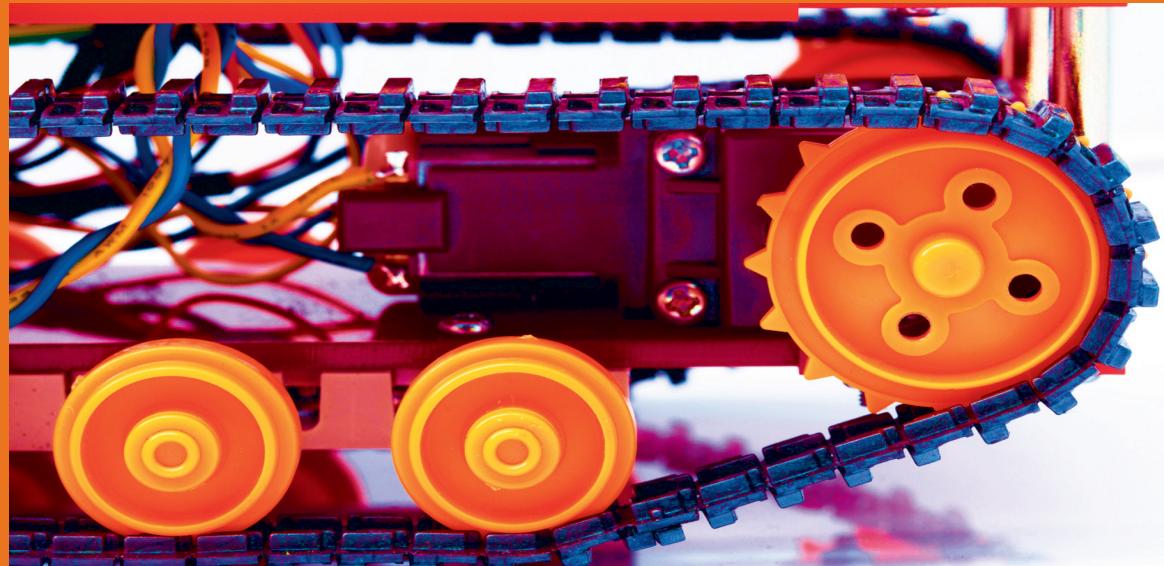


La robotique mobile

cours et exercices

Luc Jaulin



La robotique mobile

First published 2015 in Great Britain by ISTE Editions Ltd.

Apart from any fair dealing for the purposes of research or private study, or criticism or review, as permitted under the Copyright, Designs and Patents Act 1988, this publication may only be reproduced, stored or transmitted, in any form or by any means, with the prior permission in writing of the publishers, or in the case of reprographic reproduction in accordance with the terms and licenses issued by the CLA. Enquiries concerning reproduction outside these terms should be sent to the publishers at the undermentioned address:

ISTE Editions Ltd
27-37 St George's Road
London SW19 4EU
UK

© ISTE Editions Ltd 2015

The rights of the authors of this work have been asserted by them in accordance with the Copyright, Designs and Patents Act 1988.

British Library Cataloguing-in-Publication Data
A CIP record for this book is available from the British Library
ISBN: 978-1-78405-087-0 (print)
ISBN: 978-1-78406-087-9 (e-book)



Printed and bound in Great Britain by CPI Group (UK) Ltd., Croydon, Surrey CR0 4YY, December 2015

Document protégé électroniquement. Réservé à l'usage unique de Valentin GIES
(vgies@hotmail.com) Transaction : 40506

La robotique mobile

cours et exercices

Luc Jaulin



Ouvrage publié sous la direction de
Hisham Abou Kandil

Table des matières

Introduction	9
Chapitre 1. Modélisation tridimensionnelle	13
1.1. Matrices de rotation	14
1.1.1. Définition	14
1.1.2. Vecteur de rotation	15
1.1.3. Adjoint	16
1.1.4. Changement de repère	17
1.2. Angles d'Euler	19
1.2.1. Définition	19
1.2.2. Dérivée d'une matrice d'Euler	20
1.2.3. Vecteur rotation d'une matrice d'Euler	21
1.3. Modèle cinématique d'un robot solide	22
1.4. Exercices	25
1.5. Corrections	37
Chapitre 2. Commande par bouclage linéarisant	51
2.1. Commande d'une chaîne d'intégrateurs	51
2.1.1. Régulateur proportionnel et dérivées	52
2.1.2. Régulateur proportionnel intégral et dérivées	53
2.2. Exemple introductif	54
2.3. Principe de la méthode	55
2.3.1. Principe	55
2.3.2. Degré relatif	57
2.3.3. Matrice des retards différentiels	58
2.3.4. Singularités	59

2.4. Char	61
2.4.1. Premier modèle	61
2.4.2. Deuxième modèle	63
2.5. Régulation d'un tricycle	64
2.5.1. Commande en vitesse et en cap	64
2.5.2. Commande en position	66
2.5.3. Choix d'une autre sortie	67
2.6. Voilier	67
2.6.1. Polaire des vitesses	68
2.6.2. Retards différentiels	69
2.6.3. Méthode par bouclage linéarisant	70
2.6.4. Commande par la polaire des vitesses	72
2.7. Modèle cinématique et modèle dynamique	73
2.7.1. Principe	73
2.7.2. Exemple du pendule inversé	74
2.7.2.1. Modèle dynamique	74
2.7.2.2. Modèle cinématique	75
2.7.3. Servo-moteurs	77
2.8. Exercices	78
2.9. Corrections	85
Chapitre 3. Commande sans modèle	99
3.1. Commande sans modèle d'un robot char	100
3.1.1. Commande proportionnelle en cap et vitesse	100
3.1.2. Commande proportionnelle et dérivée en cap	102
3.2. Skate car	103
3.2.1. Modèle	104
3.2.2. Commande sinusoïdale	105
3.2.3. Commande à poussée maximale	106
3.2.4. Simplification de la dynamique rapide	108
3.3. Voilier	110
3.3.1. Problème	110
3.3.2. Régulateur	112
3.3.3. Navigation	117
3.3.4. Expérience	118
3.4. Exercices	120
3.5. Corrections	127
Chapitre 4. Guidage	135
4.1. Guidage sur une sphère	135
4.2. Planification de trajectoires	138

4.2.1. Exemple simple	138
4.2.2. Polynômes de Bézier	139
4.3. Diagramme de Voronoï	141
4.3.1. Triangulation de Delaunay	141
4.4. Méthode des champs de potentiels artificiels	142
4.5. Exercices	143
4.6. Corrections	149
Chapitre 5. Localisation instantanée	161
5.1. Capteurs	161
5.2. Localisation goniométrique	165
5.2.1. Formulation du problème	165
5.2.2. Arcs capables	166
5.2.3. Triangulation statique d'un robot plan	167
5.2.3.1. Deux amers et une boussole	167
5.2.3.2. Trois amers	168
5.2.4. Triangulation dynamique	169
5.2.4.1. Un amer, une boussole, des odomètres	169
5.2.4.2. Un amer sans boussole	169
5.3. Multilatération	170
5.4. Exercices	171
5.5. Corrections	175
Chapitre 6. Identification	185
6.1. Fonctions quadratiques	185
6.1.1. Définition	185
6.1.2. Dérivée d'une forme quadratique	186
6.1.3. Valeurs propres d'une fonction quadratique	187
6.1.4. Minimisation d'une fonction quadratique	187
6.2. Méthode des moindres-carrés	188
6.2.1. Cas linéaire	188
6.2.2. Cas non linéaire	190
6.3. Exercices	191
6.4. Corrections	194
Chapitre 7. Filtre de Kalman	203
7.1. Matrices de covariance	203
7.1.1. Définitions et interprétations	203
7.1.2. Propriétés	205

7.1.3. Ellipsoïde de confiance	206
7.1.4. Génération de vecteurs aléatoires gaussiens	208
7.2. Estimateur orthogonal non biaisé	209
7.3. Application à l'estimation linéaire	213
7.4. Filtre de Kalman	214
7.5. Lisseur de Kalman	217
7.6. Exercices	218
7.7. Corrections	236
Bibliographie	269
Index	271

Introduction

Un *robot mobile* peut se définir comme un système mécanique capable de se déplacer dans son environnement de façon autonome. Pour cela, un robot doit être équipé :

- de *capteurs* qui l'aideront à percevoir son environnement (qu'il connaît plus ou moins) et à se localiser ;
- d'*actionneurs* qui lui permettront de se mouvoir ;
- d'une *intelligence* (ou algorithme, ou régulateur), qui lui permettra de calculer, à partir des données recueillies par les capteurs, les commandes à envoyer aux actionneurs afin de réaliser la mission demandée.

Enfin, il faut rajouter à ce robot son *environnement* qui correspond au monde dans lequel il évolue et sa *mission* qui est la tâche qu'il doit accomplir. Les robots mobiles sont en constante évolution principalement depuis les années 2000, dans les domaines militaires (drones volants [BEA 12], robots sous-marins [CRE 14], etc.), le médical ou l'agriculture. Ils sont particulièrement demandés pour réaliser des tâches considérées comme pénibles ou dangereuses pour l'homme. C'est le cas des opérations de déminage, de recherche de boîtes noires des avions abîmés au fond de l'océan ou d'exploration de planètes. Les satellites artificiels, les lanceurs (comme Ariane V), les métros sans conducteurs, les ascenseurs sont des exemples de robots mobiles. Les avions de ligne, les trains et les voitures évoluent de façon continue vers des systèmes de plus en plus autonomes et deviendront très probablement des robots mobiles dans les prochaines décennies.

La *robotique mobile* est la discipline qui s'intéresse à la conception de robots mobiles [LAU 01]. Elle s'appuie sur d'autres disciplines comme l'automatique, le traitement du signal, la mécanique, l'informatique et l'électronique. L'objectif de ce livre est de donner un aperçu des outils et des méthodes de la robotique qui permettront d'aider à la conception de robots mobiles. Les robots seront modélisés

par des *équations d'état*, c'est-à-dire des équations différentielles (le plus souvent non linéaires) du premier ordre. Ces équations d'état peuvent être obtenues en utilisant les lois de la mécanique. Il n'est pas dans notre objectif d'enseigner en détail les méthodes de modélisation des robots (voir [JAU 05] et [JAU 14] pour plus d'informations sur le sujet), mais juste d'en rappeler les principes. Par *modélisation*, nous entendons ici l'obtention des équations d'état. Cette étape est indispensable pour la simulation des robots ainsi que pour la conception des régulateurs. Nous allons toutefois illustrer le principe de la modélisation au chapitre 1 sur des exemples volontairement tridimensionnels. Ce choix a pour objectif d'introduire des concepts importants pour la robotique comme les angles d'Euler ou les matrices de rotation. Par exemple, nous nous intéresserons à la dynamique d'une roue et à la cinématique d'un robot sous-marin. Les robots mobiles sont des systèmes fortement non linéaires et seule une approche non linéaire permet la synthèse de régulateurs performants. Cette synthèse fait l'objet des chapitres 2 et 3. Le chapitre 2 s'appuie essentiellement sur des méthodes de commande fondées sur l'utilisation du modèle du robot. L'approche utilisera le concept du *bouclage linéarisant* qui sera introduit et illustré à travers de nombreux exemples. Le chapitre 3 propose des méthodes plus pragmatiques n'utilisant pas le modèle d'état du robot et qui seront qualifiées de *sans modèle* ou *mimétiques*. L'approche utilise une représentation plus intuitive du robot et est adaptée aux situations où les robots sont relativement faciles à téléopérer, comme c'est le cas pour les voitures, des voiliers ou les avions. Le chapitre 4 s'intéresse au *guidage* et se place à un niveau supérieur à celui de la régulation. C'est-à-dire que l'on s'intéresse à guider ou à superviser le système déjà régulé par les outils des chapitres 2 et 3. Nous chercherons donc à trouver la consigne à donner au régulateur afin que le robot accomplisse la mission qui lui est demandée. Le guidage devra donc prendre en compte la connaissance de l'environnement, la présence d'obstacles ou bien la rotundité de la terre. Les méthodes de régulation non linéaires ou de guidage demandent une bonne connaissance des variables d'état du système, comme par exemple les variables définissant la position du robot. Ces variables de position sont les plus difficiles à trouver et le chapitre 5 se focalise sur le problème de *localisation*. Il introduit les approches non linéaires classiques utilisées depuis très longtemps par les hommes pour se localiser, avec l'observation des phares, des étoiles, l'utilisation de la boussole où le comptage du nombre de pas. Bien que la localisation puisse être considérée comme un cas particulier de l'observation d'état, les méthodes spécifiques qui en découlent justifient un chapitre à part. Le chapitre 6 sur l'*identification* cherche à estimer, avec une certaine précision, des quantités non mesurées (paramètres, position) à partir d'autres grandeurs qui elles le sont. Pour effectuer cette identification, nous allons principalement nous intéresser à l'approche dite des *moindres-carrés* qui cherche à trouver le vecteur des inconnues qui minimise la somme des carrés des erreurs. Le chapitre 7 présente le *filtre de Kalman*. Ce filtre peut être vu comme un observateur d'état pour des systèmes linéaires dynamiques à coefficients variant dans le temps.

Les codes MATLAB associés aux exercices de ce livre ainsi que des vidéos explicatives peuvent être trouvés à l'adresse suivante :

www.ensta-bretagne.fr/jaulin/isterob.html

Chapitre 1

Modélisation tridimensionnelle

Ce chapitre s'intéresse à la modélisation tridimensionnelle d'un robot solide (c'est-à-dire non articulé). Une telle modélisation est utilisée pour représenter un avion, un quadri-rotor, un sous-marin, etc. A travers cette modélisation, nous allons introduire certaines notions fondamentales de la robotique comme les notions de représentation d'état, de matrices de rotation ou d'angles d'Euler. Les robots, qu'ils soient mobiles, manipulateurs ou articulés, peuvent généralement se mettre sous la forme d'une représentation d'état :

$$\begin{cases} \dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)) \\ \mathbf{y}(t) = \mathbf{g}(\mathbf{x}(t), \mathbf{u}(t)) \end{cases}$$

où \mathbf{x} est le vecteur d'état, \mathbf{u} le vecteur d'entrée et \mathbf{y} le vecteur des mesures [JAU 05]. Nous appellerons *modélisation* l'étape qui consiste à trouver une représentation d'état, plus ou moins précise, du robot qui nous intéresse. En général, apparaissent dans les équations d'état des paramètres constants (comme la masse, le moment d'inertie d'un corps, le coefficient du frottement visqueux, etc.). Dans de tels cas, une étape d'identification peut s'avérer nécessaire. Nous supposerons que tous les paramètres sont connus. Bien sûr, il n'existe pas de méthodologie systématique à appliquer pour modéliser un robot mobile. Le but de ce chapitre est de présenter des outils pour arriver à une représentation d'état de robots solides tridimensionnels afin de permettre au lecteur d'acquérir une certaine expérience qui l'aidera à modéliser ses propres robots. Cette modélisation nous permettra aussi de rappeler quelques notions importantes de géométrie dans l'espace, qui sont fondamentales en robotique mobile. Ce chapitre débute par un rappel de quelques notions importantes en cinématique qui nous seront utiles pour la modélisation.

1.1. Matrices de rotation

Pour la modélisation tridimensionnelle, il est fondamental de bien maîtriser les notions sur les matrices de rotation, qui sont rappelées dans cette section. C'est avec cet outil que nous allons effectuer nos changements de repère et placer nos objets dans l'espace.

1.1.1. Définition

Rappelons que la $j^{\text{ième}}$ colonne de la matrice d'une application linéaire de $\mathbb{R}^n \rightarrow \mathbb{R}^n$ représente l'image du $j^{\text{ième}}$ vecteur e_j de la base canonique (voir figure 1.1). Ainsi, l'expression d'une matrice de rotation d'angle θ dans le plan \mathbb{R}^2 est bien donnée par :

$$\mathbf{R} = \begin{pmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{pmatrix}$$

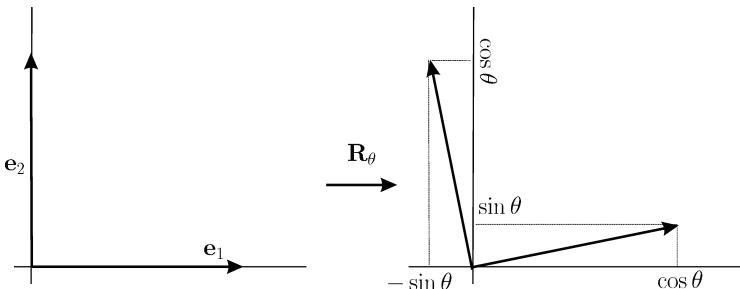


Figure 1.1. Rotation d'angle θ dans un plan

En ce qui concerne les rotations dans l'espace \mathbb{R}^3 (voir figure 1.2), il est important de préciser l'axe de rotation. Nous pouvons distinguer trois rotations principales : la rotation d'angle autour de l'axe Ox , celle autour de Oy et celle autour de Oz .

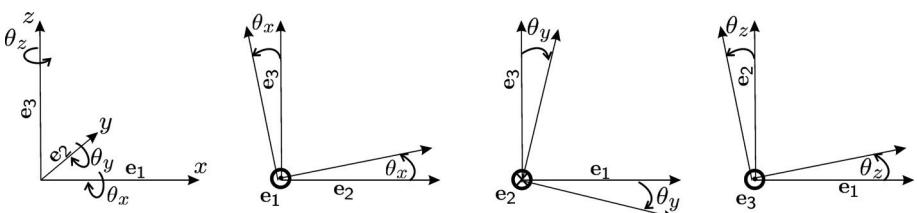


Figure 1.2. Rotations dans \mathbb{R}^3 suivant différents angles de vue

Les matrices associées sont respectivement données par :

$$\mathbf{R}_x = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos \theta_x & -\sin \theta_x \\ 0 & \sin \theta_x & \cos \theta_x \end{pmatrix}, \quad \mathbf{R}_y = \begin{pmatrix} \cos \theta_y & 0 & \sin \theta_y \\ 0 & 1 & 0 \\ -\sin \theta_y & 0 & \cos \theta_y \end{pmatrix},$$

$$\mathbf{R}_z = \begin{pmatrix} \cos \theta_z & -\sin \theta_z & 0 \\ \sin \theta_z & \cos \theta_z & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Rappelons la définition formelle d'une rotation. Une rotation est une application linéaire qui est une isométrie (c'est-à-dire qui conserve le produit scalaire) et un déplacement (c'est-à-dire qui ne change pas l'orientation de l'espace).

THÉORÈME 1.1.– *Une matrice \mathbf{R} est une matrice de rotation si et seulement si :*

$$\mathbf{R}^T \cdot \mathbf{R} = \mathbf{I} \text{ et } \det \mathbf{R} = 1$$

PREUVE.– Le produit scalaire est conservé par \mathbf{R} si quelques soient \mathbf{u} et \mathbf{v} dans \mathbb{R}^n , on a :

$$(\mathbf{R}\mathbf{u})^T \cdot (\mathbf{R}\mathbf{v}) = \mathbf{u}^T \mathbf{R}^T \mathbf{R} \mathbf{v} = \mathbf{u}^T \mathbf{v}$$

c'est-à-dire si $\mathbf{R}^T \mathbf{R} = \mathbf{I}$. Les symétries relativement à un plan, ainsi que tous les autres antidéplacements (isométries qui changent l'orientation de l'espace, comme un miroir), vérifient également la propriété $\mathbf{R}^T \cdot \mathbf{R} = \mathbf{I}$. La condition $\det \mathbf{R} = 1$ nous permet de nous limiter aux isométries qui sont des déplacements. L'ensemble des matrices de rotation de \mathbb{R}^n forme un groupe appelé *groupe spécial orthogonal* (spécial car $\det \mathbf{R} = 1$, orthogonal car $\mathbf{R}^T \cdot \mathbf{R} = \mathbf{I}$).

1.1.2. Vecteur de rotation

Si \mathbf{R} est une matrice de rotation qui dépend du temps t , en dérivant la relation $\mathbf{R}\mathbf{R}^T = \mathbf{I}$, on obtient :

$$\dot{\mathbf{R}} \cdot \mathbf{R}^T + \mathbf{R} \cdot \dot{\mathbf{R}}^T = \mathbf{0}$$

Ainsi, la matrice $\dot{\mathbf{R}} \cdot \mathbf{R}^T$ est une matrice antisymétrique (c'est-à-dire qu'elle satisfait $\mathbf{A}^T = -\mathbf{A}$ et donc sa diagonale ne contient que des éléments nuls et pour tous les éléments de \mathbf{A} , on a $a_{ij} = -a_{ji}$). Nous pouvons donc écrire, dans le cas où \mathbf{R} est de dimension 3×3 :

$$\dot{\mathbf{R}} \cdot \mathbf{R}^T = \begin{pmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{pmatrix} \quad [1.1]$$

Le vecteur $\omega = (\omega_x, \omega_y, \omega_z)$ est appelé *vecteur de rotation* associé au couple $(\mathbf{R}, \dot{\mathbf{R}})$. Il est à noter que $\dot{\mathbf{R}}$ n'est pas une matrice avec des bonnes propriétés (comme par exemple le fait d'être antisymétrique). En revanche, la matrice $\dot{\mathbf{R}} \cdot \mathbf{R}^T$ possède la structure [1.1] car elle permet de se positionner dans le repère où s'effectue le mouvement de rotation et ceci grâce à changement de base effectué par \mathbf{R}^T . Nous définissons le *produit vectoriel* entre deux vecteurs ω et $\mathbf{x} \in \mathbb{R}^3$ comme suit :

$$\begin{aligned}\omega \wedge \mathbf{x} &= \begin{pmatrix} \omega_x \\ \omega_y \\ \omega_z \end{pmatrix} \wedge \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} x_3\omega_y - x_2\omega_z \\ x_1\omega_z - x_3\omega_x \\ x_2\omega_x - x_1\omega_y \end{pmatrix} \\ &= \begin{pmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}\end{aligned}$$

1.1.3. Adjoint

A chaque vecteur $\omega = (\omega_x, \omega_y, \omega_z)$, on peut adjoindre la matrice antisymétrique :

$$\mathbf{Ad}(\omega) \stackrel{\text{def}}{=} \begin{pmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{pmatrix}$$

qui peut être interprétée comme la matrice associée à un produit vectoriel par le vecteur ω .

PROPOSITION 1.1.- *Si $\mathbf{R}(t)$ est une matrice de rotation qui dépend du temps, son vecteur de rotation est donné par :*

$$\omega = \mathbf{Ad}^{-1}(\dot{\mathbf{R}} \cdot \mathbf{R}^T) \quad [1.2]$$

PREUVE.— Cette relation est une conséquence directe de l'équation [1.1]. ■

PROPOSITION 1.2.- *Si \mathbf{R} est une matrice de rotation dans \mathbb{R}^3 et si \mathbf{a} est un vecteur de \mathbb{R}^3 , nous avons :*

$$\mathbf{Ad}(\mathbf{R} \cdot \mathbf{a}) = \mathbf{R} \cdot \mathbf{Ad}(\mathbf{a}) \cdot \mathbf{R}^T \quad [1.3]$$

qui peut aussi s'écrire :

$$(\mathbf{R} \cdot \mathbf{a}) \wedge = \mathbf{R} \cdot (\mathbf{a} \wedge) \cdot \mathbf{R}^T$$

PREUVE.– Soit \mathbf{x} un vecteur de \mathbb{R}^3 . Nous avons :

$$\begin{aligned}\mathbf{Ad}(\mathbf{R} \cdot \mathbf{a}) \cdot \mathbf{x} &= (\mathbf{R} \cdot \mathbf{a}) \wedge \mathbf{x} = (\mathbf{R} \cdot \mathbf{a}) \wedge (\mathbf{R} \cdot \mathbf{R}^T \mathbf{x}) \\ &= \mathbf{R} \cdot (\mathbf{a} \wedge \mathbf{R}^T \cdot \mathbf{x}) = \mathbf{R} \cdot \mathbf{Ad}(\mathbf{a}) \cdot \mathbf{R}^T \cdot \mathbf{x}. \blacksquare\end{aligned}$$

PROPOSITION 1.3.– (*Dualité*). *On a :*

$$\mathbf{R}^T \dot{\mathbf{R}} = \mathbf{Ad}(\mathbf{R}^T \omega) \quad [1.4]$$

Cette relation exprime que la matrice $\mathbf{R}^T \dot{\mathbf{R}}$ est associée au vecteur de rotation ω associé à $\mathbf{R}(t)$ mais exprimé dans le repère associé à \mathbf{R} alors que $\dot{\mathbf{R}} \cdot \mathbf{R}^T$ est associé au même vecteur, mais cette fois exprimé dans le repère de la base canonique.

PREUVE.– On a :

$$\mathbf{R}^T \dot{\mathbf{R}} = \mathbf{R}^T (\dot{\mathbf{R}} \cdot \mathbf{R}^T) \mathbf{R} \stackrel{[1.2]}{=} \mathbf{R}^T \cdot \mathbf{Ad}(\omega) \cdot \mathbf{R} \stackrel{[1.3]}{=} \mathbf{Ad}(\mathbf{R}^T \omega) \blacksquare$$

1.1.4. *Changement de repère*

Soient deux repères $\mathcal{R}_0 : (\mathbf{o}_0, \mathbf{i}_0, \mathbf{j}_0, \mathbf{k}_0)$ et $\mathcal{R}_1 : (\mathbf{o}_1, \mathbf{i}_1, \mathbf{j}_1, \mathbf{k}_1)$ et soit un point \mathbf{u} un vecteur de \mathbb{R}^3 (voir figure 1.3). On a la relation suivante :

$$\begin{aligned}\mathbf{u} &= x_0 \mathbf{i}_0 + y_0 \mathbf{j}_0 + z_0 \mathbf{k}_0 \\ &= x_1 \mathbf{i}_1 + y_1 \mathbf{j}_1 + z_1 \mathbf{k}_1\end{aligned}$$

où (x_0, y_0, z_0) et (x_1, y_1, z_1) sont respectivement les coordonnées de \mathbf{u} dans \mathcal{R}_0 et \mathcal{R}_1 .

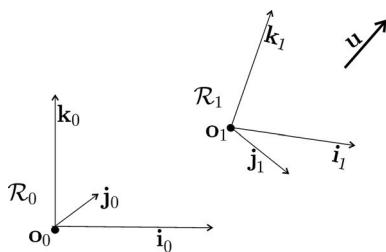


Figure 1.3. *Changement du repère \mathcal{R}_0 vers le repère \mathcal{R}_1*

Ainsi, pour tout vecteur \mathbf{v} nous avons :

$$\langle x_0 \mathbf{i}_0 + y_0 \mathbf{j}_0 + z_0 \mathbf{k}_0, \mathbf{v} \rangle = \langle x_1 \mathbf{i}_1 + y_1 \mathbf{j}_1 + z_1 \mathbf{k}_1, \mathbf{v} \rangle$$

En prenant respectivement $\mathbf{v} = \mathbf{i}_0, \mathbf{j}_0, \mathbf{k}_0$, nous obtenons les trois relations suivantes :

$$\left\{ \begin{array}{l} \langle x_0 \mathbf{i}_0 + y_0 \mathbf{j}_0 + z_0 \mathbf{k}_0, \mathbf{i}_0 \rangle = \langle x_1 \mathbf{i}_1 + y_1 \mathbf{j}_1 + z_1 \mathbf{k}_1, \mathbf{i}_0 \rangle \\ \langle x_0 \mathbf{i}_0 + y_0 \mathbf{j}_0 + z_0 \mathbf{k}_0, \mathbf{j}_0 \rangle = \langle x_1 \mathbf{i}_1 + y_1 \mathbf{j}_1 + z_1 \mathbf{k}_1, \mathbf{j}_0 \rangle \\ \langle x_0 \mathbf{i}_0 + y_0 \mathbf{j}_0 + z_0 \mathbf{k}_0, \mathbf{k}_0 \rangle = \langle x_1 \mathbf{i}_1 + y_1 \mathbf{j}_1 + z_1 \mathbf{k}_1, \mathbf{k}_0 \rangle \end{array} \right.$$

Or comme la base $(\mathbf{i}_0, \mathbf{j}_0, \mathbf{k}_0)$ de \mathcal{R}_0 est orthonormale, $\langle \mathbf{i}_0, \mathbf{i}_0 \rangle = \langle \mathbf{j}_0, \mathbf{j}_0 \rangle = \langle \mathbf{k}_0, \mathbf{k}_0 \rangle = 1$ et $\langle \mathbf{i}_0, \mathbf{j}_0 \rangle = \langle \mathbf{j}_0, \mathbf{k}_0 \rangle = \langle \mathbf{i}_0, \mathbf{k}_0 \rangle = 0$. Ainsi, ces trois équations deviennent :

$$\left\{ \begin{array}{l} x_0 = x_1 \langle \mathbf{i}_1, \mathbf{i}_0 \rangle + y_1 \langle \mathbf{j}_1, \mathbf{i}_0 \rangle + z_1 \langle \mathbf{k}_1, \mathbf{i}_0 \rangle \\ y_0 = x_1 \langle \mathbf{i}_1, \mathbf{j}_0 \rangle + y_1 \langle \mathbf{j}_1, \mathbf{j}_0 \rangle + z_1 \langle \mathbf{k}_1, \mathbf{j}_0 \rangle \\ z_0 = x_1 \langle \mathbf{i}_1, \mathbf{k}_0 \rangle + y_1 \langle \mathbf{j}_1, \mathbf{k}_0 \rangle + z_1 \langle \mathbf{k}_1, \mathbf{k}_0 \rangle \end{array} \right.$$

Soit sous forme matricielle :

$$\begin{pmatrix} x_0 \\ y_0 \\ z_0 \end{pmatrix} = \underbrace{\begin{pmatrix} \langle \mathbf{i}_1, \mathbf{i}_0 \rangle & \langle \mathbf{j}_1, \mathbf{i}_0 \rangle & \langle \mathbf{k}_1, \mathbf{i}_0 \rangle \\ \langle \mathbf{i}_1, \mathbf{j}_0 \rangle & \langle \mathbf{j}_1, \mathbf{j}_0 \rangle & \langle \mathbf{k}_1, \mathbf{j}_0 \rangle \\ \langle \mathbf{i}_1, \mathbf{k}_0 \rangle & \langle \mathbf{j}_1, \mathbf{k}_0 \rangle & \langle \mathbf{k}_1, \mathbf{k}_0 \rangle \end{pmatrix}}_{= \mathbf{R}_{\mathcal{R}_0}^{\mathcal{R}_1}} \cdot \underbrace{\begin{pmatrix} x_1 \\ y_1 \\ z_1 \end{pmatrix}}_{\mathbf{u}|_{\mathcal{R}_1}} \quad [1.5]$$

Nous voyons apparaître une matrice de rotation $\mathbf{R}_{\mathcal{R}_0}^{\mathcal{R}_1}$ dont les colonnes sont les coordonnées de $\mathbf{i}_1, \mathbf{j}_1, \mathbf{k}_1$ exprimées dans le repère absolu \mathcal{R}_0 . C'est-à-dire :

$$\mathbf{R}_{\mathcal{R}_0}^{\mathcal{R}_1} = \left(\begin{array}{c|c|c} & \mathbf{i}_1|_{\mathcal{R}_0} & \mathbf{j}_1|_{\mathcal{R}_0} & \mathbf{k}_1|_{\mathcal{R}_0} \end{array} \right)$$

Cette matrice dépend du temps et lie le repère \mathcal{R}_1 à \mathcal{R}_0 . La matrice $\mathbf{R}_{\mathcal{R}_0}^{\mathcal{R}_1}$ est souvent appelée *matrice cosinus directeur* car ses composantes font intervenir les cosinus directeurs des vecteurs de base des deux repères. De même, si nous avions plusieurs repères $\mathcal{R}_0, \dots, \mathcal{R}_n$ (voir figure 1.4), nous aurions :

$$\mathbf{u}|_{\mathcal{R}_0} = \mathbf{R}_{\mathcal{R}_0}^{\mathcal{R}_1} \cdot \mathbf{R}_{\mathcal{R}_1}^{\mathcal{R}_2} \cdot \dots \cdot \mathbf{R}_{\mathcal{R}_{n-1}}^{\mathcal{R}_n} \cdot \mathbf{u}|_{\mathcal{R}_n}$$

Considérons par exemple la situation d'un robot qui se déplace dans un environnement tridimensionnel. Appelons $\mathcal{R}_0 : (\mathbf{o}_0, \mathbf{i}_0, \mathbf{j}_0, \mathbf{k}_0)$ son repère de référence (par exemple, le repère du robot à l'instant initial). La position du robot est repérée par le vecteur $\mathbf{p}(t)$ exprimé dans \mathcal{R}_0 et son attitude (c'est-à-dire son orientation) par la matrice de rotation $\mathbf{R}(t)$ qui représente les coordonnées des vecteurs $\mathbf{i}_1, \mathbf{j}_1, \mathbf{k}_1$ du repère \mathcal{R}_1 du robot exprimés dans le repère \mathcal{R}_0 , à l'instant t . Ainsi :

$$\mathbf{R}(t) = \left(\begin{array}{c|c|c} & \mathbf{i}_1|_{\mathcal{R}_0} & \mathbf{j}_1|_{\mathcal{R}_0} & \mathbf{k}_1|_{\mathcal{R}_0} \end{array} \right) = \mathbf{R}_{\mathcal{R}_0}^{\mathcal{R}_1}(t)$$

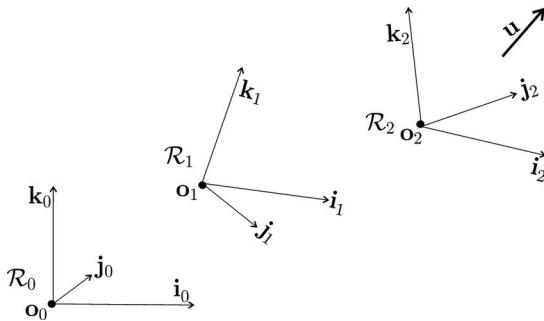


Figure 1.4. Composition dans les changements de repère

Cette matrice peut être retournée par une centrale d'attitude précise positionnée sur le robot. Si le robot est aussi équipé d'un *loch-Doppler* qui lui donne son vecteur vitesse \mathbf{v}_r relativement au sol, ou au fond marin, exprimé dans le repère \mathcal{R}_1 du robot, alors le vecteur vitesse \mathbf{v} du robot satisfait :

$$\underbrace{\mathbf{v}|_{\mathcal{R}_0}}_{\dot{\mathbf{p}}(t)} \stackrel{[1.5]}{=} \underbrace{\mathbf{R}_{\mathcal{R}_0}^{\mathcal{R}_1}}_{\mathbf{R}(t)} \cdot \underbrace{\mathbf{v}|_{\mathcal{R}_1}}_{\mathbf{v}_r(t)}$$

c'est-à-dire :

$$\dot{\mathbf{p}}(t) = \mathbf{R}(t) \cdot \mathbf{v}_r(t) \quad [1.6]$$

Une localisation à l'*estimate* (ou *dead-reckoning*) consiste à intégrer cette équation d'état à partir de la connaissance de $\mathbf{R}(t)$ et $\mathbf{v}_r(t)$.

1.2. Angles d'Euler

1.2.1. Définition

Dans la littérature, les angles proposés par Euler en 1770 pour représenter l'orientation des corps rigides dans l'espace ne sont pas définis de façon unique. On distingue principalement la formulation roulis-lacet-roulis, roulis-tangage-roulis ou roulis-tangage-lacet. C'est cette dernière que nous allons choisir car elle s'est imposée en robotique mobile. Dans cette formulation roulis-tangage-lacet, les angles d'Euler sont parfois appelés *angles de Cardan*. Toute matrice de rotation de \mathbb{R}^3 peut s'exprimer sous la forme du produit de trois matrices comme suit :

$$\mathbf{R}(\psi, \theta, \varphi) = \underbrace{\begin{pmatrix} \cos \psi & -\sin \psi & 0 \\ \sin \psi & \cos \psi & 0 \\ 0 & 0 & 1 \end{pmatrix}}_{\mathbf{R}_\psi} \underbrace{\begin{pmatrix} \cos \theta & 0 & \sin \theta \\ 0 & 1 & 0 \\ -\sin \theta & 0 & \cos \theta \end{pmatrix}}_{\mathbf{R}_\theta} \underbrace{\begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos \varphi & -\sin \varphi \\ 0 & \sin \varphi & \cos \varphi \end{pmatrix}}_{\mathbf{R}_\varphi}$$

ou sous forme développée :

$$\left(\begin{array}{ccc} \cos \theta \cos \psi & -\cos \varphi \sin \psi + \sin \theta \cos \psi \sin \varphi & \sin \psi \sin \varphi + \sin \theta \cos \psi \cos \varphi \\ \cos \theta \sin \psi & \cos \psi \cos \varphi + \sin \theta \sin \psi \sin \varphi & -\cos \psi \sin \varphi + \sin \theta \cos \varphi \sin \psi \\ -\sin \theta & \cos \theta \sin \varphi & \cos \theta \cos \varphi \end{array} \right) \quad \begin{array}{c} \overbrace{\quad \quad \quad}^{\mathbf{i}_1|_{\mathcal{R}_0}} \quad \overbrace{\quad \quad \quad}^{\mathbf{j}_1|_{\mathcal{R}_0}} \quad \overbrace{\quad \quad \quad}^{\mathbf{k}_1|_{\mathcal{R}_0}} \end{array} \quad [1.7]$$

Les angles ψ, θ, φ sont les *angles d'Euler* et sont appelés respectivement le *cap*, *l'assiette* et la *gîte*. Les termes *lacet*, *tangage*, *roulis* sont souvent utilisés bien qu'ils correspondent respectivement à des variations de cap, d'assiette et de gîte.

REMARQUE 1.1.– A partir d'une matrice de rotation \mathbf{R} , on peut retrouver aisément les trois angles d'Euler en résolvant, d'après [1.7], les équations suivantes :

$$\left\{ \begin{array}{ll} -\sin \theta = r_{31} & \cos \theta \cos \varphi = r_{33} \\ \cos \theta \sin \varphi = r_{32} & \cos \theta \sin \psi = r_{21} \\ \cos \theta \cos \psi = r_{11} & \end{array} \right.$$

En imposant $\theta \in [-\frac{\pi}{2}, \frac{\pi}{2}]$, $\varphi \in [-\pi, \pi]$, $\psi \in [-\pi, \pi]$, on trouve :

$$\theta = -\arcsin r_{31}, \varphi = \text{atan2}(r_{32}, r_{33}) \text{ et } \psi = \text{atan2}(r_{21}, r_{11})$$

Ici atan2 est la fonction arc tangente à deux arguments définie par :

$$\theta = \text{atan2}(y, x) \Leftrightarrow \theta \in]-\pi, \pi] \text{ et } \exists r > 0 \mid \begin{cases} x = r \cos \theta \\ y = r \sin \theta \end{cases} \quad [1.8]$$

1.2.2. Dérivée d'une matrice d'Euler

Considérons une matrice de rotation exprimée avec ses angles d'Euler qui dépendent du temps :

$$\mathbf{R}(t) = \mathbf{R}(\psi(t), \theta(t), \varphi(t))$$

Cherchons à exprimer $\dot{\mathbf{R}}(t)$, ou de façon équivalente $\dot{\mathbf{R}}(t)\mathbf{R}^T(t)$. On préfère en effet exprimer cette dérivée dans le repère associé à $\mathbf{R}(t)$. Pour cela, on pourrait bien sûr dériver terme à terme l'expression [1.7] mais le calcul est assez laborieux et de plus, on risque de ne pas arriver à simplifier l'expression obtenue. On a :

$$\begin{aligned} \dot{\mathbf{R}}\mathbf{R}^T &= \frac{d}{dt}(\mathbf{R}_\psi \cdot \mathbf{R}_\theta \cdot \mathbf{R}_\varphi) \cdot \mathbf{R}_\varphi^T \cdot \mathbf{R}_\theta^T \cdot \mathbf{R}_\psi^T \\ &= (\dot{\mathbf{R}}_\psi \cdot \mathbf{R}_\theta \cdot \mathbf{R}_\varphi + \mathbf{R}_\psi \cdot \dot{\mathbf{R}}_\theta \cdot \mathbf{R}_\varphi + \mathbf{R}_\psi \cdot \mathbf{R}_\theta \cdot \dot{\mathbf{R}}_\varphi) \cdot \mathbf{R}_\varphi^T \cdot \mathbf{R}_\theta^T \cdot \mathbf{R}_\psi^T \\ &= \dot{\mathbf{R}}_\psi \cdot \mathbf{R}_\psi^T + \mathbf{R}_\psi \cdot \dot{\mathbf{R}}_\theta \cdot \mathbf{R}_\theta^T \cdot \mathbf{R}_\psi^T + \mathbf{R}_\psi \cdot \mathbf{R}_\theta \cdot \dot{\mathbf{R}}_\varphi \cdot \mathbf{R}_\varphi^T \cdot \mathbf{R}_\theta^T \cdot \mathbf{R}_\psi^T \end{aligned}$$

Or, d'après [1.2], on a :

$$\begin{cases} \dot{\mathbf{R}}_\psi \mathbf{R}_\psi^T = \text{Ad}(\dot{\psi} \mathbf{k}) = \dot{\psi} \text{Ad}(\mathbf{k}) \\ \dot{\mathbf{R}}_\theta \mathbf{R}_\theta^T = \text{Ad}(\dot{\theta} \mathbf{j}) = \dot{\theta} \text{Ad}(\mathbf{j}) \\ \dot{\mathbf{R}}_\varphi \mathbf{R}_\varphi^T = \text{Ad}(\dot{\varphi} \mathbf{i}) = \dot{\varphi} \text{Ad}(\mathbf{i}) \end{cases}$$

Donc :

$$\begin{aligned} \dot{\mathbf{R}} \mathbf{R}^T &= \dot{\psi} \cdot \text{Ad}(\mathbf{k}) + \dot{\theta} \cdot \mathbf{R}_\psi \cdot \text{Ad}(\mathbf{j}) \cdot \mathbf{R}_\psi^T + \dot{\varphi} \cdot \mathbf{R}_\psi \cdot \mathbf{R}_\theta \cdot \text{Ad}(\mathbf{i}) \cdot \mathbf{R}_\theta^T \cdot \mathbf{R}_\psi^T \\ &\stackrel{[1.3]}{=} \dot{\psi} \cdot \text{Ad}(\mathbf{k}) + \dot{\theta} \cdot \text{Ad}(\mathbf{R}_\psi \cdot \mathbf{j}) + \dot{\varphi} \cdot \text{Ad}(\mathbf{R}_\psi \cdot \mathbf{R}_\theta \cdot \mathbf{i}) \end{aligned} \quad [1.9]$$

On remarque la dépendance linéaire en $(\dot{\psi}, \dot{\theta}, \dot{\varphi})$.

