Multimode navigation for degraded fixed wing unmanned aerial vehicle operation under sensor and actuator faults

Enzo Iglésis

THESE DE DOCTORAT

NNT : 2021UPAST142

Multimode navigation for degraded fixed wing unmanned aerial vehicle operation under sensor and actuator faults

Navigation multimode d'un drone avec fonctionnalités dégradées dues à des défaillances de capteurs ou actionneurs

Thèse de doctorat de l'université Paris-Saclay et de Coventry University

École doctorale n° 580, sciences et technologies de l'information et de la communication (STIC)
Spécialité de doctorat : Automatique
Graduate School : Sciences de l'ingénierie et des systèmes, Référent : Faculté des sciences d'Orsay

Thèse préparée dans les unités de recherche **Traitement de l'information et systèmes (Université Paris-Saclay, ONERA)** et **Centre for computational science and mathematical modelling (Coventry University)**, sous la direction d'**Hélène PIET-LAHANIER**, Directrice de recherche, et de **James BRUSEY**, Professeur des universités, le co-encadrement de **Karim DAHIA**, Ingénieur de recherche, et de **Nadjim HORRI**, Maître de conférences hors classe

Thèse soutenue à Paris-Saclay, le 21 décembre 2021, par

Enzo IGLÉSIS

Composition du Jury

Jean-Yves TOURNERET Professeur des universités, INP – ENSEEIHT Toulouse	Président
Kamal MEDJAHER Professeur des universités, ENIT Tarbes	Rapporteur & Examinateur
Lyudmila MIHAYLOVA Professeur des universités, The University of Sheffield	Rapporteur & Examinatrice
Dalil ICHALAL Professeur des universités, Université d'Evry Val d'Essonne	Examinateur
Tarek RAISSI Professeur des universités, CNAM Paris III	Examinateur
Hélène PIET-LAHANIER Directrice de recherche, ONERA, Université Paris-Saclay	Directrice de thèse
James BRUSEY Professeur des universités, Coventry University	Directeur de thèse

université PARIS-SACLAY

ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Title: Multimode navigation for degraded fixed wing unmanned aerial vehicle operation under sensor and actuator faults

Keywords: Navigation, Estimation, Faults, Particle Filters, UAVs.

Abstract: Actuator or sensor faults occurring in an unmanned aerial vehicle can compromise the system integrity. Fault diagnosis methods are therefore becoming a required feature for those systems. In this thesis, the focus is on fault estimation for fixed-wing unmanned aerial vehicles in the presence of simultaneous actuator and sensor faults. To deal with the challenging nature of some fault scenarios, such as simultaneous and ambiguous faults that induce multimodality, a jump-Markov regularized particle filter and enhanced versions of it are presented in this thesis.

This method is based on a regularized particle filter that improves the robustness thanks to the approximation of the posterior density by a kernel mixture, and on a jump-Markov system. The jump strategy uses a small number of particles — called sentinel particles — to continue testing the alternate hypothesis under both fault free and faulty modes.

The numerical results are obtained using linear then non-linear longitudinal dynamics of fixed wing unmanned aerial vehicle. It is compared to interacting multiple model Kalman filters and regularized particle filters and shown to outperform them in terms of accuracy, robustness and convergence time in the scenarios considered. The state estimation is also more accurate and robust to faults using the proposed approach. Performance enhancement compared to other filters is more pronounced when fault amplitudes increase.

An enhanced version of this method named the robustified jump-Markov regularized particle filter is also presented and allows one to accurately and rapidly estimate faults with no prior knowledge of the fault dynamics. Finally, a new approach to compute an adaptive transition probability matrix is introduced by computing the false alarm and missed detection probabilities using a saddlepoint approximation.

The proposed approaches significantly improve the safety and accuracy for increasingly autonomous unmanned aerial vehicles and generalize to other control environments where faults occur.

université PARIS-SACLAY

ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Titre : Navigation multimode d'un drone avec fonctionnalités dégradées dues à des défaillances de capteurs ou actionneurs

Mots clés : Navigation, Estimation, Fautes, Filtres particulaires, Drones.

Résumé : Les défaillances d'actionneurs ou de capteurs survenant dans un drone peuvent compromettre l'intégrité de la mission. La mise en œuvre de méthodes de diagnostic de fautes est alors nécessaire. Dans cette thèse, l'accent est mis sur l'estimation de fautes dues à des défaillances simultanées de capteurs et d'actionneurs pour un drone à voilure fixe. Pour faire face à certains scénarios complexes de fautes, tels que les fautes simultanées qui induisent une ambiguïté sur les mesures qui se manifeste par la multimodalité de la densité conditionnelle, un filtre particulaire régularisé de type jump-Markov (JMRPF) et des versions améliorées de celui-ci sont présentées dans cette thèse.

Cette méthode est basée sur un filtre particulaire régularisé (RPF) qui approche la densité conditionnelle par une mixture de noyaux et sur un système de Markov à sauts (JMS). La stratégie de saut utilise un petit nombre de particules — appelées particules sentinelles — qui permet de tester en continue l'hypothèse alternative en mode sans faute et en mode avec faute.

Les résultats numériques sont obtenus en utilisant un modèle dynamique linéaire puis non linéaire de la dynamique longitudinale d'un drone à voilure fixe. Les performances du JMRPF sont comparées aux performances des filtres de Kalman à modèles multiples interactifs (IMM-KF) et du RPF. Les performances du JMRPF montrent une nette amélioration de terme de précision de l'estimation des fautes capteurs et actionneurs et des paramètres cinématiques et en termes de robustesse et de vitesse de convergence par rapport aux autres filtres. L'amélioration des performances par rapport aux autres filtres est plus marquée lorsque l'amplitude des fautes augmente au cours du temps.

Une version améliorée du JMRPF, appelée filtre particulaire régularisé robuste à sauts est également présentée et permet d'estimer rapidement et précisément les fautes sans connaissance a priori de la dynamique des fautes. Enfin, une nouvelle approche pour calculer une matrice de probabilité de transition adaptative est présentée en calculant les probabilités de fausse alarme et de non-détection à l'aide de l'approximation du point-selle.

Les algorithmes de navigation proposés permettent à un drone d'atteindre son objectif de suivi de trajectoire de manière autonome, avec une sécurité et une précision accrues.

MULTIMODE NAVIGATION FOR DEGRADED FIXED WING UNMANNED AERIAL VEHICLE OPERATION UNDER SENSOR AND ACTUATOR FAULTS

ENZO IGLÉSIS

A thesis submitted in partial fulfilment of the University's requirements for the Degree of Doctor of Philosophy

Université Paris-Saclay & Coventry University

October 2018 – September 2021

DIRECTORS OF STUDIES:
Hélène Piet-Lahanier
James Brusey

SUPERVISORS:
Karim Dahia
Nadjim Horri

LOCATION:
Palaiseau (France) & Coventry (United Kingdom)

TIME FRAME:
October 2018 – September 2021

I dedicate this thesis to the loving memory of my father
Alain Iglésis, 16 August 1960 – 19 May 2021

ABSTRACT

Actuator or sensor faults occurring in an unmanned aerial vehicle can compromise the system integrity. Fault diagnosis methods are therefore becoming a required feature for those systems. In this thesis, the focus is on fault estimation for fixed-wing unmanned aerial vehicles in the presence of simultaneous actuator and sensor faults. To deal with the challenging nature of some fault scenarios, such as simultaneous and ambiguous faults that induce multimodality, a jump-Markov regularized particle filter and enhanced versions of it are presented in this thesis.

This method is based on a regularized particle filter that improves the robustness thanks to the approximation of the posterior density by a kernel mixture, and on a jump-Markov system. The jump strategy uses a small number of particles — called sentinel particles — to continue testing the alternate hypothesis under both fault free and faulty modes.

The numerical results are obtained using linear then non-linear longitudinal dynamics of fixed wing unmanned aerial vehicle. It is compared to interacting multiple model Kalman filters and regularized particle filters and shown to outperform them in terms of accuracy, robustness and convergence time in the scenarios considered. The state estimation is also more accurate and robust to faults using the proposed approach. Performance enhancement compared to other filters is more pronounced when fault amplitudes increase.

An enhanced version of this method named the robustified jump-Markov regularized particle filter is also presented and allows one to accurately and rapidly estimate faults with no prior knowledge of the fault dynamics. Finally, a new approach to compute an adaptive transition probability matrix is introduced by computing the false alarm and missed detection probabilities using a saddlepoint approximation.

The proposed approaches significantly improve the safety and accuracy for increasingly autonomous unmanned aerial vehicles and generalise to other control environments where faults occur.

ACKNOWLEDGMENTS

Countless people supported my effort on this thesis. Therefore, I can't thank all of you, so for those I couldn't mention: Thank you!

First, I would like to express my deepest gratitude to my supervisory team for their support and availability all along my thesis. *Karim Dahia*, for his level of patience, knowledge, and his enthusiasm that were needed and much appreciated. *Nadjim Horri*, for his expertise, and technical advice. *Hélène Piet-Lahanier*, for her guidance and help. *James Brusey*, for his direction, backing, and feedback.

I would also like to thank *Lyudmila Mihaylova* and *Kamal Medjaher* for having accepted to be my examiners, *Mike Blundell* for being my Coventry internal examiner, and *Jean-Yves Tourneret* as the Paris-Saclay's jury president. Thanks also to *Dalil Ichalal*, *Tarek Raissi* for having joined the jury of Paris-Saclay.

The opportunity to perform this thesis in two different countries with two different universities, widened my outlook on research and provide me with new perspectives. Then, I would like to thank all individuals with whom I could exchange while I was working on my doctoral research. At the ONERA – The French Aerospace Lab, I would like to thank *Nicolas Merlinge* for his help advises, and contributions which had an influential on my work. *Pascal Fély* and all the members of the design and assessment of aerospace vehicles unit. Also, the other Ph.D. students, among which *Camille Palmier* and *Esteban Restrepo* who for their kindness and friendship all along the thesis. As well, *Julien Pelamatti*, *Ali Hebbal*, *Nathan Michel*, *Emilien Flayac*, *Sergio Pérez Roca*, *Camille Sarotte*, *Julius Ibenthal*, *Etienne Bertin*, *Clément Chahbazian*, *Mathieu Marchand*, *Clara Leparoux*, *Hanae Labriji* and all others with whom I would have liked to be able to share my working environment for longer. At the Coventry University, I would like to thank *Elena Gaura* for having provided her help and support to me. As well as *Damien Foster*, and all other members of the research Centre for Computational Science and Mathematical Modelling, including Ph.D. students that I unfortunately didn't have much opportunity to share time with.

Last but not least, I would like to thank all my family and friends for their unwavering support and encouragement. Especially *John Wilkinson*, for without whom this thesis would have been much more challenging to complete. My mother *Astrid Iglésis* and my brother *Quentin Iglésis*, who were unfailingly supportive and unbeatable motivators. Finally, I would like to thank my father *Alain Iglésis* to whom I dedicate this thesis and who was not only an exemplary father, but who allowed himself to be an inspiring one.

Regarding financial support: I would like to acknowledge the *ONERA – The French Aerospace Lab* and the *Coventry University* for both having fully funded this work and welcomed me in their facilities.

CONTENTS

ACRONYMS

AJMRPF	adaptive jump-Markov regularized particle filter
ARJMRPF	adaptive robustified jump-Markov regularized particle filter
ASL	above sea level
CDF	cumulative distribution function
CUSUM	cumulative sum
DOP	dilution of precision
EKF	extended Kalman filter
GLRT	generalized likelihood ratio test
GNC	guidance navigation & control
GNSS	global navigation satellite system
HDOP	horizontal dilution of precision
i. i. d.	independent and identically distributed
IMM	interacting multiple model
IMM-KF	interacting multiple model Kalman filters
JMRPF	jump-Markov regularized particle filter
JMS	jump-Markov system
LQR	linear quadratic regulator
MAP	maximum a posteriori
MEMS	micro-electromechanical system
MGF	moment-generating function
MISE	mean integrated square error
ML	maximum likelihood
MMSE	minimum mean-square error
NED	North, East, Down
RJMRPF	robustified jump-Markov regularized particle filter
RMSE	root-mean-square error
ROC	receiver operating characteristic
RPF	regularized particle filter

RRPF	robustified regularized particle filter
SIR	sequential importance resampling
SPRT	sequential probability ratio test
UAV	unmanned aerial vehicle
UERE	user-equivalent range error
UKF	unscented Kalman filter
USAF	United States Air Force
VDOP	vertical dilution of precision

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

NOMENCLATURE

Latin Letters

$\acute{\mathbf{x}}_k$	State vector after the multinomial resampling		$\mathbb{R}^{n_x}$
$\bar{\mathbf{V}}_\mathbf{g}$	Trimmed velocity vector	$\mathrm{m\,s^{-1}}$	$\mathbb{R} - \mathbf{V_g}^*$
$\bar{p_d}^c$	Desired trimmed state p_d	m	$\mathbb{R} - p_d{}^*$
$\bar{p_{di}}$	Integrated error of the trimmed state $\bar{p_{di}}$	m	$\mathbb{R}$
$\bar{u_i}$	Integrated error of the trimmed state $\bar{u}$	$\mathrm{m\,s^{-1}}$	$\mathbb{R}$
$\bar{\mathbf{u}}$	Trimmed control input $\mathbf{u}$		$\mathbb{R}^{n_u} - \mathbf{u}^*$
$\bar{\mathbf{u}}_{lon}$	Trimmed longitudinal control input $\mathbf{u}_{lon}$		$\mathbb{R}^2 - \mathbf{u}^*_{lon}$
$\bar{\mathbf{z}}$	Trimmed state vector $\mathbf{z}$		$\mathbb{R}^{n_z} - \mathbf{z}^*$
$\bar{\mathbf{z}}_{lon}$	Trimmed longitudinal state vector $\mathbf{z}_{lon}$		$\mathbb{R}^5 - \mathbf{z}^*_{lon}$
$\bar{p_d}$	Trimmed state p_d	m	$\mathbb{R} - p_d^*$
$\bar{q}$	Trimmed state q	$\mathrm{rad\,s^{-1}}$	$\mathbb{R} - q^*$
$\bar{u}$	Trimmed state u	$\mathrm{m\,s^{-1}}$	$\mathbb{R} - u^*$
$\bar{u}^c$	Desired trimmed velocity along i^b	$\mathrm{m\,s^{-1}}$	$\mathbb{R} - u^*$
$\bar{w}$	Trimmed state w	$\mathrm{m\,s^{-1}}$	$\mathbb{R} - w^*$
$\Delta_{\mathrm{f_s}}$	Value of the state $\mathrm{f_s}$ when switch from fault-free to faulty mode		$\mathbb{R}^*$
Δ_{f}	Value of the state f when switch from fault-free to faulty mode		$\mathbb{R}^*$
$\Delta_{\mathbf{f_a}}$	Values of the state vector $\mathbf{f_a}$ when switch from fault-free to faulty mode		$\mathbb{R}^*$
$\Delta_{\mathbf{f_s}}$	Values of the state vector $\mathbf{f_s}$ when switch from fault-free to faulty mode		$\mathbb{R}^*$
$\mathbb{E}[\cdot]$	Probabilistic expectation		
$\hat{\mathbf{P}}_k$	Estimated covariance of the state vector		$\mathbb{R}^{n_x \times n_x}$
$\hat{\mathbf{x}}$	Estimate of state vector $\mathbf{x}$		$\mathbb{R}^{n_x}$
$\hat{N}_{eff}$	Estimate of the effective number of particles		$\mathbb{R}^{1:N}$
$\mathcal{H}_0$	Nominal hypothesis		
$\mathcal{H}_1$	Faulty hypothesis		

κ	User-defined setting parameter of the bandwidth factor of the regularization kernel		$(0,\, 1)$
$\left\lvert \bar{\mathbf{V}}_{\mathbf{g}}{}^{c} \right\rvert$	Desired trimmed modulus of the velocity vector	$\mathrm{m\,s^{-1}}$	$\mathbb{R}^{+} - \left\lvert \bar{\mathbf{V}}_{\mathbf{g}}{}^{*} \right\rvert$
$\mathbb{M}$	Set of modes		
$\mathcal{F}_{12}(\cdot)$	Twelfth output of $\mathcal{F}(\cdot)$		$\mathbb{R}$
$\mathcal{F}_{1}(\cdot)$	First output of $\mathcal{F}(\cdot)$		$\mathbb{R}$
$\mathcal{F}_{2}(\cdot)$	Second output of $\mathcal{F}(\cdot)$		$\mathbb{R}$
$\mathcal{F}(\cdot)$	Continuous dynamics of the state vector $\mathbf{z}$		$\mathbb{R}^{n_z \times n_u} \to \mathbb{R}^{n_z}$
$\mathcal{F}_{k}(\cdot)$	Discrete dynamics of the state vector $\mathbf{z}$		$\mathbb{R}^{n_z \times n_u} \to \mathbb{R}^{n_z}$
$\mathcal{H}(\cdot)$	Continuous measurement function of the state vector $\mathbf{z}$		$\mathbb{R}^{n_z} \to \mathbb{R}^{n_y}$
$\mathcal{H}_{k}(\cdot)$	Discrete measurement function of the state vector $\mathbf{z}$		$\mathbb{R}^{n_z} \to \mathbb{R}^{n_y}$
$\mathcal{K}(\cdot)$	Regularization kernel used by the regularization step of a RPF		$\mathbb{R}^{n_x} \to \mathbb{R}^{n_x}$
$\mathcal{K}_{\bar{X}}(\cdot)$	cumulant generating function		
$\mathcal{R}$	Trun radius of the UAV	m	$[\mathcal{R}_{min},\, +\infty]$
$\mathcal{R}_{min}$	Minimum turn radius of the UAV	m	$\mathbb{R}^{+}$
$m^{(0)}$	Nominal mode of the system		
$m^{(1)}$	Faulty mode of the system		
$m^{(i)}$	The i^{th} mode of a system		
$\mathcal{N}\left(\mu,\, \sigma^2\right)$	Gaussian law of mean μ and standard deviation σ		
P_{fa}	False alarm probability		
π^{j}	Transition probability associated with the j^{th} mode of the mode vector $\mathbf{m}_k$		$\mathbb{M}$
P_{md}	Missed detection probability		
M	Constant used by $\sigma(\cdot)$		$\mathbb{R}^{+}$
m	Mass of the UAV	kg	$\mathbb{R}$
$\tilde{\mathbf{y}}$	Residual vector, also known as innovation vector		$\mathbb{R}^{n_y}$
$\tilde{\mathbf{y}}^{(i)}$	Innovation vector associated with the mode $m_k^{(i)}$		$\mathbb{R}^{n_y}$
$\tilde{w}$	Weight before being normalized		$\mathbb{R}$
υ	Random variable drawn according to a standard uniform distribution		$[0,\, 1]$

$\mathcal{U}(a,\, b)$	Uniform law with a mode defined in the interval $[a,\, b]$		
$\mathbf{B}_{\mathrm{C}}$	Control input matrix of the closed loop lateral/longitudinal control		$\mathbb{R}^{n_{zi} \times n_u}$
$\mathbf{F}_{\mathrm{C}}$	Evolution matrix of the closed loop lateral/longitudinal control		$\mathbb{R}^{n_{zi} \times n_{zi}}$
$\mathbf{H}_{\mathrm{C}}$	Output matrix of the closed loop lateral/longitudinal control		$\mathbb{R}^{n_z \times n_{zi}}$
$\mathbf{H}_{\theta}$	Row of $\mathbf{H}$ that give the observability of the state $\bar{\theta}$		$\mathbb{R}^{1 \times n_z}$
$\mathbf{H}_{\mathbf{p_d}}$	Row of $\mathbf{H}$ that give the observability of the state $\bar{p_d}$		$\mathbb{R}^{1 \times n_z}$
$\mathbf{H}_{\mathbf{u}}$	Row of $\mathbf{H}$ that give the observability of the state $\bar{u}$		$\mathbb{R}^{1 \times n_z}$
$\mathbf{B}$	Input matrix		$\mathbb{R}^{n_x \times n_u}$
$\mathbf{D}_k$	Real lower triangular matrix of the covariance matrix $\mathbf{P}_k$		$\mathbb{R}^{n_x \times n_x}$
$\mathbf{f_a}$	Actuator fault state vector		$\mathbb{R}^{n_a}$
$\mathbf{f_s}$	Sensor fault state vector		$\mathbb{R}^{n_s}$
$\mathbf{F}$	State matrix		$\mathbb{R}^{n_x \times n_x}$
$\mathbf{f}$	Fault state vector		$\mathbb{R}^{n_f}$
$\mathbf{f}_k^{i,j}$	The j^{th} state of the i^{th} particle of state vector $\mathbf{f}_k$		$\mathbb{R}$
$\mathbf{H}_{\mathbf{z^j}}$	Row of the $\mathbf{H}$ matrix corresponding to the observability of the state $\mathbf{z}^j$		$\mathbb{R}^{n_i \times n_z}$
$\mathbf{H}$	Output matrix		$\mathbb{R}^{n_y \times n_x}$
$\mathbf{h}$	Angular momentum vector	$\mathrm{kg\,m^2\,s^{-1}}$	$\mathbb{R}^3$
$\mathbf{h}^b$	Angular momentum vector in F^b	$\mathrm{kg\,m^2\,s^{-1}}$	$\mathbb{R}^3$
$\mathbf{H}_{m^{(0)}}$	Output matrix of the process model for the mode $m^{(0)}$		$\mathbb{R}^{n_y \times n_x}$
$\mathbf{H}_{m^{(1)}}$	Output matrix of the process model for the mode $m^{(1)}$		$\mathbb{R}^{n_y \times n_x}$
$\mathbf{J}$	inertia matrix	$\mathrm{kg\,m^2}$	$\mathbb{R}^{3 \times 3}$
$\mathbf{K}_k$	Kalman gain		$\mathbb{R}^{n_x \times n_y}$
$\mathbf{l_c}$	Gain of the desired output $\mathbf{y^c}$		$\mathbb{R}^{n_u \times n_u}$
$\mathbf{L_z}$	Gain of the full state feedback with integrator effect associated with the state vector $\mathbf{z}$		$\mathbb{R}^{n_u \times n_z}$

$\mathbf{L}_{\theta_i}$	LQR gain of the longitudinal control associated with the state θ_i	$\mathbb{R}^{1\times 2}$
$\mathbf{L}_{\theta}$	LQR gain of the longitudinal control associated with the state θ	$\mathbb{R}^{1\times 5}$
$\mathbf{L}_{\mathbf{p_d}}$	LQR gain of the longitudinal guidance associated with the state p_d	$\mathbb{R}^{1\times 6}$
$\mathbf{L}_{\mathbf{u_i}}$	LQR gain of the longitudinal control associated with the state u_i	$\mathbb{R}^{1\times 2}$
$\mathbf{L}_{\mathbf{u}}$	LQR gain of the longitudinal control associated with the state u	$\mathbb{R}^{1\times 5}$
$\mathbf{L}_{\mathbf{z_i}}$	Gain of the full state feedback with integrator effect associated with the state vector $\mathbf{z_i}$	$\mathbb{R}^{n_u\times n_i}$
$\mathbf{L}$	Gain of the full state feedback	$\mathbb{R}^{n_u\times n_z}$
$\mathbf{m_a}$	Mode vector associated with the state vector $\mathbf{f_a}$	$\mathbb{R}^{n_a}$
$\mathbf{m_s}$	Mode vector associated with the state vector $\mathbf{f_s}$	
$\mathbf{m}$	Mode vector associated with the state vector $\mathbf{f}$	
$\mathbf{m}_k^{(0)}$	All the state of the mode vector $\mathbf{m}_k$ are in $m^{(0)}$	
$\mathbf{m}_k^{i,j}$	The j^{th} mode of the i^{th} particle of the mode vector $\mathbf{m}_k$	$\mathbb{M}$
$\mathbf{M}_{1:k}$	Vector of k modes	$\mathbb{R}^{k}$
$\mathbf{N}$	Cross product matrix of the LQR	$\mathbb{R}^{n_u\times n_u}$
$\mathbf{P_{xy}}$	Cross covariance matrix	$\mathbb{R}^{n_x\times n_y}$
$\mathbf{Q}$	Matrix of the weight attached to tracking performance of the LQR	$\mathbb{R}^{n_z\times n_z}$
$\mathbf{Q}_k$	Covariance of the process noise	$\mathbb{R}^{n_x\times n_x}$
$\mathbf{R}$	Matrix of the weight attached to the control effort of the LQR	$\mathbb{R}^{n_u\times n_u}$
$\mathbf{R}_b^v$	Rotation matrix from F^b to F^v	$\mathbb{R}^{3\times 3}$
$\mathbf{R}_i^b$	Rotation matrix of a rotation from F^i to F^b	$\mathbb{R}^{3\times 3}$
$\mathbf{R}_k$	Covariance of the measurements noise	$\mathbb{R}^{n_y\times n_y}$
$\mathbf{S}$	Covariance matrix of the innovation	$\mathbb{R}^{n_y\times n_y}$
$\mathbf{S}$	Solution to the Riccati equation	$\mathbb{R}^{n_z\times n_z}$
$\mathbf{u}$	control input vector	$\mathbb{R}^{n_u}$

$\mathbf{U}_{lon}$	Transformation matrix between the full control input vector $\mathbf{u}$ to the longitudinal control input vector $\mathbf{u}_{lon}$		$\mathbb{R}^{2\times n_u}$
$\mathbf{u}_{lon}$	Longitudinal control input of the UAV		$\mathbb{R}^2$
$\mathbf{V_a}$	Airspeed vector	$\mathrm{m\,s^{-1}}$	$\mathbb{R}^3$
$\mathbf{V_g}$	Velocity vector	$\mathrm{m\,s^{-1}}$	$\mathbb{R}^3$
$\mathbf{V_w}$	Wind vector	$\mathrm{m\,s^{-1}}$	$\mathbb{R}^3$
$\mathbf{x}$	Extended state vector		$\mathbb{R}^{n_x}$
$\mathbf{x}_k^{(i)}$	State vector associated with the mode $m_k^{(i)}$		$\mathbb{R}^{n_x}$
$\mathbf{y^c}$	Desired output		$\mathbb{R}^{n_u}$
$\mathbf{y}$	Measurement vector		$\mathbb{R}^{n_y}$
$\mathbf{Y}_{1:k}$	Vector of k measurement vector		$\mathbb{R}^{n_y\times k}$
$\mathbf{Y}_{lon}$	Transformation matrix between the full measurement vector to the longitudinal measurement vector of the UAV		$\mathbb{R}^{8\times n_y}$
$\mathbf{y}_{lon}$	Longitudinal measurement vector of the UAV		$\mathbb{R}^8$
$\mathbf{z_i}$	Integrated error state vector		$\mathbb{R}^{n_i}$
$\mathbf{z}$	State vector		$\mathbb{R}^{n_z}$
$\mathbf{z}^j$	j^{th} state of the state vector $\mathbf{z}$		$\mathbb{R}^{n_i}$
$\mathbf{Z}_{lon}$	Transformation matrix between the full state vector $\mathbf{z}$ to the longitudinal state vector $\mathbf{z}_{lon}$		$\mathbb{R}^{5\times n_z}$
$\mathbf{z}_{lon}$	Longitudinal state vector of the UAV		$\mathbb{R}^5$
A_u	Gain associated used to compute linear approximation of the angle of attack		$\mathbb{R}$
A_w	Gain associated used to compute linear approximation of the angle of attack		$\mathbb{R}$
AR	Wing aspect ratio		$\mathbb{R}$
b	Wing span	m	$\mathbb{R}^+$
c	Mean aerodynamic chord of the wing	m	$\mathbb{R}^+$
$C_X(\cdot)$	Aerodynamic force coefficient along the i^b axis		$\mathbb{R}$
C_{D_p}	Aerodynamic coefficient of drag due to roll rate		$\mathbb{R}$
C_{D_q}	Aerodynamic coefficient of drag due to pitch rate		$\mathbb{R}$
$C_{D_{\delta_e}}$	Aerodynamic coefficient of drag due to elevator deflection		$\mathbb{R}$
$C_D(\cdot)$	Aerodynamic drag coefficient		$\mathbb{R}$

C_{L_0}	Constant aerodynamic lift coefficient	$\mathbb{R}$
C_{l_0}	Constant aerodynamic coefficient of rolling moment	$\mathbb{R}$
C_{L_α}	Aerodynamic coefficient of lift due to angle of attack	$\mathbb{R}$
C_{l_β}	Aerodynamic coefficient of rolling moment due to side-slip	$\mathbb{R}$
C_{l_p}	Roll damping coefficient	$\mathbb{R}$
C_{L_q}	Aerodynamic coefficient of lift due to pitch rate	$\mathbb{R}$
C_{l_r}	Aerodynamic coefficient of rolling moment due to yaw rate	$\mathbb{R}$
$C_{l_{\delta_a}}$	Aerodynamic coefficient of rolling moment due to aileron deflection	$\mathbb{R}$
$C_{L_{\delta_e}}$	Aerodynamic coefficient of lift due to elevator deflection	$\mathbb{R}$
$C_{l_{\delta_r}}$	Aerodynamic coefficient of rolling moment due to rudder deflection	$\mathbb{R}$
$C_L(\cdot)$	Aerodynamic lift coefficient	$\mathbb{R}$
C_{m_0}	Constant aerodynamic coefficient of pitching moment	$\mathbb{R}^+$
C_{m_α}	Aerodynamic coefficient of pitching moment due to angle of attack	$\mathbb{R}$
C_{m_q}	Aerodynamic coefficient of pitching moment due to pitch rate	$\mathbb{R}$
$C_{m_{\delta_e}}$	Aerodynamic coefficient of pitching moment due to elevator deflection	$\mathbb{R}$
C_{n_0}	Constant aerodynamic coefficient of yawing moment	$\mathbb{R}$
C_{n_β}	Aerodynamic coefficient of yawing moment due to side-slip	$\mathbb{R}$
C_{n_r}	Aerodynamic coefficient of yawing moment due to yaw rate	$\mathbb{R}$
c_{n_x}	Volume of the unit hypersphere in $\mathbb{R}^{n_x}$	$\mathbb{R}$
$C_{n_{\delta_a}}$	Aerodynamic coefficient of yawing moment aileron deflection	$\mathbb{R}$
$C_{n_{\delta_r}}$	Aerodynamic coefficient of yawing moment due to rudder deflection	$\mathbb{R}$

C_{prop}	Aerodynamic coefficient of the propeller		$\mathbb{R}$
$C_{X_q}(\cdot)$	Aerodynamic force coefficient along the i^b axis		$\mathbb{R}$
$C_{X_{\delta_e}}(\cdot)$	Aerodynamic force coefficient along the i^b axis		$\mathbb{R}$
C_{Y_0}	Aerodynamic force coefficient along the j^b axis		$\mathbb{R}$
C_{Y_β}	Aerodynamic force coefficient along the j^b axis		$\mathbb{R}$
C_{Y_p}	Aerodynamic force coefficient along the j^b axis		$\mathbb{R}$
C_{Y_r}	Aerodynamic force coefficient along the j^b axis		$\mathbb{R}$
$C_{Y_{\delta_a}}$	Aerodynamic force coefficient along the j^b axis		$\mathbb{R}$
$C_{Y_{\delta_r}}$	Aerodynamic force coefficient along the j^b axis		$\mathbb{R}$
$C_{Z_q}(\cdot)$	Aerodynamic force coefficient along the k^b axis		$\mathbb{R}$
$C_{Z_{\delta_e}}(\cdot)$	Aerodynamic force coefficient along the k^b axis		$\mathbb{R}$
$C_Z(\cdot)$	Aerodynamic force coefficient along the k^b axis		$\mathbb{R}$
e	Oswald efficiency factor		$[0.8, 1.0]$
F^b	Body frame		
F^i	Inertial frame		
F^s	Stability frame		
F^v	Vehicle frame		
F^w	Wind frame		
F^{v1}	Vehicle 1		
F^{v2}	Vehicle 2		
f_b	State of the barometer fault in altitude	m	$\mathbb{R}$
F_d	Drag force	$\mathrm{kg\,m\,s^2}$	$\mathbb{R}$
F_g	Gravity force	$\mathrm{kg\,m\,s^2}$	$\mathbb{R}$
f_g	State of the GNSS receiver fault in altitude	m	$\mathbb{R}$
$f_k(\cdot)$	Discrete dynamics of the extended state vector $\mathbf{x}$		$\mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_x}$
F_l	Lift force	$\mathrm{kg\,m\,s^2}$	$\mathbb{R}$
F_p	Propulsion force	$\mathrm{kg\,m\,s^2}$	$\mathbb{R}$
f_x	Externally applied forces on the UAV along the i^b axis	$\mathrm{kg\,m\,s^{-2}}$	$\mathbb{R}$
f_y	Externally applied forces on the UAV along the j^b axis	$\mathrm{kg\,m\,s^{-2}}$	$\mathbb{R}$
f_z	Externally applied forces on the UAV along the k^b axis	$\mathrm{kg\,m\,s^{-2}}$	$\mathbb{R}$
F_{x^b}	Components about i^b axis of the lift and drag forces	$\mathrm{kg\,m\,s^2}$	$\mathbb{R}$

F_{z^v}	Components about k^v axis of the lift and drag forces	$\mathrm{kg\,m\,s^2}$	$\mathbb{R}$
g	Gravitational acceleration	$\mathrm{m\,s^{-2}}$	$\mathbb{R}^+$
g_n	Standard acceleration of gravity	$\mathrm{m\,s^{-2}}$	$\mathbb{R}$
h	Bandwidth factor of the kernel $\mathcal{K}(\cdot)$		$\mathbb{R}^{+*}$
$h_k(\cdot)$	Discrete measurement function of the extended state vector $\mathbf{x}$		$\mathbb{R}^{n_x} \rightarrow \mathbb{R}^{n_y}$
J	Youden's index		
$J(\cdot)$	Cost function of the LQR		$\mathbb{R}$
k_ω	Motor constants of the UAV		$\mathbb{R}$
k_{GNSS}	Frequency response of the Gauss-Markov process of the GNSS error model	Hz	$\mathbb{R}^+$
k_{motor}	Constant that specify the efficiency of the motor		$\mathbb{R}^+$
k_{T_p}	Motor constants of the UAV		$\mathbb{R}$
l	Externally applied moments on the UAV along the i^b axis	$\mathrm{kg\,m\,s^{-1}}$	$\mathbb{R}$
L_0	Lapse rate of the temperature decrease in the lower atmosphere	$\mathrm{K\,m^{-1}}$	$\mathbb{R}$
L_w	Gain of the anti-wind-up system of the longitudinal guidance		$\mathbb{R}$
$L_{p_{di}}$	LQR gain of the longitudinal control associated with the state p_{di}		$\mathbb{R}$
M	Standard molar mass of the atmospheric air	$\mathrm{kg\,mol^{-1}}$	$\mathbb{R}^+$
M	Sum of all externally applied moments	$\mathrm{kg\,m^2\,s^{-2}}$	$\mathbb{R}$
m	Externally applied moments on the UAV along the j^b axis	$\mathrm{kg\,m\,s^{-1}}$	$\mathbb{R}$
m	Mode of the system		$\mathbb{M}$
M^b	Sum of all externally applied moments in F^b	$\mathrm{kg\,m^2\,s^{-2}}$	$\mathbb{R}$
$m_k^{(i)}$	$m_k = m^{(i)}$		$\mathbb{M}$
M_q	Coefficient of the longitudinal state space model of the UAV		$\mathbb{R}$
M_u	Coefficient of the longitudinal state space model of the UAV		$\mathbb{R}$
M_w	Coefficient of the longitudinal state space model of the UAV		$\mathbb{R}$

M_{δ_e}	Coefficient of the longitudinal state space model of the UAV		$\mathbb{R}$
N	Number of particles		$\mathbb{N}^+$
n	Externally applied moments on the UAV along the k^b axis	$\mathrm{kg\,m\,s^{-1}}$	$\mathbb{R}$
n	Number of measurements of the saddlepoint approximation		
N_k	Total number of time step		$\mathbb{N}^+$
N_{MC}	Total number of Mote Carlo performed		$\mathbb{N}^{+*}$
P	Pressure measured by the barometer	$\mathrm{kg\,s^{-2}\,m^{-1}}$	$\mathbb{R}$
p	Roll rate measured along i^b in F^b	$\mathrm{rad\,s^{-1}}$	$\mathbb{R}$
P_0	Standard pressure at sea level	$\mathrm{kg\,s^{-2}\,m^{-1}}$	$\mathbb{R}^+$
p_d	Inertial Down position (negative of altitude) of the UAV measured along k^i in F^i	m	$\mathbb{R}$
p_e	Inertial East position of the UAV along j^i in F^i	m	$\mathbb{R}$
p_n	Inertial North position of the UAV along i^i in F^i	m	$\mathbb{R}$
p_{d0}	Initial value of the state p_d	m	$\mathbb{R}$
P_d	Detection probability		
p_{e0}	Initial value of the state p_e	m	$\mathbb{R}$
p_{n0}	Initial value of the state p_n	m	$\mathbb{R}$
q	Pitch rate measured along j^b in F^b	$\mathrm{rad\,s^{-1}}$	$\mathbb{R}$
R	Molar gas constant	$\mathrm{kg\,m^2\,s^{-2}\,mol^{-1}\,K^{-1}}$	$\mathbb{R}$
r	Yaw rate measured along k^b in F^b	$\mathrm{rad\,s^{-1}}$	$\mathbb{R}$
S_{prop}	Area of the propeller	$\mathrm{m^2}$	$\mathbb{R}^+$
T	Local temperature of the air	K	$\mathbb{R}^+$
t	Time	s	$\mathbb{R}$
T_0	Standard temperature at sea level	K	$\mathbb{R}^+$
t_{off}	Deactivation time of the fault		
T_s	Sample time of the Gauss-Markov process of the GNSS error model	s	$\mathbb{R}$
t_{on}	Activation time of the fault		
u	Body frame velocity measured along i^b in F^b	$\mathrm{m\,s^{-1}}$	$\mathbb{R}$
v	Body frame velocity measured along j^b in F^b	$\mathrm{m\,s^{-1}}$	$\mathbb{R}$
V_d	Velocity of the UAV in F^i along the Down axis	$\mathrm{m\,s^{-1}}$	$\mathbb{R}$

V_e	Velocity of the UAV in F^i along the East axis	$\mathrm{m\,s^{-1}}$	$\mathbb{R}$
V_n	Velocity of the UAV in F^i along the North axis	$\mathrm{m\,s^{-1}}$	$\mathbb{R}$
V_u	Gain associated used to compute linear approximation of the velocity vector		$\mathbb{R}$
V_w	Gain associated used to compute linear approximation of the velocity vector		$\mathbb{R}$
w	Body frame velocity measured along k^b in F^b	$\mathrm{m\,s^{-1}}$	$\mathbb{R}$
$w_k^{(i)}$	Weight of the i^{th} model		$[0,\,1]$
w_k	Particle importance weights		$\mathbb{R}^+$
X_q	Coefficient of the longitudinal state space model of the UAV		$\mathbb{R}$
X_u	Coefficient of the longitudinal state space model of the UAV		$\mathbb{R}$
X_w	Coefficient of the longitudinal state space model of the UAV		$\mathbb{R}$
X_{δ_e}	Coefficient of the longitudinal state space model of the UAV		$\mathbb{R}$
X_{δ_t}	Coefficient of the longitudinal state space model of the UAV		$\mathbb{R}$
$y_{accel,u}$	Integral of the raw measurement of the accelerometer along i^b axis	$\mathrm{m\,s^{-1}}$	$\mathbb{R}$
$y_{accel,v}$	Integral of the raw measurement of the accelerometer along j^b axis	$\mathrm{m\,s^{-1}}$	$\mathbb{R}$
$y_{accel,w}$	Integral of the raw measurement of the accelerometer along k^b axis	$\mathrm{m\,s^{-1}}$	$\mathbb{R}$
$y_{accel,x}$	Raw measurement of the accelerometer along i^b axis	$\mathrm{m\,s^{-2}}$	$\mathbb{R}$
$y_{accel,y}$	Raw measurement of the accelerometer along j^b axis	$\mathrm{m\,s^{-2}}$	$\mathbb{R}$
$y_{accel,z}$	Raw measurement of the accelerometer along k^b axis	$\mathrm{m\,s^{-2}}$	$\mathbb{R}$
$y_{baro,-p_d}$	Linear approximation of the altitude using the barometer	m	$\mathbb{R}$
$y_{baro_{nl},-p_d}$	Non-linear measurement of the altitude using the barometer	m	$\mathbb{R}$
y_{baro}	Raw measurement of the barometer	$\mathrm{kg\,s^{-2}\,m^{-1}}$	$\mathbb{R}$
y_{GNSS,p_e}	Raw measurement of the East position of the GNSS	m	$\mathbb{R}$

y_{GNSS,p_n}	Raw measurement of the North position of the GNSS	m	$\mathbb{R}$
$y_{GNSS,u}$	Raw measurement of the velocity position of the GNSS along the i^b axis	$\mathrm{m\,s^{-1}}$	$\mathbb{R}$
$y_{GNSS,v}$	Raw measurement of the velocity position of the GNSS along the j^b axis	$\mathrm{m\,s^{-1}}$	$\mathbb{R}$
$y_{GNSS,w}$	Raw measurement of the velocity position of the GNSS along the k^b axis	$\mathrm{m\,s^{-1}}$	$\mathbb{R}$
$y_{GNSS,-p_d}$	Raw measurement of the Altitude position of the GNSS	m	$\mathbb{R}$
$y_{gyro,\phi}$	Integral of the raw measurement of the rate gyro along i^b axis	$\mathrm{rad\,s^{-1}}$	$\mathbb{R}$
$y_{gyro,\psi}$	Integral of the raw measurement of the rate gyro along k^b axis	$\mathrm{rad\,s^{-1}}$	$\mathbb{R}$
$y_{gyro,\theta}$	Integral of the raw measurement of the rate gyro along j^b axis	$\mathrm{rad\,s^{-1}}$	$\mathbb{R}$
$y_{gyro,p}$	Raw measurement of the rate gyro along i^b axis	$\mathrm{rad\,s^{-1}}$	$\mathbb{R}$
$y_{gyro,q}$	Raw measurement of the rate gyro along j^b axis	$\mathrm{rad\,s^{-1}}$	$\mathbb{R}$
$y_{gyro,r}$	Raw measurement of the rate gyro along k^b axis	$\mathrm{rad\,s^{-1}}$	$\mathbb{R}$
y_{mag}	Raw measurement of the magnetometer	rad	$\mathbb{R}$
Z_q	Coefficient of the longitudinal state space model of the UAV		$\mathbb{R}$
Z_u	Coefficient of the longitudinal state space model of the UAV		$\mathbb{R}$
Z_w	Coefficient of the longitudinal state space model of the UAV		$\mathbb{R}$
Z_{δ_e}	Coefficient of the longitudinal state space model of the UAV		$\mathbb{R}$
$\mathcal{G}_{ak}(\cdot)$	Discrete dynamics of the actuators faults		$\mathbb{R}^{n_a} \to \mathbb{R}^{n_a}$
$\mathcal{G}_{sk}(\cdot)$	Discrete dynamics of the sensors faults		$\mathbb{R}^{n_s} \to \mathbb{R}^{n_s}$
$\boldsymbol{\eta}_{k\mathbf{m}_k}$	Process noise associated with the mode vector $\mathbf{m}_k$		
$f_{k\mathbf{m}_k}(\cdot)$	Discrete dynamics of the extended state vector $\mathbf{x}$ associated with the mode vector $\mathbf{m}_k$		

$h_{k\mathbf{m}_k}(\cdot)$	Discrete measurement function of the extended state vector $\mathbf{x}$ associated with the mode vector $\mathbf{m}_k$	
$\mathcal{F}_{km_k}(\cdot)$	Discrete dynamics of the state vector $\mathbf{z}$ associated with the discrete mode m_k	$\mathbb{R}^{n_z \times n_u \times n_a} \to \mathbb{R}^{n_z}$
$\mathcal{G}_{ak\mathbf{m}_{\mathbf{a}k}}(\cdot)$	Discrete dynamics of the actuator faults associated with the mode vector $\mathbf{m}_{\mathbf{a}k}$	
$\mathcal{G}_{sk\mathbf{m}_{\mathbf{s}k}}(\cdot)$	Discrete dynamics of the sensor faults associated with the mode vector $\mathbf{m}_{\mathbf{s}k}$	
$\mathcal{H}_{km_k}(\cdot)$	Discrete measurement function of the state vector $\mathbf{z}$ associated with the discrete mode m_k	$\mathbb{R}^{n_z \times n_s} \to \mathbb{R}^{n_y}$

Greek Letters

α	Angle of attack	rad	$[-\pi, \pi]$
α	Bound of the false alarm of the Neyman-Pearson criterion		
α_0	Stalling angle of attack	rad	$[0, \frac{\pi}{2}]$
$\bar{\alpha}$	Trimmed angle of attack	rad	$[-\pi, \pi] - \alpha^*$
$\bar{\gamma_a}$	Trimmed air-mass-referenced flight path angle	rad	$[-\pi, \pi] - \gamma_a^*$
$\bar{\theta}_i$	Integrated error of the trimmed state $\bar{\theta}$	rad	$\mathbb{R}$
$\bar{\theta}^c$	Desired trimmed pitch	rad	$[-\pi, \pi] - \theta^*$
$\bar{\delta}_e$	Trimmed control input δ_e	rad	$\left[-\frac{\pi}{7}, \frac{\pi}{7}\right] - \delta_e^*$
$\bar{\delta}_t$	Trimmed control input δ_t		$[0, 1] - \delta_t^*$
$\bar{\gamma_a}^c$	Desired trimmed air-mass-referenced flight path angle	rad	$[-\pi, \pi] - \gamma_a^*$
$\bar{\theta}$	Trimmed state θ	rad	$[-\pi, \pi] - \theta^*$
β	Side-slip angle	rad	$[-\pi, \pi]$
χ	Course angle	rad	$[-\pi, \pi]$
χ_c	Crab angle	rad	$[-\pi, \pi]$
δ_a	UAV aileron deflection	rad	$\left[-\frac{\pi}{7}, \frac{\pi}{7}\right]$
δ_e	UAV elevator deflection	rad	$\left[-\frac{\pi}{7}, \frac{\pi}{7}\right]$
δ_m	Local declination between the geographical and magnetic North	rad	$\mathbb{R}$
δ_r	UAV rudder deflection	rad	$\left[-\frac{\pi}{5}, \frac{\pi}{5}\right]$
δ_t	UAV throttle input		$[0, 1]$

$\delta(\cdot)$	Dirac delta functions		
η	Threshold of the likelihood ratio		
η_{GNSS}	Zero-mean white Gaussian noise of the Gauss-Markov process of the GNSS error model	m	$\mathbb{R}$
Γ	Threshold of the false alarm and missed detection probabilities		
γ	Flight path angle	rad	$[-\pi, \pi]$
$\Gamma(\cdot)$	Euler's gamma function		$\mathbb{R} \to \mathbb{R}$
γ_a	Air-mass-referenced flight path angle	rad	$[-\pi, \pi]$
Γ_{1-8}	Products of the inertia matrix		$\mathbb{R}$
Γ_{rspl}	Threshold of the effective number of particle for performing the resampling step		$[0, 1]$
$\Lambda(\cdot)$	Likelihood ratio		
μ_0	Mean of the distribution under $\mathcal{H}_0$		
μ_1	Mean of the distribution under $\mathcal{H}_1$		
$\mu_{accel,x}$	Mean of Gaussian noise of the accelerometer along i^b axis	$\mathrm{m\,s^{-2}}$	$\mathbb{R}$
$\mu_{accel,y}$	Mean of Gaussian noise of the accelerometer along j^b axis	$\mathrm{m\,s^{-2}}$	$\mathbb{R}$
$\mu_{accel,z}$	Mean of Gaussian noise of the accelerometer along k^b axis	$\mathrm{m\,s^{-2}}$	$\mathbb{R}$
μ_{baro}	Mean of Gaussian noise of the barometer	$\mathrm{kg\,s^{-2}\,m^{-1}}$	$\mathbb{R}$
$\mu_{gyro,p}$	Mean of Gaussian noise of the rate gyro along i^b axis	$\mathrm{rad\,s^{-1}}$	$\mathbb{R}$
$\mu_{gyro,q}$	Mean of Gaussian noise of the rate gyro along j^b axis	$\mathrm{rad\,s^{-1}}$	$\mathbb{R}$
$\mu_{gyro,r}$	Mean of Gaussian noise of the rate gyro along k^b axis	$\mathrm{rad\,s^{-1}}$	$\mathbb{R}$
$\nu_{accel,u}$	Integral of the Gaussian noise of the accelerometer along i^b axis	$\mathrm{m\,s^{-1}}$	$\mathbb{R}$
$\nu_{accel,v}$	Integral of the Gaussian noise of the accelerometer along j^b axis	$\mathrm{m\,s^{-1}}$	$\mathbb{R}$
$\nu_{accel,w}$	Integral of the Gaussian noise of the accelerometer along k^b axis	$\mathrm{m\,s^{-1}}$	$\mathbb{R}$
$\nu_{accel,x}$	Gaussian noise of the accelerometer along i^b axis	$\mathrm{m\,s^{-2}}$	$\mathbb{R}$

$\nu_{accel,y}$	Gaussian noise of the accelerometer along j^b axis	$\mathrm{m\,s^{-2}}$	$\mathbb{R}$
$\nu_{accel,z}$	Gaussian noise of the accelerometer along k^b axis	$\mathrm{m\,s^{-2}}$	$\mathbb{R}$
$\nu_{baro,-p_d}$	Gaussian noise of the barometer in the altitude state space	m	$\mathbb{R}$
ν_{baro}	Gaussian noise of the barometer	$\mathrm{kg\,s^{-2}\,m^{-1}}$	$\mathbb{R}$
ν_{GNSS,p_e}	Error model of the East position of the GNSS	m	$\mathbb{R}$
ν_{GNSS,p_n}	Error model of the North position of the GNSS	m	$\mathbb{R}$
$\nu_{GNSS,u}$	Zero mean white Gaussian noise of the velocity using GNSS along the i^b axis	$\mathrm{m\,s^{-1}}$	$\mathbb{R}$
$\nu_{GNSS,v}$	Zero mean white Gaussian noise of the velocity using GNSS along the j^b axis	$\mathrm{m\,s^{-1}}$	$\mathbb{R}$
$\nu_{GNSS,w}$	Zero mean white Gaussian noise of the velocity using GNSS along the k^b axis	$\mathrm{m\,s^{-1}}$	$\mathbb{R}$
$\nu_{GNSS,-p_d}$	Error model of the Altitude position of the GNSS	m	$\mathbb{R}$
$\nu_{gyro,\phi}$	Integral of the Gaussian noise of the rate gyro along i^b axis	rad	$\mathbb{R}$
$\nu_{gyro,\psi}$	Integral of the Gaussian noise of the rate gyro along k^b axis	rad	$\mathbb{R}$
$\nu_{gyro,\theta}$	Integral of the Gaussian noise of the rate gyro along j^b axis	rad	$\mathbb{R}$
$\nu_{gyro,p}$	Gaussian noise of the rate gyro along i^b axis	$\mathrm{rad\,s^{-1}}$	$\mathbb{R}$
$\nu_{gyro,q}$	Gaussian noise of the rate gyro along j^b axis	$\mathrm{rad\,s^{-1}}$	$\mathbb{R}$
$\nu_{gyro,r}$	Gaussian noise of the rate gyro along k^b axis	$\mathrm{rad\,s^{-1}}$	$\mathbb{R}$
ν_{mag}	Gaussian noise of the magnetometer	rad	$\mathbb{R}$
$\omega_{b/i}$	Angular velocity of frame F^b with respect to F^i	$\mathrm{rad\,s^{-1}}$	$\mathbb{R}$
ϕ	Roll angle defined with respect to F^{v2}	rad	$[-\pi,\,\pi]$
$\Phi\left(\cdot\right)$	Cumulative distribution function		
$\phi\left(\cdot\right)$	Standard Gaussian distribution		
π_{ij}	Transition probability of switching from a mode j to i		
ψ	Heading (yaw) angle defined with respect to F^v	rad	$[-\pi,\,\pi]$
ψ_m	Heading relative to the magnetic North	rad	$\mathbb{R}$

ρ	Density of air	$\mathrm{kg\,m^{-3}}$	$\mathbb{R}^+$
$\sigma(\cdot)$	Sigmoid function		$\mathbb{R}$
σ_0	Standard deviation of the distribution under $\mathcal{H}_0$		
σ_1	Standard deviation of the distribution under $\mathcal{H}_1$		
$\sigma_{accel,x}$	Standard deviation of Gaussian noise of the accelerometer along i^b axis	$\mathrm{m\,s^{-2}}$	$\mathbb{R}$
$\sigma_{accel,y}$	Standard deviation of Gaussian noise of the accelerometer along j^b axis	$\mathrm{m\,s^{-2}}$	$\mathbb{R}$
$\sigma_{accel,z}$	Standard deviation of Gaussian noise of the accelerometer along k^b axis	$\mathrm{m\,s^{-2}}$	$\mathbb{R}$
σ_{baro}	Standard deviation of Gaussian noise of the barometer	$\mathrm{kg\,s^{-2}\,m^{-1}}$	$\mathbb{R}$
$\sigma_{GNSS,-p_d}$	Standard deviation of the Gaussian noise of the GNSS error in Down position	m	$\mathbb{R}$
σ_{GNSS,p_e}	Standard deviation of the Gaussian noise of the GNSS error in East position	m	$\mathbb{R}$
σ_{GNSS,p_n}	Standard deviation of the Gaussian noise of the GNSS error in North position	m	$\mathbb{R}$
$\sigma_{GNSS,u}$	Standard deviation of the Gaussian noise of the velocity using GNSS along the i^b axis	$\mathrm{m\,s^{-1}}$	$\mathbb{R}$
$\sigma_{GNSS,v}$	Standard deviation of the Gaussian noise of the velocity using GNSS along the j^b axis	$\mathrm{m\,s^{-1}}$	$\mathbb{R}$
$\sigma_{GNSS,w}$	Standard deviation of the Gaussian noise of the velocity using GNSS along the k^b axis	$\mathrm{m\,s^{-1}}$	$\mathbb{R}$
σ_{GNSS}	Standard deviation of the Gaussian noise of the Gauss-Markov process of the GNSS error model	m	$\mathbb{R}$
$\sigma_{gyro,p}$	Standard deviation of Gaussian noise of the rate gyro along i^b axis	$\mathrm{rad\,s^{-1}}$	$\mathbb{R}$
$\sigma_{gyro,q}$	Standard deviation of Gaussian noise of the rate gyro along j^b axis	$\mathrm{rad\,s^{-1}}$	$\mathbb{R}$
$\sigma_{gyro,r}$	Standard deviation of Gaussian noise of the rate gyro along k^b axis	$\mathrm{rad\,s^{-1}}$	$\mathbb{R}$
θ	Pitch angle defined with respect to F^{v1}	rad	$[-\pi, \pi]$
ε_k	Regularization noise		$\mathbb{R}^{n_x}$

$\boldsymbol{\eta}$	Model uncertainties, also known as process noise	$\mathbb{R}^{n_x}$
$\boldsymbol{\nu}$	Measurement noise	$\mathbb{R}^{n_y}$
$\boldsymbol{\Pi}$	Transition probability matrix	$\mathbb{R}^{M\times M}$
$\sigma_{\mathbf{f}}$	Standard deviation of the process noise associated with the state vector $\sigma_{\mathbf{f}}$	$\mathbb{R}^{n_f}$
$\sigma_{\mathbf{z}}$	Standard deviation of the process noise associated with the state vector $\mathbf{z}$	$\mathbb{R}^{n_z}$
$\phi_{fa}(\cdot)$	Activation functions of the false alarm probability of the adaptive transition probability matrix	
$\phi_{md}(\cdot)$	Activation functions of the missed detection probability of the adaptive transition probability matrix	

Superscripts

$*$	Equilibrium point	
$1:N$	Sample from 1 to N	
MC	The MC^{th} Mote Carlo performed	$\mathbb{N}^{+*}$
$\top$	Transposition of the matrix to which it is applied	
(i)	Mode number	$\mathbb{N}^{0:M-1}$

Subscripts

$1:k$	Time step 1 to k
k	Current time step

1

GENERAL INTRODUCTION

Unmanned aerial vehicle (UAV) systems have been one of the aviation's most rapidly evolving fields in recent years, but their wider use will depend on safely increasing their autonomy. The ability to recover from actuator and sensor faults is crucial in this context.

Fault detection, isolation and recovery systems are increasingly employed in aircraft to provide alternate flight modes with adequate flight envelope protections using hardware or analytical redundancies in large aircraft. The level of redundancy is however limited in small aircraft. Fault detection malfunctions in aircraft sensing continue to be linked to loss of control in flight, often leading to serious or even catastrophic incidents such as the two recent Boeing 737 MAX crashes where a stall was erroneously detected by a faulty angle of attack sensor, which triggered the automatic anti stall system, forcing the nose of the aircraft down multiple times until it eventually crashed [1]. Inertial navigation sensor failures have also led to the crashes of the Qantas F72 and Croatia Boeing 737–200 [2]. UAV systems also face these issues. Approximately 40 % of Predator drones have indeed crashed in Class A (the highest severity) accidents and the United States Air Force (USAF) acknowledged that Predator UAVs crash more frequently than regular military aircraft, which highlights an even higher need for fault tolerance in autonomous aircraft [3]. This is also true in small UAV due to limited redundancy and lower cost sensing. UAVs legislation bodies increasingly require any applications for certification of autonomous systems to cover data integrity, including sensor data and fault flags [4]. Small autonomous UAVs will therefore increasingly be required to employ fault detection, isolation and recovery systems for multiple sensor fault modes. The architecture of UAVs with fault diagnosis and recovery modules is illustrated in Figure 1.1.

In Figure 1.1, the process faults — faults that directly occur in the UAV module — are illustrated, but this thesis focuses on actuator and sensor faults.

Fault detection and isolation (see Section 2.2) are well mastered nowadays, but fault estimation must be considered to make a fault recovery possible. Fault estimation methods are developed in this thesis for a low to medium endurance fixed wing UAV, without actuator redundancy. Estimation filters were investigated for this purpose, including Kalman Filters that have been adapted to a variety of joint state and fault estimation problems for linear Gaussian systems (see Section 2.5.4). An efficient extension of the approach to multimode systems is the use of banks of Kalman filters in a interacting multiple model (IMM) architecture (see Section 2.5.7). Kalman filters were extended to nonlinear systems using local linearization in the extended Kalman filter (EKF) (see Section 2.5.5) Another extension of

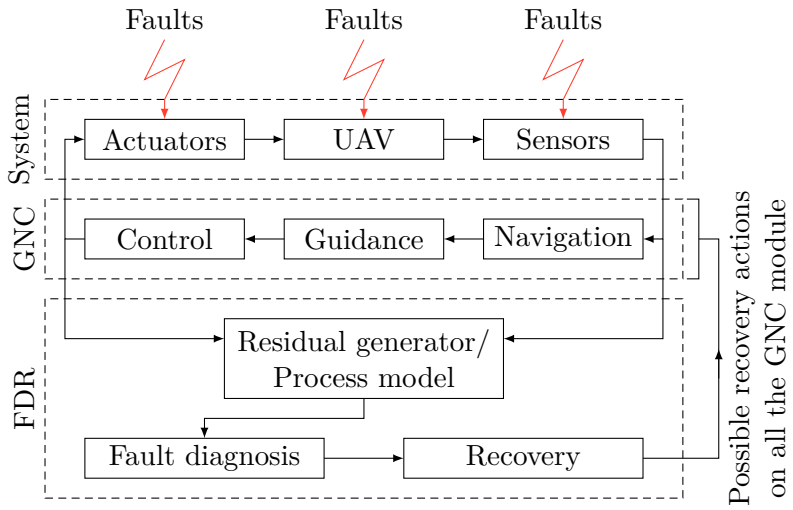


Figure 1.1: Architecture of the UAV with fault diagnosis and recovery

Kalman filtering to nonlinear systems is the propagation of a deterministic cloud of particles using the unscented Kalman filter (UKF), but both the EKF and UKF are suboptimal in the nonlinear non-Gaussian case. Particle filters are known to provide a discrete approximation to the optimal state estimation problem in nonlinear and multimodal systems [5]. Particle filters (see Section 2.5.6) have been successfully implemented for aircraft navigation [6], including terrain aided navigation in the presence of nonlinear and multimodal measurements [7], and fault detection and isolation in small UAV [8]. They were also developed for multiple model architectures but with the computational cost of testing both hypotheses using large numbers of particles at all times [9]. A jump Markov particle filter was introduced to represent the mode transitions using a Markovian jump process, where faulty and fault-free situations are modelled as discrete states of the system by Doucet, Gordon, and Krishnamurthy [10] and Tafazoli and Sun [11]. However, all these methods are either not suitable for some types of faults, embedded applications such as the UAVs considered in this thesis, or suffer from some limitations like making assumptions about the potential faults, or having knowledge of the fault magnitude or dynamics. All these identified limitations lead to the following list of questions to which this thesis aims to provide complete or at least partial answers.

1.1 RESEARCH QUESTIONS

1. Does a jump-Markov system (JMS) with a regularized particle filter (RPF) increase accuracy and speed up convergence when abrupt additive faults occur, compared to a stand-alone RPF?

   The JMS is used to represent the change in dynamics of a system and more details on this are given in Section 2.5.7. A RPF is an estimation filter that can be used for state and fault estimation and a detailed description of this filter is given in Section 2.5.

2. Is it possible to distinguish and estimate ambiguous sensor faults using only a JMS as process model of a particle filter?

   In this thesis, ambiguous sensor faults are defined as faults occurring on measurements provided by sensors that measure the same state. The measurement equations are then differentiated only by their noises, which are assumed to be independent and have different characteristics. When two sensors are used to measure the same state and faults occur, sensor fault isolation and estimation is not trivial. More details on ambiguous sensor faults are given in Section 4.1.

3. Can an abrupt additive fault, with a large amplitude with respect to the process noise, be accurately estimated in a short time period?

   Process noise is used to model the uncertainties of the system, especially in the Bayesian approach used in this thesis. Large process noise then usually means that the dynamics of the system is not well known, and in the case of abrupt faults, high process noise can be useful to account for abrupt changes in the dynamics, which may correspond to a fault. However, increasing the process noise leads to a less accurate estimate. Thus, the issue is to be able to estimate a fault with a large amplitude without degrading the fault estimate. The effect of process noise when an abrupt change occurs is explored in Section 4.2.

4. Can faults with different dynamics than the ones used in the process model be estimated accurately?

   The model-based fault estimation approach used in this thesis uses the dynamics of the fault and the dynamics of the UAV to be able to estimate the fault. Since the dynamics is the mathematical description of the behaviour of the system, it provides an analytical redundancy. A more accurate approximation of the fault dynamics tends to provide a better estimate of the faults. However, detailed knowledge of the fault dynamics is not always possible, and an approximation is often used. Therefore, in this thesis, a thorough study is performed to be able to estimate a fault despite the use of an approximation of the fault dynamics, which would not usually provide accurate estimates. The effects of having a different dynamics as a process model than the one occurring is described in Section 5.2.

5. In the case of ambiguous actuator and sensor faults, is it possible to distinguish and estimate them?

   An ambiguous actuator and sensor fault is defined in this thesis as an actuator fault that has a similar impact on the measurements as a sensor fault. This type of fault is simulated in this thesis by a fault on the elevator deflection. This fault has a direct impact on the pitch rate measurement. It is therefore difficult to distinguish between a faulty pitch rate measurement and a faulty elevator deflection. Ambiguous actuator and sensor faults are detailed in Section 5.2.

6. Can the false alarm and missed detection probability be computed in real time so that the transition probability matrix of the JMS process model can be adjusted?

   The JMS as is it described in Section 2.5.7 uses a probability transition matrix to switch between modes. These modes represent faulty and fault-free situations. The transition probabilities between modes are linked to the false alarm and missed detection probability. The idea is then to study the applicability of an adaptive transition probability matrix, computed with the current false alarm and missed detections probabilities.

7. Can the proposed solution for the previous questions be used for real-time embedded applications?

   The concern here is to be sure that the proposed solutions are suitable for an embedded application such as UAV, navigation and control, and proceed in real-time with the current data to provide a fault diagnosis that could limit the impact of faults on a mission.

1.2 CONTRIBUTIONS TO KNOWLEDGE

The work described in this thesis that contributes to current knowledge is the following:

- A state estimation filter based on a RPF and on JMS, called the jump-Markov regularized particle filter (JMRPF);
- An improved particle placement for particle filter regarding the likelihood using a Kalman update to solve ambiguity between actuator and sensor fault and to improve the robustness of the filters;
- An enhanced probability transition matrix of the JMS for fault estimation to explore the alternate mode to the current hypothesis to reduce computational demand, using an analytical expression based on a generalization of the saddlepoint approximation to independent but non identically distributed measurements, with application to the online computation of false alarm and missed detection probabilities.

1.3 PUBLICATIONS

[1] **Enzo Iglésis**, Karim Dahia, Hélène Piet-Lahanier, Nicolas Merlinge, Nadjim Horri, and James Brusey. “A Jump-Markov Regularized Particle Filter for the estimation of ambiguous sensor faults”. In: vol. 53. 2. 21th IFAC World Congress. 2020, pp. 756–763.

[2] **Enzo Iglésis**, Nadjim Horri, Karim Dahia, James Brusey, and Hélène Piet-Lahanier. “Nonlinear Estimation of Sensor Faults With Unknown Dynamics for a Fixed Wing Unmanned Aerial Vehicle”. In: *2021 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE. 2021, pp. 404–412.

[3] **Enzo Iglésis**, Nadjim Horri, James Brusey, Karim Dahia, and Hélène Piet-Lahanier. "Simultaneous Actuator and Sensor Faults Estimation for Aircraft Using a Jump-Markov Regularized Particle Filter". In: *2021 IEEE International Conference on Prognostics and Health Management (ICPHM)*. IEEE. 2021, pp. 1–10.

1.4 THESIS STRUCTURE

The thesis is organized as follows:

Chapter 2 presents a literature review on state fault diagnosis, with an in depth review of fault estimation, and in particular, including Bayesian estimation theory, particle filtering and hybrid state estimation using IMM architectures.

Chapter 3 details the application under consideration in this thesis: a fixed-wing UAV. The non-linear model, guidance, control systems are presented, as well as the UAV sensors.

Chapter 4 introduces a new approach named the JMRPF for the estimation of actuator and sensor faults as well as ambiguous sensor faults, by combining a new jump Markov strategy with regularized particle filtering. The principle, mathematical formulation and algorithms used to implement this method are presented, followed by a numerical simulation analysis that includes a comparison with the IMM for a linear jump Markov model with ambiguous sensors faults.

Chapter 5 presents an enhanced version of the JMRPF named the robustified jump-Markov regularized particle filter (RJMRPF) for the estimation of faults with unknown dynamics and amplitudes and to further improve the estimation of ambiguous actuator and sensor faults. The mathematical formulation of this filter where a Kalman correction is incorporated, are detailed in this chapter and followed by a numerical simulation analysis. The RJMRPF is compared to the RPF, JMRPF and a robustified RPF in terms of fault and state estimation accuracy.

Chapter 6 tackles the issue of the sub-optimality of the transition probability matrix of the JMS used in the JMRPF and RJMRPF, by updating this matrix in real time using the false alarm and missed detection probabilities, which are both computed using a Lugannani and Rice formula. A formulation of the transition probability matrix as well as the method and the new algorithm named the adaptive jump-Markov regularized particle filter (AJMRPF) — or adaptive robustified jump-Markov regularized particle filter (ARJMRPF) depending on which algorithm is used with the transition probability matrix update — is detailed, with a numerical simulation analysis. The AJMRPF is compared to the JMRPF in terms of fault and state estimation accuracy. The false alarm and missed detection probabilities are also simulated in the case of the AJMRPF.

Chapter 7 concludes the thesis and discusses possible directions for future work.

2

FAULT DIAGNOSIS AND ESTIMATION METHODS

This chapter provides the background for the work presented in the subsequent chapters of this thesis and is organized as follows: Section 2.1 detail the different type of faults and how to describe them in a system. The Section 2.2 define fault diagnosis. In Section 2.3, the model-based fault diagnosis is detail. Section 2.4 detail the common issues of model based fault diagnosis as well as the performance metrics. The Section 2.5 takes a look at the model based fault estimation. Section 2.6 summarize this chapter.

Without loss of generality, all dynamical models used in this thesis are time-invariant, which is not unusual in the mathematical modelling of fixed wing UAV dynamics. It is also assumed that the measurement function does not explicitly depend on control inputs, which is also a common assumption.

In this chapter, the following considerations are used: The evolution of a dynamic system can be represented using state variables, such as position, velocity, and their temporal dependencies. The state vector contains all the state variables and its dimension is the dimension of the system. Some aspects of the state of the system can be measured. The set of possible measurements can be represented as a vector space, and it is referred to a vector in this space as a measurement vector. The measurement vector contains all measurements available, these measurements are functions of the state variables.

To represent the dynamic system studied in this thesis, let us consider a state vector $\mathbf{z} \in \mathbb{R}^{n_z}$ and a measurement vector $\mathbf{y} \in \mathbb{R}^{n_y}$ described by the following discrete time system,

$$\begin{cases} \mathbf{z}_k = \mathcal{F}_k\left(\mathbf{z}_{k-1},\, \mathbf{u}_k\right) & \text{(2.1a)} \\ \mathbf{y}_k = \mathcal{H}_k\left(\mathbf{z}_k\right) & \text{(2.1b)} \end{cases}$$

where $\mathbf{u} \in \mathbb{R}^{n_u}$ is the input vector, the subscript k and $k-1$ denote the time step, $\mathcal{F}_k(\cdot) \in \mathbb{R}^{n_z \times n_u} \to \mathbb{R}^{n_z}$ is the discrete time dynamics of the state vector and $\mathcal{H}(\cdot) \in \mathbb{R}^{n_z} \to \mathbb{R}^{n_y}$ is the measurement function.

2.1 FAULT TYPES AND MODELS

According to a definition provided by Isermann [12] a 'fault' is an "*Unpermitted deviation of at least one characteristic property of the system*" This 'unpermitted deviation' can occur at

various levels of an architecture and with multiple behaviours. This section aims to provide an overview of fault types and their potential mathematical representation.

2.1.1 *Fault types*

A classification provided by Varga [13] allows various types of faults to be distinguished. The physical classification defines where the fault occurs on the system. The occurrence classification defines how the fault occurs. The time-related classification defines the duration of the fault. Finally, the model classification defines how the fault occurs on the system.

A fault diagnosis method is characterized by the type of fault that it can diagnose.

2.1.1.1 *Physical classification*

The physical classification of faults can be associated with one of the three types proposed by Varga [13], each type being exclusive. The first type is the actuator fault. It defines a variation of one or more characteristics of an actuator that leads to a loss of efficiency or a complete breakdown. The second type is the sensor fault, which defines an acquisition of incorrect data from a sensor. The last type is the parametric fault, which consists of a modification of the plant dynamical equations. It is defined as an internal malfunction such as an unexpected shift in the centre of gravity in a UAV.

The three types of physical fault classification are illustrated in Figure 1.1.

2.1.1.2 *Occurrence classification*

The occurrence classification of faults can be associated with one of the two types proposed by Varga [13] and Isermann [14], each type being exclusive. The first type is the abrupt fault. It represents a fault that occurs suddenly. The second type is the incipient fault which represents a fault that occurs gradually.

Figure 2.1 illustrates the evolution of faults according to these two types of occurrence. Note that the dynamics of the fault can differ from the one illustrated.

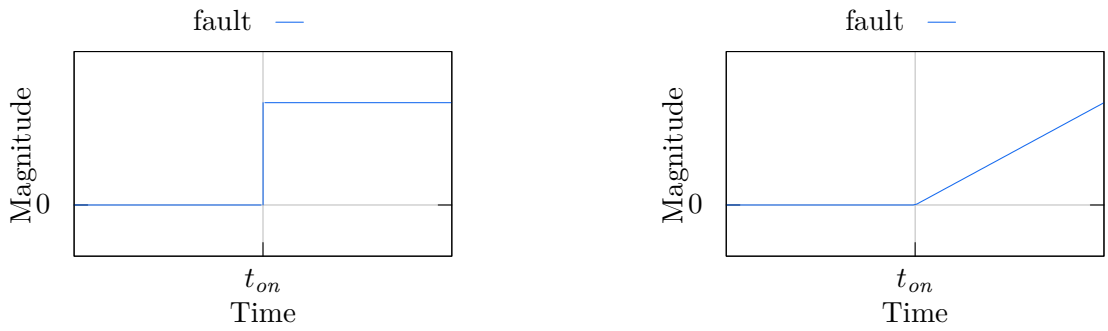
In Figure 2.1, the time t_{on} denotes the activation time of the fault.

2.1.1.3 *Time-related classification*

The time-related classification of faults can be associated with one of the two mutually exclusive types proposed by Varga [13] and Isermann [14]. The first type is the persistent fault, which describes a fault that has an effect which persists. The second type is the intermittent fault. This type of fault has an effect that lasts during a time period, then vanishes to possibly reappear later.

The time-related classification of fault is illustrated in Figure 2.2.

