Uncertainty quantification and calibration of a photovoltaic plant model : warranty of performance and robust estimation of the long-term production.

Mathieu Carmassi

université PARIS-SACLAY

AgroParisTech
INSTITUT DES SCIENCES ET INDUSTRIES DU VIVANT ET DE L'ENVIRONNEMENT
PARIS INSTITUTE OF TECHNOLOGY FOR LIFE, FOOD AND ENVIRONMENTAL SCIENCES

EDF

NNT : 2018SACLA042

Thèse de doctorat

Uncertainty quantification and calibration of a photovoltaic plant model: warranty of performance and robust estimation of the long-term production

Thèse de doctorat de l'Université Paris-Saclay
préparée à AgroParisTech (l'Institut des sciences et industries du vivant et de l'environnement)

Ecole doctorale n°581 Agriculture, Alimentation, Biologie, Environnement, Santé (ABIES)
Spécialité de doctorat : Mathématiques appliquées

Thèse présentée et soutenue à Paris, le 21 Décembre 2018, par

MATHIEU CARMASSI

Composition du Jury :

Liliane Bel Professeur, AgroParisTech (MIA Paris)	Président
Amandine Marrel Ingénieur-Chercheur, CEA Cadarache (DER/SESI)	Rapporteur
Olivier Roustant Professeur, École des Mines de Saint-Étienne (LIMOS)	Rapporteur
Luc Pronzato Directeur de Recherche, CNRS (I3S)	Examinateur
Éric Parent IGPEF, AgroParisTech (MIA Paris)	Directeur de thèse
Pierre Barbillon Maître de Conférence, AgroParisTech (MIA Paris)	Co-directeur de thèse
Matthieu Chiodetti Ingénieur, EDF (TREE)	Co-encadrant
Merlin Keller Ingénieur-Chercheur, EDF (PRISME)	Invité

REMERCIEMENTS

Je voudrais avant tout remercier mon directeur de thèse Éric Parent pour m'avoir guidé durant ces trois années de thèse. Ses qualités scientifiques et humaines ont contribué au très bon déroulement de cette thèse. Éric a su trouver un bon équilibre dans l'encadrement, en étant très présent au début lorsque le besoin s'en ressentait et en me laissant une certaine marge de manœuvre par la suite. Je tiens à souligner sa constante bonne humeur et son côté paternel qui m'a soulagé de la pression lorsque celle-ci se faisait trop importante.

Ensuite, je tiens à remercier mon co-directeur de thèse Pierre Barbillon pour m'avoir aiguillé scientifiquement durant ces trois années. J'ai pu bénéficier de ses grandes qualités scientifiques à n'importe quel moment et les échanges que nous avons eus se sont toujours révélés très productifs. Son perfectionnisme m'a aussi amené à ne rien négliger que ce soit au niveau rédactionnel ou scientifique. J'ai pu apprécier sa personnalité au quotidien, que ce soit pendant les soirées à Rochebrune, pendant les réunions de travail ou pendant les pauses-café dans le bureau des doctorants. Une chose est sûre c'est que les jeux de mots quasi quotidiens vont me manquer.

Je tiens aussi à adresser mes remerciements à Merlin Keller. Il est indéniablement la raison pour laquelle j'ai pu en arriver là. Sa constante bonne humeur, sa modestie et ses qualités scientifiques ont été la source d'une collaboration qui dure depuis mon stage de fin d'année d'école d'ingénieurs. J'ai pu le solliciter autant de fois que je le souhaitais pendant la thèse et même si je ne travaillais pas sur le même site, il a toujours été là pour m'aider. À chaque réunion que nous avons eue, j'ai pu retrouver chez lui un juste recul sur les problèmes industriels avec une grande connaissance mathématique.

Je remercie également Amy Lindsay qui m'a encadré avec une grande qualité jusqu'à sa mutation qui est intervenue au bout d'un an de thèse. C'est aussi pour cela que je remercie grandement Matthieu Chiodetti qui s'est retrouvé avec ma thèse sur les bras. Son encadrement a été d'une très grande qualité et il a su m'orienter sur les besoins industriels précis d'EDF et son expertise dans le domaine m'a permis de prendre du recul sur les résultats que l'on pouvait avoir. De manière générale, j'ai pu tisser avec chacun des membres de mon encadrement des liens d'amitié qui se sont retranscrit dans des conférences ou écoles d'été (Rochebrune, ETICS,...) où nous étions parfois amenés à sortir du cadre professionnel.

J'adresse également mes remerciements à Amandine Marrel et Olivier Roustant pour avoir accepté de rapporter ma thèse. Le travail sur les rapports a grandement été apprécié. Je remercie aussi Liliane Bel d'avoir présidé ma soutenance de thèse et Luc Pronzato d'avoir accepté de faire partie de mon jury en tant qu'examinateur.

Mes prochains remerciements seront pour mes collègues d'EDF. Même si le site des Renardières se trouvait très loin de mon domicile, c'était toujours avec un grand plaisir que je retrouvais tout le groupe. J'ai toujours apprécié les échanges, à table ou pendant les pauses, que nous avons eu. Je remercie tout particulièrement mon laboratoire MIA d'AgroParisTech qui m'a apporté énormément durant cette thèse. Au-delà des compétences mathématiques dont j'ai pu bénéficier, j'ai pu tisser des liens d'amitiés forts avec d'autres doctorants. Merci Timothée, Marie P, Pierre G (déjà vieux docteur), Félix, Rana, Paul, Anna, Marie C, Loïc d'avoir animé, à un moment ou un autre, le bureau des doctorants. Les afterworks à la Montagne, au Vieux Chêne, ou dans la rue Mouffetard nous ont permis de décompresser avec des journées de travail bien chargées.

Je tiens bien entendu à remercier toute ma famille : ma mère Marie-Christine, mon père Patrick, mon frère Guillaume et ma grand-mère Jeanine. Je pense sincèrement que le cadre familial qu'ils ont établi m'a conduit là où je suis aujourd'hui et je n'aurais pas pu rêver d'un meilleur équilibre de vie.

Je tiens désormais à remercier tous mes amis qui m'ont épaulé de près ou de loin pendant la thèse. D'abord, je remercie Quentin Huchet qui a été dans la même galère que moi et avec qui j'ai partagé quelques écoles d'été qui sont devenues mythiques (Porquerolles, Roscoff, ...). Merci de m'avoir permis de rigoler profondément à des moments où j'en avais besoin, d'avoir tourné en dérision les situations compliquées, d'avoir toujours eu le mot marrant pour dédramatiser le contexte. Merci à Lambert pour les pauses printemps de Bourges ou pour les quelques week-ends pétanque, hors du temps, à Paris. Merci à Albin pour les pauses rugby au bar ou les brainstormings non fructueux qui se terminaient quasi systématiquement en bières/rugby. Je remercie toute la clique de l'IFMA : Tareck, Pilou, Vely, Alexia, ViVi, MiMi, Norman, Naf, Matthieu, Ludo pour les week-ends de retrouvailles ou les réveillons du nouvel an passés ensemble. C'était aussi agréable de se retrouver des soirs aux Lombards, Mac Bride, Halls Beer, et autres tavernes. Les moments pendant la coupe du monde sur cette petite place pleine à craquer resteront aussi des moments marquants.

La conclusion de ces remerciements ne peut concerner que la personne qui m'a épaulé durant ces trois dernières années. Je remercie Camille pour m'avoir soutenu et supporté comme elle l'a fait même dans les moments où je devenais insupportable. Merci de m'avoir poussé à faire des breaks qui m'ont permis de mieux repartir après. Merci pour ces voyages qui n'appartiennent qu'à nous et qui m'ont permis de m'évader le temps de quelques semaines. Merci de ne m'avoir jamais laissé tomber au fond du trou et de m'avoir poussé à repartir rapidement après. Merci pour ta joie de vivre et ta constante foi en moi qui me redonnait confiance dans les mauvais moments. Merci d'avoir rendu cette thèse plus facile.

CONTENTS

List of Figures

ACRONYMS

PV	Photovoltaic
EDF	Électricté De France
LCOE	Levalized Cost Of Energy
CSP	Concentrate Solar Power
CRE	Energy Regulation Commission
OPEX	Operational Expenditures
CAPEX	Capital Expenditures
UTC	Universal Time Coordinated
UQ	Uncertainty Quantification
V & V	Verification and Validation
SA	Sensitivity Analysis
HSIC	Hilbert-Schmidt Independence Criterion
OAT	One At a Time
EE	Elementary Effect
TSI	Total Sensitivity Index
iid	independent and identically distributed
LHS	Latin Hyperspace Sampling
LHD	Latin Hyperspace Design
EGO	Efficient Global Optimization
GP	Gaussian process
BLUP	Best Linear Unbiased Predictor
EBLUP	Empirical Best Linear Unbiased Predictor
EI	Expected Improvement
DOE	Design Of Experiments
MST	Minimum Spanning Tree
SFD	Space Filling Design
ESE	Enhanced Stochastic Evolutionary
PCA	Principal Component Analysis
MCMC	Monte Carlo Markov Chain
MLE	Maximum Likelihood Estimates
DAG	Directed Acyclic Graph
SMLE	Separated Maximum of Likelihood Estimation
MAP	Maximum A *Posteriori*
CV	Cross Validation
RMSE	Root Mean Square Error
CRAN	Comprehensive R Archive Network

RÉSUMÉ

Dans la plupart des industries, l'accès aux expériences de terrain peut s'avérer coûteux économiquement et très chronophage dans certains cas. En effet, lorsque des tests sur des structures très volumineuses sont à réaliser ou lorsque les phénomènes à observer dépendent du temps, ces essais deviennent alors des enjeux majeurs pour les sociétés qui les conçoivent. Des codes numériques, représentant les phénomènes physiques en jeux, sont alors conçus pour diminuer les coûts. Cependant, un code numérique n'est qu'une représentation de ce qu'est la réalité. Il doit être en accord avec les résultats expérimentaux. C'est pour cela que l'on ne peut pas dissocier le code numérique des expériences de terrain.

Les codes numériques présentent dès lors deux avantages. Ils sont, en effet, plus rapides pour obtenir des résultats que l'expérimentation réelle et ils représentent, notamment, un moindre coût pour les industriels. Cependant, pour que les codes soient les plus représentatifs possible de la réalité, les ingénieurs les ont développés et perfectionnés à un tel point que leurs exécutions prend un temps non négligeable. Ce temps est, certes, diminué par rapport à l'expérimentation réelle, mais il assez pour remettre en question l'utilisation de méthodes qui nous permettraient d'obtenir des résultats sur la fiabilité du système. De plus, le code peut aussi transporter une erreur (appelée erreur de code ou discrépance) qui représente les difficultés du code à reproduire le système physique réel.

À EDF (Électricité De France), des codes numériques sont utilisés dans tous domaines (le nucléaire, l'éolien, l'hydraulique, le photovoltaïque, etc...). Ces domaines font appel à des codes numériques basés sur la simulation de phénomène physique qui comprennent, notamment, la thermohydraulique, l'hydraulique, la neutronique, la mécanique des fluides, la mécanique continue, la mécanique vibratoire, l'écotoxicologie, la thermique, etc... Dans certains domaines, des codes dits d'"échelle" peuvent être utilisés comme les codes éléments ou volumes finis, ou des codes système dits 0D/1D. Les enjeux de l'utilisation de tels codes numériques se concentrent sur la sûreté des installations, l'environnement, la distribution ou la production. Dans le cas du photovoltaïque (PV), le code numérique peut être utilisé à bien des égards (les smarts grids, smart cities, les offres de service liés à l'autoconsommation, les études de dégradations, la physique du panneaux photovoltaïque, etc...). Dans la thèse, nous nous intéressons à deux cas particuliers dans l'utilisation des codes numériques. Le premier concerne l'établissement du business plan d'une centrale PV avant sa construction. En France, la CRE (Commission de Régulation de l'Énergie) lance un appel à projet pour la construction d'une centrale PV et les énergéticiens comme EDF doivent calculer les coûts d'un tel projet. Pour ce faire, il faut connaître les coûts de construction et de maintenance de la centrale puis la production sur sa durée de vie ce qui permet ensuite de fixer un prix de facturation pour l'électricité produite par cette centrale. Pour calculer la production totale d'électricité de la centrale, EDF utilise un code numérique, qu'il sait imprécis. Pour chiffrer le projet, EDF applique un coefficient qui tend à sous estimer la sortie du code pour diminuer les risques financiers. Pour augmenter la rentabilité du projet et diminuer ces risques financiers, il convient donc de connaître avec plus de précision quelles sont les incertitudes introduites dans le code numérique. Dans un deuxième temps, le code PV peut être utilisé à des fins de suivi de performances. En effet, lorsqu'une centrale est d'ores et déjà construite et que des données de production sont disponibles, le code numérique basés sur les données de production est utilisé pour rectifier les prédictions sur les années suivantes. Dans les deux cas, le calage de code numérique représente un enjeu majeur pour l'obtention de résultats plus complets qui permettent de prendre une

décision financière basée sur plus d'informations.

Le calage de code

Lorsque les données expérimentales de terrains sont prélevées, une erreur de mesure est à prendre en compte. En effet, les tolérances des capteurs et l'imprécision des outils de mesure créent un bruit additionnel. De plus, si l'on considère le phénomène physique comme une fonction déterministe ξ (Sacks et al., 1989) dépendant uniquement de variables dites de contrôle, l'équation suivante peut être écrite :

$$\forall i \in [\![1,\dots,n]\!]\ \boldsymbol{Y}_{exp_i} = \xi(\boldsymbol{X}_i) + \varepsilon_i, \tag{1}$$

où $\boldsymbol{Y}_{exp_i}$ est le i^{me} point de mesure parmi les n, ε_i est un bruit blanc Gaussien tel que $\varepsilon_i \overset{iid}{\sim} \mathcal{N}(0,\sigma^2_{err})$ (où tous les n ε_i sont choisis indépendamment et identiquement distribués), ξ représente le phénomène physique réel correspondant à la i^{me} mesure, et $\boldsymbol{X}_i$ le vecteur des variables de contrôle qui correspondent aux variables observées et non modifiables (comme les données environnementales par exemple).

Le calage de code permet de mieux quantifier et d'estimer les valeurs des paramètres en entrée de code par rapport à des données de terrain. Considérons un code numérique f_c qui possède deux types d'entrées: les variables de contrôle ($\boldsymbol{X}$, définies précédemment) et les paramètres ($\boldsymbol{\theta}$). Les paramètres sont généralement des constantes physiques implémentées dans les équations sous-jacentes au code numérique. L'hypothèse que le code représente parfaitement le phénomène physique réel à condition de connaître la "vraie" valeur de $\boldsymbol{\theta}$ est faite dans un premier temps. Ainsi, une nouvelle représentation des expériences de terrain peut être écrite comme il suit :

$$\mathcal{M}_1 \ :\ \forall i \in [\![1,\dots,n]\!]\ \boldsymbol{Y}_{exp_i} = f_c(\boldsymbol{X}_i,\boldsymbol{\theta}) + \varepsilon_i. \tag{2}$$

Cependant lorsque le code f_c est long à être exécuté, l'utilisation de méthodes statistiques sur le modèle précédent n'est pas envisageable. Afin de palier ce problème, Sacks et al. (1989) proposent de mettre en place un processus Gaussien en remplacement au code. Dans le cadre du calage Cox et al. (2001), ont eu l'idée d'introduire un processus Gaussien dans le modèle statistique :

$$\mathcal{M}_2 \ :\ \forall i \in [\![1,\dots,n]\!]\ \boldsymbol{Y}_{exp_i} = \boldsymbol{F}_c(\boldsymbol{X}_i,\boldsymbol{\theta}) + \varepsilon_i, \tag{3}$$

où F_c est un processus Gaussien définit tel que $F_c(\bullet,\bullet) \sim \mathcal{PG}\Big(m_S(\{\bullet,\bullet\},\{\bullet,\bullet\}), c_S(\{\bullet,\bullet\},\{\bullet,\bullet\})\Big)$ (avec m_S la moyenne du processus Gaussien souvent considérées comme une forme linéaire avec un vecteur de coefficient $\boldsymbol{\beta}$ à estimer et c_S la fonction de covariance du processus qui dépend d'une variance σ^2_S, d'un noyau de corrélation et d'un vecteur $\boldsymbol{\psi}$ qui représente les longueurs de corrélation dans le noyau). Cependant, comme il a été introduit précédemment une discrépance peut apparaître avec l'introduction du code numérique en remplacement du phénomène physique. Des articles comme Higdon et al. (2004), Kennedy and O'Hagan (2001) et Bayarri et al. (2007) suggèrent d'introduire cette discrépance et d'effectuer le calage en considérant celle-ci comme étant une réalisation d'un processus Gaussien. Le fait de considérer la discrépance comme un processus Gaussien permet de détecter et de quantifier les erreurs structurelles qui seraient présentes dans les données expérimentales. Si le code n'est pas coûteux et que l'on ajoute une discrépance le modèle devient alors :

$$\mathcal{M}_3 \ :\ \forall i \in [\![1,\dots,n]\!]\ \boldsymbol{Y}_{exp_i} = f_c(\boldsymbol{X}_i,\boldsymbol{\theta}) + \delta(\boldsymbol{X}_i) + \varepsilon_i, \tag{4}$$

où δ représente la discrépance telle que $\delta(\bullet) \sim \mathcal{PG}\Big(m_\delta(\bullet,\bullet), c_\delta(\bullet,\bullet)\Big)$. Dans un cadre où le code numérique est considéré comme coûteux, le modèle qui généralise les deux précédents peut s'écrire :

$$\mathscr{M}_4 \ : \ \forall i \in [\![1,\dots,n]\!] \ \boldsymbol{Y}_{exp_i} = \boldsymbol{F}_c(\boldsymbol{X}_i,\boldsymbol{\theta}) + \delta(\boldsymbol{X}_i) + \varepsilon_i. \tag{5}$$

Un des enjeux majeur du calage est l'estimation des paramètres. Plus le modèle est complexe et plus l'estimation des paramètres est compliquée. En effet, dans chaque modèle statistique vient s'ajouter, aux paramètres $\boldsymbol{\theta}$ du code, les paramètres dit de nuisances qui sont les variances σ^2_{err}, σ^2_S, σ^2_δ et les vecteurs ψ_S, ψ_δ. Cela complexifie l'estimation qui peut s'effectuer de plusieurs manières (par la méthode des moindres carrés, l'inversion directe, la régression quantile, etc...), mais qui est souvent réalisée de deux sortes : par maximum de vraisemblance (qui permet une estimation simple) ou par estimation bayésienne (si un besoin de régularisation est nécessaire). Le maximum de vraisemblance est utilisé notamment par Cox et al. (2001) pour effectuer l'estimation des paramètres de $\mathscr{M}_2$ et (Wong et al., 2017) ont étendu ces résultats au cas $\mathscr{M}_4$. En calage bayésien, deux méthodes s'opposent. Higdon et al. (2004) suggèrent d'effectuer une estimation *a posteriori* directement sur la vraisemblance complète (pour l'écriture des vraisemblance se référer à la section 3.2.2) alors que Kennedy and O'Hagan (2001) et Bayarri et al. (2007) propose une estimation en deux temps appelée "approche modulaire" par Liu et al. (2009). Cette méthode permet de séparer en deux la méthode d'estimation classique (utilisée par Higdon et al. (2004)), afin de réduire les temps de calcul. Il s'agit de trouver des estimateurs des paramètres de nuisances pour le processus Gaussien $\boldsymbol{F}_c$ et d'utiliser ces estimateurs dans la vraisemblance conditionnelle (plus de précision à la section 3.2.3).

A des fins de comparaisons, nous possédons un code numérique rapide qui reproduit la puissance instantanée du stand de test expérimental de la R&D d'EDF composé de 12 panneaux nommé "PVzen". Grâce à sa flexibilité et à sa rapidité, ce code nous permet de reproduire les différents cas évoqués précédemment. Suite à cette comparaison, plusieurs conclusions émergent. La première concerne l'importance de la discrépance. En effet le calage du code dans le cas $\mathscr{M}_1$ indique que la valeur la plus probable de la variance de l'erreur de mesure doit être bien plus élevée que ce que l'on pensait *a priori*. Cependant, cette valeur indiquée par le calage n'a aucun sens physique puisqu'elle est trop élevée pour être plausible. En effectuant le calage avec $\mathscr{M}_3$, on remarque que la valeur de la variance de l'erreur de mesure diminue pour être cohérente avec l'*a priori*. La sur-estimation de σ^2_{err} était due à la présence d'une erreur de code qui n'était pas prise en compte. Dans le cas où l'on considère le code comme coûteux, le plan d'expériences pour établir le processus Gaussien doit être limité (nous avons fait le choix d'un plan de 50 points). Le calage pour le modèle $\mathscr{M}_2$ donne alors une incohérence dans les valeurs estimées par rapport aux densités *a priori*. En effet le fait de prendre un processus Gaussien pas très performant dégrade la qualité du calage. Il est donc important de ne pas négliger la qualité du processus Gaussien précédent le calage. Dans cette perspective, l'application d'une méthode d'établissement d'un plan d'expériences basé sur le critère EI (Expected Improvment) (Damblin et al., 2018)), permet d'améliorer les résultats.

CaliCo

Le codage d'un package, appelé **CaliCo**, en **R** a été effectué pour le calage bayésien. Ce qui diffère avec les précédents packages mis en ligne sur le site du CRAN (Comprehensive R Archive Network), c'est que **CaliCo** se base sur les quatre modèles introduits précédemment et offre la possibilité à l'utilisateur d'utiliser au même titre chacun des modèles pour son code numérique et non pas uniquement $\mathscr{M}_4$. L'établissement du processus Gaussien peut aussi être automatiquement géré dans le package avec la possibilité d'effectuer un calage séquentiel (Damblin et al., 2018). La majeure partie des algorithmes MCMC (Monte Carlo par Chaînes de Markov) sont implémentés en **C++** ce qui rend leurs utilisations plus rapides. De plus, beaucoup d'outils de visualisation en **ggplot2** ont été ajoutés pour donner à l'utilisateur un rapide accès à des graphiques qu'il pourra lui même modifier à sa guise.

Suivi de performances

L'application du calage bayésien dans un cadre industriel est mis en application sur une centrale PV de grande taille. Nous possédons pour cela un autre code numérique qui est plus performant que celui que nous avions utilisé précédemment. Il est ainsi plus coûteux en temps de calcul mais plus précis pour estimer la puissance PV dans des conditions particulières (ombrages, effets de missmatch, etc...). Ce code produit en sortie une série temporelle sur un an de puissances instantanées. Le calage s'appliquait jusqu'à présent à des sorties scalaires. Higdon et al. (2008) a introduit le calage de code sur une sortie multidimensionnelle en réalisant notamment une projection sur d axes qui portent plus de 99% de l'information donné par une ACP (Analyse en Composante principales). La qualité d'une telle projection est étudiée ainsi que l'erreur faite en projetant les données sur les axes de l'ACP. L'ajout d'une discrépance a aussi été faite dans l'espace de l'ACP. Ce travail a abouti à l'écriture de deux modèles supplémentaires, dont celui avec discrépance, s'écrit:

$$\mathscr{M}_4' : \quad \boldsymbol{P}\boldsymbol{Y}_{exp} = \boldsymbol{P}_F \begin{pmatrix} f_{c_1}(\boldsymbol{\theta}) \\ \vdots \\ f_{c_d}(\boldsymbol{\theta}) \end{pmatrix} + \boldsymbol{P}_\delta \boldsymbol{\delta} + \boldsymbol{E}, \tag{6}$$

où $\boldsymbol{P}$ est la matrice de passage entre l'espace de l'ACP et l'espace physique, $\boldsymbol{Y}_{exp}$ sont les puissances relevées, f_{c_i} sont les projections émulées par des processus Gaussien sur les d axes de l'ACP, $\boldsymbol{P}_\delta$ la matrice qui comporte les $T-d$ derniers vecteurs propres contenus dans $\boldsymbol{P}$, la matrice $\boldsymbol{P}_F$ représente celle composée des d premier vecteurs propres contenus dans $\boldsymbol{P}$ et $\boldsymbol{E}$ le vecteur aléatoire des bruits de mesure. Ce modèle mis en application nous permet d'obtenir des résultats de calage du code numérique coûteux afin d'actualiser les prédictions de puissance sur les années suivantes.

Conclusions et perspectives

En conclusion, cette thèse se focalise sur les méthodes de calage bayésien. L'objectif était d'améliorer les connaissances en les paramètres afin de rendre l'estimation de la sortie du code plus robuste. Cela présente un intérêt économique fort puisque le fait de mieux estimer la puissance générée par une centrale photovoltaïque permet de moins prendre de risques financiers que ce soit lors de l'établissement d'un business plan ou lorsque l'on veut mettre à jour des prévisions de productions. Le calage bayésien permet, à partir de données de terrain, de mieux connaître la loi de probabilité des paramètres pour ainsi mieux prendre des décisions.

Dans cette thèse nous avons effectué une revue des principales méthodes présentes dans la littératures. À l'aide d'un code numérique peu coûteux en temps de calcul, nous avons pu mettre en place une comparaison des différents modèles introduits. Les conclusions que nous avons pu en tirer sont que l'introduction de la discrépance dans certains cas peut s'avérer importante. En effet, lors de l'estimation des densités *a posteriori* des paramètres, un décalage par rapport à la densité *a priori* de la variance du bruit de mesure peut s'effectuer. Cela pourrait avoir du sens si l'*a priori* n'était pas bon, cependant si l'augmentation de cette variance n'est pas justifiée d'un point de vue physique, il est possible que la non prise en compte de la discrépance fausse le résultats. De plus lorsque le code est coûteux et afin de réaliser les méthodes statistiques présentées, des émulateurs ou méta-modèles peuvent être utilisés. Là aussi, la précaution doit être d'usage lorsque l'on tente de reproduire le code numérique. En effet, nous avons constaté que si le méta-modèle n'était pas d'une qualité suffisante, cela crée un décalage dans les modes *a posteriori*.

Un travail de développement informatique a également été réalisé durant cette thèse. Le package **CaliCo** permet de réaliser un calage bayésien avec une multitude de code numérique ou partir de plan d'expériences. Il offre une

flexibilité du choix du modèle pour l'utilisateur. De plus, un codage des MCMC en **C++** permet d'accélérer les parties d'estimation qui sont chronophages. Des outils de visualisation basés sur **ggplot2** permettent aussi de tirer profit des réalisations du package sans difficultés.

Un dernier cas d'étude, basé sur des données de centrale photovoltaïque de grande capacité de production, a enfin été partiellement traité. Le code numérique utilisé dans ce cas est chronophage car il est basé sur des optimisations informatiques qui alourdissent le temps de calcul mais qui améliorent ses performances. La sortie de ce code est une série temporelle ce qui ne permet pas d'appliquer les différents modèles introduits précédemment. Ce problème a abouti à la formalisation de deux nouveaux modèles qui permettent, à partir d'une ACP, de trouver une sous espace vectoriel orthonormé dans lequel le calage peut être effectué. Cette formalisation a été appliquée au cas d'étude et nous a permis d'estimer les densités *a posteriori* des paramètres du code mais aussi de la variance des erreurs de mesures ainsi que la variance de la discrépance. La valeur de la variance de l'erreur de mesure se retrouve plus élevée que ce que l'on attendait *a piori* lorsque l'on utilise le modèle sans discrépance. Ce décalage est rattrapé par l'ajout de la discrépance et nous permet de conclure que l'apport de la discrépance a permis d'expliquer une erreur qui s'était retrouvée dans la variance de l'erreur de mesure et qui représentait une erreur de code.

Cependant, les aspects prédictifs du modèle utilisant l'ACP reste à être démontré. Une validation croisée aurait pu être effectué sur un mois de données. La nécessité d'ajouter une discrépance est très discutée dans beaucoup de papiers (Kennedy and O'Hagan, 2001; Bayarri et al., 2007; Higdon et al., 2004) et fait l'objet de la validation de modèle statistique basée sur le facteur de Bayes dans Damblin et al. (2016). Une validation statistique à l'aide d'un modèle de mélange peut aussi être envisagé comme le propose Kamary (2016). La remise en question sur la qualité du processus Gaussien en tant qu'émulateur de code interroge sur la nécessité de prendre un plan d'expériences bien fourni. Des travaux comme Damblin et al. (2018) permettent dans ce cas d'améliorer le plan d'expériences en vue du calage bayésien. Les méthodes permettant d'utiliser des codes à sorties multidimensionnelles dans le cadre du calage découle d'un article fondateur (Higdon et al., 2008) mais ne restent pas très développés en pratique. Un processus Gaussien multi-fidélité pourrait aussi être envisagé en remplacement du code à sortie multidimensionnelle et ainsi être intégré dans le calage de code.

INTRODUCTION

In many industrial fields, numerical experiments have become more and more popular over the last few years. Field experiments are often really expensive and getting results from the real phenomenon is quite long. To limit this investment, numerical simulations are run as a substitute for field experiments (Santner et al., 2013; Fang et al., 2005). As numerical simulations intend to be as close as possible to the physical system, they have been continuously improved. However, the development of computer processors did not catch up with this evolution and some numerical simulations are still greedy in computational time (Sacks et al., 1989). Moreover, a difference between the numerical code and experiments is often observed. If there exists a bias between the code and the reality, what is the uncertainty in using it as a proxy of the physical system? Bayarri et al. (2007) introduce the notion of validation which consists in comparing the code outputs to the field experiments. Such a task can be difficult to achieve since it needs expensive field experiments and outputs from a code, often long to run. All along this thesis, we will use the word "code" as a proxy for numerical code, sometimes also called numerical model, simulator or computational code and field experiment for real world experiment.

In that respect, EDF (Électricité De France) uses numerical simulations in many fields, in particular to estimate the power produced by a photovoltaic (PV) plant. This thesis is motivated by the uncertainty quantification and calibration of a numerical code that intends to estimate the power produced by a PV plant. In this introduction, we will present the economic context that explains the needs of EDF for such a study. Then, a brief explanation of how a PV panel works is given that is followed by details on the different physical models developed by EDF. Finally, the numerical codes, used as application cases in this thesis, are detailed and presented in the last section.

1.1 Economic issue

Due to global warming, new "fuel-free" technologies are increasingly being developed. EDF focuses its research on some of them which are, without being exhaustive, the photovolatic, wind turbines or concentrated solar power (CSP). In each field the same economic problems appear especially in the photovoltaic (PV) where more and more PV plants are built in France and all around the would. The goal for energy suppliers such as EDF is to be competitive in this new market. In France, the apparition of new plants is regulated by an entity called CRE (Energy Regulation Commission). The right to build and manage a new plant usually comes by winning a bidding call for

project. The business model of such a project is particularly based on a factor called the levelized cost of energy also named LCOE.

LCOE

The average minimum cost at which electricity must be sold in order to break-even over the lifetime of the project.

It can be written as:

$$LCOE = \frac{sum\ of\ actualized\ costs\ over\ the\ lifetime}{sum\ of\ electrical\ energy\ produced\ over\ lifetime} = \frac{\sum_{t=1}^{n} \frac{I_t + M_t}{(1+r)^t}}{\sum_{t=1}^{n} \frac{E_t}{(1+r)^t}}\ , \tag{1.1}$$

where I_t stands for the investment expenditures in the year t, M_t for the operations and maintenance expenditures in the year t, E_t for the electrical energy generated in the year t, r for the discount rate and n for the expected lifetime of the system or power station.

The costs are relatively well estimated. Based on previous experience, the operational expenditures (also named OPEX, which encompass the operations and maintenance expenditures) and the capital expenditures (also called CAPEX, which cover investment expenditures I_t) are approximated with a narrow credibility interval. The CAPEX represents the invested money for building a photovoltaic plant and are fixed costs, when the OPEX is the money spent to build and maintain the photovolatic plant and stands for the variable costs. However, to predict the total power generated over the lifetime of the plant, energy suppliers use a homemade or commercial numerical code. For the uncertainty quantification of the code, the actual methodologies used give credibility intervals around $\pm 8\%$ of the output of the code. A compromise is found between the risk and the price based on the estimated uncertainty. This compromise tends to have a LCOE secure for investors but it increases the price of the project. Other energy supplier companies, which could have more accurately predicted the power or have a more aggressive policy on the establishment of business plans, could win the project. The main stake is to better assess the credibility interval on the power produced so that even the estimation of the less power produced is better than the final power estimation of the competition.

Once EDF has won the call for project, the PV plant is built in the specific location. When its activation is effective, data of power production are recorded. After some time, EDF engineers can compare the estimation made in the business plan to the real power produced. If, as expected, the prediction is lower than real power gathered on the field, the electricity price based on the business plan is no longer adequate. A new simulation can be run, based on these new data, which better estimates the power produced for the next few years. The business plan can then be updated based on the new estimation. This operation is called performance monitoring.

In France the economical stakes are important because it is the fifth photovoltaic field in Europe. The estimation of the total production capacity is more than 400 GWp. Wp stands for Watt peak which represents the power delivered by a photovolatic panel or plant under nominal conditions (1000 W/m^2 of enlightening and a temperature of 25°C). The development deadlines are also shorter in photovolatic (3-4 years) than onshore wind turbine (7-9 years). The costs of the electricity produced by the photovoltaic is decreasing (it was more than 200 €/MWh in 2012 and it was equal to 55 €/MWh in 2018 for large scaled PV plants). So far, in France only 8,159 MW of PV capacity is installed. In 2018 EDF has announced a Solar Plan which aims to install 30 GW of photovoltaic power between 2020 and 2035. A major part of this 30 GW will be constituted of by large PV plants. The resources mobilized by the EDF Group are the identification of the land to be mobilized, the mobilization of the subcontracting chain and EDF's partners, the development of self-consumption offers, the cooperation with public authorities to

make large areas available, the development of an industrial model adapted to the challenge, etc... In that context EDF is looking to better control the uncertainties made by the production estimations to limit financial risks when establishing the project.

The aim of this thesis is to quantify the uncertainty of the numerical codes used by EDF. Based on Bayesian calibration, the main framework will be the performance monitoring, where recorded power data are available. Then, the new predicted power can then be compared to the estimated one in the business plan and quantify plausible earnings.

1.2 Physical phenomenon

The photovoltaic principle mainly lies in characteristics of the semiconductor material used for the cell which is, for most of the technologies developed up to now, the silicon. The energy contribution of the sun to the PV cell is visible at a quantum level. Indeed, the energy present in the light spectrum changes locally the energy levels of the silicon until the emission of an electron. A panel encompasses a high number of cells and a cell is composed of two semiconductor materials. One is called the p-type and the other one the n-type. They are linked by the p-n junction. The anode corresponds to the p-type and the cathode to the n-type. The anode includes an excess of holes (it is positively charged, because a hole is a lack of electron) and the cathode contains an excess of electron (negatively charged). Figure 1.1 illustrates this dipole which can be "electrically" modeled by a diode.

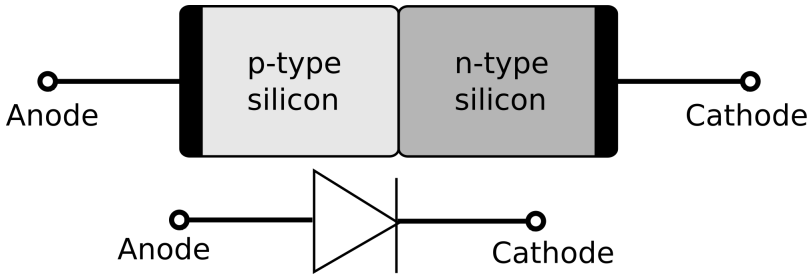

Figure 1.1: p-n junction and the equivalent electrical component (*source: Raffamaiden – CC By SA*).

The contact between both parts can lead to a displacement, by diffusion, of an electron toward a hole. The creation of an electron/hole pair can occur with a sufficient energy supply. When an electron/hole pair is created, the displacement of the electron is generating current. The energy needed to create such a displacement, is called the gap energy. It corresponds to the difference of the energy levels between conduction and valence bands. Figure 1.4 illustrates the different bands and the creation of the current. To facilitate this creation, a doping of the poles can be done. The p-side doping creates an electron deficiency to establish a new pseudo level higher than the valence band (*cf* Figure 1.3). Similarly, the n-side doping is an excess of electron production to set up a pseudo level lower than the conduction band (*cf* Figure 1.2).

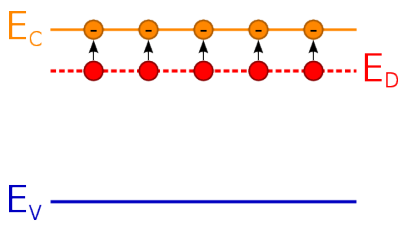

Figure 1.2: n side doping.

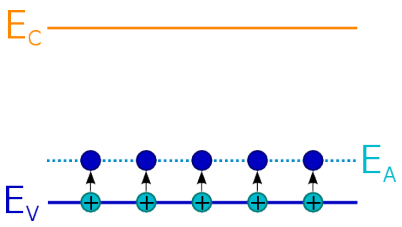

Figure 1.3: p side doping.

Once the photon (elementary particle particle brought here by the light) with enough energy has arrived on the cell, the creation of an electron/hole pair is done. Figure 1.4 illustrates the displacement, after the photon arrival,

from the n side to the p side of the electron.

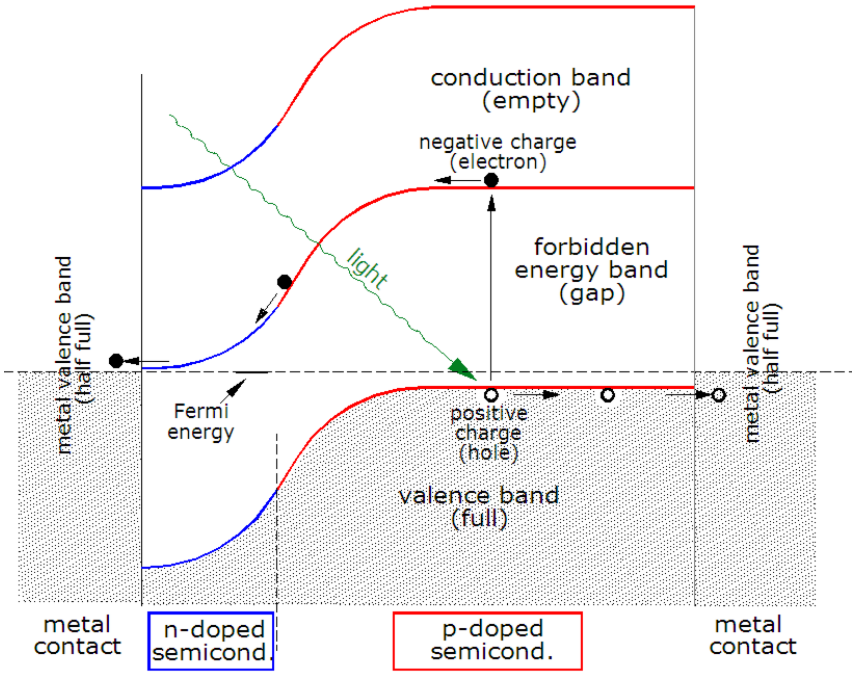

Figure 1.4: Displacement of an electron in the silicon (*source: Freshman404 – CC By SA*).

The silicon is a fragile material and needs to be protected from the environment. Anti-reflective coatings, transparent adhesive and glass cover are responsible of losses due to reflection of sun rays. The metal contacts, the gap energy, the probability to create an electron hole pair (also called *Quantum efficiency*) generate other losses. The efficiency of a panel, which is the total energy used to create electricity over the total energy arriving on the panel, stands between 15 and 20% according to manufacturers. As the global interest increases, technologies of PV cells and experiments are evolving. It is now possible to make a characterization in laboratory, very quickly, for a single panel. However, to get the total energy produced by a PV plant is a much longer and expensive experimental operation. That is why numerical codes based on a physical model have been developed.

1.3 Several modeling approaches

Several numerical codes, based on physical models have been developed to mimic the PV cell behavior. This section presents only models developed at EDF. What differentiates them from one another is their level of accuracy, especially when it comes to predict the power of a PV plant in constraining conditions (when shades or mismatch effects appear). If one is interested in modeling a PV system very quickly with a correct approximation, one can look into the first model described below. However, for a more accurate prediction in constraining conditions, one should refer to the second model detailed beneath.

1.3.1 A first simple model

First, for a good approximation of the power produced by a PV plant, a model based on physical equations is established. As the power produced by a panel is mainly proportional to the nominal power of the panel and the incident direct irradiation, the equations can be written explicitly without the use of complex computer optimization to solve

eventually challenging equations. To consider the losses described in Section 1.2, parameters are implemented in the model and can encompass the module photo-conversion efficiency or the module temperature coefficient for example (more details are given in Section 1.4.3). This model, that does not require any optimization to get the resolution of eventual differential equations, is quick to implement but presents some limitations.

1.3.2 Advanced electrical models

To model, more accurately, the PV system, a refined representation of the cell is achieved. The p-n junction can be "electrically represented" by a diode (Figure 1.1), so that an electrical representation can be set up to reproduce the power generated by a PV cell, panel or plant. Two different electrical schemes have been developed. The first drawn, on the left panel of Figure 1.5, has only one diode when the second one (on the right panel of Figure 1.5) takes two. The second diode is added because it takes into account the cases where the functioning conditions are more difficult (shades or low irradiations for example).

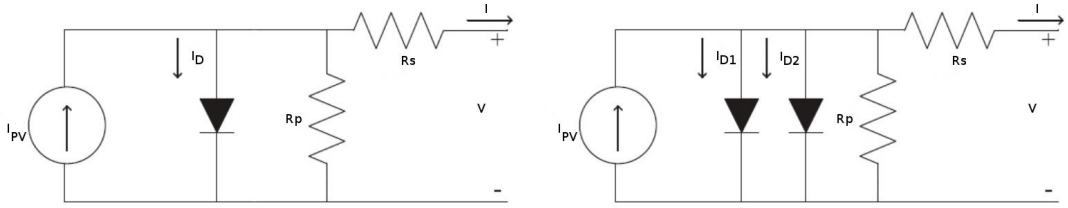

Figure 1.5: On the left panel, the electrical equivalence with 1 diode where I_{PV} stands for a photo-current that depends on the incident sun rays, I_D for the saturation current of the ideal diode, R_P the shunt resistance, R_S the series resistance representing losses proportional to I, I the current and V the voltage generated by the cell. On the right panel the electrical equivalence with 2 diodes where I_{D1} and I_{D2} stands for the saturation current of both ideal diodes.

To establish the performances of the PV cell, the power is the quantity looked for. Then, from both electrical schemes the voltage V needs to be expressed as a function of the current I. Kirchhoff's equations allow to solve the electrical model with one diode (Tian et al., 2012) and:

$$I = I_{PV} - I_D \left[exp\left\{ \frac{V + IR_s}{nV_t} - 1 \right\} \right] - \frac{V + IR_s}{R_p} , \tag{1.2}$$

where V_t and n are the thermodynamic potential and the quality factor of the diode. The solution of the electrical model with two diodes is given by:

$$I = I_{PV} - I_{D1} \left[exp\left\{ \frac{V + IR_s}{n_1 V_{t1}} - 1 \right\} \right] - I_{D2} \left[exp\left\{ \frac{V + IR_s}{n_2 V_{t2}} - 1 \right\} \right] - \frac{V + IR_s}{R_p} , \tag{1.3}$$

where V_{ti} and n_i are the thermodynamic potential and the quality factor of the diode i (Ishaque et al., 2011). A PV panel is generally made of around 60 to 72 PV cells and the voltage at the output of the panel is continuous. To be integrated in the network grid, it has to be transformed into an alternating voltage. To do so, an inverter is added at the output of a group of panel in a PV plant. The inverter finds the "best" functioning point (which is the maximum of IV, the available power) and generates the alternating power corresponding. To get this maximum power possible, the relation between I and V is studied. Figure 1.6 illustrates the typical behavior of the current as a function of the voltage for a module or a PV system.

The working conditions of the inverter are such that it physically performs an "optimization" to get to the

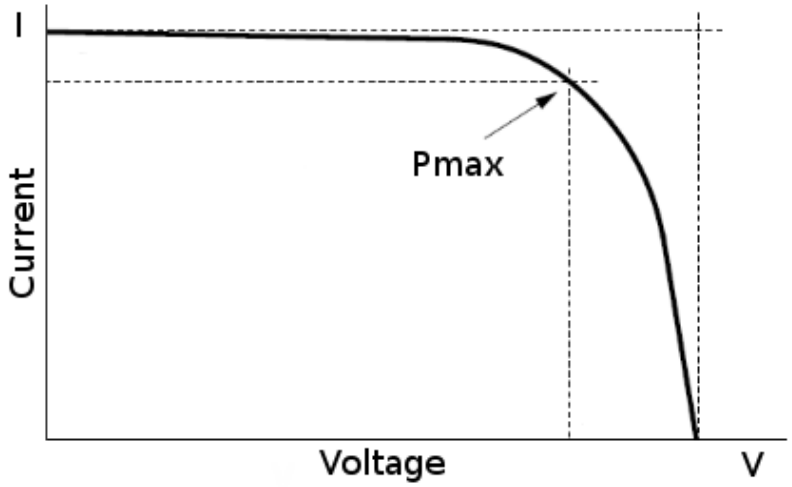

Figure 1.6: I/V curve of a toy example.

point *Pmax* (Figure 1.6) and generates the alternating voltage afterward. Having an explicit expression of I as a function of V is intractable regarding Equation (1.2) and Equation (1.3). Both equations are implicit and to solve them, computers usually run optimization operations which are time consuming. To bypass this burden, some approximation can be proposed to render explicit the equations. The use of Lambert's function W is required in many cases (Petrone et al., 2007; Ding and Radhakrishnan, 2008; Picault et al., 2010). It is defined as:

$$\forall x \geq -e^{-1} \quad W(x)exp\Big\{W(x)\Big\} = x \,, \tag{1.4}$$

and if x is near $+\infty$ or 0:

$$W(x) = \log x - log(log(x)) + \sum_{k=0}^{\infty}\sum_{m=1}^{\infty} c_{km}\frac{(\log(\log(x)))^m}{(\log(x))^{k+m}} \,, \tag{1.5}$$

with

$$c_{km} = \frac{(-1)^k}{m!}S[k+m,k+1] \,, \tag{1.6}$$

and $S[k+m,k+1]$ is the number of Stirling's cycle. Using the Lambert's function, I and V can be separated and Equation (1.2) now written:

$$I = \frac{R_p(I_{PV}+I_D)-V}{R_p+R_s} - \frac{nV_t}{R_s}W(\alpha(V)) \,, \tag{1.7}$$

with

$$\alpha(\boldsymbol{V}) = \frac{R_pR_s}{R_p+Rs}\frac{I_D}{nV_t}exp\Big\{\frac{R_p}{R_p+R_s}\frac{\boldsymbol{V}+R_s(I_{PV}+I_D)}{nV_t}\Big\} \,.$$

Newton or Halley's method allows to find the couple (I,V) maximizing the available power IV. However,the more elaborated Equation (1.3) cannot be changed to be rendered explicit. The question is to know whether one wants a numerical code fast to run or an accurate code. Is the approximation worth the saved computation time?

1.4 Numerical codes

In this section, we detail the general framework and notations of the numerical codes used in this thesis and we introduce the main sources of uncertainties that can be found in this context. Then we present, in details, the two numerical codes further used in this work.

1.4.1 General framework

A computer code generally depends on two kinds of inputs: variables and parameters. The variables represent the input variables (also called controllable variables in Higdon et al. (2008) or general inputs in Plumlee (2017)) which are set during a field experiment and can encompass environmental variables which can be measured in field experiments. In contrast, the parameters are generally interpreted as physical constants defining the mathematical model of the system of interest, but can also contain the so-called tuning parameters, which have no physical interpretation. They have to be set by the user to run the code and chosen carefully to make the code mimic the real physical phenomenon. The code can be mathematically represented by a function f_c. Let us note in what follows that, $\boldsymbol{\theta} \in \mathscr{Q} \subset \mathbb{R}^p$ to represent the parameter vector and $\mathbf{x} \in \mathscr{H} \subset \mathbb{R}^d$ which is the variable vector. The space $\mathscr{Q}$ is called the input parameter space and $\mathscr{H}$ the input variable space. The physical quantity of interest (QOI) is denoted by ζ and only depends on variables in vector $\boldsymbol{x} \in \mathscr{H}$ because the parameter vector $\boldsymbol{\theta}$ has no counterpart in field experiments.

A code output is then written as $f_c(\boldsymbol{x}, \boldsymbol{\theta})$ (considered as a deterministic code all along the thesis) whereas $\zeta(\boldsymbol{x})$ denotes the physical phenomenon for the same variable $\boldsymbol{x}$. This is of course an idealized formalization, in which we assume that the code variables $\boldsymbol{x}$ are exhaustive to describe the phenomenon of interest, in the sense that the quantity to be predicted can take a single deterministic value $\zeta(\boldsymbol{x})$ for a given $\boldsymbol{x}$.

1.4.2 Sources of uncertainties

In general, there are two main kinds of uncertainties considered: the epistemic and the statistical. The statistical uncertainty represents the random fluctuations of the input variables, and the associated measurement errors and the epistemic uncertainty comes from the uncertainty on parameters, that one could in principle know but does not in practice. The latter can be estimated but can also be reduced as the number of experiments increases. The uncertainties relative to the numerical code are then epistemic. The code is deterministic so no variability is visible between two launches. In this thesis, we focus only on the epistemic uncertainties that are detailed below.

The numerical code takes two inputs that are uncertain: $\boldsymbol{\theta}$ and $\boldsymbol{X}$. In the calibration framework, only the uncertainty on $\boldsymbol{\theta}$ is considered because we do not know the true value of $\boldsymbol{\theta}$ and we need to adjust it.

In the PV plant context, $\boldsymbol{\theta}$ represents physical constants or manufacturer values that are carrying uncertainty. Indeed, the building process of a PV panel encompasses tolerances at each step of the fabrication. At the end of the chain, the parameter, that characterizes the nominal power of the panel for example, might be altered. The input variables $\boldsymbol{X}$ represent mainly meteorological data. These are also carrying uncertainty because in both, prediction or performance monitoring, contexts, $\boldsymbol{X}$ is averaged from previous data where modification due to global warming is added.

This thesis focuses only on parametric uncertainties, because the main aim is to calibrate the parameter vector $\boldsymbol{\theta}$ given a data set, the uncertainty of the input variable being out of the scope of this study. EDF experts judge that

parameter uncertainty and input variable uncertainty are each responsible for about 4% of the output variability.

1.4.3 Python code

Code

The Python code has been developed based on the first physical model described in Section 1.3.1. It implements the physical equations that encompass production approximation estimation but also the losses relative to the panel. As no optimization are needed in this code, it runs very fast (about $36\mu s$ each run). The code, that does not take into account the inverter, produces an estimation of the instantaneous power. That means:

$$\begin{aligned} f_c : \mathscr{H} \times \mathscr{Q} &\to \mathbb{R} \\ (\boldsymbol{x}, \boldsymbol{\theta}) &\mapsto y \, . \end{aligned} \tag{1.8}$$

The code depends on some parameter vector $\boldsymbol{\theta}$ and input variables $\boldsymbol{x}$ detailed as follows: $\boldsymbol{\theta} = \begin{pmatrix} \eta \\ \mu_t \\ n_t \\ a_l \\ a_r \\ n_{inc} \end{pmatrix}$ and $\boldsymbol{x} = \begin{pmatrix} t \\ L \\ l \\ I_g \\ I_d \\ T_e \end{pmatrix}$.