1.2.3. Vecteur rotation d'une matrice d'Euler

Considérons un solide se déplaçant dans un repère \mathcal{R}_0 et un repère \mathcal{R}_1 attaché à ce solide (voir figure 1.5). Les conventions choisies ici sont celles de la SNAME (*Society of Naval and Marine Engineers*). Les deux repères sont supposés orthonormés. Soit $\mathbf{R}(t) = \mathbf{R}(\psi(t), \theta(t), \varphi(t))$ la matrice de rotation reliant les deux repères. On cherche à trouver le vecteur de rotation instantané ω du solide relativement à \mathcal{R}_0 en fonction de $\psi, \theta, \varphi, \dot{\psi}, \dot{\theta}, \dot{\varphi}$. On a :

$$\begin{aligned} \omega|_{\mathcal{R}_0} &\stackrel{[1.2]}{=} \text{Ad}^{-1}(\dot{\mathbf{R}} \cdot \mathbf{R}^T) \\ &\stackrel{[1.9]}{=} \text{Ad}^{-1}(\dot{\psi} \cdot \text{Ad}(\mathbf{k}) + \dot{\theta} \cdot \text{Ad}(\mathbf{R}_\psi \cdot \mathbf{j}) + \dot{\varphi} \cdot \text{Ad}(\mathbf{R}_\psi \cdot \mathbf{R}_\theta \cdot \mathbf{i})) \\ &= \dot{\psi} \cdot \mathbf{k} + \dot{\theta} \cdot \mathbf{R}_\psi \cdot \mathbf{j} + \dot{\varphi} \cdot \mathbf{R}_\psi \cdot \mathbf{R}_\theta \cdot \mathbf{i} \end{aligned}$$

Ainsi, après avoir calculé les quantités \mathbf{k} , $\mathbf{R}_\psi \mathbf{j}$ et $\mathbf{R}_\psi \cdot \mathbf{R}_\theta \cdot \mathbf{i}$ dans le repère \mathcal{R}_0 , on a :

$$\omega|_{\mathcal{R}_0} = \dot{\psi} \cdot \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} + \dot{\theta} \cdot \begin{pmatrix} -\sin \psi \\ \cos \psi \\ 0 \end{pmatrix} + \dot{\varphi} \cdot \begin{pmatrix} \cos \theta \cos \psi \\ \cos \theta \sin \psi \\ -\sin \theta \end{pmatrix}$$

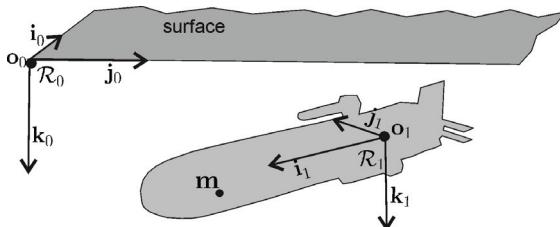


Figure 1.5. Repère $\mathcal{R}_1 : (\mathbf{o}_1, \mathbf{i}_1, \mathbf{j}_1, \mathbf{k}_1)$ fixé au robot

D'où le résultat :

$$\omega|_{\mathcal{R}_0} = \begin{pmatrix} 0 & -\sin \psi & \cos \theta \cos \psi \\ 0 & \cos \psi & \cos \theta \sin \psi \\ 1 & 0 & -\sin \theta \end{pmatrix} \begin{pmatrix} \dot{\psi} \\ \dot{\theta} \\ \dot{\varphi} \end{pmatrix} \quad [1.10]$$

On remarque que cette matrice est singulière lorsque $\cos \theta = 0$. On s'arrangera donc pour ne jamais avoir une assiette θ égale à $\pm \frac{\pi}{2}$.

1.3. Modèle cinématique d'un robot solide

Un robot (avion, sous-marin, bateau) peut être souvent considéré comme un solide dont les entrées sont les accélérations (tangentielle et angulaire). En effet, ces dernières sont des fonctions analytiques des forces qui sont à l'origine du déplacement du robot. Ici, nous allons considérer que les entrées du modèle cinématique sont les accélérations tangentielle et les vitesses angulaires. La raison pour cela est qu'elles sont directement mesurables (si exprimées dans le repère du robot) et que l'on peut considérer qu'elles sont directement commandables (même si une rotation peut prendre un temps non négligeable pour les grosses structures). Le vecteur d'état pour un modèle cinématique est composé du vecteur $\mathbf{p} = (p_x, p_y, p_z)$ donnant les coordonnées du centre du robot exprimées dans le repère inertiel absolu \mathcal{R}_0 , des trois angles d'Euler (ψ, θ, φ) et du vecteur vitesse \mathbf{v}_r du robot exprimé dans son propre repère. Les entrées du système sont d'une part l'accélération $\mathbf{a}_r = \mathbf{a}_{\mathcal{R}_1}$ du centre du robot exprimée dans son propre repère et d'autre part le vecteur $\omega_r = \omega_{\mathcal{R}_1/\mathcal{R}_0|\mathcal{R}_1} = (\omega_x, \omega_y, \omega_z)$ correspondant au vecteur de rotation du robot relativement à \mathcal{R}_0 exprimé dans le repère \mathcal{R}_1 du robot. Il est en effet classique de choisir d'exprimer \mathbf{a}, ω dans le repère du robot car ces quantités sont généralement mesurées par le robot lui-même via des capteurs fixés sur ce dernier. Elles sont donc naturellement exprimées dans le repère du robot. La première des équations d'état est :

$$\ddot{\mathbf{p}} \stackrel{[1.6]}{=} \mathbf{R}(\psi, \theta, \varphi) \cdot \mathbf{v}_r$$

Afin d'exprimer \mathbf{v}_r , dérivons cette équation. Nous obtenons :

$$\ddot{\mathbf{p}} = \dot{\mathbf{R}} \cdot \mathbf{v}_r + \mathbf{R} \cdot \dot{\mathbf{v}}_r$$

avec $\mathbf{R} = \mathbf{R}(\psi, \theta, \varphi)$, c'est-à-dire :

$$\dot{\mathbf{v}}_r = \underbrace{\mathbf{R}^T \cdot \ddot{\mathbf{p}}}_{\mathbf{a}_r} - \mathbf{R}^T \dot{\mathbf{R}} \cdot \mathbf{v}_r \stackrel{[1.4]}{=} \mathbf{a}_r - \mathbf{Ad}(\omega_{\mathcal{R}_1/\mathcal{R}_0|\mathcal{R}_1}) \cdot \mathbf{v}_r$$

Ainsi :

$$\dot{\mathbf{v}}_r = \mathbf{a}_r - \omega_r \wedge \mathbf{v}_r$$

constitue la deuxième équation d'état. Pour terminer, il nous faut aussi exprimer $\dot{\psi}, \dot{\theta}, \dot{\varphi}$ en fonction des variables d'état. La relation :

$$\omega_{|\mathcal{R}_0} = \mathbf{R}(\psi, \theta, \varphi) \cdot \omega_{|\mathcal{R}_1}$$

se traduit, d'après [1.10], par :

$$\begin{pmatrix} 0 & -\sin \psi & \cos \theta \cos \psi \\ 0 & \cos \psi & \cos \theta \sin \psi \\ 1 & 0 & -\sin \theta \end{pmatrix} \begin{pmatrix} \dot{\psi} \\ \dot{\theta} \\ \dot{\varphi} \end{pmatrix} = \mathbf{R}(\psi, \theta, \varphi) \cdot \omega_r$$

En isolant dans cette expression le vecteur $(\dot{\psi}, \dot{\theta}, \dot{\varphi})$, nous obtenons la troisième équation d'état. En réunissant les trois équations d'état, nous obtenons le modèle cinématique suivant pour robot :

$$\begin{cases} \dot{\mathbf{p}} = \mathbf{R}(\psi, \theta, \varphi) \cdot \mathbf{v}_r \\ \dot{\mathbf{v}}_r = \mathbf{a}_r - \omega_r \wedge \mathbf{v}_r \\ \begin{pmatrix} \dot{\psi} \\ \dot{\theta} \\ \dot{\varphi} \end{pmatrix} = \begin{pmatrix} 0 & \frac{\sin \varphi}{\cos \theta} & \frac{\cos \varphi}{\cos \theta} \\ 0 & \cos \varphi & -\sin \varphi \\ 1 & \tan \theta \sin \varphi & \tan \theta \cos \varphi \end{pmatrix} \cdot \omega_r \end{cases} \quad [1.11]$$

Dans un plan horizontal : pour un robot se déplaçant sur un plan horizontal, on a $\varphi = \theta = 0$. La dernière équation nous donne $\dot{\psi} = \omega_{r3}$, $\dot{\theta} = \omega_{r2}$ et $\dot{\varphi} = \omega_{r1}$. Dans un tel cas, il y a une correspondance parfaite entre les composantes de ω_r et les dérivées des angles d'Euler. Il existe des cas singuliers, par exemple quand $\theta = \frac{\pi}{2}$ (c'est le cas lorsque le robot pointe vers le haut), où les dérivées des angles d'Euler ne peuvent pas être définies. L'utilisation du vecteur de rotation est souvent préférée car il n'admet pas de telles singularités.

Localisation à l'estime : pour la localisation à l'estime (c'est-à-dire sans capteur extéroceptifs) on dispose généralement de gyromètres laser qui sont extrêmement précis (de l'ordre de 0.001 deg/s.). Ils utilisent l'effet Sagnac (dans une fibre optique circulaire tournant sur elle-même, le temps que met la lumière pour faire un tour complet dépend du sens de parcours). A l'aide de trois fibres, ces gyromètres génèrent le vecteur $\omega_r = (\omega_x, \omega_y, \omega_z)$. On dispose également d'accéléromètres capables de mesurer l'accélération \mathbf{a}_r avec une très bonne précision. En mode inertiel pur, on se localise en intégrant les équations [1.11] à partir uniquement de l'accélération \mathbf{a}_r et de la vitesse de rotation ω_r toutes les deux exprimées dans le repère du robot. Dans le cas où l'on mesure grâce à un *loch-Doppler* la quantité \mathbf{v}_r (aussi exprimée dans le repère du robot), il nous suffit d'intégrer la première et la dernière de ces trois équations. Enfin, lorsque le robot est un sous-marin correctement lesté ou un robot terrestre qui se déplace dans une zone relativement plane, on sait *a priori* qu'en moyenne la gîte et l'assiette sont nulles. On peut donc incorporer ces informations à travers un filtre de

Kalman pour limiter la dérive de la localisation. Une centrale inertie performante intègre une fusion de toutes les informations disponibles.

Centrale inertie : une centrale inertie pure (sans hybridation et sans prise en compte de la gravité terrestre) représente le robot comme par le modèle cinématique de la figure 1.6, qui reprend lui-même les équations d'état [1.11]. Ce système se met sous la forme $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u})$ où $\mathbf{u} = (\mathbf{a}_r, \omega_r)$ est le vecteur des entrées inertielles mesurées (accélérations et vitesses de rotation vues par un observateur au sol, mais exprimées dans le repère du robot) et $\mathbf{x} = (\mathbf{p}, \mathbf{v}_r, \psi, \theta, \varphi)$ est le vecteur d'état. Pour l'intégrer, on utilise une méthode d'intégration numérique comme la méthode d'Euler. Cette dernière revient à remplacer l'équation différentielle $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u})$ par la récurrence :

$$\mathbf{x}(t + dt) = \mathbf{x}(t) + dt \cdot \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t))$$

$$\begin{aligned} \dot{\mathbf{p}} &= \mathbf{R}(\psi, \theta, \varphi) \cdot \mathbf{v}_r \\ \dot{\mathbf{v}}_r &= \mathbf{a}_r - \boldsymbol{\omega}_r \wedge \mathbf{v}_r \\ \begin{pmatrix} \dot{\psi} \\ \dot{\theta} \\ \dot{\varphi} \end{pmatrix} &= \begin{pmatrix} 0 & \frac{\sin\varphi}{\cos\theta} & \frac{\cos\varphi}{\cos\theta} \\ 0 & \cos\varphi & -\sin\varphi \\ 1 & \tan\theta\sin\varphi & \tan\theta\cos\varphi \end{pmatrix} \cdot \boldsymbol{\omega}_r \end{aligned}$$

Figure 1.6. Modèle cinématique utilisé par une centrale inertie

Modélisation dynamique : pour la modélisation dynamique d'un sous-marin, l'ouvrage de référence est le livre de Fossen [FOS 02]. Pour obtenir un modèle dynamique, il suffit de reprendre les équations de la cinématique, et de considérer que les accélérations angulaires et tangentielles sont engendrées par des forces et des couples. Ces quantités deviennent les nouvelles entrées de notre système. Le lien entre les accélérations et les forces se fait par la deuxième loi Newton (ou le principe fondamental de la dynamique). Ainsi, par exemple, si \mathbf{f} est la résultante des forces extérieures exprimées dans le repère inertiel et m est la masse du robot, on a :

$$m\ddot{\mathbf{p}} = \mathbf{f}$$

Cette relation vue du repère inertiel, mais exprimée dans le repère du robot donne $m\mathbf{a}_r = \mathbf{R}^T \mathbf{f}$ ou encore :

$$\mathbf{a}_r = \frac{1}{m} \mathbf{R}^T \cdot \mathbf{f}$$

On a donc l'accélération tangentielle (qui intervient comme une entrée du modèle cinématique) est une fonction algébrique des forces agissant sur le robot.

1.4. Exercices

Exercice 1.1. Propriétés de la matrice adjointe

Soit le vecteur $\omega = (\omega_x, \omega_y, \omega_z)$ et sa matrice adjointe $\mathbf{Ad}(\omega)$.

1) Montrer que les valeurs propres de $\mathbf{Ad}(\omega)$ sont $\{0, ||\omega||i, -||\omega||i\}$. Donner un vecteur propre associé à 0. Interpréter.

2) Montrer que le vecteur $\mathbf{Ad}(\omega) \mathbf{x} = \omega \wedge \mathbf{x}$ est un vecteur perpendiculaire à ω et \mathbf{x} , tel que le trièdre $(\omega, \mathbf{x}, \omega \wedge \mathbf{x})$ soit direct.

3) Montrer que la norme de $\omega \wedge \mathbf{x}$ est la surface du parallélogramme \mathcal{A} porté par ω et \mathbf{x} .

Exercice 1.2. Identité de Jacobi

L'identité de Jacobi s'écrit sous la forme :

$$\mathbf{a} \wedge (\mathbf{b} \wedge \mathbf{c}) + \mathbf{c} \wedge (\mathbf{a} \wedge \mathbf{b}) + \mathbf{b} \wedge (\mathbf{c} \wedge \mathbf{a}) = \mathbf{0}$$

1) Montrer que cette identité est équivalente à :

$$\mathbf{Ad}(\mathbf{a} \wedge \mathbf{b}) = \mathbf{Ad}(\mathbf{a}) \mathbf{Ad}(\mathbf{b}) - \mathbf{Ad}(\mathbf{b}) \mathbf{Ad}(\mathbf{a})$$

où $\mathbf{Ad}(\omega)$ est la matrice adjointe du vecteur $\omega \in \mathbb{R}^3$.

2) Dans l'espace des matrices antisymétriques, on définit le crochet de Lie comme suit :

$$[\mathbf{A}, \mathbf{B}] = \mathbf{A} \cdot \mathbf{B} - \mathbf{B} \cdot \mathbf{A}$$

Montrer que :

$$\mathbf{Ad}(\mathbf{a} \wedge \mathbf{b}) = [\mathbf{Ad}(\mathbf{a}), \mathbf{Ad}(\mathbf{b})]$$

3) Une *algèbre* est une structure algébrique $(\mathcal{A}, +, \times, \cdot)$ sur un corps \mathbb{K} , si (i) $(\mathcal{A}, +, \cdot)$ est un espace vectoriel sur \mathbb{K} ; (ii) la loi de multiplication \times de $\mathcal{A} \times \mathcal{A} \rightarrow \mathcal{A}$ est distributive à gauche et à droite par rapport à $+$ et (iii) pour tout $\alpha, \beta \in \mathbb{K}$, et tout $x, y \in \mathcal{A}$, $\alpha \cdot x \times \beta \cdot y = (\alpha \beta) \cdot (x \times y)$. Attention, en général, une algèbre est non commutative ($x \times y \neq y \times x$) et non associative ($(x \times y) \times z \neq x \times (y \times z)$). Une *algèbre de Lie* $(\mathcal{G}, +, [\cdot, \cdot])$ est une algèbre non commutative et non associative où la multiplication, notée par un crochet dit de Lie, vérifie (i) $[\cdot, \cdot]$ est bilinéaire, c'est-à-dire linéaire par rapport à chaque variable; (ii) $[x, y] = -[y, x]$ (antisymétrie) et

(iii) $[x, [y, z]] + [y, [z, x]] + [z, [x, y]] = 0$ (relation de Jacobi). Vérifier que l'ensemble $(\mathbb{R}^3, +, \wedge, \cdot)$ forme une *algèbre de Lie*.

Exercice 1.3. Formule de Varignon

Soit un corps solide dont le centre de gravité reste à l'origine d'un repère galiléen et en rotation autour d'un axe Δ avec un vecteur de rotation ω . Donner l'équation de la trajectoire décrite par un point \mathbf{x} du corps.

Exercice 1.4. Formule de Rodrigues

Soit un corps solide dont le centre de gravité reste à l'origine d'un repère galiléen et en rotation autour d'un axe Δ . La position d'un point \mathbf{x} du solide satisfait l'équation d'état (formule de Varignon) :

$$\dot{\mathbf{x}} = \omega \wedge \mathbf{x}$$

où ω est parallèle à l'axe de rotation Δ et $\|\omega\|$ est la vitesse de rotation du solide (en $\text{rad} \cdot \text{s}^{-1}$).

1) Montrer que cette équation d'état peut se mettre sous la forme :

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x}$$

Expliquer pourquoi la matrice \mathbf{A} est souvent notée $\omega \wedge$.

2) Donner l'expression de la solution de l'équation d'état.

3) En déduire que l'expression de la matrice de rotation \mathbf{R} d'angle $\|\omega\|$ autour de ω est donnée par la formule, dite *formule de Rodrigues* :

$$\mathbf{R} = e^{\omega \wedge}$$

4) Calculer les valeurs propres de \mathbf{A} et montrer que ω est le vecteur propre associé à la valeur propre nulle. Interpréter.

5) Quelles sont les valeurs propres de \mathbf{R} ?

6) En utilisant les questions précédentes, donner l'expression d'une rotation autour du vecteur $\omega = (1, 0, 0)$ et d'angle α .

7) Donner un code MATLAB, `eulermat(phi, theta, psi)` utilisant la formule de Rodrigues, retournant la matrice d'Euler.

Exercice 1.5. Formule de Rodrigues, approche géométrique

Considérons la rotation $\mathcal{R}_{\mathbf{n},\varphi}$ d'angle φ autour du vecteur \mathbf{n} unitaire. Soit \mathbf{u} un vecteur quelconque auquel nous allons faire subir cette rotation. Le vecteur \mathbf{u} peut se décomposer comme suit :

$$\mathbf{u} = \underbrace{\langle \mathbf{u}, \mathbf{n} \rangle \cdot \mathbf{n}}_{\mathbf{u}_{\parallel}} + \underbrace{\mathbf{u} - \langle \mathbf{u}, \mathbf{n} \rangle \cdot \mathbf{n}}_{\mathbf{u}_{\perp}}$$

où \mathbf{u}_{\parallel} est colinéaire à \mathbf{n} et \mathbf{u}_{\perp} est dans le plan P_{\perp} orthogonal à \mathbf{n} (voir figure 1.7).

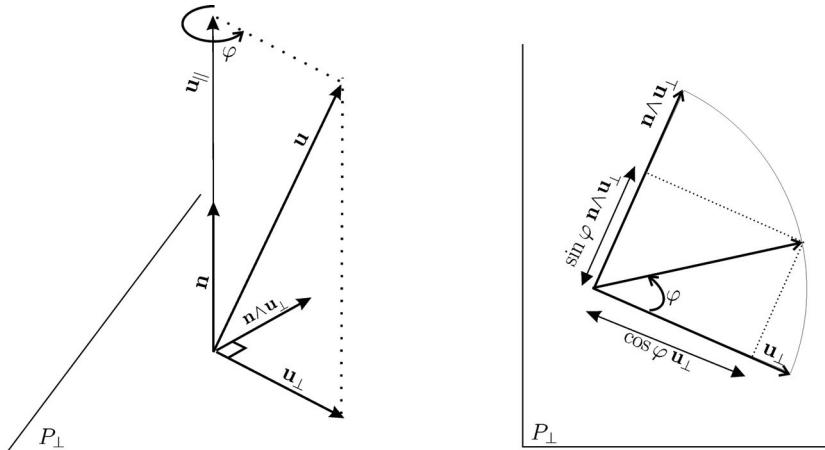


Figure 1.7. Rotation du vecteur \mathbf{u} autour du vecteur \mathbf{n} ; à gauche : vue en perspective ; à droite : vue de dessus

1) Démontrer la formule de Rodrigues donnée par :

$$\mathcal{R}_{\mathbf{n},\varphi}(\mathbf{u}) = \langle \mathbf{u}, \mathbf{n} \rangle \cdot \mathbf{n} + (\cos \varphi) (\mathbf{u} - \langle \mathbf{u}, \mathbf{n} \rangle \cdot \mathbf{n}) + (\sin \varphi) (\mathbf{n} \wedge \mathbf{u})$$

2) En utilisant la formule du double produit vectoriel $\mathbf{a} \wedge (\mathbf{b} \wedge \mathbf{c}) = (\mathbf{a}^T \mathbf{c}) \cdot \mathbf{b} - (\mathbf{a}^T \mathbf{b}) \cdot \mathbf{c}$, sur l'élément $\mathbf{n} \wedge (\mathbf{n} \wedge \mathbf{u})$, montrer que la formule de Rodrigues peut aussi s'écrire :

$$\mathcal{R}_{\mathbf{n},\varphi}(\mathbf{u}) = \mathbf{u} + (1 - \cos \varphi) (\mathbf{n} \wedge (\mathbf{n} \wedge \mathbf{u})) + (\sin \varphi) (\mathbf{n} \wedge \mathbf{u})$$

En déduire que la matrice associée à l'opérateur linéaire $\mathcal{R}_{\mathbf{n},\varphi}$ s'écrit :

$$\begin{aligned} \mathbf{R}_{\mathbf{n},\varphi} &= \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} + (1 - \cos \varphi) \begin{pmatrix} -n_y^2 - n_z^2 & n_x n_y & n_x n_z \\ n_x n_y & -n_x^2 - n_z^2 & n_y n_z \\ n_x n_z & n_y n_z & -n_x^2 - n_y^2 \end{pmatrix} \\ &\quad + (\sin \varphi) \begin{pmatrix} 0 & -n_z & n_y \\ n_z & 0 & -n_x \\ -n_y & n_x & 0 \end{pmatrix} \end{aligned}$$

3) Inversement, on dispose d'une matrice de rotation $\mathbf{R}_{n,\varphi}$ pour laquelle on souhaite retrouver l'axe de rotation \mathbf{n} et l'angle de la rotation φ . Donner une expression pour $\mathbf{R}_{n,\varphi} - \mathbf{R}_{n,\varphi}^T$ et en déduire comment obtenir \mathbf{n} et φ en fonction de $\mathbf{R}_{n,\varphi}$. Pour une illustration géométrique, on pourra s'aider de la figure 1.8.

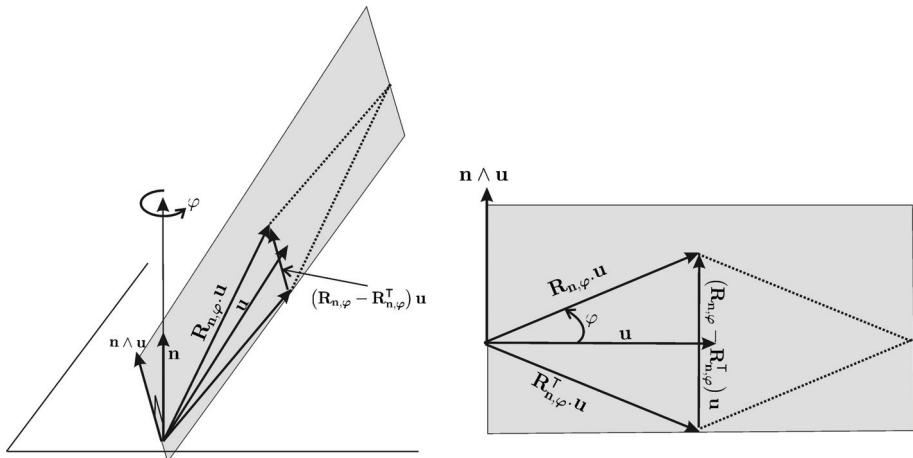


Figure 1.8. A gauche, on a une vision de la rotation d'angle φ autour de \mathbf{n} ; à droite, on visualise la coupe correspondant au losange de Rodrigues

4) En utilisant un développement en série de Mac-Laurin de $\sin\varphi$ et $\cos\varphi$, montrer que :

$$\mathbf{R}_{n,\varphi} = \exp(\varphi \cdot \mathbf{Ad}(\mathbf{n}))$$

relation que l'on écrit parfois sous la forme :

$$\mathbf{R}_{n,\varphi} = \exp(\varphi \cdot \mathbf{n} \wedge)$$

Exercice 1.6. Oscillations de Schuler

Une des composantes fondamentales d'une centrale inertielle est l'inclinomètre. Un tel capteur est censé donner la verticale. Traditionnellement, on utilise un pendule (ou un fil à plomb) pour cela. Or, lorsque l'on bouge, du fait des accélérations, le pendule se met à osciller et le pendule ne peut plus être utilisé comme un indicateur de la verticale. On veut ici concevoir un pendule pour lequel toute accélération horizontale n'engendre pas d'oscillations. Considérons un pendule avec deux masses m à chaque extrémité situées à une distance ℓ_1 et ℓ_2 de l'axe de rotation de la tige (voir figure 1.9). L'axe se déplace à la surface de la terre. On suppose que ℓ_1 et ℓ_2 sont petits devant le rayon r de la terre.

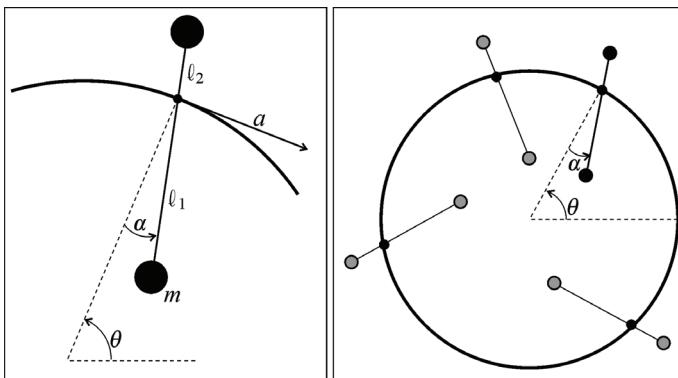


Figure 1.9. Pendule inclinomètre se déplaçant sur la surface de la terre

1) Trouver les équations d'état du système.

2) On suppose que $\alpha = \dot{\alpha} = 0$. Pour quelles valeurs de ℓ_1 et ℓ_2 , le pendule reste à la verticale quels que soient les mouvements horizontaux du pendule ? Que doit-on prendre pour ℓ_2 si l'on choisit $\ell_1 = 1$ m et un rayon de la terre $r = 6\ 400$ km ?

3) On suppose que, suite à des perturbations, le pendule se mette à osciller. La période de ces oscillations est appelée *période de Schuler*. Calculer cette période.

4) Simuler le système avec graphisme en mettant comme entrée (pour l'accélération) un bruit blanc gaussien. Comme le système est conservatif, une méthode d'intégration par Euler ne se comportera pas bien (le pendule prendra de l'énergie). Il conviendra d'utiliser un schéma d'intégration d'ordre plus élevé comme la méthode de *Runge Kutta* donnée par :

$$\mathbf{x}(t+dt) \simeq \mathbf{x}(t) + dt \cdot \left(\frac{\mathbf{f}(\mathbf{x}(t), \mathbf{u}(t))}{4} + \frac{3}{4}\mathbf{f}(\mathbf{x}(t) + \frac{2dt}{3}\mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)), \mathbf{u}(t + \frac{2}{3}dt)) \right)$$

Pour plus de facilité dans la représentation graphique, on prendra $r = 10$ m, $\ell_1 = 1$ m et $g = 10$ ms⁻². Interpréter les résultats.

Exercice 1.7. DéTECTEUR de freinage

Nous allons ici considérer un problème utilisant les changements de base et les matrices de rotation. Une voiture est précédée par un autre véhicule **m** (que nous supposerons ponctuel). A cette voiture, nous attachons le repère \mathcal{R}_1 : $(\mathbf{o}_1, \mathbf{i}_1, \mathbf{j}_1)$ comme représenté sur la figure 1.10. Le repère \mathcal{R}_0 : $(\mathbf{o}_0, \mathbf{i}_0, \mathbf{j}_0)$ est un repère sol supposé fixe.

Cette voiture est munie de capteurs suivants :

- des odomètres placés sur les roues arrières permettent de mesurer la vitesse v du milieu de l'essieu arrière ;
- un gyromètre donne la vitesse angulaire de la voiture $\dot{\theta}$, ainsi que l'accélération angulaire $\ddot{\theta}$;
- un accéléromètre placé en \mathbf{o}_1 permet de mesurer le vecteur accélération (α, β) de \mathbf{o}_1 exprimé dans le repère \mathcal{R}_1 ;
- à l'aide de deux radars positionnés à l'avant, notre voiture est capable de mesurer (indirectement) les coordonnées (a_1, b_1) du point \mathbf{m} dans le repère \mathcal{R}_1 ainsi que les deux premières dérivées (\dot{a}_1, \dot{b}_1) et (\ddot{a}_1, \ddot{b}_1) .

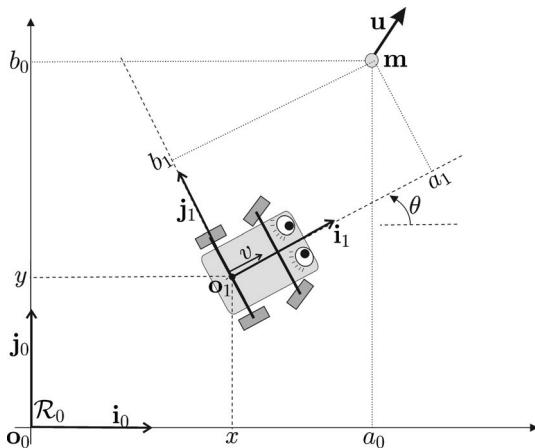


Figure 1.10. Voiture cherchant à détecter si le point \mathbf{m} freine

En revanche, la voiture n'est pas munie d'un système de localisation (de type GPS, par exemple) qui lui permettrait de connaître x, y, \dot{x}, \dot{y} . Elle n'est pas munie de boussole pour avoir l'angle θ . Les quantités en jeu sont les suivantes :

Mesurées $v, \dot{\theta}, a_1, b_1, \alpha, \beta$
 Inconnues $x, y, \dot{x}, \dot{y}, \theta, a_0, b_0$

Si une quantité est mesurée, on supposera que ses dérivées sont aussi mesurées, mais pas leur primitive. Par exemple $\dot{a}_1, \dot{b}_1, \ddot{a}_1, \ddot{b}_1$ sont considérées comme mesurées car a_1, b_1 sont mesurées. En revanche, $\dot{\theta}$ est mesuré, mais pas θ . On ne connaît pas les équations d'état de notre voiture. Le but du problème est de trouver une condition sur les variables mesurées (et leurs dérivées) qui nous permettent de dire si le point \mathbf{m} est en train de freiner ou non. On comprend bien qu'une telle condition nous permettrait de fabriquer un avertisseur qui nous informe que le véhicule qui nous précède freine,

et ceci même si ses feux de freinage arrière sont invisibles (brouillard, remorque non équipée de feux) ou défectueux.

1) En exprimant la relation de Chasle ($\mathbf{o}_0\mathbf{m} = \mathbf{o}_0\mathbf{o}_1 + \mathbf{o}_1\mathbf{m}$) dans le repère \mathcal{R}_0 , montrer la formule de changement de repère :

$$\begin{pmatrix} a_0 \\ b_0 \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix} + \mathbf{R}_\theta \begin{pmatrix} a_1 \\ b_1 \end{pmatrix}$$

où \mathbf{R}_θ est une matrice de rotation.

2) Montrer que $\mathbf{R}_\theta^T \dot{\mathbf{R}}_\theta$ est une matrice antisymétrique et donner son expression.

3) Soit \mathbf{u} le vecteur vitesse du point \mathbf{m} vu par un observateur fixe. Donner une expression $\mathbf{u}_{|\mathcal{R}_1}$ du vecteur vitesse \mathbf{u} exprimé dans le repère \mathcal{R}_1 . Exprimer $\mathbf{u}_{|\mathcal{R}_1}$ en fonction des variables mesurées $v, \dot{\theta}, a_1, \dot{a}_1, b_1, \ddot{b}_1$.

4) Nous allons maintenant utiliser l'accéléromètre de notre véhicule qui nous donne le vecteur accélération (α, β) de \mathbf{o}_1 , exprimé dans \mathcal{R}_1 . En dérivant deux fois $\mathbf{m}_{|\mathcal{R}_0}$, donner l'expression de l'accélération $\mathbf{a}_{|\mathcal{R}_0}$ de \mathbf{m} dans le repère \mathcal{R}_0 . En déduire son expression $\mathbf{a}_{|\mathcal{R}_1}$ dans le repère \mathcal{R}_1 . Donner l'expression de $\mathbf{a}_{|\mathcal{R}_1}$ uniquement en fonction des variables mesurées.

5) Trouver une condition sur les mesures $(v, \dot{\theta}, a_1, b_1, \dot{a}_1, \ddot{b}_1, \alpha, \beta)$ qui permettent de détecter si le véhicule de devant est en train de freiner.

Exercice 1.8. Modélisation d'un robot sous-marin

Le robot que nous allons chercher à modéliser est le Redermor (*coureur des mers* en Breton). Il se trouve représenté sur la figure 1.11. Il s'agit d'un robot sous-marin entièrement autonome. Ce robot, développé par le GESMA (Groupe d'Etude Sous-Marine de l'Atlantique), a une longueur de 6 m, un diamètre de 1 m et un poids de 3 800 kg. Il possède un système de propulsion et de commande très performant dans le but de rechercher des mines dans les fonds de l'océan.

Sur la zone parcourue par le robot, construisons un repère local \mathcal{R}_0 : $(\mathbf{o}_0, \mathbf{i}_0, \mathbf{j}_0, \mathbf{k}_0)$. Le point \mathbf{o}_0 est posé sur la surface de l'océan. Le vecteur \mathbf{i}_0 indique le nord, \mathbf{j}_0 indique l'est et \mathbf{k}_0 est orienté vers le centre de la terre. Soit $\mathbf{p} = (p_x, p_y, p_z)$ les coordonnées du centre du robot exprimées dans le repère \mathcal{R}_0 . Les variables d'état du sous-marin sont sa position \mathbf{p} dans le repère \mathcal{R}_0 , sa vitesse v tangentielle et ses trois angles d'Euler ψ, θ, φ . Ses entrées sont l'accélération tangentielle \dot{v} ainsi que trois gouvernes qui agissent respectivement sur $\omega_x, \omega_y, \omega_z$. Plus formellement, on a :

$$\begin{cases} u_1 = \dot{v} \\ vu_2 = \omega_y \\ vu_3 = \omega_z \\ vu_4 = \omega_x \end{cases}$$



Figure 1.11. Le Redermor construit par le GESMA (Groupe d'Etude Sous-Marine de l'Atlantique), à la surface de l'eau, encore proche du bateau d'où il vient d'être lancé

où le facteur v devant u_1, u_2, u_4 indique que le robot ne peut tourner que s'il avance. Donner le modèle d'état cinématique pour ce système.

Exercice 1.9. Graphisme robot 3D

Le dessin d'objets ou de robots bidimensionnels ou tridimensionnels sur un écran est très utilisé pour la simulation en robotique. La méthode classique (et utilisée par OpenGL) repose sur une modélisation de la pose des objets par une suite de transformations affines (rotations, translations, homothéties) de la forme :

$$\begin{aligned} \mathbf{f}_i : \mathbb{R}^n &\rightarrow \mathbb{R}^n \\ \mathbf{x} &\mapsto \mathbf{A}_i \mathbf{x} + \mathbf{b}_i \end{aligned}$$

avec $n = 2$ ou 3 . Or, la manipulation de compositions de fonctions affines est moins aisée que celle d'applications linéaires. L'idée de la transformation en *coordonnées homogènes* est de transformer un système d'équations affines en système d'équations linéaires. Remarquons tout d'abord qu'une équation affine du type $\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{b}$ peut s'écrire :

$$\begin{pmatrix} \mathbf{y} \\ 1 \end{pmatrix} = \begin{pmatrix} \mathbf{A} & \mathbf{b} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \mathbf{x} \\ 1 \end{pmatrix}$$

Nous définirons donc la *transformation homogène* d'un vecteur comme suit :

$$\mathbf{x} \mapsto \mathbf{x}_h = \begin{pmatrix} \mathbf{x} \\ 1 \end{pmatrix}$$

Ainsi, une équation du type :

$$\mathbf{y} = \mathbf{A}_3 (\mathbf{A}_2 (\mathbf{A}_1 \mathbf{x} + \mathbf{b}_1) + \mathbf{b}_2) + \mathbf{b}_3$$

où intervient la composition de trois transformations affines, pourra s'écrire :

$$\mathbf{y}_h = \begin{pmatrix} \mathbf{A}_3 & \mathbf{b}_3 \\ \mathbf{0} & 1 \end{pmatrix} \begin{pmatrix} \mathbf{A}_2 & \mathbf{b}_2 \\ \mathbf{0} & 1 \end{pmatrix} \begin{pmatrix} \mathbf{A}_1 & \mathbf{b}_1 \\ \mathbf{0} & 1 \end{pmatrix} \mathbf{x}_h$$

Un *motif* est une matrice à deux ou trois lignes (suivant que l'objet est dans le plan ou l'espace) et n colonnes qui représentent les n sommets d'un polygone indéformable, censé représenter l'objet. Il est important que l'union de tous les segments formés par deux points consécutifs du motif forme toutes les arêtes du polygone que l'on souhaite représenter.

1) On considère le robot sous-marin (ou AUV pour *Autonomous Underwater Vehicle*) dont le motif en coordonnées homogènes est le suivant :

$$\begin{pmatrix} 0 & 0 & 10 & 0 & 0 & 10 & 0 & 0 \\ -1 & 1 & 0 & -1 & -0.2 & 0 & 0.2 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

Dessiner ce motif, en perspective sur une feuille de papier.

2) Les équations d'état du robot sont les suivantes :

$$\begin{cases} \dot{p}_x = v \cos \theta \cos \psi \\ \dot{p}_y = v \cos \theta \sin \psi \\ \dot{p}_z = -v \sin \theta \\ \dot{v} = u_1 \\ \dot{\psi} = \frac{\sin \varphi}{\cos \theta} \cdot v \cdot u_2 + \frac{\cos \varphi}{\cos \theta} \cdot v \cdot u_3 \\ \dot{\theta} = \cos \varphi \cdot v \cdot u_2 - \sin \varphi \cdot v \cdot u_3 \\ \dot{\varphi} = -0.1 \sin \varphi + \tan \theta \cdot v \cdot (\sin \varphi \cdot u_2 + \cos \varphi \cdot u_3) \end{cases}$$

où (φ, θ, ψ) sont les trois angles d'Euler. Les entrées du système sont l'accélération tangentielle u_1 , le tangage u_2 et lacet u_3 . Le vecteur d'état est donc $\mathbf{x} = (p_x, p_y, p_z, v, \psi, \theta, \varphi)$. Donner une fonction MATLAB capable de dessiner le robot en 3D, avec son ombre sur le plan x - y . Vérifier que le dessin est correct en bougeant un par un les six degrés de liberté du robot. Utiliser la fonction `plot3` de MATLAB pour obtenir une représentation 3D comme celle illustrée par la figure 1.12.