In Figure 2.2, the time t_{off} denotes the deactivation time of the fault. The State *off* indicates that the fault is not active while the state *on* indicates that the fault is active.



(a) Abrupt fault. (b) Incipient fault.

Figure 2.1: Occurrence classification of faults

(a) Persistent fault. (b) Intermittent fault.

Figure 2.2: Time-related classification of faults

A fault diagnosis method that can also diagnose intermittent faults can diagnose persistent faults but must also detect the deactivation of the fault.

2.1.1.4 *Model classification of faults*

The model classification of faults can be associated with one of the two types proposed by Varga [13], each type being exclusive. The additive fault can be modelled by a superposition of a signal with the original input, state or measurement signals. The multiplicative fault results in changes in the parametric representations of the process in state or measurement equations.

The model classification of fault is illustrated in Figure 2.3.

(a) Additive fault.

(b) Multiplicative fault.

Figure 2.3: Model classification of faults

2.1.2 *State representation of faults*

This section aims to represent the system with actuator and sensor faults. Then, a model with mode switch between faulty and fault-free dynamics is presented. Since "*a fault is a state within the system*" [14], the actuator and sensor faults are then denoted respectively as state vector $\mathbf{f_a} \in \mathbb{R}^{n_a}$ and $\mathbf{f_s} \in \mathbb{R}^{n_s}$. For the sake of brevity in this section, actuator and sensor faults vectors have the same dimension as the input vector and the measurement vector respectively. However, faults can also be considered on a subset of actuators or measurements.

2.1.2.1 *Actuator faults*

Actuator faults can be modelled by an extra signal on the control input vector. For sake of brevity, in this thesis it is assumed that the actuator faults directly modify the control input values [15]. This means that the actuator fault and the control input vector are in the same space. Then, for an additive fault this signal is added to the control input vector $\mathbf{u}$, and the new system is then given by:

$$\begin{cases} \mathbf{z}_k = \mathcal{F}_k\left(\mathbf{z}_{k-1},\, \mathbf{u}_k + \mathbf{f}_{\mathbf{a}k-1}\right) & \text{(2.2a)} \\ \mathbf{y}_k = \mathcal{H}_k\left(\mathbf{z}_k\right) & \text{(2.2b)} \end{cases}$$

2.1.2.2 *Sensor faults*

Sensor faults can be modelled by an extra signal in the measurement equation. For sake of brevity, in this thesis it is assumed that the sensor faults directly act on the process measurement [15, 16]. This means that the sensor faults and the measurements are in the same space. For an additive fault this signal is added to the measurement function $\mathcal{H}_k\left(\cdot\right)$, and the new system is then given by:

$$\begin{cases} \mathbf{z}_k = \mathcal{F}_k\left(\mathbf{z}_{k-1},\, \mathbf{u}_k\right) & \text{(2.3a)} \\ \mathbf{y}_k = \mathcal{H}_k\left(\mathbf{z}_k\right) + \mathbf{f}_{\mathbf{s}k} & \text{(2.3b)} \end{cases}$$

2.2 FAULT DIAGNOSIS

In the literature, 'fault diagnosis' is used to encompass multiple tasks [13, 15]. The definition given by Isermann [12] is no exception since it defines fault diagnosis as a "*Determination of kind, size, location and time of detection of a fault by evaluating symptoms. Follows fault detection. Includes fault detection, isolation and identification*"

Fault detection is the most basic task of fault diagnosis. For Isermann [12], fault detection means "*Determination of faults present in a system and time of detection*" Then, as suggested by Isermann [12], fault detection only focuses on the detection of the occurrence of a fault. Its purpose is only to provide information that a fault is active or not — in other words *on* or *off* — and consequently, provide information on the time of activation of the fault — that is t_{on}. The deactivation time — that is t_{off} — can be provided, but it is not a required feature. If the deactivation time can be provided by the fault diagnosis method, the only fault type information — according to the ones given in Section 2.1 — that is provided is the time-related classification of the fault.

In fault diagnosis the fault detection is followed by the fault isolation. In [12], fault isolation is the "*Determination of kind, location and time of detection of a fault by evaluating symptoms*". The main additional feature of fault isolation is the localization of the fault. This localization goes beyond the simple physical classification of the fault. Indeed, it does not simply determine if the fault originates from the actuator or the sensor, but also indicates which actuator or sensor is faulty.

A more challenging task than fault isolation is fault identification. According to the definition provided by Isermann [12] fault identification consists in the "*Determination of the size and time-variant behaviour of a fault*". However, the definitions of 'fault identification' vary in the literature. The definition of Isermann [12] is close to the definition of fault estimation by Varga [13], where 'fault estimation' means "*the reconstruction of the fault signal*". However, Varga [13] uses the term 'fault estimation' because 'fault identification' is defined as "*a precise classification of the detected faults and their characteristics*". Nevertheless, this thesis only addresses the problem of fault diagnosis up to the fault estimation. The fault identification as defined by Varga [13] is not discussed here. In other words, the term fault diagnosis in this thesis includes fault detection, isolation and estimation.

When there is no hardware redundancy, fault estimation allows one to keep using the faulty hardware since by reconstructing the fault signal it can be removed from the raw signal. In the application considered — a small UAV — it is assumed that no hardware redundancy is available.

2.2.1 *Fault diagnosis methods*

The two main approaches currently being adopted in research into fault diagnosis are the data-driven and model-based fault diagnosis approaches [17]. The data-driven approach uses a large amount of process data and statistical decision methods to perform fault diagnosis [18]. Recent trends in data-driven approaches have led to a proliferation of studies that use neural

networks to perform fault diagnosis [19–23]. The quality of the data used is a key element of this approach. The model-based approach on the other side takes advantage of the knowledge of the dynamics of the system [15, 24]. Also called analytical redundancy, this approach aims to predict the output of the process by using a mathematical representation that reconstructs the process behaviour on-line. The difference between the process output and the predicted output is called the residual [25]. In a fault-free situation regardless of the noise, the residual should be equal to zero, while in a faulty situation, it should be different from zero. The quality of the mathematical representation of the process is a key element of this approach.

This thesis focuses on the model-based approach since it is applied to fixed-wing UAVs and it is assumed that the dynamics of the fixed-wing UAV considered are known. This assumption is rationalized in Chapter 3 by a full description of this model.

2.3 MODEL-BASED FAULT DIAGNOSIS

The general architecture of model-based fault diagnosis is illustrated in Figure 2.4.

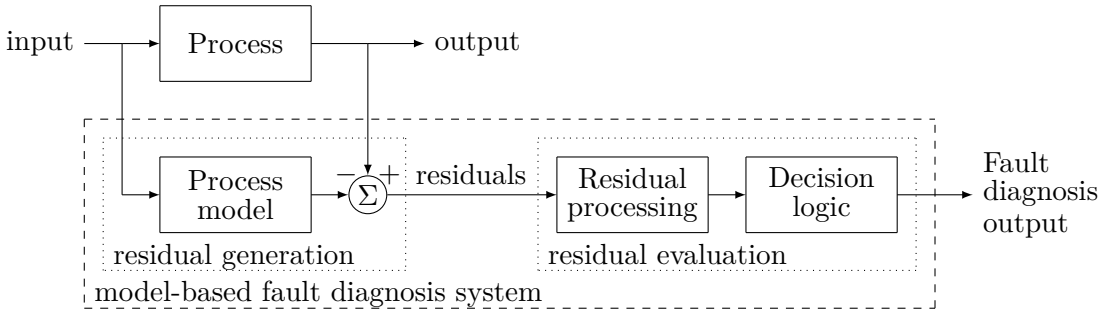


Figure 2.4: General architecture of model-based fault diagnosis

As shown in Figure 2.4, a model-based fault diagnosis system consists of two sequential steps: (i) the residual generation, and (ii) the residual evaluation. These steps are described in the Section 2.3.1 and Section 2.3.2.

2.3.1 *Residual generator*

As previously explained and as illustrated in Figure 2.4, the residual is the difference between the process model output and the corresponding measured output. Herein, the process is the system considered, the inputs are the actuator command $\mathbf{u}$, and the outputs are the sensors measurements $\mathbf{y}$. Then, the residual is the difference between the measurements $\mathbf{y}$ and the process model outputs. This process model output is obtained by evaluating the function $\mathcal{H}_k(\cdot)$ for the state predicted state vector $\mathbf{z}_{k|k-1}$, given the input value and the model equation. Then, the residual is given by:

$$\tilde{\mathbf{y}}_k = \mathbf{y}_k - \mathcal{H}_k\left(\mathbf{z}_{k|k-1}\right) \tag{2.4}$$

The prediction of the state vector $\mathbf{z}$ is performed using the discrete state dynamics function $f_k(\cdot)$. Estimation filters provide a predicted state vector that can be used for the residual generation. This technique is detailed in Section 2.5.

2.3.2 *Residual evaluation*

As previously explained, in the absence of noise and uncertainty, a residual is equal to zero in a fault-free system. However, in practice due to the measurement and process noises — in other words, the system uncertainties — the residual is almost always different from zero, even in a fault-free system. Then, the residual evaluation module aims to analyse the residuals to be able to know if it is significantly different from zero or not. To do so, this analysis is usually performed by a statistical test. In Figure 2.4, the residual evaluation is divided into two sequential steps: (i) the residual processing and (ii) the decision logic. The residual processing aims to provide statistical characteristics of the residuals, while the decision logic use these statistical characteristics to decide if there is a fault or not. The statistical test can be used to compare the residual signal observed to the residual signal expected in a fault free case. Usually the monitored change between the fault free residuals and the observed residuals is the mean, but the variance can sometimes also be monitored. Some residual generators are able to perform residual evaluation. To be able to achieve fault isolation without any additional feature, it is necessary that each fault affects a distinct set of residuals.

The most common residual evaluator methods are described by Basseville and Nikiforov [26]. Most of these tests are based on hypothesis testing. To do so, two hypotheses are considered and confronted, the most likely according to the test outcome is then chosen. These hypotheses are $\mathcal{H}_0$ and $\mathcal{H}_1$ which are the nominal and faulty hypotheses respectively. They are given by:

$$\begin{cases} \mathcal{H}_0 : \tilde{\mathbf{y}}_k \sim p\left(\tilde{\mathbf{y}}_k | \mathcal{H}_0\right) & \text{(2.5a)} \\ \mathcal{H}_1 : \tilde{\mathbf{y}}_k \sim p\left(\tilde{\mathbf{y}}_k | \mathcal{H}_1\right) & \text{(2.5b)} \end{cases}$$

where $\tilde{\mathrm{y}}_k$ is the residual signal with a mean $\mathbb{E}\left[\tilde{\mathrm{y}}_k\right]$. The two hypotheses mean are then given by:

$$\begin{cases} \mathcal{H}_0 : \quad \mathbb{E}\left[\tilde{\mathrm{y}}_k\right] = \mu_0 & \text{(2.6a)} \\ \mathcal{H}_1 : \quad \mathbb{E}\left[\tilde{\mathrm{y}}_k\right] = \mu_1 & \text{(2.6b)} \end{cases}$$

where μ_0 is the mean of the nominal signal which is assumed to be 0, and μ_1 is the mean of the faulty signal which should be different from 0. The value μ_1 is often unknown. To deal with this, most of the hypothesis testing methods consider the hypothesis $\mathcal{H}_1$ more likely if the hypothesis $\mathcal{H}_0$ likelihood is under a specified threshold. Then, the hypotheses on the mean of the residual can be rewritten as:

$$\begin{cases} \mathcal{H}_0 : \quad \mathbb{E}\left[\tilde{\mathrm{y}}_k\right] = 0 & \text{(2.7a)} \\ \mathcal{H}_1 : \quad \mathbb{E}\left[\tilde{\mathrm{y}}_k\right] \neq 0 & \text{(2.7b)} \end{cases}$$

Note that these hypotheses are based on the means of the residuals. The standard deviation of the hypotheses $\mathcal{H}_0$ and $\mathcal{H}_1$ are respectively denoted σ_0 and σ_1. The variance — or any higher moment of the distribution — of the residual is considered unchanged between the two hypotheses in this thesis. However, the hypotheses can be performed on other statistical properties with the suitable tests. Moreover, the residual distribution is often considered as Gaussian or at least known, but other distributions can be considered too with the suitable tests.

2.3.2.1 *Thresholding*

The thresholding is the simplest residual evaluation. No residual processing is performed, other than a decision logic. The purpose of this method is to compare the residuals to a pre-defined threshold. Then, if the residual is above this threshold, the system is evaluated as faulty. Otherwise, it is determined as fault-free. This technique is efficient for abrupt fault with a significant amplitude when the threshold is well-designed. However, incipient or small amplitude faults can remain undetected.

2.3.2.2 *Student's t-test*

The Student's t-test is detailed in [27]. It is based on the assumption that the noise of the residual follows a normal distribution with mean μ_0 and variance σ_0^2. The aim of this test is to detect an unexpected change of the mean of the residuals. To do so, this test analyses a sliding window of residual values. The threshold value for the decision logic is determined by the level of significance desired in the t-distribution and the size of the window used.

2.3.2.3 *The two-sided cumulative sum*

The two-sided cumulative sum (CUSUM) test detailed in [28] makes the assumption that the residuals follow a normal distribution with a mean μ_0 and variance σ_0^2. The aim of this test is to detect an unexpected change of the mean of the residuals. This test is cumulative, as each iteration is based on the previous outcome. It is two-sided because it is based on two CUSUM, one for the positive side — that is the residual with a positive value — and one for the negative side — that is the residual with a negative value. The minimum size of a residual mean different from 0 to be cumulated must be specified. Its specification can be performed by using the cumulative distribution function (CDF) of the normal distribution, for example to only consider residuals with 1 % chance to be fault-free, the value to be taken is $2.58\sigma_0$. However, there is no rule for the specification of the threshold.

2.3.2.4 *Generalized likelihood ratio test*

The generalized likelihood ratio test (GLRT) detailed in [26] aims to detect an unexpected change in the mean of the residuals. The GLRT is based on a likelihood ratio $\Lambda(\tilde{y}) = \frac{p(\tilde{y}|\mathcal{H}_1)}{p(\tilde{y}|\mathcal{H}_0)}$ of a sliding window of size n. This likelihood ratio is then compared to a user defined threshold to decide if the system is faulty or not.

2.3.2.5 *Sequential probability ratio test*

A commonly used residual evaluator is the sequential probability ratio test (SPRT) [26, 29, 30], also known as Wald's test. This test aims to detect an unexpected change of mean of the residuals. Like the GLRT, this test is based on a likelihood ratio, but unlike the GLRT this test does not use a sliding window, but it is performed sequentially by using the previous result of the test and the last value of the residuals. This means that only the last SPRT outcome must be saved and only the current residual value is used, while the GLRT must save the previous residual values up to a user defined windows to be used for the computation of the outcome of the GLRT. This sequentiality of the test makes it more efficient than the GLRT for real-time application. However, this test has three decision outputs: $\mathcal{H}_0$, $\mathcal{H}_1$ and undefined — or in other words, no decision made. This third option usually results from the lack of informative residual data. For fault detection, it can be overruled by selecting $\mathcal{H}_0$.

2.4 PERFORMANCE AND ISSUES OF FAULT DIAGNOSIS

Given all the types and classification of fault and the decision process of fault diagnosis, some methods are more efficient than others according to certain metrics. This section aims to define what these metrics are, to then be able to compare fault diagnosis methods introduced in this thesis.

A first metric to qualify the performance of a fault diagnosis method is the minimal fault amplitude required to be detected. Indeed, the larger the fault, the easier it is to detect.

The delay to detect a fault is also an important criterion [13]. Indeed, to limit the impact of a fault as much as possible the fault must be detected as soon as possible. This delay is given by the difference between the time of activation of a fault t_{on} and the time when the decision $\mathcal{H}_1$ is taken by the fault diagnosis method.

The false alarm and missed detection probabilities should also be considered. Indeed, the decisions taken by a fault diagnosis method are not necessarily correct. A wrong decision can either be a false alarm or a missed detection. If a correct decision is made then it is a hit or a correct rejection. All possible states of the fault diagnosis [31] are illustrated in 2.1.

		Fault diagnosis decision	
		$\mathcal{H}_0$	$\mathcal{H}_1$
System state	$\mathcal{H}_0$	Correct rejection	False alarm
	$\mathcal{H}_1$	Missed detection	Hit

Table 2.1: Possible states of a fault diagnosis module

The false alarm probability is given by:

$$P_{fa_k} = \mathbb{P}\left(\tilde{\mathrm{y}}_k > \Gamma\right) = \int_{\Gamma}^{+\infty} p\left(\tilde{\mathrm{y}}_k | \mathcal{H}_0\right) \mathrm{d}\tilde{\mathrm{y}}_k \tag{2.8}$$

where Γ denote the threshold used to decide if the residual value corresponds to $\mathcal{H}_0$ or $\mathcal{H}_1$. Similarly, the missed detection probability is given by:

$$P_{mdk} = \mathbb{P}\left(\tilde{\mathrm{y}}_k < \Gamma\right) = \int_{-\infty}^{\Gamma} p\left(\tilde{\mathrm{y}}_k | \mathcal{H}_1\right) \mathrm{d}\tilde{\mathrm{y}}_k \tag{2.9}$$

The probability of detection is sometime used, and it is given by:

$$P_{dk} = 1 - P_{mdk} \tag{2.10}$$

These probabilities are illustrated in Figure 2.5 using a residual with a Gaussian distribution.

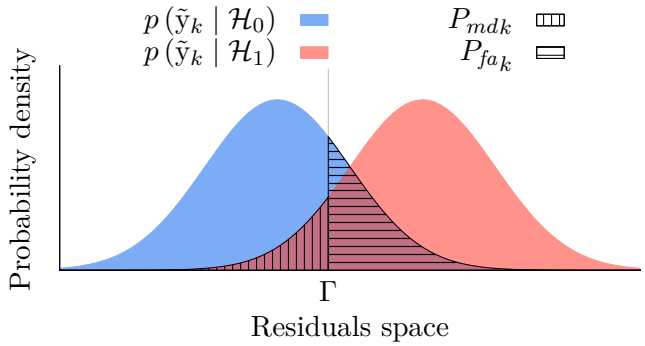


Figure 2.5: False alarm and missed detection for Gaussian distribution given a threshold Γ

An optimal trade-off between the false alarm rate and the hit rate is a key element of a fault diagnosis [15]. The trade-off can be used to define optimality criterion. Multiple criteria exist in the literature, the most frequently used are described hereafter.

2.4.1 *The Neyman-Pearson criterion*

The Neyman-Pearson criterion [32] aims to reach the false alarm and missed detection trade-off by bounding the false-alarm probability and then to maximizing the detection probability within this constraint [33]. This criterion is given by:

$$\max\left\{P_d\right\} \text{ such that } P_{fa} \leq \alpha \tag{2.11}$$

where α is the upper bound of the false alarm rate, also known as the significance level of the test. The above optimization problem has an explicit solution given by:

$$\Lambda\left(\tilde{\mathrm{y}}_k\right) = \frac{p\left(\tilde{\mathrm{y}}_k|\mathcal{H}_1\right)}{p\left(\tilde{\mathrm{y}}_k|\mathcal{H}_0\right)} \underset{\mathcal{H}_0}{\overset{\mathcal{H}_1}{\gtrless}} \eta \tag{2.12}$$

where η is the threshold that satisfies the constraint:

$$P_{fa_k} = \int_{\eta}^{+\infty} p\left(\Lambda\left(\tilde{\mathrm{y}}_k\right)|\mathcal{H}_0\right) \mathrm{d}\Lambda\left(\tilde{\mathrm{y}}_k\right) = \alpha \tag{2.13}$$

2.4.2 *The receiver operating characteristic curve criterion*

The receiver operating characteristic (ROC) curve [34] represents the detection probability given the false alarm probability. This curve applies to Gaussian distributions with a fixed mean $\mu_0 = 0$ and for multiple faulty means μ_1 is illustrated in Figure 2.6.

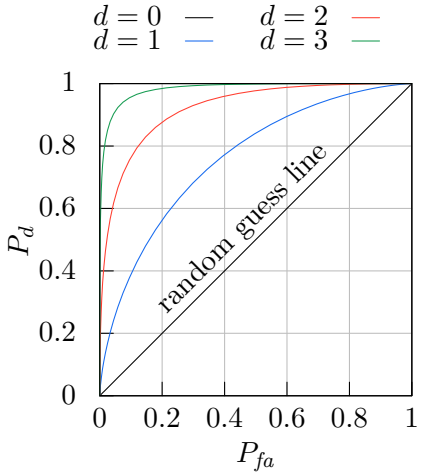


Figure 2.6: ROC curves with Gaussian error $d = \frac{\mu_1 - \mu_0}{\sigma}$, with $\sigma = \sigma_0 = \sigma_1$

When both means are equal — in other words $\mu_1 = \mu_0$ —, then all the points of the curve are defined on the random guess line (the diagonal line). The aim of this ROC curve criterion is to provide a point of the curve to either:

- maximize the vertical distance from the guess line, this is performed by maximizing the value of Youden's index [35] denoted J, and it is given by:

$$\max\{J\} = \max\{P_d - P_{fa}\}; \tag{2.14}$$

- minimize the distance from the optimal $P_d = 1$, $P_{fa} = 0$, this is performed by solving the following optimization function:

$$\min\left\{\sqrt{(1-P_d)^2 + {P_{fa}}^2}\right\} = \min\left\{\sqrt{{P_{md}}^2 + {P_{fa}}^2}\right\}. \tag{2.15}$$

2.5 MODEL-BASED FAULT ESTIMATION

Model-based fault estimation is performed in this thesis using estimation theory [36] to estimate the state vector and the faults. The estimation methods used in this thesis require the introduction of fault dynamic models which are given by:

$$\mathbf{f}_{\mathbf{a}k} = \mathcal{G}_{ak}\left(\mathbf{f}_{\mathbf{a}k-1}\right) \tag{2.16a}$$
$$\mathbf{f}_{\mathbf{s}k} = \mathcal{G}_{sk}\left(\mathbf{f}_{\mathbf{s}k-1}\right) \tag{2.16b}$$

where $\mathcal{G}_{ak}(\cdot) \in \mathbb{R}^{n_a} \to \mathbb{R}^{n_a}$ denotes the discrete dynamics of the actuators faults and $\mathcal{G}_{sk}(\cdot) \in \mathbb{R}^{n_s} \to \mathbb{R}^{n_s}$ the discrete dynamics of the sensor faults.

Then, the dynamics of the system completed with additive faults and their dynamics is given by:

$$\begin{cases} \begin{bmatrix} \mathbf{z}_k \\ \mathbf{f}_{\mathbf{a}k} \\ \mathbf{f}_{\mathbf{s}k} \end{bmatrix} = \begin{bmatrix} \mathcal{F}_k\left(\mathbf{z}_{k-1},\, \mathbf{u}_k + \mathbf{f}_{\mathbf{a}k-1}\right) \\ \mathcal{G}_{ak}\left(\mathbf{f}_{\mathbf{a}k-1}\right) \\ \mathcal{G}_{sk}\left(\mathbf{f}_{\mathbf{s}k-1}\right) \end{bmatrix} & \text{(2.17a)} \\ \mathbf{y}_k = \mathcal{H}_k\left(\mathbf{z}_k\right) + \mathbf{f}_{\mathbf{s}k} & \text{(2.17b)} \end{cases}$$

For sake of brevity, an extended state vector $\mathbf{x} \in \mathbb{R}^{n_x}$ is defined as:

$$\mathbf{x}_k = \begin{bmatrix} \mathbf{z}_k \\ \mathbf{f}_{\mathbf{a}k} \\ \mathbf{f}_{\mathbf{s}k} \end{bmatrix} \tag{2.18}$$

and then the dynamics of the system are given by:

$$\begin{cases} \mathbf{x}_k = f_k\left(\mathbf{x}_{k-1},\, \mathbf{u}_k\right) & \text{(2.19a)} \\ \mathbf{y}_k = h_k\left(\mathbf{x}_k\right) & \text{(2.19b)} \end{cases}$$

where the dynamics function $f_k(\cdot) \in \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_x}$ and the measurement function $h_k(\cdot) \in \mathbb{R}^{n_x} \to \mathbb{R}^{n_y}$ are given by:

$$\begin{cases} f_k\left(\mathbf{x}_k,\, \mathbf{u}_k\right) = \begin{bmatrix} \mathcal{F}_k\left(\mathbf{z}_k,\, \mathbf{u}_k + \mathbf{f}_{\mathbf{a}k}\right) \\ \mathcal{G}_{ak}\left(\mathbf{f}_{\mathbf{a}k}\right) \\ \mathcal{G}_{sk}\left(\mathbf{f}_{\mathbf{s}k}\right) \end{bmatrix} & \text{(2.20a)} \\ h_k\left(\mathbf{x}_k\right) = \mathcal{H}_k\left(\mathbf{z}_k\right) + \mathbf{f}_{\mathbf{s}k} & \text{(2.20b)} \end{cases}$$

Note that the state vector $\mathbf{f_a}$ and $\mathbf{f_s}$ does not have always the same number of state than there is actuator or sensor respectively. Indeed, the fault can be estimated on one or more actuator, but not necesarilly on all of them. Likewise for the sensor fault, not all sensor must be systematically considered for fault estimation. However, for sake of brevity, unless it is specified otherwise, it is considered that $\mathbf{f_a}$ and $\mathbf{f_s}$ has the same dimension as $\mathbf{u}$ and $\mathbf{y}$ respectively.

2.5.1 *Estimation of the extended state vector*

As pointed out previously, the fault detection and identification processes can be efficiently tackled by estimating the extended state vectors of the system as described by equations (2.19). However, it is assumed that there is no full knowledge of the state vector. Indeed, it is straightforward that the faults are unknown, but it is common that some other state cannot also be perfectly predicted or directly measured. The prediction of a state is possible by using its evolution function. However, some uncertainties such as unmodelled dynamics or unknown disturbances may provide a prediction away from the true state. Moreover, the sensors used usually suffer from unwanted measurements changes that are generally unknown, and which may have originated during the capture, storage, transmission, processing, or conversion of the data. These modifications induce uncertainties into the measured data. Then, to tackle these uncertainties, the system to be surveyed can be considered as a stochastic process model. The process evolution and the measurement noises are here considered to be additive, and the stochastic process model is given by:

$$\begin{cases} \mathbf{x}_k = f_k\left(\mathbf{x}_{k-1},\, \mathbf{u}_k\right) + \boldsymbol{\eta}_k & \text{(2.21a)} \\ \mathbf{y}_k = h_k\left(\mathbf{x}_k\right) + \boldsymbol{\nu}_k & \text{(2.21b)} \end{cases}$$

where $\boldsymbol{\eta}_k \in \mathbb{R}^{n_x}$ represents the uncertainties of the model and is called the process noise and $\boldsymbol{\nu}_k \in \mathbb{R}^{n_y}$ represents the measurement noise of the sensors. They are assumed to be of zero mean and independent, $\mathbb{E}\left[\boldsymbol{\eta}_k {\boldsymbol{\nu}_k}^\top\right] = 0$. As this thesis aims at dealing with on-line fault monitoring, the focus is on recursive state estimation method for stochastic process, which leads us to the concept of the optimal filter.

2.5.2 *Optimal estimation filter*

The optimal estimation filter aims to estimate the density of $\mathbf{x}_k$ given all the previous measurements $\mathbf{Y}_{1:k-1} \in \mathbb{R}^{n_y \times k-1}$. This corresponds to the posterior density $p\left(\mathbf{x}_k|\mathbf{Y}_{1:k}\right)$.

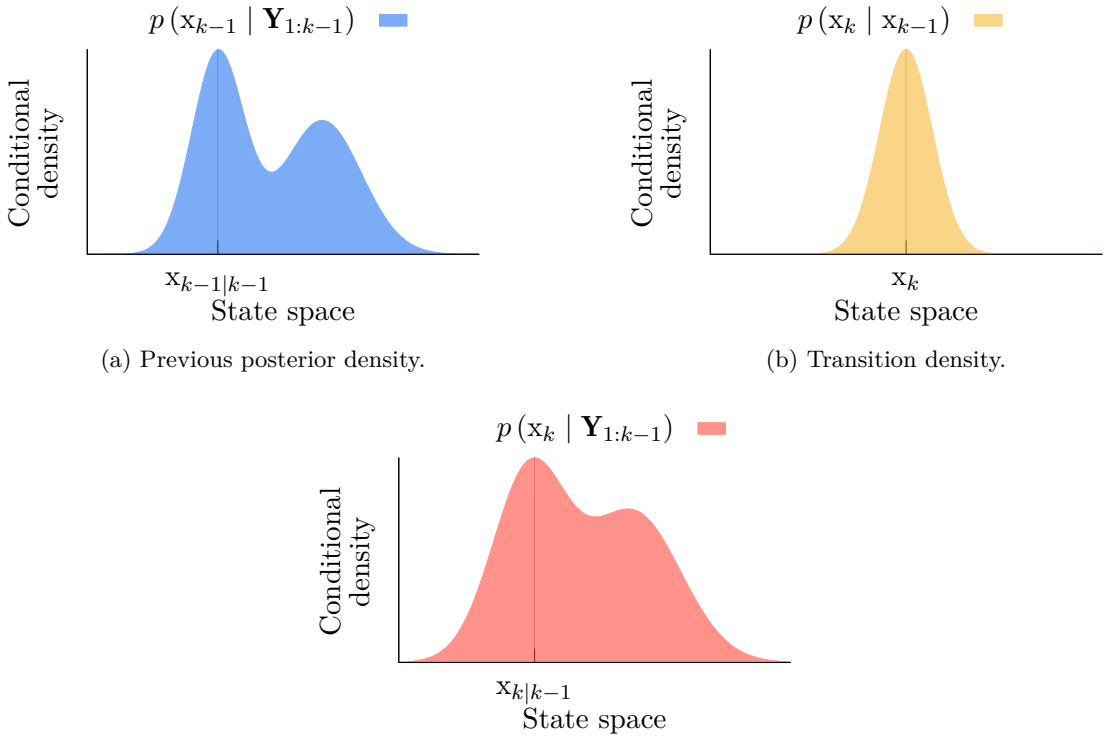
Estimation filters usually have two steps: the prediction and the update. The prediction seeks to compute the prior state density $p\left(\mathbf{x}_k|\mathbf{Y}_{1:k-1}\right)$ while the update aims to update the prior state density using the up-to-date measurements by computing the posterior density $p\left(\mathbf{x}_k|\mathbf{Y}_{1:k}\right)$.

2.5.2.1 *Prediction step*

The prior state density is obtained by a convolution between the state transition density $p(\mathbf{x}_k|\mathbf{x}_{k-1})$ from (2.21a) and the posterior density of the previous time step $p(\mathbf{x}_{k-1}|\mathbf{Y}_{1:k-1})$. This convolution is known as the Chapman-Kolmogorov equation and is obtained by the recursive process given by:

$$p(\mathbf{x}_k|\mathbf{Y}_{1:k-1}) = \int_{\mathbb{R}^{n_x}} p(\mathbf{x}_k|\mathbf{x}_{k-1})\, p(\mathbf{x}_{k-1}|\mathbf{Y}_{1:k-1})\, \mathrm{d}\mathbf{x}_{k-1} \tag{2.22}$$

At the first time-step when there is no knowledge of the previous posterior density, an initial state density denoted $p(\mathbf{x}_0)$ is used instead. This initial state density then represents the initial state uncertainties. The prediction step is illustrated in Figure 2.7.



(a) Previous posterior density.

(b) Transition density.

(c) Prior state density.

Figure 2.7: Previous posterior density (a) and transition density (b) convoluted to obtain the prior state density (c).

2.5.2.2 *Update step*

The posterior density is computed when a new measurement becomes available. Then, the posterior density $p(\mathbf{x}_k|\mathbf{Y}_{1:k})$ is given by the Bayes' formula [37]:

$$p(\mathbf{x}_k|\mathbf{Y}_{1:k}) = \frac{1}{p(\mathbf{Y}_{1:k})} p(\mathbf{y}_k|\mathbf{x}_k)\, p(\mathbf{x}_k|\mathbf{Y}_{1:k-1}) \tag{2.23}$$

where $p(\mathbf{y}_k|\mathbf{x}_k)$ is the likelihood, $p(\mathbf{x}_k|\mathbf{Y}_{1:k-1})$ is the prior state density and the normalizing constant $p(\mathbf{Y}_{1:k})$ is given by:

$$p(\mathbf{Y}_{1:k}) = \int_{\mathbb{R}^{n_x}} p(\mathbf{y}_k|\mathbf{x}_k)\, p(\mathbf{x}_k|\mathbf{Y}_{1:k-1})\, \mathrm{d}\mathbf{x}_k \tag{2.24}$$

The likelihood $p(\mathbf{y}_k|\mathbf{x}_k)$ depends on the law of measurement noise given by (2.21b).

Equations (2.22) and (2.23) represent the theoretical filter named optimal filter. A filter is considered to be optimal if it is theoretically equivalent to the optimal filter. The update step is illustrated in Figure 2.8.

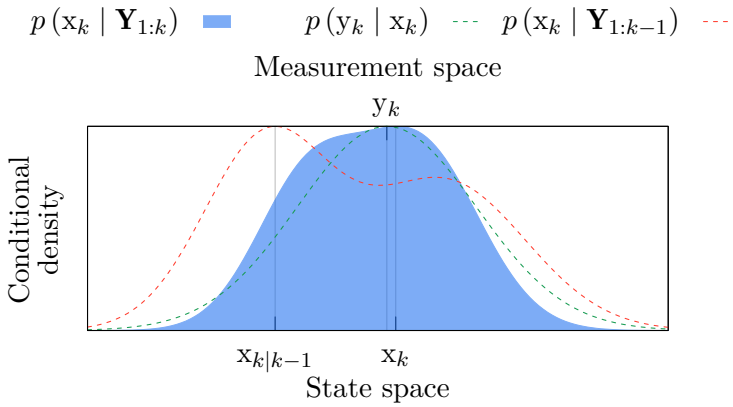


Figure 2.8: Update step.

2.5.3 *State estimators*

The state density is the most exhaustive way to represent a state estimate. However, some applications like the one considered in this thesis need to pick a single value from the estimated posterior density. This single value is a point that is to serve as the best guess or the best estimate. The estimated state vector is then denoted $\hat{\mathbf{x}}$. Then, the following are the most common criteria that are used to select this best estimate.

2.5.3.1 *Maximum a posteriori*

The maximum a posteriori (MAP) estimation [38] aims to estimate the parameters of a probability distribution by selecting a point in the state space that maximizes the posterior density $p(\mathbf{x}_k|\mathbf{Y}_{1:k})$. This point is then called the MAP estimate, and it is given by:

$$\hat{\mathbf{x}}_k = \arg\max_{\mathbf{x}_k} p(\mathbf{x}_k|\mathbf{Y}_{1:k}) \tag{2.25}$$

The MAP estimate is illustrated in Figure 2.9.

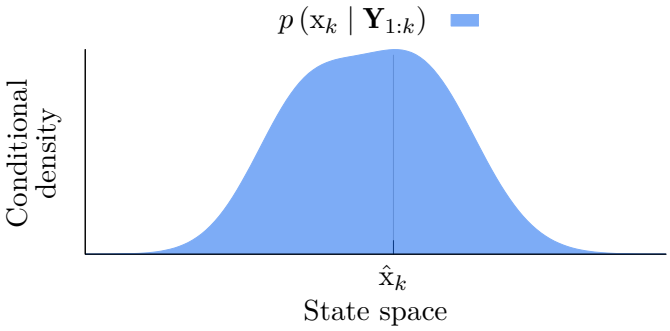


Figure 2.9: MAP estimate.

2.5.3.2 *Maximum likelihood*

The maximum likelihood (ML) estimation [39] aims to estimate the parameters of a probability distribution by selecting a point in the measurements space that maximizes the likelihood $p(\mathbf{y}_k|\mathbf{x}_k)$. This point is then called the ML estimate, and it is given by:

$$\hat{\mathbf{x}}_k = \arg\max_{\mathbf{x}_k} p(\mathbf{y}_k|\mathbf{x}_k) \tag{2.26}$$

The ML estimate is illustrated in Figure 2.10.

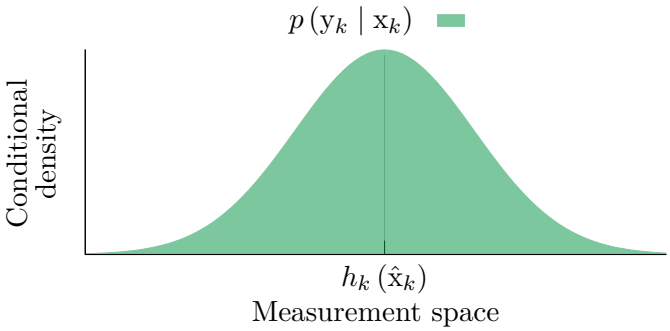


Figure 2.10: ML estimate.

2.5.3.3 *Minimum mean-square error*

The minimum mean-square error (MMSE) estimation [40] aims to provide an estimate that minimizes the error variance. The resulting MMSE estimator of $\mathbf{x}_k$ in terms of $\mathbf{Y}_{1:k}$ corresponds to:

$$\hat{\mathbf{x}}_k = \mathbb{E}\left[\mathbf{x}_k | \mathbf{Y}_{1:k}\right] \tag{2.27a}$$

$$= \int_{\mathbb{R}^{n_x}} \mathbf{x}_k p\left(\mathbf{x}_k | \mathbf{Y}_{1:k}\right) \mathrm{d}\mathbf{x}_k \tag{2.27b}$$

The MMSE estimate is illustrated in Figure 2.11.

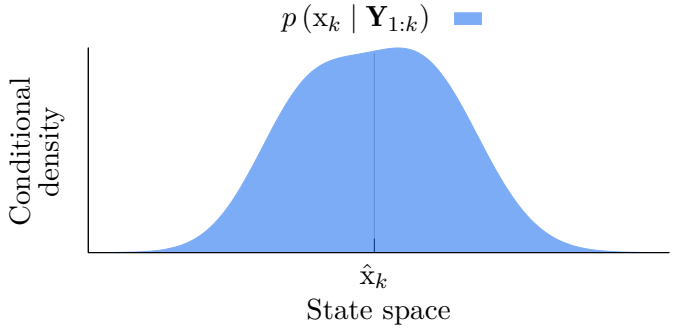


Figure 2.11: MMSE estimate.

Note that in the case where the posterior density is given by a symmetric probability distribution — for example a Normal distribution — the MMSE estimate is equivalent to the MAP estimate.

The implementation of the optimal filter with state estimator yields various types of filters. The choice of these filters depends on the assumption made on the system. These assumptions are often related to the linearity of the system and the nature of the noises. The following sections aims to present filters that cover the most common assumption pertaining to linearity and nature of the noise.

2.5.4 *Linear Gaussian estimation filter*

The optimal estimation problem for a linear system with Gaussian noise has long been solved. Indeed, the Kalman filter, first introduced by Kalman [41], is theoretically equivalent to the optimal filter under these assumptions [42]. In the linear case, (2.21) can be rewritten as:

$$\begin{cases} \mathbf{x}_k = \mathbf{F}\mathbf{x}_{k-1} + \mathbf{B}\mathbf{u}_k + \boldsymbol{\eta}_k & \text{(2.28a)} \\ \mathbf{y}_k = \mathbf{H}\mathbf{x}_k + \boldsymbol{\nu}_k & \text{(2.28b)} \end{cases}$$

where $\mathbf{F} \in \mathbb{R}^{n_x \times n_x}$ is the state matrix, $\mathbf{B} \in \mathbb{R}^{n_x \times n_u}$ is the input matrix, and $\mathbf{H} \in \mathbb{R}^{n_y \times n_x}$ is the output matrix. In the Gaussian case, the noises are given by:

$$\begin{cases} \boldsymbol{\eta}_k \sim \mathcal{N}\left(\mathbf{0}_{n_x,1},\, \mathbf{Q}_k\right) & \text{(2.29a)} \\ \boldsymbol{\nu}_k \sim \mathcal{N}\left(\mathbf{0}_{n_y,1},\, \mathbf{R}_k\right) & \text{(2.29b)} \end{cases}$$

where $\mathbf{Q}_k \in \mathbb{R}^{n_x \times n_x}$ is the covariance of the process noise, and $\mathbf{R}_k \in \mathbb{R}^{n_y \times n_y}$ is the covariance of the measurements noise.

The Kalman filter estimate is given by:

$$\hat{\mathbf{x}}_k = \left(1 - \mathbf{K}_k \mathbf{H}\right) \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k \mathbf{y}_k \tag{2.30}$$

where $\hat{\mathbf{x}}_{k|k-1}$ is given by:

$$\hat{\mathbf{x}}_{k|k-1} = \mathbf{F}\hat{\mathbf{x}}_{k-1} + \mathbf{B}\mathbf{u}_k \tag{2.31}$$

and $\mathbf{K}_k \in \mathbb{R}^{n_x \times n_y}$ is the Kalman gain, and it is given by:

$$\mathbf{K}_k = \hat{\mathbf{P}}_{k|k-1}\mathbf{H}^\top \left(\mathbf{R}_k + \mathbf{H}\hat{\mathbf{P}}_{k|k-1}\mathbf{H}^\top\right)^{-1} \tag{2.32}$$

where the superscript $\top$ denotes the matrix transposition, $\hat{\mathbf{P}}_{k|k-1} \in \mathbb{R}^{n_x \times n_x}$ is the prior estimated covariance, and it is given by:

$$\hat{\mathbf{P}}_{k|k-1} = \mathbf{F}\hat{\mathbf{P}}_{k-1}\mathbf{F}^\top + \mathbf{Q}_k \tag{2.33}$$

where $\hat{\mathbf{P}}_k$ is the posterior estimated covariance, which is given by:

$$\hat{\mathbf{P}}_k = \left(\mathbf{I}_{n_x} - \mathbf{K}_k \mathbf{H}\right) \hat{\mathbf{P}}_{k|k-1} \tag{2.34}$$

Based on (2.30) to (2.34) and on the assumption of Gaussian process and measurement noises, the estimated posterior density $\hat{p}\left(\mathbf{x}_k | \mathbf{Y}_{1:k}\right)$ is given by $\mathcal{N}\left(\hat{\mathbf{x}}_k,\, \hat{\mathbf{P}}_k\right)$, and the theoretically posterior density $p\left(\mathbf{x}_k | \mathbf{Y}_{1:k}\right)$ is given by $\mathcal{N}\left(\mathbf{x}_k,\, \mathbf{P}\right)_k$. Hence, the Kalman filter gives an optimal solution to the estimation problem — with a linear system and Gaussian noises. Moreover, since the point estimate given by the Kalman filter is the expected value of the posterior density, which corresponds to the MMSE estimate [41], and given that the density is symmetrical, the point estimate also corresponds to the MAP.

The Kalman filter structure is illustrated in Figure 2.12

The Kalman filter algorithm is presented in Algorithm 2.1

2.5.5 *Non-linear Gaussian filter*

In the non-linear case when the densities are Gaussian, the simplest approach is to linearize (2.21) and then use the Kalman filter. This approach is named the EKF [43], but unlike the Kalman filter in the linear case, it is not an optimal filter. The linearization is

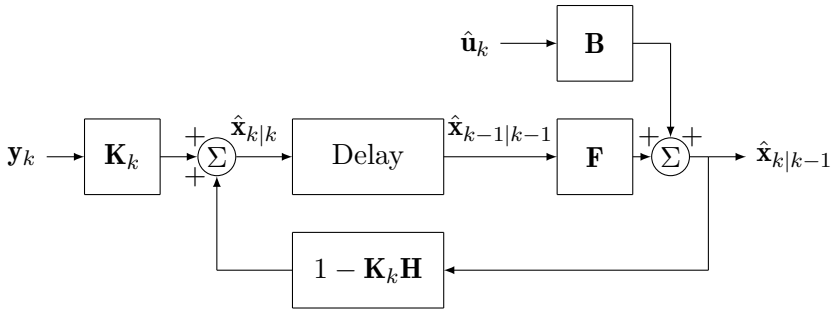


Figure 2.12: Kalman filter structure

Algorithm 2.1 Kalman filter

$k \leftarrow 0$

⋮ //Initialization

Loop

$k \leftarrow k + 1$

//Prediction

$\hat{\mathbf{x}}_{k|k-1} = \mathbf{F}\hat{\mathbf{x}}_{k-1|k-1} + \mathbf{B}\mathbf{u}_k$ //See (2.31)

$\hat{\mathbf{P}}_{k|k-1} = \mathbf{F}\hat{\mathbf{P}}_{k-1|k-1}\mathbf{F}^\top + \mathbf{Q}_k$ //See (2.33)

//Kalman gain update

$\mathbf{K}_k = \hat{\mathbf{P}}_{k|k-1}\mathbf{H}^\top \left(\mathbf{R}_k + \mathbf{H}\hat{\mathbf{P}}_{k|k-1}\mathbf{H}^\top\right)^{-1}$ //See (2.32)

//Measurement update

$\hat{\mathbf{x}}_{k|k} = (1 - \mathbf{K}_k\mathbf{H})\,\hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k\mathbf{y}_k$ //See (2.30)

$\hat{\mathbf{P}}_{k|k} = (\mathbf{I}_{n_x} - \mathbf{K}_k\mathbf{H})\,\hat{\mathbf{P}}_{k|k-1}$ //See (2.34)

obtained by computing the Jacobian matrix of the dynamics function and the measurement function, which is respectively given by:

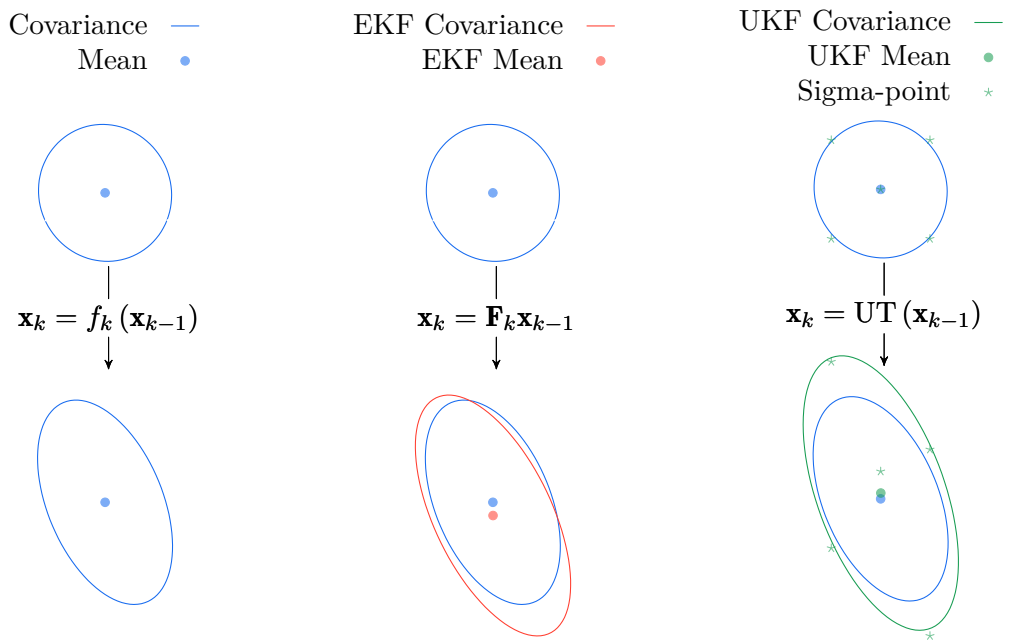
$$\begin{cases} \mathbf{F}_k = \left.\dfrac{\partial f_k(\mathbf{x}_k, \mathbf{u}_k)}{\partial \mathbf{x}_k}\right|_{\mathbf{x}_k = \hat{\mathbf{x}}_k} & \text{(2.35a)} \\ \mathbf{H}_k = \left.\dfrac{\partial h_k(\mathbf{x}_k)}{\partial \mathbf{x}_k}\right|_{\mathbf{x}_k = \hat{\mathbf{x}}_k} & \text{(2.35b)} \end{cases}$$

The linearization must be performed before each prediction step.

This approach is commonly used in many applications, such as the global navigation satellite system (GNSS) receivers. In [44], the EKF was applied to fault diagnosis for a speed sensor, with the estimation of false alarm and detection probabilities. However, it has some limitations, that prevent the use of this approach for some applications. One of these limitations is that, in some applications, the covariance estimates performed by the EKF tend to underestimate the true covariance of the state [45]. This is mainly due to the fact that it is propagated through linearization. Moreover, the initial estimation error has to be

small with the EKF, otherwise the filter may diverge. In addition to that, when the system is subject to severe non-linearity the EKF may diverge too. Finally, for fault estimation, the linearization techniques used by the EKF, tend to suffer from poor fault detection or high false alarm rates [46].

Various enhancements to the EKF filters have been proposed to enhance its robustness. However, one of the most advanced Kalman filters, that is able to deal with significant non-linearity, is the UKF [47, 48]. The UKF takes its name from the deterministic sampling technique used to approximate the Gaussian densities. This technique, known as the unscented transformation, approximates the distribution by a minimal set of carefully chosen sample points, called sigma-points. These sigma points are generally four to ten points, selected around the mean, and propagated using the non-linear dynamic. Each sigma-point is associated with multiple weights. Finally, the posterior density is approximated by computing the weighted sum of the propagated sigma-points. The UKF is known to accurately estimate the posterior density up to the 3^{rd} order of the Taylor series expansion of the non-linear system. A comparison between the EKF and the UKF approaches is illustrated using a two-dimensional state vector in Figure 2.13.



(a) True transformation. (b) Linear transformation. (c) Unscented transformation.

Figure 2.13: Comparison of the true transformation (a), the linearization approach taken by a EKF (b), and the unscented transformation approach taken by the UKF (c) on a two-dimensional state vector.

Other approaches exist to tackle estimation of non-linear Gaussian systems, like the ensemble Kalman filter or the change of space to bring back the system in a linear geometry, but are relatively less common and not used in this thesis.

2.5.6 *Non-Linear non-Gaussian filter*

To tackle the estimation of non-linear systems, with non-Gaussian distributions, and including kernel mixtures, the most commonly used approach is the particle filter. The idea behind this filter is to use the sequential Monte Carlo method to provide an estimate. A Monte Carlo method aims to generate input as randomly distributed values, that are then processed by the system dynamics to provide a range of solutions. The use of this method for estimation application has been initiated by the development of the sequential importance resampling (SIR) particle filters [49, 50], and this approach is increasingly used because of the increase in computational performance of modern microprocessors.

2.5.6.1 *Principle of the Monte Carlo approximation*

Let X be a random variable on $\mathbb{R}^d$ distributed according to the probability density function denoted $p(\cdot)$ and with $X^{1:N}$ a set of independent random variables on $\mathbb{R}^d$ with the same distribution as X. Then, for any bounded function $\Phi(\cdot) : \mathbb{R}^d \to \mathbb{R}$, the mean of $\Phi(X)$ is given by:

$$\mathbb{E}[\Phi(X)] = \int_{\mathbb{R}^d} \Phi(X)\, p(X)\, \mathrm{d}X \tag{2.36}$$

The Monte Carlo methods approximate the expectancy by the empirical mean:

$$\bar{\Phi}(X) = \frac{1}{N} \sum_{i=1}^{N} \Phi\left(X^i\right), \quad X^i \sim p() \tag{2.37}$$

The law of large numbers ensures that the empirical mean converges almost surely to the expected value:

$$\bar{\Phi}(X) = \frac{1}{N} \sum_{i=1}^{N} \Phi\left(X^i\right) \xrightarrow[N\to\infty]{} \mathbb{E}[\Phi(X)] \tag{2.38}$$

The variance of the Monte Carlo estimator $\bar{\Phi}(X)$ is equal to:

$$\mathrm{Var}\left(\bar{\Phi}(X)\right) = \frac{\sigma^2}{N}, \tag{2.39}$$

where

$$\sigma^2 = \int_{\mathbb{R}^d} (\Phi(X) - \mathbb{E}[\Phi(X)])^2 p(X)\, \mathrm{d}X \tag{2.40}$$

The law of the error $\Phi(X) - \mathbb{E}[\Phi(X)]$ when the number of samples N tends to infinity is given by the central limit theorem:

$$\frac{\sqrt{N}}{\sigma}\left(\Phi(X) - \mathbb{E}[\Phi(X)]\right) \xrightarrow[N\to\infty]{} \mathcal{N}(0,\,1) \tag{2.41}$$

The average error is of order $\frac{\sigma}{\sqrt{N}}$. The Monte Carlo error does not depend on the dimension of the state.

2.5.6.2 *The sequential importance resampling particle filter*

The SIR particle filter aims to approximate the posterior density $p(\mathbf{x}_k|\mathbf{Y}_{1:k})$ using a weighted set of $N \gg 1$ samples, called particles. To do so, the particles are represented by weighted Dirac delta functions, where the position of the particle determines the value of the state associated with the particle. The posterior density is then approximated by:

$$p(\mathbf{x}_k|\mathbf{Y}_{1:k}) \approx \sum_{i=1}^{N} w_k^i \delta\left(\mathbf{x}_k - \mathbf{x}_k^i\right) \tag{2.42}$$

where $\delta(\cdot)$ denotes the Dirac delta functions, $w_k^i \in \mathbb{R}^+$ are the importance weight of a particle, and the superscript i denotes the index of the particle. The weights are normalized to ensure and satisfy the equation:

$$\sum_{i=1}^{N} w_k^i = 1 \tag{2.43}$$

Assuming that the previous posterior density can be approximated in the same way as (2.42), and based on (2.22), the prediction can be approximated by:

$$p(\mathbf{x}_k|\mathbf{Y}_{1:k-1}) \approx \int_{\mathbb{R}^{n_x}} p(\mathbf{x}_k|\mathbf{x}_{k-1}) \sum_{i=1}^{N} w_{k-1}^i \delta\left(\mathbf{x}_{k-1} - \mathbf{x}_{k-1}^i\right) \mathrm{d}\mathbf{x}_{k-1} \tag{2.44a}$$

$$= \sum_{i=1}^{N} w_{k-1}^i \int_{\mathbb{R}^{n_x}} p(\mathbf{x}_k|\mathbf{x}_{k-1})\, \delta\left(\mathbf{x}_{k-1} - \mathbf{x}_{k-1}^i\right)\, \mathrm{d}\mathbf{x}_{k-1} \tag{2.44b}$$

$$= \sum_{i=1}^{N} w_{k-1}^i p\left(\mathbf{x}_k|\mathbf{x}_{k-1}^i\right) \tag{2.44c}$$

$$= \sum_{i=1}^{N} w_{k-1}^i \delta\left(\mathbf{x}_k - \mathbf{x}_{k|k-1}^i\right) \tag{2.44d}$$

Equation (2.44d) highlights the fact that the weights are not updated during the prediction step, however the positions of the particles given by the Dirac delta function are updated. The error of the approximation given by (2.44d) only depends on the number of particles

used. Indeed, the benefit of using Monte Carlo methods is to approximate the state transition density $p(\mathbf{x}_k|\mathbf{x}_{k-1})$ by numerically integrating multiple particles with multiple draws of the process noise $\boldsymbol{\eta}_k$. Then, the prediction of a particle is obtained using the true non-linear equation. Using the discrete dynamics function, the prediction of a particle is given by:

$$\mathbf{x}_{k|k-1}^i = f_k\left(\mathbf{x}_{k-1}^i, \mathbf{u}_k\right) + \boldsymbol{\eta}_k^i \tag{2.45}$$

If the particles are independent and given that $\boldsymbol{\eta}_k$ is a zero mean process noise, then the unbiased estimate of the prior state density according to the law of large numbers is almost certainly given by the weighted sum of the predicted particles given by (2.44d).

The algorithm of the prediction step of the particle filter is performed by the function PREDICT detailed in Algorithm 2.2, in the case where the process noise is Gaussian.

Algorithm 2.2 Prediction step of the particle filter

Function PREDICT($\mathbf{x}_{k|k-1}^{1:N}$, $\mathbf{x}_{k-1}^{1:N}$, $\mathbf{u}_k$)
  for each $i \in [1, N]$ **do**
    $\boldsymbol{\eta}_k^i \sim \mathcal{N}(0, \mathbf{Q}_k)$
    $\mathbf{x}_{k|k-1}^i \leftarrow f_k\left(\mathbf{x}_{k-1}^i, \mathbf{u}_k\right) + \boldsymbol{\eta}_k^i$ //See (2.45)

The update step, based on (2.23), and the approximation of the prediction given by (2.44d), gives:

$$p(\mathbf{x}_k|\mathbf{Y}_{1:k}) \approx \sum_{i=1}^{N} \frac{w_{k-1}^i p\left(\mathbf{y}_k|\mathbf{x}_k^i\right) \delta\left(\mathbf{x}_k - \mathbf{x}_{k|k-1}^i\right)}{\sum_{j=1}^{N} w_{k-1}^j p\left(\mathbf{y}_k|\mathbf{x}_k^j\right)}. \tag{2.46}$$

The weights can then be updated and given by:

$$w_k^i = \frac{w_{k-1}^i p\left(\mathbf{y}_k|\mathbf{x}_k^i\right)}{\sum_{j=1}^{N} w_{k-1}^j p\left(\mathbf{y}_k|\mathbf{x}_k^j\right)}. \tag{2.47}$$

Then, by substituting (2.47) into (2.46), the posterior density can then be approximated by:

$$p(\mathbf{x}_k|\mathbf{Y}_{1:k}) \approx \sum_{i=1}^{N} w_k^i \delta\left(\mathbf{x}_k - \mathbf{x}_{k|k-1}^i\right). \tag{2.48}$$

This shows that the update step does not change the position of the particles, then (2.48) is equivalent to (2.42). However, unlike the prediction step, the update step updates the particle weights. In practice, the update of the weights is performed by computing the weights without the normalization constant and using the equation:

$$w_k^i \propto w_{k-1}^i p\left(\mathbf{y}_k|\mathbf{x}_k^i\right), \tag{2.49}$$

and then ensuring (2.43).

The algorithm of the update step of the particle filter is performed by the function UPDATE detailed in Algorithm 2.3, where $g\left(\mathbf{x}_{k|k-1}^{i}\right) = p\left(\mathbf{y}_k|\mathbf{x}_k^i\right)$, is the likelihood density with argument. Since an extended state vector is considered now, the innovation used to update the weights is not equal to the one described by (2.4) and is now given by:

$$\tilde{\mathbf{y}}_k^i = \mathbf{y}_k - h_k\left(\mathbf{x}_{k|k-1}^{i}\right). \tag{2.50}$$

Algorithm 2.3 Update step of the particle filter

Function UPDATE($w_k^{1:N}$, $w_{k-1}^{1:N}$, $\mathbf{x}_{k|k-1}^{1:N}$, $\mathbf{y}_k$)

  for each $i \in [1, N]$ **do**

    $\tilde{w}_k^i \leftarrow w_{k-1}^i g\left(\mathbf{x}_{k|k-1}^{i}\right)$ //See (2.49)

  for each $i \in [1, N]$ **do**

    $w_k^i \leftarrow \frac{\tilde{w}_k^i}{\sum_{j=1}^{N} \tilde{w}_k^j}$ //Normalization of the weights

The particle filter prediction and update steps are illustrated in Figure 2.14, where it is shown that the particle positions are updated at the prediction step only, while the weights are updated at the update step only.

Having the approximation of the posterior density by weighted Dirac delta functions makes the computation of the MAP estimate straightforward. This estimate corresponds to the particle with the highest weight. However, in practice, looking for the particle with the highest weight can be more computationally expensive than the MMSE estimate. Then, from (2.27b), the point estimate of the particle filter is usually given by:

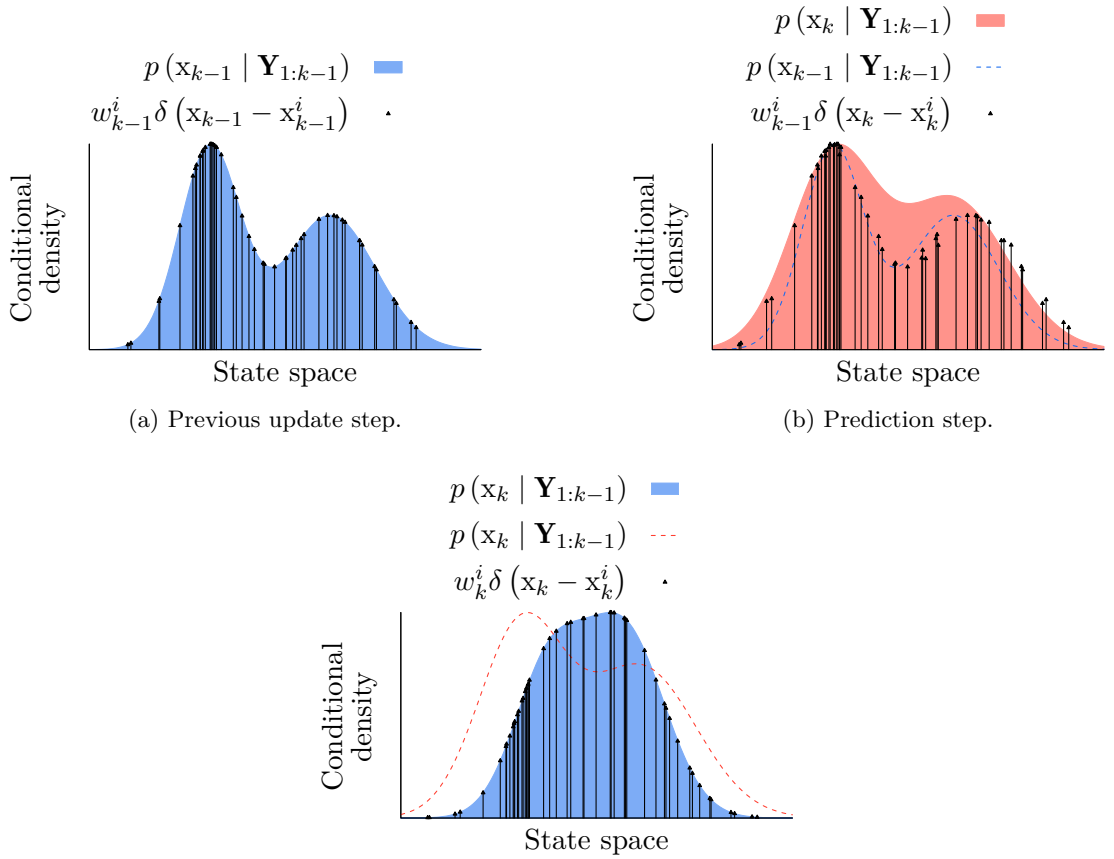
$$\hat{\mathbf{x}}_k = \sum_{i=1}^{N} w_k^i \mathbf{x}_{k|k-1}^{i}. \tag{2.51}$$

Its associated estimated covariance matrix is then given by:

$$\hat{\mathbf{P}}_k = \sum_{i=1}^{N} w_k^i \left(\mathbf{x}_{k|k-1}^{i} - \hat{\mathbf{x}}_k\right)\left(\mathbf{x}_{k|k-1}^{i} - \hat{\mathbf{x}}_k\right)^{\top}. \tag{2.52}$$

The algorithm that produces the estimate of the particle filter is performed by the function ESTIMATE detailed in Algorithm 2.4.

However, the recursive update of the weights and the normalization might lead to the degeneracy of the algorithm. This degeneracy is characterized by the fact that after multiple iterations, all the weights but one tend to zero. Then, only one particle contributes to the estimation of the state vector which drastically decreases the efficiency and the benefit of



(a) Previous update step.

(b) Prediction step.

(c) Update step.

Figure 2.14: Prediction step (b) and update step (c) of a particle filter based on the Previous update step (a).

Algorithm 2.4 Estimate step of the particle filter

Function ESTIMATE($\hat{\mathbf{x}}_k$, $\hat{\mathbf{P}}_k$, $\mathbf{x}^{1:N}_{k|k-1}$, $w^{1:N}_k$)

$$\hat{\mathbf{x}}_k \leftarrow \sum_{i=1}^{N} w^i_k \mathbf{x}^i_{k|k-1} \qquad \text{//See (2.51)}$$

$$\hat{\mathbf{P}}_k \leftarrow \sum_{i=1}^{N} w^i_k \left(\mathbf{x}^i_{k|k-1} - \hat{\mathbf{x}}_k\right)\left(\mathbf{x}^i_{k|k-1} - \hat{\mathbf{x}}_k\right)^\top \qquad \text{//See (2.52)}$$

the algorithm, and can lead to a divergence of the filter. A commonly used metric of the degeneracy phenomenon is the estimate of the effective number of particles, given by:

$$\hat{N}_{eff} = \frac{1}{\sum_{i=1}^{N} \left(w^i_k\right)^2}, \tag{2.53}$$

where the effective number of particles is equal to 1 when all the weights but one are equal to 0, and is equal to N when the weights are all equal. These two cases represent complete degeneration and no degeneration respectively. More generally the effective number of particles is bounded, with $1 \leq \hat{N}_{eff} \leq N$.

To avoid the degeneracy phenomenon an additional step called the resampling step is added to the particle filter. This step aims to remove the particles with the lowest weights and to duplicate the ones with the highest weights. There are multiple methods to perform this step, and the performance of the particle filter can be also affected by the method used. It is usually done in such a way that the number of particles remains the same. The most commonly used is the multinomial resampling method that aims to select N particles with a probability for the particle to be selected given by its weight. Then, the particles with the highest weights are the one that are more likely to be preserved and duplicated while the particles with a weight equal to zero are removed. Then, the probability to select a particle is given by:

$$\mathbb{P}\left(\acute{\mathbf{x}}_k^j = \mathbf{x}_k^i\right) = w_k^i, \tag{2.54}$$

where $\acute{\mathbf{x}}_k$ denote the state vector after the multinomial resampling.

Multinomial resampling is performed by the function MULTINOMIAL detailed in Algorithm 2.5. There exist algorithms that are more computationally efficient. The description provided here is given to illustrate the main steps of the method.

Algorithm 2.5 Multinomial resampling step

Function MULTINOMIAL($\acute{\mathbf{x}}_k^{1:N}$, $\mathbf{x}_{k|k-1}^{1:N}$, $w_k^{1:N}$)

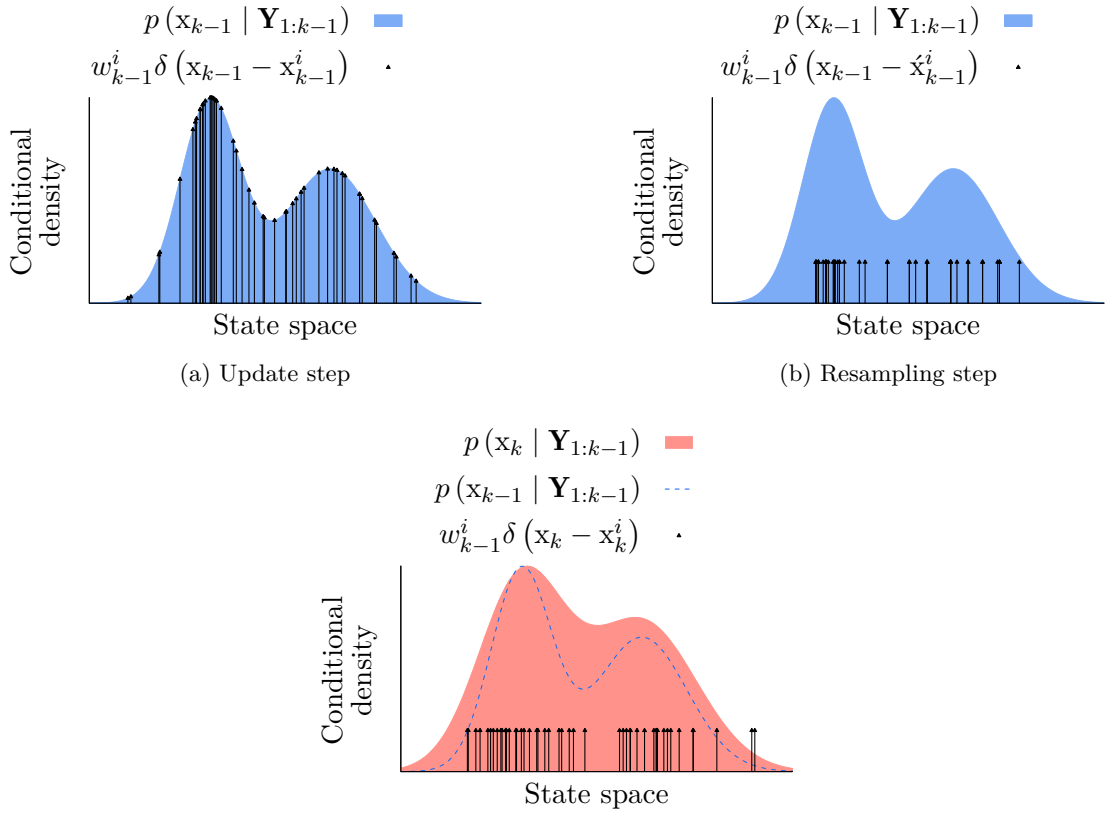
```
for each i ∈ [1, N] do
    X ~ U(0, 1)
    j ← 1
    while Σ_{l=1}^{j} w_k^i − X < 0 do
        j ← j + 1
    ẋ_k^i ← x_{k|k−1}^j
```

After performing the multinomial resampling, the weights are usually reset to be all equal, and the new weight is given by:

$$w_k^i = \frac{1}{N}. \tag{2.55}$$

The resampling step is, however, not usually performed at every time step. Indeed, the degeneracy phenomenon metric is usually compared to a user-defined threshold. In the case of the effective number of particles given by (2.53), this threshold herein denoted $\Gamma_{rspl} \in [0, 1]$ is multiplied by the number of particles N, and if it is below this value then the resampling step is performed.

The above-mentioned prediction, update and resampling algorithms are the fundamental steps of the SIR particle filter. The Figure 2.15 illustrates the prediction and update when the resampling step is performed.



(a) Update step

(b) Resampling step

(c) Prediction step.

Figure 2.15: SIR particle filter after the update step (a), the resampling step (b) and the prediction step when the resampling has been performed (c).

In Figure 2.15, the removal and duplication of particles at the resampling step are illustrated, as well as the reset of the weights. The impact of the prediction and particles positions is also shown.

The SIR particle filter is presented in Algorithm 2.6 using the previously defined functions PREDICT, UPDATE, ESTIMATE and MULTINOMIAL.

The SIR particle filter presented here is the basic particle filter that is commonly used. However, many variations of this filter have been proposed.

Algorithm 2.6 Sequential importance resampling particle filter

$k \leftarrow 0$
⋮ //Initialization
Loop
  $k \leftarrow k+1$
  PREDICT($\mathbf{x}_{k|k-1}^{1:N}$, $\mathbf{x}_{k-1}^{1:N}$, $\mathbf{u}_k$) //SeeAlgorithm 2.2
  UPDATE($w_k^{1:N}$, $w_{k-1}^{1:N}$, $\mathbf{x}_{k|k-1}^{1:N}$, $\mathbf{y}_k$) //See Algorithm 2.3
  ESTIMATE($\hat{\mathbf{x}}_k$, $\hat{\mathbf{P}}_k$, $\mathbf{x}_{k|k-1}^{1:N}$, $w_k^{1:N}$) //See Algorithm 2.4
  $\hat{N}_{eff} \leftarrow \frac{1}{\sum_{i=1}^{N} \left(w_k^i\right)^2}$ //See (2.53)
  if $\hat{N}_{eff} \leq N\Gamma_{rspl}$ **then** //if true then resample
    MULTINOMIAL($\acute{\mathbf{x}}_k^{1:N}$, $\mathbf{x}_{k|k-1}^{1:N}$, $w_k^{1:N}$) //See Algorithm 2.5
    for each $i \in [1, N]$ **do**
      $\mathbf{x}_k^i \leftarrow \acute{\mathbf{x}}_k^i$ //Particles replaced by the multinomial resampling
      $w_k^i \leftarrow \frac{1}{N}$ //Reset the weights, See (2.55)

2.5.6.3 *The regularized particle filter*

The RPF [51], was designed to improve particle filter diversity compared to the previously presented SIR particle filter. Indeed, the resampling step that aims to solve the degeneracy phenomenon introduces a new issue, namely the loss of diversity of the particles. This is due to the fact that the multinomial draw is performed according to a discrete distribution that leads to the duplication of the particles at certain positions as shown in Figure 2.15b. This duplication process can affect the same particles at multiple times. Other methods than the RPF exist to solve this issue [52, 53], but the RPF is one of the most widely used techniques.