The physical meaning of the parameters $\boldsymbol{\theta}$ is explained below (Duffie and Beckman, 2013):

- η: module photo-conversion efficiency in nominal test conditions ($1000W/m^2$, 25°C),
- μ_t: module temperature coefficient (the efficiency decreases when the temperature rises) in $\%/°C$,
- n_t: reference temperature for the normal operating conditions of the module in $°C$,
- a_l: reflection power of the ground (albedo),
- a_r: describes the transmission of the radiation as a function of the incidence angle of solar rays, which depends on optical properties and the cleanliness,
- n_{inc}: transmission factor for normal incidence.

The input variables $\boldsymbol{x}$ contain all measurable data:

- t: the UTC time since the beginning of the year in s,
- L: the latitude in °,
- l: the longitude in °,
- I_g: global irradiation (normal incidence of the sun rays to the panel) in W/m^2,
- I_d: diffuse irradiation (horizontal incidence of the sun rays to the panel) in W/m^2,

- T_e: ambient temperature in $°C$.

Note that temporal aspects are taken into account through the input variables. We do not consider any delay in the PV reaction to the forcing conditions. Time t indicates here a snap shot corresponding to the instant when the power has to be computed. This code only focuses on a specific time and if the evolution of the power over a day is what we look for, a repetition over the specific durations has to be made. This operation has to consider the number of time steps available. For example, if 300 configurations of $\boldsymbol{x}$ are accessible for one day, the code will have to be executed 300 times to have the power evolution over a day. For the rest of this thesis, we will denote the code output referring to the i$^{\text{th}}$ time step by $f_c(\boldsymbol{x}_i, \boldsymbol{\theta})$ and by $f_c(\mathbf{X}, \boldsymbol{\theta})$ the code outputs corresponding to the whole time frame contained in matrix $\mathbf{X}$.

Experimental data

The Python code has been created to reproduce the instantaneous power of a test stand at EDF called "PVzen". The stand is a group of 12 panels connected together. Data over two months are available at a time step of $10s$: August and September 2014. Figure 1.7 is an example of data gathered on the stand. Note that on the left panel of Figure 1.7, several days have no production. It is mainly because of the sensor malfunctions and data need to be cleaned before being used. The power collected on the stand is the one before the inverter and physically matches with the one simulated by the numerical code introduced in Section 1.4.3.

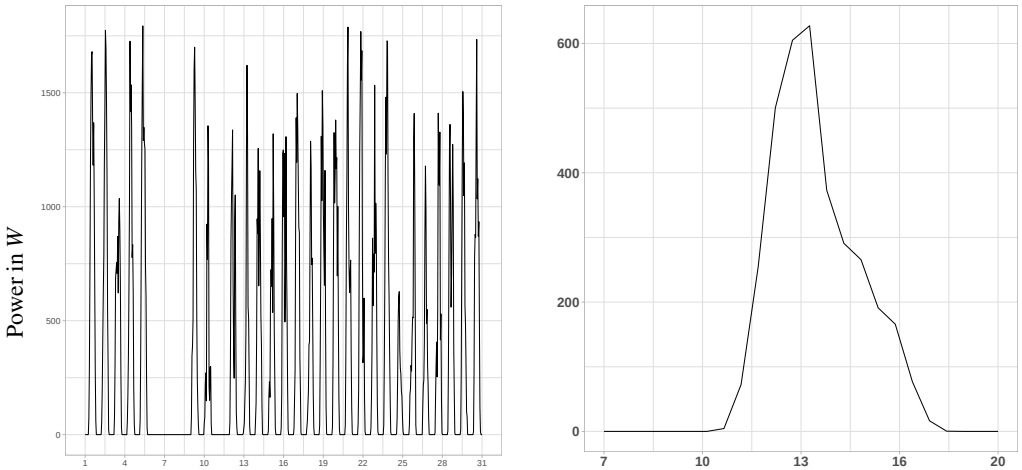

Figure 1.7: The power production by PVzen for August 2014 (on the left) and the power production averaged by hour for August 25$^{\text{th}}$ 2014 (on the right).

1.4.4 Dymola code

Code

The code Dymola is based on the "electrical" modeling introduced in Section 1.3.2, especially the physical model with one diode. Dymola is a modeling and simulation environment based on the language Modelica. This code has been developed for a specific PV plant that EDF maintains. It implements the shades that appear in a big plant configuration but also takes into account the mismatch effects. The mismatch effects are when the panels from a PV plant do not possess the same nominal power value. The inverter has to find the minimal

one and not the averaged. Mismatch effects is also a concern when one or several cells are shaded but not the whole panel. Shunt resistances are then activated so this part of the panel does not affect the panel overall production.

This code is then much longer to run than the previous one (about 20s for each call) but the output is a temporal trajectory over one year of the instantaneous power with the time step of 900s. This means that, for n points in the trajectory:

$$\begin{aligned} f_c : \mathscr{Q} &\to \mathbb{R}^n \\ \boldsymbol{\theta} &\mapsto y \ . \end{aligned} \tag{1.9}$$

This numerical code does not take $\boldsymbol{X}$ as input variables because they are implicitly implemented in Dymola. As a matter of fact, $\boldsymbol{X}$ represent the meteorological, the mismatch data and the projected rays files for one year corresponding to the n points produced by f_c. The mismatch and the projected rays files are input data that give the information of the mismatch effects and the shades on the PV plant panels we are focusing on. In these conditions, the output power is more complex to determine. The parameter vector $\boldsymbol{\theta}$ takes 26 components that we will not detail here. The parameter vector encompasses those which have an electrical meaning such as I_{pv}, R_p or R_s of Figure 1.5 (on the left panel) but also those which characterize the inverter. That means the output given by the Dymola code corresponds to the power after the optimization performed by the inverter.

Experimental data

The data available for the Dymola code is the power gathered during one year 2015 (sometimes data are partially collected). Data are scaled in Figure (1.8) for confidentiality matters. Identically as in Figure 1.7, there is some days where the production is null. These issues are common and also correspond to recording errors. Figure 1.8 represents the temporal series given by the PV plant for the year 2015.

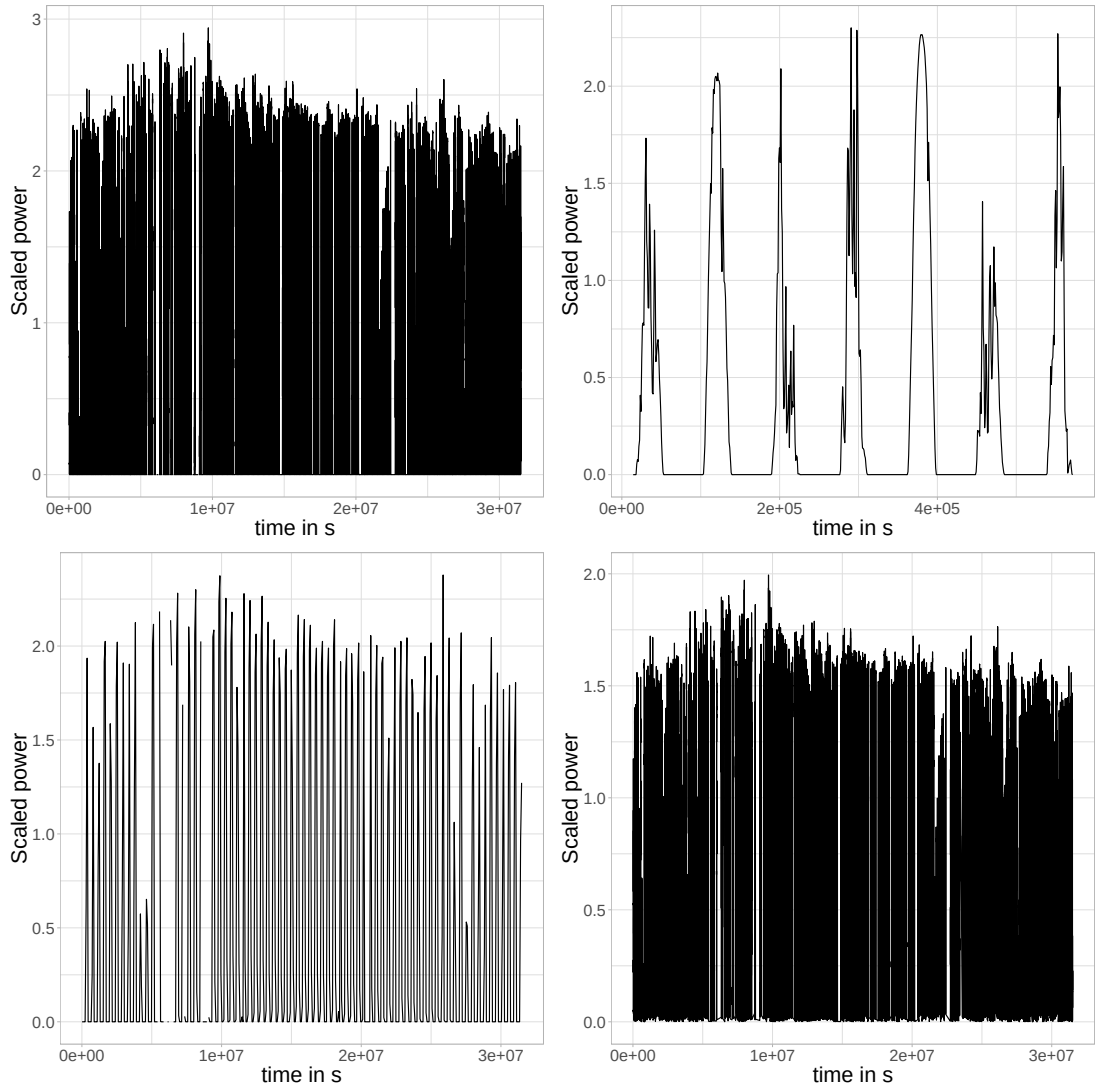

Figure 1.8: On the top left, the original scaled power production gathered on the the PV plant during the year 2015. On the top, right the same data but only on the first week. On the bottom left, the original data but averaged by hour. On the bottom right, only the positive power is kept among the origin data.

1.5 Thesis organization

This thesis presents the work done on Bayesian calibration especially conditioned by the two different application cases detailed above. First, Chapter 2 recalls the main tools in sensitivity analysis, design of experiments, principle component analysis, Monte Carlo Markov chains and Gaussian processes for a good understanding of Bayesian Calibration. Chapter 3 gives a state of the art of Bayesian calibration methods. This chapter uses the application case of the Python code to illustrate and compare the different statistical models that are existing. Then, Chapter 4 presents a package, called **CaliCo**, that completes Bayesian calibration in **R** which has been developed in the frame of this thesis. Chapter 5 illustrates a comprehensive industrial study of calibration using the Dymola code and data from a real PV plant. The document then concludes with a discussion and perspectives to be explored.

CHAPTER 2

STATISTICAL TOOLS FOR NUMERICAL CODE CALIBRATION

Uncertainty Quantification (UQ) in an industrial context has become important over the last few years. All along the process in an industrial cycle, from the research to the in-service and maintenance, UQ has an equivalent impact on business and risk reliability. However, the procedure for establishing the impact of the variability of several quantities on the output of interest has to be performed in multiple steps. From the identification of which parameter is responsible for the most of the output variation to the propagation of uncertainty, there is several steps that are detailed in Rocquigny (2009). Figure 2.1 is a graphical representation of the main steps in the UQ in an industrial context:

- Step A is the problem specification. An identification of the quantities of interest, input variables, input parameters, and of the numerical codes have to be done.

- Step B is the quantification of the uncertainty sources. Some components of the input variables or of the input parameters might be randomly distributed as mentioned in Section 1.4.2. Their variabilities are identified and the distribution densities determined in this step.

- Step B' is the V& V step (Verification and Validation) which can be followed by calibration of the numerical code.

- Step C is the propagation of the uncertainty achieved through the code. The distribution of the quantity of interest is looked for and particularly its moments (generally the expectancy or the variance), quantiles or modes.

- Step C' is the sensitivity analysis. The variables or parameters can be sorted in order of importance to identify which variable or parameter has the most responsibility for the variability of the quantity of interest.

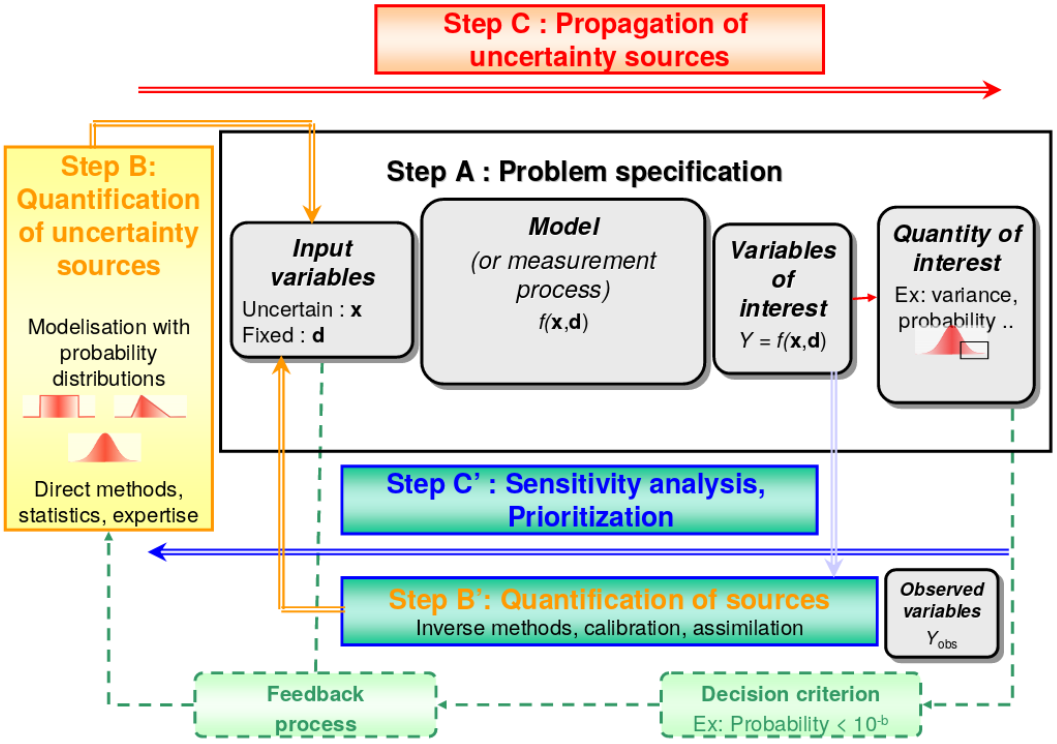

Figure 2.1: Major steps in uncertainty treatment for industrial matters (*source: Bertrand Iooss – ENBIS-EMSE 2009 Conference*).

This thesis focuses essentially on step B' and code calibration. However, before running any calibration on a numerical code, preliminary studies have to be conducted. This chapter aims to introduce some of them in order to help the reader in understanding the main steps of code calibration in an industrial context. First, two different methods of sensitivity analysis are presented. The second section provides theoretical developments on Gaussian processes that can be used as much in machine learning as in modeling some structural error. The third section introduces the aspects of design of experiments and recalls some of the major tools used in the thesis. Then, the two last sections present, theoretically, the principal component analysis and Monte Carlo Markov Chains techniques that are useful for a complete understanding of the extensions used in the further work.

2.1 Sensitivity analysis

The quantification of the influence of each input parameter on the output is the task performed by sensitivity analysis. It is interesting to access such an information because if the numerical code is time consuming, it can be helpful to focus on a reduced number of parameters. In that sense, sensitivity analysis (SA) can be considered as a prerequisite for model building in any setting (Saltelli et al., 2000). SA has a different meaning in different contexts. For an engineer, it could mean to move each component of the input parameter vector at the same range and compare the impacts on the quantity of interest. For a statistician, SA is the study of the variation of the distribution of the density of the quantity of interest according to the change of specific parameters.

There are two major categories in SA: global and local methods. Local SA tends to quantify the impact of a parameter around reference values. It is mainly done with partial derivatives. This kind of analysis is useful when one is interested in understanding the behavior of the physical model nearby these values. However, these methods do not allow to study the effect of the input parameters on the output when they have an important area of uncertainty. Contrary to local SA, global SA aims to study the variability of the quantity of interest driven by input parameters variation on all their area of uncertainty. In this section, we present only global SA methods mainly because in this thesis we are interested on the overall impact of the parameters on the variability of the quantity of interest.

Global SA can be performed in several different ways. First, screening methods allow to explore quickly the variability of the output, given by a numerical code, induced by a variation of the, potentially large, input parameter vector. The importance criterion in such methods is the amplitude of the variations of the code output obtained for different input parameter values. Screening methods can also be categorized but we only focus on Morris method Section 2.1.1 (Morris, 1991) because it requires less assumptions about the model compared to the other screening methods. These methods only allow to identify the non-influent parameters. If one is interested in classifying by order of importance the parameter influence, sensibility indices can be used. Sensibility indices are defined as a measure of the influence of an input parameter on the variability of the output. If the chosen measure of importance is the variance, then the indices are called Sobol indices (Sobol', 1990) and are presented in Section 2.1.2. However, these methods require a lot of code calls and if the code is time consuming they become quickly intractable. To estimate, with a limited computational impact, sensitivity indices, smoothing methods or surrogate of the code function have been developed. For example, Sudret (2008) demonstrates that Sobol indices straightforwardly arise from the chaos polynomial decomposition. Kriging surrogate also allows to obtain analytically the sensitivity indices formulation as it is shown in Oakley and O'Hagan (2004); Marrel et al. (2009). Da Veiga (2015) also introduced global SA methods which use dependence measures as the mutual information, the distance correlation or the Hilbert-Schmidt Independence Criterion (HSIC). Then, De Lozzo and Marrel (2016) have extended this work for a screening purpose and allow to decrease the computational time burden.

2.1.1 Morris method

Morris method aims to evaluate the influence of each input parameter by considering the impact of its variation on the output considering the other ones as constant. This way to operate consists in moving each parameter one at a time ("One At a Time" or OAT method). Then, the input parameter space is discretized on a grid.

Sampling space

Since the parameter range values are not all of the same order, the sampling design of the Morris method is standardized over the interval $[0,1]$. Thus, for p parameters, the sampling plan will be contained in a hypercube $\omega = [0,1]^p$. To generate a comparable variation for each parameters, a step δ is defined. For each parameter, the

sampling plan is divided in Q levels such as $D = [0, \frac{1}{Q-1}, \frac{2}{Q-1}, ..., 1]^p$ (where D stands for the sampling grid). Each axis is split into $Q-1$ equals sections. Usually, the step δ is chosen equally as the sampling step $\frac{1}{Q-1}$ but one can also pick a $k \in \mathbb{N}^{+*}$ such as $\delta = \frac{k}{Q-1}$. Figure 2.2 illustrates these variations. To define them, a point is randomly selected on the grid D. Then, a sign is also randomly selected (practically a Bernoulli random variable) to indicate the direction of displacement. The new point being found, a new random test on the sign of δ is run to move in the second direction. This procedure is repeated $p-1$ times and is called a trajectory. In its paper Morris (1991) advocates to select $\delta = \frac{Q}{2(Q-1)}$ and Q even such as not to favor any area of the space. However, the step of such selection might imply some lost of information on the parameter. For the rest of the thesis, the choice of $\delta = \frac{1}{Q-1}$ will always be made.

Let us define a toy function f_t such as:

$$\begin{aligned} [0,1]^2 &\mapsto \mathbb{R} \\ (\theta_1, \theta_2) &\to f_t(\theta_1, \theta_2) \\ \theta &\to f_t(\theta). \end{aligned} \tag{2.1}$$

The parameter vector θ can be extended to a vector with p component as $\theta = \begin{pmatrix} \theta_1 \\ \vdots \\ \theta_p \end{pmatrix}$ (if the function takes p input parameters as it is the case for f_c). If we choose a sampling level of $Q = 9$, Figure 2.2 illustrates the grid obtained and 4 different trajectories for f_t.

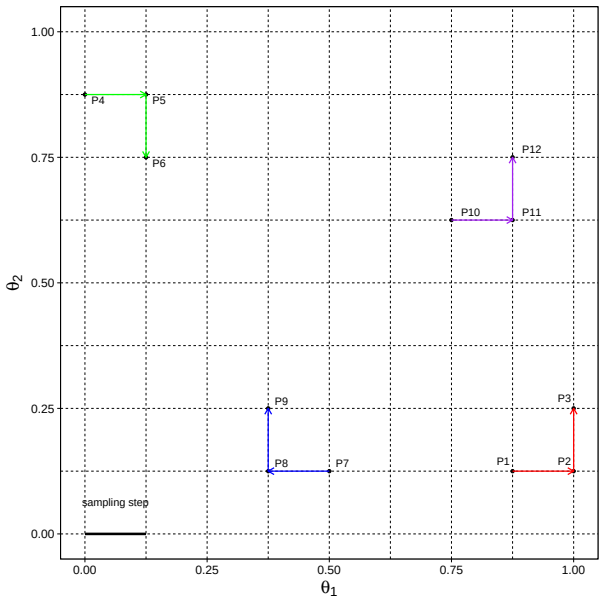

Figure 2.2: Sampling grid on the scaled space.

Let us call r the number of trajectories accomplished (Figure 2.2, $r = 4$). In his paper, Morris (1991) advices not to take a too high r. The aim is to screen the input parameter space with some displacements, not to generate a design of experiments representative of the original space (some of them are developed in Section 2.3). Note that if one of the chosen point is taken on the boundary of the grid, then the sign of δ is not randomly chosen. Indeed, to stay in the input parameter space, the sign is set such as the displacement stays in the grid.

Elementary effects

These variations of each parameter are quantified by elementary effects (EE) (Morris, 1991). For the function f_c defined on $[0,1]^p$, the EEs can be written as:

Elementary effects

$$\Delta_k^{(i)} = \frac{f_c(\theta_i + \varepsilon_k^{(i)}\delta) - f_c(\theta_i + \varepsilon_{k-1}^{(i)}\delta)}{\delta \; \varepsilon_k^{(i)}(k)}, \tag{2.2}$$

where:

$$\varepsilon^{(i)} = \begin{pmatrix} s_1 & \dots & s_1 & \dots & s_1 \\ \vdots & \ddots & & & \vdots \\ 0 & & s_k & & s_k \\ \vdots & \ddots & & \ddots & \vdots \\ 0 & \dots & 0 & \dots & s_p \end{pmatrix}^{(i)},$$

with $\epsilon_0^{(i)} = (0,\dots,0)_{(i)}{}^T$ and where $\varepsilon_k^{(i)}$ stands for the k^{th} column vector of the matrix $\varepsilon^{(i)}, \forall i \in [\![1,r]\!]$ $\varepsilon_k^{(i)}(k) = s_k^{(i)}$, $\varepsilon^{(i)}$ possesses p lines with $s_k^{(i)} = \pm 1 \; \forall k \in [\![1,p]\!]$, $\theta_i = (\theta_1,\dots,\theta_p)_{(i)}^T$ is the i^{th} point in the design, and $\Delta_k^{(i)} f_c$ is the elementary effect of the k^{th} parameter at the iteration i and δ stands for the step.

In the 2 dimensional example (Figure 2.2), P_2 coordinates are given by $f_t\begin{pmatrix}\theta_1+\delta\\ \theta_2\end{pmatrix}$ and P_1 coordinates by $f_t\begin{pmatrix}\theta_1\\ \theta_2\end{pmatrix}$. The direction is positive so: $\varepsilon_1 = \begin{pmatrix}1\\0\end{pmatrix}$. From this example, the elementary effect for the first trajectory and for θ_1 is defined by:

$$\Delta_1^1 = \frac{f_t\left(\begin{pmatrix}\theta_1\\ \theta_2\end{pmatrix} + \begin{pmatrix}1\\0\end{pmatrix}\delta\right) - f_t\left(\begin{pmatrix}\theta_1\\ \theta_2\end{pmatrix} + \begin{pmatrix}0\\0\end{pmatrix}\delta\right)}{\delta} = \frac{f_t\begin{pmatrix}\theta_1+\delta\\ \theta_2\end{pmatrix} - f_t\begin{pmatrix}\theta_1\\ \theta_2\end{pmatrix}}{\delta}.$$

This matches with the finite difference on θ_1 and for a step δ. Visually in Figure 2.2, it coincides with the difference of the values of f_t at the unnormalized points P_2 and P_1 and divided it by δ.

For r Morris trajectories, there will be $N = r(p+1)$ calls to the function f_c. Two indices computed with the EEs allow to compare the effect of the variation of each parameter on the output. The first is the expectancy of the EEs for each parameter (Faivre et al., 2013; Saltelli et al., 2000, 2004; Morris, 1991):

$$\widehat{\mu_k} = \frac{1}{r}\sum_{i=1}^{r} \Delta_k^{(i)} f_c. \tag{2.3}$$

If f_c is periodic, the mean of the EEs is near zero. This does not mean that the parameter has no impact, that is why it is better to use the mean of the absolute value of the EEs μ^* (Faivre et al., 2013; Saltelli et al., 2000, 2004):

the estimated expectancy $\widehat{\mu^*}$

$$\widehat{\mu_k^*} = \frac{1}{r}\sum_{i=1}^{r} | \Delta_k^{(i)} f_c | . \tag{2.4}$$

The second index is the standard deviation of the EEs (Morris, 1991; Saltelli et al., 2004):

the estimated standard deviation $\widehat{\sigma_k}$

$$\widehat{\sigma_k} = \sqrt{\frac{1}{r-1}\sum_{i=1}^{r}(\Delta_k^{(i)} f_c - \mu_k)^2}. \tag{2.5}$$

Indices $\widehat{\mu_k}$, $\widehat{\mu_k^*}$ and $\widehat{\sigma_k^2}$ are Monte-Carlo estimators of respectively $\mu_k = \mathbb{E}_\Theta[\Delta_k f_c(\theta)]$, $\mu_k^* = \mathbb{E}_\Theta[|\Delta_k f_c(\Theta)|]$ and $\sigma_k^2 = Var(\Delta_k f_c(\Theta))$ with $\Theta \sim Unif[\{0, ..., \frac{k}{Q-1}, ..., 1\}^p]$. Thus, under regularity conditions of f_c and its derivatives, when $\delta \to 0$ these estimators converge toward $\widetilde{\mu_k} = \int_{[0,1]^k} \frac{\partial f_c}{\partial \theta_k}(\theta) d\theta$, $\widetilde{\mu_k^*} = \int_{[0,1]^k} |\frac{\partial f_c}{\partial \theta_k}|(\theta) d\theta = ||\frac{\partial f_c}{\partial \theta_k}||_1$ and $\widetilde{\sigma_k^2} = \int_{[0,1]^k} \frac{\partial f_c}{\partial \theta_k}^2(\theta) d\theta - \widetilde{\mu_k^2} = ||\frac{\partial f_c}{\partial \theta_k}||_2^2 - \widetilde{\mu_k^2}$.

If the relation between the k^{th} input parameter and the output is linear then the mean of the elementary effect is proportional to the intensity of the relation. The index σ_k, on the other hand, allows to detect interactions between input parameters and is sensitive to the non-linearity of the output function of the k^{th} input parameter (Faivre et al., 2013). A graph of the elementary effects can be display considering the x-axis as $\widehat{\mu^*}$ and the y-axis as $\widehat{\sigma}$. Then, three areas appear:

- for low values of $\widehat{\mu_k^*}$ and $\widehat{\sigma_k}$, the k^{th} parameter is considered as having a negligible impact on the output,
- for hight values of $\widehat{\mu_k^*}$ compared to $\widehat{\sigma_k}$, the k^{th} parameter has a strong linear effect on the output,
- for hight values of $\widehat{\sigma_k}$ the k^{th} parameter has either a strong non-linear effect on the output and/or includes one or several interactions with other ones.

One of the limits of this method is that we cannot differentiate parameters that have interaction with other ones from those which have a non-linear effect on the output (Faivre et al., 2013). To access this information another study has to be performed afterward. Computing the Sobol indices on the parameters left can be a solution (see Section 2.1.2 for further details).

Morris function

In its paper, Morris (1991) introduced a test function of the method that allows to visualize the three areas. The function takes 20 input parameters and is defined as:

$$Y = \beta_0 + \sum_{i=1}^{20}\beta_i w_i + \sum_{i<j}^{20}\beta_{i,j} w_i w_j + \sum_{i<j<l}^{20}\beta_{i,j,l} w_i w_j w_l + \sum_{i<j<l<s}^{20}\beta_{i,j,l,s} w_i w_j w_l w_s, \tag{2.6}$$

where,

$$w_i = \begin{cases} 2(1.1\frac{X_i}{X_i+0.1} - 0.5) & \text{for i= 3, 5, 7,} \\ 2(X_i - 0.5) & \text{otherwise.} \end{cases}$$

The β are defined as:

$$\begin{aligned}
\beta_i &= 20 && \text{for i=1,...,10,}\\
\beta_{i,j} &= -15 && \text{for i,j=1,...,6,}\\
\beta_{i,j,l} &= -10 && \text{for i,j,l=1,...,5,}\\
\beta_{i,j,l,s} &= 5 && \text{for i,j,l,s=1,...,4.}
\end{aligned}$$

All the β_i, $\beta_{i,j}$ left and β_0 are sampled according to a standard normal distribution. The number of parameter p is equal to 20 and $r = 5$ trajectories are achieved on a sampled space with $Q = 9$. Figure 2.3 illustrates on the left the results of the method with the, previously defined, settings for the Morris function and on the right 10 repetitions of the method on the same function.

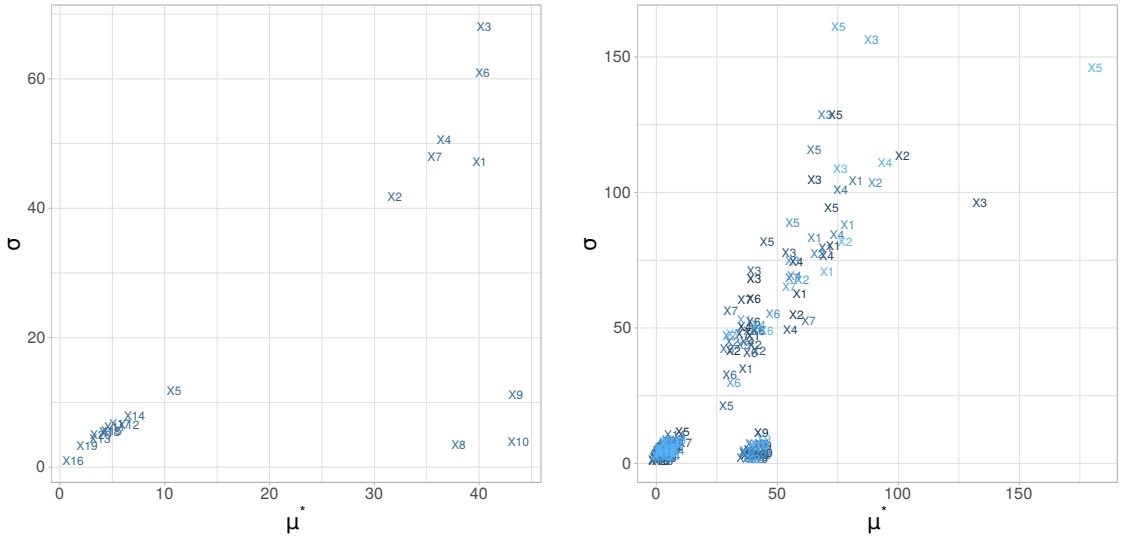

Figure 2.3: On the left panel a result of Morris method on the Morris function and on the right panel 10 repetitions of the method.

The three areas presented above are visible in Figure 2.3. The parameters $X8$, $X9$ and $X10$ are in the area where the mean of the EEs are high compared to the standard deviation of the EEs. That means these parameters have a linear impact on the output. Then, parameters $X1$, $X2$, $X3$, $X4$, $X5$, $X6$ and $X7$ are in the area where σ is high. All other parameters (from $X11$ to $X20$) are negligible since they possess low values of μ^* and σ. This method only allows to identify the parameters to neglect for further study. However, some variability is visible on the right panel of Figure 2.3 because 10 repetitions of the method gives different results.

Non-parametric bootstrap

To quantify the variability of the method, a non-parametric bootstrap is run. This has the advantage to run only once the Morris method. A N re-sample is done on the matrix of the EEs called Δ. In this case the aim is to estimate the following: bias, variance and credibility interval at 95%:

$$b(\widehat{\theta}) = \mathbb{E}(\widehat{\theta} - \theta) \tag{2.7}$$

$$\sigma(\widehat{\theta})^2 = Var(\widehat{\theta}) = Var(\widehat{\theta} - \theta) \tag{2.8}$$

$$CI(\widehat{\theta}) = \widehat{\theta} + [q_{97.5\%}(\theta - \widehat{\theta}); q_{2.5\%}(\theta - \widehat{\theta})] \tag{2.9}$$

These values straightforwardly depend on the distribution $L(\widehat{\theta} - \theta)$. Bootstrapping allows, here, to simulate another distribution $L(\widehat{\theta}^* - \widehat{\theta})$ to estimate these coefficients.

$$\theta = (\mu^*, \sigma)$$
$$\downarrow$$
$$\Delta = \begin{pmatrix} \Delta_1^1 & \dots & \Delta_p^1 \\ \vdots & & \vdots \\ \Delta_1^r & \dots & \Delta_p^r \end{pmatrix} \quad \rightarrow \quad \widehat{\theta} = \begin{pmatrix} \widehat{\mu_1^*} & \widehat{\sigma}_1 \\ \vdots & \vdots \\ \widehat{\mu_p^*} & \widehat{\sigma}_p \end{pmatrix}$$
$$\downarrow$$
$$\Delta^* = \begin{pmatrix} \Delta_1^{1*} & \dots & \Delta_p^{1*} \\ \vdots & & \vdots \\ \Delta_1^{r*} & \dots & \Delta_p^{r*} \end{pmatrix} \quad \rightarrow \quad \widehat{\theta}^* = \begin{pmatrix} \widehat{\mu_1^*}^* & \widehat{\sigma}_1^* \\ \vdots & \vdots \\ \widehat{\mu_p^*}^* & \widehat{\sigma}_p^* \end{pmatrix}$$

Because of the low number of simulations used in Morris method, the uncertainty on the estimated matrix $\widehat{\theta}$ can be high. The lines Δ^* $(\Delta_k^{i*}, \dots, \Delta_k^{r*})$ are each of them sampled uniformly and re-injected among the lines Δ $(\Delta_k^i, \dots, \Delta_k^r)$. This operation is completed N times to get N estimations of $\widehat{\theta}^*$ $(\widehat{\theta}_1^*, \dots, \widehat{\theta}_N^*)$. Then for each sample, the coefficients introduced above can be estimated as:

$$\widehat{b(\widehat{\theta})} = \frac{1}{N}\sum_{i=1}^{N} \widehat{\theta_i^*} - \widehat{\theta_i}, \tag{2.10}$$

$$\widehat{\sigma(\widehat{\theta})}^2 = \frac{1}{N}\sum_{i=1}^{N} \widehat{\theta_i^*}^2 - \left(\frac{1}{N}\sum_{i=1}^{N} \widehat{\theta_i^*}\right)^2, \tag{2.11}$$

$$\begin{aligned} \widehat{IC(\widehat{\theta})} &= \widehat{\theta} - \left[\widehat{\theta} - \widehat{\theta^*}_{[97.5\%*N]}; \widehat{\theta} - \widehat{\theta^*}_{[2.5\%*N]}\right] \\ &= 2\widehat{\theta} - \left[\widehat{\theta^*}_{[97.5\%*N]}; \widehat{\theta^*}_{[2.5\%*N]}\right], \end{aligned} \tag{2.12}$$

where $\widehat{\theta^*}_{[97.5\%*N]}$ corresponds to $\widehat{\theta^*}_{[1]} \leq \dots \leq \widehat{\theta^*}_{[x]}$ and $[x] = E(x)$.

To visualize graphically the results given by the bootstrap, scatter plots can be displayed. Figure 2.4 gives the bootstrap results for $N = 1500$ and for $r = 100$ trajectories of Morris. This shows the trustworthiness of the hypothesis about the variability of the method. The biggest variability concerns only the parameters from $X1$ to $X7$. However even with the variability, the three areas are distinct and the decision about the parameter to neglect can still be achieved.

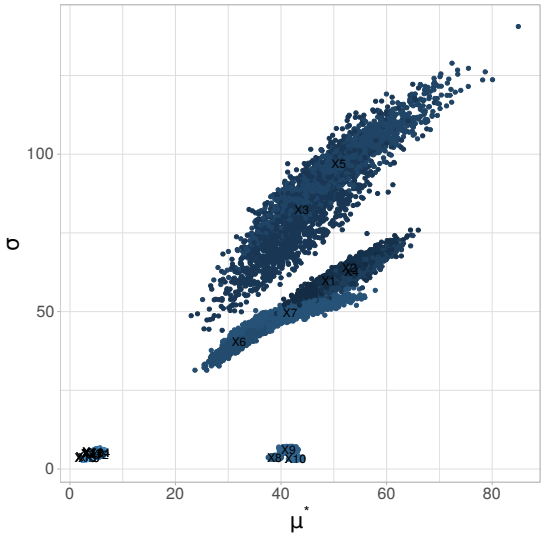

Figure 2.4: Scatter plot of the Morris indices given by the 1500 iterations bootstrap.

2.1.2 Sobol indices

The Morris method only allows to detect parameters that have no impact on the output and hence can be negligible. However to get more information on which parameter has the more influence on the output, another method needs to be applied. Sensitivity index based on a measure of importance can be used and in this section, the variance-based sensitivity analysis is presented. This method is also called Sobol method and uses the variance as the measure of influence of the output to get sensitivity index.

First order sensitivity index

Sobol method has been introduced by Sobol' (1990) and, as it is the case for the Morris method, the parameter input space is normalized between 0 and 1. It means that the space Ω is the k sized hypercube. From Fourier Haar's series, Sobol, in a previous work, had proposed a decomposition of the function f_c such as:

$$f_c(\theta_1,\dots,\theta_k) = f_{c_0} + \sum_{i=1}^{k} f_{c_i}(\theta_i) + \sum_{1\le i<j\le k} f_{c_{ij}}(\theta_i,\theta_j) + \cdots + f_{c_{1,2,\dots,k}}(\theta_1,\dots,\theta_k). \tag{2.13}$$

For Equation (2.13) to make sense, f_{c_0} must be constant and the integral of each part of the sum must be null:

$$\int_0^1 f_{c_{i_1,\dots,i_s}}(\theta_{i_1},\dots,\theta_{i_s})\,d\theta_{i_k} = 0 \quad \text{if} \quad 1 \le k \le s. \tag{2.14}$$

The consequence of Equation (2.13) and Equation (2.14) is that all the parts of the sum are orthogonal. So, if $(i_1,\dots,i_s) \neq (j_1,\dots,j_l)$ then:

$$\int_\Omega f_{c_{i_1,\dots,i_s}} f_{c_{j_1,\dots,j_s}}\,d\boldsymbol{\theta} = 0. \tag{2.15}$$

Sobol shows, in its article Sobol (1993), that the decomposition given in Equation (2.13) is unique and all the terms can be evaluated thanks to multidimensional integrals:

$$f_{c_i}(\theta_i) = -f_{c_0} + \int_0^1 \cdots \int_0^1 f_c(\boldsymbol{\theta})\,d\boldsymbol{\theta}_{\sim i},$$

$$f_{c_{ij}}(\theta_i,\theta_j) = -f_{c_0} - f_{c_i}(\theta_i) - f_{c_j}(\theta_j) + \int_0^1 \cdots \int_0^1 f_c(\boldsymbol{\theta})\,d\boldsymbol{\theta}_{\sim(ij)}.$$

where $d\boldsymbol{\theta}_{\sim i}$ and $d\boldsymbol{\theta}_{\sim(ij)}$ stands for the integrations over the whole domain Ω without (respectively) θ_i and, θ_i

and θ_j. Total variance D of $f_c(\boldsymbol{\theta})$ can be expressed by:

$$D = \int_{\Omega^k} f_c^2(\boldsymbol{\theta}) d\boldsymbol{\theta} - f_{c_0}^2. \tag{2.16}$$

In a similar way and thank to the decomposition introduced in Equation (2.13), partial variances can be written as:

$$D_{i_1,\ldots,i_s} = \int_0^1 \cdots \int_0^1 f_{c_{i_1,\ldots,i_s}}^2(\theta_{i_1},\ldots,\theta_{i_s}) d\theta_{i_1} \ldots d\theta_{i_s}, \tag{2.17}$$

where $1 \leq i_1 < \cdots < i_s \leq k$ and $s = 1,\ldots,k$. When the square of Equation (2.13) is taken and then integrated over Ω^k, the total variance is written:

$$D = \sum_{i=1}^{k} D_i + \sum_{1\leq i<j\leq k} D_{ij} + \cdots + D_{1,2,\ldots,k}. \tag{2.18}$$

The first index in the Sobol method is called the first order Sobol index and is defined as:

First order Sobol's index

$$S_i = \frac{D_i}{D}. \tag{2.19}$$

The first order Sobol's index is S_i for the parameter θ_i. This index allows to quantify the principal effects of this parameter on the variance of the output. The formula $S_i = \frac{Var[Y|\theta_i]}{Var[Y]}$ is, here, found back. The S_{ij} for $i \neq j$ are called second order index and allow to quantify the interactions between θ_i and θ_j that are not taken into account in S_i. If Equation (2.18) is divided by D and replaced by Sobol's index defined in Equation (2.19), then:

$$\sum_{i=1}^{k} S_i + \sum_{1\leq i<j\leq k} S_{ij} + \cdots + S_{1,2,\ldots,k} = 1. \tag{2.20}$$

Total effect index

Total effect index or total sensitivity index (TSI) are defined as the sum of the all the indices encompassing the studied parameter. This definition can be rewritten as:

Total sensitivity index

$$TS(i) = \sum_{I \subseteq i} S_i. \tag{2.21}$$

In a 3-dimensional toy example, the TSI of the parameter θ_1 is given by $TS(1) = S_1 + S_{12} + S_{13} + S_{123}$ where S_1 is the first-order index of the parameter θ_1 on the output y_c, S_{12} and S_{13} are the second order index which are quantifying the interactions of θ_1 with θ_2 and θ_3, and S_{123} the third order index. If $TS(i) \approx S_i$, that means only the first-order index has a significant impact on the output and the superior order indices are then negligible. It is common to compute the first order index and the total index to check if interactions are present.

Compared to the Morris method, Sobol indices allow to visualize interactions between parameters. However, the computation of the first order and the total effect Sobol's indices are performed with a design $\boldsymbol{D}$. To have a global study on the overall impact, for the range of $\boldsymbol{D}$, the analysis has to be run for each $\boldsymbol{\theta}$ in $\boldsymbol{D}$ which is really

time consuming. Some optimal design of experiments can be used to compute the Sobol indices such as optimal Latin Hypercube Sampling (LHS detailed in Section 2.3) described in Saltelli (2002) for example.

Test on Ishigami functions

The Ishigami function is a non-monotonic test function which is defined by:

$$Y = \sin\theta_1 + A\sin^2\theta_2 + B\theta_3^4\sin\theta_1, \tag{2.22}$$

with $\theta_i \sim U(-\pi,\pi)$, $A = 7$ and $B = 0.1$. From a first prospect, no parameter seems having a bigger impact on the output than another. As a matter of fact trigonometric functions and different coefficients lead to a difficult interpretation of the function. The scatter plots in Figure 2.5, represent the output function of each parameter and allow to visualize the behavior of the function according to a specific parameter.

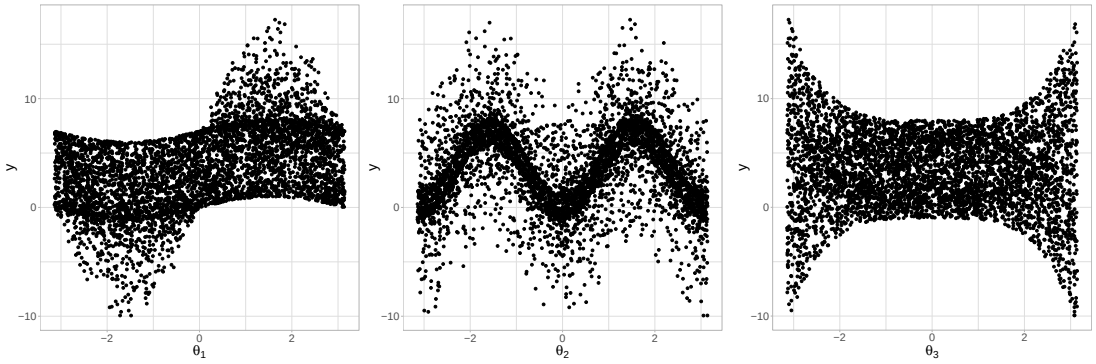

Figure 2.5: Scatter plots of the Ishigami function where the output is given function of the each parameter.

The results of the first order and the total effect Sobol's indices with the Ishigami function are represented in Figure 2.6. The can point out a first remark that the total index is close to the first index for the parameter θ_2 but they are far away for θ_1 and θ_3. It means that for θ_2 there is likely no interaction with other parameters. However, a strong interaction is denoted for θ_1 and θ_3 by the Sobol's indices. As there is only three parameters, the interaction is only between θ_1 and θ_3. Figure 2.6 also represents the box-plot of a bootstrap of 1000 iterations. The variability of this method is lower than the one obtained by Morris.

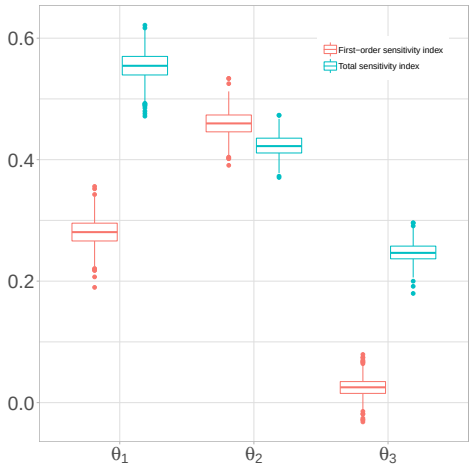

Figure 2.6: Sobol's index computed for the Ishigami function and the boxplots representing the variability of 1000 bootstrap iterations.

2.2 Kriging / Gaussian processes

In mining engineering, predicting the content at site, knowing the content of neighbor sites by a Gaussian process comes from Krige (1951) and is called Kriging. Then Matheron, in Matheron (1963), has proposed the Kriging method to model spatial data in geostatistics (Cressie and Noel, 1993; Stein, 2012). This is Sacks et al. (1989) that have used this formal modeling as surrogate in a numerical experiment framework. More than a surrogate, this kind of modeling brings also an indicator of the uncertainty according to a prediction of the surrogate at a given point. This section will focus, first, on the general framework of the Gaussian process and in a second part on the main parameter estimation methods. Then, on a third part some general covariance functions, used to depict correlation between nearby sites, are developed. To optimize the design of experiments regarding the quality of the Gaussian process, some Gaussian process-based optimization method have been developed and one of them, named Efficient Global Optimization (EGO), is presented in the fourth part.

2.2.1 General framework

Let us consider a probability space $(\Omega, \mathscr{F}, \pi)$ where Ω stands for a sample space, $\mathscr{F}$ a σ-algebra on Ω and π a probability on $\mathscr{F}$. A stochastic process X is a family as $\{X_t\ ;\ t \in \mathscr{T}\}$ where $\mathscr{T} \subset \mathbb{R}^d$. It is said that the aleatory process is indexed by t. At t fixed, the application $X_t\ :\ \Omega \to \mathbb{R}$ is a random variable. At $\omega \in \Omega$ fixed, the application $t \to X_t(\omega)$ is a trajectory of the stochastic process.

For $t_1 \in \mathscr{T}, \ldots, t_n \in \mathscr{T}$, the probability distribution of the random vector $(X_{t_1}, \ldots, X_{t_n})$ is called finite-dimensional distributions of the stochastic process $\{X_t\}_{t\in\mathscr{T}}$. Hence, the probability distribution of an aleatory process is determined by its finite-dimensional distributions. Kolmogorov's theorem guaranties the existence of such a stochastic process if a suitably collection of coherent finite-dimensional distributions is provided.

A random vector $\boldsymbol{Z}$ such as $\boldsymbol{Z} = (Z_1, \ldots, Z_n)$ is Gaussian if $\forall \lambda_1, \ldots, \lambda_n \in \mathbb{R}$ the random variable $\sum_{i=1}^n \lambda_i Z_i$ is Gaussian. The distribution of $\boldsymbol{Z}$ is straightforwardly determined by its two first moments: the mean $\boldsymbol{\mu} = (\mathbb{E}[Z_1], \ldots, \mathbb{E}[Z_n])$ and the variance covariance matrix $\boldsymbol{\Sigma} = cov(Z_i, Z_j)_{1\leq i,\ j\leq n}$. When $\boldsymbol{\Sigma}$ is positive definite, $\boldsymbol{Z}$ has a probability density defined by Equation (2.23).

$$\pi(\boldsymbol{z}) = \frac{|\boldsymbol{\Sigma}|^{-1/2}}{(2\pi)^{n/2}} \exp\Big\{ -\frac{1}{2}(\boldsymbol{z}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{z}-\boldsymbol{\mu}) \Big\}. \tag{2.23}$$

Let us consider two Gaussian vectors called $\boldsymbol{U_1}$ and $\boldsymbol{U_2}$ such as:

$$\begin{pmatrix} \boldsymbol{U_1} \\ \boldsymbol{U_2} \end{pmatrix} \sim \mathscr{N}\left(\begin{pmatrix} \boldsymbol{\mu_1} \\ \boldsymbol{\mu_2} \end{pmatrix}, \begin{pmatrix} \boldsymbol{\Sigma}_{1,1} & \boldsymbol{\Sigma}_{1,2} \\ \boldsymbol{\Sigma}_{2,1} & \boldsymbol{\Sigma}_{2,2} \end{pmatrix} \right).$$

The conditional distribution $\boldsymbol{U_2}|\boldsymbol{U_1}$ is also Gaussian (Equation (2.24)). This property is the base of Kriging when a Gaussian process is used as a surrogate of a function.

