.8 the impedance modulus and phase are shown for one the *Mirnov* coils. This measurement is acquired with an *Helmholtz* coil field apparatus.[34] The fitting is performed with a recursive algorithm and it is found to be of orders 3/2.[35] In Fig. 3.9 the related transfer function is plotted. Notice that this has the same resonance of $Z(\omega)$.

The acquisition system has a limit on the maximum peak-to-peak measurable voltage. This is 20 V, meaning that $-10\,\text{V} < V_{\text{meas}} < +10\,\text{V}$. Moreover, the D-TACQ acquisition board divides the measurement interval in discrete steps, which depend on the number of bits dedicated to the store the values. For the installed system this is $n = 14$, meaning $2^{14} - 1$ possible steps. The system also reserves some bits to set a noise threshold, which in this case is $n = 3$. This is important because it determines the minimum magnetic field amplitude that the system can detect.

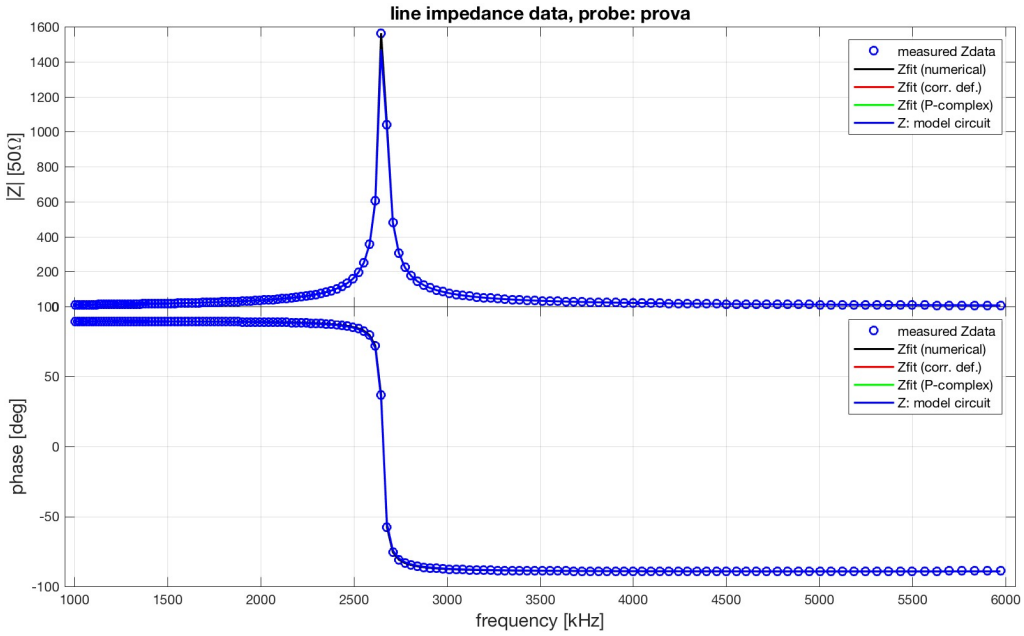

Figure 3.8: Impedance measurement and polynomial fitting for a *Mirnov* coil. Notice the presence of a single resonance at 2.7 MHz.

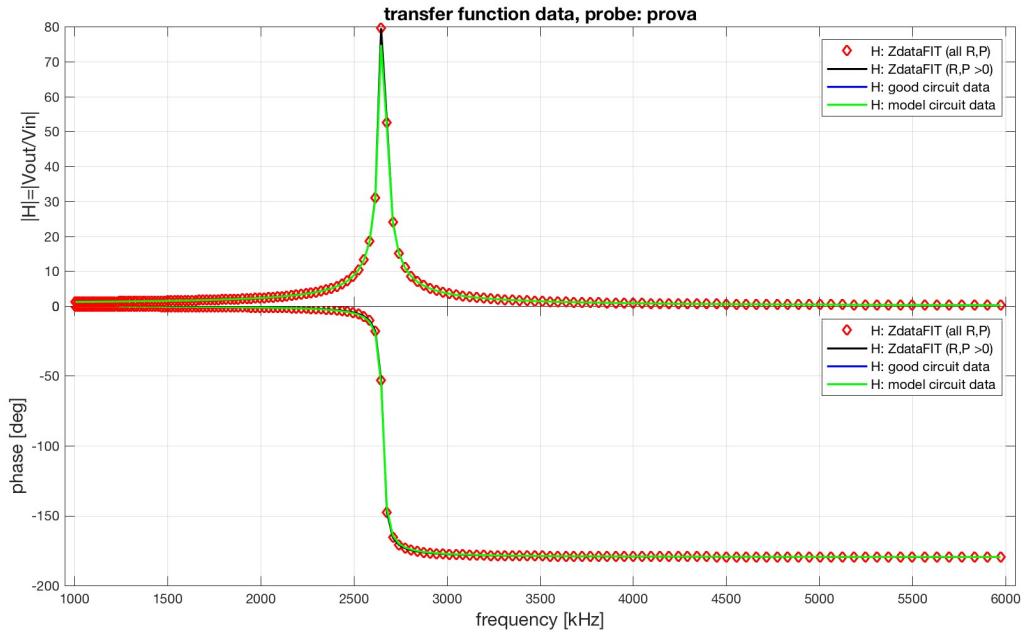

Figure 3.9: Transfer function of a *Mirnov* coil. Notice the presence the same resonance of $Z(\omega)$ at 2.7 MHz.

Chapter 4

Edge turbulent magnetic field characterization : a statistical approach

The aim of this chapter is to introduce the statistical analysis exploited to characterize the time-variant part of the edge magnetic field during a TCV shot. This physical quantity is denoted as $\delta B(t)$ and it is defined as follows:

$$\delta B(t) = B(t) - \langle B \rangle = B(t) - \frac{1}{t_2 - t_1} \int_{t_1}^{t_2} B(t) dt, \tag{4.1}$$

where the mean value is computed over the chosen time interval.

The statistical analysis described in this chapter can be performed on both scalar and vector physical quantities. In fact, the following set of tools is absolutely general, meaning that it can be used to analyze all sort of signals.[17]

The analysis is going to be performed on the normalized time-varying part of $\delta B(t)$, which is $b_{\text{norm}}(t)$ and it is computed starting form the following:

$$b(t) = \delta B(t) - B_0 = \delta B(t) - \langle \delta B \rangle, \tag{4.2}$$

where B_0 is the mean value of the time-varying field over a proper time window. $b_{\text{norm}}(t)$ is finally obtained through normalization:

$$b_{\text{norm}}(t) = \frac{b(t) - \mu}{\sigma}, \tag{4.3}$$

where μ and σ are, respectively, the mean value and standard deviation of $b(t)$ over the selected time period. The normalization is performed over σ in order to avoid divergent values that a normalization over μ would produce in those cases when the mean is very close, or is equal to, zero. In the rest of the document $b(t)$ is used in place of $b_{\text{norm}}(t)$ for better readability.

The analysis is carried out exploiting the following statistical tools: autocorrelation function, power spectrum density, probability density function of

temporal increments, temporal structure functions, permutation entropy and complexity.[17] All of these tools are used to analyze a time domain signal. However, a space domain characterization is possible thanks to the *Taylor*'s hypothesis.[16, 17] This states that when turbulent structures are are convected with the plasma flow, meaning that they do not move in the plasma frame of reference, a signal at frequency f is related to a wave-vector k through the following:

$$f = u \cdot \frac{k}{2\pi}, \tag{4.4}$$

where u is the characteristic advection speed. Another way of stating the validity of Eq. 4.4 is that $v/u \ll 1$ needs to hold true, that is that the velocity of the turbulent structures is much lower than the advection one.[17] Thanks to Eq. 4.4 an information encoded in a time series can be related to length space information, and vice versa. To the phenomenon of advection of the magnetic field it is usually referred to as magnetic filed lines freezing in the plasma flow.

While the first two methods are considered to be basic, or even 'classical', tools for statistical analysis of turbulent flows, the temporal structure function and the permutation entropy and complexity methods are considered to be more advanced mathematical tools. All of the following analysis can be performed also on neutral fluids turbulence studies.[36]

In Sec. 4.1 further details on the auto-correlation function are given.

In Sec. 4.2 the power spectrum density analysis is discussed. The power spectrum density is compared to the one predicted by three turbulence theories.

In Sec. 4.3 the concept of time increments is introduced. A record of time increments is computed and this is exploited for the construction of a probability density function, which is compared to the one obtained from a pure noise signal.

In Sec. 4.4 the temporal structure function is introduced as a generalization of the record of time increments. This tool is useful to study the self-affinity of the phenomenon that generates the signal, and to understand whether dissipation processes take place over the chosen time scales.

In Sec. 4.5 the concept of *Hurst*'s exponent and fractal dimension of the signal are given. These are exploited in order to understand the long time correlation of a signal.

In Sec. 4.6 the CH plot of a signal is constructed. This allows the quantification of noise content of a signal and the identification of the mathematical function that the signal follows.

Finally, some astrophysical results are presented in Section 4.7. Since no relevant statistical analysis has been previously done on tokamak-level magnetic turbulence, the solar wind results are taken as a reference for this work.

4.1 Auto-correlation function

In this section an introduction to the auto-correlation function is given.

This tool allows the computation of the auto-correlation time of a signal, denoted as τ_c. This is the correlation time between the signal and itself or, in other words, it is the time after which it becomes de-correlated. For this reason it is also called de-correlation time. During a period of time of order τ_c the signal shows a self-similar wave-form and, therefore, it is commonly said that 'it loses its memory' on larger time scales. The computation of this parameter is quite important since it sets a characteristic time scale over which turbulent structures live.

In order to calculate τ_c, the normalized auto-correlation function is obtained multiplying the time series with a shifted copy of itself, as follows:

$$\underline{\underline{R}}(\tau) = \frac{\langle \mathbf{b}(t+\tau)\mathbf{b}(t)\rangle}{\langle \mathbf{b}(t)\mathbf{b}(t)\rangle}, \tag{4.5}$$

where $\underline{\underline{R}}$ is the correlation tensor and τ is called time delay, or lag time. The denominator serves as the normalization factor. Eq. 4.5 is the time analogous of Eq. 2.29. Taking into account only one component of the field, only one component of the correlation tensor becomes relevant. In the frame of this work only the poloidal component of the magnetic field is acquired, therefore just the $R_{\phi\phi}$ component of the correlation tensor is analyzed. Form here on the subscript is dropped for better readability.

$R(\tau)$ is an even dumped oscillating function that starts from unity and rapidly converges toward zero. In Fig. 4.1 an example computed form one of the acquired TCV signals is presented. On the left plot the whole function is shown, while on the right only the portion for positive delays is shown. Notice that the function is symmetric with respect to the vertical axis.

τ_c is mathematically defined as that value of τ at which $R(\tau)$ drops under a chosen value. This is usually set to be equal to e^{-1} or $1/2$. Looking at Fig. 4.1 it is easy to understand that there is a degree of freedom in the choice of τ_c. In fact, the function might cross the reference factor multiple times because of its oscillating behavior. Eventually, the choice is made between two options: define τ_c either as the delay of the first or of the last crossing. The former one, however, allows the writing of a more robust search algorithm for the crossing delay coordinate, meaning that such an algorithm will always provide comparable de-correlation times. See App. A for the the actual code.

The computation of the de-correlation factor is also useful in order to calculate an estimate of an effective turbulent diffusion coefficient.[37, 38] In fact, this can be estimated using the following:

$$D_\perp = \frac{{\lambda_c}^2}{\tau_c}, \tag{4.6}$$

where $D_\perp$ is the coefficient perpendicular to a magnetic flux surface and λ_c is the turbulent correlation length. Although the related algorithm has not been tested, the computation of the diffusion coefficient is possible with the developed code. The computation of both $D_\perp$ and λ_c is one of the future feature that are going to be developed in the frame of the toolbox project.

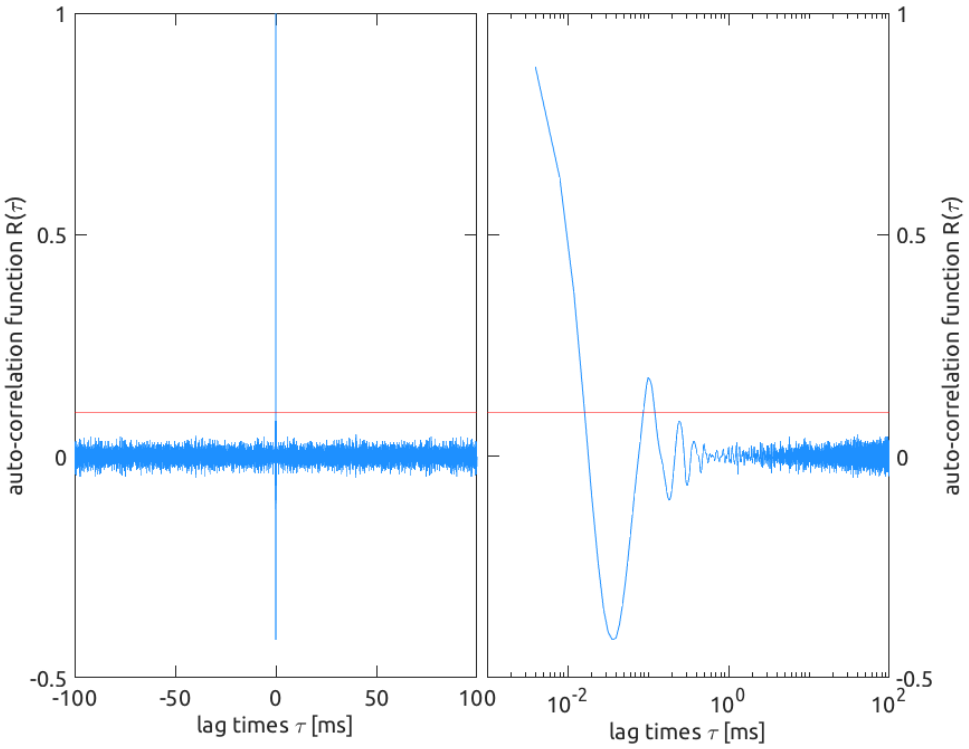

Figure 4.1: Example of auto-correlation function $b\,(t)$ measured by one of the *Mirnov*'s coil placed inside TCV. The signal is HFS-MID-003-001, acquired during TCV shot #55597. On the left side the complete function is plotted, whereas on the right only the part fro positive delays is shown. In red the e^{-1} factor is shown.

4.2 Power spectrum density

The aim of this section is to provide an introduction to the power spectrum density (PSD) analysis.

The PSD is a representation of the spectral power content of a time series. With PSD it is usually referred to the experimentally obtained energy spectrum, which is introduced in Sec. 2.3. A turbulent signal is divided in harmonics, each having an energy content. The PSD describes the power density content of all such harmonics. Energy can flow among different frequencies, giving rise to flows usually called energy cascades, or power cascades. These can either be direct or not, based on energy flowing respectively from low towards high frequencies or vice versa. The energy cascade related to turbulence phenomena is expected to be direct, since the length scale of turbulent energy flows decreases as time evolves. When length scales become smaller, the related wave-vector increase and, thanks to Eq. 4.4, the linked frequencies increase as well. On a PSD vs frequency plot an higher energy content is therefore shown at lower frequencies in the case of direct cascade.

For simplicity, and to achieve faster computation, the real power spectrum is actually chosen as the tool for the analysis; this is defined as follows:

$$E_b\,(\omega) = \frac{1}{t_2 - t_1}\left[\int_{t_2}^{t_1} b\,(t)\,e^{-i\omega t}dt\right]^2 . \tag{4.7}$$

Notice that the developed algorithm actually calculates $E_b(f)$. However, frequencies and angular frequencies are proportional physical quantities and then the two PSDs present the same information content.

Over the frequency interval that corresponds to the magnetic turbulence's inertial range, the PSD shows a decreasing linear behavior on a log-log plot. In fact, over this interval the PSD is described by a power law:

$$E_b(f) \propto f^{a_1}, \tag{4.8}$$

where a_1 is the above-mentioned slope and, therefore, $a_1 < 0$. As written in Sec. 2.3, usually the energy spectrum is described as a function of wave-vector k:

$$E_b(k) \propto k^{a_2}. \tag{4.9}$$

However, as previously stated, spatial information are encoded into a time series when the *Taylor*'s hypotheses holds true. In such cases the following equality holds true: $a_1 = a_2$. In this way the slope computed from the PSD in frequency domain is compared to the exponents obtained by the reference turbulence theories. Such exponents are:

1. **-5/3.** In this case the power law is given by the well known *K41* theory. Turbulence is expected to be homogeneous, isotropic, incompressible and dissipationless over the inertial range of frequencies. The underlying mechanism is expected to be related to *Alfvén* waves.[17]

2. **-3/2.** The power law that has this exponent is given by the *IK* theory. Turbulence has the same characteristic as the previous case. However, the underlying mechanism is the interaction of opposite-traveling MHD waves.[22, 23]

3. **-7/3.** In this last case turbulence is not guaranteed to be incompressible, since the underlying mechanism can be related to either *Whistler* or kinetic *Alvfén* waves.[17] In this work this exponent is referred to as *Hall*'s exponent.

The computation of the slope in a linear section of the PSD log-log plot gives an insight on the underlying process to which turbulent flow is linked. In Fig. 4.2, an example of PSD plot is shown.

The energy spectra of pure noise signals are characterized by power laws analogous to the ones expected from this analysis. Nevertheless, the exponents of these functions are different. Here are some examples: $a_1 = 0$ for a white noise signal, $-2 < a_1 < 0$ for pink noise, $a_1 = -2$ for brown noise, $a_1 = +1$ for blue noise and $a_1 = +2$ for violet noise.

4.3 Temporal increments analysis

In this section, an introduction to the temporal increments analysis is given.

The aim of this tool is the detection of the presence of coherent structures in a time series. This is achieved through the computation of the probability density function (PDF) of time increments. Thanks to this analysis an actual signal

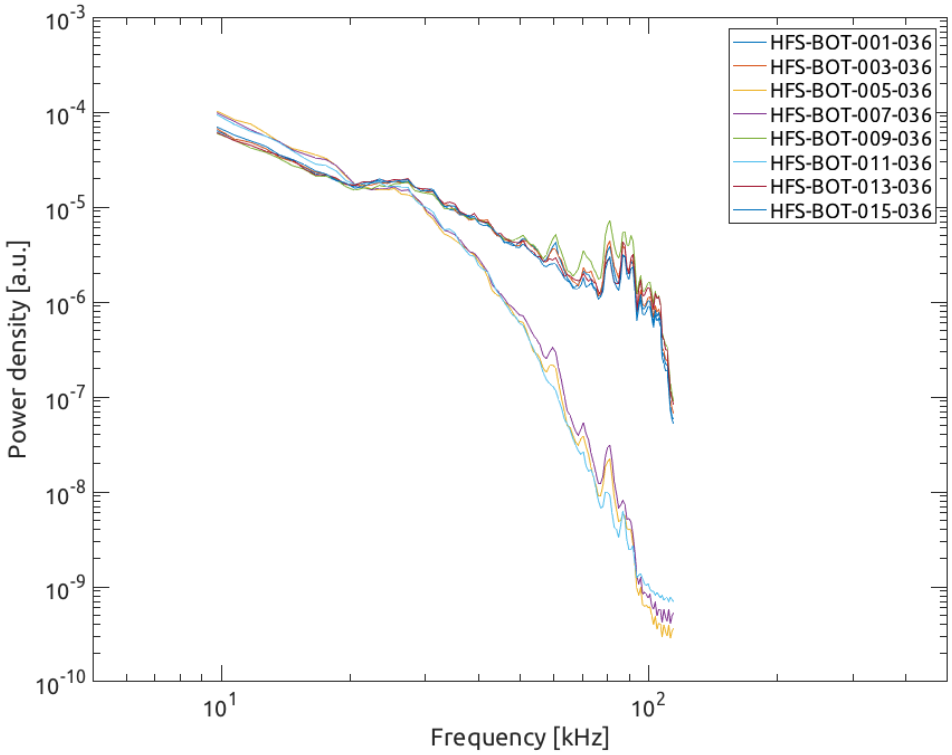

Figure 4.2: Example of PSD plot computed for eight signals acquired during TCV shot #55597.

is discriminated form noise. In this way, signals are characterized as having an information content, which noise does not have.

The foundation of the tool is the construction of the so-called record of time increments. This is obtained taking the difference between the signal and its value at a delayed time coordinate, for all instants of the chosen time interval:

$$\Delta \mathbf{b} = \mathbf{b}\left(t+\tau\right) - \mathbf{b}\left(t\right). \tag{4.10}$$

The time increments are then normalized to their standard deviation. Thus, Δb is used to denote the normalized quantities. Notice that the exploited component of the vector is the poloidal one.

The next step is the computation of the frequency of occurrence of all normalized increments. This occurrence frequency is assumed to be the experimentally obtained PDF for the analyzed time series. The results of the analysis are obtained through the study of this PDF, which is compared to the one given by a noise signal.[17, 39] The PDF of noise is Gaussian shaped, since after any delay upward and downward changes are equally expected for a truly random signal. Such a PDF is not expected to appear as result of the analysis of TCV signals.

In order quantitatively comparison two moments of the PDF are computed, namely: kurtosis, K, and skewness, S. The former one is also called statistical metric flatness and it is a measure of the distribution's wideness. It is mathematically defined as follows:

$$K\left(\tau\right) = \frac{\langle \Delta b^4 \rangle}{\langle \Delta b^2 \rangle^2}. \tag{4.11}$$

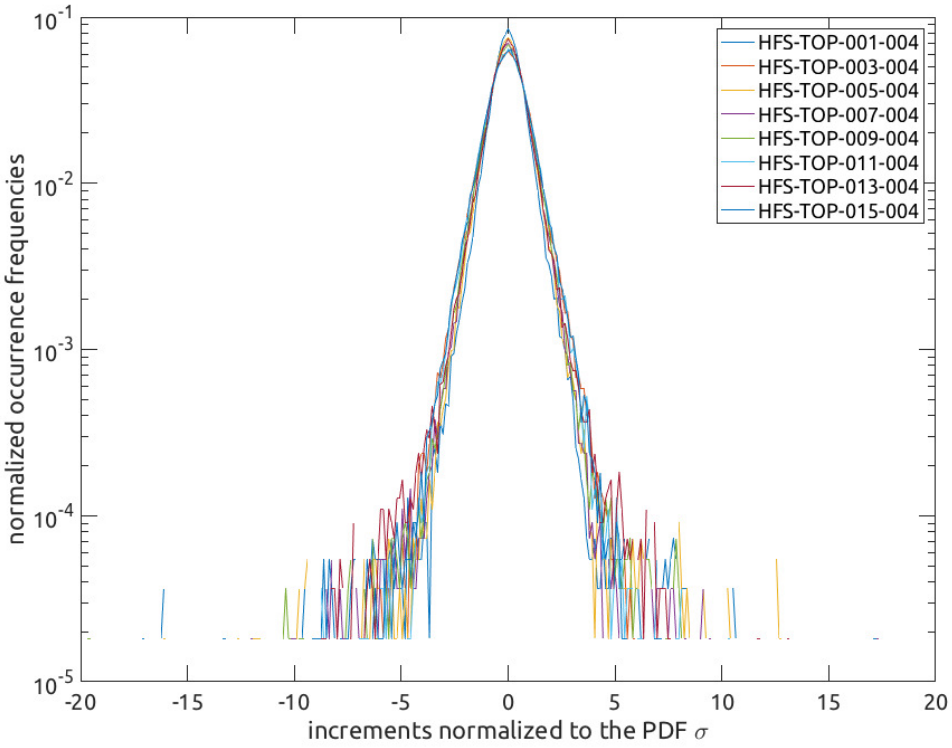

Figure 4.3: Example of PDF computed for eight signals acquired during TCV shot #55597.

For a Gaussian curve $K = 3$.
The skewness measures the symmetry of the PDF with respect to the most probable increment, that is represented as the maximum of the curve. When $S > 0$ higher variations are more frequent over the given time interval, whereas lower increments are more frequent when $S < 0$. For this work the maximum of the curve is found at zero normalized increments and, therefore, for $S > 0$ more positive variations are found, while more negative ones are observed in the opposite case. For a symmetric distribution, as the Gaussian one is, $S = 0$, meaning that over the selected time interval the distribution of increments is perfectly symmetrical. The mathematical definition of skewness is:

$$S(\tau) = \frac{\langle \Delta b^3 \rangle}{\langle \Delta b^2 \rangle^{\frac{3}{2}}}. \tag{4.12}$$

The comparison of the experimental K and S values with the ones of the Gaussian distribution allows for the discrimination of the experimental signals from noise.

The presence of the non-Gaussian flat tails is related to large variations in the signals, or even to discontinuities. These discontinuities are associated to the presence of coherent structures in the flow, such as plasma flux tube and current sheets.[17]

A coherence check between the results of this tool and the previous one is performed. In fact, a noise PDF is foreseen when a noise PSD is obtained, and vice versa.

4.4 Temporal structure functions

In this section, an introduction to the temporal structure function (TSF) is given.

The aim of this tool is the detection of dissipation processes acting over given time scales. Moreover, thanks to the TSF it is possible to compute the *Hurst*'s exponents related to the signal. This topic, however, is covered in Sec. 4.5.

A TSF is computed as the average the record of time increments elevated at some power; mathematically:

$$\mathbf{S}^{p}{}_{\mathrm{b}} = \langle (\mathbf{b}\,(t+\tau) - \mathbf{b}\,(t))^{p} \rangle\,. \tag{4.13}$$

Again, in the frame of this analysis only scalar quantities are taken into account. It easy to see that both kurtosis and skewness of a PDF are properly normalized TSFs.

The computation is carried out for multiple time lags and multiple powers; the latter are usually referred to as orders of the TSF. The results are then plotted on a log-log plot, as in Fig. 4.4. For positive powers, the values of the TSF increases with an increase in the order order. It is important to perform the calculation of multiple powers because at higher orders the oscillations of the TSF are accentuated. This allows a better observation of the results. Since energy injection is related to local maxima of the TSF, whereas dissipation is linked to local minima, it is easy to understand that higher powers accentuates these distortions of the TSF. Multiple energy injections and/or dissipations can be found at different time scales.

