Disruption prediction at the JET tokamak using the geometry of wavelet distributions

Giorgos Karagounis

Promotor: prof. dr. ir. Guido Van Oost
Begeleider: dr. Geert Verdoolaege

Masterproef ingediend tot het behalen van de academische graad van
Master in de ingenieurswetenschappen: toegepaste natuurkunde

Vakgroep Toegepaste Fysica
Voorzitter: prof. dr. ir. Christophe Leys
Faculteit Ingenieurswetenschappen en Architectuur
Academiejaar 2011-2012

UNIVERSITEIT
GENT

Foreword

The courses on plasma physics and plasma and fusion technology were my first touch with the subject of nuclear fusion. The subject interested me immediately: it was a subject linked to the most advanced physics of our time, but it carried an immense promise. I cannot think of a second field in physics with the same degree of importance for our future. The choice of a master thesis was thus not difficult for me. I never visited a second department in search of an alternative.

During my thesis year, I learned a lot, but certainly not most about fusion. Writing a thesis has more to do with learning to handle unexpected situations, periods which seem void, but are not; and communication. For all of these, I got help from different sides and I would like to thank each of them for being there for me.

First of all, I want to thank my supervising professor, Guido Van Oost. Because of him it was possible to have an intership at JET, last summer. Working in this international environment had a tremendous impact on my professional and personal self. Of course I would like to sincerely thank Geert Verdoolaege, for the administrative burdens which preceded the internship, his guidance during the first days in England and the time he spended to help me with each of my concerns, no matter how small they were.

In my personal environment, I would like to thank my mother first. She probably heard most of my frustrations, which of course did not miss. Thanks mom, for reading each page of this work, no matter the difficulties with the scientific language. I would like to thank my father, with whom I talked numerous times on the phone about the evolution of my work. And also thanks dad, and granddad, to stimulate my interest in science so fervently. Finally, I would like to thank my girlfriend, Britt, for supporting me this year and long before that.

I would like to thank dr. Murari and dr. Vega for their guidance during my internship at JET and for providing me with the necessary JET data, without which this work would not be possible. Also thanks to dr. de Vries, for the access he provided to his results concerning the disruption causes at JET.

For all persons I thanked here, probably I forget to thank one more. Therefore, once more, thanks to everyone which helped me to become the person I am now and thus contributed to this work.

Giorgos Karagounis
Ghent, 4^{th} of June, 2012

Permission for usage

The author gives permission to make this master dissertation available for consultation and to copy parts of this dissertation for personal use.
In the case of any other use, the limitations of the copyright have to be respected, in particular with regard to the obligation to state expressly the source when quoting results from this master dissertation.
De auteur geeft de toelating deze scriptie voor consultatie beschikbaar te stellen en delen van de scriptie te kopiëren voor persoonlijk gebruik.
Elk ander gebruik valt onder de beperkingen van het auteursrecht, in het bijzonder met betrekking tot de verplichting de bron uitdrukkelijk te vermelden bij het aanhalen van resultaten uit deze masterproef.

Giorgos Karagounis
Ghent, 4^{th} of June, 2012

Disruption prediction at the JET tokamak using the geometry of wavelet distributions

by

Giorgos Karagounis

Master thesis submitted in fulfillment of the requirements for the degree of

Master of Engineering Sciences:

Engineering Physics

Academic year 2011–2012

Promotor: Prof. Dr. Ir. G. Van Oost

Supervisor: Dr. G. Verdoolaege

Faculty of Engineering and Architecture

Ghent University

Department of Applied Physics

Head of Department: Prof. Dr. Ir. C. Leys

Summary

In this study, we investigate the use of wavelet statistics in the prediction of plasma disruptions. A probabilistic model is fitted to the wavelet statistics. A geometric framework is presented, which allows to measure differences between probability distributions. This combination improves the classification rates as compared to previous approaches based on Fourier statistics. The proposed method especially improves the period between the prediction and the actual disruption. The most limiting factor proves to be the sampling rate of the diagnostics.

Keywords: Plasma disruption, wavelet decomposition, generalised Gaussian distribution, geodesic distance, fusion.

Herkenning van disrupties in de JET-tokamak gebruikmakend van de geometrie van wavelet distributies

Giorgos Karagounis

Begeleiders: prof. dr. ir. Guido Van Oost, dr. Geert Verdoolaege

Abstract— **Plasmadisrupties zijn instabiliteiten die voorkomen in experimentele fusieplasmas opgesloten in *tokamaks*. Disrupties induceren grote krachten op het raamwerk van een tokamak en moeten vermeden worden. In deze studie werd automatische detectie van disrupties verwezenlijkt door het gebruik van waveletdecompositie van plasmasignalen. De statistiek van de waveletcoëfficiënten werd gemodelleerd door een veralgemeende Gaussiaanse distributie. Verschillende classifiers werden ontwikkeld. Om de gelijkenis tussen distributies te bepalen werd gebruik gemaakt van de *Rao geodetische afstand* afkomstig uit de informatiemeetkunde. De resultaten werden vergeleken met classifiers gebaseerd op de statistiek van fouriercoëfficiënten. Onze methode levert een verbetering op in het voorspellen van disrupties. Het wordt mogelijk om disrupties tot 360 ms voor de disruptie betrouwbaar te voorspellen. Tenslotte werd het vermogen van de classifiers getest om de oorzaak van een disruptie te herkennen. Opnieuw werden betere resultaten behaald door het gebruik van wavelet statistieken.**

Trefwoorden— **Plasmadisrupties, fusie, wavelet decompositie, veralgemeende Gaussiaanse distributie, geodetische afstand**

I. Inleiding

GE controleerde nucleaire fusie vormt een milieuvriendelijke en een nagenoeg onuitputtelijke bron van energie voor de toekomst. Het meest geavanceerde concept om dit te bereiken is de gecontroleerde fusie van een plasma bestaande uit isotopen van waterstof in een magnetische configuratie die de *tokamak* wordt genoemd. In deze configuratie kan het plasma echter op verschillende manieren destabiliseren, met verlies van controle tot gevolg. Deze instabiliteiten worden *plasmadisrupties* genoemd. Plasmadisrupties induceren grote krachten op het metalen raamwerk van de tokamak en moeten tegen elke prijs vermeden worden.

Wegens de complexiteit van de verschijnselen die tot plasmadisrupties leiden, wordt voor het vermijden van disrupties veelvuldig gebruik gemaakt van *patroonherkenningstechnieken*. Voorbeelden zijn de *support vector machines* (SVM)[1] of *artificiële neurale netwerken*[2]. De huidige benaderingen slagen er reeds in om een groot percentage van de aankomende disrupties te voorspellen, maar er blijft ruimte voor verbetering. Bovendien is er nood aan een automatisch herkenningsmechanisme dat ook de oorzaak van de disruptie kan herkennen, zodanig dat de nodige preventieve maatregelen kunnen genomen worden[3].

In deze studie werden verschillende alternatieven bestudeerd voor de herkenning van disrupties. De zogenaamde *classifiers* werden getest met data uit experimenten die uitgevoerd zijn in de *Joint European Torus* (JET), de grootste operationele tokamak op dit moment. Tijdsvensters van 30 ms voor dertien indicatieve signalen werden naar frequentieinhoud onderzocht. Dit gebeurde enerzijds a.d.h.v. fourieranalyse, naar analogie met vorige experimenten[1]. Om de data compact te maken en overtollige informatie te verwijderen, werd de standaardafwijking van het fourierspectrum (met exclusie van de statische component) gebruikt als relevant kenmerk. De verdeling van de fouriercoëfficiënten werd aldus vereenvoudigd tot een Gaussiaanse verdeling gecentreerd rond nul. Wij stellen anderzijds een waveletdecompositie voor, gezien het sterk transiënte karakter van disrupties. Wavelet decompositie is een recent ontwikkelde techniek voor frequentieanalyse, die onder andere frequent wordt gebruikt in de analyse van beelden[4], [5]. De waveletdecompositie werd doorgevoerd op drie schalen met de *Daubechies 4-tap wavelets*. De verdeling van de wavelet detailcoëfficiënten wordt beter beschreven door een *veralgemeende Gaussiaan* gecentreerd rond nul[5]. De waarschijnlijkheidsdistributie (PDF) van een veralgemeende Gaussiaan wordt gegeven door[6]

$$f(x \mid \alpha, \beta) = \frac{\beta}{2\alpha\Gamma(1/\beta)} \exp\left[-\left(\frac{\mid x \mid}{\alpha}\right)^{\beta}\right] \qquad (1)$$

Merk op dat voor de vormparameter $\beta = 2$, deze verdeling zich herleidt tot een normale verdeling. Het verband tussen de standaardafwijking σ en de variabele α is $\alpha^2 = 2\sigma^2$. Voor $\beta = 1$ herleidt deze verdeling zich tot een Laplaciaanse verdeling.

De vernieuwing in onze aanpak bestaat erin om de intrinsiek probabilistische natuur van de data in rekening te brengen. De statistiek van fouriercoëfficiënten werd reeds gebruikt in het herkennen van disrupties[1], maar de probabilistische natuur van de data werd verwaarloosd, t.t.z. de standaardafwijking werd behandeld als een Euclidische variabele. Dit is impliciet aanwezig in de exponent van de radiële basisfunctie (RBF) voor de SVM (zie ook uitdrukking (5)). De probabilistische aard van de data kan in rekening gebracht worden door concepten die ontstaan zijn in de tak van *informatiemeetkunde*. Een familie van PDFs wordt gezien als een variëteit, een oppervlak met een niet-Euclidische *metriek*[6]. De *geodetische afstand* tussen twee distributies van dezelfde familie moet dan gemeten worden langsheen dit oppervlak. De afstand tussen twee veralgemeende Gaussianen met dezelfde β wordt gegeven door[6]

$$d_G(\alpha_1; \alpha_2) = \sqrt{\beta}\left|\ln\frac{\alpha_2}{\alpha_1}\right| \qquad (2)$$

Er bestaat geen analytische expressie voor de geodetische afstand tussen twee veralgemeende Gaussianen met verschillende β. In het geval dat β niet constant gehouden werd in de testen, werd daarom gebruik gemaakt van de *Kullback-Leibler divergentie* (KLD). De KLD is een bekende maat uit de waarschijn-

G. Karagounis is student bij de vakgroep Toegepaste Fysica, Universiteit Gent (UGent), Gent, België. E-mail: Giorgos.Karagounis@UGent.be.

lijkheidstheorie en wordt voor twee veralgemeende Gaussianen met verschillende β gegeven door[7]

$$KLD(\alpha_1,\beta_1;\alpha_2,\beta_2) = \ln\left(\frac{\beta_1\alpha_2\Gamma(1/\beta_2)}{\beta_2\alpha_1\Gamma(1/\beta_1)}\right) + \left(\frac{\alpha_1}{\alpha_2}\right)^{\beta_2}\frac{\Gamma((\beta_2+1)/\beta_1)}{\Gamma(1/\beta_1)} - \frac{1}{\beta_1} \quad (3)$$

De KLD afstandsmaat is niet symmetrisch. Een eenvoudige oplossing bestaat erin om de J-divergentie te gebruiken, gedefinieerd door

$$J-div(\alpha_1,\beta_1;\alpha_2,\beta_2) = 1/2\cdot[KLD(\alpha_1,\beta_1;\alpha_2,\beta_2) + KLD(\alpha_2,\beta_2;\alpha_1,\beta_1)] \quad (4)$$

II. Dataverwerking

Informatie in verband met de frequentieinhoud van de signalen werd op de volgende manier geëxtraheerd. De *time traces* van de dertien signalen werden na verwerking (herschaling naar [0,1] en herbemonstering met 1 kHz) gesplitst in tijdsvensters van 30 ms. Dit tijdsinterval zorgt voor een compromis tussen enerzijds voldoende hoge tijdsresolutie en anderzijds de inhoud van voldoende informatie over de tendenzen van het plasma[8]. Dertien verschillende signalen werden gebruikt om de toestand van het plasma te beschrijven. De combinatie van signalen is gebaseerd op vorige studies[8]. De signalen worden gegeven in tabel I. Voor elk tijdsvenster werd een waveletdecomposi-

TABLE I

Lijst van predictor signalen.

	Signaal	Eenheid
(1)	Plasmastroom	A
(2)	Poloïdale beta	
(3)	Tijdsafgeleide van (2)	s^{-1}
(4)	Mode lock amplitude	T
(5)	Veiligheidsfactor bij 95% van kleine straal	
(6)	Tijdsafgeleide van (5)	s^{-1}
(7)	Totaal ingevoerd vermogen	W
(8)	Interne inductantie van het plasma	
(9)	Tijdsafgeleide van (8)	s^{-1}
(10)	Vertikale positie plasma	m
(11)	Plasmadichtheid	m^{-3}
(12)	Tijdsafgeleide van diamagnetische energie	W
(13)	Netto vermogen	

tie uitgevoerd en de statistiek van de waveletcoëfficiënten werd beschreven aan de hand van een veralgemeende Gaussiaan met $\beta = 1$. Bij het uitvoeren van de testen werd het duidelijk dat het gebruik van een variabele β niet mogelijk was. Voor tijdsvensters voldoende verwijderd van de tijd voor disruptie (TvD), was de energieinhoud van het signaal te laag. Als gevolg was de statistiek van de waveletcoëfficiënten sterk gecentreerd rond nul en zorgde dit voor onaanvaardbaar hoge waarden van β. Om de meerwaarde van het gebruik van een variabele β alsnog te testen, werd aan dezelfde statistiek van waveletcoëfficiënten een veralgemeende Gaussiaan met variabele β gefit. Indien de β-waarde echter groter was dan vijf, werd de fit herdaan met vaste $\beta = 5$. Parameters α en β zijn gelijkgesteld aan hun meest aannemelijke schatters[7].

De classificatieresultaten aan de hand van de waveletdecompositie werden vergeleken met deze voor een dataset gerelateerd aan fourieranalyse. De statistiek van de fouriercoëfficiënten werd gemodelleerd door een rond nul gecentreerde Gaussiaan (mits exlusie van de statische component). Dit komt overeen met een veralgemeende Gaussiaan zoals in (1), met $\beta = 2$ en $\alpha = \sqrt{2}\sigma$.

III. Classifiers

De dataset bestaande uit standaardafwijkingen van fourierspectra werd geclassificeerd aan de hand van drie verschillende technieken. De meest eenvoudige was een dichtste naburen (k-NN) classifier[9] met als similariteitscriterium de Euclidische afstand. Elke feature werd als een onafhankelijke variabele beschouwd. De tweede classifier was een k-NN classifier met als similariteitscriterium de geodetische afstand gedefinieerd in (2). De laatste classifier was een SVM classifier met een RBF kern gegeven door

$$K(\boldsymbol{\sigma}_i;\boldsymbol{\sigma}_j) = \exp\left(-\frac{\|\boldsymbol{\sigma}_i-\boldsymbol{\sigma}_j\|^2}{2\sigma^2}\right) \quad (5)$$

$\boldsymbol{\sigma}_i$ en $\boldsymbol{\sigma}_j$ zijn vectoren bestaande uit de aaneenschakeling van de standaardafwijkingen van de fourierspectra voor de dertien signalen, zogenaamde *feature vectors*. Een goede introductie in verband met SVM classifiers kan gevonden worden in [10]. De hier voorgestelde SVM classifier lijkt sterk op deze die reeds gebruikt werd in vroegere classificatietesten in JET[1].

Voor de dataset bestaande uit GGD parameters van waveletstatistieken werden drie classifiers ontwikkeld. Een eerste classifier was een k-NN classifier met als similariteitscriterium de geodetische afstand gedefinieerd in (2). De input voor deze classifier is de set features waar een GGD met vaste $\beta = 1$ aan is gefit. Een tweede k-NN classifier werd ontwikkeld met similariteitscriterium de J-divergentie gedefinieerd in (4). Deze classifier neemt als input de set features waar een GGD met $0 < \beta \leq 5$ aan gefit is. Tenslotte werd een meer geavanceerde classifier ontwikkeld die gebaseerd is op de dataset van features waar een GGD met vaste $\beta = 1$ aan werd gefit. De classifier is gebaseerd op de Mahalanobis afstand tussen een testpunt $\boldsymbol{x}$ en een cluster punten die (multivariaat) Gaussiaans verdeeld is met gemiddelde $\boldsymbol{\mu}$ en covariantiematrix $\boldsymbol{\Sigma}$. Uitleg in verband met de Mahalanobis afstand kan gevonden worden in [11]. De Mahalanobis afstand wordt gegeven door

$$d_M(\boldsymbol{x};\boldsymbol{\mu},\boldsymbol{\Sigma}) = (\boldsymbol{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}) \quad (6)$$

De bovenindex T verwijst naar de transpositie operator. Het idee is dat in de trainingset, reguliere punten enerzijds en disruptieve punten anderzijds in twee aparte clusters gescheiden zijn. Het testpunt wordt in de dichtstbijzijnde cluster geclassificeerd. Met *punten* wordt verwezen naar punten in de ruimte van GGD distributies en meer specifiek naar de productruimte van verschillende, onafhankelijke families van GGD distributies voor de verschillende signalen. De parameters $\boldsymbol{\mu}$ en $\boldsymbol{\Sigma}$ worden

voor elk van de twee clusters apart berekend met formules[11]

$$\begin{aligned} \boldsymbol{\mu} &= \mathbb{E}\left[\boldsymbol{X}\right] \\ &= \frac{1}{n}\sum_{i=1}^{n}\boldsymbol{x}_i \\ \boldsymbol{\Sigma} &= \mathbb{E}\left[(\boldsymbol{X}-\mathbb{E}[\boldsymbol{X}])(\boldsymbol{X}-\mathbb{E}[\boldsymbol{X}])^{\mathbb{T}}\right] \\ &= \frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{x}_i-\boldsymbol{\mu})(\boldsymbol{x}_i-\boldsymbol{\mu})^T \end{aligned} \tag{7}$$

De sommaties lopen over alle punten in de corresponderende cluster.
Een bijkomend probleem was dat de schattingen in (7) enkel gelden in affiene ruimtes, zoals de Euclidische ruimte. De productruimte van onafhankelijke families van univariate Laplacianen is niet affien. Om dit te verhelpen, werden parameters van de clusters in de (Euclidische) raakruimte bepaald. Het raakpunt was het geodetisch zwaartepunt van de cluster. Een gedetailleerde beschrijving van de procedure wordt gegeven in het artikel van Pennec[12].
Er dient opgemerkt te worden dat de k-NN classifiers parametrische classifiers zijn en de resultaten dus afhangen van het aantal dichtste naburen k dat in rekening werd gebracht. De resultaten bleken niet sterk afhankelijk te zijn van dit aantal. k werd daarom gelijk aan een gekozen. Ook de SVM classifier is een parametrische classifier. De resultaten hingen af van de schaalparameter σ van de RBF kern. Uit een uitvoerige test bleek $\sigma = 6$ een goede keuze voor de schaalparameter.

IV. Resultaten

Een classifier moet aan verschillende voorwaarden voldoen. Ten eerste mag de classifier geen enkel regulier tijdsvenster als disruptief herkennen, aangezien de machine dan onnodig stilgelegd wordt. Het aandeel shots waar een classifier toch die fout beging wordt *vals alarm aandeel* (VA) genoemd. Bovendien moet de classifier minstens een disruptief tijdsvenster als disruptief herkennen. Het aandeel van de shots waar de classifier geen enkel disruptief venster als disruptief herkende, wordt *gemist alarm aandeel* (MA) genoemd. Het totaal aandeel van fouten (TF) is de som van VA en MA. Het *succes aandeel* (SA) is het complement van TF (100%-TF). Tenslotte moet de classifier voldoende op voorhand een disruptie detecteren. De gemiddelde tijd voor disruptie waarop de classifier een correcte voorspelling deed wordt afgekort door AVG.
Wegens de beperkte geheugenmogelijkheden werden uit de periode [1180 ms-TvD;1000 ms-TvD[van elk shot zes reguliere tijdsvensters van 30 ms gebruikt om de reguliere features te extraheren. De disruptieve features werden geëxtraheerd uit zes tijdsvensters van de periode [210 ms-TvD;30 ms-TvD[. Het laatste tijdsvenster voor de disruptie werd weggelaten, aangezien het op dat moment niet meer mogelijk is om de disruptie af te wenden. Door de beperkte set van tijdsvensters die in rekening worden gebracht, is het niet mogelijk om de resultaten te vergelijken met real-time resultaten van andere studies. Deze testen dienen als onafhankelijk beschouwd te worden. De voorlopige resultaten zijn interessant omdat ze een vergelijking van het vermogen van verschillende classifiers toelaten in dezelfde omstandigheden.

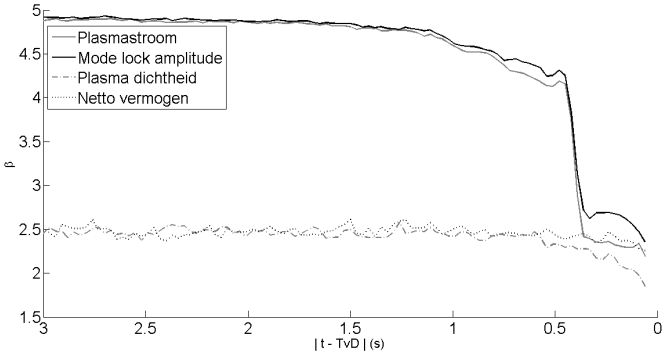

Fig. 1. Tijdsevolutie van β voor veschillende signalen.

A. Evolutie van de vormparameter β

Voor vier verschillende signalen werd de tijdsevolutie van de gemiddelde vormparameter β voor veralgemeende Gaussianen die gefit waren aan de waveletstatistiek op de grootste tijdsschaal weergegeven in figuur 1. De β-waarden voor de plasmadichtheid en het netto vermogen vertonen geen grote verandering bij het naderen van een disruptie. Merk op dat de β-waarden gemiddelde waarden zijn over alle shots. De vrij constante waarde van β wijst erop dat het transiënt gedrag van de signalen slechts in een klein aandeel van de shots te zien is in de statistiek van de waveletcoëfficiënten. Deels is dit te wijten aan de lage resolutie van de signalen.
De vormparameter voor de mode lock grootte en de plasmastroom vertonen daarentegen een sterke overgang ongeveer 360 ms voor de TvD. Het zijn tevens deze twee signalen die in de meeste gevallen het alarm hebben getriggerd in volgende testen. De signalen hebben gemiddeld de hoogste tijdsresolutie in tabel I. De overgang bij 360 ms voor disruptie kan de maximale tijd voor disruptie zijn waar een classifier gebaseerd op wavelet features betrouwbaar een disruptie herkent. In studies gebaseerd op fourier features ligt deze grens ongeveer bij 180 ms voor disruptie.

B. Classificatietest JET campagnes C15-C20

Het vermogen van de classifiers om geen vals alarm te melden en een aankomende disruptie correct te voorspellen werd getest met een dataset gecreëerd uit 442 disruptieve shots van JET campagnes C15-C20. 65% van de shots werden willekeurig gekozen en gebruikt om de trainingset te vormen. De data verkregen uit de rest van de shots is geclassificeerd en het resultaat werd getoetst aan de evolutie van elk shot bepaald door deskundigen. De precieze keuze van shots in training- en testset zorgen voor een variatie op de classificatieresultaten. Om deze variatie te evalueren, werd de test twintig keer herhaald. Voor elke grootheid in tabel II wordt steeds de gemiddelde waarde gegeven met de standaardafwijking op die waarde na meerdere iteraties.

C. Veralgemeningsvermogen naar JET campagnes C21-C27

Een classifier voor disrupties moet tevens goed presteren in situaties waarvoor hij niet getraind is. Om dit te testen, werden de training- en testsets geconstrueerd met shots uit twee verschillende periodes, waarbij de experimentele condities van de periodes verschillend waren. Bij overgang van campagne C20 naar

TABLE II
CLASSIFICATIEVERMOGEN.

	Fourier k-NN Euclidisch	Fourier k-NN geodetisch	Wavelet k-NN geodetisch
MA	0.8 ± 1.0	0.7 ± 0.8	0.3 ± 0.5
VA	65.2 ± 6.3	63.4 ± 5.8	11.3 ± 4.1
TF	66.1 ± 6.1	64.0 ± 5.6	11.6 ± 4.0
SA	33.9 ± 6.1	36.0 ± 5.6	88.4 ± 4.0
AVG	165.4 ± 9.5	173.5 ± 6.5	184.7 ± 3.1
	Wavelet Mahalanobis	Fourier SVM	Wavelet k-NN J-divergentie
MA	0.3 ± 0.5	9.3 ± 2.8	0.2 ± 0.5
VA	8.9 ± 2.4	19.2 ± 3.7	15.2 ± 3.8
TF	9.2 ± 2.3	28.5 ± 4.1	15.5 ± 3.8
SA	90.8 ± 2.3	71.5 ± 4.1	84.5 ± 3.8
AVG	186.9 ± 2.7	137.7 ± 6.4	186.0 ± 2.8

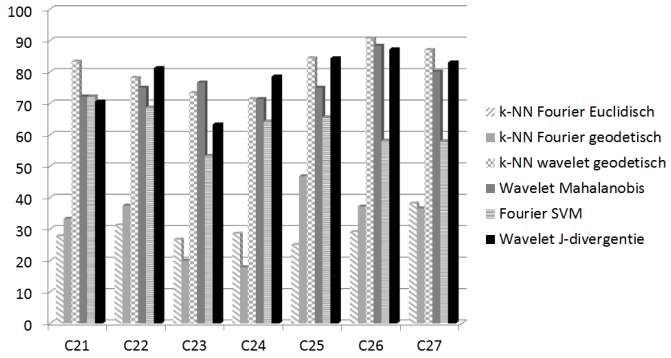

Fig. 2. Succes aandeel voor de veralgemeningstest.

C21, werd de ITER gelijkende *Ion Cyclotron Resonance Heating* (ICRH) *antenne* geïnstalleerd in de JET machine. Daarom werd het veralgemeningsvermogen van de verschillende classifiers getest met een trainingset bestaande uit shots van de periode C15-C20. Er werden zeven testsets geklasseerd, elk bestaande uit data van alle shots van een campagne tussen C21 en C27. Het succes aandeel voor elke campagne wordt weergegeven in figuur 2.

D. Bepalen van de disruptieoorzaak

In toekomstige *real-time* classifiers zal het voorspellen van een aankomende disruptie niet volstaan. Bij elke voorspelling moet ook de oorzaak van de disruptie bepaald worden, zodat de juiste preventieve maatregelen kunnen genomen worden[3]. Gezien de hoge classificatieresultaten van de vorige testen, was het interessant om te kijken of de ontwikkelde classifiers ook de disruptieoorzaak konden bepalen. Een test werd uitgevoerd met data van shots uit de periode C15-C27 (921 shots). Er werden acht disruptieoorzaken opgenomen in de test. De oorzaken, samen met het aantal disrupties te wijten aan elke oorzaak, worden gegeven in tabel III. Deze tabel is gebaseerd op de resultaten van een vorige studie[3]. Merk op dat veel van de disruptieoorzaken (uitschakelen vermogenbron, snelle stroomstijging, problemen in verschillende controlesystemen) te maken hebben met de besturing van de machine en niet met fysische oorzaken van disrupties. Merk bovendien op dat de signalen die voor deze studie gebruikt zijn, niet noodzakelijk aansluiten bij de signalen om

TABLE III
DISRUPTIE-OORZAKEN PERIODE C15-C27.

Disruptieoorzaak	Aantal
Uitschakelen externe vermogenbron (H-L transitie)	67
Te sterke interne transportbarrière	14
Te snelle stroomstijging	13
Probleem in de onzuiverheidscontrole	111
Te lage dichtheid en lage veiligheidsfactor	39
Neo-klassieke "tearing"mode	38
Probleem in de dichtheidscontrole	144
Greenwald dichtheidslimiet	8

elk van deze oorzaken te herkennen. Het was bijvoorbeeld niet mogelijk om een probleem in de onzuiverheidscontrole te herkennen aan de hand van de signalen in tabel I.
De resultaten zijn samengevat in tabel IV. Een classifier kon in deze test een extra fout maken, namelijk het correct voorspellen van een disruptie, maar de foute disruptieoorzaak doorgeven. Het aandeel van shots waarin deze fout gemaakt werd, wordt het *fout alarm aandeel* (FA) genoemd. In deze test werd de SVM classifier terzijde gelaten, aangezien de veralgemening naar classificatie met meerdere groepen niet triviaal is. Bovendien werd het optimaal aantal dichtste naburen van k-NN classifiers voor deze test geherevalueerd. Het bleek voordelig te zijn om met $k = 6$ te werken.

TABLE IV
HERKENNING VAN DISRUPTIEOORZAAK.

	Fourier k-NN Euclidisch	Fourier k-NN geodetisch	Wavelet k-NN geodetisch
MA	6.7 ± 1.4	1.1 ± 0.8	0.4 ± 0.3
VA	32.6 ± 2.9	32.5 ± 2.6	7.2 ± 1.1
FA	35.9 ± 3.1	35.9 ± 2.1	49.8 ± 2.9
TF	75.3 ± 2.1	69.6 ± 3.3	57.4 ± 3.3
SA	24.7 ± 2.1	30.4 ± 3.3	42.6 ± 3.3
AVG	153.5 ± 6.2	162.7 ± 4.7	184.0 ± 3.7
	Wavelet Mahalanobis	Wavelet k-NN J-divergentie	
MA	0.8 ± 0.7	0.5 ± 0.5	
VA	8.5 ± 2.1	10.1 ± 1.9	
FA	48.0 ± 3.0	50.5 ± 2.1	
TF	57.3 ± 2.9	61.1 ± 3.0	
SA	42.7 ± 2.9	38.9 ± 3.0	
AVG	185.1 ± 3.1	181.4 ± 3.7	

V. CONCLUSIES

Er werd een nieuwe methode voorgesteld om disrupties te herkennen. De frequentie-inhoud van een signaal werd in het waveletdomein beschreven. De statistiek van de waveletcoëfficiënten werd voor de compactheid gemodelleerd door een veralgemeende Gaussiaan. Vanwege de hoge β waarden die voorkwamen bij het fitten van een veralgemeende Gaussiaan werden twee alternatieven voorgesteld. Enerzijds werd de β constant gehouden ($\beta = 1$) en anderzijds mocht de β variëren in het interval $0 < \beta \leq 5$. Naargelang de gekozen representatie van de data, werd een gepast similariteitscriterium gekozen. Bij een vaste β was dit similariteitscriterium gebaseerd op de afstand

tussen distributies op een Riemanniaans oppervlak.

De evolutie van β in figuur 1 wijst erop dat classifiers die gebruik maken van wavelet features, aankomende disrupties betrouwbaar kunnen herkennen vanaf 360 ms voor de TvD. Dit wordt bevestigd door de AVG tijden uit tabellen II en IV. Classifiers gebaseerd op de statistiek van waveletcoëfficiënten hebben een AVG tijd die de 180 ms overstijgt. Door het interval dat echter gebruikt is om de disruptieve features te extraheren, is de maximaal mogelijke AVG in die testen 195 ms (het midden van het tijdsvenster waarin de disruptie herkend werd, werd gebruikt om de AVG te bepalen). De AVG voor een SVM classifier is ongeveer 138 ms. Een vergelijkbare AVG tijd voor een SVM classifier wordt ook gemeld in [1].

Het in rekening brengen van de probabilistische structuur via een gepast similariteitscriterium levert een voordeel op in de classificatieresultaten. Dit is reeds zichtbaar voor de classifiers gebaseerd op fourier features. De k-NN classifier die gebaseerd is op de statistiek van fouriercoëfficiënten en die gebruik maakt van de Rao geodetische afstand geeft hogere classificatieresultaten dan dezelfde classifier die gebruik maakt van de Euclidische afstand. Het verschil wordt nog duidelijker voor de classifier gebaseerd op wavelet features. Het gebruik van meer gesofisticeerde classifiers in combinatie met wavelet features, zoals de Mahalanobis classifier, blijkt nog effectiever te zijn in de voorspelling van disrupties.

Tenslotte zijn de classifiers getest op hun vermogen om de oorzaak van disrupties te bepalen. Slechts in een op twee shots wordt de oorzaak van de disruptie correct herkend. Merk op dat het vermogen van de classifiers om disrupties te herkennen niet gedaald is, alleen wordt voor correct voorspelde disrupties, de oorzaak van de disruptie niet altijd herkend.

Uit deze testen blijkt dat de combinatie van de statistiek van waveletcoëfficiënten en het correct in rekening brengen van diens probabilistisch karakter een meerwaarde kan bieden voor detectie van plasmadisrupties. In een volgende stap moet deze combinatie ook in real-time getest worden op haar potentieel. De test voor de bepaling van de disruptieoorzaak is in deze studie eerder rudimentair gebeurd. Het is interessant om in de toekomst een nieuwe combinatie van signalen te gebruiken die meer aanleunt bij de oorzaken van disrupties die de classifier moet kunnen onderscheiden. Eventueel kan dit gebeuren door verschillende classifiers in parallel te laten werken, waarbij iedere classifier getraind is om een welbepaalde klasse van disrupties te detecteren. Het voordeel van deze aanpak zou het beperkte aantal signalen zijn dat iedere classifier zou moeten verwerken, wat een duidelijker onderscheid tussen reguliere en disruptieve tijdsvensters als gevolg zou hebben.

DANKWOORD

Vooreerst dank aan prof. dr. ir. Guido Van Oost die dit onderzoek mogelijk heeft gemaakt. Heel veel dank aan dr. Geert Verdoolaege voor de verschillende keren dat ik bij hem ten rade ben geweest en voor zijn begeleiding in het algemeen. Ook dank aan dr. Jesus Vega en dr. Andrea Murari voor de mogelijkheid om stage te lopen in JET en voor hun begeleiding daar. Tenslotte speciale dank aan dr. Peter de Vries voor het leveren van informatie in verband met de oorzaak van disrupties in JET.

REFERENTIES

[1] G.A. Rattá, J. Vega, A. Murari, G. Vagliasindi, M.F. Johnson, P.C. De Vries, and JET-EFDA contributors, "An advanced disruption predictor for JET tested in a simulated real-time environment," *Nuclear Fusion*, vol. 50, no. 025005, pp. 025005–1–025005–10, 2010.

[2] B. Cannas, A. Fanni, E. Marongiu, and P. Sonato, "Disruption forecasting at JET using neural networks," *Nuclear Fusion*, vol. 44, no. 1, pp. 68, 2004.

[3] P.C. De Vries, M.F. Johnson, B. Alper, P. Buratti, T.C. Hender, H.R. Koslowski, V. Riccardo, and JET-EFDA contributors, "Survey of disruption causes at JET," *Nuclear Fusion*, vol. 51, pp. 053018–1–053018–12, April 2011.

[4] M. Antonini, M. Barlaud, P. Mathieu, and I. Daubechies, "Image coding using wavelet transform," *IEEE Transactions on Image Processing*, vol. 1, no. 2, pp. 205 –220, apr 1992.

[5] G. Verdoolaege and P. Scheunders, "Geodesics on the manifold of multivariate generalized gaussian distributions with an application to multicomponent texture discrimination," *Journal of Mathematical Imaging and Vision*, vol. 10.1007/s11263-011-0448-9, May 2011.

[6] G. Verdoolaege and P. Scheunders, "On the geometry of multivariate generalized gaussian models," *Journal of Mathematical Imaging and Vision*, vol. DOI 10.1007/s10851-011-0297-8, May 2011.

[7] Minh N. Do and Martin Vetterli, "Wavelet-based texture retrieval using generalized gaussian density and kullback-leibler distance," *IEEE Transaction on Image Processing*, vol. 11, no. 2, pp. 146–158, Februari 2002.

[8] G. A. Rattá, J. Vega, A. Murari, M. Johnson, and JET-EFDA contributors, "Feature extraction for improved disruption prediction analysis in JET," *Review of scientific instruments*, vol. 79, pp. 10F328–1–10F328–2, 2008.

[9] S. Kotsiantis, I. Zaharakis, and P. Pintelas, "Machine learning: a review of classification and combining techniques," *Artificial Intelligence Review*, vol. 26, pp. 159–190, 2006, 10.1007/s10462-007-9052-3.

[10] Christopher J.C. Burges, "A tutorial on support vector machines for pattern recognition," *Data Mining and Knowledge Discovery*, vol. 2, pp. 121–167, 1998, 10.1023/A:1009715923555.

[11] Brian H. Russell and Laurence R. Lines, *Mahalanobis clustering, with applications to AVO classification and seismic reservoir parameter estimation*, Consortium for Research in Elastic Wave Exploration Seismology, 2003.

[12] X. Pennec, "Intrinsic statistics on riemannian manifolds: Basic tools for geometric measurements," *Journal of Mathematical Imaging and Vision*, vol. 25, pp. 127–154, 2006, 10.1007/s10851-006-6228-4.

Disruption prediction at the JET tokamak using the geometry of wavelet distributions

Giorgos Karagounis

Supervisors: prof. dr. ir. Guido Van Oost, dr. Geert Verdoolaege

Abstract— **Plasma disruptions are instabilities that occur in experimental fusion plasmas confined in so-called *tokamaks*. Disruptions induce large forces on the framework of a tokamak and need to be avoided at all costs. In this study, automatic detection of disruptions is presented by decomposition of plasma signals in the wavelet domain. The statistics of wavelet coefficients are modelled by a generalised Gaussian distribution. Different classifiers have been developed on this basis. As similarity measure between distributions, the *Rao geodesic distance* was used. The results are compared to classifiers based on the statistics of Fourier coefficients. Our approach shows an improvement as compared to already used methods in this domain. In addition it is possible to reliably predict disruptions up to 360 ms before disruption. Finally, the ability to recognise the disruption cause was tested. Again, better results are obtained using wavelet statistics as predictive features.**

Keywords— **Plasma disruption, fusion, wavelet decomposition, generalised Gaussian distribution, geodesic distance**

I. Introduction

CO ntrolled nuclear fusion can provide an environmental friendly and a virtually inexaustible source of energy in the future. The currently most advanced method to reach this goal is the controlled fusion of nuclei of hydrogen isotopes in a hot plasma confined by a magnetic configuration called the *tokamak*. In such a configuration however, the plasma can destabilise in different ways, leading to a loss of the confinement. The instabilities are called *plasma disruptions*. Plasma disruptions induce large force loads on the metalic vessel and need to be avoided in future machines.

Because of the complexity of phenomena leading to disruptions, plasma disruptions are avoided by means of *pattern recognition techniques*. Examples used in practice are *support vector machines* (SVM)[1] or *artificial neural networks*[2]. Current disruption predictors are already capable of detecting a large number of upcoming disruptions, but there still exist room for improvement. In addition, an automatic disruption predictor is required that is able to recognise the cause of a disruption, in order to be able to undertake mitigating actions[3].

In this study, different classifiers were developed. The so-called *classifiers* were tested with data acquired in disruptive shots at the *Joint European Torus* (JET), the biggest operational tokamak at this moment. The frequency content of time windows of 30 ms of thirteen indicative signals was determined. First this was done using Fourier decomposition, a method which was already used in previous tests[1]. To construct a compact data set in order to avoid redundant information, only the standard deviation of the spectrum (excluding the static component) was held as a relevant feature. The true distribution of Fourier coefficients was thus modelled by a zero-mean Gaussian distribution. Second, we propose to use the wavelet decomposition, because of the highly transient behaviour of disruptions. The wavelet decomposition is a recently developed frequency analysis technique, that is already being used often in image analysis[4], [5]. The wavelet decomposition was performed at three different time scales using the *Daubechies 4-tap wavelet basis*. The distribution of wavelet coefficients is better described by a zero-mean *generalised Gaussian distribution* (GGD)[5]. The probability density function (PDF) of a GGD is given by[6]

$$f(x \mid \alpha, \beta) = \frac{\beta}{2\alpha\Gamma(1/\beta)} \exp\left[-\left(\frac{\mid x \mid}{\alpha}\right)^{\beta}\right] \quad (1)$$

Note that for the shape parameter $\beta = 2$, this distribution simplifies to a Gaussian distribution. The relation between the standard deviation σ en the scale parameter α is $\alpha^2 = 2\sigma^2$. For $\beta = 1$, this distribution simplifies to a Laplace distribution.

The innovation in our approach is to take the intrinsic probabilistic nature of the data into account. The statistics of Fourier coefficients has already been used for disruption prediction[1], but the probabilistic nature of the data was neglected, i.e. the standard deviation was treated as a Euclidean variable. This is implicitly present in the exponent of the radial basis function kernel (RBF) of the SVM (see also (5)). The correct way to measure similarities between distributions has been the subject of *information geometry*. A family of PDFs has to be considered as a Riemannian *manifold*, a surface with a non-Euclidean *metric*[6]. De *geodesic distance* between distributions of the same family then needs to be measured along the surface. We used the so-called *Rao geodesic distance* between distibutions of the family of zero-mean GGD distribution with fixed β. The Rao geodesic distance is given by[6]

$$d_G(\alpha_1; \alpha_2) = \sqrt{\beta} \left| \ln \frac{\alpha_2}{\alpha_1} \right| \quad (2)$$

There exists no analytical expression for the geodesic distance between two GGDs with different β. In tests where β was allowed to vary, the *Kullback-Leibler divergence* (KLD) was used as similarity measure. The KLD is a well-known measure in probability theory and for two GGDs it is given by[7]

$$KLD(\alpha_1, \beta_1; \alpha_2, \beta_2) = \ln\left(\frac{\beta_1\alpha_2\Gamma(1/\beta_2)}{\beta_2\alpha_1\Gamma(1/\beta_1)}\right) + \left(\frac{\alpha_1}{\alpha_2}\right)^{\beta_2} \frac{\Gamma((\beta_2+1)/\beta_1)}{\Gamma(1/\beta_1)} - \frac{1}{\beta_1} \quad (3)$$

De KLD measure is not symmetric. A simple solution is the use of the J-divergence, defined by

$$J - div(\alpha_1, \beta_1; \alpha_2, \beta_2) = 1/2 \cdot [KLD(\alpha_1, \beta_1; \alpha_2, \beta_2) + KLD(\alpha_2, \beta_2; \alpha_1, \beta_1)] \quad (4)$$

G. Karagounis is a student at the department of Applied Physics, University of Ghent (UGent), Ghent, Belgium. E-mail: Giorgos.Karagounis@UGent.be.