Conditional distribution

$$\boldsymbol{U_2}|\boldsymbol{U_1} \sim \mathscr{N}\Big(\boldsymbol{\mu_2} + \boldsymbol{\Sigma}_{2,1}\boldsymbol{\Sigma}_{1,1}^{-1}(\boldsymbol{U_1}-\boldsymbol{\mu_1}), \boldsymbol{\Sigma}_{2,2} - \boldsymbol{\Sigma}_{2,1}\boldsymbol{\Sigma}_{1,1}^{-1}\boldsymbol{\Sigma}_{1,2}\Big). \tag{2.24}$$

A stochastic process $\{X_t\}_{t\in\mathscr{T}}$ is a Gaussian process if each of its finite-dimensional distributions are Gaussian. Let us introduce the mean function such as $m : t \in \mathscr{T} \to m(t) = \mathbb{E}[X_t]$ and the correlation function such as $K : (t,t') \in \mathscr{T}\times\mathscr{T} \to K(t,t') = corr(X_t, X_{t'})$ (as well noted $r(t,t')$ in the thesis). A Gaussian process with a certain variance noted σ^2 will be defined as Equation (2.25).

$$X(.) \sim \mathscr{PG}(m(.), \sigma^2 K(.,.)). \tag{2.25}$$

Gaussian processes are used in this thesis in two cases. In the fist one, f_c is a code function long to run and the Gaussian process emulates its behaviour. Then the Gaussian process is the surrogate of the code. The second case is when we want to model the error made by the code (called code error or discrepancy). For the former, we want to create a surrogate $\hat{f}_c$ of a deterministic function f_c. Let us define this function to emulate f_c such as:

$$\begin{aligned} f_c : \mathscr{Q} \subset \mathbb{R}^p &\to \mathbb{R} \\ \boldsymbol{\theta} &\mapsto f_c(\boldsymbol{\theta}). \end{aligned} \tag{2.26}$$

Following previous statement, f_c is a realization of a Gaussian process. In a Bayesian framework, the Gaussian process is a "functional" *a priori* on f_c (Currin et al., 1991). The natural idea of the Gaussian process emulation is to use known evaluations of f_c at some selected points from a design of experiments D, and access information of the evaluation of f_c at some point $\boldsymbol{\theta_0}$ where $\boldsymbol{\theta_0} \notin D$. The modeling with a Gaussian process of the function f_c can be written as:

$$f_c(\bullet) \sim \mathscr{PG}(\boldsymbol{H}(\bullet)^T \beta, \sigma^2 K_\psi(\bullet,\bullet)), \tag{2.27}$$

where β is a parameter vector such as $\beta = (\beta_1, \ldots, \beta_p)$, σ^2 and ψ respectively stands for the variance and the correlation length of the covariance function $c_{\sigma,\psi}(\bullet,\bullet) = c(\bullet,\bullet) = \sigma^2 K_\psi(\bullet,\bullet)$ and $\boldsymbol{H}(\bullet)$ the matrix of regressors. The covariance function is defined as:

Covariance function

$$(\boldsymbol{\theta}, \boldsymbol{\theta'}) \in \mathscr{Q} \times \mathscr{Q},\ c(\boldsymbol{\theta}, \boldsymbol{\theta'}) = \sigma^2 K_\psi(\boldsymbol{\theta}, \boldsymbol{\theta'}). \tag{2.28}$$

Equation (2.27) can also be written as:

Gaussian process

$$\boldsymbol{\theta} \in \mathscr{Q},\ f_c(\boldsymbol{\theta}) = \boldsymbol{H}(\boldsymbol{\theta})^T \beta + Z(\boldsymbol{\theta}), \tag{2.29}$$

where $Z(\boldsymbol{\theta})$ represents a centered Gaussian process characterized by its covariance function $c(Z(\boldsymbol{\theta}), Z(\boldsymbol{\theta'}) = \sigma^2 K_\psi(\boldsymbol{\theta}, \boldsymbol{\theta'})$. Let us consider a toy function f_t we try to emulated by a Gaussian process with 5 points in the design of experiments:

$$\begin{aligned} f_t : [0,1] &\to \mathbb{R} \\ \theta &\mapsto y = (6\theta - 2)^2 sin(11\theta - 4)\ . \end{aligned} \tag{2.30}$$

To visualize the impact of parameter values, some Gaussian process realizations are performed with different values of σ^2 and ψ. The results are displayed in Figure 2.7 and allows to note that these parameters values impacts the Gaussian process quality.

Usually β, σ^2 and ψ are unknown. Some modeling consider them as determined values but, practically, they are estimated upstream with methods described in Section 2.2.2. They also can be considered as unknown during the Gaussian process modeling with their associated uncertainty encompassed. However, this could lead to complex

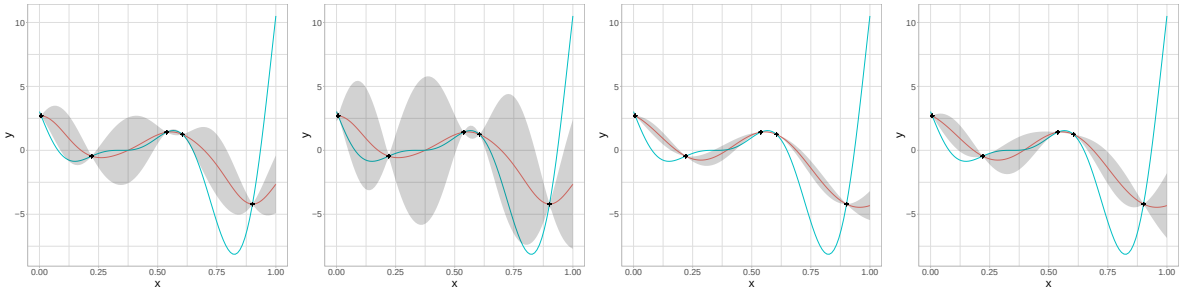

Figure 2.7: Different Gaussian process emulations for the toy function defined Equation (2.30). On the first panel (on the left) the Gaussian process is estimated with $\sigma^2 = 1$ and $\psi = 0.1$. On the second panel (on the middled left), $\sigma^2 = 5$ and $\psi = 0.1$. On the third panel (on the middle right), $\sigma^2 = 1$ and $\psi = 0.2$. And, on the fourth panel (on the right), $\sigma^2 = 5$ and $\psi = 0.2$.

modelings. In what follows, we consider β, σ^2 and ψ as determined values.

Let us consider that N evaluations of f_c are available (where f_c is the general function of Equation (2.27)) at N different points of a design of experiments D. We note the design of experiments (DOE) $D = (\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_N)^T$ and the corresponding outputs of D by f_c, $\boldsymbol{y_c} = (y_1, \dots, y_N)^T$. To get the evaluation of $\boldsymbol{\theta}_0 \in \mathscr{Q} - D$ by f_c, we are interested in knowing the distribution of $f_c(\boldsymbol{\theta_0})$ conditionally to $\boldsymbol{Y}_c$ where $\boldsymbol{Y}_c = (f_c(\boldsymbol{\theta_1}), \dots, f_c(\boldsymbol{\theta}_N))$. This can be written as:

$$\begin{pmatrix} f_c(\boldsymbol{\theta_0}) \\ \boldsymbol{Y}_c \end{pmatrix} \sim \mathscr{N}\left(\begin{pmatrix} \boldsymbol{H}(\boldsymbol{\theta_0})^T\beta \\ \boldsymbol{H}(D)^T\beta \end{pmatrix}, \sigma^2 \begin{pmatrix} \Sigma_\psi(\boldsymbol{\theta_0}) & \boldsymbol{\Sigma}_\psi(\boldsymbol{\theta_0}, D) \\ \boldsymbol{\Sigma}_\psi(\boldsymbol{\theta_0}, D)^T & \boldsymbol{\Sigma}_\psi(D) \end{pmatrix} \right), \tag{2.31}$$

where $\boldsymbol{H}(D) = (\boldsymbol{H}(\boldsymbol{\theta_1}), \dots, \boldsymbol{H}(\boldsymbol{\theta_N}))$, $\boldsymbol{H}(\boldsymbol{\theta_0}) = (\boldsymbol{H}(\theta_0^1), \dots, \boldsymbol{H}(\theta_0^p))$, $\Sigma_\psi(\boldsymbol{\theta_0}) = K_\psi(\boldsymbol{\theta_0}, \boldsymbol{\theta_0}) = \mathbb{I}_p$ (because of the kernel regularity), $(\boldsymbol{\Sigma}_\psi(\boldsymbol{\theta_0}, D))_{1 \le i \le N} = K_\psi(\boldsymbol{\theta_0}, \boldsymbol{\theta_i})$ and $(\boldsymbol{\Sigma}_\psi(D))_{1 \le i,j \le N} = K_\psi(\boldsymbol{\theta_i}, \boldsymbol{\theta_j})$. Using Equation (2.24), it comes straightforwardly that:

$$f_c(\boldsymbol{\theta_0}) | \boldsymbol{Y}_c = \boldsymbol{y_c} \sim \mathscr{N}(\mu_{\boldsymbol{\theta_0}|D}, \sigma^2_{\boldsymbol{\theta_0}|D}),$$

with,

$$\mu_{\boldsymbol{\theta_0}|D} = \boldsymbol{H}(\boldsymbol{\theta_0})^T\beta + \boldsymbol{\Sigma}_\psi(\boldsymbol{\theta_0}, D)\boldsymbol{\Sigma}_\psi(D)^{-1}(\boldsymbol{y_c} - \boldsymbol{H}(D)^T\beta), \tag{2.32}$$

$$\sigma^2_{\boldsymbol{\theta_0}|D} = \sigma^2\Big(1 - \boldsymbol{\Sigma}_\psi(\boldsymbol{\theta_0}, D)^T\boldsymbol{\Sigma}_\psi(D)^{-1}\boldsymbol{\Sigma}_\psi(\boldsymbol{\theta_0}, D)\Big). \tag{2.33}$$

In prediction, the mean can be used to approach the value of $f_c(\boldsymbol{\theta_0})$ and $\hat{f}_c : \boldsymbol{\theta_0} \mapsto \mu_{\boldsymbol{\theta_0}|D}$. The variance $\sigma^2_{\boldsymbol{\theta_0}|D}$ represents the uncertainty associated to the prediction of $f_c(\boldsymbol{\theta_0})$ by $\hat{f}_c(\boldsymbol{\theta_0})$. A confidence interval can be established for the surrogate because:

$$\frac{f_c(\boldsymbol{\theta_0}) - \mu_{\boldsymbol{\theta_0}|D}}{\sqrt{\sigma^2_{\boldsymbol{\theta_0}|D}}} \sim \mathscr{N}(0,1). \tag{2.34}$$

Jones et al. (1998) proposes to validate the surrogate with a "leave-one-out" cross validation method which consists in testifying that 99.7% of the $i = 1, ..., n$:

$$\frac{f_c(\boldsymbol{\theta_i}) - \hat{f}_{c-i}(\boldsymbol{\theta_i})}{\sqrt{\sigma^2_{\boldsymbol{\theta_0}|D}}} \in [-3, 3], \tag{2.35}$$

where $\hat{f}_{c-i}(\boldsymbol{\theta_i})$ and $\sigma^2_{\boldsymbol{\theta_0}|D_{-i}}$ respectively stands for the *posterior* mean and the *posterior* variance obtained by Equation (2.32) and Equation (2.33) for a design of experiments $D = \{\boldsymbol{\theta_1}, \dots, \boldsymbol{\theta}_{i-1}, \boldsymbol{\theta}_{i+1}, \dots, \boldsymbol{\theta}_N\}$.

The predictor $\hat{f}_c(\boldsymbol{\theta_0})$ is the Best Linear Predictor (BLP) if it minimizes the following Mean Square Error (MSE):

$$MSE(\boldsymbol{\theta_0}) = \mathbb{E}[(f_c(\boldsymbol{\theta_0}) - \hat{f}_c(\boldsymbol{\theta_0}))^2]. \tag{2.36}$$

With β, σ^2 and ψ fixed, and under hypotheses of Equation (2.27), the best linear predictor of $f_c(\boldsymbol{\theta_0})$ is:

Best linear predictor

$$\boldsymbol{\theta_0} \mapsto \boldsymbol{H}(\boldsymbol{\theta_0})^T\beta + \boldsymbol{\Sigma_\psi}(\boldsymbol{\theta_0}, D)\boldsymbol{\Sigma_\psi}(D)^{-1}(\boldsymbol{y_c} - \boldsymbol{H}(D)^T\beta). \tag{2.37}$$

This predictor is unbiased and its mean square error is equal to $MSE(\boldsymbol{\theta_0}) = \sigma^2_{\boldsymbol{\theta_0}|D}$. If the evaluated point is taken in the original design D (*i.e.* $\boldsymbol{\theta_0} = \boldsymbol{\theta}_i \in \mathscr{D}$), then $\mu^2_{\boldsymbol{\theta_0}|D} = y_i$ and $\sigma^2_{\boldsymbol{\theta_0}|D} = 0$.

2.2.2 Parameter estimation

Maximum likelihood estimates

Practically, β is unknown, and, to estimate it, a generalized least square method is commonly applied which corresponds to the maximum likelihood of Equation (2.23). This gives:

β prediction

$$\hat{\beta} = (\boldsymbol{H}(D)^T\boldsymbol{\Sigma_\psi}(D)^{-1}\boldsymbol{H}(D))^{-1}\boldsymbol{H}(D)^T\boldsymbol{\Sigma_\psi}(D)^{-1}\boldsymbol{y_c}. \tag{2.38}$$

The predictor given Equation (2.37) can be rewritten as:

$$\hat{f}_c(\boldsymbol{\theta_0}) = \boldsymbol{H}(\boldsymbol{\theta_0})^T\hat{\beta} + \Sigma_\psi(\boldsymbol{\theta_0}, D)\Sigma_\psi(D)^{-1}(\boldsymbol{y_c} - \boldsymbol{H}(D)^T\hat{\beta}), \tag{2.39}$$

and its mean square error is:

$$\begin{aligned} Var(\hat{f}_c(\boldsymbol{\theta_0})) &= \mathbb{E}(\hat{f}_c(\boldsymbol{\theta_0}) - f_c(\boldsymbol{\theta_0}))^2 \\ &= \sigma^2(1 + u(\boldsymbol{\theta_0})^T(\boldsymbol{H}(D)^T\Sigma_\psi(D)^{-1}\boldsymbol{H}(D))^{-1}u(\boldsymbol{\theta_0}) - \Sigma_\psi(\boldsymbol{\theta_0}, D)^T\Sigma_\psi(D)^{-1}\Sigma_\psi(\boldsymbol{\theta_0}, D), \end{aligned} \tag{2.40}$$

where $u(\boldsymbol{\theta_0}) = \boldsymbol{H}(\boldsymbol{\theta_0})^T\Sigma_\psi(D)^{-1}\Sigma_\psi(\boldsymbol{\theta_0}, D) - \boldsymbol{H}(\boldsymbol{\theta_0})$. This prediction mean square error is applied when β is not fixed. If it is fixed the prediction mean square error is given Equation (2.33) but in both cases the predictor mean square error corresponds to the predictor variance since it is unbiased. If we consider σ^2 and β as unknown, the covariance matrix known, then $\hat{f}_c(\boldsymbol{\theta_0})$ is the Best Linear Unbiased Predictor (BLUP) of $f_c(\boldsymbol{\theta_0})$.

To estimate the parameter σ^2, that only intervenes in the calculation of the root mean square error, a maximum likelihood can be applied and:

σ^2 prediction

$$\hat{\sigma}^2 = \frac{1}{N}(\boldsymbol{Y_c} - \boldsymbol{H}(D)\hat{\beta})^T\Sigma_\psi(D)^{-1}(\boldsymbol{Y_c} - \boldsymbol{H}(D)\hat{\beta}). \tag{2.41}$$

Then, the parameter ψ is estimated after having plugged $\hat{\beta}$ and $\hat{\sigma}^2$ in the likelihood and getting the maximum. After parameter plugging, the opposite log-likelihood can be written as:

$$\mathscr{L}(\boldsymbol{Y_c};\psi) \propto N log(\sigma^2) + log(|\Sigma_\psi(D)|), \tag{2.42}$$

and, $\hat{\psi} = \underset{\psi}{argmin}\ \mathscr{L}(\boldsymbol{Y_c};\psi)$.

The resulting predictor

$$\hat{f}_c(\boldsymbol{\theta_0}) = \boldsymbol{H}(\boldsymbol{\theta_0})^T\hat{\beta} + \Sigma_{\hat{\psi}}(\boldsymbol{\theta_0},D)^T\Sigma_{\hat{\psi}}(D)(\boldsymbol{Y_c} - \boldsymbol{H}(D)\hat{\beta}) \tag{2.43}$$

is called EBLUP (Empirical Best Linear Unbiased Predictor) although it is neither linear nor unbiased. Maximum likelihood estimation, and "plug-in" techniques under estimate the prediction variance (Handcock and Stein, 1993; Helbert et al., 2009) and does not take into account any uncertainty affecting the parameters β, σ^2 and ψ.

Bayesian estimation

In this section we focus on Bayesian estimation of σ^2 and β. We will consider a three level hierarchical model, in the sense of Robert (2007), in where the lowest level only deals with the parameter β. At the second level, the parameter σ^2 controls the distribution of β and at the top level, the parameter ψ controls the distribution of σ^2 and β. Note that the estimation of σ^2 causes the loss of normality *a posteriori*. The *prior* density of (β,σ^2) can be written:

$$\pi(\beta,\sigma^2) = \pi(\beta|\sigma^2)\pi(\sigma^2). \tag{2.44}$$

In this section we will work with the four configurations of *priors* present in Santner et al. (2013); Le Gratiet (2013):

1. $\sigma^2 \sim \mathscr{IG}(\alpha,\gamma)$ and $\beta|\sigma^2 \sim \mathscr{N}(\boldsymbol{b_0},\sigma^2\boldsymbol{V_0})$
2. $\pi(\sigma^2) \propto \frac{1}{\sigma^2}$ and $\beta|\sigma^2 \sim \mathscr{N}(\boldsymbol{b_0},\sigma^2\boldsymbol{V_0})$
3. $\sigma^2 \sim \mathscr{IG}(\alpha,\gamma)$ and $\pi(\beta|\sigma^2) \propto 1$
4. $\pi(\sigma^2) \propto \frac{1}{\sigma^2}$ and $\pi(\beta|\sigma^2) \propto 1$

The first case is the more common in Bayesian Kriging because both *priors* are conjugate to a normal likelihood and the posterior density can, explicitly, be written. In the presented configurations, $\mathscr{IG}(\alpha,\gamma)$ stands for the inverse Gamma distribution

$$\pi(x) = \frac{\gamma^\alpha}{\Gamma(\alpha)}\frac{e^{-\gamma/x}}{x^{\alpha+1}}\mathbb{1}_{x>0}.$$

The computation of the *posterior* density can be done for each cases and for ψ fixed. In each cases, the *posterior* can be expressed as $\pi(\beta|\boldsymbol{y_c},\sigma^2) \sim \mathscr{N}(\nu\Sigma,\Sigma)$ with:

$$\Sigma^{-1} = \begin{cases} (\boldsymbol{H}^T\Sigma_\psi(D)\boldsymbol{H} + \boldsymbol{V_0})/\sigma^2 & \text{for the cases (1) and (2),} \\ (\boldsymbol{H}^T\Sigma_\psi(D)\boldsymbol{H})/\sigma^2 & \text{for the cases (3) and (4),} \end{cases} \tag{2.45}$$

and,

$$\nu = \begin{cases} (\boldsymbol{H}^T\Sigma_\psi(D)\boldsymbol{y_c} + \boldsymbol{V_0}\boldsymbol{b_0})/\sigma^2 & \text{for the cases (1) and (2),} \\ (\boldsymbol{H}^T\Sigma_\psi(D)\boldsymbol{y_c})/\sigma^2 & \text{for the cases (3) and (4).} \end{cases} \tag{2.46}$$

In the second level of the hierarchical model, we are now considering σ^2. We can write:

$$\pi(\sigma^2|\boldsymbol{y_c}) = \frac{\mathscr{L}(\boldsymbol{y_c};\beta,\sigma^2)\pi(\beta|\sigma^2)\pi(\sigma^2)}{\pi(\boldsymbol{y_c})}, \tag{2.47}$$

and this equation leads the posterior distribution $\sigma^2|\boldsymbol{y_c} \sim \mathscr{IG}(\alpha_\sigma, \gamma_\sigma)$ where

$$\gamma_\sigma = \begin{cases} 2\gamma + (\boldsymbol{b}_0 - \beta^*)(\boldsymbol{V}_0 + (\boldsymbol{H}^T\Sigma_\psi^{-1}\boldsymbol{H})^{-1}(\boldsymbol{b}_0 - \beta^* + \gamma^*) & \text{for (1)}, \\ (\boldsymbol{b}_0 - \beta^*)(\boldsymbol{V}_0 + (\boldsymbol{H}^T\Sigma_\psi^{-1}\boldsymbol{H})^{-1}(\boldsymbol{b}_0 - \beta^* + \gamma^*) & \text{for (2)}, \\ 2\gamma + \gamma^* & \text{for (3)}, \\ \gamma^* & \text{for (4)}, \end{cases} \tag{2.48}$$

with $\beta^* = (\boldsymbol{H}^T\Sigma_\psi^{-1}\boldsymbol{H})^{-1}(\boldsymbol{H}^T\Sigma_\psi^{-1}\boldsymbol{y_c})$, $\gamma^* = \boldsymbol{y_c}^T(\Sigma_\psi^{-1} - \Sigma_\psi^{-1}\boldsymbol{H}(\boldsymbol{H}^T\Sigma_\psi^{-1}\boldsymbol{H})^{-1}\boldsymbol{H}^T\Sigma_\psi^{-1})\boldsymbol{y_c}$ and:

$$\alpha_\sigma = \begin{cases} \frac{N}{2} + \alpha & \text{for (1)}, \\ \frac{N}{2} & \text{for (2)}, \\ N - \frac{p}{2} + \alpha & \text{for (3)}, \\ N - \frac{p}{2} & \text{for (4)}. \end{cases} \tag{2.49}$$

A posterior predictive distribution can also be available by integrating, over the domain of β, $\pi(\beta|\boldsymbol{y_c}, \sigma^2)$ and then integrating the result over the domain of σ^2. Details of this method are pursued in Le Gratiet (2013). Another method, based on cross validation, is used by Bachoc et al. (2014) to estimate the parameters.

2.2.3 Covariance functions

The choice of the covariance function is crucial in establishing a Gaussian process. Figure 2.8 illustrates the different results that one can have with different covariance functions.

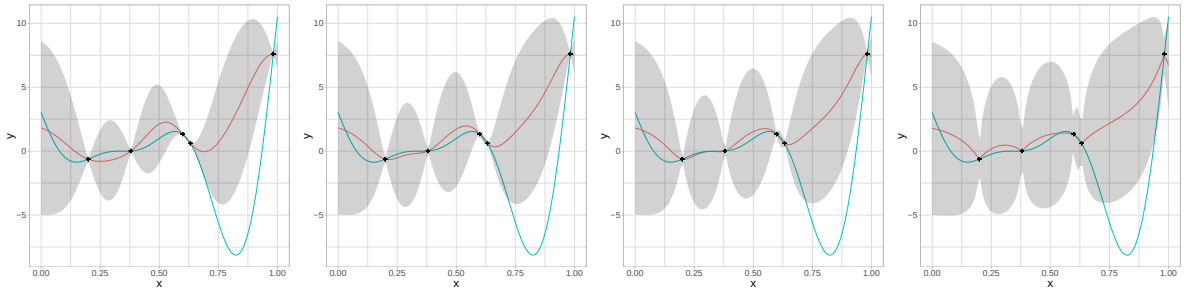

Figure 2.8: Different Gaussian process estimation for the toy function Equation (2.30) with $\sigma^2 = 5$ and $\psi = 0.1$ but with different covariance functions. On the first panel (on the left) the Gaussian process is estimated with Gaussian covariance function. On the second panel (on the middled left), with a Matérn $5/2$. On the third panel (on the middle right), with a Matérn $3/2$. And, on the fourth panel (on the right), with an exponential.

This section recalls the main correlation functions, that will be used for further purpose in the thesis, but the reader is invited to refer to Rasmussen (2004) or Santner et al. (2013) to have more details. In this section and in all the thesis, when it not specified, the Gaussian processes will be considered as stationary (the covariance function is a function of the only argument $\boldsymbol{\theta} - \boldsymbol{\theta}'$ and depends on the distance and the direction of two points in the space) and isotropic (the covariance function is a function of the only argument $|\boldsymbol{\theta} - \boldsymbol{\theta}'|$ the euclidean distance between $\boldsymbol{\theta}$ and $\boldsymbol{\theta}'$). When the Gaussian process is used to emulate the function f_c, the choice of the correlation function depends on the regularity of the function. Just as a recall, $c_{\sigma^2,\psi} = \sigma^2 K_\psi$ where c represents the covariance and K_ψ

the correlation functions.

The exponential functions

The exponential family can be written for a euclidean distance $\boldsymbol{d}$ such as $\boldsymbol{d} = |\boldsymbol{\theta} - \boldsymbol{\theta}'|$ with $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_p)^T$ and $\boldsymbol{\theta}' = (\theta'_1, \ldots, \theta'_p)^T$:

Exponential function family

$$\text{for } 0 < \nu \leq 2 \quad K_{\psi,\nu}(\boldsymbol{d}) = \prod_{j=1}^{p} exp\Big\{ -\frac{\boldsymbol{d}^{\nu}}{\psi} \Big\}. \tag{2.50}$$

This function is a product of unidimensional correlation functions. The parameter ψ is a vector that specifies the correlation length in each direction. If ψ is a scalar the function is isotropic. This parameter influence the correlation scale of the process. If $\nu = 1$ the function is called exponential, but if $\nu = 2$ the function is called Gaussian (otherwise it is a generalized exponential function). For $0 < \nu \leq 2$, the covariance function is differentiable in root mean square and the trajectories are almost likely continuous. However, the Gaussian covariance function is infinitely differentiable which generates a smooth process.

The Matérn functions

The Matérn family (Matérn, 1960) can be written for a distance $\boldsymbol{d}$ such as $\boldsymbol{d} = |\boldsymbol{\theta} - \boldsymbol{\theta}'|$ with $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_p)^T$ and $\boldsymbol{\theta}' = (\theta'_1, \ldots, \theta'_p)^T$:

Matérn function family

$$K_{\psi,\nu}(\boldsymbol{d}) = \prod_{j=1}^{p} \frac{\psi_j |\boldsymbol{d}|^{\nu}}{\Gamma(\nu) 2^{\nu-1}} \boldsymbol{J}_{\nu}(\psi_j |\boldsymbol{d}|), \tag{2.51}$$

where $\boldsymbol{J}_{\nu}$ is a modified ν order Bessel function. Identically as for the exponential family, the parameter ν controls the regularity of the process. For the Matérn family, if $\nu > m$ then the process is m differentiable in root mean square. The parameter ψ still influences the correlation scale. Usually, the Matérn $5/2$ and Matérn $3/2$ are considered and correspond to $\nu = 5/2$ and $\nu = 3/2$ (Rasmussen, 2004).

2.2.4 Gaussian process-based optimization

When the code to emulate is time consuming, the number of code calls is limited. Once the Gaussian process is established, if its variance is too high, it is possible to identify sequentially the inputs locations where the code f_c has to be run to improve the estimation. These new points are generally found around global minimum of f_c over the initial design of experiments D. This method is called Expected Improvements (EI) strategy (Jones et al., 1998) and EI is defined as:

Expected improvement (EI)

$$EI(\boldsymbol{\theta}) = \mathbb{E}(m - f_c(\boldsymbol{\theta})) \mathbb{1}_{f_c(\boldsymbol{\theta}) > m}, \tag{2.52}$$

where $\boldsymbol{\theta}$ is the parameter vector, m the global minimum of f_c over the initial design of experiments D and the expectancy is taken on the distribution of f_c. Figure 2.9 illustrates the values of the EI for the emulated Gaussian process on the function defined Equation (2.30).

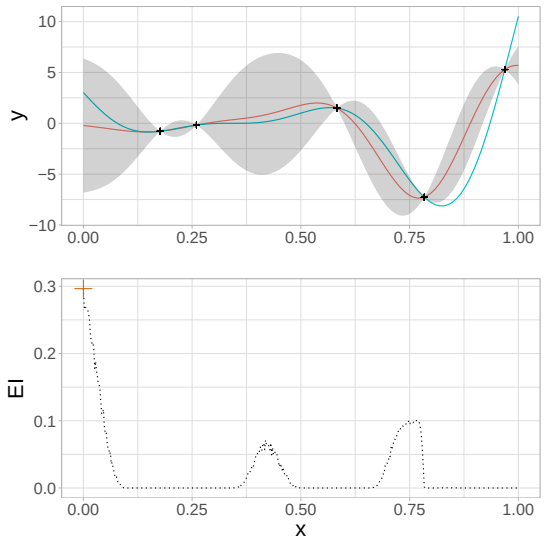

Figure 2.9: Expected improvement computed for the Gaussian process established on the function defined Equation (2.30) with 5 points in the original design of experiments.

Let us consider that k simulations are run and the outputs are expressed by $f_c(D_k)$ where D_k is the design of experiment used to establish the k^{th} Gaussian process. The EI criterion expresses the expected improvement brought by a new point close to the global minimum of f_c. The new point $\boldsymbol{\theta}_{k+1}$ will then be the one that maximizes the expected improvement:

$$\begin{aligned} \boldsymbol{\theta}_{k+1} &= \underset{\boldsymbol{\theta}}{argmax}\, EI_k(\boldsymbol{\theta}), \\ &= \underset{\boldsymbol{\theta}}{argmax}\, \mathbb{E}\Big[((m_k) - F_k(\boldsymbol{\theta}_k))\mathbb{1}_{F_k(\boldsymbol{\theta}_k)>m}\Big], \end{aligned} \tag{2.53}$$

where $m_k = minf(D_k)$ and F_k is the current Gaussian process (established with D_k). If the Gaussian process (GP) used to emulate f_c was deterministic, the EI criterion would be $m_k - \mu_F(\boldsymbol{\theta})$ if $\mu_F(\boldsymbol{\theta}) < m_k$ (with $\mu_F(\boldsymbol{\theta})$ the mean of the Gaussian process) and 0 otherwise. In the case where F is stochastic, the expectation of its truncated difference with respect to the distribution of F_k represents the EI criterion. The algorithm that consists in improving the Gaussian process by updating each time the design of experiments based on the EI criterion is called Efficient Global Optimization (EGO) (Jones et al., 1998). Figure 2.10 illustrates two iterations of the algorithm and shows at each time the new point to add in the new DOE.

The convergence of the EGO has been proven by Vazquez and Bect (2010) under some assumptions such as non-degeneracy of the covariance function of the GP (called the Non-Empty-Ball property) and π-almost all continuous functions, where π is the probability distribution of the GP. Then, Bull (2011) has shown that expected improvement can converge near-optimally, but a naive implementation may not converge at all. Practically the algorithm is stopped after a limited (and authorized) number of calls or when the improvement m_k becomes negligible. In Ginsbourger (2009) it is shown that with comparable levels of performance and based the second stop criterion, EGO algorithm does not need as much code call as for other regular optimization methods. Figure 2.11 illustrates the end of the EGO algorithm for the toy function of Equation (2.30).

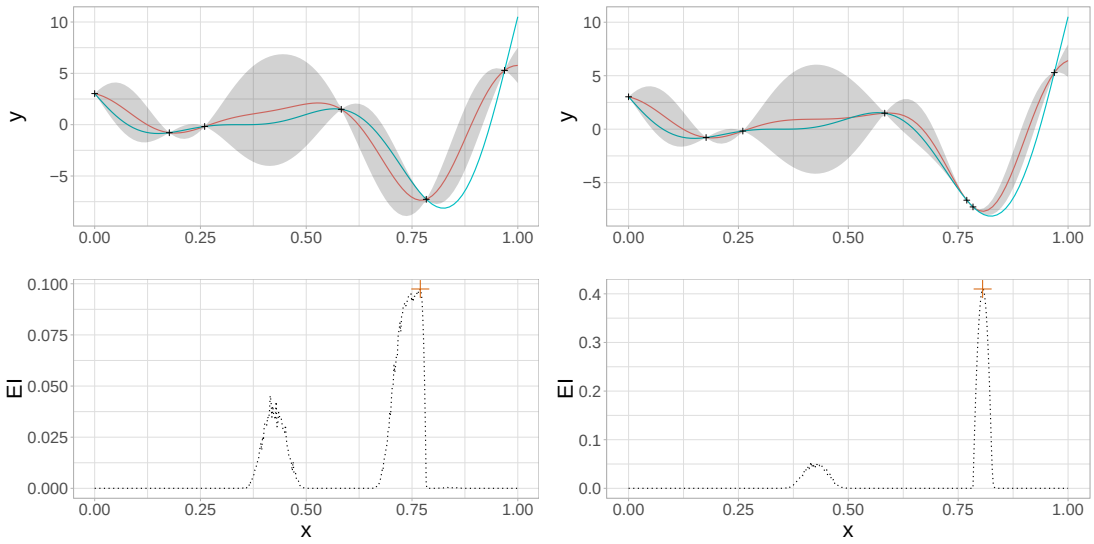

Figure 2.10: 2 EGO iterations with on top the GP updated based on the previous point found with the EI criterion and on the bottom the EI values corresponding to the GP on top. The point in orange is the EI maximum used to establish the following GP.

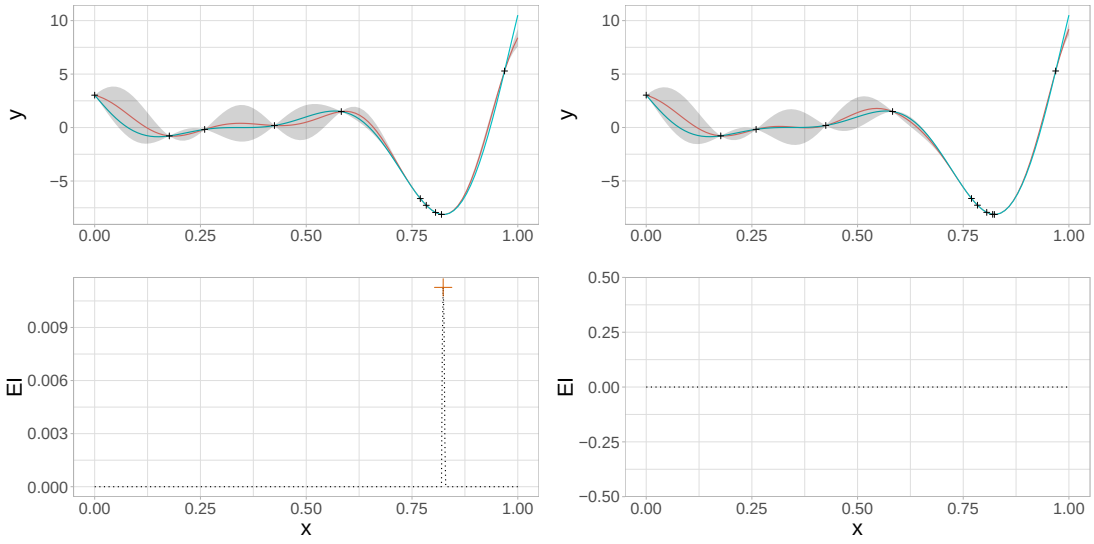

Figure 2.11: 2 last of the 6 iterations of the EGO algorithm.

2.3 Design of experiments

So far, the Design Of Experiments (DOE) to build a Gaussian process as a surrogate or for the Sobol sensitivity analysis was considered known. In both cases, the number of code calls matches with the number of points in the DOE. So it is important, especially when the code is time consuming, to find the design that gives the maximum of information but with a limited number of points. Let us keep the same notation introduced in Subsection 2.1.2 where the DOE was represented by $D = \{\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_N\}$ with $D \subset \mathscr{Q}^N$ and $\mathscr{Q} \subset \mathbb{R}^p$. A mathematical criterion, that specifies that the points are well spread in the parameter input space, needs to be defined. Several criteria exist such as the sampling property of the points in D or the distances between the points. The DOE introduced for the Morris method, in Subsection 2.1.1, is particular to the method because it focuses on evaluating the impact of a displacement in the parameter input space. This design, called factorial design, is particular to this sensitivity analysis method, and is not relevant here. In this section the space $\mathscr{Q}$ will be consider hypercubic, which means that each dimension is bounded. First the sampling criterion of the points in D is presented and will be followed by details on the distance criterion between the points.

2.3.1 Sampling criteria

In the particular cases of the sensitivity analysis or the establishment of the surrogate of a function f_c, considered as a deterministic function, it is interesting to sample points in $\mathscr{Q}$ such as to have a fairly good representation of $f_c(D)$. The $\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_N$ are generated such as the expected value of $\mathbb{E}(f_c(\boldsymbol{\theta}))$ (for the sake of simplicity, $f_c(D) = y(D)$) can be estimated by:

$$y(\bar{D}) = \frac{1}{N} \sum_{i=1}^{N} f_c(\boldsymbol{\theta_i}). \tag{2.54}$$

For a time consuming code function f_c, it is important to chose a sampling scheme that allows to estimate $\mathbb{E}(f_c(\boldsymbol{\theta}))$ well but with N as small as possible. The first idea for sampling D is to generate N *iid* random vectors with an identical probability distribution (usually uniform) on D. This is called the *simple random sampling*. A second sampling method, called *stratified sampling*, considers that a population is represented by multiple subpopulations (or strata) and sampled in each stratum. However, McKay et al. (1979) introduced an alternative method called *Latin Hypercube Design* (LHD), also named *Latin Hypercube Sampling* (LHS). Considering that $\mathscr{Q} = [0,1]^p$, it consists in placing the point of D as for $i = 1, \dots, N$ and $j = 1, \dots, p$:

Coordinates in the LHD

$$\boldsymbol{\theta}_{ij} = \frac{p_{ij} - u_{ij}}{N}, \tag{2.55}$$

where $\boldsymbol{P} = (p_{ij})_{1 \leq i \leq N, 1 \leq j \leq p}$ is a matrix which takes in the j^{th} column random permutations of the integers $\{1, \dots, N\}$ and where $\boldsymbol{U}(u_{ij})_{1 \leq i \leq N, 1 \leq j \leq p}$ is a matrix which takes in the j^{th} column a N sampling in a Uniform distribution between 0 and 1. The LHS is based on the stratified sampling and allows to separate the points in N subdivisions of the space $\mathscr{Q}$. Let us take, as an example, $\mathscr{Q}$ such as $\mathscr{Q} = [0,1]^2$ with $\boldsymbol{\theta} = (\theta_1, \theta_2)^T$. The LHS of $N = 6$ points in $\mathscr{Q}$ obtained is displayed in Figure 2.12.

McKay et al. (1979) shows that if $D1$ is a simple random sample and if $D2$ is a LHS then:

$$Var(Y(\bar{D}_1)) \geq Var(Y(\bar{D}_2)), \tag{2.56}$$

where $Y(\bar{D})$ is a random variable which takes as a realization $y(\bar{D})$ defined in Equation (2.54). It shows that the sampling error is lower when the LHS is performed. Without any hypotheses on the monotony of f_c, Stein (1987) has demonstrated that $Var(Y(\bar{D}_2))$ is asymptotically lower than $Var(Y(\bar{D}_1))$ if the second order moment exists.

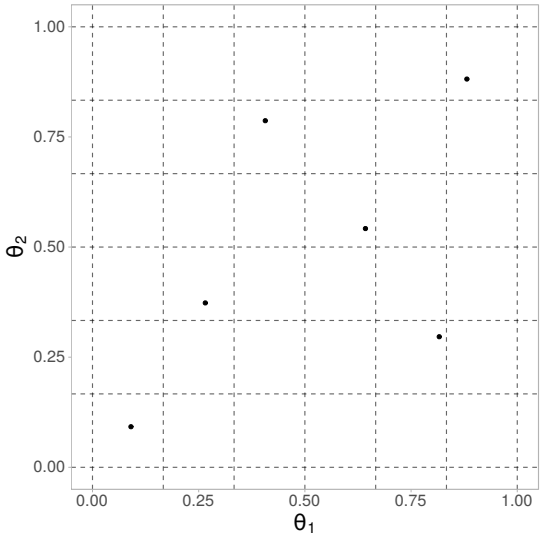

Figure 2.12: 6 points sampled with a LHS for $\mathscr{Q} = [0,1]^2$.

Moreover, the LHS performs well when there are worthless dimensions. However, the LHD is not necessarily a good design for an exploratory DOE. So far the LHD has been build under the hypothesis that the input parameter vector $\boldsymbol{\theta}$ was following an uniform distribution on $[0,1]^p$ with independence between each components. The LHS can be performed with a different distribution for $\boldsymbol{\theta}$ on $[0,1]^p$ but also with dependence between the components of $\boldsymbol{\theta}$ (Stein, 1987). A generalization of the LHD design called randomized orthogonal arrays will not be developed here. For details the reader is referred to the articles of Owen (1992); Tang (1993).

2.3.2 Distance between the points criteria

As it is said above, the LHD can be not optimal regarding space exploration. For example in Figure 2.13 shows that some configuration of LHD does not cover properly the entire input parameter space.

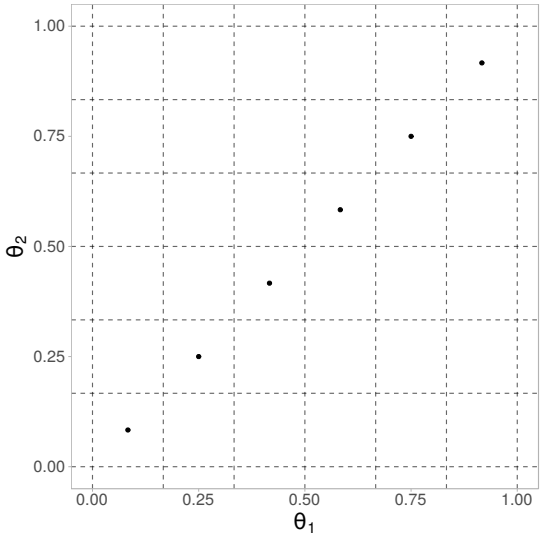

Figure 2.13: 6 points sampled with a LHS for $\mathscr{Q} = [0,1]^2$.

Another criterion can be added to the original design of the LHD to optimize the exploration of the space. Johnson et al. (1990) has introduced two criteria based on the distance between two points. These criteria are used to testify the exploration quality of a DOE. First, a DOE D (defined above) is called *minimax* if it minimizes:

minimax

$$h_D = \sup_{\boldsymbol{\theta} \in \mathscr{Q}} \min_{1 \leq i \leq N} ||\boldsymbol{\theta} - \boldsymbol{\theta}_i||^2. \tag{2.57}$$

The distance distance by $||\bullet||$ refers in this section and along the thesis for the euclidean distance. Similarly, a DOE D is called *maximin* if it maximizes:

maximin

$$\delta_D = \min_{1 \leq i,j \leq N} ||\boldsymbol{\theta}_i - \boldsymbol{\theta}_j||^2. \tag{2.58}$$

As the parameter input space is a compact and as the functions $D \mapsto h_D$ and $D \mapsto \delta_D$ are continuous , the existence of *minimax* and *maximin* design is guarantied. A *minimax* design will ensure that all points in $\mathscr{Q}$ will not be far from a point in D. A *maximin* design tries to create a maximum space between the points to avoid replications.

Morris and Mitchell (1995) proposed to look for a *maximin* design in the latin hypercube classes. It is then possible to use the *maximin* criterion on a LHS to add dispersion property on the original DOE. Figure 2.14 illustrates two different maximin LHS designs obtained with the algorithm introduced in Morris and Mitchell (1995). Several optimal designs exists and they can be compared to Figure 2.12. In the rest of the thesis, maximin LHS refers to this algorithm of Morris and Mitchell (1995).

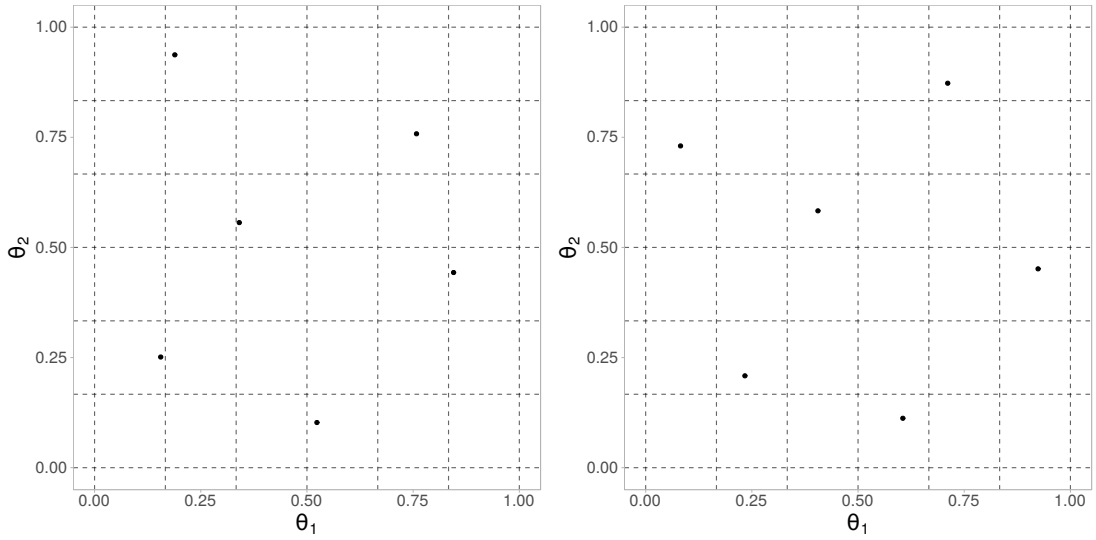

Figure 2.14: 2 6-sized *maximin* LHS performed with the algorithm of Morris and Mitchell (1995) for 2 parameters.

When the number of input parameter increase, the question of the optimization according distance criterion can be highlighted. Indeed, it exists several other distance criteria than maximin distance such as the L^2-discrepancy, the minimum spanning tree (MST) (Dussert et al., 1986; Franco et al., 2009) etc... Some optimizations have also been developed as the Enhanced Stochastic Evolutionary algorithm (ESE) (Jin et al., 2003) or the Simulated Annealing (SA). A review of the optimization methods is available in Viana et al. (2010). Morris and Mitchell (1995) have proposed a version called MM SA (for Morris and Mitchell SA, that is detailed above) and Marrel (2008) has proposed a version called the Boussouf SA. In their paper, Damblin et al. (2013) aim to compare several optimization

methods in the high dimensional case. They analyzed that the ESE algorithm behaves more efficiently than the MM SA and seems to be a good choice as an optimization. They also state that the MST criterion, which analyzes the geometrical profile design according to the distance between the points, is preferable to the the maximin design.

The maximin design is nonetheless good. Indeed, in its paper Schaback (2007) had provided convergence proofs for a generalized non-square version of Kansa's collocation method (Kansa, 1985), showing that the convergence rates are determined by approximation results for non-stationary meshless-kernel-based trial spaces. Furthermore, Auffray et al. (2010) had shown that the maximin design which ensures that any point of $\mathscr{Q}$ is not far from the points of the design that leads to the best performances.

2.4 Principal component analysis (PCA)

The Principal Component Analysis (PCA) (introduced in Pearson (1901)) belongs to the group of multivariate analysis descriptive methods called factorial methods (Husson et al., 2017). In the sense where these methods are descriptive, they are not based on a probabilistic model, but depend on a geometric model. The PCA proposes, from a rectangular table of data composed of n observations (also called individuals) with p variables, graphical representations of these observations and variables. These data can be derived from a sampling procedure or from the observation of an entire population. Graphical representations allow to visualize if it exists a structure, *a priori* unknown, on these data. Similarly, graphical representations can help to identify the linear link structures on variables considered.

Let us consider that the measures are performed on n units $\{u_1,\dots,u_n\}$ and the p quantitative variables that represent these measures are $\{v_1,\dots,v_p\}$. The table of data can be written as:

$$\boldsymbol{X} = \begin{pmatrix} x_{11} & \dots & x_{1p} \\ \vdots & & \vdots \\ x_{n1} & \dots & x_{np} \end{pmatrix}, \tag{2.59}$$

where the j^{th} column represents the data on the n units for the quantitative variable v_j and the i^{th} line the data of p variables for the unit u_i. Let us call $\boldsymbol{U_i}$ the vector of the data for the i^{th} unit such as $\boldsymbol{U_i} = (x_{i1},\dots,x_{ip})^T$ and $\boldsymbol{V_j}$ the vector of data for the variable j such as $\boldsymbol{V_j} = (x_{1j},\dots,x_{nj})^T$.

For the graphical representation of the units, we choose an affine space with , as an origin, a particular point of $\mathbb{R}^p$ (for example the point with the null coordinates). Similarly, in $\mathbb{R}^n$ each variable can be represented by a point. The set of points that represent the variables is called a "scatter plot". However, the dimension of these spaces is in general larger than 2 and even at 3 and we cannot visualize these representations. The general idea of factorial methods is to find a new plan or axis system such as the projections of these scatter plots on this axis or plan allow to reconstitute points positions relative to each other (*i.e.* having the least distorted images as possible).

2.4.1 Distance

To make a geometric representation, it is necessary to choose a distance between two points of the space. The distance used by the PCA is the classical Euclidean distance:

$$d^2(u_i,u_{i'}) = \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2. \tag{2.60}$$

With this distance, all the variables are playing the same role and the axes defined by the variables are constituting an orthogonal basis. A dot product can be associated to this distance:

$$\langle \vec{ou_i}, \vec{ou_{i'}} \rangle = \sum_{j=1}^{p} x_{ij} xi' j = \boldsymbol{U_i^T U_{i'}}, \tag{2.61}$$

as well as a norm:

$$||\vec{ou_i}||^2 = \sum_{j=1}^{p} x_{ij}^2 = \boldsymbol{U_i^T U_i}. \tag{2.62}$$

We can then define the angle α between two vectors by its cosine:

$$cos(\alpha) = \frac{\langle \vec{ou_i}, \vec{ou_{i'}} \rangle}{||\vec{ou_i}||\ ||\vec{ou_{i'}}||} = \frac{\boldsymbol{U_i^T U_{i'}}}{\sqrt{(\boldsymbol{U_i^T U_i})(\boldsymbol{U_{i'}^T U_{i'}})}}. \tag{2.63}$$

The point with null coordinates is not always a satisfactory origin because if the coordinates of the points in the scatter plot are high, the plotted points are far away from the origin. It seems wiser to chose an origin linked to the scatter plot itself: the center of gravity. We will consider for the following that we center the matrix $\boldsymbol{X}$ such as:

$$\boldsymbol{X_c} = \begin{pmatrix} x_{11} - x_{\bullet 1} & \dots & x_{1p} - x_{\bullet p} \\ \vdots & & \vdots \\ x_{n1} - x_{\bullet 1} & \dots & x_{np} - x_{\bullet p} \end{pmatrix}. \tag{2.64}$$

where $x_{\bullet p}$ is the mean of the corresponding column in $\boldsymbol{X}$. The coordinates of the vectors $\boldsymbol{U_i}$ and $\boldsymbol{V_j}$ are also centered and can be written as:

$$\boldsymbol{U}_{ci} = \begin{pmatrix} x_{i1} - x_{\bullet 1} \\ \vdots \\ x_{ip} - x_{\bullet p} \end{pmatrix}, \text{and } \boldsymbol{V}_{cj} = \begin{pmatrix} x_{ij} - x_{\bullet j} \\ \vdots \\ x_{nj} - x_{\bullet j} \end{pmatrix}. \tag{2.65}$$

2.4.2 Moments of inertia

Les us note I_G the moment of inertia of the individuals scatter plot with respect to the center of gravity G:

$$I_G = \frac{1}{n} \sum_{i=1}^{n} d^2(G, u_i) = \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{p} (x_{ij} - x_{\bullet j})^2 = \frac{1}{n} \sum_{i=1}^{n} \boldsymbol{U_{ci}^T U_{ci}}. \tag{2.66}$$

This moment of total inertia is of interest because it measures the dispersion of the individual scatter plot from the center of gravity. If this moment of inertia is large, it means the scatter plot is dispersed. However, if it is small, the scatter plot is highly concentrated on its center of gravity. Another way to write Equation (2.66) can be obtained by inverting the sums:

$$I_G = \sum_{j=1}^{p} \left[\frac{1}{n} \sum_{i=1}^{n} (x_{ij} - x_{\bullet j})^2 \right] = \sum_{j=1}^{p} Var(v_j), \tag{2.67}$$

where $Var(v_j)$ is the empirical variance of the variable v_j. From this form, it can be seen that total inertia is equal to the trace of the covariance matrix $\boldsymbol{\Sigma}$ of the p variables v_j:

$$I_G = trace(\boldsymbol{\Sigma}). \tag{2.68}$$

The inertia of the scatter plot from an axis Δ passing by G is by definition:

$$I_\Delta = \frac{1}{n}\sum_{i=1}^{n} d^2(h_{\Delta i}, u_i), \tag{2.69}$$

where $h_{\Delta i}$ is the orthogonal projection of u_i on Δ. This inertia measures the proximity of the scatter plot to the axis Δ.