If turbulence is fully developed, as seen in Sec. 2.3, an inertial range is established over a proper time scale interval. This range correspond to a linear non-flat behavior of the TSF on log-log plot. Over this range the TSF is expressed by a power law, as follows:

$$S^{p}{}_{\mathrm{b}}\,(\Delta s) \sim \Delta s^{\zeta(p)}. \tag{4.14}$$

This result comes from hydrodynamic turbulence theories and it is expressed as function of length space. However, thanks to Eq. 4.4, the same scaling is found in time coordinates, i.e.:

$$S^{p}{}_{\mathrm{b}}\,(\tau) \sim \tau^{\zeta(p)}. \tag{4.15}$$

Multiple values of $\zeta\,(p)$ are predicted by the turbulence models discussed previously in Sec. 2.3. However, for this analysis two are taken as reference:

1. ***K41.*** In this case $\zeta\,(p) = p/3$.
2. ***IK.*** In this other case $\zeta\,(p) = p/4$.

Notice that both predictions are linear functions. This is due to the fact that no dissipation is taken into account during the inertial range. Therefore, a departure from linearity indicates the presence of coherent structures over the inertial range hence allowing dissipation to take place. However, little information on the origin of these structures is obtained from this analysis. In fact, characteristic time scales, and therefore frequencies, are obtained. Form this piece of information a guess on the nature of these structures might be performed.

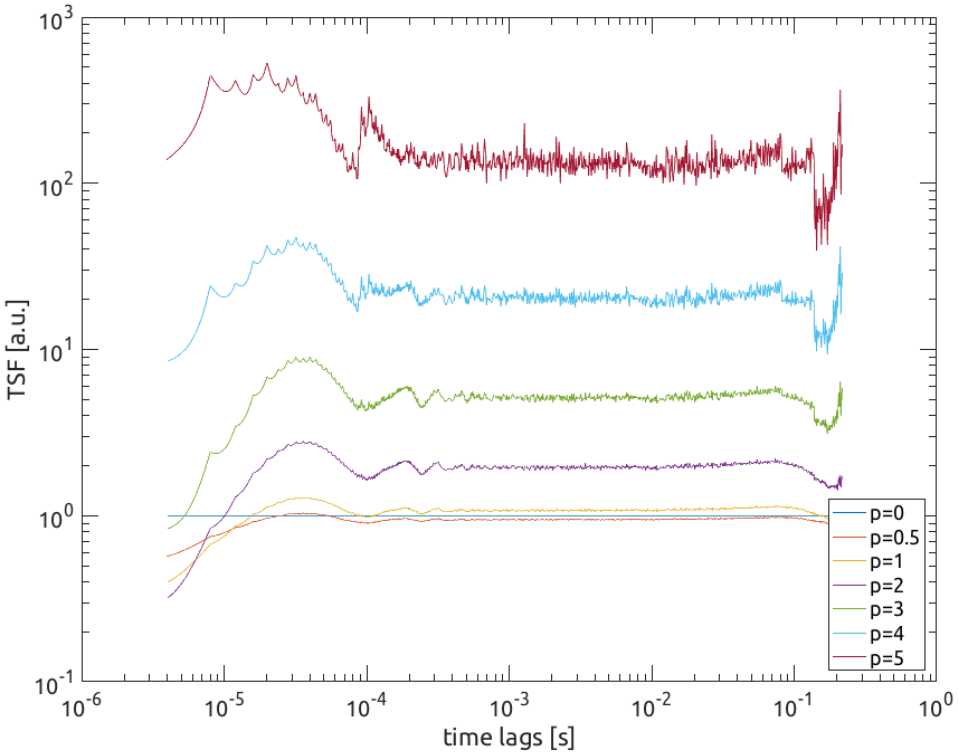

Figure 4.4: Example fo TSF plot computed from the HFS-MID-009-001 signal acquired during TCV shot #55597. Multiple orders and time lags are explored.

Another coherence check is performed. The asymptotic behavior of $\zeta(p)$ at low orders, typically $p < 2$, is compared with the theoretical references. For signals that do not follow one of the PSD exponents of reference, a different linear behavior at low orders is observed.

4.5 *Hurst*'s exponents and fractal dimensions

In this section, an introduction to the *Hurst*'s exponents and fractal dimensions analysis is given.

The *Hurst*'s exponent, also called self-similarity scale exponent, is a figure of merit that is exploited for the evaluation of the time persistence of a signal.[40] This characteristic is also called long-time correlation. This quantity is denoted as H and there are multiple mathematical definitions for it. For this work the following is adopted:

$$H_p = \frac{\zeta(p)}{p}. \tag{4.16}$$

It is now clear why the computation of the TSF is necessary for the calculation of H in the frame of this work. However, other methods for the computation of this exponent are exploited in literature. Among these, the most common is the re-scaled range statistics method (R/S).[41–43] H is a quantity usually comprised between 0 and 1. If H_p is an almost constant function of p, the signal is said to be mono-fractal. Otherwise, the signal is characterized as being multi-fractal and

a spectrum of H_p values is needed in order to describe it. See Sec. 4.4 for more information on the temporal structure function.

It is easier to understand the nature of H from the concept of fractional Brownian motion (fBm) and the role that the exponent has in the definition of its underlying function.[44] The fBm is a generalization of the Brownian motion function, which is also called Wiener process. An fBm signal is characterized by two important features:

1. **Self-similarity.** As already stated, this is the characteristic of shape retention of a signal over different scales. Mathematically, for a fBm designated as $B_H(at)$:

$$B_H(at) \sim |a|^H B_H(t), \tag{4.17}$$

   where H is the *Hurst*'s exponent and a is an arbitrary scaling factor.

2. **Stationarity of increments.** This means that the value of $B_H(t)$ translated through time is given by the difference between the value in the two points:

$$B_H(t_2) - B_H(t_1) \sim B_H(t_2 - t_1). \tag{4.18}$$

Both fBm and Bm are computed on a step basis, meaning that the evolution of their functions are computed step-by-step. The characteristic of a Bm signal is that two subsequent variations of its value are uncorrelated. This means that each variation is completely independent from all the others. For a fBm this is not the case, meaning that two subsequent steps does not need to be uncorrelated. The type of correlation is described by H_1, that is the first *Hurst*'s exponent. H_1 is computed form the firs order TSF, as it can be seen from Eq. 4.16.

For $H_1 < 0.5$, the signal is said to be anti-correlated. This means that each variation is likely to occur in the opposite direction that the the previous one had. An anti-correlated signal tends to oscillate around its mean value. For $H_1 > 0.5$ the signal is said to be correlated, or persistent. This mean that each variation is likely to occur in the same direction that the previous one had. A persistent signal diverges from its local average values. The third case is found for $H_1 = 0.5$. This is the exponent typical of Bm signals. Thus, fBms having this H_1 value are Bm signals, which are not correlated. The last case is for $H_1 = 0$. These signals are said to be stationary, since they present an oscillatory behavior around a global mean value. In Fig. 4.5 five fBm signals are shown. It is interesting to notice the various anti-correlation and persistence features of the plotted signals.

Through the computation of H it is possible to state that the analyzed signal presents the same properties of a fBm that is defined form the very same exponent.

The *Hurst*'s exponent is also exploited for the computation of the fractal dimension of the signal, denoted as D.[45] Whereas H is a global property of the signal, D is a local characteristic. In fact, the fractal dimension of a system describes the local dimensions of the underlying structure. When the signal is self-similar, the following holds true:[46, 47]

$$D + H_1 = n + 1, \tag{4.19}$$

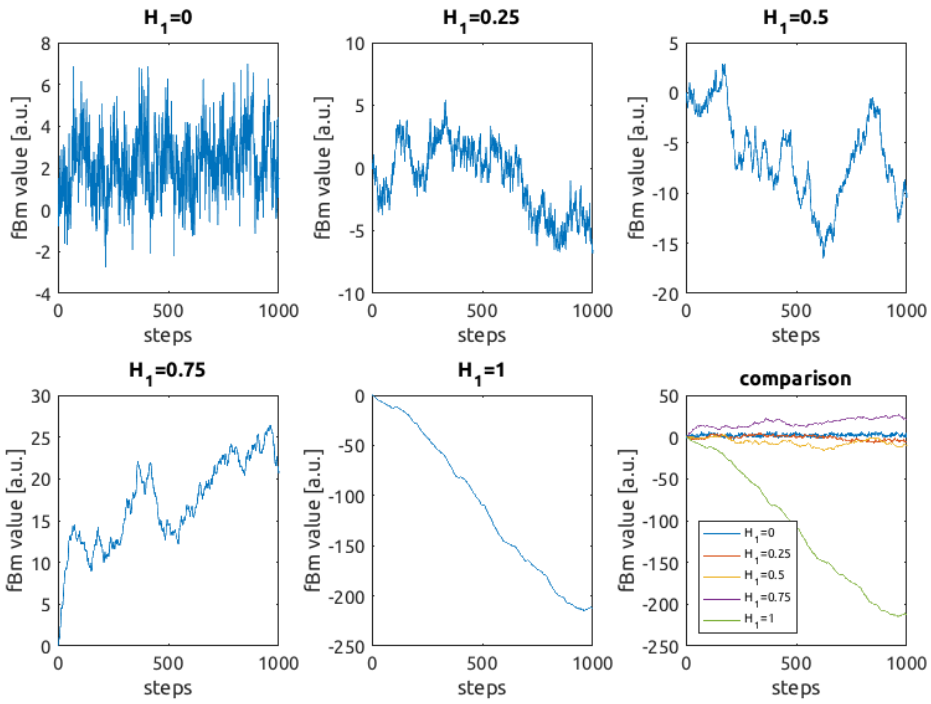

Figure 4.5: Five examples of fBm signals having different H_1 values. Notice how the first signal oscillates around a mean value that is set around 2. The signal at H_1 is actually a Bm signal.

where n is the number of dimensions exploited for the description of the system. Since H_1 is generally bounded to the interval $[0, 1]$, then $D \in [n, n+1)$. Therefore, the fractal dimension gives information about how many dimensions are needed for the local description of turbulent structures. This is useful in order to construct the equation describing such phenomena.

4.6 Permutation entropy and complexity

The aim of this section is the presentation of the last tool that is exploited for this analysis of edge magnetic field turbulence.

The study of ordinal patterns of the values of a time series gives another level of insight on the physics that generates a signal. Two figures of merit of time series are computed as a result of the application of this tool, namely: permutation entropy, $H[P]$, and complexity $C_{\mathrm{JS}}[P]$.[17, 48, 49] In order to compute these, a PDF of the ordinal patterns of the signal need to be calculated. This is constructed as an occurrence frequency, as in Sec. 4.3.

Taking into account N subsequent points of the time series, it is possible to establish their ordinal pattern. This is the pattern of relative variations between the two subsequent points among the chosen ones. For N points there are $N!$ possible permutations of their relative order. In Fig. 4.6 three possible permutations for $N = 5$ are shown.[17] Once N, called embedding dimension, is set, then the occurrence frequency of each permutation is computed. Each frequency is designated

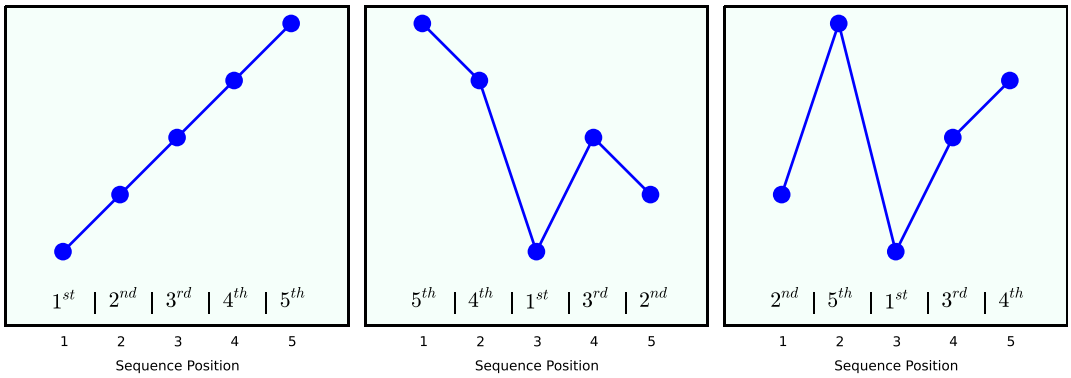

Figure 4.6: Example of possible permutations for $N = 5$.[17]

as P_i, while the complete PDF is indicated as P. The entropy of P is computed following the *Shannon*'s approach and it is, therefore, called *Shannon* permutation entropy. Mathematically it is computed as follows:

$$S[P] = -\sum_{i=1}^{N!} P_i \log_e (P_i), \tag{4.20}$$

where the summation is performed over all possible permutations. A completely stochastic time series ,for which the number of elements is much larger than the embedding dimension, all occurrence frequencies are equal. This means that $P_1 = 1/N!$. In this case the permutation entropy is maximized and is given by:

$$S_{\max}[P_e] = \log_e (N!), \tag{4.21}$$

where P_e is the PDF that has been just described. Since S has a theoretical maximum value for each N, a normalized entropy value is introduced:

$$H[P] = \frac{S[P]}{S_{\max}[P_e]}. \tag{4.22}$$

An example of null entropy signal is given by a linear ramp, whereas $H = 1$ is obtained from a white noise signal.

The second parameter, that is the complexity, is computed from the normalized entropy. The *Jensen-Shannon* complexity is given by:

$$C_{JS}[P] = H[P] Q[P], \tag{4.23}$$

where $Q[P]$ is a normalization factor that describes how far a PDF of ordinal patterns is from the related P_e. The factor is given by the following:

$$Q[P] = S\left[\frac{P + P_e}{2}\right] - \frac{S[P]}{2} + \frac{S[P_e]}{2}. \tag{4.24}$$

It is interesting to notice that both the examples previously used share the same complexity value, which is null.

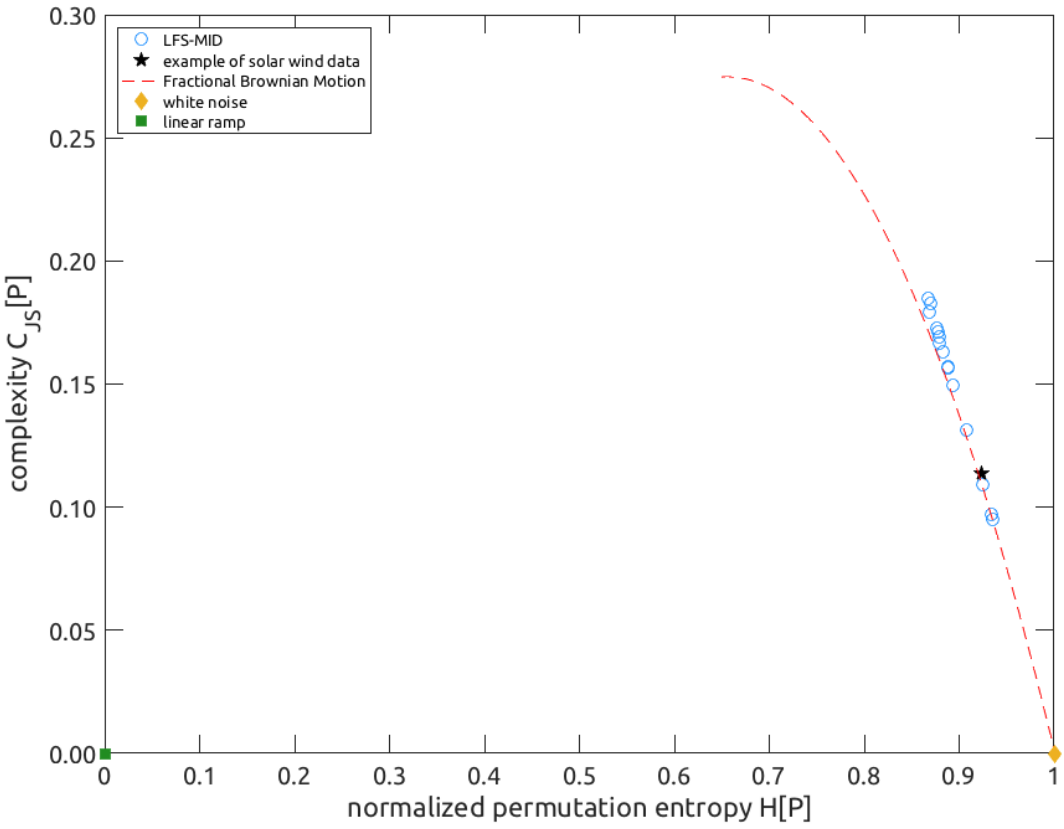

Figure 4.7: Example of CH plot, computed for fifteen signals acquired with the LFS-MID probes during TCV shot #55597. The dashed line is related to varying fBm signals, while the star point is an example of solar wind data result. Two reference points, related to the linear ramp and white noise, are also shown.

The complexity is a measure of the 'non-triviality'of the distribution of the ordinal patterns of time series. The more complicated is the function describing a signal, the more complex this is.

These two quantities are exploited in order to build a plot, called CH plot. On its plane a signal has solidly defined coordinates that are bound to an area of theoretically allowed set, which is computed for each N value. In Fig. 4.7 an example of CH plot is shown. The result of the analysis on fifteen signals is shown in the plot. Alongside this, two reference points, located at the coordinates $(0, 0)$ and $(1, 0)$ are plotted. These are, respectively, the linear ramp and white noise entropy-complexity features. A solar wind data point is also shown, as well as the line drawn by fBm signals having different H_1. In Fig. 4.8 another CH plot is shown. In this case the entropy and complexity features of various mathematical functions are reported.

This analysis is particularly interesting because it allows to determine the function that describes the signal. Since a known function gives a particular set of coordinates then, if an experimental signal has the very same entropy and complexity properties, the link between the two is legit. While, for example, the *Hurt*'s exponent analysis enables the comparison of long term correlation between

the studied signal and a given fBm, the CH plot allows to infer that the signal is described by the very same fBm.

4.7 An astrophysical reference: the solar wind turbulence

In this section some results regarding the analysis of solar wind turbulence are presented. The statistical analysis of magnetic turbulence regarding solar wind data is a well establish practice. This is the reason why a brief introduction on this topic is given. The article in reference (*17*) is used as the basic tutorial upon which the developed analysis toolbox is created.

The solar wind is considered to probably be "the best studied turbulence laboratory".(*15*, *17*) The solar wind is an high velocity, low density hydrogen plasma. In fact, typical values are $u \geq 400\,\mathrm{km/s}$ and $n \sim 10\,\mathrm{cm}^{-3}$.(*17*) At the distance of 1 AU from the Sun the characteristic magnetic field is $B \sim 10\,\mathrm{nT}$.(*17*) The solar wind's plasma beta parameter is of order unity, indicating that neither kinetic nor magnetic pressure dominate the dynamics.(*17*) Typical temperatures are of the order of 10 keV, with $T_\mathrm{i} \geq T_\mathrm{e}$.(*17*) The mean-free path for collisions between different particles populations is of the order of 1 AU, thus the solar wind is considered to be a collisionless system.(*17*) Turbulence in the the solar wind flow is fully developed since a turbulent cascade is established for all MHD dynamical physical quantities, such as $\mathbf{u}$, n and $\mathbf{B}$.(*17*) Solar wind turbulence is usually studied near planet Earth, but studies that involved spacecraft sent at a distance of up to 120 AU are available.(*17*) Both temporal and spatial statistical analysis are performed with spacecraft acquired data. However, to perform the latter, an array of satellites is necessary. One of such arrays is deployed in the frame of the Cluster II space mission, jointly operated by ESA and NASA.* To have a more exhaustive picture of solar wind MHD turbulence, please refer to (*15*).

In Fig. 4.9 an example of solar wind magnetic waveform is shown. This time series, that is the sun-ward component of the magnetic field, is acquired by the Wind satellite, located at the L1 Lagrangian point between the Sun an planet Earth.(*50*) A period of very fast wind stream ($\geq 600\,\mathrm{km/s}$) is shown. The data is acquired over multiple days, namely January 14-21 2008. These type of data usually shows stationarity, meaning that any average quantity is independent of the choice of time origin. Turbulent analysis is performed over stationary time periods in order to exploit time averaging procedures in place of ensemble averaging, as explained in Sec. 2.3.

In Fig. 4.10 a typical example of auto-correlation function for data coming from the Wind satellite is shown. Considering a threshold value of e^{-1} for the computation of τ_c, the calculated value is $\tau_\mathrm{c} = 0.54\,\mathrm{h}$.(*17*) Taking into account an estimated value of the flow velocity of $u = 600\,\mathrm{km/s}$, this de-correlation time gives a correlation length of about $1.17 \cdot 10^6\mathrm{km}$, which is of the order of the values obtained from actual two-point correlation length computations.(*51*)

In Fig. 4.11 an example of PSD plot is given for other data acquired by the

*See `http://sci.esa.int/cluster/` for further information.

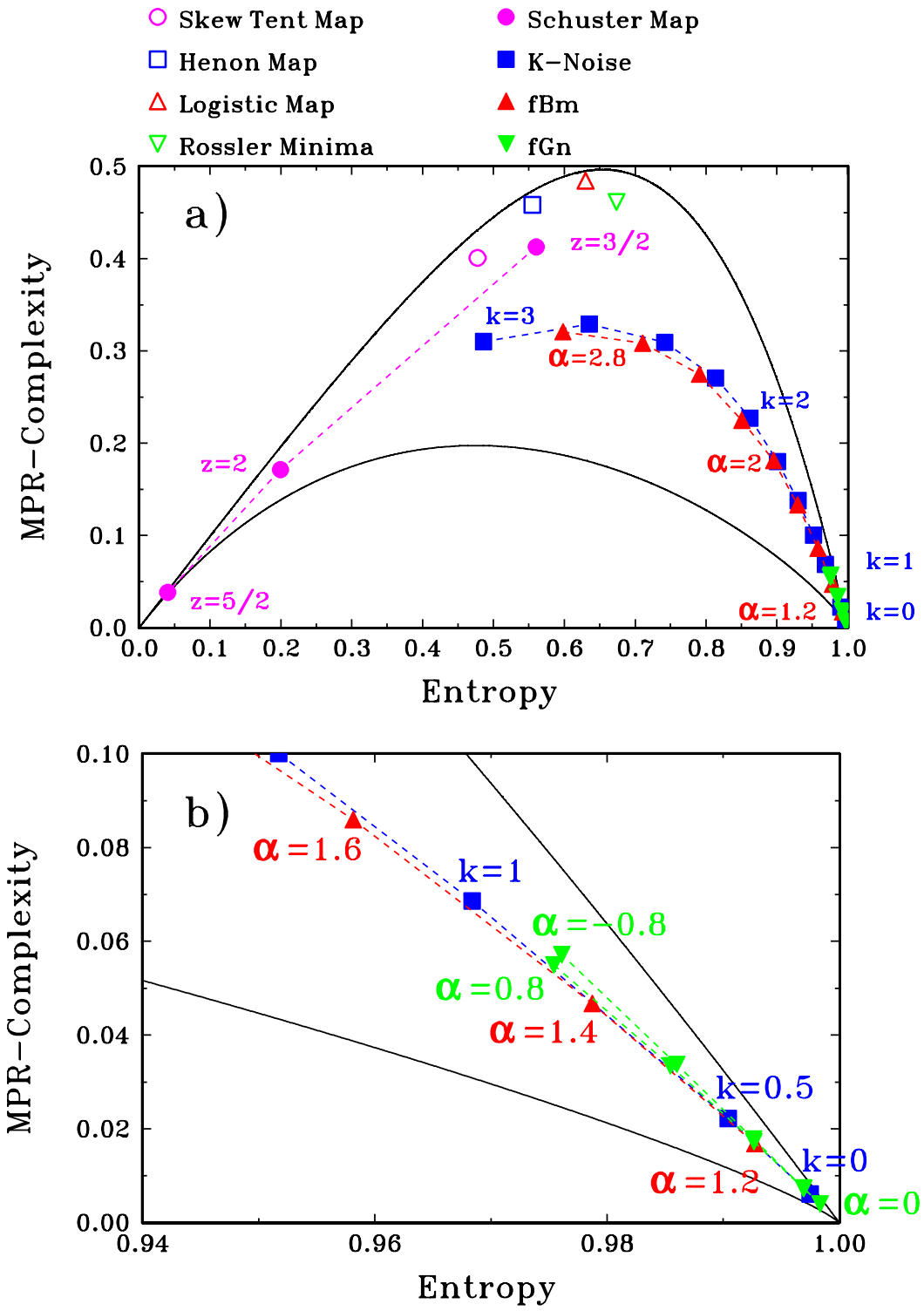

Figure 4.8: In the plots the CH coordinates of multiple mathematical functions are drawn. In b) the area around the maximum entropy area is enlarged.[49]