II. Data processing

The *time traces* of thirteen signals (after normalisation to [0,1] and resampling with resampling rate 1 kHz) were split into time windows of 30 ms. This time interval provides a good time resolution and includes enough information about the plasma tendencies[8]. The signals which were used to monitor the plasma status are based on a previous study and are given in table I[8]. Each time window and signal was decomposed by wavelet analysis on three different time scales and the statistics of the coefficients of each time scale was described by a GGD with $\beta = 1$. During the tests it became clear that it was not possible to use a variable β. For time windows sufficiently prior to time of disruption (ToD), the energy content of the signal was too low. As a consequence, the wavelet coefficients were highly centered around zero and this resulted in unacceptable high values of β. Another approach to this issue, was to fit a GGD with a variable β within certain limits. First the β was fitted with no limits. If the β value exceeded five, the fit was repeated with a fixed $\beta = 5$. α and β were determined by maximum likelihood estimation[7].

TABLE I

List of predictor signals.

	Signal	Unit
(1)	Plasma current	A
(2)	Poloidal beta	
(3)	Time derivative of (2)	s^{-1}
(4)	Mode lock amplitude	T
(5)	Safety factor at 95% of minor radius	
(6)	Time derivative of (5)	s^{-1}
(7)	Total input power	W
(8)	Internal plasma inductance	
(9)	Time derivative of (8)	s^{-1}
(10)	Vertical position of plasma centroid	m
(11)	Plasma density	m^{-3}
(12)	Time derivative of stored diamagnetic energy	W
(13)	Net power	W

The classification results were compared to those of classifiers based on the statistics of Fourier coefficients. The statistics were modeled by a zero-mean Gaussian (excluding the static component). This distribution corresponds to a GGD with $\beta = 2$ and $\alpha = \sqrt{2}\sigma$.

III. Classifiers

The data set consisting of standard deviations of Fourier statistics was classified in three different ways. The simplest classifier is a k-nearest neighbour (k-NN) classifier[9] with as similarity measure the Euclidean distance. Each feature was considered to be an independent Euclidean variable. The second classifier is a k-NN classifier with as similarity measure the geodesic distance defined in (2). The last classifier is a SVM classifier with an RBF kernel given by

$$K(\boldsymbol{\sigma}_i; \boldsymbol{\sigma}_j) = \exp\left(-\frac{\|\boldsymbol{\sigma}_i - \boldsymbol{\sigma}_j\|^2}{2\sigma^2}\right) \tag{5}$$

$\boldsymbol{\sigma}_i$ and $\boldsymbol{\sigma}_j$ are vectors constructed by the concatenation of the standard deviations of Fourier statistics of different signals and are called *feature vectors*. An introduction about SVM classifiers can be found in [10]. The proposed SVM classifier resembles well the classifier used in previous studies at JET[1].

The data set consisting of GGD parameters of wavelet statistics was also classified by three different classifiers. The first classifier was a k-NN classifier with as similarity measure the geodesic distance defined in (2) (fixed $\beta = 1$). The second classifier was a k-NN classifier with as similarity measure the J-divergence defined in (4) (variable $0 < \beta \leq 5$). The third classifier is a more advanced classifier for the data set with fixed $\beta = 1$. The classifier is based on the Mahalanobis distance between a test object $\boldsymbol{x}$ and a cluster of points which is described by a multivariate Gaussian with mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$. Information about the Mahalanobis distance can be found in[11]. The Mahalanobis distance is given by

$$d_M(\boldsymbol{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) = (\boldsymbol{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{x} - \boldsymbol{\mu}) \tag{6}$$

The upper index T is the transposition operator. The idea is that in a training set, regular points and disruptive points are separated in two clusters. The test point is classified to the closest cluster. With *points* we refer to points in the space of a family of GGD distributions, or better, to the product space of different independent families of GGD distributions. The wavelet statistics of each signal and timescale correspond to one such space of GGD distributions. Estimates of $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ for each regime are found using[11]

$$\begin{aligned} \boldsymbol{\mu} &= \mathbb{E}[\boldsymbol{X}] \\ &= \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{x}_i \\ \boldsymbol{\Sigma} &= \mathbb{E}\left[(\boldsymbol{X} - \mathbb{E}[\boldsymbol{X}])(\boldsymbol{X} - \mathbb{E}[\boldsymbol{X}])^{\mathbb{T}}\right] \\ &= \frac{1}{n}\sum_{i=1}^{n} (\boldsymbol{x}_i - \boldsymbol{\mu})(\boldsymbol{x}_i - \boldsymbol{\mu})^T \end{aligned} \tag{7}$$

The summations run over all points of the cluster.

An additional issue is that the estimates in (7) are only valid in affine spaces, e.g. the Euclidean space. The product space of independent families of univariate Laplacians is not affine. To circumvent this problem, the parameters of the clusters are computed in the (Euclidean) tangent space. The point at which the tangent space was constructed was chosen to be the geodesic centre-of-mass of the cluster. A more detailed description is given in [12].

k-NN classifiers are parametric classifiers and the results will only depend on the number of nearest neighbours k. The results were found to be insensitive to this number. k was therefore chosen equal to one for simplicity. Also the SVM classifier is a parametric classifier. The results depend on the value of the scale parameter σ of the RBF kernel. From an extensive test, $\sigma = 6$ proved to be a good choice.

IV. Results

A classifier needs to fullfil several conditions. First, the classifier should not classify any regular time window as disruptive, as the tokamak operation would then be unnecessarily interrupted. The share of shots in which the classifier made this error

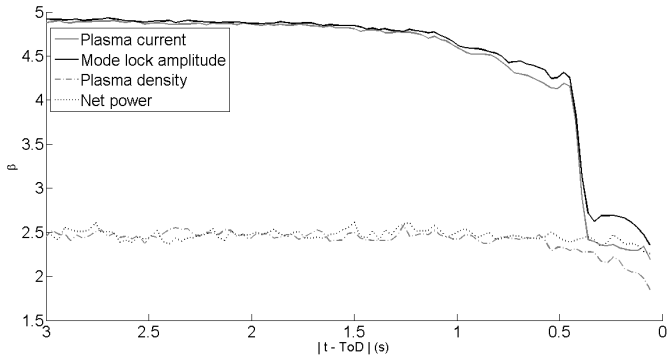

Fig. 1. Time evolution of β for several signals.

is called the *false alarm rate* (FA). Second, the classifier needs to recognise the disruptive behaviour in at least one disruptive time window in order to avoid the disruption. The share of shots for which the classifier did not predict the upcoming disruption is called the *missed alarm rate* (MA). The *total error* (TE) is the sum of FA and MA. The *success rate* (SR) is the complement of the total error (100%-TE). Finally, the classifier should detect the upcoming disruption as early as possible. The mean time before the ToD for which the classifier was able to correctly predict the disruption is called the *average time* (AVG).

Because of the limited available memory, the regular regime was represented in the tests by the features of six time windows of 30 ms long, drawn from the period [1180 ms-ToD;1000 ms-ToD[. The disruptive features were extracted from six time windows of the period [210 ms-ToD;30 ms-ToD[. The last time window before ToD has been left out, as it is not possible to mitigate the disruption in such a limited amount of time. Because of the limited share of regular time windows, it is not possible to compare our results with real-time equivalent results of other studies. The presented tests are to be seen as independent. However, the tests are still interesting because they allow to compare the performance of different classifiers under the same conditions.

A. Evolution of the shape parameter β

The time evolution of the mean shape parameter β for GGDs fitted to the wavelet statistics of four different signals at the highest time scale is shown in figure 1. The value of β does not display a significant change for the plasma density and for the net power when approaching the disruption. The β values in the figure are mean values. The almost constant value of β thus implies that only in a limited amount of shots the transient behaviour of the signals is present in the fit of a GGD to the wavelet statistics. This is partly explained by the low temporal resolution of the signals.

On the other hand, the shape parameters for the mode lock amplitude and the plasma current decrease sharply at about 360 ms before ToD. In the other tests, the alarm was almost always triggered because of the transient behaviour of one of those two signals. Both signals have on average the highest time resolution in table I. The abrupt decrease of the shape parameter at 360 ms before ToD could be indicative for the maximal time before disruption at which a classifier based on wavelet features could reliably detect a disruption. In studies based on Fourier features, this time is reported to be of the order of 180 ms before ToD[1].

B. Classification test JET campaigns C15-C20

The ability of each classifier to avoid a false alarm and to predict an upcoming disruption well in advance was tested with a data set constisting of 442 disruptive shots of JET campaigns C15-C20. 65% of the shots have been selected randomly and the features vectors of these shots were used to construct a training set. The data of the rest of the shots was classified and the labels of the classifier were compared to the ground truth, which was determined by experts. The choice of shots in the training and test sets affect the classification rates. The test was therefore repeated twenty times with different training and test sets. Each quantity in II is given by its mean value $\pm$ the standard deviation after different iterations.

TABLE II
CLASSIFICATION PERFORMANCE.

	Fourier k-NN Euclidean	Fourier k-NN geodesic	Wavelet k-NN geodesic
MA	0.8 ± 1.0	0.7 ± 0.8	0.3 ± 0.5
FA	65.2 ± 6.3	63.4 ± 5.8	11.3 ± 4.1
TE	66.1 ± 6.1	64.0 ± 5.6	11.6 ± 4.0
SR	33.9 ± 6.1	36.0 ± 5.6	88.4 ± 4.0
AVG	165.4 ± 9.5	173.5 ± 6.5	184.7 ± 3.1
	Wavelet Mahalanobis	Fourier SVM	Wavelet k-NN J-divergence
MA	0.3 ± 0.5	9.3 ± 2.8	0.2 ± 0.5
FA	8.9 ± 2.4	19.2 ± 3.7	15.2 ± 3.8
TE	9.2 ± 2.3	28.5 ± 4.1	15.5 ± 3.8
SR	90.8 ± 2.3	71.5 ± 4.1	84.5 ± 3.8
AVG	186.9 ± 2.7	137.7 ± 6.4	186.0 ± 2.8

C. Generalisation capability to JET campaigns C21-C27

A disruption classifier should in addition be able to perform well in instances for which is was not trained. To evaluate the performance of the classifiers in such a situation, the classifiers were made to classify data from shots were something was changed in the experiment as compared to the shots with which the classifier was trained. In the period between campaign C20 and C21, the ITER-like *Ion Cyclotron Resonance Heating* (ICRH) *antenna* was installed at JET.

The classifiers were first trained with disruptive shots from campaigns C15-C20. Then seven data sets, each containing all disruptive shots of one campaign from C21 to C27, were classified. The success rate for each campaign is shown in figure 2.

D. Recognition of disruption cause

In future real-time classifiers, the detection of an upcoming disruption will not be sufficient. The detection should be accompanied by the detection of the disruption *cause*, in order to undertake the appropriate mitigating actions[3]. Because of the high classification rates of previous tests, it becomes interesting to investigate whether the proposed classifiers are able to determine also the disruption cause. The test was performed by data of shots from campaigns C15-C27 (921 shots). Eight disruption causes were considered. The causes are given in table III, together with the number of disruptions within the period C15-C27 due to each cause. The shots which did not disrupt for one

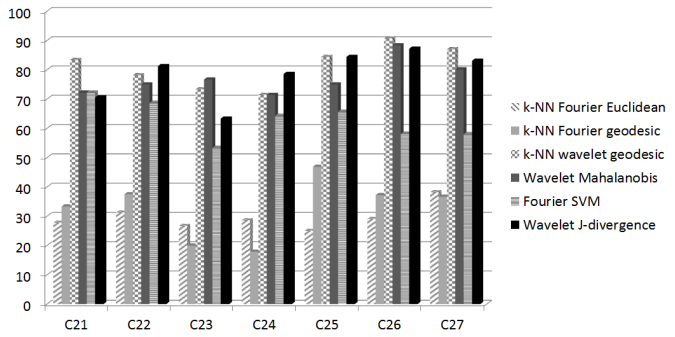

Fig. 2. Success rate for the generalisation test.

of these causes were collected in an additional class. The disruption cause is based on the results presented in [3]. Note that different disruption causes (power shut-down, fast current ramp-up, problems in different control systems) are a consequence of the control systems and do not originate in physical disruption mechanisms. Furthermore, some disruption causes can simply not be detected with the thirteen predictor signals of table I, e.g. there is no signal in the table which can clearly distinguish impurity control problems from other causes.

TABLE III
DISRUPTION CAUSES FOR THE PERIOD C15-C27.

Disruption cause	Number
Auxialiary power shut-down (H-L transition)	67
Too strong internal transport barrier	14
Too fast current ramp-up	13
Impurity control problem	111
Too low density and low safety factor	39
Neo-classical tearing mode	38
Density control problem	144
Greenwald density limit	8

The results are summarised in table IV. A classifier can now make an additional mistake, i.e. to correctly predict a disruption, but give an incorrect disruption cause. The share of shots for which the classifier made this mistake is called the *wrong alarm rate* (WA). The SVM was not used in this test, as the generalisation to a multiple-class problem is not trivial. In addition the optimal number of nearest neighbours was re-investigated. $k = 6$ was found to be the ideal choice.

V. CONCLUSIONS

A new method was presented for disruption prediction. The frequency content of plasma signals was extracted in the wavelet domain. The statistics of wavelet coefficients was modelled by a GGD for compactness. Because of the occurence of high β values, two alternatives were considered. The β was held fixed ($\beta = 1$) or it was allowed to vary within the range $0 < \beta \leq 5$. Depending on this choice, an appropriate similarity measure was used which takes into account the probabilistic nature of the data. When the β was held fixed, this measure was based on the distance between points on a Riemannian surface.
The evolution of β in figure 1 suggests that classifiers that are based on wavelet features are able to reliably detect disruptions from about 360 ms before ToD. The AVG times from tabels II and IV confirm this conclusion. Classifiers based on wavelet features have AVG times that surpass the 180 ms. However, because of the choice of the period from which the disruptive features are extracted, the maximum possible AVG time in the tests is 195 ms, i.e. the middle of the first disruptive time window. Thus most of the disruptions were predicted by wavelet classifiers already in the first disruptive time window. The AVG for the SVM classifier based on Fourier features is about 135 ms. A similar AVG time has already been reported for a SVM classifier[1].

TABLE IV
DISRUPTION CAUSE RECOGNITION.

	Fourier k-NN Euclidean	Fourier k-NN geodesic	Wavelet k-NN geodesic
MA	6.7 ± 1.4	1.1 ± 0.8	0.4 ± 0.3
FA	32.6 ± 2.9	32.5 ± 2.6	7.2 ± 1.1
WA	35.9 ± 3.1	35.9 ± 2.1	49.8 ± 2.9
TE	75.3 ± 2.1	69.6 ± 3.3	57.4 ± 3.3
SR	24.7 ± 2.1	30.4 ± 3.3	42.6 ± 3.3
AVG	153.5 ± 6.2	162.7 ± 4.7	184.0 ± 3.7
	Wavelet Mahalanobis	Wavelet k-NN J-divergence	
MA	0.8 ± 0.7	0.5 ± 0.5	
FA	8.5 ± 2.1	10.1 ± 1.9	
WA	48.0 ± 3.0	50.5 ± 2.1	
TE	57.3 ± 2.9	61.1 ± 3.0	
SR	42.7 ± 2.9	38.9 ± 3.0	
AVG	185.1 ± 3.1	181.4 ± 3.7	

In addition, taking into account the probabilistic nature of the data by using an appropriate similarity measure results in higher classification rates. This is already clear in the different results between the k-NN classifier based on Fourier features using the Euclidean distance as similarity measure and the classifier that uses Fourier features and the Rao geodesic distance as similarity measure. The classifier which uses the Rao geodesic distance and thus takes the probabilistic nature into account, delivers slightly better results. The k-NN classifier based on wavelet features which uses the Rao geodesic distance delivers still better results. Finally, the use of more sophisticated classifier models in combination with the wavelet decomposition, as in the Mahalanobis classifier, delivers the best results.
At last, the classifiers were tested on their ability to determine the cause of a disruption. For only one on two shots the correct cause is identified (WA rates in table IV). This test should be considered as preliminary, as the predictor signals should be chosen according to the disruption causes one wants to investigate. Note, however, that the ability of the predictor to detect disruptions has not lowered (MA and FA in table IV), but that the SR only lowered because of the incorrect detected disruption cause.
The combination of the use of wavelet decomposition and the probabilistic nature of the wavelet statistics proves to be useful in the detection of disruptions. In a next step, this combination should be tested for its potential in a real-time application. The detection of the disruption cause is to treated more in detail. It could be interesting to work with different, parallel classifiers, each trained to detect only one class of disruptions. The advan-

tage would be the limited amount of signals that each classifier should process, resulting in less redundant information and a clearer distinction between regular and disruptive events.

ACKNOWLEDGMENTS

First I would like to thank prof. dr. ir. Guido Van Oost that made this study possible. Many thanks to dr. Geert Verdoolaege for the several times he offered me advice and for his guidance in general. Also thanks to dr. Jesus Vega and dr. Andrea Murari for their guidance during the internship at JET. Finally, special thanks to dr. Peter de Vries for the information concerning the disruption causes at JET.

REFERENCES

[1] G.A. Rattá, J. Vega, A. Murari, G. Vagliasindi, M.F. Johnson, P.C. De Vries, and JET-EFDA contributors, "An advanced disruption predictor for JET tested in a simulated real-time environment," *Nuclear Fusion*, vol. 50, no. 025005, pp. 025005–1–025005–10, 2010.

[2] B. Cannas, A. Fanni, E. Marongiu, and P. Sonato, "Disruption forecasting at JET using neural networks," *Nuclear Fusion*, vol. 44, no. 1, pp. 68, 2004.

[3] P.C. De Vries, M.F. Johnson, B. Alper, P. Buratti, T.C. Hender, H.R. Koslowski, V. Riccardo, and JET-EFDA contributors, "Survey of disruption causes at JET," *Nuclear Fusion*, vol. 51, pp. 053018–1–053018–12, April 2011.

[4] M. Antonini, M. Barlaud, P. Mathieu, and I. Daubechies, "Image coding using wavelet transform," *IEEE Transactions on Image Processing*, vol. 1, no. 2, pp. 205 –220, apr 1992.

[5] G. Verdoolaege and P. Scheunders, "Geodesics on the manifold of multivariate generalized gaussian distributions with an application to multicomponent texture discrimination," *Journal of Mathematical Imaging and Vision*, vol. 10.1007/s11263-011-0448-9, May 2011.

[6] G. Verdoolaege and P. Scheunders, "On the geometry of multivariate generalized gaussian models," *Journal of Mathematical Imaging and Vision*, vol. DOI 10.1007/s10851-011-0297-8, May 2011.

[7] Minh N. Do and Martin Vetterli, "Wavelet-based texture retrieval using generalized gaussian density and kullback-leibler distance," *IEEE Transaction on Image Processing*, vol. 11, no. 2, pp. 146–158, Februari 2002.

[8] G. A. Rattá, J. Vega, A. Murari, M. Johnson, and JET-EFDA contributors, "Feature extraction for improved disruption prediction analysis in JET," *Review of scientific instruments*, vol. 79, pp. 10F328–1–10F328–2, 2008.

[9] S. Kotsiantis, I. Zaharakis, and P. Pintelas, "Machine learning: a review of classification and combining techniques," *Artificial Intelligence Review*, vol. 26, pp. 159–190, 2006, 10.1007/s10462-007-9052-3.

[10] Christopher J.C. Burges, "A tutorial on support vector machines for pattern recognition," *Data Mining and Knowledge Discovery*, vol. 2, pp. 121–167, 1998, 10.1023/A:1009715923555.

[11] Brian H. Russell and Laurence R. Lines, *Mahalanobis clustering, with applications to AVO classification and seismic reservoir parameter estimation*, Consortium for Research in Elastic Wave Exploration Seismology, 2003.

[12] X. Pennec, "Intrinsic statistics on riemannian manifolds: Basic tools for geometric measurements," *Journal of Mathematical Imaging and Vision*, vol. 25, pp. 127–154, 2006, 10.1007/s10851-006-6228-4.

Contents

List of abbreviations

D-T	Deuterium-Tritium
DWT	Discrete Wavelet Transform
ELM	Edge Localised Mode
EU	European Union
FFT	Fast Fourier Transform
GGD	Generalised Gaussian Distribution
i.a.	inter alia (among other things)
ITER	International Thermonuclear Experimental Reactor
ITB	Internal Transport Barrier
JET	Joint European Torus
k-NN	k-Nearest Neigbours
MHD	MagnetoHydroDynamic (equations)
MDS	MultiDimensional Scaling
NDWT	Non-Decimated Wavelet Transform
NTM	Neo-classical Tearing Mode
PDF	Probability Density Function
RBF	Radial Basis Function
SVM	Support Vector Machine
ToD	Time of Disruption
UK	United Kingdom
USA	United States of America
VDE	Vertical Displacement Event

Chapter 1

Introduction

Nuclear fusion can provide a clean and nearly inexhaustible source of energy for future generations. One of the ways to reach this objective is the magnetic confinement of plasma in the tokamak configuration. However, sudden losses of plasma confinement occur in tokamaks, which are called *plasma disruptions*. Disruptions should be avoided at all costs in next-generation devices such as the *International Thermonuclear Experimental Reactor* (ITER), as they can harm the structural integrity of the machine. ITER is schematically represented in figure 1.1.
The physical characterisation of disruptions is an extremely complex task. It is thus impossible to distinguish disruptives events from the available data in an unambiguous way. Therefore, considerable effort has been put into the prediction of upcoming disruptions using machine learning techniques. Machine learning refers to a class of techniques which allow a computer to detect significant patterns in the data. Such techniques can be used to make predictions about new data originating from the same source. Disruption prediction has mainly been addressed by means of *support vector machines* (SVM) and *artificial neural network* techniques [1, 2]. There still exists, however, wide interest to improve the performance of automatic disruption predictors.
In this study an alternative approach to automatic disruption prediction is presented. The study is mainly based on the methodology of previous research of Rattá et al. [1, 3]. In this study, an automated SVM disruption predictor is presented, specifically designed for the *Joint European Torus* (JET) device situated at Culham, UK. The signals were split in small time windows after pre-processing and the frequency content was analysed. This is done by Fourier decomposition. To allow a compact description of the data, only the statistics of the Fourier spectrum was retained. The Fourier spectrum was described by a zero-mean Gaussian distribution, i.e. it was described by its standard deviation. This already allowed to obtain good classification rates.
Our goal is to improve the performance of automated disruption predictors in two aspects. First, the frequency content of plasma signals close to disruption can be better described using a *wavelet decomposition*. Wavelet decomposition is a relatively new frequency analysis method. Wavelets decompose the signal at multiple resolutions and the decomposition at each resolution scale is clearly split in an average part and a detail part. It is the detail part that is retained to describe transient behaviour or, equivalently, the frequency content of a signal. Wavelets are heavily used in image compression [5] and image classification [6], where abrupt changes in color are common. Wavelet decomposition is also expected to describe well the highly transient

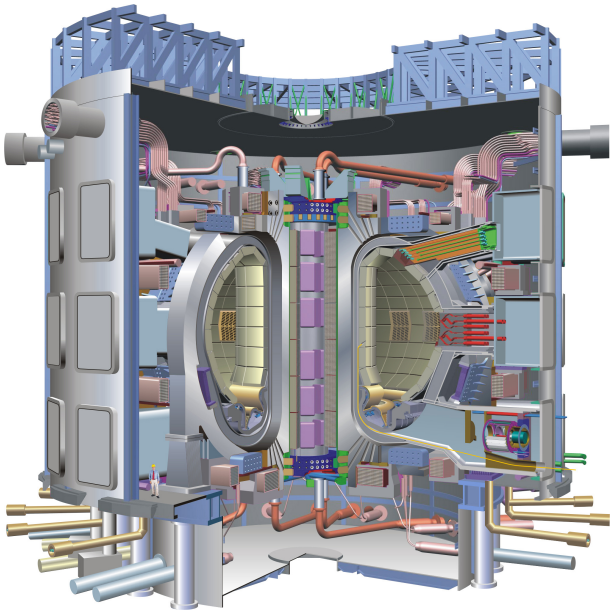

Figure 1.1: Schematic representation of the ITER device [4].

behaviour of plasma signals. The wavelet spectrum will be described by a statistical model, in analogy with the work of Rattá et al. It has been observed earlier that wavelet spectra are heavy-tailed when describing highly transient data. An appropriate distribution which models well this heavy tailed behaviour is the *generalised Gaussian distribution* (GGD) [6].
The developed classifiers are essentially based on two different models. The first model is a simple *k-nearest neighbours* (k-NN) classifier. k-NN searches for the closest neighbours of a new event in an available training set and classifies the event based on the class of its neighbours. A more advanced classifier clusters the training data per regime and describes the spread of the cluster by a multivariate Gaussian distribution. New events are then classified according to the closest cluster. The distance to the cluster is the *Mahalanobis distance*. The Mahalanobis distance is a measure for the probability that the new event is a sample of the multivariate Gaussian distribution of the corresponding cluster.
Both classifiers are based on geometrical concepts. It is therefore interesting to geometrise the probabilistic space of the data. Our second innovation is to introduce some elements from *information geometry*, in which a family of probability distributions is seen as a differentiable Riemannian *manifold*, which is a space equipped with a (non-Euclidean) metric [7]. Concepts as the distance between probability distributions arise naturally in this framework as the distance between points along the manifold. This distance measure is called the *Rao geodesic distance* and will be used to compute the distance between the statistics of wavelet spectra when using a k-NN classifier. Also, the estimation of the parameters of the multivariate Gaussian distribution of a cluster relies implicitly on the properties of affine spaces, which the probabilistic manifold is not. In the case of the Mahalanobis classifier, the *exponential* and *logarithmic* maps were used to project the data from the manifold to a tangent, Euclidean space, which is an affine space.
In this chapter, basics about nuclear fusion and plasma disruptions will be given. The data used in the tests is also presented. In the second chapter, the pattern recognition techniques

necessary for this study will be described in detail. How the different techniques are combined to create the disruption predictor will be explained. The experiments are presented in chapter 3, together with the obtained results and some conclusions. The work is completed in chapter 4 with an overview of the results, conclusions and considerations for future work.

1.1 Nuclear fusion

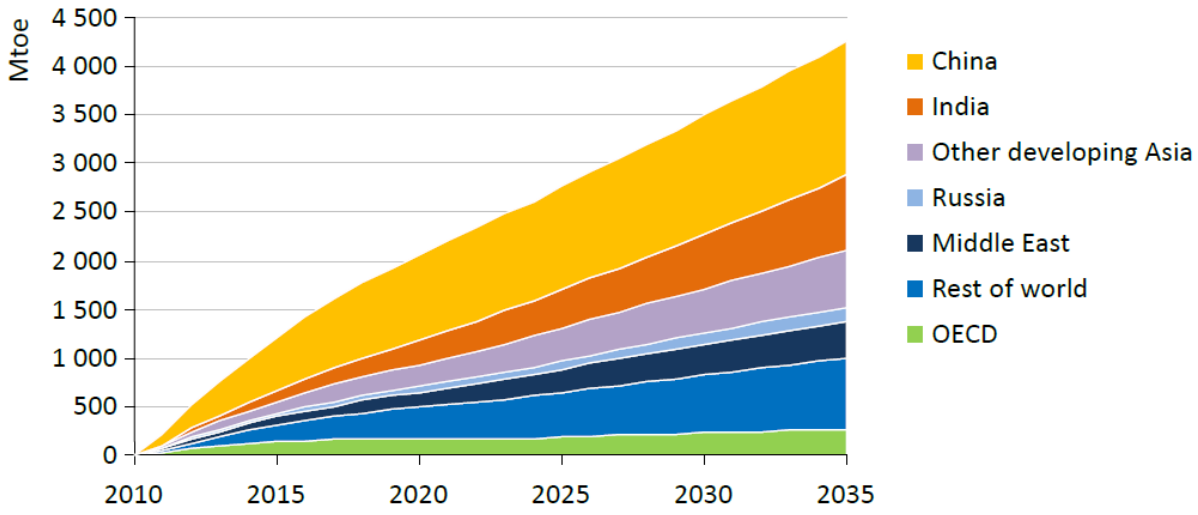

Figure 1.2: Growth in primary energy demand from 2010 to 2035 [8].

Mankind is confronted with a continuously rising world energy demand. Figure 1.2 represents the expected growth of energy demand up until 2035, showing that it will grow with one third of current energy demand. Other estimates expect a doubling of this demand between 2050 and 2100 [9, 10, 11]. The rise in energy demand is accompanied by limited reserves of fossil fuels. Crude oil and natural gas are expected to be available for the next 70 years at current rates of consumption [11]. At the same time, environmental issues are becoming increasingly important. Without further action, by 2017 all CO_2 emissions permitted by the *450 scenario* of the International Energy Agency will be locked-in by existing power plants, factories and buildings [8]. The 450 scenario consists of a series of measures which could stabilise greenhouse gases at 450 ppm CO_2 equivalent [12]. This concentration will increase the world mean temperature with 2°C as compared to pre-industrial ages. It becomes clear that the scientific community is confronted with the challenge to develop new, environmentally friendly technologies to solve the energy problem.

The only long-term alternatives to burning fossil fuels are renewables, fission and fusion. Renewables, although environmentally friendly, have only a limited potential and will only be able to complement other clean energy sources [11]. The available uranium will be exhausted within a few decades at the current rate of consumption, although breeder reactors could extend present reserves to several thousands of years [10, 11]. Fission reactors have also suffered from public criticism by the Chernobyl and Fukushima events. Although safer reactor designs are available [10], adverse effects are expected in the development of nuclear fission energy in the EU and

USA [13, 14]. Another critical issue is the production of long-lived nuclear waste. A breeder reactor could provide a solution to this problem as nuclear waste is recycled to produce nuclear energy [10]. For the moment no commercial breeder reactors exist because of the high cost and safety issues [15]. Fission thus remains a valid energy source for the future, but some problems will remain.

At this point fusion should be introduced as an environmentally friendly, inherently safe and virtually inexhaustible source of energy. The primary fuels of the most likely fusion scenario at present, deuterium and lithium, are not radioactive and do not contribute to environmental pollution. In addition, natural reserves will last for several thousands of years [11]. The direct end product, helium, is no pollutant, is chemically inert and is a useful byproduct for industry. The *Deuterium-Tritium* (D-T) reaction is considered the most feasible fusion reaction for the first power plants. This choice is justified by the highest cross-section in the reaction rates at the lowest temperature [10]. The D-T reaction is thus the easiest way towards fusion energy. Working with tritium has however some disadvantages. First, there are no natural tritium reserves on earth. A simple solution is the breeding of tritium by the reaction of neutrons, originating from the fusion reactions, with lithium, which is installed in a blanket surrounding the reactor. Tritium is radioactive with a half-time of about 12 years [16]. The fact, however, that the tritium cycle will internally be closed makes a fusion reactor extremely attractive for safety purposes. A commercial fusion power plant will only need reserves of a few kilograms of tritium. Moreover, only about 1 kg of tritium could be released in case of an accident, assuming hypothetical, ex-plant events such as an earthquake of hitherto never experienced magnitude [17]. For events of this magnitude, the consequences of the external hazard are expected to be higher than the health effects of irradiation.

Working with tritium delivers neutrons as byproduct. The neutrons will be used to create the necessary tritium and transport a part of the heat, generated by the fusion reactions, to the vessel walls. The heat will be used to heat a coolant flowing through the vessel walls. This coolant will transport the heat outside the reactor where electricity will be generated in the same way as in a conventional power plant [10]. However, the neutrons will unavoidably irradiate and activate materials in the vessel structures. This will give rise, by component replacement and decommissioning, to activated material similar in volume to that of fission reactors [17]. It has to be noted that the radiotoxicity of activated materials falls off significantly within a few decades. It is estimated that 60% of the decommissioned materials will be below international clearance levels within 30 years and 80% of the material will be available after 100 years [18].

A last issue involves the use of beryllium as a material for the first wall of a fusion device. The use of beryllium is justified by its low-Z number, leading to low Bremsstrahlung radiation and better tritium retention properties [19]. Beryllium however is a toxic material and should be treated with care. Experience with beryllium at JET has shown that current safety measures are effective, with no identifiable health effect up until now [20].

Despite the apparent advantages, commercial fusion is likely to be commercially available only within 30 to 50 years [10]. Other sources of energy will be needed until then. It is nonetheless important to pursue the goal of a functional power plant for its enormous potential in energy production, environmentally friendly properties and safety features. Fusion will in addition be an energy source available for each nation as both deuterium and lithium are omni-present. Fu-

sion seems to be at present the only option towards a sustainable energy source in a technological society.

1.2 Plasma disruptions

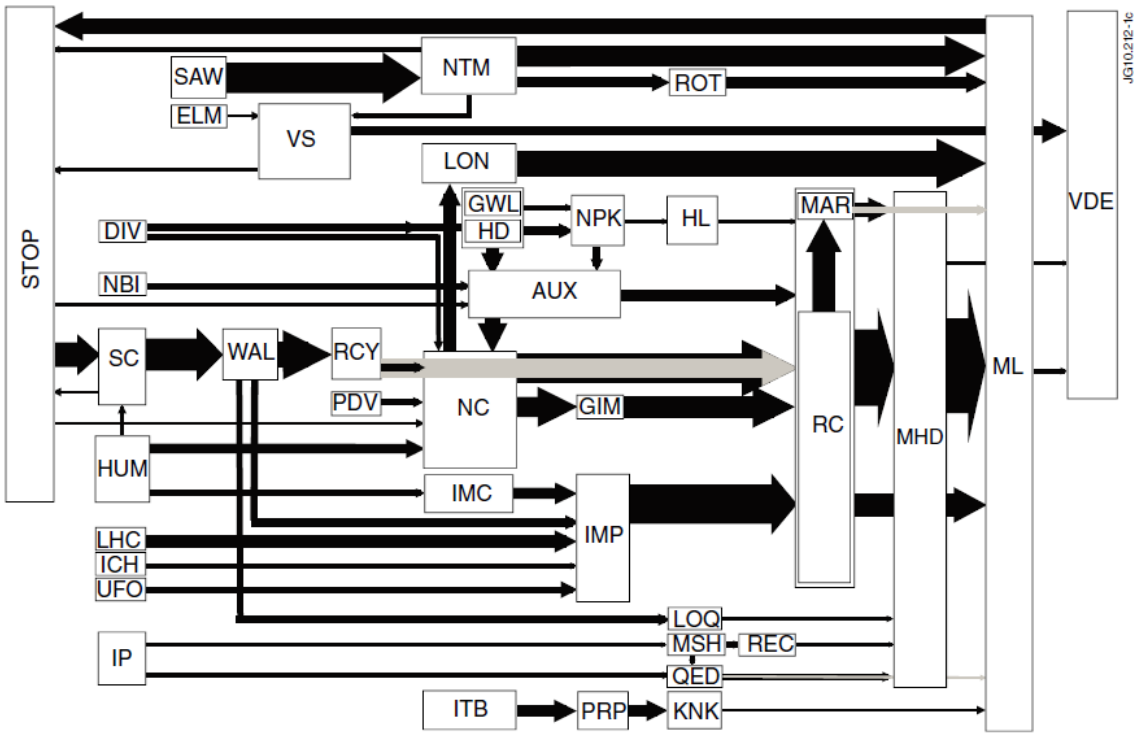

Figure 1.3: Main unintentional disruptions causes at JET for shots between 2000 and 2010. The thickness of the arrow represents the frequency with which the sequence took place [21].

Controlled nuclear fusion aims at the development of a clean and nearly inexhaustible source of energy. At present the most promising and technological advanced design of a fusion device is the tokamak. However, under certain circumstances, instabilities in a tokamak plasma can lead to a collapse of the plasma pressure and current [22]. This is called a *plasma disruption*. In present devices, disruptions induce forces up to 2 MN and heat loads up to 2 MJ/m^2 which lead to serious damage of plasma facing components [21, 23]. In ITER the force loads are expected to increase with two orders of magnitude. Forces of this magnitude can harm the integrity of the machine [23]. Disruption prediction will thus become vital in next-generation devices [24].
A typical disruption develops in four stages: *(i)* an initiating event leads to *(ii)* the development of some precursors of the disruption. Typically the thermal energy of the plasma is lost: *(iii)* the thermal quench. The disruption ends by the loss of plasma confinement and plasma current: *(iv)* the current quench [23]. A disruption predictor should be able to detect precursors in order to initiate mitigating actions [21].
Figure 1.3 represents a schematic overview of the sequence of events that eventually led to unintended disruptions at JET between 2000 and 2010. It is clear that a quite complicated structure arises, with multiple possible paths leading to disruptions. The most important physical ini-

tiating events of disruptions of this scheme will be qualitatively discussed. It should be noted however that a large part of the disruptions at JET were caused by control errors and human mistakes, bringing a random factor into the occurence of disruptions [21].

Table 1.1: Definition of some abbreviations used in figure 1.3.

Abbreviation	Disruption cause
ELM	Edge localised mode
GWL	Greenwald limit
HD	Operation near Greenwald limit
ITB	Too strong internal transport barrier
KNK	Internal kink mode
LOQ	Low q or $Q_{95} \approx 2$
MHD	General rotating $n = 1, 2$ MHD
ML	Mode lock
NTM	Neo-classical tearing mode
RC	Radiative collapse
SAW	Sawtooth crash
SC	Shape control problem
VDE	Vertical displacement event
VS	Vertical stability control problem

Disruptions very often occur in the form of *magnetohydrodynamic* (MHD) instabilities. Important experimental and theoretical advances have been made recently in the understanding and control of this class of instabilities [25]. From a theoretical point of view, the instabilities are understood in terms of small perturbations on MHD equilibrium. Some perturbations are unstable and eventually lead to a disruption. Those perturbations are called ideal MHD instabilities. Hereafter, a summary of the theoretical understanding is given, mainly based on the book of Freidberg [10].

It may be interesting for the reader to repeat briefly how the MHD equations are derived. The MHD equations arise by considering the plasma as a two-fluid model (ions and electrons are treated separately). Using this model, equations describing the different macroscopic properties of the plasma (such as density, temperature and pressure) are derived from conservation of mass, momentum and energy. Those equations are coupled with Maxwell's equations because of the electromagnetic forces acting on the plasma which appear in the conservation of momentum. The two-fluid model is simplified by inspection of the different terms in the model. Only terms relevant on the length and time scale of the macroscopic behaviour of the plasma are conserved. This results in a single-fluid model which is known as MHD. MHD is a system of coupled partial differential equations describing the macroscopic quantities of the plasma as a whole.

The MHD model allows a determination of the magnetic configurations that are able to confine a plasma at fusion conditions. The tokamak is such a configuration. The plasma exhibits a natural tendency to expand in a tokamak, which results in a radial expansion force. This force is counteracted by the generation of toroidal (along the plasma column) and poloidal forces (in a plane perpendicular to the plasma column). By the curvature of the plasma in a toroidal

configuration there also exist toroidal forces, causing the plasma expand to larger major radius. These forces are counteracted by a poloidal (vertical) field. For a tokamak configuration, the equilibrium MHD equations also set an upper limit for β/I^2, where I is the plasma current. β is an important dimensionless parameter in reactor design which gives the ratio of plasma pressure to magnetic pressure. β is a measure of the effectiveness by which the magnetic field confines the plasma [10].