The inertia of the scatter plot from a linear subspace V passing by G, is similarly defined, and is equal to:

$$I_V = \frac{1}{n}\sum_{i=1}^{n} d^2(h_{Vi}, u_i), \tag{2.70}$$

where h_{Vi} is the orthogonal projection of u_i on the linear subspace V. Let us not V^* the orthogonal complement of V in $\mathbb{R}^p$ and h_{V^*i} the orthogonal projection of u_i on V^*. By the Pythagorean theorem:

$$d^2(h_{Vi}, u_i) + d^2(h_{V^*i}, u_i) = d^2(G, u_i) = d^2(G, h_{Vi}) + d^2(G, h_{V^*i}). \tag{2.71}$$

From Equation (2.71), it can be concluded that:

Huygens theorem

$$I_V + I_{V^*} = I_G. \tag{2.72}$$

In the particular case where the linear subspace V has a dimension of 1, I_{V^*} is a measure of the scatter plot elongation along the axis of the linear subspace. By projecting the scatter plot on a linear subspace V, the inertia measured by I_V is lost (only I_{V^*} is kept). If $\mathbb{R}^p$ is decomposed in a sum of orthogonal linear subspace such as:

$$\Delta_1 \oplus \Delta_2 \oplus \cdots \oplus \Delta_p, \tag{2.73}$$

then (with a Pythagorean theorem):

$$I_G = I_{\Delta_1^*} + \cdots + I_{\Delta_p^*}. \tag{2.74}$$

2.4.3 Axis of minimum inertia

The axis Δ_1 which is the closest to all the individuals in the scatter plot is an axis passing by G and has the minimum inertia I_{Δ_1}. This axis is interesting because, when the projection will be done, this will give the least distorted image of the scatter plot. Looking for Δ_1 such as I_{Δ_1} is minimum is equivalent to look for Δ_1 such as $I_{\Delta_1^*}$ is maximum (Equation (2.74)). Let us note the unitary vector $\vec{Ga_1}$ of the axis Δ_1. Then, the vector $\vec{Ga_1}$ is looked for such as $I_{\Delta_1^*}$ is maximum under the constraint of $||\vec{Ga_1}||^2 = 1$.

We can write:

$$d^2(G, h_{\Delta_{1i}}) = \langle \vec{Gu_i}, \vec{Ga_1} \rangle^2 = a_1^T \boldsymbol{U}_{ci} \boldsymbol{U}_{ci}^T a_1. \tag{2.75}$$

Using the property of symmetry of the dot product, it can be deduced that:

$$I_{\Delta_1^*} = \frac{1}{n}\sum_{i=1}^{n} a_1^T \boldsymbol{U}_{ci} \boldsymbol{U}_{ci}^T a_1 = a_1^T \left[\frac{1}{n}\sum_{i=1}^{n} \boldsymbol{U}_{ci} \boldsymbol{U}_{ci}^T\right] a_1. \tag{2.76}$$

The empirical variance Σ of the p variables is identified under the bracket of Equation (2.76). Then:

$$I_{\Delta_1^*} = a_1^T \Sigma a_1, \tag{2.77}$$

with,

$$||\vec{Ga_1}||^2 = a_1^T a_1. \tag{2.78}$$

The problem is summarized at finding a_1 that maximize $a_1^T \Sigma a_1$ under the constraint $a_1^T a_1 = 1$. The components of a_1 are unknown and are linked by a constraint. To solve such a problem, a Lagrange multiplier method can be used. This method consists in finding the optimums of a function $f(t_1,\ldots,t_p)$ where the p variables are linked by a relation $l(t_1,\ldots,t_p) = cte$. The p partial derivatives with respect to each variable of the function $g(t_1,\ldots,t_p) = f(t_1,\ldots,t_p) - \lambda(l(t_1,\ldots,t_p) - cte)$ are calculated and annulled to get a system of $p+1$ equations with $p+1$ variables (after adding the constraint term). The term λ is called the Lagrange multiplier. In the previous case, partial derivatives of

$$g(a_1) = g(a_{11},\ldots,a_{1p}) = a_1^T \Sigma a_1 - \lambda_1(a_1^T a_1 - 1) \tag{2.79}$$

has to be found. Using the derivative of a quadratic form with respect of a vector, it can be proven that:

$$\frac{\partial g(a_1)}{\partial a_1} = 2\Sigma a_1 - 2\lambda_1 a_1 = 0. \tag{2.80}$$

The system to solver is then:

$$\begin{cases} \Sigma - \lambda_1 a_1 = 0 & (1) \\ a_1^T a_1 - 1 = 0 & (2) \end{cases} \tag{2.81}$$

From Equation (2.81), matrix Equation (1) allows to show that a_1 is an eigenvector of Σ with eigenvalue λ_1. Multiplying matrix Equation (1) by a_1^T on the left

$$a_1^T \Sigma a_1 - \lambda_1 a_1^T a_1 = 0 \tag{2.82}$$

is obtained and using the Equation (2) in Equation 2.81, it is shown that:

$$a_1^T \Sigma a_1 = \lambda_1. \tag{2.83}$$

From Equation (2.77), the first term of Equation (2.83) is replaced and it can be stated that $\lambda_1 = I_{\Delta_1^*}$. That means the value λ_1 is the highest eigenvalue of Σ and this eigenvalue is the inertia carried by Δ_1. Then, the axis Δ_1 has for unit vector the first eigenvector associated at the highest eigenvalue of the covariance matrix Σ.

However, only one axis might not be sufficient to represent all the data. Then, a second axis Δ_2 is looked for which is orthogonal to Δ_1 and has a minimal inertia. Identically as before, let us define the axis Δ_2 passing by G and defined by its unit vector u_2. The inertia of the individuals scatter plot with respect to it orthogonal complement is:

$$I_{\Delta_2^*} = a_2^T \Sigma a_2. \tag{2.84}$$

The inertia defined Equation (2.84) has to be maximum under the constraints:

$$a_2^T a_2 = 1 \quad \text{and} \quad a_2^T a1 = 0. \tag{2.85}$$

The second constraint Equation (2.85) represents the orthogonality of u_1 with u_2. With the Lagrange multipliers method, but with two constraints, it can be shown that a_2 is the eigenvector of the covariance matrix $\mathbf{\Sigma}$ which corresponds to the second highest eigenvalue. It can be also demonstrated that the inertia is maximum in the linear subspace defined with the unit vectors u_1 and u_2.

The same procedure can be extended but can only be done p times. The covariance matrix $\mathbf{\Sigma}$ is real and symmetrical, thus it has only p eigenvectors (building an orthogonal basis of $\mathbb{R}^p$).

2.4.4 Contribution to the total inertia

All the axes does not have the same contribution on the total inertia. Using the Huygens theorem Equation 2.72, the total inertia can be decomposed as:

$$I_G = I_{\Delta_1^*} + \cdots + I_{\Delta_p^*} = \lambda_1 + \cdots + \lambda_p. \tag{2.86}$$

The absolute contribution to the total inertia of the axis Δ_k is equal to:

$$ca(\Delta_k / I_G) = \lambda_k, \tag{2.87}$$

but its relative contribution is defined as:

$$cr(\Delta_k / I_G) = \frac{\lambda_k}{I_G}. \tag{2.88}$$

For the rest of the thesis, the term "percent of inertia explained by the axis Δ_k" will be used as a reference of the relative contribution. It is possible to extend these definitions at all the generated linear subspaces. For example, the percent of inertia explained by the linear subspace generated by the two eigenvector associated at the two highest eigenvalues is defined as:

$$cr(\Delta_1 \oplus \Delta_2 / I_G) = \frac{\lambda_1 + \lambda_2}{I_G}. \tag{2.89}$$

These inertia percentages are indicators of the part of variability of the scatter plot explained by these subspaces. Practically, only the d ($d < p$) first axis are considered because they explain a percentage of inertia close to 1. The other axes are then neglected.

2.4.5 Graphical representations

Individuals representation quality

To have a graphical representation of the individuals in the plans defined by the new axis, it is necessary to calculate the coordinates of each individuals in the new axis. To get y_{ik} the coordinate of the unit u_i on the axis Δ_j, an orthogonal projection of the vector $\vec{Gu_i}$ is achieved:

$$y_{ik} = \langle \vec{Gu_i}, \vec{a_k} \rangle = a_k^T \boldsymbol{U}_{ci}, \tag{2.90}$$

and,

$$\boldsymbol{Y_i} = \boldsymbol{A}^T \boldsymbol{U}_{ci}, \tag{2.91}$$

where $\boldsymbol{Y_i}$ is the vector of coordinates of the unit u_i and $\boldsymbol{A}$ is the change of basis matrix ($\boldsymbol{A}$ is the matrix composed of the orthogonal eigenvectors, which have a norm equal to 1, is an orthogonal matrix and its inverse is equal to its

transposed).

The square of the cosine of the angle α_{ik} between $\vec{Gu_i}$ and an axis Δ_k is equal to:

$$cos^2(\alpha_{ik}) = \frac{\langle \vec{Gu_i}, \vec{Ga_k} \rangle^2}{||\vec{Gu_i}||^2} = \frac{a_k^T \boldsymbol{U}_{ci} \boldsymbol{U}_{ci}^T a_k}{\boldsymbol{U}_{ci}^T \boldsymbol{U}_{ci}} = \frac{\left[\sum_{j=1}^{p} (x_{ij} - x_{\bullet j}) a_{kj}\right]^2}{\sum_{j=1}^{p} (x_{ij} - x_{\bullet j})^2}. \tag{2.92}$$

As mentioned above, by using the Pythagorean theorem, the square of the cosine of the angle $\alpha_{ikk'}$ between $\vec{Gu_i}$ and the plan generated by the two axes $\Delta_k \oplus \Delta_{k'}$ can be calculated and:

$$cos^2(\alpha_{ikk'}) = cos^2(\alpha_{ik}) + cos^2(\alpha_{ik'}). \tag{2.93}$$

Contribution of an individual to an axis

When $I_{\Delta_k^*}$ (the inertia carried by the axes Δ_k) is calculated, it is possible to see the part of the inertia attributable to an individual u_i in particular. From Equation (2.70), $I_{\Delta_k^*} = \frac{1}{n} \sum_{i=1}^{n} d^2(h_{\Delta_{ki}}, G)$ and the absolute contribution of u_i of this inertia is:

$$ca(u_i/\Delta_k) = \frac{1}{n} d^2(h_{\Delta_{ki}}, G), \tag{2.94}$$

because all the individuals have the same weight. Practically, an individual will have a greater contribution to the axis when its projection will be far from the center of gravity. The other way around, an individual which has a projection close to the center of gravity will have a weak contribution to the axis. It is also possible, for the same particular individual u_i, to give its relative contribution to the inertia carried by Δ_k:

$$cr(u_i/\Delta_k) = \frac{\frac{1}{n} d^2(h_{\Delta_{ki}}, G)}{I_{\Delta_k^*}} = \frac{\frac{1}{n} \langle \vec{Gu_i}, \vec{Ga_k} \rangle^2}{\lambda_k} = \frac{\frac{1}{n} a_k^T \boldsymbol{U}_{ci} \boldsymbol{U}_{ci}^T a_k}{\lambda_k}. \tag{2.95}$$

Note that $\sum_{i=1}^{n} cr(u_i/\Delta_k) = 1$.

Graphical representation of the variables

The individuals have been represented in the space of the original variables and the change of basis has been made in this original space. New axes are then linear combination of original axes and can be considered as new variables which are a linear combination of original variables. These new variables are called “principal components”. Let us note $Z1, \ldots, Z_p$ the principal components, Z_k being the new variable corresponding to the axes Δ_k and:

$$Z_k = \sum_{j=1}^{p} a_{kj} \boldsymbol{V}_{cj} = \boldsymbol{X_c} a_k, \tag{2.96}$$

where $\boldsymbol{V}_{cj}$ is the centered variable vector defined in Equation (2.65) and $\boldsymbol{X_c}$ the matrix of centering data introduced Equation (2.64). In general, the principal components can be written as:

$$\boldsymbol{Z} = [Z_1, \ldots, Z_p] = \boldsymbol{X_c} \boldsymbol{A}. \tag{2.97}$$

It is also interesting to visualize how original variables are linked to the principal components by calculating the correlations. Original variables representation will be done by taking as coordinates the correlation coefficients with the new variables. This representation is called “correlation circle” because of the fact that a correlation coefficient is always between -1 and 1 which leads to have all the points in a circle of radius 1. The variances of the principal components are:

$$Var(Z_k) = \frac{1}{n} a_k^T \boldsymbol{X}_{\boldsymbol{c}}^T \boldsymbol{X}_{\boldsymbol{c}} a_k = a_k^T \boldsymbol{\Sigma} a_k = \lambda_k, \tag{2.98}$$

their covariances are:

$$Cov(Z_k, V_{cj}) = \frac{1}{n} a_k^T \boldsymbol{X}_{\boldsymbol{c}}^T \boldsymbol{V}_{cj} = \frac{1}{n} a_k^T \boldsymbol{X}_{\boldsymbol{c}}^T \boldsymbol{X}_{\boldsymbol{c}} \begin{pmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{pmatrix} = a_k^T \boldsymbol{\Sigma} \begin{pmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{pmatrix} = \lambda_k a_k^T \begin{pmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{pmatrix} = \lambda_k a_{kj}, \tag{2.99}$$

and their correlations with the original variables are:

$$Cor(Z_k, \boldsymbol{V}_{cj}) = \sqrt{\lambda_k} \frac{a_{kj}}{\sqrt{Var(\boldsymbol{V}_j)}}, \tag{2.100}$$

where a_{kj} is the j^{th} coordinates of the unit vector a_k of Δ_k. Generally, the covariance matrix of the principal components $\boldsymbol{\Sigma}_{\boldsymbol{Z}}$ can be written as:

$$\boldsymbol{\Sigma}_{\boldsymbol{Z}} = \frac{1}{n} \boldsymbol{A}^T \boldsymbol{X}_{\boldsymbol{c}}^T \boldsymbol{X}_{\boldsymbol{c}} \boldsymbol{A} = \boldsymbol{A}^T \boldsymbol{\Sigma} \boldsymbol{A} = \boldsymbol{\Lambda}, \tag{2.101}$$

where $\boldsymbol{\Lambda}$ is the diagonal matrix of the eigenvalues of Σ:

$$\boldsymbol{\Lambda} = \begin{pmatrix} \lambda_1 & & (0) \\ & \ddots & \\ (0) & & \lambda_p \end{pmatrix}, \tag{2.102}$$

and the covariance matrix between the principal components and original variables is:

$$Cov(\boldsymbol{Z}, \boldsymbol{V}) = \frac{1}{n} \boldsymbol{X}_{\boldsymbol{c}}^T \boldsymbol{X}_{\boldsymbol{c}} \boldsymbol{A} = \boldsymbol{\Sigma} \boldsymbol{A} = \boldsymbol{A} \boldsymbol{\Lambda}. \tag{2.103}$$

Example on the decathlon data set

As an example, let us consider the decathlon data set in the **FactoMineR** (Husson et al., 2018) package. The data set is an example of decathlon data which refers to athletes' performance during two athletic meetings (2004 Olympic Games or 2004 Decastar). The data set is made of 41 rows and 13 columns where the ten first column only are kept. These columns correspond to the performances of the athletes of the ten events of the decathlon.

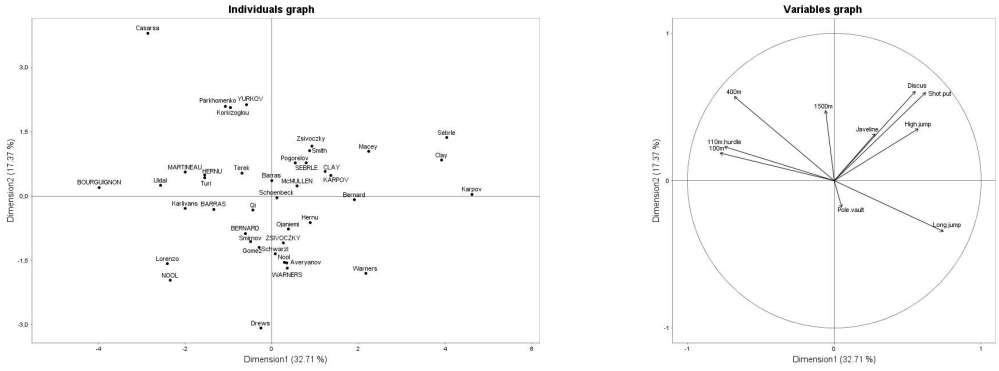

Figure 2.15: Graphical representation of the individuals (on the left) and the variables (on the right) of the decathlon data set.

Figure 2.15 represents the individuals and the variables of the decathlon data set. For each axis, the percentage of the inertia carried is indicated and both axis (the ones used for the representation) contain 50% of the total inertia. The right panel of Figure 2.15 illustrates the correlation between the variables and for example the variable "100m" is negatively correlated with the variable "long.jump". This means that when an athlete jumps far, he does not achieve so good performances in the 100m. The left panel of Figure 2.15 illustrates in two axis a global representation of the performances of the athletes. The first axis strongly represents the speed events and long jump event (given from the right panel of Figure 2.15). This means for an individual that has a high value on this axis, he is more likely better on these events than an individual that has a weak value on the axis. For example Karpov seems to have better performed than Bourguignon in these events. The second axis represents the variables "Discuss" and "shot put" and similarly a high value of the individual on this axis means that he performed well on these events. The winner of the 2004 Olympic Games is Sebrie who has reasonable value on both axis. The variable "Discus", "Shot.put" and "High.jump" are not correlated to "100m", "400m", "110m.hurdle" and "Long.jump" which means that strength and speed are not correlated.

2.5 Monte Carlo Markov Chains techniques

Sampling from a posterior distribution is the only mean to get information about the posterior distribution if there is no conjugacy in the model. Several Monte Carlo Markov Chains (MCMC) have been developed in that sense (Robert and Casella, 2013; Andrieu et al., 2003). Let us consider a general example, in a Bayesian framework, where the posterior density of a p dimensional parameter vector $\boldsymbol{\theta}$ is looked for. Let us define a probability space $(\Omega, \mathscr{F}, \pi)$ where Ω stands for a sample space, $\mathscr{F}$ a σ-algebra on Ω and π a probability on $\mathscr{F}$. The parameter vector is defined as $\boldsymbol{\theta} \in \mathscr{Q} \subset \mathbb{R}^p$ and the n dimensional experimental vector as $\boldsymbol{y} \in \mathscr{E} \subset \mathbb{R}^n$. From Bayes formula, it is possible to write straightforwardly that:

$$\pi(\boldsymbol{\theta}|\boldsymbol{y}) = \frac{\mathscr{L}(\boldsymbol{y};\boldsymbol{\theta})\pi(\boldsymbol{\theta})}{m(\boldsymbol{y})}, \tag{2.104}$$

where $m(\boldsymbol{y}) = \int_{\boldsymbol{\theta}\in\mathscr{Q}} \mathscr{L}(\boldsymbol{y};\boldsymbol{\theta})\pi(\boldsymbol{\theta})$, but as the term $m(\boldsymbol{y})$ is independent from $\boldsymbol{\theta}$, Equation (2.104) can be rewritten as:

$$\pi(\boldsymbol{\theta}|\boldsymbol{y}) \propto \mathscr{L}(\boldsymbol{y};\boldsymbol{\theta})\pi(\boldsymbol{\theta}). \tag{2.105}$$

When the *prior* distribution is not conjugate to the the likelihood, $\pi(\boldsymbol{\theta}|y)$ cannot be expressed analytically. The choice of the sampling method is partly conditioned by the knowledge of full conditional distributions (*i.e.* the knowledge of $\pi(\theta_i|\boldsymbol{y},\boldsymbol{\theta}_{(-i)})\ \forall i \in [1,\dots,n]$, $\boldsymbol{\theta}_{(-i)}$ representing the vector $\boldsymbol{\theta}$ without the i^{th} component). In case where these full conditional distributions are known, the Gibbs sampler can be used. Otherwise, Metropolis Hastings algorithms could be applied to sample in the *posterior* distribution. The full conditional distribution can be written proportionally to the joint distribution such as:

$$\pi(\theta_i|\boldsymbol{y},\boldsymbol{\theta}_{-i}) = \frac{\pi(\boldsymbol{\theta},\boldsymbol{y})}{\pi(\theta_1,\dots,\theta_{i-1},\theta_{i+1},\dots,\theta_p,\boldsymbol{y})} \propto \pi(\boldsymbol{\theta},\boldsymbol{y}). \tag{2.106}$$

Then from Equation (2.105):

$$\begin{aligned}\pi(\theta_i|\boldsymbol{\theta}_{-i},\boldsymbol{y}) &\propto \mathscr{L}(\boldsymbol{y};\boldsymbol{\theta})\pi(\boldsymbol{\theta}) \\ &\propto \mathscr{L}(\boldsymbol{y};\boldsymbol{\theta})\pi(\theta_1|\theta_2,\dots,\theta_p)\pi(\theta_2|\theta_3,\dots,\theta_p)\dots\pi(\theta_{p-1}|\theta_p)\pi(\theta_p).\end{aligned} \tag{2.107}$$

And, if all the components of $\boldsymbol{\theta}$ are independent then:

$$\pi(\theta_i|\boldsymbol{\theta}_i,\boldsymbol{y}) \propto \mathscr{L}(\boldsymbol{y};\boldsymbol{\theta}) \prod_{i=1}^{p} \pi(\theta_i). \tag{2.108}$$

MCMC goal

The aim of the MCMC is then to produce a Markov chain whose invariant distribution is the *posterior* distribution

2.5.1 Gibbs sampler

If the full conditional distributions are known, then it is possible to sample in each of them. This way, if an initial state $\boldsymbol{\theta}^{(s)}$, such as $\boldsymbol{\theta}^{(s)} = (\theta_1^{(s)}, \ldots, \theta_p^{(s)})$, is considered, a new state can be sampled from:

1. sample $\theta_1^{(s+1)} \sim \pi(\theta_1^{(s)}|\boldsymbol{\theta}_2^{(s)}, \ldots, \boldsymbol{\theta}_p^{(s)}, \boldsymbol{y})$

2. sample $\theta_2^{(s+1)} \sim \pi(\theta_2^{(s)}|\boldsymbol{\theta}_1^{(s)}, \boldsymbol{\theta}_3^{(s)}, \ldots, \boldsymbol{\theta}_p^{(s)}, \boldsymbol{y})$

$\vdots$

(p). sample $\theta_p^{(s+1)} \sim \pi(\theta_p^{(s)}|\boldsymbol{\theta}_1^{(s)}, \ldots, \boldsymbol{\theta}_{p-1}^{(s)}, \boldsymbol{y})$

(p+1). $\boldsymbol{\theta}^{(s+1)} = (\theta_1^{(s+1)}, \ldots, \theta_p^{(s+1)})$

The algorithm that repeats this sequence S times is called *Gibbs sampler* and generates a dependent sequence of parameters: $\boldsymbol{\theta} = \{\boldsymbol{\theta}^{(1)}, \ldots, \boldsymbol{\theta}^{(S)}\}$ (Robert and Casella, 2013; Hoff, 2009). The algorithm can be written as:

Algorithm 1 *Gibbs sampler*

$\boldsymbol{\theta}^{(1)} = \boldsymbol{\theta}_{init}$
for i in $2 : S$ **do**
  for j in $1 : p$ **do**
    $\theta_j^{(i)} \sim \pi(\theta_j^{(i-1)}|\boldsymbol{\theta}_{1\ldots j-1}^{(i)}, \boldsymbol{\theta}_{j+1\ldots p}^{(i)}, \boldsymbol{y})$
  end for
  $\boldsymbol{\theta}^{(i)} = (\theta_1^{(i)}, \ldots, \theta_p^{(i)})$
end for

For example, let us consider a sample of data y such as each component are a realization of the random variable Y which is following $Y \sim \mathscr{N}(\mu, \frac{1}{\tau})$ where μ is the population mean and τ the population precision (inverse of the variance). Let us consider n the sample size, $\bar{y}$ the sample mean and s^2 the sample variance. *A priori*, there is no correlation between the population mean and the population variance:

$$\pi(\mu,\tau) = \pi(\mu)\pi(\tau). \tag{2.109}$$

Considering that $\pi(\mu) \propto 1$ and $\pi(\tau) \propto \tau^{-1}$ as *prior* densities (Casella and George, 1992), the conditional *posterior* densities are explicit and can be expressed as:

$$\mu|\tau,y \sim \mathcal{N}\left(\bar{y}, \frac{1}{n\tau}\right), \tag{2.110}$$

$$\tau|\mu,y \sim \Gamma\left(\frac{n}{2}, \frac{2}{(n-1)s^2 + n(\mu-\bar{y})^2}\right). \tag{2.111}$$

As the conditional *posterior* densities can be written, the *Gibbs sampler* can be run and then have access to the joint *posterior* density. Figure 2.16 is the result of the *Gibbs sampler* in this example.

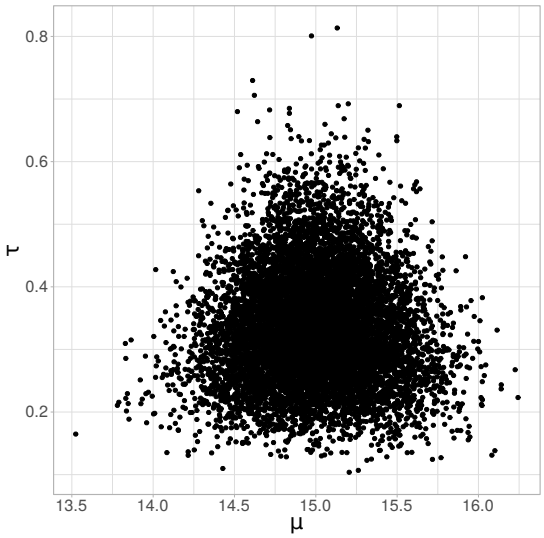

Figure 2.16: *Gibbs sampler* completed for 10000 iteration with a 1000 burn-in sample.

When conjugate *prior* densities are not accessible, the *Gibbs sampler* cannot be used. To sample in the *posterior* joint density, another algorithm can be performed.

2.5.2 Metropolis Hastings

The Metropolis Hastings algorithm has been introduced in Metropolis et al. (1953) by Metropolis but with a specific Boltzmann distribution. In Hastings (1970), Hastings has generalized the algorithm for every specific cases. To approximate the *posterior* density, it is not possible to generate independent and identically distributed (*i.i.d.*) samples from $\pi(\boldsymbol{\theta}|\boldsymbol{y})$. To do so, a large collection of $\boldsymbol{\theta}$ values $\{\boldsymbol{\theta}^{(1)},\dots,\boldsymbol{\theta}^{(s)}\}$, whose empirical distribution approximates $\pi(\boldsymbol{\theta}|\boldsymbol{y})$, are generated. Let us suppose that we have a working collection such as $\{\boldsymbol{\theta}^{(1)},\dots,\boldsymbol{\theta}^{(s)}\}$ to which we would like to add a new value $\boldsymbol{\theta}^{(s+1)}$. Let us consider a new point $\boldsymbol{\theta}^{(*)}$ which is nearby $\boldsymbol{\theta}^{(s)}$. If $\pi(\boldsymbol{\theta}^{(*)}|\boldsymbol{y}) > \pi(\boldsymbol{\theta}^{(s)}|\boldsymbol{y})$, it means that $\boldsymbol{\theta}^{(*)}$ has a probability to be in the set higher than $\boldsymbol{\theta}^{(s)}$ which is already in the set. In that particular case $\boldsymbol{\theta}^{(s+1)}$ can take the value of $\boldsymbol{\theta}^{(*)}$. However, if $\pi(\boldsymbol{\theta}^{(*)}|\boldsymbol{y}) < \pi(\boldsymbol{\theta}^{(s)}|\boldsymbol{y})$, $\theta^{(*)}$ should not be necessarily rejected. A decision, based on the comparison of $\pi(\boldsymbol{\theta}^{(*)}|\boldsymbol{y})$ and $\pi(\boldsymbol{\theta}^{(s)}|\boldsymbol{y})$ has to be established. Fortunately, this comparison can be made even if $\pi(\boldsymbol{\theta}|\boldsymbol{y})$ cannot be computed (from Equation (2.104)):

$$r = \frac{\pi(\boldsymbol{\theta}^{(*)}|\boldsymbol{y})}{\pi(\boldsymbol{\theta}^{(s)}|\boldsymbol{y})} = \frac{\mathcal{L}(\boldsymbol{\theta}^{(*)};\boldsymbol{y})\pi(\boldsymbol{\theta}^{(*)})}{m(\boldsymbol{y})} \frac{m(\boldsymbol{y})}{\mathcal{L}(\boldsymbol{\theta}^{(s)};\boldsymbol{y})\pi(\boldsymbol{\theta}^{(s)})} = \frac{\mathcal{L}(\boldsymbol{\theta}^{(*)};\boldsymbol{y})\pi(\boldsymbol{\theta}^{(*)})}{\mathcal{L}(\boldsymbol{\theta}^{(s)};\boldsymbol{y})\pi(\boldsymbol{\theta}^{(s)})}. \tag{2.112}$$

The intuition is when $r > 1$, $\pi(\boldsymbol{\theta}^{(*)}|\boldsymbol{y}) > \pi(\boldsymbol{\theta}^{(s)}|\boldsymbol{y})$ and $\boldsymbol{\theta}^{(*)}$ should be accepted in the set. On the other hand if $r < 1$, the probability to add $\boldsymbol{\theta}^{(*)}$ in the set is equal to the $min(1,r)$. It is similarly as comparing r to a value that has been sampled from an uniform distribution between 0 and 1 (let us call this value u). Then if $r > u$, $\boldsymbol{\theta}^{(*)}$ is accepted in the set, if $r < u$, then $\boldsymbol{\theta}^{(*)}$ is rejected.

The new point $\boldsymbol{\theta}^{(*)}$, which is nearby $\boldsymbol{\theta}^{(s)}$, is generated from a proposal distribution (also called *jumping distribution* in Gelman et al. (1995)). This distribution $q(\boldsymbol{\theta}^{(*)}|\boldsymbol{\theta}^{(s)})$ can be distinguished in two cases. The first, the distribution is symmetric such as $q(\boldsymbol{\theta}^{(*)}|\boldsymbol{\theta}^{(s)}) = q(\boldsymbol{\theta}^{(s)}|\boldsymbol{\theta}^{(*)})$ which is the case for:

- $q(\boldsymbol{\theta}^{(*)}|\boldsymbol{\theta}^{(s)}) = \text{uniform}(\boldsymbol{\theta}^{(s)} - \delta, \boldsymbol{\theta}^{(s)} + \delta)$,
- $q(\boldsymbol{\theta}^{(*)}|\boldsymbol{\theta}^{(s)}) = \text{normal}(\boldsymbol{\theta}^{(s)}, k\boldsymbol{\Sigma}^{\mathbf{2}})$.

In that particular case the algorithm is called Metropolis algorithm and the computation of the ratio is straightforwardly obtained from Equation (2.112). Algorithm 2 recalls the steps of this particular MCMC.

Algorithm 2 Metropolis algorithm

$\boldsymbol{\theta}^{(1)} = \boldsymbol{\theta}_{init}$
$\tau_{accept} = 0$
for i in $1 : S$ **do**
  $\boldsymbol{\theta}^{(*)} \sim \mathcal{N}(\boldsymbol{\theta}^{(i)}, k\Sigma)$
  $r = \frac{\pi(\boldsymbol{\theta}^{(*)}|y)}{\pi(\boldsymbol{\theta}^{(i)}|y)} = \frac{\pi(\boldsymbol{\theta}^{(*)})\mathcal{L}(\boldsymbol{\theta}^{(*)};y)}{\pi(\boldsymbol{\theta}^{(i)})\mathcal{L}(\boldsymbol{\theta}^{(i)};y)}$
  if $r > u$, with $u \sim \mathcal{U}(0,1)$ **then**
    $\boldsymbol{\theta}^{(i+1)} = \boldsymbol{\theta}^{(*)}$
    $\tau_{accept} = \tau_{accept} + 1$
  else
    $\boldsymbol{\theta}^{(i+1)} = \boldsymbol{\theta}^{(i)}$
  end if
end for
$\boldsymbol{\theta}_M = \boldsymbol{\theta}[N_{burn-in} : S]$

On the other hand if $q(\boldsymbol{\theta}^{(*)}|\boldsymbol{\theta}^{(s)})$ is not symmetric, the ratio is now given by:

$$r = \frac{\mathcal{L}(\boldsymbol{\theta}^{(*)};\boldsymbol{y})\pi(\boldsymbol{\theta}^{(*)})}{\mathcal{L}(\boldsymbol{\theta}^{(s)};\boldsymbol{y})\pi(\boldsymbol{\theta}^{(s)})} \frac{q(\boldsymbol{\theta}^{(s)}|\boldsymbol{\theta}^{(*)})}{q(\boldsymbol{\theta}^{(*)}|\boldsymbol{\theta}^{(s)})}. \tag{2.113}$$

The ratio is almost surely defined because a jump can only occur if both $q(\boldsymbol{\theta}^{(*)}|\boldsymbol{\theta}^{(s)})$ and $q(\boldsymbol{\theta}^{(s)}|\boldsymbol{\theta}^{(*)})$ are non zero. Such a proposal can be useful for increasing the speed of the random walk. Algorithm 3 details the steps of the, so-called, Metropolis Hastings algorithm.

Algorithm 3 Metropolis Hastings algorithm

$\boldsymbol{\theta}^{(1)} = \boldsymbol{\theta}_{init}$
$\tau_{accept} = 0$
for i in $1 : S$ **do**
  $\boldsymbol{\theta}^{(*)} \sim q(\boldsymbol{\theta}^{(i)}, .)$
  $r = \frac{\pi(\boldsymbol{\theta}^{(*)}|\boldsymbol{y})q(\boldsymbol{\theta}^{(*)},\boldsymbol{\theta}^{(i)})}{\pi(\boldsymbol{\theta}^{(i)}|\boldsymbol{y})q(\boldsymbol{\theta}^{(i)},\boldsymbol{\theta}^{(*)})} = \frac{\pi(\boldsymbol{\theta}^{(*)})\mathcal{L}(\boldsymbol{\theta}^{(*)};\boldsymbol{y})q(\boldsymbol{\theta}^{(*)},\boldsymbol{\theta}^{(i)})}{\pi(\boldsymbol{\theta}^{(i)})\mathcal{L}(\boldsymbol{\theta}^{(i)};\boldsymbol{y})q(\boldsymbol{\theta}^{(i)},\boldsymbol{\theta}^{(*)})}$
  if $r > u$, with $u \sim \mathcal{U}(0,1)$ **then**
    $\boldsymbol{\theta}^{(i+1)} = \boldsymbol{\theta}^{(*)}$
    $\tau_{accept} = \tau_{accept} + 1$
  else
    $\boldsymbol{\theta}^{(i+1)} = \boldsymbol{\theta}^{(i)}$
  end if
end for
$\boldsymbol{\theta}_{MH} = \boldsymbol{\theta}[N_{burn-in} : S]$

The choice of the nature of q is problem relevant. To simplify the programing, it is common to choose a Metropolis algorithm. It is difficult to tune all the parameters of a Metropolis algorithm, so it simplifies this problem

if a symmetric proposal distribution is taken. Between the normal or the uniform random walk, the major difficulty is to tune the parameter k or δ. The performance of such an algorithm is evaluated by its faculty to explore the input parameter space $\mathscr{Q}$. So, if $\mathscr{Q} \subset \mathbb{R}^p$ with p high, then it is hard to find the right convergence direction.

In the case where the Metropolis is chosen with a normal proposal distribution, some improvements can be brought especially when p is high.

2.5.3 Metropolis within Gibbs

When the number of parameter of parameter is high, the input parameter space becomes a space where it is complicated to move in the right direction. If the access of conditional distributions are available, one solution could be to compute the ratio according to the conditional distribution relative to the active parameter. Then, the algorithm would move parameter by parameter, that increases the efficiency of the displacement. The ratio introduced in Equation (2.112) becomes:

$$r = \frac{\pi(\boldsymbol{\theta}_j^{(*)}|\boldsymbol{\theta}_{-j}^{(*)}, y)}{\pi(\boldsymbol{\theta}_j^{(s)}|\boldsymbol{\theta}_{-j}^{(s)}, y)} = \frac{\pi(\boldsymbol{\theta}_j^{(*)})\mathscr{L}(\boldsymbol{\theta}_j^{(*)}; \boldsymbol{\theta}_{1\dots j-1}^{(*)}, \boldsymbol{\theta}_{j+1\dots p}^{(*)}, y)}{\pi(\boldsymbol{\theta}_j^{(S)})\mathscr{L}(\boldsymbol{\theta}_j^{(s)}; \boldsymbol{\theta}_{1\dots j-1}^{(s)}, \boldsymbol{\theta}_{j+1\dots p}^{(s)}, y)}. \tag{2.114}$$

Algorithm 4 Metropolis within Gibbs algorithm

$\boldsymbol{\theta}^{(1)} = \boldsymbol{\theta}_{init}$
$\tau_{accept} = (0, \dots, 0)^T$
for i in $1 : S$ **do**
  for j in $1 : p$ **do**
    $\boldsymbol{\theta}_j^{(*)} \sim \mathscr{N}(\theta_j^{(i)}, k\boldsymbol{\Sigma}[j, j])$
    $r = \frac{\pi(\boldsymbol{\theta}_j^{(*)})\mathscr{L}(\boldsymbol{\theta}_j^{(*)}; \boldsymbol{\theta}_{1\dots j-1}^{(*)}, \boldsymbol{\theta}_{j+1\dots p}^{(*)}, y)}{\pi(\boldsymbol{\theta}_j^{(i)})\mathscr{L}(\boldsymbol{\theta}_j^{(i)}; \boldsymbol{\theta}_{1\dots j-1}^{(i)}, \boldsymbol{\theta}_{j+1\dots p}^{(i)}, y)}$
    if $r > u$, with $u \sim \mathscr{U}(0, 1)$ **then**
      $\theta_j^{(i+1)} = \theta^{(*)}$
      $\tau_{accept_j} = \tau_{accept_j} + 1$
    else
      $\theta_j^{(i+1)} = \theta_j^{(i)}$
    end if
  end for
end for
$\boldsymbol{\theta}_{MWG} = \boldsymbol{\theta}[N_{burn-in} : S]$

The strength of Algorithm 4 is to sample well in a multidimensional space. However, as it is moving component by component, it is more time consuming than a regular Metropolis or Metropolis Hastings algorithm.

2.5.4 Improvements of the Metropolis Hastings

The major difficulty in applying a Metropolis Hastings algorithm is to use a right proposal distribution such as the new point point is neither too far from the previous one or too close. The acceptation ratio controls the quality of the chain. If the acceptation ratio is too high, it means that the number of the new accepted points is too high and that the proposal distribution is not adequate to the problem because it generates points too close to the previous one. Similarly, a too low acceptation ratio is synonym of a proposal distribution that generates points that are too far from the previous one. Let us consider for the rest of this section (and for the major part of the thesis) that the considered proposal distribution is normal centered on the previous point and with variance covariance $k\boldsymbol{\Sigma}$. The proposal distribution is in that case symmetric and the major difficulty is to find the right k and $\boldsymbol{\Sigma}$ that make a good

proposal distribution.

To tune the matrix Σ is tricky because before running the Metropolis algorithm, the structure of the covariance matrix that is representative of the space is unknown. Especially, if correlation exists between parameters. A Laplace approximation can be performed upstream to get an initial point (which is the estimate Maximum A Posteriori) and a covariance structure which comes from the Hessian of the likelihood. This can be burdensome if the number of parameter is high. An algorithm, called Adaptive Metropolis, has been developed by Haario et al. (2001) that tries to improve the original Metropolis algorithm. It mainly proposes to run a certain amount, say 1000, of samples, with at first $\Sigma = \mathbb{I}_p$ and $k = 1$, and evaluates the covariance matrix on the 1000 samples generated. Then, by changing Σ to this new calculated covariance matrix, it improves the proposal distribution based on the samples already simulated. The direction of the proposal distribution has better features than the previous one because it takes now into account, possible, correlation between parameters and variances corresponding to the right intensities to move in the posterior density. The Adaptive Metropolis (Algorithm 5) also continues this adaptation each 500 iterations to converge toward the right covariance matrix and to finally have a proper proposal distribution.

Algorithm 5 Adaptive Metropolis

$\theta^{(1)} = \theta_{init}$
$\tau_{accept} = 0$
$\Sigma = \mathbb{I}_p$
for i in $1 : S$ **do**
  $\theta^{(*)} \sim \mathcal{N}(\theta^{(i)}, k\Sigma)$
  $r = \frac{\pi(\theta^{(*)}|y)}{\pi(\theta^{(i)}|y)} = \frac{\pi(\theta^{(*)})\mathcal{L}(\theta^{(*)};y)}{\pi(\theta^{(i)})\mathcal{L}(\Theta^{i};y)}$
  if $r > u$, with $u \sim \mathcal{U}(0,1)$ **then**
    $\theta^{(i+1)} = \theta^{(*)}$
    $\tau_{accept} = \tau_{accept} + 1$
  else
    $\theta^{(i+1)} = \theta^{(i)}$
    if $i \; mod \; A = 0 \quad \& \quad i > B$ **then**
      $\Sigma = cov(\theta[1 : i])$
    end if
  end if
end for
$\Theta_{AM} = \Theta[N_{burn-in} : S]$

The condition added in Algorithm 5 compared to Algorithm 2 introduces the adaptive behavior of the algorithm where B stands for the threshold to compute the covariance (in the previous example 1000) and A stands for the value that specifies at which steps the covariance matrix is computed after the threshold (in the previous example 500). Ergodicity, reversibility and convergence criteria of MCMC chains of such an algorithm are well explained in Haario et al. (2001) and will not be developed here.

It is also common to focus only on the tuning parameter k. As the mixing quality is related to the acceptance ratio, it is possible to establish an algorithm that, function of the actual acceptation rate, modify the parameter k. In Roberts et al. (1997), it is demonstrated that the "optimal efficiency" of the MCMC chain is obtained for an acceptation rate at 0.234. Algorithm 6 introduces such an algorithm that adapts the parameter k, each C iterations, function of the actual acceptation rate.

Algorithm 6 k tuned Metropolis

$\theta^{(1)} = \theta_{init}$
$\tau_{accept} = 0$
for i in $1 : S$ **do**
  $\theta^{(*)} \sim \mathcal{N}(\theta^{(i)}, k\Sigma)$
  $r = \frac{\pi(\theta^{(*)}|y)}{\pi(\theta^{(i)}|y)} = \frac{\pi(\theta^{(*)})\mathcal{L}(\theta^{(*)};y)}{\pi(\theta^{(i)})\mathcal{L}(\Theta^{i};y)}$
  if $r > u$, with $u \sim \mathcal{U}(0,1)$ **then**
    $\theta^{(i+1)} = \theta^{(*)}$
    $\tau_{accept} = \tau_{accept} + 1$
  else
    $\theta^{(i+1)} = \theta^{(i)}$
    if $i \; mod \; C = 0$ **then**
      if $\tau_{accept}/S < 0.2$ **then**
        $k = k(1-r)$
      else
        if $\tau_{accept}/S > 0.5$ **then**
          $k = k(1+r)$
        end if
      end if
    end if
  end if
end for
$\Theta_{KM} = \Theta[N_{burn-in} : S]$

To improve drastically the performances the MCMC chain, the Adaptive Metropolis can be adapted by running upstream a Metropolis within Gibbs algorithm on a limiter number of iterations. The aim is to move in the right direction (but slowly) with the Metropolis within Gibbs, to estimate a covariance structure. Once this covariance is known, a regulation on the parameter k can still be useful to better tune the displacement in the space $\mathcal{Q}$. The regulation is stopped after an amount of iterations to keep convergence properties. Indeed, the convergence properties are preserved when changes in the chain decrease. This hybrid algorithm (detailed Algorithm 7) is the one used all along the thesis to perform MCMC sampling estimation.

Algorithm 7 Hybrid Metropolis

$\theta^{(1)} = \theta_{init}$
$\tau_{accept} = (0,\dots,0)^T$
for i in $1 : S_{MWG}$ **do**
  for j in $1 : p$ **do**
    $\theta_j^{(*)} \sim \mathcal{N}(\theta_j^{(i)}, k\Sigma[j,j])$
    $r = \frac{\pi(\theta_j^{(*)})\mathcal{L}(\theta_j^{(*)};\theta^{(*)}_{1\dots j-1},\theta^{(*)}_{j+1\dots p},y)}{\pi(\theta_j^{(i)})\mathcal{L}(\theta_j^{(i)};\theta^{(i)}_{1\dots j-1},\theta^{(i)}_{j+1\dots p},y)}$
    if $r > u$, with $u \sim \mathcal{U}(0,1)$ **then**
      $\theta_j^{(i+1)} = \theta^{(*)}$
      $\tau_{accept_j} = \tau_{accept_j} + 1$
    else
      $\theta_j^{(i+1)} = \theta_j^{(i)}$
    end if
  end for
end for
$\theta_{MWG} = \theta[N_{burn-in} : S_{MWG}]$
$\Sigma = cov(\theta_{MWG})$
$\theta^{(1)} = \theta_{init}$
$\tau_{accept} = 0$
for i in $1 : S_M$ **do**
  $\theta^{(*)} \sim \mathcal{N}(\theta^{(i)}, k\Sigma)$
  $r = \frac{\pi(\theta^{(*)}|y)}{\pi(\theta^{(i)}|y)} = \frac{\pi(\theta^{(*)})\mathcal{L}(\theta^{(*)};y)}{\pi(\theta^{(i)})\mathcal{L}(\Theta^i;y)}$
  if $r > u$, with $u \sim \mathcal{U}(0,1)$ **then**
    $\theta^{(i+1)} = \theta^{(*)}$
    $\tau_{accept} = \tau_{accept} + 1$
  else
    $\theta^{(i+1)} = \theta^{(i)}$
    if $i \; mod \; C = 0$ **then**
      if $\tau_{accept}/S_M < 0.2$ **then**
        $k = k(1-r)$
      else
        if $\tau_{accept}/S_M > 0.5$ **then**
          $k = k(1+r)$
        end if
      end if
    end if
  end if
end for
$\Theta_{HM} = \Theta[N_{burn-in} : S_M]$

Some other developments have been completed for improving the Metropolis algorithm as the Delayed rejection introduced in Mira et al. (2001). It consists in, in case of rejection, proposing another point, based on the same proposal distribution, with a probability of 0.5 (sampled from a binomial distribution). If the realization of the binomial is 1, then a new point is proposed and the ratio is computed as if it was the first point. If another rejection occurs, the same procedure is repeated but if it is accepted, the current iteration in the algorithm can end. If the realization if the binomial is 0, the rejection is accepted and the iteration of the algorithm can also stop. This algorithm allows to postpone the rejection and gives another chance to accept a specific point. The aim of this method is not to stay on the same point which tends to increase autocorrelation.

CHAPTER 3

REVIEW OF THE MAIN CALIBRATION METHODS

The general framework of this chapter is the one introduce in Section 1.4.1 where we considered, the outputs of ζ and f_c lie in $\mathbb{R}$. A vector of experimental data ($\boldsymbol{y}_{exp}$) which are noisy measurements of ζ is observed as a realization of the statistical model:

$$\mathscr{M}_0 \ : \ \forall i \in [\![1,\dots,n]\!] \quad y_{exp_i} = \zeta(\boldsymbol{x}_i) + \varepsilon_i$$

where $\forall i \in [\![1,\dots,n]\!] \quad \varepsilon_i \overset{iid}{\sim} \mathcal{N}(0,\sigma_{err}^2)$. The corresponding values of the variables $\boldsymbol{x}_i$ are also observed.

Code calibration

Calibrating the code consists in setting the vector of parameters $\boldsymbol{\theta}$ consistent in some sense with these n field data.

Several statistical modeling strategies have been proposed in the literature. When only measurement errors are considered, Cox et al. (2001) use a rather simple model, considering that the code does not differ from the phenomenon under study. As for Higdon et al. (2004), Kennedy and O'Hagan (2001) and Bayarri et al. (2007), they advocate some extensions, which additionally encompass a *model bias* or a *model error* term, also dubbed *discrepancy* in the following. All of these models are reviewed and discussed in Section 3.2. The identifiability issues between the parameter θ and the discrepancy were already discussed in the written discussions of Kennedy and O'Hagan (2001). Tuo et al. (2015) consider the calibration task as a minimization of a loss function between

the code and the physical reality. In Tuo and Wu (2016), they show that this loss function leads to an estimation of θ depending on the the chosen prior distribution of the discrepancy. Then, Plumlee (2017) advices an orthogonality specification for the discrepancy i.e. the discrepancy should be orthogonal to the gradient of the computer code with respect to a loss function. Finally, Konomi et al. (2017) also propose a methodology to model the discrepancy as a non-stationary Gaussian process and apply it on the carbon capture with AX sorbent mathematical function.

This chapter presents the main calibration statistical model that one can found in literature which are illustrated by an application case introduced in Section 1.4.3. Section 3.1 focuses on sensitivity analysis performed for the numerical code and the *prior* propagation of uncertainty, then Section 3.2 recalls the main statistical models for code calibration with the associated likelihoods and the main estimation methods. Then Section 3.3 applies and compares the four statistical models through the application case.

3.1 Numerical code

3.1.1 Sensitivity analysis

Based on the code presented in Section 1.4.3, a sensitivity analysis is run upstream calibration to identify the important parameter to keep for further study. A Morris method is first applied to screen, upstream, the non-influent parameters that can be set to their nominal value for the rest of the study. Figure 3.1 illustrates the Morris study performed on the numerical code.

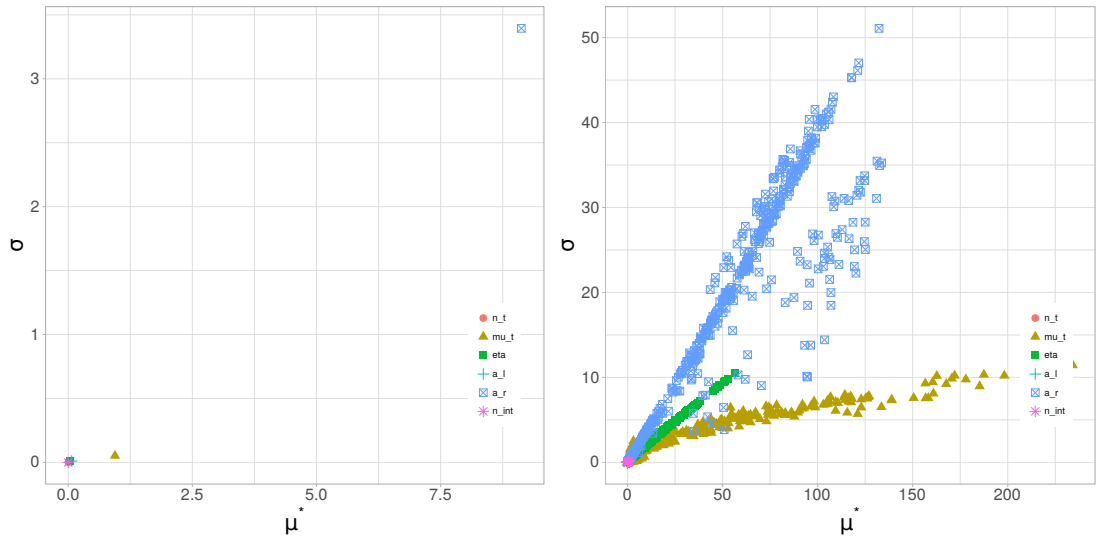

Figure 3.1: On the left panel Morris method at noon the 24^{th} of September 2014 and all the EEs computed at each time step over the two months of data.