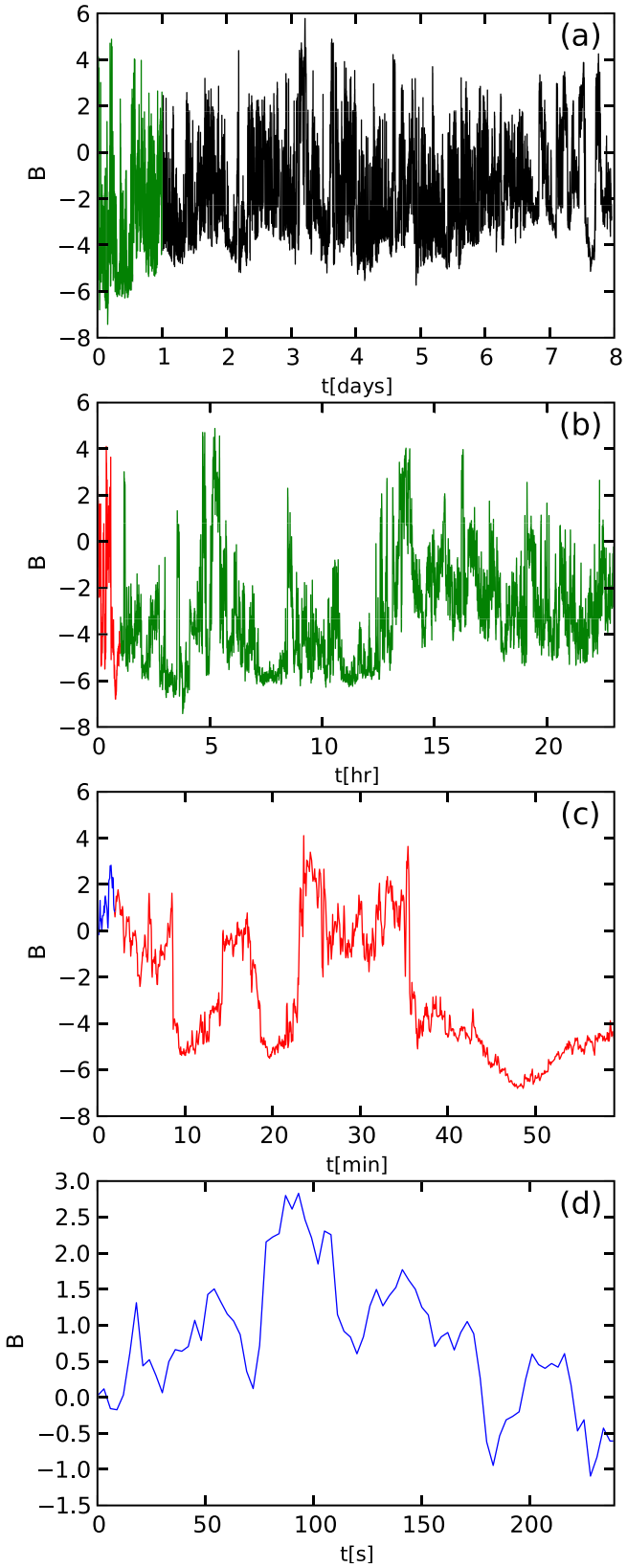

Figure 4.9: Magnetic field waveforms acquired by the Wind satellite. The signal is the sunward component of the magnetic field measured in nT. The colors indicate the data enlarged in the next plot. a) Eight days acquisition period. b) Twenty-four hours period. c) One hour period. d) 4 minutes period.[17]

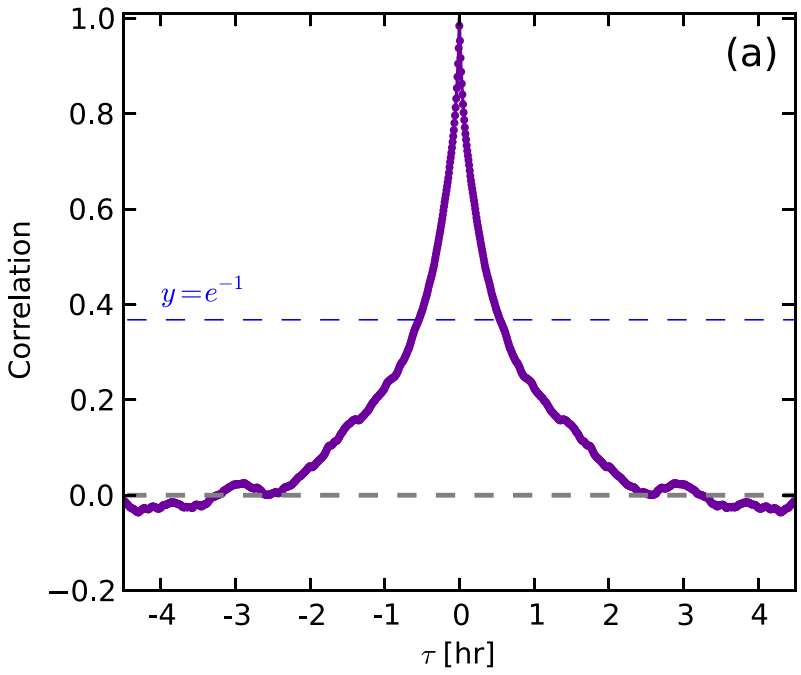

Figure 4.10: Auto-correlation function computed from a time series acquired with the Wind satellite.[17]

Wind spacecraft. Although a fast *Fourier* transform is usually exploited to compute the power spectrum, here it is computed through a wavelet transform.[17, 52] Notice the inertial energy cascade, which follows the famous *Kolmogorov*'s prediction. This feature is found over twelve orders of magnitude of the considered scale, either wave-numbers or frequencies, and it is sometimes referred to as "the Great Power law in the Sky".[53] Even though the $-5/3$ scaling is found in many research publications, it is often found that MHD turbulence is actually anisotropic with respect to the magnetic field, with perpendicular fluctuations having an higher energy content.[53, 54]

In Fig. 4.12 multiple PDF plots are shown. Notice how the distribution regresses towards the Gaussian one as the time delay used for the computation increases. This means that coherent structures are revealed when the analysis is performed only on short enough time scales, based on the actual de-correlation time. In other words, any analysis performed with time delays that are much larger than τ_c cannot discriminate the signals from pure noise.

In Fig. 4.13 and example of TSF computation is shown. Here the function is computed for multiple time lags and orders. In red the inertial range is highlighted; notice the non-flat linear behavior. In Fig. 4.14 the evolution of the related slope is shown. In this case the *K41* behavior is lost at $p \approx 2$, but other authors have computed TSFs up to order 20, showing a departure only at $p = 6$.[55]

In Fig. 4.15 an example of CH plot is shown. Notice the solid black lines that define the area of allowed coordinates. In the plot some reference point are shown, namely: chaotic skew tent, Henon, and logistic maps.[49] It is interesting to point out how the experimental results drift toward the pure noise coordinates, that are

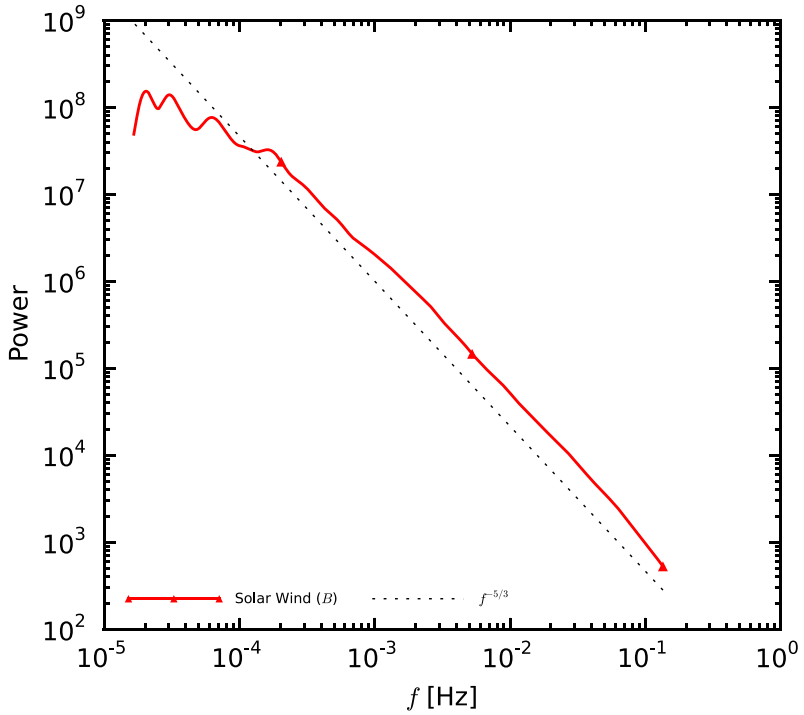

Figure 4.11: PSD computed from a time series acquired with the Wind satellite.[17] Notice the *K41* scaling drawn as a dashed line.

$(1, 0)$, as the selected embedding dimension increases. This is in accordance with the PDF analysis, meaning that an higher embedding dimension translates in a longer time delay for the computation of the PDF.

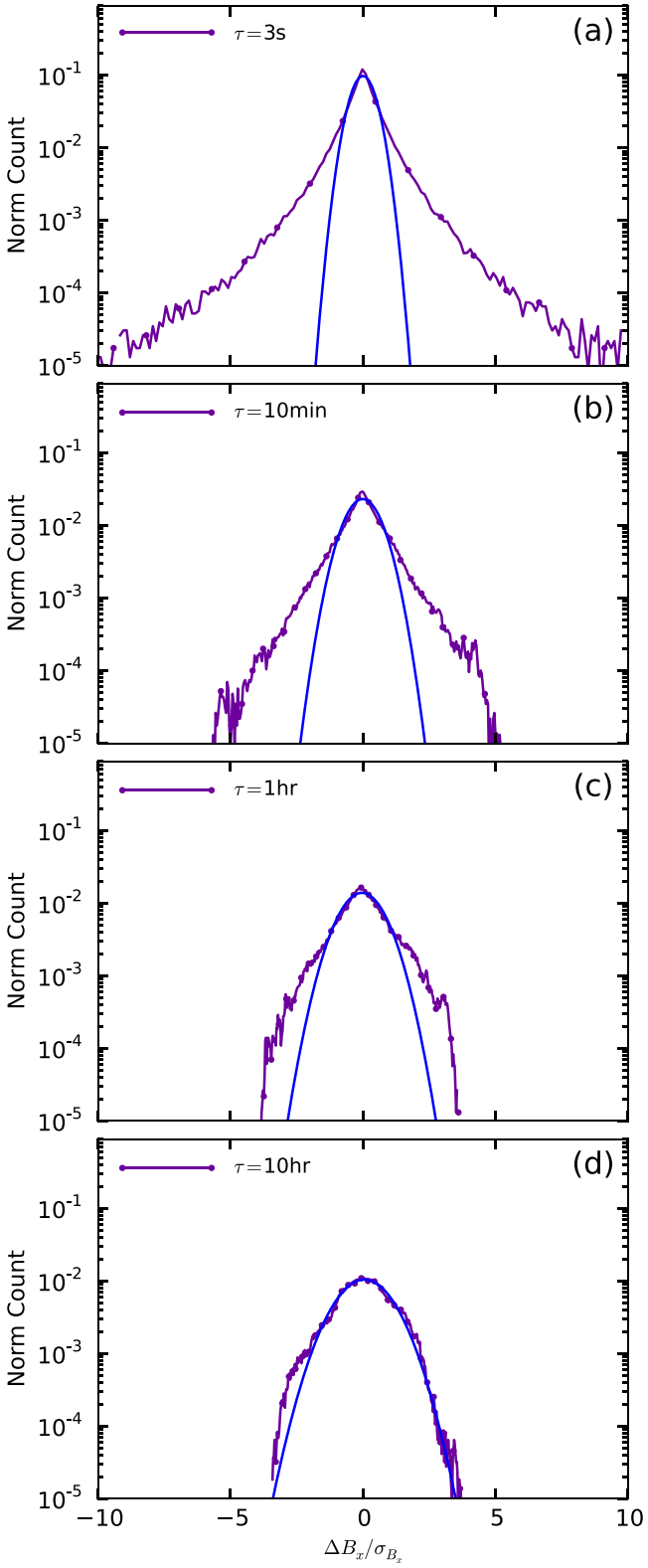

Figure 4.12: PDFs of increments computed for multiple time lags. notice how the PDF is much larger at low time lags, indicating the presence og coherent structures.[17]

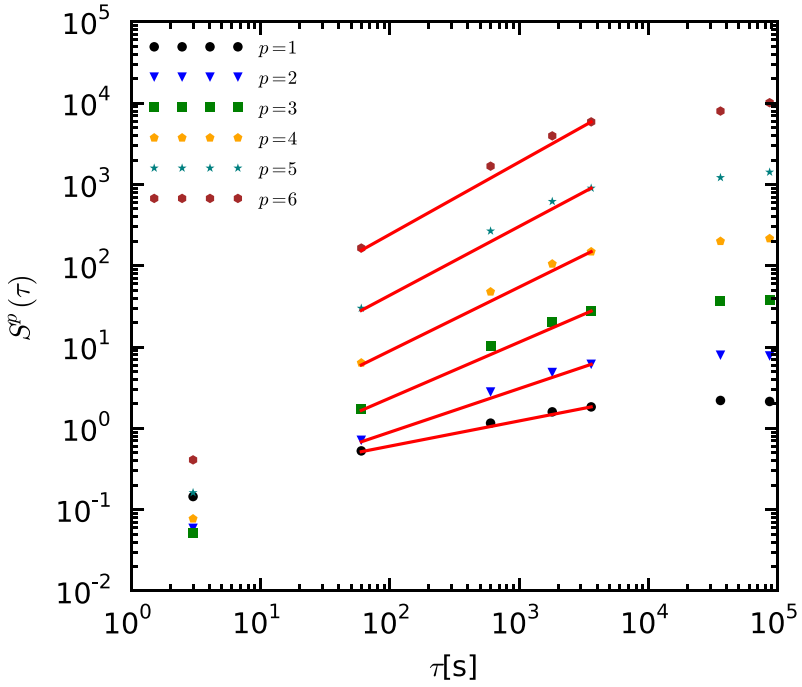

Figure 4.13: TSFs computed from a time series acquired with the Wind satellite.[17] Notice the inertial range highlighted in red.

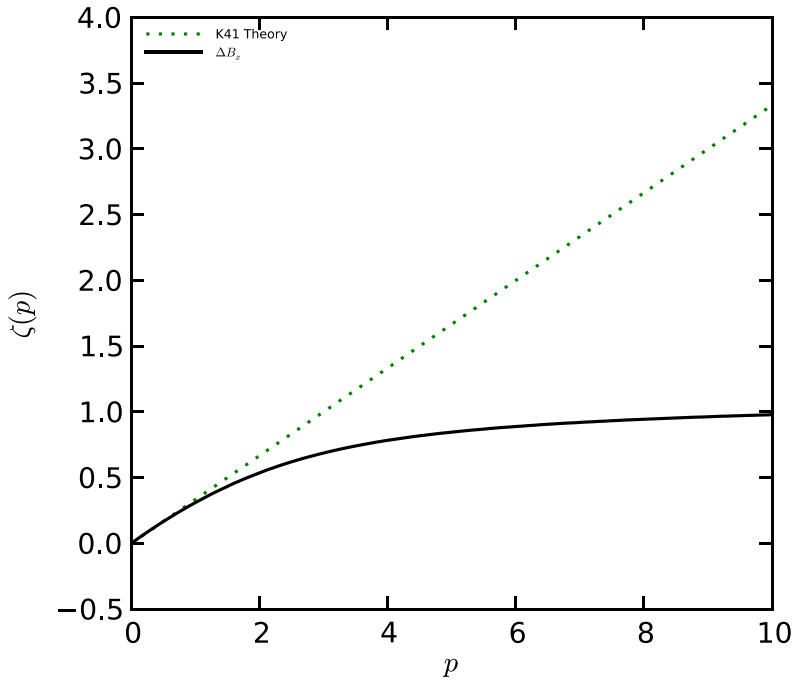

Figure 4.14: The slope of the TSF over the inertial range are here shown.[17] Notice how the experimental behavior deviates from the *K41* prediction at $p \approx 2$.

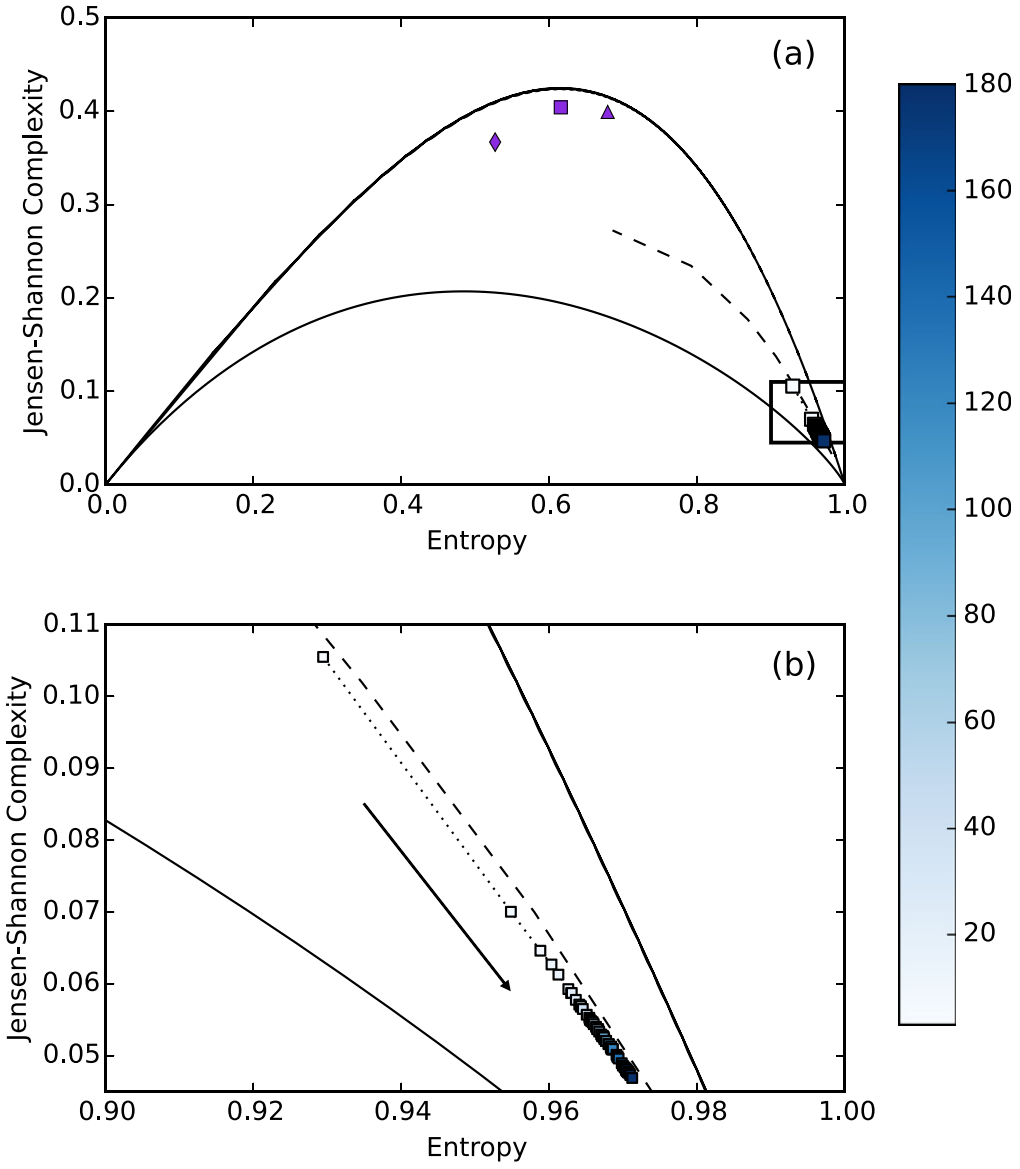

Figure 4.15: In a) the whole CH plot is shown. The solid black lines delimits the area of allowed coordinates, while the dashed one is related to fBm signals. The purple markers are the chaotic skew tent (diamond), Henon (square), and logistic (triangle) maps. The boxed region is enlarged in b). The arrow indicates the direction of increasing embedding dimension, or time delay.[17]

Chapter 5

Magnetic turbulence analysis on TCV plasma discharges

In this chapter the results of the application of the analysis tools presented in Ch. 4 is given. The analyzed experimental data comes from multiple plasma discharges, also referred to as shots. The chosen shots are part of two different TCV experimental missions, which are mission #1531 and #1594. These missions have different goals: the former focuses on L-H transition physics, while the latter studies core electrons' temperature turbulence.[56, 57] The choice of these mission is made because they offer a set of coherent experimental data that enables a proper test of the developed algorithms. This means that plasmas having similar characteristic are studied in the frame of a single mission. Therefore, consistent results are obtained from different plasma discharges of a single mission. Twenty-seven plasma discharges are taken into account in this work, amounting to several GB of data. Notice that each computation is performed on the signals coming from all the probes in a independent manner. This means that the analysis can be performed on an arbitrary number of probes at the same time. For this work all the probes are always taken into account; this is also the reason why the amount of analyzed data is huge.

In Sec. 5.1 an explanation of the analysis strategy of the experimental data is given. In particular, it is presented how the data is chosen and how it is processed before the actual computation.

In Sec. 5.2 the first part of the results are presented. Here the more basic analysis are taken into account, namely: auto-correlation, PSD and temporal increments.

In Sec. 5.3 the more advanced analysis are presented. In fact, the last three tools are taken into account, namely: temporal structure functions ,*Hurst*'s exponents, and CH plot.

Please, for further information on the actual implementation of the code refer to App. A.

5.1 Data selection and processing

The aim of this section is to explain how the whole data relative to one plasma discharge is divided into multiple subsets and how these are processed before the actual analysis.

The shots of mission #1531 are characterized by multiple plasma current plateaus, which are set in order to try achieve the L-H mode of confinement transition. There are two possible mode of operation for a tokamak plasma. These are the low confinement mode (L) and high confinement mode (H). Qualitatively, the two differ in the established transport behavior, with τ_E in the H case being as high as two times the one typical of L-mode plasmas.[8] This is, therefore, a favorable situation towards the goal of sustaining burning plasma. However, the H-mode of operation shows the presence of edge instabilities known as edge localized instabilities (ELMs). Practically, these act as pressure relief valves, stabilizing the edge plasma pressure. ELMs are driven by the edge pressure gradient; when this becomes too large, one of such events is initiated. There are three types of ELM-y H-mode of operation. These are characterized by different frequency of occurrence and violence of ELM events. An ELM-free H-mode of operation is also possible. Eight plasma discharges are taken from this mission. All these shots are located in the center of the TCV vacuum vessel. The relative data is sliced in multiple subsets, called time windows. These are chosen in order to perform the analysis over a plasma current plateau, because a changing current leads to a change in the poloidal magnetic field. Since this is the physical quantity of interest, this choice guarantees to perform the acquisition in a close to steady-state situation. This is important in order to perform unbiased averaging. A total of ten time windows are selected, with up to eight . For the duration of a time window similar plasma properties are shown

The shots of mission #1594 are peculiar for at least a couple of reasons: these are performed at different vertical positions inside the vacuum vessel of TCV, and they are characterized by different values of triangularity. Specifically, the triangularity values are both positive and negative. The latter configuration is of particular interest since such a divertor configuration would be characterized by a larger divertor surface with respect to the positive case, because of the toroidal geometry. Maintaining fixed plasma parameters, a larger area means lower heat power fluxes on the divertor itself, which is a advantageous feature. Nineteen plasma discharges are taken from this mission and only one time window is selected for each. This is done in order to perform the analysis on multiple plasmas having similar density values. This allows to search for any difference in the results, that can be due either to a different plasma position or to a change in triangularity.

Once the data is selected, it is processed in order to prepare it for the analysis. First, the acquired edge magnetic field is manipulated in order to obtain another quantity, which is the normalized time-varying part of the signal. This procedure is introduced at the beginning of Ch. 4. Then, any drift in the time series is individuated and removed, through the computation of a polynomial fit. This is done in order to remove any residual bias that there might be in the data set.[58] Finally, the signal is filtered using a Kaiser window filter. The default window

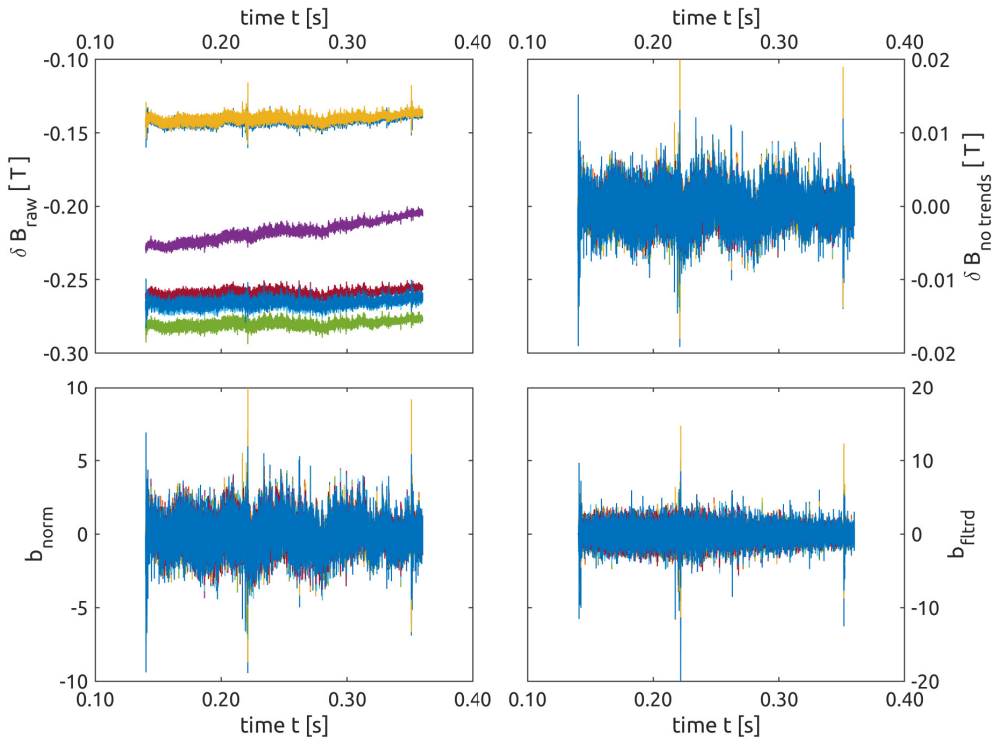

Figure 5.1: Example of signal processing. Here eight signals acquired from the TCV shot #55597. In the first plot the raw data is shown. Then, from the second plot to the last one the signal is shown after: trends removal, normalization and filtering.

is designed for the exclusion of unwanted features, such as: low frequency MHD coherent structures, power supply noise, and all those harmonics already distorted by the cutoff frequency of the measurement system.

In Fig. 5.1, an example of signal processing is shown.