Once MHD equilibrium is reached, perturbations can occur. However, small perturbations of this equilibrium are sometimes unstable. Unstable perturbations are MHD instabilities and can lead to disruptions. Perturbations are accounted for in the MHD model by the introduction of the *perturbed displacement vector* $\boldsymbol{\xi}(\boldsymbol{r})$. The vector represents a small displacement of each fluid element in the plasma as compared to its equilibrium position. Perturbations of the position of fluid elements generate perturbations in the macroscopic quantities of the plasma. For small displacements, the perturbations of these quantities are assumed to be linearly dependent on the displacement vector. Introducing the small perturbations of all quantities in the MHD equations delivers once more a set of coupled differential equations but now for the perturbed quantities:

$$\begin{cases} \rho_1 & = -\nabla \cdot (\rho \boldsymbol{\xi}) \\ p_1 & = -\boldsymbol{\xi} \cdot \nabla p - \gamma p \nabla \cdot \boldsymbol{\xi} \\ \boldsymbol{B_1} & = \nabla \times (\boldsymbol{\xi} \times \boldsymbol{B}) \\ \boldsymbol{J_1} & = (1/\mu_0) \nabla \times [\nabla \times (\boldsymbol{\xi} \times \boldsymbol{B})] \end{cases} \tag{1.1}$$

In the above expression ρ stands for plasma density, p for pressure, $\boldsymbol{B}$ for the magnetic field and $\boldsymbol{J}$ for plasma current. Quantities without a subscript are the quantities at equilibrium. The quantities with a subscript *1* are the perturbations on the equilibrium quantities. γ is the adiabatic coefficient and equals 5/3. In those equations, the time dependency has already been eliminated by means of a normal mode expansion, meaning that all quantities X vary harmonically in time: $X(\boldsymbol{r}, t) = X(\boldsymbol{r}) \exp(-i\omega t)$. The equations illustrate the most important results of this approach. The equations are partial differential equations in space and need to be accompanied by boundary conditions. Depending on the structure, a different set of eigenmodes and eigenfrequenties will be supported by the geometry. Instabilities correspond to exponentially growing eigenmodes. The mathematics in a toroidal configuration are rather complex. In a simple model, the *2-D surface model*, the perturbed displacement vector reduces to only a radial component. In an axisymmetric system this component can be Fourier analysed in the toroidal direction. The solutions hence depend on a toroidal wave number n.

In a circular tokamak, the perturbed displacement vector can be further Fourier decomposed in the poloidal direction. A poloidal wave number m is introduced. MHD instabilities are therefore often labeled by their toroidal and poloidal wave numbers in the following form: m/n. This convention will be of further use.

Numerical studies have been carried out that require stability against all MHD modes [26]. The result is a simple empirical limit on the safety factor, the so-called *Troyon limit*. The safety factor describes the number of times a magnetic field line goes around the long way, for each time it circles the torus once along the short way [27]. See also figure 1.4 for an illustration. A relation for the safety factor is derived in an equivalent straight cylinder representation, giving

[10]

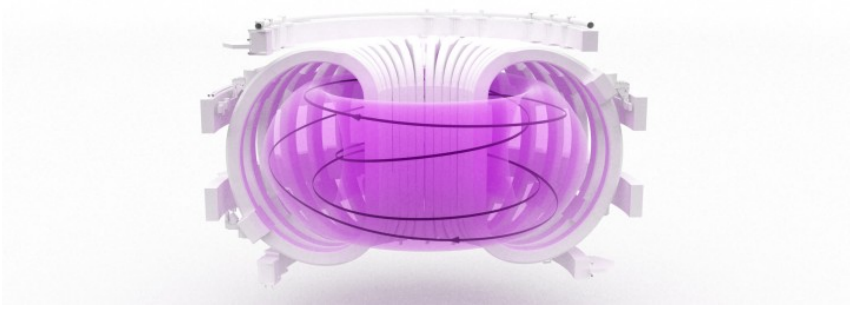

Figure 1.4: A magnetic field line circling around the plasma in a tokamak [28].

$$q(r) = \frac{rB_\phi(r)}{R_0 B_\theta(r)} \tag{1.2}$$

r can vary from the minor radius a to the major radius, R_0. B_ϕ is the magnetic field in the toroidal direction. Similarly, B_θ is the magnetic field in the poloidal direction. A small correction due to toroidicity is neglected in previous expression. Larger values of q are associated with higher ratios of external magnetic field to plasma induced poloidal field and consequently correspond to more stable and safe plasmas [27]. The poloidal magnetic field is proportional to the plasma current.

The Troyon limit is given by

$$q = \frac{2\pi a^2 B_T}{\mu_0 R_0 I} > 2 \tag{1.3}$$

The Troyon limit is directly taken for $r = a$, as the safety factor is smallest at smallest radius. In the relationship, a is the plasma minor radius in metres, R_0 is the plasma major radius in metres, B_T is the toroidal magnetic field in Tesla, and I is the plasma current in mega-amperes. μ_0 is the permeability of the vacuum. In most of the numerical tests the so-called *external ballooning-kink mode* is the most restrictive mode. The word *external* refers to the fact that the mode changes the outer shape in the cross-section of the plasma. The ballooning kink-mode is dominated by a 2/1 mode [10]. The shape of the ballooning-kink mode is shown in figure 1.5.

MHD modes exist in small islands rotating with the plasma. The rotation of the island induces eddy currents in the vessel structure which stabilises the island. However, when external stray fields with a same Fourier component as the island are present, the mode can lock to the wall [23]. This happens when the amplitude of the external stray field is large enough and when the toroidal momentum exerted by the plasma rotation is too weak to prevent it. External stray fields are generated by asymmetric busbars and coil misalignments. The stabilising eddy current ceases to exist for a locked mode. The mode grows even larger and causes edge cooling by convention. This mechanism leads to the thermal quench. The 2/1 mode also dominates locked modes. Most important, stray fields impose a low density limit. Below this density limit, stray fields penetrate easily through the plasma and large convective cells build up around locked 2/1 islands.

Criterion (1.3) for the safety factor can be used to find an upper limit for β. The maximum β allowed in a circular tokamak is about 2.8% for $R_0/a = 2.5$. However, values in a reactor

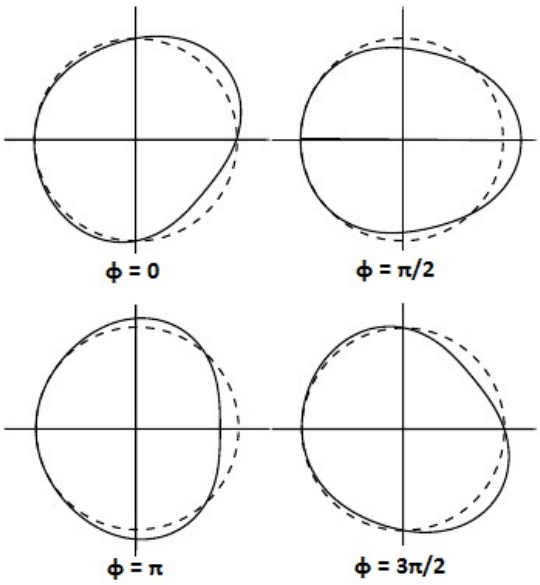

Figure 1.5: A contour plot of the shape of the ballooning-kink mode in the high-β limit for four poloidal cross-sections. The kink shape is clearly situated at the outward radial part of the plasma [10].

are desired to reach 8% [10]. This has inspired fusion research to concentrate on elongated, non-circular plasmas. Both experimental and numerical studies have proved that working with elongated plasmas, β stability limits are increased. Although elongated toroidal configurations stabilise the MHD $n = 1$ mode, the $n = 0$ mode is unstable and gives rise to *vertical displacement events* (VDEs) which eventually lead to disruptions.

One can understand how VDEs develop by considering how the plasma is elongated in an experiment. The plasma at JET has a natural elongation of 1.2 [23]. By external currents with the same polarity as the plasma current, an attractive force is induced at the top and bottom of the plasma, elongating it even further. However, when the plasma centre is perturbed in the vertical direction, the attractive force from the closest coil increases and the force from the furthest coil decreases, as the force between two parallel current-carrying wires is inversely proportional to the distance between them. The direction of this force moves the plasma even further away from its equilibrium position. Vertical displacement events are thus unstable. At the same time, eddy currents that are generated in the external coils and the conducting walls by the movement of the plasma counteract the plasma motion [10, 23]. The plasma can be vertically stable at the resistive timescale of the eddy current. Long time stability still requires a feedback circuit. VDEs should certainly be avoided, as they are responsible for generating halo currents in the conducting internal structures of the machine, leading to large forces [21, 23]. Although VDEs represent a small fraction of the causes of disruptions in JET, they are responsible for a large part of the disruptions leading to highest forces. Often the VDE develops because of vertical stability control problems [21].

Other disruption causes include high density limits (Greenwald limit), edge localised modes, neo-classical tearing modes and strong internal transport barriers. Those causes are understood in terms of transport theory. It should nevertheless be noted that still much work has to be

done before a rigorous approach of transport theory is formulated in fusion plasmas [10]. The reason is that transport is almost always dominated by turbulence driven micro-instabilities. This branch of transport is called anomalous transport and requires kinetic models which are, in general, difficult to solve.

Radiation cooling at the edge is another cause of disruptions. Radiation of low-Z impurities at the edge cools the plasma. As the low-Z impurity radiation increases with decreasing temperature [10, 21], this effect is clearly unstable. While radiation increases and the temperature drops at the plasma edge, the core plasma is contracted. The total current is carried by a smaller plasma area (and thus a smaller minor radius) which causes the safety factor at the plasma edge to decrease (see also (1.3)). This can lead to MHD instabilities. This sequence of events is called a radiative collapse.

A radiative collapse can be caused by too high plasma density at the edge. As the plasma density is increased at a fixed heating power, the temperature at the edge decreases to keep the plasma pressure constant. Radiation cooling will drastically increase at temperatures of about 10 eV. The above reasoning holds for an ohmically heated plasma. In plasma with auxiliary heating the theory becomes much more complicated. In this case, an empirically upper density limit was determined by Greenwald [10]

$$n \leq \frac{I}{\pi a^2} \tag{1.4}$$

Here, n is the density in $10^{20}\,\mathrm{m}^{-3}$, I is the plasma current in mega-amperes and a is the minor plasma radius in metre. In JET, exceeding the Greenwald density limit has not been found to be a primal cause of disruptions in the past ten years [21].

Another disruption mechanism active at the edge of the plasma is called *edge localised modes* (ELMs). The physical mechanism driving ELMs is still not understood and subject of current investigation. There is speculation that ELMs are MHD instabilities [10]. During ELM-free operation, the plasma edge density increases and the plasma is contaminated by impurities. This is an unstable situation as was explained in the case of radiation cooling. However, ELMs often appear before the Greenwald density limit. ELMs are radially outward bursts of energy which relieve the high pressure at the plasma edge. Those bursts of energy carry also impurities out of the plasma. The removal of impurities can be regarded as an advantage and a moderate presence of ELMs is considered beneficial [10]. ELMs are divided into three classes according to the amplitude of the energy burst. Type III ELMs correspond to low-level bursts at low edge pressure. Type II is an intermediate mode of operation and type I ELMs produce large amplitude, narrow time energy bursts. From a reactor's design point of view, type II ELMs could be beneficial, as the pressure remains at acceptable levels, increasing the energy confinement time. In addition, a type II ELM will carry impurities out of the plasma as was mentioned before. From a disruption point of view, type I ELMs should be avoided as they are often precursors for *neo-classical tearing modes* (NTMs) [21].

NTMs are an important cause of disruptions and are at this moment subject of ongoing investigation [29]. NTMs start by the growth of a magnetic island. Such magnetic islands are created by MHD instability events, a sawtooth crash in the core, or ELMs. Magnetic islands short-circuit the magnetic flux surfaces [30]. For very small magnetic islands, diffusion across magnetic surfaces is dominant over diffusion along the magnetic field lines. In this case magnetic

islands are stable and disappear after a certain period of time. For islands with a width of about one centimetre however, the diffusion along the magnetic field lines becomes dominant, leading to a loss of the pressure gradient across the magnetic island. This results in the loss of the bootstrap current in the magnetic island. The bootstrap current is a plasma current in the toroidal direction. The origin of bootstrap current is explained in neoclassical transport. The bootstrap current depends on the pressure gradient. A high bootstrap current is of great importance in a fusion reactor, as this considerably reduces the cost of steady-state operation [10]. The loss of bootstrap current from within the magnetic island seems to result in an unstable situation in which the loss of bootstrap current results in the growth of the island, loss of the pressure gradient within the bigger magnetic island, loss of the bootstrap current in the bigger island and so forth. The resulting magnetic islands can be as big as several tens of centimetres. NTMs are found to be the major cause of disruptions in JET [21].

Too strong *internal transport barriers* (ITBs) can also lead to disruptions [21]. An internal transport barrier is a region within the plasma core where the ion thermal conductivity is substantially decreased as a result of anomalous transport [31]. This results in high temperature gradients, a high temperature at the plasma core and subsequently a corresponding high energy confinement time [10]. Internal transport barriers could be an important feature in advanced tokamak scenarios to produce large fractions of bootstrap current. Therefore, different mechanisms have been devised to create ITBs, such as off-axis current drive or fueling the plasma with high-density solid D pellets. Nonetheless, a number of problems have still to be solved when generating ITBs [31]. Too strong ITBs represent a large fraction (42%) of the causes of highly energetic disruptions (plasma energies $> 4\,\mathrm{MJ}$). Disruptions at these energy levels are certainly harmful for the ITER-like wall installed at JET [21].

1.3 JET disruption data

The previous section points out that for most initiating events, the physical understanding is not complete. Furthermore, the necessary data is not always available to diagnose the disruption cause. It is thus not possible to unambiguously detect a disruption simply based on the values of the plasma parameters. Pattern recognition techniques are used at present in order to predict disruptive behaviour using data of previous disruptions. In this study, JET data was used for training and assessment of the performance of the developed disruption predictor.

A data set consisting of 2309 disruptions which occurred during the last decade at JET was used to assess the performance of the classifiers. Based on previous work, each discharge is represented by the time trace of thirteen predictor signals [3]. A list of all predictor signals is given in table 1.2. For almost all of the signals, the average sampling time is also given. An example of the time trace of five predictor signals is shown in figure 1.6. The shot in the figure disrupted because of a problem in the density control.

Some issues need to be addressed concerning the used data set. Several predictor signals are missing in the database for earlier shotnumbers. Therefore only a subset of the original data was used in this work. The subset contains shots starting from campaign C15, which started at JET in 2006 after the installation of the ITER-like magnetic configurations [28]. The subset consists of 921 shots in which a disruption occurred. This subset was further split in two smaller subsets.

Table 1.2: JET predictor signals.

	Signal name	Unit	Sampling time (ms)
(1)	Plasma current	A	0.2
(2)	Poloidal beta		10
(3)	Poloidal beta time derivative	s^{-1}	10
(4)	Mode lock amplitude	T	0.2
(5)	Safety factor at 95% of minor radius		15
(6)	Safety factor at 95% of minor radius time derivative	s^{-1}	15
(7)	Total input power	W	10
(8)	Plasma internal inductance		10
(9)	Plasma internal inductance time derivative	s^{-1}	10
(10)	Plasma vertical centroid position	m	10
(11)	Plasma density	m^{-3}	1
(12)	Stored diamagnetic energy time derivative	W	10
(13)	Net power (total input power minus total radiated power)	W	

The subsets are presented in table 1.3. In the discussion of tests and results, the abbreviations given in the table will be used to address the different subsets.

Table 1.3: Different subsets of shots used in this work.

Abbreviation	Campaigns	First shotnumber	Last shotnumber	Number of shots
SA	C15-C27	65863	79831	921
SB	C15-C20	65863	73128	442
SC	C21-C27	73177	79831	479

Although the experiments deliver good results at a resampling rate of 1 kHz, it should be noted that the original sampling rate of some signals is lower than 1 kHz. The benefit of resampling is discussed in 2.1. The high resampling rate possibly results in unreliable wavelet features. It will become clear from the tests that a high sampling rate is essential. The low-sampled signals were, however, still used in the tests in accordance with previous studies. This issue will have to be addressed in future work.
Finally, it should be noted that in a limited number of shots, the actual *time of disruption* (ToD) determined by experts appears to be incorrect. Those shots could not be used in the classification tests. An example of the appearance of a false ToD is shown in figure 1.7.

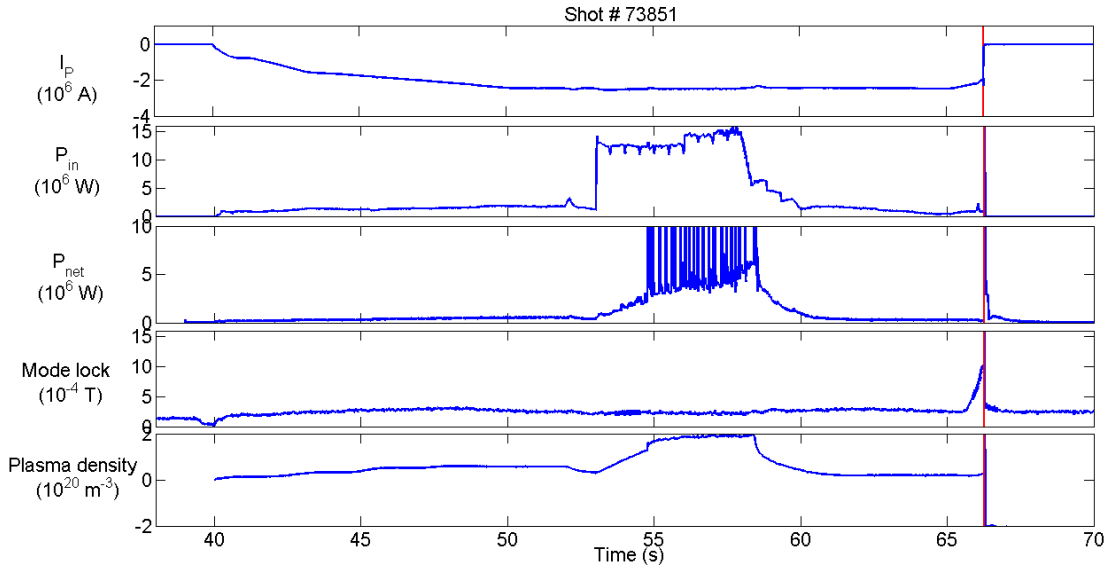

Figure 1.6: Temporal evolution of five predictor signals. The disruption time is indicated by a full red vertical line.

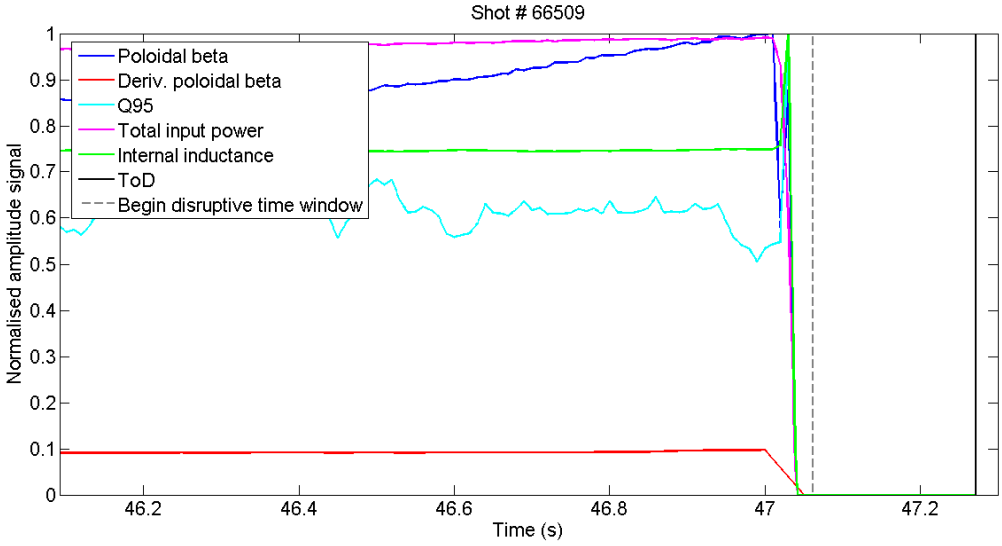

Figure 1.7: In shot 66509, the actual ToD appears to be incorrect. All signals fall off to minimal value more than 200 ms before the actual ToD.

Chapter 2

Data analysis techniques

Different data analysis techniques are necessary in order to create an automatic disruption predictor. These techniques will be presented in this chapter. Multidimensional scaling is discussed briefly as a method to identify the clustering structure in the processed data by visual inspection. The remainder of this chapter is devoted to the different steps in the construction of a disruption predictor. The consecutive steps will appear in the same order as they are used in the classification process.

The first step in classification is to process the original data and extract only information relevant for the classifier. Here, the data to be processed is the JET data which was presented in 1.3. This step is called *feature extraction* [32]. The end result consists of a concatenation of *features*, basic properties describing an event in a compact way. The result of the concatenation is called a *feature vector*. All feature vectors are collected into a single data set which is used by the classifier.

Processing the data has two intentions. First, the presence of higher frequency components in the time signals has been noted in previous work as the discharge approaches a disruption [3]. Therefore, the time trace will be split into small time windows. The frequency content of each window is considered to be a more appropriate starting point than the time evolution for classification purposes. Second, it is known that working with high dimensional feature vectors limits the performance of classifiers because of redundant information in the data. Therefore, the statistics of the frequency content are modeled by an appropriate probabilistic model. In general, the corresponding probability distribution will depend only on a limited set of parameters, allowing a compact description of the relevant information.

The data set is given as input to the classifier, from which it derives information that can be used to classify new events. The data set will be split in two parts, the *training set* and the *test set*. The training set is given to the classifier in the training phase. Once the classifier has been trained, the test set is classified to assess the performance of the classifier. This study focuses on a two-class problem, the classification into either a regular or a disruptive time window. Only one test will concern the classification of disruptions according to disruption cause, which is a multiple-class problem. Different classifier models were developed to solve the two-class problem. Each classifier will be introduced separately.

Particular attention will be given to the geometry and clustering structure of the statistics of wavelet coefficients. It will become clear from the experimental results that the use of an appro-

priate distance measure between probability distributions increases considerably the classification rates. The probability distributions are interpreted as points on a differentiable, Riemannian manifold. This allows to measure distances between distributions on the curved surface. The discussion will begin with the Euclidean distance. A similarity measure from probability theory will, in addition, be revised, the Kullback-Leibler divergence. Lastly, the Rao geodesic distance will be introduced as a measure between distributions on the probabilistic manifold.
The chapter ends with a short introduction of the exponential and logarithmic maps. Once the curved nature of the space of probability distributions is accepted, it becomes clear that traditional estimates of the mean and covariance do not hold for a data set of probability distributions. One of the classifiers, the Mahalanobis classifier, relies heavily on both concepts. The exponential and logarithmic maps will prove useful to project the data from the Riemannian manifold to a tangent, Euclidean space, in which the classifier is able to operate.

2.1 Feature extraction

The frequency content of the data is considered to be more useful in the classification of disruptions. The data contains high frequency components in a certain time interval before the development of a disruption. This time interval is typically a few tens to hundreds of milliseconds long [1]. The signals were split into time windows of 30 ms in order to extract information about the frequency content. This time interval was chosen based on previous work, as a compromise between time resolution and the assurance that each time window contains sufficient information about the plasma tendencies [3].
It has to be noted that, as a feature vector is constructed by the combination of different signals, some measures need to be taken to assure that the classifier treats each signal with equal weight. First, the signals were not equally sampled and therefore each signal was first resampled with a sampling rate of 1 kHz to a common time base. Second, the signal values differ over several orders of magnitude. See also figure 1.6, in which e.g. the mode lock amplitude is of the order of 10^{-4} T, while the density n is of the order of $10^{20}\,\mathrm{m}^{-3}$. Therefore, the time traces were normalised to values between 0 and 1. The resampled and normalised signals were then split in windows of 30 ms. The resampled and normalised time windows were subjected in the next step to a Fourier or wavelet transform.

2.1.1 Fourier decomposition

To analyse the frequency content of a signal, the signal can be decomposed in its Fourier components. A short summary of the discrete Fourier transform (DFT) is given here. Consider a resampled, normalised time window for one plasma signal. This signal consists of values of the normalised plasma signal $f_k = f(t_k)$ at discrete time instants t_k, where $t_k = t_0 + k\Delta$. k is a discrete number between 0 and $N = 29$ (for time windows of 30 ms), t_0 is the instant at which the time window begins and Δ is the sampling rate (here 1 ms). This allows us to rewrite the signal as [33]

$$f_k = \frac{1}{N} \sum_{n=0}^{N-1} F_n \exp\left(\frac{2\pi i k n}{N}\right) \tag{2.1}$$

The coefficients F_n in (2.1) are the Fourier coefficients and can be calculated from

$$F_n = \sum_{k=0}^{N-1} f_k \exp\left(-\frac{2\pi i k n}{N}\right) \tag{2.2}$$

The idea behind the DFT is that the periodic extension of a finite signal in time is projected onto a basis of periodic functions over this time interval. The periodic functions have a well defined frequency. The coefficients given in (2.2) provide then information about the frequency content of the signal.

The Fourier coefficients were computed in the tests by the built-in Matlab function for the Fast Fourier Transform (FFT), `fft(.)`. In order to remove redundant information, a zero-mean Gaussian was fitted to the spectrum of the signal, excluding the static component. The only relevant parameter for the zero-mean Gaussian is the standard deviation. The standard deviation was computed by the Matlab command `std(.)`.

An example is the time trace of the total input power of shot 74530, shown in figure 2.1. The part of the signal which is indicated by the black line has been decomposed by DFT, and the resulting spectrum, together with the fit of a Gaussian distribution, are shown. Note the large amount of Fourier coefficients with a relatively small value.

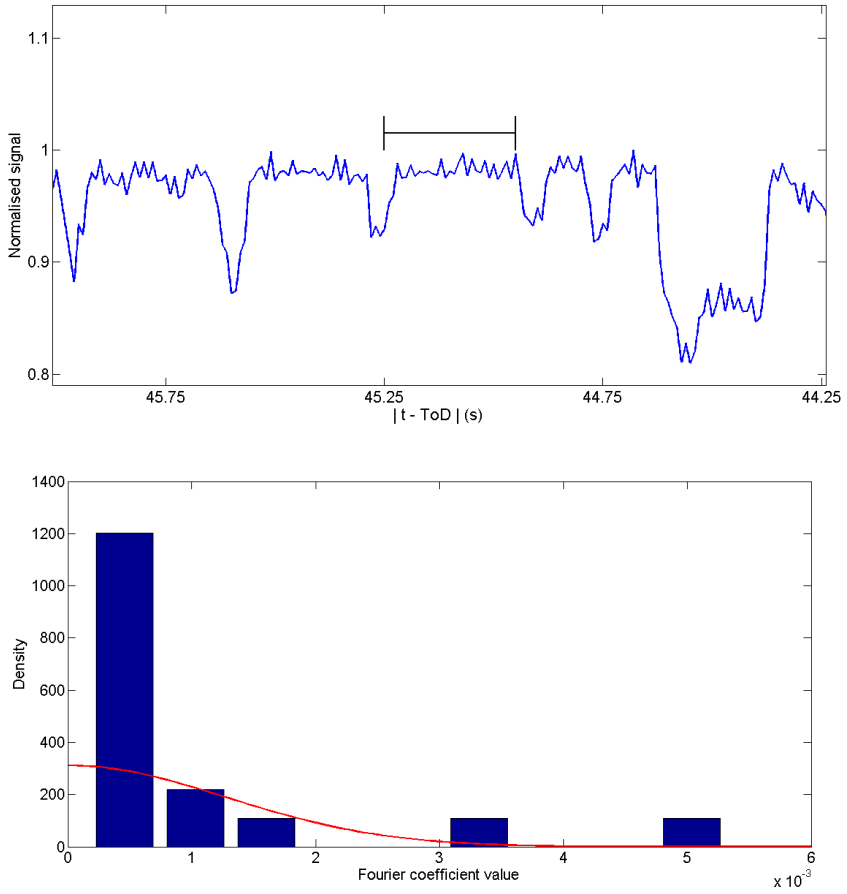

Figure 2.1: Processed total input power, accompanied by the statistics of the Fourier coefficients for a time window of 30 ms.

The resulting feature vector for one time interval of 30 ms consists of thirteen features, resulting

from the concatenation of the standard deviations for the spectra of the thirteen signals discussed in 1.3.

2.1.2 Wavelet decomposition

Disruptions are highly transient phenomena. It is known that highly transient behaviour, as is the case for disruptions, is better described by wavelet analysis. For example, wavelets are heavily used in image compression [7, 6, 5, 34], where edges with an abrupt change in colour or grayscale often occur.

The wavelet transform consists, as is the case with the Fourier transform, of projecting the original time signal onto a new set of basis functions, the wavelets [5]. However, the wavelet transform allows to decompose the original signal on different time scales or resolutions. In a multiresolution wavelet analysis there are actually two families of basis functions which are called the *mother wavelet* (or detail function) and the *father wavelet* (or scaling function). In the case of a continuous time signal $f(t)$, the decomposition can be written as follows [35]

$$f(t) = \sum_k a_{J,k}\Phi_{J,k}(t) + \sum_{j\leq J}\sum_k d_{j,k}\Psi_{j,k}(t) \tag{2.3}$$

In this form, Φ represents the father wavelet and Ψ represents the mother wavelet. $a_{J,k}$ are the approximation coefficients and $d_{j,k}$ are the detail coefficients. The functions $\Phi_{J,k}(t)$ and $\Psi_{j,k}(t)$ are dilated and translated versions of respectively the father and mother wavelets

$$\begin{aligned} \Phi_{j,k}(t) &= \frac{1}{\sqrt{2^j}}\Phi\left(\frac{t-2^j k}{2^j}\right) \\ \Psi_{j,k}(t) &= \frac{1}{\sqrt{2^j}}\Psi\left(\frac{t-2^j k}{2^j}\right) \end{aligned} \tag{2.4}$$

Note that in this expression a positive j corresponds to a time dilated replicate of the original wavelet. Examples of wavelet families, consisting of a father and a mother wavelet, are shown in figure 2.2. The wavelet family on the left is called the *Haar* family. The Haar family is the simplest example of wavelets. The wavelets on the right are the *Daubechies 4-tap wavelets*. This wavelet family was used in the tests.

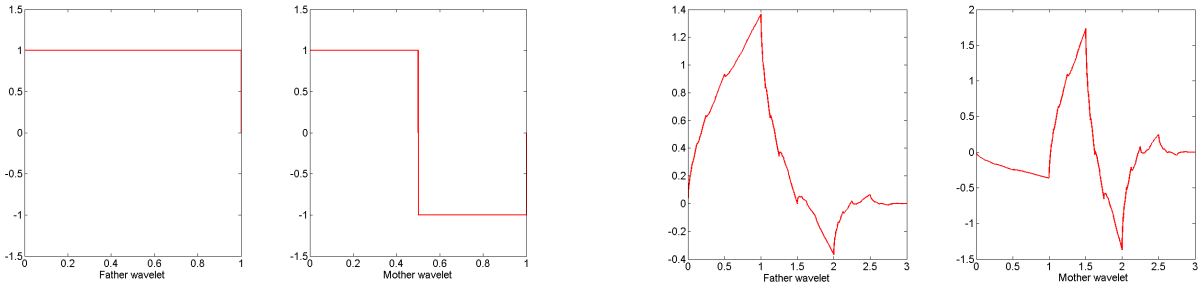

Figure 2.2: Two wavelet families.

How information is extracted about the frequency content is explained in terms of an example. Figure 2.3 shows a signal $f(t)$ with its wavelet decomposition on different decomposition levels.

The first decomposition level is a coarse approximation at the highest time scale using only the father wavelets (upper left). It is clear that father wavelets take into account the average behaviour of f. At a higher wavelet decomposition level (upper right), a detail wavelet subband is added. Some detail about the transient behaviour of the signal is already taken into account. At still higher decomposition levels, more detail subbands are added, making the wavelet decomposition more and more accurate. Mother wavelets on different time scales restore more and more details which were lost in the averaging process by the father wavelets.

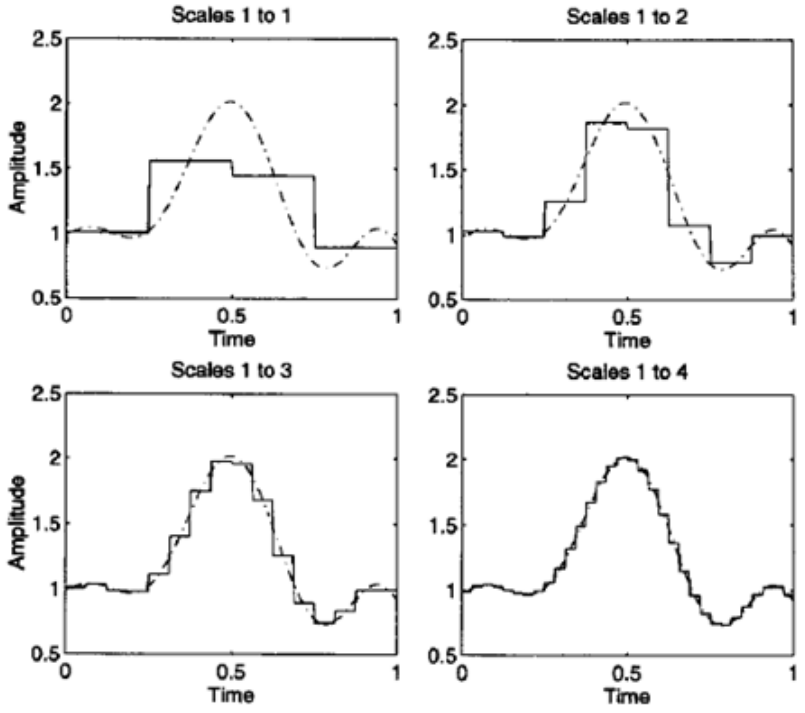

Figure 2.3: Wavelet decomposition of a time signal at four different decomposition levels. The original signal is shown by the broken line. The wavelet reconstruction is given by the solid line [36].

Clearly, the transient behaviour of f is described by the detail functions. Their weighting coefficients, the detail coefficients in (2.3), carry thus information about the transient behaviour of f. This is the desired information to classify JET data in regular and disruptive time windows. As the JET signals are discrete signals, a discrete version of the wavelet transform was used. The wavelet decomposition was performed by the built-in Matlab function for the non-decimated wavelet transform, `ndwt(.)`. The non-decimated wavelet transform does not subsample the signals. In this way, the histogram at each decomposition level will contain sufficient samples (approximately 30).
The wavelet decomposition of one time window of 30 ms delivers detail coefficients of different subbands (assuming higher decomposition levels) which are independent. For each subband, the statistics are now modeled by a generalised Gaussian distribution.

2.1.3 Generalised Gaussian distributions

When plotting a histogram of the values of detail coefficients (i.e. looking at its statistics), two important things should become clear. First, the histogram is highly peaked at zero. The peak at zero can be understood by the function of the detail functions in a wavelet decomposition. Detail functions give higher order corrections to the coarse approximation of the scaling functions. As detail coefficients only describe corrections to this first approximation, many of them have a very low value and lie symmetrically around zero. This immediately indicates that the relevant information is to be found in the high valued detail coefficients, as high detail coefficients describe a large overlap between the original signal and the corresponding detail function. Those detail coefficients will appear with a low frequency. A good approximation of the histogram tails is thus required. A flexible model able to simulate the heavy tailed behaviour of wavelet histograms is the *generalised Gaussian distribution* (GGD) [6, 34]. Wavelet histograms can often be approximated to be centered at zero. The univariate, zero-mean GGD *probability density function* (PDF) is given by

$$f(x \mid \alpha, \beta) = \frac{\beta}{2\alpha\Gamma(1/\beta)} \exp\left[-\left(\frac{\mid x \mid}{\alpha}\right)^{\beta}\right] \tag{2.5}$$

Γ stands for the Gamma function. This distribution still depends on two parameters. $\alpha > 0$ is called the scale parameter. $\beta > 0$ controls the fall-off rate at the vicinity of the mode and is called the shape parameter. A GGD with heavy tails has a low β value. Note that for $\beta = 2$ the GGD reduces to the Gaussian distribution (with standard deviation σ related to α: $\alpha^2 = 2\sigma^2$) and $\beta = 1$ to the Laplace distribution.

An example is shown in figure 2.4. A part of the time trace of the processed plasma current of shotnumber 74530 is shown. The plasma has reached the disruptive stage. The time window indicated by the black line was decomposed by a wavelet transform on three different time scales. The wavelet statistics of the highest time scale are shown in the figure, together with a fit of a Gaussian distribution (green) and a Laplace distribution (red). The Laplace distribution, having a lower β value, describes better the wavelet statistics in the disruptive, transient state of the plasma.

Estimation of the parameters of the GGD were computed by the maximum-likelihood method described in [34]. Consider a set of wavelet detail coefficients $\mathbf{x} = (x_1, \ldots, x_N)$. The method involves the maximisation of the likelihood function $\mathrm{L}(\mathbf{x}|\ \alpha, \beta)$ given by

$$L(\boldsymbol{x} \mid \alpha, \beta) = \log \prod_{i=1}^{N} p(x_i \mid \alpha, \beta) \tag{2.6}$$

p is the probability distribution. In the case of a GGD, p has the same form as f in (2.5). The likelihood function $\mathrm{L}(\mathbf{x}|\ \alpha, \beta)$ becomes extremal when its partial derivatives to α and β vanish, which results in two equations

$$\begin{cases} \frac{\partial L}{\partial \alpha} & = -\frac{L}{\alpha} + \sum_{i=1}^{N} \frac{\beta |x_i|^{\beta} \alpha^{-\beta}}{\alpha} = 0 \\ \frac{\partial L}{\partial \beta} & = \frac{L}{\beta} + \frac{L\Psi(1/\beta)}{\beta^2} - \sum_{i=1}^{N} \left(\frac{|x_i|}{\alpha}\right)^{\beta} \log\left(\frac{|x_i|}{\alpha}\right) = 0 \end{cases} \tag{2.7}$$

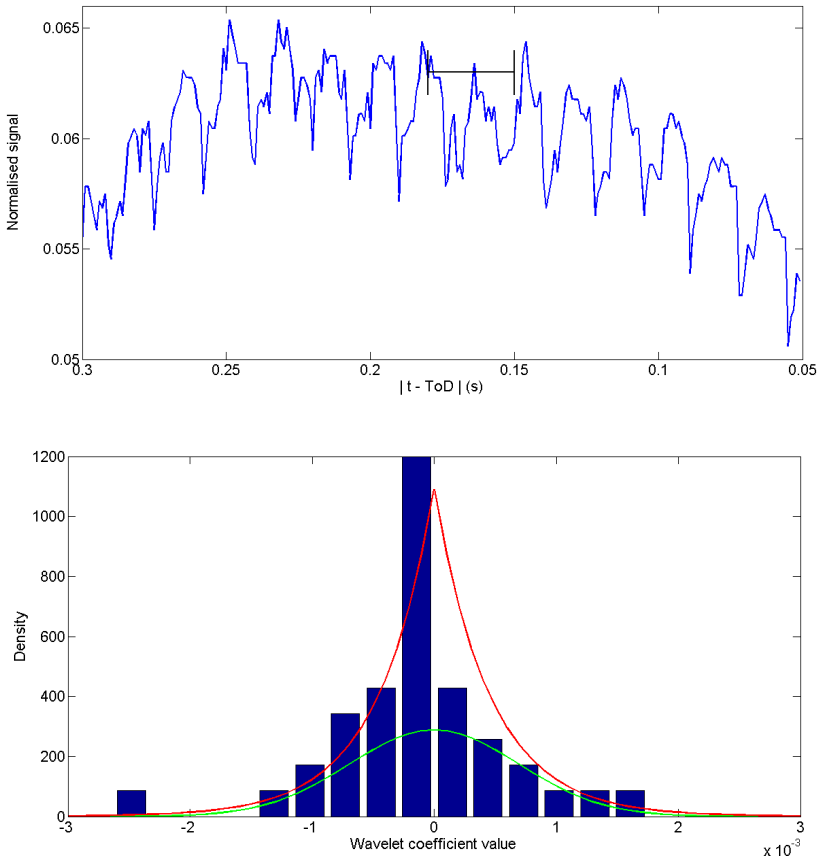

Figure 2.4: Processed plasma current, accompanied by the statistics of the wavelet coefficients for a time window of 30 ms.

Ψ represents the digamma function, defined as $\Psi(x) = \Gamma'(x)/\Gamma(x)$, where $\Gamma(x)$ is the Gamma function. The first equation can be solved to deliver an estimation for α

$$\hat{\alpha} = \left(\frac{\beta}{L} \sum_{i=1}^{N} \mid x_i \mid^{\beta} \right)^{1/\beta} \tag{2.8}$$

Back-substitution of (2.8) in the second equation of (2.7) delivers a transcendental equation in β

$$1 + \frac{\Psi(1/\hat{\beta})}{\hat{\beta}} - \frac{\sum_{i=1}^{N} \mid x_i \mid^{\hat{\beta}} \log \mid x_i \mid}{\sum_{i=1}^{N} \mid x_i \mid^{\hat{\beta}}} + \frac{1}{\hat{\beta}} \log \left(\frac{\hat{\beta}}{L} \sum_{i=1}^{N} \mid x_i \mid^{\hat{\beta}} \right) = 0 \tag{2.9}$$

This highly non-linear equation is solved using the Newton-Raphson iterative procedure with an appropriate initial guess [34].

During tests this method delivered unrealistic high β values for time windows of the regular regime. This can be understood by noting that regular time windows contain mainly low-frequency components. This results in a sparse wavelet representation and thus an unreliable estimation of the distribution parameters. An example is shown in figure 2.5. The time evolution of the time derivative of the poloidal beta of shot 74391 is shown. The time window shown by the black line was decomposed on three different time scales by a wavelet transform. The wavelet statistics of the highest time scale are also shown in the figure. The bars of the histogram

were shortened by a factor of hundred to fit the figure. The value of the wavelet coefficients is thus highly centered at zero, implying a low frequency content in the signal. The fit delivered $\beta = 65.1$ and clearly does not represent well the statistics of the wavelet coefficients. It is also possible to note the low sampling sampling rate of the original signal in the time evolution. The dots show the values of the resampled signal with a rate of 1 kHz. It is, however, clear, that several points lie on a straight line in between two consecutive points of the original signal. The occurence of high β values in the GGD fit was closely related to the time resolution of the signals.