However, as described in Section 2.1.1, Morris method only concerns functions that create a scalar output. In this particular case, it is necessary to have an indication of the parameter impacts over the whole time frame. On the left panel of Figure 3.1, the Morris method is applied at noon the 24^{th} of September 2014. The impact of several parameters seems negligible in the graph but they could not be negligible on another moment of the day. When the Morris method is completed for each time step, a "dynamic" version is visible on the right panel of Figure 3.1. Thus, it is not possible to conclude on which parameter has no influence on all the time frame of the data. That is why, a PCA (Section 2.4) on all the trajectories of the Morris DOE is performed. In the new subspace, "new" indices μ^* and σ^2 can be computed and then summarize the information contained in the whole time frame. It means that the PCA allows to visualize eventual temporal correlation and to find new representation axis for the

Morris indices. The left panel of Figure 3.2 represents the circle of correlation obtained by the PCA and on the right panel the eigenvalues for the ten first axis found.

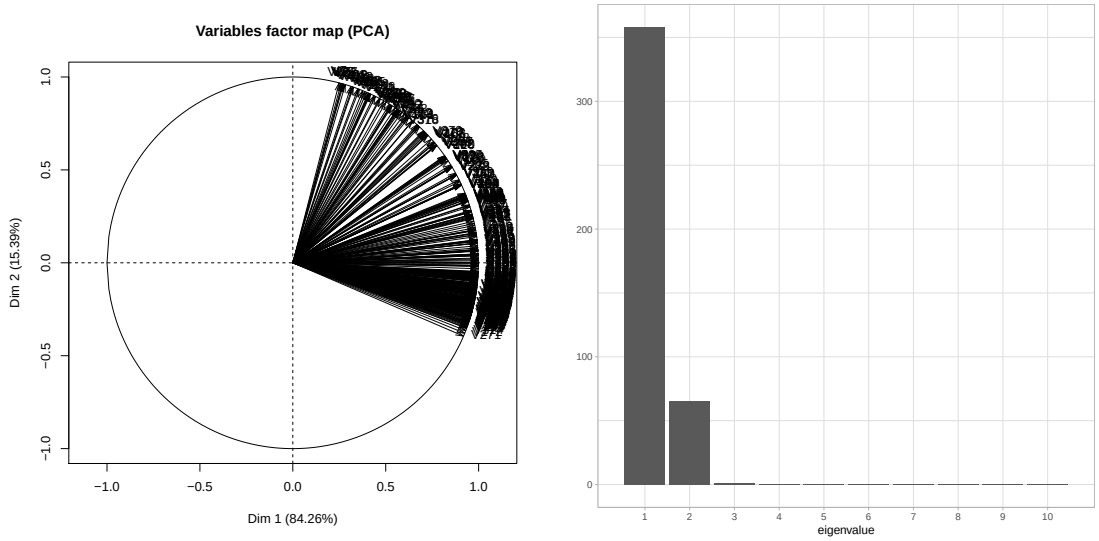

Figure 3.2: Results of the PCA done on the trajectories of the Morris DOE. On the right panel the correlation circle and on the right panel the eigenvalues.

From Figure 3.2, three axis of representation can be chosen because they are covering more than 99% of the information. The different Morris indices in the PCA subspace composed of these axes are visible in Figure 3.3.

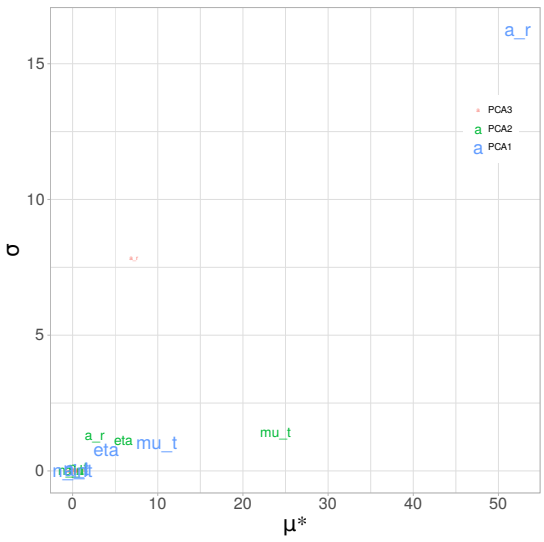

Figure 3.3: Projection on the PCA axis of the Morris indices.

Figure 3.3 is more visual than the one presented on the right panel of Figure 3.1. It allows to conclude on the non-influent parameter and to keep as random η, μ_t and a_r. As this numerical code is quick to be executed, these results are double checked with a Sobol method. It is also a way to better acknowledge the input parameters, especially if interaction between them and/or non linear effects are quantifiable. The results of the first order indices and the total effects indices are visible in Figure 3.4. The Sobol indices has been computed for each time step, which allows to state on the evolution of the influence of a parameter over the time frame.

Figure 3.4 confirms the results obtained by the Morris method after the PCA visible in Figure 3.3. The three

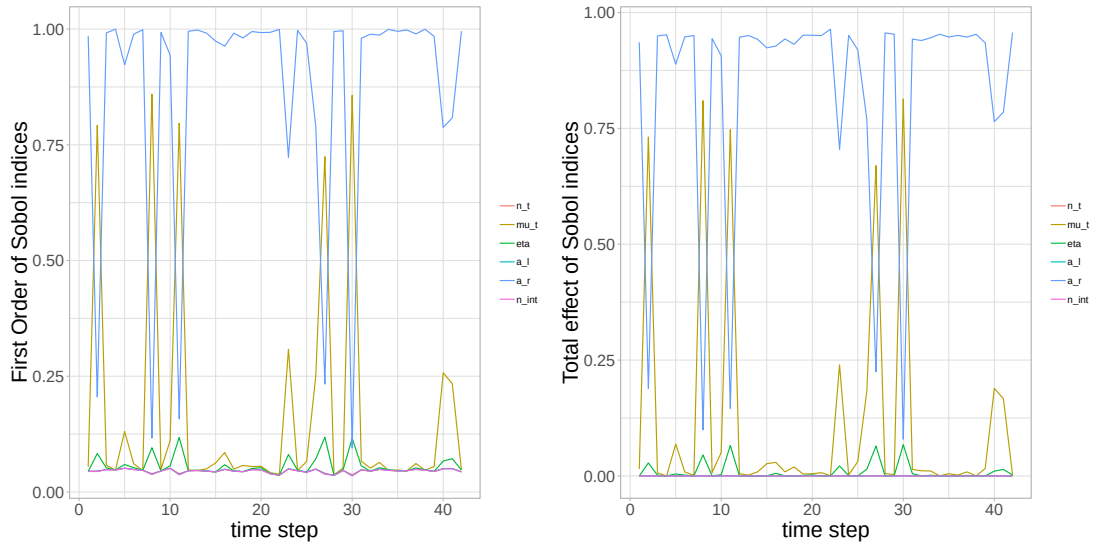

Figure 3.4: Sobol method completed for each time steps. On the left panel the first order indices and on the right panel the total effects indices.

parameters to consider as random are η, μ_t and a_r. With the Sobol method, it is also possible to sort the order of importance of the parameters and in this case a_r is the most influent all along the time frame. One can also note that the differences between the first order indices and the total effects are practically equal which means that no non-linear effects affect the input parameters.

3.1.2 Prior propagation of uncertainty

So far, experts are using the code with some parameter values with the knowledge that these parameters are uncertain (the so called reference values). They can also provide more expertise on the nature of the parameter. For example for a_r, the nominal value is 0.17 and experts state that the parameter lies within the 95% confidence interval $[0.05, 0.29]$. We chose to consider a_r as Gaussian with $a_r \sim \mathcal{N}(\mu = 0.17, \sigma^2 = 3.6.10^{-3})$. The standard deviation is chosen equal to 0.06 because we have considered the upper bound and the lower bound of the given interval as respectively the quantiles $a_{r_{0.975}}$ and $a_{r_{0.025}}$. Similarly, η and μ_t are taken as Gaussian such as $\eta \sim \mathcal{N}(\mu = 0.143, \sigma^2 = 2.5.10^{-3})$ and $\mu_t \sim \mathcal{N}(\mu = -0.4, \sigma^2 = 10^{-2})$. If 100 realizations are drawn from the joint distribution of η, μ_t and a_r, the production curve and the *prior* credibility interval can be simultaneously plotted on a same graph to see how uncertain the predicted power is over a day. Figure 3.5 illustrates on the left the distribution of η, μ_t and a_r and on the right the production curve obtained for reference values and the *prior* credibility interval at 90%. On the right side, experiments collected that same day are also displayed. One can check that the prior credibility interval, build given by the experts, looks coherent regarding the experimental data.

If one is interested in the energy produced rather than the power (the energy in kWh is the power in kW multiplied by a duration), one can easily compute the maximum and the minimum energy for say 100 realizations. The energy for collected power is $W_{exp} = 3.44kWh$, the maximum energy computed $W_{max} = 3.65kWh$ and the minimum energy $W_{min} = 2.93kWh$. Straightforwardly $W_{min} < W_{exp} < W_{max}$ which means that the experts interval seems correct for that day. With the considered uncertainty on η, μ_t and a_r, the error made is about 20% over only one day. Considering this error over a day, cumulative error over a lifetime plant could be too prejudicial. The aim of the calibration is to quantify this error and, at the same time, increase the knowledge on the parameter distribution. The results of calibration for this application case are detailed in Section 3.3.

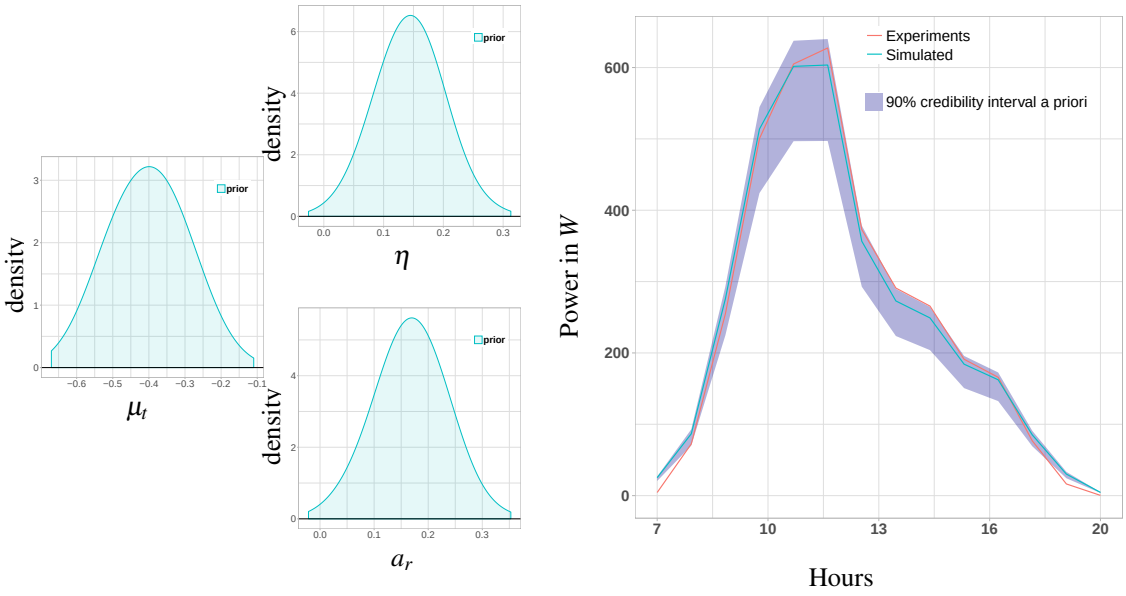

Figure 3.5: $\pi(\eta)$, $\pi(\mu_t)$ and $\pi(a_r)$ prior densities (represented on the left panel) and induced credibility interval of the instantaneous power (right panel).

3.2 Calibration through statistical models

Calibration intends to find the "best fitting" parameters of a computational code, in order to minimize the difference between the output and the experiments. It can be used in two cases. In a forecasting context (Craig et al., 2001), calibrated code on data collected on site can be used to compute the behavior of the power plant over the next time period. In a prediction context, data from an experimental stand are used to predict the behavior of a non-existing stand (assuming they have the same features).

A simple way to express calibration is to write down a first, straightforward model. The computational code is set up to entirely replace the physical system. Intuitively, we can assume that $\forall x \in \mathscr{H}, \zeta(x) = f_c(x,\theta)$ for some well-chosen θ which leads to the following equation:

Model1

$$\mathscr{M}_1 \ : \ \forall i \in [\![1,\dots,n]\!] \quad y_{exp_i} = f_c(x_i,\theta) + \varepsilon_i, \tag{3.1}$$

with $\forall i \in [\![1,\dots,n]\!] \quad \varepsilon_i \overset{iid}{\sim} \mathscr{N}(0,\sigma_{err}^2)$.

The likelihood of such a model is a function of f_c. In methods such as Maximum Likelihood Estimation (MLE) or as in Bayesian estimation (making recourse to many MCMC iterations), it becomes intractable to work with a time consuming f_c.

For the sake of simplicity we will consider, in what follows, the code as deterministic. It means that for the same inputs, the output of the code is identical, which is generally the case. Even in a deterministic context, a gap between the code and the physical system is often unavoidable. This gap is called code error or discrepancy. Some papers advocate for adding this discrepancy in the statistical models (Kennedy and O'Hagan, 2001; Higdon et al., 2004; Bayarri et al., 2007; Bachoc et al., 2014). In the following, we present three models which take into account a

time consuming code and/or an additional discrepancy.

3.2.1 Presentation of the models

A time consuming code

Let us consider a time consuming code. As said above, in this particular case, the computational burden become too huge to perform calibration. That is why Sacks et al. (1989) introduced an emulation of the not yet computed outputs from the code by a random function, *i.e.* a stochastic process. The common choice is oriented toward the Gaussian process because the conditional Gaussian process is still a Gaussian process (see Section 2.2 for more details). It is, parsimoniously, defined by its mean and covariance functions. The first "simple" and straightforward model was introduced by Cox et al. (2001) which uses this emulation of f_c.

Model 2

$$\begin{aligned} \mathscr{M}_2 \ : \ \forall i \in [\![1,\dots,n]\!] \quad y_{exp_i} &= F(x_i,\theta)+\varepsilon_i \\ F(\bullet,\bullet) &\sim \mathscr{GP}\Big(m_S(\bullet,\bullet), c_S\{(\bullet,\bullet),(\bullet,\bullet)\}\Big) \end{aligned} \tag{3.2}$$

where $\forall i \in [\![1,\dots,n]\!] \quad \varepsilon_i \overset{iid}{\sim} \mathscr{N}(0,\sigma^2_{err})$ and the random function $F(x_i,\theta)$ stands for a Gaussian process (GP) over the joint domain of x_i and θ. Its mean function $m_S(x_i,\theta)$ is generally a linear form of simple functions of x_i and θ and its covariance function $c_S\{(x_i^*,\theta^*),(x_i,\theta)\} = \sigma_S^2 r_{\psi_S}\{(x_i^*,\theta^*),(x_i,\theta)\}$ is such as the function $r_{\psi_S}\{(\bullet,\bullet),(\bullet,\bullet)\}$ is the correlation function with a vector parameter ψ_S which is the scale and the regularity of the kernel and where σ_S^2 represents the variance. The mean $m_S(x_i,\theta)$ can be written:

$$m_S(x_i,\theta) = m_{\beta_S}(x_i,\theta) = \mathbb{E}[F(x_i,\theta)] = \beta_{S_0} + \sum_{j=1}^{M} \beta_{S_j} h_{S_j}(x_i,\theta) = h_S(x_i,\theta)\beta_S \tag{3.3}$$

where $\beta_S^T = (\beta_{S_0},\dots,\beta_{S_M})$ is the coefficient vector to be estimated and $h_S(\bullet,\bullet) = (h_{S_0}(\bullet,\bullet),\dots,h_{S_M}(\bullet,\bullet))$ the row vector of regression functions where $h_{S_0} = 1$. Similarly we define the $n \times (M+1)$ matrix $H_S(X,\theta)$ such as its i^{th} row is $h_S(x_i,\theta)$. The correlation function can take multiple forms as Gaussian or Matérn for instance (see Santner et al., 2013, for more examples). We will consider, for now and for all theoretical developments, the general form of $c_S\{(\bullet,\bullet),(\bullet,\bullet)\} = \sigma_S^2 r_{\psi_S}\{(\bullet,\bullet),(\bullet,\bullet)\}$ where σ_S^2 is the variance and r is the correlation function with a parameter vector ψ_S. The advantage of using a surrogate model, for $f_c(X,\theta)$ is to alleviate the computational burden, at the cost of adding an additional source of uncertainty, and of increasing the number of uncertain parameters. Specific hypotheses, for instance a known smoothness of the random field, may help to choose the size of the parametric family in which the correlation shape is to be assessed.

When the code is time consuming, a fixed number N of simulations is set up. The ensuing simulated data (we will call them y_c) are usually the image of a design of experiments (DOE) representative of the input space. Some interesting developments have been made on using the least possible points in the input space with some wise repartitions (the *Latin Hilbert Space* sampling is one example, some good insights are available in Pronzato and Müller (2012)).

Let us call D a DOE, a set of N points sampled in the input space defined as the product of $\mathscr{H}$ and $\mathscr{Q}$. We can write $D = \{(x_1^D,\tau_1^D),\dots(x_N^D,\tau_N^D)\}$ where $\forall i \in [\![1,\dots,N]\!]$ (x_i^D,τ_i^D) are chosen in $\mathscr{H}\times\mathscr{Q}$. The establishment of the

DOE will lead to simulated data which are defined as $y_c = f_c(D)$. The error made by the surrogate strongly depends on the numerical design of experiments used to fit the emulator. Adaptive numerical designs introduced in Damblin et al. (2018) is a way to enhance the emulator when the goal is to calibrate the code.

With a code error

Considering the computational code as a perfect representation of the physical system may be a too strong hypothesis and it is legitimate to wonder whether the code might differ from the phenomenon. This error (called discrepancy and introduced above) can be defined as:

$$\delta(x_i) = \zeta(x_i) - f_c(x_i, \theta).$$

In all the works cited above, this unknown discrepancy is modeled as a realization of a Gaussian process, this time yielding a random function over the domain of X variables only. For the sake of simplicity we will denote by m_s, c_S ($c_{\sigma_S^2,\psi_S}$) and r_S (r_{ψ_S}) the mean, covariance (with σ_S^2 as the variance) and correlation function relative to the surrogate and by m_δ, c_δ ($c_{\sigma_\delta^2,\psi_\delta}$) and r_δ (r_{ψ_δ}) the same functions relative to the discrepancy (respectively σ_δ^2 for the variance in the covariance function). Note that m_δ and c_δ are functions of x only and not θ. The aim of adding the discrepancy lies in the fact that correlation is sometimes visible in the residuals and/or that no value of θ makes the computer close to experiments. However, the discrepancy could lead to an identifiability issue. For example, it could easily exist two couples $(\theta, \delta(x_i))$ and $(\theta^*, \delta^*(x_i))$ that verify the two equalities: $\delta(x_i) = \zeta(x_i) - f_c(x_i, \theta)$ and $\delta^*(x_i) = \zeta(x_i) - f_c(x_i, \theta^*)$. Some papers (Higdon et al., 2004; Bachoc et al., 2014; Bayarri et al., 2007) advocate to set the mean of the discrepancy to 0 to solve this identifiability issue. The contribution of the discrepancy is widely discussed in literature and make the object of comparative studies in validation (Damblin, 2015).

When the code is not time consuming, the real code f_c is used:

Model 3

$$\mathscr{M}_3 \ : \ \forall i \in [\![1,\dots,n]\!] \quad \begin{array}{rcl} y_{exp_i} & = & f_c(x_i,\theta) + \delta(x_i) + \varepsilon_i \\ \delta(\bullet) & \sim & \mathscr{GP}\Big(m_\delta(\bullet), c_\delta(\bullet,\bullet)\Big) \end{array} \tag{3.4}$$

where $\forall i \in [\![1,\dots,n]\!] \quad \varepsilon_i \overset{iid}{\sim} \mathscr{N}(0,\sigma_{err}^2)$, and $\delta(\bullet)$ stands for a Gaussian process which mimics the discrepancy and will only depends on the input variables x. We can write $\delta(\bullet) \sim \mathscr{GP}(m_\delta(\bullet), c_\delta(\bullet,\bullet))$ with $\forall x,\ m_\delta(x) = h_\delta(x)\beta_\delta$ (where h_δ is a row vector and β_δ is a column vector if we choose a parametric representation of the mean) and c_δ the covariance function of the discrepancy. We also denote $H_\delta(X)$ the n row matrix, the i^{th} row of which is $h_\delta(x_i)$.

When the code is time consuming, the systematic use of f_c is not computationally acceptable. Then, as for Model $\mathscr{M}_2$, the code is replaced with a Gaussian process. This leads to the more generic model introduced in Kennedy and O'Hagan (2001).

Model 4

$$\mathscr{M}_4 \ : \ \forall i \in [\![1,\dots,n]\!] \quad y_{exp_i} = F(x_i,\theta) + \delta(x_i) + \varepsilon_i \tag{3.5}$$

where $\forall i \in [\![1,\dots,n]\!]$ $\varepsilon_i \overset{iid}{\sim} \mathcal{N}(0,\sigma^2_{err})$, $F(x_i,\theta)$ and $\delta(x_i)$ are two Gaussian processes defined as before. In their model, Kennedy and O'Hagan (2001) also used a scale parameter ρ in front of F. This parameter is usually set to 1 in many works in order to achieve the best estimate on θ. Thus, we omit this parameter in the model definition.

A quantification of the bias form is the aim of both models. If we are interested in improving the computational code or its surrogate, it is usually fair to set the mean of the discrepancy to zero and find the best tuning parameter vector which compensates a potential bias (Higdon et al., 2004; Bachoc et al., 2014).

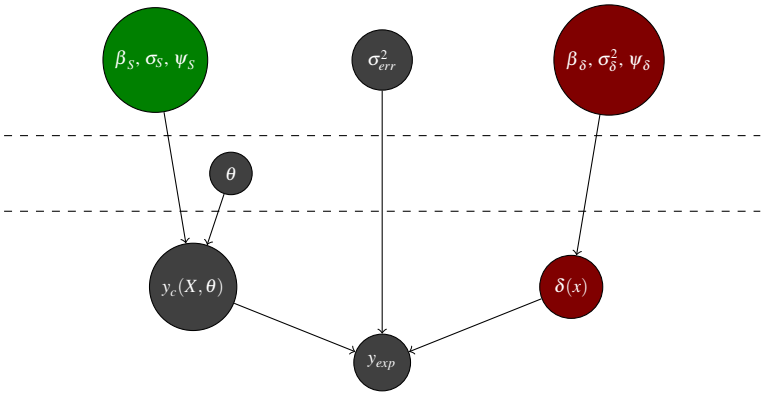

Figure 3.6: Directed Acyclic Graph (DAG) representation of the different models.

Figure 3.6 is a summary of all the models introduced above. The directed acyclic graph (DAG) allows us to compare the structures of all the previously introduced models. Specifically: if one considers only the grey nodes, the obtained DAG corresponds to Model $\mathscr{M}_1$. Adding the green node, the resulting DAG represents $\mathscr{M}_2$. Considering the grey and red nodes, yields a DAG for model $\mathscr{M}_3$ and the whole DAG represents the general model $\mathscr{M}_4$. Note that two categories of parameters are considered. The tuning parameters are only related to the code and other parameters (also called nuisance parameter) concern the measurement error, the surrogate or the discrepancy introduced in the models. In calibration, we only focus on the value of θ but the other parameters introduced need to be estimated as well. We will dig into these estimation issues in Section 3.2.3.

All these models introduce new parameters and need to be estimated as well as tuning parameters. Estimation needs to dive into technical aspects such as writing the likelihood for each model. The following section provides all the elements required to go one step further and carry out estimation.

3.2.2 Likelihood

For estimating parameters (whatever framework used, Bayesian or Maximization Likelihood Estimation (MLE)), expressing the likelihood comes as the first requirement. Two major categories stand out. When the code is not time consuming, the main issue in code calibration (*i.e.* the computational time burden) is avoided. When the code is time consuming, new parameters have to be taken into account and to be estimated. That is why, in the cases 1 and 3 data are only field data y_{exp} and in the cases 2 and 4, numerical data (outputs of the code) are added to form the whole data vector $y^T = (y^T_{exp}, y^T_c)$. In what follows, we will denote by θ^* the true parameter vector. Note that it is well-defined only in Models $\mathscr{M}_1$ and $\mathscr{M}_2$, as the value of θ which satisfies : $\zeta(x) = f_c(x,\theta^*)$ for all possible x, (assuming such a θ exists and is unique). On the other hand, the models $\mathscr{M}_3$ and $\mathscr{M}_4$ are both defined by the relation $\zeta(x) = f_c(x,\theta) + \delta(x)$, which holds for infinitely many couples $(\theta, \delta(\bullet))$, as discussed earlier. Kennedy and O'Hagan (2001) avoid this issue by defining θ^* as a "best-fitting" value, but it is unclear what this means

exactly (see the discussion section of their paper for further details).

In order to simplify the notation, we will use for the rest of the paper $\Phi = \{\sigma_S^2, \sigma_\delta^2, \psi_S, \psi_\delta\}$ and $\Phi_S = \{\sigma_S^2, \psi_S\}$ and $\Phi_\delta = \{\sigma_\delta^2, \psi_\delta\}$, where σ_S^2 and σ_δ^2 are the variances of the two Gaussian processes respectively relative to the surrogate and the discrepancy. The two parameter vectors ψ_S and ψ_δ are relative to the correlation functions. Let us call $\beta^T = (\beta_S^T, \beta_\delta^T)$ the vector of collected coefficient vectors.

Both cases of time consuming or not time consuming code will be dealt with. The likelihood equations will be written for the generic forms of $\mathscr{M}_3$ and $\mathscr{M}_4$. The likelihoods for the simpler models $\mathscr{M}_1$ and $\mathscr{M}_2$ will be then derived since $\mathscr{M}_1 \subset \mathscr{M}_3$ and $\mathscr{M}_2 \subset \mathscr{M}_4$.

A fast code

The generic model which deals with calibration with a code quick to run is detailed in Equation (3.4). Experimental data are the only one needed because simulation data are free but will not bring additional information for the parameters of Model $\mathscr{M}_3$. Experimental data follow a Gaussian distribution, the expectation of which is:

$$\mathbb{E}[y_{exp}|\theta, \beta_\delta; X] = m_{exp}^{\beta_\delta}(X, \theta) = m_{exp}(X, \theta) = f_c(X, \theta) + H_\delta(X)\beta_\delta.$$

Then, the expression of the variance is given by:

$$\mathbb{V}ar[y_{exp}|\Phi_\delta; X] = V_{exp}^{\Phi_\delta, \sigma_{err}^2}(X) = V_{exp}(X) = \Sigma_\delta(X) + \sigma_{err}^2 I_n$$

with $\forall (i,j) \in [\![1, \ldots, n]\!]^2 : (\Sigma_\delta(X))_{i,j} = (\Sigma_\delta^{\Phi_\delta}(X))_{i,j} = c_\delta(\{x_i, x_j\})$. The likelihood in this particular case can be written as

$$\mathscr{L}^F(\theta, \beta_\delta, \Phi_\delta; y_{exp}, X) = \frac{1}{(2\pi)^{n/2}|V_{exp}(X)|^{1/2}} \exp\left\{ -\frac{1}{2}\Big(y_{exp} - m_{exp}(X,\theta)\Big)^T V_{exp}(X)^{-1} \Big(y_{exp} - m_{exp}(X,\theta)\Big)\right\}. \tag{3.6}$$

This likelihood is relative to Model $\mathscr{M}_3$ (Equation (3.4)). For the specific case, where no discrepancy is considered (corresponding to $\mathscr{M}_1$ Equation (3.1)) the likelihood can be written in a similar way but with $m_{exp}(X,\theta) = f_c(X,\theta)$ and $V_{exp}(X) = \sigma_{err}^2 I_n$. Note that the covariance matrix depends only on ${\sigma_{err}}^2$. It implies that if we seek to estimate the *posterior* density on θ (in a Bayesian framework), this covariance term is superfluous.

Then the likelihood can be rewritten in an simpler way:

$$\mathscr{L}^F(\theta, \sigma_{err}^2; y_{exp}, X) = \frac{1}{(2\pi)^{n/2}\sigma_{err}^n} \exp\left\{ -\frac{1}{2\sigma_{err}^2}||y_{exp} - f_c(X,\theta)||_2^2\right\}. \tag{3.7}$$

The models using the code with or without the discrepancy do look quite similar. For theoretical development, it might be easier to work with the one without discrepancy. From an experimental point of view, it could be interesting to study the role of the code error.

A time consuming code

When a code is time consuming and replaced by a surrogate, additional parameters are to be estimated. As introduced above, a DOE is set up and intends to be a representative sample of the input space (variable and parameter input space). Simulated data from this DOE (called y_c) will constitute additional data for the estimation of the nuisance parameters. Depending on how we consider that two sources of data are linked, multiple likelihoods can be set up. For the theoretical development, we will consider the general model $\mathscr{M}_4$ and we will detail the particular case $\mathscr{M}_2$ hereafter.

The first likelihood useful in estimation is the full likelihood. This one concerns the distribution of all collected data ($y^T = (y_{exp}^T, y_c^T)$). That means, we are interested in estimating the parameters of the distribution $\pi(y|\theta,\beta,\Phi,\sigma_{err}^2;X,D)$ which is Gaussian. The expectations can be written from both expectancies of $\pi(y_{exp}|\theta,\beta,\Phi,\sigma_{err}^2;X)$ and $\pi(y_c|\theta,\beta_S,\Phi_S;X)$.

$$\begin{cases} \mathbb{E}[y_c|\beta_S;D] = m_c^{\beta_S}(D) = m_c(D) = H_S(D)\beta_S \\ \mathbb{E}[y_{exp}|\theta,\beta;X] = m_{exp}^{\beta}(X,\theta) = m_{exp}(X,\theta) = H_S(X,\theta)\beta_S + H_\delta(X)\beta_\delta \end{cases} \tag{3.8}$$

This can be summed up for two component vectors $y^T = (y_{exp}^T, y_c^T)$:

$$\begin{aligned} \mathbb{E}[y|\theta,\beta;X,D] = m_y^{\beta}((X,\theta),D) = m_y((X,\theta),D) = H((X,\theta),D)\beta \\ = \begin{pmatrix} H_S(X,\theta) & H_\delta(X) \\ H_S(D) & 0 \end{pmatrix}\beta. \end{aligned} \tag{3.9}$$

The variance matrix now includes the covariance functions of the discrepancy and the surrogate.

$$\begin{aligned} \mathbb{V}ar[y|\theta,\Phi,\sigma_{err}^2;X,D] = V^{\Phi,\sigma_{err}^2}((X,\theta),D) = V((X,\theta),D) \\ = \begin{pmatrix} \Sigma_{exp,exp}(X,\theta) + \Sigma_\delta(X) + \sigma_{err}^2 I_n & \Sigma_{exp,c}((X,\theta),D) \\ \Sigma_{exp,c}((X,\theta),D)^T & \Sigma_{c,c}(D) \end{pmatrix} \end{aligned} \tag{3.10}$$

where

- $\forall (i,j) \in [\![1,\dots,n]\!]^2 : (\Sigma_{exp,exp}(X,\theta))_{i,j} = c_S\{(x_i,\theta),(x_j,\theta)\}$,
- $\forall (i,j) \in [\![1,\dots,n]\!] \times [\![1,\dots,N]\!] : (\Sigma_{exp,c}((X,\theta),D))_{i,j} = c_S\{(x_i,\theta_i),(x_j^D,\tau_j^D)\}$,
- $\forall (i,j) \in [\![1,\dots,n]\!]^2 : (\Sigma_\delta(X))_{i,j} = c_\delta\{(x_i,x_j)\}$,
- $\forall (i,j) \in [\![1,\dots,N]\!]^2 : (\Sigma_{c,c}(D))_{i,j} = c_S\{(x_i^D,\tau_i^D),(x_j^D,\tau_j^D)\}$.

As a reminder D is the DOE set up to build the surrogate and is defined as $D = \{(x_1^D,\tau_1^D),\dots(x_N^D,\tau_N^D)\}$. The general expression of the full likelihood can then be expressed:

$$\begin{aligned} \mathscr{L}^F(\theta,\beta,\Phi,\sigma_{err}^2;y,X,D) = \frac{1}{(2\pi)^{(n+N)/2}|V((X,\theta),D)|^{1/2}} \exp\Bigg\{ -\frac{1}{2}\Big(y - m_y((X,\theta),D)\Big)^T \\ V((X,\theta),D)^{-1}\Big(y - m_y((X,\theta),D)\Big)\Bigg\}. \end{aligned} \tag{3.11}$$

The particular cases of Bayarri et al. (2007) and Higdon et al. (2004) a zero mean for the discrepancy is considered. Under this condition, we have $m_y((X,\theta),D) = \begin{pmatrix} H_S(X,\theta) \\ H_S(D) \end{pmatrix} \beta_S$ and the other terms remain the same. For the model $\mathscr{M}_2$ where a surrogate is used without any discrepancy (Cox et al., 2001), the expectation becomes:

$$\mathbb{E}[y|\theta,\beta_S;X,D] = m_y((X,\theta),D) = H((X,\theta),D)\beta_S = \begin{pmatrix} H_S(X,\theta) \\ H_S(D) \end{pmatrix} \beta_S \tag{3.12}$$

and the covariance:

$$\mathbb{V}ar[y|\theta,\Phi,\sigma^2_{err};X,D] = V((X,\theta),D) = \begin{pmatrix} \Sigma_{exp,exp}(X,\theta) + \sigma^2_{err} I_n & \Sigma_{exp,c}((X,\theta),D) \\ \Sigma_{exp,c}((X,\theta),D)^T & \Sigma_{c,c}(D) \end{pmatrix} \tag{3.13}$$

where covariances matrices are the same as defined before.

The estimation can be separate into different steps where the partial likelihood (Equation (3.14)) could be useful. This one only concerns simulated data and the corresponding surrogate. The partial likelihoods of the model $\mathscr{M}_2$ and $\mathscr{M}_4$ are then the same. That means we are only interesting in estimating the distribution $\pi(\beta_S,\Phi_S|y_c)$ where $\Phi_S = \{\sigma^2_S,\psi_S\}$. The expectation can be obtained by considering only the mean function of the surrogate (Equation (3.8)) and the variance is straightforwardly linked to the variance of the surrogate.

$$\mathbb{V}ar[y_c|\Phi_S;D] = V_c^{\Phi_S}(D) = V_c(D) = \Sigma_{c,c}(D),$$

where $\forall (i,j) \in [1,\dots,N]^2 : (\Sigma_{c,c}(D))_{i,j} = c_S\{(x_i^D,\theta_i^D),(x_j^D,\theta_j^D)\}$. Let us recall that Equation (3.8) established that $m_c(D) = H_S(D)\beta_S$. It implies that the partial likelihood relative to $\mathscr{M}_4$ and $\mathscr{M}_2$ is:

$$\mathscr{L}^M(\beta_S,\Phi_S;y_c,D) = \frac{1}{(2\pi)^{N/2}|V_c(D)|^{1/2}} \exp\left\{ -\frac{1}{2}\Big(y_c - m_c(D)\Big)^T V_c(D)^{-1}\Big(y_c - m_c(D)\Big)\right\}. \tag{3.14}$$

As the other model introduced by Higdon et al. (2004) and Bayarri et al. (2007) only deals with changes on the discrepancy, the partial likelihood remains identical as the one for the model in Bayarri et al. (2007).

From what has been introduced before, one can write the conditional distribution $\pi(y_{exp}|y_c)$ (see Section 2.2 for more details) from the joint distribution $\pi(y_{exp},y_c)$ and for $\mathscr{M}_4$:

$$\begin{pmatrix} y_{exp} \\ y_c \end{pmatrix} \sim \mathscr{N}\left(\begin{pmatrix} m_{exp}(X,\theta) \\ m_c(D) \end{pmatrix}, \begin{pmatrix} \Sigma_{exp,exp}(X,\theta) + \Sigma_\delta + \sigma^2_{err} I_n & \Sigma_{exp,c}((X,\theta),D) \\ \Sigma_{exp,c}((X,\theta),D)^T & \Sigma_{c,c}(D) \end{pmatrix} \right)$$

where m_c and m_{exp} are defined Equation (3.8) and covariance matrices defined above before Equation (3.11). Then,

$$y_{exp}|y_c \sim \mathscr{N}(\mu_{exp|c}((X,\theta),D),\Sigma_{exp|c}((X,\theta),D))$$

with:

$$\mu_{exp|c}((X,\theta),D) = m_{exp}(X,\theta) + \Sigma_{exp,c}((X,\theta),D)\Sigma_{c,c}(D)^{-1}(y_c - m_c(D)), \tag{3.15}$$

$$\Sigma_{exp|c}((X,\theta),D) = \Sigma_{exp,exp}(X,\theta) + \Sigma_\delta + \sigma^2_{err} I_n - \Sigma_{exp,c}((X,\theta),D)\Sigma_{c,c}(D)^{-1}\Sigma_{exp,c}((X,\theta),D)^T. \tag{3.16}$$

For $\mathscr{M}_2$, the variance term $\Sigma_{exp,exp}(X,\theta) + \Sigma_\delta + \sigma^2_{err} I_n$ in Equation (3.2.2) becomes $\Sigma_{exp,exp}(X,\theta) + \sigma^2_{err} I_n$ because there is no discrepancy. It means that $\Sigma_{exp|c}((X,\theta),D)$ for $\mathscr{M}_2$ can be rewritten as:

$$\Sigma_{exp|c}((X,\theta),D) = \Sigma_{exp,exp}(X,\theta) + \sigma_{err}^2 I_n - \Sigma_{exp,c}((X,\theta),D)\Sigma_{c,c}(D)^{-1}\Sigma_{exp,c}((X,\theta),D)^T. \tag{3.17}$$

The conditional likelihood can then be expressed as:

$$\begin{aligned}\mathscr{L}^C(\theta,\beta_\delta,\Phi_\delta;\beta_S,\Phi_S,y_{exp}|y_c,X,D) \propto & |\Sigma_{exp|c}((X,\theta),D)|^{-1/2} \\ & \exp\Big\{-\frac{1}{2}(y_{exp}-\mu_{exp|c}((X,\theta),D))^T\Sigma_{exp|c}((X,\theta),D)^{-1} \\ & (y_{exp}-\mu_{exp|c}((X,\theta),D))\Big\}.\end{aligned} \tag{3.18}$$

Usually in a Bayesian framework, β is distributed according to a Jeffreys *prior*. In this case, $\pi(\beta) = \pi(\beta_S,\beta_\delta) \propto 1$ and we can integrate out β from the full likelihood expressed by Equation (3.11).

3.2.3 Estimation

Maximum likelihood estimator

In this section, we comment remarkable insights developed in Cox et al. (2001). For estimating the parameters θ, β and Φ, a first approach (for $\mathscr{M}_1$ and $\mathscr{M}_2$) would be to maximize the full likelihood introduced in the previous section. This method is called Full Maximization of Likelihood Estimator. The major drawback of this method is to deal with a high number of parameters and in certain cases this leads to a very heavy computational operation.

A second method, introduced in Cox et al. (2001) only for $\mathscr{M}_2$ to overcome this issue, is called the Separated Maximization of Likelihood Estimation (SMLE). The estimation is made in two steps. The first step is to maximize the partial likelihood (Equation (3.14)) to get estimators of the parameters of the Gaussian Process. Then these estimators ($\hat{\Phi}$ and $\hat{\beta}$) are plugged into $\mu_{exp|c}((X,\theta),D)$ and $\Sigma_{exp|c}((X,\theta),D)$ which are the mean and the variance of the conditional distribution. A likelihood is set up from those quantities and maximized to get $\hat{\theta}$. The SMLE method can also be seen as an approximation of the generalized non linear least squares.

These methods are applied in Cox et al. (2001) for $\mathscr{M}_2$. For models $\mathscr{M}_3$ and $\mathscr{M}_4$, (Wong et al., 2017) have developed a new approach which deals with the identifiability problem when the discrepancy is added in this framework. Then, the estimation part is performed in two times. The first step consists in estimating $\hat{\theta}$ in

$$\hat{\theta} = \underset{\theta\in\mathscr{Q}}{argmin}\, M_n(\theta) \quad with \quad M_n(\theta) = \frac{1}{n}\sum_{i=1}^{n}\{y_{exp_i} - F(x_i,\theta)\}^2. \tag{3.19}$$

Then the estimation of the discrepancy is done by applying a nonparametric regression to the data $\{x_i, y_{exp_i} - F(x_i,\hat{\theta})\}_{i=1,\dots,n}$. Any nonparametric regressions are subject to offer working alternatives with this method which shows an interesting flexibility of the approach.

Bayesian estimation

Under the Bayesian framework, there are several *ad hoc* short cuts to find estimators without evaluating and sampling from the entire joint *posterior* distribution of the unknowns. The idea behind is to consider a *prior* distribution on each parameters which we will separate into two different categories. The first category represents the nuisance parameters which are typically $\{\sigma_S^2,\sigma_\delta^2,\psi_S,\psi_\delta\}$, σ_{err}^2 and β. Those parameters are added because of the modeling. The second category regroups the other parameters to estimate such as θ. We will work on the two generic models $\mathscr{M}_3$ and $\mathscr{M}_4$ with the corresponding sets of parameters to estimate.

The difference between both models lies in the fact that for $\mathscr{M}_3$ the code can be used as such and for $\mathscr{M}_4$ a surrogate is used to avoid running the code. In the further developments, the parameters to estimate will be relative to $\mathscr{M}_4$ and for going back to $\mathscr{M}_3$ it will be just necessary to omit the nuisance parameters relative to the surrogate.

As introduced before, it is common to take a weakly informative *prior* on β such as $\pi(\beta_S, \beta_\delta) \propto 1$. It is also reasonable to suppose that *prior* information about θ is independent from the *prior* information about Φ and β. The *prior* density can then be expressed as

$$\pi(\theta, \beta, \Phi) = \pi(\theta) \times 1 \times \pi(\Phi). \tag{3.20}$$

Once the full likelihood integrated $\mathscr{L}^F$ on the *prior* distribution of β, the *posterior* distribution can be expressed (all details are pursued in Kennedy and O'Hagan (2001)).

For a full Bayesian analysis, integrating Φ out is needed to finally get $\pi(\theta|y)$. However this integration can be quite difficult because of the high number of nuisance parameters. It would also demand a full and careful consideration of the *prior* $\pi(\Phi)$. Two methods are mainly used for estimating θ and Φ. In Higdon et al. (2004), the choice made is to jointly estimate all parameters from Equation (3.11). The strength of this method is to stand within the pure Bayesian tracks: recourse is made to all collected data (the simulated with the DOE and experimental data) to estimate all parameters and nuisance parameters at the same time.

However, Kennedy and O'Hagan (2001) and Bayarri et al. (2007) have chosen an estimation in separate steps. This method called modularization by Liu et al. (2009) makes inference simpler but gives only a convenient approximation of the exact *posterior* (that separates the components of parameter Φ for each Gaussian Process involved). The first step consists in maximizing the likelihood $\mathscr{L}^M(\beta_S, \Phi_S|y_c; D)$ (Equation (3.14)) to get the maximum likelihood estimates (MLE) $\hat{\beta}_S$ and $\hat{\Phi}_S$ of β_S and Φ_S. In the second stage, these estimators are plugged into the conditional likelihood $\mathscr{L}^C(\theta, \beta_\delta, \Phi_\delta; \beta_S, \Phi_S, y_{exp}|y_c, X, D)$ (Equation 3.18) from which the posterior density is sampled with MCMC methods. Note that this last step is the only one that differs from SMLE method from Cox et al. (2001).

Kennedy and O'Hagan (2001) arrange to first estimate the nuisance parameters of the surrogate of the code with the partial likelihood (Equation (3.8)), then they make an integration on the *prior* density of $\boldsymbol{\theta}$ to estimate the nuisance parameters of the discrepancy and the variance of the measurement error. They finally estimate the parameter vector θ by sampling in the *posterior* density with a MCMC. Bayarri et al. (2007) are doing the same thing in estimating first the nuisance parameters of the surrogate with the partial likelihood. However, they use "virtual" residuals (defined as $y_{exp} - f_c(X, \theta_{prior})$ where θ_{prior} is a *prior* value on θ) to compute a maximum likelihood estimates for estimating the nuisance parameters relative to the discrepancy and to the measurement error. Then the *posterior* densities of θ, σ_δ^2 and σ_{err}^2 are sampled with a Gibbs algorithm based on conditional complete distribution. Practically, this estimation is very time consuming. Indeed, the Gibbs sampler will compute at each iteration the full likelihood which contains a $(n+N) \times (n+N)$ matrix to invert.

We made the choice to use the modularization method for estimating first the nuisance parameters of the surrogate with the partial likelihood. Then, to avoid the computational time burden of the method introduced by Bayarri et al. (2007) and the integration on the *prior* density of $\boldsymbol{\theta}$ of Kennedy and O'Hagan (2001), we chose to sample, with the Algorithm 7, in the *posterior* densities of the parameter vector θ, of the nuisance parameters of the discrepancy and of the variance of the measurement error. The first part of the algorithm is a Metropolis Hastings within Gibbs that is run for a limited number of times. This algorithm allows to estimate the covariance structure on the samples generated and to use it in a Metropolis algorithm. These way to proceed help the Metropolis algorithm

to better perform.

3.3 Application to the prediction of power from a photovoltaic (PV) plant

In this section, the PV plant code is a toy example to try out all the models. First, we test the model $\mathscr{M}_1$ (Equation (3.1)), in which only the initial code and the measurement error are considered. The code is supposed, in this case, quick to run although, in most industrial case studies, numerical codes are time consuming. This is the first issue of feasibility met by engineers. In a second part, we apply Model $\mathscr{M}_2$ on our example to mimic the case when the code cannot be run at will. This model introduces a surrogate of the code and its characteristics will be detailed below. $\mathscr{M}_3$ is motivated by the gap between the reality and the code observed, most of the time, by engineers. In this case, we will add to $\mathscr{M}_1$ an error term for the discrepancy between the code and the phenomenon. This code error will be represented by a Gaussian process also detailed below. The final case is when both issues are occurring. That will lead to the consideration of $\mathscr{M}_4$ for the application case.

The Bayesian framework starts with the elicitation of *priors* densities (that will not be discussed here (Albert et al., 2012)). According to the experts we choose:

- $\eta \sim \mathscr{N}(0.143, 2.5.10^{-3})$,
- $\mu_t \sim \mathscr{N}(-0.4, 10^{-2})$,
- $a_r \sim \mathscr{N}(0.17, 3.6.10^{-3})$,
- $\sigma^2_{err} \sim \Gamma(2, 169)$,
- $\sigma^2_{\delta} \sim \Gamma(3, 1)$,
- $\psi_{\delta} \sim \mathscr{U}(0, 1)$.

This application section is developed in two subsections. The first subsection details the practical implementation procedure of the inference for each model. In the second subsection, we discuss all the results obtained for the models that we tried out.

3.3.1 Inference

The sensitivity analysis run Section 3.1.1 on the parameter vector θ allowed to conclude that only η, μ_t and a_r are relevant considering the power output. The inference only concerns these three parameters and the additional nuisance parameters depending on the model. For the sake of simplicity, data, recorded every $10s$, is averaged per hour and only data corresponding to a strictly positive power are kept. The Bayesian framework is chosen for the following study. It is motivated by the availability of strongly informative *priors*, elicited from experts, on the parameters we want to estimate. To perform the inference, a Markov Chain Monte Carlo algorithm is used (Robert, 1996) (especially the algorithm 7 in Section 2.5). Here, the Metropolis within Gibbs is launched for 3000 iterations. The values of this first sampling phase are kept to improve the covariance structure of the auxiliary distribution used to make proposals by the algorithm. This will lead to better mixing properties for the following Metropolis Hastings (10000 iterations including a burn in phase of 3000).

Two months of data are studied. The PV production over August and September 2014 are available. We used those two months of data averaged per hour which makes 1019 points. For the cross validation, three days of instantaneous power (51 points) are taken off the learning set and used to evaluate the predictive power of the model considering the rest of the available data.

The Gaussian process

As said in Section 1.4.3, 6 input variables are needed to run the code. These are t the UTC time, L the latitude, l the longitude, I_g the global irradiation, I_d the diffuse irradiation and T_e the ambient temperature. The test stand is precisely located. Therefore, the latitude and the longitude are not to be considered since they do not change by records.

The major issue in emulating the behavior of the code is to deal with correlated variables. Actually, the global irradiation, diffuse irradiation and ambient temperature depend on the time which defines the sun position. If a space filling DOE, is taken into $[0,1]^4$ and then unnormalized between the upper and lower bounds of the 4 input variables and the parameter, some configurations tested would not make any sense. We could obtain for example, a time which indicates the morning and a global irradiation value which corresponds at noon. To solve this problem, we choose to run a PCA (Principle Component Analysis) on the matrix containing all the x_i's (over the duration used for the calibration). The aim is to access an uncorrelated space in which we could sample a DOE which would keep a physical sense and then go back to the original space with the transformation matrix.

The main steps of this method are:

1. the PCA is performed on the matrix X where the i^{th} line contains $x_i = \begin{pmatrix} t_i \\ I_{g_i} \\ I_{d_i} \\ T_{e_i} \end{pmatrix}$ where $x_i \in \mathbb{R}^4$,
2. the maximin LHS is sampled in the uncorrelated space given by the PCA,
3. the transformation matrix T allows us to go back in the original space,
4. the code is run for those points and gives the computed data y_c.

The Gaussian processes emulated from this method reveal to work much better. We also could have developed the method with an adaptive numerical design (see (Damblin et al., 2018)) to the correlated input variables.

To position the application case to a time consuming context, a limited number of experiments is allowed for the DOE which establishes the surrogate. We will limit the number of code calls to 50 to investigate the time consuming situation and compare it to other situations more favorables. This number of experiments is taken when computer codes are extremely time consuming. To compare the quality of $\mathscr{M}_2$ with a DOE of 50 points, we chose to compare it with a model $\mathscr{M}_2$ established with a DOE of 100 points.

Table 3.1: Results in cross validation for $\mathscr{M}_2$ for 2 different DOE sizes

	50 points	100 points
coverage rate at 90% (in %)	32	65
RMSE of the instantaneous power (W)	21.61	19.7

The degradation in prediction with the decrease of the number of points in the DOE (Table 3.1) is in line with our desire to place ourselves in the most unfavorable case possible.