3) En utilisant la relation :

$$\omega|_{\mathcal{R}_0} = \begin{pmatrix} 0 & -\sin \psi & \cos \theta \cos \psi \\ 0 & \cos \psi & \cos \theta \sin \psi \\ 1 & 0 & -\sin \theta \end{pmatrix} \begin{pmatrix} \dot{\psi} \\ \dot{\theta} \\ \dot{\varphi} \end{pmatrix}$$

tracer le vecteur de rotation instantané du robot.

4) Simuler sous MATLAB le robot dans différentes conditions par une méthode d'Euler.

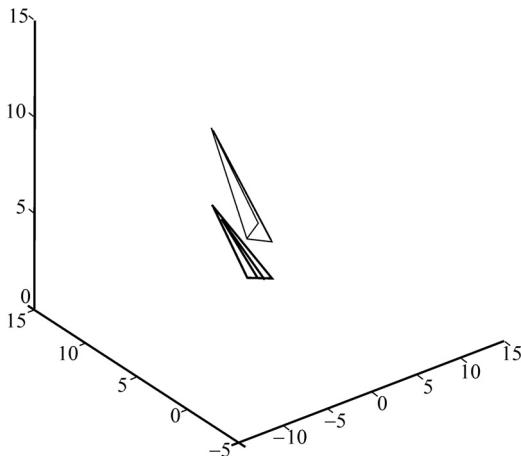


Figure 1.12. Représentation 3D du robot avec son ombre sur le plan horizontal

Exercice 1.10. Robot manipulateur

Un robot manipulateur, tel que *Staubli* représenté sur la figure 1.13, est constitué de plusieurs bras indéformables. On repère les coordonnées de l'organe terminal, à l'extrémité du robot, par une suite de transformations géométriques. On peut montrer qu'une paramétrisation à quatre degrés de liberté permet de représenter ces transformations. Il existe plusieurs paramétrisations possibles avec chacune des avantages et des inconvénients. La plus utilisée est probablement celle de *Denavit-Hartenberg*. Dans le cas où les articulations sont de type rotoïde (comme c'est le cas pour le robot Staubli où les joints sont tournants), la paramétrisation représentée par la figure peut s'avérer pratique car elle facilite le tracé du robot. Cette transformation est la composition de quatre transformations élémentaires : (i) une translation de longueur r suivant z ; (ii) une translation de longueur d suivant x ; (iii) une rotation de α autour de y et (iv) une rotation de θ (la variable actionnée) autour de z . En s'inspirant de la figure pour le tracé des bras et de la photo pour le robot, effectuer sous MATLAB une simulation réaliste du mouvement du robot.

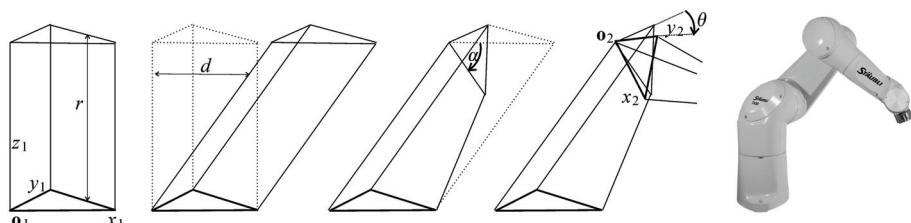


Figure 1.13. Paramétrisation pour le modèle géométrique direct d'un robot manipulateur

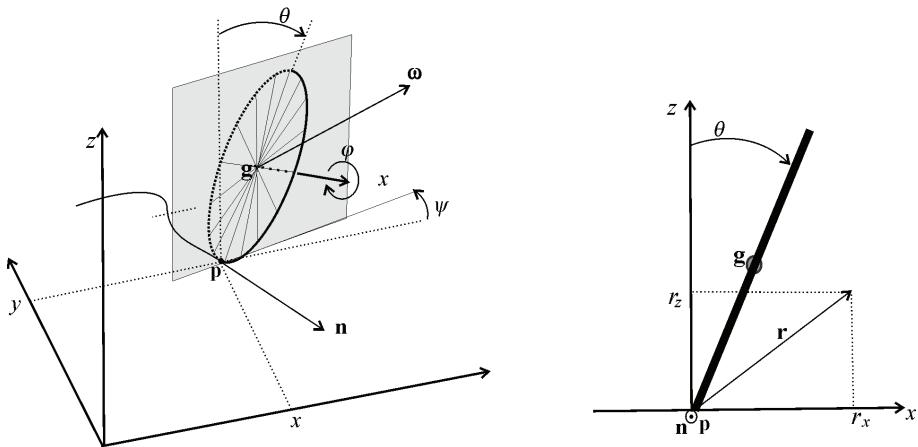


Figure 1.14. Roue roulant sur un plan. A gauche, vue tridimensionnelle ; à droite, coupe dans un plan perpendiculaire à \mathbf{u} suivant le plan en gris

Exercice 1.11. Modélisation tridimensionnelle d'une roue

On considère la roue roulant sur un plan comme sur la figure 1.14 et pour laquelle on souhaiterait obtenir des équations d'état. Dans cette figure \mathbf{u} est le vecteur unitaire indiquant la direction de déplacement du point de contact \mathbf{p} . Le roulement est supposé sans glissement et donc la force de réaction du sol \mathbf{r} est orthogonale à \mathbf{n} . Tout d'abord, nous devons définir les angles d'Euler dans le contexte de la roue, où les notions d'assiette et de gîte n'ont pas grande signification. Choisissons pour ψ l'angle de la projection horizontale de l'axe de la roue (indiquant la direction horizontale à gauche de \mathbf{n}). Pour θ , nous prenons l'inclinaison de la roue et pour φ , l'angle de la roue sur elle-même. Ce choix réside dans le fait que l'angle θ sera bien compris dans l'intervalle $[-\frac{\pi}{2}, \frac{\pi}{2}]$ conformément à ce qui se produit avec les angles d'Euler. Ainsi, les singularités des équations correspondront à des singularités physiques. En effet, la matrice intervenant dans [1.10] est singulière si $\cos \theta = 0$ et dans un tel cas, la roue ne peut plus rouler. On supposera que la roue est pleine, assimilable à un disque homogène, de masse m et de rayon ρ . Sa matrice d'inertie est donnée par :

$$\mathbf{I} = \begin{pmatrix} \frac{m\rho^2}{2} & 0 & 0 \\ 0 & \frac{m\rho^2}{4} & 0 \\ 0 & 0 & \frac{m\rho^2}{4} \end{pmatrix}$$

Donner les équations d'état de cette roue.

2) Supposons que nous puissions déplacer des masses dans le plan de la roue uniquement, tout en conservant la symétrie cylindrique de la roue. Le centre de gravité est donc toujours au centre de la roue et nous pouvons influencer le vecteur de rotation

ω seulement suivant l'axe de la roue. Est-il possible de contrôler la trajectoire de la roue ainsi que sa vitesse ?

Exercice 1.12. Stabilité mécanique d'un robot sous-marin

Considérons un corps homogène dans l'eau de même densité que l'eau. Son centre de gravité est noté g . On pose sur ce corps en un point noté a une masse ponctuelle (donc de densité infinie) exerçant une force f , comme illustré par la figure 1.15 à gauche.

1) Quelle est la condition d'équilibre en rotation (c'est-à-dire qui n'engendre pas une rotation du corps) ?

2) Supposons que l'on ait cet équilibre en rotation. Sous quelle condition a-t-on un équilibre stable ?

3) On considère un robot sous-marin que l'on souhaite voir évoluer dans un plan horizontal. A l'équilibre sous l'eau, on remarque que le robot est légèrement penché en avant comme sur la figure 1.15 à droite. Que faire pour remédier à cela ?

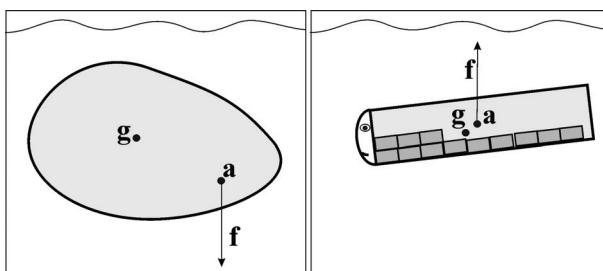


Figure 1.15. A gauche : corps immergé homogène ; à droite : robot mal équilibré

4) Le robot est maintenant en équilibre à l'horizontal et c'est ce que l'on souhaite. En revanche, cet équilibre est instable, c'est-à-dire que le robot tend à se retourner dès que nous l'écartons de sa position d'équilibre. Que faire pour avoir la stabilité ?

5) Le robot est désormais bien équilibré relativement à la force d'Archimède. On lui donne une vitesse et on le lâche à l'horizontal. Le robot plonge. Pourquoi ? Que faire pour lui éviter de plonger ?

6) Le robot est maintenant équilibré et stable relativement à la poussée d'Archimède et à la trainée. On agit maintenant sur le propulseur, mais le robot plonge à nouveau. Pourquoi ? Que faire pour qu'il ne plonge pas ?

7) Le robot en mode régulé en cap grâce à sa boussole, cherche à se mettre proche d'un mur en béton et parallèle à ce dernier. Il se met alors à osciller fortement tout en restant dans un plan horizontal. Pourquoi ? Comment éviter cela ?

8) La figure 1.16 représente un robot entièrement équilibré avec différentes configurations pour la position de ses propulseurs. La configuration (a) est-elle plus stable ou plus manœuvrante que la configuration en (b) ? La configuration (c) est-elle plus stable ou plus manœuvrante que la configuration en (d) ?

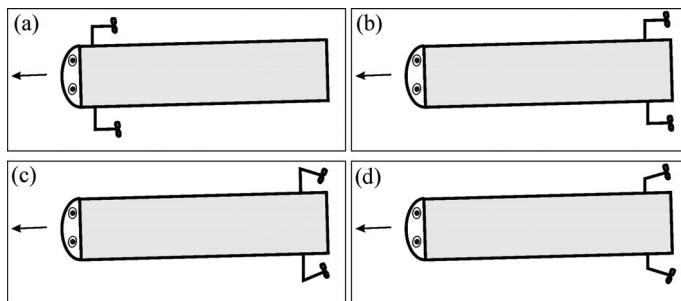


Figure 1.16. Un robot sous-marin vu de dessus avec différentes configurations pour ses propulseurs

1.5. Corrections

Correction de l'exercice 1.1 (propriétés de la matrice adjointe)

1) Le polynôme caractéristique de la matrice $\mathbf{Ad}(\omega)$ se calcule relativement facilement. Il est donné par :

$$s^3 + (\omega_x^2 + \omega_y^2 + \omega_z^2) s = s (s^2 + (\omega_x^2 + \omega_y^2 + \omega_z^2))$$

D'où les valeurs propres $\{0, ||\omega||i, -||\omega||i\}$. Enfin, on a :

$$\begin{pmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{pmatrix} \begin{pmatrix} \omega_x \\ \omega_y \\ \omega_z \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

et donc le vecteur propre associé à 0 est ω . La matrice $\mathbf{Ad}(\omega)$ est associée à un champ de vecteurs vitesse d'un repère tournant avec ω . L'axe ω ne bougeant, on aura $\mathbf{Ad}(\omega) \cdot \omega = \mathbf{0}$.

2) (i) Nous allons montrer que $\mathbf{x} \perp (\omega \wedge \mathbf{x})$. Pour cela, il nous suffit de prouver que $\mathbf{x}^T \mathbf{Ad}(\omega) \mathbf{x} = 0$. Ainsi, nous aurons $\mathbf{x} \perp \mathbf{Ad}(\omega) \mathbf{x}$. On a :

$$\begin{aligned} 2\mathbf{x}^T \mathbf{Ad}(\omega) \mathbf{x} &= \mathbf{x}^T \mathbf{Ad}(\omega) \mathbf{x} + \mathbf{x}^T \mathbf{Ad}^T(\omega) \mathbf{x} \quad (\text{car scalaires}) \\ &= \mathbf{x}^T (\mathbf{Ad}(\omega) + \mathbf{Ad}^T(\omega)) \mathbf{x} \\ &= 0 \quad (\text{car } \mathbf{Ad}(\omega) \text{ est antisymétrique}) \end{aligned}$$

(ii) Nous allons montrer que $\omega \perp (\omega \wedge \mathbf{x})$. On a :

$$\omega^T \mathbf{Ad}(\omega) \mathbf{x} = \mathbf{x}^T \mathbf{Ad}^T(\omega) \omega$$

Or ω est le vecteur propre associé à 0 de la matrice $\mathbf{Ad}(\omega)$ et donc aussi pour sa transposée. Donc $\mathbf{Ad}^T(\omega) \cdot \omega = \mathbf{0}$, ce qui nous donne $\omega \perp (\mathbf{Ad}^T(\omega) \cdot \omega) = 0$.

(iii) On montre facilement que :

$$\det \begin{pmatrix} & & \\ \omega & \mathbf{x} & \omega \wedge \mathbf{x} \\ & & \end{pmatrix} = \|\omega \wedge \mathbf{x}\|^2$$

Pour cela, il faut développer les deux expressions ci-dessus et vérifier l'égalité. La positivité du déterminant implique que le trièdre $(\omega, \mathbf{x}, \omega \wedge \mathbf{x})$ est direct.

3) Le parallélépipède porté par ω, \mathbf{x} et $\omega \wedge \mathbf{x}$ a pour volume :

$$v = \det \begin{pmatrix} & & \\ \omega & \mathbf{x} & \omega \wedge \mathbf{x} \\ & & \end{pmatrix} = \|\omega \wedge \mathbf{x}\|^2$$

Or, puisque $\omega \wedge \mathbf{x}$ est orthogonal à \mathbf{x} et à ω , le volume du parallélépipède est la surface de sa base \mathcal{A} multipliée par sa hauteur $h = \|\omega \wedge \mathbf{x}\|$, c'est-à-dire :

$$v = \mathcal{A} \cdot \|\omega \wedge \mathbf{x}\|$$

En égalisant les deux expressions pour v obtient, $\mathcal{A} = \|\omega \wedge \mathbf{x}\|$.

Correction de l'exercice 1.2 (identité de Jacobi)

1) On a :

$$\underbrace{\mathbf{a} \wedge (\mathbf{b} \wedge \mathbf{c})}_{=\mathbf{Ad}(\mathbf{a})\mathbf{Ad}(\mathbf{b})\cdot\mathbf{c}} + \underbrace{\mathbf{c} \wedge (\mathbf{a} \wedge \mathbf{b})}_{=-(\mathbf{a}\wedge\mathbf{b})\wedge\mathbf{c}=-\mathbf{Ad}(\mathbf{a}\wedge\mathbf{b})\cdot\mathbf{c}} + \underbrace{\mathbf{b} \wedge (\mathbf{c} \wedge \mathbf{a})}_{=-\mathbf{b}\wedge(\mathbf{a}\wedge\mathbf{c})=-\mathbf{Ad}(\mathbf{b})\mathbf{Ad}(\mathbf{a})\cdot\mathbf{c}} = \mathbf{0}$$

On a donc pour tout \mathbf{c} :

$$\mathbf{Ad}(\mathbf{a}) \mathbf{Ad}(\mathbf{b}) \cdot \mathbf{c} - \mathbf{Ad}(\mathbf{a} \wedge \mathbf{b}) \cdot \mathbf{c} - \mathbf{Ad}(\mathbf{b}) \mathbf{Ad}(\mathbf{a}) \cdot \mathbf{c} = \mathbf{0}$$

c'est-à-dire :

$$\mathbf{Ad}(\mathbf{a} \wedge \mathbf{b}) = \mathbf{Ad}(\mathbf{a}) \mathbf{Ad}(\mathbf{b}) - \mathbf{Ad}(\mathbf{b}) \mathbf{Ad}(\mathbf{a})$$

2) On a tout simplement :

$$\mathbf{Ad}(\mathbf{a} \wedge \mathbf{b}) = \mathbf{Ad}(\mathbf{a}) \mathbf{Ad}(\mathbf{b}) - \mathbf{Ad}(\mathbf{b}) \mathbf{Ad}(\mathbf{a}) = [\mathbf{Ad}(\mathbf{a}), \mathbf{Ad}(\mathbf{b})]$$

3) La vérification est triviale et nous ne la donnerons pas ici. Notons que de ce résultat nous pouvons en déduire que l'ensemble des matrices antisymétriques muni de l'addition, du crochet et de la multiplication externe classique est également une algèbre de Lie.

Correction de l'exercice 1.3 (formule de Varignon)

La position d'un point \mathbf{x} du solide satisfait l'équation d'état :

$$\dot{\mathbf{x}} = \mathbf{Ad}(\omega) \cdot \mathbf{x}$$

où ω est parallèle à l'axe de rotation Δ et $||\omega||$ est la vitesse de rotation du solide (en rad.s^{-1}). Après intégration de cette équation d'état linéaire, on trouve :

$$\mathbf{x}(t) = e^{\mathbf{Ad}(\omega)t} \cdot \mathbf{x}(0)$$

On aurait pu aussi obtenir cette formule en appliquant la formule de Rodrigues étudiée dans l'exercice 1.4. Cette propriété s'interprète par le fait que $\mathbf{Ad}(\omega)$ représente un mouvement de rotation, alors que son exponentielle représente le résultat de ce mouvement (c'est-à-dire une rotation).

Correction de l'exercice 1.4 (formule de Rodrigues)

1) Il suffit de vérifier que :

$$\omega \wedge \mathbf{x} = \begin{pmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{pmatrix} \mathbf{x}$$

2) La solution de l'équation d'état est :

$$\mathbf{x}(t) = e^{\mathbf{A}t} \mathbf{x}(0)$$

3) A l'instant t le corps solide a tourné d'un angle égal à $||\omega|| \cdot t$ et donc pour $t = 1$, il a tourné de $||\omega||$. Ainsi, la rotation \mathbf{R} d'angle $||\omega||$ autour de ω est donnée par :

$$\mathbf{R} = e^{\mathbf{A}} = e^{\omega \wedge}$$

4) Le polynôme caractéristique de \mathbf{A} est $s^3 + (\omega_x^2 + \omega_y^2 + \omega_z^2)s$. Les valeurs propres de \mathbf{A} sont $0, i||\omega||, -i||\omega||$. Les vecteurs propres associés à la valeur propre 0

sont colinéaires à ω . Ceci est logique car les points de l'axe de rotation ont une vitesse nulle.

5) Les valeurs propres de \mathbf{R} s'obtiennent par le théorème de correspondance des valeurs propres et sont donc $e^0, e^{i\|\omega\|}, e^{-i\|\omega\|}$.

6) L'expression d'une rotation autour du vecteur $\omega = (1, 0, 0)$ et d'angle α est :

$$\mathbf{R} = \exp \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & -\alpha \\ 0 & \alpha & 0 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos \alpha & -\sin \alpha \\ 0 & \sin \alpha & \cos \alpha \end{pmatrix}$$

7) La formule de Rodrigues nous dit que la matrice de rotation autour du vecteur ω et d'angle $\varphi = \|\omega\|$ est donnée par :

$$\mathbf{R}_\omega = \exp \begin{pmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{pmatrix}$$

Le script MATLAB nous rentrant la matrice d'Euler est donc le suivant :

```
eulermat(phi,theta,psi)
Apsi=psi*[0 -1 0 ;1 0 0 ;0 0 0];
Atheta=theta*[0 0 1 ;0 0 0 ;-1 0 0];
Aphi=phi*[0 0 0 ;0 0 -1 ;0 1 0];
R=expm(Apsi)*expm(Atheta)*expm(Aphi);
```

Correction de l'exercice 1.5 (formule de Rodrigues, approche géométrique)

1) On a :

$$\begin{aligned} \mathcal{R}_{\mathbf{n},\varphi}(\mathbf{u}) &= \mathcal{R}_{\mathbf{n},\varphi}(\mathbf{u}_{\parallel} + \mathbf{u}_{\perp}) \\ &= \mathcal{R}_{\mathbf{n},\varphi}(\mathbf{u}_{\parallel}) + \mathcal{R}_{\mathbf{n},\varphi}(\mathbf{u}_{\perp}) \text{ (par linéarité de l'opérateur de rotation)} \\ &= \mathbf{u}_{\parallel} + (\cos \varphi) \mathbf{u}_{\perp} + (\sin \varphi) (\mathbf{n} \wedge \mathbf{u}_{\perp}) \text{ (voir figure de droite)} \\ &= \langle \mathbf{u}, \mathbf{n} \rangle \cdot \mathbf{n} + (\cos \varphi) (\mathbf{u} - \langle \mathbf{u}, \mathbf{n} \rangle \cdot \mathbf{n}) + (\sin \varphi) (\mathbf{n} \wedge (\mathbf{u} - \langle \mathbf{u}, \mathbf{n} \rangle \cdot \mathbf{n})) \\ &= \langle \mathbf{u}, \mathbf{n} \rangle \cdot \mathbf{n} + (\cos \varphi) (\mathbf{u} - \langle \mathbf{u}, \mathbf{n} \rangle \cdot \mathbf{n}) + (\sin \varphi) (\mathbf{n} \wedge \mathbf{u}) \end{aligned}$$

D'où la formule de Rodrigues :

$$\mathcal{R}_{\mathbf{n},\varphi}(\mathbf{u}) = (\cos \varphi) \cdot \mathbf{u} + (1 - \cos \varphi) (\langle \mathbf{u}, \mathbf{n} \rangle \cdot \mathbf{n}) + (\sin \varphi) (\mathbf{n} \wedge \mathbf{u})$$

2) On a :

$$\mathbf{n} \wedge (\mathbf{n} \wedge \mathbf{u}) = (\mathbf{n}^T \mathbf{u}) \cdot \mathbf{n} - (\mathbf{n}^T \mathbf{n}) \cdot \mathbf{u} = (\mathbf{n} \cdot \mathbf{u}^T) \mathbf{n} - \|\mathbf{n}\|^2 \mathbf{u} = \langle \mathbf{u}, \mathbf{n} \rangle \mathbf{n} - \|\mathbf{n}\|^2 \mathbf{u}$$

Donc :

$$\langle \mathbf{u}, \mathbf{n} \rangle \cdot \mathbf{n} = \mathbf{n} \wedge (\mathbf{n} \wedge \mathbf{u}) + \mathbf{u}$$

La formule de Rodrigues peut donc aussi s'écrire :

$$\begin{aligned}\mathcal{R}_{\mathbf{n},\varphi}(\mathbf{u}) &= \mathbf{n} \wedge (\mathbf{n} \wedge \mathbf{u}) + \mathbf{u} + (\cos \varphi) (\mathbf{u} - \mathbf{n} \wedge (\mathbf{n} \wedge \mathbf{u}) + \mathbf{u}) + (\sin \varphi) (\mathbf{n} \wedge \mathbf{u}) \\ &= \mathbf{u} + (1 - \cos \varphi) (\mathbf{n} \wedge (\mathbf{n} \wedge \mathbf{u})) + (\sin \varphi) (\mathbf{n} \wedge \mathbf{u})\end{aligned}$$

L'opérateur $\mathcal{R}_{\mathbf{n},\varphi}$ peut être représenté par la matrice de rotation suivante :

$$\mathbf{R}_{\mathbf{n},\varphi} = \mathbf{I} + (1 - \cos \varphi) (\mathbf{Ad}^2(\mathbf{n})) + (\sin \varphi) (\mathbf{Ad}(\mathbf{n}))$$

ou, sous forme développée :

$$\begin{aligned}\mathbf{R}_{\mathbf{n},\varphi} &= \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} + (1 - \cos \varphi) \begin{pmatrix} -n_y^2 - n_z^2 & n_x n_y & n_x n_z \\ n_x n_y & -n_x^2 - n_z^2 & n_y n_z \\ n_x n_z & n_y n_z & -n_x^2 - n_y^2 \end{pmatrix} \\ &\quad + (\sin \varphi) \begin{pmatrix} 0 & -n_z & n_y \\ n_z & 0 & -n_x \\ -n_y & n_x & 0 \end{pmatrix}\end{aligned}$$

3) On a :

$$\mathbf{R}_{\mathbf{n},\varphi} - \mathbf{R}_{\mathbf{n},\varphi}^T = 2 (\sin \varphi) \begin{pmatrix} 0 & -n_z & n_y \\ n_z & 0 & -n_x \\ -n_y & n_x & 0 \end{pmatrix}$$

Les vecteurs $\mathbf{R}_{\mathbf{n},\varphi} \cdot \mathbf{u}$ et $\mathbf{R}_{\mathbf{n},\varphi}^T \cdot \mathbf{u}$ forment les deux côtés d'un losange (le losange de Rodrigues) dont le vecteur :

$$(\mathbf{R}_{\mathbf{n},\varphi} - \mathbf{R}_{\mathbf{n},\varphi}^T) \mathbf{u} = 2 (\sin \varphi) \cdot \mathbf{n} \wedge \mathbf{u}$$

correspond à la diagonale de ce losange.

4) Rappelons que le développement en série de Mac-Laurin des fonctions sinus et cosinus s'écrit :

$$\begin{aligned}\sin \varphi &= \varphi - \frac{\varphi^3}{3!} + \frac{\varphi^5}{5!} - \frac{\varphi^7}{7!} + \dots \\ \cos \varphi &= 1 - \frac{\varphi^2}{2!} + \frac{\varphi^4}{4!} - \frac{\varphi^6}{6!} + \dots\end{aligned}$$

Posons $\mathbf{H} = \mathbf{Ad}(\mathbf{n})$. Puisque \mathbf{n} est un vecteur propre de \mathbf{H} associé à la valeur propre nulle, on a $\mathbf{H} (\mathbf{n} \cdot \mathbf{n}^T) = \mathbf{0}$. De plus :

$$\mathbf{H}^2 = (\mathbf{n} \cdot \mathbf{n}^T - \mathbf{I})$$

Donc :

$$\begin{aligned}\mathbf{H}^3 &= \mathbf{H} \cdot (\mathbf{n} \cdot \mathbf{n}^T - \mathbf{I}) = -\mathbf{H} \\ \mathbf{H}^4 &= \mathbf{H} \cdot \mathbf{H}^3 = -\mathbf{H}^2 \\ \mathbf{H}^5 &= \mathbf{H} \cdot \mathbf{H}^4 = \mathbf{H} (-\mathbf{H}^2) = -\mathbf{H}^3 = \mathbf{H} \\ \mathbf{H}^6 &= \mathbf{H} \cdot \mathbf{H}^5 = \mathbf{H}^2 \\ \mathbf{H}^7 &= \mathbf{H} \cdot \mathbf{H}^6 = \mathbf{H}^3 = -\mathbf{H} \dots\end{aligned}$$

Ainsi, la formule de Rodrigues s'écrit :

$$\begin{aligned}
 \mathbf{R}_{\mathbf{n},\varphi} &= \mathbf{I} + (\sin \varphi) \cdot \mathbf{H} + (1 - \cos \varphi) \cdot \mathbf{H}^2 \\
 &= \mathbf{I} + \left(\varphi - \frac{\varphi^3}{3!} + \frac{\varphi^5}{5!} - \frac{\varphi^7}{7!} + \dots \right) \cdot \mathbf{H} + \left(\frac{\varphi^2}{2!} - \frac{\varphi^4}{4!} + \frac{\varphi^6}{6!} + \dots \right) \cdot \mathbf{H}^2 \\
 &= \mathbf{I} + \varphi \cdot \mathbf{H} + \frac{\varphi^2}{2!} \cdot \mathbf{H}^2 - \frac{\varphi^3}{3!} \cdot \mathbf{H} - \frac{\varphi^4}{4!} \cdot \mathbf{H}^2 + \frac{\varphi^5}{5!} \cdot \mathbf{H} + \frac{\varphi^6}{6!} \cdot \mathbf{H}^2 - \frac{\varphi^7}{7!} \cdot \mathbf{H} + \dots \\
 &= \mathbf{I} + \varphi \cdot \mathbf{H} + \frac{\varphi^2}{2!} \cdot \mathbf{H}^2 + \frac{\varphi^3}{3!} \cdot \mathbf{H}^3 + \frac{\varphi^4}{4!} \cdot \mathbf{H}^4 + \frac{\varphi^5}{5!} \cdot \mathbf{H}^5 + \frac{\varphi^6}{6!} \cdot \mathbf{H}^6 + \dots \\
 &= \exp(\varphi \cdot \mathbf{H})
 \end{aligned}$$

C'est-à-dire :

$$\mathbf{R}_{\mathbf{n},\varphi} = \exp(\varphi \cdot \mathbf{Ad}(\mathbf{n})) = \exp(\varphi \cdot \mathbf{n} \wedge)$$

Correction de l'exercice 1.6 (oscillations de Schuler)

1) Le vecteur d'état est $\mathbf{x} = (\theta, \alpha, \dot{\theta}, \dot{\alpha})$. Pour avoir un déplacement horizontal, il nous faut une force f horizontale. D'après le principe fondamental de la dynamique, on a :

$$J(\ddot{\theta} + \ddot{\alpha}) = -mg\ell_1 \sin \alpha + mg\ell_2 \sin \alpha - f \frac{\ell_1 - \ell_2}{2} \cos \alpha$$

avec $J = m(\ell_1^2 + \ell_2^2)$ et $f = 2ma$. Comme $\ddot{\theta} = \frac{a}{r}$, les équations d'état du système s'écrivent :

$$\begin{cases} \dot{\theta} = \dot{\theta} \\ \dot{\alpha} = \dot{\alpha} \\ \ddot{\theta} = \frac{a}{r} \\ \ddot{\alpha} = -\frac{a}{r} + \frac{\ell_2 - \ell_1}{\ell_1^2 + \ell_2^2} (g \sin \alpha - a \cos \alpha) \end{cases}$$

2) Le pendule reste à l'horizontale si, pour $\alpha = 0$, on a $\ddot{\alpha} = 0$. C'est-à-dire :

$$-\frac{a}{r} + \frac{\ell_2 - \ell_1}{\ell_1^2 + \ell_2^2} (g \sin \alpha - a \cos \alpha) = 0$$

ou de façon équivalente :

$$-\frac{\ell_2 - \ell_1}{\ell_1^2 + \ell_2^2} = \frac{1}{r}$$

Il nous faut donc satisfaire l'équation $\ell_2^2 + r\ell_2 - \ell_1r + \ell_1^2 = 0$. La résolution nous donne :

$$\ell_2 = \frac{-r + \sqrt{r^2 + 4(\ell_1r - \ell_1^2)}}{2}$$

Pour $\ell_1 = 1$, on obtient :

$$\ell_2 = \frac{-64 \cdot 10^5 + \sqrt{(64 \cdot 10^5)^2 + 4(64 \cdot 10^5 - 1)}}{2} = 1 - 3.1 \times 10^{-7} \text{ m}$$

3) L'équation décrivant les oscillations est :

$$\ddot{\alpha} = -\frac{a}{r} - \frac{1}{r} (g \sin \alpha - a \cos \alpha)$$

Pour $a = 0$, on a :

$$\ddot{\alpha} = -\frac{g}{r} \sin \alpha$$

qui est l'équation d'un pendule de longueur $\ell = r$. En linéarisant cette équation, on obtient le polynôme caractéristique $s^2 + \frac{g}{r}$ et donc la pulsation $\omega = \sqrt{\frac{g}{r}}$. La période de Schuler est donc :

$$T = 2\pi \sqrt{\frac{r}{g}} = 5072 \text{ sec} = 84 \text{ min}$$

4) Le programme, rangé dans **schuler.m**, est le suivant :

```
r=10 ; l1=1 ; l2=(-r+sqrt(r^2+4*(l1*r-l1^2)))/2 ;
dt=0.05 ; x=[1 ; 0.1 ; 0] ;
for t=0 :dt :10,
a=randn(1) ;
x=x+dt*(0.25*f(x,a)+0.75*(f(x+dt*(2/3)*f(x,a),a))) ;
end
```

On remarque que pour une initialisation à $\alpha = \dot{\alpha} = 0$, le pendule pointe toujours vers le centre de la terre. Sinon, il oscille et conserve cette oscillation à la fréquence de Schuler. Cette oscillation peut être observée dans les centrales inertielles modernes et il existe des moyens de la compenser grâce à des informations collectées par les autres capteurs inertIELS.

Correction de l'exercice 1.7 (détecteur de freinage)

1) On a :

$$\begin{aligned} \mathbf{o}_0 \mathbf{m} &= \mathbf{o}_0 \mathbf{o}_1 + \mathbf{o}_1 \mathbf{m} \\ \Leftrightarrow a_0 \mathbf{i}_0 + b_0 \mathbf{j}_0 &= x \mathbf{i}_0 + y \mathbf{j}_0 + a_1 \mathbf{i}_1 + b_1 \mathbf{j}_1 \end{aligned}$$

Exprimons cette relation dans le repère \mathcal{R}_0 :

$$\begin{pmatrix} a_0 \\ b_0 \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix} + a_1 \begin{pmatrix} \cos \theta \\ \sin \theta \end{pmatrix} + b_1 \begin{pmatrix} -\sin \theta \\ \cos \theta \end{pmatrix}$$

ou encore :

$$\begin{pmatrix} a_0 \\ b_0 \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix} + \underbrace{\begin{pmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{pmatrix}}_{\mathbf{R}_\theta} \begin{pmatrix} a_1 \\ b_1 \end{pmatrix}$$

2) On a :

$$\mathbf{R}_\theta^T \cdot \dot{\mathbf{R}}_\theta = \dot{\theta} \cdot \begin{pmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{pmatrix} \begin{pmatrix} -\sin \theta & -\cos \theta \\ \cos \theta & -\sin \theta \end{pmatrix} = \dot{\theta} \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$$

3) Dans le repère \mathcal{R}_0 ce vecteur \mathbf{u} s'exprime par :

$$\mathbf{u}_{|\mathcal{R}_0} = \begin{pmatrix} \dot{a}_0 \\ \dot{b}_0 \end{pmatrix} = \begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix} + \dot{\mathbf{R}}_\theta \begin{pmatrix} a_1 \\ b_1 \end{pmatrix} + \mathbf{R}_\theta \begin{pmatrix} \dot{a}_1 \\ \dot{b}_1 \end{pmatrix}$$

Donc :

$$\mathbf{u}_{|\mathcal{R}_1} = \mathbf{R}_\theta^T \cdot \mathbf{u}_{|\mathcal{R}_0} = \mathbf{R}_\theta^T \cdot \begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix} + \mathbf{R}_\theta^T \cdot \dot{\mathbf{R}}_\theta \begin{pmatrix} a_1 \\ b_1 \end{pmatrix} + \begin{pmatrix} \dot{a}_1 \\ \dot{b}_1 \end{pmatrix}$$

c'est-à-dire :

$$\begin{aligned} \mathbf{u}_{|\mathcal{R}_1} &= \mathbf{R}_\theta^T \cdot \begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix} + \dot{\theta} \begin{pmatrix} -b_1 \\ a_1 \end{pmatrix} + \begin{pmatrix} \dot{a}_1 \\ \dot{b}_1 \end{pmatrix} \\ &= \begin{pmatrix} v \\ 0 \end{pmatrix} + \dot{\theta} \cdot \begin{pmatrix} -b_1 \\ a_1 \end{pmatrix} + \begin{pmatrix} \dot{a}_1 \\ \dot{b}_1 \end{pmatrix} = \begin{pmatrix} v - \dot{\theta}b_1 + \dot{a}_1 \\ \dot{\theta}a_1 + \dot{b}_1 \end{pmatrix} \end{aligned}$$

4) On a :

$$\begin{aligned} \mathbf{a}_{|\mathcal{R}_0} &= \begin{pmatrix} \ddot{a}_0 \\ \ddot{b}_0 \end{pmatrix} = \frac{d}{dt} \left(\begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix} + \dot{\mathbf{R}}_\theta \begin{pmatrix} a_1 \\ b_1 \end{pmatrix} + \mathbf{R}_\theta \begin{pmatrix} \dot{a}_1 \\ \dot{b}_1 \end{pmatrix} \right) \\ &= \begin{pmatrix} \ddot{x} \\ \ddot{y} \end{pmatrix} + \ddot{\mathbf{R}}_\theta \begin{pmatrix} a_1 \\ b_1 \end{pmatrix} + 2\dot{\mathbf{R}}_\theta \begin{pmatrix} \dot{a}_1 \\ \dot{b}_1 \end{pmatrix} + \mathbf{R}_\theta \begin{pmatrix} \ddot{a}_1 \\ \ddot{b}_1 \end{pmatrix} \end{aligned}$$

Donc :

$$\begin{aligned} \mathbf{a}_{|\mathcal{R}_1} &= \mathbf{R}_\theta^T \mathbf{a}_{|\mathcal{R}_0} \\ &= \mathbf{R}_\theta^T \begin{pmatrix} \ddot{x} \\ \ddot{y} \end{pmatrix} + \mathbf{R}_\theta^T \ddot{\mathbf{R}}_\theta \begin{pmatrix} a_1 \\ b_1 \end{pmatrix} + 2\mathbf{R}_\theta^T \dot{\mathbf{R}}_\theta \begin{pmatrix} \dot{a}_1 \\ \dot{b}_1 \end{pmatrix} + \begin{pmatrix} \ddot{a}_1 \\ \ddot{b}_1 \end{pmatrix} \end{aligned}$$

Or :

$$\begin{aligned} \mathbf{R}_\theta^T \ddot{\mathbf{R}}_\theta &= \mathbf{R}_\theta^T \frac{d}{dt} \dot{\mathbf{R}}_\theta = \mathbf{R}_\theta^T \frac{d}{dt} \left(\dot{\theta} \begin{pmatrix} -\sin \theta & -\cos \theta \\ \cos \theta & -\sin \theta \end{pmatrix} \right) \\ &= \begin{pmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{pmatrix} \left(\dot{\theta} \begin{pmatrix} -\sin \theta & -\cos \theta \\ \cos \theta & -\sin \theta \end{pmatrix} + \dot{\theta}^2 \begin{pmatrix} -\cos \theta & \sin \theta \\ -\sin \theta & -\cos \theta \end{pmatrix} \right) \\ &= \ddot{\theta} \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} - \dot{\theta}^2 \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} -\dot{\theta}^2 & -\ddot{\theta} \\ \ddot{\theta} & -\dot{\theta}^2 \end{pmatrix} \end{aligned}$$

Finalement :

$$\begin{aligned}\mathbf{a}_{|\mathcal{R}_1} &= \begin{pmatrix} \alpha \\ \beta \end{pmatrix} + \begin{pmatrix} -\dot{\theta}^2 & -\ddot{\theta} \\ \ddot{\theta} & -\dot{\theta}^2 \end{pmatrix} \begin{pmatrix} a_1 \\ b_1 \end{pmatrix} + 2\dot{\theta} \begin{pmatrix} -b_1 \\ \dot{a}_1 \end{pmatrix} + \begin{pmatrix} \ddot{a}_1 \\ \ddot{b}_1 \end{pmatrix} \\ &= \begin{pmatrix} \alpha - \dot{\theta}^2 a_1 - \ddot{\theta} b_1 - 2\dot{\theta} \dot{b}_1 + \ddot{a}_1 \\ \beta + \ddot{\theta} a_1 - \dot{\theta}^2 b_1 + 2\dot{\theta} \dot{a}_1 + \ddot{b}_1 \end{pmatrix}\end{aligned}$$

5) Le véhicule qui nous précède freine si :

$$\langle \mathbf{a}_{|\mathcal{R}_1}, \mathbf{u}_{|\mathcal{R}_1} \rangle \leq 0$$

c'est-à-dire si :

$$\begin{pmatrix} \alpha - \dot{\theta}^2 a_1 - \ddot{\theta} b_1 - 2\dot{\theta} \dot{b}_1 + \ddot{a}_1 \\ \beta + \ddot{\theta} a_1 - \dot{\theta}^2 b_1 + 2\dot{\theta} \dot{a}_1 + \ddot{b}_1 \end{pmatrix}^T \cdot \begin{pmatrix} v - \dot{\theta} b_1 + \dot{a}_1 \\ \dot{\theta} a_1 + \dot{b}_1 \end{pmatrix} \leq 0$$

Correction de l'exercice 1.8 (modélisation d'un robot sous-marin)

La dérivée pour le vecteur position s'obtient en remarquant que :

$$\dot{\mathbf{p}} = v \mathbf{i}_1 \stackrel{[1.7]}{=} v \begin{pmatrix} \cos \theta \cos \psi \\ \cos \theta \sin \psi \\ -\sin \theta \end{pmatrix}$$

car \mathbf{i}_1 correspond à la première colonne de la matrice [1.7]. Finalement, en prenant en compte les équations [1.11], nous pouvons écrire les équations d'état pour notre sous-marin :

$$\begin{cases} \dot{p}_x = v \cos \theta \cos \psi \\ \dot{p}_y = v \cos \theta \sin \psi \\ \dot{p}_z = -v \sin \theta \\ \dot{v} = u_1 \\ \dot{\psi} = \frac{\sin \varphi}{\cos \theta} \cdot vu_2 + \frac{\cos \varphi}{\cos \theta} \cdot vu_3 \\ \dot{\theta} = \cos \varphi \cdot vu_2 - \sin \varphi \cdot vu_3 \\ \dot{\varphi} = vu_4 + \tan \theta \cdot (\sin \varphi \cdot vu_2 + \cos \varphi \cdot vu_3) \end{cases}$$

Nous avons ici un modèle cinématique (c'est-à-dire qu'il ne fait intervenir ni de forces ni de couples). Il n'y a aucun paramètre et le modèle peut donc être considéré comme exact si le robot sous-marin si le robot est solide (c'est-à-dire qui ne peut pas se tordre), et si sa trajectoire est tangente à l'axe du robot. Avoir un tel modèle permet d'utiliser des méthodes de commande non linéaire comme la méthode par bouclage linéarisant qui sera vue au chapitre 2. Ces méthodes sont en effet peu robustes par rapport à une erreur de modèle, mais sont d'une redoutable efficacité si un modèle précis pour notre système est disponible.