The only difference between the SIR particle filter and the RPF is at the resampling step. Compared to the SIR particle filter, the RPF has an additional step called the regularization that aims to add diversity to the duplicated particles. This improvement in diversity of the particles is obtained by randomly moving the duplicated particles $\acute{\mathbf{x}}_k^i$ according to a regularization kernel denoted $\mathcal{K}(\cdot)$. This then yields the following regularization equation:

$$\mathbf{x}_k^i = \acute{\mathbf{x}}_k^i + h\mathbf{D}_k\varepsilon_k^i \tag{2.56}$$

where $h \in \mathbb{R}^{+*}$ is the kernel bandwidth factor, $\mathbf{D}_k$ is the real lower triangular matrix of the covariance matrix $\mathbf{P}_k$ with positive diagonal elements defined such that $\mathbf{P}_k = \mathbf{D}_k\mathbf{D}_k^\top$, and $\varepsilon_k^i \sim \mathcal{K}(\mathbf{x})$ is a regularization noise. The kernel density is a symmetric probability density function such that:

$$\int \mathbf{x}\mathcal{K}(\mathbf{x})\, d\mathbf{x} = 0, \quad \int \|\mathbf{x}\|^2\mathcal{K}(\mathbf{x})\, d\mathbf{x} < \infty \tag{2.57}$$

The optimal kernel $\mathcal{K}(\cdot)$ and bandwidth factor h are chosen to minimize the mean integrated square error (MISE) between the theoretical posterior density and the estimated one. The MISE criterion is given by:

$$\text{MISE}(\hat{p}) = \mathbb{E}\left[\int \left(\hat{p}(\mathbf{x}_k|\mathbf{Y}_{1:k}) - p(\mathbf{x}_k|\mathbf{Y}_{1:k})\right)^2 \mathrm{d}\mathbf{x}_k\right], \tag{2.58}$$

where $\hat{p}(\mathbf{x}_k|\mathbf{Y}_{1:k})$ is here the estimated posterior density of the RPF and is given by:

$$\hat{p}(\mathbf{x}_k|\mathbf{Y}_{1:k}) = \sum_{i=1}^{N} w_k^i \mathcal{K}_h\left(\mathbf{x}_k - \mathbf{x}_{k|k-1}^i\right), \tag{2.59}$$

where the kernel $\mathcal{K}_h(\cdot)$ is assumed to be symmetric such that $\mathcal{K}_h(-x) = \mathcal{K}_h(x)$, and it is given by:

$$\mathcal{K}_h(\mathbf{x}) = \frac{1}{h^{n_x}} \mathcal{K}\left(\frac{\mathbf{x}}{h}\right). \tag{2.60}$$

The approximation of the posterior density by kernels' mixture in (2.59) based on the example of the Dirac's mixture illustrated in Figure 2.15 is shown in Figure 2.16

Then, the optimal kernel that minimizes (2.58) in the case that all weights are equivalent — which is the case after the resampling — is the Epanechnikov kernel [51, 54] defined by:

$$\mathcal{K}(\mathbf{x}) = \begin{cases} \frac{n_x+2}{2c_{n_x}}\left(1 - \|\mathbf{x}\|^2\right) & \text{if} \quad \|\mathbf{x}\| < 1 \\ 0 & \text{else} \end{cases}, \tag{2.61}$$

where c_{n_x} is the volume of the unit hypersphere in $\mathbb{R}^{n_x}$, given by

$$c_{n_x} = \frac{\pi^{\frac{n_x}{2}}}{\Gamma\left(\frac{n_x}{2}+1\right)}; \tag{2.62}$$

where $\Gamma(\cdot) \in \mathbb{R} \to \mathbb{R}$ is Euler's gamma function. Then, the optimal bandwidth factor [51, 54] associated with this optimal kernel is then given by:

$$h = \kappa A N^{-\frac{1}{n_x+4}} \tag{2.63}$$

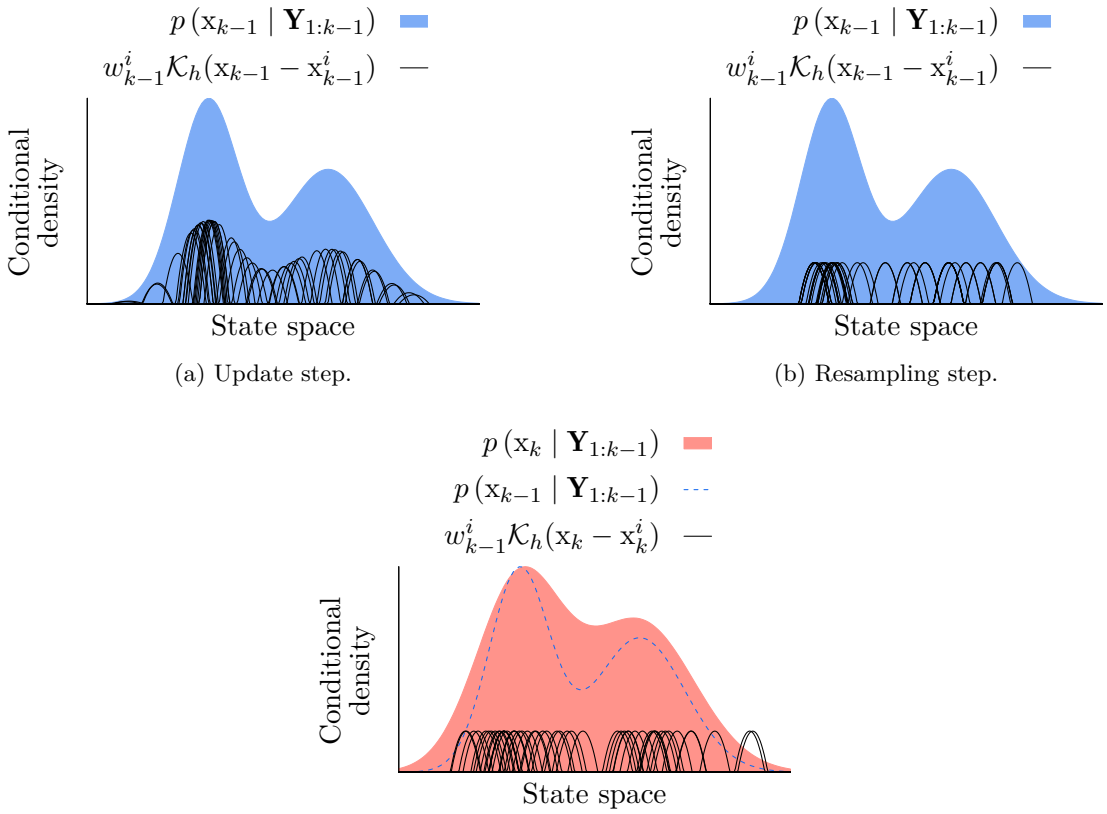
where $\kappa \in (0, 1)$ is a user-defined setting parameter, and A is given by:

$$A = \left[8c_{n_x}^{-1}(n_x+4)\left(2\sqrt{\pi}\right)^{n_x}\right]^{\frac{1}{n_x+4}} \tag{2.64}$$

The algorithm of the regularization step is performed by the function REGULARIZE detailed in Algorithm 2.7.

The RPF is presented in Algorithm 2.8, using the previously defined function REGULARIZE, and the function used by the SIR particle filter.

In theory, the kernel approximation performed by the RPF is less and less relevant as the dimension of state vector increases.



(a) Update step.

(b) Resampling step.

(c) Prediction step.

Figure 2.16: RPF after the update step (a), the resampling step (b) and the prediction step when the resampling has been performed (c).

Algorithm 2.7 Regularization step of the regularized particle filter

Function REGULARIZE($\mathbf{x}_k^{1:N}$, $\acute{\mathbf{x}}_k^{1:N}$, $\hat{\mathbf{P}}_k$)

$\mathbf{D}_k \leftarrow \{\mathbf{D}_k : \hat{\mathbf{P}}_k = \mathbf{D}_k\mathbf{D}_k^\top\}$ //Compute $\mathbf{D}_k$ such that $\hat{\mathbf{P}}_k = \mathbf{D}_k\mathbf{D}_k^\top$

for each $i \in [1, N]$ **do**

$\varepsilon_k^i \sim \mathcal{K}\left(\acute{\mathbf{x}}_k^i\right)$ //Regularization noise

$\mathbf{x}_k^i \leftarrow \acute{\mathbf{x}}_k^i + h\mathbf{D}_k\varepsilon_k^i$ //See (2.56)

2.5.7 *Multiple model architecture for estimation filter*

A multiple model architecture can be used to implement estimation filters. Multiple model architecture is used for the hybrid state estimate problem. A hybrid state is a state that has multiple components, but unlike a state vector, these components are not of the same kind.

Algorithm 2.8 Regularized particle filter

$k \leftarrow 0$
⋮ //Initialization
Loop
  $k \leftarrow k+1$
  PREDICT($\mathbf{x}_{k|k-1}^{1:N}$, $\mathbf{x}_{k-1}^{1:N}$, $\mathbf{u}_k$) //See Algorithm 2.2
  UPDATE($w_k^{1:N}$, $w_{k-1}^{1:N}$, $\mathbf{x}_{k|k-1}^{1:N}$, $\mathbf{y}_k$) //See Algorithm 2.3
  ESTIMATE($\hat{\mathbf{x}}_k$, $\hat{\mathbf{P}}_k$, $\mathbf{x}_{k|k-1}^{1:N}$, $w_k^{1:N}$) //See Algorithm 2.4
  $\hat{N}_{eff} \leftarrow \frac{1}{\sum_{i=1}^{N} (w_k^i)^2}$ //See (2.53)
  if $\hat{N}_{eff} \leq N\Gamma_{rspl}$ **then** //if true then resample
    MULTINOMIAL($\acute{\mathbf{x}}_k^{1:N}$, $\mathbf{x}_{k|k-1}^{1:N}$, $w_k^{1:N}$) //See Algorithm 2.5
    for each $i \in [1, N]$ **do**
      $w_k^i \leftarrow \frac{1}{N}$ //Reset the weights, See (2.55)
    REGULARIZE($\mathbf{x}_k^{1:N}$, $\acute{\mathbf{x}}_k^{1:N}$, $\hat{\mathbf{P}}_k$) //See Algorithm 2.7

Indeed, one of the components is a continuous-value state, while the other is a discrete-value variable often referred to as a mode. This approach is commonly used in tracking estimation with manoeuvring targets [49, 55], where the continuous state is associated with the target state while the other is associated with its modes, which are defined by different dynamics. Thus, a multiple model architecture switches between multiple dynamical models.

The hybrid state that is estimated is then given by the following couple $(\mathbf{x}_k, m_k)$.

If the MMSE is used to provide the estimate of the continuous state vector $\mathbf{x}_k$, then the overall estimate is the probabilistically weighted sum of all filter estimates [56], which for $\mathbf{x}_k$ yields:

$$\hat{\mathbf{x}}_k = \sum_{i=0}^{M-1} \hat{\mathbf{x}}_k^{(i)} \mathbb{P}\left(\mathbf{M}_{1:k} | \mathbf{Y}_{1:k}\right), \tag{2.65}$$

where $\hat{\mathbf{x}}_k^{(i)}$ denotes the estimate of the state vector $\mathbf{x}_k$ associated with the mode $m_k^{(i)}$, and $\mathbf{M}_{1:k} \in \mathbb{R}^k$ represents all the previous modes. The covariance associated with the continuous state vector $\mathbf{x}_k$ is given by:

$$\hat{\mathbf{P}}_k = \sum_{i=0}^{M-1} \left[\hat{\mathbf{P}}_k^{(i)} + \left(\hat{\mathbf{x}}_k - \hat{\mathbf{x}}_k^{(i)}\right) \left(\hat{\mathbf{x}}_k - \hat{\mathbf{x}}_k^{(i)}\right)^\top \right] \mathbb{P}\left(\mathbf{M}_{1:k} | \mathbf{Y}_{1:k}\right). \tag{2.66}$$

The probability $\mathbb{P}\left(\mathbf{M}_{1:k} | \mathbf{Y}_{1:k}\right)$ in (2.65) and (2.66) represents the probability of being in a specific mode sequence $\mathbf{M}_{1:k}$, given all the previous measurements $\mathbf{Y}_{1:k}$.

To estimate the state vectors and the mode of the multiple model architecture, one often relies on Kalman filters or its derivatives. Particle filter can also be used for that purpose but

as a filter must be defined for each mode, the computational cost of each particle filter makes its use in a bank of filters prohibitive. It is an interesting and commonly used approach in fault diagnosis [57–60].

Multiple model architectures are then hybrid state estimation methods, since they aim to estimate both the state of the system and the mode. To do so, they use a JMS.

2.5.7.1 *Jump-Markov system*

An efficient way of representing the switch between these subsystems is to use a Markov transition probability matrix. This representation is known as JMS. Using a JMS allows one to introduce a large variety of subsystems that represent the potential dynamics of the process studied with or without faults, no matter which types of faults are considered. The different subsystems are represented by discrete different dynamics. They correspond to the modes of the system denoted $m^{(i)}$, where i is the index of the mode within a finite set of M modes denoted $\mathbb{M} = \left\{m^{(0)}, m^{(1)}, \ldots, m^{(M-1)}\right\}$. The mode $m^{(0)}$ is usually associated with the nominal mode — that is the nominal dynamics of the system — while the others are associated with a faulty mode. Each mode is then characterized by specific dynamics, m_k denotes the mode of the system at time step k. For sake of brevity herein, when the mode of the system m_k is equal to a specific mode, — for example $m^{(i)}$ —, then it is denoted m_k with the mode number as superscript between parentheses — for example $m_k^{(i)}$ —, then $m_k^{(i)} \triangleq \{m_k = m^{(i)}\}$.

Since the system is Markovian, the switching between modes can be modelled by a time homogeneous M-state first order Markov chain and the transition probability of switching from a mode $m^{(j)}$ to $m^{(i)}$ is given by:

$$\pi_{ij} \triangleq \mathbb{P}\left(m_k^{(j)} | m_{k-1}^{(i)}\right), \quad \forall i, j \in \mathbb{N}^{[0:M-1]} \tag{2.67}$$

All other transition probabilities then jointly form a $M \times M$ transition probability matrix given by:

$$\mathbf{\Pi} \triangleq \begin{bmatrix} \pi_{00} & \pi_{10} & \cdots & \pi_{(M-1)0} \\ \pi_{01} & \pi_{11} & \cdots & \pi_{(M-1)1} \\ \vdots & \vdots & \ddots & \vdots \\ \pi_{0(M-1)} & \pi_{1(M-1)} & \cdots & \pi_{(M-1)(M-1)} \end{bmatrix} \tag{2.68}$$

Each element of the transition probability matrix must satisfy:

$$\pi_{ij} \geq 0, \quad \forall i, j \in \mathbb{N}^{[0:M-1]} \qquad \text{and} \qquad \sum_{i=0}^{M-1} \pi_{ij} = 1 \quad \forall j \in \mathbb{N}^{[0:M-1]} \tag{2.69}$$

The associated Markov chain is illustrated in Figure 2.17.

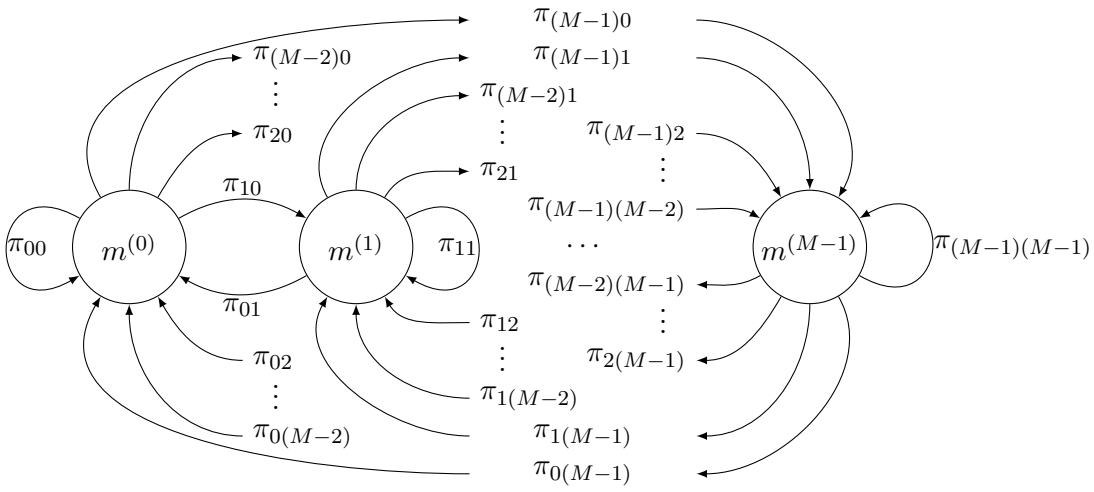


Figure 2.17: Time homogeneous M-state first order Markov chain with transition probability.

Then a discrete-time stochastic JMS is used to represent the dynamics of the system, including the transitions between modes. This generic system model, based on the representation of Tafazoli and Sun [11] is given by:

$$\begin{cases} m_k \sim p\left(m_k | m_{k-1}\right) & \text{(2.70a)} \\ \mathbf{z}_k = \mathcal{F}_{km_k}\left(\mathbf{z}_{k-1},\, \mathbf{u}_k,\, \mathbf{f}_{\mathbf{a}k-1}\right) + \boldsymbol{\eta}_k & \text{(2.70b)} \\ \mathbf{y}_k = \mathcal{H}_{km_k}\left(\mathbf{z}_k,\, \mathbf{f}_{\mathbf{s}k}\right) + \boldsymbol{\nu}_k & \text{(2.70c)} \end{cases}$$

where $\mathcal{F}_{km_k}(\cdot)$ represents the discrete dynamics of the state vector $\mathbf{z}$ associated with the mode m_k, and $\mathcal{H}_{km_k}(\cdot)$ represents the discrete measurement function of the state vector $\mathbf{z}$ associated with the mode m_k. Note that it is possible to consider that noises also depend on the current mode m_k.

The JMS representation allows considering several dynamics, even multiplicative and additive faults in the same system in both actuator and sensor, and it is the one used by multiple model architecture presented in Section 2.5.7.

The JMS approach requires then to store all the possible modes sequences from initial to current time step, for which the number grows exponentially with time. It is the major drawback of this method, which makes it not suitable for real-time implementation. Therefore, it cannot be used for fault estimation for the application considered in this thesis. To overcome this issue, a suboptimal algorithm that provides an approximation of these equations is used.

2.5.7.2 *The interacting multiple model framework*

The IMM is the most used architecture for hybrid state estimation [61–64], and is considered as a state-of-the-art approach for fault estimation applications. The IMM only considers the model at the current time-step, which makes it possible to implement it for real-time applications. This algorithm runs a bank of estimation filters, with different models. Each

estimation filter provides an estimate. These estimates are then mixed together to provide an overall estimate.

To provide an overall estimate, weights are associated with the estimate of each model. The weight $w_k^{(j)}$ that corresponds to the probability of the j^{th} model at time step k is given by:

$$w_k^{(j)} = \mathbb{P}\left(m_k^{(j)}|\mathbf{Y}_{1:k}\right) \tag{2.71a}$$

$$\propto p\left(\mathbf{y}_k|m_k^{(j)}\right)\mathbb{P}\left(m_k^{(j)}|\mathbf{Y}_{1:k-1}\right), \tag{2.71b}$$

where $\mathbb{P}\left(m_k^{(j)}|\mathbf{Y}_{1:k-1}\right)$ is given by:

$$\mathbb{P}\left(m_k^{(j)}|\mathbf{Y}_{1:k-1}\right) = \sum_{i=0}^{M-1}\mathbb{P}\left(m_k^{(j)}|m_{k-1}^{(i)}\right)\mathbb{P}\left(m_{k-1}^{(i)}|\mathbf{Y}_{1:k-1}\right) \tag{2.72a}$$

$$= \sum_{i=0}^{M-1}\pi_{ij}w_{k-1}^{(i)}. \tag{2.72b}$$

By substituting (2.72b) into (2.71b) it gives:

$$w_k^{(j)} \propto p\left(\mathbf{y}_k|m_k^{(j)}\right)\sum_{i=0}^{M-1}\pi_{ij}w_{k-1}^{(i)}. \tag{2.73}$$

The weights are normalized to ensure:

$$\sum_{j=0}^{M-1} w_k^{(j)} = 1. \tag{2.74}$$

The algorithm of the weight update of the multiple model architecture is given by the function WEIGHTUPDATE in Algorithm 2.9, where $\tilde{\mathbf{y}}_k^{(i)}$ denotes the innovation vector of the i^{th} model at time step k.

Algorithm 2.9 Weight update in multiple model architecture

Function WEIGHTUPDATE($w_k^{(0:M-1)}$, $\tilde{\mathbf{y}}_k^{(0:M-1)}$, $\mathbf{S}_k^{(0:M-1)}$)

  for each $i \in [0,\, M-1]$ **do**

    $\tilde{w}_k^{(i)} \leftarrow \mathcal{N}\left(\tilde{\mathbf{y}}_k^{(i)};\, 0,\, \mathbf{S}_k^{(i)}\right)\sum_{i=0}^{M-1}\pi_{ij}w_{k-1}^{(i)}$ //See (2.73)

  for each $i \in [0,\, M-1]$ **do**

    $w_k^{(i)} \leftarrow \frac{\tilde{w}_k^{(i)}}{\sum_{j=1}^{N}\tilde{w}_k^{(j)}}$ //Normalization of the weights, see (2.74)

Then, the overall estimate is given by:

$$\hat{\mathbf{x}}_k = \sum_{i=0}^{M-1} w_k^{(j)} \hat{\mathbf{x}}_k^{(j)}. \tag{2.75}$$

And its associated covariance is then:

$$\hat{\mathbf{P}}_k = \sum_{i=0}^{M-1} w_k^{(j)} \left[\hat{\mathbf{P}}_k^{(j)} + \left(\hat{\mathbf{x}}_k - \hat{\mathbf{x}}_k^{(j)} \right) \left(\hat{\mathbf{x}}_k - \hat{\mathbf{x}}_k^{(j)} \right)^\top \right]. \tag{2.76}$$

The algorithm of the multiple model estimate is given by the function MMESTIMATE in Algorithm 2.10.

Algorithm 2.10 Multiple model estimate

Function MMESTIMATE($\hat{\mathbf{x}}_k$, $\hat{\mathbf{P}}_k$, $\hat{\mathbf{x}}_k^{(0:M-1)}$, $w_k^{(0:M-1)}$)

$\hat{\mathbf{x}}_k = \sum_{i=0}^{M-1} w_k^{(i)} \hat{\mathbf{x}}_k^{(i)}$ //See (2.75)

$\hat{\mathbf{P}}_k \leftarrow \sum_{i=0}^{M-1} w_k^{(i)} \left[\hat{\mathbf{P}}_k^{(i)} + \left(\hat{\mathbf{x}}_k - \hat{\mathbf{x}}_k^{(i)} \right) \left(\hat{\mathbf{x}}_k - \hat{\mathbf{x}}_k^{(i)} \right)^\top \right]$ //See (2.76)

At the next iteration the filters are fed with the previous estimates given all the previous weights. These estimates are obtained through the mixing step and are used to compute the prior density $p\left(\mathbf{x}_{k-1} | m_k^{(j)}, \mathbf{Y}_{1:k-1}\right)$. To do so, all the possible modes at time step $k-1$ are considered and since the previous mode $m_{k-1}^{(i)}$ is known, the current mode $m_k^{(j)}$ is conditionally independent of the previous continuous state vector $\mathbf{x}_{k-1}$ and the all the previous measurement vectors $\mathbf{Y}_{1:k-1}$. Thus, using the law of total probability,

$$p\left(\mathbf{x}_{k-1} | m_k^{(j)}, \mathbf{Y}_{1:k-1}\right) = \sum_{i=0}^{M-1} p\left(\mathbf{x}_{k-1} | m_{k-1}^{(i)}, \mathbf{Y}_{1:k-1}\right) \mathbb{P}\left(m_{k-1}^{(i)} | m_k^{(j)}, \mathbf{Y}_{1:k-1}\right) \tag{2.77a}$$

$$= \sum_{i=0}^{M-1} w_{k-1}^{(i|j)} p\left(\mathbf{x}_{k-1} | m_{k-1}^{(i)}, \mathbf{Y}_{1:k-1}\right), \tag{2.77b}$$

where $w_{k-1}^{(i|j)}$ represents the mixed weights, and is given by:

$$w_{k-1}^{(i|j)} = \frac{\pi_{ij} w_{k-1}^{(i)}}{\sum_{i=0}^{M-1} \pi_{ij} w_{k-1}^{(i)}}, \tag{2.78}$$

Then, the mixed state vector is given by:

$$\hat{\mathbf{x}}_{k-1}^{(i|j)} = \sum_{i=0}^{M-1} w_{k-1}^{(i|j)} \hat{\mathbf{x}}_{k-1}^{(i)}, \tag{2.79}$$

and its associated covariance is given by:

$$\hat{\mathbf{P}}_{k-1}^{(i|j)} = \sum_{i=0}^{M-1} w_{k-1}^{(i|j)} \left[\hat{\mathbf{P}}_{k-1}^{(i)} + \left(\hat{\mathbf{x}}_{k-1}^{(i)} - \hat{\mathbf{x}}_{k-1}^{(i|j)} \right) \left(\hat{\mathbf{x}}_{k-1}^{(i)} - \hat{\mathbf{x}}_{k-1}^{(i|j)} \right)^{\top} \right]. \tag{2.80}$$

The algorithm of the mixing step of the IMM is given by the function MIXING in Algorithm 2.11.

Algorithm 2.11 Mixing step of the interacting multiple model

Function MIXING($\hat{\mathbf{x}}_{k-1}^{(i|j)}$, $\hat{\mathbf{P}}_{k-1}^{(i|j)}$, $w_{k-1}^{(0:M-1)}$)

  for each $j \in [0,\, M-1]$ **do**

    for each $i \in [0,\, M-1]$ **do**

      $w_{k-1}^{(i|j)} \leftarrow \frac{\pi_{ij} w_{k-1}^{(i)}}{\sum_{i=0}^{M-1} \pi_{ij} w_{k-1}^{(i)}}$ //See (2.78)

    $\hat{\mathbf{x}}_{k-1}^{(i|j)} \leftarrow \sum_{i=0}^{M-1} w_{k-1}^{(i|j)} \hat{\mathbf{x}}_{k-1}^{(i)}$ //See (2.79)

    $\hat{\mathbf{P}}_{k-1}^{(i|j)} \leftarrow \sum_{i=0}^{M-1} w_{k-1}^{(i|j)} \left[\hat{\mathbf{P}}_{k-1}^{(i)} + \left(\hat{\mathbf{x}}_{k-1}^{(i)} - \hat{\mathbf{x}}_{k-1}^{(i|j)} \right) \left(\hat{\mathbf{x}}_{k-1}^{(i)} - \hat{\mathbf{x}}_{k-1}^{(i|j)} \right)^{\top} \right]$ //See (2.80)

The second step is then the prediction of the continuous state vector $\mathbf{x}_{k-1}$, by computing the prior density denoted $p\left(\mathbf{x}_k | m_k^{(j)}, \mathbf{Y}_{1:k-1}\right)$, and is obtained by applying the prediction step from (2.22), which gives:

$$p\left(\mathbf{x}_k | m_k^{(j)}, \mathbf{Y}_{1:k-1}\right) = \int_{\mathbb{R}^{n_x}} p\left(\mathbf{x}_k | \mathbf{x}_{k-1}, m_k^{(j)}\right) p\left(\mathbf{x}_{k-1} | m_k^{(j)}, \mathbf{Y}_{1:k-1}\right) \mathrm{d}\mathbf{x}_{k-1}. \tag{2.81}$$

The next step which is the update step of the continuous state vector $\mathbf{x}_k$, aims to compute the posterior density denoted $p\left(\mathbf{x}_k | m_k^{(j)}, \mathbf{Y}_{1:k}\right)$, using (2.23) which then gives:

$$p\left(\mathbf{x}_k | m_k^{(j)}, \mathbf{Y}_{1:k}\right) = \frac{p\left(\mathbf{y}_k | \mathbf{x}_k, m_k^{(j)}\right) p\left(\mathbf{x}_k | m_k^{(j)}, \mathbf{Y}_{1:k-1}\right)}{\int_{\mathbb{R}^{n_x}} p\left(\mathbf{y}_k | \mathbf{x}_k, m_k^{(j)}\right) p\left(\mathbf{x}_k | m_k^{(j)}, \mathbf{Y}_{1:k-1}\right)}. \tag{2.82}$$

The IMM is given by the Algorithm 2.12. In this algorithm, the function ESTIMATIONFILTER refers to any compatible estimation filter, such as the ones previously introduced, that take as an input the previous state estimate $\hat{\mathbf{x}}_{k-1}$, its associated estimated covariance $\hat{\mathbf{P}}_{k-1}$, the control vector $\mathbf{u}_k$ and the measurement vector $\mathbf{y}_k$ and provide as output an estimated state $\hat{\mathbf{x}}_k$, its associated estimated covariance $\hat{\mathbf{P}}_k$, the innovation $\tilde{\mathbf{y}}_k$ to compute the likelihood, and its associated covariance $\hat{\mathbf{S}}_k$.

The architecture of the IMM is illustrated in Figure 2.18.

Algorithm 2.12 Interacting multiple model

$k \leftarrow 0$
⋮ //Initialization
Loop
  $k \leftarrow k + 1$
  MIXING($\hat{\mathbf{x}}_{k-1}^{(i|j)}$, $\hat{\mathbf{P}}_{k-1}^{(i|j)}$, $w_{k-1}^{(0:M-1)}$) //See Algorithm 2.11
  for each $i \in [0, M-1]$ **do**
    ESTIMATIONFILTER($\hat{\mathbf{x}}_k^{(i)}$, $\hat{\mathbf{P}}_k^{(i)}$, $\tilde{\mathbf{y}}_k^{(i)}$, $\mathbf{S}_k^{(i)}$, $\hat{\mathbf{x}}_{k-1}^{(i|j)}$, $\hat{\mathbf{P}}_{k-1}^{(i|j)}$, $\mathbf{u}_k$, $\mathbf{y}_k$) //i^{th} model estimation
  WEIGHTUPDATE($w_k^{(0:M-1)}$, $\tilde{\mathbf{y}}_k^{(i)}$, $\mathbf{S}_k^{(i)}$) //See Algorithm 2.9
  MMESTIMATE($\hat{\mathbf{x}}_k$, $\hat{\mathbf{P}}_k$, $\hat{\mathbf{x}}_k^{(0:M-1)}$, $w_k^{(0:M-1)}$) //See Algorithm 2.10

The IMM estimator with a bank of EKF yields significantly better fault detection performance than a stand-alone EKF, indeed IMM and multiple model architecture in general allows for better estimation of abrupt changes in the system dynamics [46], which is essential for this application, especially for the estimation of abrupt fault.

2.5.8 *Jump-Markov particle filters*

Particle filters for Markovian jump linear systems were introduced by Doucet, Gordon, and Krishnamurthy [10] and Tafazoli and Sun [11]. These filters aim to perform fault estimation and detection alongside state estimation using hybrid state vector where fault detections are modelled as transitions from nominal mode to faulty modes. The jump step of Doucet, Gordon, and Krishnamurthy [10] is designed to select particles from a proposal density, which must satisfy the following conditions:

- The support of the proposal density must contain the support of the posterior density;
- The proposal density must take into account recent observations.

In the jump step of Tafazoli and Sun [11], the particles are generated from a prior density. A mode selection probability is calculated as a function of the weights of all particles for all modes. The likeliest mode with the higher probability is selected.

The fixed-lag Rao-blackwelization particle filter was proposed by Giremus, Tourneret, and Calmettes [65]. This particle filter tackles the detection and estimation of multipath errors while inferring the vehicle dynamics. Multipath events were considered as abrupt changes affecting the navigation state space model.

Jump Markov particle filters were also developed for nonlinear systems by Driessen and Boers [66], where the user has control on the number of particles in order to avoid degeneracy, with application to radar target tracking. Due to their computational demand, those approaches are not well suited for real-time embedded applications.

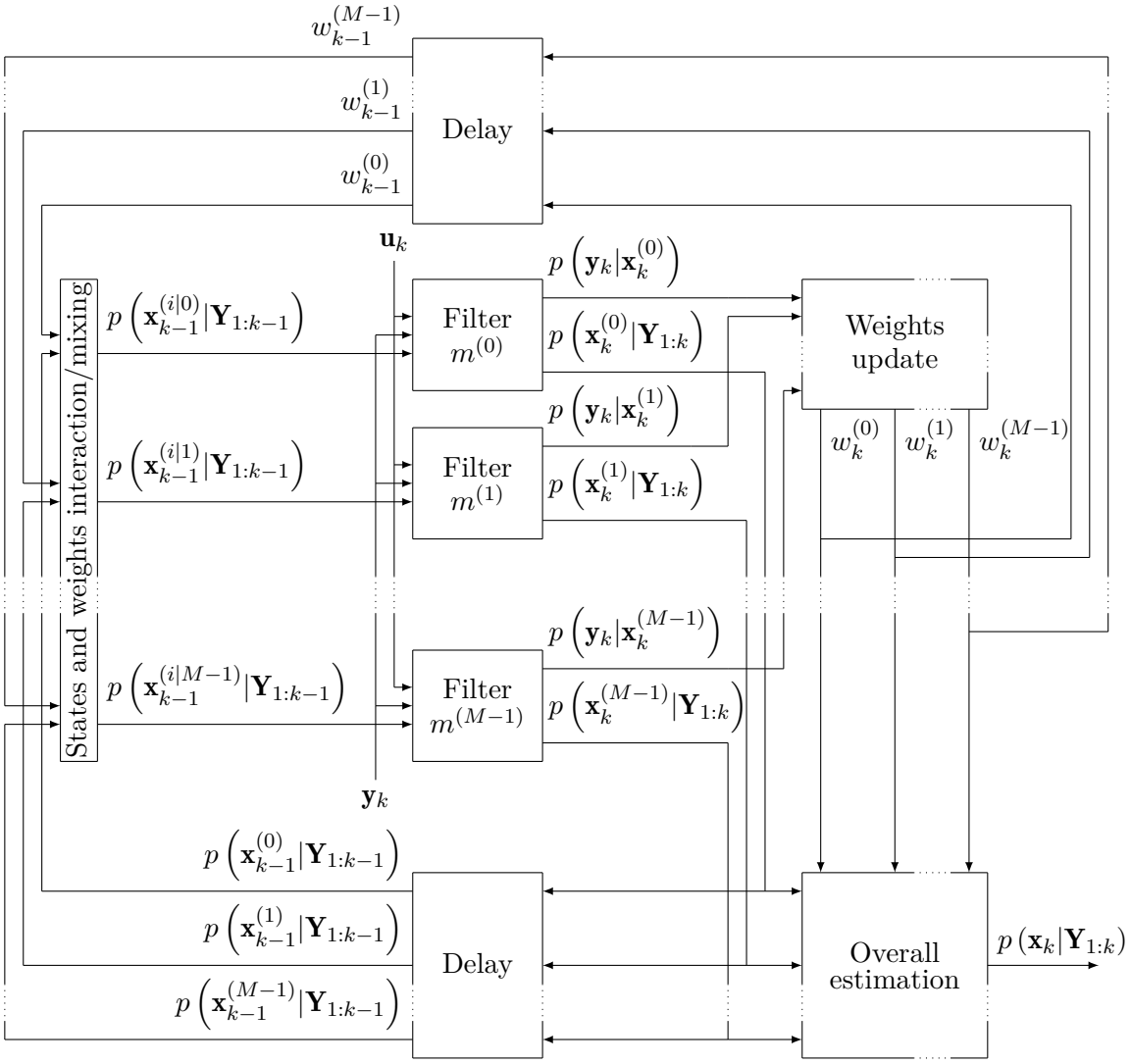


Figure 2.18: Architecture of the IMM.

2.6 CHAPTER SUMMARY

In this chapter, a literature review on fault diagnosis, including fault detection, isolation and estimation was presented. The focus was on model-based approaches because the model of the aircraft is known, and more data would often be required for a data driven approach. However, given the uncertainty on the fixed wing UAV model, a Bayesian approach is considered for fault and state estimation. This is performed by estimation filters. The most commonly used filters were described in this chapter for various situations. State-of-the-art multiple

model, such as the IMM were presented in more detail, to highlight the benefit of hybrid state estimation. The IMM is however not designed to deal with ambiguous faults. The RPF and other particle filters for JMS were also reviewed, but those approaches are either not suitable for real time applications or lack robustness. The limitations of the IMM and of the RPF will be further detailed in Chapter 4 and 5, respectively.

3

FIXED WING UNMANNED AERIAL VEHICLE DYNAMICS, GUIDANCE, NAVIGATION AND CONTROL

This chapter aims to present the model under consideration in this thesis. Without loss of generality, the focus of the thesis is on the guidance navigation & control (GNC) of UAV longitudinal dynamics.

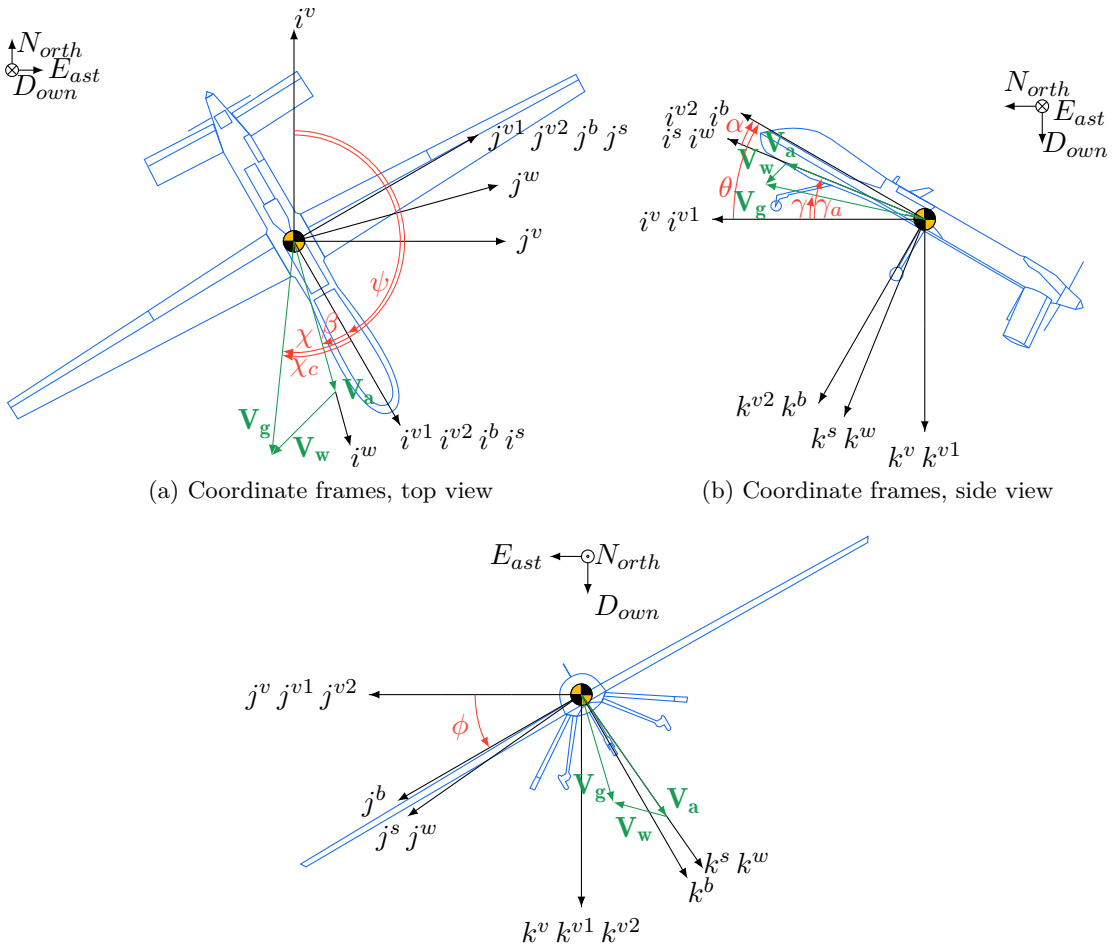
The chapter is organized as follows: Section 3.1 details the coordinate frame of the UAV used in the rest of the thesis. The Section 3.2 gives the aircraft kinematics and dynamics equations used. In Section 3.3, the forces and moments that are involved in the kinematics and dynamics equations are detailed. Section 3.4 provides the measurement equation of the UAV. The Section 3.5 provides a linearized model used in particular for the control of the UAV. In Section 3.6, the navigation of the UAV is explained. Section 3.7 describes the control of the UAV used. The Section 3.8 details the guidance of the UAV used. Section 3.9 summarize this chapter.

3.1 COORDINATE FRAMES

The coordinate frames [67] are defined in Figure 3.1. The $Z\,Y\,X$ Euler rotational sequence is used with unit vectors notations along the current axes of the current frame $i \equiv X$, $j \equiv Y$ and $k \equiv Z$.

In Figure 3.1, the black arrows are frame axes, the green arrows are vectors and the red arc are angles. The inertial frame axes (not represented here) are collinear with the North, East, Down (NED) axis and represent the origin of the position states. The superscripts v, $v1$ and $v2$ denote respectively the vehicle, vehicle 1 and vehicle 2 frames. The vehicle 1 and vehicle 2 frames are respectively obtained after the first and second rotations of the Euler sequence. The superscripts b denotes the body frame obtained after the full sequence of three rotations, the superscript s the stability frame and the superscript w the wind frame. The vector $\mathbf{V_a}$ denotes the airspeed vector, $\mathbf{V_w}$ represents the wind vector and $\mathbf{V_g}$ is the velocity vector. The angle ψ, θ, ϕ respectively denote the yaw, pitch and roll, β represents the side-slip angle, χ is the course angle, χ_c is the crab angle, α is the angle of attack, γ the flight path angle and γ_a the air-mass-referenced flight path angle. Finally, the yellow and black coloured disk represent the centre of mass.

In this thesis, the inertial frame is denoted F^i, the vehicle frame F^v, the vehicle 1 and 2 frames F^{v1} and F^{v2}, the body frame F^b, the stability frame F^s and the wind frame F^w.



(a) Coordinate frames, top view

(b) Coordinate frames, side view

(c) Coordinate frames, front view

Figure 3.1: Coordinate frames of the UAV.

3.2 AIRCRAFT KINEMATICS AND DYNAMICS

The kinematic and dynamic equations of the UAV are taken from reference [67]. These equations define the non-linear model of a UAV.

3.2.1 *State variables definitions*

The state variables used for the UAV kinematics and dynamics are described in table 3.1, with a differentiation between longitudinal and lateral variables. All these state variables concatenated constitute the state vector **z**.

Name	Description	Lateral	Longitudinal
p_n	Inertial North position of the UAV along i^i in F^i		
p_e	Inertial East position of the UAV along j^i in F^i		
p_d	Inertial Down position (negative of altitude) of the UAV measured along k^i in F^i		✓
u	Body frame velocity measured along i^b in F^b		✓
v	Body frame velocity measured along j^b in F^b	✓	
w	Body frame velocity measured along k^b in F^b		✓
ϕ	Roll angle defined with respect to F^{v2}	✓	
θ	Pitch angle defined with respect to F^{v1}		✓
ψ	Heading (yaw) angle defined with respect to F^v	✓	
p	Roll rate measured along i^b in F^b	✓	
q	Pitch rate measured along j^b in F^b		✓
r	Yaw rate measured along k^b in F^b	✓	

Table 3.1: State variables of the UAV

3.2.2 *Kinematics and dynamic equations*

3.2.2.1 *Rotational motion*

For rotational motion, Newton's second law states that:

$$\frac{d\mathbf{h}}{dt_i} = \frac{d\mathbf{h}}{dt_b} + \omega_{b/i}\mathbf{h} = M, \tag{3.1}$$

where $\frac{d}{dt_i}$ is the time derivative in the F^i, $\frac{d}{dt_b}$ is the time derivative in F^b, $\mathbf{h}$ is the angular momentum vector, M is the sum of all externally applied moments and $\omega_{b/i}$ denote the angular velocity of frame F^b with respect to F^i. As with translational motion, it is most convenient to express this equation in the body frame, giving:

$$\frac{d\mathbf{h}^b}{dt_b} + \omega^b_{b/i}\mathbf{h}^b = M^b. \tag{3.2}$$

where $\mathbf{h}^b$ is the angular momentum vector expressed in F^b.

For a rigid body, angular momentum is defined as the product of the inertia matrix $\mathbf{J}$ and the angular velocity vector $\mathbf{h}^b \triangleq \mathbf{J}\omega^b_{b/i}$, and rotational dynamics are described by the equation:

$$\mathbf{J}\frac{d\omega^b_{b/i}}{dt_b} + \omega^b_{b/i}\left(\mathbf{J}\omega^b_{b/i}\right) = M^b, \tag{3.3}$$

where:

$$\mathbf{J} = \begin{bmatrix} J_x & -J_{xy} & -J_{xz} \\ -J_{xy} & J_y & -J_{yz} \\ -J_{xz} & -J_{yz} & J_z \end{bmatrix}. \tag{3.4}$$

3.2.2.2 *Equations of motion in component form*

The equations of motion of the UAV are given by[1]:

$$\begin{bmatrix} \dot{p}_n \\ \dot{p}_e \\ \dot{p}_d \end{bmatrix} = \begin{bmatrix} c_{(\theta)}c_{(\psi)} & s_{(\phi)}s_{(\theta)}c_{(\psi)} - c_{(\phi)}s_{(\psi)} & c_{(\phi)}s_{(\theta)}c_{(\psi)} + s_{(\phi)}s_{(\psi)} \\ c_{(\theta)}s_{(\psi)} & s_{(\phi)}s_{(\theta)}s_{(\psi)} + c_{(\phi)}c_{(\psi)} & c_{(\phi)}s_{(\theta)}s_{(\psi)} - s_{(\phi)}c_{(\psi)} \\ -s_{(\theta)} & s_{(\phi)}c_{(\theta)} & c_{(\phi)}c_{(\theta)} \end{bmatrix} \begin{bmatrix} u \\ v \\ w \end{bmatrix} \tag{3.5a}$$

$$\begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{bmatrix} = \begin{bmatrix} rv - qw \\ pw - ru \\ qu - pv \end{bmatrix} + \frac{1}{\mathsf{m}} \begin{bmatrix} f_x \\ f_y \\ f_z \end{bmatrix} \tag{3.5b}$$

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & \sin(\phi)\tan(\theta) & \cos(\phi)\tan(\theta) \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \frac{\sin(\phi)}{\cos(\theta)} & \frac{\cos(\phi)}{\cos(\theta)} \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{3.5c}$$

$$\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} \Gamma_1 pq - \Gamma_2 qr \\ \Gamma_5 pr - \Gamma_6 \left(q^2 - r^2\right) \\ \Gamma_7 pq - \Gamma_1 qr \end{bmatrix} + \begin{bmatrix} \Gamma_3 l + \Gamma_4 n \\ \frac{1}{J_y} m \\ \Gamma_4 l + \Gamma_8 n \end{bmatrix} \tag{3.5d}$$

1 For the position equation the function $\cos(\cdot)$ is denoted $c_{(\cdot)}$ and $\sin(\cdot)$ is denoted $s_{(\cdot)}$

where m denotes the mass of the aircraft, $\begin{bmatrix} f_x & f_y & f_z \end{bmatrix}^\top$ and $\begin{bmatrix} l & m & n \end{bmatrix}^\top \triangleq M^b$ are the externally applied forces and moments on the UAV about the i^b, j^b, and k^b axes, and Γ_1 to Γ_8 are product of the inertia terms, given by:

$$\Gamma_1 = \frac{J_{xz}\left(J_x - J_y + J_z\right)}{J_x J_z - J_{xz}^2} \tag{3.6a}$$

$$\Gamma_2 = \frac{J_z\left(J_z - J_y\right) + J_{xz}^2}{J_x J_z - J_{xz}^2} \tag{3.6b}$$

$$\Gamma_3 = \frac{J_z}{J_x J_z - J_{xz}^2} \tag{3.6c}$$

$$\Gamma_4 = \frac{J_{xz}}{J_x J_z - J_{xz}^2} \tag{3.6d}$$

$$\Gamma_5 = \frac{J_z - J_x}{J_y} \tag{3.6e}$$

$$\Gamma_6 = \frac{J_{xz}}{J_y} \tag{3.6f}$$

$$\Gamma_7 = \frac{\left(J_x - J_y\right) J_x + J_{xz}^2}{J_x J_z - J_{xz}^2} \tag{3.6g}$$

$$\Gamma_8 = \frac{J_x}{J_x J_z - J_{xz}^2}, \tag{3.6h}$$

The equations of motion (3.5) are the components of $\mathcal{F}(\cdot)$ the non-linear dynamics of the state vector $\mathbf{z}$.

3.2.2.3 *Equations of decoupled*

The lateral equations of motion of the UAV are given by:

$$\begin{bmatrix} \dot{v} \end{bmatrix} = \frac{1}{\mathsf{m}} \begin{bmatrix} f_y \end{bmatrix} \tag{3.7a}$$

$$\begin{bmatrix} \dot{\phi} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} p \\ \cos(\phi) r \end{bmatrix} \tag{3.7b}$$

$$\begin{bmatrix} \dot{p} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} \Gamma_3 l + \Gamma_4 n \\ \Gamma_4 l + \Gamma_8 n \end{bmatrix} \tag{3.7c}$$

The longitudinal equations of motion of the UAV are given by:

$$\begin{bmatrix}\dot{p}_d\end{bmatrix} = \begin{bmatrix}-\sin(\theta)u + \cos(\theta)w\end{bmatrix} \tag{3.8a}$$

$$\begin{bmatrix}\dot{u}\\ \dot{w}\end{bmatrix} = \begin{bmatrix}qw\\ qu\end{bmatrix} + \frac{1}{\mathsf{m}}\begin{bmatrix}f_x\\ f_z\end{bmatrix} \tag{3.8b}$$

$$\begin{bmatrix}\dot{\theta}\end{bmatrix} = \begin{bmatrix}q\end{bmatrix} \tag{3.8c}$$

$$\begin{bmatrix}\dot{q}\end{bmatrix} = \begin{bmatrix}\Gamma_6\left(q^2\right)\end{bmatrix} + \begin{bmatrix}\frac{1}{J_y}m\end{bmatrix} \tag{3.8d}$$

3.3 FORCES AND MOMENTS

The forces and moments equations of the UAV are taken from reference [67]. This section describes the forces $\begin{bmatrix}f_x & f_y & f_z\end{bmatrix}^\top$ and moments $\begin{bmatrix}l & m & n\end{bmatrix}^\top$ that act on the UAV dynamics.

The control inputs of the UAV are described in table 3.2 with a differentiation between lateral and longitudinal control inputs. All the control inputs concatenated constitute the control input vector $\mathbf{u}$.

Name	Description	Lateral	Longitudinal
δ_e	Elevator deflection		✓
δ_a	Aileron deflection	✓	
δ_r	Rudder deflection	✓	
δ_t	Throttle input		✓

Table 3.2: Control inputs of the UAV

The total forces along the body axes of the UAV can be written as follows:

$$\begin{aligned}\begin{bmatrix}f_x\\ f_y\\ f_z\end{bmatrix} = &\begin{bmatrix}-\mathsf{m}g\sin(\theta)\\ \mathsf{m}g\cos(\theta)\sin(\phi)\\ \mathsf{m}g\cos(\theta)\cos(\phi)\end{bmatrix} +\\ &\frac{\rho\mathbf{V_a}^2 S}{2}\begin{bmatrix}C_X(\alpha) + C_{X_q}(\alpha)\frac{c}{2\mathbf{V_a}}q + C_{X_{\delta_e}}(\alpha)\delta_e\\ C_{Y_0} + C_{Y_\beta}\beta + C_{Y_p}\frac{b}{2\mathbf{V_a}}p + C_{Y_r}\frac{b}{2\mathbf{V_a}}r + C_{Y_{\delta_a}}\delta_a + C_{Y_{\delta_r}}\delta_r\\ C_Z(\alpha) + C_{Z_q}(\alpha)\frac{c}{2\mathbf{V_a}}q + C_{Z_{\delta_e}}(\alpha)\delta_e\end{bmatrix} +,\\ &\frac{\rho S_{prop}C_{prop}}{2}\begin{bmatrix}(k_{motor}\delta_t)^2 - \mathbf{V_a}^2\\ 0\\ 0\end{bmatrix}\end{aligned} \tag{3.9}$$

where g is the gravitational acceleration, ρ the density of air, S the surface area of the wing, c the mean aerodynamic chord of the wing, b the wing span, S_{prop} the area of the propeller, C_{prop} the aerodynamic coefficient of the propeller, k_{motor} the constant that specify the efficiency of the motor, and C_{Y_0}, C_{Y_β}, C_{Y_p}, C_{Y_r}, $C_{Y_{\delta_a}}$ and $C_{Y_{\delta_r}}$ are aerodynamic force coefficients along the j^b axis. All these parameters except g and ρ that are independent of the aircraft are given in Appendix A.1. Finally, $C_X(\cdot)$, $C_{X_q}(\cdot)$ and $C_{X_{\delta_e}}(\cdot)$ are aerodynamic force coefficients along the i^b axis, and $C_Z(\cdot)$, $C_{Z_q}(\cdot)$ and $C_{Z_{\delta_e}}(\cdot)$ are aerodynamic force coefficients along the k^b axis. There are given by:

$$C_X(\alpha) \triangleq -C_D(\alpha)\cos(\alpha) + C_L(\alpha)\sin(\alpha) \tag{3.10a}$$
$$C_{X_q}(\alpha) \triangleq -C_{D_q}\cos(\alpha) + C_{L_q}\sin(\alpha) \tag{3.10b}$$
$$C_{X_{\delta_e}}(\alpha) \triangleq -C_{D_{\delta_e}}\cos(\alpha) + C_{L_{\delta_e}}\sin(\alpha) \tag{3.10c}$$
$$C_Z(\alpha) \triangleq -C_D(\alpha)\sin(\alpha) - C_L(\alpha)\cos(\alpha) \tag{3.10d}$$
$$C_{Z_q}(\alpha) \triangleq -C_{D_q}\sin(\alpha) - C_{L_q}\cos(\alpha) \tag{3.10e}$$
$$C_{Z_{\delta_e}}(\alpha) \triangleq -C_{D_{\delta_e}}\sin(\alpha) - C_{L_{\delta_e}}\cos(\alpha), \tag{3.10f}$$

where $C_D(\cdot)$, C_{D_q} and $C_{D_{\delta_e}}$ are aerodynamic drag coefficients, and $C_L(\cdot)$, C_{L_q} and $C_{D_{\delta_e}}$ are lift coefficients. There are given in Appendix A.1 except for $C_D(\cdot)$ and $C_L(\cdot)$ which are given by:

$$C_L(\alpha) = (1-\sigma(\alpha))\left[C_{L_0} + C_{L_\alpha}\alpha\right] + \sigma(\alpha)\left[2\,\mathrm{sign}(\alpha)\sin(\alpha)^2\cos(\alpha)\right] \tag{3.11a}$$
$$C_D(\alpha) = C_{D_p} + \frac{C_L(\alpha)^2}{\pi e AR} \tag{3.11b}$$

where C_{L_0} and C_{L_α} are aerodynamic lift coefficients, C_{D_p} is an aerodynamic drag coefficient, e is the Oswald efficient factor, and AR is the wing aspect ratio. There are given in Appendix A.1. Finally, $\sigma(\cdot)$ is a sigmoid function given by:

$$\sigma(\alpha) = \frac{1 + e^{-\mathsf{M}(\alpha-\alpha_0)} + e^{-\mathsf{M}(\alpha+\alpha_0)}}{\left(1+e^{-\mathsf{M}(\alpha-\alpha_0)}\right)\left(1+e^{\mathsf{M}(\alpha+\alpha_0)}\right)}, \tag{3.12}$$

where M is a positive constant, and α_0 is the stalling angle of attack. There are given in Appendix A.1.

A nominal flight is conducted with an angle of attack bounded between $\pm\alpha_0$. Above this limit, the aircraft is in stall. This phenomenon is illustrated on the Figure 3.2. Figure 3.2 also illustrates that the lift and drag equation can be approximated by a linear and second order function if the angle of attack is bounded by α_0.

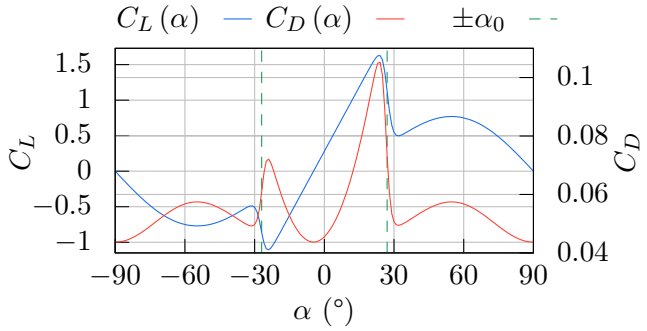


Figure 3.2: Lift coefficient (C_L) and drag coefficient (C_D) versus the angle of attack α, with parameters from Appendix A.1

The total torque vector of the UAV can be written as follows:

$$\begin{bmatrix} l \\ m \\ n \end{bmatrix} = \frac{\rho \mathbf{V_a}^2 S}{2} \begin{bmatrix} b\left(C_{l_0} + C_{l_\beta}\beta + C_{l_p}\frac{b}{2\mathbf{V_a}}p + C_{l_r}\frac{b}{2\mathbf{V_a}}r + C_{l_{\delta_a}}\delta_a + C_{l_{\delta_r}}\delta_r\right) \\ c\left(C_{m_0} + C_{m_\alpha}\alpha + C_{m_q}\frac{c}{2\mathbf{V_a}}q + C_{m_{\delta_e}}\delta_e\right) \\ b\left(C_{n_0} + C_{n_\beta}\beta + C_{n_p}\frac{b}{2\mathbf{V_a}}p + C_{n_r}\frac{b}{2\mathbf{V_a}}r + C_{n_{\delta_a}}\delta_a + C_{n_{\delta_r}}\delta_r\right) \end{bmatrix} + \begin{bmatrix} -k_{T_p}(k_\omega \delta_t)^2 \\ 0 \\ 0 \end{bmatrix} \tag{3.13}$$

where C_{l_0}, C_{l_β}, C_{l_p}, C_{l_r}, $C_{l_{\delta_a}}$ and $C_{l_{\delta_r}}$ are aerodynamic moments coefficients along the i^b axis, C_{m_0}, C_{m_α}, C_{m_q}, and $C_{m_{\delta_e}}$ are aerodynamic moments coefficients along the j^b axis, C_{n_0}, C_{n_β}, C_{n_p}, C_{n_r}, $C_{n_{\delta_a}}$ and $C_{n_{\delta_r}}$ are aerodynamic moments coefficients along the k^b axis, and k_{T_p} and k_ω are motor constants. All these parameters are given in Appendix A.1.

3.4 SENSORS

The typical sensors used for the GNC of a UAV [67] are:

- Rate gyros;
- Accelerometers;
- Pressure sensors;
- Digital compass;
- GNSS receiver.

Note that most of the sensors in embedded systems are usually digital sensors. This means that their accuracies depend on the resolution of the analog-to-digital converter. The resolution herein is assumed to be negligible, and therefore it is not considered.

3.4.1 *Rate gyros*

In small aircraft, vibratory and micro-electromechanical system (MEMS) rate gyros are commonly used, and they typically operate based on the principle of the Coriolis acceleration. Then, equations of gyros rates are given by:

$$\begin{bmatrix} y_{gyro,p} \\ y_{gyro,q} \\ y_{gyro,r} \end{bmatrix} = \begin{bmatrix} p \\ q \\ r \end{bmatrix} + \begin{bmatrix} \nu_{gyro,p} \\ \nu_{gyro,q} \\ \nu_{gyro,r} \end{bmatrix}, \tag{3.14}$$

where $\nu_{gyro,p}$, $\nu_{gyro,q}$ and $\nu_{gyro,r}$ are Gaussian processes with variance $\sigma^2_{gyro,p}$, $\sigma^2_{gyro,q}$ and $\sigma^2_{gyro,r}$, respectively, and means $\mu_{gyro,p}$, $\mu_{gyro,q}$ and $\mu_{gyro,r}$, respectively. The gyros rate $y_{gyro,p}$, $y_{gyro,q}$ and $y_{gyro,r}$ are expressed in $\mathrm{rad\,s^{-1}}$. The attitude state vector, can be expressed by integration of the angle rates, that yields:

$$\begin{bmatrix} y_{gyro,\phi} \\ y_{gyro,\theta} \\ y_{gyro,\psi} \end{bmatrix} = \begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix} + \begin{bmatrix} \nu_{gyro,\phi} \\ \nu_{gyro,\theta} \\ \nu_{gyro,\psi} \end{bmatrix}, \tag{3.15}$$

where the measurement noise $\nu_{gyro,\phi}$, $\nu_{gyro,\theta}$ and $\nu_{gyro,\psi}$ depend respectively on the noise in the measurements of p, q and r.

For low-cost MEMS gyros, drift in the bias term can be significant and care must be taken to zero the gyro bias periodically during flight. This is done by flying a straight and level path and resetting the gyro bias so that $y_{gyro,p}$, $y_{gyro,q}$ and $y_{gyro,r}$ averages zero over a period of 100 or so samples. Moreover, the biases will never be perfectly estimated, and non-zero biases must be expected especially for the attitude computation where the biases are integrated. Then, a drift should be expected.

3.4.2 *Accelerometers*

The measured acceleration is the total acceleration minus gravity. The equations of accelerometers are:

$$\begin{bmatrix} y_{accel,x} \\ y_{accel,y} \\ y_{accel,z} \end{bmatrix} = \begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{bmatrix} + \begin{bmatrix} qw - rv \\ ru - pw \\ pv - qu \end{bmatrix} - g \begin{bmatrix} -\sin(\theta) \\ \sin(\phi)\cos(\theta) \\ \cos(\phi)\cos(\theta) \end{bmatrix} + \begin{bmatrix} \nu_{accel,x} \\ \nu_{accel,y} \\ \nu_{accel,z} \end{bmatrix}, \tag{3.16}$$

where $\nu_{accel,x}$, $\nu_{accel,y}$ and $\nu_{accel,z}$ are Gaussian processes with variance $\sigma^2_{accel,x}$, $\sigma^2_{accel,y}$ and $\sigma^2_{accel,z}$ respectively and means $\mu_{accel,x}$, $\mu_{accel,y}$ and $\mu_{accel,z}$ respectively. The accelerations $y_{accel,x}$, $y_{accel,y}$ and $y_{accel,z}$ are expressed in $\mathrm{m\,s^{-2}}$. Each accelerometer measures elements of linear acceleration, Coriolis acceleration, and gravitational acceleration.

The measurement equation to observe the velocity state vector can be expressed by integrating acceleration terms from (3.16) which yields:

$$\begin{bmatrix} y_{accel,u} \\ y_{accel,v} \\ y_{accel,w} \end{bmatrix} = \begin{bmatrix} u \\ v \\ w \end{bmatrix} + \begin{bmatrix} \nu_{accel,u} \\ \nu_{accel,v} \\ \nu_{accel,w} \end{bmatrix}, \tag{3.17}$$

where the measurement noise $\nu_{accel,u}$, $\nu_{accel,v}$ and $\nu_{accel,w}$ depend respectively on the noise on the measurements of $\dot{u}$, $\dot{v}$ and $\dot{w}$.

The accelerometers can then be used to derive pseudo-measures of the velocity if the angle rate and θ and ϕ are known. However, even with a good calibration, the biases are never fully removed, resulting in a drift of the velocity measurement.

3.4.3 *Pressure sensors*

Pressure sensors can provide pseudo measurements of altitude, with an absolute pressure sensor, and airspeed with a differential pressure sensor. Herein only an absolute pressure sensor is considered.

An absolute pressure sensor — that is a barometer — measures the atmospheric pressure. The equation of the sensor is given by:

$$y_{baro} = P + \nu_{baro}, \tag{3.18}$$

where P is the pressure measured by the barometer, and ν_{baro} is the Gaussian noise with variance σ^2_{baro} and mean μ_{baro} which is a temperature-related bias drift.

In the troposphere, typically below altitude of 11 000 m above sea level (ASL), the pressure of the atmosphere can be calculated using the barometric formula:

$$P = P_0 \left[\frac{T_0}{T_0 - p_d L_0} \right]^{\frac{g_n M}{R L_0}}, \tag{3.19}$$

where P_0 is the standard pressure at sea level, T_0 is the standard temperature at sea level, L_0 is the lapse rate of the temperature decrease in the lower atmosphere, g_n is the standard acceleration of gravity, R the molar gas constant, and M is the standard molar mass of the atmospheric air. All these values are given in Table B.1 Then, the measurement equation depending on the position down state can be written as:

$$y_{baro} = P_0 \left[\frac{T_0}{T_0 - p_d L_0} \right]^{\frac{g_n M}{R L_0}} + \nu_{baro}. \tag{3.20}$$

Then, the measurement of the position state p_d can be expressed as:

$$y_{baro_{nl},-p_d} = \left[\frac{T_0}{\left(\frac{y_{baro} - \nu_{baro}}{P_0} \right)^{\frac{R L_0}{g_n M}}} - T_0 \right] \frac{1}{L_0} \tag{3.21}$$

The measurement equation is nonlinear[2]. It can be approximated for sake of simplicity by a linearized measurement equation as:

$$y_{baro,-p_d} = -\frac{y_{baro} - \nu_{baro} - P_0}{\rho g_n} \tag{3.22}$$

The air density at a specific altitude can be computed by the formula:

$$\rho = \frac{MP}{RT} \tag{3.23}$$

where T is the local temperature parameters. If the model is linearized at 0 m of altitude then the error with the nonlinear model at 100 m is about 0.5 m and 12 m at 500 m. The linearized model at 0 m and the nonlinear model are shown in Figure 3.3.

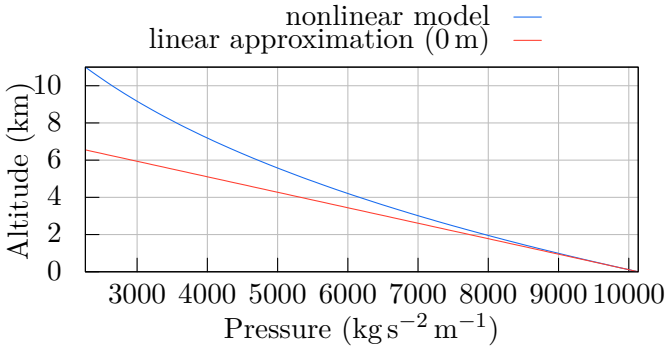


Figure 3.3: Altitude versus pressure with nonlinear and linear model

Then, the measurement equation depending on the position down state can be written as:

$$y_{baro,-p_d} = -p_d + \frac{\nu_{baro}}{\rho g_n} \tag{3.24a}$$

$$= -p_d + \nu_{baro,-p_d} \tag{3.24b}$$

3.4.4 *Digital compass*

A digital compass (magnetometer) measures the direction field locally and provides an indication of heading relative to the magnetic North ψ_m. However, ψ is the heading relative to the geographical North. There is a declination δ_m between the geographical and magnetic North, which depends on the location on Earth. The heading is then the sum of the magnetic heading measurement and the declination angle. The measurement equation of the digital compass is given by:

$$y_{mag} = \psi_m + \delta_m + \nu_{mag} \tag{3.25}$$

2 The subscript nl is used here to distinguish the nonlinear and the linear equation

where ν_{mag} is the Gaussian noise with variance σ^2_{mag} and mean μ_{mag}.

Given that the heading is given by $\psi = \psi_m + \delta_m$, it also gives:

$$y_{mag} = \psi + \nu_{mag} \tag{3.26}$$

3.4.5 *Global navigation satellite system receiver*

A GNSS receiver uses one or multiple satellite constellations to provide a 3-D position information on or near the Earth's surface.

3.4.5.1 *Position*

By measuring the time of flight of a minimum of four satellites (at least three for the trilateration and one for the receiver clocks synchronization and the need to resolve the ambiguity linked to the fact that there are often two possible points of intersection between 3 spheres). Then, the measurement equation of the GNSS receiver for the position in Earth coordinates and altitude ASL are:

$$\begin{bmatrix} y_{GNSS,p_n} \\ y_{GNSS,p_e} \\ y_{GNSS,-p_d} \end{bmatrix} = \begin{bmatrix} p_n \\ p_e \\ -p_d \end{bmatrix} + \begin{bmatrix} \nu_{GNSS,p_n} \\ \nu_{GNSS,p_e} \\ \nu_{GNSS,-p_d} \end{bmatrix}, \tag{3.27}$$

where ν_{GNSS,p_n}, ν_{GNSS,p_e} and $\nu_{GNSS,-p_d}$ are the error model of the North East and Altitude position respectively. An error model is necessary for the GNSS because there are multiple measurements errors sources. Kaplan and Hegarty [68] characterize the GNSS error solution by:

$$\text{(error in GNSS solution)} = \text{(pseudo-range error factor)}\,\text{(geometry factor)} \tag{3.28}$$

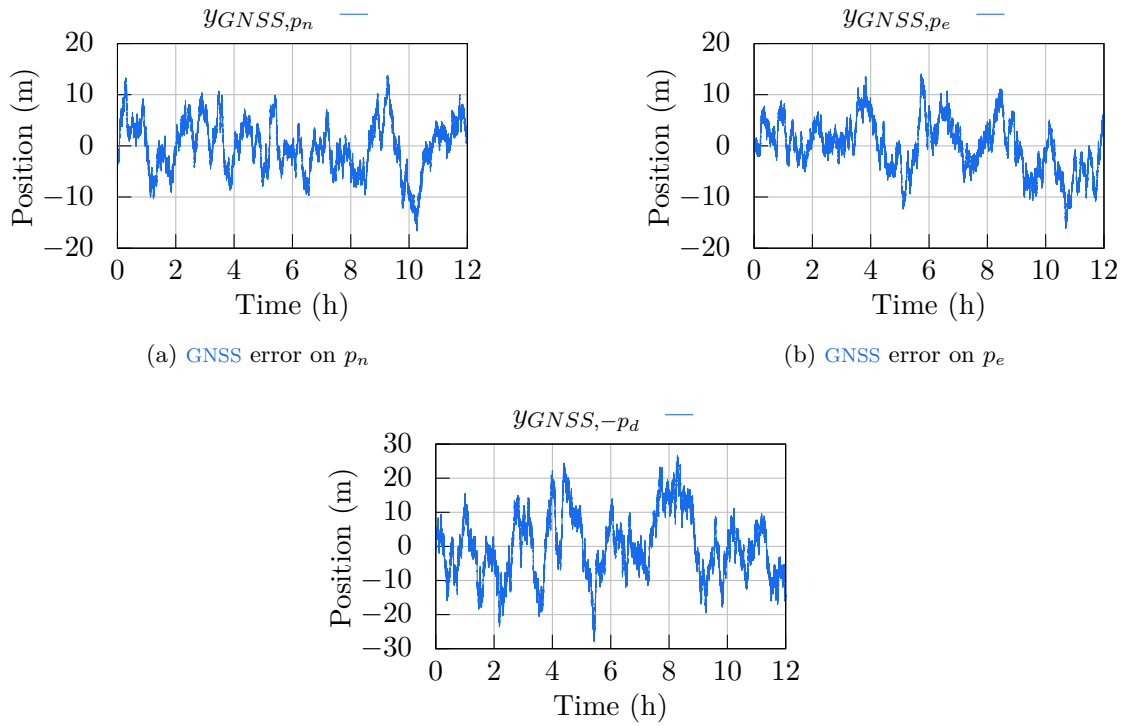
The GNSS pseudo-range error factor for a dual-frequency receiver is described by Kaplan and Hegarty [68] and Spilker Jr et al. [69]. It is due to the accuracy of the satellite clock and the ephemeris data, various atmospheric effect, multipath at the reception, and the receiver noise and its resolution. The cumulative effect of each of these errors sources on the pseudo-range measurement is called the user-equivalent range error (UERE).

The GNSS geometry factor is the satellite/user geometry effect on the GNSS solution error. It is generically called the dilution of precision (DOP). The DOP is composed of the vertical dilution of precision (VDOP) and the horizontal dilution of precision (HDOP). The terms describe the receiver location error due to the satellite location on the constellation.

To model the transient behaviour of the GNSS error [70], a Gauss-Markov process is used:

$$\nu_{k+1} = e^{-k_{GNSS}T_s}\nu_k + \eta_{GNSS}, \tag{3.29}$$

where k_{GNSS} is the frequency response of the Gauss-Markov process, T_s is the sample time and η_{GNSS} is a zero-mean white Gaussian noise with a standard deviation σ_{GNSS}. The



(a) GNSS error on p_n

(b) GNSS error on p_e

(c) GNSS error on $-p_d$

Figure 3.4: Example of GNSS position error simulated over 12 h with the model described in (3.29) and parameters given by Table B.2

Figure 3.4 shows an example of the GNSS error with the model described in (3.29) and parameters given by Table B.2.

However, some estimation methods described in Section 2.5 can only process a Gaussian noise. Then, to be able to use these methods, a Gaussian approximation of this noise can be performed.