The first model $\mathscr{M}_1$

Model $\mathscr{M}_1$ described by Equation (3.1) only deals with the measurement error. The code used in its simplest form only makes recourse to the parameters η, μ_t and a_r. In this case the parameters to infer on are η, μ_t, a_r and σ^2_{err} (where $\varepsilon_i \overset{iid}{\sim} \mathcal{N}(0, \sigma^2_{err})$).

The second model $\mathscr{M}_2$

As defined in Section 3.2, when the code is time consuming, the solution is to mimic it with a Gaussian process (GP). For the GP emulator, we made the choice to consider the mean function $m_S(\bullet,\bullet)$ as a linear combination of linear functions. That means $\mathbf{H}_S$ is a matrix of linear functions. The correlation function r_S ($c_S = \sigma^2_S r_S$) chosen is defined by the following equation that corresponds to a Gaussian kernel (Equation (2.50)):

$$r_S\{(x,\theta),(x^*,\theta^*)\} = \exp\Big\{-\frac{1}{2}\frac{||(x,\theta)-(x^*,\theta^*)||_2^2}{\psi_S^2}\Big\} \tag{3.21}$$

where $||\bullet||_2$ stands for the Euclidean norm.
In this case, five parameters have to be estimated: η, μ_t, a_r, σ^2_{err}, σ^2_S and ψ_S.

The third model $\mathscr{M}_3$

The third model introduces another GP for the discrepancy. We choose a different covariance kernel which is a Matérn 5/2 (Equation (3.22) from Equation (2.51) with $\nu = 5/2$). Note that compared to Equation (3.4), the discrepancy mean has been set to 0 (*i.e.* $m_\delta(.) = 0$). These choices are motivated by the fact that the purpose of calibration is to estimate the "best-fitting" vector parameter θ. We do not want any compensation into any additional bias. This decision is consistent with Bachoc et al. (2014) where the same hypothesis has been made.

$$r_\delta(x,x^*) = \Big(1+\frac{\sqrt{5}||x-x^*||_2}{\psi_\delta}+\frac{5||x-x^*||_2^2}{3\psi_\delta^2}\Big)\exp\Big\{-\frac{\sqrt{5}||x-x^*||_2}{\psi_\delta}\Big\}. \tag{3.22}$$

In this case, there are also five parameters to estimate that are η, μ_t, a_r σ^2_δ, ψ_δ and σ^2_{err}.

The fourth model $\mathscr{M}_4$

This part focuses on a time consuming code with discrepancy. This model uses the same surrogate and discrepancy defined above. The two correlation function for the surrogate and the discrepancy were chosen with different regularity in order to distinguish the two Gaussian processes. It seems relevant to assume that the discrepancy is smoother than the code. That is why a Matérn correlation function is chosen for the code and a Gaussian correlation function for the discrepancy. In this case seven parameters need to be estimated which are η, μ_t, a_r, σ^2_{err}, σ^2_S, ψ_S, σ^2_δ and ψ_δ.

Estimation of the nuisance parameters

In our Bayesian framework, the choice of an estimation by modularization is made. It concerns only the second and the fourth model. As it is the case in Kennedy and O'Hagan (2001), a maximization of the probability $\pi(\Phi_S|y_c)$ is done to estimate β_S, σ^2_S and ψ_S where y_c are the outputs of the code for all the points given by the DOE. This maximization is included in the R function *km* from the package *DiceKriging* (Roustant et al., 2012).

3.3.2 Results

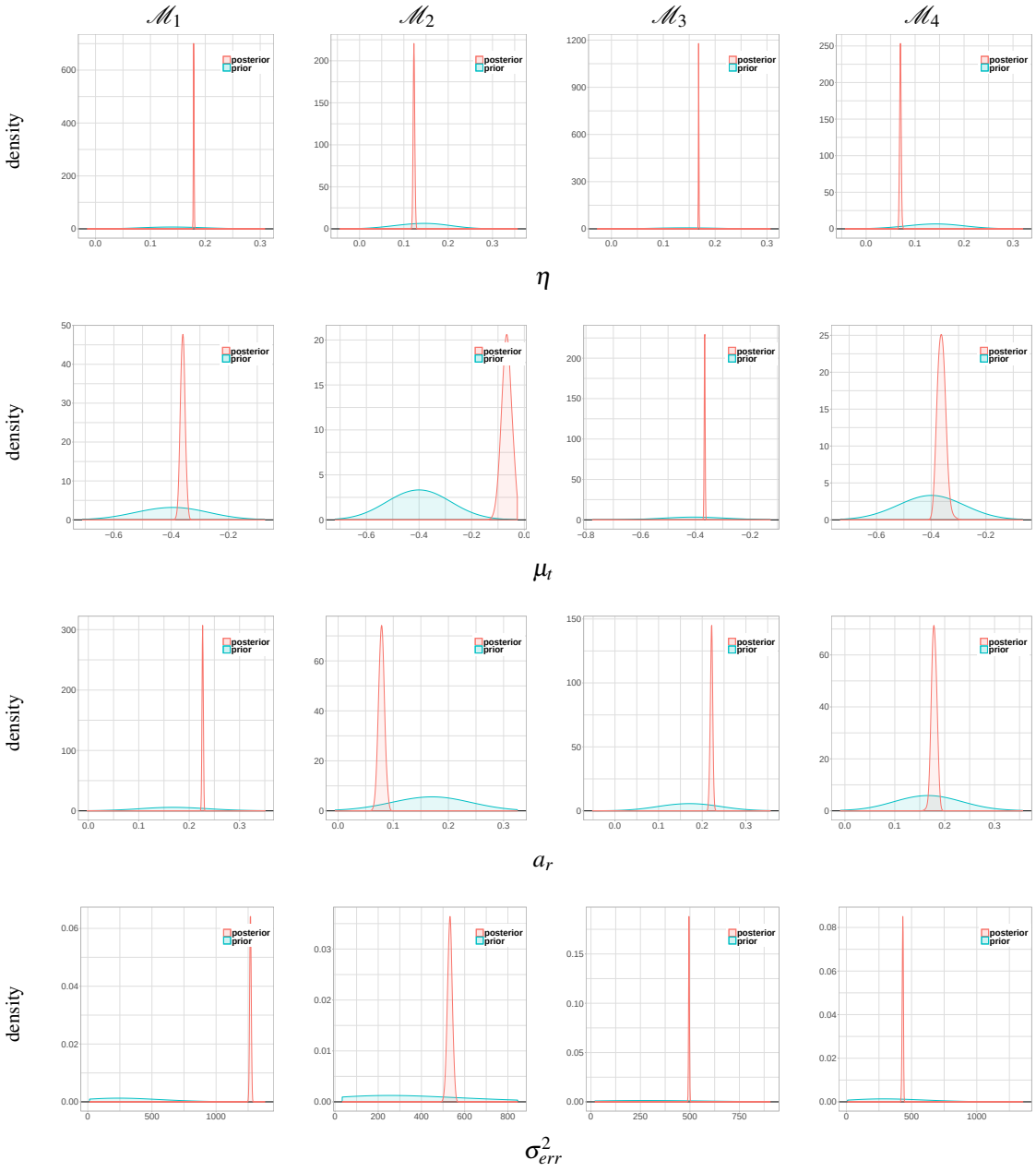

Figure 3.7: *Prior* (in blue) and posterior (in red) densities of η, μ_t, a_r and σ^2_{err} for each model. On the two first column the two first models (without and with surrogate) which have only these four parameters to estimate. The two other columns represent the third and the fourth models which have two more parameters to estimate (see Figure 3.9).

Figure 3.7 compares the results obtained (with the help of the **R** package CaliCo (Carmassi, 2018)) for η, μ_t, a_r and σ^2_{err} for each model. In each case, the MCMC chains converge. For the first model, a strong disagreement has appeared between the *prior* and *posterior* densities for σ^2_{err}. The Maximum A Posteriori (MAP) of σ^2_{err}'s density, for $\mathscr{M}_1$, is 1283 W^2. That makes a standard deviation of $36W$ which is too high and has no physical trustworthiness. For $\mathscr{M}_2$, $\mathscr{M}_3$ and $\mathscr{M}_4$ the MAP estimations of σ^2_{err}'s densities seem to be coherent with physics. The addition of the discrepancy between $\mathscr{M}_1$ and $\mathscr{M}_3$ then between $\mathscr{M}_2$ and $\mathscr{M}_4$ depicts a correlation (an error structure) in y_e. When, no code error is applied, the variance from this covariance matrix is added to the variance of the measurement error.

From $\mathscr{M}_1$ to $\mathscr{M}_2$, a surrogate emulates the numerical code. Figure 3.7 illustrates a bigger variance *a posteriori* for the parameters densities of $\mathscr{M}_2$ than $\mathscr{M}_1$. Replacing the code by a surrogate had brought more uncertainty. Moreover, the densities for $\mathscr{M}_2$ appear to be out of step with $\mathscr{M}_1$. Calibration behaves as if a bias has appeared with the surrogate. This is worth to note that using a surrogate not only increase the variance of the posterior distribution of the parameter θ but may also change the mode. This is also observed when moving from $\mathscr{M}_3$ to $\mathscr{M}_4$. Adding the discrepancy (from $\mathscr{M}_1$ to $\mathscr{M}_3$ and from $\mathscr{M}_2$ to $\mathscr{M}_4$) has almost always reduced the variances of the posterior distributions.

We also depict correlation between the parameters. As a matter of fact, a strong positive and linear correlation links every parameters (η, μ_t and a_r) with each other as illustrated in Figure 3.8. A strong correlation appears between μ_t and a_r. A lower, but still meaningful, correlation is also visible between η and μ_r, and a_r and η.

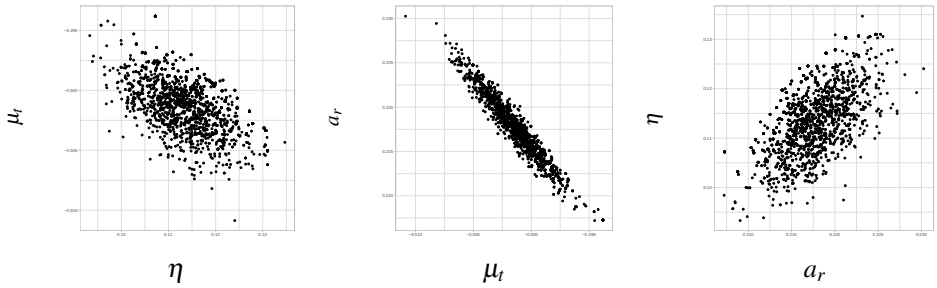

Figure 3.8: Correlation representation between the parameters.

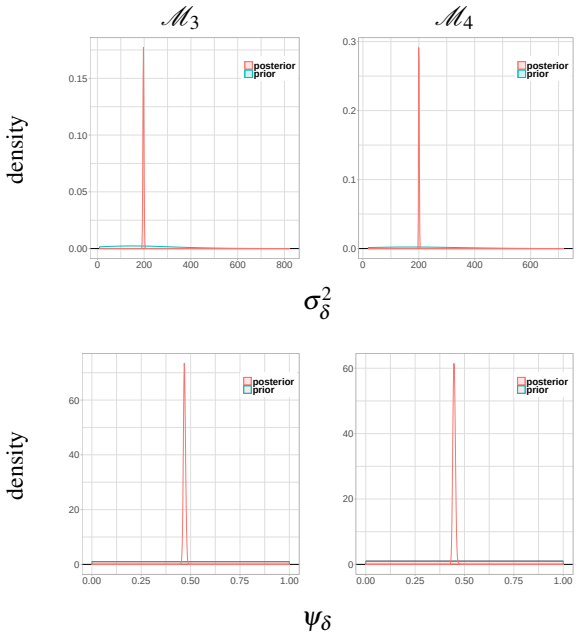

Figure 3.9: *Prior* (in blue) and posterior (in red) densities of σ^2_δ and ψ_δ for $\mathscr{M}_3$ and $\mathscr{M}_4$.

Figure 3.9 illustrates the estimation of the parameters from the discrepancy term. As expected, learning from data has improved our *prior* belief by decreasing the *prior* uncertainty of the parameters. It shows that in both cases (with and without surrogate) that the convergence seems to be reached at some point.

3.3.3 Comparison

To compare the prediction ability of the four models, a cross validation (CV) is performed. Three days of data (chosen randomly) are taken off the calibration dataset for each of the 100 repetitions of the CV. The power densities, generated from the MCMC samples, allow us to compute, for each model, the 90% predictive credibility intervals. The coverage rate at 90% represents the quantity of validation experiments contained in these credibility intervals. The Root Mean Square Error (RMSE) is also computed for the instantaneous power. The results are displayed Table 3.2.

Table 3.2: Comparison of the RMSEs and coverage rates in prediction of 100 test-sets on three randomly selected days.

	$\mathscr{M}_1$	$\mathscr{M}_2$	$\mathscr{M}_3$	$\mathscr{M}_4$
coverage rate at 90% (in %)	91	32	87	23
RMSE of the instantaneous power (W)	12.69	21.61	5.91	18.7

The coverage rates for $\mathscr{M}_1$ and $\mathscr{M}_3$ corresponds to the chosen credibility level. However for $\mathscr{M}_2$ and $\mathscr{M}_4$ the coverage rates are below this level. This was expected since the coverage rates for the code emulation displayed in Table 3.1 were below the fixed credibility level especially when the DOE had only 50 points. We recall that these coverage rates only account for the surrogate error and not for the uncertainty on θ. We also notice that the predictive power increases when the discrepancy is added.

Overall, the model $\mathscr{M}_3$ has better results than the others. This conclusion can be explained twofold. First, the code achieves a better prediction than the surrogate. Second, a correlation structure remains in the error. Adding the discrepancy in the model allows to catch up the real results. The fact that the models, encompassing a surrogate, produces worse results is expected. The Gaussian process used for the surrogate had trouble to fill every variation of power. To compensate this lack of information, the posterior credibility interval becomes wide and less informative.

Figure 3.7 illustrates that the use of a Gaussian process with a low number of points creates a bad estimation of the parameters. Indeed the modes *a posteriori* found with $\mathscr{M}_2$ or $\mathscr{M}_4$ are shifted compared to the *a priori* modes. A particular attention has to be given to the Gaussian process quality regarding further calibration. A better DOE could have been proposed as for example the sequential design introduced in Damblin et al. (2018). If we start from an initial maximin LHS DOE with a higher number of points, say 150 points, and if we use the sequential design to add say 30 points, calibration is modified and the results are illustrated in Figure 3.10.

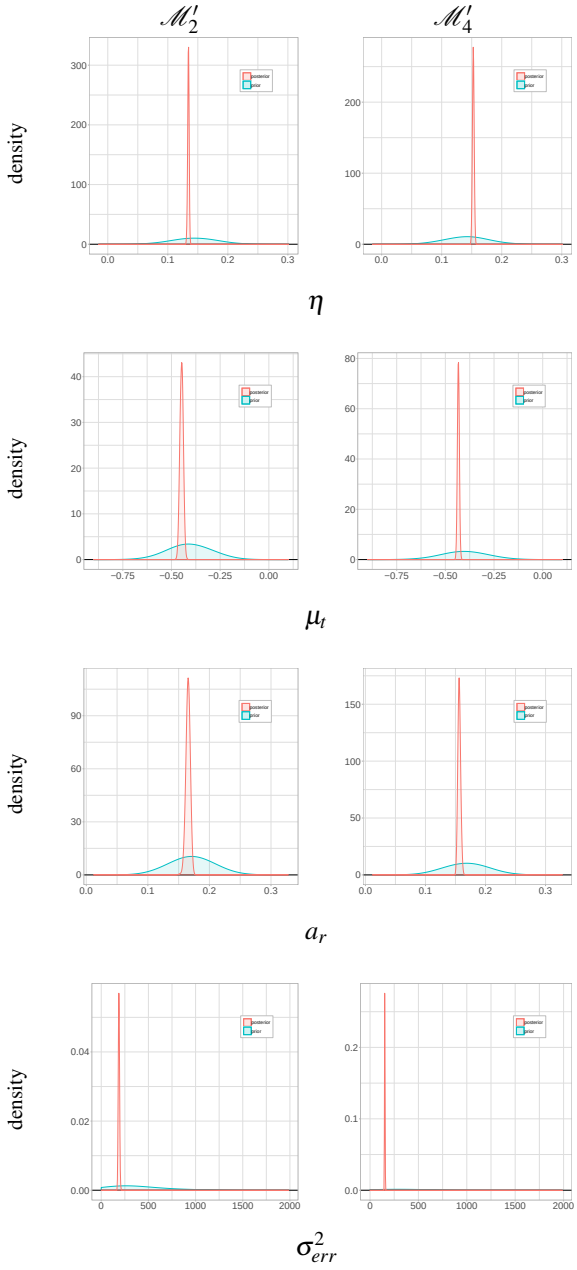

Figure 3.10: Calibration results for $\mathscr{M}_2$ and $\mathscr{M}_4$ that are using Gaussian processes build on a DOE extended by the sequential design.

With the new DOE, the estimation of the nuisance parameters is better than the previous one and allow to be consistent with the *prior* densities and the physical meaning. The question highlighted here is to wonder how many points to add in the design to have a proper Gaussian process which leads to a coherent calibration.

3.4 Conclusion and discussion

This chapter focuses on code calibration which can be a very interesting way to deal with uncertainties in numerical experiments. The code used in this chapter, to allow comparisons with time consuming codes, is a quick code predicting power from small PV plant. This work can be extended to bigger computational codes in application at larger PV plants. As we are working with a physical code, it is important in this case to keep in mind the reality of the physical boundaries. This aspect had allowed us to confirm the presence of the discrepancy.

In a case where input variables are correlated, additional issues of DOE appear when the surrogate is fitted. To cope with these issues we made recourse to a PCA. The design of numerical experiments could have been enhanced by using adaptive designs proposed by Damblin et al. (2018).

The hypotheses made for the application case can also be discussed. For example setting ρ and $m_\delta(.)$ to 1 and 0 is a preliminary decision which goes along with calibration. We do not want to quantify the bias because the aim of calibration is to find the parameter value to compensate that bias. If one's goal is to check where the uncertainty goes, other hypotheses could have been made. For example, a non zero discrepancy expectation would quantify the mean of the gap between the code and the experiments. In calibration we want this gap to be taken into account in the code through adjusted θ.

The quality of the Gaussian process is also mentioned. Indeed, in code calibration if the estimation of the parameters of the Gaussian process is based on a not satisfactory DOE, calibration quality is then affected. Some techniques as the sequential design, based of the expected improvement criterion, allow to enhance the initial DOE by finding new points regarding further calibration.

One can wonder which models to use in a given particular case. There is no obvious answer to this question but it depends first on the numerical code. If it is time consuming, the first model to try on is $\mathscr{M}_2$ and, if it is not, one can use $\mathscr{M}_1$. One may then wonder whether it is worth adding a discrepancy term, going from $\mathscr{M}_1$ to $\mathscr{M}_3$ or from $\mathscr{M}_2$ to $\mathscr{M}_4$. In-depth work on statistical validation had been developed in Damblin et al. (2016) in which the comparison between two models (with and without discrepancy) is studied in a simplified context. The Bayes factor helps to decide whether the discrepancy is relevant or not. However, this Bayes factor is burdensome to compute in a general context.

CHAPTER 4

CALICO: A R PACKAGE FOR BAYESIAN CALIBRATION

This chapter presents and illustrates the package **CaliCo** that has been published on CRAN (Comprehensive R Archive Network). It is based on all theoretical developments given Chapter 3. Three packages have been developed for Bayesian calibration. The package **BACCO** (Hankin, 2013b) is a bundle of several other packages. It imports **emulator** (Hankin, 2014), **mvtnorm** (Genz et al., 2018), **calibrator** (Hankin, 2013c) and **approximator** (Hankin, 2013a). These packages contain functions that perform Bayesian calibration and also prediction. The statistical model implemented concerns only the case where the numerical code is time consuming and when a code error is added (the model introduced by Kennedy and O'Hagan (2001)). Moreover, **BACCO** explores the prior distribution, of the parameters from the statistical model, using analytic or numerical integration. Similarly, another package called **SAVE** (Palomo et al., 2017) deals with Bayesian calibration through four main functions: `SAVE`, `bayesfit`, `predictcode` and `validate`. The function `SAVE` creates the statistical model when `bayesfit`, `predictcode` and `validate` perform respectively Bayesian calibration, prediction and validation of a model. Calibration is done in **SAVE** in a similar way to **BACCO** because it is based on the same statistical model. Both packages are not flexible with the numerical code and the statistical model. A design of experiments has to be run upstream on the code before running calibration. The package **RobustCalibration** based on Gu and Wang (2017) (Gu, 2018b) is a package that achieves calibration of inexact mathematical models and implements the discrepancy with a "scaled Gaussian process".

CaliCo offers more flexibility on the statistical model choice. If one is interested in calibrating a numerical code inexpensive in computation time, **CaliCo** allows the user to upload the numerical code in the model and run Bayesian calibration with it. A very intuitive perspective is given to the user by using four functions called `model`, `prior`, `calibrate` and `forecast`, that are detailed in Section 4.1. **CaliCo** also allows the user to access several **ggplot2** (Wickham and Chang, 2016) graphs very easily and to load each of them to change the layout at one's convenience. At some point, if the code is time consuming, calibration needs a surrogate to emulate it. Usually, a Gaussian process is chosen (Sacks et al., 1989; Cox et al., 2001). Three packages are related to the establishment of a Gaussian process as a surrogate: **GPfit** (MacDoanld et al., 2015), **DiceKriging** (Roustant et al.,

2015) and **RobustGaSP** based on Gu (2018a) (Mengyang Gu and Berger, 2018). We use in **CaliCo**, the package **DiceDesign** (Franco et al., 2015) to establish design of experiments (DOE). For compatibility matters, we have chosen **DiceKriging** to generate surrogates. Moreover, the time consuming steps of the Bayesian calibration are coded in **C++** and linked to **R** via the package **Rcpp** (Eddelbuettel et al., 2018).

In this section, the first part (Section 4.1) presents the main functions and functionalities of the package. The second section provides a Bayesian calibration illustration on a toy example. It is based on a physical model of a damped harmonic oscillator with five parameters to calibrate.

4.1 Guidelines for users

CaliCo performs a Bayesian calibration through 3 different steps:

1. creation of the statistical model,
2. selection of the *prior* distributions,
3. running calibration with some simulation options.

CaliCo allows the user to easily take advantage of the calibration performed. Indeed, a prediction can be performed on a new data set using the calibrated code in the statistical model. The main functions of the package **CaliCo** are detailed Table 4.1.

Function	Description
`model`	generates a statistical model
`prior`	creates one or a list of prior distributions
`calibrate`	performs calibration for the `model` and `prior` specified
`forecast`	predicts the output over a new data set

Table 4.1: Main functions necessary for calibration in **CaliCo**.

CaliCo is coded in **R6** (Chang, 2017) which is an oriented object language. Each function generates an **R6** object that can be used by other functions (in this case methods) that are proper to the object. The **R6** layer is not visible to the user. The main classes implemented with the associated functions are detailed Table 4.2.

Function	**R6** class called
`model`	`model.class`
`prior`	`prior.class`
`calibrate`	`calibrate.class`
`forecast`	`forecast.class`

Table 4.2: **R6** classes called by the main functions in **CaliCo**.

To define the statistical model, which is the first step in calibration, several elements are necessary. The code function must be defined in **R** and takes two arguments X and θ respecting this order. For example:

```
code <- function(X,theta)
{
   return((6*X-2)^2*sin(theta*X-4))
}
```

If the numerical code is called from another language, one can implement a wrapper that calls from **R** the numerical code according to the above writing. It is also possible to build a design of experiments and run the code outside **R** to get the outputs. Then, the DOE and the relative outputs are used instead of the numerical code in the statistical model (more details below). The function `model` takes several other arguments (Table 4.3) as for example the vector or the matrix of the input variables (described in Section 1.4.1), the vector of experimental data or the statistical model chosen for calibration.

`model` description	Arguments to be specified
$f_c(x,\theta)$ the function to calibrate	`code` (defined as `code(x,theta)`)
X the matrix of the input variables	`X`
y_e the vector of experimental data	`Yexp`
$\mathscr{M}$ the statistical model selected	(Default value `model1`) `model1`, `model2`, `model3`, `model4`
Gaussian process options (optional only for $\mathscr{M}_2$ and $\mathscr{M}_4$)	(Optional) `opt.gp` (is a `list`)
Emulation options (optional only for $\mathscr{M}_2$ and $\mathscr{M}_4$)	(Optional) `opt.emul` (is a `list`)
Simulation options (optional only for $\mathscr{M}_2$ and $\mathscr{M}_4$)	(Optional) `opt.sim` (is a `list`)
Discrepancy options (necessary only for $\mathscr{M}_3$ and $\mathscr{M}_4$)	(Optional) `opt.disc` (is a `list`)

Table 4.3: description of the arguments of the function `model`.

If the chosen model is $\mathscr{M}_2$ or $\mathscr{M}_4$, then a Gaussian process will be created as a surrogate of the function `code`. In each case the Gaussian process option (`opt.gp`, see Table 4.3) is needed. It is a `list` which encompasses:

- `type`: type of covariance function chosen for the surrogate established by the package **DiceKriging** (Roustant et al., 2015),
- `DOE`: design of experiments for the surrogate (default value NULL).

Three cases can occur. First, the numerical code is available and the user does not possess any Design Of Experiments (DOE). In this case, only the Gaussian process option `opt.gp` and the emulation option `opt.emul` (Table 4.3) are needed. The emulation option controls the establishment of the DOE. It is a `list` which contains:

- `p`: the number of parameters in the model,
- `n.emul`: the number of points for constituting the DOE,
- `binf`: the lower bound of the parameter vector,
- `bsup`: the upper bound of the parameter vector.

The second possible case is when the user want to enforce a specific DOE. Note that in `opt.gp`, the `DOE` option is `NULL`. One can upload a specific DOE in this option. As the new DOE will be used, the emulation option `opt.emul` is not needed anymore.

The third case is when no numerical code is available. The user is only in possession of a DOE and the corresponding code evaluations. Then, the simulation option `opt.sim` is added. This option encompasses:

- `Ysim`: the code evaluations of `DOEsim`,
- `DOEsim`: the specific DOE used to get simulated data.

When this option is added, the emulation option is not necessary anymore. The `code` argument in the function `model` can then be set to `code=NULL`. Table 4.4 presents a summary of these three cases and the options to add in the function `model`.

cases	options needed in the function `model`
numerical code without DOE	`opt.gp` and `opt.emul`
numerical code with DOE	`opt.gp`
no numerical code	`opt.gp` and `opt.sim`

Table 4.4: Summary of the options needed depending on the case.

If $\mathscr{M}_3$ or $\mathscr{M}_4$ is chosen, a discrepancy term is added in the statistical model. This discrepancy is created in **CaliCo** with the option `opt.disc` in the function `model`. It is a `list` composed of one component called `type.kernel` which corresponds to the correlation function of the discrepancy chosen. The list of the correlation functions implemented are detailed in Table 4.5.

`kernel.type`	covariance implemented
`gauss`	$g(d) = \sigma^2 exp\left(-\frac{1}{2}(\frac{d}{\psi})^2\right)$
`exp`	$g(d) = \sigma^2 exp\left(-\frac{1}{2}\frac{d}{\psi}\right)$
`matern3_2`	$g(d) = \sigma^2\left(1+\sqrt{3}\frac{d^2}{\psi}\right)exp\left(-\sqrt{3}\frac{d^2}{\psi}\right)$
`matern5_2`	$g(d) = \sigma^2\left(1+\sqrt{5}\frac{d^2}{\psi}+5\frac{d^2}{3\psi^2}\right)exp\left(-\sqrt{5}\frac{d^2}{\psi}\right)$

Table 4.5: Kernel implemented for the discrepancy covariance.

The `model` function creates an **R6** object in which two methods have been coded and are able to be used as regular functions. These function are `plot` and `print`. The function `print` gives an access to a short summary of the statistical model created. The function `plot` allows to get a visual representation. However, to get a visual representation, parameter values have to be specified in the model. A pipe `%<%` is available in **CaliCo** to parametrize a model. Let us consider a created random model called `myModel`. The code line `myModel %<% param` is the way to give the model parameter values. The `param` variable is a `list` containing values of θ (named `theta` in the `list`), θ_δ for $\mathscr{M}_3$ and $\mathscr{M}_4$ (variance and correlation length of the discrepancy, named `thetaD` in the `list`) and σ_e^2 (named `var`). Section 4.2 gives an overview of how the pipe works for each models. The `plot` function takes two arguments that are the model generated by `model` and the x-axis to draw the results. An additional option `CI` (by default `CI="all"`) allows to select which credibility interval at 95% one wants to display:

- `CI="err"` only the credibility interval of the measurement error with (or without) the discrepancy is given,
- `CI="GP"`, only the credibility interval of the Gaussian process is plotted,
- `CI="all"` all credibility intervals are displayed.

The second step for calibration is to define the *prior* distributions of the parameters we seek to estimate. At least, two *prior* distributions have to be set and it is in the case where the `code` function takes only one input parameter θ. That means, only the posterior distributions of this parameter and the variance σ_e^2 (for the model $\mathscr{M}_1$) are what we seek to sample in calibration.

Table 4.6 describes the options needed into the function `prior`. Three choices of `type.prior` are available so far (`gaussian`, `gamma` and `unif` which respectively stands for Gaussian, Gamma and Uniform distributions, see Table 4.7 for details). For calibration with 2 parameters (which is the lower dimensional case), `type.prior` is a vector (`type.prior=c("gaussian","gamma")` for example). Then, `opt.prior` is a list containing characteristics of each distribution. For the Gaussian distribution, it will be a vector of the mean and the variance, for the

prior arguments	description
type.prior	vector or scalar of string among ("gaussian", "gamma" and "unif")
opt.prior	list of vector corresponding to the distribution parameters

Table 4.6: description of the arguments of the function prior.

Gamma distribution it will be the shape and the scale and for the Uniform distribution the lower bound and the upper bound (opt.prior=list(c(1,0.1),c(0.01,1)) for example).

type.prior	distribution	arguments in opt.prior vector
gaussian	$f(x) = \frac{1}{\sqrt{2\pi V}} exp\left(-\frac{1}{2}\left(\frac{(x-m)^2}{V}\right)\right)$	c(m,V)
gamma	$f(x) = \frac{1}{(k^a * \Gamma(a))} x^{(a-1)} exp(-\frac{x}{k})$	c(a,k)
unif	$f(x) = \frac{1}{b-a}$	c(a,b)

Table 4.7: description of the arguments of the function prior.

When the prior distributions and the model are defined, calibration can be run. The function calibrate implements a Markov Chain Monte Carlo (MCMC) according to specific conditions all controlled by the user.

calibrate arguments	Description
md	the model generated with the function model
pr	the list of prior generated by the function prior
opt.estim	estimation options for calibration
opt.valid	(optional) cross validation options (default value NULL)

Table 4.8: description of the arguments of the function calibrate.

The MCMC implemented in **CaliCo** is composed of two algorithms as described in Section 2.5. The implemented algorithm (Algorithm 7) is coded in **C++** thanks to **Rcpp** package (Eddelbuettel et al., 2018) in order to limit the time consuming aspect of these non-parallelizable loops. Note that there is an adaptability present to regulate the parameter k according to the acceptation rate. The user is free to set that regulation at the wanted percentage.

Then, the opt.estim option is a list composed of:

- Ngibbs: the number of iterations of the Metropolis within Gibbs algorithm,
- Nmh: the number of iteration of the Metropolis Hastings algorithm,
- thetaInit: the starting point,
- r: the vector of regulation of the covariance in the Metropolis within Gibbs algorithm (in the proposition distribution the variance is $k\Sigma$),
- sig: the variance of the proposition distribution Σ,
- Nchains: (default value 1) the number of MCMC chains to run,
- burnIn: the number of iteration to withdraw from the Metropolis Hastings algorithm.

In the function calibrate, one optional argument is available to run a cross validation. This option called opt.valid, is a list composed of two options which have to be filled:

- type.CV: the type of cross validation wanted (leave one out is the only cross validation implemented so far type.CV="loo"),

- `nCV`: the number of iteration to run in the cross validation.

After calibration is complete, an **R6** object is created and two methods (`print` and `plot`) are available and are also able to be used as regular functions. The `print` function is a summary that recalls the selected model, the code used for calibration, the acceptation rate of the Metropolis within Gibbs algorithm, the acceptation rate of the Metropolis Hastings algorithm, the maximum *a posteriori* and the mean *a posteriori*. It allows to quickly check the acceptation ratios and see if the chains have properly mixed. The `plot` function generates automatically, a series of graphs that displays, notably, the output of the calibrated code. Two arguments are necessary to run this function: the calibrated model and the x-axis to draw the results. An additional option `graph` (by default `graph="all"`) allows to control which graphic layout one wants to plot:

- if `graph="chains"` a layout containing the autocorrelation graphs, the MCMC chains and the *prior* and *posterior* distributions for each parameter is given,
- if `graph="corr"` a layout containing in the diagonal the *prior* and *posterior* distributions for the parameter vector θ and the scatterplot between each pair of parameters is plotted,
- if `graph="results"` the result of calibration is displayed,
- if `graph="all` all of them are printed.

Note that all these graphs (made in **ggplot2**) are proposed in a particular layout but one can easily load all of them into a variable and extract the particular graph one wants. Indeed, if a variable `p` is used to store all the graphs, then `p` is a `list` containing "ACF" (the autocorrelation graphs), "MCMC" (the MCMC chains), "corr" (the scatter plot between each pair of parameters), "dens" (*prior* and *posterior* distributions) and "out" (the calibration result graph) variables.

Two external functions can be run on an object generated by the function `calibrate`:

- `chain`: function that allows to extract the chains sampled in the *posterior* distribution. If the variable `Nchains`, in `opt.estim` option, is higher than 1 then the function `chain` return a **coda** (Plummer et al., 2016) object with the sampled chains,
- `estimators`: function that accesses the maximum *a posteriori*(MAP) and the mean *a posteriori*.

Sequential design introduced in Damblin et al. (2018) allows to improve the Gaussian process estimation for $\mathscr{M}_2$ and $\mathscr{M}_4$. Based on the expected improvement (EI), introduced in Jones et al. (1998), new points are added to an initial DOE in order to improve the quality of calibration. The arguments of the function are given Table 4.9.

`sequentialDesign` arguments	Description
`md`	the model generated with the function `model` (for $\mathscr{M}_2$ or $\mathscr{M}_4$)
`pr`	the list of prior generated by the function `prior`
`opt.estim`	estimation options for calibration
`k`	number of points to add in the design

Table 4.9: description of the arguments of the function `sequentialDesign`.

The last main function in **CaliCo** is `forecast` which produces a prediction of a selected model on a new data and based on previous calibration.

The object generated by `forecast.class` possesses the two similar methods `print` and `plot`. The `print` function gives a summary identical to the one in `model.class` except that it adds the MAP estimator. The `plot`

`forecast` arguments	Description
`modelfit`	calibrated model (run by `calibrate` function)
`x.new`	new data for prediction

Table 4.10: description of the arguments of the function `calibrate`.

function displays the calibration results and also adds the predicted results. The arguments of `plot` are the predicted model and the x-axis which is the axis corresponding to calibration extended with the axis corresponding to the forecast.

4.2 Multidimensional example with CaliCo

An illustration is provided in this section to help the user to easily handle the functionalities of **CaliCo**. This example, represents a damped harmonic oscillator and experimental data are simulated for specific values of the parameter vector θ. These parameters to calibrate are A the constant amplitude, the damping ratio ξ, the spring constant k, the mass of the spring m and ϕ the phase. The recorded displacement of the damped oscillator is represented in Figure 4.1 and the equation of the displacement of a damped harmonic oscillator is:

$$x(t,\theta) : \mathbb{R} \times \mathbb{R}^5 \mapsto \mathbb{R} \tag{4.1}$$

$$(t, \theta = (A, \xi, k, m, \phi)^T) \rightarrow Ae^{-\xi\sqrt{\frac{k}{m}}t} sin(\sqrt{1-\xi^2}\sqrt{\frac{k}{m}}t + \phi) \tag{4.2}$$

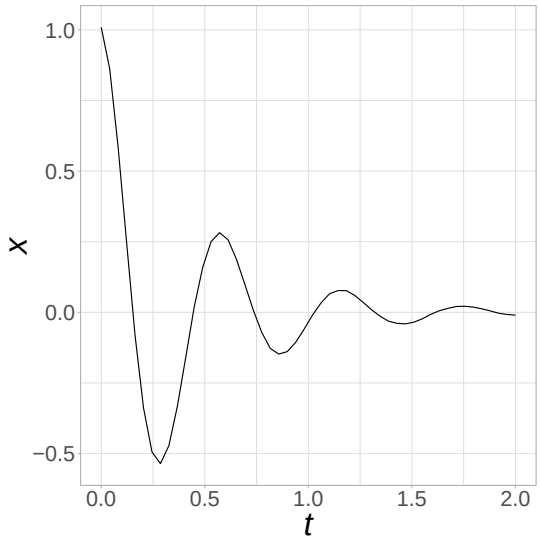

Figure 4.1: Displacement of the oscillator simulated.

There is five parameters to calibrate. Let us consider that experiments are available for the 2 first seconds of the movement (these experiments have been simulated for the interval time $[0,2]$ with the time step of $40ms$ and for specific parameter values). Visually, at time $t = 0$ the position of the mass seems to be at the position $x = 1$. So the values *a priori* of A and ϕ are $A = 1$ and $\phi = \frac{\pi}{2}$. The company states that the spring has a constant of $k = 6N/m$ and the mass is weighing at $m = 50g$. The major uncertainty lies in the knowledge of ξ. It is indeed a difficult parameter to estimate. However, the value of the damping ratio ξ determines the behavior of the system. A damped harmonic oscillator can be:

- over-damped ($\xi > 1$): the system exponentially decays to steady state without oscillating,
- critically damped ($\xi = 1$): the system returns to steady state as quickly as possible without oscillating,
- under-damped ($\xi < 1$): The system oscillates with the amplitude gradually decreasing to zero.

Physical experts provide us a value of $\xi = 0.3$ but says that the parameter can oscillate between the value $[0.15, 0.45]$ at 95%.

4.2.1 The models

To define the first statistical model, the function `code` has to be defined such as:

```
n <- 50
t <- seq(0,2,length.out=n)
code <- function(t,theta)
{
  w0 <- sqrt(theta[3]/theta[4])
  return(theta[1]*exp(-theta[2]*w0*t)*sin(sqrt(1-theta[2]^2)*w0*t+theta[5]))
}
```

In **CaliCo**, one function allows to define the statistical model. This function `model` takes as inputs the `code` function, the input variables `X`, experimental data and the model choice. If a numerical code has no input variables, it is just enough to put `X=0`.

```
model1 <- model(code,X=t,Yexp,"model1")
```

In this particular case where, the input variables are unidimensional, it is easy to choose a graphical representation of the model. As mentioned in Section 4.1, when the function `model` is called, a `model.class` object is generated. This object owns several methods as `plot` or `print` that behave as regular functions.

```
print(model1)

## Call:
## [1] "model1"
##
## With the function:
## function(t,theta)
## {
##   w0 <- sqrt(theta[3]/theta[4])
##   return(theta[1]*exp(-theta[2]*w0*t)*sin(sqrt(1-theta[2]^2)*w0*t+theta[5]))
## }
##
## No surrogate is selected
##
## No discrepancy is added
```

To get a visual representation, parameter values need to be added to the model. To achieve such an operation in **CaliCo**, one can use the defined pipe %<%. Following the pipe, a list containing all parameter values allows to select these values for the visual representation. The parameter vector (`theta`) and the value of the variance of the

measurement error (`var`), here, are needed in the list to set a proper parametrization of the model:

```
model1 %<% list(theta=c(1,0.3,6,50e-3,pi/2),var=1e-4)

## Warning:  Please be careful to the size of the parameter vector
```

The `Warning` is present at each use of the pipe. It appears as a reminder for the user to be careful with the size of the parameter vector. When the model is defined nothing indicates the number of parameter within. To get a visual representation of the model with such parameters values, the `plot` function can be straightforwardly applied on the model object created by `model` and completed by the pipe `%<%`. The x-axis needs to be filled in `plot` to get an x-axis for display. The left panel of Figure 4.3 is the result of:

```
plot(model1,t)
```

If no parameter value is added to the model and the visual representation is required, a `Warning` appears and remind the user that no parameter value has been defined and only experiments are plotted (Figure 4.2):

```
model1bis <- model(code,X=t,Yexp,"model1")
plot(model1bis,t)

## Warning:  no theta and var has been given to the model, experiments only are
                                        plotted
```

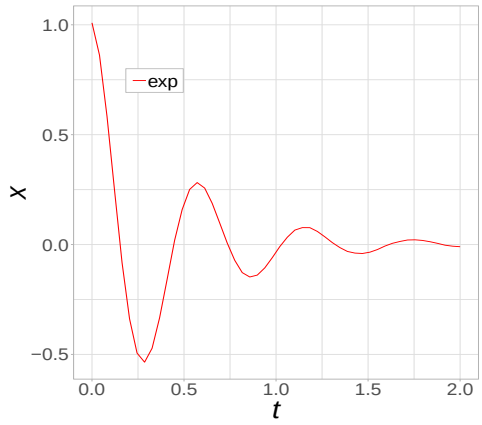

Figure 4.2: Experimental data displayed when no parameter values are set in the model.

If no x-axis is defined, then no visual representation is possible and the function `plot` breaks:

```
model1bis <- model(code,X=t,Yexp,"model1")
plot(model1bis)

## Error:  No x-axis selected, no graph is displayed
```

For $\mathscr{M}_2$ several cases may occur (see Table 4.4 for more details). First the user only has the time consuming code without any Design Of Experiments (DOE). Then, the definition of the model is done by delimiting the boundaries of the parameters. The option `opt.gp` allows the user to set the kernel type of the Gaussian process and to specify if the user has a particular DOE. In this first case the DOE is not available, so `DOE=NULL` in the `opt.gp` option. To parametrize the DOE created in the function `model`, the option called `opt.emul` needs to be filled by `p`, `n.emul`,

`binf`, `bsup`. Where `p` stands for the number of parameter to calibrate, `n.emul` for the number of experiments in the DOE, `binf` and `bsup` for the lower and upper bounds of the parameter vector.

```
binf <- c(0.9,0.15,5.8,48e-3,1.49)
bsup <- c(1.1,0.45,6.2,52e-3,1.6)

model2 <- model(code,t,Yexp,"model2",
                opt.gp = list(type="matern5_2",DOE=NULL),
                opt.emul = list(p=5,n.emul=60,binf=binf,bsup=bsup))
```

The second case is when the users has a numerical code and a specific DOE. In **CaliCo**, the option `DOE` in `opt.gp` allows to consider a particular DOE wanted by the user. As no DOE is build with the function `model`, the option `opt.emul` is not necessary anymore:

```
library(DiceDesign)
DOE <- maximinSA_LHS(lhsDesign(60,6)$design)$design
DOE <- unscale(DOE,c(0,binf),c(2,bsup))

model2doe <- model(code,t,Yexp,"model2",
                   opt.gp=list(type="matern5_2",DOE=DOE))
```

When one does not possess any numerical code, but only the DOE and the corresponding output, another option, called `opt.sim`, needs to be filled. The `opt.gp` option is still needed to specify the chosen kernel but the `opt.emul` option is no longer necessary (for the same reasons as in the second case). The `opt.sim` option is the list containing the DOE and the output of the code. As the user does not possess the numerical code, the `code` option in the function `model` can be set to `code=NULL`.

```
Ysim <- code(DOE[1,1],DOE[1,2:6])
for (i in 2:60){Ysim <- c(Ysim,code(DOE[i,1],DOE[i,2:6]))}

model2code <- model(code=NULL,t,Yexp,"model2",
                    opt.gp = list(type="matern5_2", DOE=NULL),
                    opt.sim = list(Ysim=Ysim,DOEsim=DOE))
```

The package **CaliCo** is comfortable with these three situations and bring flexibility according to the different problems of the users. Similarly as before, parameter values need to be added to each models and the function `print` and `plot` can be directly used:

```
ParamList <- list(theta=c(1,0.3,6,50e-3,pi/2),var=1e-4)
model2 %<% ParamList
model2doe %<% ParamList
model2code %<% ParamList
```

```
plot(model2,t)
plot(model2doe,t)
plot(model2code,t)
```

These three lines of code produce the same graphs because **CaliCo** uses a maximin Latin Hypercube Sample (LHS) to establish the DOE. Several credibility interval are displayed. For the first model only the 95% credibility

interval of the measurement error is available. For the second model, the 95% credibility interval of the Gaussian process can also be shown. Figure 4.3 illustrates $\mathscr{M}_1$ and $\mathscr{M}_2$.

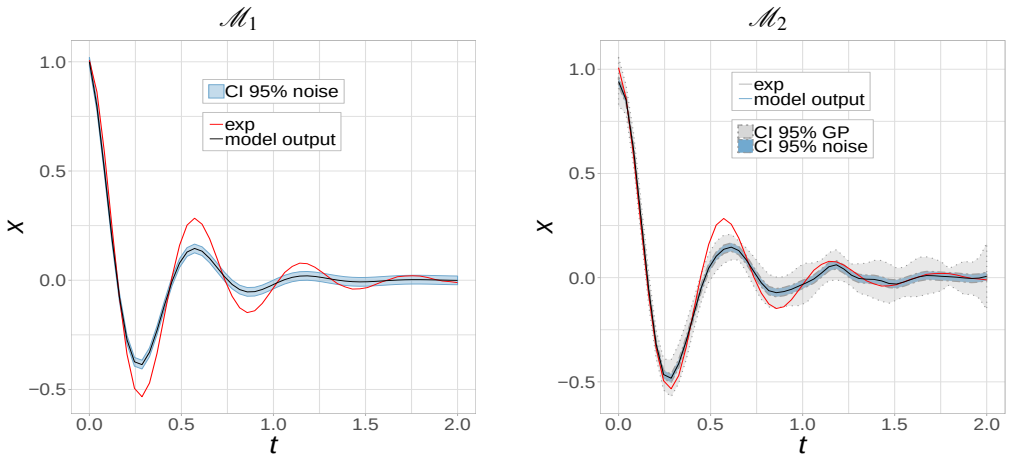

Figure 4.3: First and second model output for prior belief on parameter values. The left panel illustrates the first model and the right panel the second model with the Gaussian process estimated.

One can be interested in modeling a code discrepancy. When the code is not time consuming, the model to choose is $\mathscr{M}_3$. The `opt.disc` option allows to specify the kernel type of the discrepancy. Note that the visual representation requires initial values for the discrepancy and are needed in the pipe `%<%`. This vector `thetaD` is composed of σ^2 and ψ according to Table 4.5.

```
model3 <- model(code,t,Yexp,"model3",
                opt.disc = list(kernel.type="gauss"))
model3 %<% list(theta=c(1,0.3,6,50e-3,pi/2),thetaD=c(1e-4,0.2),var=1e-4)
```

When the code is time consuming, then $\mathscr{M}_4$ is selected. The same cases can occur as for $\mathscr{M}_2$ but only the case where the code is not available will be considered, here, for $\mathscr{M}_4$.

```
model4 <- model(code=NULL,t,Yexp,"model4",
                opt.gp = list(type="matern5_2", DOE=NULL),
                opt.sim = list(Ysim=Ysim,DOEsim=DOE),
                opt.disc = list(kernel.type="gauss"))
model4 %<% list(theta=c(1,0.3,6,50e-3,pi/2),thetaD=c(1e-4,0.2),var=1e-4)
```

To get a visual representation of $\mathscr{M}_3$ and $\mathscr{M}_4$, the function `plot` is defined identically as before. The results are displayed in Figure 4.4.

```
plot(model3,t)
plot(model4,t)
```

Note that several credibility intervals can be displayed. By default all of them are shown (for example see right panels of Figure 4.3 and Figure 4.4). For $\mathscr{M}_1$ and $\mathscr{M}_3$ only one credibility interval is given. It represents the 95% credibility interval of the measurement error, in the case of $\mathscr{M}_1$, and the 95% credibility interval of the measurement error plus the discrepancy, in the case of $\mathscr{M}_3$. For $\mathscr{M}_2$ and $\mathscr{M}_4$, two credibility intervals are available. Compared to $\mathscr{M}_1$ and $\mathscr{M}_3$ the credibility interval at 95% of the Gaussian process, that emulates the code, is added. It allows to

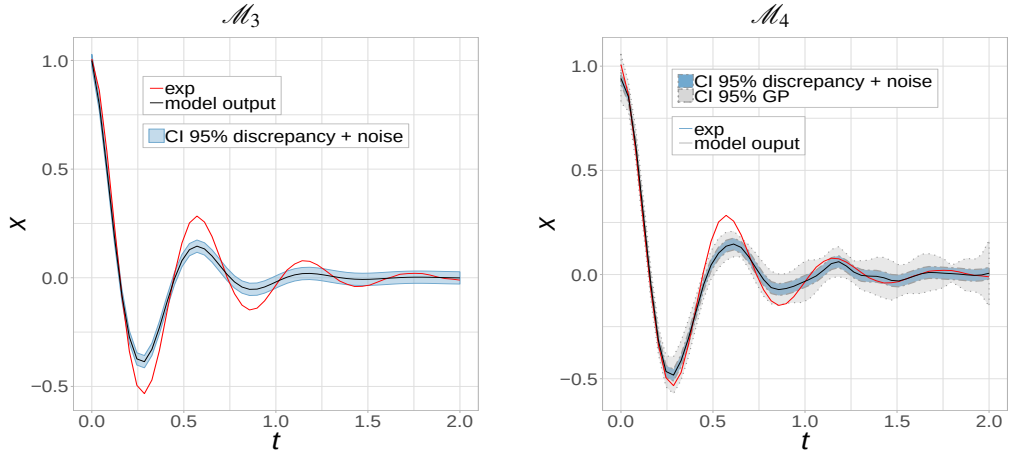

Figure 4.4: Third and fourth model output for prior belief on parameter values. The left panel illustrates the third model and the right one, the fourth model with the Gaussian process estimated. Both are encompassing the discrepancy.

quickly visualize from where the variability of the model comes before calibration.

With the option `CI`, one can deactivate or select which credibility interval (CI) one wants to display. By default `CI="all"`, but if `CI="err"` only the 95% CI of the measurement error with, or without, the discrepancy is given. Similarly, for $\mathscr{M}_2$ and $\mathscr{M}_4$, if `CI="GP"`, only the 95% CI of the Gaussian process is shown. For example, for $\mathscr{M}_4$ the three possibilities are obtained with the following code and are displayed in Figure 4.5.

```
plot(model4,t,CI="err")
plot(model4,t,CI="GP")
plot(model4,t,CI="all")
```

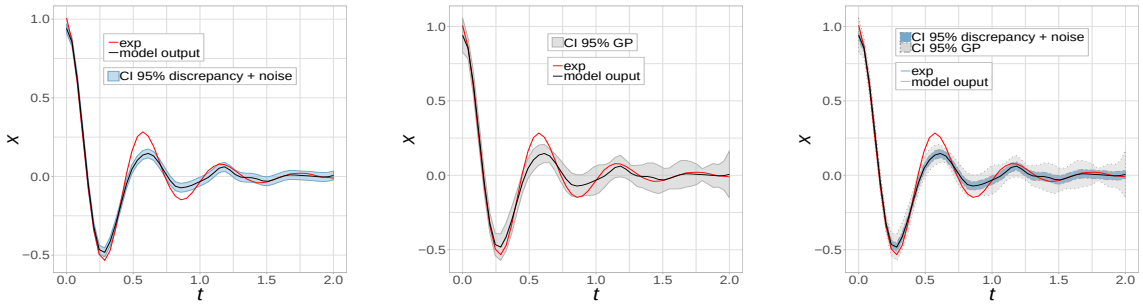

Figure 4.5: $\mathscr{M}_4$ displayed for some guessed values with the CI relative to the measurement error on the left panel, with the CI relative to the Gaussian process only on the middle panel and both credibility intervals on the right panel.