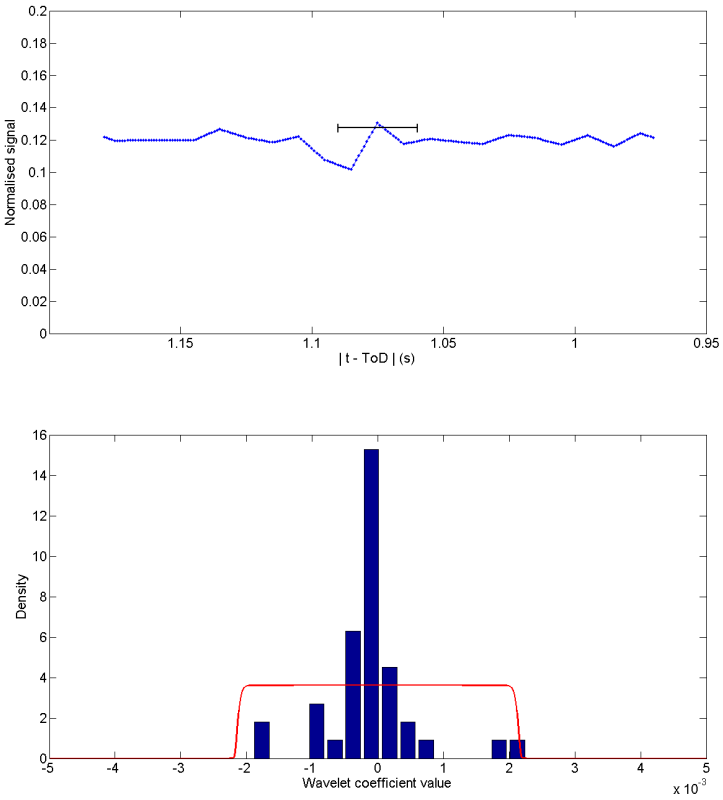

Figure 2.5: Processed time derivative of the poloidal beta, accompanied by the statistics of the wavelet coefficients for a time window of 30 ms.

For this reason, either a fixed $\beta = 1$ was used or the β was allowed to vary within the interval $0 < \beta \leq 5$. To fit β within certain limits, a first fit was performed with no constraints on its value. If, however, $\beta > 5$, the fit was repeated with a fixed $\beta = 5$.
For one time interval, a feature vector will consist of $2 \times 13 \times N_S$ features, with N_S the number of wavelet subbands. There are thirteen signals for each time window, described in 1.3. The time window for each signal is decomposed into N_S sets of detail coefficients by the wavelet transformation. For each of those $13 \times N_S$ sets of detail coefficients, two parameters of the GGD are computed. This explains the number of features in a feature vector.
The resampling, normalisation and splitting of the time traces into time windows was implemented in Matlab, together with the fit of the probabilistic model for the Fourier and wavelet features. The code can be found in appendix D.

2.2 Dimensionality reduction

Dimensionality reduction usually refers to techniques which are used to extract the most essential information from a high-dimensional data set [37]. *Multidimensional scaling* (MDS) is an example of a dimensionality reduction technique. The goal is to avoid the curse of dimensionality in statistical data analysis and also allow a better interpretation of the results [38]. The curse of dimensionality refers to the fact that in high-dimensional spaces, data becomes unavoidably sparse. This effect is problematic for the statistical significance of data analysis.
In this section MDS is introduced solely as a visualisation tool. As mentioned before, the feature vectors acquired by wavelet decomposition lie in a $2 \times 13 \times N_S$ dimensional space, with N_S the decomposition level. In most tests a decomposition level of three was chosen (see also 3.2.2), leading to a 78-dimensional space. In addition, it will be shown in 2.4.3 that this space is non-Euclidean because of the inherent probabilistic nature of the data. To understand the classification results in chapter 3, some intuitive insights will be given by looking at a three-dimensional embedding of the data.

2.2.1 Multidimensional scaling

MDS relies on the notion of dissimilarities between observations. In the case of a data set consisting of parameters of GGDs, the dissimilarity will be a distance measure on the probabilistic manifold. The main goal of MDS is to find a configuration of points in a low-dimensional, Euclidean space, so that the pairwise distances match as closely as possible the dissimilarities. In this way, the data is said to be embedded into a lower-dimensional Euclidean space and can be visualised.
The dissimilarities between pairs of observations (or feature vectors) (i, j) can be collected into a dissimilarity matrix. Note that the dissimilarity matrix should *(i)* have diagonal elements equal to zero, *(ii)* be symmetric, *(iii)* have strictly positive non-diagonal elements (assuming no identical observations). The three restrictions arise from some basic rules that distance measures need to fulfill.
In MDS, the differences between the dissimilarities and the distances of the points in the lower-dimensional space are minimised [39]. Consider a data set consisting of n observations in the m-dimensional feature space. The data is embedded in a p-dimensional space, with $p < m$. The dissimilarity between observations i and j is written as δ_{ij}. The distance between the corresponding points in the lower-dimensional embedding is written as $d_{ij}(\boldsymbol{X})$. $\boldsymbol{X}$ represents a $n \times p$ matrix. Each row in $\boldsymbol{X}$ corresponds to a point, while each column corresponds to a coordinate value for all points. $d_{ij}(\boldsymbol{X})$ is given by

$$d_{ij}(\boldsymbol{X}) = \left(\sum_{k=1}^{p} (\boldsymbol{X}_{ik} - \boldsymbol{X}_{jk})^2 \right)^{1/2} \tag{2.10}$$

which is exactly the Euclidean distance between points X_i and X_j. The first criterion ever used in MDS to minimise the difference between δ_{ij} and $d_{ij}(\boldsymbol{X})$ was the Kruskal raw stress $\sigma^2(\boldsymbol{X})$ given by [39]

$$\sigma^2(\boldsymbol{X}) = \sum_{i=1}^{n} \sum_{j<i} (\delta_{ij} - d_{ij}(\boldsymbol{X}))^2 \tag{2.11}$$

Often a weight factor w_{ij} is added to the sum which is user-defined. Here, however, $w_{ij} = 1$ for all i and j. In this study a slightly different stress criterion was preferred which is called the normalised raw stress $\sigma_n^2(\boldsymbol{X})$

$$\sigma_n^2(\boldsymbol{X}) = \frac{\sum_{i=1}^{n} \sum_{j<i} (\delta_{ij} - d_{ij}(\boldsymbol{X}))^2}{\sum_{i=1}^{n} \sum_{j<i} \delta_{ij}^2} \tag{2.12}$$

The advantage of this stress is that it is insensitive to the scale and the number of dissimilarities [39].

The minimisation of (2.12) is a complex problem and there exists no closed-form solution [39]. Iterative schemes are used to minimise (2.12). For the tests use was made of the Matlab built-in function `mdscale(.)`.

The quality of an MDS embedding can be evaluated by a Shepard's plot [39]. The Shepard's plot is a two-dimensional scatter plot. The x-axis represents the original dissimilarities while the y-axis represents the Euclidean distances for the corresponding embedding. When a dissimilarity matrix is available for the original data (which is not always the case), the vertical distance between principal bisector and a point of the scatter plot represents the error on the lower-dimensional embedding for this pair of objects. A Shepard's plot, made during a test with a data set of wavelet coefficients, is shown in figure 2.6. The exact details of the used set will be discussed in 3.3. It is apparent from the figure that only a limited number of pairwise distances are not well-represented in the MDS embedding. The scatter plot is also reasonably symmetrical along the principal bisector. No systematic deviations are evident. The lower-dimensional embedding can be considered a reasonable representation of the original data set.

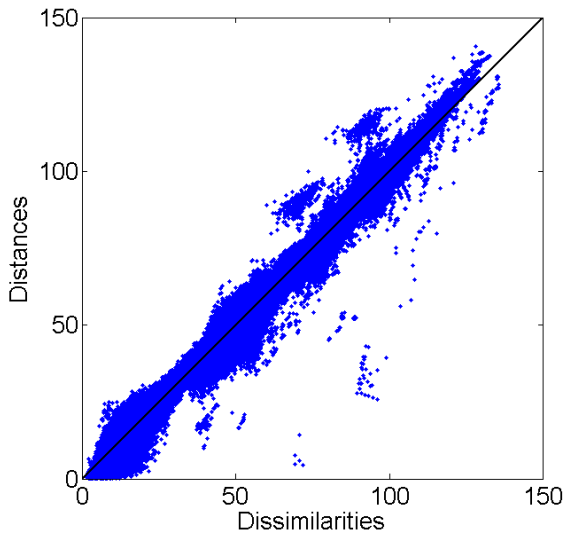

Figure 2.6: A Shepard's plot for MDS performed on a data set of wavelet coefficients. The black solid line represents the principal bisector.

Up until now, the dimension of the Euclidean space has not been mentioned. An approach

to choose the appropriate dimension for the embedding space, is to perform MDS for several dimensions and plot the stress against the dimension. Typically, the stress decreases at higher dimensions. The embedding dimension is chosen at the point where an elbow appears in the plot [40]. It would not be interesting to embed the data at a still higher-dimensional space, as this would not significantly decrease the stress anymore. An example is shown in figure 2.7 for the same wavelet data set as for figure 2.6. From the figure, the conclusion could be drawn easily that the elbow appears for a two-dimensional embedding. However, the one-dimensional embedding should be treated with care [39]. It is best to only consider embeddings in Euclidean spaces with dimension higher than one. If the one-dimensional embedding is neglected in the figure, the three-dimensional embedding appears to be a better choice. Figure 2.6 was made using the three-dimensional embedding of the data set.
The elbow criterion is a rather weak criterion. When MDS is used for exploration of the data, the main criterion is the interpretability of the embedding. This has been the driving criterion in this example. Nevertheless, the elbow criterion is mentioned to further justify the choice of a three-dimensional embedding, which allows to acquire insights by visual inspection.

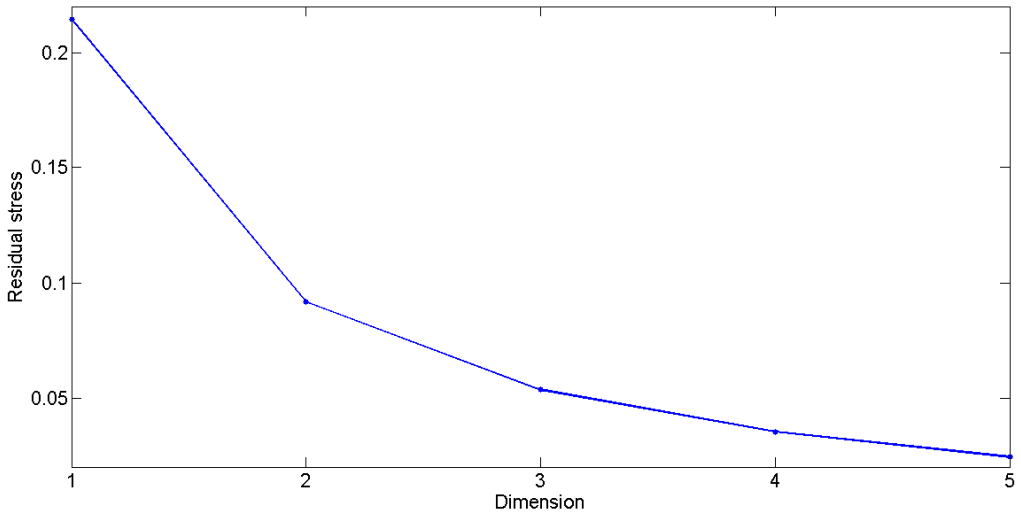

Figure 2.7: Plot of the normalised raw stress for embeddings in two-dimensional to six-dimensional Euclidean spaces.

2.2.2 Hubert's statistic

The results of MDS show a clear cluster structure where observations of regular and disruptive time windows are separated (see also 3.3). To support the visual evidence of cluster structure, statistical tests exist to assess the cluster validity. More specifically, the normalised Hubert's Gamma statistic was used to validate the cluster structure in the MDS embedding.
In the following, a definition is required of the mean and the standard deviation of a matrix. The mean $\mu_{\boldsymbol{A}}$ and standard deviation $\sigma_{\boldsymbol{A}}$ of a matrix $\boldsymbol{A}$ are defined as [41]

$$\mu_{\boldsymbol{A}} = \frac{1}{N(N-1)/2} \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \boldsymbol{A}(i,j) \tag{2.13}$$

$$\sigma_{\boldsymbol{A}}^2 = \frac{1}{N(N-1)/2} \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \left(\boldsymbol{A}(i,j)^2 - \mu_{\boldsymbol{A}}\right)^2 \tag{2.14}$$

The Hubert's Gamma statistic is a measure for the correlation between two matrices. Consider matrices $\boldsymbol{P}$ and $\boldsymbol{Q}$, both of dimension $N \times N$. The normalised Hubert's Gamma statistic for matrices $\boldsymbol{P}$ and $\boldsymbol{Q}$ is then given by [41]

$$\hat{\Gamma} = \frac{(1/M)\sum_{i=1}^{N-1}\sum_{j=i+1}^{N}(\boldsymbol{P}(i,j) - \mu_{\boldsymbol{P}})(\boldsymbol{Q}(i,j) - \mu_{\boldsymbol{Q}})}{\sigma_{\boldsymbol{P}}\sigma_{\boldsymbol{Q}}} \tag{2.15}$$

M in the last expression is a shorthand notation for the number of non-redundant elements in a similarity matrix: $M = N(N-1)/2$.

The goal is to measure the degree in which the *proximity matrix* of $\boldsymbol{X}$ matches the partition $\wp$ which is imposed on $\boldsymbol{X}$ by visual inspection [41]. The term proximity matrix describes the dissimilarity matrix for the observations in the low-dimensional embedding, i.e. the dissimilarity matrix of $\boldsymbol{X}$. The partition $\wp$ can also be regarded as a mapping g of $\boldsymbol{X}$ to the set $\{1, \ldots, m\}$, with m representing the number of partitions[1]. In the next step, a matrix $\boldsymbol{Y}$ is constructed in the following way

$$\boldsymbol{Y}(i,j) = \begin{cases} 1, & \text{if} \quad \mathrm{g(X_i)} \neq \mathrm{g(X_j)} \\ 0, & \text{otherwise} \end{cases} \tag{2.16}$$

It is clear that, when the Gamma statistic of (2.15) is computed for the proximity matrix of $\boldsymbol{X}$ and $\boldsymbol{Y}$ and the partition matches the cluster structure in $\boldsymbol{X}$, pairs of points from different clusters will have a larger impact on the value of $\hat{\Gamma}$ than pairs which belong to the same cluster. Pairs of points from different clusters will lie further away from each other and the corresponding element in the proximity matrix will have a high value in this case. In the case that $\wp$ represents the underlying structure of $\boldsymbol{X}$, the Gamma statistic will thus have a high value.

A statistical test based on the *random label hypothesis* is proposed [41]. The null hypothesis is that there exists no underlying structure in $\boldsymbol{X}$ and the labels are thus assigned randomly. The statistical test is right-tailed. Under the random label hypothesis there is no correlation between $\boldsymbol{X}$ and $\boldsymbol{Y}$ and $\hat{\Gamma}$ is low. On the other hand, when the partition described by the mapping g corresponds to the underlying structure in $\boldsymbol{X}$, $\hat{\Gamma}$ is high and the null hypothesis should be rejected. The decision rule for the statistical test with a significance level α is

$$\begin{cases} \hat{\Gamma} \leq \hat{\Gamma}(1-\alpha) \Rightarrow \text{accept} \quad \mathrm{H_0} \\ \hat{\Gamma} > \hat{\Gamma}(1-\alpha) \Rightarrow \text{reject} \quad \mathrm{H_0} \end{cases} \tag{2.17}$$

In the decision rule, $\hat{\Gamma}(1-\alpha)$ represents the $\hat{\Gamma}$ at which the cumulative distribution function under the null hypothesis reaches $1-\alpha$.

[1] Practically, the partition $\wp$ was determined by visual inspection. The minimum and maximum values of each coordinate in each cluster were determined manually. In a next step, a label was assigned to each point by comparison of its coordinate values and the limits of each cluster.

There exists, however, no closed form expression for the PDF of $\hat{\Gamma}$ under the H_0 hypothesis. The PDF is estimated by Monte Carlo simulations [41]. Monte Carlo simulations require the construction of r mappings g_i, $i = 1, \ldots, r$ under the random label hypothesis. Each point in $\boldsymbol{X}$ is assigned a random label from the set $\{1, \ldots, m\}$. With this mapping, a matrix $\boldsymbol{Y}_i$ is constructed in a similar way as in (2.16). $\hat{\Gamma}_i$ can be computed between the proximity matrix of $\boldsymbol{X}$ and $\boldsymbol{Y}_i$ from (2.15). The decision rule can now be adjusted for the estimated cumulative distribution function under the random label hypothesis to

$$\begin{cases} \#\{i; \hat{\Gamma} > \hat{\Gamma_i}\} \leq ((1-\alpha)r) \Rightarrow \text{accept} \quad \text{H}_0 \\ \#\{i; \hat{\Gamma} > \hat{\Gamma_i}\} > ((1-\alpha)r) \Rightarrow \text{reject} \quad \text{H}_0 \end{cases} \tag{2.18}$$

The decision rule states: accept the null hypothesis if $\hat{\Gamma}$ exceeds at most $((1-\alpha)r)$ of the $\hat{\Gamma_i}$ under the random label hypothesis; reject the hypothesis otherwise.

2.3 Classification

Classifiers are examples of supervised learning machines [32]: classification of a new feature vector depends on knowledge of previous, similar events. The development of an automatic classifier is divided into two phases, the training phase and the test phase. During the training phase, the predictor is given a set of feature vectors together with the label of the class they belong to. The labels are almost always whole numbers. The set of training feature vectors is called the *training set*. The learning machine is able to build a model from this training set. The model then enables the machine to make predictions about new data without the help of an external supervisor. The machine, with inclusion of the derived model, will be called a predictor. During the test phase, feature vectors for which the correct labels are known but which are completely new to the machine, are classified. This set of feature vectors is called the *test set*. The predictor uses its prior knowledge to classify the new objects. The percentage of correctly classified objects is called *success rate* and is used to determine the accuracy of the model [42].

2.3.1 Support vector machines

The first supervised learning machine which will be introduced, is the *support vector machine* (SVM). The SVM approach follows best the sequence presented in 2.3 and will therefore be treated first. It will afterwards be easier to point out where k-NN differs from an ordinary supervised learning machine.

Consider a training set, for which each feature vector is a member of one of the two possible classes. The classes are e.g. regular and disruptive time windows. The training data will be labelled as $(\boldsymbol{x}_i, y_i)$ with $i = 1, \ldots, l$ the index of the feature vector. $\boldsymbol{x}_i \in \mathbb{R}^d$ is the d-dimensional feature vector. The feature vectors should be thought of as points in a Euclidean space. y_i is a label given to the feature vector representing its class and can either be $+1$ or -1.

The simplest implementation of SVM is the linear machine trained on separable data. The non-linear machines trained on non-separable data are treated in a similar way and the linear case therefore will be treated first. In this case, a hyperplane can be constructed, which separates

the two classes in the training set [43]. A hyperplane is the equivalent of a plane in a higher-dimensional Euclidean space. The hyperplane can be described by $H : \boldsymbol{w} \cdot \boldsymbol{x} + b = 0$. $\boldsymbol{w}$ is the vector perpendicular to the plane. $\mid b \mid /\|\boldsymbol{w}\|$ is the perpendicular distance between the origin and the hyperplane. $\|\boldsymbol{w}\|$ represents the Euclidean norm of $\boldsymbol{w}$. Consider d_+ and d_-, which are the perpendicular distances between the closest points of the respective classes and the hyperplane. See figure 2.8 for an illustration. The SVM will maximise the margin of the hyperplane defined as $d_+ + d_-$.

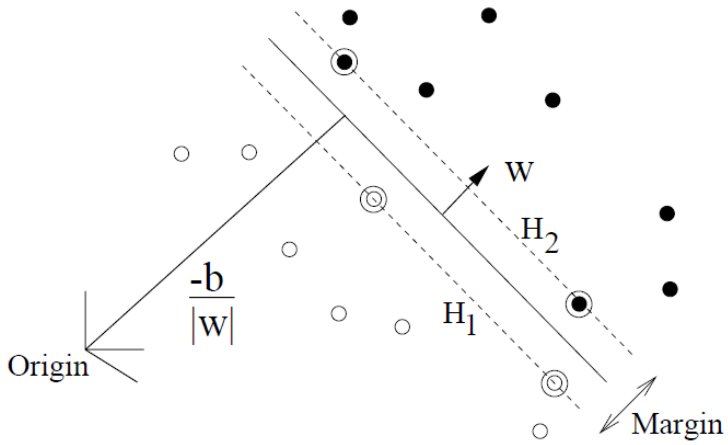

Figure 2.8: A two-dimensional example of a hyperplane separating the two classes in the linear separable case [43].

The next step is to introduce constraints on the training data

$$\begin{cases} \boldsymbol{w} \cdot \boldsymbol{x}_i + b \geq +1, \mathrm{y_i} = +1 \\ \boldsymbol{w} \cdot \boldsymbol{x}_i + b \leq -1, \mathrm{y_i} = -1 \end{cases} \tag{2.19}$$

The constraints can be combined to yield

$$y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b) - 1 \geq 0 \quad \forall i \tag{2.20}$$

By choosing an appropriate scale for $\boldsymbol{w}$ and b, points of each class can be found which fullfil (2.19) with an equality sign. Some points of class $y_i = +1$ lie on a hyperplane $H_+ : \boldsymbol{w} \cdot \boldsymbol{x} + b = +1$. H_+ is parallel to H and has a perpendicular distance to the origin $\mid b - 1 \mid /\|\boldsymbol{w}\|$. Similarly, some points of class $y_i = -1$ lie on a hyperplane $H_- : \boldsymbol{w} \cdot \boldsymbol{x} + b = -1$. H_- is parallel to H and H_+ and has a perpendicular distance to the origin $\mid b+1 \mid /\|\boldsymbol{w}\|$. No points will lie between H_+ and H_-. See also figure 2.8. The margin thus reduces to the distance between H_+ and H_- and is given by $2/\|\boldsymbol{w}\|$. The maximisation of the margin is then equivalent to

$$\begin{cases} \boldsymbol{w} = \mathrm{argmin}_{\tilde{\boldsymbol{w}}} \|\tilde{\boldsymbol{w}}\|^2 \\ y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b) - 1 \geq 0 \quad \forall i \end{cases} \tag{2.21}$$

An equivalent Lagrangian formulation exists for this problem. The derivation of the Lagrangian formulation lies outside the scope of this introduction. Nonetheless, the equations of the Wolfe

dual formulation are given in order to understand how the SVM develops a model for classification [43]. In the Wolfe dual formulation, Lagrangian multipliers $\alpha_i > 0$ are introduced. The problem is rewritten as follows

$$\begin{cases} \text{Maximise} \quad L = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j \boldsymbol{x}_i \cdot \boldsymbol{x}_j \\ \sum_{i=1}^{l} \alpha_i y_i = 0 \\ \alpha_i \geq 0 \end{cases} \tag{2.22}$$

There is also an extra constraint, which can be used to determine $\boldsymbol{w}$ after solving previous convex quadratic problem. The expression for $\boldsymbol{w}$ is

$$\boldsymbol{w} = \sum_{i=1}^{l} \alpha_i y_i \boldsymbol{x}_i \tag{2.23}$$

It should be noted that $\alpha_i \neq 0$ in the solution of (2.22) holds only for points which lie on one of the hyperplanes H_+ or H_-. Those points are called *support vectors* and are the only points which are of interest in a SVM classifier. Removing or rearranging the other feature vectors in such a way that they do not cross H_+ or H_- will not affect the solution of (2.22).

In the non-separable case, no hyperplane exists that is able to separate all feature vectors in two different classes. In this situation an adjustment is made which allows support vectors to lie off the hyperplanes H_+ and H_-. A penalisation is introduced which depends on the distance between the vector and its corresponding hyperplane [43]. The only difference in the Wolfe dual formulation is that the Lagrange multipliers will further be subject to constraints $\alpha_i \leq C/l$, C being a regularisation parameter [42].

In the non-linear case, the data is mapped to a higher-dimensional space which can even be infinite dimensional. Consider a mapping Φ from the original feature space $\mathbb{R}^d$ to the new, higher-dimensional space $\mathcal{H}$, $\Phi : \mathbb{R}^d \to \mathcal{H}$. Because the Lagrangian problem in the linear case (2.22) depends only on scalar products between the support vectors, the same will hold after mapping to $\mathcal{H}$. The Lagrangian formulation for the non-linear case will then only depend on $\Phi(\boldsymbol{x}_i) \cdot \Phi(\boldsymbol{x}_j)$. In practice, the precise mapping or even the space to which one projects is often unknown. As the Lagrangian problem only depends on dot products in $\mathcal{H}$, it suffices to prescribe a function which computes the dot products in this space directly from the original feature vectors. Mathematically, this translates to the knowledge of the kernel function $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \Phi(\boldsymbol{x}_i) \cdot \Phi(\boldsymbol{x}_j)$. The *radial basis function* (RBF) kernel is an example which is used for i.a. classification purposes [1, 42] and is given by

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}{2\sigma^2}\right) \tag{2.24}$$

σ is called the scale parameter of the RBF kernel and is in practice a user-defined parameter, thus chosen that maximum classification rates are achieved. In this particular example, $\mathcal{H}$ is an infinite dimensional space. The ideas about linear separation now hold in $\mathcal{H}$. The Lagrangian formulation for the non-linear case is acquired by replacing $\boldsymbol{x}_i \cdot \boldsymbol{x}_j$ by $K(\boldsymbol{x}_i, \boldsymbol{x}_j)$ in (2.22).

Figure 2.9 illustrates the separating plane for linearly non-separable data. The linear SVM finds a hyperplane, which however is not able to separate the two groups. The RBF kernel performs better. The separating curve in the figure represents a hyperplane in the infinite dimensional

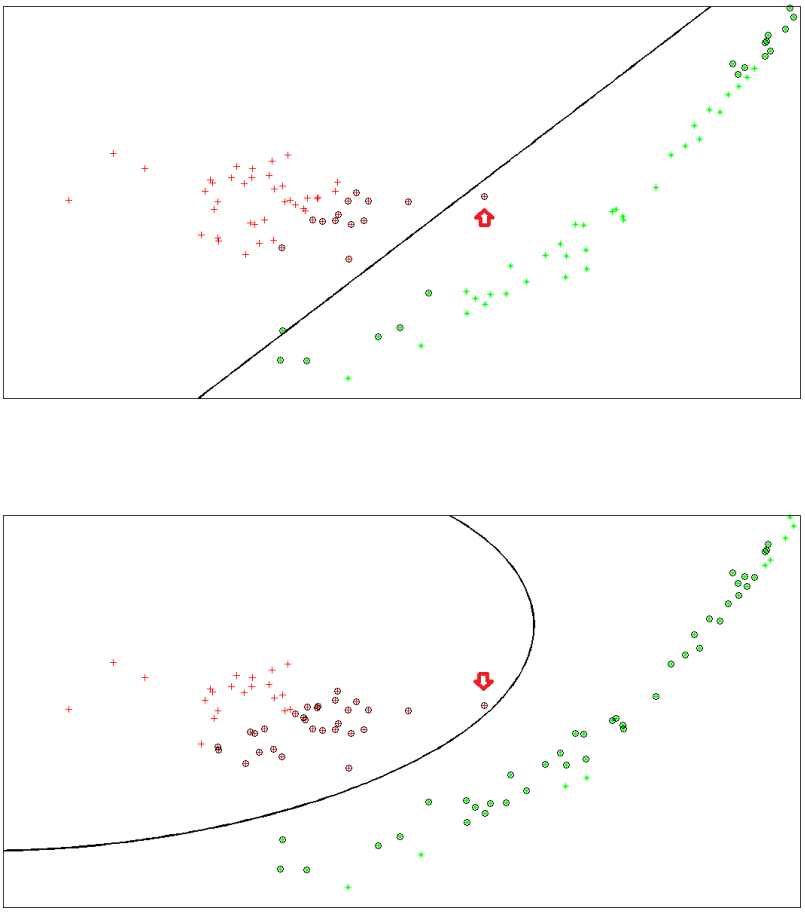

Figure 2.9: Example of the separating hyperplane in two cases. Top: non-separable linear SVM. Bottom: separable non-linear SVM (RBF kernel, $\sigma = 2$). Support vectors are shown by black circles.

space.

Once the training has been accomplished, which in terms of SVM translates to solving (2.22), the learning machine generates a model, or equivalently called a decision function, which allows the machine to make predictions about the class of new objects. In the case of SVM a new object will be classified according to its position compared to the separating hyperplane. When the object lies on the side of H_+, it will be given the label $y = +1$. Otherwise it will be labelled with $y = -1$. The decision function, which determines the class of a new object, is given by

$$D(\boldsymbol{x}) = \sum_{i=1}^{n_s} \alpha_i y_i K(\boldsymbol{s}_i, \boldsymbol{x}) \tag{2.25}$$

The sum runs over all n_s support vectors $\boldsymbol{s}_i$. When the decision function is positive, the object will be labeled $y = +1$, if not it is labeled $y = -1$.

The SVM classifier has been implemented using built-in Matlab routines. The training is done using `svmtrain(.)`. This Matlab function supports various kernel functions. In the tests only the RBF kernel was used. The classification is performed using `svmclassify(.)`, which requires

the output of `svmtrain(.)` as input.

2.3.2 k-NN classifiers

k nearest neighbours (k-NN) is a learning machine with an easy-to-grasp learning procedure. A test object will be classified in the same class as its nearest neighbours in the training set. k-NN does not follow the learning machine scheme of 2.3. First the technique is presented. The procedure is illustrated with an example. At the end, the k-NN and the SVM classifiers are compared based on general properties.

k-NN is an instance-based algorithm and does not develop a decision function [32]. Instead, the learning process is delayed until the classification takes place. The training set is thus not used in the training phase. It is only stored for future access. In the test phase, a new object is given to the k-NN machine. The k-NN machine then computes the distance between the new object and all the objects of the training set. The labels of the k nearest neighbours form a subset. The new object is given the same label as the most frequent class in this subset.

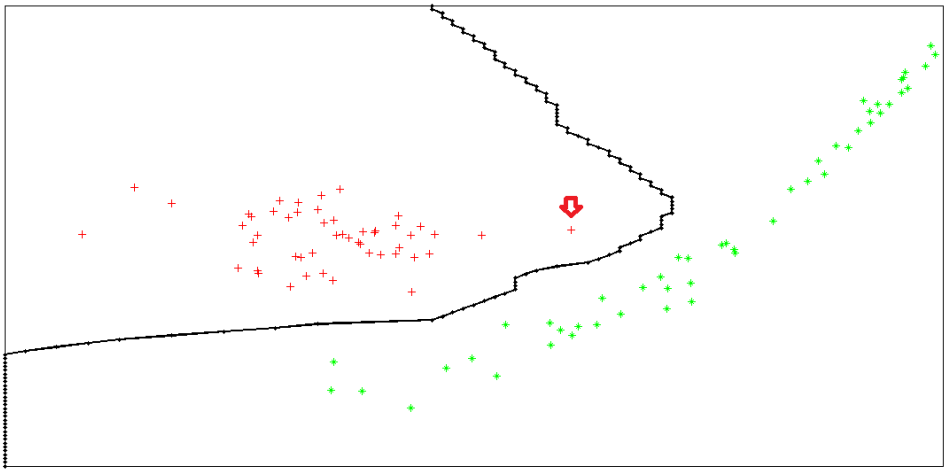

Figure 2.10: An estimation of the separating surface for a 1-NN classifier.

The classification still depends on k and the distance measure. k will be determined in order to acquire the best classification results. Usually $k = 1$ is sufficient. The distance measure is the subject of section 2.4.

As an example, the training set used in the illustration of the SVM training is again considered. In this example, two classes of objects were to be separated by an SVM hyperplane, possibly in a high-dimensional space. The separation surface is inherently present in the k-NN classification. The space can be split into two parts by a surface. A point will be classified according to its position from this surface. The separating surface is not determined explicitly by the k-NN classifier. However, in figure 2.10, an estimation is shown which was found by the k-NN classification of an extensive number of points. In this example, the k-NN separation surface matches well the surface found by the SVM machine using an RBF kernel. The most profound difference is to be found in the middle of the figure. Because of the one red outlier shown by

the arrow, the separating surface approaches considerably the points of the green class. This indicates the sensitivity of k-NN to noise. The SVM separating surface does not exhibit this behaviour.
The k-NN algorithm has been implemented in Matlab. The Matlab code can be found in appendix D, together with some guiding notes.
As the basic concepts of k-NN and SVM have been introduced, a short comparison is given of advantages and disadvantages for both techniques. The summary given here is mainly based on a review article about supervised learning [32].
The machine discussed first was the SVM. The SVM training phase needs a considerable amount of computational time. SVMs also have disadvantages in interpretability and transparency. It is difficult to interpret why a certain vector is classified in one of the classes by the SVM. This is mainly a consequence of the non-transparency of kernels. It is not clear in which high-dimensional space the data is projected to. However, SVMs are heavily used in practice. This is because of their excellent classification performance for multi-dimensional data sets. Furthermore, SVMs have a high tolerance level to irrelevant attributes. Also, once the SVM model is constructed, classification is relatively fast as it only depends on the computation of dot products (possibly by the use of a kernel function) between the new feature vector and a limited number of support vectors.
The k-NN classifier overcomes some disadvantages of the SVM classifier but often does not possess the positive features of the SVM. It is clear that the k-NN classifier does not need any computational time in the training phase. The training set is only stored for future reference. This means that k-NN poses some memory requirements. Although k-NN classifies data in a very transparent way, it has the same interpretability issues as the SVM. A further disadvantage of the k-NN is its sensitivity to irrelevant features and noise. This was already pointed out in the example of figure 2.10. In contrast to the SVM, the k-NN needs more computational time in the test phase, as the distances between the new object and all feature vectors of the training set are computed.
The k-NN is a simple classifier and is computationally inefficient. Despite the disadvantages reported in literature, the k-NN will prove to work remarkably well in the case of disruption prediction. The good performance of the classifier is mainly the result of the use of the geometry of wavelet statistics. The interpretability issue will be solved partly by the visualisation of the data using MDS.

2.3.3 Mahalanobis classifier

A third classifier is based on the Mahalanobis distance. The classifier will in short be addressed by *Mahalanobis classifier*. Consider a multivariate random variable $\boldsymbol{X}$. Experiments give access to the distribution of $\boldsymbol{X}$. In practice, $\boldsymbol{X}$ is the random variable which describes the distribution of the (multivariate) feature vectors of a data set. The complicated distribution of feature vectors in the data space is reduced by using only the two first moments of the distribution, the mean $\boldsymbol{\mu}$ and the covariance or dispersion matrix $\boldsymbol{\Sigma}$[44]. Consider n experimental, multivariate observations (feature vectors) $\boldsymbol{x}_i$ $(i = 1, \ldots, n)$. The moments are

$$\boldsymbol{\mu} = \mathbb{E}[\boldsymbol{X}] = \frac{1}{n}\sum_{i=1}^{n}\boldsymbol{x}_i \tag{2.26}$$

$$\boldsymbol{\Sigma} = \mathbb{E}\left[(\boldsymbol{X}-\mathbb{E}[\boldsymbol{X}])(\boldsymbol{X}-\mathbb{E}[\boldsymbol{X}])^T\right] = \frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{x}_i-\boldsymbol{\mu})(\boldsymbol{x}_i-\boldsymbol{\mu})^T \tag{2.27}$$

The upper index T indicates the transposition operation. The last equality gives the maximum-likelihood sample estimations for respectively $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ [45]. The corresponding statistical model for the distribution of the variable $\boldsymbol{X}$ is the multivariate normal distribution, with PDF given by [7]

$$f(\boldsymbol{X} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{m/2} \mid \Sigma \mid^{1/2}} \exp\left(-\frac{1}{2}(\boldsymbol{X}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{X}-\boldsymbol{\mu})\right) \tag{2.28}$$

m is the number of non-redundant elements in $\boldsymbol{\Sigma}$. For d-dimensional data, $m = d\cdot(d+1)/2$, since $\boldsymbol{\Sigma}$ is symmetric [7]. The idea of the Mahalanobis distance is that it represents the probability that a test point belongs to the cluster. The Mahalanobis distance between a test point $\boldsymbol{x}$ and the cluster represented by the random variable $\boldsymbol{X}$ is given by [46]

$$d_M(\boldsymbol{x}, \boldsymbol{X}) = \left((\boldsymbol{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right)^{1/2} \tag{2.29}$$

It is apparent from (2.29) that points $\boldsymbol{x}$, at the same Mahalanobis distance from the cluster, lie on a surface with a constant value of the density in (2.28). The density is proportional to the probability of $\boldsymbol{X}$ lying in a very small neighbourhood around $\boldsymbol{x}$[46]. This is also illustrated in figure 2.11. The Mahalanobis distance is thus indeed representative for the probability that $\boldsymbol{x}$ belongs to the cluster.

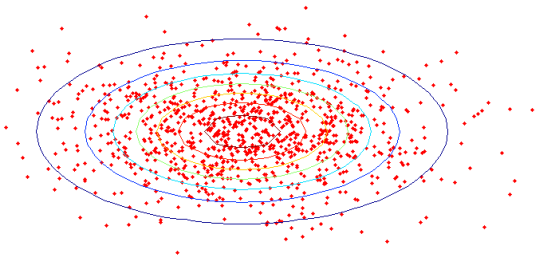

Figure 2.11: Points drawn from a bivariate normal distribution. The points are distributed according to the density function of the random variable $\boldsymbol{X}$. Points which lie on the same ellipse, are also at the same Mahalanobis distance from the cluster.

The Mahalanobis distance can be used to develop a classifier. Consider a two-class problem. The training data of each class forms a cluster. The clusters are well-separated. Consider for example the same set of two-dimensional points as in 2.3.1 and 2.3.2. During the training phase, the machine will calculate the moments for both clusters according to (2.26) and (2.27). Simple Matlab built-in routines exist to this end. `mean(.)` and `cov(.)` are used to respectively estimate $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$. The classifier will classify a new point in the cluster for which the Mahalanobis distance

is minimal.
The result for the two-dimensional data set is shown in figure 2.12. Once more the separation surface was estimated by the classification of an extensive number of points. The ellipses in the figure represent contour plots of constant density according to (2.28) for each cluster. Note that the red point of the training set in the middle of the figure is classified in the wrong class. This is the result of loss of information about individual points of the training set in the model. The effect of this point was also discussed for the other two classifiers in 2.3.1 and 2.3.2. The k-NN classifier was sensitive to the outlier.

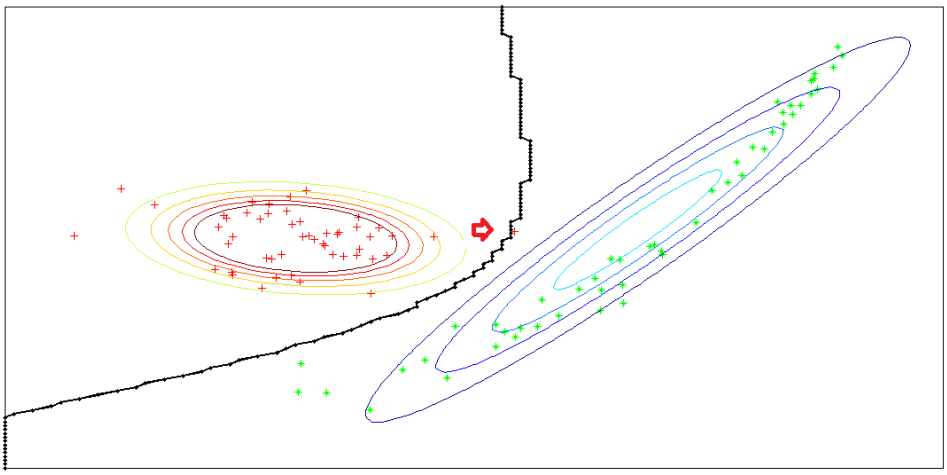

Figure 2.12: Estimated separating surface for the Mahalanobis classifier.

2.4 Similarity measures

Pattern recognition techniques such as k-NN are essentially based on geometric concepts, such as distance, or alternatively called the *similarity measure*. The Euclidean distance is the most simple example of a similarity measure. It will be treated in section 2.4.1. However, statistics of the frequency spectra are inherently probabilistic. The Euclidean distance is not able to account properly for the probabilistic nature of the data. Other similarity measures, originating from the field of *information geometry*, treat similarities between probabilistic data in a optimal way [47]. The proposed similarity measure will be the *Rao geodesic distance*. The idea is that probability distributions lie on a curved *manifold*. Information geometry thus provides a geometrical framework for spaces of probability density functions. How this is done will be the subject of section 2.4.3.