3.4.5.2 *Velocity*

Using carrier phase Doppler measurements from the GNSS signal, a GNSS receiver computes its velocity with a standard deviation. Then, the measurement equation of the GNSS receiver for the velocity in the inertial frame is given by:

$$\begin{bmatrix} y_{GNSS,u} \\ y_{GNSS,v} \\ y_{GNSS,-w} \end{bmatrix} = \mathbf{R}_i^b \begin{bmatrix} V_n \\ V_e \\ -V_d \end{bmatrix} + \begin{bmatrix} \nu_{GNSS,u} \\ \nu_{GNSS,v} \\ \nu_{GNSS,-w} \end{bmatrix}, \tag{3.30}$$

where V_n, V_e and V_d are velocity components of the North, East and downward directions given by the GNSS with respect to the inertial frame, $\mathbf{R}_i^b$ denotes the rotation matrix of a rotation from inertial coordinate frame to body coordinate frame and $\nu_{GNSS,u}$, $\nu_{GNSS,v}$ and $\nu_{GNSS,w}$ are a zero mean white Gaussian noise with a standard deviation $\sigma_{GNSS,u}$, $\sigma_{GNSS,v}$ and $\sigma_{GNSS,w}$ respectively. Equation 3.30 can then be rewritten as:

$$\begin{bmatrix} y_{GNSS,u} \\ y_{GNSS,v} \\ y_{GNSS,-w} \end{bmatrix} = \begin{bmatrix} u \\ v \\ -w \end{bmatrix} + \begin{bmatrix} \nu_{GNSS,u} \\ \nu_{GNSS,v} \\ \nu_{GNSS,-w} \end{bmatrix} \tag{3.31}$$

3.4.6 *Measurement model*

The main navigation sensors used by UAV were described above. All the measurement equations are linear or can be approximated by a linear function. Then, the measurement model can be written as follows:

$$\mathbf{y} = \mathbf{H}\mathbf{z} + \boldsymbol{\nu}. \tag{3.32}$$

In this thesis only the linear measurement model is considered. Therefore:

$$\mathbf{H}\mathbf{z} + \boldsymbol{\nu} \equiv \mathcal{H}(\mathbf{z}) + \boldsymbol{\nu} \equiv \mathcal{H}_k(\mathbf{z}_k) + \boldsymbol{\nu}_k, \tag{3.33}$$

with $\mathcal{H}(\cdot)$ the continuous measurement function of the state vector $\mathbf{z}$.

Then, from (3.17) to (3.31), the measurement model is given by:

$$\begin{bmatrix} y_{GNSS,p_n} \\ y_{GNSS,p_e} \\ y_{GNSS,-p_d} \\ y_{baro,-p_d} \\ y_{GNSS,u} \\ y_{GNSS,v} \\ y_{GNSS,-w} \\ y_{accel,u} \\ y_{accel,v} \\ y_{accel,w} \\ y_{mag,\psi} \\ y_{gyro,\psi} \\ y_{gyro,\theta} \\ y_{gyro,\phi} \\ y_{gyro,p} \\ y_{gyro,q} \\ y_{gyro,r} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \mathbf{z} + \begin{bmatrix} \nu_{GNSS,p_n} \\ \nu_{GNSS,p_e} \\ \nu_{GNSS,-p_d} \\ \nu_{baro,-p_d} \\ \nu_{GNSS,u} \\ \nu_{GNSS,v} \\ \nu_{GNSS,-w} \\ \nu_{accel,u} \\ \nu_{accel,v} \\ \nu_{accel,w} \\ \nu_{mag,\psi} \\ \nu_{gyro,\psi} \\ \nu_{gyro,\theta} \\ \nu_{gyro,\phi} \\ \nu_{gyro,p} \\ \nu_{gyro,q} \\ \nu_{gyro,r} \end{bmatrix} \tag{3.34}$$

The lateral and longitudinal measurement model can easily be obtained from (3.34) by separately considering longitudinal and lateral measurement equation.

3.4.6.1 *Longitudinal state-space model*

For the longitudinal state-space model, the state vector is $\mathbf{z}_{lon} = \begin{bmatrix} p_d & u & w & \theta & q \end{bmatrix}^\top$ and the longitudinal state-space model is given by:

$$\mathbf{y}_{lon} = \mathbf{Y}_{lon}\mathbf{H}\mathbf{Y}_{lon}^\top \mathbf{z}_{lon} + \mathbf{Y}_{lon}\boldsymbol{\nu}, \tag{3.35}$$

where

$$\mathbf{Y}_{lon} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}. \tag{3.36}$$

3.5 LINEARIZED MODEL

The non-linear equations of motion with Euler angle representation are described in (3.5), (3.9) and (3.13).

However, these equations can be linearized at trim points, enabling the use of linear control laws. To further simplify the problem, the longitudinal and lateral states are decoupled.

To linearize the flight dynamics, a trim point must be chosen inside the flight envelope to ensure that the UAV can maintain flight at the chosen trim point.

3.5.1 *Trim conditions*

A non-linear system described by the differential equations $\dot{\mathbf{z}} = \mathcal{F}(\mathbf{z}, \mathbf{u})$, the system is said to be in equilibrium (or trimmed for aerospace vehicles) at the state $\mathbf{z}^*$ and input $\mathbf{u}^*$ if $\mathcal{F}(\mathbf{z}^*, \mathbf{u}^*) = 0$. Letting $\bar{\mathbf{z}} \equiv \mathbf{z} - \mathbf{z}^*$ it gives:

$$\dot{\bar{\mathbf{z}}} = \dot{\mathbf{z}} - \dot{\mathbf{z}}^* = \mathcal{F}(\mathbf{z}, \mathbf{u}) - \mathcal{F}(\mathbf{z}^*, \mathbf{u}^*) = \mathcal{F}(\mathbf{z}^* + \bar{\mathbf{z}}, \mathbf{u}^* + \bar{\mathbf{u}}) - \mathcal{F}(\mathbf{z}^*, \mathbf{u}^*) \tag{3.37}$$

Taking the Taylor series expansion of the first term about the trim state, gives:

$$\dot{\bar{\mathbf{z}}} \approx \frac{\partial \mathcal{F}(\mathbf{z}^*, \mathbf{u}^*)}{\partial \mathbf{z}}\bar{\mathbf{z}} + \frac{\partial \mathcal{F}(\mathbf{z}^*, \mathbf{u}^*)}{\partial \mathbf{u}}\bar{\mathbf{u}} \tag{3.38}$$

In the computation of trimmed state, it is assumed that the wind speed is zero, in other words $\mathbf{V_a} = \mathbf{V_g}$, $\psi = \chi$, and $\gamma = \gamma_a$. The trim states and inputs are computed when the aircraft simultaneously satisfies the following three conditions:

- It is travelling at a constant speed $\mathbf{V_a}^*$;
- It is climbing at a constant flight path angle of γ^*;
- It is in a constant orbit with radius of turn $\mathcal{R}^*$, where $\mathcal{R}^* \in [\mathcal{R}_{min}, +\infty)$ and $\mathcal{R}_{min}$ is the minimum turning radius of the aircraft. A turning radius equal to $+\infty$ represents a straight flight.

Note that the right-hand side of equations (3.5), (3.9) and (3.13) are independent of the position components p_n, p_e and p_d. If the trimmed flight condition is a constant climb, it gives: $\dot{\psi}^* = \frac{\mathbf{V_a}^*}{\mathcal{R}^*}\cos(\gamma^*)$ and $\dot{p}_d^* = -\mathbf{V_a}^* \sin(\gamma^*)$ and the new equation to satisfy $\mathcal{F}(\mathbf{z}^*, \mathbf{u}^*) = 0$ is:

$$\dot{\mathbf{z}}^* = \begin{bmatrix} \dot{p}_n^* \\ \dot{p}_e^* \\ \dot{p}_d^* \\ \dot{u}^* \\ \dot{v}^* \\ \dot{w}^* \\ \dot{\phi}^* \\ \dot{\theta}^* \\ \dot{\psi}^* \\ \dot{p}^* \\ \dot{q}^* \\ \dot{r}^* \end{bmatrix} = \begin{bmatrix} - \\ - \\ -\mathbf{V_a}^* \sin(\gamma^*) \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \frac{\mathbf{V_a}^*}{\mathcal{R}^*}\cos(\gamma^*) \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad \dot{\mathbf{u}}^* = \begin{bmatrix} \dot{\delta}_e^* \\ \dot{\delta}_a^* \\ \dot{\delta}_r^* \\ \dot{\delta}_t^* \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{3.39}$$

where '$-$' denotes any values.

To determine $\mathbf{z}^*$ and $\mathbf{u}^*$ such that $\dot{\mathbf{z}}^* = \mathcal{F}(\mathbf{z}^*, \mathbf{u}^*)$, the following non-linear equations has to be solved to obtain $\mathbf{z}^*$:

$$\begin{bmatrix} p_n^* \\ p_e^* \\ p_d^* \end{bmatrix} = \int_0^t \left(\mathbf{R}_b^v \begin{bmatrix} u \\ v \\ w \end{bmatrix} \right) dt + \begin{bmatrix} p_{n0} \\ p_{e0} \\ p_{d0} \end{bmatrix} \tag{3.40a}$$

$$\begin{bmatrix} u^* \\ v^* \\ w^* \end{bmatrix} = {\mathbf{V_a}}^* \begin{bmatrix} \cos(\alpha^*)\cos(\beta^*) \\ \sin(\beta^*) \\ \sin(\alpha^*)\cos(\beta^*) \end{bmatrix} \tag{3.40b}$$

$$\begin{bmatrix} \phi^* \\ \theta^* \\ \psi^* \end{bmatrix} = \int_0^t \left(\begin{bmatrix} p^* \\ q^* \\ r^* \end{bmatrix} \right) dt = \begin{bmatrix} \phi^* \\ \alpha^* + \gamma^* \\ \psi^* \end{bmatrix} \tag{3.40c}$$

$$\begin{bmatrix} p^* \\ q^* \\ r^* \end{bmatrix} = \frac{{\mathbf{V_a}}^* \cos(\gamma^*)}{\mathcal{R}^*} \begin{bmatrix} -\sin(\alpha^* + \gamma^*) \\ \sin(\phi^*)\cos(\alpha^* + \gamma^*) \\ \cos(\phi^*)\cos(\alpha^* + \gamma^*) \end{bmatrix} \tag{3.40d}$$

where $\mathbf{R}_b^v$ is the rotation matrix from coordinate body frame to vehicle frame, and $\begin{bmatrix} p_{n0} & p_{e0} & p_{d0} \end{bmatrix}^\top$ is the position at $t = 0$.

Given result of (3.40), it is possible to solve equation (3.5) for δ_e^*, δ_a^*, δ_r^*, and δ_t^*, giving $\mathbf{u}^*$:

$$\delta_e^* = \frac{\left[\frac{J_{xz}(p^{*2} - r^{*2}) + (J_x - J_z)p^* r^*}{0.5\rho({\mathbf{V_a}}^*)^2 cS} \right] - C_{m_0} - C_{m_\alpha}\alpha^* - C_{m_q}\frac{cq^*}{2{\mathbf{V_a}}^*}}{C_{m_{\delta_e}}} \tag{3.41a}$$

$$\begin{bmatrix} \delta_a^* \\ \delta_r^* \end{bmatrix} = \begin{bmatrix} C_{p_{\delta_a}} & C_{p_{\delta_r}} \\ C_{r_{\delta_a}} & C_{r_{\delta_r}} \end{bmatrix}^{-1} \begin{bmatrix} \frac{-\Gamma_1 p^* q^* + \Gamma_2 q^* r^*}{\frac{1}{2}\rho({\mathbf{V_a}}^*)^2 Sb} - C_{p_0} - C_{p_\beta}\beta^* - C_{p_p}\frac{bp^*}{2{\mathbf{V_a}}^*} - C_{p_r}\frac{br^*}{2{\mathbf{V_a}}^*} \\ \frac{-\Gamma_7 p^* q^* + \Gamma_1 q^* r^*}{\frac{1}{2}\rho({\mathbf{V_a}}^*)^2 Sb} - C_{r_0} - C_{r_\beta}\beta^* - C_{r_p}\frac{bp^*}{2{\mathbf{V_a}}^*} - C_{r_r}\frac{br^*}{2{\mathbf{V_a}}^*} \end{bmatrix} \tag{3.41b}$$

$$\delta_t^* = \sqrt{\frac{2\mathsf{m}\left(-r^* v^* + q^* w^* + g\sin(\theta^*)\right) - {\mathbf{V_a}}^{*2}\rho S\left(C_X^* + C_{X_q}^*\frac{cq^*}{2{\mathbf{V_a}}^*} + C_{X_{\delta_e}}^*\delta_e^*\right)}{\rho S_{prop} C_{prop} k_{motor}^2}} + \frac{{\mathbf{V_a}}^{*2}}{k_{motor}^2} \tag{3.41c}$$

where $C_X^* = C_X(\alpha^*)$, $C_{X_q}^* = C_{X_q}(\alpha^*)$ and $C_{X_{\delta_e}}^* = C_{X_{\delta_e}}(\alpha^*)$.

The system is expressed in terms of ${\mathbf{V_a}}^*$, γ^*, $\mathcal{R}^*$, α^*, β^* and ϕ^*. Since ${\mathbf{V_a}}^*$, γ^* and $\mathcal{R}^*$ are user-specified inputs, computing the trim state will then consist of an optimization

algorithm over α, β and ϕ to find α^*, β^* and ϕ^*. To find those three parameter, the following optimization problem must be solved:

$$(\alpha^*, \phi^*, \beta^*) = \arg\min \|\dot{\mathbf{z}}^* - \mathcal{F}(\mathbf{z}^*, \mathbf{u}^*)\|^2, \tag{3.42}$$

where $\dot{\mathbf{z}}^*$ is (3.39), and $\mathcal{F}(\mathbf{z}^*, \mathbf{u}^*)$ is (3.5).

3.5.2 *Linearized aircraft state-space model*

A linear state-space model can be expressed as:

$$\dot{\mathbf{z}} = \mathbf{F}\mathbf{z} + \mathbf{B}\mathbf{u} \tag{3.43a}$$

$$\mathbf{y} = \mathbf{C}\mathbf{z} \tag{3.43b}$$

This state-space model approximates $\mathcal{F}(\mathbf{z}, \mathbf{u})$ with:

$$\mathbf{F} = \begin{bmatrix} \frac{\partial \mathcal{F}_1(\mathbf{z}^*,\mathbf{u}^*)}{\partial p_n^*} & \frac{\partial \mathcal{F}_1(\mathbf{z}^*,\mathbf{u}^*)}{\partial p_e^*} & \cdots & \frac{\partial \mathcal{F}_1(\mathbf{z}^*,\mathbf{u}^*)}{\partial r^*} \\ \frac{\partial \mathcal{F}_2(\mathbf{z}^*,\mathbf{u}^*)}{\partial p_n^*} & \frac{\partial \mathcal{F}_2(\mathbf{z}^*,\mathbf{u}^*)}{\partial p_e^*} & \cdots & \frac{\partial \mathcal{F}_2(\mathbf{z}^*,\mathbf{u}^*)}{\partial r^*} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial \mathcal{F}_{12}(\mathbf{z}^*,\mathbf{u}^*)}{\partial p_n^*} & \frac{\partial \mathcal{F}_{12}(\mathbf{z}^*,\mathbf{u}^*)}{\partial p_e^*} & \cdots & \frac{\partial \mathcal{F}_{12}(\mathbf{z}^*,\mathbf{u}^*)}{\partial r^*} \end{bmatrix}, \tag{3.44a}$$

$$\mathbf{B} = \begin{bmatrix} \frac{\partial \mathcal{F}_1(\mathbf{z}^*,\mathbf{u}^*)}{\partial \delta_e^*} & \frac{\partial \mathcal{F}_1(\mathbf{z}^*,\mathbf{u}^*)}{\partial \delta_a^*} & \cdots & \frac{\partial \mathcal{F}_1(\mathbf{z}^*,\mathbf{u}^*)}{\partial \delta_t^*} \\ \frac{\partial \mathcal{F}_2(\mathbf{z}^*,\mathbf{u}^*)}{\partial \delta_e^*} & \frac{\partial \mathcal{F}_2(\mathbf{z}^*,\mathbf{u}^*)}{\partial \delta_a^*} & \cdots & \frac{\partial \mathcal{F}_2(\mathbf{z}^*,\mathbf{u}^*)}{\partial \delta_t^*} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial \mathcal{F}_{12}(\mathbf{z}^*,\mathbf{u}^*)}{\partial \delta_e^*} & \frac{\partial \mathcal{F}_{12}(\mathbf{z}^*,\mathbf{u}^*)}{\partial \delta_a^*} & \cdots & \frac{\partial \mathcal{F}_{12}(\mathbf{z}^*,\mathbf{u}^*)}{\partial \delta_t^*} \end{bmatrix}, \tag{3.44b}$$

where $\mathcal{F}_1(\cdot)$ and $\mathcal{F}_{12}(\cdot)$ are respectively the first and the twelfth output of $\mathcal{F}(\cdot)$. The lateral and longitudinal state-space model can be easily obtained from (3.44a) by separately considering longitudinal and lateral state variables and control inputs.

The decoupling between longitudinal and lateral dynamics is generically valid when the side-slip angle is sufficiently small and the trajectory is a straight line. Since it is assumed that the wind speed is zero, then $\beta = 0$ which means that the decoupling can be performed under a straight flight.

3.5.2.1 *Longitudinal state-space model*

For the longitudinal state-space the state vector and control input are:

$$\bar{\mathbf{z}}_{lon} = \begin{bmatrix} \bar{p_d} \\ \bar{u} \\ \bar{w} \\ \bar{\theta} \\ \bar{q} \end{bmatrix}, \quad \bar{\mathbf{u}}_{lon} = \begin{bmatrix} \bar{\delta}_e \\ \bar{\delta}_t \end{bmatrix} \tag{3.45}$$

The longitudinal state-space model is given by:

$$\dot{\bar{\mathbf{z}}}_{lon} = \mathbf{Z}_{lon}\mathbf{F}\mathbf{Z}_{lon}^{\top}\bar{\mathbf{z}}_{lon} + \mathbf{Z}_{lon}\mathbf{B}\mathbf{U}_{lat}^{\top}\bar{\mathbf{u}}_{lon} \tag{3.46}$$

where:

$$\mathbf{Z}_{lon} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}, \quad \mathbf{U}_{lon} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.47}$$

The corresponding longitudinal state-space obtained from (3.44a) is given by:

$$\begin{bmatrix} \dot{\bar{p}}_d \\ \dot{\bar{u}} \\ \dot{\bar{w}} \\ \dot{\bar{\theta}} \\ \dot{\bar{q}} \end{bmatrix} = \begin{bmatrix} 0 & \sin(\theta^*) & -\cos(\theta^*) & u^*\cos(\theta^*) + w^*\sin(\theta^*) & 0 \\ 0 & X_u & X_w & -g\cos(\theta^*) & X_q \\ 0 & Z_u & Z_w & -g\sin(\theta^*) & Z_q \\ 0 & 0 & 0 & 0 & 1 \\ 0 & M_u & M_w & 0 & M_q \end{bmatrix} \begin{bmatrix} \bar{p}_d \\ \bar{u} \\ \bar{w} \\ \bar{\theta} \\ \bar{q} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ X_{\delta_e} & X_{\delta_t} \\ Z_{\delta_e} & 0 \\ 0 & 0 \\ M_{\delta_e} & 0 \end{bmatrix} \begin{bmatrix} \bar{\delta}_e \\ \bar{\delta}_t \end{bmatrix}, \tag{3.48}$$

where the coefficients X_*, Z_* and M_* are given in Table A.2.

3.5.3 *Flight envelope*

The trim condition value can be chosen based on a flight envelope (airspeed and flight altitude variation). The flight envelope is computed with: the lift force (F_l), the drag force (F_d), the propulsion force (F_p) and the gravity force (F_g) which are given by:

$$F_l = \frac{1}{2}\rho {\mathbf{V_a}}^2 S \left(C_L(\alpha) + C_{L_q}\frac{c}{2\mathbf{V_a}}q + C_{L_{\delta_e}}\delta_e \right) \tag{3.49a}$$

$$F_d = \frac{1}{2}\rho {\mathbf{V_a}}^2 S \left(C_D(\alpha) + C_{D_q}\frac{c}{2\mathbf{V_a}}q + C_{D_{\delta_e}}\delta_e \right) \tag{3.49b}$$

$$F_p = \frac{1}{2}\rho S_{prop} C_{prop} \left((k_{motor}\delta_t)^2 - {\mathbf{V_a}}^2 \right) \tag{3.49c}$$

$$F_g = \mathsf{m}g \tag{3.49d}$$

The drag and lift forces must be compared with the propulsion and gravity forces, to determine if the aircraft is capable of flight and what its maximum velocity can be. However, the force F_l and F_d are expressed in the stability frame while F_p is expressed in the body frame and F_g in the vehicle frame. The F_l and F_d must change their coordinate frame to be compared with F_p and F_g. Hence:

$$F_{x^b} = -\cos(\alpha) F_d + \sin(\alpha) F_l, \tag{3.50a}$$
$$F_{z^v} = -\sin(\gamma_a) F_d - \cos(\gamma_a) F_l, \tag{3.50b}$$

where F_{x^b} is the force on apply on the aircraft along the i^b axis due to the drag and lift, while F_{z^v} is the force on along the k^v axis also due to the drag and lift. However, $\|\mathbf{V_w}\| = 0$ implies $\gamma_a \equiv \gamma$. Then, to determine the flight envelope with a chosen flight path angle γ^*, the aircraft can fly if:

$$\exists \alpha^* : \frac{F_{z^v} + F_g}{\mathsf{m}} = \frac{\partial \mathbf{V_a} \sin(\gamma^*)}{\partial t} \tag{3.51}$$

If multiple solutions exist for α^*, it is better to retain only the one which gives the lowest F_{x^b}. Then, the aircraft can reach a velocity if with the previously computed α^*:

$$\exists \mathbf{V_a}^* : F_{x^b} + F_p = 0. \tag{3.52}$$

The flight envelope of the Aerosonde UAV (aircraft parameters available in Appendix A.1) for a straight levelled flight is shown in Figure 3.5.

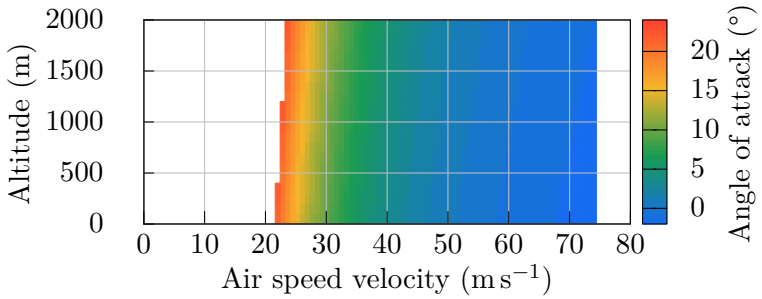


Figure 3.5: Flight envelope of the Aerosonde UAV between 0 and 2000 m. The values that are not plotted are the airspeed and altitude configuration that cannot be reached by the aircraft for a straight ($\mathcal{R}^* = \infty$) level flight ($\gamma^* = 0$).

Based on Figure 3.5, an airspeed velocity of 40 m s^{-1} at 500 m is a valid choice for the linearization.

The radius of turn $\mathcal{R}^*$ must be verified with airspeed and load variation flight envelope. However, if $\mathcal{R}^* = \infty$, then the altitude and airspeed flight envelope is sufficient. A lower radius of trun implies a larger load factor and a narrower flight envelope. In this thesis, the radius of turn is taken to be infinite (straight flight) because of teh focus on longitudinal dynamics.

3.6 NAVIGATION

This section aims to present UAV navigation. The reference navigation filters considered in this thesis are the same as the estimation filters presented in Chapter 2.

The navigation is fed by the sensors and feeds the guidance and control module, with the estimated longitudinal states, as shown in Figure 3.6

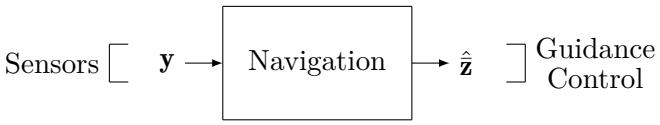


Figure 3.6: Input-output representation of the navigation module

3.7 CONTROL

This section aims to define the control law of the UAV. The navigation module is assumed to estimate all the components of the state vector $\mathbf{z}$. Using the decoupled linearized model, the control law that has been developed here is a full-state feedback with an integrator effect. The method used for this design is presented in Appendix C.

3.7.1 *Linear longitudinal control*

This section then describes the longitudinal control module of the UAV. In this section for sake of brevity and since only the longitudinal state and input are considered, the state vector $\mathbf{z}$ is used as $\mathbf{z}_{lon}$ and the control input vector $\mathbf{u}$ is used as $\mathbf{u}_{lon}$. The system considered here is given by (3.43) where $\mathbf{H} = \mathbf{I} \in \mathbb{R}^{5\times5}$. The state vector trimmed, and control inputs trimmed used to represent the longitudinal dynamics are:

$$\bar{\mathbf{z}} = \begin{bmatrix} \bar{p_d} \\ \bar{u} \\ \bar{w} \\ \bar{\theta} \\ \bar{q} \end{bmatrix}, \quad \bar{\mathbf{u}} = \begin{bmatrix} \bar{\delta}_e \\ \bar{\delta}_t \end{bmatrix} \tag{3.53}$$

The inputs of the control law are the desired and current state values provided by the guidance and navigation modules. They are, respectively, the desired airspeed velocity and flight path angle and the longitudinal state. The outputs are the throttle setting δ_t and the elevator deflection δ_e. Since it is assumed that there is no wind, then $\mathbf{V_g} \equiv \mathbf{V_a}$ and $\gamma \equiv \gamma_a$. The external view of the module is shown in Figure 3.7, where $\bar{\gamma_a}^c$ is the desired trimmed air-mass-referenced flight path angle and $\left|\bar{\mathbf{V}_\mathbf{g}}^c\right|$ is the desired trimmed modulus of the velocity vector.

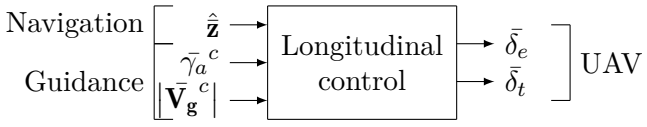


Figure 3.7: Input-output representation of the longitudinal guidance module

Since $\bar{\gamma_a} = \bar{\theta} - \bar{\alpha}$, then a possible solution for the control law of the elevator deflection is first to the control the pitch with $\bar{\theta}^c$, the desired pitch, and then compute $\bar{\alpha}$ to feed the desired pitch with $\bar{\theta}^c = \bar{\gamma_a}{}^c + \bar{\alpha}$. However, to command the pitch with $\bar{\gamma_a}{}^c$, the angle of attack must be computed. Moreover, the angle of attack formula is nonlinear since $\alpha(u,\, w) = \arctan\left(\frac{w}{u}\right)$. Then, to get a linear model for the control law, this formula must be linearized, hence:

$$\bar{\alpha}(\bar{u},\, \bar{w}) = \frac{\partial \alpha(u^*,\, w^*)}{\partial u}\bar{u} + \frac{\partial \alpha(u^*,\, w^*)}{\partial w}\bar{w} \tag{3.54a}$$

$$= \frac{-w^*}{u^{*2} + w^{*2}}\bar{u} + \frac{u^*}{w^{*2} + u^{*2}}\bar{w} \tag{3.54b}$$

The gain $\frac{-w^*}{u^{*2}+w^{*2}}$ is denoted A_u and the gain $\frac{u^*}{w^{*2}+u^{*2}}$ is denoted A_w.

To command the throttle input with $|\bar{\mathbf{V}}_\mathbf{g}{}^c|$, the modulus of the velocity vector must be computed. However, the modulus of the velocity vector is nonlinear since without wind, $|\mathbf{V_g}(u,\, w)| = \sqrt{u^2 + w^2}$. Then, to get a linear model for the control law, this formula must be linearized, as follows:

$$\left|\bar{\mathbf{V}}_\mathbf{g}(\bar{u},\, \bar{w})\right| = \frac{\partial \left|\bar{\mathbf{V}}_\mathbf{g}(u^*,\, w^*)\right|}{\partial u}\bar{u} + \frac{\partial \left|\bar{\mathbf{V}}_\mathbf{g}(u^*,\, w^*)\right|}{\partial w}\bar{w} \tag{3.55a}$$

$$= \frac{u^*}{\sqrt{u^{*2} + w^{*2}}}\bar{u} + \frac{w^*}{\sqrt{w^{*2} + u^{*2}}}\bar{w} \tag{3.55b}$$

The gain $\frac{u^*}{\sqrt{u^{*2}+w^{*2}}}$ is denoted V_u and the gain $\frac{w^*}{\sqrt{w^{*2}+u^{*2}}}$ is denoted V_w.

Since $\left|\bar{\mathbf{V}}_\mathbf{g}(\bar{u},\, \bar{w})\right| = V_u\bar{u} + V_w\bar{w}$, a possible solution for the control law of the throttle input is first to the control the velocity with $\bar{u}^c$ and feed it with $\bar{u}^c = \frac{|\bar{\mathbf{V}}_\mathbf{g}{}^c| - V_w\bar{w}}{V_u}$.

To minimize the error between $\bar{\theta}$ and its desired output $\bar{\theta}^c$ a new state $\bar{\theta}_i$ is added and $\dot{\bar{\theta}}_i = \bar{\theta}^c - \mathbf{H}_\theta\bar{\mathbf{z}}$, where $\mathbf{H}_\theta$ is the row of $\mathbf{H}$ that give the observability of the state $\bar{\theta}$. Likewise, to minimize the error between $\bar{u}$ and its desired output $\bar{u}^c$ a new state $\bar{u_i} = \bar{u}^c - \mathbf{H_u}\bar{\mathbf{z}}$ is

created, where $\mathbf{H_u}$ is the row of $\mathbf{H}$ that make the state $\bar{u}$ observable. The new state-space model with $\bar{\theta}^c$ and $\bar{u}^c$ as input in open loop is defined as:

$$
\begin{cases}
\begin{bmatrix} \dot{\bar{p}}_d \\ \dot{\bar{u}} \\ \dot{\bar{w}} \\ \dot{\bar{\theta}} \\ \dot{\bar{q}} \\ \dot{\bar{\theta}}_i \\ \dot{\bar{u}}_i \end{bmatrix} = \begin{bmatrix} 0 & \sin\theta^* & -\cos\theta^* & u^*\cos\theta^* + w^*\sin\theta^* & 0 & 0 \\ 0 & X_u & X_w & -g\cos\theta^* & X_q & 0 \\ 0 & Z_u & Z_w & -g\sin\theta^* & Z_q & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & M_u & M_w & 0 & M_q & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \bar{p}_d \\ \bar{u} \\ \bar{w} \\ \bar{\theta} \\ \bar{q} \\ \bar{\theta}_i \\ \bar{u}_i \end{bmatrix} + \\
\qquad \begin{bmatrix} 0 & 0 \\ X_{\delta_e} & X_{\delta_t} \\ Z_{\delta_e} & 0 \\ 0 & 0 \\ M_{\delta_e} & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \bar{\delta}_e \\ \bar{\delta}_t \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \bar{\theta}^c \\ \bar{u}^c \end{bmatrix} \qquad (3.56a) \\
\mathbf{y} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} \bar{p}_d \\ \bar{u} \\ \bar{w} \\ \bar{\theta} \\ \bar{q} \\ \bar{\theta}_i \\ \bar{u}_i \end{bmatrix} \qquad (3.56b)
\end{cases}
$$

The full state feedback gain of the system described by (3.56) is obtained using the linear quadratic regulator (LQR) method detailed in Appendix C.1, with zero weight on the cross product matrix $\mathbf{N}$. A good compromise between actuator effort and performance for the UAV

Aerosonde — with the airframe parameter in Appendix A.1 — is obtained by choosing the following $\mathbf{Q}$ and $\mathbf{R}$ matrix:

$$\mathbf{Q} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{R} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{3.57}$$

The output gain of the LQR denoted $\mathbf{L}$ is of the following form:

$$\mathbf{L} = \begin{bmatrix} L_{11} & L_{12} & L_{13} & L_{14} & L_{15} & L_{16} & L_{17} \\ L_{21} & L_{22} & L_{23} & L_{24} & L_{25} & L_{26} & L_{27} \end{bmatrix} \tag{3.58}$$

The block $\begin{bmatrix} L_{11} & \cdots & L_{15} \end{bmatrix}$ is denoted $\mathbf{L}_\theta$ and $\begin{bmatrix} L_{21} & \cdots & L_{25} \end{bmatrix}$ is denoted $\mathbf{L}_\mathbf{u}$, $\begin{bmatrix} L_{16} & L_{17} \end{bmatrix}$ is denoted $\mathbf{L}_{\theta_\mathbf{i}}$ and $\begin{bmatrix} L_{26} & L_{27} \end{bmatrix}$ is denoted $\mathbf{L}_{\mathbf{u_i}}$.

Then, the closed loop state space model with $\bar{\theta}^c$ and $\bar{u}^c$ as input is the following:

$$\begin{cases} \begin{bmatrix} \dot{\bar{\mathbf{z}}} \\ \dot{\bar{\theta}}_i \\ \dot{\bar{u}}_i \end{bmatrix} = \begin{bmatrix} \mathbf{F} - \mathbf{B}\begin{bmatrix} \mathbf{L}_\theta \\ \mathbf{L}_\mathbf{u} \end{bmatrix} & -\mathbf{B}\begin{bmatrix} \mathbf{L}_{\theta_\mathbf{i}} \\ \mathbf{L}_{\mathbf{u_i}} \end{bmatrix} \\ -\mathbf{H}_\theta & 0 \\ -\mathbf{H}_\mathbf{u} & 0 \end{bmatrix} \begin{bmatrix} \bar{\mathbf{z}} \\ \bar{\theta}_i \\ \bar{u}_i \end{bmatrix} + \begin{bmatrix} \mathbf{0}_{5,2} \\ \mathbf{I}_{2,2} \end{bmatrix} \begin{bmatrix} \bar{\theta}^c \\ \bar{u}^c \end{bmatrix} & \text{(3.59a)} \\ \mathbf{y} = \begin{bmatrix} \mathbf{H} & \mathbf{0}_{5,2} \end{bmatrix} \begin{bmatrix} \bar{\mathbf{z}} \\ \bar{\theta}_i \\ \bar{u}_i \end{bmatrix} & \text{(3.59b)} \end{cases}$$

The block diagram of the longitudinal control with $\bar{\theta}^c = \bar{\gamma_a}^c + \bar{\alpha}$ and $\bar{u}^c = \frac{|\bar{\mathbf{V}}_\mathbf{g}{}^c| - V_w \bar{w}}{V_u}$ is shown in Figure 3.8.

The control outputs $\bar{\delta_e}$ and $\bar{\delta_t}$ with the guidance inputs $|\bar{\mathbf{V}}_\mathbf{g}{}^c|$ and $\bar{\gamma_a}^c$ and the navigation input $\hat{\bar{\mathbf{z}}}$ are given by:

$$\bar{\delta_e} = -\mathbf{L}_\theta \hat{\bar{\mathbf{z}}} - \frac{\mathbf{L}_{\theta_\mathbf{i}}}{s}\left[\bar{\gamma_a}^c + \left(A_u \mathbf{H}_\mathbf{u} \hat{\bar{\mathbf{z}}} + A_w \mathbf{H}_\mathbf{w} \hat{\bar{\mathbf{z}}}\right) - \mathbf{H}_\theta \hat{\bar{\mathbf{z}}}\right] \tag{3.60a}$$

$$\bar{\delta_t} = -\mathbf{L}_\mathbf{u} \hat{\bar{\mathbf{z}}} - \frac{\mathbf{L}_{\mathbf{u_i}}}{s}\left(\frac{|\bar{\mathbf{V}}_\mathbf{g}{}^c| - V_w \mathbf{H}_\mathbf{w} \hat{\bar{\mathbf{z}}}}{V_u} - \mathbf{H}_\mathbf{u} \hat{\bar{\mathbf{z}}}\right) \tag{3.60b}$$

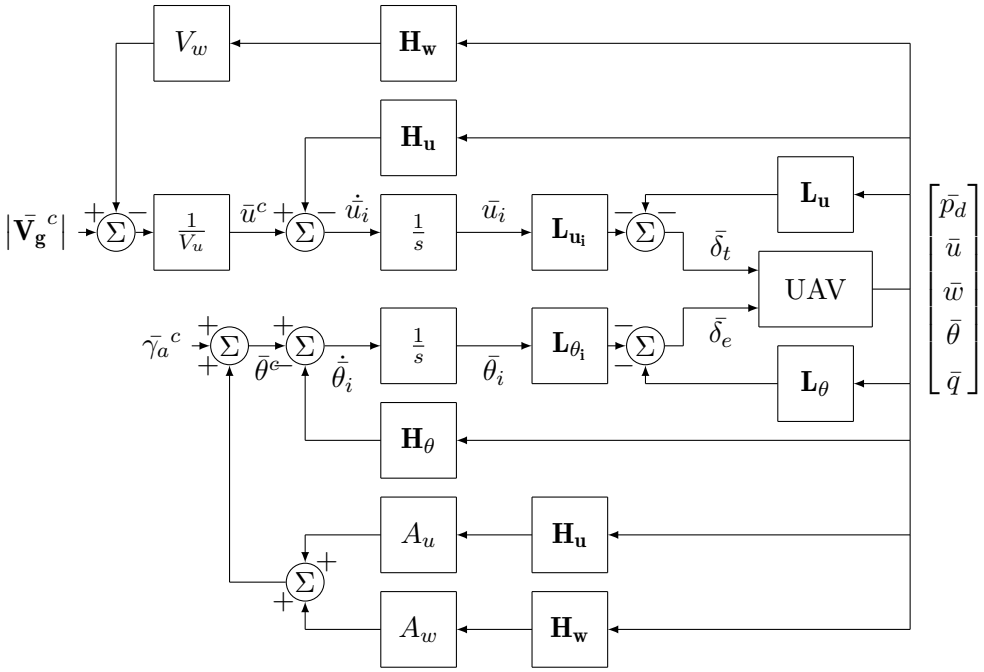


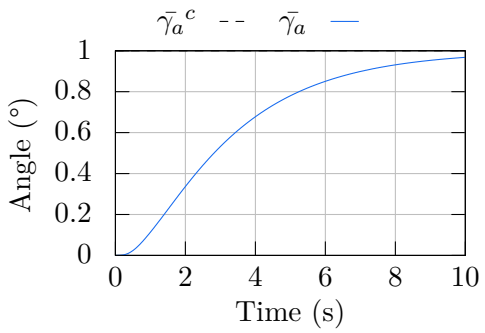
Figure 3.8: Longitudinal control systems of the UAV with $|\bar{\mathbf{V}}_\mathbf{g}{}^c|$ and $\bar{\gamma}_a{}^c$.

The state space model of the longitudinal control with the linearized longitudinal UAV dynamics is then:

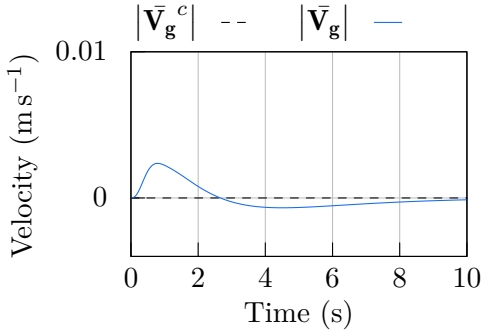
$$\begin{bmatrix} \dot{\bar{\mathbf{z}}} \\ \dot{\bar{\theta}}_i \\ \dot{\bar{u}}_i \end{bmatrix} = \begin{bmatrix} \mathbf{F} - \mathbf{B}\begin{bmatrix} \mathbf{L}_\theta \\ \mathbf{L_u} \end{bmatrix} & -\mathbf{B}\begin{bmatrix} \mathbf{L}_{\theta_\mathbf{i}} \\ \mathbf{L_{u_i}} \end{bmatrix} \\ \begin{bmatrix} A_u & A_w & 0 & 0 \\ 0 & \frac{-V_w}{V_u} & 0 & 0 \end{bmatrix} - \begin{matrix} \mathbf{H}_\theta \\ \mathbf{H_u} \end{matrix} & \begin{matrix} 0 & 0 \\ 0 & 0 \end{matrix} \end{bmatrix} \begin{bmatrix} \bar{\mathbf{z}} \\ \bar{\theta}_i \\ \bar{u}_i \end{bmatrix} + \begin{bmatrix} \mathbf{0}_{5,2} \\ \mathbf{I}_{2,2} \end{bmatrix} \begin{bmatrix} \bar{\gamma}_a{}^c \\ |\bar{\mathbf{V}}_\mathbf{g}{}^c| \end{bmatrix} \tag{3.61a}$$

$$\mathbf{y} = \begin{bmatrix} \mathbf{H} & \mathbf{0}_{5,2} \end{bmatrix} \begin{bmatrix} \bar{\mathbf{z}} \\ \bar{\theta}_i \\ \bar{u}_i \end{bmatrix} \tag{3.61b}$$

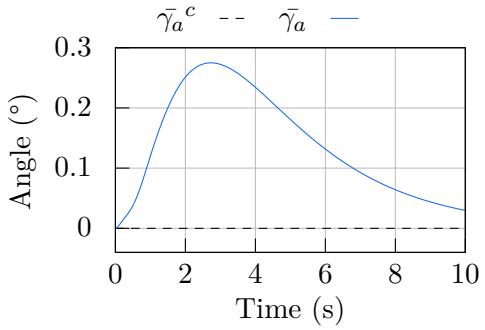
This system of equation is applied to the longitudinal model (3.48) with the airframe parameters of the Appendix A.1. By linearizing around $40\,\mathrm{m\,s^{-1}}$ and 500 m, the step responses of figures 3.9 are obtained:



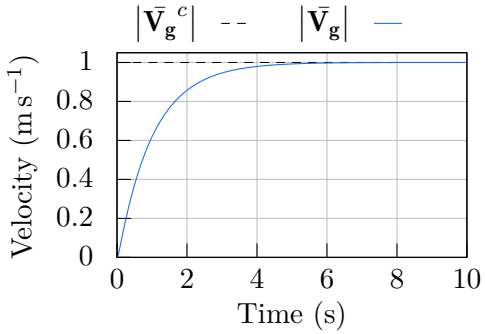
(a) Step response of $\bar{\gamma_a}$ with $\bar{\gamma_a}^c = 1°$ and $\left|\bar{\mathbf{V}_\mathbf{g}}^c\right| = 0\,\mathrm{m\,s^{-1}}$



(b) Step response of $\left|\bar{\mathbf{V}_\mathbf{g}}\right|$ with $\bar{\gamma_a}^c = 1°$ and $\left|\bar{\mathbf{V}_\mathbf{g}}^c\right| = 0\,\mathrm{m\,s^{-1}}$



(c) Step response of $\bar{\gamma_a}$ with $\bar{\gamma_a}^c = 0°$ and $\left|\bar{\mathbf{V}_\mathbf{g}}^c\right| = 1\,\mathrm{m\,s^{-1}}$



(d) Step response of $\left|\bar{\mathbf{V}_\mathbf{g}}\right|$ with $\bar{\gamma_a}^c = 0°$ and $\left|\bar{\mathbf{V}_\mathbf{g}}^c\right| = 1\,\mathrm{m\,s^{-1}}$

Figure 3.9: Step response of $\bar{\gamma_a}$ and $\left|\bar{\mathbf{V}_\mathbf{g}}\right|$ with step amplitude of $\bar{\gamma_a}^c = 1°$ ((a) and (b)) and $\left|\bar{\mathbf{V}_\mathbf{g}}^c\right| = 1\,\mathrm{m\,s^{-1}}$ ((c) and (d)) of the longitudinal plant with longitudinal control

3.8 GUIDANCE

This section aims to define the guidance of the UAV. The flight phases can be divided into multiple phases. Only the level flight phases — that can include small altitude adjustment — is considered in this section. It is assumed that the UAV is initially in a steady flight at 500 m ASL.

3.8.1 *Longitudinal guidance law*

The longitudinal guidance law is fed by the navigation module with the longitudinal states and feeds the control module with the desired air-mass-referenced flight path angle $\bar{\gamma_a}^c$ and desired velocity $\left|\bar{\mathbf{V}_\mathbf{g}}\right|$. The external view of the module is shown in Figure 3.10

The longitudinal guidance considered in this thesis contains a pre-registered waypoint to reach. In order to decouple the longitudinal and lateral guidance, the waypoint considered

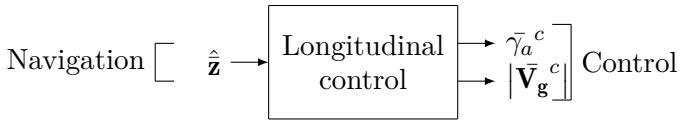


Figure 3.10: Input-output representation of the longitudinal guidance module

in this longitudinal module is composed of a desired altitude and velocity to reach after a specific time.

The control module already has a desired velocity. However, it takes as input a desired flight path angle. The first step to guide the aircraft is then to get a desired flight path angle from a desired altitude. This is done by using control techniques explained in Appendix C.

To minimize the error between $\bar{p_d}$ and its desired output $\bar{p_d}^c$, a new state $\bar{p_{di}}$ is added and $\dot{\bar{p_{di}}} = \bar{p_d}^c - \mathbf{H_{p_d}}\bar{\mathbf{z}}$, where $\mathbf{H_{p_d}}$ is the row of $\mathbf{H}$ that give the observability of the state $\bar{p_d}$. Since the altitude of the aircraft is controlled by the air-mass-referenced flight path angle input $\bar{\gamma_a}^c$, then the state space model considered for the control with $\bar{p_d}^c$ only uses state $\bar{\mathbf{z}}$ and $\bar{\theta_i}$ and input $\bar{\gamma_a}^c$ of state space model (3.61). For sake of brevity in this section the evolution matrix, input matrix and output matrix of (3.61) are denoted $\mathbf{F}_\mathrm{C}$, $\mathbf{B}_\mathrm{C}$ and $\mathbf{H}_\mathrm{C}$, respectively. Then, new state-space model with $\bar{p_d}^c$ as input in open loop is defined as:

$$\begin{cases} \begin{bmatrix} \dot{\bar{\mathbf{z}}} \\ \dot{\bar{\theta}}_i \\ \dot{\bar{p}}_{di} \end{bmatrix} = \begin{bmatrix} \mathbf{F}_{\mathrm{C}1:6,1:6} & 0 \\ -\mathbf{H_{p_d}} & 0 \end{bmatrix} \begin{bmatrix} \bar{\mathbf{z}} \\ \bar{\theta}_i \\ \bar{p_{di}} \end{bmatrix} + \begin{bmatrix} \mathbf{B}_{\mathrm{C}1:6,1} \\ 0 \end{bmatrix} \begin{bmatrix} \bar{\gamma_a}^c \end{bmatrix} + \begin{bmatrix} \mathbf{0}_{6,1} \\ \mathbf{I} \end{bmatrix} \begin{bmatrix} \bar{p_d}^c \end{bmatrix} & \text{(3.62a)} \\ \mathbf{y} = \begin{bmatrix} \mathbf{H}_{\mathrm{C}n_z,1:6} & 0_{n_z,1} \end{bmatrix} \begin{bmatrix} \bar{\mathbf{z}} \\ \bar{\theta}_i \\ \bar{p_{di}} \end{bmatrix} & \text{(3.62b)} \end{cases}$$

The full state feedback gain is obtained using the LQR method on (3.62), with zero weight on the cross product matrix $\mathbf{N}$. A good compromise between actuator effort and performance for the UAV Aerosonde (airframe parameter in Appendix A.1) is obtained by choosing the following $\mathbf{Q}$ and $\mathbf{R}$ matrix:

$$\mathbf{Q} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{R} = \begin{bmatrix} 1000 \end{bmatrix}. \tag{3.63}$$

The output gain of the LQR computation denoted $\mathbf{L}$ is of the following form:

$$\mathbf{L} = \begin{bmatrix} L_{11} & L_{12} & L_{13} & L_{14} & L_{15} & L_{16} & L_{17} \end{bmatrix} \tag{3.64}$$

The block $\begin{bmatrix} L_{11} & \cdots & L_{16} \end{bmatrix}$ is denoted $\mathbf{L}_{\mathbf{p_d}}$ and L_{17} is denoted $L_{p_{di}}$. Then, the closed loop state space model with $\bar{p_d}^c$ as input is the following:

$$\begin{cases} \begin{bmatrix} \dot{\bar{\mathbf{z}}} \\ \dot{\bar{\theta}}_i \\ \dot{\bar{p_{di}}} \end{bmatrix} = \begin{bmatrix} \mathbf{F}_{\mathrm{C}1:6,1:6} - \mathbf{B}_{\mathrm{C}1:6,1}\mathbf{L}_{\mathbf{p_d}} & -\mathbf{B}_{\mathrm{C}1:6,1}L_{p_{di}} \\ -\mathbf{H}_{\mathbf{p_d}} & 0 \end{bmatrix} \begin{bmatrix} \bar{\mathbf{z}} \\ \bar{\theta}_i \\ \bar{p_{di}} \end{bmatrix} + \begin{bmatrix} \mathbf{0}_{6,1} \\ \mathbf{I} \end{bmatrix} \begin{bmatrix} \bar{p_d}^c \end{bmatrix} & \text{(3.65a)} \\ \mathbf{y} = \begin{bmatrix} \mathbf{H}_{\mathrm{C}n_z,1:6} & \mathbf{0}_{n_z,1} \end{bmatrix} \begin{bmatrix} \bar{\mathbf{z}} \\ \bar{\theta}_i \\ \bar{p_{di}} \end{bmatrix}. & \text{(3.65b)} \end{cases}$$

However, the weakness of this control method for the altitude is that it will take the same time to reach an increment of 1 m or 100 m. This is due to the fact that the same linear approximation is used in both cases and the model does not account for saturation in terms of angles or angular rates. A large step in the desired altitude will result in a large input in the desired air-mass-referenced flight path angle $\bar{\gamma_a}^c$. This input can be above the small angle assumption used for the linear approximation and even above physical limitation — for example a desired $\bar{\gamma_a}^c$ above 90° to get a higher climb-rate. To avoid this situation a solution is to use a saturator to limit the command applied on $\bar{\gamma_a}^c$. However, using a saturator with an integrator on the output error will wind-up the integrator that will probably provide an overshoot in the desired output and take time to unwind or even destabilized the system. To avoid this problem, a solution is to implement an integral anti wind-up system. The block diagram of the longitudinal guidance with the control and UAV with a saturator and an integrator anti-wind-up system is shown in Figure 3.11.

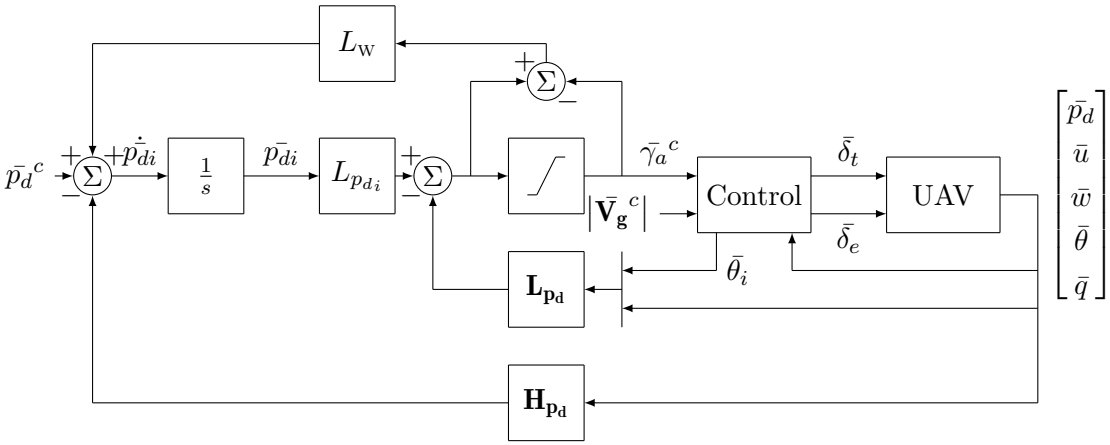
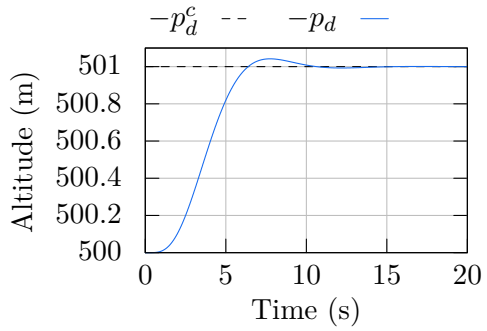


Figure 3.11: Longitudinal guidance of the longitudinal control for UAV

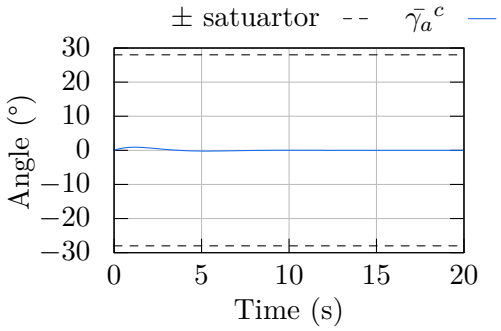
The gain L_{W} controls the integrator anti-wind-up system. The anti-wind up uses a gain that was determined here empirically by simulation.

The dynamics with a saturator are non-linear. The controller is applied to the longitudinal model 3.61 with the airframe parameter of the Appendix A.1 linearized around $40\,\mathrm{m\,s^{-1}}$

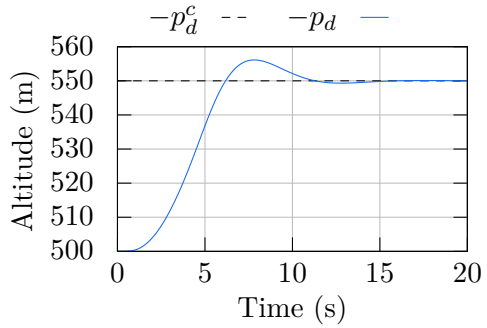
and 500 m with saturation on $\bar{\gamma_a}^c$ above and below 28° and with a gain $L_W = 2$. The step responses and input on $\bar{\gamma_a}^c$ of Figure 3.12 are obtained.



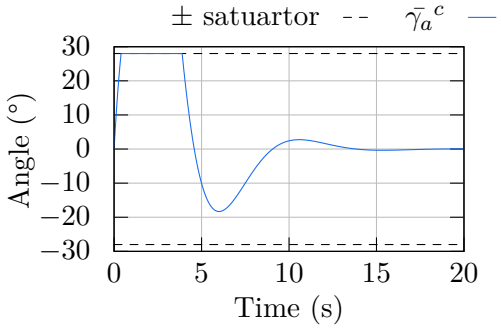
(a) Step response of $-p_d$ with $-\bar{p_d}^c = 1$



(b) Command $\bar{\gamma_a}^c$ with step input $-\bar{p_d}^c = 1$



(c) Step response of $-p_d$ with $-\bar{p_d}^c = 100$



(d) Command $\bar{\gamma_a}^c$ with step input $-\bar{p_d}^c = 100$

Figure 3.12: Step response of $-\bar{p_d}^c$ with step amplitude of $-\bar{p_d}^c = 1\,\text{m}$ ((a) and (b)) and $-\bar{p_d}^c = 50\,\text{m}$ ((c) and (d)) of the longitudinal control with longitudinal guidance. All results are obtained with $\left|\bar{\mathbf{V}_\mathbf{g}}^c\right| = 0\,\text{m}\,\text{s}^{-1}$

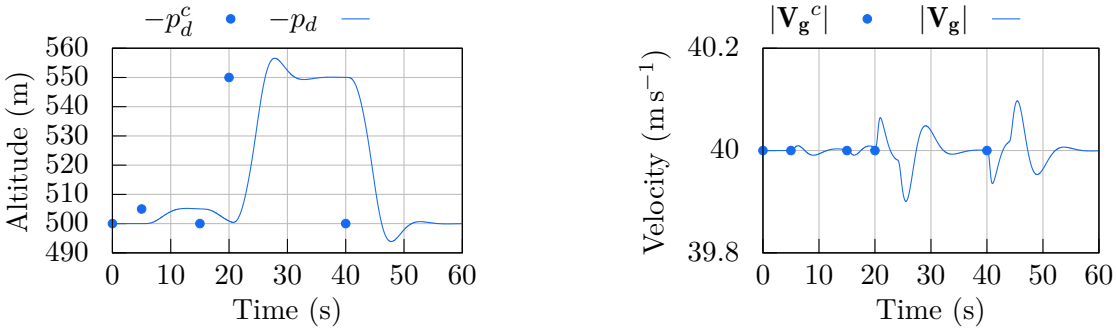
Figure 3.12 shows that the saturator with the integrator anti-wind-up system performs well. The $\bar{\gamma_a}^c$ is constrained inside its physical limitation and small enough to satisfy the linearization assumption. As expected it takes more time to complete a large altitude step (when the $\bar{\gamma_a}^c$ reaches saturation) than it takes to do a small step (when the $\bar{\gamma_a}^c$ is not the saturated).

This guidance module is thus well-suited to be used for following the set of pre-defined way points.

Figure 3.13 shows a list of waypoints and the trajectory performed by the UAV with the same parameters used for Figure 3.12. The second altitude waypoint is set to 5 m while the fourth is set to 50 m. Then, the second waypoint is reached without saturation of the $\bar{\gamma_a}^c$ input while fourth saturates it.

Figure 3.13 shows that the altitude waypoints are not reached at the desired time, the same would have been observed on the velocity if waypoints were not the same during all

the simulation. This behaviour is due to the guidance that sets a new desired output of the waypoint at its associated time. To reach the waypoints at the specified time, the guidance module must be more sophisticated. A possible solution is to anticipate the command to send to the controller. However, for the purposes of this thesis, the simple longitudinal guidance module presented in this section is suitable.



(a) Altitude waypoints

(b) Velocity waypoints

Figure 3.13: Altitude (a) and velocity (b) of the UAV with waypoints

3.9 CHAPTER SUMMARY

In this chapter, a detailed dynamic model is introduced for the fixed wing UAV under consideration in this thesis, including sensor models. This model is essential for fault estimation using model-based approaches, which require a process model. Under realistic assumptions, including a low side-slip angle, the longitudinal and lateral dynamics can be decoupled and both models can be linearized about their desired trim conditions. Moreover, the longitudinal control and guidance of the UAV was presented to allow for tracking of simple trajectories with a close loop system. This is necessary to visualize the impact of a fault on the trajectory of the UAV. The numerical results presented in this thesis are then obtained using the model, control module and guidance module presented in this chapter.

4

JUMP-MARKOV REGULARIZED PARTICLE FILTER

Model-based approaches were introduced in Chapter 2 for both fault isolation and fault estimation. It has been highlighted that fault estimation is of higher interest for autonomous vehicles without hardware redundancies. The mathematical model of a small autonomous fixed wing UAV without hardware redundancy was introduced in Section 3.2 for the case of non-linear dynamics and in Section 3.5 when the dynamics were linearized. In the presence of sensor and actuator faults, the model presented in this chapter is an extension of the small UAV model, which is hybrid and multimodal, to account for the possible faulty and fault free modes of operations. A JMRPF will be shown to be well-adapted to detect and estimate the fault for this class of system models.

This chapter is organized as follows: Section 4.1 details some of the issues that fault estimation on UAV is facing. In Section 4.2, the limitations of state-of-the-art methods facing these issues are highlighted, and Section 4.3 introduces the idea behind the mechanism of the JMRPF to overcome them. In Section 4.4, a formulation of the JMRPF is introduced with the associated algorithm. Section 4.5 presents a detailed numerical simulation analysis of the new algorithm applied to a fixed-wing UAV under sensor and actuator faults. Section 4.6 summarizes the lessons learnt from this chapter.

4.1 UNMANNED AERIAL VEHICLE FAULT ESTIMATION REQUIREMENTS

Due to the application considered, it is clear that any fault must be estimated quickly and accurately to avoid compromising the mission, run down the UAV integrity, or even the integrity of elements in the UAV's surrounding environment. Thus, the method proposed must be able to estimate faults quickly and accurately, with a good false alarm and missed detection rate.

Moreover, as presented in Section 3.4, the UAV has multiple sensors that provide the same information with different measurement noise. This can be seen as an advantage for state estimation, but it also makes the sensor fault estimation more complex. Indeed, the use of multiple sensors to measure the same variable is what makes the measurement function ambiguous in a faulty situation. This is due to the fact that only the sensor noise parametrization makes it possible to differentiate between the observations. Figure 4.1 shows

the situation where two sensors with different noise standard deviations are used to measure a state x with measurement equations given by:

$$\begin{bmatrix} \mathrm{y}_a \\ \mathrm{y}_b \end{bmatrix} = \begin{bmatrix} \mathrm{x}_k \\ \mathrm{x}_k \end{bmatrix} + \begin{bmatrix} \nu_a \\ \nu_b \end{bmatrix} \tag{4.1}$$

The only difference between these two equations are the measurement noises ν_a and ν_b. The measurements y_a produced by a sensor a are faulty whereas the measurements y_b from a second sensor b are fault-free. The state estimation of x_k is based on the posterior density of $p\left(\mathrm{x}_k|\mathbf{Y}_{1:k}\right)$ after updating the predicted state density $p\left(\mathrm{x}_k|\mathbf{Y}_{1:k-1}\right)$.

In this situation, basing the decision on the posterior density $p\left(\mathrm{x}_k|\mathbf{Y}_{1:k}\right)$ highlights the fact that there is an ambiguous choice to decide which mode of the posterior density corresponds to the state estimate.

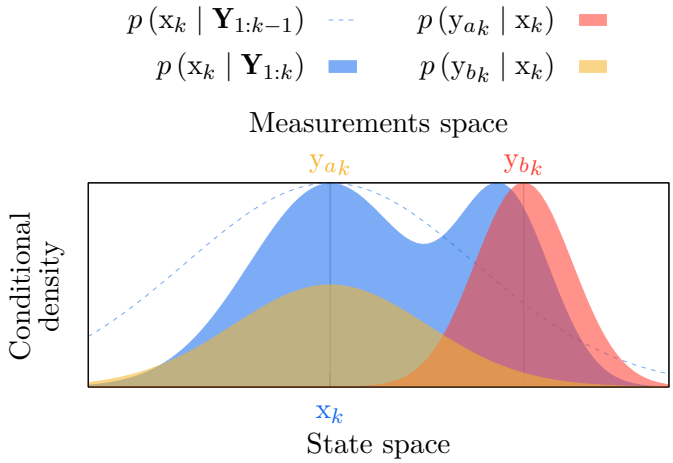


Figure 4.1: State estimate of x with a fault free and a faulty measurement estimate density

Therefore, the methods proposed must be able to estimate ambiguous sensor faults — in other words, there is a multimodal probability distribution with one possible states.

4.2 LIMITATIONS OF EXISTING METHODS

Since an approach that can estimate multimodal states must be considered to estimate the fault in the application treated, the RPF, which is a state-of-the-art method for multimodal state estimation, is used as a benchmark. Indeed, it has been highlighted in Section 2.5 that this algorithm is suited for non-linear state estimation, even in the presence of multimodality and non-Gaussian noise. Then, to provide an understanding of the underlying principles of the methods proposed in this section, the main steps of the RPF are briefly recalled: The RPF uses a model to propagate the particles during the prediction step. If the model has high uncertainty, it will spread the particles over a wider region of the state space than the

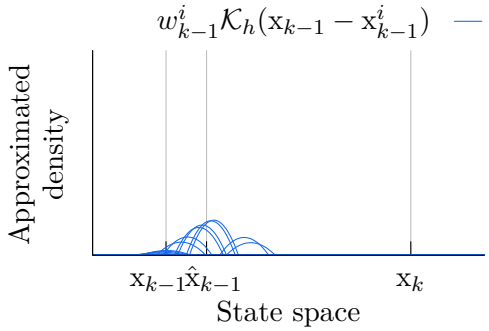
low uncertainty case. At the update step, the likelihood of the particles is evaluated, and their weights are updated accordingly. Then, at the estimation step, a global state estimate is made based on the likelihood and the state of each particle. Finally, the resampling step, where the efficiency — or another criterion — is computed and if the criterion is triggered by comparison against a threshold, then the resampling step is preformed. During the resampling, the most likely particles are kept and duplicated while others are removed. The RPF adds noise when particles are duplicated. This allows the particles to cover a wider area and reduces the chances of superposed solutions compared to the SIR particle filter. The regularization noise has an impact on the area covered by the kernel at the resampling step. However, this regularization noise has a more limited impact than the process noise in terms of area covered.

When faults are considered, the particle placement performed by the RPF is not flawless. Indeed, when a fault occurs, the fault state moves from a value close to zero to the value of the fault. However, this value of the fault can take several time steps to be reached or may not be reached at all. Indeed, even when the particles are propagated, if the value of the fault is not covered by the current particles, then the RPF may never reach this value. To reduce this risk, the solution may be to increase the process noise and the regularization noise, even if the latter has a less significant impact. Doing this, however, degrades the accuracy of the estimate, and choosing an ad-hoc process noise that complies with the amplitude of the additive abrupt faults that the system may encounter, given assumptions on the amplitude of faults. This is possible since, for example, an actuator is physically bounded to its reachable range, and it is safe to assume that a fault cannot have an amplitude outside this range. However, having a process noise that is able to cover all this range will lead to a poor estimate in terms of accuracy. The estimation of a state with process noises suited and not suited to the amplitude of an abrupt change is illustrated in Figure 4.2, where x_k denotes the fault state at time step k.

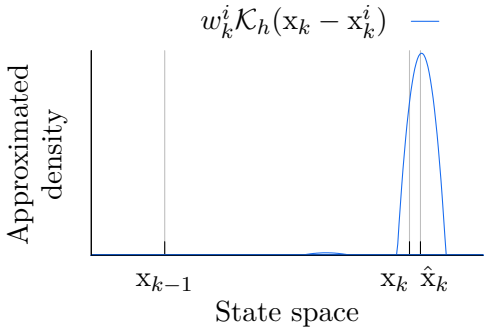
In Figure 4.2, the poor accuracy of the estimate due to a large process noise is visible in Figure 4.2a and Figure 4.2b, even when the system is fault-free. This is illustrated in Figure 4.2a where the particles are spread on a wider area than in Figure 4.2c where the process noise is smaller, and then the accuracy is better. However, when the fault occurs, the system with the large process noise is able to place particles around the fault. This is illustrated in Figure 4.2b, while in Figure 4.2d the particles cannot be placed around the fault since the process noise is not large enough to reach the fault. In Figure 4.2b the estimate of the fault is not yet close to the real fault since the resampling step does not occur yet, but the likelihood show to be high on around the fault.

This result is confirmed in Figure 4.3 where the 10 steps before and after the fault occurring are shown. Indeed, in Figure 4.3a the system with the large process noise is able to converge to the fault in few time steps but with a poor accuracy, while in Figure 4.3b the system with the small process noise is not able to converge to the fault within the 10 time steps shown after the fault occurred, but it has a better accuracy.

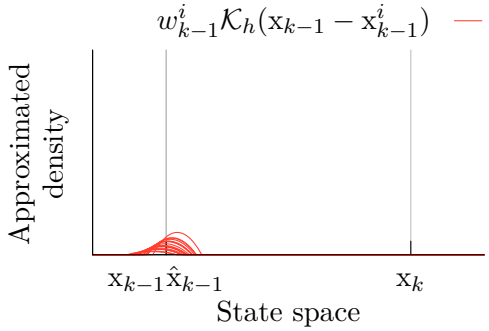
Then the main limitation of the RPF for fault estimation is that a trade-off must be chosen between the accuracy and the maximum amplitude of fault able to be estimated.



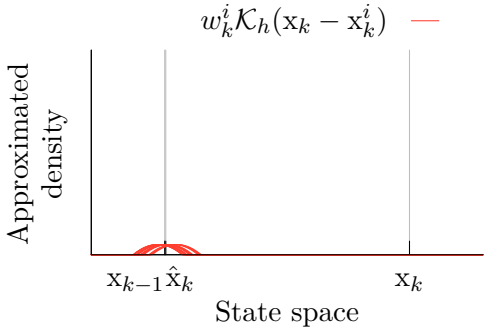
(a) Abrupt additive change and the previous approximated posterior density with a large process noise.



(b) Abrupt additive change and the current approximated posterior density with a large process noise.



(c) Abrupt additive change and the previous approximated posterior density with a small process noise.

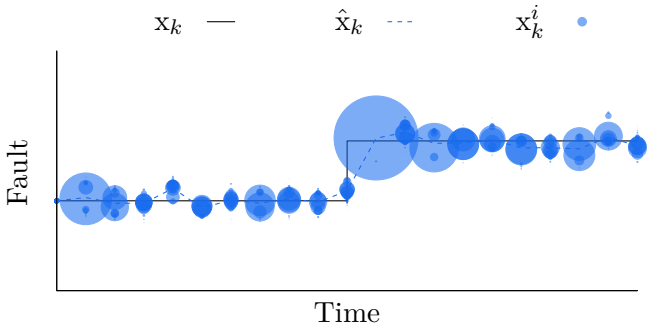


(d) Abrupt additive change and the current approximated posterior density with a small process noise.

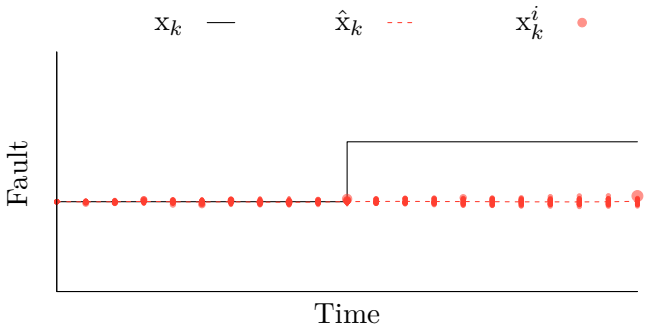
Figure 4.2: RPF performing estimation of an additive abrupt change, with a fault occurring close enough from the previous approximated posterior density — (a), (b) — and far from the previous approximated posterior density — (c), (d) — in comparison to the process noise.

However, this limitation can be partially overcome by other existing methods like the IMM. Indeed, the advantage of the IMM is to run several models, including one for the nominal mode. Then, a model with a small process noise can be used to estimate the faulty free situation, and a model with a large process noise can be used to estimate the fault. This lead to an accurate estimate of the state in the fault free situation, since no fault estimate is used in the estimate of the state vector. However, the accuracy of the fault and thus, the accuracy of the estimate of the state components impacted by the occurrence of the fault are lowered. Moreover, the use of a IMM with a RPF is not possible for real time application, as explained in Section 2.5.7.2. Then the advantages of the RPF, including estimating multimodal states, are lost.

To overcome all these limitations listed above, a new method is proposed in the following section, allowing estimation of a multimodal state and abrupt additive fault without changing



(a) RPF with a large process noise compared to the fault amplitude estimating a fault.



(b) RPF with a small process noise compared to the fault amplitude estimating a fault.

Figure 4.3: RPF performing estimation of additive abrupt faults over 21 time step, with a system with a large process noise regarding the fault amplitude — (a) — and one with a small process noise regarding the fault amplitude — (b). The size of the dots corresponding to the weight of the particles at the update step. The system used is the one of Figure 4.2

the process noise regarding the amplitude of the fault, and without degrading the state estimate in a fault free situation.

4.3 PRINCIPLE OF THE JUMP-MARKOV REGULARIZED PARTICLE FILTER

A way to overcome the limitations listed above is to provide a process model that takes into account abrupt changes. To do so, a JMS is used. Indeed, JMS can handle abrupt changes by switching from one mode to another. This mode switching is performed on the basis of transition probabilities that model the potentiality of changing the current system mode to another mode, for example non-faulty to faulty or faulty to non-faulty. These transition probabilities are used in the allocation of the particles by moving only some particles proportionally to the probabilities of the other mode. Then, a JMS can be associated with each particle and each particle can switch between modes and then be moved when they switch. For fault estimation, this JMS represents two modes:

- The mode $m^{(0)}$, which represents the fault-free (or the nominal) mode;
- The mode $m^{(1)}$, which represents the faulty mode.

The transition between those two modes are defined by a transition probability matrix here denoted $\mathbf{\Pi}$, it is given by:

$$\mathbf{\Pi} = \begin{bmatrix} \pi_{00} & \pi_{10} \\ \pi_{01} & \pi_{11} \end{bmatrix} \tag{4.2}$$

where π_{10} is the probability to switch from fault-free mode $m^{(0)}$ to faulty mode $m^{(1)}$ while the probability π_{01} is the probability to switch from faulty mode $m^{(1)}$ to fault-free mode $m^{(0)}$. The π_{ii} entries are the probabilities that the system remain in the same mode. Therefore, $\pi_{00} = 1 - \pi_{10}$ and $\pi_{11} = 1 - \pi_{01}$.

The associated Markov chain of the JMS used is illustrated in Figure 4.4.