5.2 Basic results

In this section the results of the first three analysis tools are presented. An example related to a single shot is given for each statistics. Then, global averages are taken into account. With this concept an averaging procedure over all the probes is intended. Thus, a global average parameter is obtained over a time window for each analysis and for each shot. The estimate of this mean value is improved via an outliers elimination procedure. This means that once the global average is computed, all signals, related to a result that is farther than 3σ from it, are excluded form the averaging procedure. Then a new global mean value is calculated. This can be arbitrarily repeated; however, for this work such refinement procedure is done twice. Notice that once a signal is found to be invalid for one analysis tool, it is excluded also form the other ones that are computed during the same call of the code.

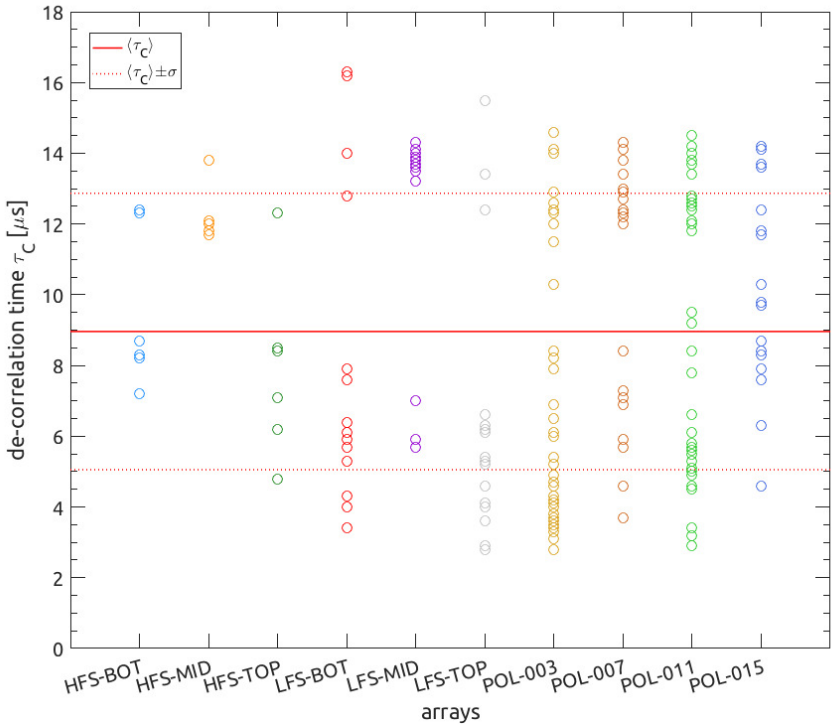

Figure 5.2: De-correlation times for all the signals for the L-mode current plateau of TCV shot #55597.

5.2.1 Auto-correlation function

Since the measurements are relative to a single component of the magnetic field, the computation of the auto-correlation function, as presented in Sec. 4.1, is straightforward. In fact, only scalar quantities have to be dealt with.

As anticipated in Sec. 4.1, there is a degree of freedom in the choice of τ_c, which for this work is defined as shortest the time delay after which the value fo the auto-correlation function drops below a selected threshold. This factor is chosen to be e^{-1}. In order to improve the robustness of the search algorithm, the choice of τ_c is bounded to the first section of $R(\tau)$, which is comprised between the global maximum and the first local minimum of the function. Since $R(\tau)$ is symmetric with respect to the vertical axis, the first local minimum is found restricting the analysis to the positive lag times part of the function and imposing a control on the value of the function's derivative. Thanks to these restrictions, the first threshold crossing is consistently found.

In Fig. 5.2 an example of computed de-correlation times is shown. The underlying data comes form TCV shot #55597, which is part of the first considered mission. The selected time window spans over a L-mode current plateau set at $I_p = 210\,\text{kA}$. This result is representative of all the analyzed shots and from this figure a couple of interesting comments are obtained.

The first one is that the results show some degree of sparseness. This may be due to various causes, however, the main one is a possibly poor signal to noise ration. This is demonstrated with the last analysis tool, that is the CH plot. In Sec. 5.3.2 the CH plot of this very time series is presented and a comment regarding the

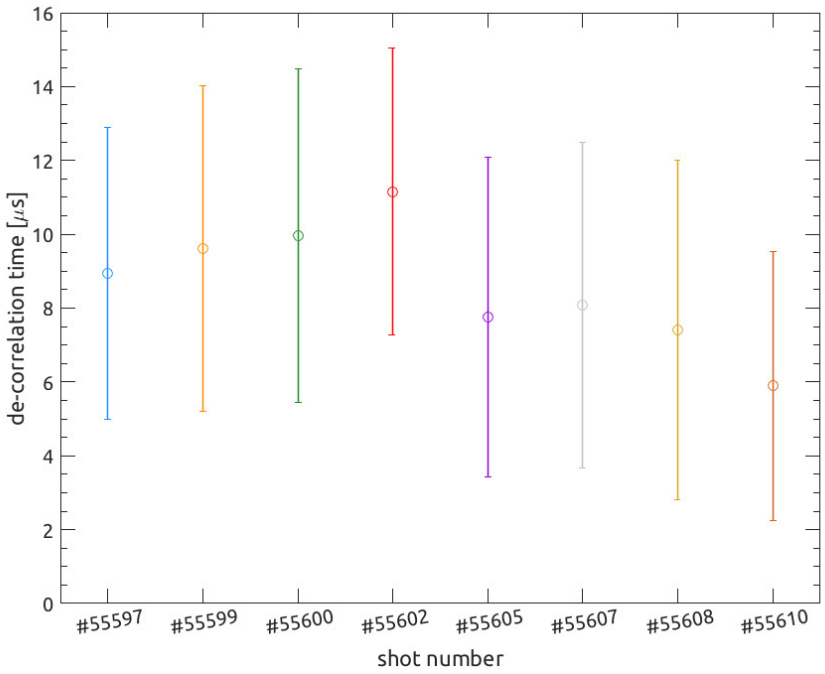

Figure 5.3: Comparison of multiple de-correlation times computed during L-mode current plateaus of different TCV shots.

hypothesis of poor signal to noise ratio is given.

The second comment deals with the possible origins of the noise content of the signals. Such sources are hypothesized to be two. An intrinsic noise content is due to the electronic acquisition chain system. Its calibration is date to the beginning of year 2016 and it is therefore considered valid. however, some of electronic cards and cabling are quite old. These, for sure, insert a noise component in the signal. The second source is considered to be the relative position between the plasma and each probe. In fact, from the figure it can be easily shown that the probes located closest to the plasma have less scattered de-correlation times.

In order to eliminate as much as possible the contribution of noise, a global averaging strategy is adopted. Computing a mean value from all of the results minimizes the noise content of the final information. In Fig. 5.2 the global mean τ_c value is shown along side the one standard deviation interval. In Fig. 5.3 all of the average de-correlation times, related to the same time window, are shown. Also this figure is representative of all the other results, meaning that it is always possible to find common results across similar time windows of different experiments.

$\tau_c \sim 10\,\mu s$ is a result that is consistently found across all data. In particular, considering the first mission, no relevant trend is seen as the plasma current is changed, nor when the mode of confinement is varied. Regarding the second mission, slightly longer de-correlation times are found for positive triangularity plasmas with respect to their negative counterparts. From the point of view of an effective turbulent diffusion coefficient this means that a positive triangularity plasma is characterized by lower transport properties, which is desirable in order to achieve the burning plasma condition.

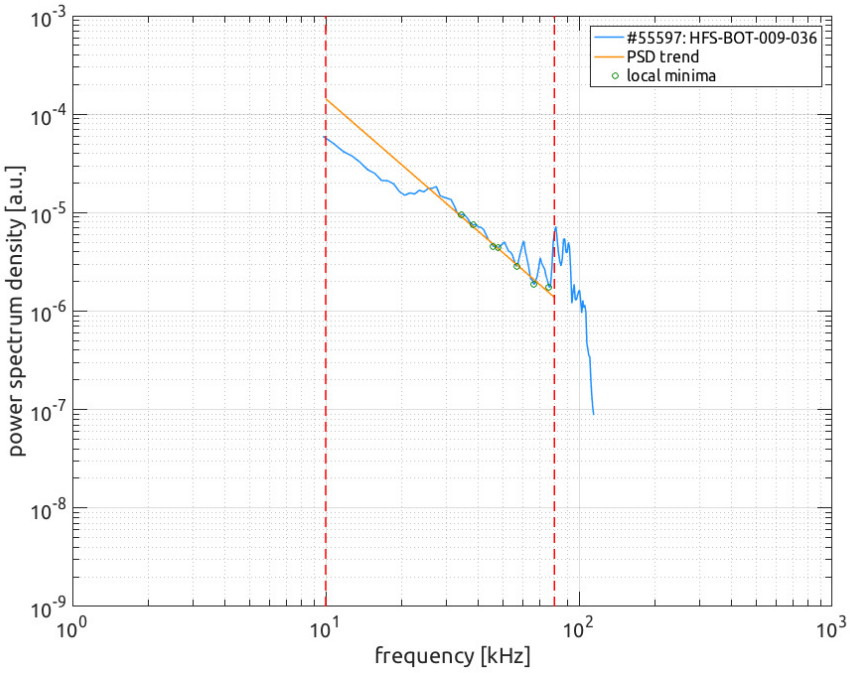

Figure 5.4: Example of PSD plot for a signal acquired during TCV shot #55597. The computed PSD trend is shown.

5.2.2 Power spectrum density

As previously presented in Sec. 4.2, the purely real power spectrum is computed for each acquired signal.

In Fig. 5.4 a representative example is shown. This is the PSD plot for one of the signals acquired during the L-mode plateau of shot #55597. The frequency interval over which the PSD is shown is determined by the filtering process that is applied prior to the analysis. It is interesting to notice how the plot significantly drops after about 100 kHz. This is due to the fact that the cut-off frequency of the measurement system is located at 120 kHz. Therefore, signal attenuation is considered relevant for frequencies over 100 kHz. This is also one of the two reasons why the PSD slope is computed in the $1 \div 80$ kHz interval. The other reason is that the energy cascade often stops in the $80 \div 100$ kHz interval, as shown in the figure. This phenomenon is not explored in the frame of the present work and, therefore, it is neglected from the computation of the slope.

Another feature that is found in all PSD plots is the presence of sudden peaks in the energy cascade interval. These are due to high harmonics coherent MHD phenomena that overlap to the underlying turbulent signal. In order to cancel their contribution from the slope computation, an exclusion algorithm is developed. This searches for local minima in the energy cascade interval and then calculates their linear interpolation. Finally, the slope of the PSD is computed as the slope of the interpolation. In Fig.5.4 both the local minima and the interpolation are shown.

Also for this analysis the results show some degree of scatter. It is therefore more interesting to comment the comparison of global averages. Different behaviors characterize the two missions.

For the first mission, although a common slope is found for each time window, it is difficult to find a trend as the current and mode of confinement change. Moreover, such common slope cannot be correlated to any of the reference one, that are presented in Sec. 4.2. On the other hand, it is much easier to exclude the *K41* and *IK* exponents, since the experimental ones are generally even more negative than the *Hall*'s one.

Regarding the second mission, it is always possible to exclude all three reference exponents. This is because the experimental values are much more negative than $-7/2$.

Both results are interesting, in the sense that it is found that the three theories chosen as a reference are not adequate to describe edge magnetic filed turbulence in a tokamak. This is of course in contrast with other MHD turbulence studies that find the famous *Kolmogorov* scaling regarding solar wind data.[17] This conclusion, however, is in accordance with the TSF analysis presented in 5.3.1.

5.2.3 Temporal increments

As stated in Sec. 4.3, this analysis is very important since it allows to state that the chosen signals are not random noise and, therefore, all other analysis are performed on time series that have actual information content.

In Fig. 5.5 the probability density function of a single signal is shown. This comes from one the coils of the HFS-TOP array and it is acquired during plasma discharge #55597. Its time window is the same as the previous two examples, that is L-mode of confinement, $I_\mathrm{p} = 210\,\mathrm{kA}$. In the figure the PDF computed for the signal is compared to a Gaussian function having the same standard deviation value. It is straightforward to notice that the experimental PDF is much broader than the Gaussian curve. Alongside a qualitative comparison, a quantitative one is performed. In fact, both kurtosis and skewness are computed. In this case $K = 283$ and $S = 0.66$. While the former value is much larger than then one of a Gaussian function, for which $K = 3$, the latter is close to the reference value, that is $S = 0$. K represents the broadness of the distribution, while S its symmetry with respect to its maximum. It is easy to understand the reason why S is slightly greater than zero in this case, since there are more variations on the positive side of the plot.

In Fig. 5.6 and 5.7 the comparison among various global averages is shown. Huge values of K are computed, in fact, up to two orders of magnitude higher than the Gaussian reference. The standard deviations are very large as well. This is due to particularly high or low values that are not excluded by the outliers elimination algorithm. In some cases several values, that seem to be invalid, are actually found to be closer than the chosen 3σ limit. Moreover, the intrinsic variability of the acquisition system, that comes from the multiple positions in which the coils are placed, adds up to the scatter of the analysis of this result. In other words, coils placed particularly far from the plasma suffer from a poorer signal pickup. This means that higher fluctuations are expected from such coils as a results of a poorer signal to noise ratio. Regarding skewness, a value close to zero means that positive and negative variations of the signal are equally likely to occur, and. $S \approx 0$ is a consistent feature across all data sets.

The kurtosis of the signals related to the first mission are about two orders of

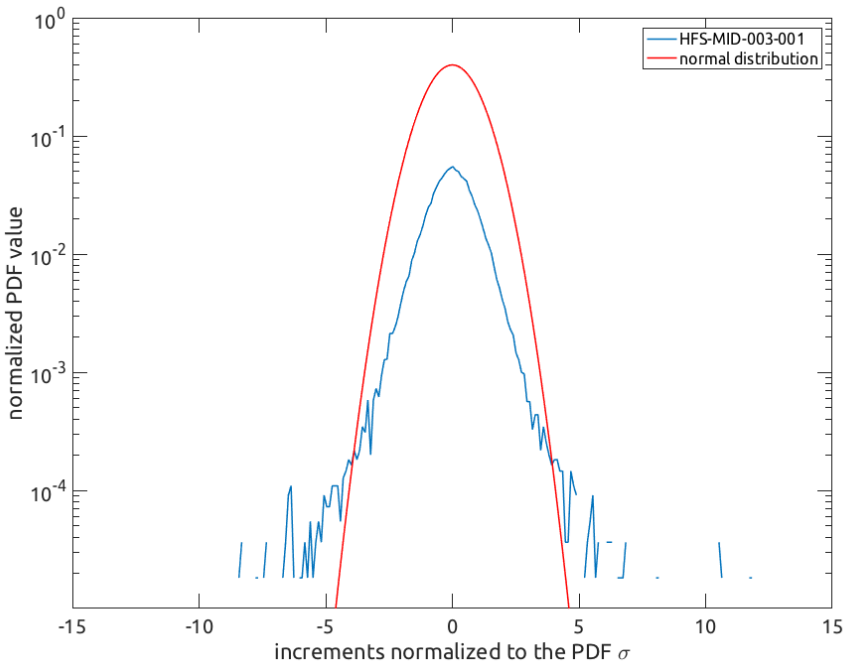

Figure 5.5: Comparison between a experimental PDF and the normal distribution that has same mean and standard deviation values.

magnitude higher than the reference of noise. Moreover, K generally increases as the plasma current grows, while staying in L-mode. On the other hand, it decreases when an L-H transition occurs. In the case of the second mission, values of K up to three orders of magnitude are computed. As anticipated, all signals show almost symmetrical PDFs since $S \approx 0$ is always found.

In conclusion, the characteristic of the experimental data clearly cannot be associated to the ones of pure noise. This guarantees that the acquired signals have a physics information content.

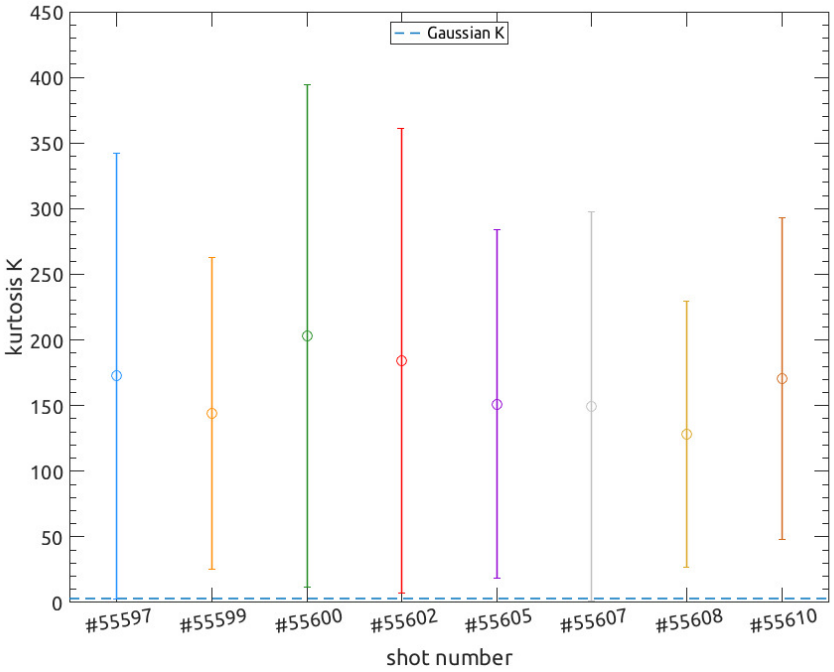

Figure 5.6: Comparison of multiple K values computed during L-mode current plateaus of different TCV shots.

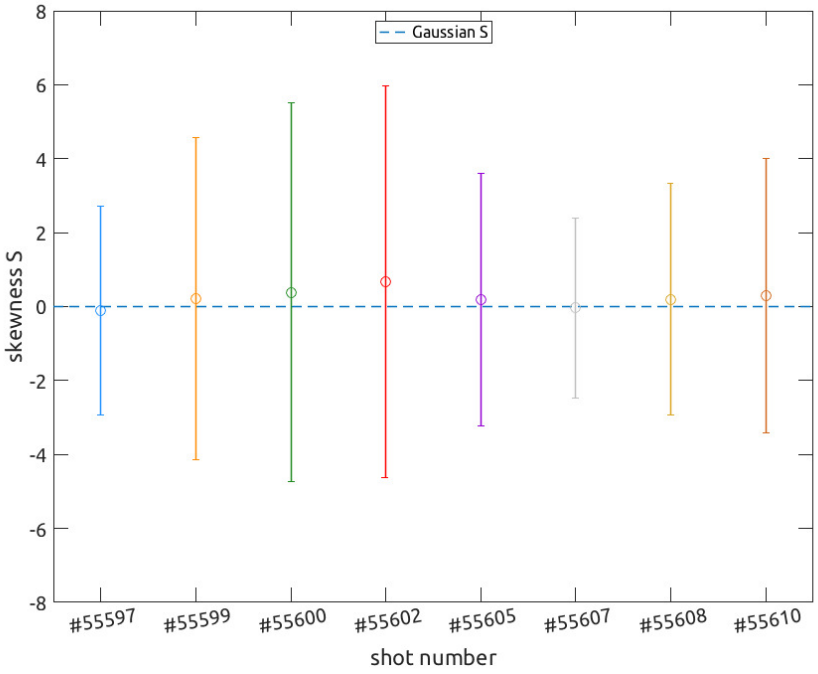

Figure 5.7: Comparison of multiple S values computed during L-mode current plateaus of different TCV shots.

5.3 Advanced Analysis

In this section the results of the last three analysis tools are presented. As for the previous section, for each tool an example is given. Then the discussion is made on the comparison of the global averages.

5.3.1 Temporal structure functions and *Hurst*'s exponents

The temporal structure function analysis is the most complex of all the implemented analysis. Not for the involved computation, but instead for what regards the interpretation of the results.

In Fig. 4.4 the temporal structure function for one signal is shown. Seven orders are computed, as explained in Sec. 4.4, for a very large number of time lags. The chosen signal is, again, acquired during the L-mode current plateau of TCV shot #5597. At order zero the TSF is a straight line. This is expected and this order is actually computed in order to check the validity of the algorithm. As expected, the value of the TSF increases as the order is increased over the value $p = 1$.

The plot is usually read either going from long to short time lags, or vice versa. The former approach is chosen here. At very long time scales, higher than 0.1 s, the TSF shows a drop in value with respect to the flat behavior found at subsequent time lags. This is due to the fact that, at these time scales, τ is of the order of magnitude of the chosen time window. Thus, the computation of the TSF value does not take into account all the values of the signal, since for some of these the correlated shifted value is not found in the time series. Hence, this part of the plot is considered not to be valid. In the next interval, that is for $0.5\,\text{ms} < \tau < 100\,\text{ms}$, the TSF shows a flat feature. Over these time scales the turbulent structures average out their contributions. Thus, turbulence is said to be stationary over these time scales. Over shorter time lags the TSF shows an oscillatory behavior, which is more difficult to explain in terms of underlying physical phenomena. This is because energy injections are related to peaks in the TSF, whereas local minima are related to energy dissipation. Even though the underlying physics is not explored, it is possible to state that there might be multiple phenomena competing over these time scales. The choice of computing the TSF over multiple orders allows to catch this feature, since the TSF values are increased at higher orders. The last interval, for $4\,\mu\text{s} < \tau < 40\,\mu\text{s}$, shows the inertial cascade. In fact, as stated in Sec. 4.4, a linear non-flat part of the TSF is correlated to the inertial energy cascade. The computation of the TSF is stopped at 5 µs because of a limit imposed by the sampling frequency of the acquisition system. In fact, the selection of a time lag close to the inverse of the sampling frequency leads to the creation of artifacts similar to the one present a long time scales. However, this time the artifact is due to the choice of performing a computation that requires an higher precision of the acquisition system.

An interesting characteristic, that is consistently found with this analysis, is the presence of the peak at $\tau \approx 30\,\mu\text{s}$. The start of the inertial cascade is located at an higher value, with respect to the stationary flat portion of the TSF. This can be explained in two ways. Reading the plot from the right, it is possible to state that the overall contribution of the oscillatory feature is, in fact, an energy injection in

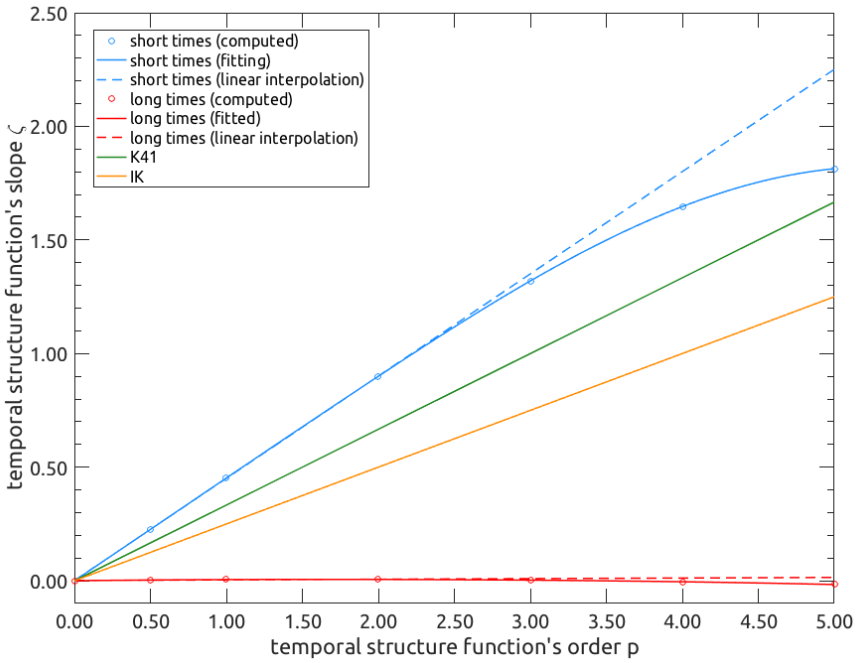

Figure 5.8: Slope of the TSF of one signal acquired during TCV shot #55597. Both slopes at short and long time delays are shown.

the system. On the other hand, reading from the left, it is possible to state that the net contribution is the one of an energy dissipation. Further investigation is here needed.

In FIg. 5.8 the evolution of the slope of the TSF, that is $\zeta(p)$, is shown. Two experimental curves are plotted: one for short and one for long time lags. In the former case the experimental slopes are located at $\zeta(p) \approx 0$ for all orders. This is expected since the TSF is flat over this region. For the short time lags case an asymptotic linear behavior is found at low orders. Again, this is expected, but this linear feature does not follow the reference ones. However, this is in accordance with the PSD analysis presented in Sec. 5.2.2. At orders higher than $p \approx 2.5$ the experimental trend deviates from linearity, showing the presence of dissipative processes acting over the inertial cascade.

Also for this analysis global averages are computed. The plot shown in Fig. 5.8 is representative of all the performed analysis. In fact, the main features of the slope of the TSF are consistently found through all data. At short time scales these are: linear behavior, for low orders, that is different form the reference ones, and departure from linearity at higher orders. For the experiments of the first mission linearity is followed up to $p \approx 1$, while for the shots of the second one this characteristic holds until $p \approx 2$. However, while for mission #1594 the plots of $\zeta(p)$ present saturation at high order, for the ones of mission #1531 drops are sometimes observed. In the case of long time lags, the TSF is consistently flat, showing stationarity over long time periods.