2.4.1 Euclidean distance

Euclidean geometry is vastly used in different domains because it is the geometry we encounter in our everyday life. Concepts based on properties of Euclidean spaces are also often present in pattern recognition. An example is the Euclidean distance in the exponent of the RBF kernel

of an SVM machine. Also expressions (2.26) and (2.27) for the mean and the covariance matrix is only valid in affine spaces, such as a Euclidean space.
The Euclidean geometry cannot take into account the inherent probabilistic nature of the data in a proper way. An introduction is nevertheless given. The goal is to use the Euclidean distance in the tests and prove its limited potential in the prediction of disruptions. Furthermore, the Euclidean geometry will be introduced in a general way to illustrate how in a next step, distances between distributions will be defined.
Consider a two-dimensional Euclidean space. The length of an infinitesimal line element between two points, the *metric* ds, obeys following relationship

$$\begin{aligned} ds^2 = dx^2 + dy^2 &\equiv g_{xx}(x,y)dx^2 + (g_{xy}(x,y) + g_{yx}(x,y))dxdy + g_{yy}(x,y)dy^2 \\ &\equiv g_{\mu\nu}(\boldsymbol{x})d\boldsymbol{x}^\mu d\boldsymbol{x}^\nu \end{aligned} \tag{2.30}$$

The second equality is the definition of the metric tensor. The metric tensor is of fundamental importance in differential geometry. It describes how distances are to be measured in a certain coordinate system. The last equality introduces the Einstein convention. For each index which appears both as lower and upper index in a single term, a summation is assumed over all its possible values. In (2.30) for example, a summation is present over $\mu \in \{x, y\}$ and $\nu \in \{x, y\}$.
In the case of a Euclidean space, the metric tensor reduces to the Kronecker delta tensor: $g_{\mu\nu}(\boldsymbol{x}) = \delta_{\mu\nu}$. The relation between ds^2 and $g_{\mu\nu}$ is a very general expression. The distance between distributions will be described by the metric tensor for a curved probabilistic space.
The metric defines the topology of a metric space [44], meaning distances and paths between two arbitrary points are exactly defined once the metric is chosen. In the context of a Euclidean space, the Euclidean distance can be derived from the Euclidean metric. The Euclidean distance is given by

$$d_E(\boldsymbol{x}_i, \boldsymbol{x}_j) = \left(\sum_{\mu=1}^{d} \left(\boldsymbol{x}_i^\mu - \boldsymbol{x}_j^\mu \right)^2 \right)^{1/2} \tag{2.31}$$

2.4.2 Kullback-Leibler divergence

The Euclidean similarity measure does not take into account the probabilistic nature of the data. A popular distance measure between distributions in probability theory is the Kullback-Leibler divergence (KLD) [48]. The KLD between distribution P and Q with respective PDFs $p(\boldsymbol{x})$ and $q(\boldsymbol{x})$ is given by

$$KLD(P\|Q) = \int p(\boldsymbol{x}) \log \left(\frac{q(\boldsymbol{x})}{p(\boldsymbol{x})} \right) d\boldsymbol{x} \tag{2.32}$$

When the log base in this expression is 2, the KLD can be seen as the number of additional bits that are required to get samples from P, using a code erroniously based on Q [49]. Because of its resemblance to entropy, the KLD is sometimes called *relative entropy*. The KLD for two GGD distributions P and Q with respectively parameters (α_1, β_1) and (α_2, β_2) is given by [34]

$$KLD(P\|Q) = \ln \left(\frac{\beta_1 \alpha_2 \Gamma(1/\beta_2)}{\beta_2 \alpha_1 \Gamma(1/\beta_1)} \right) + \left(\frac{\alpha_1}{\alpha_2} \right)^{\beta_2} \frac{\Gamma((\beta_2+1)/\beta_1)}{\Gamma(1/\beta_1)} - \frac{1}{\beta_1} \tag{2.33}$$

This expression is not symmetric for interchange of indices *1* and *2*. In the tests, a symmetrised alternative of the KLD was used, the J-divergence which is given by $J(P\|Q) = 1/2 \cdot [KLD(P\|Q) + KLD(Q\|P)]$ [48]. When β is held fixed ($\beta_1 = \beta_2 = \beta$), the J-divergence can be rewritten after some manipulations as

$$J - div(\alpha_1\|\alpha_2) = \frac{1}{\beta}\left[\cosh\left(\beta \ln\left(\frac{\alpha_2}{\alpha_1}\right)\right) - 1\right] \tag{2.34}$$

This expression will be used in section 2.4.3.

The GGDs for different signals or different wavelet scales are assumed to be independent. For independent distributions, the joint distribution is the product of the individual distributions. For the joint PDF, the total KLD can be computed by additionality. In the case of joint PDFs $\prod_i P_i$ and $\prod_i Q_i$, where P_i and Q_i ($i = 1, \ldots, n$) describe the same variable, and where the variables for different values of i are independent, the total KLD is given by [48]

$$KLD\left(\prod_{i=1}^{n} P_i \| \prod_{i=1}^{n} Q_i\right) = \sum_{i=1}^{n} KLD(P_i\|Q_i) \tag{2.35}$$

The assumption of independent wavelet statistics for the same wavelet scale and for different signals could be questioned. It is logical to expect some correlation between different signals on physical grounds. However, the correlation is almost not present in a single time window of 30 ms.

The KLD measure exhibits some disadvantages. In particular, it does not satisfy the symmetry criterion $d(P\|Q) = d(Q\|P)$ and it does not follow the triangle inequality. It is therefore sometimes called a *pseudo-distance*. The KLD will be used in the tests because of the existence of a closed form expression for the distance between GGDs with different β. A closed form expression for the Rao geodesic distance does not exist in this case. A closed form expression is convenient for classification, as the algorithms need to work sufficiently fast in real-time applications.

2.4.3 The Rao geodesic distance

For the purpose of classification, a proper notion of similarity between distributions is necessary. In the field of information geometry, the similarity between distributions arises in a natural way. Probability density families are interpreted as Riemannian, differentiable manifolds. A point on the manifold corresponds to a PDF within the family. The family parameters provide a coordinate system on the manifold. The distance between two distributions is the length of the shortest path between the distributions on this manifold. The shortest path is also called the *geodesic*.

An illustration is given in figure 2.13. The surface represents the manifold of univariate Gaussian distributions. The manifold is two-dimensional, as Gaussian PDFs are described by a mean μ and a standard deviation σ. The manifold has been visualised by an MDS embedding into a three-dimensional Euclidean space [50]. The end points of the geodesic are $p_1 : \mu = -4, \sigma = 0.7$ and $p_2 : \mu = 3, \sigma = 0.2$.

Another example is the family of univariate, zero-mean GGDs with fixed β. For this one-dimensional Riemannian space, it is possible to construct a faithful representation in a two-dimensional Euclidean space. The representation is a curved line. The curvature along the line is

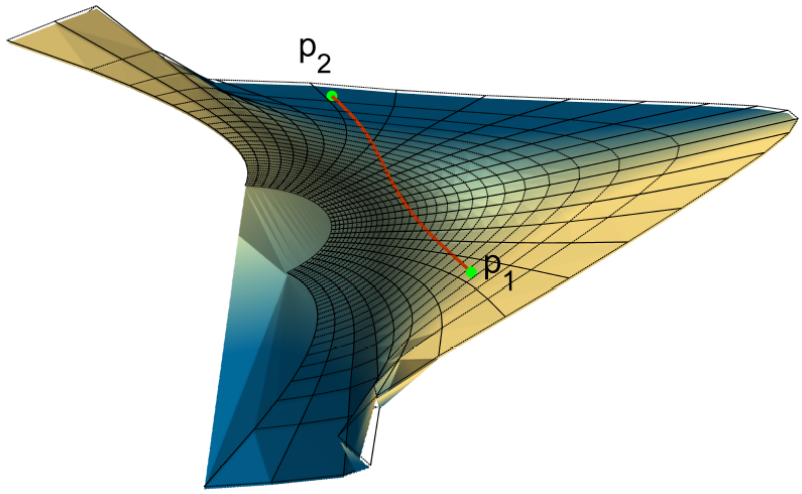

Figure 2.13: Embedding of the univariate Gaussian manifold and a geodesic between distributions p_1 and p_2.

identical to the curvature of the original manifold. Pairwise distances have to be measured along the curve. The representation of the manifold of univariate, zero-mean Laplacian distributions ($\beta = 1$) is shown in figure 2.14. How this equivalent representation is constructed, is shown in appendix C.

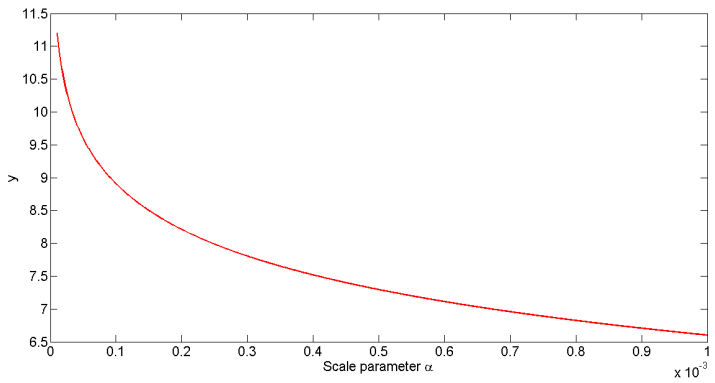

Figure 2.14: Equivalent representation of the manifold of univariate, zero-mean Laplacian distributions in a two-dimensional Euclidean space.

The manifold of a family of probability distributions is a metric space and is defined by a metric, as was the Euclidean space in 2.4.1. The metric coefficients were first derived by Fisher [47]. The interpretation of the work of Fisher as a metric on a curved manifold was given later by Crámer and Rao. The metric of the probabilistic space is also called the Rao-Fisher metric. Consider a family of PDFs of the form $f(\boldsymbol{x} \mid \boldsymbol{\theta})$. The parameters of the distribution are collected in a single vector $\boldsymbol{\theta}$. In the case of GGDs for example, $\boldsymbol{\theta} = (\alpha, \beta)$. For a random vector $\boldsymbol{X}$, the coefficients of the Fisher-Rao metric are found through the relationships [47]

$$g_{\mu\nu}(\boldsymbol{\theta}) = -\mathbb{E}\left[\frac{\partial^2}{\partial\theta^\mu \partial\theta^\nu} \ln f(\boldsymbol{X} \mid \boldsymbol{\theta})\right] \tag{2.36}$$

The idea of the Fisher-Rao metric is that the distance between distributions of the same family can be measured by the amount of information that $\boldsymbol{X}$ yields on the parameters of the distribution through the likelihood [7]. The distance between distributions with parameters $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$ can be found by the integration of the distance metric

$$d_G(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2) = \int_{\boldsymbol{\theta}_1}^{\boldsymbol{\theta}_2} ds = \int_0^1 \sqrt{g_{\mu\nu}(\boldsymbol{\theta}(t)) \frac{d\boldsymbol{\theta}^\mu(t)}{dt} \frac{d\boldsymbol{\theta}^\nu(t)}{dt}} dt \tag{2.37}$$

For the second equality use was made of expression (2.30). Again the Einstein convention is used. In addition, the components of $\boldsymbol{\theta}$ were parametrised along the geodesic which starts at $\boldsymbol{\theta}_1$ ($t = 0$) and ends at $\boldsymbol{\theta}_2$ ($t = 1$). There exists a well-known solution scheme to find the geodesic between two points on a Riemannian manifold. The geodesic can be found as the solution of the Euler-Lagrange equations [51].

For the GGDs, a closed-form expression for the geodesic distance only exists in case of a fixed shape parameter β [7]. In the case of univariate GGDs, the geodesic distance reduces to

$$d_G((\alpha_1, \beta); (\alpha_2, \beta)) = \sqrt{\beta} \left| \ln\left(\frac{\alpha_2}{\alpha_1}\right) \right| \tag{2.38}$$

The derivation of this expression is given in appendix A. For a Gaussian distribution, the geodesic distance is given by $\sqrt{2} \mid \ln(\alpha_2/\alpha_1) \mid$ and for a Laplacian distribution by $\mid \ln(\alpha_2/\alpha_1) \mid$.

As mentioned for the KLD, the statistics for wavelet coefficients of different wavelet scales or signals are assumed to be independent. The joint distribution function over all different signals and wavelet scales is again the product of several univariate PDFs. Consider the joint PDF of two different time windows with respective parameters $\boldsymbol{\theta}_1 = (\alpha_{11}, \beta_1, \ldots, \alpha_{1n}, \beta_n)$ and $\boldsymbol{\theta}_2 = (\alpha_{21}, \beta_1, \ldots, \alpha_{2n}, \beta_n)$. n represents the number of independent GGDs. The geodesic distance for the joint PDF is then given by the square root of the sum of the squared individual distances, i.e. [22]

$$d_G(\boldsymbol{\theta}_1; \boldsymbol{\theta}_2) = \sqrt{\sum_{k=1}^{n} d_G^2((\alpha_{1k}, \beta_k); (\alpha_{2k}, \beta_k))} \tag{2.39}$$

Equations (2.33) and (2.38) express two different distance measures between univariate GGDs. In case of a fixed β, both measures become a function of only the fraction α_2/α_1 and β. Instead of the KLD, the J-divergence is a more appropriate measure for our purposes. In figure 2.15, the geodesic distance given by (2.38) and the J-divergence given by (2.34) are compared for different values of the shape parameters. The solid lines represent the geodesic distance, while the broken lines represent the J-divergence. The same color is used to indicate the same value of β. The plot was made for a range of α_2/α_1 close to one, as the distinctive capability of the measures for nearly identical distributions is the most critical issue in classification. In addition, as both measures are symmetrical under the substitution $\alpha_2/\alpha_1 \to \alpha_1/\alpha_2$, it is sufficient to limit the range of α_2/α_1 to values higher than one.

Some conclusions can be drawn from the figure. First of all, all curves start at zero for $\alpha_2/\alpha_1 = 1$. This is necessary, as the distance between identical distributions needs to be zero. Second,

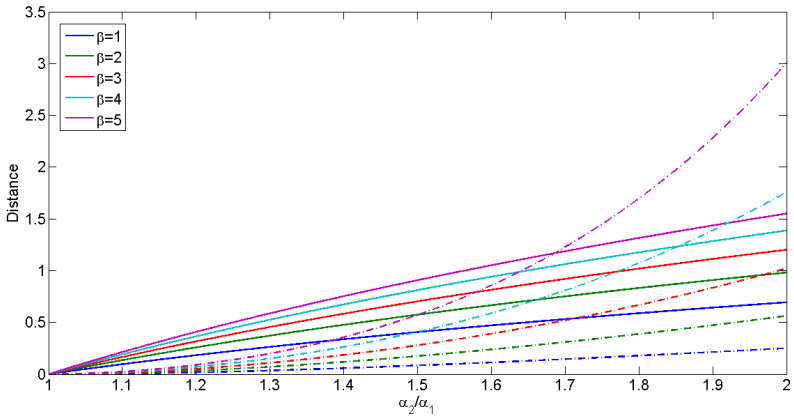

Figure 2.15: Geodesic distance and J-divergence measures for different fixed values of β.

both measures appear to distinguish better distributions with a fixed ratio of α_2/α_1, when the distributions have a higher value of the shape parameter β. The most important conclusion is, however, that at small values of the ratio α_2/α_1, the geodesic distance increases faster than the J-divergence. This implies that, for very similar distributions, a classifier based on wavelet statistics will make a better distinction if the geodesic distance is used.

2.5 Exponential and logarithmic maps

The Mahalanobis classifier cannot be used in case data lies on curved manifolds. The use of the Mahalanobis distance is only justified when the multivariate observations $\boldsymbol{x}_i$ are points in a d-dimensional Euclidean space. Euclidean spaces are affine spaces, in which the substraction of two vectors results in a vector pointing from one vector to the other. This has been implicitly used for the maximum likelihood estimations in (2.26) and (2.27). Riemannian manifolds, e.g. the one on which the wavelet statistics lie, is not affine. The Mahalanobis distance has thus to be generalised. The Mahalanobis distance will be computed in the *tangent space* at a point of the manifold. The notion of a tangent space is equivalent to the well known tangent line in a two-dimensional Euclidean space, or the tangent plane in a three-dimensional Euclidean space. Figure 2.16 illustrates this. The sphere is in fact the three-dimensional embedding of a two-dimensional Riemannian manifold in a Euclidean space. In the figure, the tangent space (a plane, or equivalently a two-dimensional Euclidean space) at the red point of the sphere is shown. The tangent space is Euclidean and thus affine. Equations (2.26), (2.27) and (2.29) will hold in the tangent space.

Generally speaking, a d-dimensional Euclidean space $\mathbb{R}^d$ can be constructed at each point of a d-dimensional differentiable manifold $\mathcal{M}$, tangent to $\mathcal{M}$ [52]. The tangent space at a point $\boldsymbol{x} \in \mathcal{M}$ is denoted by $T_{\boldsymbol{x}}\mathcal{M}$. There exists locally a diffeomorphism between $\mathcal{M}$ and $T_{\boldsymbol{x}}\mathcal{M}$ [44] in each point $\boldsymbol{x} \in \mathcal{M}$. A diffeomorphism is an isomorphism between two differentiable manifolds [53]. The transformation from the tangent space to the manifold is called the *exponential map* [44]

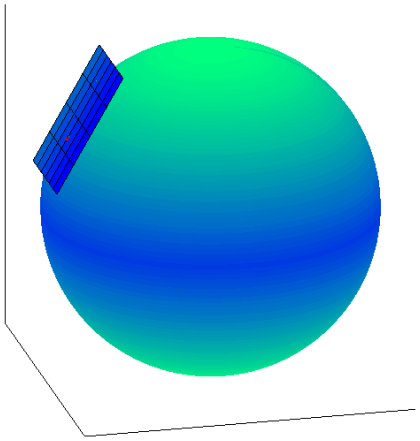

Figure 2.16: Tangent plane at a point of a sphere.

$$\exp_{\boldsymbol{x}} \quad : \begin{cases} T_{\boldsymbol{x}}\mathcal{M} \to \mathcal{M} \\ \vec{x\boldsymbol{y}} \in T_{\boldsymbol{x}}\mathcal{M} \to \exp_{\boldsymbol{x}}(\vec{x\boldsymbol{y}}) = y, \quad y \in \mathcal{M} \end{cases} \tag{2.40}$$

The second line in the above expression represents the transformation of a vector in the tangent space to a point on the manifold. In a close neighbourhood of $\boldsymbol{x}$, a unique geodesic in $\mathcal{M}$ connecting $\boldsymbol{x}$ and $\boldsymbol{y} \in \mathcal{M}$ corresponds to the vector $\vec{x\boldsymbol{y}} \in T_{\boldsymbol{x}}\mathcal{M}$. Furthermore, the length of the geodesic equals the length of the tangent vector: $d_G(\boldsymbol{x}, \boldsymbol{y}) = \|\vec{x\boldsymbol{y}}\|$, where $\|\cdot\|$ represents the Euclidean norm of the vector. In the region where the diffeomorphism exists, the inverse transformation is called the *logarithmic map*

$$\log_{\boldsymbol{x}} \quad : \begin{cases} \mathcal{M} \to T_{\boldsymbol{x}}\mathcal{M} \\ y \in \mathcal{M} \to \log_{\boldsymbol{x}}(y) = \vec{x\boldsymbol{y}}, \quad \vec{x\boldsymbol{y}} \in T_{\boldsymbol{x}}\mathcal{M} \end{cases} \tag{2.41}$$

A famous example of the application of the logarithmic map is the azimuthal equidistant projection of the earth's surface. The earth is approximated by a sphere, the manifold, and the logarithmic map allows to project the earth's surface on a two-dimensional plane. This representation is useful, as distances from the point $\boldsymbol{x} \in \mathcal{M}$, at which the projection is taken, are conserved [54]. An example is shown in figure 2.17.

The exponential and logarithmic maps are used in the Mahalanobis classifier to project the data of wavelet coefficients to an affine space, in which the Mahalanobis distance can be computed. The wavelet statistics are modelled by univariate Laplacian distributions. The derivation of the exponential and logarithmic maps in this case is presented in appendix B. The logarithmic map of a Laplacian distribution with scale parameter α_2, taken at a Laplacian distribution with scale parameter α_1 is

$$\alpha_1\vec{\alpha}_2 = \operatorname{sgn}\left(\alpha_1 \ln\left(\frac{\alpha_2}{\alpha_1}\right)\right) d_G(\alpha_1 \| \alpha_2) \tag{2.42}$$

$d_G(\alpha_1\|\alpha_2)$ is the geodesic distance between the distributions. The inverse transformation, the exponential map from a tangent vector to a point on the manifold is

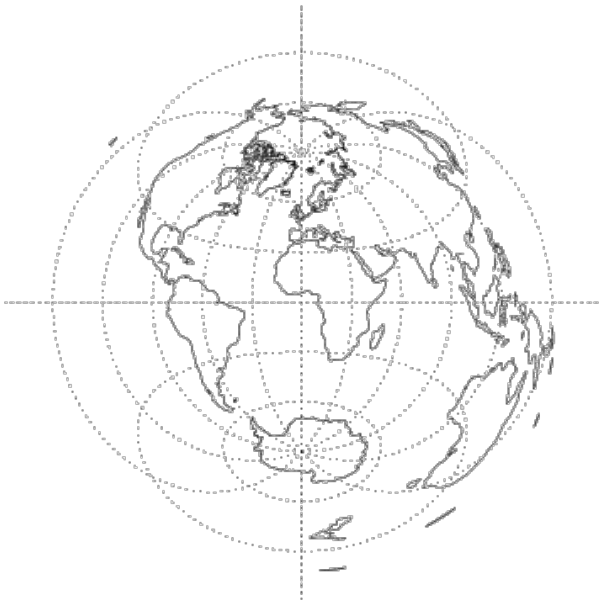

Figure 2.17: Azimuthal equidistant projection [54].

$$\alpha_2 = \alpha_1 \exp(\alpha_1 \vec{\alpha}_2) \tag{2.43}$$

The logarithmic map allows to project a cluster of wavelet distributions to an affine space in which the cluster is described by a multivariate Gaussian distribution. An issue to be addressed is the choice of the point at which the logarithmic map should be taken. A good choice is the *geodesic centre-of-mass* [44]. Note that the centre-of-mass of a set of points in a Euclidean space minimises the sum of squared distances [55]

$$\boldsymbol{\mu} = \arg\min_{\boldsymbol{y}\in\mathbb{R}^d} \sum_{i=1}^{n} d_E^2(\boldsymbol{x}_i, \boldsymbol{y}) \tag{2.44}$$

The generalisation for a manifold is straightforward: the minimisation is now done over $\boldsymbol{y} \in \mathcal{M}$ and the Euclidean distance is replaced by the geodesic distance: $d_E \to d_G$. The minimisation is performed by a gradient descend algorithm [44]. In each step, all the points are projected to the tangent space at the temporary centre-of-mass. A centre-of-mass is computed in the tangent space using the formula for the mean in a Euclidean space (similar to (2.26)). The centre-of-mass in the tangent space is projected back to the manifold by the exponential map. It then serves as a starting point for the next iteration step. The procedure is repeated until convergence is reached. The mean is initialised at the beginning of the procedure by taking the Euclidean mean of the non-Euclidean coordinates.
All the steps for the Mahalanobis classifier in Riemannian spaces have been discussed. A short summary is given here:

1. Training phase: For each class
    - Compute the geodesic centre-of-mass using the gradient descend algorithm.
    - Project all points to the tangent space at the geodesic centre-of-mass using the logarithmic map.

- Compute the mean and covariance matrix according to (2.26) and (2.27).

2. Test phase: For each test object

   - Project to the tangent space at the geodesic centre-of-mass of each class using the logarithmic map.
   - Compute the Mahalanobis distance between the point and each cluster.
   - Classify the point in the same class as the closest cluster.

Chapter 3

Experiments

In chapter 2, several techniques were presented to develop a disruption predictor. Different classifier models were introduced: SVM, k-NN and Mahalanobis classifiers. Moreover, different similarity measures were proposed in the case of a k-NN classifier. The choice of the similarity measure depends on the description of the data. In this chapter, the performance of different classifier models will be presented. A good disruption predictor should perform well in different aspects:

- The feature extraction and classification for a time window of 30 ms should last shorter than 30 ms in order to make real-time classification possible.
- Regular time windows should be classified correctly in order to avoid false alarms.
- At least one disruptive time window should be classified as disruptive to detect a disruption.
- The disruption should be predicted well in advance in order to be able to undertake mitigating actions.
- Predict the initiating event of the disruption in order to undertake the correct mitigating actions.

Section 3.1 describes by which criterion each of those aspects will be quantified in the tests. Different parameters need to be tuned in the SVM and k-NN classifiers. The parameters are determined so that optimal results are acquired. This is the subject of section 3.2. Section 3.3 presents a visualisation of the data by an MDS embedding. The MDS embedding is useful to acquire intuitive insights about the wavelet statistics and the value of using the geometry of wavelet distributions. More precisely, it becomes clear that the use of wavelet features in combination with the Rao geodesic distance distincts clearly regular and disruptive features in different clusters. In sections 3.4 to 3.8, different tests are presented to compare the performance of the classifiers. The classifiers have been tested on basis of disruption prediction, generalisation performance to data sets of more recent campaigns, the ability to detect the disruption cause, relevance of the signals in the data set and computational cost. Each aspect is discussed in a different section.

3.1 Classification rates

A widely used measure for the performance of a classifier is the *success rate*. First all feature vectors of the test set are classified. The correct class for each object of the test set is known[1]. Once the classification is performed, the success rate is defined as the ratio, in percent, of correctly classified objects [42].
In following tests, the success rate was calculated for each class individually. Each window can either be disruptive or regular. The share of correctly classified disruptive windows is called the *true positive rate* (TPR). Similarly, the share of correctly classified regular windows is the *true negative rate* (TNR). *False positive rate* (FPR) and *false negative rate* (FNR) are respectively the share of incorrectly classified regular and disruptive events. Note that when the number of regular and disruptive events in the test set are the same, FPR is the complement of TNR. Similarly, FNR is the complement of TPR. In the tests, the number of regular events was indeed the same as the number of disruptive events. It thus suffices to mention only the TPR and the FPR. TPR is a measure for the ability of the classifier to detect disruptive events. FPR represents the share of regular events in which the classifier erroniously predicts a disruption.
The TPR and FPR are not the most appropriate measures for a real-time disruption predictor. One is rather interested in the ability of the classifier to avoid false alarms and to predict upcoming disruptions, sufficiently in advance to undertake mitigating actions. Therefore all regular windows should be classified as regular and at least one disruptive time window should be classified as disruptive. In further tests, *success rate* (SR) is used to indicate the share of shots, rather than time windows, for which the predictor successfully detected an upcoming disruption and did not produce a false alarm earlier in the shot. There are two ways by which a predictor could be mistaken. The possibility of a false alarm has been mentioned, i.e. a shot in which a disruption is detected too early. To test whether the classifier was able to detect correctly the regular regime, time windows of more than one second before ToD were considered regular. Such time windows do not contain information about the upcoming disruption. One second is intented as a safe margin. It has been noted in previous work that 180 ms is the maximum time in advance to predict reliably an upcoming disruption, when using an SVM classifier trained on Fourier statistics [3]. The predictor could also miss an upcoming disruption. In this case, none of the time windows close to the disruption is classified as disruptive. Such an error is called a *missed alarm*. In the following sections, the terms *false alarm rate* (FA) and *missed alarm rate* (MA) will be used to indicate the fraction of shots in which respectively a false alarm or a missed alarm was found. The *total error* (TE) is the sum of the false alarm rate and the missed alarm rate. Finally, one is also interested in the mean time before the disruption at which the predictor is able to detect upcoming disruptions, as this determines the time interval in which preventive measures are to be taken. This time is called the *average time* (AVG). The average time should be as high as possible.
The classification results are subject to variations due to the exact choice of the training and test set. All classification rates and the average time were therefore computed for different training and test sets in consecutive iterations. The quantities are presented in the form of $\mu \pm \sigma$, where μ represents the mean value and σ the standard deviation after multiple iterations.

[1] The time of disruption determined by experts was used as ground truth.

Mainly due to computational considerations, the real-time classification rates were computed using the same number of regular and disruptive time windows in the test set. The mean duration of a disruptive pulse at JET between campaigns C15 and C27 was about 15.5 s. Only the last 210 ms were considered disruptive in the classification tests. This resulted in about 500 regular and six disruptive time windows per shot. The classification of all time windows in the test set was not possible, because of the memory demands in the computation of the proximity matrix between all objects in the training set and the objects in the test set. The experiments were conducted on a Intel Core i3-2310M, 2.1 GHz processor with 4 GB of RAM.
Use of a limited number of regular windows limits the interpretation of the results as real-time equivalent simulations. However, the approach still permits to compare the classification rates of the different classifiers. The simulations in a real-time environment should be treated in future work. Note that for a real-time predictor, the memory constraint will not be present. Such a classifier will only need to classify the features of one time window. The memory constraint will only depend on the number of objects in the training set.

3.2 Choice of appropriate parameters

3.2.1 Shape parameter for the GGD

The statistics of wavelet detail coefficients are modeled by a GGD. For regular time windows however, unrealistic values of the shape parameter β appear. The time evolution of the shape parameter on the highest wavelet decomposition level has been investigated to some extent. To this end, the thirteen time traces from all shots of data set SA were processed to extract the parameters of the wavelet statistics. β was allowed to vary within a certain interval. First, a fit of the GGD was performed unconstrained according to the procedure discussed in section 2.1.3. If β exceeded 5, the fit was repeated by using a fixed $\beta = 5$. The shape parameter was thus allowed to vary in the interval $0 < \beta \leq 5$.

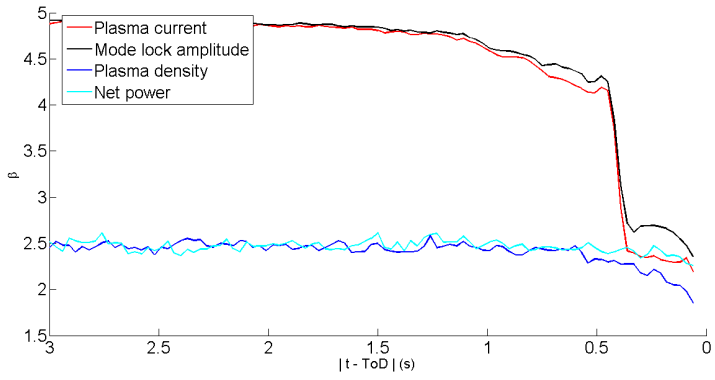

Figure 3.1: Evolution of the shape parameter before disruption.

The evolution of the shape parameter during the last 3 s before disruption in shown in figure 3.1. The plotted values are mean values over all shots of the data set. The shape parameter is plotted for wavelet statistics of the plasma current, the mode lock amplitude, the plasma density and net power. The mean shape parameter for all other signals did not differ substantially from $\beta = 5$

during the last 3 s before ToD. This is mainly a consequence of the low original sampling rate of those signals. Too low sampling rates have a low energy content, which leads to very short tails in the GGD distribution and thus in high values for the shape parameter. The evolution of the shape parameter is most interesting for the plasma current and the mode lock amplitude. During the majority of the time interval, the shape parameter remains at high values, which indicates that the current and the mode lock amplitude vary only slightly during the normal operation of the machine (within the time window of 30 ms). The shape parameters for both signals fall off significantly at about 360 ms before ToD. Lower values of β indicate the transient behaviour in the data, which is exactly what is expected in the last instants before a disruption occurs. The sudden change in the shape parameter at 360 ms before disruption indicates that the disruption becomes, on average, visible in the wavelet spectra at this instant. 360 ms is thus representative for the time scale at which classifiers based on wavelet statistics could reliably detect a disruption.
The shape parameters of the wavelet statistics for the plasma density and net power do not vary significantly in figure 3.1, although some decreasing trend is visible at about 200 ms before ToD. As figure 3.1 only shows mean values for β, this implies that only in a limited amount of shots, the transient behaviour is apparent in the shape parameter fitted to the wavelet statistics. Again this is a consequence of the low sampling rate for these signals.
Classifiers based on wavelet statistics to which a GGD with variable β was fitted were compared with classifiers for which a GGD with a fixed β was fitted to the wavelet statistics. The variable β was allowed to vary in the range $0 < \beta \leq 5$. The fixed β was chosen in such a way, as to obtain the highest classification rates. Two data sets were constructed to determine the optimal β. Time traces of shots of SB in table 1.3 were processed. The signals were decomposed with at a wavelet decomposition level $N_S = 3$. For the first data set, the β was held fixed to one, i.e. the wavelet statistics were modelled by a univariate, zero-mean Laplace distribution. For the second data set, the β was held to fixed to two. The wavelet statistics in this set were thus modelled by a univariate, zero-mean Gaussian distribution. Both sets were classified by a k-NN classifier. The Rao geodesic distance was used as similarity measure. The tests were repeated twenty times. At each iteration, 65% of the shots were randomly selected to construct the training set. The data of the remaining shots was classified and compared to the ground truth, i.e. the evolution of the shot determined by experts.
The success rate of the data set where a Laplace distribution was used, was $86.9\% \pm 4.2\%$. The classification rate using a Gaussian distribution model was $83.2\% \pm 3.6\%$. In all following tests where the β was held fixed, the wavelet statistics were modelled by a Laplace distribution.
The fixed value of $\beta = 1$ will prove to be sufficient for disruption detection. In the future, however, it will also become interesting to detect the disruption cause. The use of a variable β does not seem to have a significant added value in the detection of the disruption cause (see also section 3.6). In this case, it could however prove interesting to study the evolution of β for different disruption causes, in order to choose different fixed β values for each signal.

3.2.2 Wavelet decomposition level

The wavelet decomposition level is an important parameter in the feature extraction. The use of a low decomposition level results in an inadequate description of the original signal. This became

clear in figure 2.3. The decomposition with only one subband of detail functions (upper right figure) results in a poor approximation of the original signal. A too high decomposition level will, on the other hand, contain redundant information. This is a result of the use of the non-decimated wavelet transform. Furthermore, as more subbands are added in the decomposition, the wavelet transform becomes more computationally expensive. A compromise between the two demands will be important in a real-time application.
A simulation was performed to find the optimal number of wavelet subbands. Time windows of five different data sets were classified, each having a different number of wavelet subbands, ranging from $N_S = 1$ to $N_S = 5$. The wavelet statistics were modelled by a GGD with $\beta = 1$. A k-NN classifier was used with the Rao geodesic distance being the similarity measure ($k = 1$). Shotnumbers of subset SB in table 1.3 were used. 65% of the shots were chosen randomly to construct the training set. Data from the other shots were used in the test set. The test was repeated twenty times. The same shotnumbers were used for the five different data sets in each iteration. The results are summarised in table 3.1.

Table 3.1: Success rates of a k-NN classifier for different number of subbands.

N_S	SR
1	80.1 ± 3.8
2	79.7 ± 3.8
3	81.7 ± 4.9
4	83.3 ± 4.7
5	86.6 ± 3.4

Even higher decomposition levels have not been tested, as the computational load would increase to unacceptable levels. See section 3.8. In addition, as the non-decimated wavelet transform has been used, signal values are added automatically at the edges, to retain the same number of samples at higher decomposition levels (the signal is extended). Because of the limited amount of samples, it is possible that the higher classification rates for the higher decomposition levels in the table are a consenquence of the extension of the signal. The default extension mode of the `ndwt()` Matlab command is the symmetric extension mode.
In all further tests, three wavelet subbands were used to extract information about the frequency content of the signals.

3.2.3 Number of nearest neigbours

The number of nearest neighbours in the k-NN classifier was chosen such as to obtain high classification rates. The test was performed using subset SB in table 1.3. The subset consists of 442 disruptive shots between campaigns C15 and C20. Three wavelet subbands were taken into account. The wavelet statistics were described by a GGD with a fixed $\beta = 1$. The classifier was trained with features of 65% of the shots in SB. The shotnumbers were chosen randomly. Features of all other shots were used to create the test set. The test was repeated twenty times and in each iteration different training and test sets were constructed.
Ten k-NN classifiers were developed. Each classifier worked with a different number of nearest

neighbours, $k = 1, \ldots, 10$. The similarity measure was the Rao geodesic distance of (2.38). The results are summarised in table 3.2.

Table 3.2: Success rates for k-NN classifiers with a different number of nearest neighbours.

k	SR
1	86.7 ± 2.6
2	90.8 ± 2.5
3	87.9 ± 2.7
4	89.2 ± 3.5
5	87.8 ± 2.7
6	88.4 ± 3.4
7	87.8 ± 4.0
8	88.1 ± 3.4
9	89.7 ± 3.1
10	86.9 ± 2.9

The variation of the SR for the different classifiers is limited. It was chosen to use $k = 1$ for simplicity.

3.2.4 Scale parameter of the RBF kernel

Rattá et al. have presented a disruption predictor for JET using a SVM classifier with a RBF kernel [1]. Statistics of Fourier coefficients were used. In our tests, a similar classifier was used to compare its performance with the classifiers based on wavelet statistics. The SVM classifier with an RBF kernel is a parametric classifier. The scale parameter σ in (2.24) has to be chosen a priori. To this end, 24 SVM classifiers were developed. The range of the scale parameter varied between $0.25 \leq \sigma \leq 6$ in steps of $\Delta\sigma = 0.25$. Events of 65% of the shots in data set SB were used to train each classifier. Each classifier was trained with the same training set. Events of the other 35% of the shots were then used to test the classification performance. The test was performed only once, due to the long required training time.
Figure 3.2 shows the SR in function of the scale parameter. The SR increases with higher σ values and reaches more or less a saturation for $\sigma \geq 5.25$. The scale parameter in the tests was chosen to be $\sigma = 6$.

3.3 Dimensionality reduction

Dimensionality reduction allows to visualise data if the lower-dimensional, Euclidean embedding is either two- or three-dimensional. The use of MDS as visualisation tool has the additional advantage that the embedding is approximately isometric. As distances are conserved, the visualisation of the data represents well the dissimilarity matrix which is used by the k-NN classifier. The embedding allows to deduce some qualitative insights about the classification process.
1251 disruptive and 1251 regular features of 330 different shots were extracted from data set SB in table 1.3. Both wavelet and Fourier features, for the same time windows, were collected in

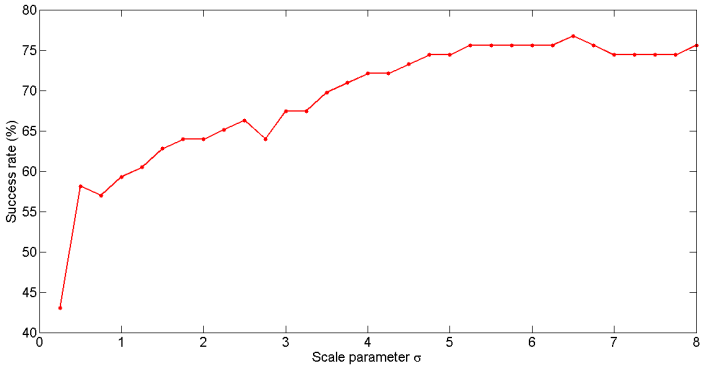

Figure 3.2: SR for SVM classifiers with different σ for the RBF kernel.

two different data sets. A dissimilarity matrix was computed for the wavelet set using the Rao geodesic distance between Laplace distributions. Two dissimilarity matrices were computed for the Fourier features. For the first dissimilarity matrix, the Euclidean distance was computed between standard deviations of the Fourier spectra of each signal. The geometry of the Gaussian distributions was thus neglected. Each feature was assumed to be independent from the other features. The total Euclidean distance is thus the square root of the sum of the squares of the individual distances. For the second dissimilarity matrix, the geometric structure of the space of zero-mean Gaussian distributions was taken into account by using the Rao geodesic distance. Metric MDS was performed on the three dissimilarity matrices for embedding spaces of dimensions 1 to 5. The normalised raw stress was used in the MDS algorithm. The residual stress as function of dimension is shown in figure 3.3. Using the elbow criterion, the embedding dimension was chosen to be three. Note that an embedding in a one-dimensional space should be treated with care and thus the elbow appearing for the two-dimensional embeddings is to be neglected (the elbow at dimension two appears because of the high value of residual stress for the one-dimensional embeddings).

The three embeddings are shown in figure 3.4. The embedding for the Fourier features in the case of the Euclidean distance measure indicates that the Euclidean distance does not make a clear distinction between regular and disruptive events. The majority of the points lie in a close neighbourhood around the origin. As both classes heavily overlap in this region, it is expected that a k-NN classifier is not able to classify points correctly. The embedding of the Fourier dissimilarity matrix constructed with the Rao geodesic distance already indicates that the similarity measure distinguishes regular and disruptive events in a much better way. There are essentially three clusters of data, indicated in the figure by black boxes. Each cluster is clearly split in a subcluster of disruptive events and a subcluster of regular events. This partially explains the higher classification rates in section 3.4, obtained by a k-NN classifier using Fourier statistics with the Rao geodesic distance as similarity measure.