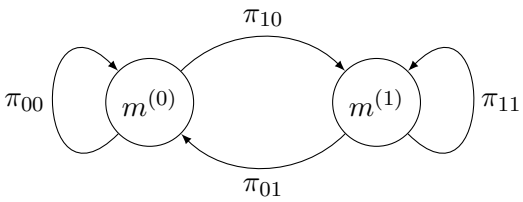
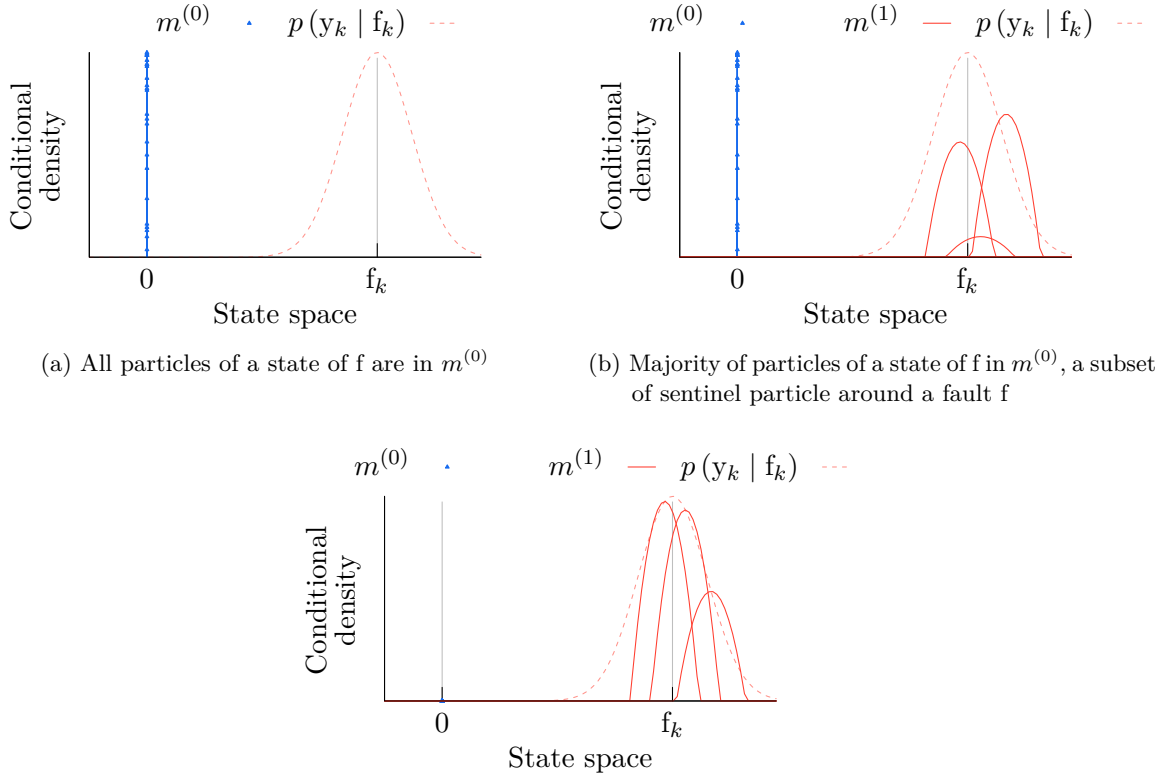


Figure 4.4: Markov chain of the JMS for fault estimation

Unlike the multiple model architecture where multiple modes are propagated in parallel and interact to determine the estimated state as shown in Figure 2.18, here a Markovian jump model is used to switch between model corresponding to the fault modes. In the approach described here, only two discrete fault modes are considered. The mode vector associated with the actuator faults vector $\mathbf{f_a}$ is denoted $\mathbf{m_a}$. The mode vector associated with the actuator faults vector $\mathbf{f_s}$ is denoted $\mathbf{m_s}$. The concatenation of the actuator fault vector $\mathbf{f_a}$ and sensor fault vector $\mathbf{f_s}$ is denoted $\mathbf{f}$, and the concatenation of the mode vector $\mathbf{m_a}$ and $\mathbf{m_s}$

is denoted $\mathbf{m}$. The value of each fault estimate vector $\mathbf{f_a}$ and $\mathbf{f_s}$ depends on the mode. Indeed, if a mode is associated with a fault estimate which is set to $m^{(0)}$, then the state estimate cannot be different from 0 since the state is in a fault-free mode. In the same way, if a mode is associated with a fault estimate which is set to $m^{(1)}$, then its value cannot be equal to 0 since the state is estimating a fault.

When a state of a particle of $\mathbf{f}$ is in a fault-free mode, the process noise and the regularization have no effect on the state of this particle, since it is set to zero. The propagation of the particle with the process noise and the regularization noise only affects the state in faulty modes. When a state of a particle switches from a fault-free to a faulty mode, a value different from zero must be assigned to it. This value, denoted Δ_{f}, must be as close as possible to the potential fault for the method to be effective. The particles that switch from fault-free to faulty modes are sometimes called sentinel particles in this thesis to indicate that they are placed around a potential alternate fault mode to test its likelihood. This process is illustrated in Figure 4.5 at time step k.



(a) All particles of a state of f are in $m^{(0)}$

(b) Majority of particles of a state of f in $m^{(0)}$, a subset of sentinel particle around a fault f

(c) The more likely particles are the sentinel particles around a fault f

Figure 4.5: Fault estimation using a JMS with a RPF, with $\Delta_{\mathrm{f}_k}^i = \mathrm{f}_k^i + \eta_k^i$

In Figure 4.5a, all the particles are in a fault-free mode $m^{(0)}$, while a fault occurs. By keeping all the particle in this mode, the fault cannot be estimated. A few particles are therefore selected according to the transition probability π_{10} to be in mode $m^{(1)}$ and then moved around the fault value f_k. This is performed at the prediction step, as illustrated in Figure 4.5b. Since the fault is not yet estimated, the value Δ_f at which the particles are moved is a prediction. In the illustration of the method, the prediction is assumed to be correct. The detail of how this predicted value is chosen is presented later in this chapter. After moving the particles around the fault, the likelihoods of the particles are computed. Then, since the sentinel particles are the particles nearest to the maximum likelihood value — which is at the fault value — the likelihood of these particles should be the highest of all the particles of the state. This is illustrated in Figure 4.5c. Since the likelihood of the sentinel particles is higher than the fault-free particles, the majority of the particles should be placed around the sentinel particles due to the resampling and regularization step, and the fault is then estimated by this cloud of particles.

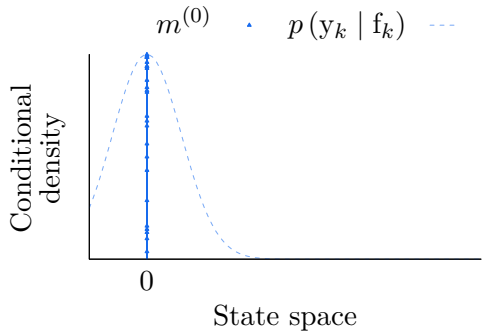
In the previously described case, illustrated by Figure 4.5, the sentinel particles — particles with discrete state $m^{(1)}$ — are placed around a potential fault. For this method to work, a potential fault value will have to be obtained, as described in Section 4.4. Moreover, since this method is performed with a subset of particles selected at each time step whose cardinality depends on a constant transition probability, it is clear that particles can be moved away from zero while the system is in the fault-free state. This situation is illustrated in Figure 4.6.

In Figure 4.6a, all the particles are in a fault-free mode $m^{(0)}$, and no fault occurs. However, since the transition from the fault-free to the faulty mode depends on probabilities, particles can switch to faulty mode when there is no fault. This is illustrated in Figure 4.6b. However, if the particles are moved away from zero while there is no fault, then their likelihoods at the update step should be smaller than the likelihood of the particles in the fault-free mode. This is illustrated in Figure 4.6c. Then, they are not contributing — or more precisely, their contributions are negligible — at the estimation step. At the resampling and regularization step, the majority of the particles should be placed in $m^{(0)}$, around 0 and the sentinel particles are likely not to be duplicated, and even removed.

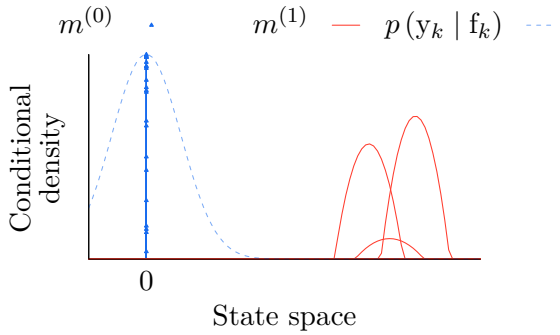
Note that in both Figure 4.5 and Figure 4.6, the weight is computed without considering potential state. However, in a real situation or simulation, a single weight is computed for the whole state vector $\mathbf{x}$.

Then, this method should allow the use of a small process noise and track a fault with a large amplitude compared to the process noise. Figure 4.7 illustrates an example of the solution proposed over 21 time steps with the same process noise as Figure 4.3b and the same amplitude. Using the JMS, it becomes possible to estimate the fault.

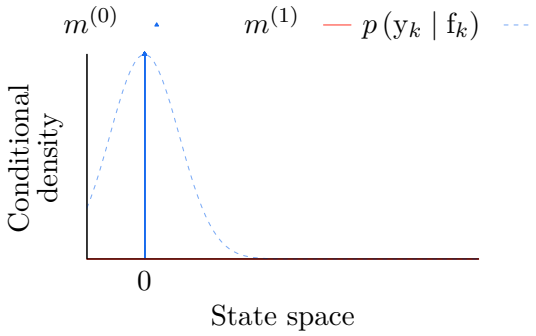
In Figure 4.7, the RPF associated with a JMS corresponds to the same precision obtained with the small process from 4.3b when the system is in a faulty situation, and even enhances accuracy in a faulty-free situation. Moreover, the fast convergence to the fault is comparable to the RPF with the large process from 4.3a.



(a) All particle of fault state f are in $m^{(0)}$



(b) The Majority of particles of fault state f in $m^{(0)}$, a subset of sentinel particle moved away from 0



(c) The more likely particles are the particle in $m^{(0)}$

Figure 4.6: Fault estimation using a JMS with a RPF, in a fault free situation ($\mathrm{f}_k = 0$) with $\Delta^i_{\mathrm{f}_k} \neq \mathrm{f}_k + \eta^i_k$

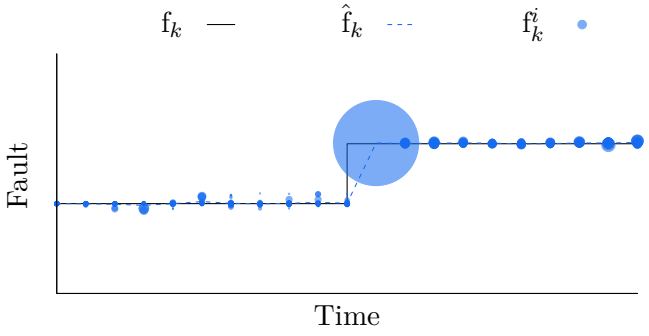


Figure 4.7: RPF associated with a JMS performing estimation of additive abrupt sensor faults over 21 time step. The size of the dots correspond to the weight of the particles at the update step. The system used is the one of Figure 4.2 for figure (c) and (d).

Since the proposed solution is based on a RPF and uses a JMS to achieve all the above requirements, the proposed filter is called the JMRPF [71]. This JMRPF belongs to the class of jump-Markov particle filters presented in Section 2.5.8.

4.4 FORMULATION OF THE JUMP-MARKOV REGULARIZED PARTICLE FILTER

The stochastic process model of the JMRPF, for additive actuator and sensor fault, is given by:

$$\begin{cases} \mathbf{m}_{\mathbf{a}k} \sim p\left(\mathbf{m}_{\mathbf{a}k} | \mathbf{m}_{\mathbf{a}k-1}\right) & (4.3a) \\ \mathbf{m}_{\mathbf{s}k} \sim p\left(\mathbf{m}_{\mathbf{s}k} | \mathbf{m}_{\mathbf{s}k-1}\right) & (4.3b) \\ \begin{bmatrix} \mathbf{z}_k \\ \mathbf{f}_{\mathbf{a}k} \\ \mathbf{f}_{\mathbf{s}k} \end{bmatrix} = \begin{bmatrix} \mathcal{F}_k\left(\mathbf{z}_{k-1},\, \mathbf{u}_k + \mathbf{f}_{\mathbf{a}k-1}\right) \\ \mathcal{G}_{ak\mathbf{m}_{\mathbf{a}k}}\left(\mathbf{f}_{\mathbf{a}k-1}\right) \\ \mathcal{G}_{sk\mathbf{m}_{\mathbf{s}k}}\left(\mathbf{f}_{\mathbf{s}k-1}\right) \end{bmatrix} + \boldsymbol{\eta}_{k\mathbf{m}_k} & (4.3c) \\ \mathbf{y}_k = \mathcal{H}_k\left(\mathbf{z}_k\right) + \mathbf{f}_{\mathbf{s}k} + \boldsymbol{\nu}_k & (4.3d) \end{cases}$$

where $\mathcal{G}_{ak\mathbf{m}_{\mathbf{a}k}}(\cdot)$ and $\mathcal{G}_{sk\mathbf{m}_{\mathbf{s}k}}(\cdot)$ respectively represent the dynamics of the actuator and sensor faults associated with their mode vector, and $\boldsymbol{\eta}_{k\mathbf{m}_k}$ is the process noise associated with the mode vector $\mathbf{m}_k$. The process noise $\boldsymbol{\eta}_{k\mathbf{m}_k}$ depends on the mode and when a state of $\mathbf{f}$ is associated with the mode $m^{(0)}$, the process noise associated with this state is set to 0. The measurement noises are assumed to be zero-means Gaussian noises with a constant variance. The occurrence of faults on sensors translates only in a variation of mean which expresses as a variation of $\mathbf{f}_{\mathbf{s}}$. In other words, the fault type under consideration in the filter design is on the mean, not on the variance of the measurements. A generalization would be possible by adding the variance as another piecewise constant state of the extended state vector, but this fault type is beyond the scope of this thesis.

Since only the state vectors $\mathbf{f}_{\mathbf{a}}$ and $\mathbf{f}_{\mathbf{s}}$ are respectively associated with the mode vector $\mathbf{m}_{\mathbf{a}}$ and $\mathbf{m}_{\mathbf{s}}$, then the extended hybrid state vector of the JMRPF for fault estimation is given by the two-ple:

$$\left(\mathbf{x}_k,\, \mathbf{m}_k\right) = \left(\begin{bmatrix} \mathbf{z}_k \\ \mathbf{f}_{\mathbf{a}k} \\ \mathbf{f}_{\mathbf{s}k} \end{bmatrix},\, \begin{bmatrix} \mathbf{m}_{\mathbf{a}k} \\ \mathbf{m}_{\mathbf{s}k} \end{bmatrix} \right) \tag{4.4}$$

The system model (4.3) can then be written as an extended state space model:

$$\begin{cases} \mathbf{m}_k \sim p\left(\mathbf{m}_k | \mathbf{m}_{k-1}\right) & (4.5a) \\ \mathbf{x}_k = f_{k\mathbf{m}_k}\left(\mathbf{x}_{k-1},\, \mathbf{u}_k\right) + \boldsymbol{\eta}_{k\mathbf{m}_k} & (4.5b) \\ \mathbf{y}_k = h_{k\mathbf{m}_k}\left(\mathbf{x}_k\right) + \boldsymbol{\nu}_k & (4.5c) \end{cases}$$

where $f_{k\mathbf{m}_k}(\cdot)$ and $h_{k\mathbf{m}_k}(\cdot)$ respectively represent the dynamics and the measurement function of the extended state vector $\mathbf{x}$ associated with the mode vector $\mathbf{m}_k$. They are respectively given by:

$$f_{k\mathbf{m}_k}(\mathbf{x}_{k-1},\, \mathbf{u}_k) = \begin{bmatrix} \mathcal{F}_k(\mathbf{z}_{k-1},\, \mathbf{u}_k + \mathbf{f}_{\mathbf{a}k-1}) \\ \mathcal{G}_{ak\mathbf{m}_{\mathbf{a}k}}(\mathbf{f}_{\mathbf{a}k-1}) \\ \mathcal{G}_{sk\mathbf{m}_{\mathbf{s}k}}(\mathbf{f}_{\mathbf{s}k-1}) \end{bmatrix} \tag{4.6a}$$

$$h_{k\mathbf{m}_k}(\mathbf{x}_k) = \mathcal{H}_k(\mathbf{z}_k) + \mathbf{f}_{\mathbf{s}k} \tag{4.6b}$$

Since only two different modes are considered, and given that the state of $\mathbf{f}$ associated with the mode must be equal to zero when the mode is in the fault free state $m^{(0)}$, then in a fault free situation the system and measurements functions $f_k(\cdot)$ and $h_k(\cdot)$ are respectively given in the fault free mode by:

$$f_{k\mathbf{m}_k^{(0)}}(\mathbf{x}_{k-1},\, \mathbf{u}_k) = \begin{bmatrix} \mathcal{F}_k(\mathbf{z}_{k-1},\, \mathbf{u}_k) \\ 0 \\ 0 \end{bmatrix} \tag{4.7a}$$

$$h_{k\mathbf{m}_k^{(0)}}(\mathbf{x}_k) = \mathcal{H}_k(\mathbf{z}_k) \tag{4.7b}$$

where $\mathbf{m}_k^{(0)}$ represents the case when all the states of the mode vector $\mathbf{m}_k$ are in $m^{(0)}$. Moreover, the process noise is set to 0 for a state associated with a mode in $m^{(0)}$ — but the process noise on the state vector $\mathbf{z}_k$ cannot be deactivated since it is not associated with a mode vector. Then, when a particle $\mathbf{x}_k^i$ has all its modes set to $m^{(0)}$, it is equivalent to performing the estimation of $\mathbf{z}_k$.

4.4.1 *Prediction step*

The prediction step of the JMRPF aims to predict the hybrid state vector $(\mathbf{x}_k,\, \mathbf{m}_k)$. The prior state density $p(\mathbf{x}_k|\mathbf{m}_k, \mathbf{Y}_{k-1})$ is obtained by the Chapman-Kolmogorov equation (2.22), and is given by:

$$p(\mathbf{x}_k|\mathbf{m}_k, \mathbf{Y}_{k-1}) = \int_{\mathbb{R}^{n_x}} p(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{m}_k)\, p(\mathbf{x}_{k-1}|\mathbf{m}_k, \mathbf{Y}_{k-1})\, \mathrm{d}\mathbf{x}_{k-1}. \tag{4.8}$$

This prior state density is then approximated by the distribution of Dirac:

$$p(\mathbf{x}_k|\mathbf{m}_k, \mathbf{Y}_{k-1}) \approx \sum_{i=1}^{N} w_{k-1}^i \delta\left(\mathbf{x}_k - \mathbf{x}_{k|k-1}^i\right) \tag{4.9}$$

where $\mathbf{x}_{k|k-1}^i$ are propagated using the following probability transition density:

$$\mathbf{x}_{k|k-1}^i \sim p\left(\mathbf{x}_k|\mathbf{x}_{k-1}^i, \mathbf{m}_k^i\right) \tag{4.10}$$

where $\mathbf{m}_k^i$ is the mode vector of the i^{th} particle that has been predicted. The prediction of the modes is performed using (4.5a), where the density $p(\cdot)$ is chosen to be uniform, since it is assumed that there is no prior information on the fault mode. The prediction of a discrete mode is illustrated in Figure 4.8.

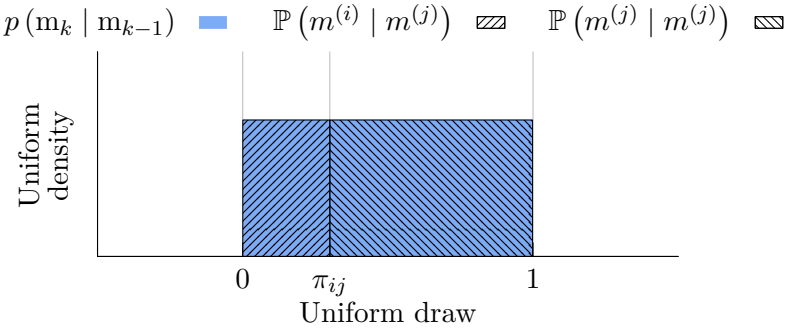


Figure 4.8: Prediction of a mode of the JMRPF.

Then, the j^{th} mode of the i^{th} particle of the mode vector $\mathbf{m}_k$ denoted $\mathbf{m}_k^{i,j}$ is predicted as follows:

$$\mathbf{m}_k^{i,j} = \begin{cases} m^{(0)} & \text{if } \upsilon_k \leq \pi_{00}^j \text{ and } \mathbf{m}_{k-1}^{i,j} = m^{(0)} \\ m^{(1)} & \text{if } \upsilon_k \leq \pi_{10}^j \text{ and } \mathbf{m}_{k-1}^{i,j} = m^{(0)} \\ m^{(0)} & \text{if } \upsilon_k \leq \pi_{01}^j \text{ and } \mathbf{m}_{k-1}^{i,j} = m^{(1)} \\ m^{(1)} & \text{if } \upsilon_k \leq \pi_{11}^j \text{ and } \mathbf{m}_{k-1}^{i,j} = m^{(1)} \end{cases} \tag{4.11}$$

where π^j is the transition probability associated with the j^{th} mode of the mode vector $\mathbf{m}_k$, and $\upsilon_k \sim \mathcal{U}(0, 1)$

Then, the j^{th} state of the i^{th} particle of state vector $\mathbf{f}_k$ denoted $\mathbf{f}_k^{i,j}$ and which is associated with the mode $\mathbf{m}_k^{i,j}$ is given by:

$$\mathbf{f}_{k|k-1}^{i,j} = \begin{cases} 0 & \text{if } \mathbf{m}_k^{i,j} = m^{(0)} \text{ and } \mathbf{m}_{k-1}^{i,j} = m^{(0)} \\ \Delta_{\mathbf{f}_k}^{i,j} & \text{if } \mathbf{m}_k^{i,j} = m^{(1)} \text{ and } \mathbf{m}_{k-1}^{i,j} = m^{(0)} \\ 0 & \text{if } \mathbf{m}_k^{i,j} = m^{(0)} \text{ and } \mathbf{m}_{k-1}^{i,j} = m^{(1)} \\ \mathbf{f}_{k|k-1}^{i,j} & \text{if } \mathbf{m}_k^{i,j} = m^{(1)} \text{ and } \mathbf{m}_{k-1}^{i,j} = m^{(1)} \end{cases} \tag{4.12}$$

The prediction of the mode vector $\mathbf{m}_k$ and the computation of the state vector $\mathbf{f}_k$ regarding the value of the mode vector is the jump step. The jump step is described in Algorithm 4.1, where the transition from $m^{(1)}$ to $m^{(1)}$ is not covered since no change to the value of f_k is needed. However, the transition from $m^{(0)}$ to $m^{(0)}$ is covered to avoid potential noise introduced on this state before calling the function JUMP.

One difficulty with this method is the need to provide a value for the predicted fault amplitude Δ_{f}. This value is computed differently in the case of sensor or actuator faults.

Algorithm 4.1 Jump step of the jump-Markov regularized particle filter

Function JUMP(f_k, m_k, $\Delta_{\mathrm{f}k}$)
- $\upsilon_k \sim \mathcal{U}(0, 1)$
- **if** $\mathrm{m}_k = m^{(0)}$ **then**
  - **if** $\upsilon_k \leq \pi_{10}$ **then** //Transition $m^{(0)} \rightarrow m^{(1)}$
    - $\mathrm{f}_k \leftarrow \Delta_{\mathrm{f}k}$ //See (4.12)
    - $\mathrm{m}_k \leftarrow m^{(1)}$ //See (4.11)
  - **else** //Transition $m^{(0)} \rightarrow m^{(0)}$
    - $\mathrm{f}_k \leftarrow 0$ //See (4.12)
- **else if** $\mathrm{m}_k = m^{(1)}$ **then**
  - **if** $\upsilon_k \leq \pi_{01}$ **then** //Transition $m^{(1)} \rightarrow m^{(0)}$
    - $\mathrm{f}_k \leftarrow 0$ //See (4.12)
    - $\mathrm{m}_k \leftarrow m^{(0)}$ //See (4.11)

4.4.1.1 *Predicted value of the fault amplitude for actuator faults*

The predicted value of the fault amplitude for actuator fault is denoted $\Delta_{\mathrm{f_a}}$. Based on the work by Saif and Guan [72], on a linear system it is deduced that the actuator fault can be given by:

$$\mathbf{f_a}_{k-1} = \mathbf{B}_k^* \left(-\mathbf{F}_k \mathbf{z}_{k-1} + \mathbf{z}_k - \mathbf{B}_k \mathbf{u}_{k-1}\right), \tag{4.13}$$

where $\mathbf{B}_k^* = \left(\mathbf{B}_k^\top \mathbf{B}_k\right)^{-1} \mathbf{B}_k^\top$. However, $\mathbf{z}_k$ is not known before the estimation step, therefore the predicted state $\mathbf{z}_{k|k-1}$ is used. The predicted values of the fault amplitudes for actuator fault vector $\mathbf{f_a}$ denoted $\Delta_{\mathbf{f_a}}$ are then given by:

$$\Delta_{\mathbf{f_a}k} = \mathbf{B}_k^* \left(-\mathbf{F}_k \hat{\mathbf{z}}_{k-1} + \mathbf{z}_{k|k-1}^i - \mathbf{B}_k \mathbf{u}_{k-1}\right) \tag{4.14}$$

4.4.1.2 *Predicted value of the fault amplitude for sensor faults*

The predicted value of the fault amplitude for sensor faults is denoted $\Delta_{\mathrm{f_s}}$. The measurements' equation with an additive fault and without considering the noise is given by (2.3b). The additive sensor fault is then given by:

$$\mathbf{f_s}_k = \mathbf{y}_k - \mathcal{H}_k\left(\mathbf{z}_k\right) \tag{4.15}$$

The true value of the state vector $\mathbf{z}_k$ is unknown, hence the value of $\mathbf{f}_k$ cannot be obtained. However, the idea here is to obtain a predicted value of the fault when the state vector is in $m^{(0)}$ and switches to $m^{(1)}$. This means that the predicted value of the particle $\mathbf{z}_k^i$ does not depend on the mode and can be obtained using (4.7). Then, $\mathbf{z}_{k|k-1}^i$ is known, and the

predicted values of the fault amplitudes for the sensor faults vector $\mathbf{f_s}$ denoted $\Delta_{\mathbf{f_s}}$ is given by:

$$\Delta_{\mathbf{f_s}k} = \mathbf{f_s}^i_{k|k-1} \tag{4.16a}$$

$$= \mathbf{y}_k - \mathcal{H}_k\left(\mathbf{z}^i_{k|k-1}\right) \tag{4.16b}$$

The prediction step of the JMRPF is described in Algorithm 4.2, using the previously defined function JUMP.

Algorithm 4.2 Prediction step of the jump-Markov regularized particle filter

Function PREDICT($\mathbf{x}^{1:N}_{k|k-1}$, $\mathbf{x}^{1:N}_{k-1}$, $\mathbf{m}^{1:N}_k$, $\mathbf{u}_k$, $\mathbf{y}_k$)

- **for each** $i \in [1, N]$ **do**
  - $\boldsymbol{\eta}^i_k \sim \mathcal{N}(0, \mathbf{Q}_k)$
  - $\mathbf{x}^i_{k|k-1} \leftarrow f_k\left(\mathbf{x}^i_{k-1}, \mathbf{u}_k\right) + \boldsymbol{\eta}^i_k$ //See (4.10)
  - //Jump step of state vector $\mathbf{f_a}_k$
  - $\Delta_{\mathbf{f_a}}{}^i_k \leftarrow \mathbf{u}_k - \mathrm{CTL}\left(\mathbf{z}^i_{k|k-1}, \mathbf{r}^c_k\right)$ //See (4.14)
  - **for each** $j \in [1, n_{f_a}]$ **do**
    - JUMP($\mathbf{f_a}^{i,j}_{k|k-1}$, $\mathbf{m_a}^{i,j}_k$, $\Delta_{\mathbf{f_a}}{}^{i,j}_k$) //See Algorithm 4.1
  - //Jump step of state vector $\mathbf{f_s}_k$
  - $\Delta_{\mathbf{f_s}}{}^i_k \leftarrow \mathbf{y}_k - h_k\left(\mathbf{x}^i_{k|k-1}\right)$ //See (4.16)
  - **for each** $j \in [1, n_{f_s}]$ **do**
    - JUMP($\mathbf{f_s}^{i,j}_{k|k-1}$, $\mathbf{m_s}^{i,j}_k$, $\Delta_{\mathbf{f_s}}{}^{i,j}_k$) //See Algorithm 4.1

4.4.2 *Update step*

The update step of the JMRPF aims to update the hybrid state vector $(\mathbf{x}_k, \mathbf{m}_k)$.

The posterior density $p(\mathbf{x}_k|\mathbf{m}_k, \mathbf{Y}_k)$ is obtained by the Bayes formula given in (2.23) which gives:

$$p(\mathbf{x}_k|\mathbf{m}_k, \mathbf{Y}_k) = \frac{p(\mathbf{y}_k|\mathbf{x}_k, \mathbf{m}_k)\, p(\mathbf{x}_k|\mathbf{m}_k, \mathbf{Y}_{k-1})}{\displaystyle\int_{\mathbb{R}^{n_x}} p(\mathbf{y}_k|\mathbf{x}_k, \mathbf{m}_k)\, p(\mathbf{x}_k|\mathbf{m}_k, \mathbf{Y}_{k-1})\, \mathrm{d}\mathbf{x}_k} \tag{4.17}$$

This posterior density is then approximated by a weighted sum of Dirac distributions:

$$p(\mathbf{x}_k|\mathbf{m}_k, \mathbf{Y}_k) \approx \sum_{i=1}^{N} w^i_k \delta_{\mathbf{x}^i_k}(\mathbf{x}_k) \tag{4.18}$$

where the weights w_k^i are proportional to the likelihood and are calculated by the following formula:

$$w_k^i \propto w_{k-1}^i p\left(\mathbf{y}_k | \mathbf{x}_{k|k-1}^i, \mathbf{m}_k^i\right). \tag{4.19}$$

The weights are then normalized to ensure (2.74). The algorithm of the update step of the JMRPF is the same as the one from the SIR particle filter detailed in Algorithm 2.3.

4.4.3 *Estimation step*

The estimation step aims to obtain a global estimate of the state vector $\hat{\mathbf{x}}_k$, with its associated covariance matrix $\hat{\mathbf{P}}_k$. This step is the same as the one from the SIR particle filter, detailed in Section 2.5.6.2. The algorithm of the estimation step of the JMRPF is the same as the one from the SIR particle filter detailed in Algorithm 2.4.

4.4.4 *Regularization-Resampling Step*

The regularization-resampling step aims to remove the particles with a low likelihood and replace them by duplicating the particles with a high likelihood and regularizing the duplicated particles. This step is the same as the one from the RPF detailed in Section 2.5.6.3. However, the joint posterior density is now approached by:

$$p\left(\mathbf{x}_k, \mathbf{m}_k | \mathbf{Y}_k\right) \approx \sum_{i=1}^{N} w_k^i \mathcal{K}_h\left(\mathbf{x}_k - \mathbf{x}_k^i\right) \delta_{\mathbf{m}_k^i}\left(\mathbf{m}_k\right) \tag{4.20}$$

The algorithm for the regularization performed by the JMRPF is given by function REGULARIZE in Algorithm 2.7.

The JMRPF is presented in Algorithm 4.3, using the previously defined function PREDICT, and the function used by the RPF.

Algorithm 4.3 Jump-Markov regularized particle filter

$k \leftarrow 0$
⋮ //Initialization
Loop
  $k \leftarrow k+1$
  PREDICT($\mathbf{x}_{k|k-1}^{1:N}$, $\mathbf{x}_{k-1}^{1:N}$, $\mathbf{m}_k^{1:N}$, $\mathbf{u}_k$, $\mathbf{y}_k$) //See Algorithm 4.2
  UPDATE($w_k^{1:N}$, $w_{k-1}^{1:N}$, $\mathbf{x}_{k|k-1}^{1:N}$, $\mathbf{y}_k$) //See Algorithm 2.3
  ESTIMATE($\hat{\mathbf{x}}_k$, $\hat{\mathbf{P}}_k$, $\mathbf{x}_{k|k-1}^{1:N}$, $w_k^{1:N}$) //See Algorithm 2.4
  $\hat{N}_{eff} \leftarrow \frac{1}{\sum_{i=1}^{N} \left(w_k^i\right)^2}$ //See (2.53)
  if $\hat{N}_{eff} \leq N\Gamma_{rspl}$ **then** //if true then resample
    MULTINOMIAL($\acute{\mathbf{x}}_k^{1:N}$, $\mathbf{x}_{k|k-1}^{1:N}$, $w_k^{1:N}$) //See Algorithm 2.5
    for each $i \in [1, N]$ **do**
      $w_k^i \leftarrow \frac{1}{N}$ //Reset the weights, See (2.55)
    REGULARIZE($\mathbf{x}_k^{1:N}$, $\acute{\mathbf{x}}_k^{1:N}$, $\hat{\mathbf{P}}_k$) //See Algorithm 2.7

4.5 COMPARATIVE NUMERICAL SIMULATION ANALYSIS

In this section the ability of the JMRPF to perform fault estimation when an ambiguous sensor fault scenario occurs is evaluated. For the sake of brevity, only the longitudinal system is considered.

Since it is one of the main limitations of the interacting multiple model Kalman filters (IMM-KF), a comparative simulation of both methods is performed to see how the JMRPF overcomes this limitation compared to the IMM-KF. To separately evaluate the effect of sensor ambiguities with the IMM-KF and JMRPF without the added effect of non-linearities, linear process and observation models are used. The model used in this section is then the one described by the longitudinal equation from (3.48) with trim point set to $40\,\mathrm{m\,s^{-1}}$ for the airspeed velocity, 500 m for the altitude, 0 rad for the flight path angle in a straight flight. The control and guidance are designed as described in Section 3.7 and Section 3.8 for the longitudinal system. The desired altitude is set to 500 m and the desired velocity to $40\,\mathrm{m\,s^{-1}}$. Since ambiguous sensor faults must be simulated, the measurements vector contains two measurements of the altitude from two different sensors: a GNSS receiver and a barometer.

The linear system used for the true state computation is then given by:

$$\begin{cases} \begin{bmatrix} \dot{\bar{p}}_d \\ \dot{\bar{u}} \\ \dot{\bar{w}} \\ \dot{\bar{\theta}} \\ \dot{\bar{q}} \end{bmatrix} = \begin{bmatrix} 0 & -0.05 & 1 & -40 & 0 \\ 0 & -0.42 & 0.27 & -9.80 & -1.91 \\ 0 & -0.40 & -1.81 & -0.47 & 39.95 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0.04 & -0.82 & 0 & -0.74 \end{bmatrix} \begin{bmatrix} \bar{p}_d \\ \bar{u} \\ \bar{w} \\ \bar{\theta} \\ \bar{q} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ -0.35 & 32.23 \\ 7.39 & 0 \\ 0 & 0 \\ -43 & 0 \end{bmatrix} \begin{bmatrix} \bar{\delta}_e \\ \bar{\delta}_t \end{bmatrix} & \text{(4.21a)} \\ \mathbf{y} = \begin{bmatrix} -1 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \bar{p}_d \\ \bar{u} \\ \bar{w} \\ \bar{\theta} \\ \bar{q} \end{bmatrix} + \begin{bmatrix} f_g \\ f_b \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} \nu_{GNSS,-p_d} \\ \nu_{baro,-p_d} \\ \nu_{accel,u} \\ \nu_{accel,w} \\ \nu_{gyro,\theta} \\ \nu_{gyro,q} \end{bmatrix} & \text{(4.21b)} \end{cases}$$

where f_g is the fault state estimate of associated with the GNSS receiver fault, and f_b is the fault state estimate of associated with the barometer fault, and where the measurement noise is a zero mean Gaussian noise with standard deviations for each sensors respectively given by $\sigma_{GNSS,-p_d} = 5\,\mathrm{m}$, $\sigma_{baro,-p_d} = 1\,\mathrm{m}$, $\sigma_{accel,u} = 1\,\mathrm{m\,s^{-1}}$, $\sigma_{accel,w} = 1\,\mathrm{m\,s^{-1}}$, $\sigma_{gyro,\theta} = 0.01\,\mathrm{rad}$ and $\sigma_{gyro,q} = 0.002\,\mathrm{rad\,s^{-1}}$. The standard deviation of the measurement noise used by the estimation filters is 1.5 times these standard deviation used for the true measurements.

The faults considered for the simulation are abrupt additive faults. Since only one dynamic fault model is considered, the IMM-KF is designed with a bank of two Kalman filters. One for the nominal mode, and one for the faulty mode. The output matrix for the process model of the nominal mode denoted $\mathbf{H}_{m^{(0)}}$ is given by:

$$\mathbf{H}_{m^{(0)}} = \begin{bmatrix} -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}, \tag{4.22}$$

and the output matrix for the process model of the faulty mode denoted $\mathbf{H}_{m^{(1)}}$ of the IMM-KF which is also the output matrix of the JMRPF is given by:

$$\mathbf{H}_{m^{(1)}} = \begin{bmatrix} -1 & 0 & 0 & 0 & 0 & 1 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}. \tag{4.23}$$

Since the faults are on the measurements only, the evolution and control input matrices for the process without the discrete mode are the same for both modes of the IMM-KF, and they are also used by the JMRPF. The linear process model used by filters is of the following form:

$$\begin{cases} \mathbf{x}_k = \mathbf{F}_k \mathbf{x}_{k-1} + \mathbf{B}_k \mathbf{u}_k + \boldsymbol{\eta}_k & \text{(4.24a)} \\ \mathbf{y}_k = \mathbf{H}_k \mathbf{x}_k + \boldsymbol{\nu}_k & \text{(4.24b)} \end{cases}$$

where the evolution matrix $\mathbf{F}_k$ and the control input matrix $\mathbf{B}_k$ are given by:

$$\mathbf{F}_k = \begin{bmatrix} 1 & 0 & 0.05 & -2 & 0 & 0 & 0 \\ 0 & 0.98 & 0.01 & -0.49 & -0.09 & 0 & 0 \\ 0 & -0.02 & 0.88 & -0.02 & 1.85 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0.05 & 0 & 0 \\ 0 & 0 & -0.04 & 0 & 0.93 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.25}$$

and

$$\mathbf{B}_k = \begin{bmatrix} 0.01 & 0 \\ 0.08 & 1.59 \\ -1.70 & -0.01 \\ -0.05 & 0 \\ -2.09 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}. \tag{4.26}$$

Since both the JMRPF and the IMM-KF use a JMS, they both use a transition probability matrix given by:

$$\boldsymbol{\Pi} = \begin{bmatrix} 0.99 & 0.01 \\ 0.01 & 0.99 \end{bmatrix} \tag{4.27}$$

The fault scenario under consideration lasts 50 s. For the first 10 s, no fault is active in the system. At 10 s the first abrupt fault occurs on the GNSS receiver with an amplitude of 50 m. After 10 more seconds, another abrupt fault occurs on the barometer with an amplitude of 30 m. The GNSS receiver abrupt fault is then deactivated at 30 s, followed by the barometer fault at 40 s. The scenario is illustrated in Figure 4.9.

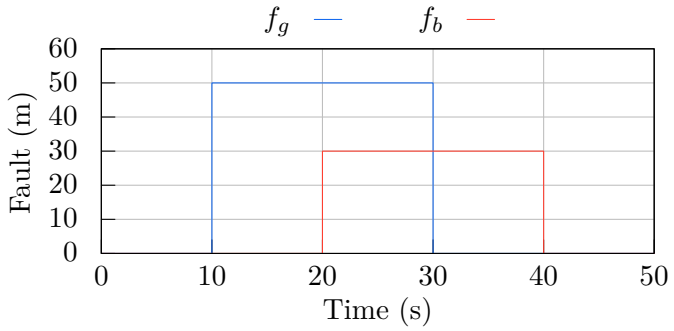


Figure 4.9: Fault scenario for simulation of ambiguous sensor faults with a GNSS receiver fault in altitude and a barometer fault in altitude.

The process noise of the JMRPF and the IMM-KF is the same for the state vector $\mathbf{z}$ for all modes. It is a Gaussian process noise with a standard deviation $\sigma_{\mathbf{z}}$ given by:

$$\sigma_{\mathbf{z}} = \begin{bmatrix} 1\,\mathrm{m} \\ 0.1\,\mathrm{m\,s^{-1}} \\ 0.1\,\mathrm{m\,s^{-1}} \\ 0.02\,\mathrm{rad} \\ 0.002\,\mathrm{rad\,s^{-1}} \end{bmatrix} \tag{4.28}$$

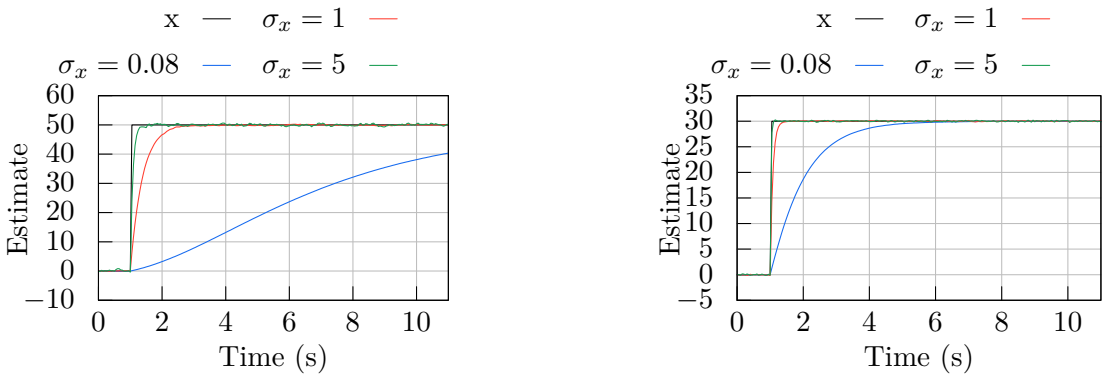
However, for the process noise of the state vector $\mathbf{f}$, the JMRPF can estimate the fault with a process noise that does not depend on the faults amplitudes, it has been empirically set with a standard deviation $\sigma_{\mathbf{f}}$ for both fault states given by:

$$\sigma_{\mathbf{f}} = \begin{bmatrix} 0.08\,\mathrm{m} \\ 0.08\,\mathrm{m} \end{bmatrix}. \tag{4.29}$$

On the other hand, to estimate faults with amplitudes of 50 m and 30 m the process noise of the Kalman filters used for the process model of the mode $m^{(1)}$ of the IMM-KF must be dimensioned accordingly. Since the mode $m^{(0)}$ of the IMM-KF is the fault-free mode, then there is no fault estimation on the process model of this mode, and the process noise of the associated state vector $\mathbf{f}$ is set with a standard deviation for both fault states of the mode $m^{(0)}$ given by:

$$\sigma_{\mathbf{f}} = \begin{bmatrix} 0\,\mathrm{m} \\ 0\,\mathrm{m} \end{bmatrix}. \tag{4.30}$$

To select the process noise of the mode $m^{(1)}$ of the IMM-KF several simulations with different process noises magnitude have been performed with a Kalman filter, to estimate a state with an abrupt change of an amplitude of 50 and 30, using the same measurements noise and fault dynamics of the UAV simulated. 100 simulations per process noise magnitude have been performed and mean estimate of the process noises selected are shown in Figure 4.10.



(a) Abrupt change of 50.

(b) Abrupt change of 30.

Figure 4.10: Mean estimate over 100 simulations performed by Kalman filter of a state x with $\dot{x} = 0$, an abrupt change of x at 1 s of 50 (a) and 30 (b), a white Gaussian measurement noise of 5 (a) and 1 (b) and different standard deviation σ_x for the Gaussian process noise.

In Figure 4.10, the estimation of a state component with an abrupt change of 50 gives a relatively fast and fairly accurate estimate of the state with a standard deviation of the process noise of 5, and 1 for an amplitude variation of 30. The process noise of the JMRPF on the other hand shows that it is too small to estimate any gap of the amplitude of the faults considered within a relative short time, if the fault is estimated using a Kalman filter. Then for the mode $m^{(1)}$ of the IMM-KF, the process noise of the state vector $\mathbf{f}$ is set with a standard deviation for both fault states given by:

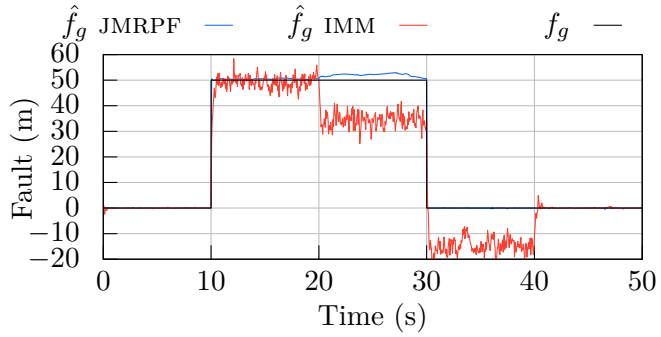
$$\sigma_{\mathbf{f}} = \begin{bmatrix} 5\,\mathrm{m} \\ 1\,\mathrm{m} \end{bmatrix} \tag{4.31}$$

The standard deviation used to compute the initial covariance matrix $\mathbf{P}_0$ for both filters, denoted $\sigma_{\mathbf{x0}}$ is given by:

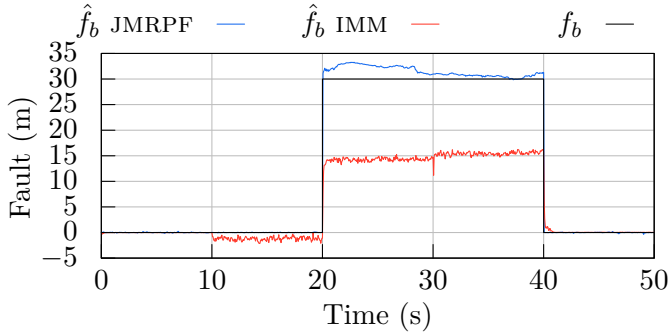
$$\sigma_{\mathbf{x0}} = \begin{bmatrix} 1\,\mathrm{m} \\ 1\,\mathrm{m\,s^{-1}} \\ 1\,\mathrm{m\,s^{-1}} \\ 0.04\,\mathrm{rad} \\ 0.01\,\mathrm{rad\,s^{-1}} \\ 0\,\mathrm{m} \\ 0\,\mathrm{m} \end{bmatrix} \tag{4.32}$$

Finally, for the JMRPF, the number of particles is set to 1000, the resampling threshold Γ_{rspl} is set to 0.15 and the bandwidth factor h of the Epanechnikov kernel of equation (2.61) is set 0.3115. The number of simulations performed is 100 with a time step of 0.05 s.

The first results illustrated in Figure 4.11 are the estimates by the JMRPF and the IMM-KF of the median results of the 100 simulations performed. The selection of the median result is detailed in Appendix D.



(a) Fault on the altitude of the GNSS receiver.



(b) Fault on the altitude of the barometer.

Figure 4.11: Median result of the fault states of the UAV under additive abrupt ambiguous sensor faults estimated by a JMRPF and a IMM. Median results based on 100 simulations.

In both Figure 4.11a and Figure 4.11b, the fault free situation of the first 10 s is properly estimated by both IMM-KF and JMRPF. The small process noise of the JMRPF and the mode $m^{(0)}$ of the IMM-KF provide a clear zero estimate. However, when the first GNSS receiver fault is activated at 10 s, differences are beginning to be noticeable between the two filters. On the JMRPF side, the estimate of the GNSS receiver fault is performed in only a time step while it took a little more time steps for the IMM-KF to converge to the fault amplitude. This is an expected result given the choice of a standard deviation of 5 m for the process noise of the GNSS receiver fault, as it is illustrated in Figure 4.10. Nonetheless, a larger process noise for the estimation of the GNSS receiver would decrease its convergence time, but also degrade its estimate by increasing its variance, which is already much larger than the one of the estimate performed by the JMRPF. The use of the mode $m^{(1)}$ by the IMM-KF also degrade the estimate of the barometer fault while it is in fault-free situation since it use a larger process noise for the fault estimate of it too. While the estimate of the barometer fault by the JMRPF in fault free situation has produced no significant change in its estimate. The use of a separate mode for the GNSS receiver fault and the barometer fault could solve this issue for the IMM-KF however the scenario used aims to highlight the capacity of the methods used in when both faults are active at the same time. This happens, at 20 s when the barometer fault is activated too. From this point, the estimate of the IMM-KF is having a significant error on both faults estimate. Indeed, only half the fault of the barometer is getting estimated and the estimate of the GNSS fault is also getting impacted by a sudden increase of the error of the fault estimate by 15 m. The JMRPF on the other side, estimates the barometer fault quickly and accurately. This deactivation of the GNSS fault slightly improve the estimate of the barometer fault by the IMM-KF but a 15 m error on both GNSS receiver and barometer fault estimate is visible. The estimate provided by the JMRPF on the other and return to zero when the GNSS fault is deactivated, no impact on the barometer fault estimated is visible and when the barometer fault is deactivated too, the situation looks similar to the first 10 s of the simulation for both IMM-KF and JMRPF estimates.

The mode selection of the IMM-KF is illustrated for the median result in Figure 4.12.

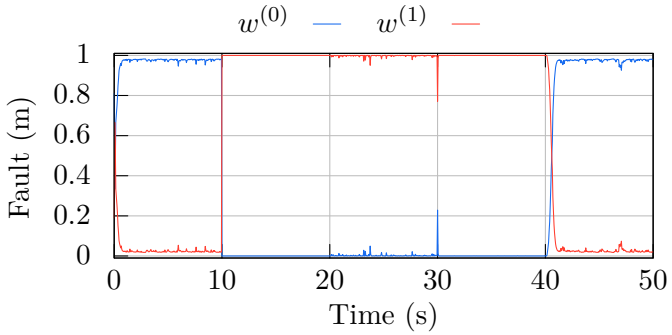
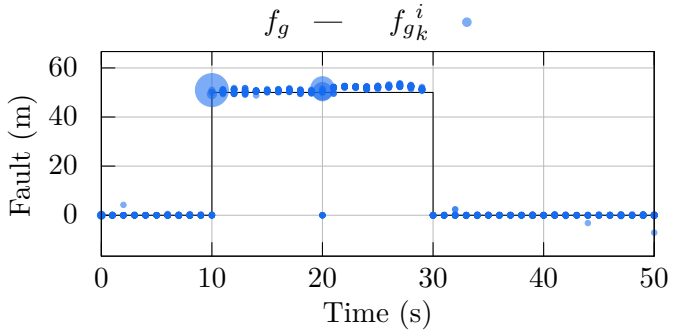


Figure 4.12: Weights of modes of the IMM-KF. Median results based on 100 simulations.

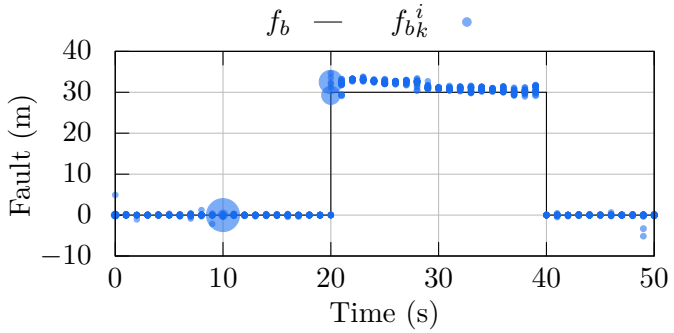
In Figure 4.12 the IMM-KF has worked as expected since the mode $m^{(0)}$ is selected with a high weight when the system is in fault free mode, and the mode $m^{(1)}$ is selected with a high

weight too when the system is in faulty mode. Hence, the noisy estimate of the IMM-KF when fault are activated due to the use of a larger process noise associated with the mode $m^{(1)}$.

The behaviour of the median results of the JMRPF is illustrated in detail in Figure 4.13, by showing the particles positions with their associated weights.



(a) Fault on the altitude of the GNSS receiver.



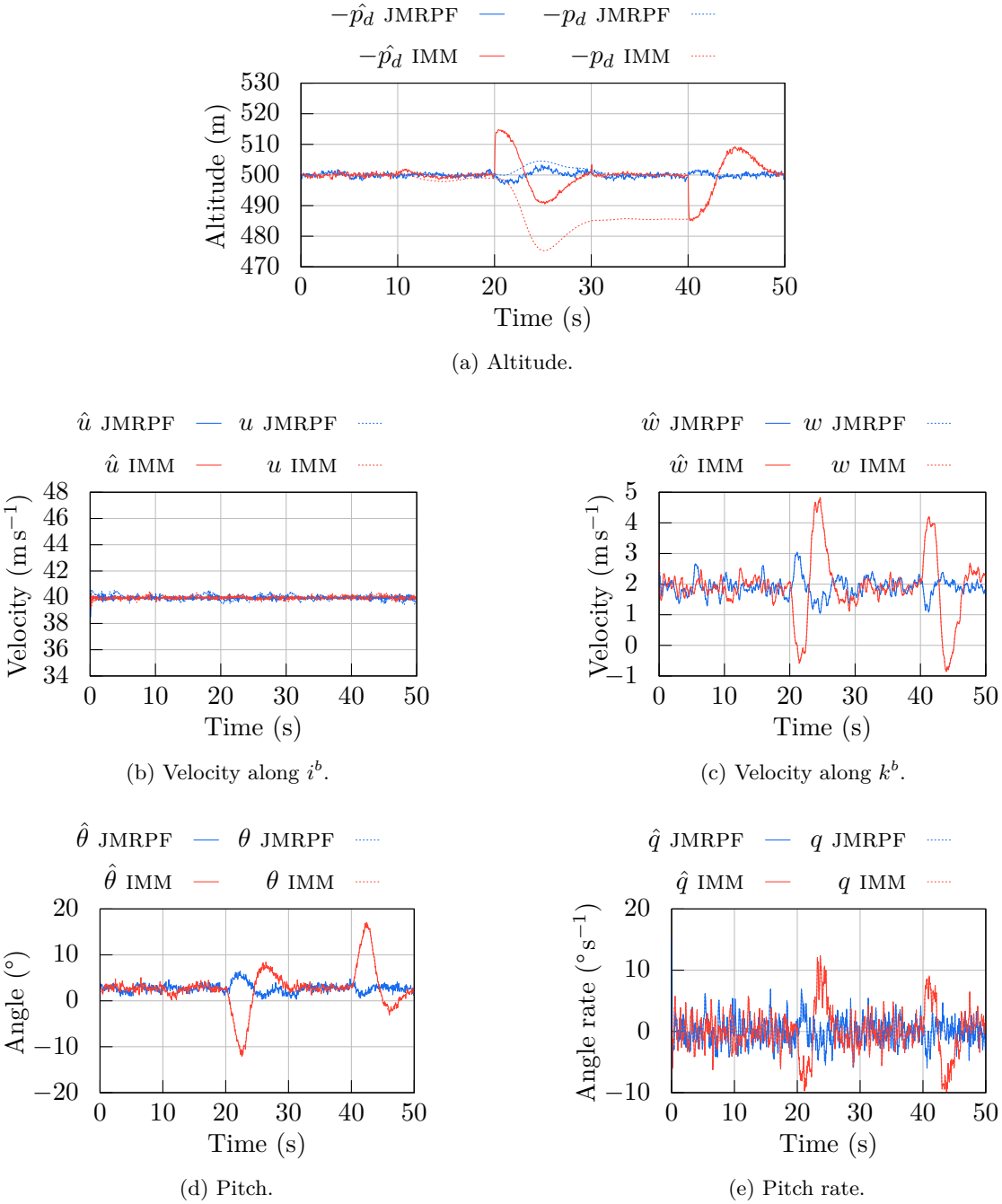
(b) Fault on the altitude of the barometer.

Figure 4.13: The 20 most weighted particles at every second of the median result of the UAV fault states under additive abrupt ambiguous sensor faults, estimated by a JMRPF. Median results based on 100 simulations.

In Figure 4.13, the particle placement has shown to be efficient since particles have been place close to the fault amplitude. The size of the dots that representing the weights shows that the likelihood of the particles is almost the same during all the simulation, expected at 10 s and 20 s, which are the moments of activation of the fault. This is due to the fact that the small amount of sentinel particles are correctly positioned after the fault occurred. For example at 10 s a small amount of particles is place around 50 m for the GNSS receiver fault and around 0 m — or even at 0 m if the particles are in $m^{(0)}$ — for the barometer fault. Since this configuration is the one that match the most the true situation of the system, it is the one that has the best likelihood. Since it only concerns a small amount of particles their likelihood is significantly higher than the rest of the particles. Then, the resampling step duplicate most of the particles in this situation — particles at 50 m for the GNSS receiver

fault and at 0 m for the barometer fault — and then the weights are closer in value to each others. Same explanation can be done for the situation at 20 s for the barometer fault.

The median results of the state vector $\mathbf{z}$ estimated by the JMRPF and the IMM-KF are shown in Figure 4.11.



(a) Altitude.

(b) Velocity along i^b.

(c) Velocity along k^b.

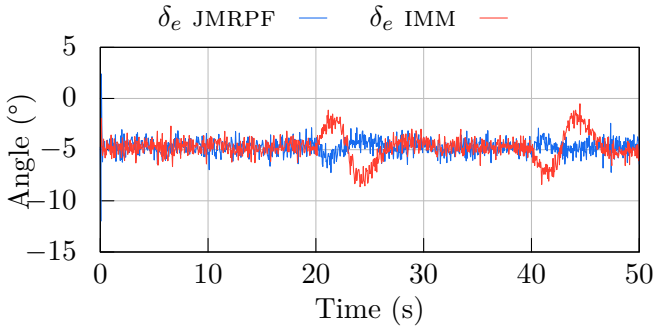
(d) Pitch.

(e) Pitch rate.

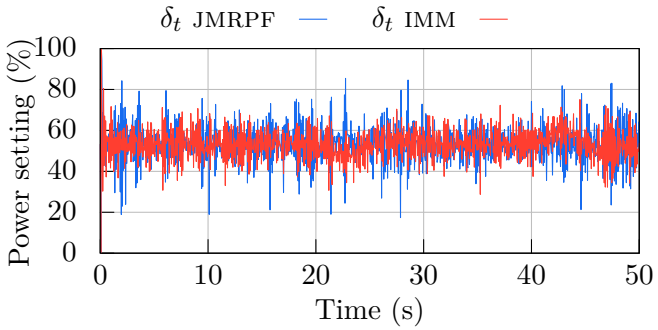
Figure 4.14: Median result of the longitudinal states of the UAV under additive abrupt ambiguous sensor faults, estimated by a JMRPF and a IMM. Median results based on 100 simulations.

In Figure 4.14, all the longitudinal states of the UAV and their estimates are represented. As expected, a bad estimation of the altitude sensor led to a bad estimation of the altitude state. In Figure 4.14a the error between the true state and the estimated state is significance for the IMM-KF estimate. The error is about 15 m as the error of the estimated fault, and it produces a dangerous deviation of trajectory for the UAV. Indeed, the UAV changes its altitude from 500 m to 485 m, without considering the overshoot due to control. The bad estimation of the error could then lead to a deterioration of the integrity of the UAV. On the others states shown in Figure 4.14, no estimation error is visible. For the estimate performed by the JMRPF, the altitude is also affected by an error, but the error is about 1.77 m at compared to the 15 m of the IMM-KF. The error then led to a slight change in the altitude trajectory but on a shorter duration. All the others states shown in Figure 4.14, and estimated by the JMRPF does not show any significant error in the estimation.

The control inputs in the case of the JMRPF and the IMM-KF estimation are shown in Figure 4.15.



(a) Elevator deflection.



(b) Throttle.

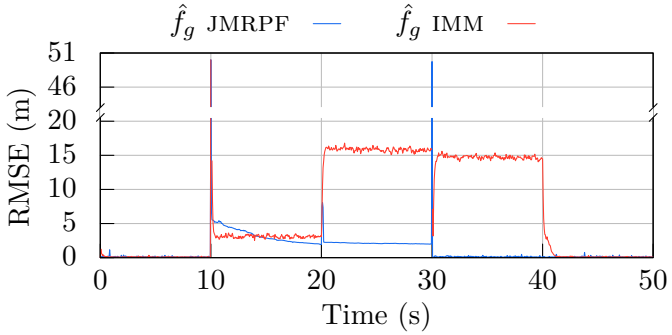
Figure 4.15: Control inputs of the UAV under additive abrupt ambiguous sensor faults, estimated by a JMRPF and a IMM-KF. Median results based on 100 simulations.

In Figure 4.15 the controls are not saturated. The change in the trajectory of the UAV due to the fault is also visible here for both filter at 20 s and 40 s. Indeed, for the elevator

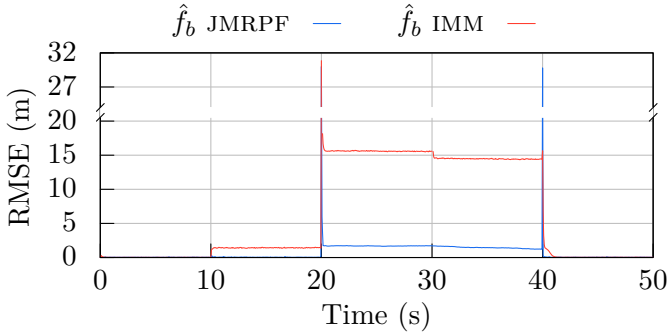
deflection, a change in the input is visible even if it is more significant for the input associated with the state estimated by a IMM-KF than the one for the JMRPF.

The results shown so far aims to show a typical result in order to illustrate the inconvenient and advantages of the methods used. However, no generalization can be made from a single simulation. To be able to conclude on the superiority of the JMRPF over the IMM-KF for the specific situation illustrated here, the root-mean-square error (RMSE) has been computed.

For the fault state vector $\mathbf{f}$, the RMSE of the JMRPF and IMM-KF are shown in Figure 4.16.



(a) RMSE of the fault estimate of the altitude of the GNSS receiver.

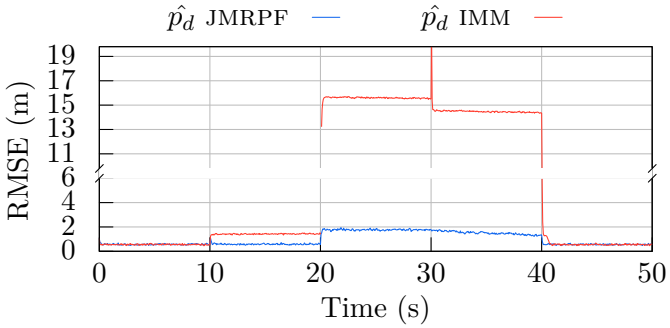


(b) RMSE of the fault estimate of the altitude of the barometer.

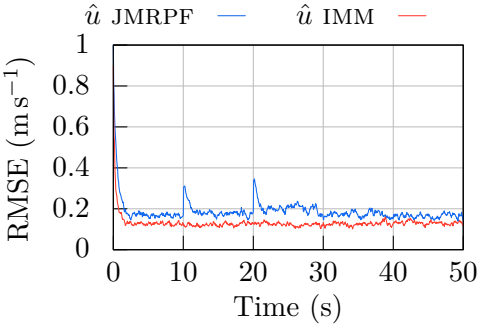
Figure 4.16: RMSE of the fault states of the UAV under additive abrupt ambiguous sensor faults, estimated by a JMRPF and a IMM-KF. Results are based on 100 simulations.

In Figure 4.16, the bad estimation of the IMM-KF from the moment when both faults are activated is confirmed by these results. However, the noisy estimate of the IMM-KF when it is in mode $m^{(1)}$ is not bad compared to the estimate of the JMRPF when it is also in mode $m^{(1)}$. This is mainly due to the fact that the particles of the JMRPF are placed accordingly to the innovation. That is, having the same standard deviation on the GNSS receiver and the barometer as the process noise of the IMM-KF of the fault state vector. Nothing else is different from what has been already observed in previous median results.

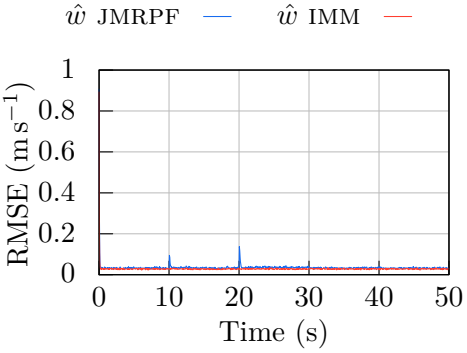
The RMSE of the states are shown in Figure 4.17.



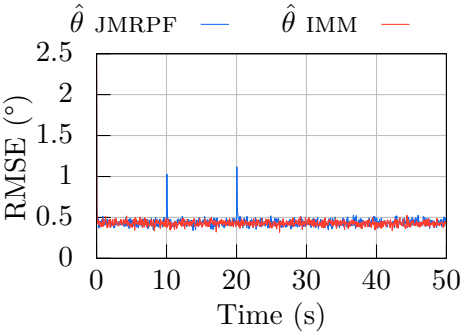
(a) RMSE of the altitude estimate.



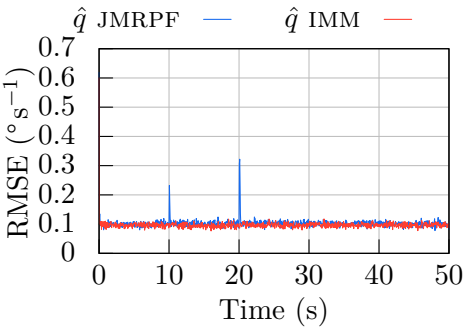
(b) RMSE of the velocity along i^b estimate.



(c) RMSE of the velocity along k^b estimate.



(d) RMSE of the pitch estimate.



(e) RMSE of the pitch rate estimate.

Figure 4.17: RMSE of the longitudinal states of the UAV under additive abrupt ambiguous sensor faults, estimated by a JMRPF and a IMM. Results are based on 100 simulations.

In Figure 4.17a, the significant error of more than 15 m for the altitude estimate performed by the IMM-KF is confirmed by the RMSE. On the other hand a mean error of 2 m is obtained for the JMRPF. For all others states shown in Figure 4.17a, both IMM-KF and JMRPF are comparable in terms of accuracy for the state estimation.

Table 4.1 gives the RMSE values at key time steps and the $\overline{\text{RMSE}}$ which is defined in Appendix D.

State	Time (s)				$\overline{\text{RMSE}}$
	10.05 s	20.05 s	30.05 s	40.05 s	
JMRPF					
RMSE p_d (m)	1.087	1.434	1.667	0.875	0.998
RMSE u ($\mathrm{m\,s^{-1}}$)	0.307	0.337	0.146	0.159	0.187
RMSE w ($\mathrm{m\,s^{-1}}$)	0.094	0.137	0.035	0.035	0.035
RMSE θ (°)	1.028	1.117	0.414	0.493	0.438
RMSE q ($^\circ\,\mathrm{s^{-1}}$)	0.232	0.306	0.093	0.111	0.103
RMSE f_g (m)	6.730	7.540	0.000	0.165	1.261
RMSE f_b (m)	0.316	5.970	1.713	0.227	0.723
IMM					
RMSE p_d (m)	1.221	9.780	20.710	5.661	6.526
RMSE u ($\mathrm{m\,s^{-1}}$)	0.119	0.129	0.125	0.133	0.131
RMSE w ($\mathrm{m\,s^{-1}}$)	0.029	0.028	0.028	0.031	0.029
RMSE θ (°)	0.375	0.461	0.439	0.439	0.429
RMSE q ($^\circ\,\mathrm{s^{-1}}$)	0.102	0.086	0.098	0.092	0.097
RMSE f_g (m)	26.359	6.679	7.099	9.263	6.886
RMSE f_b (m)	1.121	21.814	20.965	6.746	6.379

Table 4.1: RMSE values of the JMRPF and the IMM estimates at key time steps, and $\overline{\text{RMSE}}$.

4.6 CHAPTER SUMMARY

This chapter introduced a new approach for estimating actuator and sensor faults as well as ambiguous sensor faults, called the JMRPF. This algorithm has been tested and validated with numerical results and compared to a IMM-KF. The idea behind the JMRPF was to move particles according to a transition probability matrix to a potential value of a fault. The associated algorithm of this method was fully described in Section 4.4. 100 simulations were performed in Section 4.5 and their analysis has shown sound estimation performance with this filter.

High accuracy and low convergence time are usually competing objectives, but using a JMS, the proposed JMRPF has shown the ability to estimate faults in a very short time and with a fair accuracy (research question 1), with good robustness to the amplitude of the

fault. Its accuracy was comparable to the IMM-KF in a fault free situation, but does not suffer from its limitations, such as its limited ability to estimate ambiguous sensor faults.

However, this filter still suffers from some limitations because some knowledge of the fault type and dynamics had to be assumed in order to estimate it. To deal with this limitation, some methods such as the IMM-KF can use multiple candidate models of fault dynamics. The problem with the JMRPF as presented in this chapter is that it can only deal with two models, one fault-free model and one faulty model. Even though the ability of the JMRPF to estimate ambiguous sensor faults was demonstrated (research question 2), the estimation of ambiguous actuator and sensor faults (which is covered in the following chapter) is actually not possible with the JMRPF as formulated in this chapter.

The following chapter will therefore aim to solve those two main limitations of the JMRPF.

5

ROBUSTIFIED JUMP-MARKOV REGULARIZED PARTICLE FILTER

In Chapter 4, the JMRPF was introduced and evaluated for ambiguous sensor faults. However, the method in its current form presents some limitations and needs to be extended to enable estimating faults with unknown dynamics and estimating ambiguous actuator and sensor faults. This chapter aims to overcome this limitation by introducing additional features on the JMRPF introduced in Chapter 4.

This chapter is organized as follows: Section 5.1 highlights some of the issues that affect fault estimation in UAV. In Section 5.2, some limitations faced by the JMRPF from Chapter 4 are presented. Section 5.3 introduces the idea behind the mechanism of an enhanced JMRPF to overcome the issue of Section 5.1. In Section 5.4, a formulation of the RJMRPF is introduced with the associated algorithm. Section 5.5 presents a detailed numerical simulation analysis of estimation performance, with a comparison between the new RJMRPF algorithm, the JMRPF and a robustified RPF, with application to a fixed-wing UAV under sensor and actuator faults. Section 5.6 summarizes the lessons learnt from this chapter.

5.1 FAULT ESTIMATION ON UNMANNED AERIAL VEHICLE

The method introduced in this thesis must handle a system subject to faults, for which the amplitudes and dynamics are seldom known in advance. In the application under consideration, the estimation method used must handle unknown fault dynamics and amplitude.

Moreover, the method must allow to estimate both actuator and sensor faults. A fault from an actuator or sensor with an impact on the same measurement is hereby referred to as an ambiguous fault. It is common in feedback control systems when a sensor is used to measure a state variable and an actuator is used to control the same variable to a setpoint. If an actuator is faulty, the associated state variable and measurement will be affected. Likewise, a sensor fault will have a direct impact on the same output measurement. Therefore, if measurements are detected as being faulty, it is not trivial to determine if the fault originated from the sensor or the actuator.

This ambiguous fault case may lead to a multimodality in the likelihood and posterior density. Indeed, consider the extended state vector $\mathbf{x}$ given by (2.18), with one actuator

and one sensor. When a faulty measurement occurs, the likelihood $p\left(\mathrm{y}_k|\mathbf{x}_k\right)$ has two peaks corresponding to two possible modes or solutions that are:

$$\mathbf{x}_k^a = \begin{bmatrix} \mathbf{z}_k{}^\top & \mathrm{f}_{\mathrm{a}k}{}^\top & 0 \end{bmatrix}^\top \tag{5.1}$$

and

$$\mathbf{x}_k^b = \begin{bmatrix} \mathbf{z}_k{}^\top & 0 & \mathrm{f}_{\mathrm{s}k}{}^\top \end{bmatrix}^\top . \tag{5.2}$$

In other words, several states of $\mathbf{x}_k$ may be associated with the same measurement. This results in the multimodality of posterior density $p\left(\mathbf{x}_k|\mathbf{Y}_{1:k}\right)$ as illustrated in Figure 5.1.

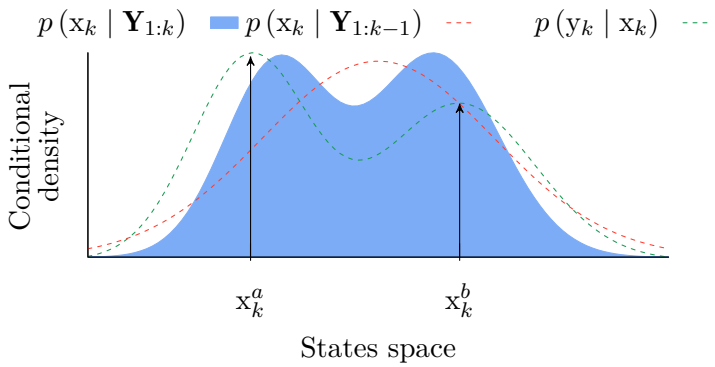


Figure 5.1: A representation of the posterior density $p\left(\mathbf{x}_k|\mathbf{Y}_{1:k}\right)$ in the case of sensor and actuator faults

In this case, an estimation method that can handle multimodality is needed.

5.2 LIMITATIONS OF EXISTING METHODS

Since the JMRPF introduced in Chapter 4 already overcomes most of the issues that must be considered in the UAV's application, it is used as the basis of the approach and additional features are developed to handle the issues that have been underlined in the previous section. It has been shown in Chapter 4 that the JMRPF allows the estimation of abrupt faults with an unknown amplitude, since the process noise can be adapted according to the amplitude of the fault considered. However, one of the limitations of the JMRPF it is that it can only have one process model, and the use of sentinel particles does not help, since a particle must be in discrete state $m^{(0)}$ — in other words set to 0 — to be able to move to Δ_{f}. An estimation of an incipient fault with a zero order fault model with a JMRPF is illustrated in Figure 5.2.