As explained in Sec. 4.4 and 4.5, the TSF is exploited for the computation of the *Hurst*'s exponents. H_p are computed for both the short and long time lags intervals. As expected, $H_p \approx 0$ for long time scales. This means that the signal

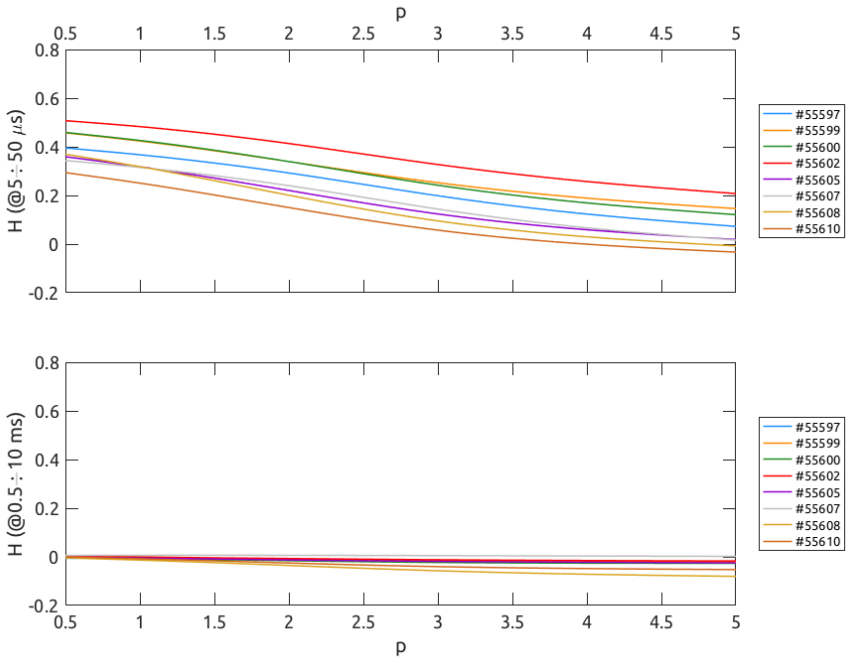

Figure 5.9: Comparison of multiple H values computed during L-mode current plateaus of different TCV shots.

is stationary over these time periods, result that is also found form the previous analysis. Moreover, the signal is said to be mono-fractal over such scales. Regarding short time lags, a multi-fractal behavior is observed. This originates from the departure of $\zeta(p)$ from linearity. In Fig. 5.9 an example of globally averaged H_p for multiple experiments is shown for both time intervals. The multi-fractal behavior is clearly visible in the short time lags case.

A further comment is needed for H_1. This is, in fact, the exponent used to establish both the time persistence of the signal and its fractal dimension. It is interesting to point out that all the analysis performed on the shots of mission #1531 show $H_1 < 0.5$. This means that these signals are anti-correlated, regardless of the actual plasma behavior. An anti-correlated signal tends to regress to its local mean value, as it is shown in Fig. 5.10. The related fractal dimension is, therefore, $D \approx 3$. This is because the acquisition system enables a two dimensional description, that is along the toroidal and poloidal coordinates, hence setting $n = 2$ in Eq. 4.19. In the case of the shots of the other mission, the majority of the signals present a persistent behavior, meaning that, even though $H_p \approx 0.5$, this value is often found to be slightly lover the threshold. Therefore, for the majority of these signals $D \approx 2.5$.

5.3.2 CH plot

This last analysis is the one that probably gives the most interesting results.

In Fig. 5.11 and example of occurrence frequencies is shown. For this work the embedding dimension is chosen to be $N = 5$. In fact, in the plot all possible permutations are shown along the horizontal axis. These are sorted in order to have

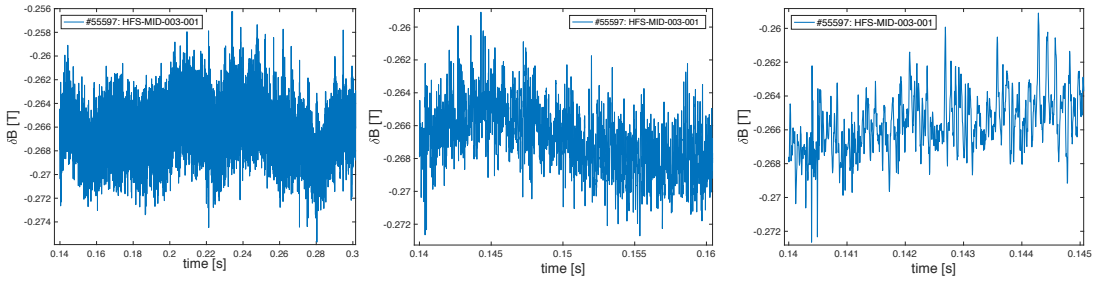

Figure 5.10: Comparison the waveform of a signal over different time scales. Notice how the signal tend to oscillate around a local mean value instead of diverging.

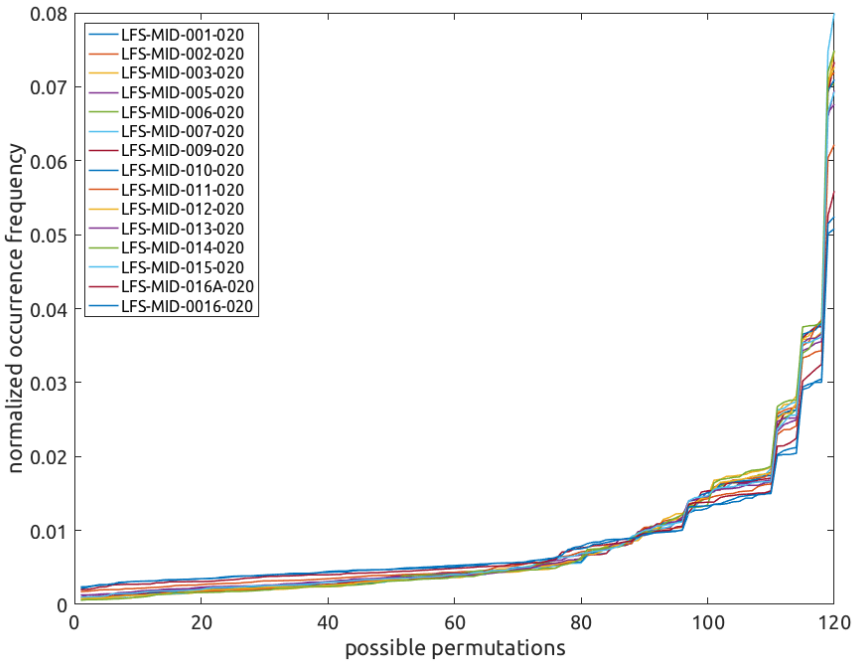

Figure 5.11: Occurrence frequencies of the 120 possible permutations for 8 signals acquired during the L-mode plateau of TCV shot #55597.

an always increasing plot, for better readability. Multiple lines are shown as there is one for each considered signal. The time series taken into account are coming from the HFS-MID probes and are acquired during the L-mode current plateau of TCV shot #55597. A pure noise signal would show a flat distribution so, at first sight it is possible to state that the analyzed signals are not pure noise.

In Fig. 5.12 the permutation entropy and complexity characteristic of the previous signals are shown on the CH plot. In order to give some reference, the characteristics of a linear ramp function, a pure noise signal, and an example from solar wind analysis[17] are plotted. Moreover, the dashed line shows the coordinates of fBm signals as their defining *Hurst*'s exponent changes. The entropy content of the signals is quite relevant, being over $H\,[P] > 0.7$.

In Fig. 5.13 a comparison between two different time windows is shown. All of the selected shot come from the first mission, so the difference in the two plots is that the first one shows the results for an L-mode $I_p = 315\,\mathrm{kA}$ plateau, while

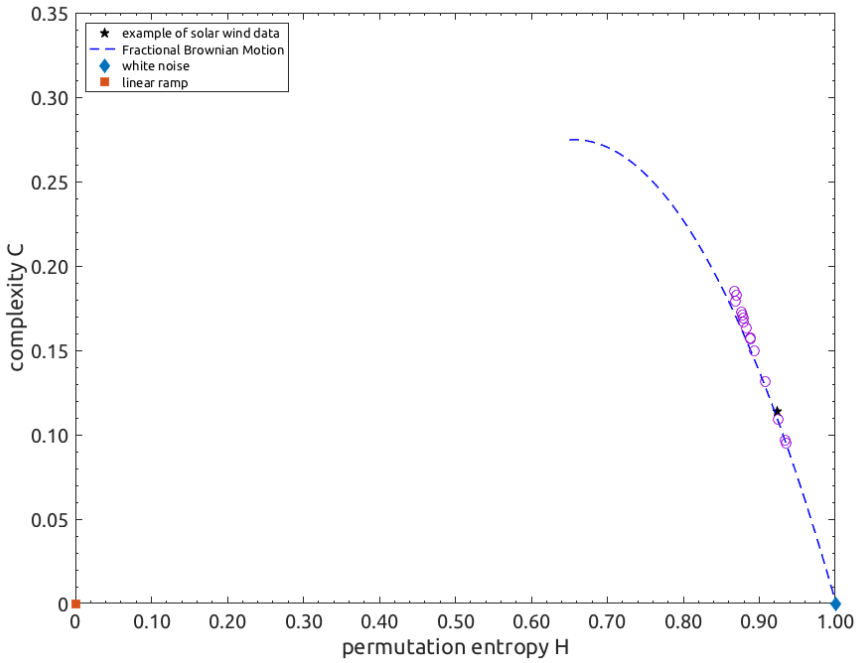

Figure 5.12: CH plot of the signals of Fig. 5.11.

for the second one the related plasmas are in ELM-free H-mode at the same current. This comparison is representative of the behavior of all the shots of the first mission, which is characterized by a decrease in complexity when the L-H transition occurs. This drops is explained by the establishment of the pressure pedestal during the H-mode of operation. When the pedestal is established, the turbulent flow is dominated by the critical balance dynamics. This states that the turbulent cascade is generated when critical values of the gradients of the involved physical quantities are reached. During the L-mode of operation, instead, there are multiple contributions that compete in order to establish the overall dynamics. An higher complexity is, therefore, correlated to this competition among different mechanisms.

In Fig. 5.14 all of the results for the second mission are shown. The division is between positive and negative triangularities. Higher complexity values are observed for the latter case. As a matter of fact, negative triangularity configurations are characterized by more important transport properties. In this sense, this result is expected. The light-blue points are related to two plasmas centered at $z = 0\,\mathrm{m}$, their presence is useful to easily track the trend in complexity change.

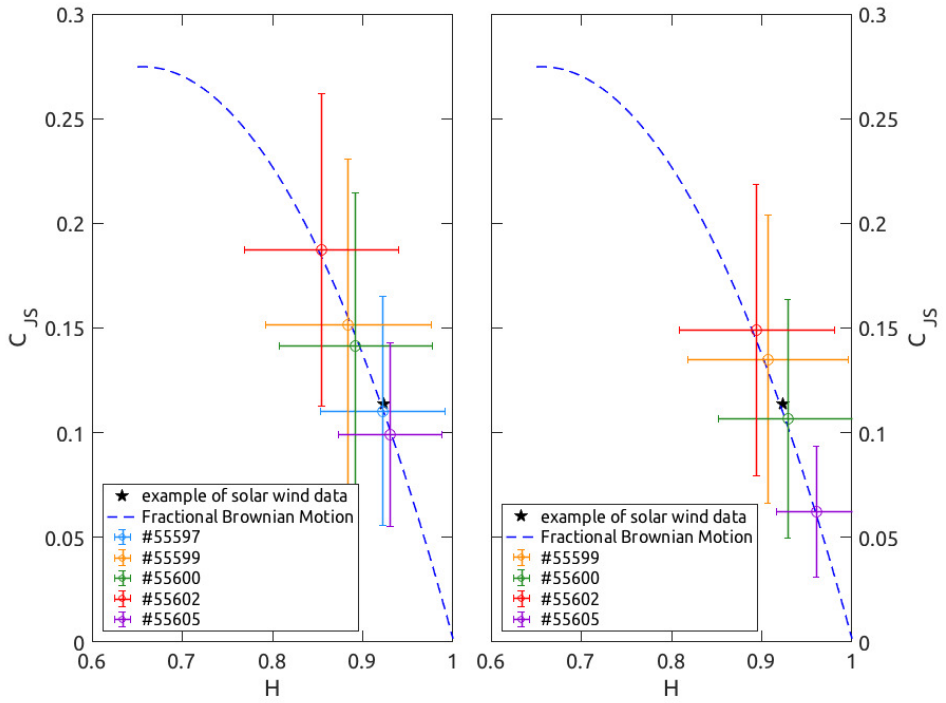

Figure 5.13: Comparison of global CH characteristics for multiple shots during two time windows: L-mode (left) and ELM-free H-mode (right). Notice how the complexity feature decreases as the plasma undergoes the L-H transition.

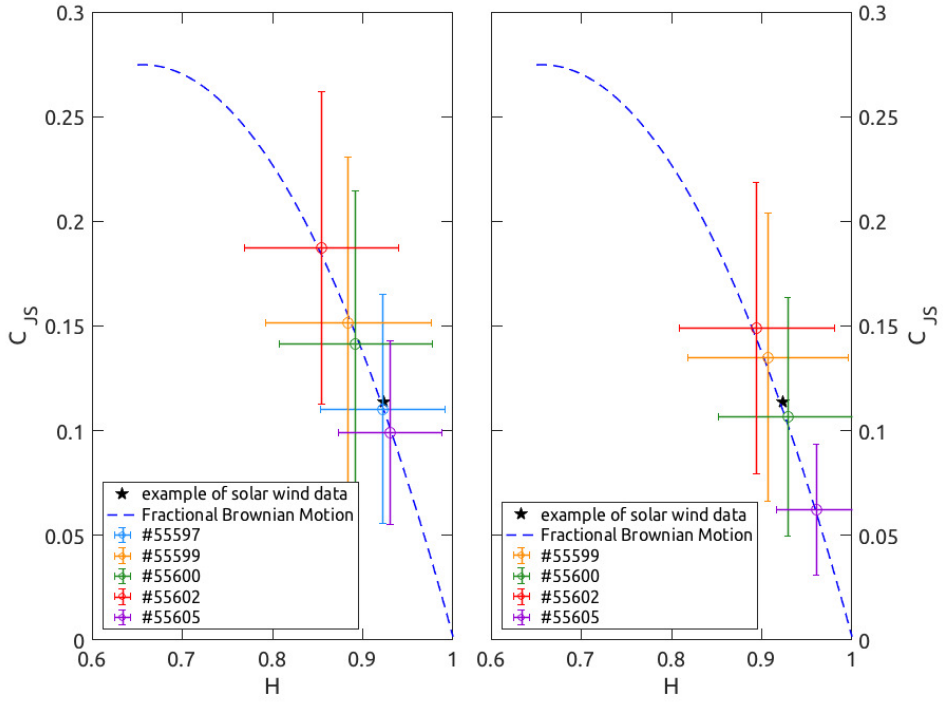

Figure 5.14: Comparison of global CH characteristics for all the shot of the second mission; negative triangularity on the left and positive on the right. Notice how the complexity values lower for positive triangularities.

Chapter 6

Conclusions

In this Chapter the results of the presented thesis are summarized, and some ideas for future works are presented.

The need of a robust algorithm to characterize the edge magnetic field turbulence in a statistical fashion has been fulfilled. The developed code, reported in Appendix A, offers a standardized, but flexible, way of invoking multiple statistical tools on the measurements acquired with the fast magnetic acquisition system installed in the TCV tokamak.

The code is able of performing the following list of analysis: computation of the auto-correlation function and de-correlation time, computation of the power spectrum density, computation of the probability density function of time increments, computation of multiple orders of the temporal structure function, computation of the *Hurst*'s exponent of the time series, computation of the permutation entropy and complexity features of the signal.

The code was validated through the analysis of actual TCV magnetic data. In fact, the analysis of 27 plasma discharges was performed. These were chosen from a pool of two different TCV experimental missions that aims to study the L-H transition physics and core electrons temperature turbulence, respectively.

De-correlation times of the order of 10 µs are consistently computed across all data and the obtained PSD spectra do not follow the theoretical scaling laws of reference over the analyzed frequency interval.
The results of the PDF analysis clearly show that the analyzed signals are not random noise, but instead they have a physical information content.
The TSF analysis showed the presence of dissipation during the energy cascade, as well as the presence of competing phenomena, prior to the cascade, that can either give a net energy injection or dissipation contribution.
The computed *Hurst*'s exponents showed that the signals are generally anti-correlated, even though in some cases H is about 0.5 or even slightly larger.
The permutation entropy and complexity analysis showed that the acquired signal follow the properties of fBm functions. This means that the experimental data can be described by these functions. Moreover, the entropy content of the time series is generally large, meaning that the signals have a high noise contribution. This

means that the signal to noise ratio is generally poor, which, in turn, causes a high degree of scattering in the results of the single signals.

The strategy of performing global averages is a successful solution towards this scattering issue. This is because the contribution of noise is canceled when averaging is carried out. Thus, having more than 200 signals is a strength of this analysis procedure.

Nevertheless, other strategies are valid as well. As a matter of fact, there might be more meaningful ways to deal with the high noise content of the signals. An example is to adopt a strategy that considers as not valid all signals having a normalized entropy greater than a chosen threshold. More investigation in this sense could improve the robustness of the analysis.

Given the modular nature of the code, it is possible to modify and upgrade it with relatively little effort. This means that multiple analysis strategy could be easily implemented and then compared.

An higher degree of automation of the code is for sure an interesting feature. Regarding this point, an automatic detection of the intervals over which perform the analysis would be much appreciated. For example, an automatic energy cascade range-detection algorithm for the PSD and TSF analysis could be implemented.

Some analysis tools were not considered during the development of the code, but it might be useful to take these into account. Some examples are: the spatial and cross-correlation analysis among all the probes, the wavelet analysis and the R/S method for the computation of the *Hurst*'s exponents.

The frequency interval of the analysis can be widened exploiting other types of probes, such as the LTCC3D ones, that have a sampling frequency that is one order of magnitude higher than the one of the *Mirnov* coils. However, the number of these very high frequency probes is quite low with respect to the one used for this work. This means that the statistical characteristic of such analysis would be diminished with the LTCC3D.

Appendix A

Implemented code

The developed code is organized in multiple files. The analysis is called either invoking the StatComp function or the MultiExecution one. The latter is designed to perform multiple analysis with the same parameters with one call. The code is very flexible; in fact it relies on an variable input arguments call, which allows to modify any computation parameter from the command line. The execution of the code relies on other functions that are part of a MATLAB® toolbox that contains all the validated code developed at the Swiss Plasma Center. The execution is also very demanding on the machine it is performed on. This is because the code is highly parallelized in order to take advantage of multi-core, multi-thread systems. However, the parallelization dramatically lowers the total run time of the code.

All of the plotting functions are not reported in this documents since their purpose is to give to the user a quick look at the results. The outputs are given in a sort of raw state, in order to allow maximum customization of their handling. Scripts used for start up and shut down of the computation procedure are also not reported here.