At first sight the embedding of the dissimilarity matrix of wavelet features appears to be very similar to the embedding of Fourier features using the Rao geodesic distance. However, the splitting of each of the three clusters into a regular and a disruptive subcluster is more pronounced in the case of wavelet statistics. This is especially clear for the biggest cluster, for which it is

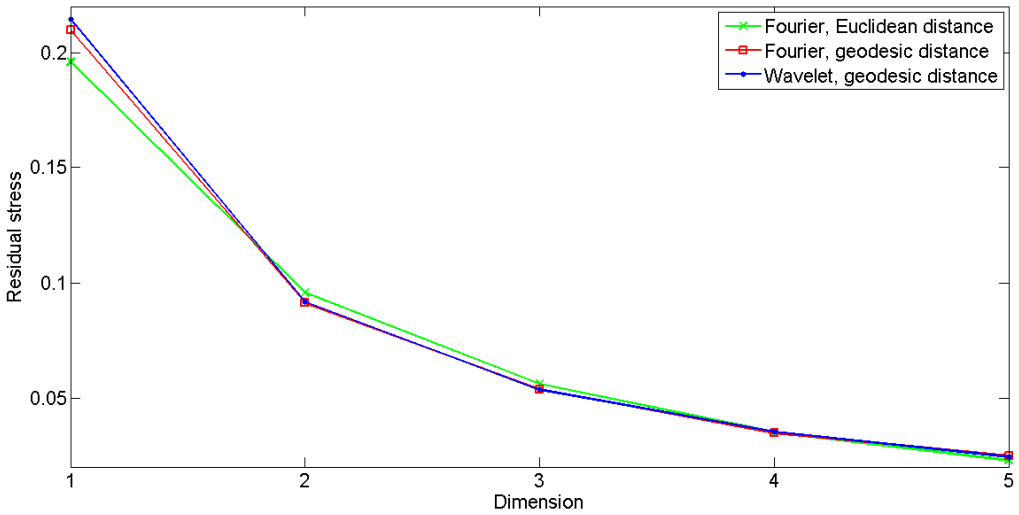

Figure 3.3: Residual variance in function of dimension.

almost possible to draw a separation plane between disruptive and regular points as is shown in figure 3.5. This is not the case for the Fourier statistics. This already indicates that the use of wavelet detail statistics and the exploitation of the geometry of GGDs is able to distinguish regular and disruptive features in a better way than Fourier statistics.

In both cases where the Rao geodesic distance was used, the data is ordered in three clusters (neglecting a minor number of outliers). The clusters do not correspond to different JET campaigns, neither do they correspond to shots with a different disruption cause. The underlying reason for the clustering remains unclear. The close resemblance of the structure of the embedding of the Fourier features and the embedding of the wavelet features could indicate that the clustering is related to the geometry of the manifolds of GGD distributions with a fixed β value. Indeed, the distance between two Gaussian or two Laplace distributions differs only by a constant ((2.38) in section 2.4.3).

It is possible to test statistically the validity of the clustering structure in the original, high-dimensional space, using the Hubert's Gamma statistic. The null hypothesis is the random label hypothesis as explained in section 2.2.2. The distribution of the Hubert's Gamma statistic under the random label hypothesis depends on the configuration of points. The distribution has been estimated from 1000 Monte-Carlo simulations for both the Fourier and wavelet data. A histogram of the resulting distributions is shown in figure 3.6. The histograms are normalised so that the areas beneath the bars sum up to one.

The resulting distributions are quite similar. Note the very low values of $\hat{\Gamma}$ under the random label hypothesis. This is consistent, as the correlation between the dissimilarity matrix for the clustered data and a random partition is low. The distribution is moreover almost centered at zero, as random partitions have on average the same probability to slightly anticorrelate with the data as to correlate with it. The highest correlation values under the random label hypothesis for the Fourier and wavelet data set are respectively $2.77 \cdot 10^{-3}$ and $3.44 \cdot 10^{-3}$. These values should be compared to the $\hat{\Gamma}$ between the proximity matrices of the data sets and the

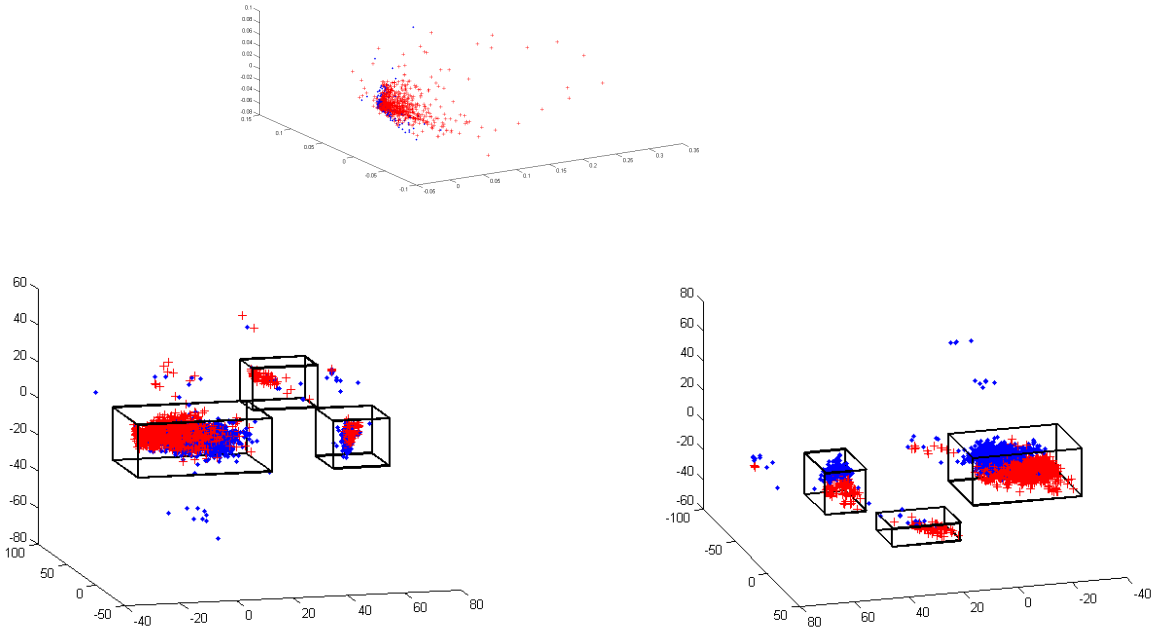

Figure 3.4: MDS embedding of processed JET data. Top: Fourier statistics, Euclidean distance. Bottom left: Fourier statistics, Rao geodesic distance. Bottom right: Wavelet statistics, Rao geodesic distance. The blue dots stand for regular events and red crosses represent disruptive events.

corresponding partitions indicated in figure 3.4 by black boxes. Therefore, a fourth partition was introduced which included all outliers. The $\hat{\Gamma}$ for the Fourier and wavelet data are respectively 0.896 and 0.928; i.e. there is almost a perfect correlation between the proximity matrix and the proposed partition in both cases. The null hypothesis is thus rejected in both cases and for both Fourier and wavelet data the partition represents the true underlying structure on the probabilistic manifold.

3.4 Classification results

Six different classifiers were developed for disruption prediction. The classifiers are:

- *Fou-Eucl*: k-NN classifier based on Fourier statistics. Each feature was seen as a independent variable in Euclidean space. The Euclidean distance was used as similarity measure. $k = 1$.
- *Fou-GD*: k-NN classifier based on Fourier statistics. The geometry of the Gaussian distributions fitted to the Fourier spectra was taken into account by the use of the Rao geodesic distance. $k = 1$.
- *Wav*: k-NN classifier based on wavelet detail statistics. The statistics were modeled by a Laplace distribution ($\beta = 1$). The geometry of the distributions was taken into account by the use of the Rao geodesic distance. $k = 1$.

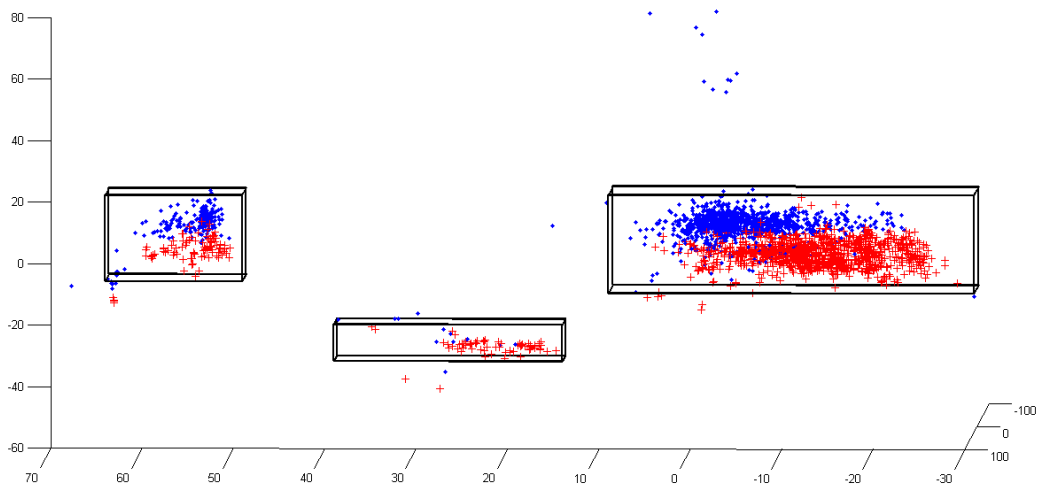

Figure 3.5: Clear separation between regular and disruptive subclusters in the right cluster.

- *Maha*: Mahalanobis classifier based on wavelet detail statistics. The statistics were modeled by a Laplace distribution ($\beta = 1$). The geometry of the distributions was taken into account by projection of the test data to the tangent plane at the geodesic center-of-mass of each cluster.
- *SVM*: SVM classifier with an RBF kernel. $\sigma = 6$.
- *J-div*: k-NN classifier based on wavelet detail statistics. The statistics were modeled by a GGD with variable $0 < \beta \leq 5$. The probabilistic nature of the distributions was taken into account by the use of the J-divergence. $k = 1$.

65% of the shots in data set SB of table 1.3 were chosen randomly and were used as training data. To overcome a preferential classification for the regular regime by k-NN classifiers, due to the high ratio of number of regular to disruptive features, the regular features were limited for each shot to the same number as the disruptive features of this shot. These regular features were chosen as close as possible to 1 s before ToD. The other 35% of the shots were used to create the test set in a similar way.
The six classifiers were tested using the same time windows for both training and test set. The classification was iterated twenty times with different training and test sets. The results are summarised in table 3.3.
Several conclusions can be drawn from this table:

- The SR of Fou-Eucl and the SR of Fou-GD are similar. The reason is that, although Fou-GD has a higher TPR than Fou-Eucl, it has almost the same FPR. This illustrates that the correct detection of a regular shot is a stricter demand for a real-time classifier than the correct detection of a disruptive event. To work succesfully, the classifier should classify *all* regular events correctly (to avoid a false alarm), and at least recognise *one* disruptive event (to avoid a missed alarm). For the Fou-Eucl and Fou-GD, the incorrect

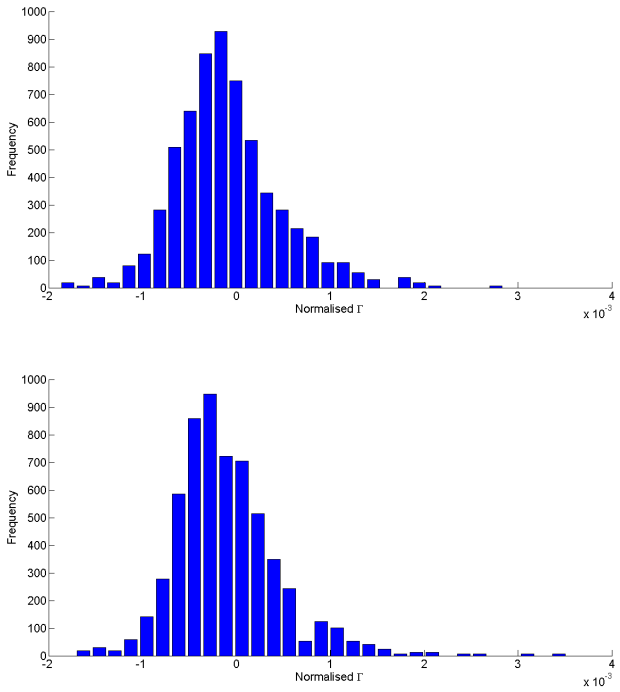

Figure 3.6: Estimated distibution of $\hat{\Gamma}$ for the Fourier and wavelet data under the random label hypothesis.

Table 3.3: Classification performance.

	Fou-Eucl	Fou-GD	Wav	Maha	SVM	J-div
TPR	74.5 ± 3.2	82.8 ± 2.5	94.9 ± 1.9	97.7 ± 1.8	59.6 ± 3.7	93.5 ± 1.6
FPR	25.0 ± 3.5	23.9 ± 3.8	5.8 ± 2.5	5.2 ± 1.9	8.8 ± 2.2	7.5 ± 2.3
MA	0.8 ± 1.0	0.7 ± 0.8	0.3 ± 0.5	0.3 ± 0.5	9.3 ± 2.8	0.2 ± 0.5
FA	65.2 ± 6.3	63.4 ± 5.8	11.3 ± 4.1	8.9 ± 2.4	19.2 ± 3.7	15.2 ± 3.8
TE	66.1 ± 6.1	64.0 ± 5.6	11.6 ± 4.0	9.2 ± 2.3	28.5 ± 4.1	15.5 ± 3.8
SR	33.9 ± 6.1	36.0 ± 5.6	88.4 ± 4.0	90.8 ± 2.3	71.5 ± 4.1	84.5 ± 3.8
AVG	165.4 ± 9.5	173.5 ± 6.5	184.7 ± 3.1	186.9 ± 2.7	137.7 ± 6.4	186.0 ± 2.8

classification of less than one on four regular events leads to false alarms in almost two thirds of the shots.

- The use of wavelet statistics leads to a great improvement of the classification rates. Wav exhibits an increase of 12% of TPR and a decline of almost 18% of FPR compared to Fou-GD, which leads to an improvement of the SR up 50%. Especially the decline of the FPR leads to a decline of FA from 63% for Fou-GD to 11% for Wav. The improved separation of regular and disruptive events was already shown by the MDS embedding in section 3.3.

- The SVM outperforms classifiers Fou-Eucl and Fou-GD in success rate. This is mainly because of the lower FPR in the case of SVM. However, the SVM predictor classifies wrongly 40% of the disruptive events. The SVM is therefore the only classifier with a considerable share of missed alarms: in 9% of the shots, the upcoming disruption was not

recognised. For all other classifiers this share does not exceed the 1%.

- The SVM detects the upcoming disruption much later than the other classifiers. The average time of detection is 138 ms before the ToD. This is in accordance with results found in literature, which report an average prediction time of 145 ms before disruption [3]. For the Wav classifier, this time is about 185 ms before ToD. The average prediction time before disruption needs to be high in order to be able to undertake mitigating actions.

- All classifiers based on wavelet statistics have an AVG close to the maximum possible AVG. The maximum possible AVG in the test is 195 ms, in which case all disruptions are predicted in the first disruptive event. 195 ms is the mean time in the first disruptive time window, $[\text{ToD} - 210\,\text{ms}; \text{ToD} - 180\,\text{ms}[$. This indicates that the classifiers based on wavelet features could, in principle, detect a disruption still earlier in the shot. This is in accordance with the evolution of the shape parameter in section 3.2.1. Classifiers based on wavelet statistics could possibly reliably detect upcoming disruptions up to 360 ms before ToD.

- The Mahalanobis classifier is the best disruption predictor in this test. As the data is projected to the appropriate tangent spaces, the geometry of the configuration is only partly taken into account. One could expect that this would lower the results. The higher classification rates in comparison to the Wav classifier could be explained by the fact that the Mahalanobis classifier is less sensitive to noise than a k-NN classifier. As was shown in figure 3.4, clusters appeared in the MDS embedding. This clustering structure was further validated in the original feature space by the statistical test of section 3.3. This means that the data of wavelet statistics is grouped into clusters, which are further split in two subclusters of regular events and of disruptive events. Especially at the boundary of the subclusters, the k-NN classifier will tend to misclassify events. The Mahalanobis classifier does not exhibit this deficiency, as only collective information about the training set is considered. Individual training points do not affect the classification.

- It does not prove beneficial to work with GGD distributions of variable shape parameter. This becomes clear in the classification rates of the J-div classifier, in comparison to the rates of the other two classifiers based on wavelet features.

3.5 Generalisation capability

It is interesting to test the performance of the classifiers for more recent campaigns. The training set consisted of events of 99% of the shots from campaigns C15 to C20 (data set SB in table 1.3). Seven test sets were extracted from data set SC. Events from 99% of the shots of each campaign between C21 and C27 were used. For each shot, an equal number of regular and disruptive events were chosen in the same way as in section 3.4.
The TPR and FPR are shown in figure 3.7. All classifiers show similar values for TPR and FPR as in the previous classification test. The Maha classifier has a TPR of 100% for campaigns C21, C23 and C25. Thus, the Maha classifier did not miss any disruption for the shots of those campaigns (but can still generate a false alarm earlier in the shot).

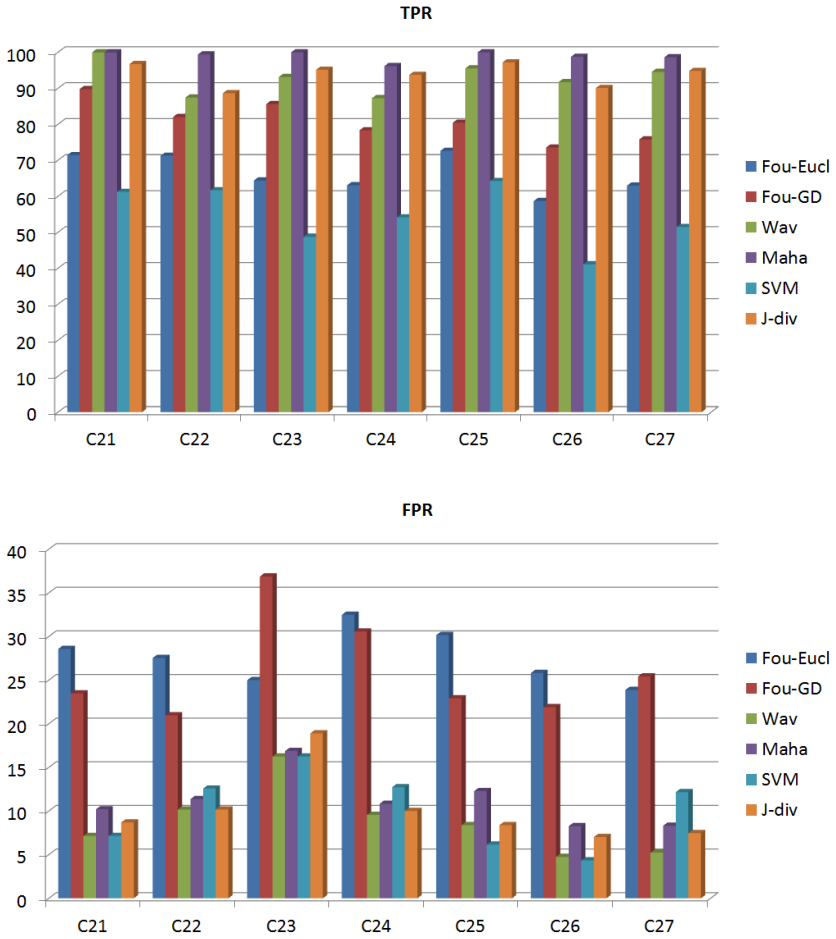

Figure 3.7: TPR and FPR results of the generalisation test (in percent).

Figure 3.8 shows the SR obtained for campaigns C21-C27. The SR has decreased for all classifiers as compared to the test of section 3.4. The decrease of the SR is expected, as test and training sets are not constructed with events of the same series of experiments. The classifier is not able to deal with the changes in the data, as the experimental conditions have changed. For example, the ITER-like ICRH antenna was commisioned on JET plasmas in May 2008 [56], which almost coincides with the beginning of campaign C21 (at the start of June 2008). However, the performance of the classifiers relative to each other remains almost the same. The Wav classifier outperforms slightly the Maha classifier, which was not the case in the test of section 3.4.

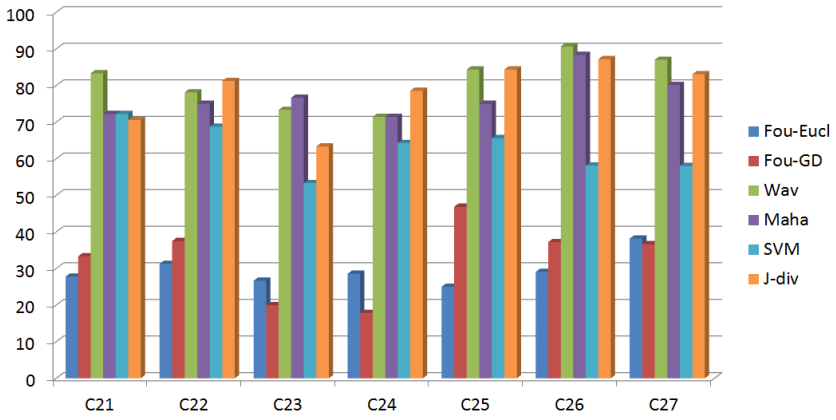

Figure 3.8: SR results of the generalisation test (in percent).

3.6 Prediction of disruption cause

All classifiers are able to distinguish regular from disruptive behaviour up to a certain extent, as was proven in sections 3.4 and 3.5. A restriction to these two classes will be insufficient in real-time applications [21]. The prediction system should be able to detect the disruption cause or class in order to couple the system to the response and mitigate the adverse effects of the disruption. For example, most of the disruptions could be avoided by a fast-emergency shut-down using high-pressure gas injection of low-Z noble gasses [57]. For some disruption causes with more damaging consequences, such as ITBs and VDEs, faster measures could, however, be necessary [21].

Following test concerned the classification of disruptions according to disruption class. As disruptions are caused by a multitude of reasons, this is a multiple-class recognition problem. Only the k-NN and Mahalanobis classifiers are considered, as a straightforward generalisation to a multiple-class problem is possible. The k-NN method of section 2.3.2 and the Mahalanobis classifier of 2.5 were presented directly in terms of a multiple-class problem. Although SVMs are used in practice in multiple-class recognition [32], the generalisation is not trivial.

The results of an extensive study which determined the cause of all disruptions in a decade of JET operations (2000-2010) was made available to this end. The results include sets of shots for eight different disruption causes. The causes are summarised in table 3.4. The third column in this table gives the number of shots for each disruption cause in campaigns C15-C27.

The training set consisted of features of 65% of the shots between campaign C15 and C27. The shot numbers were chosen randomly. Features of the other 35% were used to construct the test set. All classifiers, apart from the SVM classifier, were used. For each shot, the same number of regular time windows and disruptive windows were added to the corresponding data set. Again the regular time windows closest to 1 s before ToD were used. The test was iterated twenty times with different training and test sets.

As this test is a multiple-class problem, the number of nearest neighbours for the k-NN classifiers was re-investigated. Ten k-NN classifiers were developed with the Rao geodesic distance as similarity measure and $k = 1, \ldots, 10$. Wavelet features were used to determine the ideal number

Table 3.4: Disruption causes.

Disruption cause	Abbreviation	Disruptive events
Auxiliary power shut-down (H-L transition)	ASD	67
Too strong internal transport barrier	ITB	14
Too fast current ramp-up	IP	13
Impurity control problem	IMC	111
Too low density and low q	LON	39
Neo-classical tearing mode	NTM	38
Density control problem	NC	144
Greenwald density limit	GWL	8

of neighbours. The results are shown in table 3.5. The maximum SR is found for $k = 6$. This number of neighbours was used in following tests.

Table 3.5: Success rates for k-NN classifiers with $k = 1 \dots 10$ in the multiple-class problem.

k	SR
1	34.6 ± 5.1
2	37.0 ± 4.9
3	35.8 ± 5.4
4	37.7 ± 4.5
5	38.0 ± 4.2
6	39.5 ± 4.8
7	38.8 ± 4.6
8	39.5 ± 3.9
9	39.4 ± 3.9
10	39.4 ± 4.0

The results are summarised in figures 3.9 and 3.10, together with table 3.6. TPR in figure 3.9 now represents the share of time windows of a certain class that were classified correctly. FPR in figure 3.10 represents the share of time windows of all other classes that were classified incorrectly to this class. A successful classification is now obtained if the classifier does not produce any false alarms, does not miss an upcoming disruption *and* predicts correctly the disruption cause close to disruption time. Only the first warning of the system was used for this purpose, i.e. it is assumed that the prediction system would directly undertake mitigating actions. The fraction of shots for which the system predicted correctly an upcoming disruption, but failed to indicate the correct cause, is called the *wrong alarm rate* (WA).

From figure 3.9 the conclusion can be drawn that four disruption causes are recognised better than the others. The four causes are ASD, ITB, LON and GWL. Figure 3.10 shows that wrong classifications are spread between the same four classes. The main reason why those four classes get a preferential treatment is because the appropriate signals for their detection are available. The main diagnostic to identify an ASD disruption is the input power. ITBs are characterised by an increased output power and thus by a sharp increase in the net output power. Finally,

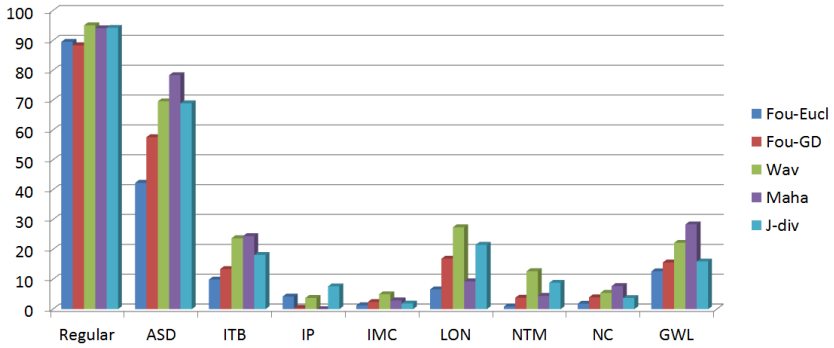

Figure 3.9: TPR per disruption cause.

LON and GWL can be detected in the tendencies of the plasma density. IMC disruptions are detected only indirectly through of the increase of the radiated power and thus a decrease of the net power. They can therefore not be distinguished from other sources of radiation. Despite the large amount of shots disrupted by an IMC, almost no data is wrongly classified as an IMC. The disruptive events of IMCs are thus highly distributed in the probabilistic space. IMC disruptions which are classified incorrectly lead to a high numbers of WA in table 3.6. IP problems are highly transient and are not well detected by the predictors. Another reason is the limited number of shots, which limits the learning capability of the classifiers. The classification rates of NTMs and NCs are more difficult to interpret. Most probably NC problems are classified either as LON or GWL, as the only diagnostic for identification of NC problems is the plasma density.

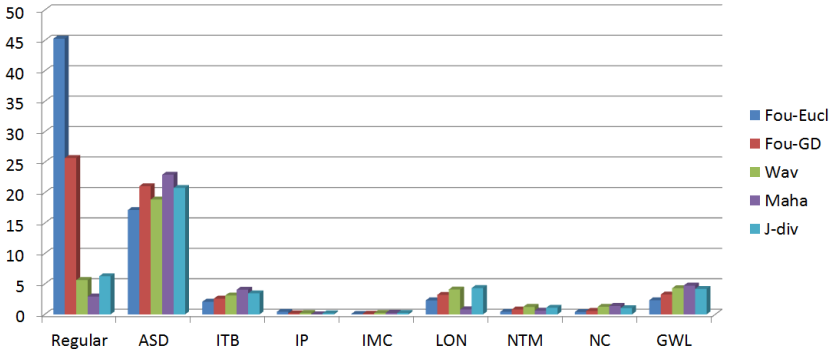

Figure 3.10: FPR per disruption cause.

Comparing the different classifiers in table 3.6, Fourier features seem to lead to less WA. It should be noted, however, that classifiers based on Fourier statistics have a FA rate of over 30%, while all classifiers based on wavelet statistics have a FA rate of under 10%. Shots for which a false alarm has been triggered cannot account anymore for a wrong prediction of the disruption cause. It is possible that a large share of the shots for which classifiers based on Fourier statistics already produced a false alarm, would also trigger a wrong alarm in the disruptive phase. Note also that although the success rate of the three classifiers based on wavelet statistics has decreased sharply, the ability of the classifiers to detect disruptive behaviour has not. However, for only one out of two recognised disruptions, the classifier is also able to predict its cause.

The overall success rates are rather limited. Classifiers based on Fourier statistics produce only in one out of four shots a correct alarm, while classifiers based on wavelet features raise this rate above one out of three shots. A classifier used in practice will need to substantially improve these figures. The tests shown here are only meant as preliminary tests. Sufficient information about each disruption cause is necessary. The precursor signals should be chosen carefully based on the disruption causes one wants to predict. A close collaboration with experts in the field could already improve substantially the prediction ability of the classifiers. Finally, it should be noted that when classifiers are built to detect several disruption causes, the test set should also contain data of shots with other disruption causes than that for which the classifier was trained. Far too often one finds in literature high classification rates, but the test sets always contain shots with only the same disruption cause as the classifier was trained for. Although current classification rates are rather low, they are more realistic, in the sense that a real disruption predictor will often encounter discharges which disrupt by another cause than the classifier was trained for.

Table 3.6: Equivalent real-time classification results for disruption cause prediction.

	Fou-Eucl	Fou-GD	Wav	Maha	J-div
MA	6.7 ± 1.4	1.1 ± 0.8	0.4 ± 0.3	0.8 ± 0.7	0.5 ± 0.5
FA	32.6 ± 2.9	32.5 ± 2.6	7.2 ± 1.1	8.5 ± 2.1	10.1 ± 1.9
WA	35.9 ± 3.1	35.9 ± 2.1	49.8 ± 2.9	48.0 ± 3.0	50.5 ± 2.1
TE	75.3 ± 2.1	69.6 ± 3.3	57.4 ± 3.3	57.3 ± 2.9	61.1 ± 3.0
SR	24.7 ± 2.1	30.4 ± 3.3	42.6 ± 3.3	42.7 ± 2.9	38.9 ± 3.0
AVG	153.5 ± 6.2	162.7 ± 4.7	184.0 ± 3.7	185.1 ± 3.1	181.4 ± 3.7

3.7 Signal relevance

The experiment of section 3.6 raises the question whether the chosen set of signals is the appropriate set for the detection of disruptions. For the prediction of disruption causes, this set will most probably have to be adjusted.

The relevance of each signal for the prediction of disruptions was investigated as follows. The test and training sets were constructed in a similar way as in section 3.4, with data from 65% of randomly chosen shots used to construct the training set. The test consisted of thirteen different stages. In the first stage, thirteen different training and test sets were used for classification. Each set consisted only of features of one precursor signal. Each signal was used once. Once the SR for all thirteen sets was acquired, the signal which produces the highest SR was memorised by the system. In the second stage, twelve different training and test sets were used. Here, each set consisted of the features derived from the memorised signal of the first stage, together with features of each of the remaining signals. In each consecutive stage, the most relevant signal for the SR was memorised. In the following stage the already memorised signals were combined with one additional signal of the remaining set. In this way, an estimation was made of the relevance of the signals in the classification.

The ideal way to proceed in order to find the optimal set of signals, would be to check all possible

combinations of all possible signals. There are $2^{13} - 1$ such combinations and it is thus obvious that iteration over all possibilities is computationally impossible. The presented method allows to assess the relevance of the signals by classification of $13 \times 14/2 = 91$ different data sets.
For each different combination of signals, the test was iterated five times. Only the mean value of the SR is given.
The ranking of the relevance of the signals derived from this test is given in table 3.7. The second column gives the name of corresponding signal. The third column gives the SR in the classification of the data set in the corresponding stage, when the signal of the second column is added.

Table 3.7: Relevance of predictor signals.

Ranking	Signal name	SR
(1)	Mode lock amplitude	92.9
(2)	Plasma current	91.8
(3)	Poloidal beta time derivative	92.3
(4)	Total input power	92.4
(5)	Plasma density	93.5
(6)	Safety factor at 95% of minor radius	92.9
(7)	Net power	92.6
(8)	Poloidal beta	91.8
(9)	Plasma vertical centroid position	91.9
(10)	Stored diamagnetic energy time derivative	91.0
(11)	Plasma internal inductance	90.4
(12)	Plasma internal inductance time derivative	93.4
(13)	Safety factor at 95% of minor radius time derivative	89.5

The variation of the SR due to the different training and test sets makes the interpretation of the results difficult. The relatively small differences between consecutive stages could, in some cases, be the result of the variation in the SR. Nevertheless, the ranking is up to a certain degree as expected. The two most relevant signals, the mode lock amplitude and the plasma current, are also the two signals for which the evolution of the shape parameter was the most pronounced (see section 3.2.1). As explained before, the evolution of the shape parameter is probably the result of the higher time resolution for these two signals. It is thus possible that the performance of the classifiers is limited by the time resolution of the signals. The first five signals appear to be the most relevant for disruption prediction. Adding more signals only decreases the classification rate. The high SR in the twelfth stage is probably the result of variations on the SR. Possibly the set of signals should be revised in future work.

3.8 Computational cost

Two important issues for a realistic disruption predictor are the memory requirement and the computational time. No issues concerning the memory requirements were encountered during the tests presented in sections 3.2 to 3.5. The computational time is treated in closer detail.

Computational time is needed for the feature extraction, for the training of the machine and lastly, for classification. The computational time in the training phase is of secondary importance, as this can be done off-line. On the contrary, the time needed for feature extraction and classification should not exceed the time interval of one time window (here 30 ms). If this were the case, the disruption machine would not be able to start processing data of a new time window once the data is acquired, leading to unacceptable delays.
In present tests, the feature extraction proceeded as follows

1. Load data.
2. Control for errors and process corrupt data.
3. Resampling.
4. Normalisation.
5. Fourier and wavelet decomposition.
6. Fit of appropriate distribution to both spectra.
7. Control for errors.
8. Add to dataset if no errors have occurred.

The control for errors in the last but one step is required, as the maximum likelihood method discussed in 2.1.3 does not perform well for high-β distributions [58] and diverges for some time windows.
The extraction of the data set used in the test of section 3.2.2 lasted 23 hours and 58 minutes. 392892 regular and 3098 disruptive time windows were analysed during this time. This results in a mean time for feature extraction of 218 ms per time window. In this data set, the wavelet decomposition included five wavelet subbands. As a comparison, the construction of a similar data set with three wavelet subbands lasted 148 ms per time window. Comparing the times at different decomposition levels, it is clear that the wavelet decomposition step and the GGD fit require a large fraction of the total time of feature extraction.
This amount of computational time is unacceptable in a realistic application. However, it should be noted first that a regular personal computer was used (processor Intel Core i3-2310M 2.1 GHz, 1×4 GB DDR3 RAM at 1.333 MHz). The use of better hardware would certainly accelerate the process. Second, once the resampling is done, which is to be done for all signals on the same time base, all other steps can be performed in parallel for each signal. The computational times above include the wavelet and fourier decomposition of all thirteen signals described in section 1.3. Parallel feature extraction could thus reduce the computational time by a factor of thirteen.
Computational time in the training phase is only relevant for the SVM classifier and the Mahalanobis classifier. As an example, the training for the SVM classifier of the test in section 3.5 lasted 41 minutes. On the other hand, the training time of the Mahalanobis classifier was 117 ms. The training phase of the SVM is computationally expensive because of the iterative solution scheme to construct the decision function. The long training time was already mentioned in section 2.3.2 as a general property of SVM machines. The training time of the Mahalanobis classifier is considerably shorter. Although the geodesic centre-of-mass of a cluster is found iteratively, the iteration process converges relatively fast.

The computational time in the classification step is also important. In table 3.8, the average computational times are summarised for the tests of sections 3.4 and 3.5. These are the times needed for the classification of one time window of the test set.
There is a large difference in classification time for k-NN classifiers, depending on the similarity measure. The Wav classifier demands approximately four times more time than Fou-GD to classify the data. This is mainly a consequence of the fact that there are three times more independent features for which the Rao geodesic distance is computed, originating in the wavelet decomposition level $N_S = 3$. Furthermore, the Euclidean distance is less computationally expensive than the Rao geodesic distance. This is apparent in the difference of computational time between the Fou-Eucl and Fou-GD classifiers. The J-div classifier needs exceptionally more time than the other k-NN classifiers. The classification time of the J-div classifier is even too high to use it as a real-time disruption predictor. Note finally the difference in computational time for the four k-NN classifiers between tests of section 3.4 and section 3.5. This is due to the fact that the computational time for k-NN classifiers also depends on the size of the training set, as the distance between all training and test points is computed.
The Mahalanobis and SVM classifiers are less computationally expensive in the test phase. For the SVM classifier, this was mentioned in section 2.3.2. This is a general property of non instance-based algorithms. The decision function has been constructed beforehand and needs less computational time in the test phase. The Mahalanobis is also a non instance-based algorithm, with a low classification time as a result.

Table 3.8: Classification time.

	Time class (ms), test section 3.4	Time class (ms), test section 3.5
Fou-Eucl	0.40	0.85
Fou-GD	2.45	4.20
Wav	9.77	15.88
Maha	1.23	1.28
SVM	0.09	0.18
J-div	93.69	172.83

Chapter 4

Conlusions and future work

The main goal of this study was to prove the potential of the use of wavelet statistics for disruption prediction. The statistics were modelled by a generalised Gaussian distribution. The classifiers thus essentially compared probability distributions. In order to take the probabilistic nature into account, concepts of information geometry were considered, where a family of probability distributions is regarded as a Riemannian manifold. This leads in a natural way to the use of the Rao geodesic distance as a similarity measure between distributions.
The classifiers were tested with data of disruptions which occurred at JET in the period 2000-2010. Classifiers using the geometry of wavelet statistics prove to work well as compared with classifiers based on the statistics of Fourier coefficients, which have already been used in the past. In a classification test with data of JET campaigns C15-C20, the classifiers using the geometry of wavelet statistics predicted successfully a disruption in more than 85% of the shots. In comparison, the most advanced classifier based on Fourier statistics, a SVM classifier with a RBF kernel, predicted the upcoming disruption in only 71.5% of the shots. The generalisation capability of the classifiers was also higher: in a classification test with a training set of JET campaigns C15-C20 and test sets of campaigns C21 to C27, the successful prediction rates varied between the 65% and 90%. The SVM classifier based on Fourier features performed worse, with success rates between the 55% and 70%. Finally, the classifiers using the geometry of wavelet statistics performed best in the recognition of the disruption cause for JET campaigns C15-C27.
An important result of the tests was the early warning time produced by classifiers based on wavelet statistics. An early warning time is essential in order to undertake mitigating actions well enough in advance. Current tests were, however, not simulated in real-time environment due to considerations about the computational time. Real-time simulations should be performed in future work.
Some additional tests were run. First, the time evolution of the essential parameter of the generalised Gaussian distribution, the shape parameter β, was investigated. The value of the shape parameter shows whether the transient behaviour in the time traces of the plasma diagnostics is also present in the modelling of the wavelet statistics. It became clear from this test that the time resolution of the precursor signals plays an essential role in the ability of the machine to detect the disruption. β showed an abrupt change only for the plasma current and the mode lock amplitude at about 360 ms before disruption. These two signals were the ones with the highest time resolution in the set of the thirteen plasma signals. The time of 360 ms is much higher

than the already reported maximum time before disruption at which classifiers based on Fourier statistics can reliably detect a disruption, i.e. 180 ms. A disruption predictor based on wavelet statistics could thus detect disruptions earlier in the shot as compared to other automatic predictors. Second, a test was performed to investigate the relevance of each of the thirteen plasma signals. It turns out that, again, the plasma current and the mode lock amplitude, the signals with the highest time resolution, were the most relevant signals. This is a second indication that the resolution of the diagnostics is essential for the detection of disruptions. Also, the use of too large numbers of plasma signals seems to lower the classification performance. It is thus interesting to consider the possibility of working with limited data sets in the future. In a last test, the required classification time was examined. The data processing and classification should not exceed the length of a time window (in this study 30 ms). Feature extraction times of 150 ms to 220 ms are thus obviously too high for real-time applications. However, solutions could be found to this issue. As in this test the processing of the thirteen plasma signals was performed sequentially, parallel processing would already lower the required time by an order of magnitude. Such a reduction would prove sufficient. In addition, almost all classifiers which used the geometry of wavelet statistics were able to classify the data sufficiently fast. One exception is the classifier based on the generalised Gaussian modelling of the wavelet statistics with a variable β. As this classifier did not perform exceptionally well in other tests, this should not be considered an issue.

The general conclusion is that the use of the geometry of wavelet statistics can add value to the prediction of plasma disruptions. In future work, the classifiers which were presented should, in the first place, be tested in a real-time environment. This should be accompanied by a thoughtful choice of the precursor signals, in collaboration with experts in the field. For this choice, special attention should be given to the time resolution of the signals and the number of signals. In addition, the detection of the disruption cause will be indispensable in the future. It is therefore essential to focus on automatic prediction of the disruption cause, and not on the detection of the disruption as such. A proposal is to construct several automatic predictors based on the geometry of wavelet statistics, each able to detect a single disruption cause. This allows to use a limited amount of signals. Each cause of disruptions can namely be detected by the careful combination of a couple of plasma diagnostics. The warning system would then consist of a parallel implementation of the classifiers.