Figure 5.2, illustrates that the JMRPF must revert to a nominal mode $m^{(0)}$ to estimate the fault by moving the sentinel particles next to the fault. This is why the estimated fault follows a sawtooth pattern.

A IMM on the other hand is designed to handle multiple process models. This allows one to consider faults with different dynamics. However, having multiple process models means

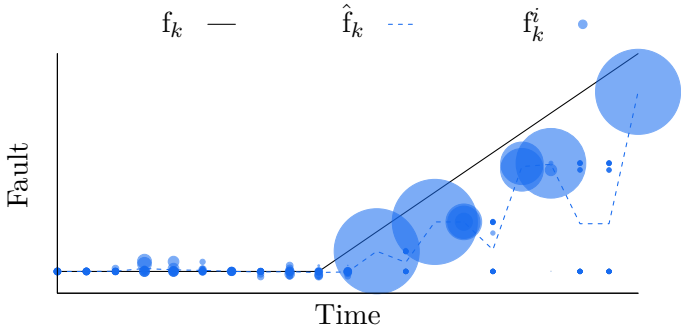


Figure 5.2: Incipient fault estimated by the JMRPF with a process model given by $[\ \dot{z} \quad \dot{f}\ ]^\top = \mathbf{0}_{2,1}$ and $y = z + f$. The process noise of the fault state is small regarding the fault dynamics, the number of particle is set to 20 and the jump probabilities π_{10} and π_{01} are both set to 10 %.

that one can only consider faults that have dynamics that correspond to — or close enough to — the one listed in the process model of the bank of filters used. Then, multiple fault dynamics must be considered to cover all possible fault dynamics, which leads to numerous and complex combinations of models to cover the whole spectrum of faults. This method does not solve the estimation of unknown dynamics.

In Figure 5.2, the process noise is small regarding the fault dynamics and then the particles stay at a plateau after being placed next to the fault. Another solution is to increase the process noise of the fault to spread the particles and allow them to match dynamics that differ more significantly from the ones of the process model. However, this solution leads to a degradation of the accuracy of the fault estimate and assumes some prior knowledge of the possible fault dynamics since the setting of the process noise will be done according to the deviation from the dynamics of the fault of the process model.

5.3 PROPOSED SOLUTION

A way to overcome the limitations listed above is to update, in addition to the particle weight, the particle placement. This solution aims to place the predicted particles to a potential better location. Indeed, the choice of the state transition density to move the particles is not optimal with the JMRPF. The particles are indeed moved based on the dynamic stochastic model, which may lead to their misplacement, outside regions of interest. A Kalman correction is therefore added to the weights update in order to bring the particles to the likeliest state space regions. The RJMRPF filter can be seen as a JMRPF with a proposal density calculated by a Kalman filter. As a consequence, the filter uses resampling less frequently for the same threshold and the Monte Carlo approximation error of the filter decrease.

5.4 FORMULATION OF THE ROBUSTIFIED JUMP-MARKOV REGULARIZED PARTICLE FILTER

The stochastic process model of the RJMRPF [73, 74] for additive actuator and sensor fault is given by 4.5.

5.4.1 *Prediction step*

The prediction step is the same as the one described by the JMRPF from Chapter 4.

5.4.2 *Update step*

The update step of the RJMRPF is different from the JMRPF from Chapter 4, since in addition to the weight update the predicted particles are also updated. Then, the weight update is still performed and given by (4.19), but since the update of the predicted particles states is performed using a Kalman gain and the innovation, it is then given by:

$$\mathbf{x}_k^i = \mathbf{x}_{k|k-1}^i + \mathbf{K}_k \tilde{\mathbf{y}}_k^i, \tag{5.3}$$

where the Kalman gain is computed at each time step with the formula given by (2.32) in the case of a linear measurement function. However, in the case of a non-linear measurements' equation, the Kalman gain is given by:

$$\mathbf{K}_k = \mathbf{P}_{\mathbf{xy}_k} \mathbf{S}_k^{-1}, \tag{5.4}$$

where $\mathbf{P}_{\mathbf{xy}}$ is the cross covariance matrix, and $\mathbf{S}$ is the covariance matrix of the innovation. The cross covariance matrix is then given by:

$$\mathbf{P}_{\mathbf{xy}} = \sum_{i=1}^{N} w_{k-1}^i \left(\mathbf{x}_{k|k-1}^i - \hat{\mathbf{x}}_{k|k-1} \right) \left(\mathbf{y}_{k|k-1}^i - \hat{\mathbf{y}}_{k|k-1} \right)^\top, \tag{5.5}$$

where $\hat{\mathbf{x}}_{k|k-1}$ is the estimated predicted state vector, $\hat{\mathbf{y}}_{k|k-1}$ the estimated predicted measurement vector and $\mathbf{y}_{k|k-1}^i$ the predicted measurement vector, given by:

$$\mathbf{y}_{k|k-1}^i = h_k \left(\mathbf{x}_{k|k-1}^i \right). \tag{5.6}$$

The estimated predicted state vector is given by:

$$\hat{\mathbf{x}}_{k|k-1} = \sum_{i=1}^{N} w_{k-1}^i \mathbf{x}_{k|k-1}^i, \tag{5.7}$$

and the estimated predicted measurement vector is given by:

$$\hat{\mathbf{y}}_{k|k-1} = \sum_{i=1}^{N} w_{k-1}^i \mathbf{y}_{k|k-1}^i, \tag{5.8}$$

Finally, the covariance matrix of the innovation is given by:

$$\mathbf{S}_k = \sum_{i=1}^{N} w_{k-1}^i \left(\mathbf{y}_{k|k-1}^i - \hat{\mathbf{y}}_{k|k-1}\right)\left(\mathbf{y}_{k|k-1}^i - \hat{\mathbf{y}}_{k|k-1}\right)^\top + \mathbf{R}_k. \tag{5.9}$$

The computation of the Kalman gain is given by the function KALMANGAIN detailed in Algorithm 5.1

Algorithm 5.1 Computation of the Kalman update of the robustified jump-Markov regularized particle filter

Function KALMANGAIN($\mathbf{K}_k$, $\mathbf{x}_{k|k-1}^{1:N}$, $w_k^{1:N}$)

- $\mathbf{y}_{k|k-1}^i \leftarrow h_k\left(\mathbf{x}_{k|k-1}^i\right)$ //See (5.6)
- $\hat{\mathbf{y}}_{k|k-1} \leftarrow \sum_{i=1}^{N} w_{k-1}^i \mathbf{y}_{k|k-1}^i$ //See (5.8)
- $\hat{\mathbf{x}}_{k|k-1} \leftarrow \sum_{i=1}^{N} w_{k-1}^i \mathbf{x}_{k|k-1}^i$ //See (5.7)
- $\mathbf{P}_{\mathbf{xy}} \leftarrow \sum_{i=1}^{N} w_{k-1}^i \left(\mathbf{x}_{k|k-1}^i - \hat{\mathbf{x}}_{k|k-1}\right)\left(\mathbf{y}_{k|k-1}^i - \hat{\mathbf{y}}_{k|k-1}\right)^\top$ //See (5.5)
- $\mathbf{S}_k = \sum_{i=1}^{N} w_{k-1}^i \left(\mathbf{y}_{k|k-1}^i - \hat{\mathbf{y}}_{k|k-1}\right)\left(\mathbf{y}_{k|k-1}^i - \hat{\mathbf{y}}_{k|k-1}\right)^\top + \mathbf{R}_k$ //See (5.9)
- $\mathbf{K}_k = \mathbf{P}_{\mathbf{xy}_k}\mathbf{S}_k^{-1}$ //See (5.4)

The algorithm of the update step of the JMRPF is performed by the function UPDATE detailed in Algorithm 2.3.

Algorithm 5.2 Update step of the robustified jump-Markov regularized particle filter

Function UPDATE($w_k^{1:N}$, $\mathbf{x}_k^{1:N}$, $w_{k-1}^{1:N}$, $\mathbf{x}_{k|k-1}^{1:N}$, $\mathbf{y}_k$)

- KALMANGAIN($\mathbf{K}_k$, $\mathbf{x}_{k|k-1}^{1:N}$, $w_k^{1:N}$) //See Algorithm 5.1
- **for each** $i \in [1, N]$ **do**
  - $\tilde{w}_k^i \leftarrow w_{k-1}^i \mathcal{N}\left(\tilde{\mathbf{y}}_k^i; 0, \mathbf{S}_k^i\right)$ //See (4.19)
  - $\mathbf{x}_k^i \leftarrow \mathbf{x}_{k|k-1}^i + \mathbf{K}_k \tilde{\mathbf{y}}_k^i$ //See (5.3)
- **for each** $i \in [1, N]$ **do**
  - $w_k^i \leftarrow \frac{\tilde{w}_k^i}{\sum_{j=1}^{N} \tilde{w}_k^j}$ //Normalization of the weights

5.4.3 *Estimation step*

The estimation step aims to perform a global estimate of the state vector $\hat{\mathbf{x}}_k$, with its associated covariance matrix $\hat{\mathbf{P}}_k$. This step is different from the SIR particle filter, since now

the global estimate is computed using the updated state vector $\mathbf{x}_k^i$. Then from (2.27b) the global estimate is obtained using the updated state vectors is given by:

$$\hat{\mathbf{x}}_k = \sum_{i=1}^{N} w_k^i \mathbf{x}_k^i, \tag{5.10}$$

and its associated estimated covariance matrix is then given by:

$$\hat{\mathbf{P}}_k = \sum_{i=1}^{N} w_k^i \left(\mathbf{x}_k^i - \hat{\mathbf{x}}_k\right)\left(\mathbf{x}_k^i - \hat{\mathbf{x}}_k\right)^\top. \tag{5.11}$$

The algorithm that produces the estimate of the RJMRPF is detailed in the function estimate of Algorithm 5.3.

Algorithm 5.3 Estimate step of the robustified jump-Markov regularized particle filter

Function ESTIMATE($\hat{\mathbf{x}}_k$, $\hat{\mathbf{P}}_k$, $\mathbf{x}_k^{1:N}$, $w_k^{1:N}$)

$\hat{\mathbf{x}}_k \leftarrow \sum_{i=1}^{N} w_k^i \mathbf{x}_k^i$ //See (5.10)

$\hat{\mathbf{P}}_k \leftarrow \sum_{i=1}^{N} w_k^i \left(\mathbf{x}_k^i - \hat{\mathbf{x}}_k\right)\left(\mathbf{x}_k^i - \hat{\mathbf{x}}_k\right)^\top$ //See (5.11)

5.4.4 *Regularization-Resampling Step*

The regularization-resampling step is the same as the one described by the JMRPF.

The RJMRPF is presented in Algorithm 5.4, using the previously defined functions UPDATE and ESTIMATE, and the function used by the JMRPF from Chapter 4.

Algorithm 5.4 Robustified jump-Markov regularized particle filter

$k \leftarrow 0$
⋮ //Initialization
Loop
  $k \leftarrow k+1$
  PREDICT($\mathbf{x}_{k|k-1}^{1:N}$, $\mathbf{x}_{k-1}^{1:N}$, $\mathbf{m}_k^{1:N}$, $\mathbf{u}_k$, $\mathbf{y}_k$) //See Algorithm 4.2
  UPDATE($w_k^{1:N}$, $\mathbf{x}_k^{1:N}$, $w_{k-1}^{1:N}$, $\mathbf{x}_{k|k-1}^{1:N}$, $\mathbf{y}_k$) //See Algorithm 5.2
  ESTIMATE($\hat{\mathbf{x}}_k$, $\hat{\mathbf{P}}_k$, $\mathbf{x}_k^{1:N}$, $w_k^{1:N}$) //See Algorithm 5.3
  $\hat{N}_{eff} \leftarrow \frac{1}{\sum_{i=1}^{N} (w_k^i)^2}$ //See (2.53)
  if $\hat{N}_{eff} \leq N\Gamma_{rspl}$ **then** //if true then resample
    MULTINOMIAL($\acute{\mathbf{x}}_k^{1:N}$, $\mathbf{x}_{k|k-1}^{1:N}$, $w_k^{1:N}$) //See Algorithm 2.5
    for each $i \in [1, N]$ **do**
      $w_k^i \leftarrow \frac{1}{N}$ //Reset the weights, See (2.55)
    REGULARIZE($\mathbf{x}_k^{1:N}$, $\acute{\mathbf{x}}_k^{1:N}$, $\hat{\mathbf{P}}_k$) //See Algorithm 2.7

5.5 COMPARATIVE NUMERICAL SIMULATION ANALYSIS

This section aims to demonstrate the state and fault estimation efficiency of the RJMRPF compared to previously presented particle filtering methods, in the presence of sensor faults with unknown dynamics and a combination of actuator and sensor faults on a fixed-wing UAV. For sake of brevity, only the longitudinal system is considered.

5.5.1 *Fault with unknown dynamics*

This section aims to demonstrate the ability of the RJMRPF to perform fault estimation in the presence of faults with unknown dynamics and unknown amplitudes. Since the unknown dynamics are one of the limitations of the JMRPF and the unknown amplitude of abrupt faults is one of the limitations of the RPF, a comparative simulation of both methods is performed to evaluate how the RJMRPF overcomes these limitations compared to the RPF and the JMRPF. Given that all algorithms under comparison are particle filters, a non-linear system is used. The UAV has an initial longitudinal velocity of $40\,\mathrm{m\,s^{-1}}$, an initial altitude of 500 m, an initial flight path angle 0 rad and is initiated in a straight cruise flight condition. The control and guidance are then performed as described in Section 3.7 and Section 3.8 for the longitudinal system around the trim point of the initial flight condition. The desired altitude is set to 500 m and the desired velocity to $40\,\mathrm{m\,s^{-1}}$.

To illustrate an abrupt sensor fault on the pitch measurement with an unknown amplitude and unknown dynamics, an arbitrary fault signal is added to the measurement equation.

This true fault model is unknown to the filter. The nonlinear longitudinal model used to compute the true state is given by:

$$\begin{cases} \dot{\mathbf{z}} = \mathcal{F}(\mathbf{z}, \mathbf{u}) & \text{(5.12a)} \\ \mathbf{y} = \mathcal{H}(\mathbf{z}) + \begin{bmatrix} 0 \\ 0 \\ 0 \\ f_\theta \\ 0 \end{bmatrix} + \boldsymbol{\nu} & \text{(5.12b)} \end{cases}$$

where f_θ denotes the pitch measurement fault and the measurement noise $\boldsymbol{\nu}$ is a zero mean Gaussian noise given by:

$$\boldsymbol{\nu} = \begin{bmatrix} \nu_{baro,-p_d} \\ \nu_{accel,u} \\ \nu_{accel,w} \\ \nu_{gyro,\theta} \\ \nu_{gyro,q} \end{bmatrix} \tag{5.13}$$

with standard deviations respectively given by $\sigma_{baro,-p_d} = 1\,\mathrm{m}$, $\sigma_{accel,u} = 1\,\mathrm{m\,s^{-1}}$, $\sigma_{accel,w} = 1\,\mathrm{m\,s^{-1}}$, $\sigma_{gyro,\theta} = 0.01\,\mathrm{rad}$ and $\sigma_{gyro,q} = 0.002\,\mathrm{rad\,s^{-1}}$. The state vector $\mathbf{z}$ is given by:

$$\mathbf{z} = \begin{bmatrix} p_d \\ u \\ w \\ \theta \\ q \end{bmatrix}. \tag{5.14}$$

The input vector $\mathbf{u}$ is given by:

$$\mathbf{u} = \begin{bmatrix} \delta_e \\ \delta_t \end{bmatrix}. \tag{5.15}$$

The non-linear function $\mathcal{F}(\cdot)$ is obtained from (3.8) and the observation function $\mathcal{H}(\cdot)$ is given by:

$$\mathcal{H}(\mathbf{z}) = \begin{bmatrix} -1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \mathbf{z} \tag{5.16}$$

Since the fault is only on the measurements, the evolution and control input matrices for the process models are the same as the ones given in (5.12) for the state vector $\mathbf{z}$. An extended vector $\mathbf{x}$ is defined as:

$$\mathbf{x} = \begin{bmatrix} \mathbf{z} \\ f_\theta \end{bmatrix}. \tag{5.17}$$

The sensor fault is assumed to follow a zero order model. The process noise is a white Gaussian noise with a standard deviation $\sigma_{\mathbf{x}}$ given by:

$$\sigma_{\mathbf{x}} = \begin{bmatrix} 1\,\mathrm{m} \\ 0.1\,\mathrm{m\,s^{-1}} \\ 0.1\,\mathrm{m\,s^{-1}} \\ 0.02\,\mathrm{rad} \\ 0.002\,\mathrm{rad\,s^{-1}} \\ 0.002\,\mathrm{rad} \end{bmatrix} \tag{5.18}$$

The measurement noise of the process model is 1.5 times the measurements noise of (5.12);

The transition probability matrix of the RJMRPF and the JMRPF is given by:

$$\mathbf{\Pi} = \begin{bmatrix} 0.99 & 0.01 \\ 0.01 & 0.99 \end{bmatrix} \tag{5.19}$$

The standard deviation used to compute the initial covariance matrix $\mathbf{P}_0$ for all filters, denoted $\sigma_{\mathbf{x0}}$ is given by:

$$\sigma_{\mathbf{x0}} = \begin{bmatrix} 1\,\mathrm{m} \\ 1\,\mathrm{m\,s^{-1}} \\ 1\,\mathrm{m\,s^{-1}} \\ 0.005\,\mathrm{rad} \\ 0.002\,\mathrm{rad\,s^{-1}} \\ 0.005\,\mathrm{rad} \end{bmatrix} \tag{5.20}$$

The fault scenario considered lasts 50 s. For the first 10 s, no fault is active in the system. At 10 s, a first fault occurs on the pitch measurement. It is an abrupt fault with an amplitude of 5°. The amplitude of the abrupt fault is too large to be estimated accurately and rapidly by the filters introduced in Chapter 2, given the process noise. This fault is deactivated at 20 s. At 30 s a second fault occurs on the pitch measurement. This time, it is an incipient fault with exponential dynamics that last 10 s and reach an amplitude of 10°. The dynamics of the incipient fault from 30 s to 40 s are given by:

$$f_\theta(t) = 10e^{t-40} \tag{5.21}$$

This incipient fault is clearly described by dynamics that are different from the assumed zero order process model of the fault. Hence, the fault dynamics are unknown. The scenario is illustrated in Figure 5.3.

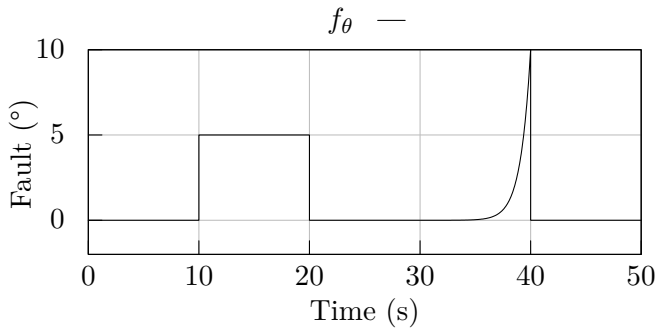


Figure 5.3: Fault scenario for the simulation of unknown dynamic of sensor fault with a fault on the pitch measurements.

Finally, the number of particles is set to 1000 and the resampling threshold Γ_{rspl} is set to 0.5 and the bandwidth factor h of the Epanechnikov kernel is set 0.2817. The number of simulations performed is 100 with a time step of 0.05 s.

The first results illustrated in Figure 5.4 are the estimates of the RJMRPF, the JMRPF and the RPF by taking the median results from 100 simulations. The selection of the median result is detailed in Appendix D.

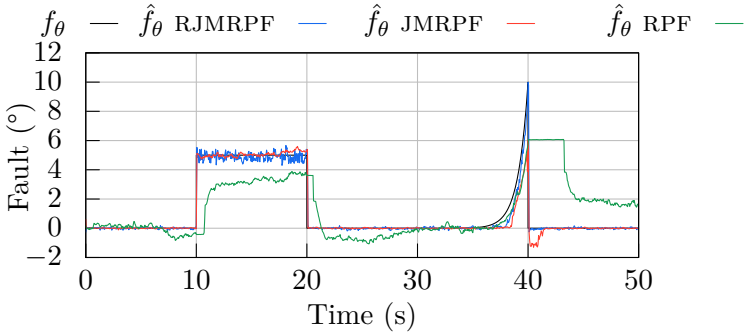


Figure 5.4: Median result of the fault states of the UAV under unknown dynamic additive sensor fault estimated by a RJMRPF, a JMRPF and a RPF. Median results based on 100 simulations.

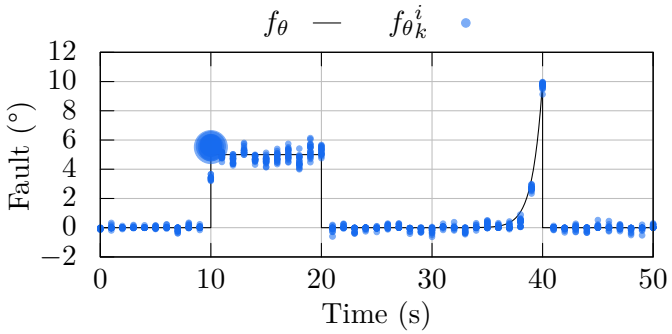
In Figure 5.4, during the first 10 s while there is no fault, the RPF is less effective to reduce noise than the JMRPF and the RJMRPF. This is due to the fact that unlike the JMRPF and the RJMRPF, it does not force the particle to zero when it is in a fault-free mode. When the first abrupt fault occurs at 10 s, the RPF takes longer time to converge to the fault than the other filters. This is mainly due to the fact that its process noise and regularization noise are not adapted to such a fault amplitude, while the JMRPF and the RJMRPF do not

need to adapt their process noise to the fault amplitude to converge to it, as long as they have a particle placement close to the true fault state. This shows that they can estimate abrupt faults with unknown amplitudes. The estimate of the abrupt fault obtained with the JMRPF is less noisy than the one obtained with the RJMRPF. However, both provide a fairly accurate estimate of the fault. The deactivation of the fault at 20 s is slower with the RPF due to its process noise. The JMRPF and the RJMRPF on the other hand return to zero in a one time step, with a better precision than the RPF.

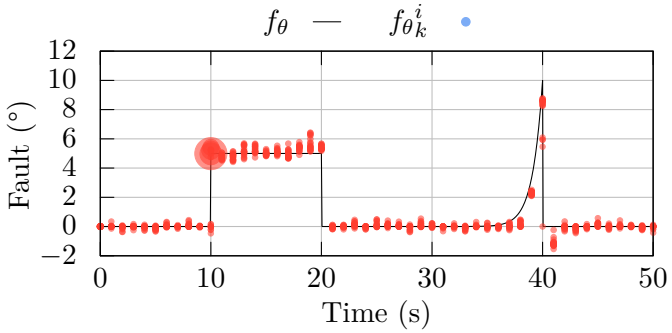
Until 37 s, the incipient exponential fault is not very significant. Then after that time, differences start to appear. Indeed, the RPF starts to converge to the fault while other filters reject the change for a brief moment. This is due to the fact that the particles of the JMRPF and the RJMRPF are forced to be at zero until sentinel particle start to be more likely than particles in $m^{(0)}$, which happens when the fault begin to be more significant. A short time after that, the JMRPF and the RJMRPF start converging to the fault, both reaching the same level of the RPF very quickly. However, after approximately 38.5 s the fault becomes too steep — or in other words, too different from the fault dynamics of the process model —, then the JMRPF and the RPF present an increasing error between the fault estimate and the true fault, and they evolve with similar dynamics. The RJMRPF estimates the true fault with a better accuracy, and is even able to estimate the fault when it reaches 10° at 40 s while the other filters have converged to 6°. This shows that the Kalman correction of the particle state, improves the ability of the JMRPF to estimate faults with different dynamics. At the deactivation of the incipient fault at 40 s, the RPF remains at a constant value for 3.25 s, before slowly converging to zero. This can be due to the fact that the RPF is diverging, since the process noise is too small compared to the abrupt change due to the fault. The JMRPF and the RJMRPF do not present similar behaviours. However, due to the poor estimation of the fault by the JMRPF at 40 s, the fault estimate does not return to zero, but around −1°, for a short time and then retuning to a fault free mode. The RJMRPF is more accurate than the JMRPF and the return to zero and the fault free mode takes one time step.

The behaviour of the median results of the RJMRPF, the JMRPF and the RPF is illustrated in detail in Figure 5.5, by showing the particles positions with their associated weights.

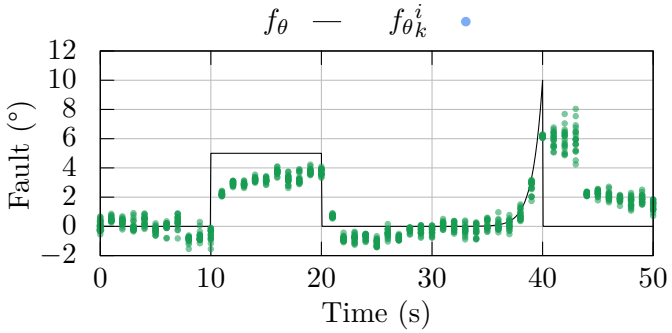
In Figure 5.5, no significant differences are visible between the RJMRPF and the JMRPF for the abrupt fault. Both estimate the abrupt fault similarly and with a strategy already detailed in Chapter 4. For the abrupt fault, no unexpected behaviour is observed on the RPF. For the incipient exponential fault, all the particles shown in Figure 5.5a seem correctly placed with the RJMRPF, except around 38 s, which is consistent with the result of Figure 5.4. The particles of the JMRPF are however more poorly positioned than the RJMRPF at 40 s. Indeed, some particles are around 6° and others around 8.5°. Finally, the particles of the RPF are better positioned at 38 s than the other filters. However, as observed in Figure 5.4, the end of the fault is poorly estimated, and at the fault deactivation time, the filter diverges for around 3 s.



(a) Estimation of the fault state with a RJMRPF.



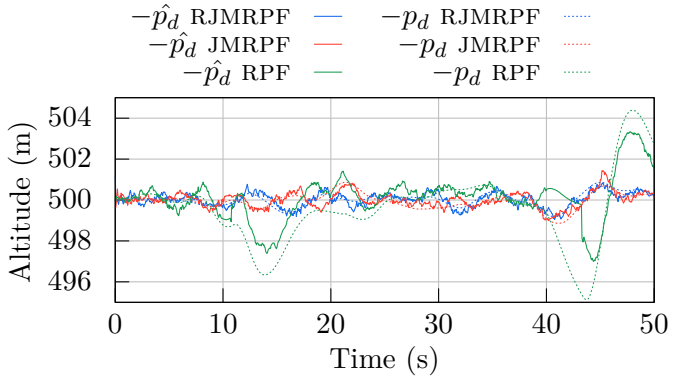
(b) Estimation of the fault state with a JMRPF.



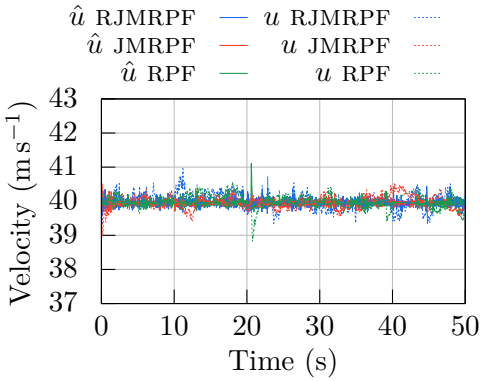
(c) Estimation of the fault state with a RPF.

Figure 5.5: The 20 most weighted particles of every second of the median result of the fault states of the UAV under unknown dynamic additive sensor fault estimated by a RJMRPF, a JMRPF and a RPF. Median results based on 100 simulations.

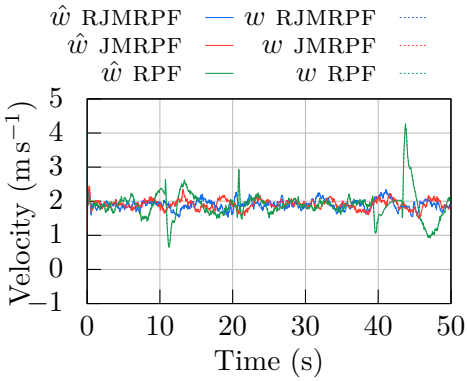
The median results of the state vector $\mathbf{z}$ estimated by the RJMRPF, the JMRPF and the RPF are shown in Figure 5.6.



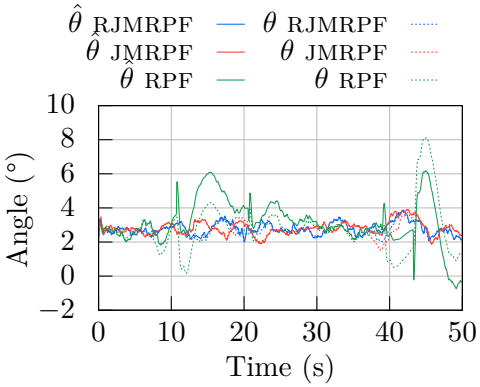
(a) Altitude.



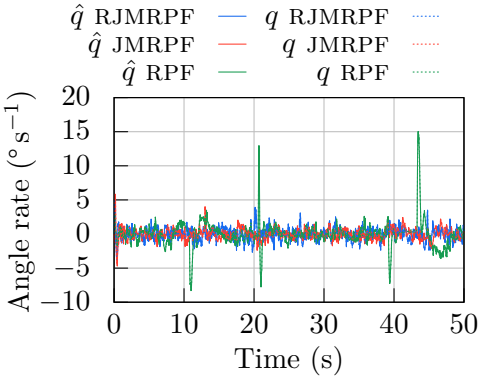
(b) Velocity along i^b.



(c) Velocity along k^b.



(d) Pitch.

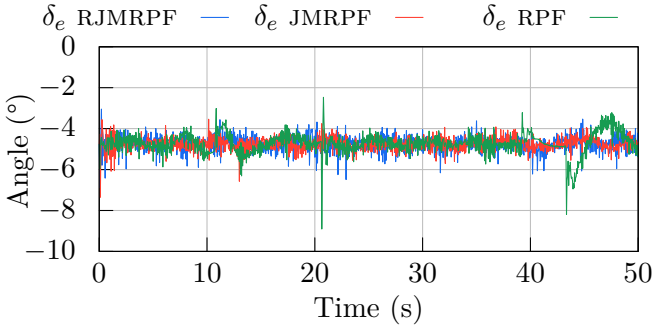


(e) Pitch rate.

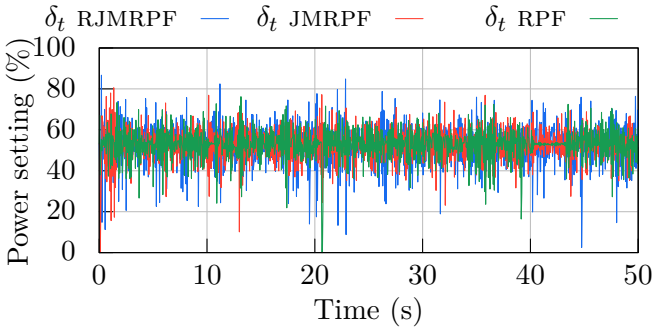
Figure 5.6: Median result of the longitudinal states of the UAV under unknown dynamic additive sensor fault estimated by a RJMRPF, a JMRPF and a RPF. Median results based on 100 simulations.

In Figure 5.6, no significant error in the RJMRPF error is visible. The UAV with the states estimated by the RJMRPF remains in a straight level flight as desired. This is consistent with the fact that the sensor fault is well estimated by this filter. For the JMRPF, only the estimate of the pitch state is slightly affected by the poor estimate of the fault at around 40 s, but no significant altitude deviation is observed. For the RPF, the poor estimate of the abrupt fault has significantly affected the estimate of the pitch and the altitude, with an altitude variation of almost 4 m. The poor estimate of the incipient fault and its late and slow convergence to zero at the deactivation of the fault induce a significant error in pitch and altitude estimation with a significant altitude variation of almost 5 m.

The control inputs of the RJMRPF, the JMRPF and the RPF are shown in Figure 5.7.



(a) Elevator deflection.



(b) Throttle.

Figure 5.7: Control inputs of the median result of the UAV under unknown dynamic additive sensor fault estimated by a RJMRPF, a JMRPF and a RPF. Median results based on 100 simulations.

In Figure 5.7, the control input is consistent with the trajectory observed for each filter. Moreover, the divergence of the RPF from 40 s for more than 3 s is again visible in both inputs of the system estimated by the RPF.

The RMSE of the fault state f_θ for the RJMRPF, the JMRPF and the RPF estimates are shown in Figure 5.8.

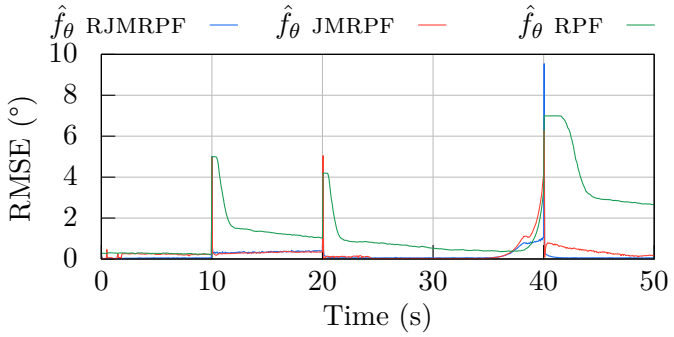
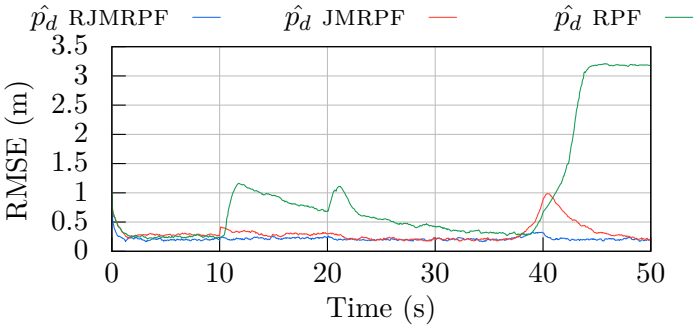


Figure 5.8: RMSE of the fault states of the UAV under unknown dynamic additive sensor fault estimated by a RJMRPF, a JMRPF and a RPF. Results are based on 100 simulations.

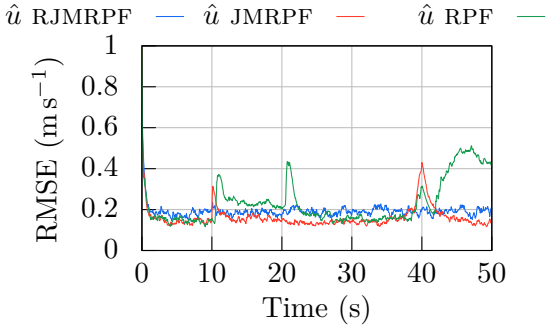
In Figure 5.8, the RJMRPF performs better in terms of RMSE than the RPF and JMRPF during the first 10 s. The JMRPF and RJMRPF perform similarly as expected at 30 s. These, RMSE results confirm that the RJMRPF has a better ability to estimate fault with unknown dynamics thanks to the addition of the Kalman correction. Neither the JMRPF nor the RPF are able to estimate the incipient exponential fault with a comparable accuracy to that of the RJMRPF. The estimation of abrupt fault with unknown amplitude is however possible thanks to the use of a JMS, and the JMRPF had a qualitatively similar result and the RPF is further behind the curve.

The RMSE of the state vector $\mathbf{z}$ is shown in Figure 5.9.

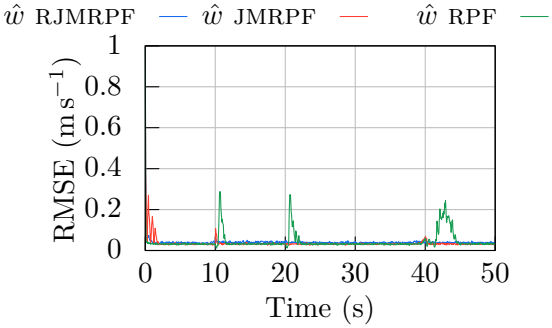
In Figure 5.9, the RMSE of the states show that, thanks to the good estimation of the RJMRPF, no significant deviation of the trajectory is observed over the 100 simulations performed. The JMRPF is however unable to accurately estimate the incipient exponential fault and then a significant error in the altitude RMSE is visible at 40 s. The RPF shows that the late return to zero at the deactivation of the abrupt and incipient faults induce a high RMSE on the pitch but also on the altitude and the pitch rate states.



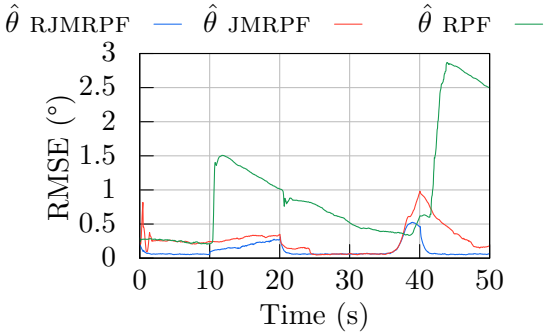
(a) RMSE of the altitude estimate.



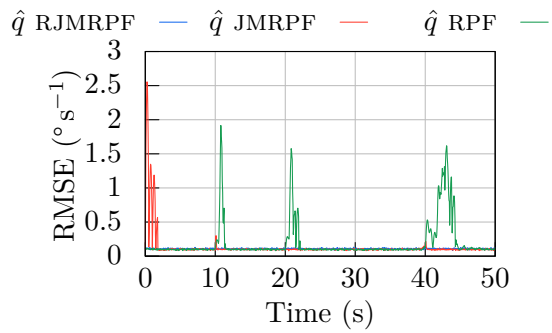
(b) RMSE of the velocity along i^b estimate.



(c) RMSE of the velocity along k^b estimate.



(d) RMSE of the pitch estimate.



(e) RMSE of the pitch rate estimate.

Figure 5.9: RMSE of the longitudinal states of the UAV under unknown dynamic additive sensor fault estimated by a RJMRPF, a JMRPF and a RPF. Results are based on 100 simulations.

Table 5.1 gives the RMSE values at key time steps and the $\overline{\text{RMSE}}$ which is defined in Appendix D.

State	Time (s)				$\overline{\text{RMSE}}$
	10.05 s	20.05 s	30.05 s	40.05 s	
RJMRPF					
RMSE p_d (m)	0.236	0.239	0.213	0.316	0.215
RMSE u ($\mathrm{m\,s^{-1}}$)	0.219	0.178	0.188	0.203	0.192
RMSE w ($\mathrm{m\,s^{-1}}$)	0.056	0.042	0.043	0.046	0.041
RMSE θ (°)	0.096	0.272	0.062	0.459	0.115
RMSE q ($^\circ\,\mathrm{s^{-1}}$)	0.104	0.112	0.108	0.116	0.108
RMSE f_θ (°)	0.549	4.734	0.051	9.542	0.177
JMRPF					
RMSE p_d (m)	0.363	0.303	0.203	0.905	0.308
RMSE u ($\mathrm{m\,s^{-1}}$)	0.267	0.150	0.141	0.428	0.163
RMSE w ($\mathrm{m\,s^{-1}}$)	0.107	0.030	0.028	0.068	0.036
RMSE θ (°)	0.236	0.349	0.066	0.982	0.260
RMSE q ($^\circ\,\mathrm{s^{-1}}$)	0.188	0.095	0.087	0.213	0.132
RMSE f_θ (°)	0.305	5.049	0.027	6.299	0.295
RPF					
RMSE p_d (m)	0.239	0.682	0.419	0.685	0.930
RMSE u ($\mathrm{m\,s^{-1}}$)	0.150	0.210	0.161	0.315	0.233
RMSE w ($\mathrm{m\,s^{-1}}$)	0.023	0.033	0.031	0.060	0.045
RMSE θ (°)	0.210	1.009	0.513	0.620	0.969
RMSE q ($^\circ\,\mathrm{s^{-1}}$)	0.153	0.103	0.102	0.164	0.191
RMSE f_θ (°)	5.001	4.193	0.522	6.992	1.526

Table 5.1: RMSE values of the RJMRPF, the JMRPF and the RPF estimates at key time steps, and $\overline{\text{RMSE}}$.

To further highlight the strengths of the RJMRPF, the effect of a tenfold increase in the fault amplitude is evaluated. The median fault estimate of this simulation is then illustrated in Figure 5.10.

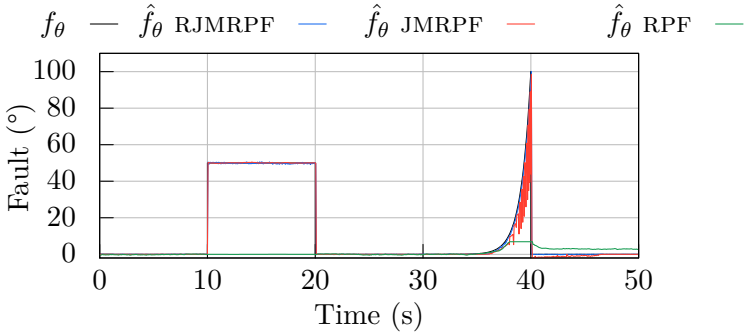


Figure 5.10: Median result of the fault states of the UAV under unknown dynamic additive sensor fault with a fault amplitude increased by a factor of 10 and estimated by a RJMRPF, a JMRPF and a RPF. Median results based on 100 simulations.

In Figure 5.10, the RJMRPF estimates the fault rapidly and accurately. The JMRPF estimates the abrupt fault as expected but provides a noisy estimate of the incipient fault. This is an expected behaviour. The noisy estimate is due to the fact that the JMRPF estimates the incipient fault by returning to zero and then replacing a sentinel particle near the true fault. A closer look at the estimate between 37 s to 40 s shows a sawtooth pattern as explained in Figure 5.2. This behaviour is significant in Figure 5.10 since the process noise is not large enough to helping the filter to converge. This is also confirmed by the fact that the RPF cannot estimate the abrupt fault, and does not converge when the difference is too large between the current estimate and the true state with respect to the process noise. This happens during the time of the abrupt fault, but also for last 2 s of the incipient fault when the JMRPF also stats to converge to the fault thanks to the JMS and not to the process noise.

The RMSE of the fault state f_θ of the RJMRPF, the JMRPF and the RPF are shown in Figure 5.11.

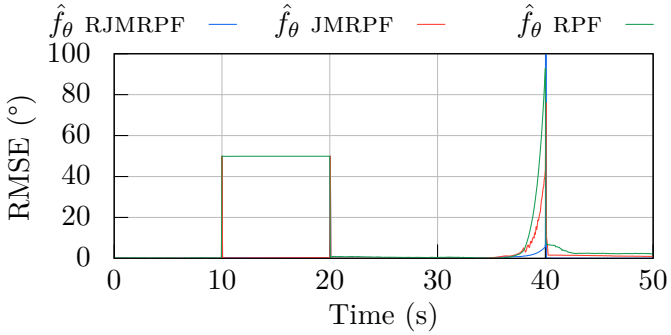
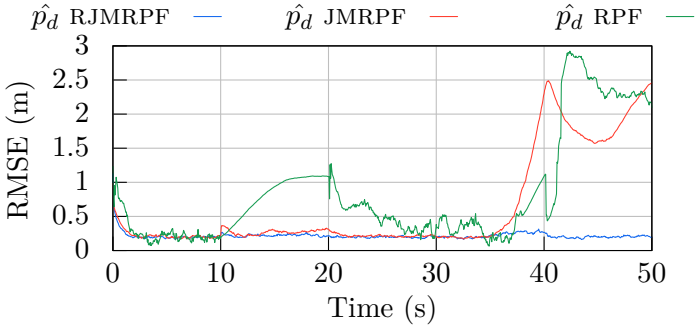


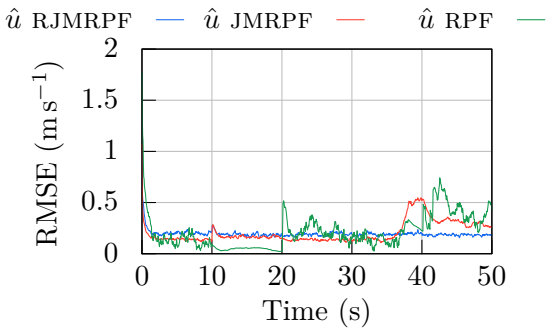
Figure 5.11: RMSE of the fault states of the UAV under unknown dynamic additive sensor fault with a fault amplitude increased by a factor of 10 and estimated by a RJMRPF, a JMRPF and a RPF. Results are based on 100 simulations.

Figure 5.11, is consistent with all previous results and highlights the improvement obtained by the Kalman correction of the particles with a better ability to estimate faults with unknown dynamics.

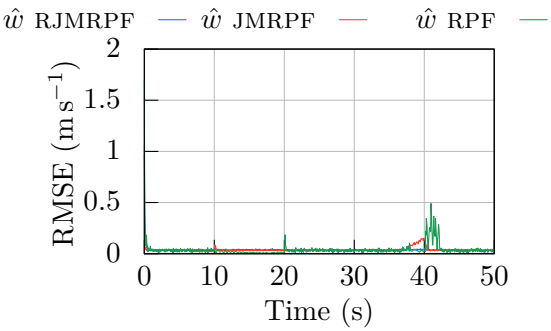
The RMSE of the states are shown in Figure 5.12.



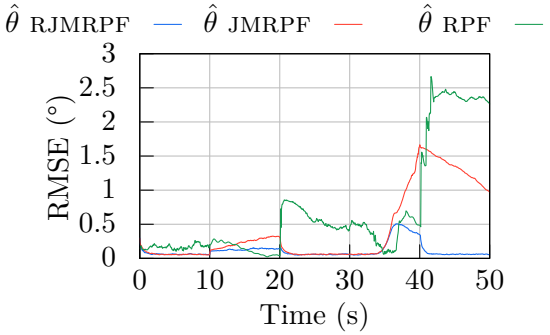
(a) RMSE of the altitude estimate.



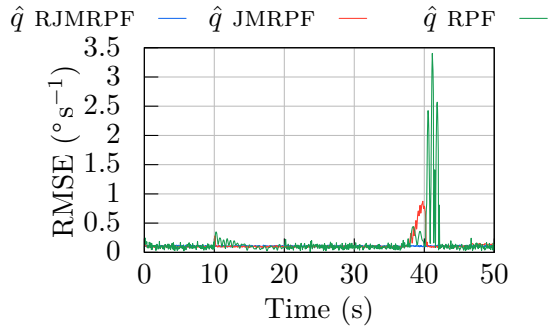
(b) RMSE of the velocity along i^b estimate.



(c) RMSE of the velocity along k^b estimate.



(d) RMSE of the pitch estimate.



(e) RMSE of the pitch rate estimate.

Figure 5.12: RMSE of the longitudinal states of the UAV under unknown dynamic additive sensor fault with a fault amplitude increased by a factor of 10 and estimated by a RJMRPF, a JMRPF and a RPF. Results are based on 100 simulations.

In Figure 5.12, the results show that the RJMRPF has significantly lower RMSE compared to the JMRPF and the RPF. The RPF has a comparatively high RMSE for all estimated states.

Table 5.2 gives the RMSE values at key time steps and the $\overline{\text{RMSE}}$ which is defined in Appendix D.

State	Time (s) 10.05 s	20.05 s	30.05 s	40.05 s	$\overline{\text{RMSE}}$
RJMRPF					
RMSE p_d (m)	0.262	0.213	0.196	0.263	0.214
RMSE u ($\mathrm{m\,s^{-1}}$)	0.279	0.173	0.189	0.179	0.194
RMSE w ($\mathrm{m\,s^{-1}}$)	0.071	0.041	0.033	0.038	0.040
RMSE θ (°)	0.090	0.140	0.057	0.342	0.114
RMSE q ($^\circ\mathrm{s^{-1}}$)	0.103	0.104	0.108	0.108	0.107
RMSE f_θ (°)	0.303	49.936	0.067	99.684	0.451
JMRPF					
RMSE p_d (m)	0.343	0.306	0.197	2.352	0.642
RMSE u ($\mathrm{m\,s^{-1}}$)	0.269	0.155	0.125	0.526	0.206
RMSE w ($\mathrm{m\,s^{-1}}$)	0.087	0.034	0.030	0.136	0.037
RMSE θ (°)	0.102	0.313	0.061	1.660	0.436
RMSE q ($^\circ\mathrm{s^{-1}}$)	0.191	0.101	0.094	0.776	0.122
RMSE f_θ (°)	0.335	50.017	0.023	76.011	1.250
RPF					
RMSE p_d (m)	0.200	1.084	0.329	1.103	0.847
RMSE u ($\mathrm{m\,s^{-1}}$)	0.089	0.026	0.153	0.224	0.213
RMSE w ($\mathrm{m\,s^{-1}}$)	0.037	0.012	0.026	0.022	0.038
RMSE θ (°)	0.196	0.042	0.470	0.485	0.710
RMSE q ($^\circ\mathrm{s^{-1}}$)	0.152	0.079	0.053	0.095	0.167
RMSE f_θ (°)	49.871	0.125	0.432	7.039	12.249

Table 5.2: RMSE values of the RJMRPF, the JMRPF and the RPF estimates at key time steps, and $\overline{\text{RMSE}}$.

5.5.2 *Ambiguous actuator and sensor faults*

This section aims to demonstrate the ability of the RJMRPF to perform fault estimation of ambiguous actuator and sensor faults. To evaluate the capacity of the RJMRPF to estimate these fault, this filter is compared to a robustified regularized particle filter (RRPF). The RRPF is a RPF with a Kalman correction — in other word it is the Algorithm 2.8, with the update step given by Algorithm 5.2. The two other filters, RPF and JMRPF have diverged on most of the trials corresponding to this scenario, which is why their results are not presented here. Moreover, unlike the JMRPF previously introduced, to be able to estimate such fault, the particles in mode zero are not forced to zero, but only set to zero when there is a transition from $m^{(1)}$ to $m^{(0)}$.

Since the actuator fault estimation using JMRPF needs a linearized system (see (4.14)), the true system is a non-linear system, however, the process model is the linearized system. The UAV has an initial longitudinal velocity of $40\,\mathrm{m\,s^{-1}}$, an initial altitude of 500 m, an initial flight path angle 0 rad and is initiated in a straight cruise flight condition. The control and guidance are then performed as described in Section 3.7 and Section 3.8 for the longitudinal system around the trim point of the initial flight condition. The desired altitude is set to 500 m and the desired velocity to $40\,\mathrm{m\,s^{-1}}$.

To illustrate an abrupt actuator and sensor fault on the elevator deflection and the pitch rate measurement, fault signals are added to the state and measurement equation. This true fault model is unknown to the filter. The nonlinear longitudinal model used to compute the true state is given by:

$$\begin{cases} \dot{\mathbf{z}} = \mathcal{F}\left(\mathbf{z}, \mathbf{u} + \mathbf{f_a}\right) & \text{(5.22a)} \\ \mathbf{y} = \mathcal{H}\left(\mathbf{z}\right) + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ f_q \end{bmatrix} + \boldsymbol{\nu} & \text{(5.22b)} \end{cases}$$

where f_q denotes the pitch rate measurement fault, $\mathbf{f_a}$ is the fault actuator state given by $\begin{bmatrix} f_{\delta_e} & 0 \end{bmatrix}^\top$ with f_{δ_e} the actuator fault, and the measurement noise $\boldsymbol{\nu}$ is a zero mean Gaussian noise given by:

$$\boldsymbol{\nu} = \begin{bmatrix} \nu_{baro,-p_d} \\ \nu_{accel,u} \\ \nu_{accel,w} \\ \nu_{gyro,\theta} \\ \nu_{gyro,q} \end{bmatrix} \tag{5.23}$$

with standard deviations respectively given by $\sigma_{baro,-p_d} = 1\,\mathrm{m}$, $\sigma_{accel,u} = 1\,\mathrm{m\,s^{-1}}$, $\sigma_{accel,w} = 1\,\mathrm{m\,s^{-1}}$, $\sigma_{gyro,\theta} = 0.01\,\mathrm{rad}$ and $\sigma_{gyro,q} = 0.002\,\mathrm{rad\,s^{-1}}$. The state vector $\mathbf{z}$ is given by:

$$\mathbf{z} = \begin{bmatrix} p_d \\ u \\ w \\ \theta \\ q \end{bmatrix}. \tag{5.24}$$

The input vector $\mathbf{u}$ is given by:

$$\mathbf{u} = \begin{bmatrix} \delta_e \\ \delta_t \end{bmatrix}. \tag{5.25}$$

The non-linear function $\mathcal{F}(\cdot)$ is obtained from (3.8) and the observation function $\mathcal{H}(\cdot)$ is given by:

$$\mathcal{H}(\mathbf{z}) = \begin{bmatrix} -1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \mathbf{z} \tag{5.26}$$

The extended vector $\mathbf{x}$ is given by:

$$\mathbf{x} = \begin{bmatrix} \mathbf{z} \\ f_{\delta_e} \\ f_q \end{bmatrix}. \tag{5.27}$$

The actuator and sensor faults follow a zero order model. The process model is then the form of:

$$\begin{cases} \bar{x}_k = \mathbf{F}_k \bar{x}_{k-1} + \mathbf{B}_k \mathbf{u}_k + \boldsymbol{\eta}_k & \text{(5.28a)} \\ \mathbf{y}_k = \mathbf{H}_k \mathbf{x}_k + \boldsymbol{\nu}_k & \text{(5.28b)} \end{cases}$$

The process noise $\boldsymbol{\eta}$ is a white Gaussian noise with a standard deviation $\sigma_{\mathbf{x}}$ given by:

$$\sigma_{\mathbf{x}} = \begin{bmatrix} 0.01\,\mathrm{m} \\ 0.02\,\mathrm{m\,s^{-1}} \\ 0.02\,\mathrm{m\,s^{-1}} \\ 0.005\,\mathrm{rad} \\ 0.002\,\mathrm{rad\,s^{-1}} \\ 0.005\,\mathrm{rad} \\ 0.005\,\mathrm{rad\,s^{-1}} \end{bmatrix} \tag{5.29}$$

The measurement noise of the process model is 1.5 times the measurements noise of (5.12); At a sampling rate of 20 Hz, the matrix $\mathbf{F}_k$, $\mathbf{B}_k$ and $\mathbf{H}_k$ are respectively given by:

$$\mathbf{F}_k = \begin{bmatrix} 1 & 0 & 0.05 & -2 & 0 & 0.01 & 0 \\ 0 & 0.98 & 0.01 & -0.48 & -0.09 & 0.08 & 0 \\ 0 & -0.02 & 0.88 & -0.02 & 1.85 & -1.70 & 0 \\ 0 & 0 & 0 & 1 & 0.05 & -0.05 & 0 \\ 0 & 0 & -0.04 & 0 & 0.93 & -2.09 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.30}$$

,

$$\mathbf{B}_k = \begin{bmatrix} 0.01 & 0 \\ 0.08 & 1.59 \\ -1.70 & -0.01 \\ -0.05 & 0 \\ -2.09 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \tag{5.31}$$

and

$$\mathbf{H}_k = \begin{bmatrix} -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 \end{bmatrix} \tag{5.32}$$

The transition probability matrix of the RJMRPF is given by:

$$\mathbf{\Pi} = \begin{bmatrix} 0.99 & 0.01 \\ 0.01 & 0.99 \end{bmatrix} \tag{5.33}$$

The standard deviation used to compute the initial covariance matrix $\mathbf{P}_0$ for all filters, denoted $\sigma_{\mathbf{x0}}$ is given by:

$$\sigma_{\mathbf{x0}} = \begin{bmatrix} 1\,\mathrm{m} \\ 1\,\mathrm{m\,s^{-1}} \\ 1\,\mathrm{m\,s^{-1}} \\ 0.002\,\mathrm{rad} \\ 0.002\,\mathrm{rad\,s^{-1}} \\ 0.002\,\mathrm{rad} \\ 0.002\,\mathrm{rad\,s^{-1}} \end{bmatrix} \tag{5.34}$$

The fault scenario considered lasts 50 s. For the first 10 s, no fault is active in the system. At 10 s, an actuator fault occurs on the elevator deflection. It is an abrupt fault with an amplitude of 10°. This fault is deactivated at 30 s. However, before that, at 20 s, a sensor fault occurs on the pitch rate measurement. It is also an abrupt fault, with an amplitude of $10\,^\circ\,\mathrm{s^{-1}}$.

The scenario is illustrated in Figure 5.13.

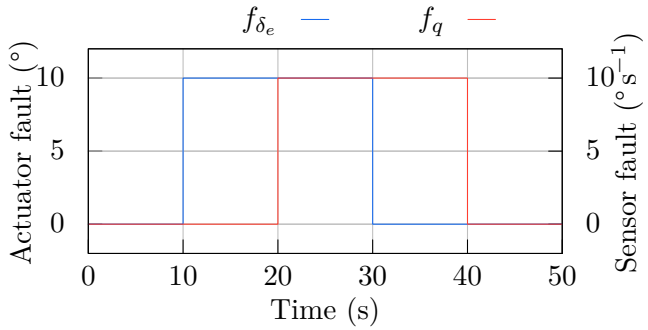
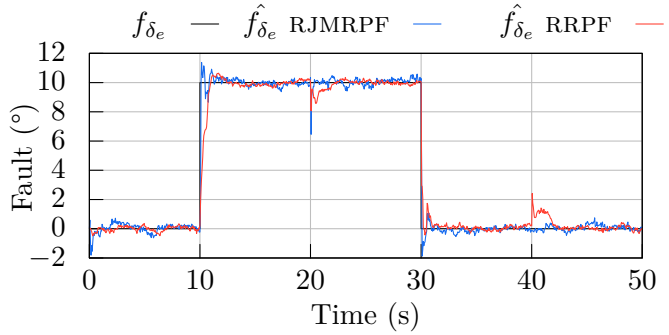


Figure 5.13: Fault scenario for the simulation of ambiguous actuator and sensor fault with a fault on the elevator deflection and the pitch rate measurements.

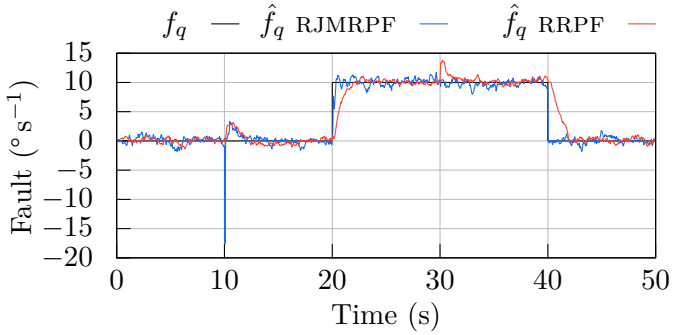
Finally, the number of particles is set to 1000 and the resampling threshold Γ_{rspl} is set to 0.75 and the bandwidth factor h of the Epanechnikov kernel is set 0.2817. The number of simulations performed is 100 with a time step of 0.05 s.

The first results illustrated in Figure 5.14 are the estimates of the RJMRPF and the RRPF by taking the median results from 100 simulations. The selection of the median result is detailed in Appendix D.

In Figure 5.14a, the RJMRPF estimates the actuator fault faster than the RRPF when it occurs at 10 s, although the estimate of the sensor fault in Figure 5.14b is briefly disrupted for the RJMRPF estimate. This deviation due to the ambiguity between the actuator and the sensor is however so brief that the estimate of the state vector $\mathbf{z}$ should not be affected, as hypothesis testing quickly resolves the ambiguity between sensor and actuator faults.



(a) Actuator fault.

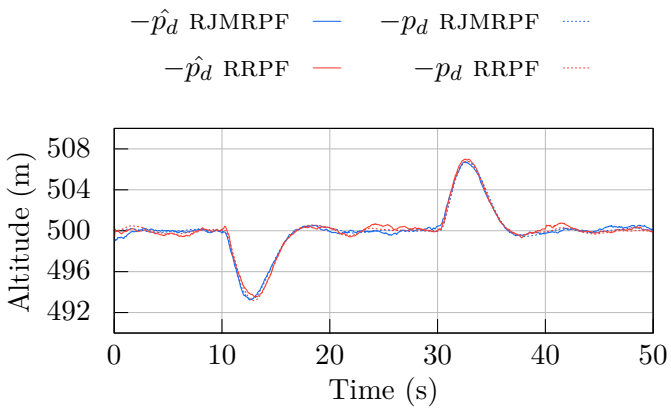


(b) Sensor fault.

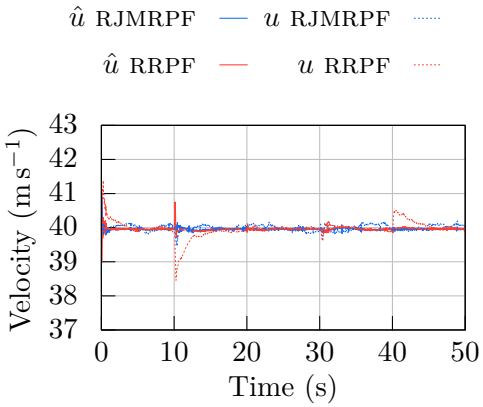
Figure 5.14: Median result of the fault states of the UAV under ambiguous actuator and sensor fault estimated by a RJMRPF and a RRPF. Median results based on 100 simulations.

Between 20 s and 30 s, both faults are simultaneously active and the RJMRPF converges faster to the fault compared to the RRPF. Both estimators then accurately track the sensor fault. When the sensor fault is no longer active, the RJMRPF quickly converges to zero but the RRPF response to this change is approximately 2 s slower because the response of the RPF is more heavily restricted by the model dynamics, while the jump strategy of the RJMRPF has a more instantaneous effect. This is also visible in a less significant way in the actuator fault estimate at the activation, which shows that the jump strategy introduced in Chapter 4 on the actuator fault is working.

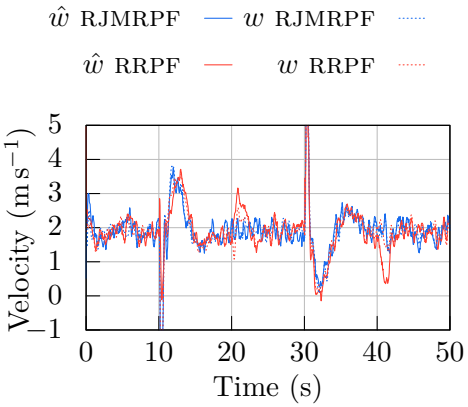
The median results of the state vector $\mathbf{z}$ estimated by the RJMRPF and the RRPF are shown in Figure 5.15.



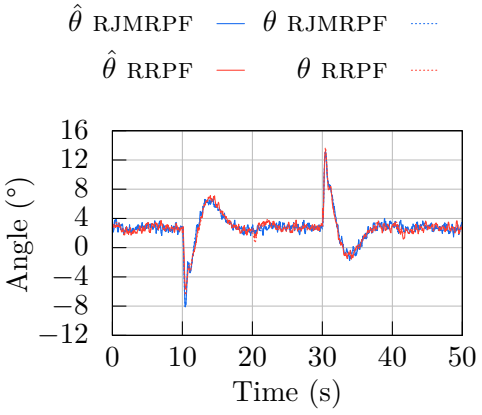
(a) Altitude.



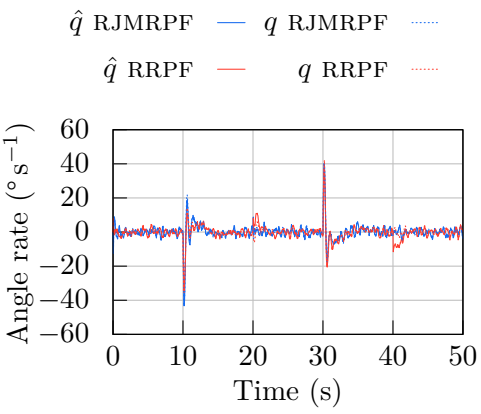
(b) Velocity along i^b.



(c) Velocity along k^b.



(d) Pitch.

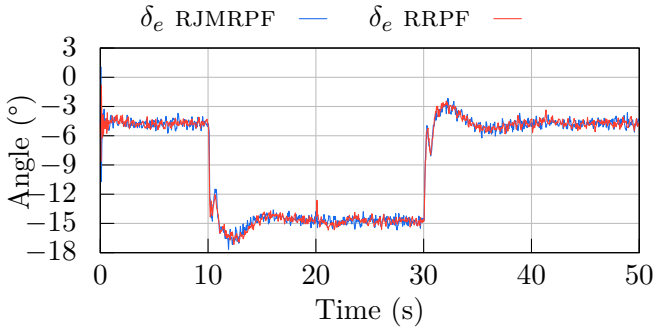


(e) Pitch rate.

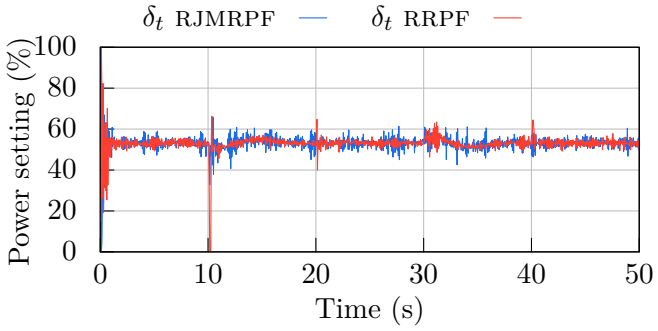
Figure 5.15: Median result of the longitudinal states of the UAV under ambiguous actuator and sensor fault estimated by a RJMRPF and a RRPF. Median results based on 100 simulations.

In Figure 5.15 the effect of the abrupt actuator fault is visible on the altitude even if the fault is quickly and accurately estimated, because the actuator dynamics take a while to compensate the error. However, the variations are only visible at the activation and deactivation of the actuator fault.

The control inputs of the RJMRPF and the RRPF are shown in Figure 5.16.



(a) Elevator deflection.

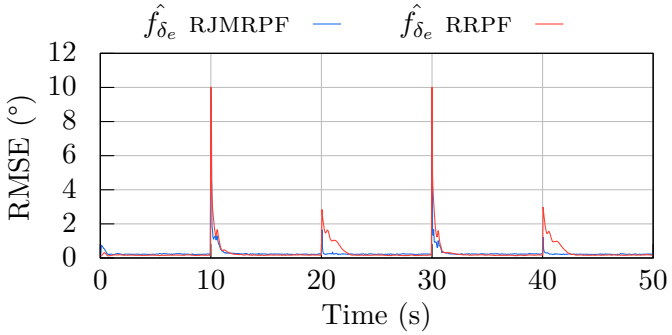


(b) Throttle.

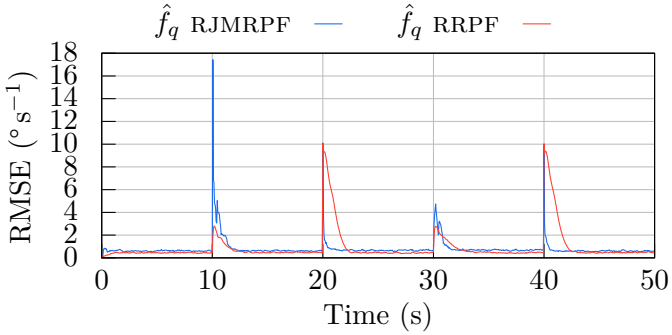
Figure 5.16: Control inputs of the median result of the UAV under ambiguous actuator and sensor fault estimated by a RJMRPF and a RRPF. Median results based on 100 simulations.

In Figure 5.16, the actuator fault compensation is visible on the elevator deflection.

The RMSE of the fault state f_{δ_e} and f_q for the RJMRPF and the RRPF estimates are shown in Figure 5.17.



(a) Actuator fault.

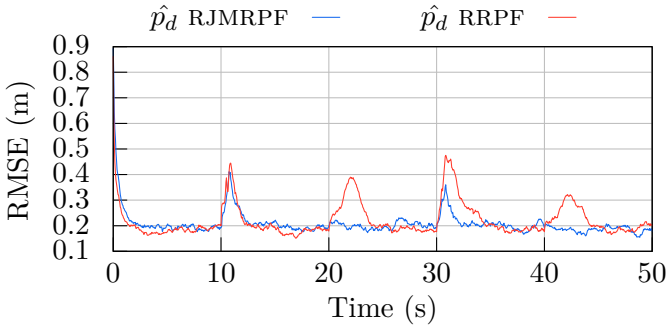


(b) Sensor fault.

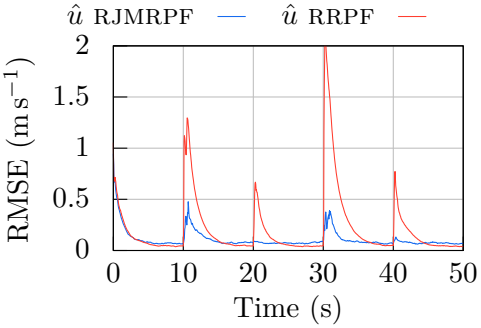
Figure 5.17: RMSE of the fault states of the UAV under ambiguous actuator and sensor fault estimated by a RJMRPF and a RRPF. Results are based on 100 simulations.

In Figure 5.17, the RMSE results are better overall for the RJMRPF, except at 10 s where a short peak is visible on the RMSE of the sensor fault, this is due to the ambiguity created by the actuator fault and is being attempted to be resolved by the sensor fault estimate. The RRPF having this bump too on its RMSE but lower since without jump strategy it cannot reach these level as fast as the RJMRPF.

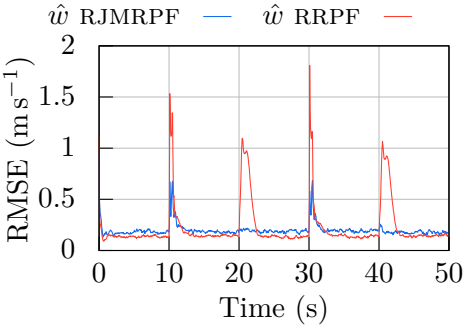
The RMSE of the states is shown in Figure 5.18.



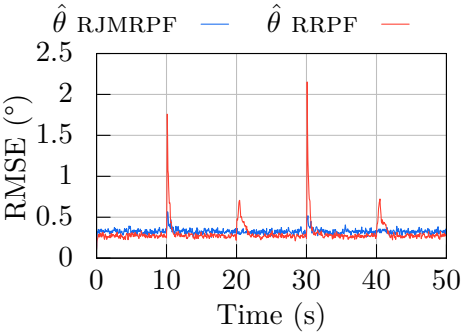
(a) RMSE of the altitude estimate.



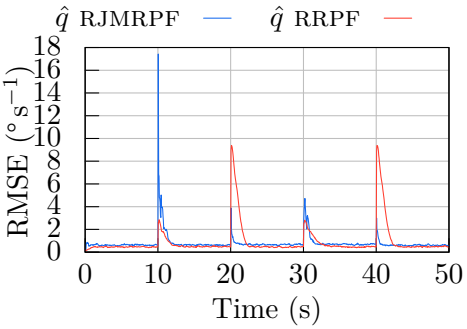
(b) RMSE of the velocity along i^b estimate.



(c) RMSE of the velocity along k^b estimate.



(d) RMSE of the pitch estimate.



(e) RMSE of the pitch rate estimate.

Figure 5.18: RMSE of the longitudinal states of the UAV under ambiguous actuator and sensor fault estimated by a RJMRPF and a RRPF. Results are based on 100 simulations.

In Figure 5.18, the RMSE of all the states are better for the RJMRPF, except at 10 s for the pitch rate state where the short peak is visible on the RMSE of the RJMRPF. However, as mentioned before, this peak does not affect the trajectory of the UAV or any other state since it is too short. The convergence of the RRPF is slower for all estimated states, both when the fault is present and when the system is fault free.

Table 5.3 gives the RMSE values at key time steps and the $\overline{\text{RMSE}}$ which is defined in Appendix D.

State	Time (s)				$\overline{\text{RMSE}}$
	10.05 s	20.05 s	30.05 s	40.05 s	
RJMRPF					
RMSE p_d (m)	0.196	0.204	0.208	0.203	0.208
RMSE u ($\mathrm{m\,s^{-1}}$)	0.080	0.087	0.102	0.077	0.110
RMSE w ($\mathrm{m\,s^{-1}}$)	0.235	0.200	0.399	0.222	0.198
RMSE θ (°)	0.373	0.326	0.467	0.338	0.326
RMSE q ($^\circ\,\mathrm{s^{-1}}$)	17.429	3.875	2.867	2.998	0.797
RMSE f_{δ_e} (°)	8.053	1.659	1.600	1.211	0.290
RMSE f_q ($^\circ\,\mathrm{s^{-1}}$)	17.428	3.882	2.876	3.015	0.807
RRPF					
RMSE p_d (m)	0.275	0.196	0.211	0.204	0.222
RMSE u ($\mathrm{m\,s^{-1}}$)	0.247	0.128	0.239	0.130	0.213
RMSE w ($\mathrm{m\,s^{-1}}$)	0.520	0.253	0.516	0.231	0.228
RMSE θ (°)	0.719	0.453	0.630	0.371	0.302
RMSE q ($^\circ\,\mathrm{s^{-1}}$)	2.634	8.944	2.540	8.870	0.931
RMSE f_{δ_e} (°)	4.618	2.833	4.809	2.973	0.339
RMSE f_q ($^\circ\,\mathrm{s^{-1}}$)	1.912	9.249	1.876	9.186	0.940

Table 5.3: RMSE values of the RJMRPF and the RRPF estimates at key time steps, and $\overline{\text{RMSE}}$.