Correction de l'exercice 1.9 (graphisme robot 3D)

2) Nous donnons maintenant la fonction MATLAB faisant le dessin 3D :

```
function draw(x)
Auv0=[ 0 0 10 0 0 10 0 0 ;
-1 1 0 -1 -0.2 0 0.2 1 ;
0 0 0 0 1 0 1 0 ;
1 1 1 1 1 1 1 1 ] ;
E=eulermat(x(7),x(6),x(5)); %phi,theta,psi
R=[E,[x(1);x(2);x(3)];0 0 0 1] ;
Auv=R*Auv0 ;
plot3(Auv(1, :),Auv(2, :),Auv(3, :),'blue') ;
plot3(Auv(1, :),Auv(2, :),0*Auv(3, :),'black') ; % ombre
```

4) Afin de simuler notre robot, il nous faut tout d'abord écrire la fonction d'évolution suivante :

```
function xdot = f(x,u)
v=x(4) ; psi=x(5) ; theta=x(6) ; phi=x(7) ;
xdot=[v*cos(theta)*cos(psi) ;
v*cos(theta)*sin(psi) ;
-v*sin(theta) ; u(1) ;
(sin(phi)/cos(theta))*v*u(2)+(cos(phi)/cos(theta))*v*u(3) ;
cos(phi)*v*u(2)-sin(phi)*v*u(3) ;
-0.1*sin(phi)*cos(theta)+tan(theta)*(sin(phi)*v*u(2)+cos(phi)*v*u(3))] ;
```

La simulation pourra se faire par la méthode d'Euler. Cette simulation sera reprise à l'exercice 2.4 page 82 pour effectuer une régulation (voir aussi le fichier auv3d.m) :

```
dt=0.1 ;
x=[0 ;0 ;10 ;0.1 ;0 ;0 ;0] ;
for t=0 :dt :10,
u=... ;
x=x+dt*f(x,u) ;
draw(x) ;
end
```

Correction de l'exercice 1.10

Pour bien s'y prendre, il faut dessiner les bras les uns après les autres et dessiner les repères associés. Pour dessiner le repère associé à une matrice de transformation homogène 4×4 , on peut utiliser la fonction suivante :

```

function drawaxis(R)
A=R*[0 1;0 0; 0 0; 1 1]; plot3(A(1, :),A(2, :),A(3, :),'red');
% axes des x
A=R*[0 0;0 1; 0 0; 1 1]; plot3(A(1, :),A(2, :),A(3, :),'green');
% axes des y
A=R*[0 0;0 0; 0 1; 1 1]; plot3(A(1, :),A(2, :),A(3, :),'blue');
% axes des z
end

```

Chacun des sept bras peut être dessiné en reliant un triangle du plan xy du repère i à son homologue dans le repère $i + 1$. Cela peut être fait par la procédure suivante :

```

function drawArm(R1,R2)
J0=[-0.1 0.3 -0.1 -0.1; -0.1 -0.1 3*0.1 -0.1; 0 0 0 0; 1 1 1
1];
J1=R1*R0; J2=R2*R0;
J=[J1( :,1),J2( :,1),J2( :,2),J1( :,2),J1( :,3),J2( :,3),...
J2( :,1),J2( :,2),J2( :,3),J1( :,3),J1( :,1),J1( :,2)];
plot3(J(1, :),J(2, :),J(3, :),'black');
end

```

Enfin, l'ensemble des sept bras du robot dans une configuration q (dont les composantes sont les coordonnées articulaires) peut être dessiné comme suit :

```

function draw(q)
R=eye(4,4); drawaxis(R);
for j=1 :7,
Rold=R;
R=R*Transl([0 0 r(j)])*Transl([d(j) 0 0])*Rot([0 a(j) 0]);
drawArm(Rold,R); R=R*Rot([0 0 q(j)]);
drawaxis(R);
end; end

```

Il reste alors à configurer les paramètres du robot dans le programme principal et à le faire bouger dans une boucle. Cela peut être fait par le programme suivant :

```

a=[0,pi/2,0,-pi/2,-pi/2,-pi/2,0];
d=[0,0.5,0,-0.1,-0.3 -1,0];
r=[1,0.5,1, 0.1,1,0.2,0.2];
q=[0.3 ;0.3 ;0.3 ;0 ;1.5 ;0.1 ;1];
dt=0.05;
for i=1 :length(a),
for h=0 :dt :2*pi, draw(q); q(i)=q(i)+dt; end;
end

```

L'ensemble du programme est donné dans le fichier `staubli.m`.

Correction de l'exercice 1.11 (modélisation tridimensionnelle d'une roue)

1) Les variables d'état que nous allons choisir sont (i) les coordonnées (x, y) du point de contact, (ii) le vecteur de rotation ω exprimé dans le repère absolu et (iii) l'orientation (ψ, θ, φ) de la roue. Nous avons donc un système d'ordre 8. D'après l'équation du gyroscope (ou aussi le principe fondamental de la dynamique pour les corps en rotation), le couple \mathbf{c} exercé par les forces extérieures est proportionnel à la dérivée du vecteur de rotation ω suivant la relation :

$$\mathbf{c} = \mathbf{I}\dot{\omega}$$

où \mathbf{I} est la matrice d'inertie. Or, en prenant comme point de référence le centre de gravité de la roue, on comprend que le couple \mathbf{c} qui fait tourner la roue est généré par la réaction \mathbf{r} du sol appliquée au point de contact \mathbf{p} . Plaçons-nous dans le plan orthogonal au déplacement. On a, d'après le principe fondamental de la dynamique en translation appliqué au point \mathbf{g} :

$$\underbrace{\begin{pmatrix} r_x \\ r_z - mg \end{pmatrix}}_{\text{forces}} = m \underbrace{\begin{pmatrix} v\omega_z \\ 0 \end{pmatrix}}_{\text{accélération de } \mathbf{p}} + m \frac{\dot{\theta}^2}{\rho} \underbrace{\begin{pmatrix} -\sin \theta \\ -\cos \theta \end{pmatrix}}_{\text{accélération deg } \mathbf{p}}$$

C'est-à-dire :

$$\begin{pmatrix} r_x \\ r_y \end{pmatrix} = m \begin{pmatrix} v\omega_z - \frac{\dot{\theta}^2}{\rho} \sin \theta \\ g - \frac{\dot{\theta}^2}{\rho} \cos \theta \end{pmatrix}$$

Le couple \mathbf{c} exercé par \mathbf{r} est donc donné par :

$$\mathbf{c} = \underbrace{\rho(r_x \cos \theta - r_z \sin \theta)}_{=\rho m(v\omega_z \cos \theta - g \sin \theta)} \cdot \underbrace{\begin{pmatrix} \sin \psi \\ -\cos \psi \\ 0 \end{pmatrix}}_{\mathbf{n}}$$

De plus, d'après la relation [1.10], on a :

$$\begin{pmatrix} \dot{\psi} \\ \dot{\theta} \\ \dot{\varphi} \end{pmatrix} = \begin{pmatrix} 0 & -\sin \psi & \cos \theta \cos \psi \\ 0 & \cos \psi & \cos \theta \sin \psi \\ 1 & 0 & -\sin \theta \end{pmatrix}^{-1} \cdot \omega$$

Enfin, la vitesse du point de contact est $\rho \cdot \dot{\varphi}$. Ce point se déplace dans le sens de \mathbf{n} , ce qui nous donne :

$$\begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix} = \rho \cdot \dot{\varphi} \cdot \mathbf{n} = \begin{pmatrix} v \sin \psi \\ -v \cos \psi \end{pmatrix}$$

Finalement, les équations d'état sont :

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\omega} \\ \dot{\psi} \\ \dot{\theta} \\ \dot{\phi} \end{pmatrix} = \begin{pmatrix} \rho \cdot \dot{\phi} \sin \psi \\ -\rho \cdot \dot{\phi} \cos \psi \\ m (\rho^2 \dot{\phi} \omega_z \cos \theta - g \rho \sin \theta) \cdot \mathbf{I}^{-1} \cdot \begin{pmatrix} \sin \psi \\ -\cos \psi \\ 0 \end{pmatrix} \\ \begin{pmatrix} 0 & -\sin \psi & \cos \theta \cos \psi \\ 0 & \cos \psi & \cos \theta \sin \psi \\ 1 & 0 & -\sin \theta \end{pmatrix}^{-1} \cdot \omega \end{pmatrix}$$

Bien que $\dot{\phi}$ apparaisse dans le membre de droite, il s'agit bien d'équations d'état explicites puisque $\dot{\phi}$ s'exprime, par la dernière de ces six équations, en fonction des variables d'état. La fonction d'évolution se programme sous MATLAB comme suit (voir fichier `wheel.m`) :

```
function xdot=f(x)
w=x(3 :5); m=1; g=9.81;
psi=x(6); theta=x(7);
I=diag([1 0.5 0.5])*(m/2)*rho^2
n=[sin(psi);-cos(psi);0];
deuler=[0 -sin(psi) cos(theta)*cos(psi);
0 cos(psi) cos(theta)*sin(psi);
1 0 -sin(theta) ]\w;
dphi=deuler(3);
dw=m*(rho^2*dphi*w(3)*cos(theta)-g*rho*sin(theta))*inv(I)*n;
xdot=[rho*dphi*n(1 :2) ;dw ;deuler];
end
```

2) Si par exemple on écarte les masses du centre, la roue tend à accélérer, du fait du principe de conservation du moment cinétique. On pourra se servir de transferts de masse pour asservir le vecteur ω de la roue et donc pouvoir contrôler sa trajectoire.

Correction de l'exercice 1.12 (stabilité mécanique d'un robot sous-marin)

1) Comme le corps est homogène de même densité que l'eau, la poussée d'Archimède compense le poids du corps. Tout se passe comme si le corps était dans l'espace. Le corps immergé ne tourne pas si $(\mathbf{a} - \mathbf{g}) \wedge \mathbf{f} = \mathbf{0}$.

2) On aura une stabilité en rotation si en plus de la relation d'équilibre, on a $\langle \mathbf{a} - \mathbf{g}, \mathbf{f} \rangle > 0$.

3) Le centre de carène \mathbf{a} (là où s'exerce la force d'Archimède \mathbf{f} , qui est verticale vers le haut) est en arrière du centre de gravité. Il faut reculer les masses pour qu'à l'horizontal, on ait la condition d'équilibre.

4) Ici, f correspond à la force d'Archimède. Son point d'application est a est en dessous de g et la condition de stabilité n'est pas satisfaite. Il faut descendre g en bougeant les masses.

5) Ici, la force f est la trainée (résistance de l'eau). Il faut que le centre de la trainée (qui ici est trop bas) soit dans le même plan horizontal que g (équilibre). Il faut présenter moins de résistance à l'eau dans la partie basse du robot. Il faut aussi prendre en compte la stabilité. Le centre de trainée (attention à ce concept, il n'a de sens que sur un voisinage d'une direction) doit être derrière g (principe de l'empennage la fléchette). On peut par exemple rajouter des ailerons à l'arrière du robot.

6) Les propulseurs doivent pointer sur g pour empêcher les rotations (sauf si bien sûr l'objectif est de tourner). Pour être plus stable, on pourrait penser qu'il faut mettre les propulseurs à l'avant, c'est-à-dire dans le sens du mouvement. Il n'en est rien car les propulseurs tournent avec le robot. Il n'y a donc ni stabilité, ni instabilité.

7) La boussole est perturbée par le métal. Quand le robot est près du mur, il cherche à réguler dans une direction qui n'est pas la bonne. De cette façon, il écarte la boussole du mur, là où la boussole n'est plus perturbée. On crée ainsi un cycle. On peut éviter cela en fusionnant (par un filtre de Kalman) les données de la boussole avec les données des gyros.

8) La configuration (a) se comporte de façon identique à la configuration en (b) car les moments relatifs au centre de carène sont identiques. La configuration (c) est plus manœuvrante que la configuration en (d) car les moments engendrés par les propulseurs sont plus grands.

Chapitre 2

Commande par bouclage linéarisant

Du fait des multiples rotations, les robots sont considérés comme des systèmes fortement non linéaires. Dans ce chapitre, nous allons chercher à proposer des régulateurs non linéaires afin de contraindre le vecteur d'état du robot à suivre une trajectoire fixée à l'avance ou bien à rester dans une zone bien déterminée de son espace de travail. Contrairement à l'approche linéaire qui offre une méthodologie générale mais limitée au voisinage d'un point de l'espace d'état [BOU 06, JAU 14] les approches non linéaires ne s'appliquent qu'à des classes limitées de systèmes, mais elles permettent d'étendre la plage de bon fonctionnement du système. Il n'existe en effet pas de méthode générale pour stabiliser globalement les systèmes non linéaires. En revanche, il existe une multitude de méthodes qui s'appliquent à des cas particuliers [FAN 01]. Le but de ce chapitre est d'en présenter une assez représentative des méthodes théoriques existantes (alors que dans le chapitre suivant, nous verrons des approches plus pragmatiques). Il s'agit de la méthode de *linéarisation par bouclage* ou par *bouclage linéarisant* qui nécessite la connaissance d'un modèle d'état précis et fiable pour notre robot. Les robots considérés ici sont des systèmes mécaniques dont la modélisation peut être trouvée dans [JAU 05]. Nous allons supposer dans ce chapitre que le vecteur d'état est entièrement connu. En pratique, il doit être estimé à partir des mesures capteurs. Nous verrons au chapitre 7 comment une telle estimation peut être faite.

2.1. Commande d'une chaîne d'intégrateurs

Comme nous allons le montrer plus tard dans ce chapitre, la méthode par bouclage linéarisant nous ramène à un problème de régulation d'un système formé de plusieurs chaînes d'intégrateurs découplées entre elles. Nous allons donc considérer dans ce paragraphe une chaîne d'intégrateurs dont l'entrée u et la sortie y sont reliées par l'équation différentielle :

$$y^{(n)} = u$$

2.1.1. Régulateur proportionnel et dérivées

On se propose tout d'abord de stabiliser ce système par un régulateur *proportionnel et dérivées* du type :

$$u = \alpha_0 (w - y) + \alpha_1 (\dot{w} - \dot{y}) + \cdots + \alpha_{n-1} (w^{(n-1)} - y^{(n-1)}) + w^{(n)}$$

où w est la consigne souhaitée pour y . Notons que w peut dépendre du temps. Le fait que ce régulateur nécessite les dérivées de y n'est pas un problème dans le cadre défini par le bouclage linéarisant. En effet, toutes ces dérivées peuvent être écrites comme des fonctions analytiques de l'état x du système et de l'entrée u . En ce qui concerne la consigne $w(t)$, elle est choisie par l'utilisateur et une expression analytique de $w(t)$ peut être supposée connue (par exemple $w(t) = \sin(t)$). Ainsi, le calcul des dérivées de w se fait de façon formelle et aucune sensibilité de l'opérateur dérivation par rapport au bruit n'est à craindre.

Le système bouclé est décrit par l'équation différentielle :

$$y^{(n)} = u = \alpha_0 (w - y) + \alpha_1 (\dot{w} - \dot{y}) + \cdots + \alpha_{n-1} (w^{(n-1)} - y^{(n-1)}) + w^{(n)}$$

Si l'on définit l'erreur e entre la consigne w et la sortie y par $e = w - y$, cette équation devient :

$$e^{(n)} + \alpha_{n-1} e^{(n-1)} + \cdots + \alpha_1 \dot{e} + \alpha_0 e = 0$$

Cette équation différentielle est appelée *dynamique de l'erreur*. Son polynôme caractéristique donné par :

$$P(s) = s^n + \alpha_{n-1} s^{n-1} + \cdots + \alpha_1 s + \alpha_0 \quad [2.1]$$

peut donc être choisi arbitrairement parmi les polynômes de degré n . Bien sûr, on choisira un polynôme dont les racines sont toutes à parties réelles négatives, afin d'assurer la stabilité du système. Par exemple, si $n = 3$ et si l'on souhaite que tous les pôles soient égaux à -1 , on posera :

$$s^3 + \alpha_2 s^2 + \alpha_1 s + \alpha_0 = (s + 1)^3 = s^3 + 3s^2 + 3s + 1$$

D'où :

$$\alpha_2 = 3, \alpha_1 = 3, \alpha_0 = 1$$

Le régulateur alors obtenu est donné par :

$$u = (w - y) + 3(\dot{w} - \dot{y}) + 3(\ddot{w} - \ddot{y}) + \dddot{w}$$

REMARQUE 2.1.– Dans ce document, nous choisirons, par simplicité, de placer tous nos pôles en -1 . Le raisonnement précédent appliqué pour différents degrés n , nous amène aux commandes suivantes :

$$\begin{aligned} n = 1 \quad u &= (w - y) + \dot{w} \\ n = 2 \quad u &= (w - y) + 2(\dot{w} - \ddot{y}) + \ddot{w} \\ n = 3 \quad u &= (w - y) + 3(\dot{w} - \ddot{y}) + 3(\ddot{w} - \dddot{y}) + \dddot{w} \\ n = 4 \quad u &= (w - y) + 4(\dot{w} - \ddot{y}) + 6(\ddot{w} - \dddot{y}) + 4(\dddot{w} - \ddot{\ddot{y}}) + \ddot{\ddot{w}} \end{aligned} \quad [2.2]$$

On remarque que les coefficients correspondent à ceux du triangle de Pascal :

$$\begin{array}{cccccc} 1 & & & & & \\ 1 & 1 & & & & \\ 1 & 2 & 1 & & & \\ 1 & 3 & 3 & 1 & & \\ 1 & 4 & 6 & 4 & 1 & \end{array}$$

2.1.2. Régulateur proportionnel intégral et dérivées

Afin de compenser les perturbations constantes, nous pouvons décider de rajouter un terme intégral. Nous obtenons une commande de type PID (proportionnel intégral et dérivée) de la forme :

$$\begin{aligned} u &= \alpha_{-1} \int_{\tau=0}^t (w(\tau) - y(\tau)) d\tau \\ &+ \alpha_0 (w - y) + \alpha_1 (\dot{w} - \dot{y}) + \cdots + \alpha_{n-1} (w^{(n-1)} - y^{(n-1)}) + w^{(n)} \end{aligned} \quad [2.3]$$

Le système bouclé est décrit par l'équation différentielle :

$$\begin{aligned} y^{(n)} &= \alpha_{-1} \int_{\tau=0}^t (w(\tau) - y(\tau)) d\tau \\ &+ \alpha_0 (w - y) + \alpha_1 (\dot{w} - \dot{y}) + \cdots + \alpha_{n-1} (w^{(n-1)} - y^{(n-1)}) + w^{(n)} \end{aligned}$$

D'où, en dérivant une fois :

$$e^{(n+1)} + \alpha_{n-1} e^{(n)} + \cdots + \alpha_1 \ddot{e} + \alpha_0 \dot{e} + \alpha_{-1} e = 0$$

Le polynôme caractéristique :

$$P(s) = s^{n+1} + \alpha_{n-1} s^n + \cdots + \alpha_1 s^2 + \alpha_0 s + \alpha_{-1}$$

peut être choisi arbitrairement, comme pour la commande proportionnelle et dérivées.

2.2. Exemple introductif

Avant de donner les principes de la méthode de régulation par bouclage linéarisant, nous allons considérer un exemple introductif. Il s'agit du pendule de la figure 2.1. L'entrée de ce système est le couple u exercé sur le pendule.

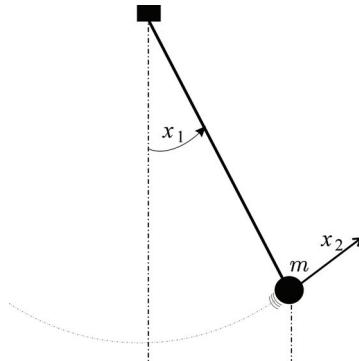


Figure 2.1. Pendule simple de vecteur d'état $\mathbf{x} = (x_1, x_2)$

Sa représentation d'état est supposée être :

$$\begin{cases} \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} x_2 \\ -\sin x_1 + u \end{pmatrix} \\ y = x_1 \end{cases}$$

Il s'agit bien sûr d'un modèle normalisé où les coefficients (masse, gravité, longueur) ont tous été fixés à 1. On souhaite que la position $x_1(t)$ du pendule soit égale à une consigne $w(t)$ pouvant varier dans le temps. En utilisant une méthode par bouclage linéarisant, qui sera expliquée par la suite, on cherche à proposer un régulateur par retour d'état tel que l'erreur $e = w - x_1$ converge vers 0 en $\exp(-t)$ (ce qui signifie que l'on place les pôles en -1). Dérivons y jusqu'à faire apparaître l'entrée u . On a :

$$\begin{aligned} \dot{y} &= x_2 \\ \ddot{y} &= -\sin x_1 + u \end{aligned}$$

Choisissons :

$$u = \sin x_1 + v \quad [2.4]$$

où v correspond à la nouvelle entrée dite intermédiaire. On obtient :

$$\ddot{y} = v \quad [2.5]$$

Le bouclage ainsi réalisé s'appelle *bouclage linéarisant* car il transforme le système non linéaire en un système linéaire. Le système ainsi obtenu peut être stabilisé par des techniques linéaires classiques. Prenons par exemple une commande proportionnelle et dérivée :

$$\begin{aligned} v &= (w - y) + 2(\dot{w} - \dot{y}) + \ddot{w} \\ &= (w - x_1) + 2(\dot{w} - x_2) + \ddot{w} \end{aligned}$$

En injectant cette expression de v dans [2.5], on obtient :

$$\ddot{y} = (w - x_1) + 2(\dot{w} - x_2) + \ddot{w}$$

Soit :

$$e + 2\dot{e} + \ddot{e} = 0$$

où $e = w - x_1$ est l'erreur entre la position du pendule et sa consigne. Le régulateur complet s'exprime par :

$$u \stackrel{[2.4]}{=} \sin x_1 + (w - x_1) + 2(\dot{w} - x_2) + \ddot{w}$$

Si maintenant, on souhaite que l'angle x_1 du pendule soit égal à $\sin t$ une fois le régime transitoire passé. Il nous suffit de prendre $w(t) = \sin t$. Ainsi, $\dot{w}(t) = \cos t$ et $\ddot{w} = -\sin t$. Par conséquence, la commande est donnée par :

$$u = \sin x_1 + (\sin t - x_1) + 2(\cos t - x_2) - \sin t$$

Dans cet exemple très simple, on comprend que le régulateur proposé est non linéaire et dépendant du temps. Aucune approximation issue d'une linéarisation n'a été effectuée. Bien sûr, une linéarisation a été faite par un premier bouclage de rendre le système linéaire, mais cette linéarisation n'a introduit aucune approximation.

2.3. Principe de la méthode

2.3.1. *Principe*

Nous cherchons à généraliser ici la méthode décrite à la section précédente. Considérons le système non linéaire décrit par :

$$\begin{cases} \dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}) + \mathbf{g}(\mathbf{x})\mathbf{u} \\ \mathbf{y} = \mathbf{h}(\mathbf{x}) \end{cases} \quad [2.6]$$

dont le nombre d'entrées et de sorties sont tous les deux égaux à m . L'idée de la méthode par bouclage linéarisant est de boucler le système par une

commande du type $\mathbf{u} = \mathbf{r}(\mathbf{x}, \mathbf{v})$, où \mathbf{v} est la nouvelle entrée, aussi de dimension m . Cette opération nécessite que l'état soit complètement accessible. Si cela n'est pas le cas, on se doit de fabriquer un observateur, mais dans un contexte non linéaire, cela est une opération très difficile. Puisqu'ici l'état est supposé accessible, le vecteur \mathbf{y} ne doit pas vraiment être considéré comme une sortie, mais plutôt comme le vecteur des variables consignées.

Pour effectuer ce bouclage, il nous faut exprimer les dérivées successives de chacun des y_i en fonction de l'état et de l'entrée. On s'arrête de dériver y_i , dès que les entrées commencent à intervenir dans l'expression de la dérivée. Nous disposons ainsi d'une équation du type :

$$\begin{pmatrix} y_1^{(k_1)} \\ \vdots \\ y_m^{(k_m)} \end{pmatrix} = \mathbf{A}(\mathbf{x}) \mathbf{u} + \mathbf{b}(\mathbf{x}) \quad [2.7]$$

où k_i désigne le nombre de fois qu'il nous faut dériver y_i pour y voir apparaître une entrée (voir les exemples dans les paragraphes qui suivent pour une meilleure compréhension). Sous l'hypothèse que la matrice $\mathbf{A}(\mathbf{x})$ soit inversible, le bouclage :

$$\mathbf{u} = \mathbf{A}^{-1}(\mathbf{x})(\mathbf{v} - \mathbf{b}(\mathbf{x})) \quad [2.8]$$

où \mathbf{v} est notre nouvelle entrée (voir figure 2.2), forme un système linéaire \mathcal{S}_L de m entrées à m sorties décrit par les équations différentielles :

$$\mathcal{S}_L : \begin{cases} y_1^{(k_1)} = v_1 \\ \vdots = \vdots \\ y_m^{(k_m)} = v_m \end{cases}$$

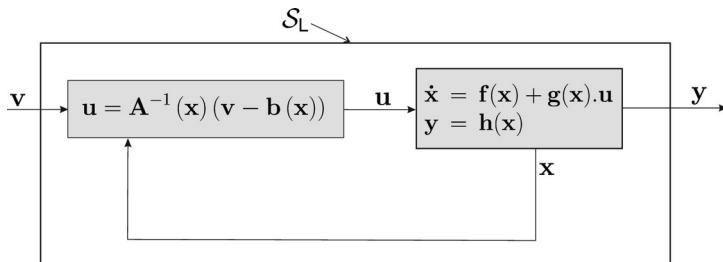


Figure 2.2. Le système non linéaire, une fois bouclé, devient linéaire et découpé ; il devient donc facile à commander

Ce système est linéaire et complètement découplé (c'est-à-dire que chaque entrée v_i agit sur une et une seule sortie y_i). Il est donc très facile à commander par les techniques linéaires classiques. Ici, comme le système à commander est constitué de chaînes d'intégrateurs découplés, nous utiliserons m régulateurs de types PID (proportionnel intégral et dérivées) dont nous avons rappelé les principes à la section 2.1. Notons que pour utiliser de tels régulateurs, il est nécessaire de disposer des dérivées des sorties. Comme nous avons supposé avoir accès à toutes les variables d'état x_i du système, une expression formelle de ces dérivées en fonction des x_i s'obtient facilement en utilisant les équations d'état.

REMARQUE 2.2.– Les robots sont dits *redondants* s'il y a plus d'entrées que nécessaire, c'est-à-dire si $\dim \mathbf{u} > \dim \mathbf{y}$. Dans ce cas, la matrice $\mathbf{A}(\mathbf{x})$ est rectangulaire. Afin d'appliquer le bouclage [2.8], on peut utiliser l'inverse généralisée de *Moore Penrose*. Si \mathbf{A} est de rang plein (c'est-à-dire égal à $\dim \mathbf{y}$), cette inverse est donnée par :

$$\mathbf{A}^\dagger = \mathbf{A}^T \cdot (\mathbf{A} \cdot \mathbf{A}^T)^{-1}$$

Ainsi, nous aurons :

$$\dim \mathbf{v} = \dim \mathbf{y} < \dim \mathbf{u}$$

et on se retrouve dans une situation identique à celle du robot carré (c'est-à-dire non redondant).

2.3.2. Degré relatif

En analysant bien le cheminement pour obtenir l'équation [2.7], on se rend compte que la $k^{\text{ième}}$ dérivée de la $i^{\text{ième}}$ sortie $y_i^{(k)}$ s'exprime sous la forme :

$$\begin{aligned} y_i^{(k)} &= \hat{b}_{ik}(\mathbf{x}) \text{ si } j < k_i \\ y_i^{(k)} &= \hat{\mathbf{a}}_{ik}^T(\mathbf{x}) \cdot \mathbf{u} + \hat{b}_{ik}(\mathbf{x}) \text{ si } k = k_i \\ y_i^{(k)} &= \hat{\mathbf{a}}_{ik}(\mathbf{x}, \mathbf{u}, \dot{\mathbf{u}}, \ddot{\mathbf{u}}, \dots) \text{ si } k > k_i \end{aligned}$$

Le coefficient k_i s'appelle le *degré relatif* de la $i^{\text{ième}}$ sortie. Si l'on mesure l'état du système \mathbf{x} et son entrée \mathbf{u} , on peut donc disposer de toutes les dérivées successives des sorties $y_i^{(k)}$, tant que k reste inférieur ou égal à k_i . En effet, en raison des bruits hautes fréquences apparaissant dans les signaux, on ne peut pas disposer de la dérivée des signaux par l'utilisation de dérivateurs de façon fiable. Nous avons donc une fonction analytique :

$$\begin{aligned} \Delta : \quad \mathbb{R}^m &\rightarrow \mathbb{R}^{(k_1+1) \times \dots \times (k_m+1)} \\ (\mathbf{x}, \mathbf{u}) &\mapsto \Delta(\mathbf{x}, \mathbf{u}) = (y_1, \dot{y}_1, \dots, y_1^{(k_1)}, y_2, \dot{y}_2, \dots, y_m^{(k_m)}) \end{aligned}$$

qui nous permet de disposer de toutes les dérivées des sorties (jusqu'à leur degré relatif), et ceci sans avoir à utiliser de dérivateurs de signaux.

EXEMPLE 2.1.– *Considérons le système décrit par :*

$$\begin{cases} \dot{x} = xu + x^3 \\ y = 2x \end{cases}$$

Nous avons :

$$\begin{aligned} y &= 2x \\ \dot{y} &= 2\dot{x} = 2xu + 2x^3 \\ \ddot{y} &= 2\dot{x}u + 2x\dot{u} + 6\dot{x}x^2 = 2(xu + x^3)u + 2x\dot{u} + 6(xu + x^3)x^2 \end{aligned}$$

Nous avons donc un degré relatif $k = 1$ pour la sortie y . On peut donc disposer de \dot{y} sans avoir à utiliser de dérivateurs numériques. Cela n'est pas le cas pour \ddot{y} car disposer de u avec une bonne précision ne signifie pas que l'on dispose de \dot{u} . Ici, nous avons $\Delta(x, u) = (2x, 2xu + 2x^3)$.

2.3.3. Matrice des retards différentiels

On appelle *retard différentiel* r_{ij} qui sépare l'entrée u_j de la sortie y_i le nombre de fois qu'il faut dériver y_i pour voir apparaître u_j . La matrice \mathbf{R} des r_{ij} est appelée *matrice des retards différentiels*. A la simple lecture des équations d'état, cette matrice peut être obtenue sans calcul, juste en comptant le nombre d'intégrations que doit subir chaque entrée u_j pour affecter algébriquement la sortie y_i (des exemples sont traités plus en détail dans les exercices). Le degré relatif pour chaque sortie peut être obtenu en prenant le minimum de chaque ligne. Prenons par exemple :

$$\mathbf{R} = \begin{pmatrix} 1 & 2 & 2 \\ 3 & 4 & 3 \\ 4 & \infty & 2 \end{pmatrix}$$

Le système associé possède trois entrées, trois sorties et les degrés relatifs (voir composantes en gras, dans la formule qui précède) sont $k_1 = 1, k_2 = 3, k_3 = 2$. S'il existe un j tel que $\forall i, r_{ij} > k_i$ (ou de façon équivalente si une colonne n'a aucun élément en gras), la matrice \mathbf{R} est dite *mal équilibrée*. Dans notre exemple, elle n'est pas équilibrée car il existe un j (ici $j = 2$) tel que $\forall i, r_{ij} > k_i$. Si la matrice est mal équilibrée alors pour tout $i, y_i^{(k_i)}$ ne dépend pas de u_j . Dans ce cas, la $j^{\text{ième}}$ colonne de $\mathbf{A}(\mathbf{x})$ sera nulle et $\mathbf{A}(\mathbf{x})$ sera toujours singulière. Ainsi, le bouclage [2.8] n'aura pas de sens. Une méthode pour s'en sortir est de retarder certaines entrées u_j en rajoutant un ou plusieurs intégrateurs devant le système. Rajouter un intégrateur devant la $j^{\text{ième}}$

entrée revient à rajouter 1 à la $j^{\text{ème}}$ colonne de \mathbf{R} . Dans notre exemple, si l'on rajoute un intégrateur devant u_1 , on obtient :

$$\mathbf{R} = \begin{pmatrix} 2 & 2 & 2 \\ 4 & 4 & 3 \\ 5 & \infty & 2 \end{pmatrix}$$

Les degrés relatifs deviennent $k_1 = 2, k_2 = 3, k_3 = 2$ et la matrice \mathbf{R} devient équilibrée.

2.3.4. Singularités

La matrice $\mathbf{A}(\mathbf{x})$ intervenant dans le bouclage [2.8] peut ne pas être inversible. Les valeurs pour \mathbf{x} telles que $\det(\mathbf{A}(\mathbf{x})) = 0$ sont appelées *singularités*. L'étude des singularités est fondamentale car, même si elles forment généralement un ensemble de mesure nulle dans l'espace d'état, elles sont parfois impossibles à éviter. On appelle *ensemble des sorties acceptables* pour le système [2.6] la quantité :

$$\mathbb{S}_y = \{\mathbf{y} \in \mathbb{R}^m \mid \exists \mathbf{x} \in \mathbb{R}^n, \exists \mathbf{u} \in \mathbb{R}^m, \mathbf{f}(\mathbf{x}) + \mathbf{g}(\mathbf{x}) \cdot \mathbf{u} = \mathbf{0}, \mathbf{y} = \mathbf{h}(\mathbf{x})\}$$

L'ensemble \mathbb{S}_y est donc constitué par la projection sur \mathbb{R}^m d'une variété (ou surface) différentiable de dimension m (car on a $m + n$ équations pour $2m + n$ variables). Ainsi, sauf cas dégénéré, \mathbb{S}_y est un sous-ensemble de \mathbb{R}^m avec un intérieur non vide et un extérieur.

Afin de bien comprendre cela, considérons l'exemple d'un chariot sur des rails comme représenté sur la figure 2.3. Ce chariot peut être propulsé par un ventilateur horizontal dont l'angle de poussée peut être commandé. Mais attention, la vitesse de rotation du ventilateur est fixée.

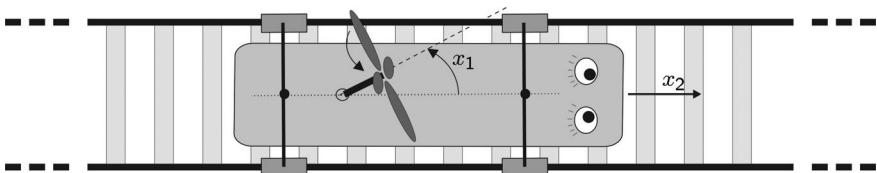


Figure 2.3. Robot chariot propulsé par un ventilateur

Les équations d'état pouvant modéliser ce système sont données par :

$$\begin{cases} \dot{x}_1 = u \\ \dot{x}_2 = \cos x_1 - x_2 \\ y = x_2 \end{cases}$$

où x_1 est l'angle de poussée du ventilateur, x_2 est la vitesse du chariot. Notons que nous avons pris en compte une force de frottement visqueux. L'ensemble des sorties acceptables est :

$$\begin{aligned}\mathbb{S}_y &= \{y \mid \exists x_1, \exists x_2, \exists u, u = 0, \cos x_1 - x_2 = 0, y = x_2\} \\ &= \{y \mid \exists x_1, \cos x_1 = y\} = [-1, 1]\end{aligned}$$

Ce qui signifie que nous ne pourrons pas stabiliser le chariot à une vitesse strictement supérieure à 1, en valeur absolue. Appliquons une méthode par bouclage linéarisant. On a :

$$\begin{aligned}\dot{y} &= \cos x_1 - x_2 \\ \ddot{y} &= -(\sin x_1) u - \cos x_1 + x_2\end{aligned}$$

et donc, la commande linéarisante est donnée par :

$$u = \frac{-1}{\sin x_1} (v + \cos x_1 - x_2)$$

Le système bouclé possède alors pour équation :

$$\ddot{y} = v$$

Il peut nous sembler que n'importe quelle valeur pour y peut être atteinte, puisque v peut être choisi arbitrairement. Cela serait juste, si nous n'avions pas la singularité qui se produit lorsque $\sin x_1 = 0$. Posons par exemple $v(t) = 1$ et $\mathbf{x}(0) = (\frac{\pi}{3}, 0)$. Nous devrions avoir :

$$\begin{aligned}\ddot{y}(t) &= v(t) = 1 \\ \dot{y}(t) &= \dot{y}(0) + \int_0^t \ddot{y}(\tau) d\tau = \cos x_1(0) - x_2(0) + t = \frac{1}{2} + t \\ y(t) &= y(0) + \int_0^t \dot{y}(\tau) d\tau = x_2(0) + t^2 + \frac{1}{2}t = t^2 + \frac{1}{2}t\end{aligned}$$

ce qui n'est physiquement pas possible. Voici ce qui se passe si nous appliquons une telle commande : l'entrée u oriente le ventilateur dans la bonne direction, et l'équation $\ddot{y} = v$ se trouve alors satisfaite, au moins au tout début. Puis x_1 s'annule et la singularité est atteinte. L'équation $\ddot{y} = v$ ne peut alors plus être satisfaite. Pour certains systèmes, il arrive qu'une telle singularité puisse être franchie. Cela n'est pas le cas ici.