4.2.2 Priors

To a proper Bayesian calibration, prior distributions have to be defined on every parameters we want to estimate (parameters of interest as θ or nuisance parameter such as θ_δ or σ^2_{err}). It means that the number of parameters to estimate differs according to the model. In **CaliCo**, the possible distributions are detailed in Table 4.7.

To define a proper prior distribution in **CaliCo**, a `prior.class` object with the function `prior` is generated. One or several prior distributions can be produced with this function. Two arguments have to be completed: `type.prior` and `opt.prior`. The argument `type.prior` can be a string (if only one prior distribution is looked for) or a vector of strings (if several prior distributions are wanted). Similarly, the argument `opt.prior` can be a vector of the distribution parameters or a list of vectors.

In this example, 5 parameters have to be calibrated. For $\mathscr{M}_3$ and $\mathscr{M}_4$, the discrepancy is added and the variance σ_δ^2 with the correlation length ψ have to be estimated as much as the other parameters. It means that for these models, two more prior distributions have to be added compared at $\mathscr{M}_1$ and $\mathscr{M}_2$. The order to define them are, first the parameters θ, then θ_δ and σ_{err}^2. In the following code lines, `pr1` stands for the prior distributions for $\mathscr{M}_1$ and $\mathscr{M}_2$ where `pr2` for $\mathscr{M}_3$ and $\mathscr{M}_4$. In the first prior definition, only the 5 parameters and σ_{err}^2 prior distributions are defined. In the second definition, the θ_δ prior distributions are added between θ and σ_{err}^2 ones.

```
type.prior <- c(rep("gaussian",5),"gamma")
opt.prior <- list(c(1,1e-3),c(0.3,1e-3),c(6,1e-3),c(50e-3,1e-5),
                  c(pi/2,1e-2),c(1,1e-3))
pr1 <- prior(type.prior,opt.prior)
```

```
type.prior <- c(rep("gaussian",5),"gamma","unif","gamma")
opt.prior <- list(c(1,1e-3),c(0.3,1e-3),c(6,1e-3),c(50e-3,1e-5),
                  c(pi/2,1e-2),c(1,1e-3),c(0,1),c(1,1e-3))
pr2 <- prior(type.prior,opt.prior)
```

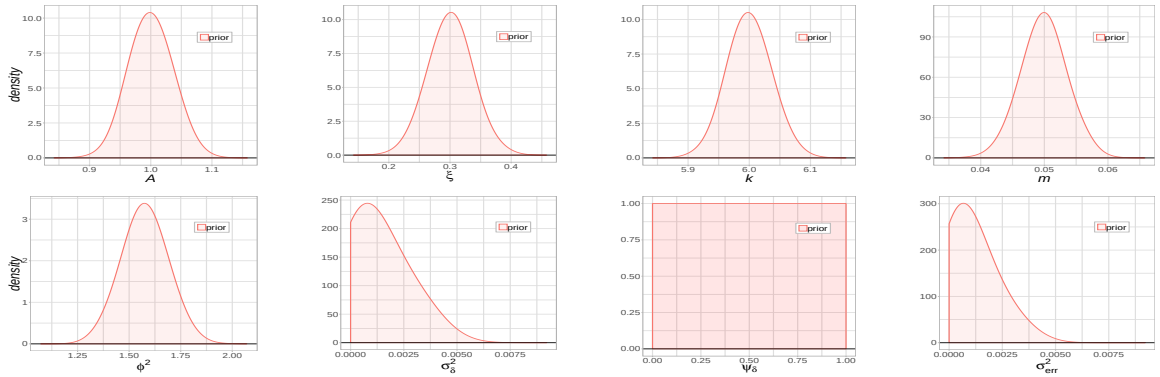

Figure 4.6: Prior distributions for each parameter to calibrate in the application case.

Figure 4.6 illustrates the prior distributions considered for the parameters of the damped harmonic oscillator code. For $\mathscr{M}_1$ and $\mathscr{M}_2$, only A, ξ, k, m, ϕ^2 and σ_{err}^2 distributions are useful. Calibration with $\mathscr{M}_3$ or $\mathscr{M}_4$ the two last distributions (σ_δ^2 and ψ_δ) are then used.

4.2.3 Calibration

Calibration is run thanks to the function `calibrate` in **CaliCo**. Estimation option (`opt.estim`) has to be filled to run the algorithm properly. As it is described in Section 4.1, two MCMC algorithms are run by the function `calibrate`.

```
opt.estim = list(Ngibbs=1000,Nmh=5000,thetaInit=c(1,0.25,6,50e-3,pi/2,1e-3),
                 r=c(0.05,0.05),sig=diag(6),Nchains=1,burnIn=2000)
mdfit1 <- calibrate(model1,pr1,opt.estim)
```

In the terminal, a loading bar represents the execution time of the inference algorithm. Then, the method `print` can be used to quickly access some information (see Section 4.1).

```
print(mdfit1)

## Call:
##
## With the function:
## function(t,theta)
## {
##   w0 <- sqrt(theta[3]/theta[4])
##   return(theta[1]*exp(-theta[2]*w0*t)*sin(sqrt(1-theta[2]^2)*w0*t+theta[5]))
## }
## <bytecode: 0x561a96e4f068>
##
## Selected model : model1
##
## Acceptation rate of the Metropolis within Gibbs algorithm:
## [1] "46.6%" "27%"   "27.9%" "8.9%"  "5.1%"  "4.4%"
##
## Acceptation rate of the Metropolis Hastings algorithm:
## [1] "44.52%"
##
## Maximum a posteriori:
## [1] 1.0138252830 0.2032794864 5.9924680899 0.0505938887 1.5199179525
## [6] 0.0002565248
##
## Mean a posteriori:
## [1] 1.0107256606 0.2010363776 5.9755232004 0.0497000370 1.5042868046
## [6] 0.0002493378
```

To visualize the results, the `plot` method allows to generate **ggplot2** objects and if the option `graph` is not deactivated, the function `plot` will create a layout of graphs displayed in Figure 4.7 which contains:

1. the auto-correlation graphs,
2. the chains trajectories,
3. the *prior* and *posterior* distributions,
4. the correlation between parameters,
5. the results on the quantity of interest.

To generate all the graphs of Figure 4.7, the `plot` function is used similarly as for a `model.class` object with the following code line.

```
plot(mdfit1,t)
```

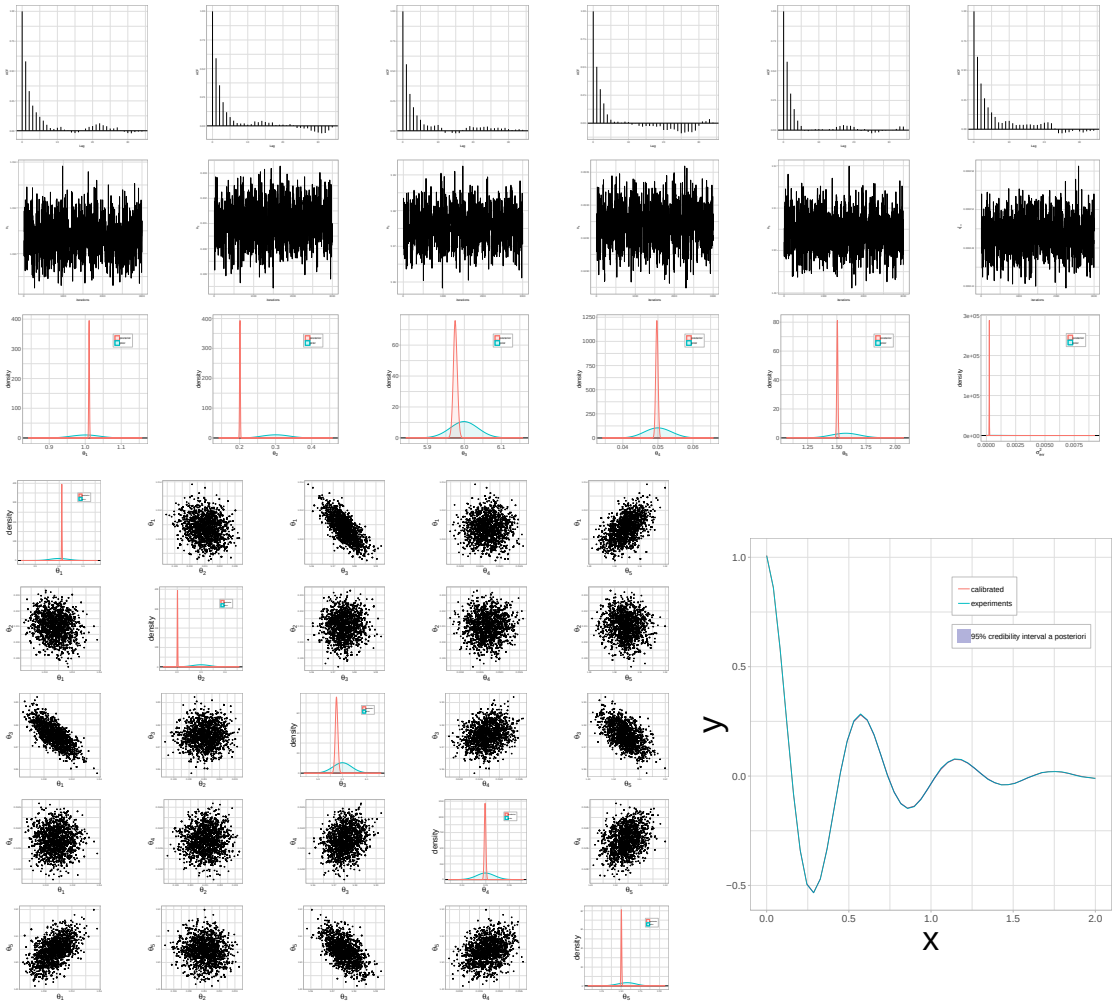

Figure 4.7: Series of plot generated by the function `plot` for calibration on $\mathscr{M}_1$.

Same procedures are available for $\mathscr{M}_2$.

```
mdfit2 <- calibrate(model2,pr1,opt.estim)
```

For $\mathscr{M}_3$ and $\mathscr{M}_4$, the estimation options are slightly different because the number of parameter to estimate has increased. The *prior* object also has changed.

```
opt.estim2=list(Ngibbs=1000,Nmh=5000,thetaInit=c(1,0.3,6,50e-3,pi/2,1e-3,0.5,1e-3),
                r=c(0.05,0.05),sig=diag(8),Nchains=1,burnIn=2000)
```

```
mdfit3 <- calibrate(model3,pr2,opt.estim2)
```

```
mdfit4 <- calibrate(model4,pr2,opt.estim2)
```

```
print(mdfit4)

## Call:
##
## With the function:
## NULL
##
## Selected model : model4
##
## Acceptation rate of the Metropolis within Gibbs algorithm:
## [1] "97.8%" "94.7%" "96.8%" "90.3%" "87.5%" "94.3%" "97.5%" "94.1%"
##
## Acceptation rate of the Metropolis Hastings algorithm:
## [1] "59.2%"
##
## Maximum a posteriori:
## [1] 1.0145661817 0.3052534056 6.0274228120 0.0521952278 1.6229728079
## [6] 0.0009970529 0.4769160005 0.0007652975
##
## Mean a posteriori:
## [1] 0.9968290203 0.2835127409 6.0032790767 0.0506295380 1.5845041957
## [6] 0.0009225639 0.4235883680 0.0006717480
```

Figure 4.7 illustrates the several graphs layout one can obtain with the use of the function `plot`. To select which specific graph one wants to display, the option `graph` can be added to the function `plot`:

- `graph="chains"`: only the table of the autocorrelation, chains points and distributions *a priori* and *a posteriori* is produced . It represents only the top part of Figure 4.7,
- `graph="corr"`: only the table of the correlation graph between each parameter is displayed. It represents only the bottom left part of Figure 4.7,
- `graph="result"`: only the result on the quantity of interest is given. It represents only the bottom right part of Figure 4.7,
- `graph=NULL`: no graphs are produced automatically.

If one does not want to produce these graphs automatically, one can set the `graph` option to `NULL`. As the `plot` function generates **ggplot2** objects, it is possible to load all the generated graphs apart.

```
p <- plot(mdfit4,t,graph=NULL)
```

The variable *p* is a `list` of all the graphs displayed in Figure 4.7. The elements in *p* are:

- `ACF` a `list` of all autocorrelation graphs in the chains for each variable,
- `MCMC` a `list` of all the MCMC chains for each variable,
- `corrplot` a `list` of all correlation graphs between each parameter,
- `dens` a `list` of all distribution *a priori* and *a posteriori* graphs for each variable,
- `out` the **ggplot2** object of the result on the quantity of interest.

Figure 4.8 illustrates the *prior* and *posterior* distributions resulted from calibration on $\mathscr{M}_4$.

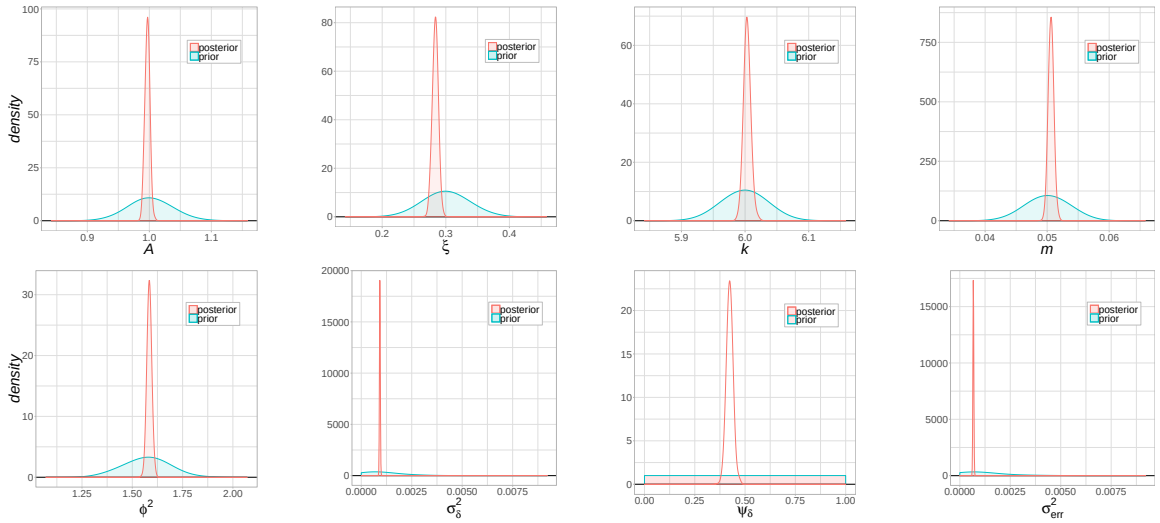

Figure 4.8: *prior* and *posterior* distributions for each parameter for calibration on $\mathscr{M}_4$.

Similarly, if one desires to access the graph of the result on the quantity of interest one only needs to run `p$out`.

```
p$out
```

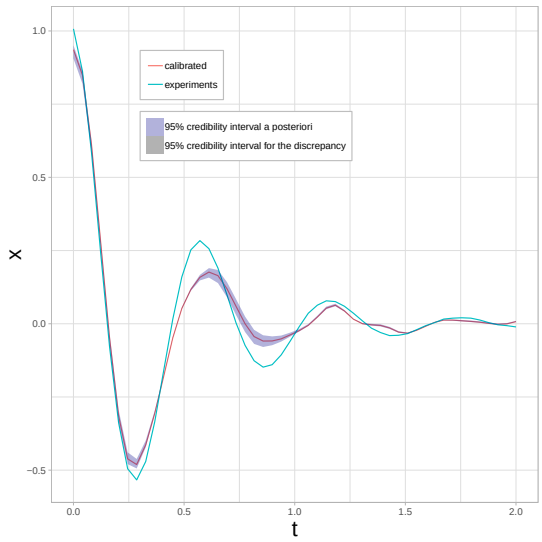

Figure 4.9: Result of calibration on $\mathscr{M}_4$ for the quantity of interest with the credibility interval at 95% *a posteriori*.

4.2.4 Additionnal tools

A function in **CaliCo** called `estimators` allows to access estimators as the MAP and the mean *a posteriori*.

```
estimators(mdfit1)

## $MAP
## [1] 1.0105967765 0.2011318973 5.9741254796 0.0496559638 1.5042583733
## [6] 0.0002488385
##
## $MEAN
## [1] 1.0107256606 0.2010363776 5.9755232004 0.0497000370 1.5042868046
## [6] 0.0002493378
```

If one is interested in running convergence diagnostics on the MCMC chains run by the function `calibrate`, one is free to increase the number of chains in the `opt.estim` options. This operation is accomplished in parallel with an automatically detected number of cores.

```
opt.estim=list(Ngibbs=1000,Nmh=5000,thetaInit=c(1,0.25,6,50e-3,pi/2,1e-3),
               r=c(0.05,0.05),sig=diag(6),Nchains=3,burnIn=2000)

mdfitMCMC <- calibrate(model1,pr1,opt.estim)
```

By setting `Nchains=3`, calibration is run 3 times. The function `chain` allows to load the **coda** object generated and then to use **coda** tools as Gelman-Rubin diagnostics (Gelman and Rubin, 1992) for example.

```
mcmc <- chain(mdfitMCMC)
library(coda)
gelman.diag(mcmc)

## Potential scale reduction factors:
##
##      Point est. Upper C.I.
## [1,]       4.00       7.81
## [2,]       1.55       2.39
## [3,]       8.58      16.65
## [4,]       1.25       1.68
## [5,]       3.52       7.23
## [6,]      38.66      75.48
##
## Multivariate psrf
##
## 31.3
```

The user can also run very easily a cross validation (a leave one out) to estimate how accurately the model prediction will perform in practice. An additional option, called `opt.valid`, is then necessary to run this cross validation. This option is a list containing the number of iteration (`nCV`) and the type cross validation method (`type.valid`).

```
mdfitCV <- calibrate(model1,pr1,
                     opt.estim = list(Ngibbs=1000,
                                      Nmh=5000,
                                      thetaInit=c(1,0.25,6,50e-3,pi/2,1e-3),
                                      r=c(0.05,0.05),
                                      sig=diag(6),
                                      Nchains=1,
                                      burnIn=2000),
                     opt.valid = list(type.valid="loo",
                                      nCV=50))
```

The activation of the cross validation will run the regular calibration and then the `nCV` iterations requested by the user. To decrease the computational burden of such operation, a parallel operation is performed by to the package **parallel** present in `R core`.

```
print(mdfitCV)
```

```
## Call:
##
## With the function:
## function(t,theta)
## {
##   w0 <- sqrt(theta[3]/theta[4])
##   return(theta[1]*exp(-theta[2]*w0*t)*sin(sqrt(1-theta[2]^2)*w0*t+theta[5]))
## }
## <bytecode: 0x561a93b33348>
##
## Selected model : model1
##
## Acceptation rate of the Metropolis within Gibbs algorithm:
## [1] "43.8%" "27.5%" "27.3%" "9.1%"  "4.3%"  "4.4%"
##
## Acceptation rate of the Metropolis Hastings algorithm:
## [1] "48.56%"
##
## Maximum a posteriori:
## [1] 1.0144067620 0.2032408379 6.0077799738 0.0508023686 1.5169972084
## [6] 0.0002536493
##
## Mean a posteriori:
## [1] 1.0121094549 0.2016018588 5.9975259667 0.0499179223 1.5060238122
## [6] 0.0002488799
##
##
## Cross validation:
##  Method: loo
##     Predicted       Real        Error
## 1 0.581834516 0.58394961 0.0021150930
```

```
## 2 0.251110861 0.25245428 0.0013434211
## 3 0.860725971 0.86208615 0.0013601740
## 4 0.860627965 0.86208615 0.0014581796
## 5 0.009987004 0.00981318 0.0001738244
## 6 0.251052820 0.25245428 0.0014014613
##
## RMSE: [1] 0.03737526
##
## Cover rate:
## [1] "94%"
```

The `print` method displays the head of the first iterations of the cross validation and the root mean square error (RMSE) associated. The coverage rate is also printed to check the accuracy of the *posterior* credibility interval.

The implemented function `sequentialDesign` is available only for $\mathscr{M}_2$ and $\mathscr{M}_4$. This function allows to run a sequential design as described in Damblin et al. (2018). Based on the expected improvement (Jones et al., 1998), it improves the estimation of the Gaussian process that emulates the code by adding new points in the design. Calibration quality is, as expected, increased.

```
binf <- c(0.9,0.05,5.8,40e-3,1.49)
bsup <- c(1.1,0.55,6.2,60e-3,1.6)

model2 <- model(code,t,Yexp,"model2",
                opt.gp = list(type="matern5_2",DOE=NULL),
                opt.emul = list(p=5,n.emul=200,binf=binf,bsup=bsup))

type.prior <- c(rep("gaussian",5),"gamma")
opt.prior <- list(c(1,1e-3),c(0.3,1e-3),c(6,1e-3),c(50e-3,1e-5),
                  c(pi/2,1e-2),c(1,1e-3))
pr1 <- prior(type.prior,opt.prior)

newModel2 <- sequentialDesign(model2,pr1,
                    opt.estim = list(Ngibbs=100,
                                     Nmh=600,
                                     thetaInit=c(1,0.25,6,50e-3,pi/2,1e-3),
                                     r=c(0.05,0.05),
                                     sig=diag(6),
                                     Nchains=1,
                                     burnIn=200),
                    k=20)
```

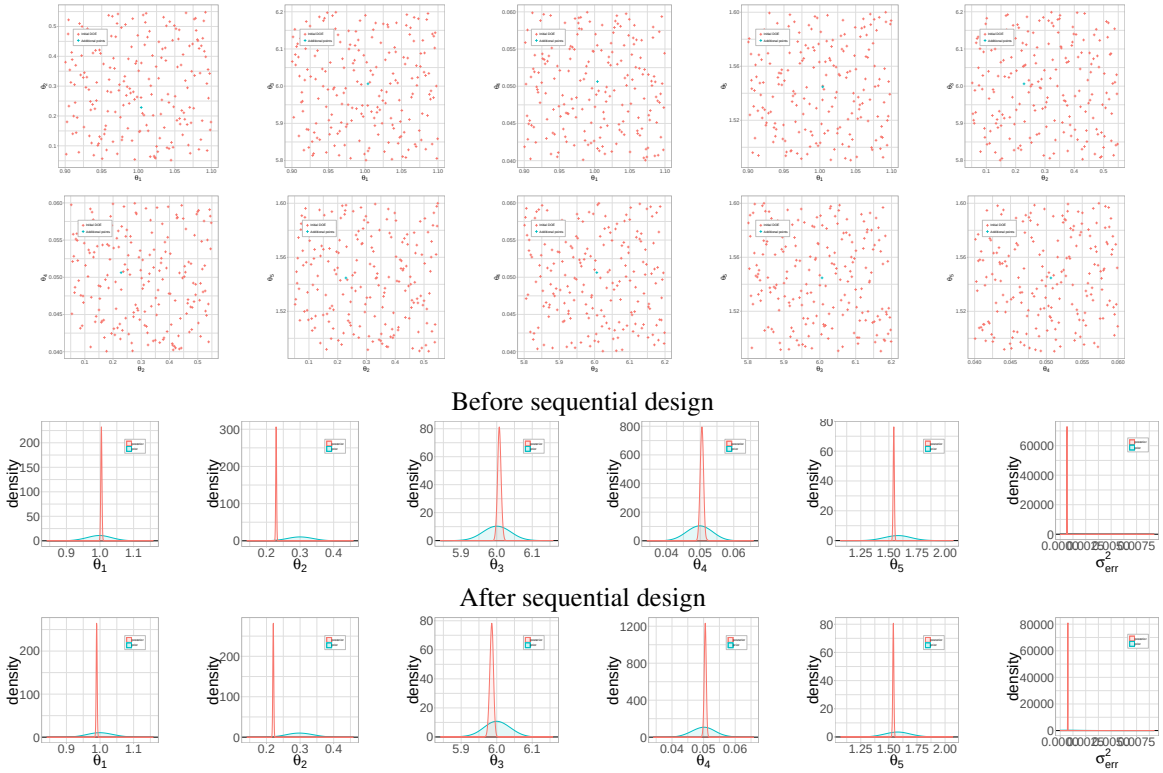

Figure 4.10: Series of plot generated by the function `plot` for the sequential design on $\mathscr{M}_2$.

4.3 Conclusion

In conclusion, **CaliCo** is a package that deals with Bayesian calibration through four main functions. For an industrial numerical code, every specific cases is covered by **CaliCo** (if the user has a DOE or not, with a numerical code or not). The **R6** classes used in the implementation makes the package more robust. Even if the class layer is not visible to the user, the standardized formulation allows a rigorous treatment. The multiple **ggplot2** graphs available for each class allow the user to take advantage of the graphical display without any knowledge of **ggplot2**. The flexibility of **ggplot2** enables also the user to modify the frame, scale, title, labels of the graphs really quickly. All the MCMC calls are implemented in **C++**, which reduces the time of these time-consuming algorithms. The Metropolis within Gibbs algorithm provides a better learning of the covariance matrix that the Metropolis Hastings will use in its proposition distribution. That improves the performance of the algorithms.

Many developments can be brought to the package. For example, statistical validation can be added to the package and permit the user to elect the best model according to the data. Based on Damblin et al. (2016) a validation using the Bayes factor or mixture models can be implemented. The dependences on **DiceKriging** or **DiceDesign** can also be a weakness of the package. When too many dependencies are implemented, the chances to have bad configuration also increase.

Chapter 5

Performance monitoring on a large PV plant

This chapter concerns the study of a PV plant built and maintained by EDF. The precise location is not given and data are normalized for a confidentiality matter. Mainly, the numerical code, based on the one diode modeling (among the advanced physical models presented in Section 1.3.2) is considered in this chapter. Particularly, the numerical code introduced in Section 1.4.4 is used to complete the study in a performance monitoring context. Based on actual production and meteorological data over one year, code calibration better assesses the uncertainties on the parameters input of the Dymola code to this specific case. Once calibration has been run, the code with the new parameter distributions, could predict better the power for a next period of time. The business plan initially conceived on the *prior* parameter distributions, can then be reviewed and the price of electricity adapted to the new estimated power. This chapter presents first the numerical code and the sensitivity analysis performed on the 26 parameters. The *prior* distributions of the parameters are also presented which leads to a propagation of uncertainties *a priori*. Then calibration, in this particular case, is performed but the following issues differ from previous chapters. Indeed, the Dymola code outputs are time series and dealing with multidimensional output in calibration requires additional work.

5.1 Sensitivity analysis

The Dymola code, as presented in Section 1.4.4, is more accurate in the prediction of a large PV plant production than the code used in Chapter 3. Indeed, one of the limitations of the code used to compare calibration models, is to over estimate the power when shades appear. This case scenario occurs every days in a large PV plant configuration because when the sun comes up, the panel in the first raw automatically creates a shade on the one behind. However, because it adds implicit equations that need to be numerically solved, such a development has the effect to slow down the code speed. Moreover, the code estimates the power of the PV plant over a year and produces a time series of $15,858$ points. Applying regular sensitivity analysis (as it was the case for the methods presented in Section 2.1) is more complicated because it has to take into account the behavior of the code on the whole time frame and not only in the instantaneous time. To overcome these issues, the same

method as the one developed in Section 3.1.1 is used. A Morris method is applied and a PCA is performed on the Morris trajectories to give a graphical representation of the indices. The results of the PCA are given in Figure 5.1.

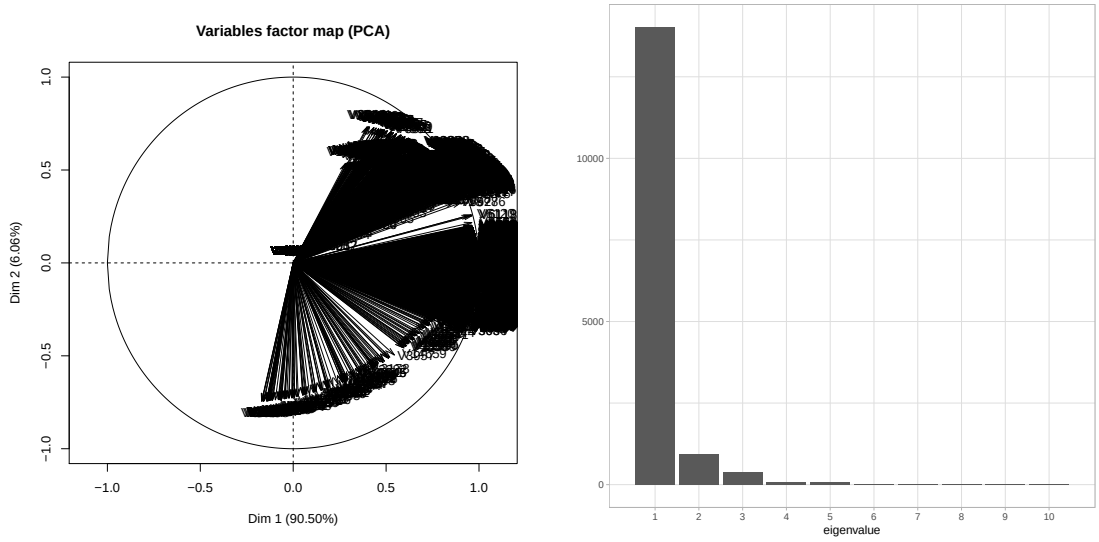

Figure 5.1: The PCA performed on the results given by the Morris method. On the left the correlation circle and on the right the eigenvalues.

Figure 5.1 shows that the time series of the outputs, given by the Morris method, can be projected on a 5 axis basis with keeping more that 99% of the information. Then, this projection in the new space is used to compute the new indices of the Morris method. Figure 5.2 illustrates them all for each of the 5 axes of the dimensional subspace given by the PCA.

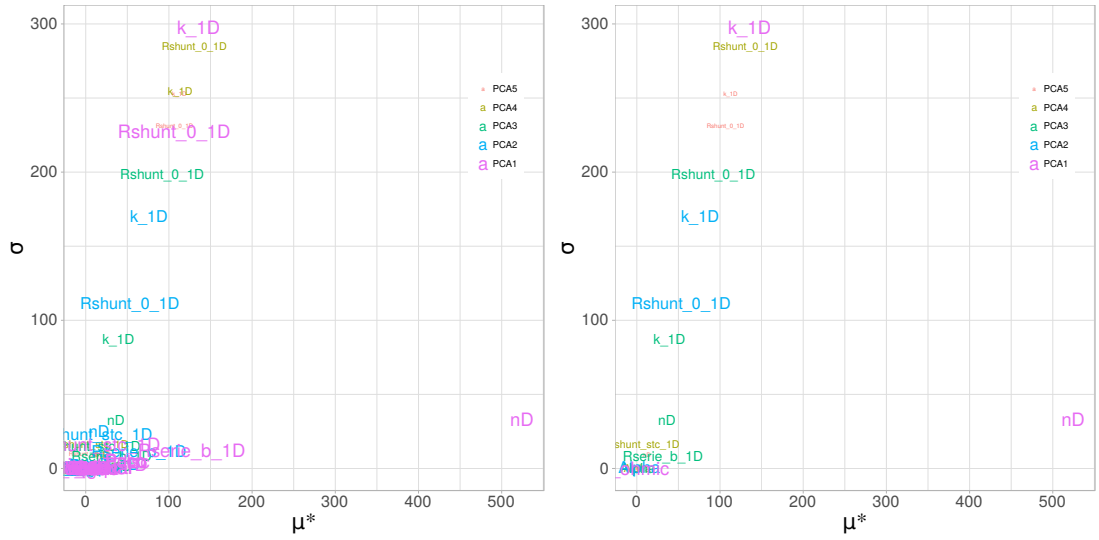

Figure 5.2: Morris indices in the new space given by the PCA. On the left all the parameters names appear and on the right only the ones that are not overlapping are displayed.

Figure 5.2 can also be seen as a representation of the Morris analysis but in a smaller dimension to allow a proper graphical interpretation. On the left panel in Figure 5.2, the new indices on the 5 axes that are carrying 99% of the total inertia is given and allow to identify the parameter that have no impact on the time series output over the time frame. Graphically, 19 parameters are identified as having no impact on the output. It means that calibration

will need to focus on only 7 parameters that are:

- I_{sc} is the short circuit current. It is the current delivered by a PV panel when the voltage at its terminals is zero.
- nD is the ideal factor of the diode in the equivalent electric circuit. It would be close to 1 for a perfect diode. The diode corresponds to the diode of the one in the left electrical scheme in Figure 1.5.
- $Rshunt\_stc\_1D$ is the shunt resistance in standard test conditions. Shunt resistances are typically due to manufacturing defects. Low shunt resistance causes power losses in solar cells by providing an alternate current path for the light-generated current. Its impact on the I/V curve and on the efficiency function of the irradiance is given in Figure 5.3.
- $Rserie\_b\_1D$ is the series resistance in standard test conditions. Series resistance in a solar cell has three causes: first, the movement of current through the emitter and base of the solar cell; second, the contact resistance between the metal contact and the silicon; and finally the resistance of the top and rear metal contacts. Its impact on the I/V curve and on the efficiency function of the irradiance is also given in Figure 5.3.
- $Rshunt\_0\_1D$ and K_1D coefficients that allow the shunt resistance to vary with the irradiation.
- $Paco$ is the maximum ac-power "rating" for inverter at reference or nominal operating condition, assumed to be an upper limit value (see Figure 5.4 for more details).

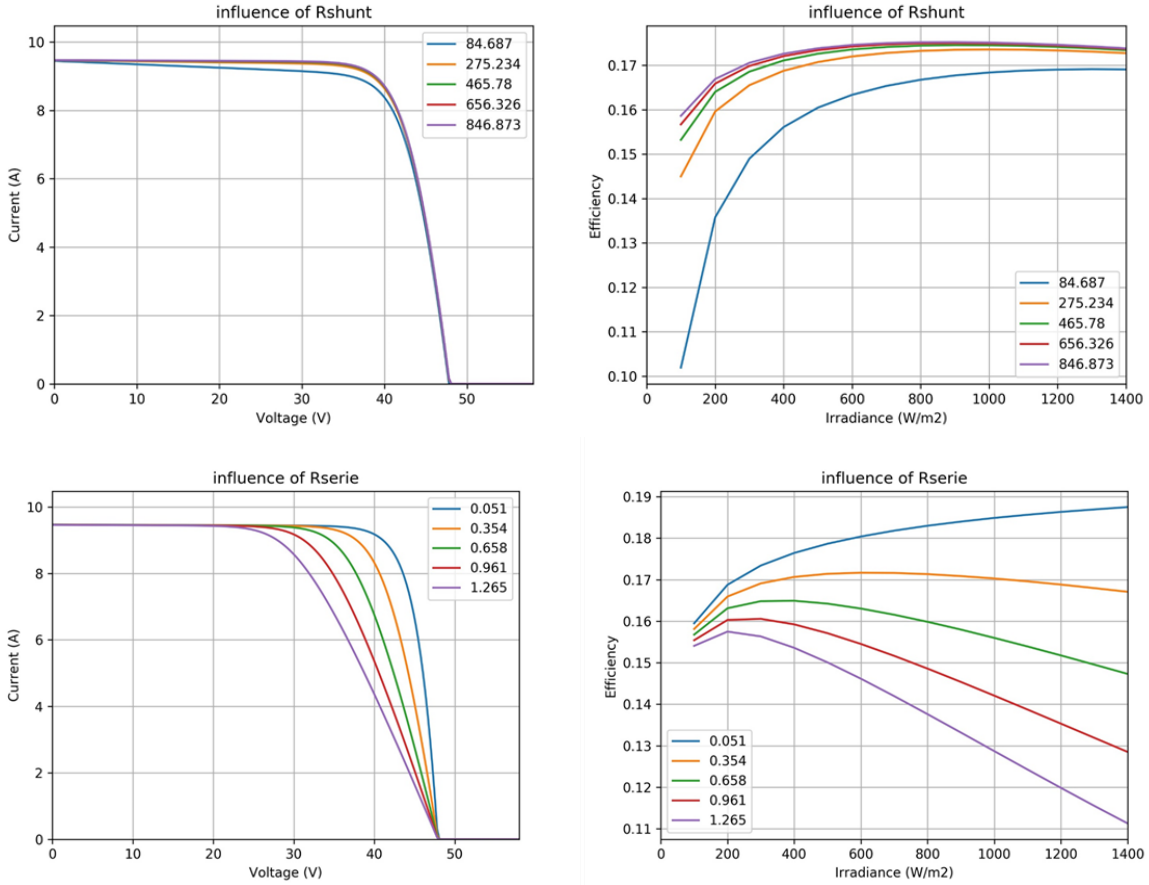

Figure 5.3: On the top left, the impact of different values of shunt resistances on the I/V curve. On the top right the impact of different values of shunt resistances on the evolution of the efficiency function of the irradiance. On the bottom left, the impact of different values of series resistances on the I/V curve. On the bottom right, the impact of different values of series resistances on the evolution of the efficiency function of the irradiance.

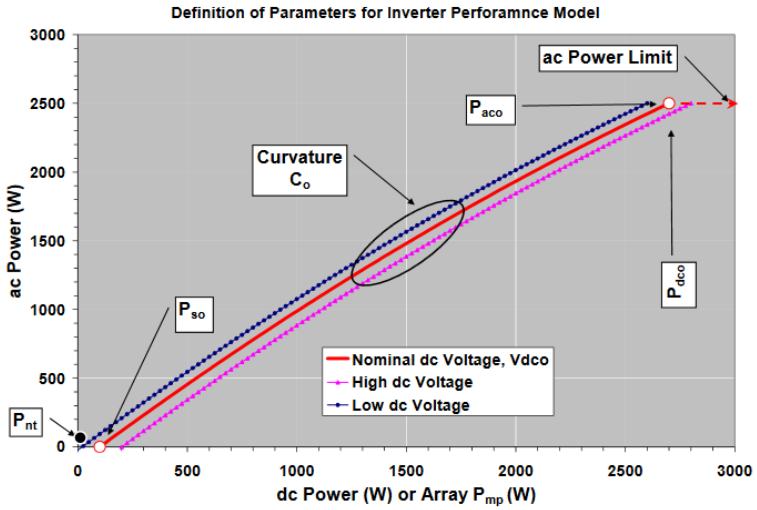

Figure 5.4: Illustration of the inverter performance model and the factors describing the relationship of the ac-output to both dc-power and dc-voltage (*source: https://energy.sandia.gov/wp-content/gallery/uploads/Performance-Model-for-Grid-Connected-Photovoltaic-Inverters.pdf*).

5.2 *Prior* densities

Prior densities are generally established from the expert's knowledge. For all of them a normal density is chosen, except for the variance of the measurement error which is a Gamma. The means of the normal densities are chosen accordingly to the data sheet of the PV panel or of the inverter. The variances are chosen thanks to the credibility intervals given by the manufacturer. As the sensitivity analysis has elected only 7 parameters, the *prior* densities will concern only these ones, the others will be fixed to nominal/reference values. Figure 5.5 illustrates all of them according the credibility intervals given by the experts.

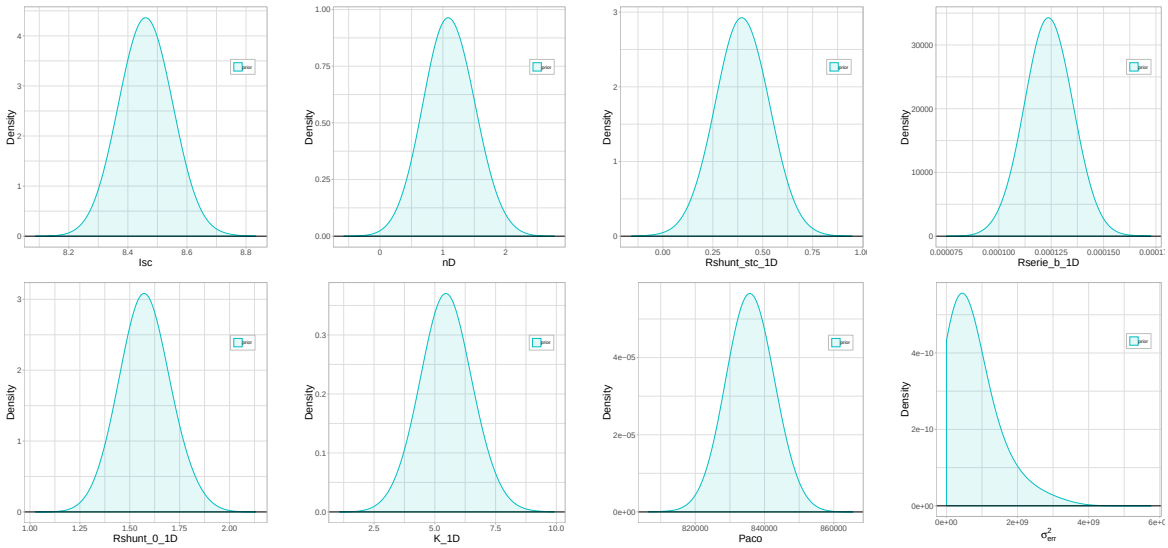

Figure 5.5: *Prior* densities for each parameter considered for further calibration.

5.3 Propagation of uncertainties

The *prior* credibility interval is obtained from sampling in *prior* densities values of $\boldsymbol{\theta}$ and getting the output of the code for each of them. The 95% credibility interval can be obtained. Figure 5.6 illustrates the *prior* credibility interval for the time series confronted with experimental data.

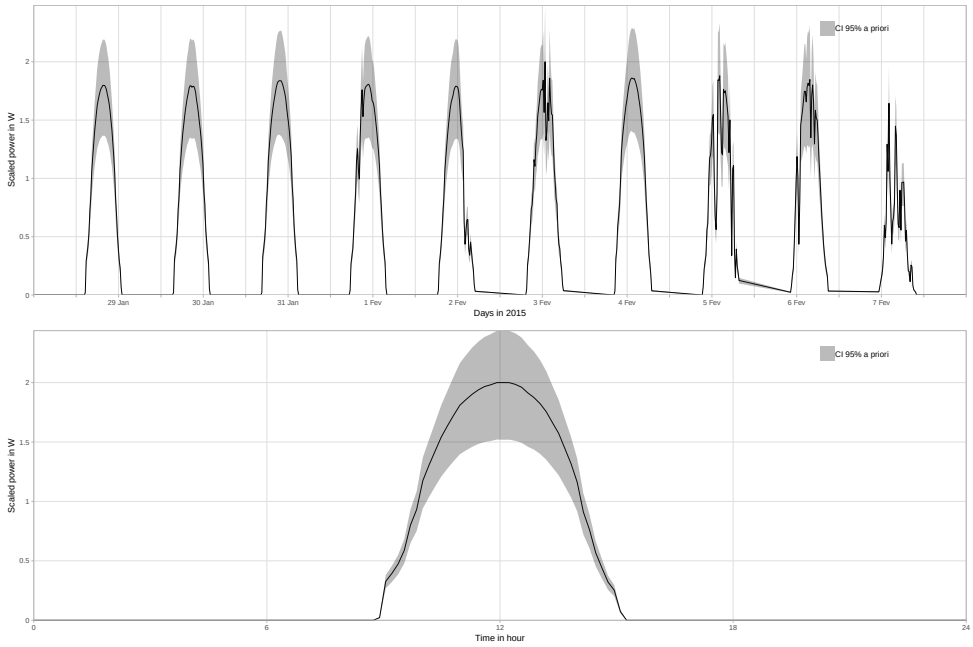

Figure 5.6: Propagation of uncertainties based on *prior* elicitation. On the top experimental data over 10 days in 2015 are displayed with the credibility interval *a priori* and on the bottom, to zoom on the phenomenon, only one day has been plotted (28^{th} of January 2015).

In Figure 5.6, the data are not centered but the scale in the y-axis has been changed. If the centered power has been considered the credibility interval *a priori* would not have been seen smaller than the reality. One can notice, in Figure 5.6 that the credibility interval is not constant during the time series. It increases when the power production grows. It is damaging when one wants to estimate the production for a next period of time. The poor knowledge carried by the *prior* parameter densities does not allow to access to a narrow prediction.

5.4 Bayesian calibration

So far, calibration model and estimation methods have been presented when the numerical code output lies in $\mathbb{R}$. In the non scalar case where the code output is a time series or multidimensional, one can wonder how to write the statistical models and the associated likelihoods to perform calibration. First, we present the models useful to deal with calibration that implies time series outputs. The likelihoods corresponding to each models are also presented and then, the results based on the Dymola code will be presented and commented.

5.4.1 Statistical models

Let us consider the framework introduced in Section 1.4.1. The numerical code with the time series output can be written as:

$$\begin{aligned} f_c : \mathscr{Q} &\to \mathbb{R}^T \\ \boldsymbol{\theta} &\mapsto \boldsymbol{Y} \ , \end{aligned} \tag{5.1}$$

where T stands for the number of instantaneous power observed and $\mathscr{Q} \subset \mathbb{R}^p$. Note that no input variables $\boldsymbol{X}$ are taken into account in this formalization since they are implicitly implemented in the Dymola code as mentioned in Section 1.4.4 which is significantly different from the Higdon et al. (2008).

The case of a time consuming code is assumed in this section. To have a surrogate that reproduces a time series, the first idea would be to generate a surrogate for each time step. When T is too high a reduction of dimension must be considered. In that respect, a LHS maximin DOE (see Section 2.3) of n points is generated and the times series are computed for each configurations of $\boldsymbol{\theta}$. The results are stored in a matrix $\boldsymbol{Y}$ of size $n \times T$. A PCA is performed on the reduced and centered results matrix $\boldsymbol{Y}$. It is, then, possible to identify d principal components that are carrying more than 99% of the total inertia. It means that the sub-space with the d dimensional PCA basis is sufficient to represent well enough the simulations. Let us note $\boldsymbol{P}$ the transition matrix from the physical basis to the PCA basis such as:

$$\boldsymbol{P} : \mathbb{R}^T \to \mathbb{R}^T. \tag{5.2}$$

The matrix $\boldsymbol{P}$ represents the $T \times T$ matrix that allows to get the coordinates of $\boldsymbol{Y}$ on the T PCA axes.
In the space of the PCA, the model $\mathscr{M}_4$ (Equation (3.5)) can be projected as:

$$\boldsymbol{P}\boldsymbol{Y}_{exp} = \boldsymbol{P} f_c(\boldsymbol{\theta}) + \boldsymbol{P}\boldsymbol{\delta} + \boldsymbol{P}\boldsymbol{E}. \tag{5.3}$$

The PCA on simulated data gives the information that all the generated trajectories by the code can be represented at 99% in a d sized orthogonal subspace. It is then possible to identify two subspaces. The first is built with the d first eigenvectors given by the PCA and the second in the orthogonal subspace generated with the $T-d$ eigenvectors left. The PCA tells that $\boldsymbol{P} f_c(\boldsymbol{\theta})$ mainly lies in a d-dimension orthogonal subspace. It is then possible to represent accurately the numerical code by using d independent Gaussian processes that emulates the projections of the code on the d axes of the PCA. The hypothesis made here is to consider that the discrepancy represents everything that is left over in the $T-d$ dimensional subspace. The statistical model would become:

Model with PCA

$$\mathscr{M}_4' : \quad \boldsymbol{P}\boldsymbol{Y}_{exp} = \boldsymbol{P}_F \begin{pmatrix} f_{c_1}(\boldsymbol{\theta}) \\ \vdots \\ f_{c_d}(\boldsymbol{\theta}) \end{pmatrix} + \boldsymbol{P}_\delta \boldsymbol{\delta} + \boldsymbol{E} = \boldsymbol{P}_F \boldsymbol{F} + \boldsymbol{P}_\delta \boldsymbol{\delta} + \boldsymbol{E}, \tag{5.4}$$

where $f_{c_1}, \ldots, f_{c_d}$ (the components of $\boldsymbol{F}$) are equal to $P_1 f_c, \ldots, P_d f_c$ with $P_1, \ldots, P_d$ the projections on the d principal components, and $\boldsymbol{E}$ is equal to $\boldsymbol{P}\boldsymbol{E}$ in probability distribution because the matrix $\boldsymbol{P}$ is orthonormal. The PCA is performed on centered data, so it is possible to formulate the hypothesis that the Gaussian processes on each axis would be defined as $f_{c_i}(\bullet) \sim \mathscr{GP}(0, \sigma_{pi}^2 r_{pi}(\bullet, \bullet))$. This is a *prior* belief that is expressed by Higdon et al. (2008) and as code parameters might not have the same impact on the different axis, we rely on anisotropic Gaussian processes. The discrepancy represents the rest of the projection of the data on the $T-d$ axes of the PCA and can be modeled as an independent and identically distributed Gaussian noise which is a random vector defined as $\boldsymbol{\delta} \sim \mathscr{N}(\boldsymbol{0}_{T-d}, \sigma_\delta^2 I_{T-d})$. The matrix $\boldsymbol{P}_F$ and $\boldsymbol{P}_\delta$ are two matrix that are composed of, respectively, the d first

eigenvectors and the $T-d$ remaining eigenvectors.

If no discrepancy is considered, the statistical model can be simplified as:

Model with PCA without discrepancy

$$\mathscr{M}_2' : \quad \boldsymbol{P}\boldsymbol{Y}_{exp} = \boldsymbol{P}_F\boldsymbol{F} + \boldsymbol{E}, \tag{5.5}$$

5.4.2 Modular estimation and likelihoods

To fulfill a proper modular estimation in the sense of modularization (Liu et al., 2009), the likelihoods have to be written. The estimation is performed in two steps. On each of the d axis that are supporting the Gaussian processes, the partial likelihood can be written from the coordinates of the n points obtained from the DOE $\boldsymbol{D}$ used to perform the PCA. The coordinates on the d axis of the PCA are called $\boldsymbol{Y}_{c_1},\dots,\boldsymbol{Y}_{c_d}$. The i^{th} partial likelihood can be expressed as:

$$\mathscr{L}_i^M(\Phi_i;\boldsymbol{Y}_{c_i},\boldsymbol{D}) = \frac{1}{(2\pi)^{n/2}|\sigma_{pi}^2 r_{pi}(\boldsymbol{D})|^{1/2}} \exp\left\{-\frac{1}{2}\left(\boldsymbol{Y}_{c_i}\right)^T(\sigma_{pi}^2 r_{pi}(\boldsymbol{D}))^{-1}\left(\boldsymbol{Y}_{c_i}\right)\right\}, \tag{5.6}$$

where σ_{pi}^2 and r_{pi} are the *prior* variance of the covariance function and the *prior* anisotropic correlation function relative to the i^{th} Gaussian process. The estimation of all the nuisance parameters of each Gaussian process can be done in a Bayesian framework where *prior* densities of all the parameters are given and the *posterior* densities are sampled with MCMC algorithms. It could also be performed using maximum likelihood estimates to get estimators of the nuisance parameters.