The version of the reported code is the one of the end of May 2017. The code is extensively commented. However, to obtain further information, or to have access to the whole package, please send an email to alessandrotolio@hotmail.com.

```
function [StatStruct,DataMHD]=StatComp(varargin)
%% STATCOMP Performs statistical operations on magnetic data for turbulence analysis.
% author: Aylwin Iantchenko, 17.11.2015.
% Last update: Alessandro Tolio, Apr. 10 2017
%
% CALL: [StatStruct,DataMHD]=StatComp(varargin);
% varargin is of type:
% 1) list of type: ..., 'param_name',param_value, ...;
%
% REQUIRED INPUT PARAMETERS:
% Shot       : shot(s) number on which the analysis is to be performed,
%              can take multiple values, use a number value format. example: ..., 'shot',[54000,51000,47565], ...
% Array      : array of probes to be analysed, can take multiple values, use a cell format.
%              example: ..., 'Array',{'LFS-MID','HFS-BOT','SAD-TOP'}, ...
%              NOTE: if multiple arrays are provided as input, then all selected shots will be analysed using all these arrays.
%
% Stat       : which statistic should be computed, use a cell array as before.
%             'Stat' name should correspond to one or more of the following function name:
%              -) AutoCorr    : results from the auto-correlation analysis ==> this is the default value!
%              -) PSD         : results from power spectral density analysis.
%              -) TempFun     : results from temporal structure function analysis.
%              -) TempInc     : results from temporal increment analysis.
%              -) PermEntr    : results from Permutation entropy analysis.
%
% OPTIONAL INPUT PARAMETERS: see ParamStruct below for the complete list of possible parameters;
%                            each one has a brief description on the side
%
% Output Parameters:
% StatStruct  : structure containing the parameters used for computation (ParamStruct)
%               and all results from computation; each analysis has its
%               field, that is a cell array of output structures
% DataMHD     : MHD data, useful to avoid reloading and save some time

tic %start timer
%% initialise variables
% checks for active parallel pool, if there is none it creates one
[~]=gcp;

% default processing structure
ProcStructDEF=struct( ...
    'TimeInt',[.5,.6], ...  %selects time interval used for computation, unit =[sec]
    'MaxVal',.2, ...        %selects value for the maximum of turbulent magnetic field, unit =[T]
    'TrendPol',5 ...        %selects degree of polynomial used for removing trends in turbulent magnetic field data
    );

% default input parameter structure
ParamStruct=struct( ...
    'Shot',[], ...                              %shot(s) selected for the statistical analysis of turbulence data, use a number value format
    ...
    'Array',{''}, ...                           %array(s) to be analysed, use a cell format, can take multiple values {array1, ..., arrayNN}
    ...                                         %it is possible to call multiple array with one input:
    ...                                         % - HFS, LFS : all arrays on the given side of the vessel
    ...                                         % - BOT, MID, TOP : all arrays at the given height
    ...                                         % - TOROIDAL, POLOIDAL : all probes of the toroidal or poloidal arrays
    ...                                         % - ALL : all arrays (from 191 to 224 probes)
    'Stat',{''},...                             %type of statistics that will be
```

```
        computed, as the one above
57      ...                                     %it is possible to call all statistics
        at once
58      'ProcParam',ProcStructDEF, ...          %structure containing parameters for
        processing
59      'BBunits',[], ...                       %unit of turbulent magnetic field,
        default =[T]
60      'fs',[], ...                            %sampling frequency for the selected
        set of data to be processed [Hz]
61      'TimeDisr',[], ...                      %disruption time in each shot, from
        FMPloader [s]
62      'StrCode','StatComp: ', ...             %string used to mark messages from the
        statComp function
63      'ProbeName',[], ...                     %probenames required to create legends
64      'ProbeNameAvg',[], ...                  %legend lines to be used when when
        averaging is performed
65      ...
66      'Average',true, ...                     %set to true to compute array averaging
67      'Plot',true, ...                        %set to true to plot results
68      'Save',false, ...                       %set to true to save results
69      'CloseAll',false,...                    %set to true to automatically close all
         figures
70      'FilterSignal',true,...                 %set to true to bandpass filter the
        signal after processing
71      'FilterFreq',[],...                     %frequency intervals used in bandpass
        filtering
72      'ExcludeOutliers',true,...              %exclude outliers from analysis, based
        on each array-mean results
73      'SaveSpace',true,...                    %if true it cancels some data to save
        space on disk
74      ...
75      'nsig',[],...                           %number of signals for each shot
76      ...
77      'AutoTimeLimit',0.1,...                 %limit used for computation in
        autocorrelation analysis [s]
78      'AutoFactorTau',exp(-1),...             %scaling factor used to determine the
        auto-correlation time scale
79      ...
80      'TauTempInc',8e-6,...                   %delay used for time increments
        analysis [s]
81      'nbins',500, ...                        %accuracy of time increments
        computation (number of bins used in the estimation of the PDF)
82      ...
83      'PSDSlopeRange',1e4*[1,8],...           %range used for estimation of slope in
        PSD analysis [kHz]
84      'NoPeaks',true,...                      %if true then computes PSD slope
        excluding coherent peaks
85      ...
86      'TempExp',[0,.5,1,2,3,4,5], ...         %exponents used in the computation
87      'TempTau',10.^[-6,-2], ...              %range of delays used for computation
88      'TempSlopeRange',[5e-6,5e-5;5e-4,1e-2], ... %range used for estimation of slope
89      ...
90      'PermEntrN',5,...                       %embedding dimension
91      'PermEntrTau',8e-6...                   %delay as input
92      );
93
94  % turning off useless warnings
95  warning('off','MATLAB:Axes:NegativeDataInLogAxis')
96  warning('off','MATLAB:elementsNowStruc')
97  warning('off','MATLAB:legend:IgnoringExtraEntries')
98  warning('off','MATLAB:MKDIR:DirectoryExists')
99
100 %% read input
101 % read and store input variables: convert list of input parameters to structure and
        replace default by custom
102 if ~isempty(varargin)
103     param_names=fieldnames(ParamStruct);
104     for j=1:2:length(varargin)-1
105         if (any(strcmp(param_names,varargin{j}))) % if field exists in varargin save
              corresponding value
106             ParamStruct.(varargin{j})=varargin{j+1};
```

```
107         end
108     end
109 end
110 clear param_names
111
112 % substitutes missing ProcParam values with default ones
113 if ~isfield(ParamStruct.ProcParam,'TimeInt')
114     ParamStruct.ProcParam.TimeInt=ProcStructDEF.TimeInt;
115 end
116 if ~isfield(ParamStruct.ProcParam,'MaxVal')
117     ParamStruct.ProcParam.MaxVal=ProcStructDEF.MaxVal;
118 end
119 if ~isfield(ParamStruct.ProcParam,'TrendPol')
120     ParamStruct.ProcParam.TrendPol=ProcStructDEF.TrendPol;
121 end
122 clear ProcStructDEF j
123
124 % perform some checks on input variables: if no shot, array or statistics provided as
      input, aborts process
125 if isempty(ParamStruct.Shot)
126     disp([ParamStruct.StrCode,'please provide choice of shot, aborting analysis ...']);
127     return
128 end
129 if numel(ParamStruct.ProcParam.TimeInt)<2 || ParamStruct.ProcParam.TimeInt(2)<
      ParamStruct.ProcParam.TimeInt(1)
130     disp([ParamStruct.StrCode,'please provide a valid choice for the time window,
          aborting analysis...'])
131     return
132 end
133 if iscell(ParamStruct.Array) %check also for cell array format
134     if isempty(ParamStruct.Array)
135         disp([ParamStruct.StrCode,'please provide choice of array, aborting analysis
              ...']);
136         return
137     end
138 else
139     disp([ParamStruct.StrCode,'please provide choice of array in a cell format, e.g.: {
          ''HFS-MID''}, aborting analysis ...']);
140     return
141 end
142 if iscell(ParamStruct.Stat) %check also for cell array format
143     if isempty(ParamStruct.Stat{1})
144         disp([ParamStruct.StrCode,'please provide choice of statistics, aborting
              analysis ...']);
145         return
146     end
147 else
148     disp([ParamStruct.StrCode,'please provide choice of statistics in a cell format, e.
          g.:{''AutoCorr''}, aborting analysis ...']);
149     return
150 end
151
152 % search for the the following array calls; if present, it uses all avaible array with
      the given position,
153 %  e.g. 'HFS' calls 'HFS-BOT','HFS-MID' and 'HFS-TOP'
154 Check={'HFS','LFS','BOT','MID','TOP','TOROIDAL','POLOIDAL','ALL'};
155 for i=1:length(Check)
156     ParamStruct.Array=ModifyArray(Check{i},ParamStruct.Array,i);
157 end
158 clear Check i
159
160 % sets all methods if requested
161 if (ismember('ALL',ParamStruct.Stat) || ismember('all',ParamStruct.Stat) || ismember('
      All',ParamStruct.Stat))
162     for i=1:length(ParamStruct.Stat)
163         ParamStruct.Stat(i)=[];
164     end
165     ParamStruct.Stat={'AutoCorr','PSD','TempInc','TempFun','PermEntr'};
166 end
167
168 % check to close all open figures
```

```
169 if ~ParamStruct.CloseAll
170     control=input('Function ready to load data. Do you want to close previous figures?
            (yes/no, default==yes) \n','s');
171     if strcmp(control,'yes') || isempty(control)
172         close all
173     end
174 else close all
175 end
176 clear control
177
178 SizeLoop=length(ParamStruct.Shot)*length(ParamStruct.Array); %size of the loop on "
        shots * arrays" used for the analysis
179
180 % read input data, if provided
181 idx=find(strcmp('Data',varargin));
182 if idx
183     DataMHD=varargin{idx+1};
184     disp([ParamStruct.StrCode,'input (magnetic) data provided as input ...']);
185 else %or fetch them using FMPloader
186     DataMHD=cell(1,SizeLoop);
187     disp([ParamStruct.StrCode,'input (magnetic) data NOT provided as input,  must fetch
            them from tcvdata server...']);
188 end
189 clear idx varargin
190
191 % initialization of some data containers
192 fs=zeros(1,SizeLoop);
193 nsig=fs;
194 TimeDisr=fs;
195 BBunits=cell(1,SizeLoop);
196 ProbeName=BBunits;
197 ProbeNameAvg=BBunits;
198 Signal=cell(1,SizeLoop);
199 Time=Signal;
200
201 % initialize structure to save the output data for the diffusion coefficients
202 % Lc estimated value for the magnetic field connection length at the plasma edge, Lc=(
        Rq)_edge~(0.88+0.25)*5~6m
203 DiffArray=cell(1,SizeLoop);
204 DiffStruct=struct( ...
205     'btorAxis',0, ...           %equilibrium toroidal magnetic field on the magnetic
            axis (unit =[T])
206     'btorEdge',0, ...           %equilibrium toroidal magnetic field at the plasma edge
            (unit =[T])
207     'BmeanRaw',0, ...           %computed average raw magnetic field (turbulent, not
            equilibrium) (unit =[T])
208     'BstdRaw',0, ...            %standard deviation on raw magnetic field (turbulent,
            not equilibrium) (unit =[T])
209     'BmeanProc',0, ...          %computed average processed magnetic field (turbulent,
            not equilibrium) (unit =[T])
210     'BstdProc',0, ...           %standard deviation on processed magnetic field (
            turbulent, not equilibrium) (unit =[T])
211     'BBunits',[], ...           %unit of magnetic field, default =[T]
212     'neEdge',0, ...             %computed volume and time averaged density at the
            plasma edge (unit =[m^-3])
213     'LneEdge',0, ...            %computed density scale length at the plasma edge (unit
            =[m])
214     'teEdge',0, ...             %computed volume and time averaged temperature at the
            plasma edge (unit =[eV])
215     'LteEdge',0, ...            %computed temperature scale length at the plasma edge (
            unit =[m])
216     'vA',0, ...                 %estimated Alfven speed at the plasma edge (unit =[m/
            sec])
217     'vS',0, ...                 %estimated sound speed at the plasma edge (unit =[m/sec
            ])
218     'vD',0, ...                 %estimated ion/electron drift velocity at the plasma
            edge (unit =[m/sec])
219     'Lc',6, ...                 %estimated connection length of the magnetic field at
            the plasma edge (unit =[m])
220     'DiffCoeff',zeros(2,2) ...  %estimated value of the EM turbulent diffusion
            coefficient (unit =[m^2/sec])
```

```
221     );
222 DiffArray(1,:)={DiffStruct};
223 clear DiffStruct
224
225 disp([ParamStruct.StrCode,'loading data from server or input parameters...'])
226 Shot=ParamStruct.Shot;
227 Array=ParamStruct.Array;
228 % loop for all signals: number of shots times selected set of arrays
229 parfor k=1:SizeLoop
230     ShotIdx=ceil(k/length(Array));                              %save index of shot, used
          for plotting purposes
231     ArrayIdx=k-length(Array)*(ShotIdx-1);                       %save index of shot, used
          for plotting purposes
232
233     %if data not provided in input, fetch new data
234     if isempty(DataMHD{k}) && ~strcmp(ParamStruct.Array{k},'LTCC3D')
235         DataMHD{k}=FMPloader('shot',Shot(ShotIdx),'array',Array{ArrayIdx}); %#ok<*PFBNS
              >
236     elseif isempty(DataMHD{k}) && strcmp(ParamStruct.Array{k},'LTCC3D')
237         DataMHD{k}=ReadDtacqLTCC3D('shot',Shot(ShotIdx),'array',Array{ArrayIdx});
238     end
239
240     % read selected input from acquired tdi object
241     Signal{k}=DataMHD{k}.tdi.data;                                                  %read
          signal values
242     fs(k)=DataMHD{k}.fs;                                                            %read
          sampling frequency for shot
243     Time{k}=DataMHD{k}.TimeFull(1):1/fs(k):DataMHD{k}.TimeFull(2);                  %create
          time vector
244     nsig(k)=size(DataMHD{k}.tdi.dim{2},2);                                          %save
          number of signals acquired
245     TimeDisr(k)=DataMHD{k}.TimeDisr;                                                %save
          disruption time
246     BBunits{k}=DataMHD{k}.tdi.units;                                                %save unit
          of measured turbulent magnetic field
247     DiffArray{k}.BBunits=DataMHD{k}.tdi.units;
248
249     % save name of probes, used for plotting purposes
250     hh=char(DataMHD{k}.signalNames);
251     ProbeName{k}=strcat(['#',int2str(Shot(ShotIdx)),': ',Array{ArrayIdx},'-'],cellstr([
          hh(:,7:9) repmat('-',[size(hh,1) 1]) hh(:,11:13)])');
252     ProbeNameAvg{k}=strcat('#',[int2str(Shot(ShotIdx)),': ', Array{ArrayIdx}]);
253 end
254
255 % filling of paramStruct and emptying of containers
256 ParamStruct.fs=fs;
257 ParamStruct.nsig=nsig;
258 ParamStruct.TimeDisr=TimeDisr;
259 ParamStruct.BBunits=BBunits;
260 ParamStruct.ProbeName=ProbeName;
261 ParamStruct.ProbeNameAvg=ProbeNameAvg;
262 clear fs nsig TimeDisr BBunits ProbeName ProbeNameAvg Array i k
263 disp([ParamStruct.StrCode,'signals from ',num2str(sum(ParamStruct.nsig)),' probes
      successfully acquired!'])
264
265 %% reorganization of data in order to process it faster
266 disp([ParamStruct.StrCode,'preparing computation structures...'])
267 ComputationArray=DataOrganizer(ParamStruct,Signal,Time,SizeLoop);
268 clear Signal Time
269
270 % performs some checks on some input parameters
271 ParamStruct.AutoTimeLimit=min(ParamStruct.AutoTimeLimit,diff(ParamStruct.ProcParam.
      TimeInt));   %check on the time lags for the auto-correlation analysis
272 a=log10(max(ParamStruct.TempTau(1),1/min(ParamStruct.fs)));
273 b=log10(ParamStruct.ProcParam.TimeInt(2)-ParamStruct.ProcParam.TimeInt(1));
274 ParamStruct.TempTau=logspace(a,b,1e3);
                                                                %check on the time lags
      for the temporal structure analysis
275 clear a b
276
277 %data container preallocation
```

```
278 AutoCorrArray=cell(1,SizeLoop);
279 XCorrPlotArray=AutoCorrArray;
280 AutoCorrPlotArray=AutoCorrArray;
281 PSDArray=AutoCorrArray;
282 PSDPlotArray=AutoCorrArray;
283 TempIncArray=AutoCorrArray;
284 TempIncPlotArray=AutoCorrArray;
285 TempStructArray=AutoCorrArray;
286 TempStructPlotArray=AutoCorrArray;
287 PermArray=AutoCorrArray;
288 PermPlotArray=AutoCorrArray;
289 Outliers=AutoCorrArray;
290
291 % temporary parameter containers
292 AutoParam=struct(...                                    %auto-correlation parameters
293     'TimeLimit',ParamStruct.AutoTimeLimit,...
294     'FactorTau',ParamStruct.AutoFactorTau,...
295     'Plot',ParamStruct.Plot,...                         %used with SaveSpace to decide
          whether cancel some unwanted data right away
296     'SaveSpace',ParamStruct.SaveSpace...
297     );
298
299 PSDParam=struct(...                                     %PSD parameter
300     'SlopeRange',ParamStruct.PSDSlopeRange,...
301     'NoPeaks',ParamStruct.NoPeaks,...
302     'Plot',ParamStruct.Plot,...                         %used with SaveSpace to decide
          whether cancel some unwanted data right away
303     'SaveSpace',ParamStruct.SaveSpace...
304     );
305
306 IncParam=struct(...                                     %time increments parameters
307     'TauTempInc',ParamStruct.TauTempInc,...
308     'nbins',ParamStruct.nbins,...
309     'Plot',ParamStruct.Plot,...                         %used with SaveSpace to decide
          whether cancel some unwanted data right away
310     'SaveSpace',ParamStruct.SaveSpace...
311     );
312
313 TempParam=struct(...                                    %temporal function parameters
314     'TempExp',sort(ParamStruct.TempExp),...
315     'TempTau',ParamStruct.TempTau,...
316     'TempSlopeRange',ParamStruct.TempSlopeRange,...
317     'Plot',ParamStruct.Plot,...                         %used with SaveSpace to decide
          whether cancel some unwanted data right away
318     'SaveSpace',ParamStruct.SaveSpace...
319     );
320
321 PermParam=struct(...                                    %permutation entropy parameters
322     'N',ParamStruct.PermEntrN,...
323     'Tau',ParamStruct.PermEntrTau,...
324     'Plot',ParamStruct.Plot,...                         %used with SaveSpace to decide
          whether cancel some unwanted data right away
325     'SaveSpace',ParamStruct.SaveSpace...
326     );
327
328 %% perform computation of statistics on selected signals, loop over all signals
329 disp([ParamStruct.StrCode,'starting computation...'])
330 disp([ParamStruct.StrCode,'now computing...'])
331
332 % display some warnings
333 if sum(ParamStruct.nsig)>30
334     disp([ParamStruct.StrCode,num2str(sum(ParamStruct.nsig)),' signals are quite a lot
          ... This may take a while: just wait for it!'])
335     t=clock;
336     if t(4)>=12 && t(4)<14
337         disp([ParamStruct.StrCode,'you may want to have your lunch in the meantime :-)'
              ])
338     else
339         disp([ParamStruct.StrCode,'you may want to go grab a coffee in the meantime :-)
              '])
340     end
```

```
341 end
342
343 parfor i=1:SizeLoop
344     ComputationArray{i}.SignalIdx=i;
345
346     % process the data and compute the Alfven speed, the sound speed and the ion/
            electron drift velocity, which are needed to calculate
347     % the turbulent diffusion coefficients in the approximation of a large mean guide
            field.
348     [ComputationArray{i},DiffArray{i}]=ProcessSignal(ComputationArray{i},DiffArray{i});
349
350     % compute the desired statistics
351     % auto-correlation analysis
352     if ismember('AutoCorr',ComputationArray{i}.Stat)
353         [AutoCorrArray{i},DiffArray{i}]=AutoCorr(ComputationArray{i},AutoParam,
                DiffArray{i});
354
355         % creates plotting structure
356         if AutoParam.Plot
357             AutoCorrPlotTemp=AutoCorrArray{i};
358             AutoCorrPlotTemp.Average=ComputationArray{i}.Average;
359             AutoCorrPlotTemp.PlotStr=ComputationArray{i}.PlotStr;
360             AutoCorrPlotTemp.ProbeNameAvg=ComputationArray{i}.ProbeNameAvg;
361             AutoCorrPlotTemp.Factor=AutoParam.FactorTau;
362             AutoCorrPlotArray{i}=AutoCorrPlotTemp;
363         end
364     end
365
366     % power spectral density analysis
367     if ismember('PSD',ComputationArray{i}.Stat)
368         PSDArray{i}=PSD(ComputationArray{i},PSDParam);
369
370         % creates plotting structure
371         if PSDParam.Plot
372             PSDPlotStruct=PSDArray{i};
373             PSDPlotStruct.Average=ComputationArray{i}.Average;
374             PSDPlotStruct.PlotStr=ComputationArray{i}.PlotStr;
375             PSDPlotStruct.ProbeNameAvg=ComputationArray{i}.ProbeNameAvg;
376             PSDPlotStruct.SlopeRange=[num2str(PSDParam.SlopeRange(1)*10^-3),'-',num2str
                    (PSDParam.SlopeRange(2)*10^-3),'kHz'];
377             PSDPlotArray{i}=PSDPlotStruct;
378         end
379     end
380
381     % temporal increment analysis
382     if ismember('TempInc',ComputationArray{i}.Stat)
383         TempIncArray{i}=TempInc(ComputationArray{i},IncParam);
384
385         % creates plotting structure
386         if IncParam.Plot
387             TempIncPlotStruct=TempIncArray{i};
388             TempIncPlotStruct.Average=ComputationArray{i}.Average;
389             TempIncPlotStruct.PlotStr=ComputationArray{i}.PlotStr;
390             TempIncPlotStruct.ProbeNameAvg=ComputationArray{i}.ProbeNameAvg;
391             TempIncPlotArray{i}=TempIncPlotStruct;
392         end
393     end
394
395     % temporal structure functions
396     if ismember('TempFun',ComputationArray{i}.Stat)
397         TempStructArray{i}=TempStruct(ComputationArray{i},TempParam);
398
399         % creates plotting structure
400         if TempParam.Plot
401             TempPlot=TempStructArray{i};
402             TempPlot.Average=ComputationArray{i}.Average;
403             TempPlot.PlotStr=ComputationArray{i}.PlotStr;
404             TempPlot.ProbeNameAvg=ComputationArray{i}.ProbeNameAvg;
405             TempPlot.StrLegend=cell(size(TempStructArray{i}.p));
406             for j=1:length(TempStructArray{i}.p)
407                 TempPlot.StrLegend(j)={['p=',num2str(TempStructArray{i}.p(j))]};
```

```
408             end
409             TempStructPlotArray{i}=TempPlot;
410         end
411     end
412
413     % permutation entropy and complexity analysis
414     if ismember('PermEntr',ComputationArray{i}.Stat)
415         PermArray{i}=PermEntropy(ComputationArray{i},PermParam);
416
417         % creates plotting structure
418         if PermParam.Plot
419             PermPlotStruct=PermArray{i};
420             PermPlotStruct.PlotStr=ComputationArray{i}.PlotStr;
421             PermPlotStruct.Average=ComputationArray{i}.Average;
422             PermPlotStruct.ProbeNameAvg=ComputationArray{i}.ProbeNameAvg;
423             PermPlotArray{i}=PermPlotStruct;
424         end
425     end
426
427     % recompute means and std if requested
428     if ComputationArray{i}.Average && ComputationArray{i}.ExcludeOutliers
429         [AutoCorrArray{i},PSDArray{i},TempIncArray{i},TempStructArray{i},PermArray{i},
                Outliers{i}]=ComputeNewAverages(AutoCorrArray{i},PSDArray{i},TempIncArray{i
                },TempStructArray{i},PermArray{i});
430     end
431 end
432 clear Average AutoParam XParam IncParam PSDSlopeRange TempParam PermParam Coordinates i
        j
433
434 %% plots, if requested
435 if ParamStruct.Plot
436
437     % plot of raw and processed deltaB signals
438     disp([ParamStruct.StrCode,'plotting raw and processed data...'])
439     ProcessPlot(ComputationArray,SizeLoop)
440
441     % plots of the results of computations
442     disp([ParamStruct.StrCode,'plotting results of computation...'])
443     if ~isempty(AutoCorrPlotArray{1})
444         AutoCorrPlot(AutoCorrPlotArray)
445     end
446     if ~isempty(XCorrPlotArray{1})
447         XCorrPlot(XCorrPlotArray)
448     end
449     if ~isempty(PSDPlotArray{1})
450         PSDPlot(PSDPlotArray)
451     end
452     if ~isempty(TempIncPlotArray{1})
453         TempIncPlot(TempIncPlotArray)
454     end
455     if ~isempty(TempStructPlotArray{1})
456         TempStructPlot(TempStructPlotArray)
457     end
458     if ~isempty(PermPlotArray{1})
459         PermPlot(PermPlotArray)
460     end
461
462     % setting uniform plots appearance
463     FigHandles=findobj('Type','figure');                                          %
            gets all figures' handles
464     AxesHandles=findobj('Type','axes');                                           %
            gets all axes' handles
465     set(AxesHandles,'XGrid','on','YGrid','on','Box','on')                         %
            sets all grids and boxes
466     set(FigHandles,'color','w','units','normalized','position',[0.1,0.1,0.8,0.8]) %
            sets color and positions
467 end
468 clear ComputationArray sizeLoop AutoCorrPlotArray PSDPlotArray TempIncPlotArray...
469       TempStructPlotArray XCorrPlotArray PermPlotArray
470
471 %% fill output structure
```

```
472 disp([ParamStruct.StrCode,'filling output data containers...'])
473 StatStruct.Parameters=ParamStruct;                    %all parameters used in the
        statistical computation
474 StatStruct.Diffusion=DiffArray;                       %output structure for the
        calculated diffusion coefficients
475 if ismember('AutoCorr',ParamStruct.Stat)
476
477     StatStruct.AutoCorr=AutoCorrArray;                %output structure for the auto-
            correlation analysis
478 end
479 if ismember('PSD',ParamStruct.Stat)
480     StatStruct.PSD=PSDArray;                          %Computed PSD
481 end
482 if ismember('TempInc',ParamStruct.Stat)
483     StatStruct.TempInc=TempIncArray;                  %Computed temporal increment
484 end
485 if ismember('TempFun',ParamStruct.Stat)
486     StatStruct.TempFun=TempStructArray;               %temporal structure
487 end
488 if ismember('PermEntr',ParamStruct.Stat)
489     StatStruct.PermEntr=PermArray;                    %permutation entropy
490 end
491 StatStruct.Outliers=Outliers;                         %outliers for each array
492 clear AutoCorrArray DiffArray PSDArray TempIncArray TempIncArray XCorrArray...
493       TempStructArray PermArray Outliers
494
495 %% saving, if requested
496 if ParamStruct.Save
497     disp([ParamStruct.StrCode,'saving...'])
498
499     % create folders to save results, foldere name format: current date/number of shot
500     if ~(exist(['~/TurbAnalysisResults/Data/',date],'dir') && exist(['~/
            TurbAnalysisResults/Plots/',date],'dir'))
501         mkdir('~/TurbAnalysisResults/Data',date)
502         mkdir('~/TurbAnalysisResults/Plots',date)
503     end
504     for i=1:length(Shot)
505         SignalString=[num2str(ParamStruct.ProcParam.TimeInt(1)),'_',num2str(ParamStruct
                .ProcParam.TimeInt(2)),'_'];
506         SignalString(SignalString=='.')=[];
507         % cheks for shot directory
508         if ~(exist(['~/TurbAnalysisResults/Data/',date,'/',num2str(Shot(i))],'dir') &&
                exist(['~/TurbAnalysisResults/Plots/',date,'/',num2str(Shot(i))],'dir'))
509             mkdir(['~/TurbAnalysisResults/Data/',date,'/',num2str(Shot(i))])
510             mkdir(['~/TurbAnalysisResults/Plots/',date,'/',num2str(Shot(i))])
511         end
512         % save data
513         save(['~/TurbAnalysisResults/Data/',date,'/',num2str(Shot(i)),'/',[SignalString
                ,datestr(now,'HH:MM:SS')]],'StatStruct','-v7.3')
514
515         % save figures
516         if ParamStruct.Plot
517             savefig(FigHandles,['~/TurbAnalysisResults/Plots/',date,'/',num2str(Shot(i)
                    ),'/',[SignalString,datestr(now,'HH:MM:SS')]])
518         end
519     end
520 else disp([ParamStruct.StrCode,'not saving data on user private file system (should
        have set ''Save''=true at input)'])
521 end
522
523 % opens figures
524 if ParamStruct.Plot
525     set(FigHandles,'visible','on')
526 end
527 disp([ParamStruct.StrCode,'code successfully executed, now showing plots (if requested)
        and ending.'])
528 toc %stop timer and display execution time
529 end
530
531 %% Local Functions
532 % ModifyArray enables the call to multiple probe arrays with one instance
```

```
533 function ArrayList=ModifyArray(Call,ArrayList,Index)
534 [Present,Location]=ismember(Call,ArrayList);
535 if Present
536     switch Index
537         case 1
538             Append={'HFS-BOT','HFS-MID','HFS-TOP'};
539         case 2
540             Append={'LFS-BOT','LFS-MID','LFS-TOP'};
541         case 3
542             Append={'HFS-BOT','LFS-BOT'};
543         case 4
544             Append={'HFS-MID','LFS-MID'};
545         case 5
546             Append={'HFS-TOP','LFS-TOP'};
547         case 6
548             Append={'HFS-BOT','HFS-MID','HFS-TOP','LFS-BOT','LFS-MID','LFS-TOP'};
549         case 7
550             Append={'POL-003','POL-007','POL-011','POL-015'};
551         case 8
552             Append={'HFS-BOT','HFS-MID','HFS-TOP','LFS-BOT','LFS-MID','LFS-TOP','POL
                -003','POL-007','POL-011','POL-015'};
553     end
554     ArrayList=[ArrayList,Append];
555     ArrayList(Location)=[];
556     while ismember(Call,ArrayList)
557         [~,Location]=ismember(Call,ArrayList);
558         ArrayList(Location)=[];
559     end
560 end
561 end
```

```
function MultiExecution(shot,time)
%MULTI_EXECUTION Calls StatComp on multiple time windows for a given shot.
%   CALL: inputs are shot number and matrix of time slices; eg:
%   time=[t1,t2], with t1, t2 being column vectors
%
%   ATTENTION: this function calls StatComp with saving option set to true,
%   ie it saves data each time; no plots produced nor saved.

length=size(time,1);
disp(['MultiExecution: ',num2str(length),' time windows selected for analysis on shot #
    ',num2str(shot),'.'])
disp('MultiExecution: WARNING: this may take a while, wait for it! Don''t warry, each
    result is going to be saved.')

for i=1:length
    proc.TimeInt=time(i,:);     %sets current time window
    disp(['MultiExecution: starting ',num2str(i),' of ',num2str(length),' iterations...
        '])
    [~,~]=StatComp('Shot',shot,'Array',{'ALL'},'Stat',{'ALL'},'Plot',false,'Save',true,
        'CloseAll',true,'ProcParam',proc,'SaveSpace',true);
end
end
```

```
function [AutoCorrStruct,DiffStruct]=AutoCorr(ComputationStruct,AutoParam,DiffStruct)
% AUTOCORR Computes the auto-correlation of the input signal and estimates the ensuing diffusion coefficients.
%
% Author: Aylwin Iantchenko
% Last update: Alessandro Tolio, Apr. 10 2017
%
% Input parameters:
% - ComputationStruct   : structures that contains all the data used for
%                         computation. See DataOrganizer.m for further details
% - AutoParam           : structure that contains the parameters used for computation
% - DiffStruct          : structure used to pass and save the required parameters
%                         and calculated diffusion coefficients
%
% Output parameters:
% - AutoCorrStruct      : structure that contains all the results of the
%                         analysis
% - DiffStruct          : structure that contains the results of the
%                         diffusion parameteres computation
%
% DiffCoeff : estimated values of the diffusion coefficients, using the formulation DperpEMturb=(deltaB/Bphi0)^2*Lc^2/tauGrowth, with
%             deltaB the EM fluctuation level at the the plasma edge, Bphi0 the toroidal magnetic field on the magnetic axis, Lc~Rq is
%             the connection length of the magnetic field, and tauGrowth=mean(tauCorr)+/-std(tauCorr) is an estimate of the most
%             efficient process in limiting the growth of EM turbulent fields, using the auto-correlation coefficient obtained from
%             this analysis and averaged over all probes.
%             NOTE1: DiffCoeff(1,:) uses the raw turbulent Bfield data and mean(tauCorr), DiffCoeff(:,2,:) used the processed (ie:
%                    temppral trends removed) turbulent Bfield data and mean(tauCorr).
%             NOTE2: DiffCoeff(:,1) is the estimated value using mean quantities, DiffCoeff(:,:,2) is the standard deviation using a
%                    Gaussian propagation of uncertainties in deltaBturb and tauCorr.
%             NOTE3: the connection length of the magnetic field Lc~Rq is estimated using q at the plasma edge, ie Lc~6m typically.

%% define and initialise variables
data=ComputationStruct.dataProc;
fs=ComputationStruct.fs;
maxLag=round(AutoParam.TimeLimit*fs);          %maximum lag time
tauCorr=zeros([1,size(data,2)]);               %estimated correlation time
corrCoeff=zeros([2*maxLag+1,size(data,2)]);    %correlation coefficients
tauDelay=zeros([2*maxLag+1,size(data,2)]);     %de-correlation (=delay) times
limit=AutoParam.TimeLimit;

%% auto-correlation computation
factor=AutoParam.FactorTau;                 %factor used to compute the auto-correlation time
x=linspace(0,limit,10^6);                   %time lags used to compute a fitting function
parfor k=1:size(ComputationStruct.Data,2)
    signal=data(:,k);

    % compute the auto-correlation using a maximum lag time
    [CorrVal,tauVal]=xcorr(signal,maxLag,'unbiased');   %auto-correlation computation
    tauVal=(tauVal/fs)';                                %convert the lag indices to time coordinates

    % compute the auto-correlation time through a fit done on results of xcorr
    f=fit(tauVal,CorrVal,'pchipinterp');
    y=feval(f,x);

    % searching for f first minimum
    d=differentiate(f,x);
    x1=x(d>0);
    x1=x1(1);

    % search for time lag x that gives minimum distance from f(x)=factor
    [~,i]=min(abs(y(x<x1)-factor));
```

```
61      tauCorr(k)=x(i);
62      corrCoeff(:,k)=CorrVal;
63      tauDelay(:,k)=tauVal;
64  end
65
66  %% diffusion coefficients computation
67  DiffCoeff(1,1)=(DiffStruct.BmeanRaw*DiffStruct.Lc/DiffStruct.btorAxis)^2/mean(tauCorr);
68  DiffCoeff(1,2)=DiffCoeff(1,1)*sqrt(4*(DiffStruct.BstdRaw/DiffStruct.BmeanRaw)^2 +(std(
        tauCorr)/mean(tauCorr))^2);
69  DiffCoeff(2,1)=(DiffStruct.BmeanProc*DiffStruct.Lc/DiffStruct.btorAxis)^2/mean(tauCorr)
        ;
70  DiffCoeff(2,2)=DiffCoeff(2,1)*sqrt(4*(DiffStruct.BstdProc/DiffStruct.BmeanProc)^2 +(std
        (tauCorr)/mean(tauCorr))^2);
71
72  %% fill output structures.
73  AutoCorrStruct.TauCorr=tauCorr;
74  AutoCorrStruct.MeanTau=mean(tauCorr);
75  AutoCorrStruct.STDTauC=std(tauCorr);
76  DiffStruct.DiffCoeff=DiffCoeff;
77  if ComputationStruct.Average                                %if Averge, then saves related
        data
78      AutoCorrStruct.CMean=mean(corrCoeff,2);
79      AutoCorrStruct.TMean=mean(tauDelay,2);
80      AutoCorrStruct.STDCorr=std(corrCoeff,0,2);
81      AutoCorrStruct.STDTauD=std(tauDelay,0,2);
82      if ComputationStruct.ExcludeOutliers                    %if outliers have to be
            excluded, then do it
83          AutoCorrStruct.Outliers=[];
84          for i=1:length(tauCorr)
85              if abs(tauCorr(i)-AutoCorrStruct.MeanTau)>3*AutoCorrStruct.STDTauC  %
                    exclusion is done for points further than 3*std from mean value
86                  AutoCorrStruct.Outliers=[AutoCorrStruct.Outliers,i];
87              end
88          end
89      end
90  end
91  if ~(~AutoParam.Plot && AutoParam.SaveSpace)    %carries some data over to StatComp
        only if needed for plottting purposes, or if SaveSpace is false
92      AutoCorrStruct.CorrCoeff=corrCoeff;
93      AutoCorrStruct.TauDelay=tauDelay;
94  end
95  end
```

```
function PSDStruct=PSD(ComputationStruct,PSDParam)
% PSD Computes the frequency power spectrum by applying the PSD analysis.
%
% Author: Aylwin Iantchenko
% Last update: Alessandro Tolio, Apr. 10 2017
%
% Input parameters:
% - ComputationStruct   : structures that contains all the data used for
%                         computation. See DataOrganizer.m for further details
% - PSDParam            : structure that contains the parameters used for computation
%
% Output parameters:
% - PSDStruct           : structure that contains all the results of the analysis