5.6 CHAPTER SUMMARY

This chapter introduces new features that have been added to extend the abilities of the previously introduced filter, the JMRPF. The new enhanced JMRPF named the RJMRPF was tested and validated with numerical results and compared to a JMRPF and RPF in the case of faults with unknown dynamics. In the case of ambiguous actuator and sensor faults (research question 5), neither the JMRPF nor the RPF could converge in a sufficient number of simulations to provide interesting comparative results. The particles placement using a Kalman update was proven to be necessary to estimate this type of faults with this type of filters. A comparison was therefore also done against the RRPF — which is a RPF with a Kalman update.

The idea behind the RJMRPF was to move particles after there prediction to more likely regions of the state space using a Kalman correction. The associated algorithm of this method was fully described in Section 5.4. 100 simulations were performed in Section 5.5 and the filter was shown to provide very good fault and state estimation performance in the case of faults with unknown dynamics (research question 4) or amplitude (research question 3), as well as in the case of ambiguous actuator and sensor faults. The filter was shown to estimate faults with unknown dynamics by only using a simplified piecewise constant (zero order) process model. This circumvents the need for more than one faulty mode, as in the case of a IMM architecture. The use of only one model also allows for better computational performances in embedded applications, such as fixed-wing UAV (research question 7).

However, a shared limitation of the RJMRPF and JMRPF is that these results were also due to a favourable configuration of the transition probability matrix. For the one or two faults under consideration, allowing 1 % of the particles to move from $m^{(0)}$ to $m^{(1)}$, or from $m^{(1)}$ to $m^{(0)}$ proved to be a good setting. With a different application or with more faults, different settings of the transition probability matrix should be tested to check if this matrix still provides optimal or nearly optimal results, depending on the user specified requirements. Moreover, multiple trials are not always possible to check if a parameter provides acceptable results. This problem is also encountered in IMM approaches, where the transition probability matrix is also a very sensitive tuning parameter.

The following chapter will therefore aim to solve this limitation of the RJMRPF and the JMRPF.

6

ADAPTIVE JUMP-MARKOV REGULARIZED PARTICLE FILTER

The need to achieve a trade-off between false alarm rate and hit rate is a well known problem in fault detection using a variety of methods, including particle filters, as in work by Kadirkamanathan et al. [75] where the effect of detection threshold selection on the trade-off between those two competing objectives was illustrated. In [76], a particle filter was also applied to state and fault estimation for an unmanned underwater vehicle in the presence of multiple sensor and actuator faults, with a threshold selection to achieve a trade-off between desired detection and false alarm probabilities.

One of the limitations of the jump-Markov filters in particular is that assumptions of constant false alarm and missed detection probabilities are often made in their mode switching. This is what has been done with the JMRPF and the RJMRPF, and this chapter aims to solve this issue by introducing the AJMRPF. The AJMRPF allows to adapt the Markov transition matrix to the experiment by estimating the detection and false alarm probabilities. Several approaches have been developed to estimate these probabilities but most of them rely on the use of computationally intensive Monte Carlo approaches. In this chapter, an analytical approach is described to evaluate these probabilities using the saddlepoint approximation [77], first introduced by Daniels [78]. The saddlepoint approximation allows the use of multiple measurements instead of a single one, which increase the accuracy of the approximation.

The false alarm probability can be expressed as a function of a likelihood ratio product of measurements up to time n, given a decision threshold. The approach is detailed in Section 6.2 and has been used to provide an analytical expression of the false alarm and missed detection probability for a number of applications, such as the optimization of sensor networks [79], and sensor fusion for detection in clutter in a stationary case [80]. It is used here together with the JMRPF to develop a multimode state and sensor fault estimation system for unmanned fixed wing aircraft. The saddlepoint approximation is extended here to the case of independent but non identically distributed measurements, as it has been used in [80]. The threshold used for the calculation of false alarm and missed detection probabilities is optimized at each time step using a ROC. The analysis of the numerical simulation on the same test case presented in Section 6.4 illustrates that the AJMRPF performs faster detection and more accurate estimation of the pitch sensor fault, with a robust and accurate state estimation compared to the JMRPF. Mode transitions are adapted to the actual false alarm

and missed detection probabilities, which are both shown to be below 0.5 at all times and close to zero when the fault is active.

The AJMRPF is formulated and applied in this chapter for the estimation of intermittent sensor faults, with a clear dependency between actual false alarm and detection probabilities on one hand and mode switching on the other. An enhanced jump strategy allows a small number of particles to explore the alternate mode to the current hypothesis to reduce computational demand, using an analytical expression based on a generalization of the saddlepoint approximation to independent but non identically distributed measurements, with application to the online computation of false alarm and missed detection probabilities. An optimized threshold selection using a ROC criterion is used to achieve a desired trade-off between false alarm probability and missed detection probability using a ROC approach for the Markovian jump system.

This chapter is organized as follows: the adaptive transition probability matrix formulation is introduced in Section 6.1. The saddlepoint approximation method is presented in Section 6.2, with a construction of the analytical expressions of the false alarm and detection probabilities. The AJMRPF algorithm is then presented in Section 6.3 with a new threshold selection strategy for probabilistic mode transitioning. The numerical simulation analysis in Section 6.4 illustrates the efficiency of the AJMRPF approach and the dependence of the mode switching on the hit-and-miss probabilities. Finally, the Section 6.5 provides an overall conclusion of this chapter.

6.1 ADAPTIVE TRANSITION PROBABILITY MATRIX

As previously defined for the JMRPF, a fault state of particle is associated with mode $m^{(0)}$ if the fault estimate of the particle is in the fault free hypothesis $\mathcal{H}_0$. Likewise, it is associated with mode $m^{(1)}$ if the fault estimate of the particle is in the faulty hypothesis $\mathcal{H}_1$. The JMRPF previously introduced in Chapter 4 and Chapter 5 had the simplest jump strategy, where the mode transition probabilities were kept constant. This strategy was not computationally efficient since particles jump at each time step, whether there is a fault or not. In this chapter, these transition probabilities are adapted to the false alarm rate and hit rate, with an optimization of the trade-off between them. This means that the number of particles allowed to explore mode $m^{(1)}$ or $m^{(0)}$ will not be constant and will depend on current false alarm and missed detection probabilities, which will themselves be dependent on fault estimation.

The elements $\pi_{ji} = \mathbb{P}\left(m_{k+1}^{(j)} | m_k^{(i)}\right)$ of the matrix $\mathbf{\Pi}$, which was defined in (4.2), determine the number of particles that jump. Estimation performance and even stability are sensitive to this transition matrix. Indeed, in Figure 6.1 for example, the same simulation as the one performed in Section 5.5.1 is done with a transition probability matrix set to 0.01 for π_{00} and π_{11}, and 0.99 for π_{10} and π_{01}. This figure shows that with a poor setting of the probability transition matrix RJMRPF and the JMRPF have difficulties to estimate the fault.

In the proposed jump strategy, when a particle jumps to a new fault state, a potential value of this state is calculated. Knowing this value before jumping allows one to compute

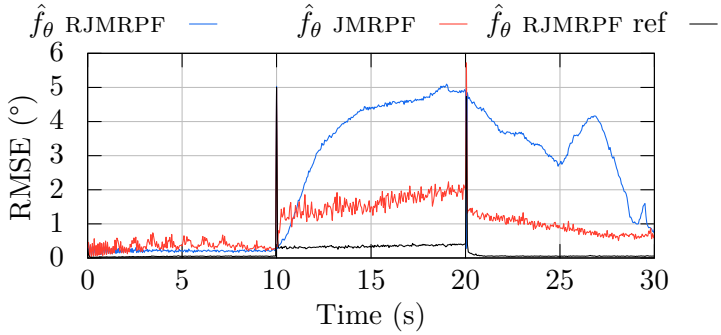


Figure 6.1: First 30 s of the RMSE of the fault states of the simulation performed in Section 5.5.1 with a RJMRPF and a JMRPF with the transition probabilities set to 0.01 for π_{00} and π_{11}, and 0.99 for π_{10} and π_{01}. And with a RJMRPF reference (RJMRPF ref) which is the RMSE obtained in Section 5.5.1. Results based on 100 simulations.

the false alarm probability associated with this potential new value. The missed detection probability is computed with the current estimate of the fault.

Figure 6.2 illustrates the density used to compute the false alarm and missed detection probabilities.

Figure 6.2 illustrates that the computation of the false alarm and missed detection probability are not using the same density $\mathcal{H}_1$ (see Section 6.3.1). Indeed, the false alarm probability is used to know if it is relevant to move the particle from 0 to Δ_f, then the mean of the $\mathcal{H}_1$ density used to compute the false alarm probability is Δ_f. On the other hand, the missed detection probability is used to know if it is relevant to revert to 0 from the current estimate of the fault $\hat{\mathrm{f}}_k$, then the mean of the $\mathcal{H}_1$ density used to compute the missed detection probability is $\hat{\mathrm{f}}_k$.

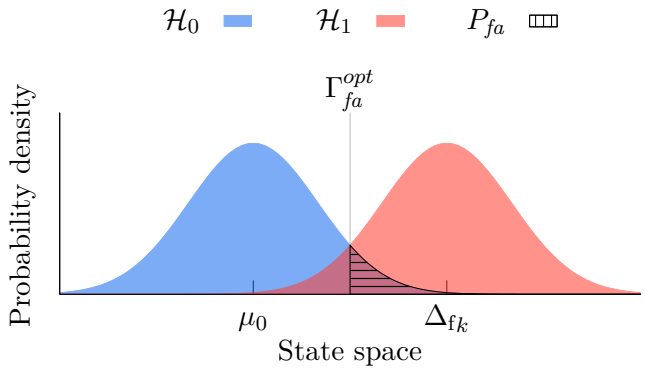
Once the false alarm and missed detection probabilities are computed, the transition probability matrix can be updated. There are several ways of computing the transition probability matrix [81], but the general idea is that the mode switching probabilities are a function of the false alarm and missed detection probabilities.

The proposed adaptation law for the transition probability matrix is given by:

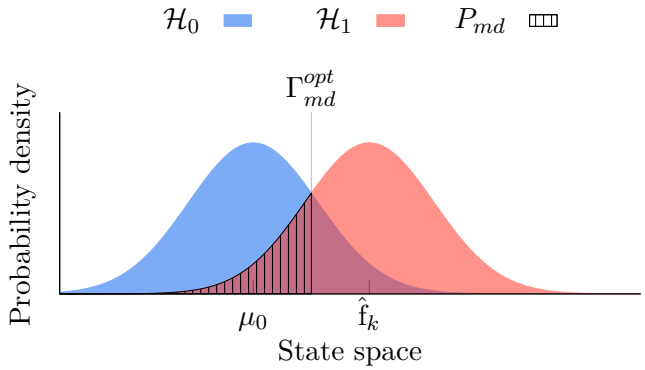
$$\Pi = \begin{bmatrix} \phi_{fa}\left(P_{fa}\right) & 1 - \phi_{fa}\left(P_{fa}\right) \\ 1 - \phi_{md}\left(P_{md}\right) & \phi_{md}\left(P_{md}\right) \end{bmatrix} \tag{6.1}$$

where $\phi_{fa}(\cdot)$ and $\phi_{md}(\cdot)$ are called activation functions. They are continuous functions in the interval [0, 1] and bounded within this interval in [0, 1]. The activation functions can be identity, binary step, sigmoid, or any other functions that satisfy the above criterion.

In the case of only considering a single measurement, the false alarm and missed detection probabilities can be calculated using a CDF as shown in Figure 2.5. To improve the accuracy of the computation of the P_{fa} and the P_{md}, the number of measurements considered must be greater than 1. In this case, this approach is however difficult to generalize and the P_{fa} and P_{md} probabilities can be calculated using Monte Carlo approximations. These approximations



(a) False alarm probability with $\mathcal{H}_0$ and $\mathcal{H}_1$ density.



(b) Missed detection probability with $\mathcal{H}_0$ and $\mathcal{H}_1$ density.

Figure 6.2: False alarm and missed detection probabilities with the $\mathcal{H}_0$ density and the $\mathcal{H}_1$ density used for the computation of each probability, with Γ_{fa}^{opt} and Γ_{md}^{opt} the optimal threshold for the false alarm and missed detection probabilities respectively.

are however computationally expensive, and then not suitable for real-time applications because of the large number of samples needed to accurately approximate the P_{fa} and P_{md}. A solution is proposed in this chapter, using the saddlepoint approximation detailed in Section 6.2 to calculate an analytical expression to both false alarm and missed detection probabilities from the last n measurements.

6.2 APPROXIMATION OF THE FALSE ALARM AND MISSED DETECTION

This section aims to present a way to approximate the false alarm and missed detection probabilities and use them to compute the adaptive transition probability matrix given by (6.1). To do so, the saddlepoint approximation method is used.

The saddlepoint approximation was introduced by Daniels [78] and has since been developed and enhanced for a wealth of applications. Given a number n of measurements, the saddlepoint approximation of the false alarm probability can be computed using the Lugannani and Rice

formula [82] and is accurate to order $1/n$ which outperforms the accuracy of a Monte Carlo approximation, known to be accurate to order $1/\sqrt{n}$.

The saddlepoint approximation then consists of approximating the probability density function of the mean of n independent and identically distributed (i. i. d.) random variables. Let us consider X_i as a random variable. The mean of n i. i. d. random variables is given by:

$$\bar{X} = \frac{1}{n}\sum_{i=1}^{n} X_i \tag{6.2}$$

The saddlepoint approximation remains accurate even at tail distribution areas, which is crucial for a number of applications, including detection problems and particularly when detection algorithms are designed to meet false alarm probability requirements. Compared to the saddlepoint approximation, the law of large numbers approximation only works for absolute error with very large values of n and provides poor accuracy in the tail distribution areas.

Many statistical problems (maximum likelihood estimator or hypothesis testing for example) can be formulated as a sum of n i. i. d. variables using the logarithm of the likelihood function.

In the following section, the saddlepoint approximation formulas in the i. i. d. and independent but not identically distributed cases, are re-demonstrated as the Lugannani and Rice formula, using new proofs.

In this section, different and simplified proofs of the formulae of the saddlepoint approximation of the mean and of Lugannani-Rice formula are provided. This is performed using carefully chosen changes of variables from the saddlepoint formula for the case $n = 1$ and integration by parts. This approach to proving the saddlepoint approximation provides a new insight into the problem and clearly differs from the proofs made in [78, 82] and based on the Edgeworth expansion. Furthermore, a generalization of the saddlepoint formula to the case of independent and non-identically distributed variables, is proposed with a new mathematical proof.

6.2.1 *Saddlepoint approximation of a sum or a mean*

In the case where the random variable X_i are i. i. d. variables, the density of $p_{\bar{X}}(x)$ is given by the Proposition 1.

Proposition 1

A random variable $\bar{X}$ is introduced such that $\bar{X} = \frac{1}{n}\sum_{i=1}^{n} X_i$. If the random variables X_i are i. i. d., then the probability density function of $\bar{X}$ is given by:

$$\hat{p}_{\bar{X}}(x) = \sqrt{\frac{n}{2\pi\mathcal{K}_X''(T_X)}}\, e^{n(\mathcal{K}_X(T_X) - T_X x)} \tag{6.3}$$

Proof . The cumulant generating function $\mathcal{K}_{\bar{X}}$ of the variable $\bar{X}$ is given by:

$$\mathcal{K}_{\bar{X}}(t) = \log\left(\mathbb{E}\left[e^{t\bar{X}}\right]\right) \tag{6.4a}$$

$$= \log\left(\mathbb{E}\left[e^{t\left(\frac{1}{n}\sum_{i=1}^{n} X_i\right)}\right]\right) \tag{6.4b}$$

$$= \log\left(\mathbb{E}\left[\prod_{i=1}^{n} e^{t\left(\frac{1}{n}X_i\right)}\right]\right) \tag{6.4c}$$

Since the variables X_i are independent, the cumulant generating function can also be written as:

$$\mathcal{K}_{\bar{X}}(t) = \log\left(\mathbb{E}\left[\prod_{i=1}^{n} e^{t\left(\frac{1}{n}X_i\right)}\right]\right) \tag{6.5a}$$

$$= \log\left(\prod_{i=1}^{n} \mathbb{E}\left[e^{t\left(\frac{1}{n}X_i\right)}\right]\right) \tag{6.5b}$$

$$= \sum_{i=1}^{n} \log\left(\mathbb{E}\left[e^{t\left(\frac{1}{n}X_i\right)}\right]\right) \tag{6.5c}$$

$$= \sum_{i=1}^{n} \mathcal{K}_{X_i}\left(\frac{t}{n}\right) \tag{6.5d}$$

Given that the variables X_i have the same probability density function, their cumulant generating function is identical $\mathcal{K}_{X_i} = \mathcal{K}_X$. Then, it can be written as:

$$\mathcal{K}_{\bar{X}}(t) = n\mathcal{K}_X\left(\frac{t}{n}\right) \tag{6.6}$$

and its derivative is then given by:

$$\mathcal{K}'_{\bar{X}}(t) = \mathcal{K}'_X\left(\frac{t}{n}\right) \tag{6.7}$$

As $\mathcal{K}'_{\bar{X}}(t) = \mathcal{K}'_X\left(\frac{t}{n}\right)$, the cumulant generating $\mathcal{K}'_{\bar{X}}$ can be written:

$$\mathcal{K}'_{\bar{X}}(t) - x = 0 \Leftrightarrow \mathcal{K}'_X\left(\frac{t}{n}\right) - x = 0 \tag{6.8}$$

Let $T_{\bar{X}}$ be the root of $\mathcal{K}'_{\bar{X}}(t) - x = 0$, and T_X the root of $\mathcal{K}'_X\left(\frac{t}{n}\right) - x$. Then, it gives $T_X = \frac{T_{\bar{X}}}{n} \Leftrightarrow T_{\bar{X}} = nT_X$.

By derivate (6.7) it gives:

$$\mathcal{K}''_{\bar{X}}(t) = \mathcal{K}''_X\left(\frac{t}{n}\right) \tag{6.9}$$

The probability density function of $\bar{X}$ is given using the Laplace approximation — detailed in Appendix E — by:

$$\hat{p}_{\bar{X}}(x) = \sqrt{\frac{1}{2\pi \mathcal{K}''_{\bar{X}}(T_{\bar{X}})}} e^{(\mathcal{K}_{\bar{X}}(T_{\bar{X}}) - T_{\bar{X}} x)} \tag{6.10}$$

Then, it gives $T_{\bar{X}} = nT_X$, $\mathcal{K}_{\bar{X}}(T_{\bar{X}}) = n\mathcal{K}_X(T_X)$ and $\mathcal{K}''_{\bar{X}}(T_{\bar{X}}) = \frac{1}{n}\mathcal{K}''_X(T_X)$. By substituting this in (6.10), it gives:

$$\hat{p}_{\bar{X}}(x) = \sqrt{\frac{1}{2\pi\left(\frac{1}{n}\mathcal{K}''_X(T_X)\right)}} e^{(n\mathcal{K}_X(T_X) - nT_X x)} \tag{6.11a}$$

$$= \sqrt{\frac{n}{2\pi \mathcal{K}''_X(T_X)}} e^{n(\mathcal{K}_X(T_X) - T_X x)} \tag{6.11b}$$

This formula is the saddlepoint approximation developed by Daniels under the assumption that the variables X_i are i. i. d..

In the case where the variables X_i are independent but not identically distributed, a generalized formulation of the saddlepoint approximation has been given in [80]. This formulation is given by the following proposition.

Proposition 2

Let the random variable $\bar{X}$ be defined such that $\bar{X} = \frac{1}{n}\sum_{i=1}^{n} X_i$. If the random variables X_i are independent but not identically distributed, the probability density function of $\bar{X}$ is given by:

$$\hat{p}_{\bar{X}}(x) = \sqrt{\frac{n}{2\pi \bar{\mathcal{K}}''_X(T_0)}} e^{n(\bar{\mathcal{K}}_X(T_0) - T_0 x)} \tag{6.12}$$

where:

$$\bar{\mathcal{K}}_X(T_0) = \frac{1}{n}\sum_{i=1}^{n} \mathcal{K}_{X_i}(T_0) \tag{6.13}$$

and

$$\bar{\mathcal{K}}''_X(T_0) = \frac{1}{n}\sum_{i=1}^{n} \mathcal{K}''_{X_i}(T_0) \tag{6.14}$$

Proof . The cumulant generating function $\mathcal{K}_{\bar{X}}$ of the variable $\bar{X}$ is given by:

$$\mathcal{K}_{\bar{X}}(t) = \log\left(\mathbb{E}\left[e^{t\bar{X}}\right]\right) \tag{6.15a}$$

$$= \log\left(\mathbb{E}\left[e^{t\left(\frac{1}{n}\sum_{i=1}^{n} X_i\right)}\right]\right) \tag{6.15b}$$

$$= \log\left(\mathbb{E}\left[\prod_{i=1}^{n} e^{t\left(\frac{1}{n}X_i\right)}\right]\right) \tag{6.15c}$$

Given that the variables X_i are independent, the cumulant generating function can also be written as follows:

$$\mathcal{K}_{\bar{X}}(t) = \log\left(\mathbb{E}\left[\prod_{i=1}^{n} e^{t\left(\frac{1}{n}X_i\right)}\right]\right) \tag{6.16a}$$

$$= \log\left(\prod_{i=1}^{n} \mathbb{E}\left[e^{t\left(\frac{1}{n}X_i\right)}\right]\right) \tag{6.16b}$$

$$= \sum_{i=1}^{n} \log\left(\mathbb{E}\left[e^{t\left(\frac{1}{n}X_i\right)}\right]\right) \tag{6.16c}$$

$$= \sum_{i=1}^{n} \mathcal{K}_{X_i}\left(\frac{t}{n}\right). \tag{6.16d}$$

And its derivative is given by:

$$\mathcal{K}'_{\bar{X}}(t) = \frac{1}{n}\sum_{i=1}^{n} \mathcal{K}'_{X_i}\left(\frac{t}{n}\right). \tag{6.17}$$

Since $T_{\bar{X}}$ is the root of:

$$\mathcal{K}'_{\bar{X}}(t) - x = 0 \Leftrightarrow \frac{1}{n}\sum_{i=1}^{n} \mathcal{K}'_{X_i}\left(\frac{t}{n}\right) - x = 0, \tag{6.18}$$

It gives $\frac{T_{\bar{X}}}{n}$ is the solution of $\frac{1}{n}\sum_{i=1}^{n} \mathcal{K}'_{X_i}(t) - x = 0$.

By deriving (6.17), it gives:

$$\mathcal{K}''_{\bar{X}}(t) = \frac{1}{n^2}\sum_{i=1}^{n} \mathcal{K}''_{X_i}\left(\frac{t}{n}\right) \tag{6.19}$$

The probability density function of $\bar{X}$ as detailed in Appendix E is given by:

$$\hat{p}_{\bar{X}}(x) = \sqrt{\frac{1}{2\pi\mathcal{K}''_{\bar{X}}(T_{\bar{X}})}}\, e^{\mathcal{K}_{\bar{X}}(T_{\bar{X}}) - T_{\bar{X}}x} \tag{6.20}$$

Let $T_0 = \frac{T_{\bar{X}}}{n}$, then it gives $T_{\bar{X}} = nT_0$, $\mathcal{K}_{\bar{X}}(T_{\bar{X}}) = \mathcal{K}_{X_i}(T_0)$ and $\mathcal{K}''_{\bar{X}}(T_{\bar{X}}) = \frac{1}{n^2}\mathcal{K}''_{X_i}(T_0)$. By substituting this in (6.20), it gives:

$$\hat{p}_{\bar{X}}(x) = \sqrt{\frac{1}{2\pi\left(\frac{1}{n^2}\sum_{i=1}^{n}\mathcal{K}''_{X_i}(T_0)\right)}}\, e^{\left(\sum_{i=1}^{n}\mathcal{K}_{X_i}(T_0) - nT_0x\right)} \tag{6.21a}$$

$$= \sqrt{\frac{n}{2\pi\left(\frac{1}{n}\sum_{i=1}^{n}\mathcal{K}''_{X_i}(T_0)\right)}}\, e^{n\left(\frac{1}{n}\sum_{i=1}^{n}\mathcal{K}_{X_i}(T_0) - T_0x\right)}. \tag{6.21b}$$

Let's define $\bar{\mathcal{K}}_X$, $\bar{\mathcal{K}}''_X$ functions as:

$$\bar{\mathcal{K}}_X(T_0) = \frac{1}{n}\sum_{i=1}^{n}\mathcal{K}_{X_i}(T_0) \tag{6.22}$$

$$\bar{\mathcal{K}}''_X(T_0) = \frac{1}{n}\sum_{i=1}^{n}\mathcal{K}''_{X_i}(T_0). \tag{6.23}$$

Then, it gives:

$$\hat{p}_{\bar{X}}(x) = \sqrt{\frac{n}{2\pi\bar{\mathcal{K}}''_X(T_0)}}\, e^{n(\bar{\mathcal{K}}_X(T_0) - T_0x)}. \tag{6.24}$$

This equation is consistent with Daniels's formula (6.11b).

6.2.2 *Lugannani and Rice Formula*

The Lugannani and Rice formula [82] is known to provide a very accurate approximation at the tail probability area. The false alarm and missed detection probabilities can be computed using the Lugannani and Rice formula. This formula is based on the saddlepoint approximation of the cumulative probability function.

Proposition 3
Let the probability density function of $\bar{X}$ be defined such that:

$$\hat{p}_{\bar{X}}(x) = \sqrt{\frac{n}{2\pi\mathcal{K}''_X(T_0)}}\, e^{n(\mathcal{K}_X(T_0) - T_0x)} \tag{6.25}$$

The saddlepoint approximation of the cumulative probability function $\mathbb{P}\left(\bar{X} > \alpha\right)$ *is given by:*

$$\mathbb{P}\left(\bar{X} > \alpha\right) \approx 1 - \Phi\left(y_\alpha\right) + \phi\left(y_\alpha\right)\left(\frac{1}{t_\alpha} - \frac{1}{y_\alpha}\right) \tag{6.26}$$

where $\phi\left(\cdot\right)$ *is the standard Gaussian distribution,* $\Phi\left(\cdot\right)$ *is the CDF of the standard normal distribution,* $t_\alpha = T_0\sqrt{n\mathcal{K}''_X\left(T_0\right)}$ *and* $y_\alpha = \text{sign}\left(t_\alpha\right)\sqrt{2n\left(T_0\alpha - \mathcal{K}_X\left(T_0\right)\right)}$ *with* T_0 *is the solution of the equation* $\mathcal{K}'_X\left(t\right) - \alpha = 0$*, and* α *is a detection threshold.*

Proof . The saddlepoint approximation of the cumulative probability function $\mathbb{P}\left(\bar{X} > \alpha\right)$ is given by:

$$\mathbb{P}\left(\bar{X} > \alpha\right) = \int_\alpha^{+\infty} \hat{p}_{\bar{X}}\left(x\right)\,\mathrm{d}x \tag{6.27a}$$

$$= \int_\alpha^{+\infty} \sqrt{\frac{n}{2\pi\mathcal{K}''_X\left(T_0\right)}} e^{n\left(\mathcal{K}_X\left(T_0\right) - T_0 x\right)}\,\mathrm{d}x \tag{6.27b}$$

A change of variables is performed by using:

$$n\left(\mathcal{K}_X\left(T_0\right) - T_0 x\right) = -\frac{y^2}{2}. \tag{6.28}$$

By replacing $\mathcal{K}'_X\left(T_0\right) = x$ in (6.28), can be written:

$$n\left(\mathcal{K}_X\left(T_0\right) - T_0\mathcal{K}'_X\left(T_0\right)\right) = -\frac{y^2}{2} \tag{6.29}$$

and then, $y\,\mathrm{d}y$ is given by:

$$y\,\mathrm{d}y = nT_0\mathcal{K}''_X\left(T_0\right)\,\mathrm{d}T_0 \tag{6.30}$$

Since $\mathcal{K}'_X\left(T_0\right) = x$, can be obtained by differentiation $\mathcal{K}''_X\left(T_0\right)\,\mathrm{d}T_0 = \mathrm{d}x$. Then, $y\,\mathrm{d}y$ can be written:

$$y\,\mathrm{d}y = nT_0\,\mathrm{d}x. \tag{6.31}$$

This implies that:

$$\mathrm{d}x = \frac{y\,\mathrm{d}y}{nT_0}. \tag{6.32}$$

Then:

$$\mathbb{P}\left(\bar{X} > \alpha\right) = \int_{y_\alpha}^{+\infty} \sqrt{\frac{1}{n\mathcal{K}''_X\left(T_0\right)}}\frac{y}{T_0}\left(\frac{1}{\sqrt{2\pi}}e^{-\frac{y^2}{2}}\right)\mathrm{d}y \tag{6.33}$$

The term $\frac{1}{\sqrt{2\pi}}e^{-\frac{y^2}{2}}$ is the standard Gaussian distribution. Let's define $\phi(y) = \frac{1}{\sqrt{2\pi}}e^{-\frac{y^2}{2}}$ and $t = T_0\sqrt{n\mathcal{K}''_X(T_0)}$. Then, the probability $\mathbb{P}\left(\bar{X} > \alpha\right)$ is given by:

$$\mathbb{P}\left(\bar{X} > \alpha\right) = \int_{y_\alpha}^{+\infty} \frac{y}{t}\phi(y)\,\mathrm{d}y \tag{6.34a}$$

$$= \int_{y_\alpha}^{+\infty} \left(1 - 1 + \frac{y}{t}\right)\phi(y)\,\mathrm{d}y \tag{6.34b}$$

$$= \int_{y_\alpha}^{+\infty} \phi(y)\,\mathrm{d}y + \int_{y_\alpha}^{+\infty} \left(\frac{1}{y} - \frac{1}{t}\right)(-y\phi(y))\,\mathrm{d}y \tag{6.34c}$$

A first integration of the first term in (6.34c) yields:

$$\int_{y_\alpha}^{+\infty} \phi(y)\,\mathrm{d}y = 1 - \Phi(y_\alpha) \tag{6.35}$$

where $\Phi(\cdot)$ is the CDF of the standard normal distribution. The second term in (6.34c) can be integrated by parts by letting $\mathrm{d}V = -y\phi(y)$ and $U = \left(\frac{1}{y} - \frac{1}{t}\right)$. Since $V = \phi(y)$ and $\mathrm{d}U = \mathrm{d}\left(\frac{1}{y} - \frac{1}{t}\right)$. Then, it gives:

$$\mathbb{P}\left(\bar{X} > \alpha\right) = 1 - \Phi(y_\alpha) + \phi(y_\alpha)\left(\frac{1}{t_\alpha} - \frac{1}{y_\alpha}\right) + \int_{y_\alpha}^{+\infty} \phi(y)\,\mathrm{d}\left(\frac{1}{y} - \frac{1}{t}\right) \tag{6.36}$$

where the term $\int_{y_\alpha}^{+\infty} \phi(y)\,\mathrm{d}\left(\frac{1}{y} - \frac{1}{t}\right)$ is a small error that is assumed to be negligible.

Equation (6.36) can then be written:

$$\mathbb{P}\left(\bar{X} > \alpha\right) \approx 1 - \Phi(y_\alpha) + \phi(y_\alpha)\left(\frac{1}{t_\alpha} - \frac{1}{y_\alpha}\right) \tag{6.37}$$

Equation (6.37) is known as the Lugannani and Rice formula [82]. Where $t_\alpha = T_0\sqrt{n\mathcal{K}''_X(T_0)}$ and $y_\alpha = \mathrm{sign}(t_\alpha)\sqrt{2n(T_0\alpha - \mathcal{K}_X(T_0))}$, with $\mathrm{sign}(\cdot)$ is the sign function, T_0 the solution of the equation $\mathcal{K}'_X(t) - \alpha = 0$ and α is a detection threshold.

6.2.3 *Hypothesis testing with the saddlepoint approximation*

The two possible fault diagnosis hypotheses of $\mathcal{H}_0$ and $\mathcal{H}_1$ were introduced in (2.5). This diagnosis is based on the Neyman-Pearson decision criterion and is given by:

$$\Lambda = \prod_{k=1}^{n} \frac{p\left(\tilde{y}_k|\mathcal{H}_1\right)}{p\left(\tilde{y}_k|\mathcal{H}_0\right)} \underset{\mathcal{H}_1}{\overset{\mathcal{H}_0}{\gtrless}} \eta \tag{6.38}$$

The likelihood ratio (6.38) can now be written in the form of (6.2):

$$\bar{X} = \frac{1}{n}\log\left(\Lambda\right) \tag{6.39a}$$

$$= \frac{1}{n}\sum_{k=1}^{n} \log\left(\frac{p\left(\tilde{y}_k|\mathcal{H}_1\right)}{p\left(\tilde{y}_k|\mathcal{H}_0\right)}\right) \tag{6.39b}$$

By substituting the variable $\tilde{y}_k$ by $\bar{X}$, the false alarm and missed detection probabilities defined in (2.8) and (2.9) can be written as:

$$P_{fa} = \mathbb{P}\left(\bar{X} > \alpha|\mathcal{H}_0\right) \tag{6.40}$$

$$P_{md} = \mathbb{P}\left(\bar{X} < \alpha|\mathcal{H}_1\right) \tag{6.41}$$

where the threshold α is equal to

$$\alpha = \frac{1}{n}\log\left(\eta\right) \tag{6.42}$$

Note that the threshold parameter α will not be kept constant at all times to enforce a constant maximum false alarm probability, as is often the case when the Neyman-Pearson criterion is used. As later shown in Section 6.3, the threshold will be selected and optimized at each time from a range of candidate thresholds. This will allow for the calculation of the false alarm and missed detection probabilities using an analytical formula, by exploiting the properties of the log likelihood ratio formulation.

In (6.39), the process $\bar{X}$ is written in the form:

$$\bar{X} = \frac{1}{n}\sum_{k=1}^{n} X_k \tag{6.43}$$

$\bar{X}$ is a complex mixture of densities. Its law cannot be analytically expressed. The samples X_k are independent but not identically distributed because they are computed at each time step k and the mean will vary in the presence of sensor faults. This process used to compute the cumulant generating function is presented in Section 6.2.4.

In the Gaussian case, with $p\left(\tilde{y}_k|\mathcal{H}_0\right) = \mathcal{N}\left(\mu_{0k}, \sigma_{0k}^2\right)$ and $p\left(\tilde{y}_k|\mathcal{H}_1\right) = \mathcal{N}\left(\mu_{1k}, \sigma_{1k}^2\right)$, X_k is given by:

$$X_k = \log\left(\frac{\sigma_{0k}}{\sigma_{1k}}\right) + \frac{1}{2\sigma_{0k}^2}(\tilde{y}_k - \mu_{0k})^2 - \frac{1}{2\sigma_{1k}^2}(\tilde{y}_k - \mu_{1k})^2 \tag{6.44}$$

This expression can be rewritten as:

$$X_k = a_k(\tilde{y}_k)^2 + b_k\tilde{y}_k + c_k \tag{6.45}$$

where a_k, b_k and c_k are the polynomial coefficients given by:

$$\begin{cases} a_k = \dfrac{1}{2{\sigma_1}_k^2} - \dfrac{1}{2{\sigma_0}_k^2} & \text{(6.46a)} \\ b_k = \dfrac{{\mu_0}_k}{{\sigma_0}_k^2} - \dfrac{{\mu_1}_k}{{\sigma_1}_k^2} & \text{(6.46b)} \\ c_k = \log\left(\dfrac{{\sigma_1}_k}{{\sigma_0}_k}\right) + \dfrac{{\mu_1}_k^2}{2{\sigma_1}_k^2} - \dfrac{{\mu_0}_k^2}{2{\sigma_0}_k^2} & \text{(6.46c)} \end{cases}$$

6.2.4 *Analytical expression of the false alarm probability*

The false alarm occurs under hypothesis $\mathcal{H}_0$. It is assumed that $\tilde{y}_k$ follows the Gaussian law $p\left(\tilde{y}_k|\mathcal{H}_0\right)$. The corresponding cumulant generating function is:

$$\begin{aligned} \mathcal{K}_{X_k}\left(t|\mathcal{H}_0\right) &= \log\left(\mathbb{E}\left[e^{tX_k}|\mathcal{H}_0\right]\right) && \text{(6.47a)} \\ &= \log\left(\int e^{t\left(a_k\tilde{y}_k^2 + b_k\tilde{y}_k + c_k\right)} p\left(\tilde{y}_k|\mathcal{H}_0\right)\mathrm{d}\tilde{y}_k\right) && \text{(6.47b)} \end{aligned}$$

The last expression can be expanded using straightforward calculations to:

$$\mathcal{K}_{X_k}\left(t|\mathcal{H}_0\right) = -\frac{1}{2{\sigma_0}_k^2}{\mu_0}_k^2 + c_k t + \frac{\left({\mu_0}_k + b_k{\sigma_0}_k^2 t\right)^2}{{\sigma_0}_k^2\left(2 - 4a_k t\right)} - \frac{1}{2}\log\left(1 - 2a_k{\sigma_0}_k^2 t\right) \tag{6.48}$$

From Proposition 2, the mean's cumulant generating function $\bar{\mathcal{K}}_0\left(t\right)$ is given by:

$$\bar{\mathcal{K}}_0\left(t\right) = \frac{1}{n}\sum_{k=1}^{n}\mathcal{K}_{X_k}\left(t|\mathcal{H}_0\right) \tag{6.49}$$

where $\bar{\mathcal{K}}_0\left(t\right)$ is defined if and only if,

$$\max_{a_k<0}\frac{1}{2a_k{\sigma_0}_k^2} < t < \min_{a_k>0}\frac{1}{2a_k{\sigma_0}_k^2} \tag{6.50}$$

In this interval, the equation $\bar{\mathcal{K}}_0'\left(t\right) - \alpha = 0$ admits a solution T_0.

The false alarm probability is given by:

$$P_{fa} = \mathbb{P}\left(\bar{X} > \alpha|\mathcal{H}_0\right) \tag{6.51}$$

where $\mathbb{P}\left(\bar{X} > \alpha|\mathcal{H}_0\right)$ is approximated by (6.37).

$$P_{fa} \approx 1 - \Phi\left(y_0\right) + \phi\left(y_0\right)\left(\frac{1}{t_0} - \frac{1}{y_0}\right) \tag{6.52}$$

where $t_0 = T_0\sqrt{n\bar{\mathcal{K}}_0''\left(T_0\right)}$ and $y_0 = \mathrm{sign}\left(t_0\right)\sqrt{2n\left(T_0\alpha - \bar{\mathcal{K}}_0\left(T_0\right)\right)}$.

6.2.5 *Analytical expression of the missed detection probability*

A missed detection occurs under hypothesis $\mathcal{H}_1$. It is assumed that $\tilde{y}_k$ follows the Gaussian law $p\left(\tilde{y}_k|\mathcal{H}_1\right)$. The corresponding cumulant generating function is:

$$\mathcal{K}_{X_k}\left(t|\mathcal{H}_1\right) = \log\left(\mathbb{E}\left[e^{tX_k}|\mathcal{H}_1\right]\right) \tag{6.53a}$$

$$= \log\left(\int e^{t\left(a_k\tilde{y}_k^2+b_k\tilde{y}_k+c_k\right)} p\left(\tilde{y}_k|\mathcal{H}_1\right) \mathrm{d}\tilde{y}_k\right) \tag{6.53b}$$

The last expression can be expanded using straightforward calculations to:

$$\mathcal{K}_{X_k}\left(t|\mathcal{H}_1\right) = -\frac{1}{2\sigma_{1_k}^2}\mu_{1_k}^2 + c_k t + \frac{\left(\mu_{1_k} + b_k\sigma_{1_k}^2 t\right)^2}{\sigma_{1_k}^2\left(2-4a_k t\right)} - \frac{1}{2}\log\left(1-2a_k\sigma_{1_k}^2 t\right) \tag{6.54}$$

From Proposition 2, the mean's cumulant generating function $\bar{\mathcal{K}}_1\left(t\right)$ is then given by:

$$\bar{\mathcal{K}}_1\left(t\right) = \frac{1}{n}\sum_{k=1}^{n}\mathcal{K}_{X_k}\left(t|\mathcal{H}_1\right) \tag{6.55}$$

where $\bar{\mathcal{K}}_1\left(t\right)$ is defined if and only if,

$$\max_{a_k<0}\frac{1}{2a_k\sigma_{1_k}^2} < t < \min_{a_k>0}\frac{1}{2a_k\sigma_{1_k}^2} \tag{6.56}$$

In this interval, the equation $\bar{\mathcal{K}}_1'\left(t\right) - \alpha = 0$ admits a solution T_1.

The missed detection probability is given by:

$$P_{md} = 1 - P_d = 1 - \mathbb{P}\left(\bar{X} > \alpha|\mathcal{H}_1\right) \tag{6.57}$$

where P_d denotes the detection probability. Finally, the missed detection probability using (6.37) is given by:

$$P_{md} \approx \Phi\left(y_1\right) - \phi\left(y_1\right)\left(\frac{1}{t_1} - \frac{1}{y_1}\right) \tag{6.58}$$

where $t_1 = T_1\sqrt{n\bar{\mathcal{K}}_1''\left(T_1\right)}$ and $y_1 = \mathrm{sign}\left(t_1\right)\sqrt{2n\left(T_1\alpha - \bar{\mathcal{K}}_1\left(T_1\right)\right)}$.

6.3 FORMULATION OF THE ADAPTIVE JUMP-MARKOV REGULARIZED PARTICLE FILTER

The stochastic process model of the AJMRPF for additive actuator and sensor fault is given by 4.5. The AJMRPF introduced in this chapter is only formulated for sensor faults.

Unlike the case of JMRPF, the prediction step of the AJMRPF also updates the transition probability matrix. The formula of the new transition probability matrix is given by (6.1). In

this equation, the transition probabilities depend on the false alarm and the missed detection probabilities. In Section 6.2, it has been shown that these probabilities can be accurately computed with a number of samples $n \geq 3$. Then the formula given by (6.52) is used here to evaluate the false alarm probability of switching a particle from a nominal mode $m^{(0)}$ to a faulty mode $m^{(1)}$. In the same way, the formula given by (6.58) is used here to evaluate the missed detection probability of switching a particle from a faulty mode $m^{(1)}$ to a nominal mode $m^{(0)}$. However, these two formula are computed for given means μ_0 and μ_1. The JMRPF and application studied so far focused on change in mean, the variance of the $\mathcal{H}_1$ hypothesis is then assumed to be the same as the variance of the $\mathcal{H}_0$ hypothesis. Then, in this section $\sigma = \sigma_0 = \sigma_1$.

6.3.1 *Means of the faults hypotheses*

It is straightforward that μ_0 represents the mean of the nominal mode distribution, which is equal to 0. The mean of the fault mode distribution μ_1, is more complex to compute. Indeed, the fault distribution is not the same depending on which mode a particle is. For example, if a particle is in mode $m^{(0)}$ then the alternate mode $m^{(1)}$ has a mean μ_1 given by (4.14) for an actuator fault or by (4.16) for a sensor fault, which corresponds to the point where the particle will be placed if it jumps to $m^{(1)}$. By doing this, what is evaluated is the false alarm between the current nominal mode distribution and the potential fault mode distribution where the sentinel particle will be sent if it switches to mode $m^{(1)}$. However, if a particle is in mode $m^{(1)}$, then the mean μ_1 is given by the current estimate of the fault by the particle in the faulty mode. Then, what is evaluated in this case, is the missed detection probability between the current distribution of the particles in mode $m^{(1)}$, and the nominal distribution where the particle will be sent if it switches to mode $m^{(0)}$. In this situation, three means must be considered. First, the mean μ_0 that correspond to the mean associated with the hypothesis $\mathcal{H}_0$ that is the same for the computation of the false alarm and missed detection probabilities. And then, the means $\mu_{1|0}$ and $\mu_{1|1}$ that correspond respectively to the hypothesis $\mathcal{H}_1$ of the false alarm probability and the hypothesis $\mathcal{H}_1$ of the missed detection probability. These different means are given by:

$$\mu_{0_k} = 0 \tag{6.59}$$

and

$$\mu_{1_k} = \begin{cases} \mu_{1|0_k} \text{for the transition to } m^{(1)} \\ \mu_{1|1_k} \text{for the transition to } m^{(0)} \end{cases} \tag{6.60}$$

where $\mu_{1|0_k}$ and $\mu_{1|1_k}$ are given by:

$$\mu_{1|0_k} = \sum_{i \in I_{m^{(0)}}} w_{0_{k-1}}^i \Delta_{\mathrm{f}_k}^i \tag{6.61a}$$

$$\mu_{1|1_k} = \sum_{i \in I_{m^{(1)}}} w_{1_{k-1}}^i \mathrm{f}_{k|k-1}^i \tag{6.61b}$$

where $I_{m^{(0)}}$ and $I_{m^{(1)}}$ are the ensembles of index of particles of state $\mathrm{f}_{k|k-1}$ that are respectively in mode $m^{(0)}$ or in mode $m^{(1)}$. The weights ${w0}^i_{k-1}$ and ${w1}^i_{k-1}$ are given by:

$$ {w0}^i_{k-1} = \frac{w^i_{k-1}}{\sum\limits_{j \in I_{m^{(0)}}} w^j_{k-1}}, \quad \forall i \in I_{m^{(0)}} \tag{6.62a} $$

$$ {w1}^i_{k-1} = \frac{w^i_{k-1}}{\sum\limits_{j \in I_{m^{(1)}}} w^j_{k-1}}, \quad \forall i \in I_{m^{(1)}} \tag{6.62b} $$

The computation of the means $\mu_{1|0_k}$ and $\mu_{1|1_k}$ are described in Algorithm 6.1.

Algorithm 6.1 Computation of the mean $\mu_{1|0_k}$ and $\mu_{1|1_k}$ for the prediction step of the adaptive jump-Markov regularized particle filter

Function MEANS$\mathcal{H}_1(\mu_{1|0_k}, \mu_{1|1_k}, \mathrm{f}^{1:N}_{k|k-1}, \Delta_{\mathrm{f}}{}^{1:N}_k, w^{1:N}_{k-1}, \mathrm{m}^{1:N}_k)$

- **for each** $i \in [1, N]$ **do**
  - **if** $\mathrm{m}^i_k = m^{(0)}$ **then**
    - ${w0}^i_{k-1} \leftarrow \frac{w^i_{k-1}}{\sum\limits_{j \in I_{m^{(0)}}} w^j_{k-1}}$ //See (6.62a)
    - ${w1}^i_{k-1} \leftarrow 0$
  - **if** $\mathrm{m}^i_k = m^{(1)}$ **then**
    - ${w1}^i_{k-1} \leftarrow \frac{w^i_{k-1}}{\sum\limits_{j \in I_{m^{(1)}}} w^j_{k-1}}$ //See (6.62b)
    - ${w0}^i_{k-1} \leftarrow 0$
- $\mu_{1|0_k} \leftarrow \sum\limits_{i \in I_{m^{(0)}}} {w0}^i_{k-1} \Delta_{\mathrm{f}}{}^i_k$ //See (6.61a)
- $\mu_{1|1_k} \leftarrow \sum\limits_{i \in I_{m^{(1)}}} {w1}^i_{k-1} \mathrm{f}^i_{k|k-1}$ //See (6.61b)

6.3.2 *Transition probability matrix update*

To compute the transition probability matrix, the false alarm and missed detection probabilities using the means of the $\mathcal{H}_1$ hypotheses given by (6.61a) and (6.61b) must be computed first. This is performed by using the analytical expression of the false alarm and missed detection probability described in Section 6.2.4 and 6.2.5. The function PROBABILITIES aims to compute the false alarm and missed detection probability used to update the transition probability matrix, it is detail in Algorithm 6.2, where $\mu_{0_{1:n}}$, $\mu_{1_{1:n}}$, and $\sigma_{1:n}$ are static variables.

In Algorithm 6.2, the function:

Algorithm 6.2 Detail of the function PROBABILITIES for the update of the transition probability matrix

Function PROBABILITIES($P_{fa}{}^{opt}$, $P_{md}{}^{opt}$, μ_1, σ, n)

- **if** $n \geq k$ **then**
  - $n \leftarrow k$
- $\mu_{0_n} \leftarrow 0$ //See (6.59)
- $\mu_{1_n} \leftarrow \mu_1$
- $\sigma_n \leftarrow \sigma$
- POLYNOMIAL($a_{1:n}$, $b_{1:n}$, $c_{1:n}$, $\mu_{01:n}$, $\sigma_{1:n}$, $\mu_{11:n}$, n)
- SAMPLING($\alpha^{1:n_\alpha}$, $\mu_{01:n}$, $\mu_{11:n}$, $a_{1:n}$, $b_{1:n}$, $c_{1:n}$) //See Algorithm 6.4
- **for each** $j \in \{0, 1\}$ **do**
  - **for each** $i \in [1, n_\alpha]$ **do**
    - **if** $a_{1:n} = 0$ **then**
      - $t_{init} \leftarrow 0$
    - **else**
      - $t_{init} \leftarrow \min\left\{\frac{1}{2a_{1:n}\sigma_{1:n}^2}\right\} + \varepsilon$ //ε is a small number to ensure (6.50) and (6.56)
    - NEWTONRAPHSON(T_j, $\bar{\mathcal{K}}'_j(t) - \alpha^i$, $\bar{\mathcal{K}}''_j(t)$, t_{init}) //With $\bar{\mathcal{K}}'_j(t)$ and $\bar{\mathcal{K}}''_j(t)$ updated
    - $t_j \leftarrow T_j\sqrt{n\bar{\mathcal{K}}''_j(T_j)}$
    - $y_j \leftarrow \text{sign}(t_j)\sqrt{2n\left(T_j\alpha^i - \bar{\mathcal{K}}_j(T_j)\right)}$
    - $P_j^i \leftarrow \Phi(y_j) - \phi(y_j)\left(\frac{1}{t_j} - \frac{1}{y_j}\right)$
- ROC($P_{fa}{}^{opt}$, $P_{md}{}^{opt}$, $P_0^{1:n_\alpha}$, $1 - P_1^{1:n_\alpha}$)
- $\mu_{0_{n-1}} \leftarrow \mu_{0_n}$ //Saving the value of μ_0
- $\mu_{1_{n-1}} \leftarrow \mu_{1_n}$ //Saving the value of μ_1
- $\sigma_{n-1} \leftarrow \sigma_n$ //Saving the value of σ

- POLYNOMIAL aims to compute the last n coefficients a, b and c using (6.46);
- NEWTONRAPHSON is the function that perform the Newton-Raphson method [83] to approximate the root of a function $f(t)$, where the first parameter is the output, the second is the function, the third is the function derivative and finally the fourth is the initial guess;
- SAMPLING is used to provide n_α different thresholds α, it is described in Algorithm 6.4;
- ROC aims to obtain the optimal false alarm and missed detection probabilities based on an optimization criterion, it is described in Algorithm 6.5.

Using the false alarm probability and the missed detection probability provided by Algorithm 6.2 the transition probability matrix can be updated. The update of the transition probability matrix is performed by the UPDATEΠ function. This function is described in Algorithm 6.3.

Algorithm 6.3 Detail of the function UPDATEΠ of the adaptive jump-Markov regularized particle filter

Function UPDATEΠ($\mathbf{\Pi}_k$, $\mu_{1|0_k}$, $\mu_{1|1_k}$, σ_k)
- PROBABILITIES($P_{fa}{}^{opt}$, $\emptyset$, $\mu_{1|0_k}$, σ_k) //See Algorithm 6.2
- PROBABILITIES($\emptyset$, $P_{md}{}^{opt}$, $\mu_{1|1_k}$, σ_k)
- $\pi_{00k} \leftarrow \phi_{fa}\left(P_{fa}{}^{opt}\right)$ //See (6.1)
- $\pi_{10k} \leftarrow 1 - \phi_{fa}\left(P_{fa}{}^{opt}\right)$
- $\pi_{11k} \leftarrow \phi_{md}\left(P_{md}{}^{opt}\right)$
- $\pi_{01k} \leftarrow 1 - \phi_{md}\left(P_{md}{}^{opt}\right)$

6.3.2.1 *Threshold sampling*

To obtain the best trade-off between the false alarm and missed detection probabilities, multiple values must be tested and therefore multiple values of threshold α must be defined. To do so, a sample of n_α threshold values is computed over an interval. From (6.43) and (6.45), $\bar{X}$ can be rewritten as:

$$\bar{X}\left(\tilde{y}_{1:n}\right) = \frac{1}{n}\sum_{k=1}^{n}\left(a_k\tilde{y}_k^2 + b_k\tilde{y}_k + c_k\right) \tag{6.63}$$

It is clear that a trade-off between the false alarm and the missed detection probability is obtained for $n = 1$ with $\Gamma \in [\mu_0, \mu_1]$, and this implies $P_{fa} \in [0, 0.5]$ and $P_{md} \in [0, 0.5]$.

Then the interval $[\mu_0, \mu_1]$ is a conservative interval of the solution of an optimal trade-off. Transposed to the state space of the threshold α, since (6.63) is a monotonically increasing function $\forall\tilde{y}_{1:n} \geq 0$, and since $\mu_0 = 0$ and $\mu_0 \leq \mu_1$, it gives:

$$\left[\bar{X}\left(\mu_{01:n}\right), \bar{X}\left(\mu_{11:n}\right)\right] = \left[\alpha_{\min}, \alpha_{\max}\right] \tag{6.64}$$

Equation (6.64) is then used to determine an interval where the threshold α will be allowed to vary. Then for a given number of samples n_α the sample step is given by:

$$\Delta\alpha = \frac{\left(\alpha_{\max} - \alpha_{\min}\right)}{n_\alpha} \tag{6.65}$$

And then the samples are given by:

$$\alpha^{1:n_\alpha} = \left[\alpha_{\min} : \Delta\alpha : \alpha_{\max}\right] \tag{6.66}$$

The sampling of the threshold α is described in Algorithm 6.4.

However, having multiple α will provide multiple P_{fa} and P_{md}, an optimal one must be selected according to a criterion to be able to compute the transition probability matrix.

Algorithm 6.4 Detail of the function SAMPLING used in Algorithm 6.2

Function SAMPLING($\alpha^{1:n_\alpha}$, $\mu_{01:n}$, $\mu_{11:n}$, $a_{1:n}$, $b_{1:n}$, $c_{1:n}$)

$\alpha_{\min} \leftarrow \frac{1}{n}\sum_{k=1}^{n}\left(a_k {\mu_{0k}}^2 + b_k\mu_{0k} + c_k\right)$ //See (6.63)

$\alpha_{\max} \leftarrow \frac{1}{n}\sum_{k=1}^{n}\left(a_k {\mu_{1k}}^2 + b_k\mu_{1k} + c_k\right)$

$\Delta\alpha \leftarrow \frac{(\alpha_{\max}-\alpha_{\min})}{n_\alpha}$ //See (6.64)

$\alpha^1 \leftarrow \alpha_{\min}$

for each $j \in [2, n_\alpha]$ **do**

  $\alpha^j \leftarrow \alpha^{j-1} + \Delta\alpha$

6.3.2.2 *Optimization Criterion*

The false alarm and missed detection probabilities given by (2.8) and (2.9) depend on a threshold α. This threshold is used to achieve a trade-off between the P_{fa} and the P_{md}. The trade-off is optimal given a criterion. Multiple criterion exist in the literature, the one used here is the ROC curve described in Section 2.4.2. The criterion selected for the formulation of the method is the optimal point given by (2.15). The false alarm and missed detection probabilities that solve (2.15) are denoted ${P_{fa}}^{opt}$ and ${P_{md}}^{opt}$. Then the threshold associated with this couple is denoted α^{opt}. Note that a weighted cost function could also have been used, but in this formulation, false alarm and missed detection probability are given equal weights.

The computation ROC curve criterion is described in Algorithm 6.5.

Algorithm 6.5 Detail of the function ROC used in Algorithm 6.2

Function ROC(${P_{fa}}^{opt}$, ${P_{md}}^{opt}$, ${P_{fa}}^{1:n_\alpha}$, ${P_{md}}^{1:n_\alpha}$)

$min_{idx} \leftarrow 1$ //Initialization

$min_{roc} \leftarrow \infty$

for each $i \in [1, n_\alpha]$ **do** //Find the min of (2.15)

  if $\sqrt{\left({P_{md}}^i\right)^2 + \left({P_{fa}}^i\right)^2} < min_{roc}$ **then**

    $min_{roc} \leftarrow \sqrt{\left({P_{md}}^i\right)^2 + \left({P_{fa}}^i\right)^2}$

    $min_{idx} \leftarrow i$

${P_{fa}}^{opt} \leftarrow {P_{fa}}^{min_{idx}}$

${P_{md}}^{opt} \leftarrow {P_{md}}^{min_{idx}}$

The prediction step is described in Algorithm 6.6.

The other step of the AJMRPF can be the same of the JMRPF from Chapter 4 or the RJMRPF from Chapter 5 depending on the needs.

Algorithm 6.6 Prediction step of the adaptive jump-Markov regularized particle filter

Function ADAPTIVEPREDICT($\mathbf{x}_{k|k-1}^{1:N}$, k, $w_{k-1}^{1:N}$, $\mathbf{x}_{k-1}^{1:N}$, $\mathbf{m}_k^{1:N}$, $\mathbf{u}_k$, $\mathbf{y}_k$, $\mathbf{R}_k$)

  for each $i \in [1, N]$ **do**

    $\boldsymbol{\eta}_k^i \sim \mathcal{N}(0, \mathbf{Q}_k)$

    $\mathbf{x}_{k|k-1}^i \leftarrow f_k\left(\mathbf{x}_{k-1}^i, \mathbf{u}_k\right) + \boldsymbol{\eta}_k^i$ //See (4.10)

    $\Delta_{\mathbf{f_s}}{}_k^i \leftarrow \mathbf{y}_k - h_k\left(\mathbf{x}_{k|k-1}^i\right)$ //See (4.16)

  for each $j \in [1, n_{f_s}]$ **do**

    ${s_k}^j \leftarrow \sqrt{\mathbf{R}_k^{j,j}}$

    MEANS$\mathcal{H}_1(\mu_{s10k}^j, \mu_{s11k}^j, \mathbf{f_s}_{k|k-1}^{1:N,j}, \Delta_{\mathbf{f_s}}{}_k^{1:N,j}, w_{k-1}^{1:N}, \mathbf{m_s}_k^{1:N,j})$ //See Algorithm 6.1

    UPDATE$\Pi(\mathbf{\Pi}_k^j, \mu_{s10k}^j, \mu_{s11k}^j, {\sigma_s}_k^j)$

    for each $i \in [1, N]$ **do**

      ADAPTIVEJUMP($\mathbf{f_s}_{k|k-1}^{i,j}$, $\mathbf{m_s}_k^{i,j}$, $\Delta_{\mathbf{f_s}}{}_k^{i,j}$, $\mathbf{\Pi}_k^{n_{fa}+j}$)

6.4 COMPARATIVE NUMERICAL SIMULATION ANALYSIS

This section aims to provide a demonstration of the AJMRPF in the presence of sensor faults with abrupt and incipient faults on a fixed-wing UAV. For the sake of brevity, only the longitudinal system is considered.

Since only the addition of the adaptive probability matrix is assessed in this section, the AJMRPF is compared to a JMRPF. Note that a ARJMRPF could have been compared to a RJMRPF, the results are less significant, and the improvements brought by the adaptive transition probability matrix are more blurred. The use of the ARJMRPF is however relevant since the AJMRPF and the RJMRPF do not solve the same issues and using the ARJMRPF solve the same issues of the AJMRPF and the RJMRPF without introducing new ones.

Given that all algorithms under comparison are particle filters, a non-linear system is used. The UAV has an initial longitudinal velocity of $40\,\mathrm{m\,s^{-1}}$, an initial altitude of 500 m, an initial null flight path angle and is initiated in a straight cruise flight condition. The control and guidance are then performed as described in Section 3.7 and Section 3.8 for the longitudinal system around the trim point of the initial flight condition. The desired altitude is set to 500 m and the desired velocity to $40\,\mathrm{m\,s^{-1}}$.

To illustrate an abrupt sensor fault on the pitch measurement, a fault signal is added to the measurement equation. This true fault model is unknown to the filter. The nonlinear longitudinal model used to compute the true state is given by:

$$\begin{cases} \dot{\mathbf{z}} = \mathcal{F}(\mathbf{z}, \mathbf{u}) & \text{(6.67a)} \\ \mathbf{y} = \mathcal{H}(\mathbf{z}) + \begin{bmatrix} 0 \\ 0 \\ 0 \\ f_\theta \\ 0 \end{bmatrix} + \boldsymbol{\nu} & \text{(6.67b)} \end{cases}$$

where f_θ denotes the pitch measurement fault and where the measurement noise $\boldsymbol{\nu}$ is a zero mean Gaussian noise given by:

$$\begin{bmatrix} \nu_{baro,-p_d} \\ \nu_{accel,u} \\ \nu_{accel,w} \\ \nu_{gyro,\theta} \\ \nu_{gyro,q} \end{bmatrix} \tag{6.68}$$

with standard deviations respectively given by $\sigma_{baro,-p_d} = 1\,\text{m}$, $\sigma_{accel,u} = 1\,\text{m}\,\text{s}^{-1}$, $\sigma_{accel,w} = 1\,\text{m}\,\text{s}^{-1}$, $\sigma_{gyro,\theta} = 0.01\,\text{rad}$ and $\sigma_{gyro,q} = 0.002\,\text{rad}\,\text{s}^{-1}$. The state vector $\mathbf{z}$ is given by:

$$\mathbf{z} = \begin{bmatrix} p_d \\ u \\ w \\ \theta \\ q \end{bmatrix}, \tag{6.69}$$

the input vector $\mathbf{u}$ by:

$$\mathbf{u} = \begin{bmatrix} \delta_e \\ \delta_t \end{bmatrix} \tag{6.70}$$

the non-linear function $\mathcal{F}(\cdot)$ by (3.8) and the observation function $\mathcal{H}(\cdot)$ is given by:

$$\mathcal{H}(\mathbf{z}) = \begin{bmatrix} -1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \mathbf{z} \tag{6.71}$$

Since the fault is only on the measurements, the evolution and control input matrices for the process models are the same as the ones given by (6.67) for the state vector $\mathbf{z}$. The extended vector $\mathbf{x}$ is given by:

$$\mathbf{x} = \begin{bmatrix} \mathbf{z} \\ f_\theta \end{bmatrix}. \tag{6.72}$$

The fault is assumed to have zero order dynamics. The process noise is a white Gaussian noise with a standard deviation $\sigma_\mathbf{x}$ given by:

$$\sigma_\mathbf{x} = \begin{bmatrix} 1\,\mathrm{m} \\ 0.1\,\mathrm{m\,s^{-1}} \\ 0.1\,\mathrm{m\,s^{-1}} \\ 0.02\,\mathrm{rad} \\ 0.002\,\mathrm{rad\,s^{-1}} \\ 0.002\,\mathrm{rad} \end{bmatrix} \tag{6.73}$$

The measurement noise of the process model is 1.5 times the measurements noise of (6.67);

The transition probability matrix of the JMRPF is given by:

$$\mathbf{\Pi} = \begin{bmatrix} 0.99 & 0.01 \\ 0.01 & 0.99 \end{bmatrix} \tag{6.74}$$

The standard deviation used to compute the initial covariance matrix $\mathbf{P}_0$ for all filters, denoted $\sigma_{\mathbf{x0}}$ is given by:

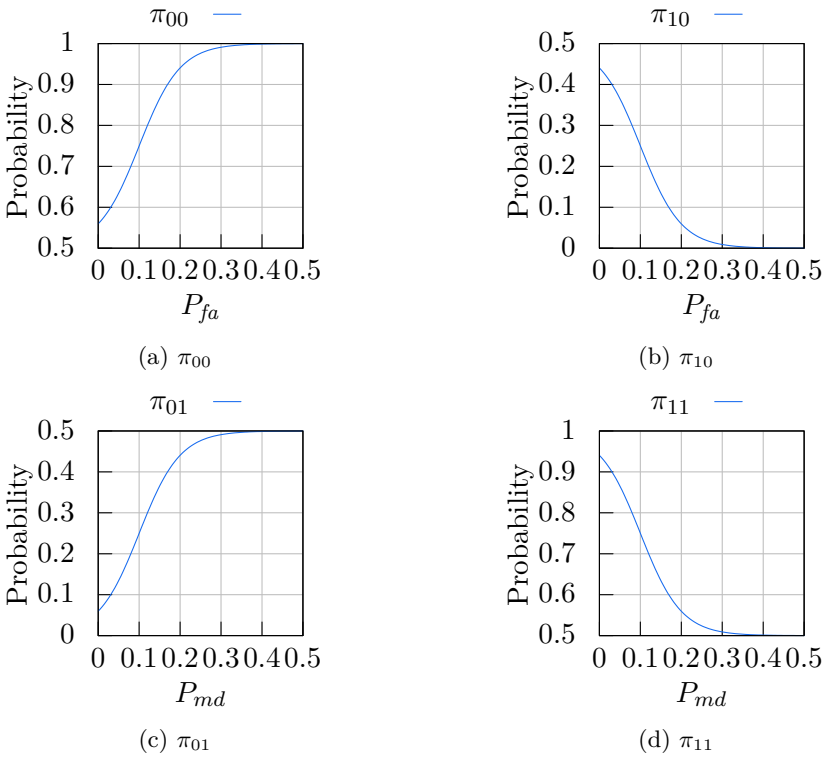
$$\sigma_{\mathbf{x0}} = \begin{bmatrix} 1\,\mathrm{m} \\ 1\,\mathrm{m\,s^{-1}} \\ 1\,\mathrm{m\,s^{-1}} \\ 0.005\,\mathrm{rad} \\ 0.002\,\mathrm{rad\,s^{-1}} \\ 0.005\,\mathrm{rad} \end{bmatrix} \tag{6.75}$$

The function $\phi_{fa}(\cdot)$ and $\phi_{md}(\cdot)$ for the transition probability matrix of the AJMRPF are respectively given by:

$$\phi_{fa}(x) = \frac{0.5}{e^{20(0.1-x)}+1} + 0.5, \tag{6.76a}$$

$$\phi_{md}(x) = 1 - \frac{0.5}{e^{20(0.1-x)}+1}. \tag{6.76b}$$

And the value of n for the saddlepoint approximation is set to 3.



(a) π_{00} (b) π_{10}

(c) π_{01} (d) π_{11}

Figure 6.3: Transition probabilities functions used by the AJMRPF

The transition probability matrix and the false alarm and missed detection are shown in Figure 6.3.

The fault scenario considered lasts 50 s. At 10 s, a first fault occurs on the pitch measurement. It is an abrupt fault with an amplitude of 10°. This fault is deactivated at 20 s. At 30 s, a second fault occurs on the pitch measurement. It is an incipient fault with linear dynamics that lasts 10 s and reaches an amplitude of 10°. The scenario is illustrated in Figure 6.4.

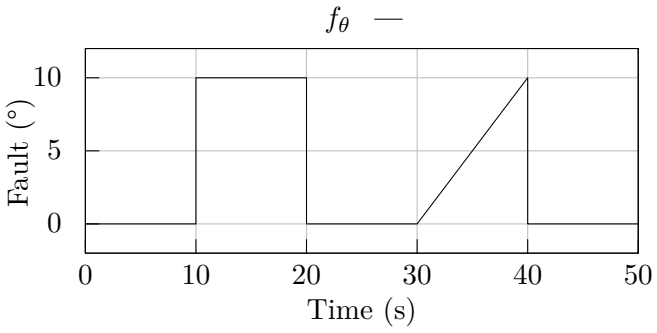


Figure 6.4: Fault scenario on the pitch measurement used to compare the AJMRPF to the JMRPF.

Finally, the number of particles is set to 1000 and the resampling threshold Γ_{rspl} is set to 0.5 and the bandwidth factor h of the Epanechnikov kernel is set 0.2817. The number of simulations performed is 100 with a time step of 0.05 s.

The first results illustrated in Figure 6.5 are the estimates of the AJMRPF and the JMRPF by taking the median results from 100 simulations. The selection of the median result is detailed in Appendix D.

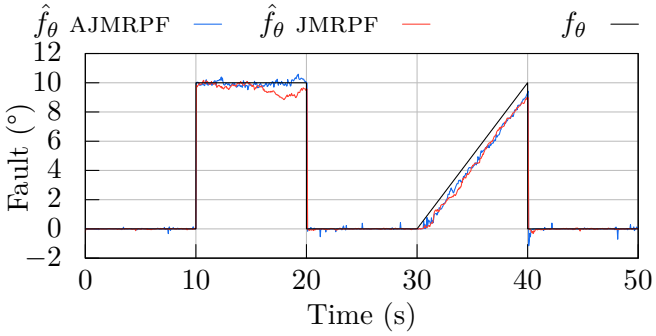


Figure 6.5: Median result of the fault states of the UAV under additive sensor fault estimated by a AJMRPF and a JMRPF. Median results based on 100 simulations.

In Figure 6.5, during the first 10 s while there is no fault both JMRPF and the RJMRPF estimate the fault free mode accurately as already expected and observed from previous simulation of the JMRPF. When the first abrupt fault occurs at 10 s, both the JMRPF and the RJMRPF estimate the fault comparably in terms of convergence time and accuracy. The AJMRPF further improves the already good result of the JMRPF for this kind of faults. The deactivation of the fault at 20 s is also efficiently handled by both filters, and the situation of the next 10 s is the same as the first 10. Similar results for both filters is also observed on the incipient fault on this simulation. The deactivation of the fault at 40 second is again efficiently handled by both filters, and the situation of the next 10 s is the same as the first 10 s.

The false alarm and missed detection probabilities used by the AJMRPF, are illustrated in Figure 6.6.

In Figure 6.6, the false alarm and missed detection probabilities are consistent with the actual fault mode. Indeed, when no fault is active, the P_{fa} and P_{md} are close to 0.5 with a P_{fa} around 0.3 and a P_{md} around 0.4. When a fault is active, both P_{fa} and P_{md} converge to 0. For the abrupt fault, this takes in one time step, and for the incipient fault it reaches zero in about 1.9 s.

The Figure 6.7, presents the components of the transition probability matrix, which depend on the P_{fa} and P_{md}.

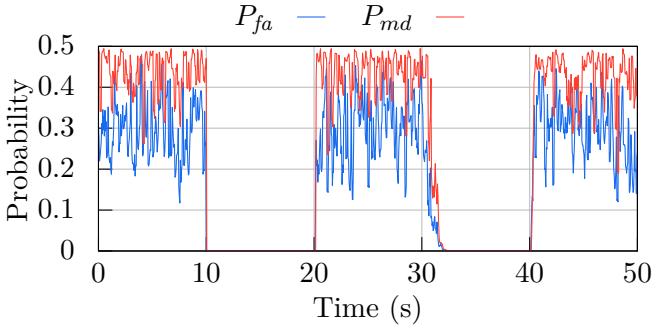


Figure 6.6: Median result of the false alarm and missed detection probabilities of the AJMRPF

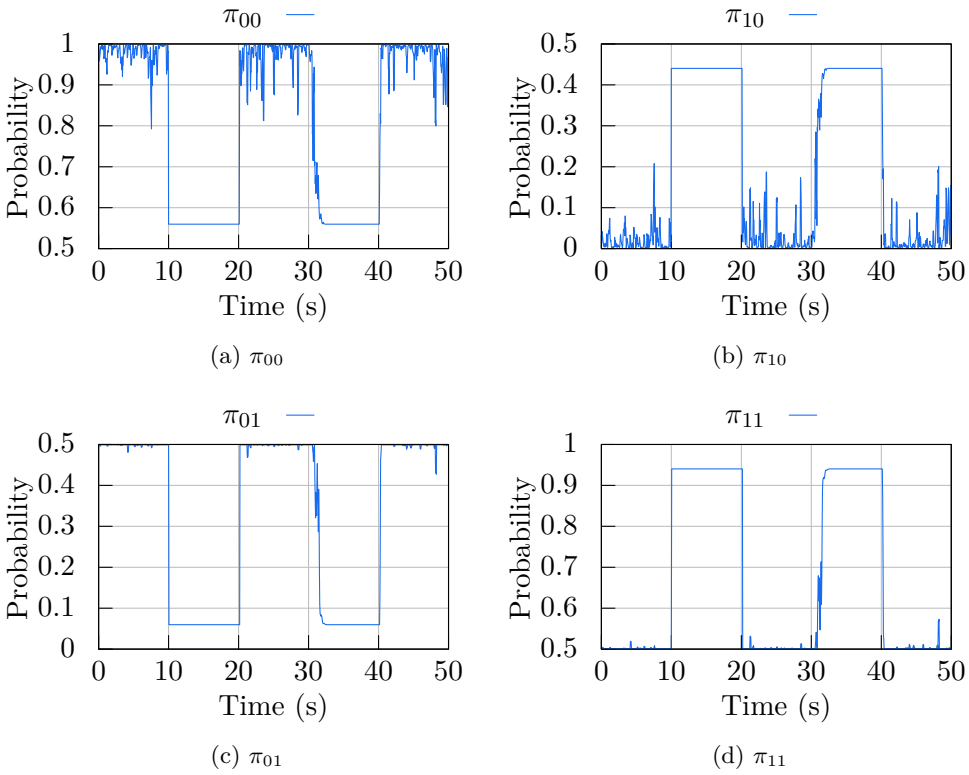
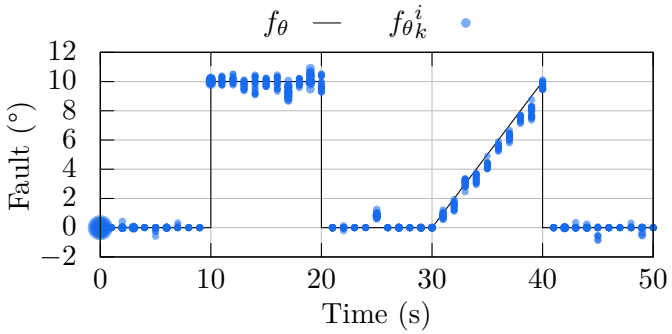


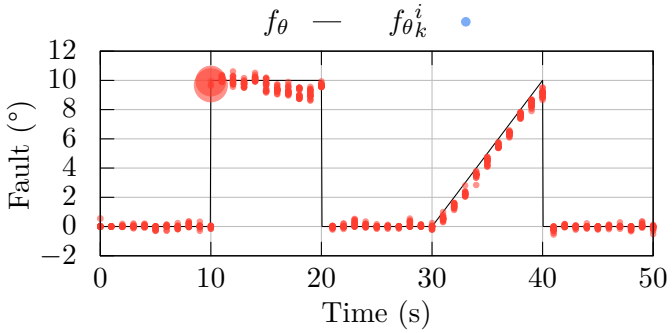
Figure 6.7: Median result of the transition probabilities of the AJMRPF

As expected from Figure 6.6, the Figure 6.7 shows that the evolution of the transition probability matrix is consistent with the missed detection and false alarm rates.