2.4. Char

2.4.1. Premier modèle

On considère un char décrit par les équations d'état suivantes :

$$\begin{cases} \dot{x} = v \cos \theta \\ \dot{y} = v \sin \theta \\ \dot{\theta} = u_1 \\ \dot{v} = u_2 \end{cases}$$

où v est la vitesse du char, θ son orientation et (x, y) les coordonnées de son centre. Le vecteur d'état est donné par $\mathbf{x} = (x, y, \theta, v)$. On se propose de calculer un régulateur qui permette de décrire une cycloïde d'équation :

$$\begin{cases} x_d(t) = R \sin(f_1 t) + R \sin(f_2 t) \\ y_d(t) = R \cos(f_1 t) + R \cos(f_2 t) \end{cases}$$

avec $R = 15$, $f_1 = 0.02$ et $f_2 = 0.12$. Pour cela, on utilise une méthode par bouclage linéarisant. On a :

$$\begin{pmatrix} \ddot{x} \\ \ddot{y} \end{pmatrix} = \begin{pmatrix} u_2 \cos \theta - u_1 v \sin \theta \\ u_2 \sin \theta + u_1 v \cos \theta \end{pmatrix} = \begin{pmatrix} -v \sin \theta & \cos \theta \\ v \cos \theta & \sin \theta \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}$$

Si l'on choisit pour entrée :

$$\begin{pmatrix} u_1 \\ u_2 \end{pmatrix} = \begin{pmatrix} -v \sin \theta & \cos \theta \\ v \cos \theta & \sin \theta \end{pmatrix}^{-1} \begin{pmatrix} v_1 \\ v_2 \end{pmatrix}$$

où (v_1, v_2) est le nouveau vecteur d'entrée, on obtient le système linéaire :

$$\begin{pmatrix} \ddot{x} \\ \ddot{y} \end{pmatrix} = \begin{pmatrix} v_1 \\ v_2 \end{pmatrix}$$

Bouclons ce dernier de façon à avoir tous les pôles en -1 . D'après [2.2], on obtient :

$$\begin{cases} v_1 = (x_d - x) + 2(\dot{x}_d - \dot{x}) + \ddot{x}_d = (x_d - x) + 2(\dot{x}_d - v \cos \theta) + \ddot{x}_d \\ v_2 = (y_d - y) + 2(\dot{y}_d - \dot{y}) + \ddot{y}_d = (y_d - y) + 2(\dot{y}_d - v \sin \theta) + \ddot{y}_d \end{cases}$$

Le système bouclé obéit alors aux équations différentielles suivantes :

$$\begin{pmatrix} (x_d - x) + 2(\dot{x}_d - \dot{x}) + (\ddot{x}_d - \ddot{x}) \\ (y_d - y) + 2(\dot{y}_d - \dot{y}) + (\ddot{y}_d - \ddot{y}) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

Si l'on définit le vecteur erreur $\mathbf{e} = (e_x, e_y) = (x_d - x, y_d - y)$, la dynamique de l'erreur s'écrit :

$$\begin{pmatrix} e_x + 2\dot{e}_x + \ddot{e}_x \\ e_y + 2\dot{e}_y + \ddot{e}_y \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

qui est stable et converge rapidement vers 0. La commande sera donc au final :

$$\begin{pmatrix} u_1 \\ u_2 \end{pmatrix} = \begin{pmatrix} -v \sin \theta & \cos \theta \\ v \cos \theta & \sin \theta \end{pmatrix}^{-1} \begin{pmatrix} (x_d - x) + 2(\dot{x}_d - v \cos \theta) + \ddot{x}_d \\ (y_d - y) + 2(\dot{y}_d - v \sin \theta) + \ddot{y}_d \end{pmatrix} \quad [2.9]$$

avec :

$$\begin{aligned} \dot{x}_d(t) &= Rf_1 \cos(f_1 t) + Rf_2 \cos(f_2 t) \\ \dot{y}_d(t) &= -Rf_1 \sin(f_1 t) - Rf_2 \sin(f_2 t) \\ \ddot{x}_d(t) &= -Rf_1^2 \sin(f_1 t) - Rf_2^2 \sin(f_2 t) \\ \ddot{y}_d(t) &= -Rf_1^2 \cos(f_1 t) - Rf_2^2 \cos(f_2 t) \end{aligned}$$

Le programme MATLAB qui suit, rangé dans le fichier `cycloide.m`, simule le comportement du système régulé :

```
x=[10 ;10 ;0 ;2] ; %x=[x,y,theta,v)
dt=0.1 ; R=15 ; f1=0.02 ;f2=0.12 ;
for t=0 :dt :10,
w = R*[sin(f1*t)+sin(f2*t) ;cos(f1*t)+cos(f2*t)] ;
dw = R*[f1*cos(f1*t)+f2*cos(f2*t) ;-f1*sin(f1*t)-f2*sin(f2*t)] ;
ddw=R*[-f1*f1*sin(f1*t)-f2*f2*sin(f2*t) ;-f1*f1*cos(f1*t)-f2*f2*cos(f2*t)] ;
u=control(x,w,dw,ddw) ;
x=x+f(x,u)*dt ;
end ;
```

Pour l'évolution du robot, le programme utilise la fonction d'évolution suivante :

```
function xdot=f(x,u)
theta=x(3) ;v=x(4) ;
xdot=[v*cos(theta) ; v*sin(theta) ; u(1) ; u(2)] ;
end
```

Quant au régulateur, il est réalisé par la fonction :

```
function u=control(x,w,dw)
v=0.25 *( w - [x(1) ;x(2)])+1*(dw - [x(4)*cos(x(3)) ;x(4)*sin(x(3))])+ddw ;
A=[-x(4)*sin(x(3)) , cos(x(3)) ; x(4)*cos(x(3)) , sin(x(3))] ;
u=inv(A)*v ;
end
```

2.4.2. Deuxième modèle

Supposons maintenant que le char soit décrit par les équations d'état :

$$\begin{cases} \dot{x} = u_1 \cos \theta \\ \dot{y} = u_1 \sin \theta \\ \dot{\theta} = u_2 \end{cases}$$

Choisissons pour sortie le vecteur $\mathbf{y} = (x, y)$. La méthode par bouclage linéarisant aboutit à une matrice $\mathbf{A}(\mathbf{x})$ qui est toujours singulière. Comme cela a été expliqué à la section 2.3.3, ce constat peut se prédir sans calcul juste en observant que la matrice des retards différentiels :

$$\mathbf{R} = \begin{pmatrix} 1 & 2 \\ 1 & 2 \end{pmatrix}$$

Cette dernière contient une colonne dont les éléments ne correspondent jamais au minimum de la ligne correspondante (c'est-à-dire une colonne sans élément en gras). Nous allons illustrer comment nous sortir d'une telle situation par l'ajout d'intégrateurs devant certaines entrées. Rajoutons par exemple un intégrateur, dont la variable d'état sera notée z , devant la première entrée. Rappelons que rajouter un intégrateur devant la j ème entrée du système revient à retarder cette entrée et donc à rajouter 1 à tous les éléments de la j colonne de \mathbf{R} . La matrice \mathbf{R} devient alors équilibrée. On obtient un nouveau système décrit par :

$$\begin{cases} \dot{x} = z \cos \theta \\ \dot{y} = z \sin \theta \\ \dot{\theta} = u_2 \\ \dot{z} = c_1 \end{cases}$$

On a :

$$\begin{cases} \ddot{x} = \dot{z} \cos \theta - z \dot{\theta} \sin \theta = c_1 \cos \theta - z u_2 \sin \theta \\ \ddot{y} = \dot{z} \sin \theta + z \dot{\theta} \cos \theta = c_1 \sin \theta + z u_2 \cos \theta \end{cases}$$

c'est-à-dire :

$$\begin{pmatrix} \ddot{x} \\ \ddot{y} \end{pmatrix} = \begin{pmatrix} \cos \theta & -z \sin \theta \\ \sin \theta & z \cos \theta \end{pmatrix} \begin{pmatrix} c_1 \\ u_2 \end{pmatrix}$$

La matrice se trouve non singulière sauf dans le cas atypique où la variable z est nulle (ici, z peut être assimilée à la vitesse du véhicule). La méthode par bouclage linéarisant peut donc fonctionner. Prenons :

$$\begin{pmatrix} c_1 \\ u_2 \end{pmatrix} = \begin{pmatrix} \cos \theta & -z \sin \theta \\ \sin \theta & z \cos \theta \end{pmatrix}^{-1} \begin{pmatrix} v_1 \\ v_2 \end{pmatrix} = \begin{pmatrix} \cos \theta & \sin \theta \\ -\frac{\sin \theta}{z} & \frac{\cos \theta}{z} \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \end{pmatrix}$$

pour avoir un système bouclé de la forme :

$$\begin{pmatrix} \ddot{x} \\ \ddot{y} \end{pmatrix} = \begin{pmatrix} v_1 \\ v_2 \end{pmatrix}$$

La figure 2.4 illustre le bouclage linéarisant que nous venons d'effectuer.

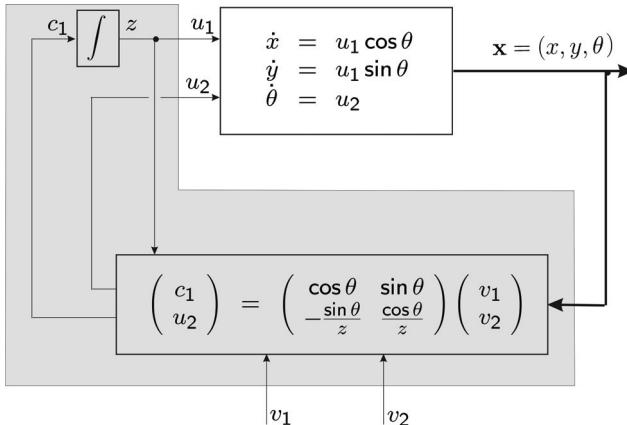


Figure 2.4. Bouclage linéarisant dynamique

Pour avoir tous les pôles en -1 , il nous faut prendre (voir [2.2]) :

$$\begin{pmatrix} c_1 \\ u_2 \end{pmatrix} = \begin{pmatrix} \cos \theta & \sin \theta \\ -\frac{\sin \theta}{z} & \frac{\cos \theta}{z} \end{pmatrix} \begin{pmatrix} (x_d - x) + 2(\dot{x}_d - z \cos \theta) + \ddot{x}_d \\ (y_d - y) + 2(\dot{y}_d - z \sin \theta) + \ddot{y}_d \end{pmatrix}$$

Les équations d'état du régulateur sont donc :

$$\begin{cases} \dot{z} = (\cos \theta)(x_d - x + 2(\dot{x}_d - z \cos \theta) + \ddot{x}_d) + (\sin \theta)(y_d - y + 2(\dot{y}_d - z \sin \theta) + \ddot{y}_d) \\ u_1 = z \\ u_2 = -\frac{\sin \theta}{z}(x_d - x + 2(\dot{x}_d - z \cos \theta) + \ddot{x}_d) + \frac{\cos \theta}{z} \cdot (y_d - y + 2(\dot{y}_d - z \sin \theta) + \ddot{y}_d) \end{cases}$$

2.5. Régulation d'un tricycle

2.5.1. Commande en vitesse et en cap

On considère le tricycle, représenté sur la figure 2.5. Son équation d'évolution est donnée par :

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{v} \\ \dot{\delta} \end{pmatrix} = \begin{pmatrix} v \cos \delta \cos \theta \\ v \cos \delta \sin \theta \\ v \sin \delta \\ u_1 \\ u_2 \end{pmatrix}$$

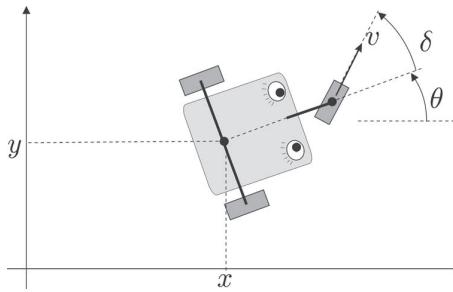


Figure 2.5. Robot tricycle à commander

Nous avons ici supposé que la distance entre le centre de l'essieu arrière et l'axe de la roue avant était de 1 m. Choisissons pour sortie le vecteur $\mathbf{y} = (v, \theta)$. Les dérivées premières des sorties y_1 et y_2 s'expriment par :

$$\dot{y}_1 = \dot{v} = u_1,$$

$$\dot{y}_2 = \dot{\theta} = v \sin \delta$$

Comme la dérivée \dot{y}_2 de y_2 ne fait pas apparaître l'entrée, on la dérive une nouvelle fois :

$$\ddot{y}_2 = \dot{v} \sin \delta + v \dot{\delta} \cos \delta = u_1 \sin \delta + u_2 v \cos \delta$$

Les expressions pour \dot{y}_1 et \ddot{y}_2 peuvent se réécrire sous forme matricielle :

$$\begin{pmatrix} \dot{y}_1 \\ \ddot{y}_2 \end{pmatrix} = \underbrace{\begin{pmatrix} 1 & 0 \\ \sin \delta & v \cos \delta \end{pmatrix}}_{\mathbf{A}(\mathbf{x})} \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}$$

En imposant le bouclage $\mathbf{u} = \mathbf{A}^{-1}(\mathbf{x}) \mathbf{v}$, où \mathbf{v} est la nouvelle entrée, notre système bouclé se réécrit :

$$\mathcal{S}_L : \begin{pmatrix} \dot{y}_1 \\ \ddot{y}_2 \end{pmatrix} = \begin{pmatrix} v_1 \\ v_2 \end{pmatrix}$$

et devient donc linéaire et découplé. Nous avons donc maintenant deux systèmes monovariables découplés. Le premier, d'ordre 1, peut être stabilisé par une commande proportionnelle. Pour le second, du deuxième ordre, une commande de type proportionnelle et dérivée est adaptée. Si $\mathbf{w} = (w_1, w_2)$ représente la consigne pour \mathbf{y} , cette commande s'exprime par :

$$\begin{cases} v_1 = (w_1 - y_1) + \dot{w}_1 \\ v_2 = (w_2 - y_2) + 2(\dot{w}_2 - \dot{y}_2) + \ddot{w}_2 \end{cases}$$

si l'on souhaite avoir pour pôles que des -1 (voir équation [2.2]). Donc les équations d'un régulateur par retour d'état pour notre système non linéaire sont données par :

$$\mathbf{u} = \begin{pmatrix} 1 & 0 \\ \sin \delta & v \cos \delta \end{pmatrix}^{-1} \begin{pmatrix} (w_1 - v) + \dot{w}_1 \\ w_2 - \theta + 2(\dot{w}_2 - \frac{v \sin \delta}{L}) + \ddot{w}_2 \end{pmatrix} \quad [2.10]$$

Notons que ce régulateur n'admet aucune variable d'état. Il s'agit donc d'un régulateur *statique*.

REMARQUE 2.3.– Puisque :

$$\det(\mathbf{A}(\mathbf{x})) = \frac{v \cos \delta}{L}$$

peut s'annuler, il existe des singularités pour lesquelles la commande \mathbf{u} n'est pas définie. Un traitement adapté doit être prévu lorsque de telles singularités sont rencontrées par le système.

2.5.2. Commande en position

Cherchons maintenant à faire suivre à notre tricycle une trajectoire désirée (x_d, y_d) . Pour cela, choisissons pour sortie le vecteur $\mathbf{y} = (x, y)$. On a :

$$\begin{cases} \dot{x} = v \cos \delta \cos \theta \\ \ddot{x} = \dot{v} \cos \delta \cos \theta - v \dot{\delta} \sin \delta \cos \theta - v \dot{\theta} \cos \delta \sin \theta \\ \quad = u_1 \cos \delta \cos \theta - v u_2 \sin \delta \cos \theta - v^2 \sin \delta \cos \delta \sin \theta \\ \dot{y} = v \cos \delta \sin \theta \\ \ddot{y} = \dot{v} \cos \delta \sin \theta - v \dot{\delta} \sin \delta \sin \theta + v \dot{\theta} \cos \delta \cos \theta \\ \quad = u_1 \cos \delta \sin \theta - v u_2 \sin \delta \sin \theta + v^2 \sin \delta \cos \delta \cos \theta \end{cases}$$

Ainsi :

$$\begin{pmatrix} \ddot{x} \\ \ddot{y} \end{pmatrix} = \underbrace{\begin{pmatrix} \cos \delta \cos \theta & -v \sin \delta \cos \theta \\ \cos \delta \sin \theta & -v \sin \delta \sin \theta \end{pmatrix}}_{\mathbf{A}(\mathbf{x})} \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} + \underbrace{\begin{pmatrix} -v^2 \sin \delta \cos \delta \sin \theta \\ v^2 \sin \delta \cos \delta \cos \theta \end{pmatrix}}_{\mathbf{b}(\mathbf{x})}$$

Or, le déterminant de $\mathbf{A}(\mathbf{x})$ est nul car les deux colonnes de la matrice $\mathbf{A}(\mathbf{x})$ sont colinéaires au vecteur $(\cos \theta, \sin \theta)$. Cela signifie que la partie commandable de l'accélération est forcément dans le sens du cap du véhicule. Ainsi, \ddot{x} et \ddot{y} ne pourront pas être commandées de façon indépendante. La méthode par bouclage linéarisant ne peut donc pas s'appliquer.

2.5.3. Choix d'une autre sortie

Afin d'éviter d'avoir une matrice $\mathbf{A}(\mathbf{x})$ singulière, choisissons maintenant pour sortie le centre de la roue avant. On a :

$$\mathbf{y} = \begin{pmatrix} x + \cos \theta \\ y + \sin \theta \end{pmatrix}$$

En dérivant une fois, nous avons :

$$\begin{pmatrix} \dot{y}_1 \\ \dot{y}_2 \end{pmatrix} = \begin{pmatrix} \dot{x} - \dot{\theta} \sin \theta \\ \dot{y} + \dot{\theta} \cos \theta \end{pmatrix} = v \begin{pmatrix} \cos \delta \cos \theta - \sin \delta \sin \theta \\ \cos \delta \sin \theta + \sin \delta \cos \theta \end{pmatrix} = v \begin{pmatrix} \cos(\delta + \theta) \\ \sin(\delta + \theta) \end{pmatrix}$$

En dérivant à nouveau, nous obtenons :

$$\begin{aligned} \begin{pmatrix} \ddot{y}_1 \\ \ddot{y}_2 \end{pmatrix} &= \begin{pmatrix} \dot{v} \cos(\delta + \theta) - v(\dot{\delta} + \dot{\theta}) \sin(\delta + \theta) \\ \dot{v} \sin(\delta + \theta) + v \cos(\delta + \theta) \end{pmatrix} \\ &= \begin{pmatrix} u_1 \cos(\delta + \theta) - v(u_2 + v \sin \delta) \sin(\delta + \theta) \\ u_1 \sin(\delta + \theta) + v(u_2 + v \sin \delta) \cos(\delta + \theta) \end{pmatrix} \end{aligned}$$

Et donc :

$$\begin{pmatrix} \ddot{y}_1 \\ \ddot{y}_2 \end{pmatrix} = \underbrace{\begin{pmatrix} \cos(\delta + \theta) & -v \sin(\delta + \theta) \\ \sin(\delta + \theta) & v \cos(\delta + \theta) \end{pmatrix}}_{\mathbf{A}(\mathbf{x})} \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} + \underbrace{v^2 \sin \delta \begin{pmatrix} -\sin(\delta + \theta) \\ \cos(\delta + \theta) \end{pmatrix}}_{\mathbf{b}(\mathbf{x})}$$

Le déterminant de $\mathbf{A}(\mathbf{x})$ n'est donc jamais nul, sauf si $v = 0$. La commande linéarisante est donc $\mathbf{u} = \mathbf{A}^{-1}(\mathbf{x}) \cdot (\mathbf{v} - \mathbf{b}(\mathbf{x}))$. Et donc la commande du tricycle (celle qui place tous les pôles en -1) s'exprime par :

$$\begin{aligned} \mathbf{u} &= \mathbf{A}^{-1}(\mathbf{x}) \left(\left(\begin{pmatrix} x_d \\ y_d \end{pmatrix} - \begin{pmatrix} x + \cos \theta \\ y + \sin \theta \end{pmatrix} \right) + 2 \left(\begin{pmatrix} \dot{x}_d \\ \dot{y}_d \end{pmatrix} - \begin{pmatrix} v \cos(\delta + \theta) \\ v \sin(\delta + \theta) \end{pmatrix} \right) \right. \\ &\quad \left. + \begin{pmatrix} \ddot{x}_d \\ \ddot{y}_d \end{pmatrix} - \mathbf{b}(\mathbf{x}) \right) \end{aligned}$$

où $\mathbf{w} = (x_d, y_d)$ est la trajectoire désirée pour la sortie \mathbf{y} .

2.6. Voilier

La commande automatique des robots voiliers [ROM 12] est un problème complexe du fait des fortes non-linéarités impliquées dans l'évolution du système.

Nous allons ici considérer le voilier de la figure 2.6 dont les équations d'état [JAU 04] sont données par :

$$\left\{ \begin{array}{l} \dot{x} = v \cos \theta \\ \dot{y} = v \sin \theta - 1 \\ \dot{\theta} = \omega \\ \dot{\delta}_s = u_1 \\ \dot{\delta}_r = u_2 \\ \dot{v} = f_s \sin \delta_s - f_r \sin \delta_r - v \\ \dot{\omega} = (1 - \cos \delta_s) f_s - \cos \delta_r \cdot f_r - \omega \\ f_s = \cos(\theta + \delta_s) - v \sin \delta_s \\ f_r = v \sin \delta_r \end{array} \right. \quad [2.11]$$

Il s'agit bien sûr d'un modèle normalisé où de nombreux coefficients (masses, longueurs, etc.) ont été choisis à 1 afin de faciliter les développements qui vont suivre. Le vecteur d'état $\mathbf{x} = (x, y, \theta, \delta_s, \delta_r, v, \omega)$, de dimension 7, est composé :

- des coordonnées de position, c'est-à-dire les coordonnées x, y du centre de gravité du bateau, l'orientation θ , et les angles δ_s et δ_r de la voile et du gouvernail ;
- des coordonnées cinématiques v et ω représentant respectivement la vitesse du centre de gravité et la vitesse angulaire du bateau.

Les entrées u_1 et u_2 du système sont les dérivées des angles δ_s et δ_r . Les indices s et r viennent de l'anglais *rudder* (gouvernail) et *sail* (voile).

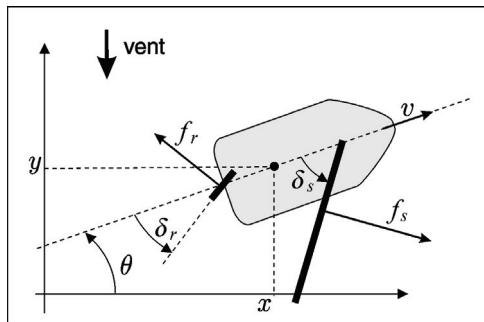


Figure 2.6. Robot voilier que l'on cherche à commander

2.6.1. Polaire des vitesses

Prenons pour sorties $\mathbf{y} = (\theta, v)$. La polaire des vitesses est l'ensemble des sorties acceptables (voir section 2.3.4), c'est-à-dire l'ensemble \mathbb{S}_y de tous les couples (θ, v) sur lesquels on peut se stabiliser. En régime permanent, nous avons :

$$\dot{\theta} = 0, \dot{\delta}_s = 0, \dot{\delta}_r = 0, \dot{v} = 0, \dot{\omega} = 0$$

Donc, d'après les équations d'état [2.11], nous obtenons :

$$\begin{aligned} \mathbb{S}_y = \{(\theta, v) \mid & f_s \sin \delta_s - f_r \sin \delta_r - v = 0 \\ & (1 - \cos \delta_s) f_s - \cos \delta_r f_r = 0 \\ & f_s = \cos(\theta + \delta_s) - v \sin \delta_s \\ & f_r = v \sin \delta_r \} \end{aligned}$$

Une méthode de calcul par intervalles [HER 10] nous permet d'obtenir l'approximation de la figure 2.7.

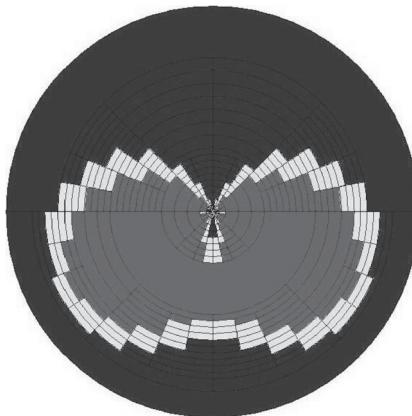


Figure 2.7. Encadrement intérieur (en gris clair) et extérieur (en gris foncé) de la polaire des vitesses

2.6.2. Retards différentiels

Aux équations d'état de notre voilier, on peut associer le *graphe des dépendances différentielles* entre les variables (voir figure 2.8). Dans ce graphe, une flèche pleine peut être interprétée, suivant la sensibilité du lecteur, soit comme une relation de cause à effet, soit comme un retard différentiel, soit comme une équation d'état. Une flèche en pointillé représente une dépendance algébrique (et non différentielle). Sur le graphe, on peut distinguer deux types de variables : les variables d'état, qui sont pointées par des flèches pleines, et les variables de liaison (en gris), qui sont pointées par des flèches en pointillés. La dérivée d'une variable d'état s'exprime comme une fonction algébrique de toutes variables qui lui sont directement en amont. De même, une variable de liaison est une fonction algébrique des variables qui lui sont directement en amont.

Le retard différentiel entre une variable et une entrée u_j est donc le nombre de minimal de flèches pleines à traverser pour atteindre cette variable à partir de u_j . Tout

comme dans [JAU 04], prenons pour sortie le vecteur $\mathbf{y} = (\delta_s, \theta)$. La matrice des retards différentiels est :

$$\mathbf{R} = \begin{pmatrix} 1 & \infty \\ 3 & 3 \end{pmatrix}$$

L'infini peut être interprété comme le fait qu'il n'existe aucun lien de cause à effet qui relie u_2 à δ_s . Les degrés relatifs sont donc $k_1 = 1$ et $k_2 = 3$.

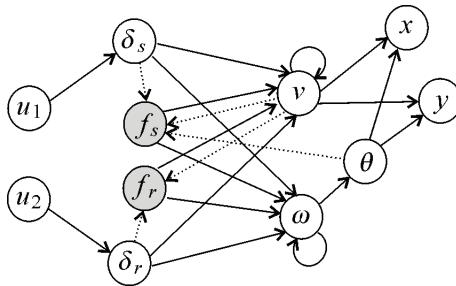


Figure 2.8. Graphe des retards différentiels pour le robot voilier

2.6.3. Méthode par bouclage linéarisant

Rappelons que les sorties (ce sont en fait des variables consignées) choisies sont l'ouverture de la voile $y_1 = \delta_s$ et le cap $y_2 = \theta$. Pour appliquer une méthode par bouclage linéarisant, il nous faut tout d'abord dériver les sorties autant de fois que le degré relatif le demande, c'est-à-dire trois fois pour θ et une fois pour δ_s . En regardant le graphe des dépendances différentielles, on comprend que pour exprimer $\ddot{\theta}$ en fonction de \mathbf{x} et \mathbf{u} , il va falloir faire de même avec $\ddot{\omega}, \dot{\omega}, \dot{\delta}_s, \dot{\delta}_r, \dot{f}_s, \dot{f}_r, \dot{v}$. Ainsi :

$$\begin{cases} \dot{v} = f_s \sin \delta_s - f_r \sin \delta_r - v \\ \dot{f}_s = -(\omega + u_1) \sin(\theta + \delta_s) - \dot{v} \sin \delta_s - vu_1 \cos \delta_s \\ \dot{f}_r = \dot{v} \sin \delta_r + vu_2 \cos \delta_r \\ \dot{\omega} = (1 - \cos \delta_s) \cdot f_s - \cos \delta_r \cdot f_r - \omega \\ \dot{\omega} = u_1 \sin \delta_s \cdot f_s + (1 - \cos \delta_s) \cdot \dot{f}_s + u_2 \sin \delta_r \cdot f_r - \cos \delta_r \cdot \dot{f}_r - \dot{\omega} \\ \ddot{\theta} = \ddot{\omega} \end{cases}$$

Nous avons :

$$\begin{aligned} \begin{pmatrix} \dot{y}_1 \\ \ddot{y}_2 \end{pmatrix} &= \begin{pmatrix} \dot{\delta}_s \\ \ddot{\theta} \end{pmatrix} = \underbrace{\begin{pmatrix} 1 & 0 \\ f_s \sin \delta_s & f_r \sin \delta_r \end{pmatrix}}_{\mathbf{A}_1(\mathbf{x})} \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} \\ &+ \underbrace{\begin{pmatrix} 0 & 0 \\ 1 - \cos \delta_s & -\cos \delta_r \end{pmatrix}}_{\mathbf{A}_2(\mathbf{x})} \begin{pmatrix} \dot{f}_s \\ \dot{f}_r \end{pmatrix} + \underbrace{\begin{pmatrix} 0 \\ -\dot{\omega} \end{pmatrix}}_{\mathbf{b}_1(\mathbf{x})} \end{aligned}$$

Or :

$$\begin{pmatrix} \dot{f}_s \\ \dot{f}_r \end{pmatrix} = \underbrace{\begin{pmatrix} -(\sin(\theta + \delta_s) + v \cos \delta_s) & 0 \\ 0 & v \cos \delta_r \end{pmatrix}}_{\mathbf{A}_3(\mathbf{x})} \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} + \underbrace{\begin{pmatrix} -\omega \sin(\theta + \delta_s) + \dot{v} \sin \delta_s \\ \dot{v} \sin \delta_r \end{pmatrix}}_{\mathbf{b}_2(\mathbf{x})}$$

Ainsi, on a une relation de la forme :

$$\begin{pmatrix} \dot{y}_1 \\ \ddot{y}_2 \end{pmatrix} = \mathbf{A}_1 \mathbf{u} + \mathbf{A}_2 (\mathbf{A}_3 \mathbf{u} + \mathbf{b}_2) + \mathbf{b}_1 \\ = (\mathbf{A}_1 + \mathbf{A}_2 \mathbf{A}_3) \mathbf{u} + \mathbf{A}_2 \mathbf{b}_2 + \mathbf{b}_1 = \mathbf{A} \mathbf{u} + \mathbf{b}$$

Pour imposer (\dot{y}_1, \ddot{y}_2) à une certaine consigne $\mathbf{v} = (v_1, v_2)$, il nous faut prendre :

$$\mathbf{u} = \mathbf{A}^{-1}(\mathbf{x})(\mathbf{v} - \mathbf{b}(\mathbf{x}))$$

Le système ainsi bouclé est régi par les équations différentielles :

$$\mathcal{S}_L : \begin{cases} \dot{y}_1 = v_1, \\ \ddot{y}_2 = v_2 \end{cases} \quad [2.12]$$

qui sont linéaires et découplées. Le système linéarisé est d'ordre 4 au lieu de 7. Nous avons donc perdu le contrôle sur trois variables qui se trouvent être x, y et v . La perte de contrôle sur x et y était prévisible (on veut que le bateau avance et il est naturel que cela corresponde à une instabilité pour ces deux variables x et y). Quant à la perte de contrôle sur v , elle est sans conséquence car la dynamique associée est stable. Comment en effet concevoir que le bateau puisse maintenir un cap et une ouverture de voile fixes, sans que sa vitesse ne converge vers une valeur finie ?

Déterminons maintenant les singularités de notre bouclage linéarisant. En calculant l'expression de $\mathbf{A}(\mathbf{x})$, on montre que :

$$\det(\mathbf{A}(\mathbf{x})) = f_r \sin \delta_r - v \cos^2 \delta_r \stackrel{[2.11]}{=} v (2 \sin^2 \delta_r - 1)$$

On a une singularité lorsque cette quantité est nulle, c'est-à-dire si :

$$v = 0 \text{ ou bien } \delta_r = \frac{\pi}{4} + k \frac{\pi}{2} \quad [2.13]$$

La singularité correspondant à $v = 0$ est relativement facile à comprendre : lorsque le bateau n'avance pas, on ne peut plus le commander. La condition sur l'angle du gouvernail δ_r est plus délicate à interpréter. En fait, la condition $\delta_r = \pm \frac{\pi}{4}$ se traduit

par une rotation maximale. La moindre action sur le gouvernail lorsque $\delta_r = \pm \frac{\pi}{4}$ se traduit par une rotation plus lente. C'est ce que traduit cette singularité.

Nous avons affaire à deux systèmes monovariables découplés. Notons $\mathbf{w} = (w_1, w_2)$ la consigne pour \mathbf{y} . Nous noterons parfois $\mathbf{w} = (\hat{\delta}_s, \theta)$ pour rappeler que w_1 et w_2 sont les consignes correspondant à l'angle d'ouverture de la voile et au cap. Choisissons le régulateur donné de type proportionnel et dérivée donné par :

$$\begin{cases} v_1 = (w_1 - y_1) + \dot{w}_1 \\ v_2 = (w_2 - y_2) + 3(w_2 - \dot{y}_2) + 3(\ddot{w}_2 - \ddot{y}_2) + \ddot{w}_2 \end{cases}$$

qui nous permet d'avoir des pôles pour le système bouclé tous égaux à -1 (voir équation [2.2]). En supposant la consigne \mathbf{w} constante, les équations d'état du régulateur par retour d'état pour notre système non linéaire sont données par :

$$\mathbf{u} = \mathbf{A}^{-1}(\mathbf{x}) \left(\begin{pmatrix} w_1 - \delta_s \\ w_2 - \theta - 3\dot{\theta} - 3\ddot{\theta} \end{pmatrix} - \mathbf{b}(\mathbf{x}) \right) \quad [2.14]$$

Or $\dot{\theta}$ et $\ddot{\theta}$ sont des fonctions analytiques de l'état \mathbf{x} . En effet, on a :

$$\begin{aligned} \dot{\theta} &= \omega \\ \ddot{\theta} &= (1 - \cos \delta_s) f_s - \cos \delta_r f_r - \omega \end{aligned}$$

L'équation [2.14] peut donc s'écrire sous la forme :

$$\mathbf{u} = \mathbf{r}(\mathbf{x}, \mathbf{w}) = \mathbf{r}(\mathbf{x}, \hat{\delta}_s, \theta) \quad [2.15]$$

Ce régulateur est *statique* car il n'admet aucune variable d'état.

2.6.4. Commande par la polaire des vitesses

Dans certaines situations, le plaisancier ne souhaite pas une totale autonomie de son bateau, mais juste une aide à la conduite. Il ne veut pas décider de l'angle des voiles mais tout simplement décider de sa vitesse et de son cap. En bref, il souhaite choisir un point sur la polaire et c'est au régulateur de s'occuper de la commande de bas niveau. En régime de croisière, on a :

$$\begin{cases} 0 = \bar{f}_s \sin \bar{\delta}_s - \bar{f}_r \sin \bar{\delta}_r - \bar{v} \\ 0 = (1 - \cos \bar{\delta}_s) \cdot \bar{f}_s - \cos \bar{\delta}_r \cdot \bar{f}_r \\ \bar{f}_s = \cos(\bar{\theta} + \bar{\delta}_s) - \bar{v} \sin \bar{\delta}_s \\ \bar{f}_r = \bar{v} \sin \bar{\delta}_r \end{cases}$$

Si $(\bar{\theta}, \bar{v})$ est dans la polaire des vitesses, on peut calculer $(\bar{f}_r, \bar{\delta}_r, \bar{f}_s, \bar{\delta}_s)$ (il existe au moins une solution, par définition de la polaire). Ainsi, il suffit d'injecter $(\bar{\theta}, \bar{\delta}_s)$ dans le régulateur [2.15] pour réaliser notre commande. La figure 2.9 illustre le rangement dans un port d'un voilier par cette approche [HER 10].

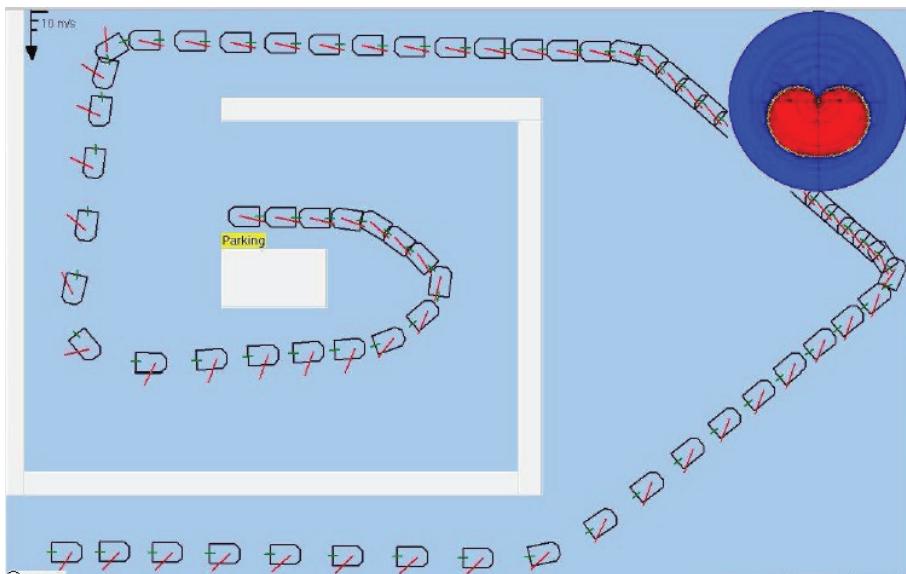


Figure 2.9. En utilisant une commande linéarisante, le robot se range à sa place dans le port ; en haut à droite est représentée la polaire des vitesses

2.7. Modèle cinématique et modèle dynamique

2.7.1. Principe

Les modèles dynamiques pour les robots sont de la forme :

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u})$$

où \mathbf{u} est le vecteur des forces extérieures (que nous commandons). La fonction \mathbf{f} fait intervenir des coefficients dynamiques (comme des masses, des moments d'inerties, des coefficients de frottement, etc.) et des coefficients géométriques (comme des longueurs). Les coefficients dynamiques ne sont généralement pas bien connus et peuvent changer dans le temps avec l'usure ou l'utilisation qui en est faite. Si maintenant, nous prenons comme nouvelle entrée, le vecteur \mathbf{a} des accélérations voulues aux points d'application des forces (dans le sens des forces), nous obtenons un nouveau modèle, dit *cinématique*, de la forme :

$$\dot{\mathbf{x}} = \varphi(\mathbf{x}, \mathbf{a})$$

mais dans ce nouveau modèle les coefficients dynamiques ont quasiment tous disparus. Il est possible de passer d'un modèle dynamique à un modèle cinématique par une commande, dite *grand gain*, de la forme :

$$\mathbf{u} = K(\mathbf{a} - \mathbf{a}(\mathbf{x}, \mathbf{u}))$$

où K est un réel très grand. La fonction $\mathbf{a}(\mathbf{x}, \mathbf{u})$ est une fonction permet d'obtenir l'accélération associée aux forces en fonction des forces \mathbf{u} et de l'état \mathbf{x} . En pratique, on ne cherche pas à exprimer $\mathbf{a}(\mathbf{x}, \mathbf{u})$ dans la commande, mais plutôt à mesurer $\mathbf{a}(\mathbf{x}, \mathbf{u})$ par des accéléromètres. La commande qui sera réellement implémentée sera :

$$\mathbf{u} = K(\mathbf{a} - \tilde{\mathbf{a}})$$

où $\tilde{\mathbf{a}}$ correspond au vecteur des accélérations mesurées. Ainsi, nous avons :

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, K(\mathbf{a} - \mathbf{a}(\mathbf{x}, \mathbf{u}))) \Leftrightarrow \dot{\mathbf{x}} = \varphi(\mathbf{x}, \mathbf{a})$$

Le simple bouclage grand gain nous a permis de débarrasser de nombreux paramètres dynamiques et de passer d'un système incertain à un système fiable, avec des coefficients géométriques qui sont bien connus. Cette rétroaction grand gain est connue en électronique sous le nom d'amplificateur opérationnel, où elle est utilisée avec la même idée de robustesse.

Passer d'un modèle dynamique à un modèle cinématique possède les avantages suivants :

- la commande linéarisante développée dans ce chapitre nécessite l'utilisation d'un modèle fiable comme un modèle cinématique. Si les coefficients (qui ne sont pas mesurés) sont mal connus (comme dans le cas des systèmes dynamiques), la commande linéarisante ne fonctionnera pas en pratique ;
- le modèle cinématique est plus facile à mettre en équations. Il n'est pas nécessaire d'avoir un modèle dynamique pour obtenir ce dernier ;
- les servo-moteurs (voir section 2.7.3) incorporent cette commande grand gain. On peut donc les voir comme des amplificateurs opérationnels mécaniques.

Nous allons illustrer l'idée dans le paragraphe suivant, à travers l'exemple du pendule inversé.

2.7.2. Exemple du pendule inversé

Considérons le pendule inversé, formé d'un pendule posé en équilibre instable sur un chariot roulant, comme représenté sur la figure 2.10.

2.7.2.1. Modèle dynamique

La quantité u est la force exercée sur le chariot de masse M , x indique la position du chariot, θ est l'angle entre le pendule et la verticale. Les équations d'état s'écrivent sous la forme :

$$\frac{d}{dt} \begin{pmatrix} x \\ \theta \\ \dot{x} \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} \dot{x} \\ \dot{\theta} \\ \frac{-m \sin \theta (\ell \dot{\theta}^2 - g \cos \theta)}{M + m \sin^2 \theta} \\ \frac{\sin \theta ((M+m)g - m \ell \dot{\theta}^2 \cos \theta)}{\ell (M + m \sin^2 \theta)} \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ \frac{1}{M + m \sin^2 \theta} \\ \frac{\cos \theta}{\ell (M + m \sin^2 \theta)} \end{pmatrix} u \quad [2.16]$$

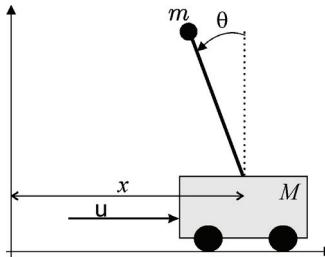


Figure 2.10. Pendule inversé à modéliser et à commander

2.7.2.2. Modèle cinématique

Reprendons les équations d'état du pendule inversé, mais prenons comme entrée, non plus la force, mais l'accélération $a = \ddot{x}$. Nous obtenons, d'après [2.16] :

$$a = \frac{1}{M + m \sin^2 \theta} \left(-m \sin \theta (\ell \dot{\theta}^2 - g \cos \theta) + u \right) \quad [2.17]$$

Donc :

$$\begin{aligned} \ddot{\theta} &\stackrel{[2.16]}{=} \frac{\sin \theta ((M+m)g - m \ell \dot{\theta}^2 \cos \theta)}{\ell (M + m \sin^2 \theta)} + \frac{\cos \theta}{\ell (M + m \sin^2 \theta)} u \\ &\stackrel{[2.17]}{=} \frac{\sin \theta ((M+m)g - m \ell \dot{\theta}^2 \cos \theta)}{\ell (M + m \sin^2 \theta)} \\ &+ \frac{\cos \theta}{\ell (M + m \sin^2 \theta)} \left(m \sin \theta (\ell \dot{\theta}^2 - g \cos \theta) + (M + m \sin^2 \theta) a \right) \\ &= \frac{1}{\ell (M + m \sin^2 \theta)} \left((M + m)g \sin \theta - gm \sin \theta \cos^2 \theta + (M + m \sin^2 \theta) \cos \theta \cdot a \right) \\ &= \frac{g \sin \theta}{\ell} + \frac{\cos \theta}{\ell} a \end{aligned}$$

Notons que cette relation aurait pu être obtenue directement en remarquant que :

$$\ell \ddot{\theta} = \underbrace{a \cdot \cos \theta}_{\text{accélération de } A \text{ qui contribue à la rotation}} + \underbrace{g \cdot \sin \theta}_{\text{accélération de } B}$$

REMARQUE 2.4.- Pour obtenir cette relation de façon plus rigoureuse, il faudrait écrire la dérivée temporelle de la formule de composition des vitesses, c'est-à-dire :

$$\dot{\mathbf{v}}_A = \dot{\mathbf{v}}_B + \overrightarrow{AB} \wedge \vec{\omega}$$

et écrire cette formule dans le repère du pendule. On obtient :

$$\begin{pmatrix} a \cos \theta \\ -a \sin \theta \\ 0 \end{pmatrix} = \begin{pmatrix} -g \sin \theta \\ n \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ \ell \\ 0 \end{pmatrix} \wedge \begin{pmatrix} 0 \\ 0 \\ \dot{\omega} \end{pmatrix}$$

où n correspond à l'accélération normale de la masse m . On obtient ainsi la relation cherchée et l'accélération normale $n = -a \sin \theta$ qui ne sera pas utilisée.