%% Initializes variables and loads parameters
% input parameters to pwelch
windowSize=2^8;
overlap=round(windowSize/2);

% variables where each computation will be locally stored
power=zeros(windowSize/2 + 1,size(ComputationStruct.Data,2));
f=zeros(windowSize/2 + 1,size(ComputationStruct.Data,2));
slope=zeros(1,size(ComputationStruct.Data,2));

%% PSD computation
for k=1:size(ComputationStruct.Data,2)
    [power(:,k),f(:,k)] = pwelch(ComputationStruct.dataProc(:,k),hamming(windowSize),
        overlap,[],ComputationStruct.fs);
end

% Convert to k using Taylors' hypotheses
% wave_k=(2*pi*f/DiffStruct.vA); % 4e-03 very rough estimate of the larmor radius

%% if requested, computes slopes without coherent peaks
if ~PSDParam.NoPeaks
    % computes slope for specified range by fitting with a straight line and taking the
         slope of this last one
    for k=1:size(ComputationStruct.dataProc,2)
        P=interp1(f(:,k),power(:,k),PSDParam.SlopeRange);
        linearApprox=polyfit(log10([(PSDParam.SlopeRange(1)),(PSDParam.SlopeRange(2))])
            ,log10([P(1),P(2)]),1);
        slope(k)=linearApprox(1);
    end
else
    % or computes the same but fitting the spectrum through its local minima
    for i=1:size(f,2)
        [~,x]=findpeaks(-power(:,i));   %finds local minima
        F=f(x,i);
        P=power(x,i);
        k=1;
        while k<length(P)
            if F(k)<PSDParam.SlopeRange(1) || F(k)>PSDParam.SlopeRange(2)   %excludes
                points outside of computation range
                F(k)=[];
                P(k)=[];
            else
                if P(k)<P(k+1)  %excludes points that are higher than the previous one
                    F(k+1)=[];
                    P(k+1)=[];
                else
                    k=k+1;
                end
            end
        end
        try
            linearfit=fit(log10(F),log10(P),'poly1');   %tries to compute the fit, if
                not enough points remains, than computes the slope with the original
                algorithm
        catch
            warning(['PSD (',ComputationStruct.ProbeName{k},'): error using the peaks
                elimination algorithm (at leat two points needed in x and y variables
```

```
                to compute fit).'])
65              disp(['PSD (',ComputationStruct.ProbeName{k},'): proceeding with normal
                    computation of the slope...'])
66              P=interp1(f(:,i),power(:,i),PSDParam.SlopeRange);
67              linearfit=fit(log10([(PSDParam.SlopeRange(1));(PSDParam.SlopeRange(2))]),
                    log10([P(1),P(2)])','poly1');
68          end
69          slope(i)=linearfit.p1;
70      end
71  end
72
73  %% invalid data removal, to be coherent with data filter
74  f(1,:)=[];
75  power(1,:)=[];
76  f1=[];
77  p1=[];
78  for k=1:size(f,2)
79      fl=f(:,k);
80      pl=power(:,k);
81      pl(fl<ComputationStruct.FilterFreq(2)-ComputationStruct.FilterFreq(2)/10 | fl>
            ComputationStruct.FilterFreq(3))=[];
82      fl(fl<ComputationStruct.FilterFreq(2)-ComputationStruct.FilterFreq(2)/10 | fl>
            ComputationStruct.FilterFreq(3))=[];
83      f1=[f1,fl]; %#ok<*AGROW>
84      p1=[p1,pl];
85  end
86  f=f1;
87  power=p1;
88
89  %% fill output structure
90  PSDStruct.Slope=slope;
91  if ComputationStruct.Average            %if Averge, then saves related data
92      PSDStruct.SMean=mean(slope);
93      PSDStruct.STDS=std(slope);
94      PSDStruct.PMean=mean(power,2);
95      PSDStruct.STDP=std(power,0,2);
96      PSDStruct.FMean=mean(f,2);
97      PSDStruct.STDF=std(f,0,2);
98      if ComputationStruct.ExcludeOutliers    %if outliers have to be excluded, then do
            it
99          PSDStruct.Outliers=[];
100         for i=1:length(slope)
101             if abs(slope(i)-PSDStruct.SMean)>3*PSDStruct.STDS %exclusion is done for
                    points further than 3*std from mean value
102                 PSDStruct.Outliers=[PSDStruct.Outliers,i];
103             end
104         end
105     end
106 end
107 if ~(~PSDParam.Plot && PSDParam.SaveSpace)  %carries some data over to StatComp only if
        needed for plottting purposes, or if SaveSpace is false
108     PSDStruct.F=f;
109     PSDStruct.Power=power;
110 end
111 end
```

```
function TempIncStruct=TempInc(ComputationStruct,IncParam)
%TEMPINC Performs the temporal increments analysis.
%
% Author: Aylwin Iantchenko
% Last update: Alessandro Tolio, Apr. 10 2017
%
% Input parameters:
% - ComputationStruct   : structures that contains all the data used for
%                         computation. See DataOrganizer.m for further details
% - IncParam            : structure that contains the parameters used for computation
%
% Output parameters:
% - TempIncTemp         : structure that contains all the results of the analysis

%% compute statistics
% compute the increments for specified delay tau. All signals and time points are
    treated directly
dval=interp1(ComputationStruct.Time,ComputationStruct.dataProc,ComputationStruct.Time(
    ComputationStruct.Time<ComputationStruct.Time(end)-IncParam.TauTempInc)+IncParam.
    TauTempInc)-...
    ComputationStruct.dataProc((ComputationStruct.Time<ComputationStruct.Time(end)-
        IncParam.TauTempInc),:);

dval=bsxfun(@minus,dval,mean(dval));    %remove mean
dval=bsxfun(@rdivide,dval,std(dval));   %divide by standard deviation;

% compute PDF of record
[counts,inc]=hist(dval,IncParam.nbins); %get counts of occurence
p=counts./sum(counts(:,1));             %compute probability of occurence, same number
    of occurencies for each column

kurt=kurtosis(dval);    %measure of width
skew=skewness(dval);    %measure of asymmetry

%% Fill output structure
TempIncStruct.kurtosis=kurt;
TempIncStruct.skewness=skew;
if ComputationStruct.Average    %if Averge, then saves related data
    Pmean=mean(p,2);
    Pmean=Pmean/sum(Pmean);     %normalization of mean probability of occurrence
    TempIncStruct.Pmean=Pmean;
    TempIncStruct.KMean=mean(kurt,2);
    TempIncStruct.SMean=mean(skew,2);
    TempIncStruct.STDP=std(p,0,2);
    TempIncStruct.STDS=std(skew);
    TempIncStruct.STDK=std(kurt);
    if ComputationStruct.ExcludeOutliers     %if outliers have to be excluded, then do
        it
        TempIncStruct.Outliers=[];
        for i=1:length(kurt)
            if abs(kurt(i)-TempIncStruct.KMean)>3*TempIncStruct.STDK    %exclusion is
                done for points further than 3*std from mean value
                TempIncStruct.Outliers=[TempIncStruct.Outliers,i];
            end
        end
        for i=1:length(skew)
            if abs(skew(i)-TempIncStruct.SMean)>3*TempIncStruct.STDS    %exclusion is
                done for points further than 3*std from mean value
                TempIncStruct.Outliers=[TempIncStruct.Outliers,i];
            end
        end
    end
end
if ~(~IncParam.Plot && IncParam.SaveSpace)  %carries some data over to StatComp only if
     needed for plottting purposes, or if SaveSpace is false
    TempIncStruct.p=p;
    TempIncStruct.inc=inc;
end
end
```

```
function TempStructTemp=TempStruct(ComputationStruct,TempParam)
% TEMPSTRUCT Computes the temporal structure function, and the associated slopetogether
     with the Hurst exponent.
%
% Author: Aylwin Iantchenko
% Last update: Alessandro Tolio, Apr. 10 2017
%
% Input parameters:
% - ComputationStruct   : structures that contains all the data used for
%                         computation. See DataOrganizer.m for further details
% - TempParam           : structure that contains the parameters used for computation
%
% Output parameters:
% - TempStructTemp      : structure that contains all the results of the analysis

%% load provided paramaters and preallocation
tau=TempParam.TempTau';                 %delay used in computation
range=TempParam.TempSlopeRange;         %specified ranges for slope computation
p=TempParam.TempExp;                    %exponents used in computation
Data=ComputationStruct.dataProc;
Time=ComputationStruct.Time;

% preallocation
S=zeros(length(tau),size(Data,2));
Slope_temp=zeros(length(p),size(Data,2));
Hurst=cell(1,size(range,1));
Slope=Hurst;
for i=1:size(range,1)
    Slope{i}=Slope_temp;
end
temp=cell(1,length(p));
tempmean=zeros(length(tau),length(p));

%% computation
for i=1:length(p) %loop over all exponents for all exponents
    for a=1:size(range,1) %loop over all lag intervals
        localp=p(i);
        % compute the increments for specified delay tau. All signals and time points
            are treated directly
        parfor j=1:length(tau)
            dval=interp1(Time,Data,Time(Time<Time(end)-tau(j))+tau(j))-Data((Time<Time(
                end)-tau(j)),:);
            S(j,:)=mean(abs(dval).^localp);             %compute temporal structure
                function
        end
        % compute slope for specified range
        localrange=range(a,:);
        P=interp1(tau,S,localrange);
        for k=1:size(Data,2)
            linearApprox=polyfit(log([localrange(1),localrange(2)]),log([P(1,k),P(2,k)
                ]),1);
            Slope{a}(i,k)=linearApprox(1);
        end
    end
    temp{i}=S;
    tempmean(:,i)=mean(S,2);
end

% Hurst's exponent computation
for i=1:length(Slope)
    for k=1:size(Slope{i},2)
        Slope_temp(:,k)=Slope{i}(:,k)./p';
    end
    Hurst{i}=Slope_temp(2:end,:);
end

%% output
TempStructTemp.Slope=Slope;
TempStructTemp.Hurst=Hurst;
TempStructTemp.p=p;
if ComputationStruct.Average    %if Averge, then saves related data
```

```
67      TempStructTemp.SMean=tempmean;
68      for i=1:length(Slope)
69          TempStructTemp.SlopeMean(:,i)=mean(Slope{i},2);
70          TempStructTemp.HMean(:,i)=mean(Hurst{i},2);
71          TempStructTemp.STDH(:,i)=std(Hurst{i},0,2);
72          TempStructTemp.STDSlope(:,i)=std(Slope{i},0,2);
73      end
74      if ComputationStruct.ExcludeOutliers    %if outliers have to be excluded, then do
            it
75          TempStructTemp.Outliers=[];
76          for k=1:length(Slope)
77              for i=1:size(Slope{k},2)
78                  if abs(Slope{k}(:,i)-TempStructTemp.SlopeMean(:,k))>3*TempStructTemp.
                        STDSlope(:,k)  %exclusion is done for points further than 3*std
                        from mean value
79                      TempStructTemp.Outliers=[TempStructTemp.Outliers,i];
80                  end
81              end
82              for i=1:size(Hurst{k},2)
83                  if abs(Hurst{k}(:,i)-TempStructTemp.HMean(:,k))>3*TempStructTemp.STDH
                        (:,k)          %exclusion is done for points further than 3*std
                        from mean value
84                      TempStructTemp.Outliers=[TempStructTemp.Outliers,i];
85                  end
86              end
87          end
88      end
89  end
90  if ~(~TempParam.Plot && TempParam.SaveSpace)    %carries some data over to StatComp
        only if needed for plottting purposes, or if SaveSpace is false
91      TempStructTemp.S=temp;
92      TempStructTemp.TempTau=TempParam.TempTau;
93  end
94  end
```

```
function PermEntropyStruct=PermEntropy(ComputationStruct,PermParam)
% PERMENTROPY Computes the permutation entropy of a time series, using precomputed
    tables for efficiency.
%
% Author: Aylwin Iantchenko
% Last update: Alessandro Tolio, Apr. 10 2017
%
% Input parameters:
% - ComputationStruct   : structures that contains all the data used for
%                         computation. See DataOrganizer.m for further details
% - PermParam           : structure that contains the parameters used for computation
%
% Output parameters:
% - PermEntropyStruct   : structure that contains all the results of the analysis

%% initiate internal variables, where computed values will be stored
Size=size(ComputationStruct.dataProc,2);
N=PermParam.N;
fact=factorial(N);
C=zeros(1,Size);
H=zeros(1,Size);
P=zeros(fact,Size);

%% compute PDF of the Shannon's permutation entropy
Tau=round(PermParam.Tau*ComputationStruct.fs);
Data=ComputationStruct.dataProc;
for j=1:Size;
    Pe=zeros(fact,1);
    Pe(:)=1/fact;                                                    %uniform
        probability distribution
    P(:,j)=OrdinalCount(N-1,Tau,Data(:,j));                          %probability
        for ordinal patterns

    % Compute Smax and C
    factor=-2/(log2(fact+1)*(fact+1)/fact-2*log2(2*fact)+log2(fact));
    Smax=log2(fact);
    Q=factor*(Shannon((P(:,j)+Pe)/2)-Shannon(P(:,j))/2-Shannon(Pe)/2);  %* factor if
        normalised or not
    H(j)=Shannon(P(:,j))/Smax;                                       %compute and
        save normalized entropy
    C(j)=Q*H(j);                                                     %compute and
        save Jensen-Shannon complexity
end

%% output
PermEntropyStruct.C=C;
PermEntropyStruct.H=H;
PermEntropyStruct.tauFactored=PermParam.Tau;
if ComputationStruct.Average    %if Averge, then saves related data
    Pmean=mean(P,2);        %ensemble average
    Pmean=Pmean/sum(Pmean); %normalization of ensemble average
    PermEntropyStruct.PMean=Pmean;
    PermEntropyStruct.CMean=mean(C);
    PermEntropyStruct.HMean=mean(H);
    PermEntropyStruct.STDH=std(H);
    PermEntropyStruct.STDC=std(C);
    PermEntropyStruct.STDP=std(P,0,2);
    if ComputationStruct.ExcludeOutliers    %if outliers have to be excluded, then do
        it
        PermEntropyStruct.Outliers=[];
        for i=1:length(C)
            if abs(C(i)-PermEntropyStruct.CMean)>3*PermEntropyStruct.STDC   %exclusion
                is done for points further than 3*std from mean value
                PermEntropyStruct.Outliers=[PermEntropyStruct.Outliers,i];
            end
        end
        for i=1:length(H)
            if abs(H(i)-PermEntropyStruct.HMean)>3*PermEntropyStruct.STDH   %exclusion
                is done for points further than 3*std from mean value
                PermEntropyStruct.Outliers=[PermEntropyStruct.Outliers,i];
            end
```

```
63          end
64      end
65  end
66  if ~(~PermParam.Plot && PermParam.SaveSpace)    %carries some data over to StatComp
        only if needed for plottting purposes, or if SaveSpace is false
67      PermEntropyStruct.P=P;
68  end
69  end
70
71  %% Local functions
72  function S=Shannon(p)
73  % SHANNON Compute Shannon permutation (if P=0, should return zero)
74  p(p==0)=1;
75  S=-sum(log2(p).*p);
76  end
77
78  function PDF=OrdinalCount(D,Tau,X)
79  % ORDINALCOUNT Computes probability for ordinal patterns
80  % initiate internal variables
81  counts=zeros(1,factorial(D+1));
82  k=zeros(1,D);
83  load(['/private/table',num2str(D),'.mat']) %load precomputed table
84  table=eval(['table',num2str(D)]);
85
86  % compute nd for first d+1 elements in time series X
87  for start=1:Tau
88      for l=1:D
89          idx=D*Tau+1+(start-1);
90          k(l)=sum(X(idx-l*Tau)>=X(idx:-Tau:idx-(l-1)*Tau)); %compute i:s required to
                calculate nd
91      end
92
93      n=sum(k*factorial(D+1)./factorial(2:D+1)); %compute label of current ordinal
            pattern
94      counts(n+1)=counts(n+1)+1; %add count to ordinal pattern labelled with nd, +1 as we
             start to count from index 1 (not 0).
95
96      % compute nd for rest
97      for idx=idx:Tau:length(X)-Tau-(start-1)
98          l=sum(X(idx+Tau)<=X(idx:-Tau:idx-D*Tau+1));
99          n=table(n*(D+1)+l+1);
100         counts(n+1)=counts(n+1)+1;
101     end
102 end
103 PDF=counts/sum(counts);
104 end
```

```
function [ComputationStruct,DiffStruct]=ProcessSignal(ComputationStruct,DiffStruct)
% PROCESSSIGNAL Processes the magnetics signals used for turbulence analysis by StatComp:
%               cuts the data according to thespecified time interval, then removes trends
%               and data values above the specified limit.
%
% Author: Aylwin Iantchenko
% Last update: Alessandro Tolio, Apr. 10 2017
%
% Input parameters:
% - ComputationStruct : data structure containing all the data needed for
%                       computation purposes (see organizer.m)
% - DiffStruct        : data structure containing various diffusion parameters
%                       computed inside this function
%
%
% Output parameters:
% - ComputationStruct : data structure now filled with only the
%                       time-varying part of the original dataset
% - DiffStruct        : this is now filled with some diffusion parameters

%% signal processing
% manually given time values encapsulating interval of interest
idx=(ComputationStruct.Time>ComputationStruct.TimeInt(1) & ComputationStruct.Time<ComputationStruct.TimeInt(2));

% check that time interval does not extend beyond the disruption time point.
if ComputationStruct.TimeInt(2)>=ComputationStruct.TimeDisr
    disp(['StatComp: #' int2str(ComputationStruct.Shot) ': processSignal: WARNING: the requested time interval extends beyond the ' , ...
        'disruption time point!']);
   disp(['StatComp: #' int2str(ComputationStruct.Shot) ': processSignal: now cutting the requested time interval to TimeDisr-5msec!']);
   idx=(ComputationStruct.Time>ComputationStruct.TimeInt(1) & ComputationStruct.Time<ComputationStruct.TimeDisr-5e-3);
end

% remove points outside the specified interval
ComputationStruct.Data=ComputationStruct.Data(idx,:);
ComputationStruct.Time=ComputationStruct.Time(idx);

% remove drifts in the input dataset: if the selected TrendPol=0, only the mean value will be removed!
dataProc=zeros(size(ComputationStruct.Data));
for i=1:size(ComputationStruct.Data,2)
    [pp,~,mu]=polyfit(ComputationStruct.Time',ComputationStruct.Data(:,i),ComputationStruct.TrendPol);
    xp=polyval(pp,ComputationStruct.Time',[],mu);
    dataProc(:,i)=ComputationStruct.Data(:,i)-xp;
end
clear pp mu xp i

% replace peaks above specified limit with mean value of signal (calculated without these peaks),
% so as to keep the same signal length for simplicity.
dataLoc=dataProc; %local variable for simplicity
dataLoc(abs(dataLoc)>ComputationStruct.MaxVal)=0;                          %set values above range to zero, to correctly calculate the mean in the next step below
dataProc(abs(dataProc)>ComputationStruct.MaxVal)=mean(abs(dataLoc(:)));  %replace too large values with mean of all probes
ComputationStruct.Data(abs(dataLoc)>ComputationStruct.MaxVal)=0;          %set values above range to zero, for data with drifts, used in diffusion calculation below
clear dataLoc

% process turbulent Bfield data: remove mean value and divide by standard deviation, deltaBproc=(deltaB-mean(deltaB))/sigma(deltaB).
dataProc=bsxfun(@minus,dataProc,mean(dataProc));        %remove mean
dataProc=bsxfun(@rdivide,dataProc,std(dataProc));       %divide by standard deviation

%% signal filtering, if requested
```

```
59  if ComputationStruct.FilterSignal
60      % design of the FIR bandpass filter.
61      if isempty(ComputationStruct.FilterFreq)
62          fcuts=[5e3,10e3,ComputationStruct.fs/2-10e3,ComputationStruct.fs/2-5e3];
63          ComputationStruct.FilterFreq=fcuts;
64      else
65          fcuts=ComputationStruct.FilterFreq;
66      end
67      mags=[0,1,0];
68      devs=[1e-5,1e-5,1e-5];
69      [n,Wn,beta,ftype]=kaiserord(fcuts,mags,devs,ComputationStruct.fs);
70      b=fir1(n,Wn,ftype,kaiser(n+1,beta),'noscale');
71      clear fcuts mags devs n Wn beta ftype
72      parfor i=1:size(dataProc,2)
73          % apply the bandpass FIR filter to the input data.
74          dataProc(:,i)=filter(b,1,dataProc(:,i));
75
76          % subtract the mean value and divide by the standard deviation, has to be re-
                done here due to the FIR filtering.
77          dataProc(:,i)=(dataProc(:,i)-mean(dataProc(:,i)))/std(dataProc(:,i));
78      end
79  end
80
81  %% compute diffusion related parameters.
82  % get time averaged electron density and temperature using the raw Thomson measurements
        .
83  % check that the specified time interval range does not extend beyond the time range
        for which the Thomson measurements are performed.
84  % if this is the case, use only the start end end values of the latter.
85  mdsopen(ComputationStruct.Shot);
86  nepr=tdi('\results::thomson:ne');                %get electron density (unit =[m^-3])
87  tepr=tdi('\results::thomson:te');                %get electron temperature (unit =[eV])
88  bphi=tdi('\magnetics::iphi');                    %current in the toroidal field coils (
        unit =[A])
89  mdsclose;
90  timeNeTe=nepr.dim{1};                            %get time vector of Thomson data (unit
        =[sec])
91  zCoords=nepr.dim{2};                             %get Z-coordinates of measurements
        points (unit =[m])
92  timeBtor=bphi.dim{1};                            %get time vector of the toroidal Bfield
         data (unit =[sec])
93  btorAxis=192.e-7*abs(bphi.data)/0.88;            %toroidal field on the magnetic axis (
        unit =[T])
94  btorEdge=192.e-7*abs(bphi.data)/(0.88+0.25);     %toroidal field at the plasma edge (
        unit =[T])
95
96  % select only the data in the correct time window.
97  if (timeNeTe(1)>ComputationStruct.TimeInt(1))
98      MeasVal=[timeNeTe(1),0];
99  else MeasVal=[ComputationStruct.TimeInt(1),0];
100 end
101 if (timeNeTe(end)<ComputationStruct.TimeInt(2))
102     MeasVal(2)= timeNeTe(end);
103 else MeasVal(2)=ComputationStruct.TimeInt(2);
104 end
105 idx=(timeNeTe>MeasVal(1) & timeNeTe<MeasVal(2));
106 ne=mean(nepr.data(idx,:));
107 Te=mean(tepr.data(idx,:));
108 idx=(timeBtor>MeasVal(1) & timeBtor<MeasVal(2));
109 btorEdge=mean(btorEdge(idx));
110 btorAxis=mean(btorAxis(idx));
111
112 % get the values of the electron density and temperature at the edge
113 neEdge=ne(1);
114 teEdge=Te(1);
115
116 % get an estimate for the turbulent magnetic field averaged over all probes and time
        points.
117 BmeanRaw=mean(abs(ComputationStruct.Data(:)-mean(ComputationStruct.Data(:)))); %mean
        value of raw deltaB data
118 BstdRaw=std(abs(ComputationStruct.Data(:)-mean(ComputationStruct.Data(:))));
```

```
119 BmeanProc=mean(abs(dataProc(:)));          %mean value of processed deltaB data
120 BstdProc=std(abs(dataProc(:)));
121
122 % estimate the Alfven speed, the ion sound speed at the plasma edge for a deuterium
        plasma
123 % (AA=2, ZZ=1) with gamma=5/3.
124 vA=2.18e16*btorEdge/sqrt(2*neEdge);     %Alfven speed, unit =[m/s]
125 vS=9.79e3*sqrt((5/3)*teEdge/2);         %ion sound speed, unit =[m/s]
126
127 % computes some diffusion-related data only if more than 1 elemtes are retrieved form
        TS system
128 if numel(ne)>1
129     % remove values outside the plasma, associated to a very low density.
130     Te(ne<100)=[];
131     zCoords(ne<100)=[];
132     ne(ne<100)=[];
133
134     % estimate the local density and temperature gradients.
135     grad_ne=abs((ne(2)-ne(1))/(zCoords(2)-zCoords(1)));
136     grad_Te=abs((Te(2)-Te(1))/(zCoords(2)-zCoords(1)));
137
138     % estimate the ion (and electron) drift velocity
139     % NOTE: assuming ne=ni and Te=Ti at the plasma edge, the ion and electron drift
            velocities are the same for ZZ=1!
140     vD=1e3*grad_ne/neEdge;                  %ion (and electron) drift velocity, unit =[
            m/s]
141
142     % partially fill the output structure
143     DiffStruct.vD=vD;                       %estimated ion/electron drift velocity at
            the plasma edge (unit =[m/sec])
144     DiffStruct.LneEdge=neEdge/grad_ne;      %computed density scale length at the
            plasma edge (unit =[m])
145     DiffStruct.LteEdge=teEdge/grad_Te;      %computed temperature scale length at the
            plasma edge (unit =[m])
146 else
147     disp('ProcessSignal: less than two Thomson points in the selected time slice:
            cannot compute gradient of ne, Te.')
148 end
149
150 %% save the output structure with the diffusion coefficients for the current signal (
        shot and array).
151 DiffStruct.btorAxis=btorAxis;
152 DiffStruct.btorEdge=btorEdge;
153 DiffStruct.BmeanRaw=BmeanRaw;
154 DiffStruct.BstdRaw=BstdRaw;
155 DiffStruct.BmeanProc=BmeanProc;
156 DiffStruct.BstdProv=BstdProc;
157 DiffStruct.vA=vA;                       %estimated Alfven speed at the plasma edge (
        unit =[m/sec])
158 DiffStruct.vS=vS;                       %estimated sound speed at the plasma edge (unit
         =[m/sec])
159 DiffStruct.neEdge=neEdge;               %computed volume and time averaged density at
        the plasma edge (unit =[m^-3])
160 DiffStruct.teEdge=teEdge;               %computed volume and time averaged temperature
        at the plasma edge (unit =[eV])
161
162 ComputationStruct.dataProc=dataProc;
163 end
```

```
function ComputationArray=DataOrganizer(ParamStruct,Signal,Time,SizeLoop)
% DATAORGANIZER Fucntions that constructs the cell array containing all the data structures needed for computation.
%
% Author: Alessandro Tolio, Nov. 8 2016
% Last update: Alessandro Tolio, Apr. 10 2017
%
% Input parameters:
% - ParamStruct:    this is the data structures containing all the parameters
%                   used for computation. I am planning to either change
%                   this structure or even getting rid of it
% - Signal:        cell array containing all the data points related to the
%                   probes' array. It is obtained from FMPloader
% - Time:           as the above one, but containing the time points of the
%                   measurements
% - SizeLoop:       total number of interations of the computation
%
% Output paramter:
% - ComputationArray: cell array containing computationStruct, which is a
%                     data structure that contains the relevant data for
%                     each combinantion of shot and probes' array



%% initialization; preallocation and shortcuts are made in order to optimize the parfor loop
ComputationStruct=struct(...
    'Shot',[],...           %shot's number
    'ShotIdx',[],...        %shot's index
    'ArrayIdx',[],...       %array's index
    'SignalIdx',[],...      %signal's index
    'Data',[],...           %data related to the given array
    'BBunits',[],...        %unit of measurement for data points
    'Time',[],...           %vector of time related to data [s]
    'TimeDisr',[],...       %disruption time [s]
    'fs',[],...             %sampling frequency
    'Stat',[],...           %statistics for current shot/array combination
    'TimeInt',[],...        %time interval index, used to select the correct time interval afterwards
    'MaxVal',[],...         %index of the maximum value
    'TrendPol',[],...       %index for the polynomial trend
    'PlotStr',[],...        %string used for plotting purposes
    'ProbeName',[],...      %string containing the probe name, used for plotting purposes
    'Average',[],...        %compute average values
    'ProbeNameAvg',[],...   %as above, but used if blnAverage==true
    'FilterSignal',[],...   %used to filter the signal
    'FilterFreq',[],...     %filter's frequency
    'ExcludeOutliers',[]... %to exclude outliers from results
    );
ComputationArray=cell(1,SizeLoop);
ComputationArray(1,:)={ComputationStruct};
shot=ParamStruct.Shot;
Array=ParamStruct.Array;
Stat=ParamStruct.Stat;
Disr=ParamStruct.TimeDisr;
TimeInt=ParamStruct.ProcParam.TimeInt;
MaxVal=ParamStruct.ProcParam.MaxVal;
TrendPol=ParamStruct.ProcParam.TrendPol;
BBunits=ParamStruct.BBunits;
fs=ParamStruct.fs;
ProbeName=ParamStruct.ProbeName;
ProbeNameAvg=ParamStruct.ProbeNameAvg;
Average=ParamStruct.Average;
FilterSignal=ParamStruct.FilterSignal;
FilterFreq=ParamStruct.FilterFreq;
ExcludeOutliers=ParamStruct.ExcludeOutliers;

%% selection of the processing values
% this is done because the calling function is said to be able to manage some processing parameters in a certain way
```

```
67  if (size(TimeInt,2)~=2) && ~mod(size(TimeInt,2),2)
68      TimeInterval=size(TimeInt,2)/2;
69  else TimeInterval=1;
70  end
71
72  if (size(MaxVal,2)~=2) && ~mod(size(MaxVal,2),2)
73      MaxValValue=size(MaxVal,2)/2;
74  else MaxValValue=1;
75  end
76
77  if (size(TrendPol,2)~=2) && ~mod(size(TrendPol,2),2)
78      TrendVal=size(TrendPol,2)/2;
79  else TrendVal=1;
80  end
81
82  llArray=length(ParamStruct.Array);
83  llShot=length(ParamStruct.Shot);
84
85  %% filling of the output cell array
86  parfor i=1:SizeLoop
87      k=ceil(i/llArray);                                  %shot index
88      h=i-llArray*(k-1);                                  %array index
89      ComputationArray{i}.Shot=shot(k); %#ok<*PFBNS>
90      ComputationArray{i}.ShotIdx=k;
91      ComputationArray{i}.Array=Array(h);
92      ComputationArray{i}.ArrayIdx=h;
93      ComputationArray{i}.Data=Signal{i};
94      ComputationArray{i}.Time=Time{i};
95      ComputationArray{i}.TimeDisr=Disr(k);
96      ComputationArray{i}.Stat=Stat;
97      ComputationArray{i}.BBunits=BBunits{i};
98      ComputationArray{i}.fs=fs(i);
99      ComputationArray{i}.ProbeName=ProbeName{h};
100     ComputationArray{i}.ProbeNameAvg=ProbeNameAvg{i};
101     ComputationArray{i}.Average=Average;
102     ComputationArray{i}.FilterSignal=FilterSignal;
103     ComputationArray{i}.FilterFreq=FilterFreq;
104     ComputationArray{i}.ExcludeOutliers=ExcludeOutliers;
105     if TimeInterval==SizeLoop
106         ComputationArray{i}.TimeInt=TimeInt([2*i-1,2*i]);
107     elseif TimeInterval==llShot
108         ComputationArray{i}.TimeInt=TimeInt([2*k-1,2*k]);
109     else ComputationArray{i}.TimeInt=TimeInt([1,2]);
110     end
111     if (MaxValValue==SizeLoop) && (MaxValValue~=1)
112         ComputationArray{i}.MaxVal=MaxVal(i);
113     elseif MaxValValue==llShot
114         ComputationArray{i}.MaxVal=MaxVal(2*k-1);
115     else ComputationArray{i}.MaxVal=MaxVal;
116     end
117     if TrendVal==SizeLoop && (MaxValValue~=1)
118         ComputationArray{i}.TrendPol=TrenPol(i);
119     elseif TrendVal==llShot
120         ComputationArray{i}.TrendPol=TrendPol(2*k-1);
121     else ComputationArray{i}.TrendPol=TrendPol;
122     end
123     ComputationArray{i}.PlotStr=['(' num2str(ComputationArray{i}.TimeInt(1),'%6.4f') '-
        ' num2str(ComputationArray{i}.TimeInt(2),'%6.4f') '[s])'];
124 end
125 end
```

```
function [AutoStruct,PSDStruct,IncStruct,FunStruct,PermStruct,Outliers] =
    ComputeNewAverages(AutoStruct,PSDStruct,IncStruct,FunStruct,PermStruct)
% COMPUTENEWAVERAGES Computes new mean and std values if outliners are found.
%
% Author: Alessandro Tolio, March 2016
% Last update: Alessandro Tolio, Apr. 10 2017
%
% Input parameters: all results structures
%
% Output parameters:
%  - same structures with new mean and std values
%  - Outliers : contains all found outliers

%% computation
% computation is done twice in order to exclude some more outliers
for k=1:2
    idx=[];

    % save all outliers from each computation
    if ~isempty(AutoStruct)
        idx=[idx,AutoStruct.Outliers];
    end
    if ~isempty(PSDStruct)
        idx=[idx,PSDStruct.Outliers];
    end
    if ~isempty(IncStruct)
        idx=[idx,IncStruct.Outliers];
    end
    if ~isempty(FunStruct)
        idx=[idx,FunStruct.Outliers];
    end
    if ~isempty(PermStruct)
        idx=[idx,PermStruct.Outliers];
    end

    if ~isempty(idx) %skips all computation if not outliesr are found

        % computation on AutoCorr results
        if ~isempty(AutoStruct)
            temp=[];
            for i=1:length(AutoStruct.TauCorr)
                if ~ismember(i,idx)
                    temp=[temp,AutoStruct.TauCorr(i)]; %#ok<*AGROW>
                end
            end
            AutoStruct.MeanTau=mean(temp);
            AutoStruct.STDTauC=std(temp);
            if k==1 %if first iteration searches for new outliers
                AutoStruct.Outliers=[];
                for i=1:length(AutoStruct.TauCorr)
                    if abs(AutoStruct.TauCorr(i)-AutoStruct.MeanTau)>3*AutoStruct.
                        STDTauC
                        AutoStruct.Outliers=[AutoStruct.Outliers,i];
                    end
                end
            end
        end

        % computation on PSD results
        if ~isempty(PSDStruct)
            temp=[];
            for i=1:length(PSDStruct.Slope)
                if ~ismember(i,idx)
                    temp=[temp,PSDStruct.Slope(i)];
                end
            end
            PSDStruct.SMean=mean(temp);
            PSDStruct.STDS=std(temp);
            if k==1 %if first iteration searches for new outliers
                for i=1:length(PSDStruct.Slope)
                    PSDStruct.Outliers=[];
```

```
                    if abs(PSDStruct.Slope(i)-PSDStruct.SMean)>3*PSDStruct.STDS
                        PSDStruct.Outliers=[PSDStruct.Outliers,i];
                    end
                end
            end
        end

        % computation on TimeInc results
        if ~isempty(IncStruct)
            temp=[];
            for i=1:length(IncStruct.kurtosis)
                if ~ismember(i,idx)
                    temp=[temp,IncStruct.kurtosis(i)];
                end
            end
            temp1=[];
            for i=1:length(IncStruct.skewness)
                if ~ismember(i,idx)
                    temp1=[temp1,IncStruct.skewness(i)];
                end
            end
            IncStruct.KMean=mean(temp);
            IncStruct.SMean=mean(temp1);
            IncStruct.STDK=std(temp);
            IncStruct.STDS=std(temp1);
            if k==1 %if first iteration searches for new outliers
                IncStruct.Outliers=[];
                for i=1:length(IncStruct.kurtosis)
                    if abs(IncStruct.kurtosis(i)-IncStruct.KMean)>3*IncStruct.STDK
                        IncStruct.Outliers=[IncStruct.Outliers,i];
                    end
                end
                for i=1:length(IncStruct.skewness)
                    if abs(IncStruct.skewness(i)-IncStruct.SMean)>3*IncStruct.STDS
                        IncStruct.Outliers=[IncStruct.Outliers,i];
                    end
                end
            end
        end

        % computation on TempStruct results
        if ~isempty(FunStruct)
            for j=1:length(FunStruct.Slope)
                temp=[];
                for i=1:size(FunStruct.Slope{j},2)
                    if ~ismember(i,idx)
                        temp=[temp,FunStruct.Slope{j}(:,i)];
                    end
                end
                temp1=[];
                for i=1:size(FunStruct.Hurst{j},2)
                    if ~ismember(i,idx)
                        temp1=[temp1,FunStruct.Hurst{j}(:,i)];
                    end
                end
                FunStruct.SlopeMean(:,j)=mean(temp,2);
                FunStruct.HMean(:,j)=mean(temp1,2);
                FunStruct.STDSlope(:,j)=std(temp,0,2);
                FunStruct.STDH(:,j)=std(temp1,0,2);
                if k==1 %if first iteration searches for new outliers
                    FunStruct.Outliers=[];
                    for i=1:size(FunStruct.Slope{j},2)
                        if abs(FunStruct.Slope{j}(:,i)-FunStruct.SlopeMean(:,j))>3*
                            FunStruct.STDSlope(:,j)
                            FunStruct.Outliers=[FunStruct.Outliers,i];
                        end
                    end
                    for i=1:size(FunStruct.Hurst{j},2)
                        if abs(FunStruct.Hurst{j}(:,i)-FunStruct.HMean(:,j))>3*
                            FunStruct.STDH(:,j)
                            FunStruct.Outliers=[FunStruct.Outliers,i];
```

```
                        end
                    end
                end
            end
        end

        % computation on PermEntr resuslts
        if ~isempty(PermStruct)
            temp=[];
            for i=1:length(PermStruct.C)
                if ~ismember(i,idx)
                    temp=[temp,PermStruct.C(i)];
                end
            end
            temp1=[];
            for i=1:length(PermStruct.H)
                if ~ismember(i,idx)
                    temp1=[temp1,PermStruct.H(i)];
                end
            end
            PermStruct.CMean=mean(temp);
            PermStruct.HMean=mean(temp1);
            PermStruct.STDC=std(temp);
            PermStruct.STDH=std(temp1);
            if k==1 %if first iteration searches for new outliers
                PermStruct.Outliers=[];
                for i=1:length(PermStruct.C)
                    if abs(PermStruct.C(i)-PermStruct.CMean)>3*PermStruct.STDC
                        PermStruct.Outliers=[PermStruct.Outliers,i];
                    end
                end
                for i=1:length(PermStruct.H)
                    if abs(PermStruct.H(i)-PermStruct.HMean)>3*PermStruct.STDH
                        PermStruct.Outliers=[PermStruct.Outliers,i];
                    end
                end
            end
        end
    end
end

%% outliers saving
if~isempty(idx)
    idx=sort(idx);
    i=1;
    while i<length(idx)     %getting rid of repeated numbers
        j=find(idx==idx(i));
        idx(j(2:end))=[];
        i=i+1;
    end
end
Outliers=idx;
end
```