Appendix A

Metric for univariate GGD with fixed β

The univariate GGD is given by

$$f(x \mid \alpha, \beta) = \frac{\beta}{2\alpha\Gamma(1/\beta)} \exp\left(-\left(\frac{\mid x \mid}{\alpha}\right)^{\beta}\right) \tag{A.1}$$

For a constant β, the metric (2.30) reduces to $ds^2 = g_{\alpha\alpha} d\alpha^2$, where

$$g_{\alpha\alpha} = -\mathbb{E}\left[\frac{\partial^2}{\partial\alpha^2} \ln f(x \mid \alpha, \beta)\right] \tag{A.2}$$

according to (2.36).
$\ln f$ can be rewriten as

$$\ln f(x \mid \alpha, \beta) = \ln \frac{\beta}{2\Gamma(1/\beta)} + \ln \frac{1}{\alpha} \exp\left(-\left(\frac{\mid x \mid}{\alpha}\right)^{\beta}\right) \tag{A.3}$$

So that $\partial/\partial_\alpha \ln f$ becomes

$$\begin{aligned}
\frac{\partial}{\partial\alpha} \ln f(x \mid \alpha, \beta) &= \frac{1}{\frac{1}{\alpha}\exp\left(-\left(\frac{|x|}{\alpha}\right)^{\beta}\right)} \left[-\frac{1}{\alpha^2}\exp\left(-\left(\frac{\mid x \mid}{\alpha}\right)^{\beta}\right)\right. \\
&\quad \left. + \frac{1}{\alpha} \cdot \left\{\beta \mid x \mid^{\beta} \alpha^{-\beta-1} \exp\left(-\left(\frac{\mid x \mid}{\alpha}\right)^{\beta}\right)\right\}\right] \\
&= -\frac{1}{\alpha} + \frac{\beta}{\alpha}\left(\frac{\mid x \mid}{\alpha}\right)^{\beta} \\
&= \frac{1}{\alpha}\left[\beta\left(\frac{\mid x \mid}{\alpha}\right)^{\beta} - 1\right]
\end{aligned} \tag{A.4}$$

From (A.4) one can calculate $\partial^2/\partial\alpha^2 \ln f$

$$\begin{aligned}
\frac{\partial^2}{\partial\alpha^2} \ln f(x \mid \alpha, \beta) &= -\frac{1}{\alpha^2}\left[\beta\left(\frac{\mid x \mid}{\alpha}\right)^{\beta} - 1\right] + \frac{1}{\alpha}\left[-\beta^2\left(\frac{\mid x \mid}{\alpha}\right)^{\beta}\frac{1}{\alpha}\right] \\
&= \frac{1}{\alpha^2}\left[1 - \beta(\beta+1)\left(\frac{\mid x \mid}{\alpha}\right)^{\beta}\right]
\end{aligned} \tag{A.5}$$

The calculation of $g_{\alpha\alpha}$ proceeds as follows

$$\begin{aligned} g_{\alpha\alpha} &= -\int_{-\infty}^{+\infty} dx f(x \mid \alpha,\beta) \frac{\partial^2}{\partial\alpha^2} \ln f(x \mid \alpha,\beta) \\ &= -\frac{1}{\alpha^2}\left[\int_{-\infty}^{+\infty} dx f(x \mid \alpha,\beta) - \beta(\beta+1)\int_{-\infty}^{+\infty} dx f(x \mid \alpha,\beta)\left(\frac{\mid x \mid}{\alpha}\right)^{\beta}\right] \\ &= -\frac{1}{\alpha^2}\left[1 - \beta(\beta+1)\int_{-\infty}^{+\infty} dx \frac{\beta}{2\alpha\Gamma(1/\beta)} \exp\left(-\left(\frac{\mid x \mid}{\alpha}\right)^{\beta}\right)\left(\frac{\mid x \mid}{\alpha}\right)^{\beta}\right] \\ &= -\frac{1}{\alpha^2}\left[1 - \frac{\beta^2(\beta+1)}{\alpha\Gamma(1/\beta)}\int_{0}^{+\infty} dx \exp\left(-\left(\frac{\mid x \mid}{\alpha}\right)^{\beta}\right)\left(\frac{\mid x \mid}{\alpha}\right)^{\beta}\right] \end{aligned} \tag{A.6}$$

In the third step, use is made of the fact that f is normalised to 1. In the last step, the even symmetry of the integrand is exploited.

The integral in the last line of (A.6) can be further simplified by the following change of variable

$$\begin{aligned} y = \left(\frac{\mid x \mid}{\alpha}\right)^{\beta} &\Leftrightarrow x = \alpha y^{1/\beta} \\ x = 0 &\Rightarrow y = 0 \\ x \to \infty &\Rightarrow y \to \infty \\ dy = \frac{\partial y}{\partial x} dx = \beta\left(\frac{x}{\alpha}\right)^{\beta}\frac{1}{x}dx &= \frac{\beta}{\alpha} y^{1-1/\beta} dx \\ \Rightarrow dx &= \frac{\alpha}{\beta} y^{1/\beta-1} dy \end{aligned} \tag{A.7}$$

Combination of (A.6) and (A.7) delivers

$$\begin{aligned} g_{\alpha\alpha} &= -\frac{1}{\alpha^2}\left[1 - \frac{\beta^2(\beta+1)}{\alpha\Gamma(1/\beta)}\int_{0}^{+\infty} dy \frac{\alpha}{\beta} y^{1/\beta-1}\exp(-y) y\right] \\ &= -\frac{1}{\alpha^2}\left[1 - \frac{\beta(\beta+1)}{\Gamma(1/\beta)}\int_{0}^{+\infty} dy \quad y^{(1/\beta+1)-1}\exp(-y)\right] \\ &= -\frac{1}{\alpha^2}\left[1 - \frac{\beta(\beta+1)}{\Gamma(1/\beta)}\Gamma(1/\beta+1)\right] \end{aligned} \tag{A.8}$$

In addition $\Gamma(1/\beta+1) = 1/\beta \cdot \Gamma(1/\beta)$, so that

$$\begin{aligned} g_{\alpha\alpha} &= -\frac{1}{\alpha^2}\left[1 - \frac{\beta(\beta+1)}{\Gamma(1/\beta)} \cdot 1/\beta \cdot \Gamma(1/\beta)\right] \\ &= -\frac{1}{\alpha^2}\left[1 - \beta - 1\right] \\ &= \frac{\beta}{\alpha^2} \end{aligned} \tag{A.9}$$

The metric thus becomes

$$\begin{aligned} ds^2 &= \frac{\beta}{\alpha^2} d\alpha^2 \\ &= \beta\left(\frac{d\alpha}{\alpha}\right)^2 \\ &= \beta(d\ln\alpha)^2 \end{aligned} \tag{A.10}$$

Taking the square root of both sides of expression (A.10) gives

$$ds = \sqrt{\beta} \mid d\ln\alpha \mid \tag{A.11}$$

which can easily be integrated. The resulting distance measure is given in (2.38).

Appendix B

Exponential and logarithmic map for GGD distributions with fixed β

The goal is to derive an expression for the logarithmic and exponential map for univariate, zero-mean GGD distributions with a fixed shape parameter β. Therefore, a more detailed explanation of the logarithmic and exponential maps is first needed. Consider, as an example, the one-dimensional manifold in figure B.1 which is represented by the red line. On the manifold, the two black points are connected by a geodesic. The logarithmic map at the lower black point will project the upper black point on the tangent space. The tangent space at the lower point is in this example simply the tangent line at that point. The tangent space is shown by a blue line. The logarithmic map of the upper point is the point on the tangent line, in the direction of the tangent vector to the geodesic connecting both points, with a length equal to the geodesic distance between the points. In the multivariate case, the logarithmic map is found in a similar way. The tangent space at a point of the manifold is spanned by the tangent vectors to all geodesics passing through that point. The logarithmic map is found for a certain point on the tangent plane in the direction of the tangent vector to the geodesic, at a distance equal to the geodesic distance. Note that in the figure, there is only one tangent vector and the direction will only depend on the sign of this vector. This will also be the case for univariate, zero-mean GGD distributions with a fixed β and will be used in the derivation.
The logarithmic map is deduced as follows

1. Find the tangent vector to the geodesic at the origin of the geodesic.
2. The logarithmic map is the point on the tangent space in the direction of the tangent vector at a distance from the origin equal to the geodesic distance. The point at which the logarithmic map is taken is always chosen as the origin of the tangent Euclidean space.

To find the tangent vector, an expression of the geodesic between two points on the manifold is required. A geodesic is found from the Euler-Lagrange equations [7]. Consider again the metric in (2.30) which is given by

$$ds^2 = g_{\mu\nu}(\boldsymbol{\theta}) d\boldsymbol{\theta}^\mu d\boldsymbol{\theta}^\nu$$

The Einstein notation is used, which means that for each index appearing as well as upper and lower index, a summation is implicitly assumed over all the index values. The Einstein notation

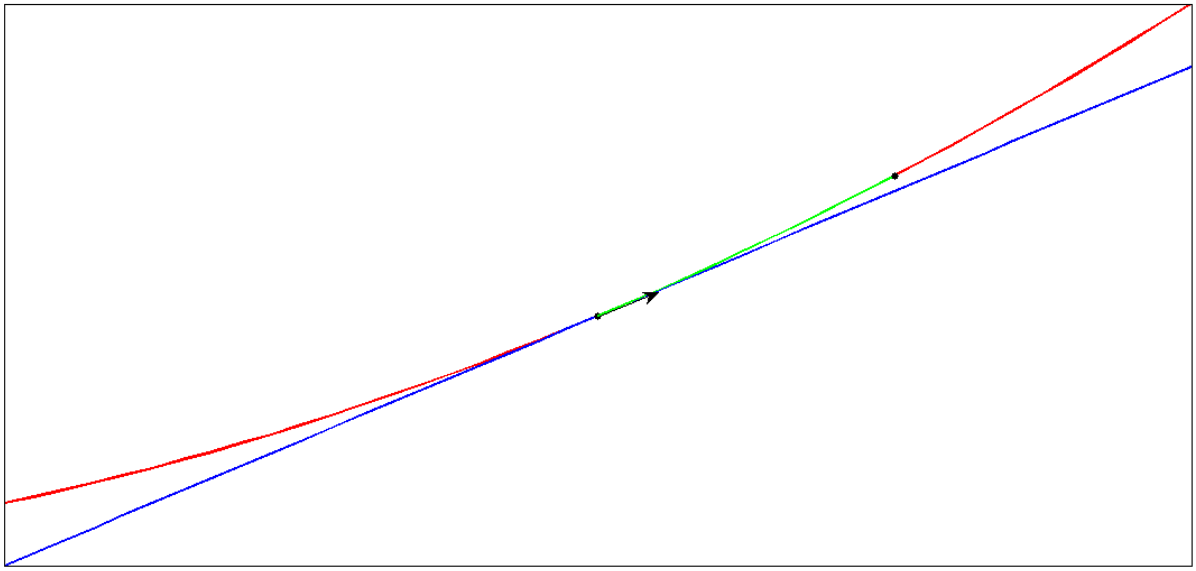

Figure B.1: Manifold in red with the tangent space in the lower black point represented by a blue line. The geodesic path between two points on the manifold is shown in green.

will be used from now on. In the above expression, μ and ν represent all N independent parameters of the metric space. In the case of a probabilistic space, the metric tensor reduces to the Rao-Fisher metric in (2.36) and is given by

$$g_{\mu\nu}(\boldsymbol{\theta}) = -\mathbb{E}\left[\frac{\partial^2}{\partial\theta^\mu \partial\theta^\nu} \log f(\boldsymbol{X} \mid \boldsymbol{\theta})\right]$$

The metric tensor delivers information about the geometry of the space. The Euler-Lagrange equations are

$$\ddot{\theta}^\kappa(t) + \Gamma^\kappa_{\mu\nu}\dot{\theta}^\mu(t)\dot{\theta}^\nu(t) = 0 \tag{B.1}$$

The Euler-Lagrange equations are a set of N coupled differential equations ($\kappa = 1, \ldots, N$). The geodesic path has been parametrised by the parameter t, with $\theta^\kappa(0) = \theta_1^\kappa$ and $\theta^\kappa(1) = \theta_2^\kappa$. $\boldsymbol{\theta_1} = (\theta_1^1, \ldots, \theta_1^N)^T$ and $\boldsymbol{\theta_2} = (\theta_2^1, \ldots, \theta_2^N)^T$ are respectively the begin and end point of the geodesic. $\Gamma^\kappa_{\mu\nu}$ are called the Christoffel symbols and are given by

$$\Gamma^\kappa_{\mu\nu} = \frac{1}{2} g^{\kappa\rho}\left(\frac{\partial g_{\nu\rho}}{\partial\theta^\mu} + \frac{\partial g_{\mu\rho}}{\partial\theta^\nu} - \frac{\partial g_{\mu\nu}}{\partial\theta^\rho}\right) \tag{B.2}$$

In the last expression, $g^{\kappa\rho}$ represents the inverse metric tensor. The definition of the inverse metric tensor is

$$g^{\kappa\rho} g_{\rho\lambda} = \delta^\kappa_\lambda \tag{B.3}$$

where δ^κ_λ is the Kronecker delta tensor. The above expression holds for all κ and λ.
For the univariate, zero mean GGDs with fixed β, the equations simplify considerably. The metric is given by (A.10), which is repeated here for convenience.

$$ds^2 = \beta \frac{d\alpha^2}{\alpha^2}$$

When the beginning and end point have scale parameters α_1 and α_2, it is possible to change the coordinate system to

$$\begin{aligned} ds^2 &= \beta \frac{\frac{d\alpha^2}{\alpha_1^2}}{\frac{\alpha^2}{\alpha_1^2}} \\ &= \beta dr^2 \end{aligned} \tag{B.4}$$

with $r = \ln(\alpha/\alpha_1)$. In this coordinate system, $g_{rr} = \beta$ is a constant, so that the derivative of g_{rr} vanishes and the Christoffel symbol becomes $\Gamma^r_{rr} = 0$. The Euler-Langrange equation in this coordinate system is

$$\ddot{r}(t) = 0 \tag{B.5}$$

The solution of this equation with boundary conditions $r(0) = 0$ and $r(1) = \ln(\alpha_2/\alpha_1)$ is

$$r(t) = \ln(\alpha_2/\alpha_1)t = \ln(\lambda_2)t \tag{B.6}$$

To find the tangent vector in the original coordinate system, we use the Ansatz $r(t) = \ln(\lambda(t))$ where $\lambda(t) = \alpha(t)/\alpha_1$. From this

$$\begin{aligned} & \lambda(t) = \exp\left(\ln(\lambda_2)t\right) \\ \Rightarrow \quad & \alpha(t) = \alpha_1 \exp\left[\ln\left(\frac{\alpha_2}{\alpha_1}\right) t\right] \end{aligned} \tag{B.7}$$

The tangent vector at the start point of the geodesic is then given by

$$d\alpha(t=0) = \alpha_1 \ln\left(\frac{\alpha_2}{\alpha_1}\right) dt \tag{B.8}$$

In the beginning of the section it was clear from figure B.1 that for one-dimensional spaces, only the sign of above expression is crucial. $dt > 0$ by definifition, so that the logarithmic map is given by

$$\alpha_1\vec{\alpha}_2 = \operatorname{sgn}\left(\alpha_1 \ln\left(\frac{\alpha_2}{\alpha_1}\right)\right) d_G(\alpha_1 \| \alpha_2) \tag{B.9}$$

Note that although $\alpha_1\vec{\alpha}_2 \in T_{\alpha_1}\mathcal{M}$ is a vector, it is a one-dimensional vector and thus also a scalar value. $d_G(\alpha_1\|\alpha_2)$ represents the geodesic distance of (2.38) and is repeated here for simplicity:

$$d_G(\alpha_1\|\alpha_2) = \sqrt{\beta}\left|\ln\left(\frac{\alpha_2}{\alpha_1}\right)\right| \tag{B.10}$$

The exponential map is the inverse transformation, projecting points from the tangent plane back to the manifold. Points in the tangent plane are determined by $\alpha_1\vec{\alpha}_2$ in (B.9) and moreover $\mid \alpha_1\vec{\alpha}_2 \mid = d_G(\alpha_1\|\alpha_2)$. Solving expression (B.10) for α_2 and using former equivalence relation, one finds

$$\alpha_2 = \alpha_1 \exp\left[\frac{\alpha_1 \vec{\alpha}_2}{\sqrt{\beta}}\right] \tag{B.11}$$

Appendix C

Euclidean representation of probabilistic spaces

Visualisation methods, such as MDS, prove to be useful in some cases, but lack information about the geometry of the original space. Even if the original pairwise distances are well conserved, the distances become Euclidean distances. This also means that the distance between two faraway points in the embedding is to be measured along the line connecting the points, which in general will not coincide with a path along consecutive neighbours. Geodesic paths are thus not conserved in an MDS embedding. Also the curvature of the original space is in general not conserved[1].

In the case of the Riemannian manifold of univariate, zero-mean GGD distributions with fixed β, it is possible to construct an equivalent representation in a Euclidean space. Equivalent representations have identical geometric properties. The goal is thus to find a diffeomorfism (an isomorphism between differentiable manifolds) between the probalistic manifold of univariate and zero-mean GGD distributions and a Euclidean space. The representation in a one-dimensional Euclidean space is not interesting for visualisation purposes[2]. We propose an equivalent representation in a two-dimensional Euclidean space. If such a representation is found, it will conserve all geometric properties and a search for still higher-dimensional representations becomes unnecessary. Two equivalent representations have the same metric. The two representations are thus linked by the constraint

$$ds^2 = \beta \frac{d\alpha^2}{\alpha^2} = dx^2 + dy^2 \tag{C.1}$$

The representation in the two-dimensional space needs to remain intrinsically one-dimensional. The representation will thus be a one-dimensional curve. The Euclidean coordinates can be parametrised along this curve. We arbitrarily choose that parameter equal to the scale parameter α of the GGD. (C.1) can then be rewritten as

$$ds^2 = \beta \frac{d\alpha^2}{\alpha^2} = \left[\left(\frac{dx}{d\alpha} \right)^2 + \left(\frac{dy}{d\alpha} \right)^2 \right] d\alpha^2 \tag{C.2}$$

[1] We refer to the *intrinsic* curvature from differential geometry. For one-dimensional Riemannian spaces, this curvature is identically zero.

[2] An equivalent one-dimensional Euclidean representation is $x(\alpha) = \sqrt{\beta} \ln \alpha$.

The representation will not be unique. $x(\alpha)$ and $y(\alpha)$ are coupled by previous expression, but different choices of $x(\alpha)$ will deliver equivalent representations, at least for a certain range of the parameter α. The most meaningful representation is one, in which one of the Euclidean coordinates is identified with the scale parameter. We choose $x(\alpha) = \alpha$. This will allow to easily identify points on the curve with GGD distributions. It should be highlighted that this choice of $x(\alpha)$ does not necessarily lead to a representation. It will in fact become clear that for this choice, it is not possible to represent the entire probabilistic manifold in the Euclidean space[3]. For this choice, (C.2) becomes

$$\frac{\beta}{\alpha^2} d\alpha^2 = \left[1 + \left(\frac{dy}{d\alpha}\right)^2\right] d\alpha^2 \tag{C.3}$$

This is a differential equation relating the dependent variable y to the independent variable α, in such a way that the infinitesimal distance between points along the curve in the Euclidean plane is the same as in the original space of distributions. This equation can be solved to find the α dependency of y by separation of variables

$$dy = \sqrt{\frac{\beta}{\alpha^2} - 1} d\alpha \tag{C.4}$$

The positive square root is chosen arbitrarily. The argument of the square root is positive when $\alpha \leq \sqrt{\beta}$. This is always satisfied for the wavelet statistics. Typically Laplacian distributions are used, with $\beta = 1$. The highest fitted α values are of the order of 10^{-3}.
The geometrical aspects do not depend on the constants of integration of above expression. Only a primitive is needed. To integrate the right hand side of (C.4), the change of variables is proposed

$$\begin{aligned} \frac{\sqrt{\beta}}{\alpha} = \cosh\theta \Leftrightarrow \theta = \operatorname{arccosh}\left(\frac{\sqrt{\beta}}{\alpha}\right) \\ d\alpha = \sqrt{\beta} d\left(\frac{1}{\cosh\theta}\right) = -\sqrt{\beta}\frac{\sinh\theta}{\cosh^2\theta} d\theta \end{aligned} \tag{C.5}$$

With this change of variables, integration of (C.4) gives

$$y(\theta) = -\sqrt{\beta} \int \tanh^2\theta \, d\theta \tag{C.6}$$

Following identity is used

$$\tanh\theta = \frac{\sinh\theta}{\cosh\theta} = \frac{\sqrt{\cosh^2\theta - 1}}{\cosh\theta} \tag{C.7}$$

Combining (C.6) and (C.7) delivers

$$\begin{aligned} y(\theta) &= \sqrt{\beta}\left[\int d\theta - \int \frac{1}{\cosh^2\theta} d\theta\right] \\ &= \sqrt{\beta}\left[\theta - \tanh\theta\right] \end{aligned} \tag{C.8}$$

[3] A Euclidean representation for the entire manifold can easily be found by choosing $x(\alpha) \propto \ln\alpha$.

$y(\theta)$ can be rewritten in function of α by combining (C.5), (C.7) and (C.8). After some simplifications, the expression for $y(\alpha)$ becomes

$$y(\alpha) = \sqrt{\beta}\left[\operatorname{arccosh}\frac{\sqrt{\beta}}{\alpha} - \frac{1}{\sqrt{\beta}}\sqrt{\beta - \alpha^2}\right] \tag{C.9}$$

The scale parameter is typically several orders of magnitude lower than the shape parameter. In the limit $\alpha \ll \sqrt{\beta}$, the second term simplifies to 1. Moreover, the following identity holds: $\operatorname{arccosh}(x) = \ln(x + \sqrt{x^2 - 1})$ for $x > 1$. For $x \gg 1$, $\operatorname{arccosh}(x) \approx \ln(2x)$. Thus (C.9) reduces to

$$y(\alpha) = \sqrt{\beta}\left[\ln\frac{2\sqrt{\beta}}{\alpha} - 1\right] \tag{C.10}$$

The representation of the manifold of univariate, zero-mean Laplace distributions is shown in figure 2.14 for $10^{-5} \leq \alpha \leq 10^{-3}$.

Appendix D

Code

D.1 Feature extraction

Feature extraction is performed with the command `FeatDisrupt.m`. `FeatDisrupt.m` requires as input cell arrays `X` and `Err`. The cell array `X` is a concatenation of JET signals. The signals were extracted from JET by use of MDSplus. Each row in `X` corresponds to a shot and each column to a signal. Each cell in `X` is a two-column vector. The first column contains the time base and the second column contains the signal values on this time base. It was for some signals and certain shots not possible to get access by MDSplus. In this case, the corresponding cell array in `X` was left empty. If this was the case, the corresponding cell in `Err` was 1. Else it was 0.
The command for `FeatDisrupt.m` looks as follows

```
[alpha, beta, sigma, FGGD, FFou, T] = FeatDisrupt(Dir, FileN, NSig, ShotRange, ...
ShotNoD, TDis, NWin30, TBD, TBR, NS, b_fix, b_max, SOmit)
```

`Dir` is a string indicating the directory in which `X` and `Err` are stored. `X` and `Err` were saved in the same Matlab file. Usually `X` and `Err` were split in blocks per hundred shots because of the large volume of the data. The generic name for a block is `FileN`. `FeatDisrupt.m` relies heavily on the specific ordening of shots in blocks. The translation of a shot number to a block and row number in `X` happens in the subfunction `FileInfo`. When used by a different user, this part of the code should be adjusted!
`NSig` is the number of signals in the data set. `NSig`=13 and corresponds to the signals in table 1.2. `ShotRange` is a 1×2 vector. It contains the first and last shotnumber of the new data set. `ShotNoD` contains the shot numbers for which signals are available in `X`, in the same order as in `X` (usually `ShotNoD` has been used to extract the data by MDSplus). `TDis` is a vector with the same length as `ShotNoD` and contains the time of disruption for each shot. `NWin30` is the number of windows of 30 ms used to extract frequency information. It was held one in all tests. The role of `TBD` and `TBR` is discussed later on. `NS` is the wavelet decomposition level. `b_max` is the maximum allowed β in the fit of a GGD distribution to the wavelet spectrum. If `b_max` is zero, the β is held fixed according to `b_fix`. If `b_fix` is also zero, then β is allowed to take all possible values. `SOmit` is a vector and contains all signals which are to be excluded in the feature extraction. It contains values from one to `NSig`. `SOmit` can also be empty.

`FeatDisrupt.m` delivers cell arrays `alpha`, `beta` and `sigma`. Each row corresponds to a shot. Each column corresponds to a signal for which the feature extraction was performed. Each cell contains a vector with the values of respectively the α and β values of the GGD fit and the standard deviations σ in the Fourier spectrum for all time windows of the corresponding shot. The data is re-ordered in `FGGD` and `FFou`. `FGGD` and `FFou` are structures with fields `reg` and `dis`. `reg` contains feature vectors for regular time windows and `dis` for disruptive time windows. Each field contains subfields `data` and `meta`. `data` is a matrix containing the feature vectors for all time windows of the corresponding class. Each row corresponds to a feature vector. Each column corresponds to a feature. `FGGD` and `FFou` are used by the classification algorithms. The `meta` field contains for values for each feature vector. The values are, in the same order as they appear in consecutive columns, the shot number, the time of disruption, the beginning and end of the time window. At last, `T` is also a structure with fields `reg` and `dis`. Each field contains a cell array with the same number of cells as there are shots in the new data set. Each cell contains the boundaries of all time windows of a shot.
The features in `FFou` are ordered per signal. For wavelet features there exist several decomposition levels for each signal. For each decomposition level, two features are extracted (the α and β parameters of the GGD). A row in `FGGD` is ordened as follows: $[\alpha_{11}, \beta_{11}, \alpha_{12}, \ldots, \beta_{1\,\mathrm{NSig}}, \alpha_{21}, \ldots, \beta_{\mathrm{NS\,NSig}}]$. After initialisation, vectors `ISE2` and `IP` are created and indicate which features should be held in respectively `FFou` and `FGGD` after deleting features for signals in `SOmit`.

```
ISE2 = setdiff(1:NSig, SOmit);
NSE2 = length(ISE2);
IP = zeros(1, NSE2 * NS * 2);
for j = 1:NSE2
    for ns = 1:NS
        IP((ns - 1)*NSE2*2+(j-1)*2+1:(ns-1)*NSE2*2+(j-1)*2+2) = ...
            [(ns-1)*NSig*2+(ISE2(j)-1)*2+1,(ns-1)*NSig*2+(ISE2(j)-1)*2+2];
    end
end
```

The original JET data is loaded per block and some pre-processing is already performed in the function `TransfData`, a subfunction in `FeatExtract.m`. The processing concerns the computation of discrete time derivatives of signals (2), (5) and (8) in table 1.2 (the time derivatives are on their own separate predictor signals) and the removal of doubles in `X`. `FileInfo` is in the following the output of subfunction `FileInfo`. The first column contains the index of the first and last block for the shots in `ShotRange`. The second column contains the row index of `X` at which the first or last shot of `ShotRange` is to be found.

```
CBlockShot = 0;  % At any moment CurShot = CBlockShot + NShotCurrent.
for ib = FileInfo(1, 1):FileInfo(2, 1)
    D = load([Dir, '\', FileN, '_', num2str(ib)]);
    [X, Err] = TransfData(D, NSig);
    clear('D');

    if ib == FileInfo(1, 1)
            istart = RedShotRange(1) - FileInfo(1, 2) + 1;
```

```
        X = X(istart:end, :);
        Err = Err(istart:end, :);
    elseif ib == FileInfo(2, 1)
       iend = RedShotRange(2) - FileInfo(2, 2) + 1;
       X = X(1:iend, :);
       Err = Err(1:iend, :);
    end
    [NShot, ~] = size(X);
```

Still in the `for`-loop with index `ib`, the data for each shot is processed. Each signal is examined for not-a-number (NaN) values and such values are replaced. NaNs are replaced in such a way as to insert as less discontinuities as possible in order to not affect the transient behaviour of the signal.

```
for i = 1:NShot
    % Current shot number among the total number of shots NShotTot
    CurShot = CBlockShot + i;
    RealCurShot = RedShotRange(1) + CurShot - 1;
    if (TDis(RealCurShot) > 0)  && (TDis(RealCurShot) < 80)
        try
            disp(['Shot# ', num2str(ShotNoD(RealCurShot)), ...
                ' (', num2str(CurShot), '/', num2str(NShotTot), ')']);
            S_raw = [];
            NSE = 0;  % Effective number of signals for shot i
            for j = 1:NSig
                if ~Err(i, j)
                    if ~isempty(X{i, j})
                        NSE = NSE + 1;
                        Temp = X{i, j};
                        INaN = isnan(Temp(:, 2));
                        if ~isempty(find(INaN, 1))
                            disp(['Found NaN in shot ', num2str(CurShot), ...
                                ' at signal ', num2str(j)]);
                            tel = 1;
                            while ~isempty(find(INaN, 1)) && tel<=5
                            % assuming right neighbour of NaN is not NaN itsself
                                 Temp(INaN, 2) = Temp(circshift(INaN,1),2);
                                 INaN = isnan(Temp(:, 2));
                                 tel = tel+1;
                            end
                            if tel >5 %too many adjacent NaN's: avoid endless loop
                                magn = ceil(mean(log10(abs(Temp(not(INaN), 2)))));
                                Temp(INaN, 2) = 10^magn;
                            end
                            if ~isempty(find(INaN, 1))
                                disp(['Found only NaN in shot ', num2str(CurShot), ...
                                    ' at signal ', num2str(j)]);
                                Temp(INaN, 2) = 0;
                            end
                        end
                        S_raw{NSE, 1} = Temp;
                    else
```

```
                    Err(i, j) = true;
                end
            end
        end
```

The next step consists of resampling at a rate of 1 kHz (`TS`=1e-3). The `for`-loop with index `j` is still running, each signal is resampled at the same time base.

```
if NSE > 0  % There should effectively be signals
    S = Resample(S_raw, TS, 40, TDis(RealCurShot));
    % Resample all signals for the current shot to 1 kHz,
    % starting at 40 s (magnetic time) and ending at the time of disruption.
```

The resampling is followed by a similar procedure as described above to remove new NaNs. The normalisation to interval [0,1] is performed as follows:

```
if any(S{jse}(:,2))
    Mi = min(S{jse}(:, 2));
    Ma = max(S{jse}(:, 2));
    S{jse}(:, 2) = (S{jse}(:, 2) - Mi)/(Ma - Mi);  % Normalize
else
    disp(['Found zero-valued vector in shot ', num2str(CurShot),...
     ' at interpolated signal ', num2str(j)]);
end
```

`jse` is an effective signal index, which increases by one in the `for`-loop with index `j` only if no error has occurred. At this point also `if ~Err(i, j)` and `for j = 1:NSig` end.
The extraction of Fourier and wavelet features is split into regular and disruptive windows. The boundaries of each time window need to be defined.

```
RT = S{1}(:, 1);  % Real time vector
NOS = length(RT);
% Total number of samples (= sample number of the approximate disruption time).
NOSWin = NWin30 * (30e-3)/TS;  % Number of samples per window
NFFT = 2^nextpow2(NOSWin);  % Next power of 2 from NOSWin for Fourier transform
TWin.reg = 1:NOSWin:(NOS - 1/TS);
%TWin.reg = (NOS-TBR/TS):NOSWin:(NOS - 1/TS);
% Boundaries of nondisruptive time windows, in number of samples, from TBR before
% ToD to one second before the disruption time at the latest.
% The first entry of TWin corresponds to T0 = 0 ms.
% The nth entry corresponds to (n - 1) * NOSWin * TS s.
NOW.reg = length(TWin.reg) - 1;  % Number of nondisruptive time windows
TWin.dis = (NOS - (30e-3)/TS):-NOSWin:(NOS - TBD/TS);
% Boundaries of disruptive time windows. The last boundary occurs at approximately
% TBD seconds before the disruption.
TWin.dis = TWin.dis(end:-1:1);  % Reverse vector
NOW.dis = length(TWin.dis) - 1;  % Number of disruptive time windows
```

`TBD`, one of the inputs of **`FeatExtract.m`**, indicates the instant before disruption which is still considered indicative for disruptive behaviour. `TBR` is the time most prior to disruption which

is used to extract information about the regular regime. The instant closest to disruption which is still used to extract regular information (1 s) has been held fixed. The use of `TBR` is optional. In the previous code fragment, this option was commented to neglect it. In real-time equivalent tests, the code should be used as indicated above.
Per regime and shot, `alpha`, `beta` and `sigma` cells are initialised.

```
FN = {'reg', 'dis'};
for fn = 1:2
    if NOW.(FN{fn}) > 0
        T.(FN{fn}){CurShot} = RT(TWin.(FN{fn}));
        % Boundaries of time windows for shot i, in seconds.
        for j = 1:NSig
            if ~Err(i, j)
                alpha.(FN{fn}){CurShot, j} = zeros(NOW.(FN{fn}), NS);
                beta.(FN{fn}){CurShot, j} = zeros(NOW.(FN{fn}), NS);
                sigma.(FN{fn}){CurShot, j} = zeros(NOW.(FN{fn}), 1);
            end
        end
```

The wavelet and Fourier features are extracted in subfunction `CalcFeat`. This function is summoned as follows:

```
% The following are temporary feature vectors that contain features for all
% NSig signals (some of which may be empty). Only the features for the
% signals with indices in ISE2 will eventually picked out of these.
Fs.(FN{fn}) = zeros(1, NSig);
Fab.(FN{fn}) = zeros(1, NSig * NS * 2);
for iw = 1:NOW.(FN{fn})  % Feature extraction
    jse = 0;  % Index for effective signal (with a maximum NSE)
    for j = 1:NSig
        if ~Err(i, j)
            jse = jse + 1;
            SS = S{jse}(TWin.(FN{fn})(iw):TWin.(FN{fn})(iw + 1), 2);
            if b_max ~= 0
                b_fix = 0;
                [alpha.(FN{fn}){CurShot, j}(iw, :), ...
        beta.(FN{fn}){CurShot, j}(iw, :), sigma.(FN{fn}){CurShot, j}(iw, 1)] = ...
            CalcFeat(SS, NOSWin, NFFT, NS, F_M, b_fix);
                if any(beta.(FN{fn}){CurShot, j}(iw, :)>b_max) || ...
                    any(isnan(beta.(FN{fn}){CurShot, j}(iw, :))) || ...
                        any(beta.(FN{fn}){CurShot, j}(iw, :)==b_max)
                    b_fix = b_max;
                    [alpha.(FN{fn}){CurShot, j}(iw, :),...
         beta.(FN{fn}){CurShot, j}(iw, :), sigma.(FN{fn}){CurShot, j}(iw, 1)] = ...
                     CalcFeat(SS, NOSWin, NFFT, NS, F_M, b_fix);
                end
            else
                [alpha.(FN{fn}){CurShot, j}(iw, :), ...
         beta.(FN{fn}){CurShot, j}(iw, :), sigma.(FN{fn}){CurShot, j}(iw, 1)] = ...
                CalcFeat(SS, NOSWin, NFFT, NS, F_M, b_fix);
```

```
            end
        Fs.(FN{fn})(1, j) = sigma.(FN{fn}){CurShot, j}(iw, 1);
        for ns = 1:NS
            Fab.(FN{fn})(1,(ns-1)*NSig*2+(j-1)*2+1:(ns-1)*NSig*2+(j-1)*2+2) = ...
            [alpha.(FN{fn}){CurShot, j}(iw, ns),beta.(FN{fn}){CurShot, j}(iw, ns)];
        end
    end
end
```

Subfunction `CalcFeat` is given below. Once the feature extraction has been performed for all signals in a time window, the features of signals which do not appear in `SOmit` are added to `FGGD` and `FFou`. To avoid any problems in the classification problem, some additional constraints are tested.

```
        if (~any(isnan(Fab.(FN{fn})(IP)))) && ...
            (~any(Fab.(FN{fn})(IP) == 0)) && (~any(isempty(Fab.(FN{fn})(IP))))
        % Only add the data vectors to FGGD.(FN{fn}) and FFou.(FN{fn}) if all GGD
        % features for all required signals (i.e. those with indices in ISE2) are OK.
            n.(FN{fn}) = n.(FN{fn}) + 1;
            FGGD.(FN{fn}).data(n.(FN{fn}), :) = Fab.(FN{fn})(IP);
            FGGD.(FN{fn}).meta(n.(FN{fn}), :) = [ShotNoD(RealCurShot), ...
            TDis(RealCurShot), T.(FN{fn}){CurShot}(iw), T.(FN{fn}){CurShot}(iw + 1)];
            FFou.(FN{fn}).data(n.(FN{fn}), 1:NSE2) = Fs.(FN{fn})(ISE2);
            FFou.(FN{fn}).meta(n.(FN{fn}), :) = [ShotNoD(RealCurShot), ...
            TDis(RealCurShot), T.(FN{fn}){CurShot}(iw), T.(FN{fn}){CurShot}(iw + 1)];
        end
    end
else
    disp(['No ', FN{fn}, ' time windows in shot ', num2str(CurShot), '. Skipping.']);
end
```

The second `end` in previous fragment closes the `for`-loop with index `fn`. The `else`-clause corresponds to `nif NOW.(FNf) > 0`.
All open `for`-loops are closed at this point. The `try`-clause of the second code fragment was added to ensure that the feature extraction does not stop by an error for one shot. If an error does occur, an error message is produced.

```
catch err
    disp(['Error processing shot ', num2str(CurShot), '. Skipping.']);
    disp(strcat(err.stack.name,': line ',num2str(err.stack(1).line)))
end
```

Once all shots of a block have been processed, `CBlockShot` is updated. The temporary loaded `X` and `Err` are deleted to replace them with `X` and `Err` of the following block.
The features of the time sequence for one signal and one time window is computed by the subfunction `CalcFeat`. This function is called as follows:

```
function [alpha, beta, sigma] = CalcFeat(SS, NOSWin, NFFT, NS, F_M, b_fix)
```

`SS` is a vector with the signal values for one time window. `NOSWin` is the length of `SS`. `NFFT` is a multiple of two which best approaches `NOSWin` in order to optimise the FFT transform. `NS` is the wavelet decomposition level. `F_M` is a look-up table to solve the highly non-linear equation in (2.9). This table needs to be stored in the same map with **`FeatExtract.m`**. It is loaded in the beginning of the initialisation part by `load('.F_M_short.mat');`. The function returns `NS` α and β GGD parameters stored in respectively vectors `alpha` and `beta`. Also a single value `sigma` is returned, the standard deviation of the Fourier power spectrum.

```
Y = fft(SS, NFFT)/NOSWin;  % Fourier transform
P = 2 * abs(Y(2:NFFT/2 + 1));
% Single-sided amplitude spectrum. Discard first component (DC).
sigma = std(P);

WT = ndwt(SS, NS, 'db2');
alpha = zeros(1, NS);
beta = zeros(1, NS);
for ns = 1:NS
    SWC = cell2mat(WT.dec(ns + 1));
    [alpha(1, ns), beta(1, ns)] = CalcAlfaBeta(SWC, F_M, b_fix);
end
```

The fit of a GGD is performed in **`CalcAlfaBeta.m`** which needs to be in the same map as **`FeatExtract.m`**. Function `CalcAlfaBeta` includes the following:

```
function [alfa, beta] = CalcAlfaBeta(CMatrix, F_M, b_fix)

        C = CMatrix(1:end);
    if ~b_fix
        m1 = mean(abs(C));
        m2 = mean(C.^2);
        zeta = m1 / sqrt(m2);
        if zeta > F_M(end, 2)
            zeta = F_M(end, 2);
        elseif zeta < F_M(1, 2)
            zeta = F_M(1, 2);
        end
        beta_i = interp1(F_M(:, 2), F_M(:, 1), zeta);

        beta = beta_i;
        tol = 1;
        NIter = 0;
        while (tol > 1E-6) && (NIter < 100)
            NIter = NIter + 1;
            [g, g_p] = gs(beta, C);
            new_beta = beta - g/g_p;
            if new_beta < 0.1
                break;
            end
            tol = abs(new_beta - beta);
```

```
            beta = new_beta;
        end
    else
        beta = b_fix;
    end

    alfa = (beta * mean((abs(C)).^beta))^(1/beta);
end



function [g, g_p] = gs(beta, C)
% Calculates g in relation (16) and g' in (25) in Do & Vetterli.

    F1 = sum((abs(C)).^beta .* (log(abs(C))).^2);
    F2 = sum((abs(C)).^beta .* log(abs(C)));
    F3 = sum((abs(C)).^beta);
    F4 = mean((abs(C)).^beta);

    g = 1 + psi(1/beta)/beta - F2/F3 + log(beta * F4) / beta;

    g_p = -psi(1/beta)/beta^2 - psi(1, 1/beta)/beta^3 + 1/beta^2 ...
        - F1/F3 + (F2/F3)^2 + F2/(beta * F3) ...
        - log(beta * F4) / beta^2;
end
```

The solution scheme is a Newton-Ralphson iterative proceduce to solve (2.9). It is described in [34].