Finalement, le modèle cinématique s'écrit :

$$\frac{d}{dt} \begin{pmatrix} x \\ \theta \\ \dot{x} \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} \dot{x} \\ \dot{\theta} \\ 0 \\ \frac{g \sin \theta}{\ell} \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 1 \\ \frac{\cos \theta}{\ell} \end{pmatrix} a \quad [2.18]$$

Ce modèle, dit *cinématique*, ne fait intervenir que des positions, des vitesses et des accélérations. Il est beaucoup plus simple que le modèle dynamique et fait intervenir moins de coefficients. En revanche, il correspond moins à la réalité puisque la vraie entrée est une force et non pas une accélération. En pratique, nous pouvons passer du modèle dynamique [2.16] d'entrée u au modèle cinématique [2.18] d'entrée a en calculant u par une *commande proportionnelle* de type *grand gain* de la forme :

$$u = K(a - \ddot{x}) \quad [2.19]$$

avec K très grand et où a est une nouvelle entrée.

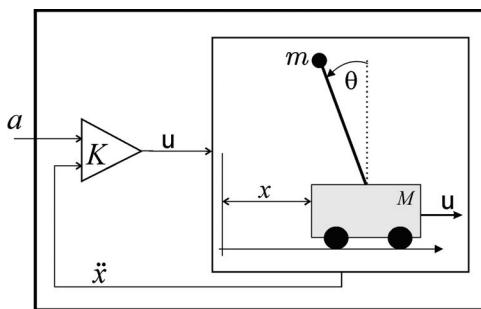


Figure 2.11. Le pendule inversé, bouclé par un grand gain K , se comporte comme un modèle cinématique

L'accélération \ddot{x} peut être mesurée par un accéléromètre. Si K est suffisamment grand, nous aurons bien évidemment la commande u que nous donnera l'accélération a voulue, c'est-à-dire que l'on pourra admettre que $\ddot{x} = a$. Ainsi, le système [2.16] pourra se décrire par les équations d'état [2.18] qui ne font intervenir aucun

des paramètres inertIELS du système. Un régulateur qui sera conçu sur le modèle cinématique sera donc plus robuste que celui conçu sur le système dynamique car le régulateur fonctionnera quels que soient les masses m, M , les moments d'inertie, les frottements, etc. Rappelons que cette commande de type grand gain est très proche du principe de l'ampli-op. En plus d'être plus robuste, une telle approche permet d'avoir un modèle plus simple et plus facile à obtenir. Pour l'implémentation de la commande [2.19], il ne faut bien sûr pas chercher à utiliser les équations d'état [2.16], pour exprimer \ddot{x} , mais plutôt à mesurer \ddot{x} . C'est cette mesure qui nous permet d'avoir une commande indépendante des paramètres dynamiques.

Reprendons notre pendule inversé et cherchons à faire osciller le pendule de gauche à droite avec un angle désiré de la forme $\theta_d = \sin t$. Appliquons pour cela une commande linéarisante. On a :

$$\ddot{\theta} = \frac{g \sin \theta}{\ell} + \frac{\cos \theta}{\ell} a$$

Donc on prendra :

$$a = \frac{\ell}{\cos \theta} \left(v - \frac{g \sin \theta}{\ell} \right)$$

où v est la nouvelle entrée. On choisira alors :

$$v = (\theta_d - \theta) + 2(\dot{\theta}_d - \dot{\theta}) + \ddot{\theta}_d = \sin t - \theta + 2 \cos t - 2\dot{\theta} - \sin t$$

Soit finalement :

$$\begin{aligned} u &= K(a - \ddot{x}) \\ &= K \left(\frac{\ell}{\cos \theta} \left(\sin t - \theta + 2 \cos t - 2\dot{\theta} - \sin t - \frac{g \sin \theta}{\ell} \right) - \ddot{x} \right) \end{aligned}$$

Notons que les paramètres inertIELS ne rentrent pas en compte dans cette commande. Cette commande nous assure que le système respectera sa consigne d'angle. En revanche, la position du chariot peut diverger, car u ne dépend pas de x . La dynamique de x est cachée et de plus, elle est ici instable. Cette dynamique cachée est classiquement appelée la *dynamique des zéros*.

2.7.3. Servo-moteurs

Un système mécanique se commande par des forces ou des couples et obéit à un modèle dynamique qui dépend de nombreux coefficients mal connus. Ce même système mécanique représenté par un modèle cinématique se commande par des positions, des vitesses ou des accélérations. Le modèle cinématique dépend

de coefficients géométriques bien connus et est beaucoup plus facile à mettre en équations. En pratique, on passe d'un modèle dynamique à son équivalent cinématique en rajoutant des servo-moteurs. En résumé, un servo-moteur est un moteur à courant continu avec un circuit électrique de commande et un capteur (de position, de vitesse ou d'accélération). Le circuit de commande calcule la tension u à donner au moteur afin que la quantité mesurée par le capteur corresponde à la consigne w . Il existe trois types de servo-moteurs :

– le *servo position*. Le capteur mesure la position (ou l'angle) x du moteur et la loi de commande s'exprime par $u = K(x - w)$. Si K est grand, on peut conclure que $x \simeq w$;

– le *servo vitesse*. Le capteur mesure la vitesse (ou la vitesse angulaire) \dot{x} du moteur et la loi de commande s'exprime par $u = K(\dot{x} - w)$. Si K est grand, on a $\dot{x} \simeq w$;

– le *servo accélération*. Le capteur mesure l'accélération (tangentielle ou angulaire) \ddot{x} du moteur et la loi de commande s'exprime par $u = K(\ddot{x} - w)$. Si K est grand, on a $\ddot{x} \simeq w$. C'est ce type de servo-moteur que nous avons choisi pour le pendule inversé.

Ainsi, lorsque nous voulons commander un système mécanique, l'utilisation de servo-moteurs permet (i) d'avoir un modèle plus facile à obtenir, (ii) d'avoir un modèle avec moins de coefficients et plus proche de la réalité et (iii) avoir une commande plus robuste par rapport à toute modification des coefficients dynamiques du système.

2.8. Exercices

Exercice 2.1. Manivelle

On considère le robot manipulateur, ou *manivelle*, de la figure 2.12 (à gauche). Ce robot possède deux bras de longueur ℓ_1 et ℓ_2 . Ses deux degrés de liberté notés x_1 et x_2 sont représentés sur la figure. Les entrées u_1, u_2 du système sont les vitesses angulaires des bras (c'est-à-dire $u_1 = \dot{x}_1, u_2 = \dot{x}_2$). On prend pour sortie le vecteur $\mathbf{y} = (y_1, y_2)$ correspondant à l'extrémité du second bras.

1) Donner les équations d'état du robot. On prendra pour vecteur d'état $\mathbf{x} = (x_1, x_2)$.

2) On souhaite que \mathbf{y} suive une consigne \mathbf{w} qui décrit un cercle cible (voir figure 2.12 à droite). Cette consigne satisfait :

$$\mathbf{w} = \mathbf{c} + r \cdot \begin{pmatrix} \cos t \\ \sin t \end{pmatrix}$$

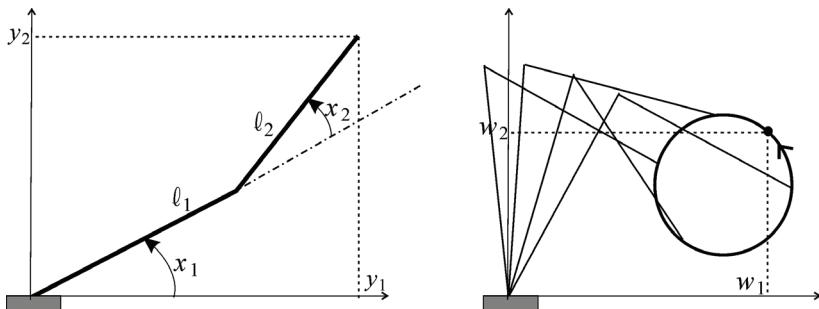


Figure 2.12. Robot manipulateur dont l'organe terminal doit suivre un cercle

Donner l'expression d'une loi de commande qui permette de réaliser cette tâche. On utilisera une méthode de linéarisation par bouclage et on placera les pôles en -1 .

3) Etudier les singularités de la commande.

4) Prenons le cas $\ell_1 = \ell_2$, $c = (3, 4)$ et $r = 1$. Pour quelles valeurs de ℓ_1 sommes-nous sûrs de pouvoir bouger librement sur le cercle cible, sans rencontrer de singularités ?

5) Faire un programme MATLAB qui illustre cette loi de commande.

Exercice 2.2. Les trois bacs

On considère un système d'écoulement dans trois bacs comme représenté sur la figure 2.13.

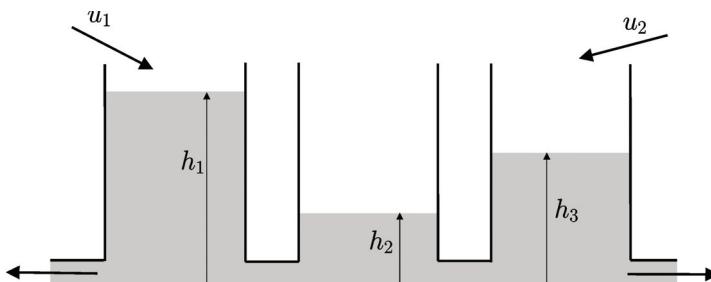


Figure 2.13. Système constitué de trois bacs contenant de l'eau et reliés par deux canaux

Ce système est décrit par les équations d'état suivantes :

$$\begin{cases} \dot{h}_1 = -\alpha(h_1) - \alpha(h_1 - h_2) + u_1 \\ \dot{h}_2 = \alpha(h_1 - h_2) - \alpha(h_2 - h_3) \\ \dot{h}_3 = -\alpha(h_3) + \alpha(h_2 - h_3) + u_2 \\ y_1 = h_1 \\ y_2 = h_3 \end{cases}$$

où $\alpha(h) = a \cdot \text{sign}(h) \sqrt{2g|h|}$. Nous avons ici choisi pour sorties les hauteurs dans le premier et le troisième bacs.

- 1) Proposer un bouclage qui rende le système linéaire et découplé.
- 2) Proposer un régulateur proportionnel et intégral pour le système linéarisé.
- 3) Donner les équations d'état du régulateur ainsi obtenu.
- 4) Proposer un programme MATLAB simulant le système et sa loi de commande.

Exercice 2.3. Train de robots

On considère un robot A (à gauche sur la figure 2.14) décrit par les équations d'état suivantes (modèle char) :

$$\begin{cases} \dot{x}_a = v_a \cos \theta_a \\ \dot{y}_a = v_a \sin \theta_a \\ \dot{\theta}_a = u_{a1} \\ \dot{v}_a = u_{a2} \end{cases}$$

où v_a est la vitesse du robot, θ_a son orientation et (x_a, y_a) les coordonnées de son centre. Nous supposons mesurer avec une très grande précision les variables d'état de notre robot.

- 1) Calculer \ddot{x}_a, \ddot{y}_a en fonction de $x_a, y_a, v_a, \theta_a, u_{a1}, u_{a2}$.
- 2) Proposer un régulateur permettant de décrire suivre la trajectoire :

$$\begin{cases} \hat{x}_a(t) = L_x \sin(\omega t) \\ \hat{y}_a(t) = L_y \cos(\omega t) \end{cases}$$

avec $\omega = 0.1$, $L_x = 15$ et $L_y = 7$. Utiliser pour cela une méthode de linéarisation par bouclage.

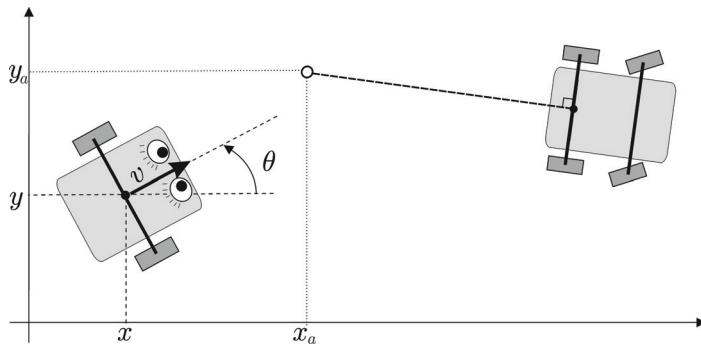


Figure 2.14. Notre robot (avec des yeux) suit un véhicule (ici une voiture) dont les équations d'état sont inconnues. Cette voiture possède un point d'accroche (petit cercle blanc) qui n'existe que par la pensée, auquel il nous faut nous raccrocher

3) Un deuxième robot B du même type que A souhaite suivre le robot A (voir figure 2.15). On définit un point d'accroche virtuel de coordonnées (\hat{x}_b, \hat{y}_b) afin que le véhicule B (à gauche sur la figure) puisse s'accrocher à notre robot A. Nous pouvons lui envoyer, par voie hertzienne, les informations associées au point d'accroche.

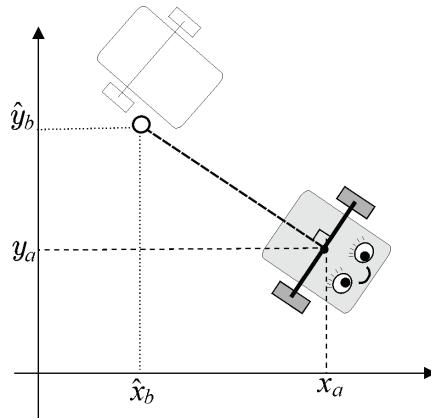


Figure 2.15. Le robot B (en pointillés) doit suivre le robot A

Ce point sera positionné à l'arrière du robot A à une distance égale à ℓ de notre point de référence (x_a, y_a) . Donner les expressions de ces quantités en fonction de l'état de notre véhicule A.

4) Simuler sous MATLAB ce deuxième véhicule B avec son régulateur qui suit le robot A.

5) Rajouter un troisième robot C, qui suit le robot B suivant le même principe. Simuler le tout sous MATLAB.

6) En jouant sur le fait que dans cet exercice, la trajectoire de référence est connue exactement, proposer un régulateur qui permettra aux robots B et C de suivre exactement le robot A.

Exercice 2.4. Commande d'un robot sous-marin 3D

On considère le robot sous-marin déjà étudié à l'exercice 1.9. Ce robot est décrit par les équations d'état suivantes :

$$\begin{cases} \dot{p}_x = v \cos \theta \cos \psi \\ \dot{p}_y = v \cos \theta \sin \psi \\ \dot{p}_z = -v \sin \theta \\ \dot{v} = u_1 \\ \dot{\psi} = \frac{\sin \varphi}{\cos \theta} \cdot v \cdot u_2 + \frac{\cos \varphi}{\cos \theta} \cdot v \cdot u_3 \\ \dot{\theta} = \cos \varphi \cdot v \cdot u_2 - \sin \varphi \cdot v \cdot u_3 \\ \dot{\varphi} = -0.1 \sin \varphi + \tan \theta \cdot v \cdot (\sin \varphi \cdot u_2 + \cos \varphi \cdot u_3) \end{cases}$$

où (p_x, p_y, p_z) est la position de son centre et (ψ, θ, φ) sont les trois angles d'Euler. Ses entrées sont l'accélération tangentielle u_1 , le tangage u_2 et le lacet u_3 . Proposer un régulateur capable de réguler le robot autour d'une cycloïde d'équation :

$$\begin{pmatrix} x_d \\ y_d \\ z_d \end{pmatrix} = \begin{pmatrix} R \cdot \sin(f_1 t) + R \cdot \sin(f_2 t) \\ R \cdot \cos(f_1 t) + R \cdot \cos(f_2 t) \\ R \cdot \sin(f_3 t) \end{pmatrix}$$

avec $f_1 = 0.01$, $f_2 = 6f_1$, $f_3 = 3f_1$ et $R = 20$. Pour la régulation, on choisira une constante de temps de 5 secondes. Simuler le comportement du régulateur sous MATLAB.

Exercice 2.5. Système plat

On considère le système décrit par les équations d'état :

$$\begin{cases} \dot{x}_1 = x_1 + x_2 \\ \dot{x}_2 = x_2^2 + u \\ y = x_1 \end{cases}$$

1) Un système est dit *plat* s'il existe deux fonctions ϕ, ψ telles que :

$$\begin{cases} \mathbf{x} = \phi(y, \dot{y}, \dots, y^{(r-1)}) \\ u = \psi(y, \dot{y}, \dots, y^{(r-1)}, y^{(r)}) \end{cases}$$

Montrer que notre système est plat. On donnera les expressions de ϕ et ψ .

2) Donner l'expression d'un bouclage linéarisant pour le système de la forme $u = \gamma(v, y, \dot{y})$, où v est une nouvelle entrée. Ce bouclage transforme notre système en un système décrit par $\ddot{y} = v$.

3) On propose de commander le système $\ddot{y} = v$ par une commande proportionnelle et dérivée. Donner l'expression de la commande $v = \eta(w, \dot{w}, \ddot{w}, y, \dot{y})$, où w est une consigne variant dans le temps, qui nous permette d'avoir une erreur $e = w - y$ qui converge vers zéro. On prendra les pôles tous égaux à -1 .

4) Déduire des questions précédentes, un régulateur par retour de sortie pour le système initial de la forme $u = \rho(w, \dot{w}, \ddot{w}, y, \dot{y})$ qui soit tel que l'erreur $e = w - y$ converge vers 0.

Exercice 2.6. Poursuite

On considère deux robots décrits par les équations d'état suivantes :

$$\begin{cases} \dot{x}_1 = u_1 \cos \theta_1 \\ \dot{y}_1 = u_1 \sin \theta_1 \\ \dot{\theta}_1 = u_2 \end{cases} \text{ et } \begin{cases} \dot{x}_2 = v_1 \cos \theta_2 \\ \dot{y}_2 = v_1 \sin \theta_2 \\ \dot{\theta}_2 = v_2 \end{cases}$$

Dans cet exercice, le robot 1 cherche à suivre le robot 2 (voir figure 2.16).

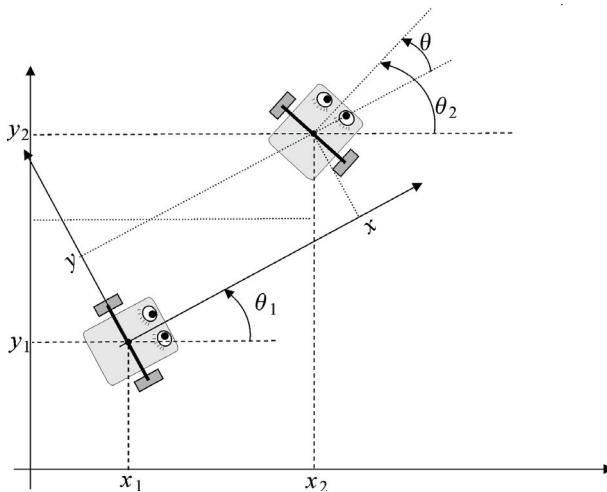


Figure 2.16. Le robot 1 cherche à poursuivre le robot 2

1) Soit $\mathbf{x} = (x, y, \theta)$ le vecteur position du robot 2 dans le repère du robot 1. Montrer que \mathbf{x} satisfait une équation d'état de la forme :

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{v}, \mathbf{u})$$

2) On suppose que les variables de commande v_1 et v_2 du robot 2 sont connues (un polynôme en t par exemple). Proposer un régulateur nous générant \mathbf{u} pour avoir $x = w_1$ et $y = w_2$, où $\mathbf{w} = (w_1, w_2)$ correspond à une consigne en position relative. Les pôles pour l'erreur seront fixés à -1 .

3) Etudier les singularités de cette commande.

4) Illustrer cette loi de commande sous MATLAB dans la situation où le robot 1 veut pointer sur le robot 2 tout en se tenant à une distance de 10 m.

Exercice 2.7. Commande du robot SAUCISSE

On considère le robot sous-marin représenté sur la figure 2.17.

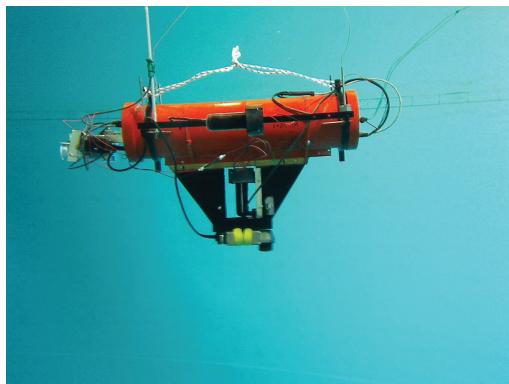


Figure 2.17. Robot SAUCISSE dans une piscine

Il s'agit du robot SAUCISSE, fabriqué par des étudiants de l'ENSTA Bretagne pour le concours SAUC'E (*Student Autonomous Underwater Challenge European*). Il comporte trois propulseurs. Les deux propulseurs 1 et 2 à gauche et à droite permettent d'agir sur la vitesse du robot et sur sa vitesse angulaire. Le propulseur 3 permet d'agir sur la profondeur du robot par l'intermédiaire. Ce robot est stable en roulis et en tangage et on supposera que ses angles de gîte φ et d'assiette θ sont toujours nuls. Les équations d'état du robot sont les suivantes :

$$\begin{cases} \dot{x} = v_x \\ \dot{y} = v_y \\ \dot{z} = v_z \\ \dot{\psi} = \omega \\ \dot{v}_x = u_1 \cos \psi \\ \dot{v}_y = u_1 \sin \psi \\ \dot{v}_z = u_3 \\ \dot{\omega} = u_2 \end{cases}$$

Remarquons qu'aucune contrainte de non holonomie n'a été supposée dans notre modèle. Le vecteur vitesse du robot (v_x, v_y) n'est pas nécessairement dans son axe comme cela était le cas dans le modèle char. Le robot peut donc marcher en crabe. En revanche, la propulsion est nécessairement dans le sens de l'axe du robot. Si l'on se limite au plan horizontal, ce modèle est connu sous le nom d'*aéroglisseur* (*hovercraft* en anglais).

1) Donner le graphe des dépendances différentielles associées à ce système.

2) Choisissons pour sortie le vecteur $\mathbf{y} = (x, y, z)$. Donner la matrice des retards différentiels et en déduire les degrés relatifs. Que peut-on en déduire ?

3) Afin d'équilibrer les retards différentiels en retardant u_1 , rajoutons deux intégrateurs devant u_1 . Notre nouveau système admettra pour nouvelles entrées $\mathbf{a} = (a_1, a_2, a_3)$ avec :

$$\begin{cases} \ddot{u}_1 = a_1 \\ u_2 = a_2 \\ u_3 = a_3 \end{cases} \quad [2.20]$$

Quelles sont les nouvelles équations d'état du système retardé ? Donner le graphe des dépendances différentielles ainsi que la matrice des retards différentiels associés.

4) Effectuer une linéarisation par bouclage du système retardé.

5) En déduire la commande correspondante pour notre robot. On placera tous les pôles en -1 .

2.9. Corrections

Correction de l'exercice 2.1 (manivelle)

1) Les équations d'état de la manivelle sont :

$$\begin{cases} \dot{x}_1 = u_1 \\ \dot{x}_2 = u_2 \\ y_1 = \ell_1 \cos x_1 + \ell_2 \cos (x_1 + x_2) \\ y_2 = \ell_1 \sin x_1 + \ell_2 \sin (x_1 + x_2) \end{cases}$$

2) En dérivant la sortie, nous obtenons :

$$\begin{aligned} \dot{y}_1 &= -\ell_1 \dot{x}_1 \sin x_1 - \ell_2 (\dot{x}_1 + \dot{x}_2) \sin (x_1 + x_2) = -\ell_1 u_1 \sin x_1 - \ell_2 (u_1 + u_2) \sin (x_1 + x_2) \\ \dot{y}_2 &= \ell_1 \dot{x}_1 \cos x_1 + \ell_2 (\dot{x}_1 + \dot{x}_2) \cos (x_1 + x_2) = \ell_1 u_1 \cos x_1 + \ell_2 (u_1 + u_2) \cos (x_1 + x_2) \end{aligned}$$

Ainsi :

$$\dot{\mathbf{y}} = \underbrace{\begin{pmatrix} -\ell_1 \sin x_1 - \ell_2 \sin(x_1 + x_2) & -\ell_2 \sin(x_1 + x_2) \\ \ell_1 \cos x_1 + \ell_2 \cos(x_1 + x_2) & \ell_2 \cos(x_1 + x_2) \end{pmatrix}}_{\mathbf{A}(\mathbf{x})} \mathbf{u}$$

On prend $\mathbf{u} = \mathbf{A}^{-1}(\mathbf{x}) \cdot \mathbf{v}$ pour avoir deux intégrateurs découplés. On choisit ensuite la commande proportionnelle :

$$\begin{aligned} \mathbf{v} = (\mathbf{w} - \mathbf{y}) + \dot{\mathbf{w}} &= \mathbf{c} + r \cdot \begin{pmatrix} \cos t \\ \sin t \end{pmatrix} - \begin{pmatrix} \ell_1 \cos x_1 + \ell_2 \cos(x_1 + x_2) \\ \ell_1 \sin x_1 + \ell_2 \sin(x_1 + x_2) \end{pmatrix} \\ &\quad + r \begin{pmatrix} -\sin t \\ \cos t \end{pmatrix} \end{aligned}$$

qui nous place tous les pôles en -1 .

3) Par multilinéarité du déterminant, on a :

$$\begin{aligned} \det \mathbf{A}(\mathbf{x}) &= -\ell_1 \ell_2 \det \underbrace{\begin{pmatrix} \sin x_1 & \sin(x_1 + x_2) \\ \cos x_1 & \cos(x_1 + x_2) \end{pmatrix}}_{=\sin x_2} \\ &\quad + \ell_2^2 \det \underbrace{\begin{pmatrix} -\sin(x_1 + x_2) & -\sin(x_1 + x_2) \\ \cos(x_1 + x_2) & \cos(x_1 + x_2) \end{pmatrix}}_{=0} \end{aligned}$$

Ce déterminant est nul si $\ell_1 \ell_2 \sin x_2 = 0$. C'est-à-dire si, $x_2 = k\pi$, $k \in \mathbb{Z}$ ou si un des deux bras est de longueur nulle.

4) Si $\ell_1 = \ell_2 \neq 0$, nous avons une singularité si $\sin x_2 = 0$. Ainsi, soit les deux bras sont repliés (et alors \mathbf{y} est pas sur le cercle) ou bien soit les deux bras sont tendus (voir figure 2.18). Dans le cas bras tendu (qui nous intéresse) le point \mathbf{y} est sur le cercle de rayon $2\ell_1$ qui intersecte le cercle cible si :

$$\ell_1 + \ell_2 \in \sqrt{4^2 + 3^2} \pm 1 = 5 \pm 1 = [4, 6]$$

où $\sqrt{4^2 + 3^2}$ correspond à la distance du centre du cercle cible à l'origine. On aura une singularité sur le cercle si $\ell_1 = \ell_2 \in [2, 3]$. Si l'on veut pouvoir se promener librement sur le cercle, il faudra choisir $\ell_1 = \ell_2 > 3$.

5) Le programme est rangé dans le fichier `crank.m`.

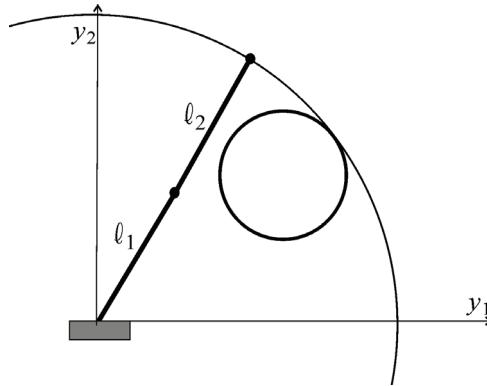


Figure 2.18. Illustration d'une singularité, lorsque les bras sont tendus. L'organe terminal risque de ne pas parvenir à faire le suivi du cercle

Correction de l'exercice 2.2 (les trois bacs)

1) Les dérivées des sorties y_1 et y_2 s'expriment par :

$$\begin{aligned}\dot{y}_1 &= \dot{h}_1 = -\alpha(h_1) - \alpha(h_1 - h_2) + u_1 \\ \dot{y}_2 &= \dot{h}_3 = -\alpha(h_3) + \alpha(h_2 - h_3) + u_2\end{aligned}$$

ou, sous forme vectorielle :

$$\begin{pmatrix} \dot{y}_1 \\ \dot{y}_2 \end{pmatrix} = \underbrace{\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}}_{\mathbf{A}(\mathbf{x})} \mathbf{u} + \underbrace{\begin{pmatrix} -\alpha(h_1) - \alpha(h_1 - h_2) \\ -\alpha(h_3) + \alpha(h_2 - h_3) \end{pmatrix}}_{\mathbf{b}(\mathbf{x})}$$

Le bouclage :

$$\mathbf{u} = \mathbf{A}^{-1}(\mathbf{x})(\mathbf{v} - \mathbf{b}(\mathbf{x})) = \mathbf{v} - \begin{pmatrix} -\alpha(h_1) - \alpha(h_1 - h_2) \\ -\alpha(h_3) + \alpha(h_2 - h_3) \end{pmatrix}$$

où \mathbf{v} est notre nouvelle entrée, rend notre système linéaire. Plus précisément, le système ainsi bouclé a la forme :

$$\begin{cases} \dot{y}_1 = v_1 \\ \dot{y}_2 = v_2 \end{cases}$$

2) Cherchons maintenant à réguler ce système linéaire par un régulateur composé de deux régulateurs PI (proportionnel et intégrale) de la forme :

$$\begin{cases} v_1(t) = \alpha_0(w_1(t) - y_1(t)) + \alpha_{-1} \int_0^t (w_1(\tau) - y_1(\tau)) d\tau + \dot{w}_1 \\ v_2(t) = \beta_0(w_2(t) - y_2(t)) + \beta_{-1} \int_0^t (w_2(\tau) - y_2(\tau)) d\tau + \dot{w}_2 \end{cases}$$

où w_1 et w_2 sont les nouvelles consignes pour y_1 et y_2 . Si l'on souhaite avoir pour pôles uniquement des -1 , il faut :

$$\begin{cases} s^2 + \alpha_0 s + \alpha_{-1} = (s + 1)^2 = s^2 + 2s + 1 \\ s^2 + \beta_0 s + \beta_{-1} = (s + 1)^2 = s^2 + 2s + 1 \end{cases}$$

c'est-à-dire $\alpha_{-1} = \beta_{-1} = 1, \alpha_0 = \beta_0 = 2$.

3) Les équations d'état pour le régulateur sont donc :

$$\begin{cases} \dot{z}_1 = w_1 - y_1 \\ \dot{z}_2 = w_2 - y_2 \\ v_1 = z_1 + 2(w_1 - y_1) + \dot{w}_1 \\ v_2 = z_2 + 2(w_2 - y_2) + \dot{w}_2 \end{cases}$$

Les équations d'état d'un régulateur par retour d'état pour notre système non linéaire sont donc :

$$\begin{cases} \dot{z}_1 = w_1 - h_1 \\ \dot{z}_2 = w_2 - h_3 \\ u_1 = z_1 + 2(w_1 - h_1) + \dot{w}_1 + \alpha(h_1) + \alpha(h_1 - h_2) \\ u_2 = z_2 + 2(w_2 - h_3) + \dot{w}_2 + \alpha(h_3) - \alpha(h_2 - h_3) \end{cases}$$

4) Le programme est rangé dans le fichier `pools.m`.

Correction de l'exercice 2.3 (train de robots)

1) On a :

$$\begin{cases} \ddot{x}_a = \dot{v}_a \cos \theta_a - v_a \dot{\theta}_a \sin \theta_a = u_{a2} \cos \theta_a - v_a u_{a1} \sin \theta_a \\ \ddot{y}_a = \dot{v}_a \sin \theta_a + v_a \dot{\theta}_a \cos \theta_a = u_{a2} \sin \theta_a + v_a u_{a1} \cos \theta_a \end{cases}$$

Donc :

$$\begin{pmatrix} \ddot{x}_a \\ \ddot{y}_a \end{pmatrix} = \underbrace{\begin{pmatrix} -v_a \sin \theta_a & \cos \theta_a \\ v_a \cos \theta_a & \sin \theta_a \end{pmatrix}}_{\mathbf{A}(v_a, \theta_a)} \begin{pmatrix} u_{a1} \\ u_{a2} \end{pmatrix}$$

2) En utilisant une méthode par bouclage linéarisant, on obtient :

$$\begin{aligned} \begin{pmatrix} u_{a1} \\ u_{a2} \end{pmatrix} &= \mathbf{A}^{-1}(v_a, \theta_a) \cdot \begin{pmatrix} (\hat{x}_a - x_a) + 2 \left(\frac{d\hat{x}_a}{dt} - v_a \cos \theta_a \right) + \frac{d^2}{dt^2} \hat{x}_a \\ (\hat{y}_a - y_a) + 2 \left(\frac{d\hat{y}_a}{dt} - v_a \sin \theta_a \right) + \frac{d^2}{dt^2} \hat{y}_a \end{pmatrix} \\ &= \mathbf{A}^{-1}(v_a, \theta_a) \cdot \begin{pmatrix} L_x \sin \omega t - x_a + 2\omega L_x \cos \omega t - 2v_a \cos \theta_a - \omega^2 L_x \sin \omega t \\ L_y \cos \omega t - y_a - 2\omega L_y \sin \omega t + 2v_a \sin \theta_a - \omega^2 L_y \cos \omega t \end{pmatrix} \end{aligned}$$

3) Appliquons le régulateur obtenu à la question précédente. Il est donné par :

$$\begin{pmatrix} u_{b1} \\ u_{b2} \end{pmatrix} = \mathbf{A}^{-1}(v_b, \theta_b) \cdot \begin{pmatrix} (\hat{x}_b - x_b) + 2 \left(\frac{d\hat{x}_b}{dt} - v_b \cos \theta_b \right) + \frac{d^2 \hat{x}_b}{dt^2} \\ (\hat{y}_b - y_b) + 2 \left(\frac{d\hat{y}_b}{dt} - v_b \sin \theta_b \right) + \frac{d^2 \hat{y}_b}{dt^2} \end{pmatrix}$$

On a :

$$\hat{x}_b = x_a - \ell \cos \theta_a$$

$$\hat{y}_b = y_a - \ell \sin \theta_a$$

$$\frac{d}{dt} \hat{x}_b = \dot{x}_a + \ell \dot{\theta}_a \sin \theta_a = v_a \cos \theta_a + \ell u_{1a} \sin \theta_a$$

$$\frac{d}{dt} \hat{y}_b = \dot{y}_a - \ell \dot{\theta}_a \cos \theta_a = v_a \sin \theta_a - \ell u_{1a} \cos \theta_a$$

Pour disposer de $\frac{d^2}{dt^2} \hat{x}_b$, $\frac{d^2}{dt^2} \hat{y}_b$, il nous faudrait avoir \dot{u}_{1a} , ce qui n'est pas le cas. Ici, nous allons tout simplement supposer que ces deux quantités sont nulles et espérer que cette approximation ne se traduira pas par une instabilité du système. Nous n'avons donc plus l'assurance d'une convergence exponentielle de l'erreur vers 0, mais on remarquera qu'en pratique le comportement reste acceptable.

5) Il suffit de reprendre le même régulateur que dans la question précédente. Le programme est disponible dans le fichier `train.m`. La figure 2.19 illustre le comportement de notre train. Le char A tourne autour d'une ellipse, le char B suit le char A et le char C suit le char B.

6) Il suffit de prendre des régulateurs complètement indépendants avec un simple retard temporel, par exemple :

$$\begin{pmatrix} \hat{x}_a \\ \hat{y}_a \end{pmatrix} = \begin{pmatrix} L_x \sin \omega t \\ L_y \cos \omega t \end{pmatrix}, \quad \begin{pmatrix} \hat{x}_b \\ \hat{y}_b \end{pmatrix} = \begin{pmatrix} L_x \sin \omega(t-1) \\ L_y \cos \omega(t-1) \end{pmatrix}$$

$$\text{et } \begin{pmatrix} \hat{x}_c \\ \hat{y}_c \end{pmatrix} = \begin{pmatrix} L_x \sin \omega(t-2) \\ L_y \cos \omega(t-2) \end{pmatrix}$$

Cette loi de commande peut être considérée comme centralisée car elle nécessite un superviseur capable d'envoyer des consignes cohérentes à tous les robots.

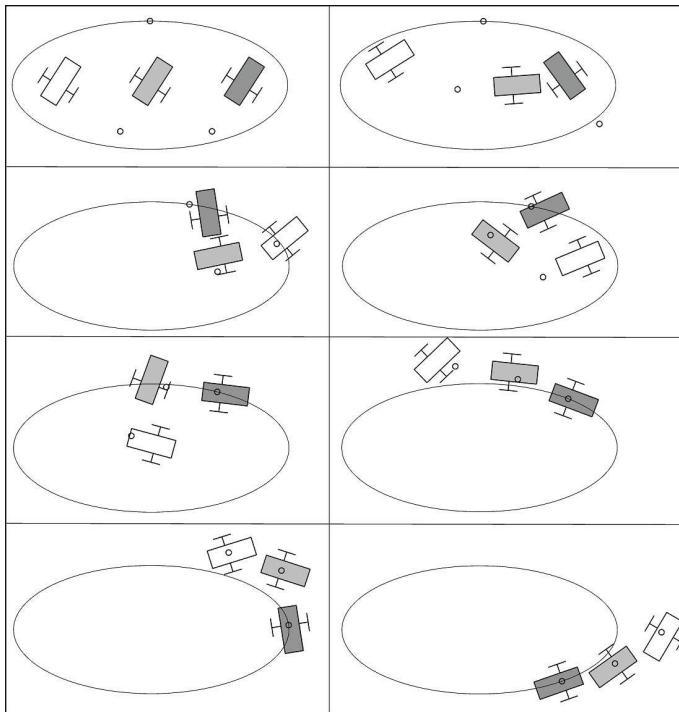


Figure 2.19. Illustration de l'exécution du programme ; le char A est en gris foncé, le char B est en gris clair et le char C est en blanc ; les consignes et points d'accroche sont représentés par des petits cercles

Correction de l'exercice 2.4 (commande d'un robot sous-marin 3D)

Choisissons pour sortie, le vecteur position du robot \mathbf{p} . Nous avons :

$$\begin{aligned}
 \ddot{\mathbf{p}}(t) &= \begin{pmatrix} \dot{v} \cos \theta \cos \psi - v \dot{\theta} \sin \theta \cos \psi - v \dot{\psi} \cos \theta \sin \psi \\ \dot{v} \cos \theta \sin \psi - v \dot{\theta} \sin \theta \sin \psi + v \dot{\psi} \cos \theta \cos \psi \\ -\dot{v} \sin \theta - v \dot{\theta} \cos \theta \end{pmatrix} \\
 &= \begin{pmatrix} \cos \theta \cos \psi & -v \cos \theta \sin \psi & -v \sin \theta \cos \psi \\ \cos \theta \sin \psi & v \cos \theta \cos \psi & -v \sin \theta \sin \psi \\ -\sin \theta & 0 & -v \cos \theta \end{pmatrix} \begin{pmatrix} \dot{v} \\ \dot{\psi} \\ \dot{\theta} \end{pmatrix} \\
 &= \underbrace{\begin{pmatrix} \cos \theta \cos \psi & -v \cos \theta \sin \psi & -v \sin \theta \cos \psi \\ \cos \theta \sin \psi & v \cos \theta \cos \psi & -v \sin \theta \sin \psi \\ -\sin \theta & 0 & -v \cos \theta \end{pmatrix}}_{\mathbf{A}(\mathbf{x})} \begin{pmatrix} 1 & 0 & 0 \\ 0 & \frac{v \sin \varphi}{\cos \theta} & v \frac{\cos \varphi}{\cos \theta} \\ 0 & v \cos \varphi & -v \sin \varphi \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix}
 \end{aligned}$$

En bouclant le système par la commande $\mathbf{u} = \mathbf{A}^{-1}(\mathbf{x})\mathbf{v}$, nous obtenons le système linéaire découplé $\ddot{\mathbf{p}} = \mathbf{v}$. Ce système possède trois entrées et trois sorties. Si $\mathbf{w} = (w_1, w_2, w_3)$ représente la trajectoire à suivre pour \mathbf{p} , nous pouvons considérer la commande suivante :

$$\mathbf{v} = 0.04 \cdot (\mathbf{w} - \mathbf{p}) + 0.4 \cdot (\dot{\mathbf{w}} - \dot{\mathbf{p}}) + \ddot{\mathbf{w}}$$

pour avoir des pôles égaux à -0.2 (ce qui correspond à un polynôme caractéristique $P(s) = (s + 0.2)^2$). Le régulateur par retour d'état pour notre robot est donc :

$$\mathbf{u} = \mathbf{A}^{-1}(\mathbf{x}) \cdot \left(0.04 \cdot (\mathbf{w} - \mathbf{p}) + 0.4 \cdot \left(\dot{\mathbf{w}} - \begin{pmatrix} v \cos \theta \cos \psi \\ v \cos \theta \sin \psi \\ -v \sin \theta \end{pmatrix} \right) + \ddot{\mathbf{w}} \right)$$

avec :

$$\begin{aligned} \mathbf{w} &= \begin{pmatrix} R \sin(f_1 t) + R \sin(f_2 t) \\ R \cos(f_1 t) + R \cos(f_2 t) \\ R \sin(f_3 t) \end{pmatrix} \\ \dot{\mathbf{w}} &= \begin{pmatrix} R f_1 \cos(f_1 t) + R f_2 \cos(f_2 t) \\ -R f_1 \sin(f_1 t) - R f_2 \sin(f_2 t) \\ R f_3 \cos(f_3 t) \end{pmatrix} \\ \ddot{\mathbf{w}} &= \begin{pmatrix} -R f_1^2 \sin(f_1 t) - R f_2^2 \sin(f_2 t) \\ -R f_1^2 \cos(f_1 t) - R f_2^2 \cos(f_2 t) \\ -R f_3^2 \sin(f_3 t) \end{pmatrix} \end{aligned}$$

La simulation avec graphisme du robot est donnée dans le fichier `auv3d.m`. Le programme reprend les résultats de l'exercice 1.9.