Bibliography

1. IEA, "Key World Energy Statistics", tech. rep.
2. FCCC, *ADOPTION OF THE PARIS AGREEMENT.* 2015, (https://tinyurl.com/jdlhklc).
3. F. Wagner, *European Physical Journal Plus* **131**, 1–21 (2016).
4. I. Pioro, *Handbook of Generation IV Nuclear Reactors* (Elsevier Science & Technology, 2016), (https://tinyurl.com/y98rph5c).
5. K. Ikeda, *Nuclear Fusion* **50**, 014002 (2010).
6. L. Coblentz, "21st ITER Council affirms steady, measurable project progress", tech. rep., pp. 1–2, (https://tinyurl.com/y7moyc9s).
7. G. Federici *et al.*, *Fusion Engineering and Design* **89**, 882–889 (2014).
8. J. P. Freidberg, *Plasma physics and fusion energy* (Cambridge University Press, 2007), p. 671, (https://tinyurl.com/y72hg9os).
9. D. R. Nicholson, *Introduction to plasma theory* (Wiley, 1983), (https://books.google.it/books?id=fyRRAAAAMAAJ).
10. J. P. Freidberg, *Ideal MHD* (Cambridge University Press, Cambridge, 2014), (https://www.cambridge.org/core/books/ideal-mhd/EB82D616B5CBE1748E8D8
11. U. Stroth, *Plasmaphysik: Phänomene, Grundlagen, Anwendungen* (Vieweg+Teubner Verlag, ed. 1, 2011), p. 488.
12. G. Dif-Pradalier *et al.*, *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics* **82** (2010).
13. L. Davidson, *An Introduction to Turbulence Models*, 2003.
14. B. Ryden, in, pp. 65–76, (https://tinyurl.com/ybdqxmr4).
15. R. Bruno *et al.*, *Living Reviews in Solar Physics* **10**, (https://tinyurl.com/yb427tjv) (2013).
16. G. I. Taylor, *Proceedings of the Royal Society of London. Series A - Mathematical and Physical Sciences* **164**, 476–490 (1938).
17. M. R. Brown *et al.*, *Physics of Plasmas* **22**, 055601 (2015).
18. P. R. Halmos, *Lectures on Ergodic Theory* (AMS Chelsea Publishing, 1956).
19. A. N. Kolmogorov, *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* **434**, 9–13 (1991).

20. A. N. Kolmogorov, *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* **434**, 15–17 (1991).

21. A. N. Kolmogorov, *Journal of Fluid Mechanics* **13**, 82 (1962).

22. R. H. Kraichnan, *Physics of Fluids* **8**, 1385 (1965).

23. P. S. Iroshnikov, *Soviet Astronomy* **7**, 556 (1964).

24. J. Cho *et al.*, in *Turbulence and Magnetic Fields in Astrophysics* (Springer Berlin Heidelberg, Berlin, Heidelberg, 2003), vol. 614, pp. 56–98, (`https://tinyurl.com/y73labns`).

25. W.-C. Müller *et al.*, *AIP Conference Proceedings* **932**, 52–57 (2007).

26. B. Sorbom *et al.*, *Fusion Engineering and Design* **100**, 378–405 (2015).

27. M. Fontana *et al.*, *Review of Scientific Instruments* **88**, 083506 (2017).

28. A. Fasoli, *Nuclear Fusion* **55** (2015).

29. B. Labit *et al.*, *Nuclear Materials and Energy* **12**, 1015–1019 (2017).

30. J. Sinha, PhD thesis, École Polytechnique Fédérale de Lausanne EPFL, 2017, (`https://tinyurl.com/yca9g6ms`).

31. J.-M. Moret *et al.*, *Review of Scientific Instruments* **69**, 2333–2348 (1998).

32. D. Testa *et al.*, *Fusion Science and Technology* **59**, 376–396 (2011).

33. D. Testa *et al.*, *Fusion Engineering and Design* **96-97**, 989–992 (2015).

34. D. J. DeTroye *et al.*, *US Army Research Laboratories*, 1–22 (1994).

35. R. F. Heeter *et al.*, *Review of Scientific Instruments* **71**, 4092–4106 (2000).

36. H. L. Grant *et al.*, *Journal of Fluid Mechanics* **12**, 241 (1962).

37. ITER, *Nuclear Fusion* **39**, 2176–2249 (1999).

38. E. J. Doyle, *Nuclear Fusion* **47**, S18–S127 (2007).

39. A. Greco *et al.*, *The Astrophysical Journal* **691**, L111–L114 (2009).

40. H. E. Hurst, *International Association of Scientific Hydrology. Bulletin* **1**, 13–27 (1956).

41. G. Wang *et al.*, *Physics of Plasmas* **7**, 1181–1183 (2000).

42. B. P. van Milligen *et al.*, *Physics of Plasmas* **7**, 5267–5268 (2000).

43. G. Antar *et al.*, *Physics of Plasmas* **7**, 5269–5271 (2000).

44. B. B. Mandelbrot, *Physica Scripta* **32**, 257–260 (1985).

45. B. B. Mandelbrot, *Science* **156**, 636–638 (1967).

46. B. B. Mandelbrot, *The Fractal Geometry of Nature*, p. 498.

47. T. Gneiting *et al.*, *SIAM Review* **46**, 269–282 (2004).

48. C. Bandt *et al.*, *Physical Review Letters* **88**, 174102 (2002).

49. O. A. Rosso *et al.*, *Physical Review Letters* **99**, 154102 (2007).

50. R. P. Lepping *et al.*, *Space Science Reviews* **71**, 207–229 (1995).

51. W. H. Matthaeus *et al.*, *Physical Review Letters* **95**, 231101 (2005).

52. M. R. Brown *et al.*, *Plasma Sources Science and Technology* **23**, 063001 (2014).

53. A. A. Schekochihin *et al.*, *The Astrophysical Journal Supplement Series* **182**, 310–377 (2009).

54. F. Sahraoui *et al.*, *Physical Review Letters* **102**, 231102 (2009).

55. E. Marsch *et al.*, *Nonlinear Processes in Geophysics* **4**, 101–124 (1997).

56. C. Marini, PhD thesis, École Polytechnique Fédérale de Lausanne EPFL, 2017, (`https://tinyurl.com/ya8qwu7m`).

57. M. Fontana *et al.*, *Review of Scientific Instruments* **88**, 83506 (2017).

58. C. X. Yu *et al.*, *Physics of Plasmas* **10**, 2772–2779 (2003).

Ringraziamenti

Arrivare alla fine del mio percorso universitario è stato molto impegnativo. Tante soddisfazioni e molti momenti di sconforto si sono alternati in questi ultimi sei anni. I miei ringraziamenti vanno soprattutto a chi mi ha aiutato, consciamente e non, nei miei momenti bassi.

Ringrazio i miei genitori, Doriano e Vania, per avermi supportato nella scelta di andare a studiare a Milano. Ciò mi ha dato la possibilità di conoscere persone fantastiche durante i miei studi.

Voglio ringraziare i professori Passoni e Testa. Il primo per essere stato un ottimo docente, avermi dato gli strumenti per decidere di cambiare rotta alla fine della Laurea Triennale e aver accettato di farmi da relatore. Il secondo per avermi calorosamente accolto allo Swiss Plasma Center ed essere stato un'ottima guida durante i miei mesi svizzeri.

Grazie infinite a Marta, che mi ha sostenuto sempre, soprattutto durante questi ultimi mesi del mio percorso. Ancora oggi non avrei finito gli esami senza di lei.

Grazie di cuore ai Saggi, al secolo Andrea e Matteo, superbi coinquilini e amici veri. Se sono cresciuto molto durante quest'avventura è anche grazie al fatto di averla affrontata giorno dopo giorno al loro fianco.

Grazie ai fenomeni di Ingegneria dei Materiali: Andrea, Federico, Francesco, Luca, Riccardo, Stefano e Tommaso. Gli amici che ho trovato in questa fantastica città sono il vero tesoro di questi anni. Menzione speciale per Akshay e Vincenzo, arrivati dopo, ma con cui mi sono trovato subito bene.

Grazie ai miei primi colleghi di ufficio: Angelica, Marco e Sonia. Senza le loro domande su troppe cose avrei finito molto prima il mio lavoro, ma non sarebbe stato altrettanto piacevole.

Grazie allo Special Nulcear Materials group: Alessandro, Andrea, Francesco, Sonia e Zeno. Andare a lezione è spesso stato più leggero grazie a voi. Ci siamo difesi bene.

Ringrazio Pietro, che si è fatto sentire più di me e non se lo merita.

Ringrazio anche l'Altra Scuola Politecnica, che non mi ha mai fatto sentire escluso, quando invece la burocrazia è stata di ferro.

Menzione d'onore per Guido, che per un game c'è sempre stato.

Last, but not least, voglio ringraziare il mio fratellino, Niccolò. Saremo sempre il più forte alleato dell'altro. Ti voglio bene.

ggwp

... et voilà!

This work has been carried out within the framework of the EUROfusion Consortium and has received funding from the European Union's Horizon 2020 research and innovation program. The views and opinions expressed herein do not necessarily reflect those of the European Commission.