D.2 Classification

D.2.1 Test and training sets

The `FGGD` and `FFou` outputs of **`FeatExtract.m`** were split in a test and training set. This was done on basis of shotnumbers. A ratio of all shotnumbers in `FGGD` and `FFou` was randomly selected to create the training set. The rest of the shots were used to generate the test set. A function to split the data set is named **`GetShotTrainData.m`**. It is called in the following way:

```
[FGGDTrain, FFouTrain, FGGDTest, FFouTest, ITrain, ITest] = ...
    GetShotTrainData(FGGD, FFou, Ratio, ITrain, ITest)
```

If input variables `ITrain` and `ITest` are not empty, the shotnumbers in those vectors are used instead. This option was included in order to be able to reconstruct the test and training sets of previous tests. This option was also used in order to test the *J-div* classifier with data of the same shots as the other classifiers. This was necessary, as the *J-div* classifier is based on a different set of wavelet features than the other classifiers.
The first part of the code is:

```
FN1 = fieldnames(FGGD);  % Contains field names 'reg' and 'dis'.
```

```
NF1 = length(FN1);

if isempty(ITrain)
    % use same shotnumbers for all classes
    shotno = unique(FGGD.(FN1{1}).meta(:,1))';
    NOS = length(shotno);
    P = randperm(NOS);
    NTrain = ceil(NOS * Ratio);
    NTest = NOS-NTrain;
    ITrain = (shotno(P(1:NTrain)))';
    ITest = (shotno(P(NTrain + 1:NOS)))';
    if isempty(ITrain) % make sure both ITrain and ITest not empty
        ITrain = shotno(P(1))';
        ITest = (shotno(P(2:NOS)))';
        NTrain = 1;
        NTest = NOS-1;
    elseif isempty(ITest)
        ITrain = shotno(P(1:NOS-1))';
        ITest = (shotno(P(NOS)))';
        NTrain = NOS-1;
        NTest = 1;
    end
else
    NTrain = length(ITrain);
    NTest = length(ITest);
end

FN2 = fieldnames(FGGD.(FN1{1}));  % Containes field names 'data' and 'meta'.
NF2 = length(FN2);
for i = 1:NF1
    for j = 1:NF2
        FGGDTrain.(FN1{i}).(FN2{j}) = [];
        FFouTrain.(FN1{i}).(FN2{j}) = [];
        FGGDTest.(FN1{i}).(FN2{j}) = [];
        FFouTest.(FN1{i}).(FN2{j}) = [];
    end
end
for k = 1:NTrain
    FGGDtemp = GetCampaignData(FGGD,[ITrain(k), ITrain(k)]);
    FFoutemp = GetCampaignData(FFou,[ITrain(k), ITrain(k)]);
    for i = 1:NF1
        for j = 1:NF2
            FGGDTrain.(FN1{i}).(FN2{j}) = ...
            [FGGDTrain.(FN1{i}).(FN2{j}); FGGDtemp.(FN1{i}).(FN2{j})];
            FFouTrain.(FN1{i}).(FN2{j}) = ...
            [FFouTrain.(FN1{i}).(FN2{j}); FFoutemp.(FN1{i}).(FN2{j})];
        end
    end
end
for k = 1:NTest
    FGGDtemp = GetCampaignData(FGGD,[ITest(k), ITest(k)]);
    FFoutemp = GetCampaignData(FFou,[ITest(k), ITest(k)]);
    for i = 1:NF1
```

```
        for j = 1:NF2
            FGGDTest.(FN1{i}).(FN2{j}) = ...
            [FGGDTest.(FN1{i}).(FN2{j}); FGGDtemp.(FN1{i}).(FN2{j})];
            FFouTest.(FN1{i}).(FN2{j}) = ...
            [FFouTest.(FN1{i}).(FN2{j}); FFoutemp.(FN1{i}).(FN2{j})];
        end
    end
end
```

GetCampaignData.m is a function which extracts from a dataset FGGD or FFou all feature vectors with shotnumbers between the two values given in its second argument. Here the function is called by giving the same begin and end shot number, in order to extract the data for each shot separately. The source code in **GetCampaignData.m** is given below.
In our tests it was not possible to work with the huge amount of regular time windows which are present in FGGD and FFou if the optional line in **FeatExtract.m** which makes use of TBR is switched off. The second part of **GetShotTrainData.m** makes it possible to create training and test sets with an equal number of disruptive and regular feature vectors.

```
FGGDTrain2.dis.data = FGGDTrain.dis.data;
FFouTrain2.dis.data = FFouTrain.dis.data;
FGGDTrain2.dis.meta = FGGDTrain.dis.meta;
FFouTrain2.dis.meta = FFouTrain.dis.meta;
FGGDTest2.dis.data = FGGDTest.dis.data;
FFouTest2.dis.data = FFouTest.dis.data;
FGGDTest2.dis.meta = FGGDTest.dis.meta;
FFouTest2.dis.meta = FFouTest.dis.meta;
FGGDTrain2.reg.data = [];
FFouTrain2.reg.data = [];
FGGDTrain2.reg.meta = [];
FFouTrain2.reg.meta = [];
FGGDTest2.reg.data = [];
FFouTest2.reg.data = [];
FGGDTest2.reg.meta = [];
FFouTest2.reg.meta = [];
% For shots for which disruptive features are available, choose
% regular time windows closest to disruption.
ShotsTrain = unique(FGGDTrain.dis.meta(:,1))';
NTraind = length(ShotsTrain);
ShotsTest = unique(FGGDTest.dis.meta(:,1))';
NTestd = length(ShotsTest);

for k = 1:NTraind
    nf = sum(FGGDTrain.dis.meta(:,1) == ShotsTrain(k));
    indtemp = find(FGGDTrain.reg.meta(:,1) == ShotsTrain(k));
    nfr = length(indtemp);
    if nfr > 0
        if nfr > nf
            indtemp = indtemp(nfr-nf+1:nfr);
        end
        FGGDTrain2.reg.data = [FGGDTrain2.reg.data;FGGDTrain.reg.data(indtemp,:)];
        FGGDTrain2.reg.meta = [FGGDTrain2.reg.meta;FGGDTrain.reg.meta(indtemp,:)];
```

```
        FFouTrain2.reg.data = [FFouTrain2.reg.data;FFouTrain.reg.data(indtemp,:)];
        FFouTrain2.reg.meta = [FFouTrain2.reg.meta;FFouTrain.reg.meta(indtemp,:)];
    end
end
for k = 1:NTestd
    nf = sum(FGGDTest.dis.meta(:,1) == ShotsTest(k));
    indtemp = find(FGGDTest.reg.meta(:,1) == ShotsTest(k));
    nfr = length(indtemp);
    if nfr > 0
        if nfr > nf
            indtemp = indtemp(nfr-nf+1:nfr);
        end
        FGGDTest2.reg.data = [FGGDTest2.reg.data;FGGDTest.reg.data(indtemp,:)];
        FGGDTest2.reg.meta = [FGGDTest2.reg.meta;FGGDTest.reg.meta(indtemp,:)];
        FFouTest2.reg.data = [FFouTest2.reg.data;FFouTest.reg.data(indtemp,:)];
        FFouTest2.reg.meta = [FFouTest2.reg.meta;FFouTest.reg.meta(indtemp,:)];
    end
end

FGGDTrain = FGGDTrain2;
FGGDTest = FGGDTest2;
FFouTrain = FFouTrain2;
FFouTest = FFouTest2;
```

In the future it will be interesting to perform real-time equivalent experiments. If this is the case, the part of above code fragment which concerns the test set should be removed. It is, however, interesting to use the same number of regular and disruptive feature vectors in the training set in order to avoid preferential treatment for one regime in the classification process because of the excess of training vectors of the regular regime. The source code of **GetCampaignData.m** is the following:

```
function FGGDcamp = GetCampaignData( FGGD,ShotRange )
    FN = fieldnames(FGGD);% contains 'reg' and 'dis'
    N = length(FN);
    for n = 1:N
        ind = (FGGD.(FN{n}).meta(:,1) >= ShotRange(1)) & ...
            (FGGD.(FN{n}).meta(:,1) <= ShotRange(2));
        FGGDcamp.(FN{n}).data = FGGD.(FN{n}).data(ind,:);
        FGGDcamp.(FN{n}).meta = FGGD.(FN{n}).meta(ind,:);
    end
end
```

D.2.2 Grouplabels

On basis of `FGGD` or `FFou`, vectors `GTest` and `GTrain` are defined which contain labels describing the class of each feature vector. The class can be regular or disruptive, but can be generalised to more group labels, for example in disruption cause detection. **GetShotGroupIndex.m** automatically makes vectors `GTest` and `GTrain`. The assumption is made in the code that in the classification step, consecutive fields in `FGGD` and `FFou` for as well training set as test set are concatenated below each other in the same order as they appear in `FGGD` and `FFou`. For example,

if in `FGGD` the field `reg` appears first and the field `dis` second, the training set will be given to the classifier as: `[FGGDTrain.reg.data; FGGDTrain.dis.data]`. The test set is treated in the same way.

```
function [ GTrain,GTest ] = GetShotGroupIndex( FTrain,FTest )

FN = fieldnames(FTrain);
NG = length(FN);

GTrain = [];
GTest = [];
telTrain = 1;
telTest = 1;

for i = 1:NG
    niTrain = size(FTrain.(FN{i}).data,1);
    niTest = size(FTest.(FN{i}).data,1);
    GTrain(telTrain:telTrain+niTrain-1,1) = i;
    GTest(telTest:telTest+niTest-1,1) = i;
    telTrain = telTrain+niTrain;
    telTest = telTest+niTest;
end

end
```

D.2.3 Classification

Support vector machine

Use was made of the built-in Matlab functions `svmtrain` and `svmclassify`.

k-nearest neighbours

The source code for k-NN classification is given below. The associated Matlab file is **`kNNClass.m`**

```
function [L, RClass, D] = kNNClass(XTrain, XTest, GTrain, GTest, NG, k, DistFun, D)

    [n1, p] = size(XTrain);
    [n2, ~] = size(XTest);

    if isempty(D)  % Distances still to be computed.
        D = zeros(n1, n2);
        for i = 1:n1
            XTemp = repmat(XTrain(i, :), n2, 1);
            D(i, :) = (DistFun(XTemp, XTest))';
        end
    end

    [~, I] = sort(D, 1);
```

```
    I = I(1:k, :);  % Select indices of k nearest neighbors within the training set
    GTemp = repmat(GTrain, 1, n2);

    NN = zeros(k, n2);
    for i = 1:n2
        NN(:, i) = GTemp(I(:, i), i);  % Group labels of the nearest neighbors
    end

    Num = zeros(NG, n2);
    for i = 1:NG  % Count the number of nearest neighbors from every group.
        Num(i, :) = sum(NN == i, 1);
    end
    [~, L] = max(Num, [], 1);
    L = L';

    if ~isempty(GTest)
        RClass.T = zeros(NG, 1);
        RClass.F = zeros(NG, 1);
        for i = 1:NG
            RClass.T(i) = 100 * length(find((GTest == i) & (L == i)))/...
                length(find(GTest == i));  % True positives
            RClass.F(i) = 100 * length(find((GTest ~= i) & (L == i)))/...
                length(find(GTest ~= i));  % False positives
        end
    else
        RClass = [];
    end
end
```

This code can be used as a generic file for all possible k-NN classification problems. The algorithm can be used for multiple-class problems. `NG` is the number of groups. The distance function can be chosen arbitrarily. `DistFun` is a function handle to the function which computes the similarity measure. An example is `DistFun = @Eucl`, if the function for the similarity measure is written in file **`Eucl.m`**. The different distance functions are given in section D.3 of this appendix.

Mahalanobis

The Mahalanobis classifier for independent, zero-mean, univariate Laplacian distributions is given below. The function file is **`MahaClass.m`**.

```
function [L, RClass, D] = MahaClass(XTest, mu, S, b, GTest, D, NG)

    [n2, ~] = size(XTest);

    if isempty(D)  % Distances still to be computed.
        D = zeros(NG, n2);
        for ng = 1:NG
            for i = 1:n2
                D(ng, i) = MahaDist(XTest(i,:), mu.(sprintf('ng_%d',ng)),...
                    S.(sprintf('ng_%d',ng)), b);
```

```
            end
        end
    end

    [~,L] = sort(D);
    L = L(1,:)';

    if ~isempty(GTest)
        RClass.T = zeros(NG, 1);
        RClass.F = zeros(NG, 1);
        for i = 1:NG
            RClass.T(i) = 100 * length(find((GTest == i) & (L == i)))/...
                length(find(GTest == i));  % True positives
            RClass.F(i) = 100 * length(find((GTest ~= i) & (L == i)))/...
                length(find(GTest ~= i));  % False positives
        end
    else
        RClass = [];
    end
end
```

Again the code is very general. The mean and covariance matrix for each class are found in structures `mu` and `S`. The fields of `mu` and `S` are constructed in a consisted way: `mu.ng_1` contains the (multivariate) mean of the cluster for the first class, `mu.ng_2` for the second class and so forth. The fields of `S` follow a similar construction.
Some clarification on the meaning of `mu` and `S` are necessary. `mu` contains the multivariate means for the cluster of each class *on the product space of zero-mean, univariate Laplacian manifolds*. In the tangent space, the mean is namely zero (the point at which the tangent space is constructed is by definition the origin of the space). The multivariate Gaussian model for the cluster is thus described solely by a covariance matrix. The covariance matrix describing the cluster of each class in the tangent space is contained in `S`.
The source code for the distance function `MahaDist.m` is the following:

```
function [D, mu, S] = MahaDist(a, mu, S, b)
    F = length(a);
    x = zeros(F,1);
        for i =1:F
        x(i) = LogMapGGD1D(mu(i), a(i), b);
    end
    D = sqrt(x.'*S^(-1)*x);
end
```

The test point is thus first projected to the tangent space at `mu` and then the Mahalanobis distance is computed in the tangent space. The exponential and logarithmic maps are treated separately in section D.4.

D.2.4 Conversion to real-time results

The classification algorithms above only give classification results in terms of TPR and FPR. In order to convert the results to real-time equivalents, the function file `RTequivs.m` should be

used. The function is called as follows:

```
[MA,FA,TE,SR,AVG, NShotT] = RTequivs(L,IFTest,GTest)
```

The inputs are the labels given by the classifier to each feature vector, `L`, the meta file of the test set, `IFTest` and the correct labels for the test set, `GTest`.
The first step is the initialisation:

```
function [MA,FA,TE,SR,AVG, NShotT] = RTequivs(L,IFTest,GTest)
    MA = 0;
    FA = 0;
    SR = 0;
    DT = [];
    Shots = unique(IFTest(:,1))';
    NShotT = length(Shots);
```

For each shot, the labels found by the classifier and the correct labels are stored in new variables for comparison.

```
    for k = 1:NShotT
        ind = find(Shots(k) == IFTest(:,1));
        Lsub = L(ind)';
        Tsub = repmat(IFTest(ind,2),1,2) - IFTest(ind,3:4);
        TWin = mean(Tsub,2); % middle of window used for DT and AVG
        Gsub = GTest(ind);
        if size(Lsub,2) == size(Gsub,1)
            Lsub = Lsub';
        end
```

Still in the `for`-loop with index `k`, a case study is performed to see whether a false alarm was generated, a succesfull prediction was made, or the disruption was not recognised. The order of the three logical tests is important.

```
        if any(Lsub == 2 & Gsub == 1) % premature alarm
            FA = FA+1;
        elseif any(Lsub == 2 & Gsub == 2) || ...
            (~any(Gsub == 2) && ~any(Lsub == 2)) % successful prediction
            SR = SR+1;
            time_pred = TWin(Lsub == 2 & Gsub == 2);
            if ~isempty(time_pred)
                DT = [DT;time_pred(1)];
            end
        else % missed alarm
            MA = MA + 1;
        end
    end
```

In the last step, the classification results are converted to percentages.

```
    MA = 100*MA/NShotT;
    FA = 100*FA/NShotT;
    TE = MA+FA;
    SR = 100*SR/NShotT;
    AVG = mean(DT);
end
```

D.3 Similarity measures

The k-NN classifier calls a function by the function handle `DistFun` to calculate the similarity matrix. All distance functions require two data sets of the same size. Each row of the data sets corresponds to a feature vector. The distance is computed between each pair of feature vectors with the same row index. The output `D` is a column vector containing the pairwise distances.

D.3.1 Euclidean distance

The Euclidean distance is computed by the function **`Eucl.m`**

```
function D = Eucl(X,Y)
D = (sum((X-Y).^2,2)).^(1/2);
end
```

D.3.2 J-divengence

The J-divergence is computed by **`Jdiv.m`**.

```
function D = Jdiv(X,Y)
l = size(X,2);
if mod(l,2)
    disp('Test/train data no right format')
    return
end
NFeat = l/2;
N = size(X,1);
D = zeros(N,1);
for j = 1:NFeat
    a1 = X(:,(j - 1) * 2 + 1)';
    b1 = X(:,(j - 1) * 2 + 2)';
    a2 = Y(:,(j - 1) * 2 + 1)';
    b2 = Y(:,(j - 1) * 2 + 2)';

    KLD12 = GGKL(a1,b1,a2,b2,1);
    KLD21 = GGKL(a2,b2,a1,b1,1);
    Dtemp = mean([KLD12; KLD21], 1);
    D = D+Dtemp';
end
end
```

`Jdiv.m` is based on function `GGKL.m`, which computes the KLD between feature vectors with the same row index. The input of `GGKL.m` is essentially the concatenation of parameters of univariate GGDs. The total distance between the joint PDFs consisting of a product of independent GGD distributions is computed above using the additionality property of the KLD.

```
function D = GGKL(a1, b1, a2, b2, Dim)
D = sum(log((b1.*a2.*gamma(1./b2))./(b2.*a1.*gamma(1./b1))) ...
    + (a1./a2).^b2.*(gamma((b2+1)./ b1)./gamma(1./b1))-1./b1, Dim);
end
```

D.3.3 Rao geodesic distance

The Rao geodesic distance is computed by `GGRao.m`.

```
function D = GGRao(X,Y)
l = size(X,2);
NFV = size(X,1);
if mod(l,2)
    disp('Test/train data no right format')
    return
end

Nfeat = l/2;
D = zeros(NFV,1);
for i = 1:Nfeat

    a1 = X(:, (i - 1) * 2 + 1);
    a2 = Y(:, (i - 1) * 2 + 1);
    bh = (X(:, (i - 1) * 2 + 2)+1)/12;

    Dtemp = (3*bh-1/4).*(log(a2.^2./a1.^2)).^2;
    D = D + Dtemp;
end
D = D.^(1/2);
end
```

This code is to be treated with care. The function only works appropriately for data sets for which β is held fixed for each GGD.
`GGRao.m` is based on the Rao geodesic distance for multivariate GGDs. Above expression however reduces to (2.38) in the univariate case. `bh` in `GGRao` is given by

$$bh = \frac{\beta + 1}{12} \tag{D.1}$$

The Rao geodesic distance in the code is given by

$$\begin{aligned} d_G((\alpha_1,\beta);(\alpha_2,\beta)) &= \sqrt{\left(3bh-\frac{1}{4}\right)\cdot \ln^2\left(\frac{\alpha_2^2}{\alpha_1^2}\right)} \\ &= \sqrt{(\frac{\beta+1}{4}-\frac{1}{4})\cdot 4\ln^2\left(\frac{\alpha_2}{\alpha_1}\right)} \\ &= \sqrt{\beta}\left|\ln\left(\frac{\alpha_2}{\alpha_1}\right)\right| \end{aligned} \tag{D.2}$$

and thus equivalent to (2.38).

D.4 Exponential and logarithmic maps

The logarithmic map for a univariate, zero-mean Laplacian distribution is computed using `LogMapGGD1D.m`.

```
function da = LogMapGGD1D(a1, a2, b)
% Logarithmic map for the univariate zero—mean GGD distribution with scale
% parameter matrix a2 to the tangent vector in the plane tangent at the GGD
% manifold in the point given by a1. a1 and a2 are scale parameters,
% or vectors of scale parameters of the same size (or one of them scalar).
% The scale parameter of the GGD is held fixed and equals b.

    da = sqrt(b)*log(a2./a1);

end
```

The exponential map is computed by use of `ExpMapGGD1D.m`.

```
function a2 = ExpMapGGD1D(da, a1, b)
% Exponential map for the zero—mean univariate GGD manifold, from the
% tangent vector da at the point a1 (scale parameter) to the point
% a2 on the manifold.
% The shape parameter of the GGD is held fixed and equals b.

     a2 = a1*exp(da/sqrt(b));

end
```

The geodesic centre-of-mass of a cluster is found using the iterative procedure of section 2.5. The procedure has been implemented in `CentrGGD1D.m`.

```
function [mu, v, da, da_mean] = CentrGGD1D(a,b)
% Searches for the centroid mu (scalar) of a cluster of n scale
% parameters in a for the GGD case with fixed shape parameter b. a is a 1 x n
% vector. An efficient gradient descent algorithm is used. Next,
% the variance v (scalar) of the cluster is calculated. da (1 x n) contains
% the tangent vectors at the mean, corresponding to the points in a
% (logarithmic maps). For good convergence of the algorithm, the mean
```

```
% da_mean of the tangent vectors in da should be 'close to' the zeros.

    mu = mean(a);  % Take the Euclidean mean of the a to initialize mu.
    tol = 100;
    NIter = 0;
    NMaxIter = 50;
    while (tol > 1) && (NIter < NMaxIter)
    % Repeat until tolerance is under 1 percent (or NIter >= NMaxIter).
        NIter = NIter + 1;
        da = LogMapGGD1D(mu, a, b);
        da_mean = mean(da);
        mu_new = ExpMapGGD1D(da_mean, mu, b);
        tol = 100 * abs((mu_new - mu)/mu);  % Relative change in percent
        mu = mu_new;
    end
    if NIter == NMaxIter
        disp('Maximum number of iterations reached.');
    end

    da = LogMapGGD1D(mu, a, b);  % Update the tangent vectors with the latest mu.
    v = var(da);
end
```

Usually the algorithm converges in a limited number of iterations.

D.5 Working example

To make the use of all code fragments more accessible, an example is given here which uses all functions above. The classification test is performed with shots of campaigns C15-C20 (396 disruptive shots with shotnumbers ranging from 65863 to 73128). First features with a constant $\beta = 1$ for the wavelet statistics (decomposition level = 3) are extracted. The Fourier features are extracted at the same time. The JET data is available in the map './Data'.
65% of the shots are randomly chosen to create the training set using `GetShotTrainData`. The group labels are determined using `GetGroupIndex`. The data is classified by a k-NN ($k = 1$) with Fourier features using the Euclidean distance. Afterwards, the same data set is classified using the geodesic distance between Gaussians. Therefore, the Fourier data set is modified. Finally, the Fourier features are classified by an SVM classifier with scale parameter of the RBF kernel $\sigma = 6$. The wavelet features are classified by a k-NN classifier and making use of the Rao geodesic distance. The wavelet features are also classified using a Mahalanobis classifier. First, the scale parameters in `FGGD` are converted to quadratic dispersions. The geodesic centre-of-mass for the cluster of regular events and the cluster of disruptive events is found using the proposed iterative prodecure. The covariance matrices are computed for both clusters. The covariance at different wavelet scales is neglected.
Afterwards, a data set with variable β for the GGD modeling of the wavelet statistics is constructed. The data set contains feature vectors for the same time windows as in the first feature extraction. The classification of this data set is performed using a k-NN classifier and the J-divergence. The same feature vectors are used as in the first test to train the classifier.
In the last line, the classification results of the k-NN classifier with wavelet features are converted

to real-time equivalent results.

```
Dir = 'Data';
FileN = 'X_Err_Disrupt';
NSig = 13;
ShotRange = [65863,73128];
NWin30 = 1;
TBD = 210e-3;
TBR = 0; % not used
NS = 3;
b_fix = 1;
b_max = 5;
SOmit = [];
load('ShotNoD');
load('TDis');

Ratio = 0.65;
sigmaSVM = 6;
NG = 2;
kneigh = 1;

[alpha, beta, sigma, FGGD, FFou, T] = ...
FeatDisrupt(Dir,FileN,NSig,ShotRange,ShotNoD,TDis,NWin30,TBD,TBR,NS,b_fix,b_max,SOmit);

[FGGDTrain,FFouTrain,FGGDTest,FFouTest,ITrain,ITest]=...
    GetShotTrainData(FGGD,FFou,Ratio,[],[]);
[GTrain,GTest]=GetShotGroupIndex(FGGDTrain,FGGDTest);

[L.FouEucl,RClass.FouEucl,~]=kNNClass([FFouTrain.reg.data;FFouTrain.dis.data],...
    [FFouTest.reg.data;FFouTest.dis.data],GTrain,GTest,NG,kneigh,@Eucl,[]);

FFouTrain2.reg.data(:,1:2:25) = sqrt(2)*FFouTrain.reg.data; % alpha = sqrt(2)*sigma
FFouTrain2.reg.data(:,2:2:26) = 2*ones(size(FFouTrain.reg.data));
FFouTrain2.dis.data(:,1:2:25) = sqrt(2)*FFouTrain.dis.data;
FFouTrain2.dis.data(:,2:2:26) = 2*ones(size(FFouTrain.dis.data));
FFouTrain2.reg.meta = FFouTrain.reg.meta;
FFouTrain2.dis.meta = FFouTrain.dis.meta;
FFouTest2.reg.data(:,1:2:25) = sqrt(2)*FFouTest.reg.data;
FFouTest2.reg.data(:,2:2:26) = 2*ones(size(FFouTest.reg.data));
FFouTest2.dis.data(:,1:2:25) = sqrt(2)*FFouTest.dis.data;
FFouTest2.dis.data(:,2:2:26) = 2*ones(size(FFouTest.dis.data));
FFouTest2.reg.meta = FFouTest.reg.meta;
FFouTest2.dis.meta = FFouTest.dis.meta;

[L.FouGD,RClass.FouGD,~]=kNNClass([FFouTrain2.reg.data;FFouTrain2.dis.data],...
    [FFouTest2.reg.data;FFouTest2.dis.data],GTrain,GTest,NG,kneigh,@GGRao,[]);

options = optimset('maxiter',50000);
SVMStruct = svmtrain([FFouTrain.reg.data;FFouTrain.dis.data],GTrain,...
    'Kernel_Function','rbf','RBF_Sigma',sigmaSVM,'quadprog_opts',options);
L.SVM = svmclassify(SVMStruct,  [FFouTest.reg.data;FFouTest.dis.data]);
for k = 1:2
```

```
    RClass.SVM.T(k) = 100 * length(find((GTest == k) & (L.SVM == k)))/...
        length(find(GTest == k));
    RClass.SVM.F(k) = 100 * length(find((GTest ~= k) & (L.SVM == k)))/...
        length(find(GTest ~= k));
end

[L.Wav,RClass.Wav,~]=kNNClass([FGGDTrain.reg.data;FGGDTrain.dis.data],...
    [FGGDTest.reg.data;FGGDTest.dis.data],GTrain,GTest,NG,kneigh,@GGRao,[]);

mu.reg = zeros(NS*NSig,1); % NS*NSig independent Laplacians
mu.dis = zeros(NS*NSig,1);
da.reg = zeros(size(FGGDTrain.reg.data,1),NS*NSig);
da.dis = zeros(size(FGGDTrain.dis.data,1),NS*NSig);
for ind = 1:NS*NSig
    % Find the centroid and variance of regular cluster.
    [mu.reg(ind),~,da.reg(:,ind),~] = ...
        CentrGGD1D(FGGDTrain.reg.data(:,2*(ind-1)+1),b_fix);
    % Find the centroid and variance of disruptive cluster.
    [mu.dis(ind),~,da.dis(:,ind),~] = ...
        CentrGGD1D(FGGDTrain.dis.data(:,2*(ind-1)+1),b_fix);
end
for ns = 1:NS % features at different wavelet scales independent
    S.reg((ns-1)*NSig+1:ns*NSig,(ns-1)*NSig+1:ns*NSig) = ...
        cov(da.reg(:,(ns-1)*NSig+1:ns*NSig));
    S.dis((ns-1)*NSig+1:ns*NSig,(ns-1)*NSig+1:ns*NSig) = ...
        cov(da.dis(:,(ns-1)*NSig+1:ns*NSig));
end

[L.Maha,RClass.Maha,~] = ...
    MahaClass([FGGDTest.reg.data(:,1:2:end-1);FGGDTest.dis.data(:,1:2:end-1)],...
        mu,S,b_fix,GTest,[],NG);

%feature extraction variable 0<beta<=5
[FGGDTrain,FFouTrain,FGGDTest,FFouTest,ITrain,ITest] = ...
    GetShotTrainData(FGGD,FFou,Ratio,ITrain,ITest);
[GTrain,GTest]=GetShotGroupIndex(FGGDTrain,FGGDTest);
[L.Jdiv,RClass.Jdiv,~]=kNNClass([FGGDTrain.reg.data;FGGDTrain.dis.data],...
    [FGGDTest.reg.data;FGGDTest.dis.data],GTrain,GTest,2,1,@Jdiv,[]);


[RT.kNN(1),RT.kNN(2),RT.kNN(3),RT.kNN(4),RT.kNN(5),RT.kNN(6)]=...
    RTequivs(L.kNN,[FGGDTest.reg.meta;FGGDTest.dis.meta],GTest);
```

Bibliography

[1] G.A. Rattá, J. Vega, A. Murari, G. Vagliasindi, M.F. Johnson, P.C. De Vries, and JET-EFDA contributors. An advanced disruption predictor for JET tested in a simulated real-time environment. *Nuclear Fusion*, 50(025005):025005–1–025005–10, 2010.

[2] B. Cannas, A. Fanni, E. Marongiu, and P. Sonato. Disruption forecasting at JET using neural networks. *Nuclear Fusion*, 44(1):68, 2004.

[3] G. A. Rattá, J. Vega, A. Murari, M. Johnson, and JET-EFDA contributors. Feature extraction for improved disruption prediction analysis in JET. *Review of scientific instruments*, 79:10F328–1–10F328–2, 2008.

[4] Commisariat à l'Énergie Atomique et aux Energies Alternatives. The ITER project. = http://www.cad.cea.fr/gb/activites/fusion/ITER.php. Visited in March 2012.

[5] M. Antonini, M. Barlaud, P. Mathieu, and I. Daubechies. Image coding using wavelet transform. *IEEE Transactions on Image Processing*, 1(2):205 –220, April 1992.

[6] G. Verdoolaege and P. Scheunders. Geodesics on the manifold of multivariate generalized Gaussian distributions with an application to multicomponent texture discrimination. *Journal of Mathematical Imaging and Vision*, 10.1007/s11263-011-0448-9, May 2011.

[7] G. Verdoolaege and P. Scheunders. On the geometry of multivariate generalized Gaussian models. *Journal of Mathematical Imaging and Vision*, DOI 10.1007/s10851-011-0297-8, May 2011.

[8] International Energy Agency. World energy outlook 2011: Press presentation. = http://www.iea.org/weo/, November 2011. Visited in March 2012.

[9] International Institute for Applied Systems Analysis. Global energy perspectives. = http://www.iiasa.ac.at/cgi-bin/ecs/book_dyn/bookcnt.py, August 1998. Visited in March 2012.

[10] J. Freidberg. *Plasma Physics and Fusion Energy*. Cambridge University Press, 2010.

[11] J. Ongena and G. Van Oost. Energy for future centuries. *Transactions of Fusion Science and Technology*, 53:3–15, Februari 2008.

[12] International Energy Agency. *How the energy sector can deliver on a climate agreement in Copenhagen*, October 2009.

[13] C. Jorant. The implications of Fukushima: The European perspective. *Bulletin of the Atomic Scientists*, 67(4):14 – 17, 2011.

[14] M. Cooper. The implications of Fukushima: The US perspective. *Bulletin of the Atomic Scientists*, 67(4):8 – 13, 2011.

[15] Thomas B. Cochran, Harold A. Feiveson, Walt Patterson, Gennadi Pshakin, M.V. Ramana, Mycle Schneider, Tatsujiro Suzuki, and Frank von Hippel. Fast breeder reactor programs: History and status. Technical report, International Panel on Fissile Materials, February 2010.

[16] Tritium (hydrogen-3). = http://www.ead.anl.gov/pub/doc/tritium.pdf, August 2005. Visited in March 2012.

[17] I. Cook, G. Marbach, L. Di Pace, C. Girard, and N. P. Taylor. *Safety and Environmental Impact of Fusion*. European Fusion Development Agreement, April 2001.

[18] European Commision: Research and Innovation. Fusion power: Safe and low-carbon. = http://ec.europa.eu/research/energy/euratom/fusion/microscope/safety-and-environment/index_en.htm. Visited in March 2012.

[19] EFDA JET. ITER-like wall project. = http://www.efda.org/jet/jet-iter/iter-like-wall-project/. Visited in March 2012.

[20] B. Patel and W. Parsons. *Operational Beryllium handling experience at JET*. EFDA JET, April 2001.

[21] P.C. De Vries, M.F. Johnson, B. Alper, P. Buratti, T.C. Hender, H.R. Koslowski, V. Riccardo, and JET-EFDA contributors. Survey of disruption causes at JET. *Nuclear Fusion*, 51:053018–1–053018–12, April 2011.

[22] G. Verdoolaege, A. Murari, J. Vega, and JET-EFDA Contributors. Recognition of disruptive behaviour in magnetically confined plasmas using the geodesic distance between wavelet distributions. Oktober 2011.

[23] F.C. Schüller. Disruptions in tokamaks. *Plasma physics and Controlled Fusion*, 37:A135–A162, 1995.

[24] M. Seki. ITER activities and fusion technology. *Nuclear Fusion*, 47(10):S489, 2007.

[25] T.C. Hender, J.C Wesley, J. Bialek, A. Bondeson, A.H. Boozer, R.J. Buttery, A. Garofalo, T.P Goodman, R.S. Granetz, Y. Gribov, O. Gruber, M. Gryaznevich, G. Giruzzi, S. Günter, N. Hayashi, P. Helander, C.C. Hegna, D.F. Howell, D.A. Humphreys, G.T.A. Huysmans, A.W. Hyatt, A. Isayama, S.C. Jardin, Y. Kawano, A. Kellman, C. Kessel, H.R. Koslowski, R.J. La Haye, E. Lazzaro, Y.Q. Liu, V. Lukash, J. Manickam, S. Medvedev, V. Mertens, S.V. Mirnov, Y. Nakamura, G. Navratil, M. Okabayashi, T. Ozeki, R. Paccagnella, G. Pautasso, F. Porcelli, V.D. Pustovitov, V. Riccardo, M. Sato, O. Sauter, M.J. Schaffer, M. Shimada, P. Sonato, E.J. Strait, M. Sugihara, M. Takechi, A.D. Turnbull, E. Westerhof, D.G. Whyte, R. Yoshino, H. Zohm, the ITPA MHD Disruption, and Magnetic Control Topical Group.

Chapter 3: MHD stability, operational limits and disruptions. *Nuclear Fusion*, 47(6):S128, 2007.

[26] F. Troyon, R. Gruber, H. Saurenmann, S. Semenzato, and S. Succi. MHD-limits to plasma confinement. *Plasma Physics and Controlled Fusion*, 26(1A):209, 1984.

[27] Lawrence Livermore National Laboratory. Plasma dictionary. `http://plasmadictionary.llnl.gov/`, March 2010. Visited in May 2012.

[28] EFDA JET. Magnetic fields. = http://www.efda.org/fusion/fusion-machine/magnetic-fields/. Visited in May 2012.

[29] Department of Physics The University of York. Neo-classical tearing modes. = http://www.york.ac.uk/physics/research/plasma-physics/research/mcf/nctm/, april 2010. Visited in March 2012.

[30] R. Fitzpatrick. Magnetic islands in plasmas. Presentation = http://www.york.ac.uk/physics/research/plasma-physics/research/mcf/nctm/, april 2010. Visited in March 2012.

[31] R. C. Wolf. Internal transport barriers in tokamak plasmas. *Plasma Physics and Controlled Fusion*, 45(1):R1, 2003.

[32] S. Kotsiantis, I. Zaharakis, and P. Pintelas. Machine learning: a review of classification and combining techniques. *Artificial Intelligence Review*, 26:159–190, 2006. 10.1007/s10462-007-9052-3.

[33] Eric W. Weisstein. Discrete Fourier transform. `http://mathworld.wolfram.com/DiscreteFourierTransform.html`. Visited in March 2012.

[34] Minh N. Do and Martin Vetterli. Wavelet-based texture retrieval using generalized Gaussian density and Kullback-Leibler distance. *IEEE Transaction on Image Processing*, 11(2):146–158, Februari 2002.

[35] G. Verdoolaege and G. Karagounis. Study of the potential of wavelet distributions for the prediction of disruptions at the JET tokamak. Presentation internship master in engineering physics at JET, August 2011.

[36] Bjørn K. Alsberg, Andrew M. Woodward, and Douglas B. Kell. An introduction to wavelet transforms for chemometricians: A time-frequency approach. *Chemometrics and Intelligent Laboratory Systems*, 37(2):215 – 239, 1997.

[37] I. Fodor. A survey of dimension reduction techniques. `www.llnl.gov/casc/sapphire/pubs/148494.pdf`, May 2002. Visited in March 2012.

[38] R. Gutierrez-Osuna. Dimensionality reduction. Lecture presentation, `http://research.cs.tamu.edu/prism/lectures/iss/iss_l10.pdf`.

[39] P.J.F. Groenen and M. van de Velden. Multidimensional scaling. Econometric Institute Report EI 2004-15, Econometric Institute, Erasmus University Rotterdam, P.O. Box 1738, 3000 DR Rotterdam, The Netherlands, April 2004.

[40] Joshua B. Tenenbaum, Vin de Silva, and John C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.

[41] Sergios Theodoridis and Kostantinos Koutroumbas. *Pattern Recognition*. Elsevier Academic Press, second edition, 2003.

[42] J. Vega, A. Murari, G. Vagliasindi, G.A. Rattá, and JET-EFDA contributors. Automated estimation of L/H transition times at JET by combining bayesian statistics and support vector machines. *Nuclear Fusion*, 49:085023–1–085023–11, 2009.

[43] Christopher J.C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2:121–167, 1998. 10.1023/A:1009715923555.

[44] X. Pennec. Intrinsic statistics on Riemannian manifolds: Basic tools for geometric measurements. *Journal of Mathematical Imaging and Vision*, 25:127–154, 2006. 10.1007/s10851-006-6228-4.

[45] T.W. Anderson and I. Olkin. Maximum-likelihood estimation of the parameters of a multivariate normal distribution. *Linear Algebra and its Applications*, 70(0):147 – 171, 1985.

[46] Prof. dr. ir. O. Thas. Multivariate Data Analysis. Ghent University, Department of Applied Mathematics, Biometrics and Process Control,157 p., 2012.

[47] A. Murari J. Vega G. Van Oost G. Verdoolaege, G. Karagounis and JET-EFDA contributors. Modeling fusion data in probabilistic metric spaces: applications to the identification of confinement regimes and plasma disruptions. In *7th Workshop on Fusion Data Processing, Validation and Analysis*, Frascati, March 2012. Submitted to Fusion Science and Technology, 2012.

[48] Don H. Johnson and Sinan Sinanović. Symmetrizing the Kullback-Leibler distance. *IEEE Transactions on Information Theory*, 2001.

[49] Jonathon Shlens. Notes on Kullback-Leibler divergence and likelihood theory. 2007.

[50] G. Verdoolaege and G. Van Oost. Advanced learning in massive fusion databases: nonlinear regression, clustering, dimensionality reduction and information retrieval. In *38th EPS Conference on Plasma Physics*, June 2011.

[51] Karel Van Acoleyen. *Course on relativity*. University of Ghent, Faculty of Sciences, Department of Physics, 2011-2012.

[52] Eric W. Weisstein. Tangent space. `http://mathworld.wolfram.com/TangentSpace.html`. Visited in March 2012.

[53] Eric W. Weisstein. Diffeomorphism. `http://mathworld.wolfram.com/Diffeomorphism.html`. Visited in March 2012.

[54] Eric W. Weisstein. Azimuthal equidistant projection. `http://mathworld.wolfram.com/AzimuthalEquidistantProjection.html`. Visited in March 2012.

[55] O. Tuzel, F. Porikli, and P. Meer. Human detection via classification on Riemannian manifolds. In *CVPR '07 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1 –8, june 2007.

[56] F. Durodié, M. Nightingale, M.-L. Mayoral, A. Argouarch, T. Blackman, M. Gauthier, R. Goulding, M. Graham, S. Huygen, P. Jacquet, E. Lerche, J. Ongena, D. Van Eester, M. Vrancken, A. Whitehurst, and E. Wooldridge. Present status of the ITER-like ICRF antenna on JET. *AIP Conference Proceedings*, 1187(1):221 – 224, 2009.

[57] D. G. Whyte, T. C. Jernigan, D. A. Humphreys, A. W. Hyatt, C. J. Lasnier, P. B. Parks, T. E. Evans, M. N. Rosenbluth, P. L. Taylor, A. G. Kellman, D. S. Gray, E. M. Hollmann, and S. K. Combs. Mitigation of tokamak disruptions using high-pressure gas injection. *Phys. Rev. Lett.*, 89:055001, Jul 2002.

[58] Mahesh K. Varanasi and Behnaam Aazhang. Parametric generalized Gaussian density estimation. *The Journal of the Acoustical Society of America*, 86(4):1404–1415, 1989.

List of Figures

List of Tables