The behaviour of the median results of the AJMRPF and the JMRPF is illustrated in detail in Figure 6.8, by showing the particles positions with their associated weights.



(a) Estimation of the fault state with a AJMRPF.

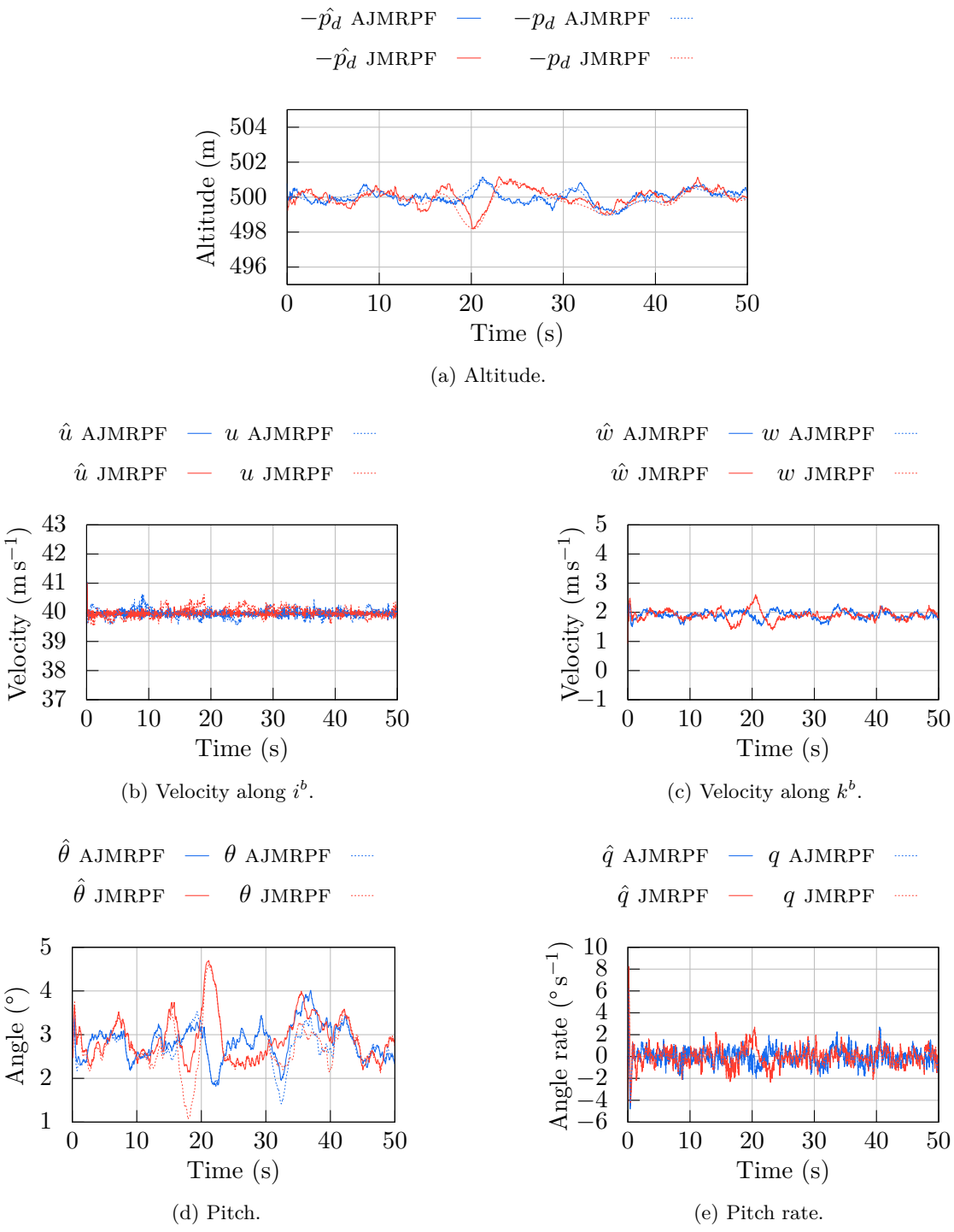


(b) Estimation of the fault state with a JMRPF.

Figure 6.8: The 20 most weighted particles of every second of the median result of the fault states of the UAV under additive sensor fault estimated by a AJMRPF and a JMRPF. Median results based on 100 simulations.

In Figure 6.8, the major difference between the AJMRPF and the JMRPF is the weight of the sentinel particles at the activation of the abrupt fault. Indeed, the JMRPF has high weights on its sentinel particles while the weights of the sentinel particles of the AJMRPF are not significantly different from other particles. This, is due to the fact that a lot more particles are used by the AJMRPF when the abrupt fault occurs since the transition probability matrix is almost at 0.5 for π_{10} at this time.

The median results of the state vector $\mathbf{z}$ estimated by the AJMRPF and the JMRPF are shown in Figure 6.9.



(a) Altitude.

(b) Velocity along i^b.

(c) Velocity along k^b.

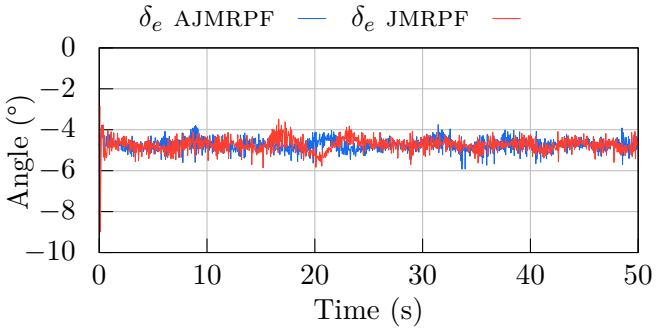
(d) Pitch.

(e) Pitch rate.

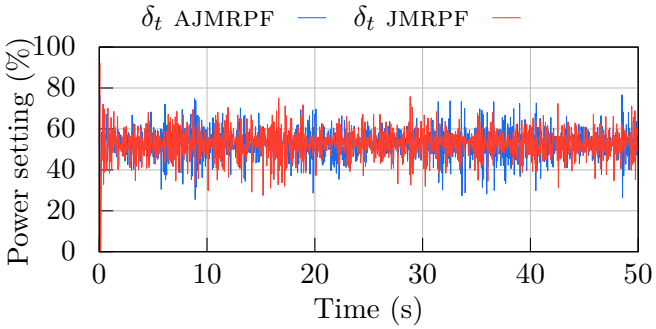
Figure 6.9: Median result of the longitudinal states of the UAV under additive sensor fault estimated by a AJMRPF and a JMRPF. Median results based on 100 simulations.

In Figure 6.9 low state estimation errors are obtained with the AJMRPF or the JMRPF are visible, the UAV remains in a straight level flight condition as required.

The median results for the control inputs of the AJMRPF and JMRPF are shown in Figure 6.10.



(a) Elevator deflection.



(b) Throttle.

Figure 6.10: Inputs of the median result of the UAV under additive sensor fault estimated by a AJMRPF, and a JMRPF. Median results based on 100 simulations.

In Figure 6.10, the control inputs are consistent with the trajectories obtained using both filters.

The RMSE of the fault state f_θ for the AJMRPF and the JMRPF estimates are shown in Figure 6.11.

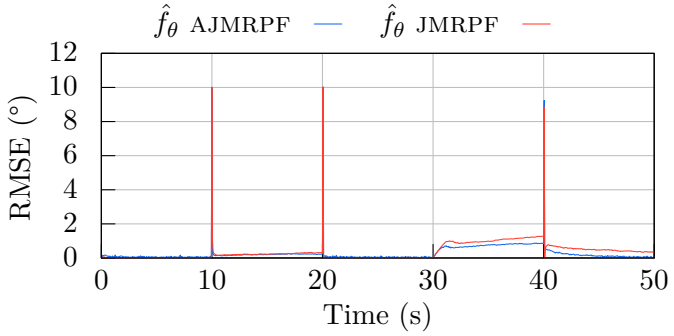
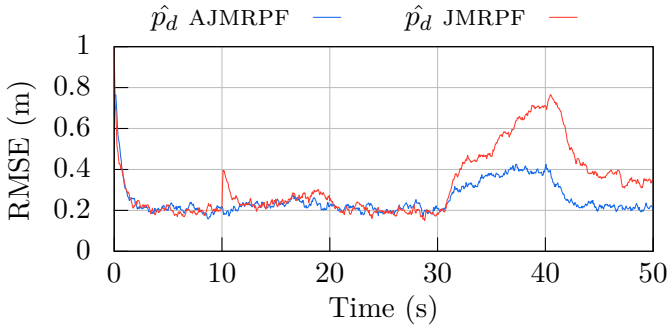


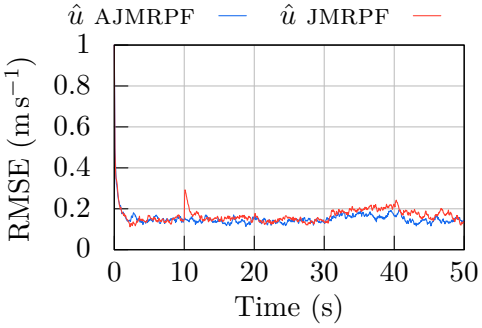
Figure 6.11: RMSE of the fault states of the UAV under additive sensor fault estimated by a AJMRPF, and a JMRPF. Results are based on 100 simulations.

In Figure 6.11 the RMSE of the AJMRPF for the incipient fault is lower than RMSE of the JMRPF. This is confirmed by the mean RMSE $\overline{\text{RMSE}}$ of f_θ equal to 0.27° for the AJMRPF and 0.38° for the JMRPF. This shows that the AJMRPF does not only adequately reset the transition probability matrix, but also improves the results compared to the JMRPF.

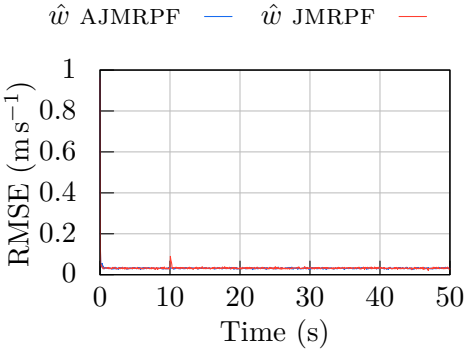
The RMSE of the states are shown in Figure 6.12 and the AJMRPF outperforms the JMRPF in terms of state estimation accuracy



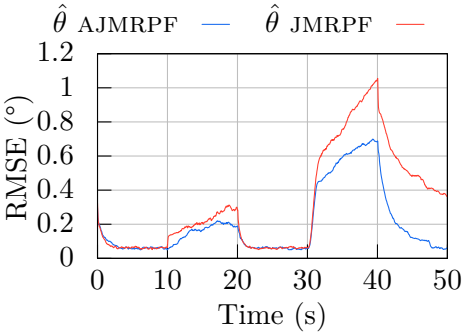
(a) RMSE of the altitude estimate.



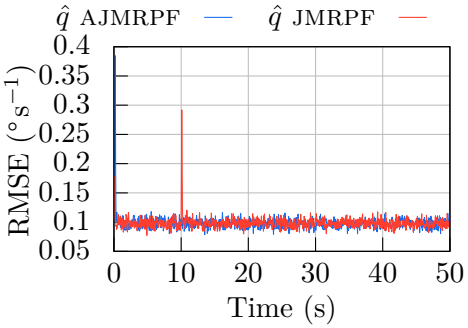
(b) RMSE of the velocity along i^b estimate.



(c) RMSE of the velocity along k^b estimate.



(d) RMSE of the pitch estimate.



(e) RMSE of the pitch rate estimate.

Figure 6.12: RMSE of the longitudinal states of the UAV under additive sensor fault estimated by a AJMRPF, and a JMRPF. Results are based on 100 simulations.

Table 6.1 gives the RMSE values at key time steps and the $\overline{\text{RMSE}}$ which is defined in Appendix D.

State	Time (s)				$\overline{\text{RMSE}}$
	10.05 s	20.05 s	30.05 s	40.05 s	
AJMRPF					
RMSE p_d (m)	0.216	0.225	0.208	0.425	0.254
RMSE u ($\mathrm{m\,s^{-1}}$)	0.155	0.156	0.128	0.163	0.152
RMSE w ($\mathrm{m\,s^{-1}}$)	0.029	0.036	0.028	0.030	0.033
RMSE θ (°)	0.053	0.190	0.061	0.687	0.205
RMSE q ($^\circ\,\mathrm{s^{-1}}$)	0.096	0.100	0.086	0.094	0.098
RMSE f_θ (°)	0.277	10.017	0.067	9.235	0.268
JMRPF					
RMSE p_d (m)	0.333	0.258	0.180	0.708	0.328
RMSE u ($\mathrm{m\,s^{-1}}$)	0.271	0.166	0.132	0.217	0.167
RMSE w ($\mathrm{m\,s^{-1}}$)	0.090	0.033	0.036	0.032	0.033
RMSE θ (°)	0.102	0.284	0.061	1.055	0.325
RMSE q ($^\circ\,\mathrm{s^{-1}}$)	0.192	0.092	0.102	0.106	0.099
RMSE f_θ (°)	1.049	10.037	0.058	8.848	0.381

Table 6.1: RMSE values of the AJMRPF and the JMRPF estimates at key time steps, and $\overline{\text{RMSE}}$.

6.5 CHAPTER SUMMARY

This chapter introduced a new way to update the transition probability matrix in real time, using estimated values of the false alarm and missed detection probabilities. This matrix was used in the previously introduced JMRPF and RJMRPF filters in Chapters 4 and 5, respectively. Using this new transition probability matrix configuration, a new filter named the AJMRPF or the ARJMRPF depending on the benchmark filter to be adapted, was shown to achieve sound estimation performance results in Section 6.4.

The idea behind the adaptive transition probability matrix was to update it using the false alarm and missed detection probabilities, which were computed using the saddlepoint approximation. This approximation allows one to consider multiple previous measurements instead of only relying on the current measurement, which leads to improved estimation accuracy. The detail of this computation and the formulation of the AJMRPF / ARJMRPF were given in Section 6.2 and Section 6.3 respectively. 100 simulations were performed in Section 6.4 have shown sound state and fault estimation results with the AJMRPF by exploiting the ability to adapt the transition probability matrix to the faulty or fault free modes. Similar and even higher performances were shown for the AJMRPF compared to the

JMRPF depending on the type of faults considered. The proposed approach also allows for real time knowledge of the false alarm and missed detection probabilities (research question 6). Morever, this method is easy to implement and requires a little of computation demand (research question 7).

7

CONCLUSIONS AND FUTURE WORK

The estimation of faults is an essential feature for fixed-wing UAV with no hardware redundancies, and required to be fault-tolerant. This thesis entitled "*Multimode navigation for degraded fixed wing unmanned aerial vehicle operation under sensor and actuator faults*" has investigated new methods for the estimation of actuator and sensor faults. Using a realistic representation of the fixed-wing UAV and its sensors, challenging fault scenarios including ambiguities and lack of knowledge of the fault models have been studied. It has been proven that estimation methods for such problems are required to handle multimodalities and the mismatch between true and assumes fault dynamics. A new method based on a RPF and a JMS called a JMRPF was introduced in Chapter 4. This method was designed and shown to successfully detect and estimate ambiguous sensor faults as well as faults with unknown amplitude. An enhanced version of it was then introduced and developed to deal with unknown fault dynamics and ambiguous actuator and sensor faults in Chapter 5 and named RJMRPF. Finally, the JMRPF — and the RJMRPF — was further enhanced by updating the transition probably matrix of the JMS in real-time. This last modification of the JMRPF named AJMRPF was introduced in Chapter 6. All these methods were evaluated on longitudinal linear and non-linear models of a fixed-wing UAV, with different fault scenarios and compared to different estimation algorithms. Significant improvement have been shown with the new methods introduced for the estimation of both sensor and actuator faults and the staes of the UAV. The accurate and fast estimation performed by the JMRPF and its enhanced versions have improved flight mission safety in the scenario considered, compared to other methods by maintaining the UAV on a trajectory closer to the one desired. In terms of impact, the proposed estimation algorithms have been shown to improve fault tolerance to a wise rang eof conditions from single faults to an ambiguous faults and unknown fault dynamics, with the ability to monitor false alarm and missed detection rates. This is in line with increasing requirements in mission integrity for increasingly autonomous UAV.

7.1 RESEARCH QUESTIONS

The research questions answered in this thesis are the following:

1. Does a JMS with a RPF increase accuracy and speed up convergence when abrupt additive faults occur, compared to a stand-alone RPF?

The use of a JMS with a RPF as presented in Chapter 4 was shown to be more efficient — or a least as efficient depending on the fault and filter parameters — as a stand-alone RPF in terms of accuracy, robustness and convergence time. This has been shown in Section 5.5. The new way to implement a JMS in a particle filter introduced in this thesis, provides better fault and state estimation results than a IMM-KF in the linear Gaussian case when abrupt additive faults are under consideration. This was shown in Section 4.5.

2. Is it possible to distinguish and estimate ambiguous sensor faults using only a JMS as process model of a particle filter?

   In Chapter 4, the ambiguous sensor fault case was investigated, and it has been shown that the JMRPF is able to accurately and rapidly estimate and distinguish between ambiguous sensor faults.

3. Can an abrupt additive fault, with a large amplitude with respect to the process noise, be accurately estimated in a short time period?

   The JMRPF has shown an outstanding ability to estimate abrupt fault without the need to artificially inflate the process noise. This was particularly highlighted in Section 5.5, where the JMRPF was successfully used to estimate abrupt faults of multiple magnitudes even with a small process noise for such abrupt changes. Indeed, a RPF with the same parameters — but naturally without JMS — was unable to converge rapidly to the fault.

4. Can faults with different dynamics than the ones used in the process model be estimated accurately?

   In Chapter 5, it has been emphasized that the JMRPF introduced in Chapter 4 cannot estimate faults accurately when the true fault dynamics do not match the ones assumes by the filter. This issue has been overcome by improving its particles placement using a Kalman update. This enhanced version of the JMRPF called the RJMRPF was shown to accurately estimate faults with a very different dynamics to the ones used by its process model, as shown in Section 5.5.1.

5. In the case of ambiguous actuator and sensor faults, is it possible to distinguish and estimate them?

   This case was investigated in Chapter 5, and was also overcome using the RJMRPF. The results shown in Section 5.5.2, show better state and fault estimation results compared to the RRPF.

6. Can the false alarm and missed detection probability be computed in real time so that the transition probability matrix of the JMS process model can be adjusted?

   A new way to compute and update the transition probability matrix has been implemented in the JMRPF to take into consideration the false alarm and missed detection probabilities. This new filter was introduced in Chapter 6 and named the AJMRPF.

It was shown to be more efficient than the JMRPF by computing and updating its probability transition matrix in real-time rather than using a user defined constant transition probability matrix. Using an adaptive transition probability matrix, more optimal mode transitions were obtained by relating them to the false alarm and missed detection probabilities, which were computed using an analytical saddlepoint approximation method. The improved state and fault results of the AJMRPF compared to the JMRPF are shown in Section 6.4.

7. Can the proposed solution for the previous questions be used for real-time embedded applications?

   The JMRPF and all its enhanced version are computationally efficient compared to other filter with similar ability. Indeed, using only one model to perform fault estimation, and using only a small number of particles — also known as sentinel particles — to test different hypothesis instead of using a whole different particle filter as a IMM with particle filters would do, participate in this computational effectiveness. The computational cost of a particle filter is mainly due to the use of hundreds or thousands of particles. The filters introduced in this thesis do not require to increase the number of particles and the added computational cost of the improvements for the fault estimation is negligible compared to overall computational cost of the RPF.

7.2 FUTURE WORK

Several future research directions and challenges can be drawn from this work.

On the application side, a test on a 6 degree of freedom fixed-wing UAV considering more than one or two potential faulty actuators or sensors might raise new issues. More complex trajectories should also be tested. If it is validated by numerical results, then an experimental test could be considered to validate the real-time capability of the algorithms and also to validate the use of a process model that does not fully match the true model of the systems.

On the theoretical side, even though challenging faults have been considered in this thesis, some scenarios have not been considered such as the cases of an actuator or sensor failure or efficiency loss. This would then lead to further investigations of the reconfiguration that was not done in this thesis where estimating the fault was sufficient to preserve the UAV flight safety. A more in depth analysis of the impact of faults on mission integrity could also be performed by accounting for the fact that actuator faults reduce the flight envelope. A more optimal fault recovery from a control system viewpoint can be developed by adapting controller gains depending on the fault mode, to prioritise robustness when the fault is present and to prioritise trajectory tracking performance in the fault free case. Moreover, in this thesis, only change in the mean and additive faults have been considered in the numerical simulation, when the presented saddlepoint approximation theory clearly extends to changes in the variance. A modified JMRPF can therefore be designed to account for changes in variance and for multiplicative faults.

Future research directions can also be defined for the GNC module to provide more awareness about the faulty situation to all the GNC modules in order to adapt the behaviour of the UAV to the fault and state estimation in real time.

A

AIRFRAME PARAMETERS

A.1 AEROSONDE UNMANNED AERIAL VEHICLE

Parameter	Value	Unit	Long. Coeff.	Value	Lat. Coeff.	Value
m	13.5	kg	C_{L_0}	0.28	C_{Y_0}	0.0
J_x	0.8244	$\mathrm{kg\,m^2}$	C_{D_0}	0.03	C_{Y_β}	−0.98
J_y	1.135	$\mathrm{kg\,m^2}$	C_{m_0}	−0.02338	C_{Y_p}	0.0
J_z	1.759	$\mathrm{kg\,m^2}$	C_{L_α}	3.45	C_{Y_r}	0.0
J_{xz}	0.1204	$\mathrm{kg\,m^2}$	C_{D_α}	0.30	$C_{Y_{\delta_a}}$	0.0
S	0.55	$\mathrm{m^2}$	C_{m_α}	−0.38	$C_{Y_{\delta_r}}$	−0.17
b	2.8956	m	C_{L_q}	0.0	C_{l_0}	0.0
c	0.189 94	m	C_{D_q}	0.0	C_{l_β}	−0.12
S_{prop}	0.2027	$\mathrm{m^2}$	C_{m_q}	−3.6	C_{l_p}	−0.26
k_{motor}	80		$C_{L_{\delta_e}}$	−0.36	C_{l_r}	0.14
k_{T_P}	0		$C_{D_{\delta_e}}$	0.0	$C_{l_{\delta_a}}$	0.08
k_Ω	0		$C_{m_{\delta_e}}$	−0.5	$C_{l_{\delta_r}}$	0.105
e	0.9		C_{prop}	1.0	C_{n_0}	0.0
ρ_{500m}	1.1680	$\mathrm{kg\,m^{-3}}$	M	50	C_{n_β}	0.25
			α_0	0.4712	C_{n_p}	0.022
			ϵ	0.1592	C_{n_r}	−0.35
			C_{D_p}	0.0437	$C_{n_{\delta_a}}$	0.06
			$C_{n_{\delta_r}}$	−0.032		

Table A.1: Aerodynamics coefficients for the Aerosonde UAV from [67]

A.2 STATE SPACE MODEL COEFFICIENTS

Coefficients	Formula
X_u	$\frac{u^*\rho S}{\mathsf{m}}\left(C_{X_0}+C_{X_\alpha}\alpha^*+C_{X_{\delta_e}}\delta_e^*\right)-\frac{\rho S w^* C_{X_\alpha}}{2\mathsf{m}}+\frac{\rho S c C_{X_q}u^*q^*}{4m\mathbf{V_a}^*}-\frac{\rho S_{prop}C_{prop}u^*}{\mathsf{m}}$
X_w	$-q^*+\frac{w^*\rho S}{\mathsf{m}}\left(C_{X_0}+C_{X_\alpha}\alpha^*+C_{X_{\delta_e}}\delta_e^*\right)+\frac{\rho S c C_{X_q}w^*q^*}{4m\mathbf{V_a}^*}+\frac{\rho S C_{X_\alpha}u^*}{2\mathsf{m}}-\frac{\rho S_{prop}C_{prop}w^*}{\mathsf{m}}$
X_q	$-w^*+\frac{\rho {\mathbf{V_a}^*} S C_{X_q} c}{4\mathsf{m}}$
X_{δ_e}	$\frac{\rho {\mathbf{V_a}^*}^2 S C_{X_{\delta_e}}}{2\mathsf{m}}$
X_{δ_t}	$\frac{\rho S_{prop}C_{prop}k^2\delta_t^*}{\mathsf{m}}$
Z_u	$q^*+\frac{u^*\rho S}{\mathsf{m}}\left(C_{Z_0}+C_{Z_\alpha}\alpha^*+C_{Z_{\delta_e}}\delta_e^*\right)-\frac{\rho S C_{Z_\alpha}w^*}{2\mathsf{m}}+\frac{u^*\rho S C_{Z_q}cq^*}{4m\mathbf{V_a}^*}$
Z_w	$\frac{w^*\rho S}{\mathsf{m}}\left(C_{Z_0}+C_{Z_\alpha}\alpha^*+C_{Z_{\delta_e}}\delta_e^*\right)+\frac{\rho S C_{Z_\alpha}w^*}{2\mathsf{m}}+\frac{\rho w^* S c C_{Z_q}q^*}{4m\mathbf{V_a}^*}$
Z_q	$-u^*+\frac{\rho {\mathbf{V_a}^*} S C_{Z_q} c}{4\mathsf{m}}$
Z_{δ_e}	$\frac{\rho {\mathbf{V_a}^*}^2 S C_{Z_{\delta_e}}}{2\mathsf{m}}$
M_u	$\frac{u^*\rho S c}{J_y}\left(C_{m_0}+C_{m_\alpha}\alpha^*+C_{m_{\delta_e}}\delta_e^*\right)-\frac{\rho S c C_{m_\alpha}w^*}{2J_y}+\frac{\rho S c^2 C_{m_q}q^*u^*}{4J_y\mathbf{V_a}^*}$
M_w	$\frac{w^*\rho S c}{J_y}\left(C_{m_0}+C_{m_\alpha}\alpha^*+C_{m_{\delta_e}}\delta_e^*\right)+\frac{\rho S c C_{m_\alpha}u^*}{2J_y}+\frac{\rho S c^2 C_{m_q}q^*w^*}{4J_y\mathbf{V_a}^*}$
M_q	$\frac{\rho {\mathbf{V_a}^*} S c^2 C_{m_q}}{4J_y}$
M_{δ_e}	$\frac{\rho {\mathbf{V_a}^*}^2 S c C_{m_{\delta_e}}}{2J_y}$

Table A.2: Longitudinal state-space model coefficients from [67]

B

SENSORS PARAMETERS

Constant denotation	Value	Unit
P_0	10 135	$\mathrm{kg\,s^{-2}\,m^{-1}}$
T_0	288.15	K
L_0	−0.0065	$\mathrm{K\,m^{-1}}$
g_n	9.806 65	$\mathrm{m\,s^{-2}}$
R	8.314 459 8	$\mathrm{kg\,m^2\,s^{-2}\,mol^{-1}\,K^{-1}}$
M	0.028 964 4	$\mathrm{kg\,mol^{-1}}$

Table B.1: Constant value of the barometric formula [84]

Axis	σ_{GNSS} (m)	${k_{GNSS}}^{-1}$ (s)	T_s (s)
North	0.21	1100	1.0
East	0.21	1100	1.0
Down	0.40	1100	1.0

Table B.2: GNSS receiver Gauss-Markov model noise parameters [67]

C

FULL STATE FEEDBACK

Full state feedback can only be implementable if all states are controllable and observable (if all state are not observable one can consider recreating a state or output feedback stabilization). Let's consider the controllable and observable system given by (3.43) and the following full state feedback control law:

$$\mathbf{u} = -\mathbf{L}\mathbf{z} + \mathbf{l_c}\mathbf{y^c}, \tag{C.1}$$

where $\mathbf{y^c}$ is the desired output, and $\mathbf{l_c}$ the gain associated with it and $\mathbf{L} = \begin{bmatrix} \mathbf{l}_0 & \mathbf{l}_1 & \dots & \mathbf{l}_{n-1} \end{bmatrix}$ the gain of the full state feedback. The new system with the full state feedback implemented is then given by:

$$\begin{cases} \dot{\mathbf{z}} = (\mathbf{F} - \mathbf{BL})\,\mathbf{z} + \mathbf{B}\mathbf{l_c}\mathbf{y^c} & \text{(C.2a)} \\ \mathbf{y} = \mathbf{H}\mathbf{z} & \text{(C.2b)} \end{cases}$$

Then, the full state feedback gain $\mathbf{L}$ changes the dynamics of the system by moving the poles of the $\mathbf{F}$ matrix. To place the new poles of the system, a commonly used method is to use the LQR method. The full state feedback representation in a block diagram is illustrated in Figure C.1. In a real implementation of this regulator, the state that feeds the gain $\mathbf{L}$ is obtained from sensors or observers or in a GNC loop from the navigation module.

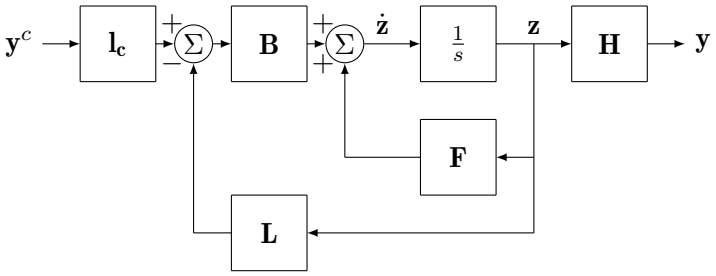


Figure C.1: Full state feedback block diagram

C.1 LINEAR QUADRATIC REGULATOR

The LQR method uses a cost function to compute the full state feedback gain $\mathbf{L}$. For a continuous time system, the cost function minimizes the following quadratic cost function:

$$J(\mathbf{u}) = \int_0^\infty \left(\mathbf{z}^\top \mathbf{Q}\mathbf{z} + \mathbf{u}^\top \mathbf{R}\mathbf{u} + 2\mathbf{z}^\top \mathbf{N}\mathbf{u} \right) \mathrm{d}t, \tag{C.3}$$

where $\mathbf{Q}$ is a diagonal positive definite matrix of the weight attached to tracking performance, and $\mathbf{R}$ is a diagonal positive definite matrix of the weight attached to the control effort. A high coefficient in the matrix $\mathbf{Q}$ optimizes the performance of the state (response time) associated with the line of $\mathbf{Q}$ whereas a high coefficient in $\mathbf{R}$ will minimize the control input. The matrix $\mathbf{N}$ is positive definite, and it acts on the cross product of $\mathbf{z}$ and $\mathbf{u}$. It is taken to be 0 if the cross product optimization is not needed.

The full state feedback gain $\mathbf{L}$ is then:

$$\mathbf{L} = \mathbf{R}^{-1} \left(\mathbf{B}^\top \mathbf{S} + \mathbf{N}^\top \right), \tag{C.4}$$

where $\mathbf{S}$ is the solution to the associated Riccati equation:

$$\mathbf{F}^\top \mathbf{S} + \mathbf{S}\mathbf{F} - (\mathbf{S}\mathbf{B} + \mathbf{N}) \mathbf{R}^{-1} \left(\mathbf{B}^\top \mathbf{S} + \mathbf{N}^\top \right) + \mathbf{Q} = \mathbf{0}_{n_z, nz} \tag{C.5}$$

C.2 FULL STATE FEEDBACK WITH INTEGRATOR EFFECT

The full state feedback with integrator effect has the same structure as the full state feedback, but an integral term is added to the error to track constant non-zero steady state error. Let us consider the controllable and observable system given by (3.43). To minimize the error between the desired output $\mathbf{y^c}$ and the output of the state $\mathbf{z}^j$ — which is the j^{th} state of the state vector $\mathbf{z}$—, a new integrated error state $\mathbf{z_i}$ is created:

$$\dot{\mathbf{z}}_\mathbf{i} = \mathbf{y^c} - \mathbf{H}_{\mathbf{z^j}} \mathbf{z} \tag{C.6}$$

where $\mathbf{H}_{\mathbf{z^j}}$ is the row of the $\mathbf{H}$ matrix corresponding to the observability of the state $\mathbf{z}^j$. Then, the new state vector is $\begin{bmatrix} \mathbf{z} \\ \mathbf{z_i} \end{bmatrix}$ and the new state space representation is:

$$\begin{cases} \begin{bmatrix} \dot{\mathbf{z}} \\ \dot{\mathbf{z}}_\mathbf{i} \end{bmatrix} = \begin{bmatrix} \mathbf{F} & \mathbf{0}_{n_z, n_i} \\ -\mathbf{H}_{\mathbf{z^j}} & \mathbf{0}_{n_i, n_i} \end{bmatrix} \begin{bmatrix} \mathbf{z} \\ \mathbf{z_i} \end{bmatrix} + \begin{bmatrix} \mathbf{B} \\ \mathbf{0}_{n_i, n_u} \end{bmatrix} \mathbf{u} + \begin{bmatrix} \mathbf{0}_{n_z, n_u} \\ \mathbf{I}_{n_i, n_u} \end{bmatrix} \mathbf{y^c} & \text{(C.7a)} \\ \mathbf{y} = \begin{bmatrix} \mathbf{H} & \mathbf{0}_{n_y, n_i} \end{bmatrix} \begin{bmatrix} \mathbf{z} \\ \mathbf{z_i} \end{bmatrix} & \text{(C.7b)} \end{cases}$$

, and with a full state feedback gain it gives:

$$\mathbf{u} = -\mathbf{L}\begin{bmatrix}\mathbf{z}\\ \mathbf{z_i}\end{bmatrix} = -\begin{bmatrix}\mathbf{L_z} & \mathbf{L_{z_i}}\end{bmatrix}\begin{bmatrix}\mathbf{z}\\ \mathbf{z_i}\end{bmatrix} \tag{C.8}$$

where:

$$\mathbf{L_z} = \begin{bmatrix}\mathbf{l_0} & \mathbf{l_1} & \dots & \mathbf{l_{n-2}}\end{bmatrix} \tag{C.9a}$$

$$\mathbf{L_{z_i}} = \begin{bmatrix}\mathbf{l_{n-1}}\end{bmatrix} \tag{C.9b}$$

The closed-loop state space representation with the full states' feedback is:

$$\begin{cases} \begin{bmatrix}\dot{\mathbf{z}}\\ \dot{\mathbf{z}}_\mathbf{i}\end{bmatrix} = \begin{bmatrix}\mathbf{F}-\mathbf{B}\mathbf{L_z} & -\mathbf{B}\mathbf{L_{z_i}}\\ -\mathbf{H_{z^j}} & \mathbf{0}_{n_i,n_i}\end{bmatrix}\begin{bmatrix}\mathbf{z}\\ \mathbf{z_i}\end{bmatrix} + \begin{bmatrix}\mathbf{0}_{n_z,n_u}\\ \mathbf{I}_{n_i,n_u}\end{bmatrix}\mathbf{y^c} & \text{(C.10a)}\\ \mathbf{y} = \begin{bmatrix}\mathbf{H} & \mathbf{0}_{n_y,n_i}\end{bmatrix}\begin{bmatrix}\mathbf{z}\\ \mathbf{z_i}\end{bmatrix} & \text{(C.10b)}\end{cases}$$

The full state feedback with integrator effect representation in the block diagram is illustrated in Figure C.2. In a real implementation of this regulator the state that feed the gain $\mathbf{L_z}$ and $\mathbf{H_{z^j}}$ is obtained through sensors or observers or in a GNC loop from the navigation module.

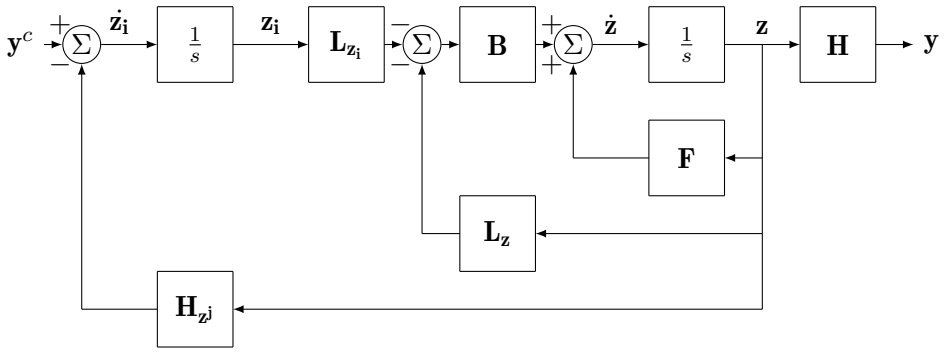


Figure C.2: Full state feedback with integrator effect block diagram

D

MEDIAN RESULTS AND ROOT-MEAN-SQUARE ERROR

D.1 MEDIAN RESULTS

A median results in the simulations presented in this thesis is the $\lfloor N_{\text{MC}}/2 + 1/2 \rfloor$ simulation sorted according to a score, where N_{MC} is the total number of Mote Carlo performed. This score aims to reflect the capacity of the simulation to estimate the faults. When there is only one fault estimated in a simulation this score is given by the temporal mean of the fault estimate error, which is given by:

$$\overline{\varepsilon_{\text{f}}} = \frac{1}{N_k} \sum_{k=1}^{N_k} \sqrt{\left(\text{f}_k - \hat{\text{f}}_k\right)^2} \tag{D.1}$$

where N_k denotes the total number of time step. Then each Monte Carlo performed is sorted according to its $\overline{\varepsilon_{\text{f}}}$ value.

However, having multiple faults estimated per simulation, leads to the computation of multiple $\overline{\varepsilon_{\text{f}}}$ value. To sort the simulation only one value must remain, and a mean or a sum of these value cannot be performed since there are not necessarily in the same state space and have been estimated with an identical noise. Then a standardization of this value is performed, by subtract to $\overline{\varepsilon_{\text{f}}}$ its mean and dividing by its standard deviation respectively given by:

$$\mu_{\overline{\varepsilon_{\text{f}}}} = \frac{1}{N_{\text{MC}}} \sum_{\text{MC}=1}^{N_{\text{MC}}} \overline{\varepsilon_{\text{f}}}^{\text{MC}} \tag{D.2}$$

and

$$\sigma_{\overline{\varepsilon_{\text{f}}}} = \sqrt{\frac{1}{N_{\text{MC}}} \sum_{\text{MC}=1}^{N_{\text{MC}}} \left(\overline{\varepsilon_{\text{f}}}^{\text{MC}} - \mu_{\overline{\varepsilon_{\text{f}}}}\right)^2} \tag{D.3}$$

where the upperscript MC of a variable denote the variable associated with the MC^{th} Mote Carlo.

Then, after being normalized, a mean between the normalized score of each fault estimate is performed and a global score of each simulation is obtained. This score is then used to sort the simulation and then a median value can be obtained.

D.2 ROOT-MEAN-SQUARE ERROR

The RMSE of the state variable x at time step k is given by:

$$\mathrm{RMSE}_k^{\mathrm{x}} = \sqrt{\frac{\sum_{\mathrm{MC}=1}^{N_{\mathrm{MC}}} \left(\mathrm{x}_k^{\mathrm{MC}} - \hat{\mathrm{x}}_k^{\mathrm{MC}}\right)^2}{N_{\mathrm{MC}}}} \tag{D.4}$$

A mean RMSE denoted $\overline{\mathrm{RMSE}}$ can be computed, and it is then given by:

$$\overline{\mathrm{RMSE}}^{\mathrm{x}} = \frac{1}{N_k} \sum_{k=0}^{N_k-1} \mathrm{RMSE}_k^{\mathrm{x}} \tag{D.5}$$

E

PROBABILITY DENSITY FUNCTION APPROXIMATED BY THE LAPLACE APPROXIMATION

The following appendix can be applied for the saddlepoint approximation with $n = 1$.

The density $p_X(x)$ can be expressed in terms of the moment-generating function (MGF) $M_X(t)$ using the inverse Fourier transform by:

$$p_X(x) = \frac{1}{2\pi} \int_{-\infty}^{+\infty} e^{-itx} M_X(it)\, \mathrm{d}t, \tag{E.1}$$

with $i^2 = -1$.

Since $M_X(t) = e^{K_X(t)}$, where $K_X(t)$ is the cumulant-generating function. Then it gives:

$$p_X(x) = \frac{1}{2\pi} \int_{-\infty}^{+\infty} e^{-itx} e^{K_X(it)}\, \mathrm{d}t \tag{E.2a}$$

$$= \frac{1}{2\pi} \int_{-\infty}^{+\infty} e^{K_X(it) - itx}\, \mathrm{d}t \tag{E.2b}$$

Let $t' = it$, then it gives:

$$p_X(x) = \frac{1}{2\pi i} \int_{-i\infty}^{+i\infty} e^{K_X(t) - tx}\, \mathrm{d}t. \tag{E.3}$$

From Cauchy's theorem, the integral is the same over all paths that are parallel to the imaginary axis and is also given by:

$$p_X(x) = \frac{1}{2\pi i} \int_{\tau - i\infty}^{\tau + i\infty} e^{K_X(t) - tx}\, \mathrm{d}t. \tag{E.4}$$

Thus, there are no constraints to choose a value for τ over which the integration is performed. This parameter is set to $\tau = T_X$, which is the saddlepoint since $K_X(t) - tx$

reaches a minimum at T_X on the real axis and the modulus of the integrand of (E.4) reaches a maximum at T_X. The density is then given by:

$$p_X(x) = \frac{1}{2\pi i} \int_{T_X - i\infty}^{T_X + i\infty} e^{K_X(t) - tx} \, \mathrm{d}t. \tag{E.5}$$

Since T_X is the real root of $K'_X(t) - x = 0$, when t is outside an immediate neighbourhood of T_X, the integrand (E.5) becomes negligible.

The 2nd order Taylor expansion of the function $f(t) = K_X(t) - tx$ around T_X that verify $f'(T_X) = K'_X(T_X) - x = 0$ is given by:

$$f(t) \approx f(T_X) + f'(T_X)(t - T_X) + \frac{f''(T_X)}{2}(t - T_X)^2 \tag{E.6a}$$

$$= K_X(T_X) - T_X x + \left(K'_X(T_X) - x\right)(t - T_X) + \frac{K''_X(T_X)}{2}(t - T_X)^2. \tag{E.6b}$$

Since $K'_X(T_X) - x = 0$, the function $f(t)$ can be simplified to:

$$f(t) \approx K_X(T_X) - T_X x + \frac{K''_X(T_X)}{2}(t - T_X)^2. \tag{E.7}$$

Then (E.5) can be rewritten as:

$$p_X(x) \approx \frac{1}{2\pi i} \int_{T_X - i\infty}^{T_X + i\infty} e^{(K_X(T_X) - T_X x) + \frac{K''_X(T_X)}{2}(t - T_X)^2} \, \mathrm{d}t \tag{E.8a}$$

$$= \frac{1}{2\pi i} \lim_{y \to +\infty} \int_{T_X - iy}^{T_X + iy} e^{(K_X(T_X) - T_X x) + \frac{K''_X(T_X)}{2}(t - T_X)^2} \, \mathrm{d}t \tag{E.8b}$$

$$= \frac{1}{2\pi i} e^{(K_X(T_X) - T_X x)} \lim_{y \to +\infty} \int_{T_X - iy}^{T_X + iy} e^{\frac{K''_X(T_X)}{2}(t - T_X)^2} \, \mathrm{d}t. \tag{E.8c}$$

Let $t = T_X + iy$, then it gives $y = -i(t - T_X)$. Then $dy = -idt$ and $(t - T_X)^2 = -y^2$ Then (E.8c) can be rewritten as:

$$p_X(x) \approx \frac{1}{2\pi} e^{K_X(T_X) - T_X x} \int_{-\infty}^{+\infty} e^{-\frac{K''_X(T_X)}{2} y^2} \, \mathrm{d}y \tag{E.9a}$$

$$= e^{K_X(T_X) - T_X x} \left(\frac{1}{\sqrt{2\pi}} \sqrt{\frac{1}{K''_X(T_X)}} \right) \left(\frac{1}{\sqrt{2\pi}\left(\sqrt{\frac{1}{K''_X(T_X)}}\right)} \int_{-\infty}^{+\infty} e^{-\frac{1}{2}\frac{y^2}{\left(\sqrt{\frac{1}{K''_X(T_X)}}\right)^2}} \, \mathrm{d}y \right). \tag{E.9b}$$

Since

$$\left(\frac{1}{\sqrt{2\pi}\left(\sqrt{\frac{1}{K_X''(T_X)}}\right)} \int_{-\infty}^{+\infty} e^{-\frac{1}{2}\frac{y^2}{\left(\sqrt{\frac{1}{K_X''(T_X)}}\right)^2}} \, \mathrm{d}y \right) = \int_{-\infty}^{+\infty} \mathcal{N}\left(y;\, 0,\, \frac{1}{K_X''\left(T_X\right)}\right) \mathrm{d}y = 1. \quad \text{(E.10)}$$

Then (E.9b) can be rewritten as:

$$p_X(x) \approx \sqrt{\frac{1}{2\pi K_X''\left(T_X\right)}} e^{K_X(T_X) - T_X x}. \quad \text{(E.11)}$$

F

RÉSUMÉ EN FRANÇAIS

Les défaillances d'actionneurs ou de capteurs survenant dans un drone peuvent compromettre l'intégrité de la mission. La mise en œuvre de méthodes de diagnostic de fautes est alors nécessaire. Dans cette thèse, l'accent est mis sur l'estimation de fautes dues à des défaillances simultanées de capteurs et d'actionneurs pour un drone à voilure fixe. Pour faire face à certains scénarios complexes de fautes, tels que les fautes simultanées qui induisent une ambiguïté sur les mesures qui se manifeste par la multimodalité de la densité conditionnelle, un filtre particulaire régularisé de type jump-Markov (JMRPF) et des versions améliorées de celui-ci sont présentées dans cette thèse.

Cette méthode est basée sur un filtre particulaire régularisé (RPF) qui approche la densité conditionnelle par une mixture de noyaux et sur un système de Markov à sauts (JMS). La stratégie de saut utilise un petit nombre de particules — appelées particules sentinelles — qui permet de tester en continue l'hypothèse alternative en mode sans faute et en mode avec faute.

Les résultats numériques sont obtenus en utilisant un modèle dynamique linéaire puis non linéaire de la dynamique longitudinale d'un drone à voilure fixe. Les performances du JMRPF sont comparées aux performances des filtres de Kalman à modèles multiples interactifs (IMM-KF) et du RPF. Les performances du JMRPF montrent une nette amélioration de terme de précision de l'estimation des fautes capteurs et actionneurs et des paramètres cinématiques et en termes de robustesse et de vitesse de convergence par rapport aux autres filtres. L'amélioration des performances par rapport aux autres filtres est plus marquée lorsque l'amplitude des fautes augmente au cours du temps.

Une version améliorée du JMRPF, appelée filtre particulaire régularisé robuste à sauts est également présentée et permet d'estimer rapidement et précisément les fautes sans connaissance a priori de la dynamique des fautes. Enfin, une nouvelle approche pour calculer une matrice de probabilité de transition adaptative est présentée en calculant les probabilités de fausse alarme et de non-détection à l'aide de l'approximation du point-selle.

Les algorithmes de navigation proposés permettent à un drone d'atteindre son objectif de suivi de trajectoire de manière autonome, avec une sécurité et une précision accrues.

Cette thèse tente de répondre aux questions de recherche suivantes :

1. Est-ce qu'un JMS combiné avec un RPF augmente la précision et accélère la vitesse de convergence par rapport à un RPF en présence de fautes additives abruptes ?

L'utilisation d'un JMS avec un RPF tel que présenté au Chapitre 4 s'est avérée plus efficace — ou au moins aussi efficace selon les types de fautes et les paramètres de réglage utilisés — que le RPF seul en termes de précision, de robustesse et de vitesse de convergence. Cela a été illustré dans la Section 5.5. La nouvelle façon d'intégrer un JMS dans un filtre particulaire fournit de meilleurs résultats d'estimation de fautes et paramètres d'état qu'un IMM-KF dans le cas gaussien avec des fautes additives abruptes.

2. Est-il possible de distinguer et d'estimer des fautes de capteurs en utilisant uniquement un JMS comme modèle de prédiction d'un filtre particulaire ?

   Dans le Chapitre 4, le cas de faute de capteurs a été étudié, et il a été montré que le JMRPF est en mesure d'estimer avec précision et rapidité les différents types de fautes capteurs.

3. Est-ce qu'une faute additive abrupte, de grande amplitude par rapport au bruit de dynamique, peut être estimée avec précision dans un court laps de temps ?

   Le JMRPF a démontré une capacité à estimer une faute abrupte sans avoir à augmenter artificiellement le bruit de dynamique. Cela a été particulièrement souligné dans la Section 5.5, où le JMRPF est utilisé avec succès pour estimer des fautes abruptes de différentes amplitudes même avec un faible bruit de dynamique. En effet, un RPF avec les mêmes paramètres de réglages n'a pas pu converger vers la faute.

4. Peut-on estimer des fautes avec des dynamiques différentes de celles utilisées dans le modèle de prédiction et avec quelle précision ?

   Dans le Chapitre 5, il a été mentionné que le JMRPF introduit dans le Chapitre 4 ne peut pas estimer correctement les fautes avec précision, lorsque la véritable dynamique des fautes ne correspond pas à celle du modèle prédictif du filtre. Ce problème est résolu en améliorant la propagation des particules en utilisant une correction de Kalman. Cette version améliorée du JMRPF appelée le RJMRPF permet d'estimer avec précision les fautes qui ont des dynamiques très différente de celles utilisées par le modèle du filtre, comme indiqué dans la Section 5.5.1.

5. En cas de mesures ambiguës dues à des fautes d'actionneurs et de capteurs, est-il possible de les distinguer et de les estimer ?

   Ce cas a fait l'objet d'une analyse approfondie dans le Chapitre 5 et a également été résolu à l'aide du RJMRPF. Les résultats présentés dans la Section 5.5.2, montrent de meilleurs résultats d'estimation des paramètres d'état et des fautes par rapport au RRPF.

6. Les probabilités de fausse alarme et de non-détection peuvent-elles être calculées en temps réel afin que la matrice de probabilité de transition du modèle du JMS soit ajustée ?

Une nouvelle façon de calculer et de mettre à jour la matrice de probabilité de transition a été mise en œuvre dans la JMRPF pour prendre en considération les probabilités de fausse alarme et de non-détection. Ce nouveau filtre a été introduit dans le Chapitre 6 et a été nommé AJMRPF.

Ce filtre s'est avéré plus efficace que le JMRPF en calculant et en mettant à jour la matrice de probabilité de transition en temps réel plutôt que d'utiliser une matrice de probabilité de transition constante fixée par l'utilisateur. La matrice de probabilité de transition adaptative est calculée en fonction des probabilités de fausse alarme et de non-détection, qui ont été calculées à l'aide d'une approximation analytique de la méthode du point-selle. Les résultats montrent une amélioration de l'estimation des paramètres d'état et des fautes par l'AJMRPF par rapport au JMRPF. Ces résultats sont présentés dans la Section 6.4.

7. Les algorithmes proposés dans cette thèse peuvent-ils être implémentés pour des applications temps réel embarquées ?

   Le JMRPF et toutes ses variantes ont un coût de calcul similaire aux autres filtres. En effet, en utilisant un seul modèle prédictif pour effectuer l'estimation des fautes, et en n'utilisant qu'un petit nombre de particules — appelées particules sentinelles — pour tester différentes hypothèses au lieu d'utiliser un banc de filtres particulaires comme les approches de type IMM évite de d'augmenter le coût de calcul. Le coût de calcul d'un filtre particulaire est principalement dû à l'utilisation de milliers de particules. Les filtres particulaires introduits dans cette thèse ne nécessitent pas d'augmenter le nombre de particules.

BIBLIOGRAPHY

[1] J. Herkert, J. Borenstein, and K. Miller. "The Boeing 737 MAX: Lessons for Engineering Ethics". In: *Science and Engineering Ethics* 84 (2020) (cit. on p. 1).

[2] S. Wang, X. Zhan, Y. Zhai, and B. Liu. "Fault detection and exclusion for tightly coupled GNSS/INS system considering fault in state prediction." In: *Sensors* 20.3 (2020), p. 590 (cit. on p. 1).

[3] A. Susini. "Lecture Notes in Information Sciences (LNIS), RIMMA Risk Information Management, Risk Models, and Application". In: *A technocritical review of drones crash risk probabilistic consequences and its societal acceptance.* Vol. 7. CODATA, Germany, 2014, pp. 27–38 (cit. on p. 1).

[4] Civil Aviation Authority. "CAP 722 Unmanned Aircraft System Operations in UK Airspace—Guidance". In: *Directorate of Airspace Policy* (2020) (cit. on p. 1).

[5] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. "A Tutorial on Particle Filters for Online Nonlinear/Non-Gaussian Bayesian Tracking". In: *IEEE Transactions on Signal Processing* 50.2 (2002), pp. 174–188 (cit. on p. 2).

[6] R. Conde, J.A. Cobano, and A. Ollero. "Method based on a particle filter for UAV trajectory prediction under uncertainties". In: *Proceedings of the 40th international symposium of robotics*. Barcelona,Spain, 2009 (cit. on p. 2).

[7] N. Merlinge, K. Dahia, H. Piet-Lahanier, J. Brusey, and N. Horri. "A Box Regularized Particle Filter for state estimation with severely ambiguous and non-linear measurements". In: *Automatica* 104 (2019), pp. 102–110 (cit. on p. 2).

[8] E. D'Amato, V. A. Nardi, I. Notaro, and V. Scordamaglia. "A Particle Filtering Approach for Fault Detection and Isolation of UAV IMU Sensors: Design, Implementation and Sensitivity Analysis". In: *Sensors* 21.9 (2021), p. 3066 (cit. on p. 2).

[9] M. Yu, H. Oh, and W. Chen. "An improved multiple model particle filtering approach for manoeuvring target tracking using airborne GMTI with geographic information". In: *Aerospace Science and Technology* 52 (2016), pp. 62–69 (cit. on p. 2).

[10] A. Doucet, N. J. Gordon, and V. Krishnamurthy. "Particle filters for state estimation of jump Markov linear systems". In: *IEEE Transactions on signal processing* 49.3 (2001), pp. 613–624 (cit. on pp. 2, 43).

[11] S. Tafazoli and X. Sun. "Hybrid system state tracking and fault detection using particle filters". In: *IEEE Transactions on Control Systems Technology* 14.6 (2006), pp. 1078–1087 (cit. on pp. 2, 39, 43).

[12] R. Isermann. "Terminology in fault detection and diagnosis". In: *Combustion Engine Diagnosis*. Springer, 2017, pp. 295–297 (cit. on pp. 7, 11).

[13] A. Varga. "Solving Fault Diagnosis Problems". In: *Studies in Systems, Decision and Control, 1st ed.; Springer International Publishing: Berlin, Germany* 84 (2017) (cit. on pp. 8, 9, 11, 15).

[14] R. Isermann. *Fault-diagnosis systems: an introduction from fault detection to fault tolerance*. Springer Science & Business Media, 2006 (cit. on pp. 8, 10).

[15] S. X. Ding. *Model-based fault diagnosis techniques: design schemes, algorithms, and tools*. Springer Science & Business Media, 2008 (cit. on pp. 10–12, 16).

[16] R. N. Clark. "The dedicated observer approach to instrument failure detection". In: *1979 18th IEEE Conference on Decision and Control including the Symposium on Adaptive Processes*. Vol. 2. 1979, pp. 237–241 (cit. on p. 10).

[17] M. Thirumarimurugan, N. Bagyalakshmi, and P. Paarkavi. "Comparison of fault detection and isolation methods: A review". In: *2016 10th International Conference on Intelligent Systems and Control (ISCO)*. IEEE. 2016, pp. 1–6 (cit. on p. 11).

[18] S. X. Ding. *Data-driven design of fault diagnosis and fault-tolerant control systems*. Springer, 2014 (cit. on p. 11).

[19] D. Guo, M. Zhong, H. Ji, Y. Liu, and R. Yang. "A hybrid feature model and deep learning based fault diagnosis for unmanned aerial vehicle sensors". In: *Neurocomputing* 319 (2018), pp. 155–163 (cit. on p. 12).

[20] G. Iannace, G. Ciaburro, and A. Trematerra. "Fault diagnosis for UAV blades using artificial neural network". In: *Robotics* 8.3 (2019), p. 59 (cit. on p. 12).

[21] I. A. Al-Zyoud and K. Khorasani. "Neural network-based actuator fault diagnosis for attitude control subsystem of an unmanned space vehicle". In: *The 2006 IEEE International Joint Conference on Neural Network Proceedings*. IEEE. 2006, pp. 3686–3693 (cit. on p. 12).

[22] B. Wang, Z. Wang, L. Liu, D. Liu, and X. Peng. "Data-driven anomaly detection for UAV sensor data based on deep learning prediction model". In: *2019 Prognostics and System Health Management Conference (PHM-Paris)*. IEEE. 2019, pp. 286–290 (cit. on p. 12).

[23] L. Wen, X. Li, L. Gao, and Y. Zhang. "A new convolutional neural network-based data-driven fault diagnosis method". In: *IEEE Transactions on Industrial Electronics* 65.7 (2017), pp. 5990–5998 (cit. on p. 12).

[24] R. Isermann and P. Ballé. "Trends in the application of model-based fault detection and diagnosis of technical processes". In: *Control Engineering Practice* 5.5 (1997), pp. 709–719. ISSN: 09670661 (cit. on p. 12).

[25] P.M. Frank and X. Ding. "Survey of robust residual generation and evaluation methods in observer-based fault detection systems". In: *Journal of Process Control* 7.6 (1997), pp. 403–424. ISSN: 0959-1524 (cit. on p. 12).

[26] M. Basseville and I. V. Nikiforov. *Detection of Abrupt Changes - Theory and Application*. Prentice Hall, Inc., 1993, p. 550 (cit. on pp. 13–15).

[27] G. K. Kanji. *100 statistical tests*. 3rd. Sage, 2006. ISBN: 141292376X (cit. on p. 14).

[28] A. Murangira. "Nouvelles approches en filtrage particulaire. Application au recalage de la navigation inertielle". PhD thesis. Université de Technologie de Troyes, 2014 (cit. on p. 14).

[29] X. R. Li and L. Wang. "Fault detection using sequential probability ratio test". In: *IEEE Power Engineering Society. 1999 Winter Meeting (Cat. No. 99CH36233)*. Vol. 2. IEEE. 1999, pp. 938–943 (cit. on p. 15).

[30] A. Dutta, M. McKay, F. Kopsaftopoulos, and F. Gandhi. "Rotor Fault Detection and Identification on a Hexacopter Based on Statistical Time Series Methods". In: *Vertical Flight Society 75th Annual Forum, Philadelphia, PA*. 2019 (cit. on p. 15).

[31] T. Fawcett. "An introduction to ROC analysis". In: *Pattern Recognition Letters* 27.8 (2006). ROC Analysis in Pattern Recognition, pp. 861–874. ISSN: 0167-8655 (cit. on p. 15).

[32] J. Neyman and E. S. Pearson. "IX. On the problem of the most efficient tests of statistical hypotheses". In: *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character* 231.694-706 (1933), pp. 289–337 (cit. on p. 16).

[33] H. V. Poor. *An introduction to signal detection and estimation*. Springer Science & Business Media, 2013 (cit. on p. 16).

[34] J. Fan, S. Upadhye, and A. Worster. "Understanding receiver operating characteristic (ROC) curves". In: *Canadian Journal of Emergency Medicine* 8.1 (2006), pp. 19–20 (cit. on p. 17).

[35] W. J. Youden. "Index for rating diagnostic tests". In: *Cancer* 3.1 (1950), pp. 32–35 (cit. on p. 17).

[36] S. M. Kay. *Fundamentals of statistical signal processing: estimation theory*. Prentice-Hall, Inc., 1993 (cit. on p. 18).

[37] P. Del Moral and S. Penev. *Stochastic Processes: From Applications to Theory*. Chapman and Hall/CRC, 2017 (cit. on p. 21).

[38] R. Bassett and J. Deride. "Maximum a posteriori estimators as a limit of Bayes estimators". In: *Mathematical Programming* 174.1 (2019), pp. 129–144 (cit. on p. 22).

[39] R. J. Rossi. *Mathematical statistics: an introduction to likelihood based inference*. John Wiley & Sons, 2018 (cit. on p. 22).

[40] D. Guo, Y. Wu, S. S. Shitz, and S. Verdú. "Estimation in Gaussian noise: Properties of the minimum mean-square error". In: *IEEE Transactions on Information Theory* 57.4 (2011), pp. 2371–2385 (cit. on p. 23).

[41] R. E. Kalman. "A New Approach to Linear Filtering and Prediction Problems". In: *Transactions of the ASME–Journal of Basic Engineering* 82.Series D (1960), pp. 35–45 (cit. on pp. 23, 24).

[42] B. D. O. Anderson and J. B. Moore. *Optimal Filtering*. Dover Books on Electrical Engineering. Dover Publications, 2012. ISBN: 9780486136899 (cit. on p. 23).

[43] M. I. Ribeiro. "Kalman and extended kalman filters: Concept, derivation and properties". In: *Institute for Systems and Robotics* 43 (2004), p. 46 (cit. on p. 24).

[44] S. Hansen and M. Blanke. "Diagnosis of Airspeed Measurement Faults for Unmanned Aerial Vehicles". In: *IEEE Transactions on Aerospace and Electronic Systems* 50.1 (2014), pp. 224–239 (cit. on p. 25).

[45] G. P. Huang, A. I. Mourikis, and S. I. Roumeliotis. "Analysis and improvement of the consistency of extended Kalman filter based SLAM". In: *2008 IEEE International Conference on Robotics and Automation*. IEEE. 2008, pp. 473–479 (cit. on p. 25).

[46] V. Kadirkamanathan, P. Li, and T. Kirubarajan. "Sequential Monte Carlo filtering vs. the IMM estimator for fault detection and isolation in nonlinear systems". In: *Component and Systems Diagnostics, Prognosis, and Health Management*. Vol. 4389. International Society for Optics and Photonics. 2001, pp. 263–274 (cit. on pp. 26, 43).

[47] K. Xiong, H. Y. Zhang, and C. W. Chan. "Performance evaluation of UKF-based nonlinear filtering". In: *Automatica* 42.2 (2006), pp. 261–270 (cit. on p. 26).

[48] K. Xiong, C. W. Chan, and H. Y. Zhang. "Detection of satellite attitude sensor faults using the UKF". In: *IEEE Transactions on Aerospace and Electronic Systems* 43.2 (2007), pp. 480–491 (cit. on p. 26).

[49] B. Ristic, S. Arulampalam, and N. Gordon. *Beyond the Kalman filter: Particle filters for tracking applications*. Artech house, 2003 (cit. on pp. 27, 37).

[50] P. Del Moral. "Nonlinear filtering: Interacting particle resolution". In: *Comptes Rendus de l'Académie des Sciences-Series I-Mathematics* 325.6 (1997), pp. 653–658 (cit. on p. 27).

[51] C. Musso, N. Oudjane, and F. Le Gland. "Improving Regularised Particle Filters". In: *Sequential Monte Carlo Methods in Practice*. Ed. by Arnaud Doucet, Nando de Freitas, and Neil Gordon. New York, NY: Springer New York, 2001, pp. 247–271. ISBN: 978-1-4757-3437-9 (cit. on pp. 34, 35).

[52] W. R. Gilks and C. Berzuini. "Following a moving target—Monte Carlo inference for dynamics Bayesian models". In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 63.1 (2001), pp. 127–146 (cit. on p. 34).

[53] O. Aharon, O. Tslil, and A. Carmi. "A fast mcmc particle filter". In: *2018 21st International Conference on Information Fusion (FUSION)*. IEEE. 2018, pp. 1874–1881 (cit. on p. 34).

[54] B. W. Silverman. *Density estimation for statistics and data analysis*. Routledge, 2018 (cit. on p. 35).

[55] J. Dezert and B. Pannetier. "A PCR-BIMM filter for maneuvering target tracking". In: *2010 13th International Conference on Information Fusion*. IEEE. 2010, pp. 1–8 (cit. on p. 37).

[56] X. Li, Y. Bar-Shalom, and W. D. Blair. "Engineer's guide to variable-structure multiple-model estimation for tracking". In: *Multitarget-multisensor tracking: Applications and advances.* 3 (2000), pp. 499–567 (cit. on p. 37).

[57] Y. Zhang and X. R. Li. "Detection and diagnosis of sensor and actuator failures using IMM estimator". In: *IEEE Transactions on aerospace and electronic systems* 34.4 (1998), pp. 1293–1313 (cit. on p. 38).

[58] B. Pourbabaee, N. Meskin, and K. Khorasani. "Sensor fault detection, isolation, and identification using multiple-model-based hybrid Kalman filter for gas turbine engines". In: *IEEE Transactions on Control Systems Technology* 24.4 (2015), pp. 1184–1200 (cit. on p. 38).

[59] C. Rago, R. Prasanth, R. K. Mehra, and R. Fortenbaugh. "Failure detection and identification and fault tolerant control using the IMM-KF with applications to the Eagle-Eye UAV". In: *Proceedings of the 37th IEEE Conference on Decision and Control (Cat. No.98CH36171)*. Vol. 4. 1998, 4208–4213 vol.4 (cit. on p. 38).

[60] L. Cork and R. Walker. "Sensor fault detection for UAVs using a nonlinear dynamic model and the IMM-UKF algorithm". In: *2007 Information, Decision and Control*. IEEE. 2007, pp. 230–235 (cit. on p. 38).

[61] C. E. Seah and I. Hwang. "State estimation for stochastic linear hybrid systems with continuous-state-dependent transitions: an IMM approach". In: *IEEE Transactions on Aerospace and Electronic Systems* 45.1 (2009), pp. 376–392 (cit. on p. 39).

[62] W. Liu and I. Hwang. "On hybrid state estimation for stochastic hybrid systems". In: *IEEE Transactions on Automatic Control* 59.10 (2014), pp. 2615–2628 (cit. on p. 39).

[63] Y. Bar-Shalom, S. Challa, and H. A. P. Blom. "IMM estimator versus optimal estimator for hybrid systems". In: *IEEE Transactions on Aerospace and Electronic Systems* 41.3 (2005), pp. 986–991 (cit. on p. 39).

[64] H. A. P. Blom. "An efficient filter for abruptly changing systems". In: *The 23rd IEEE Conference on Decision and Control*. IEEE. 1984, pp. 656–658 (cit. on p. 39).

[65] A. Giremus, J. Tourneret, and V. Calmettes. "A particle filtering approach for joint detection/estimation of multipath effects on GPS measurements". In: *IEEE Transactions on Signal Processing* 55.4 (2007), pp. 1275–1285 (cit. on p. 43).

[66] H. Driessen and Y. Boers. "Efficient particle filter for jump Markov nonlinear systems". In: *IEE Proceedings-radar, sonar and navigation* 152.5 (2005), pp. 323–326 (cit. on p. 43).

[67] R. W. Beard and T. W. Mclain. *Small Unmanned Aircraft - Theory and Practice*. Princeton University Press, 2012. ISBN: 9780691149219 (cit. on pp. 47, 48, 52, 54, 175–177).

[68] E. Kaplan and C. J. Hegarty. *Understanding GPS/GNSS: Principles and Applications*. 3rd. Norwood, MA, USA: Artech House, Inc., 2017. ISBN: 1630810584, 9781630810580 (cit. on p. 58).

[69] J. J. Spilker Jr, P. Axelrad, B. W. Parkinson, and P. Enge. *Global Positioning System: Theory and Applications, Volume I*. American Institute of Aeronautics and Astronautics, 1996 (cit. on p. 58).

[70] J. Rankin. "An error model for sensor simulation GPS and differential GPS". In: *Proceedings of 1994 IEEE Position, Location and Navigation Symposium - PLANS'94*. 1994, pp. 260–266 (cit. on p. 58).

[71] E. Iglésis, K. Dahia, H. Piet-Lahanier, N. Merlinge, N. Horri, and J. Brusey. "A Jump-Markov Regularized Particle Filter for the estimation of ambiguous sensor faults". In: vol. 53. 2. 21th IFAC World Congress. 2020, pp. 756–763 (cit. on p. 85).

[72] M. Saif and Y. Guan. "A new approach to robust fault detection and identification". In: *IEEE Transactions on Aerospace and Electronic Systems* 29.3 (1993), pp. 685–695 (cit. on p. 89).

[73] E. Iglésis, N. Horri, K. Dahia, J. Brusey, and H. Piet-Lahanier. "Nonlinear Estimation of Sensor Faults With Unknown Dynamics for a Fixed Wing Unmanned Aerial Vehicle". In: *2021 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE. 2021, pp. 404–412 (cit. on p. 110).

[74] E. Iglésis, N. Horri, J. Brusey, K. Dahia, and H. Piet-Lahanier. "Simultaneous Actuator and Sensor Faults Estimation for Aircraft Using a Jump-Markov Regularized Particle Filter". In: *2021 IEEE International Conference on Prognostics and Health Management (ICPHM)*. IEEE. 2021, pp. 1–10 (cit. on p. 110).

[75] V. Kadirkamanathan, P. Li, M. H. Jaward, and S.G. Fabri. "Particle filtering-based fault detection in non-linear stochastic systems". In: *International Journal of Systems Science* 33.4 (2002), pp. 259–265 (cit. on p. 139).

[76] B. Zhao, R. Skjetne, M. Blanke, and F. Dukan. "Particle Filter for Fault Diagnosis and Robust Navigation of Underwater Robot". In: *IEEE Transactions on Control Systems Technology* 22.6 (2014), pp. 2399–2407 (cit. on p. 139).

[77] J. L. Jensen. *Saddlepoint approximations*. 16. Oxford University Press, 1995 (cit. on p. 139).

[78] H. E. Daniels. "Saddlepoint approximations in statistics". In: *The Annals of Mathematical Statistics* (1954), pp. 631–650 (cit. on pp. 139, 142, 143, 145, 147).

[79] S. Aldosari and J. Moura. "Detection in sensor networks: The Saddlepoint approximation". In: *IEEE Transactions on Signal Processing* 55.1 (2007), pp. 327–340 (cit. on p. 139).

[80] C. Musso, E. Jay, and J. Ovarlez. "Saddlepoint approximation applied to fusion in multiple sensor and to detection in clutter". In: *Proc. Sixth Int'l Conf. Information Fusion*. 2003 (cit. on pp. 139, 145).

[81] G. Xie, L. Sun, T. Wen, X. Hei, and F. Qian. "Adaptive transition probability matrix-based parallel IMM algorithm". In: *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 51.5 (2019), pp. 2980–2989 (cit. on p. 141).

[82] R. Lugannani and S. Rice. “Saddle point approximation for the distribution of the sum of independent random variables”. In: *Advances in applied probability* 12.2 (1980), pp. 475–490 (cit. on pp. 143, 147, 149).

[83] E. T. Lee and J. Wang. “Appendix A: Newton–Raphson Method”. In: *Statistical Methods for Survival Data Analysis*. John Wiley & Sons, Ltd, 2003, pp. 428–432. ISBN: 9780471458548 (cit. on p. 155).

[84] E. Tiesinga, P. J. Mohr, D. B. Newell, and B. N. Taylor. *The 2018 CODATA Recommended Values of the Fundamental Physical Constants*. National Institute of Standards and Technology, Gaithersburg, MD 20899, 2019 (cit. on p. 177).