Correction de l'exercice 2.5 (système plat)

1) On a :

$$y = x_1, \dot{y} = x_1 + x_2$$

$$\ddot{y} = \dot{x}_1 + \dot{x}_2 = x_1 + x_2 + x_2^2 + u$$

Donc :

$$x_1 = y, \quad x_2 = \dot{y} - x_1 = \dot{y} - y$$

$$u = \ddot{y} - (x_1 + x_2 + x_2^2) = \ddot{y} - \left(y + \dot{y} - y + (\dot{y} - y)^2 \right) = \ddot{y} - \dot{y} - \dot{y}^2 - y^2 + 2y\dot{y}$$

Ainsi :

$$\phi(y, \dot{y}) = \begin{pmatrix} y \\ \dot{y} - y \end{pmatrix}$$

$$\psi(y, \dot{y}, \ddot{y}) = \ddot{y} - \dot{y} - \dot{y}^2 - y^2 + 2y\dot{y}$$

2) Puisque $u = \ddot{y} - \dot{y} - \dot{y}^2 + y^2 + 2y\dot{y}$, en choisissant :

$$u = v - \dot{y} - \dot{y}^2 - y^2 + 2y\dot{y}$$

on obtient $\ddot{y} = v$.

3) En prenant $v = (w - y) + 2(\dot{w} - \dot{y}) + \ddot{w}$, l'équation différentielle sur l'erreur est $\ddot{e} + 2\dot{e} + e = 0$. Le polynôme caractéristique est $s^2 + 2s + 1 = (s + 1)^2$. L'erreur converge donc vers zéro en e^{-t} .

4) On a :

$$\begin{aligned} u &= v - \dot{y} - \dot{y}^2 + y^2 + 2y\dot{y} \\ &= (w - y) + 2(\dot{w} - \dot{y}) + \ddot{w} - \dot{y} - \dot{y}^2 + y^2 + 2y\dot{y} \\ &= w + 2\dot{w} + \ddot{w} - y - y^2 - 3\dot{y} - \dot{y}^2 + 2y\dot{y} \end{aligned}$$

Correction de l'exercice 2.6 (poursuite)

1) L'écart relatif entre les deux robots (c'est-à-dire la pose du robot 2 exprimée dans le repère du robot 1) s'écrit :

$$\begin{pmatrix} x \\ y \\ \theta \end{pmatrix} = \begin{pmatrix} \cos \theta_1 & \sin \theta_1 & 0 \\ -\sin \theta_1 & \cos \theta_1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_2 - x_1 \\ y_2 - y_1 \\ \theta_2 - \theta_1 \end{pmatrix}$$

Dérivons par rapport à t les deux premières composantes de cette relation. Nous obtenons :

$$\begin{aligned} \begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix} &= \begin{pmatrix} \cos \theta_1 & \sin \theta_1 \\ -\sin \theta_1 & \cos \theta_1 \end{pmatrix} \begin{pmatrix} \dot{x}_2 - \dot{x}_1 \\ \dot{y}_2 - \dot{y}_1 \end{pmatrix} \\ &\quad + \dot{\theta}_1 \begin{pmatrix} -\sin \theta_1 & \cos \theta_1 \\ -\cos \theta_1 & -\sin \theta_1 \end{pmatrix} \begin{pmatrix} x_2 - x_1 \\ y_2 - y_1 \end{pmatrix} \end{aligned}$$

Or :

$$\left\{ \begin{array}{l} \begin{pmatrix} \dot{\theta}_1 \\ \dot{x}_2 \\ \dot{y}_2 \\ \dot{x}_1 \\ \dot{y}_1 \end{pmatrix} = u_2 \\ \begin{pmatrix} x_2 - x_1 \\ y_2 - y_1 \end{pmatrix} = \begin{pmatrix} \cos \theta_1 & -\sin \theta_1 \\ \sin \theta_1 & \cos \theta_1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \end{array} \right.$$

Donc :

$$\begin{aligned}
 \begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix} &= v_1 \begin{pmatrix} \cos \theta_1 & \sin \theta_1 \\ -\sin \theta_1 & \cos \theta_1 \end{pmatrix} \begin{pmatrix} \cos \theta_2 \\ \sin \theta_2 \end{pmatrix} - u_1 \begin{pmatrix} \cos \theta_1 & \sin \theta_1 \\ -\sin \theta_1 & \cos \theta_1 \end{pmatrix} \begin{pmatrix} \cos \theta_1 \\ \sin \theta_1 \end{pmatrix} \\
 &\quad + u_2 \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \\
 &= v_1 \begin{pmatrix} \cos(\theta_2 - \theta_1) \\ \sin(\theta_2 - \theta_1) \end{pmatrix} - u_1 \begin{pmatrix} 1 \\ 0 \end{pmatrix} + u_2 \begin{pmatrix} y \\ -x \end{pmatrix} \\
 &= \begin{pmatrix} -u_1 + v_1 \cos(\theta_2 - \theta_1) + u_2 y \\ v_1 \sin(\theta_2 - \theta_1) - u_2 x \end{pmatrix}
 \end{aligned}$$

De plus :

$$\dot{\theta} = \dot{\theta}_2 - \dot{\theta}_1 = v_2 - u_2$$

Ainsi :

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} -u_1 + v_1 \cos \theta + u_2 y \\ v_1 \sin \theta - u_2 x \\ v_2 - u_2 \end{pmatrix}$$

2) On a :

$$\begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix} = \begin{pmatrix} v_1 \cos \theta \\ v_1 \sin \theta \end{pmatrix} + \begin{pmatrix} -1 & y \\ 0 & -x \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}$$

On propose le régulateur qui suit :

$$\begin{pmatrix} u_1 \\ u_2 \end{pmatrix} = \begin{pmatrix} -1 & y \\ 0 & -x \end{pmatrix}^{-1} \left(\begin{pmatrix} w_1 - x \\ w_2 - y \end{pmatrix} + \begin{pmatrix} \dot{w}_1 \\ \dot{w}_2 \end{pmatrix} - \begin{pmatrix} v_1 \cos \theta \\ v_1 \sin \theta \end{pmatrix} \right)$$

Ainsi, en combinant les deux équations précédentes, on obtient les équations sur l'erreur en x et y :

$$\begin{pmatrix} w_1 - x \\ w_2 - y \end{pmatrix} + \begin{pmatrix} \dot{w}_1 - \dot{x} \\ \dot{w}_2 - \dot{y} \end{pmatrix} = 0$$

Ces deux erreurs ont pour polynôme caractéristique $P(s) = s + 1$, ce qui nous dit que l'erreur converge vers 0 en e^{-t} .

3) On a une singularité de notre loi de commande lorsque :

$$\det \begin{pmatrix} -1 & y \\ 0 & -x \end{pmatrix} = 0$$

c'est-à-dire lorsque $x = 0$, ou encore lorsque le robot 2 est à gauche ou à droite du robot 1.

4) Le programme (voir fichier `pursuit.m`) est le suivant :

```
xa=[-10 ;-10 ;0] ; xb=[-5 ;-5 ;0] ; dt=0.02 ;
for t=0 :dt :50,
v=[3 ;sin(0.2*t)] ;
x=[cos(xa(3)),sin(xa(3)),0 ;-sin(xa(3)), cos(xa(3)),0 ;0,0,1]*(xb-xa) ;
w=[10 ;0] ; dw=[0 ;0] ;
u=inv([-1 x(2) ;0 -x(1)])*(w-x(1 :2)+dw-v(1)*[cos(x(3)) ;sin(x(3))]) ;
xa=xa+f(xa,u)*dt ; xb=xb+f(xb,v)*dt ;
end ;
```

Ce programme utilise la fonction d'évolution suivante :

```
function xdot = f(x,u)
xdot=[u(1)*cos(x(3)) ; u(1)*sin(x(3)) ; u(2)] ;
end
```

Correction de l'exercice 2.7 (Commande du robot SAUCISSE)

1) Le graphe des dépendances différentielles est donné par la figure 2.20.

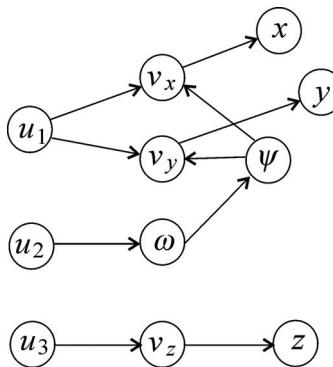


Figure 2.20. Graphe des dépendances différentielles du robot sous-marin

2) La matrice des retards différentiels est :

$$\mathbf{R} = \begin{pmatrix} 2 & 4 & \infty \\ 2 & 4 & \infty \\ \infty & \infty & 2 \end{pmatrix}$$

Les degrés relatifs sont $k_1 = 2$, $k_2 = 2$ et $k_3 = 2$. Mais puisque la deuxième colonne est telle que $\forall i, r_{i2} > k_i$, la méthode par bouclage linéarisant aboutira forcément à une matrice $\mathbf{A}(\mathbf{x})$ singulière pour tout vecteur d'état \mathbf{x} (voir section 2.3.3).

3) Nous obtenons pour nouveau système les équations d'état :

$$\begin{cases} \dot{x} = v_x \\ \dot{y} = v_y \\ \dot{z} = v_z \\ \dot{\psi} = \omega \\ \dot{v}_x = c_1 \cos \psi \\ \dot{v}_y = c_1 \sin \psi \\ \dot{v}_z = a_3 \\ \dot{\omega} = a_2 \\ \dot{c}_1 = b_1 \\ \dot{b}_1 = a_1 \end{cases}$$

où b_1 et c_1 sont les nouvelles variables d'état attachées à nos intégrateurs. Le graphe des dépendances différentielles est donné sur la figure 2.21. La matrice des retards différentiels est désormais donnée par :

$$\mathbf{R} = \begin{pmatrix} 4 & 4 & \infty \\ 4 & 4 & \infty \\ \infty & \infty & 2 \end{pmatrix}$$

Les degrés relatifs sont $k_1 = 4$, $k_2 = 4$ et $k_3 = 2$.

4) Afin de linéariser par bouclage le système retardé, il nous faut calculer \ddot{x} et \ddot{y} en fonction de \mathbf{x} et \mathbf{u} . On a :

$$\begin{cases} \ddot{x} = c_1 \cos \psi \\ \ddot{x} = b_1 \cos \psi - c_1 \omega \sin \psi \\ \ddot{y} = a_1 \cos \psi - 2b_1 \omega \sin \psi - c_1 a_2 \sin \psi - c_1 \omega^2 \cos \psi \\ \ddot{y} = c_1 \sin \psi \\ \ddot{y} = b_1 \sin \psi + c_1 \omega \cos \psi \\ \ddot{y} = a_1 \sin \psi + 2b_1 \omega \cos \psi + c_1 a_2 \cos \psi - c_1 \omega^2 \sin \psi \\ \ddot{z} = a_3 \end{cases}$$

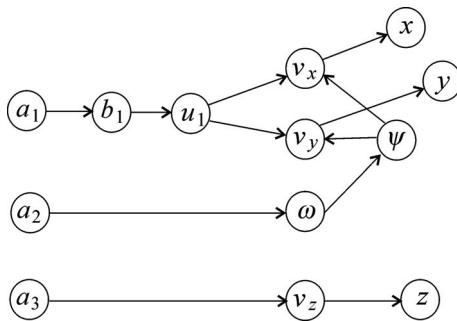


Figure 2.21. Graphe des retards différentiels après ajout des intégrateurs

Donc :

$$\begin{pmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{pmatrix} = \underbrace{\begin{pmatrix} \cos \psi & -c_1 \sin \psi & 0 \\ \sin \psi & c_1 \cos \psi & 0 \\ 0 & 0 & 1 \end{pmatrix}}_{\mathbf{A}(\mathbf{x}, c_1)} \begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} + \underbrace{\begin{pmatrix} -2b_1 \omega \sin \psi - c_1 \omega^2 \cos \psi \\ 2b_1 \omega \cos \psi - c_1 \omega^2 \sin \psi \\ 0 \end{pmatrix}}_{\mathbf{b}(\mathbf{x}, b_1, c_1)}$$

Afin d'avoir un système bouclé de la forme :

$$\begin{pmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{pmatrix} = \begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix}$$

Nous allons prendre :

$$\begin{aligned} \mathbf{a} &= \mathbf{A}^{-1}(\mathbf{x}, c_1) (\mathbf{v} - \mathbf{b}(\mathbf{x}, b_1, c_1)) \\ &= \begin{pmatrix} \cos \psi & -c_1 \sin \psi & 0 \\ \sin \psi & c_1 \cos \psi & 0 \\ 0 & 0 & 1 \end{pmatrix}^{-1} \left(\begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix} - \begin{pmatrix} -2b_1 \omega \sin \psi - c_1 \omega^2 \cos \psi \\ 2b_1 \omega \cos \psi - c_1 \omega^2 \sin \psi \\ 0 \end{pmatrix} \right) \\ &= \begin{pmatrix} \cos \psi & \sin \psi & 0 \\ -\frac{\sin \psi}{c_1} & \frac{\cos \psi}{c_1} & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix} + \begin{pmatrix} \omega^2 c_1 \\ \frac{-2\omega b_1}{c_1} \\ 0 \end{pmatrix} \end{aligned}$$

[2.21]

5) En notant que $\dot{b}_1 = a_1$ et $u_2 = a_2$, on en déduit que les équations d'état du linéariseur dynamique sont :

$$\begin{cases} \dot{c}_1 = b_1 \\ \dot{b}_1 = v_1 \cos \psi + v_2 \sin \psi + \omega^2 c_1 \\ \begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix} = \begin{pmatrix} c_1 \\ \frac{1}{c_1} (v_2 \cos \psi - v_1 \sin \psi - 2\omega b_1) \\ v_3 \end{pmatrix} \end{cases} \quad [2.22]$$

Ce bouclage linéarisant admet \mathbf{v} comme vecteur d'entrée, \mathbf{u} comme sortie et (c_1, b_1) comme vecteur d'état. Afin d'avoir tous les pôles en -1 , nous choisissons (voir équation [2.2]) :

$$\begin{cases} v_1 = (w_1 - y_1) + 4(\dot{w}_1 - \dot{y}_1) + 6(\ddot{w}_1 - \ddot{y}_1) + 4(\ddot{w}_1 - \ddot{y}_1) + \ddot{w}_1 \\ v_2 = (w_2 - y_2) + 4(\dot{w}_2 - \dot{y}_2) + 6(\ddot{w}_2 - \ddot{y}_2) + 4(\ddot{w}_2 - \ddot{y}_2) + \ddot{w}_2 \\ v_3 = (w_3 - y_3) + 2(\dot{w}_3 - \dot{y}_3) + \ddot{w}_3 \end{cases}$$

c'est-à-dire :

$$\begin{cases} v_1 = (x_d - x) + 4(\dot{x}_d - v_x) + 6(\ddot{x}_d - c_1 \cos \psi) \\ \quad + 4(\ddot{x}_d - (b_1 \cos \psi - c_1 \omega \sin \psi)) + \ddot{x}_d \\ v_2 = (y_d - y) + 4(\dot{y}_d - v_y) + 6(\ddot{y}_d - c_1 \sin \psi) \\ \quad + 4(\ddot{y}_d - (b_1 \sin \psi + c_1 \omega \cos \psi)) + \ddot{y}_d \\ v_3 = (z_d - z) + 2(\dot{z}_d - v_z) + \ddot{z}_d \end{cases} \quad [2.23]$$

où (x_d, y_d, z_d) correspond à la trajectoire désirée pour le centre du robot. En combinant les deux équations [2.22] et [2.23], nous obtenons le régulateur demandé qui est bien de la forme :

$$\begin{cases} \dot{\mathbf{x}}_r = \mathbf{f}_r(\mathbf{x}, \mathbf{x}_r, t) \\ \mathbf{u} = \mathbf{g}_r(\mathbf{x}, \mathbf{x}_r, t) \end{cases}$$

où $\mathbf{x} = (x, y, z, \psi, v_x, v_y, v_z, \omega)$ est le vecteur d'état du robot et où $\mathbf{x}_r = (c_1, b_1)$ est le vecteur d'état du régulateur. Ce dernier correspond aux deux intégrateurs rajoutés devant le système dans le but de pouvoir éliminer la singularité dans la linéarisation. L'approche présentée sur cet exemple est appelée *méthode par bouclage linéarisant dynamique*. En effet, le fait d'avoir rajouté des intégrateurs nous impose une représentation d'état pour notre régulateur, contrairement à la situation sans singularité qui nous amenait à une relation statique pour notre régulateur.

Chapitre 3

Commande sans modèle

Lorsque l'on implémente un régulateur pour un robot et que l'on lance les premiers tests, ça ne fonctionne rarement du premier coup et vient le problème du débogage. Il se peut que la boussole subisse des perturbations électromagnétiques, qu'elle soit positionnée à l'envers, qu'il y ait un problème de conversion d'unité dans les capteurs, que les moteurs saturent ou encore qu'il y ait une erreur de signe dans les équations du régulateur. Le problème du débogage est un problème complexe et il est judicieux de respecter le *principe de continuité* : chaque avancée dans la construction d'un robot doit être d'ampleur raisonnable et doit être validée avant de continuer la construction. Ainsi, pour un robot, il est souhaitable d'implémenter une commande intuitive simple et facile à déboguer, avant de mettre en place une commande plus évoluée. Ce principe ne peut pas toujours s'appliquer. En revanche, si l'on dispose d'une bonne compréhension *a priori* de la loi de commande à appliquer, alors un tel principe de continuité est envisageable. Parmi les robots mobiles pour lesquels une commande pragmatique peut être imaginée, on peut distinguer au moins deux sous-classes :

– *les robots véhicules*. Il s'agit de systèmes construits par l'homme pour être pilotés par l'homme comme le vélo, le voilier, la voiture, etc. On cherchera à copier la loi de commande utilisée par l'homme et à la transformer en un algorithme ;

– *les robots biomimétiques*. Ce sont des robots inspirés par le déplacement des êtres vivants. On a pu les observer longuement et en déduire la stratégie qu'a pu développer la nature pour réaliser sa loi de commande. Il s'agit de l'approche *biomimétique* (voir par exemple [BOY 06]). Nous ne mettons pas dans cette classe les robots marcheurs car, bien que nous sachions tous marcher, il est quasiment impossible de savoir quelle loi de commande nous utilisons pour cela. Ainsi, la conception d'une loi de commande pour les robots marcheurs [CHE 07] ne pourra se faire sans une modélisation mécanique complète de la marche et sans l'utilisation de méthodes issues de l'automatique théorique, comme celles évoquées au chapitre précédent.

Pour ces deux classes, on ne dispose souvent pas de modèles simples et fiables (c'est le cas par exemple du voilier ou du vélo). On revanche la compréhension forte que l'on a de ces derniers nous permettra de construire une loi de commande robuste.

L'objectif de ce chapitre est de montrer sur quelques exemples comment concevoir de telles lois de commande que l'on qualifiera de *commande mimétique* (on cherche à mimer la façon dont les humains ou les animaux procèdent) ou bien de *commande sans modèle* (on utilise pas les équations d'état du robot pour concevoir le régulateur). Bien que les approches sans modèle aient été largement théorisées (voir par exemple [FLI 13]), nous allons ici utiliser le plus possible l'intuition que l'on a du fonctionnement de notre robot.

3.1. Commande sans modèle d'un robot char

Afin d'illustrer le principe de la commande sans modèle, prenons le cas d'un robot de type char décrit par les équations :

$$\begin{cases} \dot{x} = v \cos \theta \\ \dot{y} = v \sin \theta \\ \dot{\theta} = u_2 \\ \dot{v} = u_1 - v \end{cases}$$

Ce modèle pourra être utilisé pour une simulation, mais pas pour l'obtention du régulateur.

3.1.1. Commande proportionnelle en cap et vitesse

Nous allons maintenant proposer une commande simple pour ce système en utilisant l'intuition que l'on peut avoir sur ce système. Posons $\tilde{\theta} = \theta_d - \theta$ où θ_d est le cap désiré et $\tilde{v} = v_d - v$ où v_d est la vitesse désirée.

– Pour la commande en vitesse, on prend :

$$u_1 = a_1 \tanh \tilde{v}$$

où a_1 est une constante qui représente l'accélération maximale (en valeur absolue) que le moteur est capable de délivrer. La tangente hyperbolique \tanh (voir figure 3.1) est utilisée comme fonction de saturation. On rappelle que :

$$\tanh x = \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad [3.1]$$

– Pour la commande en cap, on prend :

$$u_2 = a_2 \cdot \text{sawtooth}(\tilde{\theta})$$

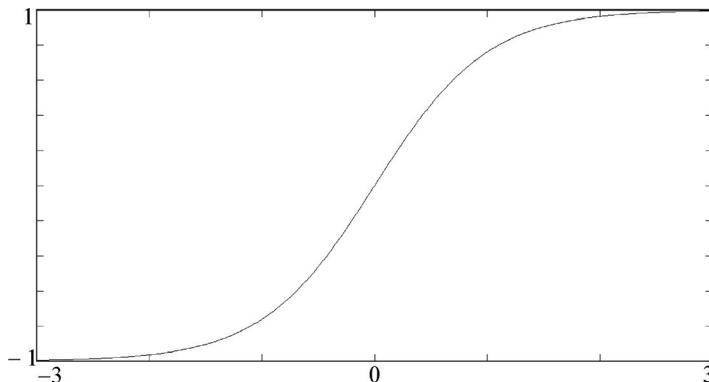


Figure 3.1. Fonction tangente hyperbolique utilisée comme fonction de saturation

Dans cette dernière formule, *sawtooth* est la fonction dite à *dents de scie* (*sawtooth function*). Elle est définie par :

$$\text{sawtooth}(\tilde{\theta}) = 2\text{atan}\left(\tan\frac{\tilde{\theta}}{2}\right) = \text{mod}(\tilde{\theta} + \pi, 2\pi) - \pi \quad [3.2]$$

Notons que pour des raisons numériques, il est préférable d'utiliser l'expression impliquant la fonction *modulo* (*mod* sous MATLAB). Comme illustré par la figure 3.2, la fonction correspond à une erreur de cap. L'intérêt de prendre une erreur $\tilde{\theta}$ filtrée par la fonction *sawtooth* est d'éviter le problème du modulo $2k\pi$: on souhaite qu'une erreur de $2k\pi$ soit considérée comme nulle.

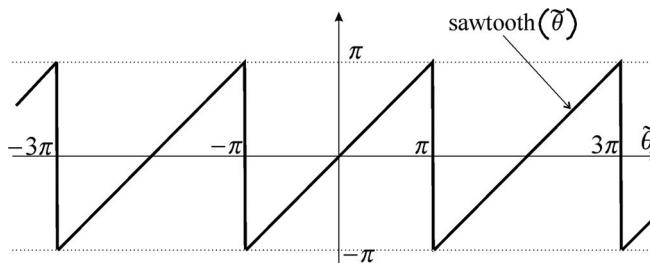


Figure 3.2. Fonction dents de scie utilisée pour éviter les sauts dans la régulation en cap

On peut résumer cette commande par :

$$\begin{pmatrix} u_1 \\ u_2 \end{pmatrix} = \begin{pmatrix} a_1 \cdot \tanh(v_d - v) \\ a_2 \cdot \text{sawtooth}(\theta_d - \theta) \end{pmatrix}$$

On effectuera ainsi une régulation en cap. Cette commande sans modèle, qui fonctionne très bien en pratique, n'a pas eu besoin d'utiliser les équations d'état du robot. Elle repose sur la compréhension que l'on a de la dynamique du système et copie notre manière de téléopérer un robot de type char. Elle possède deux paramètres a_1 et a_2 qui sont faciles à régler (a_1 représente la puissance de la propulsion et a_2 la nervosité dans la direction). Enfin, cette commande est facile à implémenter et à déboguer.

3.1.2. Commande proportionnelle et dérivée en cap

Pour de nombreux robots, une commande proportionnelle engendre des oscillations et il peut s'avérer nécessaire de rajouter un terme d'amortissement, ou de dérivée. C'est le cas pour des robots sous-marins d'exploration (de type ROV, *Remotely Operated Vehicule*) censés se stabiliser au-dessus de la zone d'intérêt. Les robots sous-marins de type torpille n'ont pas ce problème d'oscillation du fait des gouvernes qui stabilisent le cap en mouvement. Si le cap désiré est constant, une telle commande proportionnelle et dérivée est donnée par :

$$u_2 = a_2 \cdot \text{sawtooth}(\theta_d - \theta) + b_2 \dot{\theta}$$

La quantité θ est peut être obtenue par une boussole, par exemple. Quant à $\dot{\theta}$, elle est généralement obtenue par un gyromètre. Pour les robots bas coût, on a pas toujours des gyromètres à notre disposition et on doit essayer d'estimer $\dot{\theta}$ à partir des mesures de θ . Or, une boussole peut sauter de 2π et ceci pour de faibles variations de cap. Ceci est le cas si, par exemple, une boussole renvoie un angle dans l'intervalle $[-\pi, \pi]$ et que le cap varie autour de $(2k + 1)\pi$. Dans ce cas, une estimation de $\dot{\theta}$ est doit obtenue et cette estimation doit être insensible à ces sauts. Notons :

$$\mathbf{R}_t = \begin{pmatrix} \cos \theta(t) & -\sin \theta(t) \\ \sin \theta(t) & \cos \theta(t) \end{pmatrix}$$

la matrice de rotation correspondant au cap $\theta(t)$ du robot (notons que cette matrice est insensible aux sauts de $2k\pi$). Remarquons que :

$$\mathbf{R}_t^T \dot{\mathbf{R}}_t = \begin{pmatrix} 0 & -\dot{\theta}(t) \\ \dot{\theta}(t) & 0 \end{pmatrix}$$

Cette relation peut se voir comme une version bidimensionnelle de la relation [1.1] page 15, mais peut aussi être obtenue directement en utilisant l'expression de \mathbf{R}_t . Ainsi, une intégration par Euler de la matrice de rotation :

$$\mathbf{R}_{t+dt} = \mathbf{R}_t + dt \dot{\mathbf{R}}_t$$

se traduit par :

$$\mathbf{R}_{t+dt} = \mathbf{R}_t + dt \cdot \mathbf{R}_t \begin{pmatrix} 0 & -\dot{\theta}(t) \\ \dot{\theta}(t) & 0 \end{pmatrix} = \mathbf{R}_t \left(\mathbf{I} + dt \begin{pmatrix} 0 & -\dot{\theta}(t) \\ \dot{\theta}(t) & 0 \end{pmatrix} \right)$$

Donc :

$$\begin{aligned}
 dt \begin{pmatrix} 0 & -\dot{\theta}(t) \\ \dot{\theta}(t) & 0 \end{pmatrix} &= \mathbf{R}_t^T \mathbf{R}_{t+dt} - \mathbf{I} \\
 &= \begin{pmatrix} \cos(\theta(t+dt) - \theta(t)) & -\sin(\theta(t+dt) - \theta(t)) \\ \sin(\theta(t+dt) - \theta(t)) & \cos(\theta(t+dt) - \theta(t)) \end{pmatrix} \\
 &- \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}
 \end{aligned}$$

Prenons, dans cette équation matricielle, l'équation scalaire correspondant à la deuxième ligne et première colonne. Nous obtenons :

$$\dot{\theta}(t) = \frac{\sin(\theta(t+dt) - \theta(t))}{dt}$$

La commande proportionnelle et dérivée en cap pourra donc s'écrire sous la forme :

$$u_2(t) = a_2 \cdot \text{sawtooth}(\theta_d - \theta(t)) + b_2 \frac{\sin(\theta(t) - \theta(t-dt))}{dt}$$

qui sera insensible au sauts de 2π .

3.2. Skate car

On considère le véhicule patineur [JAU 10] représenté sur la figure 3.3.

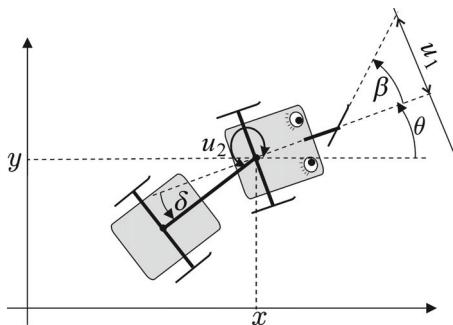


Figure 3.3. Robot patineur qui se déplace à la manière d'un serpent

Ce véhicule, que nous appellerons *skate car* est purement imaginaire. Il est conçu pour se déplacer sur un lac gelé et repose sur cinq patins à glace. Ce système a deux entrées : la tangente u_1 de l'angle β du patin avant (on a choisi la tangente pour entrée afin d'éviter les singularités) et u_2 le couple exercé au niveau de l'articulation

entre les deux traîneaux et qui correspond à l'angle δ . La propulsion provient donc uniquement du couple u_2 et rappelle le mode de propulsion du serpent ou de l'anguille [BOY 06]. Toute commande sur u_1 n'apporte donc aucune énergie au système, mais participe indirectement à la propulsion en générant l'ondulation. Dans ce paragraphe, nous allons proposer un modèle sous forme d'état pour la simulation du système. En ce qui concerne la loi de commande, les méthodes générales existantes ne permettent pas de traiter ce genre de système et il est nécessaire de prendre en compte la physique du problème. Nous allons donc proposer une loi de commande mimétique permettant d'obtenir un régulateur efficace.

3.2.1. Modèle

Cherchons ici à obtenir des équations d'état capables de représenter la dynamique du système afin de pouvoir simuler notre système. Les variables d'état choisies seront $\mathbf{x} = (x, y, \theta, v, \delta)$, où x, y, θ correspondent à la pose du chariot avant, v représente la vitesse du milieu de l'essieu du traîneau avant et δ est l'angle entre les deux chariots. La vitesse angulaire du traîneau avant est donnée par :

$$\dot{\theta} = \frac{v_1 \sin \beta}{L_1} \quad [3.3]$$

où v_1 est la vitesse du patin avant et L_1 est la distance entre le patin avant et le milieu de l'essieu du traîneau avant. Or :

$$v = v_1 \cos \beta$$

et donc :

$$\dot{\theta} = \frac{v \tan \beta}{L_1} = \frac{v u_1}{L_1} \quad [3.4]$$

Vu du traîneau arrière, tout se passe comme si au milieu de l'essieu du traîneau avant, il y avait un patin virtuel se déplaçant avec ce dernier. Ainsi, en reprenant la formule [3.3], la vitesse angulaire du traîneau arrière est :

$$\dot{\theta} + \dot{\delta} = -\frac{v \sin \delta}{L_2}$$

où L_2 est la distance entre les milieux des essieux. Et donc :

$$\dot{\delta} = -\frac{v \sin \delta}{L_2} - \dot{\theta} \stackrel{[3.4]}{=} -\frac{v \sin \delta}{L_2} - \frac{v u_1}{L_1} \quad [3.5]$$

D'après le théorème de l'énergie cinétique, la dérivée temporelle de l'énergie cinétique est égale à la somme des puissances apportées au système, c'est-à-dire :

$$\frac{d}{dt} \left(\frac{1}{2} m v^2 \right) = \underbrace{u_2 \cdot \dot{\delta}}_{\text{puissance motrice}} - \underbrace{(\alpha v) \cdot v}_{\text{puissance dissipée}} \quad [3.6]$$

où α est le coefficient de frottement visqueux. Dans un souci de simplification, nous venons ici de supposer la force de frottement est αv , ce qui revient à supposer que seul le traîneau avant freine. Nous avons donc :

$$mv\dot{v} \stackrel{[3.6]}{=} u_2 \cdot \dot{\delta} - \alpha v^2 \stackrel{[3.5]}{=} u_2 \cdot \left(-\frac{v \sin \delta}{L_2} - \frac{vu_1}{L_1} \right) - \alpha v^2$$

ou encore :

$$m\dot{v} = u_2 \cdot \left(-\frac{\sin \delta}{L_2} - \frac{u_1}{L_1} \right) - \alpha v \quad [3.7]$$

Le système peut se décrire par les équations d'état suivantes :

$$\begin{cases} \dot{x} = v \cos \theta \\ \dot{y} = v \sin \theta \\ \dot{\theta} \stackrel{[3.4]}{=} vu_1 \\ \dot{v} \stackrel{[3.7]}{=} -(u_1 + \sin \delta) u_2 - v \\ \dot{\delta} \stackrel{[3.5]}{=} -v(u_1 + \sin \delta) \end{cases} \quad [3.8]$$

où, pour simplifier, les coefficients (masse m , coefficient de frottement visqueux α , distances inter-essieux L_1, L_2 , etc.) ont été choisis unitaires. Ce système pourrait être rendu affine en la commande (voir équation [2.6]) en rajoutant un intégrateur devant u_1 , mais l'approche par bouclage linéarisant ne s'applique pas du fait de nombreuses singularités. En effet, on peut aisément montrer, que lorsque la vitesse v est nulle (facile à éviter) ou lorsque $\dot{\delta} = 0$ (ce qui arrive forcément régulièrement), on a une singularité. Une commande utilisant le *biomimétisme*, copiant la propulsion du serpent ou de l'anguille, peut être envisagée.

3.2.2. Commande sinusoïdale

En cherchant à copier la stratégie de commande d'un serpent ondulant pour se déplacer, on choisit u_1 de la forme :

$$u_1 = p_1 \cos(p_2 t) + p_3$$

où p_1 est l'amplitude, p_2 la pulsation et p_3 le biais. On choisit u_2 pour que le couple propulseur soit moteur, c'est-à-dire $\dot{\delta}u_2 \geq 0$. En effet $\dot{\delta}u_2$ correspond à la puissance apportée au robot qui se transforme en énergie cinétique. Si u_2 est bornée par l'intervalle $[-p_4, p_4]$, on choisit donc une commande de type bang-bang pour u_2 de la forme :

$$u_2 = p_4 \cdot \text{sign}(\dot{\delta})$$

ce qui revient à exercer une propulsion maximale. La commande par retour d'état choisie est donc :

$$\mathbf{u} = \begin{pmatrix} p_1 \cos(p_2 t) + p_3 \\ p_4 \operatorname{sign}(-v(u_1 + \sin \delta)) \end{pmatrix}$$

Les paramètres de la commande restent à déterminer. Le paramètre de biais p_3 permet de se diriger. La puissance du moteur couple nous donne p_4 . Le paramètre p_1 est directement lié à l'amplitude de l'oscillation engendrée lors du déplacement. Enfin, le paramètre p_2 donne la fréquence des oscillations. Les simulations peuvent nous aider fixer correctement les paramètres p_1 et p_2 . La figure 3.4 illustre deux simulations où le robot part avec une vitesse quasi nulle. Dans la simulation du haut, le biais p_3 est nul. Dans celle du bas, $p_3 > 0$.

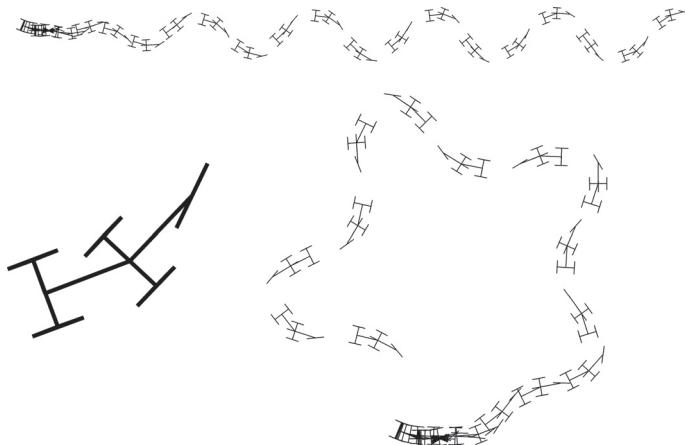


Figure 3.4. Différentes simulations pour illustrer la loi de commande pour le robot patineur

La figure 3.5 représente la poussée en fonction du temps. Il est clair que la puissance fournie par le moteur est très forte au démarrage alors qu'en régime de croisière, le moteur est sous-utilisé. Une telle commande oblige à surdimensionner notre moteur. Nous aurions intérêt à avoir une poussée aussi constante que possible. Le script `snake.m` propose une implémentation de cette loi de commande.

3.2.3. Commande à poussée maximale

La propulsion du robot se fait par la poussée $u_2 \cdot \dot{\delta} = -v(u_1 + \sin \delta) \cdot u_2$ et donc par le moteur qui génère le couple u_2 . Afin d'aller le plus vite possible, pour un

moteur donné, il convient de demander au moteur de délivrer une puissance maximale dénotée \bar{p} qui sera transformée en énergie cinétique. Ainsi :

$$\underbrace{-v(u_1 + \sin \delta) \cdot u_2}_{\dot{\delta}} = \bar{p}$$

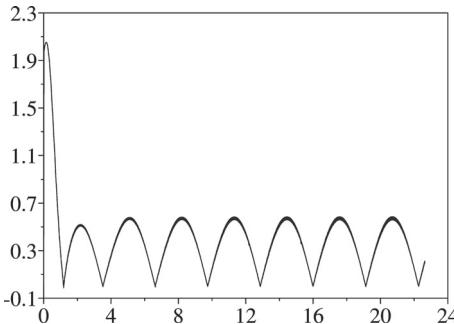


Figure 3.5. Poussée fournie par le moteur u_2

Il existe donc plusieurs couples (u_1, u_2) capables de donner la puissance désirée \bar{p} . Nous allons donc choisir pour u_2 la forme :

$$u_2 = \varepsilon \cdot \bar{u}_2 \text{ avec } \varepsilon = \pm 1$$

où $\varepsilon(t)$ est un signal créneau et \bar{u}_2 une constante. Ce choix pour u_2 peut être de borner le couple moteur et ainsi limiter les sollicitations mécaniques. Si nous choisissons la fréquence de ε trop faible, la puissance délivrée sera respectée, mais le chariot avant rentrera en collision avec le chariot arrière. Dans le cas limite où ε est constant, on pourra observer en simulation, le premier chariot s'enrouler au deuxième (ce qui signifie que δ augmente jusqu'à l'infini). Nous obtenons en isolant l'orientation u_1 du patin avant :

$$u_1 = - \left(\frac{\bar{p}}{v\varepsilon\bar{u}_2} + \sin \delta \right)$$

La commande à poussée maximale est donc donnée par :

$$\mathbf{u} = \begin{pmatrix} - \left(\frac{\bar{p}}{v\varepsilon\bar{u}_2} + \sin \delta \right) \\ \varepsilon\bar{u}_2 \end{pmatrix} \quad [3.9]$$

Ainsi, avec cette commande, non seulement on pousse toujours dans la bonne direction à travers u_2 mais on joue avec la direction u_1 pour que le couple fourni par u_2 se traduise en une poussée maximale \bar{p} . Il ne nous reste plus qu'à agir sur ε

(qui rappelons-le, est un signal créneau qui vaut ± 1) et sur la puissance \bar{p} . Le rapport cyclique du signal $\varepsilon(t)$ nous permettra de nous diriger et sa fréquence nous donnera l'amplitude des oscillations pour la trajectoire du robot. Quant à \bar{u}_2 il nous permet de régler la vitesse moyenne du robot. Sur simulation, cette commande se montre en effet plus efficace que la commande sinusoïdale. La figure 3.6 nous montre l'angle du patin β en fonction du temps, une fois le régime de croisière atteint. Notons que l'angle du patin avant $\beta = \text{atan}(u_1)$ fait apparaître des discontinuités.

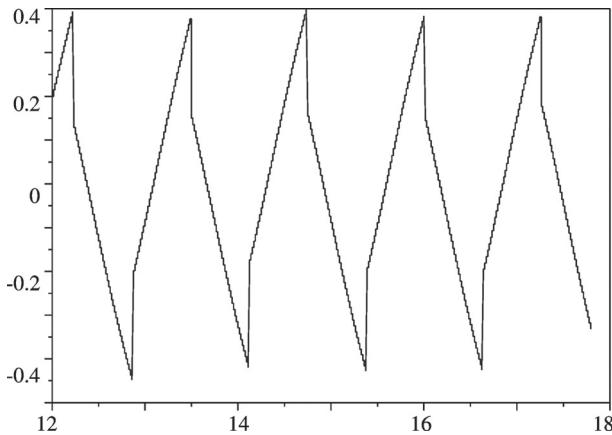


Figure 3.6. Evolution de l'angle du patin avant β en régime de croisière

3.2.4. Simplification de la dynamique rapide

Les équations d'état pour le *snakeboard* possèdent de nombreuses singularités et l'on souhaite ici les simplifier. Or, dans notre système, nous avons deux dynamiques qui interfèrent : une lente (qui représente l'évolution lissée des variables d'état) et une rapide (cadencée par ε) qui crée l'ondulation. L'idée, relativement classique en automatique, est de moyenner ces grandeurs de façon à faire disparaître la dynamique rapide. On retrouve cette idée lorsque l'on commande des moteurs à courant continu par PWM (*Pulse Width Modulation*).