In a Bayesian framework, the *prior* densities can be chosen according to Higdon et al. (2008) where they consider σ_{pi}^2 as a Gamma distribution, and each ψ_{ij} (the correlation lengths in r_{pi}) as a Beta distribution where i would be the i^{th} axis and j would be the j^{th} parameter component. Once estimators of the parameters *a posteriori*, such as the Maximum A Posteriori (MAP), are found they are plugged into the conditional likelihood. For the following, the method of using the maximum likelihood estimates to find estimators of each Gaussian processes is picked. Let us, then, consider that the i^{th} Gaussian process conditioned on the DOE is defined as $f_{c_i}(\bullet)|P_i f_c(D) \sim \mathscr{GP}(m_i(\bullet),\sigma_i^2 c_i(\bullet,\bullet))$.

In the case where the discrepancy is a white Gaussian noise and independent Gaussian processes are chosen for each as surrogates on each axis, the full likelihood can be written straightforwardly. If we consider $\hat{\Phi}$ as the set of the nuisance parameters for the d surrogates estimators, then the full likelihood of the model $\mathscr{M}_4'$ is:

$$\mathscr{L}^f(\boldsymbol{P}\boldsymbol{Y}_{exp};\theta,\sigma_{err}^2,\sigma_\delta^2,\hat{\Phi}) = \frac{1}{(2\pi)^{d/2}\Pi_{i=1}^d\sqrt{\sigma_{err}^2+\sigma_i^2(\theta)}} \exp\left(-\frac{1}{2}\sum_{i=1}^d \frac{(\boldsymbol{P}_{F_i}^T\boldsymbol{Y}_{exp}-m_i(\theta))^2}{\sigma_{err}^2+\sigma_i^2(\theta)}\right)$$
$$\frac{1}{(2\pi(\sigma^2+\sigma_\delta^2))^{(T-d)/2}} \exp\left(-\frac{1}{2(\sigma_{err}^2+\sigma_\delta^2)}\|\boldsymbol{P}_\delta^T\boldsymbol{Y}_{exp}\|^2\right). \tag{5.7}$$

And for the model without discrepancy $\mathscr{M}_2'$:

$$\mathscr{L}^f(\boldsymbol{PY}_{exp};\theta,\sigma_{err}^2,\hat{\Phi}) = \frac{1}{(2\pi)^{d/2}\Pi_{i=1}^d\sqrt{\sigma_{err}^2+\sigma_i^2(\theta)}}\exp\left(-\frac{1}{2}\sum_{i=1}^{d}\frac{(\boldsymbol{P}_{F_i}^T\boldsymbol{Y}_{exp}-m_i(\theta))^2}{\sigma_{err}^2+\sigma_i^2(\theta)}\right) \frac{1}{(2\pi(\sigma_{err}^2))^{(T-d)/2}}\exp\left(-\frac{1}{2(\sigma_{err}^2)}\|\boldsymbol{P}_\delta^T\boldsymbol{Y}_{exp}\|^2\right). \tag{5.8}$$

5.4.3 Application to the PV plant

The Dymola code is time consuming and produces a time series output. In this section, both models $\mathscr{M}_2'$ and $\mathscr{M}_4'$ will be applied to the code and the results will be commented. First, the necessary PCA is performed before carrying out calibration. Then, after getting the estimators for the maximization of the partial likelihood, calibration is performed.

The PCA

As introduced earlier, the Dymola code possesses a time series outputs of $15,858$ points. A PCA is performed on the output of the code of a maximin LHS DOE (see Section 2.3) of $n = 300$ points. The results of the PCA obtained are displayed Figure 5.7.

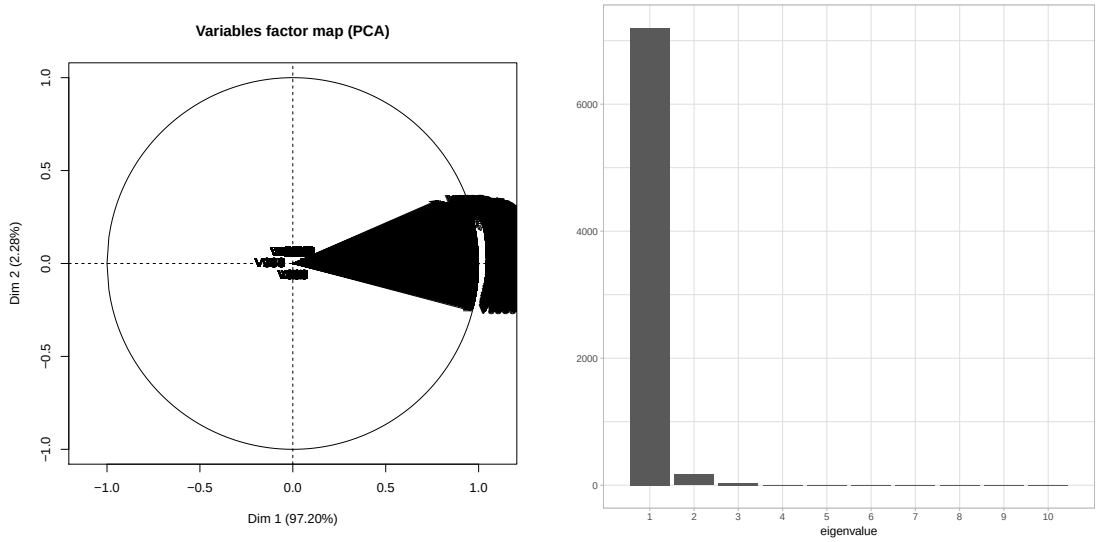

Figure 5.7: The PCA performed on the outputs gotten from the DOE of 300 points. On the left, the correlation circle and on the right, the eigenvalues.

The PCA performed on the maximin LHS gives different results from Figure 5.1 because between both 7 parameters has been selected by the sensitivity analysis. The DOE generated by the Morris method was implying displacements on a 26 dimensional space and the maximin LHS only focuses on these 7 parameters. It means that we might have lost a little piece of information which is not relevant regarding the variations of power in the time series output. That is why in the PCA performed with the Morris DOE more dispersion is visible.

Figure 5.7 illustrates the correlation circle and the eigenvalues obtained after the PCA. It allows to consider that only $d = 5$ axes are carrying more than 99% of the total inertia of the points. Regarding Figure 5.7, 3 axis could have been considered instead of 5 but, as the time consuming part is focused on the establishment of the DOE, it does not cost a lot to use 5 axis and then limit the projection error. In this example, $\boldsymbol{P}_F : \mathbb{R}^T \to \mathbb{R}^5$ is the projection on the subspace of the first 5 principal components. The matrix that is representing the projection on the

orthogonal subspace to the 5 principal components is then noted $\boldsymbol{P}_{\delta} : \mathbb{R}^{T} \rightarrow \mathbb{R}^{T-5}$. The correlation between the power computed by the numerical code and the the PCA axis (computed from Equation (2.100)) is visible in Figure 5.8. The irradiation is plotted on the figure as well because we want to link the physical phenomenon to each axis of the PCA. Each PCA axis should reproduce a different characteristic of the time series and Figure 5.8 shows that the first component of the PCA is representing the average value of the power, when the second axis seems to take care of power fluctuations. The first axis presents some peaks at low values of irradiation because when the sun is low in the sky, some non-linear effects appear which affect the production and the averaged power. The third axis looks like it re-creates the behavior of the PV plant when the irradiation is low. The values are near zero and the peaks are occurring at the same time as the peaks represented on the first axis. This may represent the non linear behavior of the panel when the sun rays angle is low. The moments when the sun rises and when it sets are when shades are occurring. So the third axis can represent the parts of the code that are dealing with the shades and missmatch effects. The fourth and the fifth axis are not given because they are almost entirely flat and does not carry a strong interpretation.

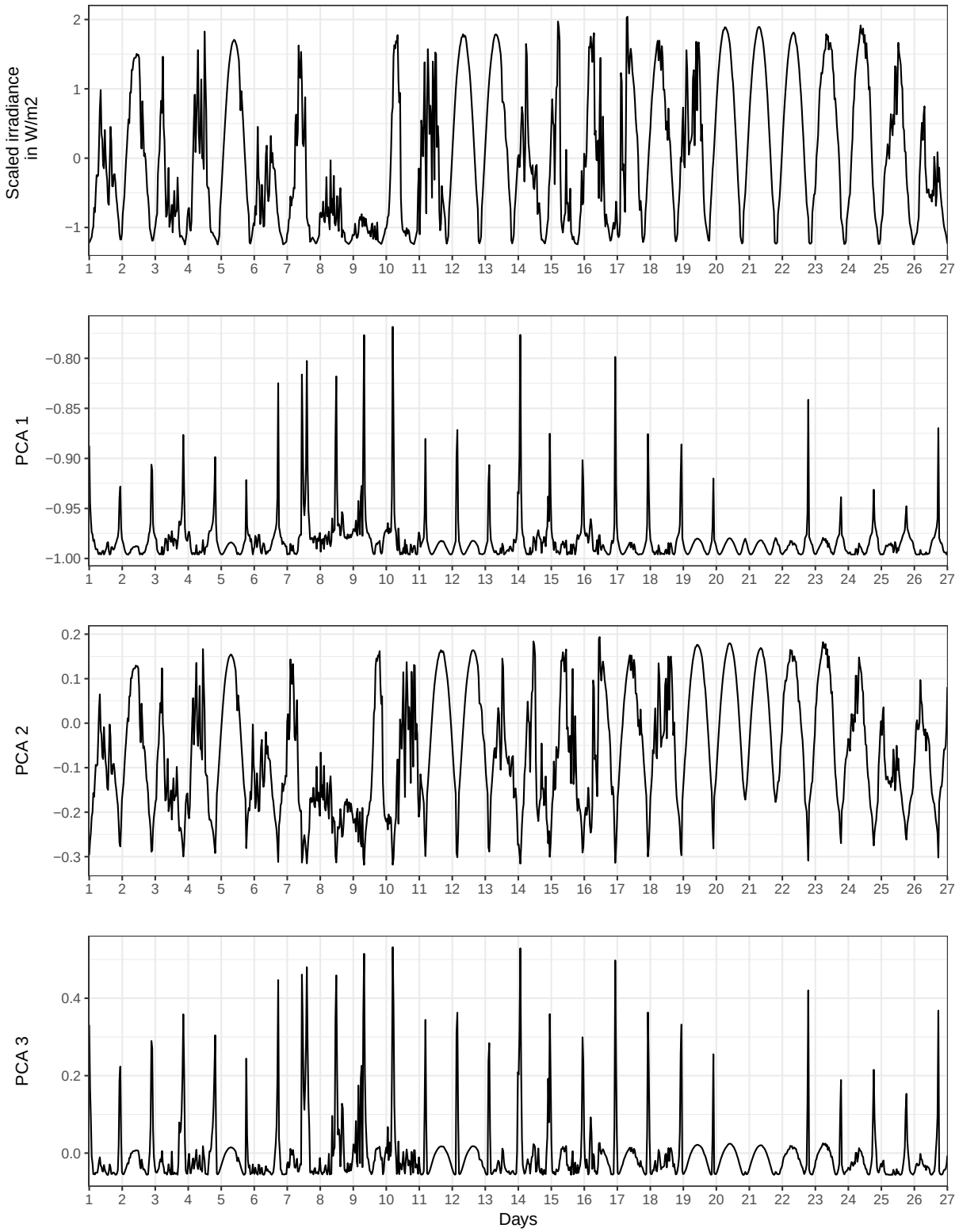

Figure 5.8: The irradiation and the correlation of the power recorded with the three first PCA axes for the 27 first days. On the top the scaled irradiance, on the middle top the correlation of power recorded with the first axis given by the PCA. On the middle bottom, the correlation between the recorded power and the second PCA axis, and on the bottom, the correlation with the third PCA axis.

Calibration

Calibration is performed under some hypotheses. The first one is to consider anisotropic Matérn $5/2$ kernels for the Gaussian processes. A maximum of the partial likelihood is used to find the nuisance parameters relative to each Gaussian process. Different values of correlation lengths according the parameters and different variances are expected because the influence of each Gaussian process is diverse. The values for these parameters are given in Figure 5.9.

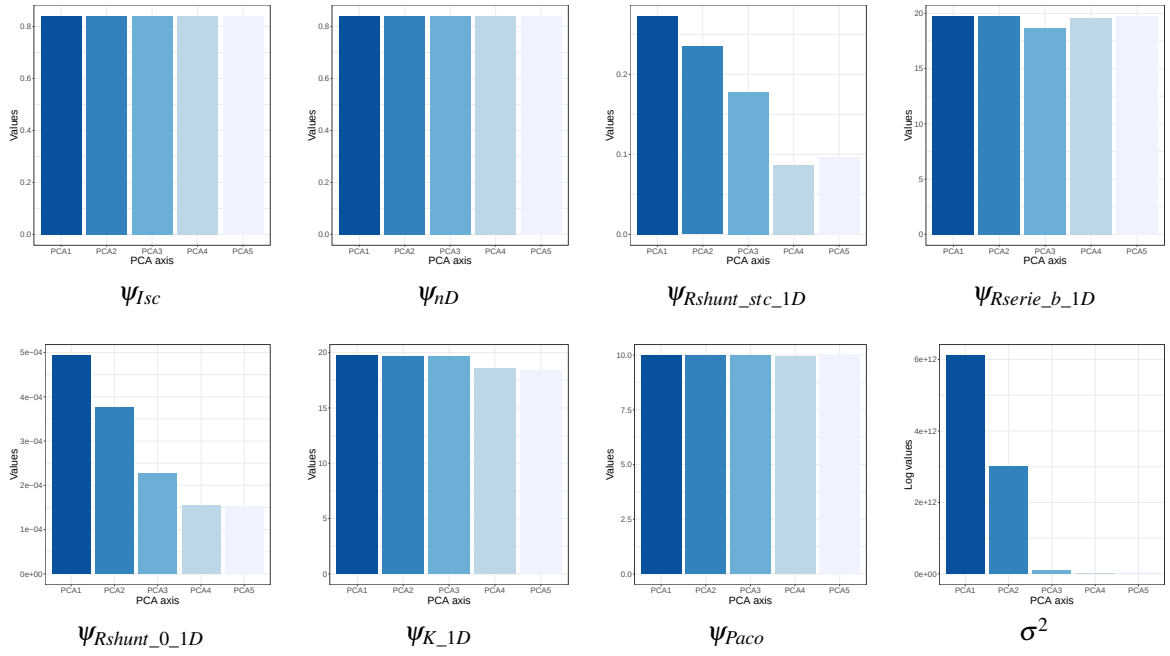

Figure 5.9: Correlation lengths for each component of the parameter vector $\boldsymbol{\theta}$ and for the variance of the 5 Gaussian processes on each PCA axis.

Figure 5.9 illustrates the histograms of each correlation length value for each parameter and the variance of the kernel for the 5 Gaussian processes used. The value of the variance is noticeable because after the third principal component the variance decrease drastically. It means that the variance of the phenomenon carried by the fourth and fifth axes is negligible regarding the three first ones. This result is in agreement with the right panel of Figure 5.7 where the the fourth and the fifth eigenvalues were significantly lower than the three first ones. The correlation lengths are homogeneous except for the component R_shunt_1D where the correlation length is decreasing. The correlation function used to establish the Gaussian process is a Matérn $5/2$ (Equation (2.51)). In the package DiceKriging (Roustant et al., 2015), the Matérn correlation function is coded such as the correlation factor is in the denominator of an exponential. It means that if the value of ψ decreases, the weight of the correlation structure is declining.

Once the Gaussian processes parameters are estimated by MLE, the estimators are plugged into the conditional likelihood that allows to sample in the full likelihood for a regular MCMC. This method might bring instability in calibration because of the computation of the full likelihood. Indeed, in Equation (5.7), the second term, that introduces the term $||\boldsymbol{P}_\delta \boldsymbol{Y}_{exp}||^2$ can potentially be huge and counterparts the first one. A separation in the estimation can be done. The way to do it would be to separate the likelihoods in two, and perform two Bayesian calibrations.

The first one, the estimation is performed on the first part of the likelihood which is:

$$\frac{1}{(2\pi)^{d/2}\Pi_{i=1}^{d}\sqrt{\sigma_{err}^2+\sigma_i^2(\theta)}}\exp\left(-\frac{1}{2}\sum_{i=1}^{d}\frac{(\boldsymbol{P}_{F_i}^T\boldsymbol{Y}_{exp}-m_i(\theta))^2}{\sigma_{err}^2+\sigma_i^2(\theta)}\right). \tag{5.9}$$

Estimators of $\boldsymbol{\theta}$, σ_{err}^2 and σ_δ^2 are then found and used as new starting points of the estimation based on the second part of the likelihood:

$$\frac{1}{(2\pi(\sigma^2+\sigma_\delta^2))^{(T-d)/2}}\exp\left(-\frac{1}{2(\sigma_{err}^2+\sigma_\delta^2)}\|\boldsymbol{P}_\delta^T\boldsymbol{Y}_{exp}\|^2\right). \tag{5.10}$$

This method allows to perform the estimation even if the second term in the likelihood creates too much weight. This case can easily occur because it is highly dependent on the size of the vector $\boldsymbol{P}_\delta\boldsymbol{Y}_{exp}$ (which is T). In time series outputs, the number of points is often high and having recourse to this split estimation solves the issue. Calibration is performed via the **CaliCo** package presented in Chapter 4. The chosen *prior* densities are the one presented in Section 5.2. First the model without discrepancy is considered. This model $\mathscr{M}_2'$ encompasses the 5 Gaussian processes estimated above. The results are displayed in Figure 5.10.

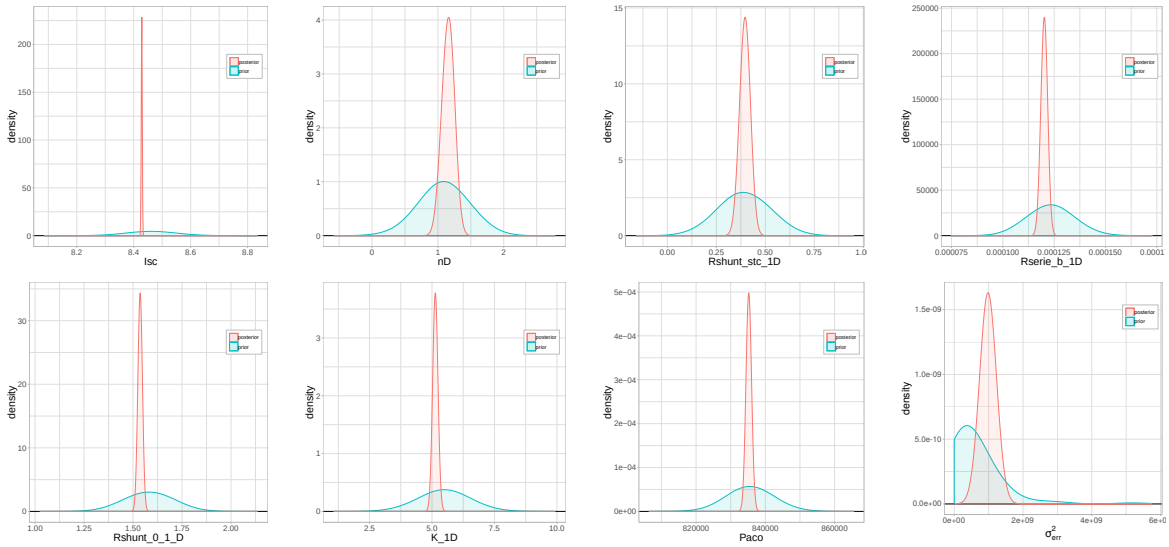

Figure 5.10: *Prior* and *posterior* densities for each parameter in a $\mathscr{M}_2'$ calibration.

The results given in Figure 5.10 are obtained after visual checking of the good mixing properties in the MCMC chains. The *posterior* density for each parameter has less variance than the *prior* density which indicates that information in the PCA subspace has been useful to improve the knowledge on the parameter densities. The variance of the measurement error looks coherent regarding the *prior* density but is a little over estimated in that case because the projection errors are not taken into account and might have introduced this increase of variance.

Let us now consider the model that encompasses the discrepancy modeled as a multidimensional Gaussian noise with a variance σ_δ^2. The results for that model $\mathscr{M}_4'$ are given in Figure 5.11.

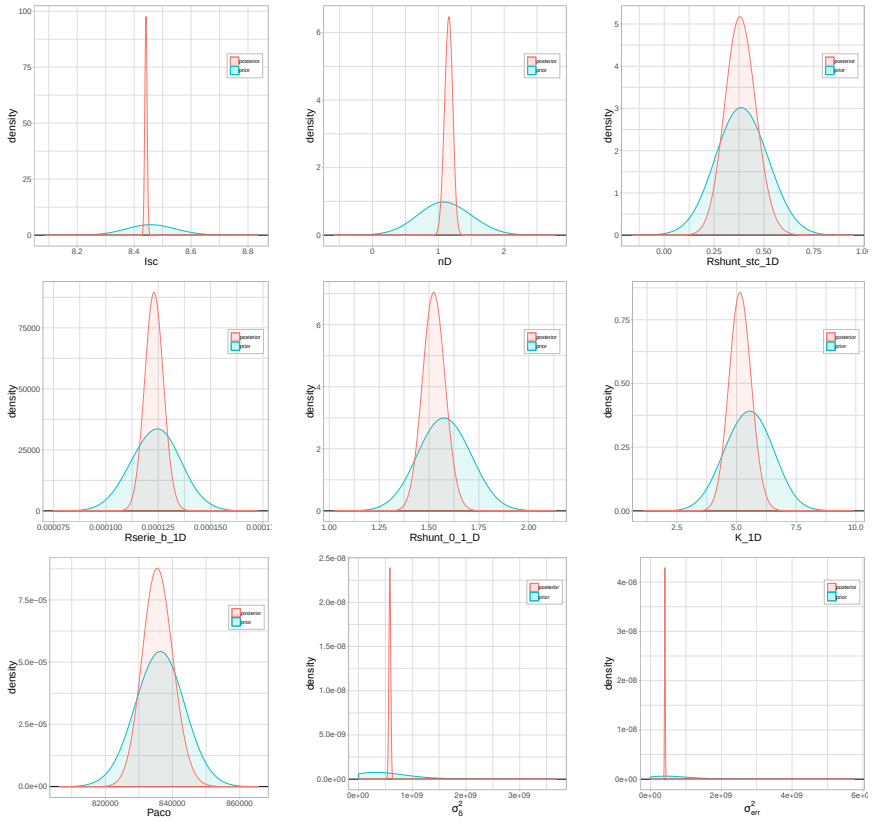

Figure 5.11: *Prior* and *posterior* densities for each parameter in a $\mathscr{M}_4'$ calibration.

Figure 5.11 shows that the maximum *a posteriori* are the same for the second model. When the projection error is taken into account in the model, the variance of the measurement error has decreased and has an acceptable value. However, when the discrepancy is introduced the *posterior* parameter densities have larger variances than the ones estimated without. It can be explained by the introduction of a new parameter σ_δ^2 that tends to increase the uncertainty when the estimation is looked for. The modes *a posteriori* look coherent for both estimation. The interest of adding the discrepancy here is low because the variances of the parameter densities have increased and the estimation of the variance of the measurement error was not so bad with $\mathscr{M}_2'$. When the numerical code is accurate, the introduction of the discrepancy is maybe not necessary. Some methods, in statistical validation, allow to pick the most likely model given experimental data between the one with discrepancy and the one without (Damblin et al., 2016).

The split estimation may have underestimated the variance *a posteriori* of the σ_δ^2 and σ_{err}^2 variances. Another practical solution could have been to compute the Maximum Likelihood Estimates and start the algorithm at this estimator. The MLE can be expressed straightforwardly by deriving Equation 5.7 regarding σ_{err}^2 or σ_δ^2. Indeed, if the log-likelihood of Equation (5.7) is derived regarding σ_δ^2 then we obtain:

$$\frac{(T-d)}{2(\sigma_{err}^2+\sigma_\delta^2)} - \frac{||\boldsymbol{P}_\delta \boldsymbol{Y}_{exp}||^2}{2(\sigma_{err}^2+\sigma_\delta^2)^2} = 0, \tag{5.11}$$

which leads to:

$$(\sigma_\delta^2+\sigma_{err}^2) = \frac{||\boldsymbol{P}_\delta \boldsymbol{Y}_{exp}||^2}{(T-d)}. \tag{5.12}$$

If the derivative is done regarding σ_{err}^2, the following relationship is found:

$$-\frac{d}{2\sigma_{err}^2}+\frac{||\boldsymbol{P}_F\boldsymbol{F}-\boldsymbol{P}_F\boldsymbol{Y}_{exp}||^2}{2\sigma_{err}^4}=\frac{(T-d)}{2(\sigma_{err}^2+\sigma_\delta^2)}-\frac{||\boldsymbol{P}_\delta\boldsymbol{Y}_{exp}||^2}{2(\sigma_{err}^2+\sigma_\delta^2)^2}.$$

Thanks to Equation (5.11) the right term is equal to 0 and:

$$\sigma_{err}^2=\frac{||\boldsymbol{P}_F\boldsymbol{F}-\boldsymbol{P}_F\boldsymbol{Y}_{exp}||^2}{d}, \tag{5.13}$$

$$\sigma_\delta^2=\frac{||\boldsymbol{P}_\delta\boldsymbol{Y}_{exp}||^2}{(T-d)}-\frac{||\boldsymbol{P}_F\boldsymbol{F}-\boldsymbol{P}_F\boldsymbol{Y}_{exp}||^2}{d}. \tag{5.14}$$

The use of the PCA had allowed us to summarize the information contained in the time series output of the code, and to emulate the projections in a restricted space. The hypotheses made on the discrepancy can be discussed. Indeed, representing the discrepancy as the projection on the orthogonal space might be a too subtle representation. One could have think of a representation of the discrepancy as an error term on several axis and not on all the orthogonal space. The results obtained after calibration state that a numerical code which reproduces well enough the physical system does not need the add of the discrepancy. Regarding the results, the discrepancy just add a parameter to estimate and increases the variance *a posteriori* of the parameter densities. It could have been interesting to apply statistical validation, with the Bayes factor for example (Damblin et al., 2016), and compare the results with the ones obtained in Chapter 3 where the discrepancy had an positive impact. The predictive aspect can also be developed. The cross validation has not been tested on these models but a month of data could have been taken off, calibration performed on the remaining test data set and predictive tests on the power could have been run on the month that have been taken off. The characterization of the meteorological data could also have been considered. To get back in the frame of the article of Higdon et al. (2008), the numerical code could have been modified such as a $\boldsymbol{X}$ matrix, which represents the input variables, would be introduced in the input of the code.

CONCLUSION AND PERSPECTIVES

The general framework of this thesis is Bayesian calibration of numerical codes. The objective was to better asses the credibility interval *a posteriori* of the quantity of interest when using an industrial code. This objective is really important for industrial companies such as EDF because numerical codes are used in prediction in many contexts and these companies have difficulties to estimate the predictive error they could make. In the economical framework at EDF, it becomes relevant especially in the forecasting context where power plants are already built and data are already gathered. Then, the numerical code initially used to predict the power before the construction of the plant is used again and the results are confronted with data. Calibration is then performed to better asses the knowledge of the input parameters of the code corresponding to the specific plant.

In this thesis, we have introduced four concepts of calibration that are using different kinds of numerical code. We reviewed the main statistical methods that are available in the literature and we have detailed the inference associated with these statistical models. The purpose of the first part is to confront the models with and without discrepancy. We have noticed that, in the case of the numerical code, we used the discrepancy appears to be really important in the modeling. Indeed, the estimation of the variance of the measurement errors was not concordant with the physical reality. Then, when the discrepancy was added, the estimation became correct which implies that the code was carrying an error not taken into account in the model without discrepancy. The second conclusion comes when the initial numerical code turns out to be long to run. In this scenario case, a Gaussian process is used to emulate the behavior of the code. In a real industrial case, numerical codes can be so expensive in time, that we only have recourse to a few code calls. In that case the estimations after calibration are deteriorating because the emulator is not precise enough. It means that the physical interpretation of the parameter density does not stand anymore and new density parameters corresponding to the, bad, emulator are found. If one is interested in finding the right parameter densities even with an emulator, it is possible to have recourse to an adaptive sequential design that finds points regarding further calibration. It drastically improves the emulation quality and allows to find estimators in correspondence with a physical meaning.

Based on the models developed in the first part, I developed a package. The **CaliCo** package is coded in object-oriented language that allows to manipulate easily users requests. It performs Bayesian calibration with recourse to MCMC that are coded in **C++** to improve the speed of these time consuming operations. The package provides flexibility on the model choice which makes it different from other packages published so far. For each step, visual representations are available and the user can easily take advantage of each graph. A graphical interface has also been developed so that EDF will be able to use the package and take advantage of the coded functions for running test and illustrates the improvements they could make using calibration.

The third part focuses on a real test case which brings lots of issues. Indeed, instead of having a scalar output, the code is producing time series outputs. It is a common fact for numerical codes in industries, so calibration has to be adapted to be performed on these codes. To deal with this issue, Higdon et al. (2008) have proposed to run a PCA on a DOE that encompass the simulations of the code. The aim is to reproduce the time series output with a limited amount of Gaussian processes. Indeed the PCA allows to specify a reasonable number of axis that are carrying a majority of the total inertia of the simulations. Then, the projected simulations on these axes are used to estimate the parameters of the Gaussian processes (when one Gaussian process is used to emulate the behavior of the numerical code on one axis). In this particular case we have considered the discrepancy as the projection

on the orthogonal subspace. We have detailed the inference associated with these statistical models and proposed and applied them on a numerical code that predicts the power of a large PV power plant. The result show that the estimation of the parameters of the code are coherent in both cases. The variance *a posteriori* of the parameter densities is higher when the estimation is performed with the model with discrepancy than with the model without. However, the variance of the measurement errors is a little higher when it is estimated with the model without discrepancy but when the discrepancy is introduced, the value stands to decrease to have more reasonable physical sense.

The predictive aspect of the models that encompasses the PCA has to be completed. It could be interesting to visualize how the models works in prediction. Data already gathered on the field could also be split by keeping a month of data as a validation data set. However, with such a complex model and code, for each test set, the DOE has to be computed again and the PCA performed at each time. Such an operation could turn out to be intractable for desktop computers. The impact of the discrepancy is also the subject of statistical validation. Some studies highlight the fact that data gathered on the field also allow to elect the right model. Bayes factor (Damblin, 2015) or mixture models (Kamary, 2016) have been developed and it could be interesting to try out these methods on the application cases provided in this thesis. Working with a time consuming code also turned out to be difficult to deal with. To run proper calibration on a function that emulates the code, this function has to be close to the initial numerical code, otherwise the prediction of the parameter could loose their physical sense. It could be interesting to study deeper the way to find adapted designs for calibration or other kind of emulators. Also calibration on nested numerical codes could be a real challenge. Indeed, emulating a nested code properly with a Gaussian process might be a difficult task and integrating it in a Bayesian calibration framework would be interesting for companies that are using this kind of codes.

BIBLIOGRAPHY

Albert, I., Donnet, S., Guihenneuc-Jouyaux, C., Low-Choy, S., Mengersen, K., Rousseau, J., et al. (2012). Combining expert opinions in prior elicitation. *Bayesian Analysis*, 7(3):503–532.

Andrieu, C., De Freitas, N., Doucet, A., and Jordan, M. I. (2003). An introduction to mcmc for machine learning. *Machine learning*, 50(1-2):5–43.

Auffray, Y., Barbillon, P., and Marin, J.-M. (2010). Maximin design on non hypercube domain and kernel interpolation. *arXiv preprint arXiv:1004.0784*.

Bachoc, F., Bois, G., Garnier, J., and Martinez, J.-M. (2014). Calibration and improved prediction of computer models by universal kriging. *Nuclear Science and Engineering*, 176(1):81–97.

Bayarri, M. J., Berger, J. O., Paulo, R., Sacks, J., Cafeo, J. A., Cavendish, J., Lin, C.-H., and Tu, J. (2007). A framework for validation of computer models. *Technometrics*, 49(2):138–154.

Bull, A. D. (2011). Convergence rates of efficient global optimization algorithms. *Journal of Machine Learning Research*, 12(Oct):2879–2904.

Carmassi, M. (2018). *CaliCo: Code Calibration in a Bayesian Framework*. R package version 0.1.0.

Casella, G. and George, E. I. (1992). Explaining the gibbs sampler. *The American Statistician*, 46(3):167–174.

Chang, W. (2017). ***R6****: Classes with Reference Semantics*. **R** package version 2.2.2.

Cox, D. D., Park, J.-S., and Singer, C. E. (2001). A statistical method for tuning a computer code to a data base. *Computational statistics & data analysis*, 37(1):77–92.

Craig, P. S., Goldstein, M., Rougier, J. C., and Seheult, A. H. (2001). Bayesian forecasting for complex systems using computer simulators. *Journal of the American Statistical Association*, 96(454):717–729.

Cressie, N. A. and Noel, A. (1993). Cassie (1993). statistics for spatial data. vol. 900.

Currin, C., Mitchell, T., Morris, M., and Ylvisaker, D. (1991). Bayesian prediction of deterministic functions, with applications to the design and analysis of computer experiments. *Journal of the American Statistical Association*, 86(416):953–963.

Da Veiga, S. (2015). Global sensitivity analysis with dependence measures. *Journal of Statistical Computation and Simulation*, 85(7):1283–1305.

Damblin, G. (2015). *Contributions statistiques au calage et à la validation des codes de calcul*. PhD thesis, PhD thesis, Université Paris Saclay.

Damblin, G., Barbillon, P., Keller, M., Pasanisi, A., and Parent, É. (2018). Adaptive numerical designs for the calibration of computer codes. *SIAM/ASA Journal on Uncertainty Quantification*, 6(1):151–179.

Damblin, G., Couplet, M., and Iooss, B. (2013). Numerical studies of space-filling designs: optimization of latin hypercube samples and subprojection properties. *Journal of Simulation*, 7(4):276–289.

Damblin, G., Keller, M., Barbillon, P., Pasanisi, A., and Parent, É. (2016). Bayesian model selection for the validation of computer codes. *Quality and Reliability Engineering International*, 32(6):2043–2054.

De Lozzo, M. and Marrel, A. (2016). New improvements in the use of dependence measures for sensitivity analysis and screening. *Journal of Statistical Computation and Simulation*, 86(15):3038–3058.

Ding, J. and Radhakrishnan, R. (2008). A new method to determine the optimum load of a real solar cell using the lambert w-function. *Solar Energy Materials and Solar Cells*, 92(12):1566–1569.

Duffie, J. A. and Beckman, W. A. (2013). *Solar engineering of thermal processes*. John Wiley & Sons.

Dussert, C., Rasigni, G., Rasigni, M., Palmari, J., and Llebaria, A. (1986). Minimal spanning tree: A new approach for studying order and disorder. *Physical Review B*, 34(5):3528.

Eddelbuettel, D., Francois, R., Allaire, J., Ushey, K., Kou, Q., Russell, N., Bates, D., and Chambers, J. (2018). ***Rcpp****: Seamless R and C++ Integration*. **R** package version 0.12.16.

Faivre, R., Iooss, B., Mahévas, S., Makowski, D., and Monod, H. (2013). *Analyse de sensibilité et exploration de modèles: application aux sciences de la nature et de l'environnement*. Editions Quae.

Fang, K.-T., Li, R., and Sudjianto, A. (2005). *Design and modeling for computer experiments*. CRC Press.

Franco, J., Dupuy, D., Roustant, O., Damblin, G., and Iooss, B. (2015). ***DiceDesign****: Designs of Computer Experiments*. **R** package version 1.7.

Franco, J., Vasseur, O., Corre, B., and Sergent, M. (2009). Minimum spanning tree: A new approach to assess the quality of the design of computer experiments. *Chemometrics and intelligent laboratory systems*, 97(2):164–169.

Gelman, A., Carlin, J. B., Stern, H. S., and Rubin, D. B. (1995). *Bayesian data analysis*. Chapman and Hall/CRC.

Gelman, A. and Rubin, D. B. (1992). Inference from iterative simulation using multiple sequences. *Statistical science*, pages 457–472.

Genz, A., Bretz, F., Miwa, T., Mi, X., Leisch, F., Scheipl, F., Bornkamp, B., Maechler, M., and Hothorn, T. (2018). ***mvtnorm****: Multivariate Normal and t Distributions*. **R** package version 1.0-7.

Ginsbourger, D. (2009). *Multiples metamodeles pour l'approximation et l'optimisation de fonctions numeriques multivariables*. PhD thesis, Ecole Nationale Supérieure des Mines de Saint-Etienne.

Gu, M. (2018a). Jointly robust prior for gaussian stochastic process in emulation, calibration and variable selection. *arXiv preprint arXiv:1804.09329*.

Gu, M. (2018b). ***RobustCalibration****: Robust Calibration of Imperfect Mathematical Models*. **R** package version 0.5.1.

Gu, M. and Wang, L. (2017). Scaled gaussian stochastic process for computer model calibration and prediction.

Haario, H., Saksman, E., Tamminen, J., et al. (2001). An adaptive metropolis algorithm. *Bernoulli*, 7(2):223–242.

Handcock, M. S. and Stein, M. L. (1993). A bayesian analysis of kriging. *Technometrics*, 35(4):403–410.

Hankin, R. K. S. (2013a). ***approximator****: Bayesian prediction of complex computer codes*. **R** package version 1.2-6.

Hankin, R. K. S. (2013b). ***BACCO****: Bayesian Analysis of Computer Code Output (BACCO)*. **R** package version 2.0-9.

Hankin, R. K. S. (2013c). ***calibrator****: Bayesian calibration of complex computer codes*. **R** package version 1.2-6.

Hankin, R. K. S. (2014). ***emulator**: Bayesian emulation of computer programs*. **R** package version 1.2-15.

Hastings, W. K. (1970). Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):97–109.

Helbert, C., Dupuy, D., and Carraro, L. (2009). Assessment of uncertainty in computer experiments from universal to bayesian kriging. *Applied Stochastic Models in Business and Industry*, 25(2):99–113.

Higdon, D., Gattiker, J., Williams, B., and Rightley, M. (2008). Computer model calibration using high-dimensional output. *Journal of the American Statistical Association*, 103(482):570–583.

Higdon, D., Kennedy, M., Cavendish, J. C., Cafeo, J. A., and Ryne, R. D. (2004). Combining field data and computer simulations for calibration and prediction. *SIAM Journal on Scientific Computing*, 26(2):448–466.

Hoff, P. D. (2009). *A first course in Bayesian statistical methods*. Springer Science & Business Media.

Husson, F., Josse, J., Le, S., and Mazet, J. (2018). ***FactoMineR**: Multivariate Exploratory Data Analysis and Data Mining*. **R** package version 1.41.

Husson, F., Lê, S., and Pagès, J. (2017). *Exploratory multivariate analysis by example using R*. Chapman and Hall/CRC.

Ishaque, K., Salam, Z., Taheri, H., et al. (2011). Modeling and simulation of photovoltaic (pv) system during partial shading based on a two-diode model. *Simulation Modelling Practice and Theory*, 19(7):1613–1626.

Jin, R., Chen, W., and Sudjianto, A. (2003). An efficient algorithm for constructing optimal design of computer experiments. In *ASME 2003 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, pages 545–554. American Society of Mechanical Engineers.

Johnson, M. E., Moore, L. M., and Ylvisaker, D. (1990). Minimax and maximin distance designs. *Journal of statistical planning and inference*, 26(2):131–148.

Jones, D. R., Schonlau, M., and Welch, W. J. (1998). Efficient global optimization of expensive black-box functions. *Journal of Global optimization*, 13(4):455–492.

Kamary, K. (2016). *Non-informative priors and modelization by mixtures*. PhD thesis, PhD thesis, Université Paris Dauphine.

Kansa, E. (1985). Application of hardy's multiquadric interpolation to hydrodynamics. Technical report, Lawrence Livermore National Lab.

Kennedy, M. and O'Hagan, A. (2001). Supplementary details on bayesian calibration of computer. rap. tech., university of nottingham. *Statistics Section*.

Kennedy, M. C. and O'Hagan, A. (2001). Bayesian calibration of computer models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 63(3):425–464.

Konomi, B. A., Karagiannis, G., Lai, K., and Lin, G. (2017). Bayesian treed calibration: an application to carbon capture with ax sorbent. *Journal of the American Statistical Association*, 112(517):37–53.

Krige, D. G. (1951). A statistical approach to some basic mine valuation problems on the witwatersrand. *Journal of the Southern African Institute of Mining and Metallurgy*, 52(6):119–139.

Le Gratiet, L. (2013). *Multi-fidelity Gaussian process regression for computer experiments*. PhD thesis, Université Paris-Diderot-Paris VII.

Liu, F., Bayarri, M., Berger, J., et al. (2009). Modularization in bayesian analysis, with emphasis on analysis of computer models. *Bayesian Analysis*, 4(1):119–150.

MacDoanld, B., Chipman, H., and Ranjan, P. (2015). ***GPfit**: Gaussian Processes Modeling*. **R** package version 1.0-0.

Marrel, A. (2008). Mise en oeuvre et exploitation du métamodèle processus gaussien pour l'analyse de modèles numériques-application à un code de transport hydrogéologique. *These de l'INSA Toulouse*.

Marrel, A., Iooss, B., Laurent, B., and Roustant, O. (2009). Calculations of sobol indices for the gaussian process metamodel. *Reliability Engineering & System Safety*, 94(3):742–751.

Matérn, B. (1960). Spatial variation: Meddelanden fran statens skogsforskningsinstitut. *Lecture Notes in Statistics*, 36:21.

Matheron, G. (1963). Principles of geostatistics. *Economic geology*, 58(8):1246–1266.

McKay, M. D., Beckman, R. J., and Conover, W. J. (1979). Comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 21(2):239–245.

Mengyang Gu, J. P. and Berger, J. (2018). ***RobustGaSP**: Robust Gaussian Stochastic Process Emulation*. **R** package version 0.5.5.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., and Teller, E. (1953). Equation of state calculations by fast computing machines. *The journal of chemical physics*, 21(6):1087–1092.

Mira, A. et al. (2001). On metropolis-hastings algorithms with delayed rejection. *Metron*, 59(3-4):231–241.

Morris, M. D. (1991). Factorial sampling plans for preliminary computational experiments. *Technometrics*, 33(2):161–174.

Morris, M. D. and Mitchell, T. J. (1995). Exploratory designs for computational experiments. *Journal of statistical planning and inference*, 43(3):381–402.

Oakley, J. E. and O'Hagan, A. (2004). Probabilistic sensitivity analysis of complex models: a bayesian approach. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 66(3):751–769.

Owen, A. B. (1992). Orthogonal arrays for computer experiments, integration and visualization. *Statistica Sinica*, pages 439–452.

Palomo, J., Garcia-Donato, G., Paulo, R., Berger, J., Bayarri, M. J., and Sacks, J. (2017). ***SAVE**: Bayesian Emulation, Calibration and Validation of Computer Models*. **R** package version 1.0.

Pearson, K. (1901). Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572.

Petrone, G., Spagnuolo, G., and Vitelli, M. (2007). Analytical model of mismatched photovoltaic fields by means of lambert w-function. *Solar energy materials and solar cells*, 91(18):1652–1657.

Picault, D., Raison, B., Bacha, S., De La Casa, J., and Aguilera, J. (2010). Forecasting photovoltaic array power production subject to mismatch losses. *Solar Energy*, 84(7):1301–1309.

Plumlee, M. (2017). Bayesian calibration of inexact computer models. *Journal of the American Statistical Association*, 112(519):1274–1285.

Plummer, M., Best, N., Cowles, K., Vines, K., Sarkar, D., Bates, D., Almond, R., and Magnusson, A. (2016). ***coda**: Output Analysis and Diagnostics for MCMC*. **R** package version 0.19-1.

Pronzato, L. and Müller, W. G. (2012). Design of computer experiments: space filling and beyond. *Statistics and Computing*, 22(3):681–701.

Rasmussen, C. E. (2004). Gaussian processes in machine learning. In *Advanced lectures on machine learning*, pages 63–71. Springer.

Robert, C. (1996). *Méthodes de Monte Carlo par chaînes de Markov*. Economica.

Robert, C. (2007). *The Bayesian choice: from decision-theoretic foundations to computational implementation*. Springer Science & Business Media.

Robert, C. and Casella, G. (2013). *Monte Carlo statistical methods*. Springer Science & Business Media.

Roberts, G. O., Gelman, A., Gilks, W. R., et al. (1997). Weak convergence and optimal scaling of random walk metropolis algorithms. *The annals of applied probability*, 7(1):110–120.

Rocquigny, E. d. (2009). Quantifying uncertainty in an industrial approach: an emerging consensus in an old epistemological debate. *SAPI EN. S. Surveys and Perspectives Integrating Environment and Society*, (2.1).

Roustant, O., Gainsbourger, D., and Deville, Y. (2015). ***DiceKriging**: Kriging Methods for Computer Experiments*. **R** package version 1.5.5.

Roustant, O., Ginsbourger, D., and Deville, Y. (2012). Dicekriging, diceoptim: Two r packages for the analysis of computer experiments by kriging-based metamodelling and optimization. *Journal of Statistical Software*, 51(1):54p.

Sacks, J., Welch, W. J., Mitchell, T. J., and Wynn, H. P. (1989). Design and analysis of computer experiments. *Statistical science*, pages 409–423.

Saltelli, A. (2002). Making best use of model evaluations to compute sensitivity indices. *Computer physics communications*, 145(2):280–297.

Saltelli, A., Chan, K., Scott, E. M., et al. (2000). *Sensitivity analysis*, volume 1. Wiley New York.

Saltelli, A., Tarantola, S., Campolongo, F., and Ratto, M. (2004). *Sensitivity analysis in practice: a guide to assessing scientific models*. John Wiley & Sons.

Santner, T. J., Williams, B. J., and Notz, W. I. (2013). *The design and analysis of computer experiments*. Springer Science & Business Media.

Schaback, R. (2007). Convergence of unsymmetric kernel-based meshless collocation methods. *SIAM Journal on Numerical Analysis*, 45(1):333–351.

Sobol', I. M. (1990). On sensitivity estimation for nonlinear mathematical models. *Matematicheskoe modelirovanie*, 2(1):112–118.

Sobol, I. M. (1993). Sensitivity estimates for nonlinear mathematical models. *Mathematical modelling and computational experiments*, 1(4):407–414.

Stein, M. (1987). Large sample properties of simulations using latin hypercube sampling. *Technometrics*, 29(2):143–151.

Stein, M. L. (2012). *Interpolation of spatial data: some theory for kriging*. Springer Science & Business Media.

Sudret, B. (2008). Global sensitivity analysis using polynomial chaos expansions. *Reliability Engineering & System Safety*, 93(7):964–979.

Tang, B. (1993). Orthogonal array-based latin hypercubes. *Journal of the American statistical association*, 88(424):1392–1397.

Tian, H., Mancilla-David, F., Ellis, K., Muljadi, E., and Jenkins, P. (2012). A cell-to-module-to-array detailed model for photovoltaic panels. *Solar energy*, 86(9):2695–2706.

Tuo, R., Wu, C. J., et al. (2015). Efficient calibration for imperfect computer models. *The Annals of Statistics*, 43(6):2331–2352.

Tuo, R. and Wu, J. (2016). A theoretical framework for calibration in computer models: parametrization, estimation and convergence properties. *SIAM/ASA Journal on Uncertainty Quantification*, 4(1):767–795.

Vazquez, E. and Bect, J. (2010). Convergence properties of the expected improvement algorithm with fixed mean and covariance functions. *Journal of Statistical Planning and inference*, 140(11):3088–3095.

Viana, F. A., Venter, G., and Balabanov, V. (2010). An algorithm for fast optimal latin hypercube design of experiments. *International journal for numerical methods in engineering*, 82(2):135–156.

Wickham, H. and Chang, W. (2016). ***ggplot2**: Create Elegant Data Visualisations Using the Grammar of Graphics*. **R** package version 2.2.1.

Wong, R. K., Storlie, C. B., and Lee, T. (2017). A frequentist approach to computer model calibration. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 79(2):635–648.

université PARIS-SACLAY
ÉCOLE DOCTORALE
Agriculture, alimentation, biologie, environnement, santé (ABIES)

Titre : Quantification des incertitudes et calage d'un modèle de centrale photovoltaïque : garantie de performance et estimation robuste de la production long-terme.

Mots clés : Centrale photovoltaïque / Calage bayésien / Quantification d'incertitudes / Code numérique

Résumé : Les difficultés de mise en œuvre d'expériences de terrain ou de laboratoire, ainsi que les coûts associés, conduisent les sociétés industrielles à se tourner vers des codes numériques de calcul. Ces codes, censés être représentatifs des phénomènes physiques en jeu, entraînent néanmoins tout un cortège de problèmes. Le premier de ces problèmes provient de la volonté de prédire la réalité à partir d'un modèle informatique. En effet, le code doit être représentatif du phénomène et, par conséquent, être capable de simuler des données proches de la réalité. Or, malgré le constant développement du réalisme de ces codes, des erreurs de prédiction subsistent. Elles sont de deux natures différentes. La première provient de la différence entre le phénomène physique et les valeurs relevées expérimentalement. La deuxième concerne l'écart entre le code développé et le phénomène physique. Pour diminuer cet écart, souvent qualifié de biais ou d'erreur de modèle, les développeurs complexifient en général les codes, les rendant très chronophages dans certains cas. De plus, le code dépend de paramètres à fixer par l'utilisateur qui doivent être choisis pour correspondre au mieux aux données de terrain. L'estimation de ces paramètres propres au code s'appelle le calage. Cette thèse propose dans un premier temps une revue des méthodes statistiques nécessaires à la compréhension du calage Bayésien. Ensuite, une revue des principales méthodes de calage est présentée accompagnée d'un exemple comparatif basé sur un un code de calcul servant à prédire la puissance d'une centrale photovoltaïque. Le package appelé CaliCo qui permet de réaliser un calage rapide de beaucoup de codes numériques est alors présenté. Enfin, un cas d'étude réel d'une grande centrale photovoltaïque sera introduit et le calage réalisé pour effectuer un suivi de performance de la centrale. Ce cas de code industriel particulier introduit des spécificités de calage numériques qui seront abordées et deux modèles statistiques y seront exposés.

Title : Uncertainty quantification and calibration of a photovoltaic plant model: warranty of performance and robust estimation of the long-term production.

Keywords : Photovoltaic power plant / Bayesian calibration / Uncertainty quantification / Numerical code

Abstract : Field experiments are often difficult and expensive to make. To bypass these issues, industrial companies have developed computational codes. These codes intend to be representative of the physical system, but come with a certain amount of problems. The code intends to be as close as possible to the physical system. It turns out that, despite continuous code development, the difference between the code outputs and experiments can remain significant. Two kinds of uncertainties are observed. The first one comes from the difference between the physical phenomenon and the values recorded experimentally. The second concerns the gap between the code and the physical system. To reduce this difference, often named model bias, discrepancy, or model error, computer codes are generally complexified in order to make them more realistic. These improvements lead to time consuming codes. Moreover, a code often depends on parameters to be set by the user to make the code as close as possible to field data. This estimation task is called calibration. This thesis first proposes a review of the statistical methods necessary to understand Bayesian calibration. Then, a review of the main calibration methods is presented with a comparative example based on a numerical code used to predict the power of a photovoltaic plant. The package called CaliCo which allows to quickly perform a Bayesian calibration on a lot of numerical codes is then presented. Finally, a real case study of a large photovoltaic power plant will be introduced and the calibration carried out as part of a performance monitoring framework. This particular case of industrial code introduces numerical calibration specificities that will be discussed with two statistical models.

Université Paris-Saclay
Espace Technologique / Immeuble Discovery
Route de l'Orme aux Merisiers RD 128 / 91190 Saint-Aubin, France