Considérons un signal créneau haute fréquence $\varepsilon(t)$. Sa moyenne temporelle $\bar{\varepsilon}$ est appelée *rapport cyclique*. Ce rapport est amené à varier très lentement dans le temps. L'opérateur *moyenne temporelle* est linéaire (tout comme l'espérance mathématique). Par exemple :

$$\overline{2\varepsilon_1(t) - 3\varepsilon_2(t)} = 2\bar{\varepsilon}_1(t) - 3\bar{\varepsilon}_2(t)$$

En revanche, on ne pourra pas écrire pour une fonction f non linéaire $\overline{f(\varepsilon)} = f(\bar{\varepsilon})$. Par exemple, $\overline{\varepsilon^{-1}} \neq \bar{\varepsilon}^{-1}$. Par contre, nous aurons $\varepsilon^{-1} = \bar{\varepsilon}$ si $\varepsilon(t) \in \{-1, 1\}$.

Cela vient du fait que les signaux ε et ε^{-1} sont égaux dans un tel cas. Si $a(t)$ et $b(t)$ sont des signaux variant lentement dans le temps, nous aurons aussi :

$$\overline{a(t) \varepsilon_1(t) + b(t) \varepsilon_2(t)} = a(t) \bar{\varepsilon}_1(t) + b(t) \bar{\varepsilon}_2(t)$$

Nous allons essayer d'appliquer ces approximations au cas du *snakeboard* pour éliminer les dynamiques rapides. En reprenant les équations d'état [3.8] et en y injectant la loi de commande [3.9], nous obtenons un système bouclé décrit par les équations d'état suivantes :

$$\begin{cases} \dot{x} = v \cos \theta \\ \dot{y} = v \sin \theta \\ \dot{\theta} = -\frac{\bar{p}}{\varepsilon \bar{u}_2} - v \sin \delta \\ \dot{v} = \frac{\bar{p}}{v} - v \\ \dot{\delta} = \frac{\bar{p}}{\varepsilon \bar{u}_2} \end{cases}$$

Rappelons, que nous pouvons agir sur les constantes \bar{p} , \bar{u}_2 et sur le signal créneau $\varepsilon = \pm 1$, que nous allons ici considérer comme de haute fréquence et de rapport cyclique $\bar{\varepsilon}$. Nous pouvons faire l'approximation :

$$\frac{\bar{p}}{\varepsilon \bar{u}_2} = \frac{\varepsilon \bar{p}}{\bar{u}_2} \xrightarrow{\text{(par linéarité)}} \bar{\varepsilon} \frac{\bar{p}}{\bar{u}_2}$$

Ainsi, le système devient :

$$\begin{cases} \dot{x} = v \cos \theta \\ \dot{y} = v \sin \theta \\ \dot{\theta} = -\bar{p}\bar{q} - v \sin \delta \\ \dot{v} = \frac{\bar{p}}{v} - v \\ \dot{\delta} = \bar{p}\bar{q} \end{cases}$$

qui ne possède plus que deux entrées : \bar{p} et $\bar{q} = \frac{\bar{\varepsilon}}{\bar{u}_2}$. Cherchons à commander les entrées θ et v en utilisant une méthode de linéarisation par bouclage. Notons que bien que ce système ne soit pas affine en ses entrées (\bar{p} et \bar{q}), la méthode s'applique car, comme nous allons maintenant le voir, l'inversion nécessaire est ici possible. Pour cela, définissons deux nouvelles entrées v_1, v_2 telles que :

$$\begin{cases} v_1 = -\bar{p}\bar{q} - v \sin \delta \\ v_2 = \frac{\bar{p}}{v} - v \end{cases}$$

En inversant ce système relativement aux entrées, nous obtenons :

$$\begin{cases} \bar{p} = v(v_2 + v) \\ \bar{q} = -\frac{v_1 + v \sin \delta}{v(v_2 + v)} \end{cases}$$

Ainsi, le système linéarisé par bouclage est :

$$\begin{cases} \dot{\theta} = v_1 \\ \dot{v} = v_2 \end{cases}$$

Une commande proportionnelle peut donc convenir. On prendra celle qui pose les pôles en -1 , c'est-à-dire :

$$\begin{cases} v_1 = w_1 - \theta + \dot{w}_1 \\ v_2 = w_2 - v + \dot{w}_2 \end{cases}$$

Récapitulons la loi de commande dans sa globalité. Elle possède pour consigne w_1, w_2 qui correspond au cap et à la vitesse désirés. Elle est donnée par la table qui suit :

$$\bar{p} = v (w_2 + \dot{w}_2)$$

$$\bar{q} = - \frac{w_1 - \theta + \dot{w}_1 + v \sin \delta}{v (w_2 + \dot{w}_2)}$$

$$\bar{\varepsilon} = \bar{q} \cdot \bar{u}_2 \text{ (choisir } \bar{\varepsilon} \in [-1, 1] \text{)}$$

ε : créneau de rapport cyclique $\bar{\varepsilon}$ et de fréquence ∞

$$\mathbf{u} = \begin{pmatrix} - \left(\frac{\bar{p}}{v \varepsilon \bar{u}_2} + \sin \delta \right) \\ \varepsilon \bar{u}_2 \end{pmatrix}$$

Le paramètre de réglage \bar{u}_2 intervenant dans cette commande est assez délicat à régler. Il faut choisir \bar{u}_2 suffisamment petit pour avoir $\bar{\varepsilon} \in [-1, 1]$. Mais pas trop proche de zéro, pour ne pas avoir u_1 trop grand (ce qui engendrerait des mouvements du patin avant trop important). Cette variable \bar{u}_2 influence la répartition entre le couple (à travers u_2) et le mouvement (à travers u_1) nécessaires à la génération de la puissance. Notons enfin que cette dernière loi de commande censée être plus efficace utilise pour sa conception les équations d'état du système et peut donc difficilement être qualifiée de *sans modèle*.

3.3. Voilier

3.3.1. Problème

Reprendons les principes de la commande sans modèle et tentons de l'adapter à une régulation par suivi de ligne pour notre voilier. Nous allons ici considérer un voilier dont seule la longueur d'écoute est variable et non pas directement l'angle de la voile comme c'était le cas jusqu'à présent (voir [2.11] page 68). Ce robot possède deux entrées qui sont l'angle du gouvernail $u_1 = \delta_r$ et l'angle maximal de la voile $u_2 = \delta_s^{\max}$ (de façon équivalente u_2 correspond à la longueur de l'écoute). Nous allons chercher à ce que le robot suive une ligne passant par les points **a** et **b** (voir figure 3.7).

Ce problème est influencé par les stratégies de commande du robot VAIMOS [GOR 11] d'IFREMER, du robot voilier de l'école navale ERWAN ou le robot Optimousse de l'ENSTA-Bretagne (voir figure 3.8).

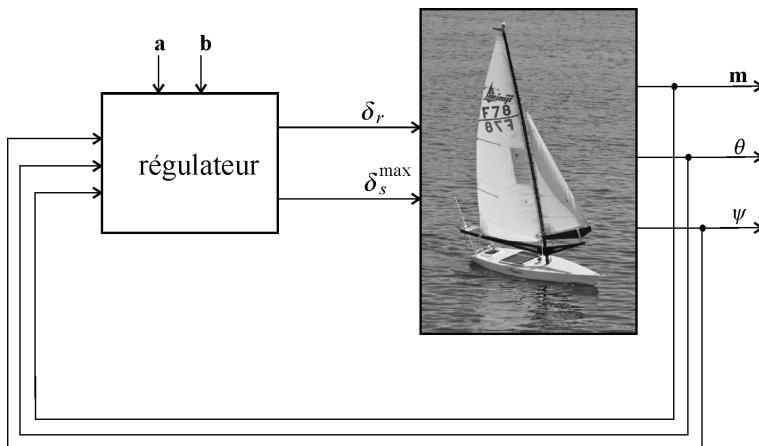


Figure 3.7. Boucle de régulation du robot voilier

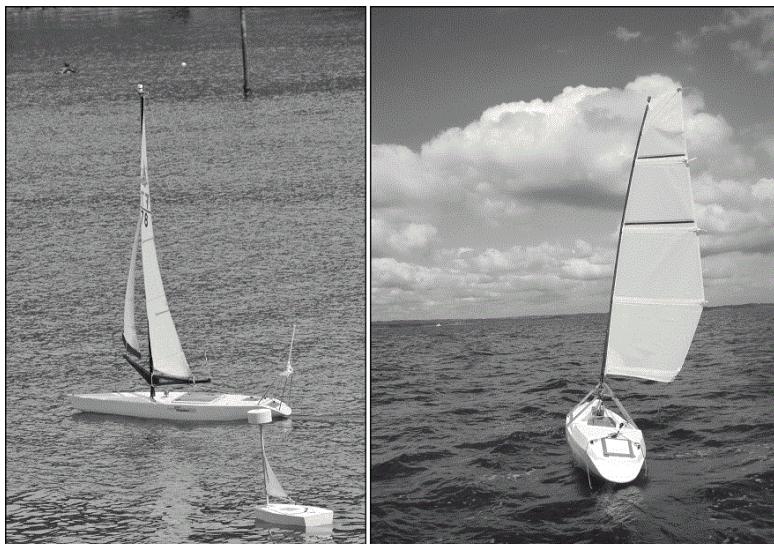


Figure 3.8. A gauche : robot voilier VAIMOS d'IFREMER (arrière-plan) et le robot Optimousse de l'ENSTA Bretagne ; à droite : robot de l'Ecole Navale qui réalise un suivi de ligne

Comme illustré par la figure 3.9, nous noterons par (x, y, θ) la pose du bateau, v sa vitesse d'avance, ω sa vitesse angulaire, f_s la force du vent sur la voile, f_r la force de l'eau sur le gouvernail, δ_s l'angle de la voile et ψ l'angle du vent.

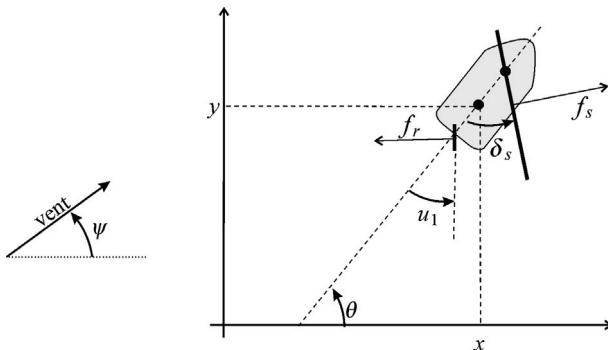


Figure 3.9. Variables utilisées dans les équations d'état du robot

3.3.2. Régulateur

Nous allons ici chercher à faire un régulateur pour faire un suivi de ligne. Le robot sera équipé de trois capteurs : une boussole qui nous donne le cap θ , une girouette qui nous mesure l'angle du vent ψ et un GPS qui nous retourne la position \mathbf{m} du bateau. Le robot sera aussi muni de deux actionneurs : un servo-moteur qui nous règle l'angle du gouvernail δ_r et un moteur pas à pas qui nous règle la longueur de l'écoute et donc l'angle maximal δ_s^{\max} de la voile (c'est-à-dire $|\delta_s| \leq \delta_s^{\max}$). Quant au régulateur il a pour consigne la ligne \mathbf{ab} à suivre et dispose d'une variable binaire $q \in \{-1, 1\}$ dite d'*hystérésis* qui sera utilisée lors de la remontée au près. Ce régulateur aura peu de paramètres qui seront de surcroît faciles à régler. Parmi ces paramètres, on trouve l'angle maximal du gouvernail δ_r^{\max} (typiquement $\delta_r^{\max} = \frac{\pi}{4}$), la distance de coupure r (c'est-à-dire que nous voulons que la distance à la ligne soit toujours inférieure à r), l'angle de près ζ (typiquement $\zeta = \frac{\pi}{4}$), et l'angle de voile en vent de travers β (typiquement $\beta = 0.3$ rad). Nous proposons le régulateur suivant, tiré de l'article [JAU 12], et que nous allons ensuite expliquer :

Régulateur in : $\mathbf{m}, \theta, \psi, \mathbf{a}, \mathbf{b}$; out : $\delta_r, \delta_s^{\max}$; inout : q
1 $e = \det \left(\frac{\mathbf{b} - \mathbf{a}}{\ \mathbf{b} - \mathbf{a}\ }, \mathbf{m} - \mathbf{a} \right)$
2 if $ e > r$ then $q = \text{sign}(e)$
3 $\varphi = \text{angle}(\mathbf{b} - \mathbf{a})$
4 $\bar{\theta} = \varphi - \text{atan} \left(\frac{e}{r} \right)$
5 if $\cos(\psi - \bar{\theta}) + \cos \zeta < 0$
6 or $(e < r \text{ and } (\cos(\psi - \varphi) + \cos \zeta < 0))$
7 then $\bar{\theta} = \pi + \psi - q\zeta$.
8 $\delta_r = \frac{\delta_r^{\max}}{\pi} \text{sawtooth}(\theta - \bar{\theta})$
9 $\delta_s^{\max} = \frac{\pi}{2} \left(\frac{\cos(\psi - \bar{\theta}) + 1}{2} \right)^{\frac{\log \left(\frac{\pi}{2\beta} \right)}{\log(2)}}$

Le régulateur dispose pour seule variable d'état, la variable binaire $q \in \{-1, 1\}$. C'est pour cela qu'elle apparaît à la fois comme entrée et sortie de l'algorithme. Commentons cet algorithme.

Ligne 1 (calcul de la distance algébrique) : nous calculons la distance algébrique entre le robot et sa ligne. Si $e > 0$ le robot est à gauche de sa ligne et si $e < 0$, il est à droite. Dans la formule, le déterminant est à comprendre dans le sens suivant :

$$\det(\mathbf{u}, \mathbf{v}) = u_1v_2 - v_1u_2$$

Ligne 2 (mise à jour de la variable d'hystérésis) : si $|e| > r$, c'est que le robot est loin de sa ligne et la variable d'hystérésis q (qui mémorise le bord privilégié) est autorisée à changer de valeur. Si par exemple $e > r$, alors q prendra la valeur 1 et elle gardera cette valeur jusqu'à ce que $e < -r$.

Ligne 3 (calcul de l'angle de la ligne) : nous calculons l'angle φ de la ligne à suivre (voir figure 3.10). Dans l'instruction, `angle(u)` représente l'angle que fait le vecteur $\mathbf{u} \in \mathbb{R}^2$ relativement à l'axe *Ox* (direction Est). Elle correspond à la fonction rangée dans `angle.m`.

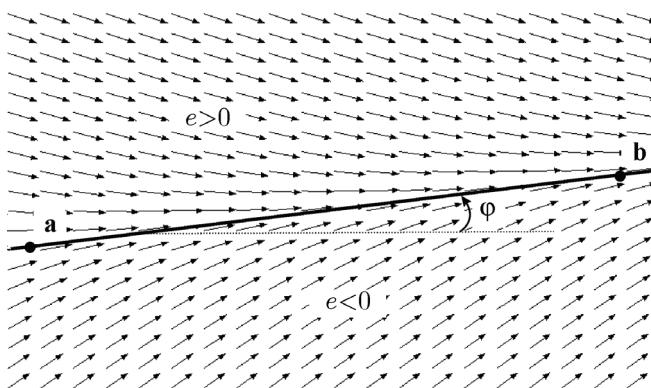


Figure 3.10. Champ de vecteur nominal que le robot cherche à suivre, lorsque cela est possible

Ligne 4 (calcul du cap nominal) : nous calculons l'angle nominal $\bar{\theta}$ (voir figure 3.10), c'est-à-dire celui que l'on souhaite avoir sans se préoccuper du vent. Nous prenons :

$$\bar{\theta} = \varphi - \text{atan}\left(\frac{e}{r}\right)$$

Cette expression pour $\bar{\theta}$ se traduit par une ligne attractive. Lorsque $e = \pm\infty$, nous avons $\bar{\theta} = \varphi \pm \frac{\pi}{2}$, ce qui signifie que le robot a un cap qui forme un angle à la ligne de

$\frac{\pi}{2}$. Pour une distance e correspondant à la distance de coupure r , c'est-à-dire $e = \pm r$, nous avons $\bar{\theta} = \varphi \pm \frac{\pi}{4}$. Enfin sur la ligne $e = 0$ et donc $\bar{\theta} = \varphi$, ce qui correspond à un cap dont la direction est celle de la ligne. Comme illustré sur la figure 3.11a, certaines directions $\bar{\theta}$ peuvent être incompatibles avec celle du vent.

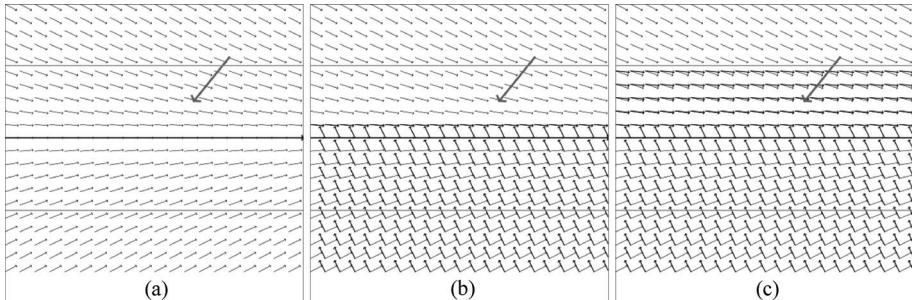


Figure 3.11. (a) Le champ de vecteurs nominal peut être incompatible avec le vent (ici représenté par la grosse flèche). (b) Champ de vecteurs généré par le régulateur si l'on enlève la ligne 6. Les flèches fines correspondent aux routes nominales et les flèches épaisses aux routes corrigées. (c) Champ de vecteurs généré par le régulateur avec la ligne 6.

Ligne 5 : lorsque $\cos(\psi - \bar{\theta}) + \cos \zeta < 0$, la route $\bar{\theta}$ correspond à une direction trop proche du vent que le robot n'est pas capable de suivre (voir figure 3.12).

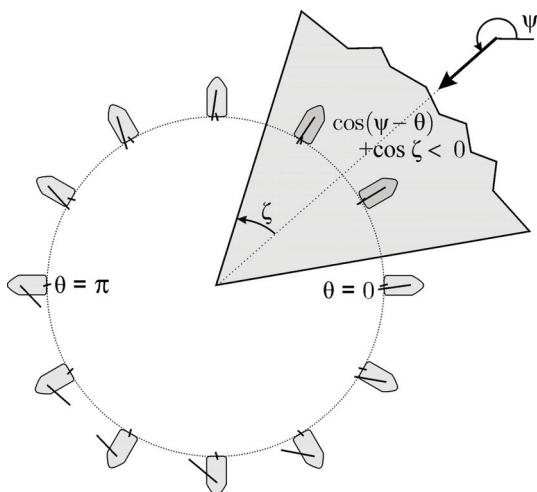


Figure 3.12. Certaines directions ne sont pas possibles pour le voilier. Ces directions infaisables forment la no-go zone, représentée en gris

Le cap $\bar{\theta}$ est alors impossible à tenir. Dans ce cas, il faut passer en mode *près*, ce qui signifie que le robot se met le plus qu'il le peut contre le vent, ou plus formellement, la nouvelle direction devient $\bar{\theta} = \pi + \psi \pm \zeta$ (voir ligne 7). La figure 3.11b représente le champ de vecteurs correspondant. Les flèches fines correspondent au champ nominal et les flèches en gras représentent de champ corrigé lorsque nécessaire. Dans cette représentation, nous avons supprimé l'effet d'hystérésis induit par la variable q (ce qui revient à dire que l'on a toujours $q = \text{sign}(e)$).

Ligne 6 (stratégie de garde le près) : cette instruction implémente ce que l'on appelle la stratégie *garde le près* (en anglais *keep close hauled strategy*). Si $|e| < r$ ou si $\cos(\psi - \varphi) + \cos \zeta < 0$, alors on force le bateau à se mettre au près, même si le cap $\bar{\theta}$ est admissible et ceci pour des raisons d'efficacité. Cette stratégie est illustrée sur la figure 3.11c. Sur cette figure nous avons choisi un angle de près $\zeta = \frac{\pi}{3}$ (ce qui correspond à un bateau qui a du mal à remonter au vent) et de ce fait, la ligne est considérée comme contre le vent.

Ligne 7 (cap au près) : le bateau est en mode près et on choisit $\bar{\theta} = \pi + \psi - q\zeta$ (la direction du vent plus ou moins l'angle de près ζ). La variable d'hystérésis q oblige à maintenir le même bord tant que la distance à la ligne de r n'a pas été atteinte. Une illustration du comportement qui en résulte est représentée sur la figure 3.13. Si le cap nominal est tenable, alors on le suit.

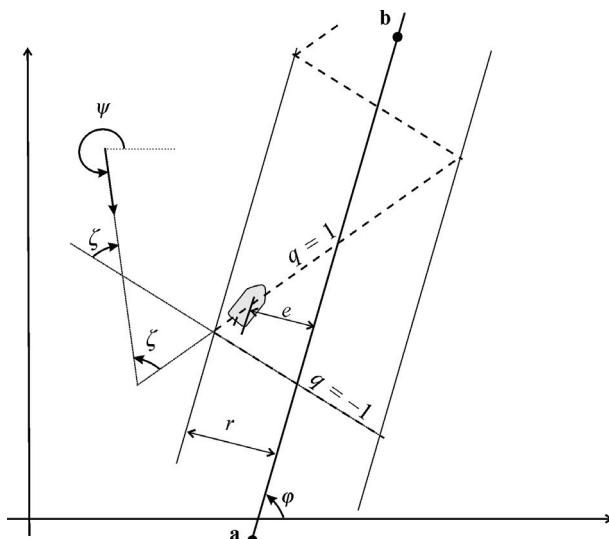


Figure 3.13. Stratégie de remonté au vent en restant dans la bande centrée sur la ligne ab et de diamètre r

Ligne 8 (commande du gouvernail) : à ce niveau, le cap à tenir $\bar{\theta}$ a été sélectionné et on cherche à le suivre par une action sur le gouvernail. On effectue une commande proportionnelle relativement à l'erreur $\theta - \bar{\theta}$. Afin de filtrer le problème du modulo 2π , on utilise la fonction en dent de scie *sawtooth* (voir formule [3.2]). On obtient ainsi :

$$\delta_r = \frac{\delta_r^{\max}}{\pi} \cdot \text{sawtooth}(\theta - \bar{\theta})$$

où δ_r^{\max} est l'angle maximal du gouvernail (par exemple $\delta_r^{\max} = 0.5$ rad). La commande qui en résulte est illustrée par la figure 3.14.

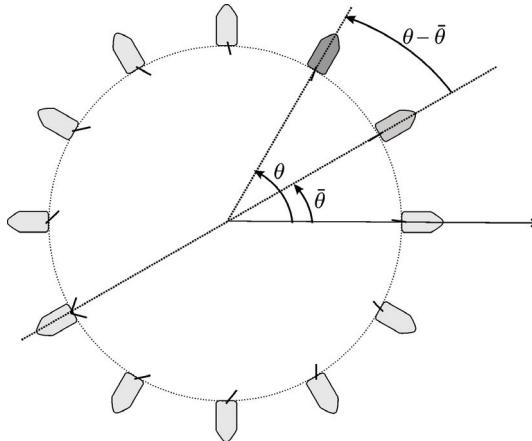


Figure 3.14. Réglage du gouvernail pour le robot voilier

Ligne 9 (réglage de la voile) : on choisit un angle de voile β (voile à moitié ouverte) que devra posséder la voile en vent de travers. Ce paramètre est déterminé de façon empirique suivant le voilier et le type de conduite que l'on souhaite adopter. L'angle maximal de la voile δ_s^{\max} est une fonction de $\psi - \theta$ qui est périodique de période 2π . Un modèle possible [JAU 12] est celui de la cardioïde :

$$\delta_s^{\max} = \frac{\pi}{2} \cdot \left(\frac{\cos(\psi - \theta) + 1}{2} \right)^\eta$$

où le paramètre η est positif. Lorsque $\psi = \theta + \pi$, le bateau est face au vent et le modèle nous donne $\delta_s^{\max} = 0$. Lorsque $\psi = \theta$, on a $\delta_s^{\max} = \frac{\pi}{2}$, ce qui signifie que la voile est grand ouverte lorsque le robot est vent arrière. Pour le choix du paramètre η , on se basera sur l'angle de voile en vent de travers, c'est-à-dire pour $\psi = \theta \pm \frac{\pi}{2}$. L'équation $\delta_s^{\max} = \beta$ pour $\psi = \theta \pm \frac{\pi}{2}$ se traduit par :

$$\frac{\pi}{2} \cdot \left(\frac{1}{2} \right)^\eta = \beta$$

c'est-à-dire :

$$\eta = \frac{\log\left(\frac{\pi}{2\beta}\right)}{\log(2)}$$

La fonction δ_s^{\max} est représentée sur la figure 3.15.

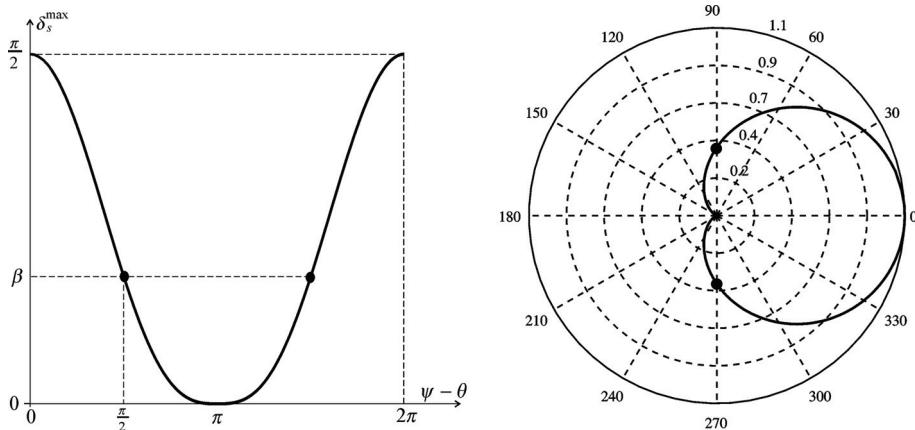


Figure 3.15. Réglage de l'angle maximal de voile (ou également de la longueur de l'écoute) ; à gauche : représentation cartésienne ; à droite : représentation polaire

Sur les tests effectués, ce réglage de voile s'est montré efficace et facile à régler, du fait de l'existence de peu de paramètres.

3.3.3. Navigation

Une fois que le suivi de ligne a été correctement implémenté et validé, il convient d'enchaîner les lignes dans le but de pouvoir effectuer des missions complexes (comme par exemple rejoindre deux points du globe). Dans un tel contexte, une stratégie par réseau de Pétri est bien adaptée pour représenter les changements d'état discrets [GUI 11]. La figure 3.16 illustre un réseau de Pétri permettant la gestion de la mission. Avant que le robot ne soit lancé, il est à l'état initial représenté par la place p_0 . La transition t_1 est franchie au lancement de la mission. Si tout se passe bien, le robot est dans l'état p_1 et suit sa première ligne a_1b_1 . La ligne a_jb_j sera validée dès que le point b_j est dépassé, c'est-à-dire si $\langle b_j - a_j, m - b_j \rangle > 0$. Ce critère d'arrêt couplé à la route peut être interprété comme une porte de validation. Une fois cette porte validée, on passe à la ligne suivante. Lorsque la liste des lignes à suivre est épuisée, la mission s'arrête (place p_3).

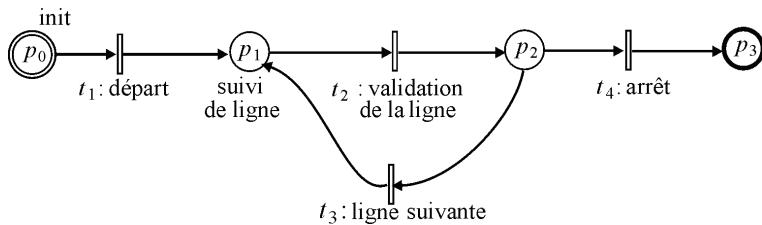


Figure 3.16. Réseau de Pétri qui supervise la navigation du robot

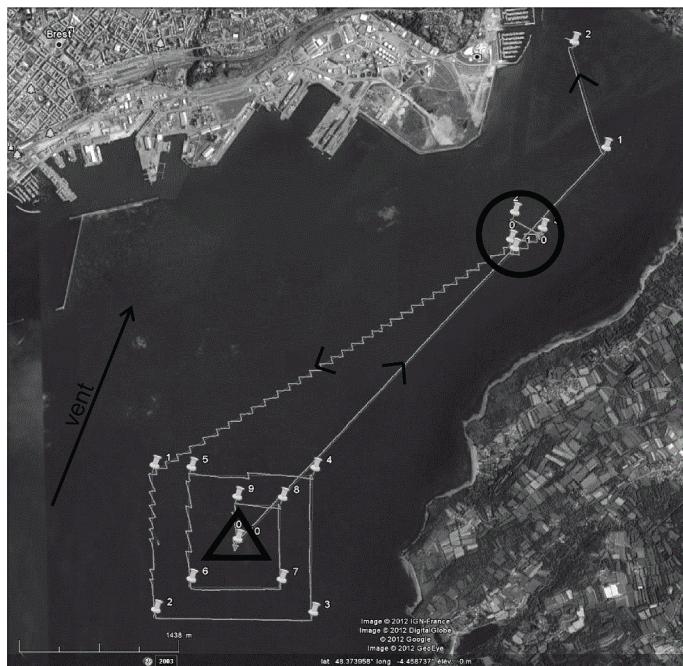


Figure 3.17. Expérience de la spirale composée de cinq étapes : (a) le robot commence par un triangle (dans le cercle) ; (b) il remonte au vent en suivant une ligne ; (c) décrit une spirale ; (d) s'ancre virtuellement au centre de la spirale pour quelques minutes ; (e) rentre au port

3.3.4. Expérience

A partir de septembre 2011, nous avons fait une série d'expériences avec le voilier VAIMOS en mode autonome. Nous allons en décrire une à la fois simple et représentative qui s'est tenue le jeudi 28 juin 2012 dans le rade de Brest, près du port du moulin blanc. La trace du trajet effectué par le robot se trouve représentée sur la figure 3.17. Le vent provient du sud-sud-ouest, comme indiqué par la flèche qui représente une moyenne déduite du capteur de vent du robot sur toute la mission. Dans

cette zone à forte circulation maritime, les interruptions de mission sont à prévoir et un lien wifi permanent entre le robot et le bateau de suivi est nécessaire afin de pouvoir arrêter à tout moment de la mission et éviter les collisions avec les autres bateaux. Mis à part ces interruptions de sécurité (qui d'ailleurs n'ont pas été nécessaires dans cette mission), le robot est entièrement autonome. La mission demandée se décompose en cinq sous-missions. Tout d'abord, le robot commence par un triangle (dans le cercle) afin de vérifier que tout fonctionne parfaitement. Ensuite, il se dirige sud-ouest en remontant le vent et en suivant la ligne demandée. Il décrit une ensuite spirale. Une fois rendu au centre de la spirale, le robot s'ancre virtuellement, c'est-à-dire qu'il manoeuvre pour rester autour de son point d'accroche. Enfin, le robot rentre au port en vent arrière.

D'autres expériences de plus grande ampleur ont aussi effectuées, comme celle de Brest à Douarnenez (voir figure 3.18) effectuée le 17-18 janvier 2012, faisant ainsi un trajet de plus de 100 km. De très haut (comme sur la figure), les lignes semblent être suivies parfaitement. De plus près, les choses ne sont pas aussi idéales : le voilier louvoie pour remonter le vent, se recalibre, ou subit l'effet de fortes vagues. Cependant, dans les deux expériences décrites précédemment, le robot n'est jamais éloigné de sa ligne de plus de 50 mètres (sauf bien sûr, dans les situations d'évitement imposées où ce dernier se trouve remorqué).



Figure 3.18. *Trajet de Brest à Douarnenez fait par VAIMOS :*
 (a) *le robot part du port du moulin blanc (dans le cercle) ;*
 (b) *il évite un sous-marin (dans le carré) ; (c) il évite un cargo (triangle)*

REMARQUE 3.1.– Même si le robot ne dépasse jamais sa ligne de plus de 50 m, on pourrait faire mieux et améliorer cette précision lorsque le robot suit le cap nominal (ce qui signifie que l'angle φ de la ligne correspond à un cap tenable). En effet, sur nos expériences, un biais de 10 m peut être observé en mode nominal, ce qui signifie que la distance à la ligne ne converge pas vers zéro (à la précision GPS). C'est le rôle d'un intégrateur que de supprimer un tel biais. Pour implémenter un tel intégrateur, il suffit de remplacer la ligne 4 du régulateur par les deux instructions suivantes :

$$\begin{cases} z = z + \alpha dt e \\ \theta = \varphi - \text{atan} \left(\frac{e+z}{r} \right) \end{cases}$$

où dt est la période d'échantillonnage. La variable z correspond à la valeur de l'intégrateur et converge naturellement vers le biais constant que l'on avait sans intégrateur et que l'on cherche à supprimer. Le coefficient α doit être suffisamment petit pour éviter un changement de comportement de notre robot (qui pourrait apparaître en régime transitoire). Par exemple, si $e = 10$ m pendant 100 secondes, nous pourrions vouloir une correction de 1 m du biais. Pour cela, il faudrait prendre $\alpha = 0.001$. Notons que dès que la distance à la ligne est supérieure à r (c'est le cas par exemple pendant l'initialisation), ou bien si le robot valide une ligne et passe à la ligne suivante, ou encore si le robot est en mode près, alors l'intégrateur doit être forcé à zéro. En effet, un intégrateur ne doit prendre sa fonction qu'une fois le régime permanent établi.

3.4. Exercices

Exercice 3.1. Robot char sur une ligne

On considère un robot se déplaçant sur un plan et décrit par les équations d'état suivantes :

$$\begin{cases} \dot{x} = \cos \theta \\ \dot{y} = \sin \theta \\ \dot{\theta} = u \end{cases}$$

où θ est le cap du robot et (x, y) les coordonnées de son centre. Il s'agit ici du modèle correspondant à la voiture de Dubins [DUB 57]. Le vecteur d'état est donné par $\mathbf{x} = (x, y, \theta)$.

- 1) Simuler sous MATLAB ce système avec graphisme dans différentes situations.
- 2) Proposer une régulation en cap pour le robot.
- 3) Proposer une régulation suivant une ligne **ab**. Il faudra que cette ligne soit attractive. Arrêter le programme lorsque le point **b** est dépassé, c'est-à-dire lorsque $(\mathbf{b} - \mathbf{a})^T (\mathbf{b} - \mathbf{m}) < 0$.

4) Faire suivre au robot un parcours fermé composé d'une séquence de lignes $\mathbf{a}_j \mathbf{b}_j$, $j \in \{1, \dots, j_{\max}\}$.

5) Mettre plusieurs robots identiques à faire ce même circuit, mais avec des vitesses différentes. Modifier les lois de commande afin d'éviter les collisions.

Exercice 3.2. Voiture de Van der Pol

On considère la voiture représentée sur la figure 3.19.

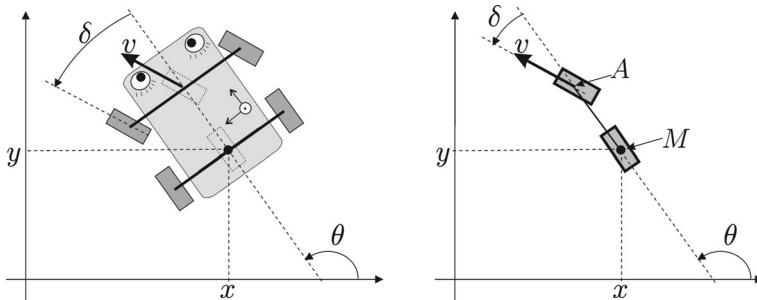


Figure 3.19. Voiture se déplaçant sur un plan (vue de dessus)

Cette voiture possède deux commandes : l'accélération des roues avant et la vitesse de rotation du volant. Les variables d'état de notre système sont constituées par les coordonnées de position (les coordonnées x, y du centre de l'essieu arrière, le cap θ de la voiture, et l'angle δ des roues avant) et la vitesse v du milieu de l'essieu avant. L'équation d'évolution de la voiture est supposée identique à celle d'un tricycle (voir page 64). Elle s'écrit :

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{v} \\ \dot{\delta} \end{pmatrix} = \begin{pmatrix} v \cos \delta \cos \theta \\ v \cos \delta \sin \theta \\ \frac{v \sin \delta}{L} \\ u_1 \\ u_2 \end{pmatrix}$$

1) Simuler sous MATLAB ce système en utilisant la méthode d'Euler.

2) Faire un premier bouclage $\mathbf{u} = \rho(\mathbf{x}, \bar{\mathbf{u}})$ de type proportionnel grand gain qui permette de passer à un modèle char de la forme :

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} \bar{u}_1 \cos \theta \\ \bar{u}_1 \sin \theta \\ \bar{u}_2 \end{pmatrix}$$

Les nouvelles entrées du système bouclé \bar{u}_1 et \bar{u}_2 correspondent respectivement à la vitesse v et à la vitesse angulaire $\dot{\theta}$.

3) Faire un deuxième bouclage $\bar{\mathbf{u}} = \sigma(\mathbf{x}, \mathbf{w})$ qui nous permette de réguler cette voiture en cap et en vitesse. La nouvelle entrée sera $\mathbf{w} = (w_1, w_2)$ où w_1, w_2 correspondent respectivement à la vitesse voulue et au cap voulu.

4) On souhaite que la voiture suive une trajectoire obéissant à l'équation de Van der Pol :

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = - (0.01 x_1^2 - 1) x_2 - x_1 \end{cases}$$

Proposer un troisième régulateur $\mathbf{w} = \tau(\mathbf{x})$ qui permette de réaliser cela. Valider avec une simulation en y superposant le champ de vecteurs à l'aide de l'instruction `quiver`.

Exercice 3.3. Ancreage

On considère le robot décrit par les équations d'état suivantes :

$$\begin{cases} \dot{x} = \cos \theta \\ \dot{y} = \sin \theta \\ \dot{\theta} = u \end{cases}$$

L'objectif de l'*ancreage* pour le robot est de rester dans un voisinage autour de zéro.

1) Le système admet-il un point d'équilibre ?

2) Motivés par le fait que le problème de rester autour de zéro pour le robot admet une symétrie de rotation, nous nous proposons de passer d'une représentation cartésienne (x, y, θ) vers une représentation polaire (α, d, φ) , comme représenté sur la figure 3.20. Donner les équations d'état dans cette représentation polaire.

3) En quoi cette nouvelle représentation est-elle intéressante pour la représentation graphique de la dynamique du système ?

4) Pour résoudre le problème de l'ancreage nous proposons la loi de commande :

$$u = \begin{cases} +1 & \text{si } \cos \varphi \leq \frac{1}{\sqrt{2}} \quad (\text{le robot tourne à gauche}) \\ -\sin \varphi & \text{sinon} \quad (\text{commande proportionnelle}) \end{cases}$$

Une illustration de cette loi de commande est donnée sur la figure 3.21. Expliquer en quoi cette commande résout le problème de l'ancreage. Y a-t-il une configuration qui permette au robot de partir arbitrairement loin de 0 ?

5) Simuler sous MATLAB cette loi de commande avec différentes conditions initiales.

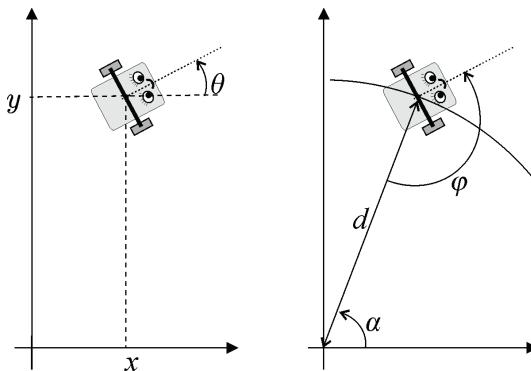


Figure 3.20. Changement de repère permettant de tirer avantage de la symétrie de rotation

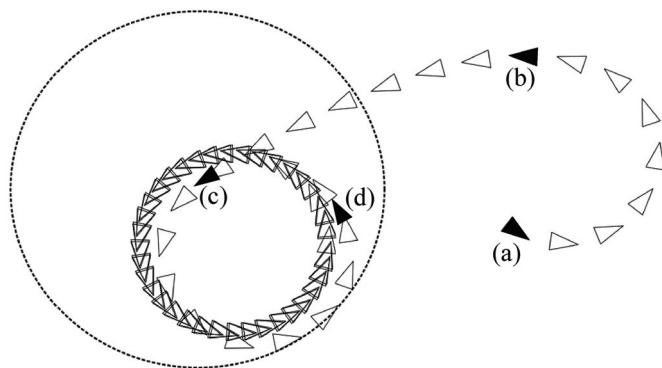


Figure 3.21. Trajectoire du système régulé afin de rester dans le disque

Exercice 3.4. Discréétisation de l'espace d'état

On considère le système suivant (qui découle de l'exercice précédent) :

$$\begin{cases} \text{(i)} \quad \dot{\varphi} = \begin{cases} \frac{\sin \varphi}{d} + 1 & \text{si } \cos \varphi \leq \frac{1}{\sqrt{2}} \\ \left(\frac{1}{d} - 1\right) \sin \varphi & \text{sinon} \end{cases} \\ \text{(ii)} \quad \dot{d} = -\cos \varphi \end{cases}$$

Le champ de vecteurs associé est représenté sur la figure 3.22, où $\varphi \in [\pi, \pi]$ et $d \in [0, 10]$. Une trajectoire $\phi(t, \mathbf{x}_0)$ qui correspond à la simulation visualisée dans l'exercice précédent est aussi dessinée.

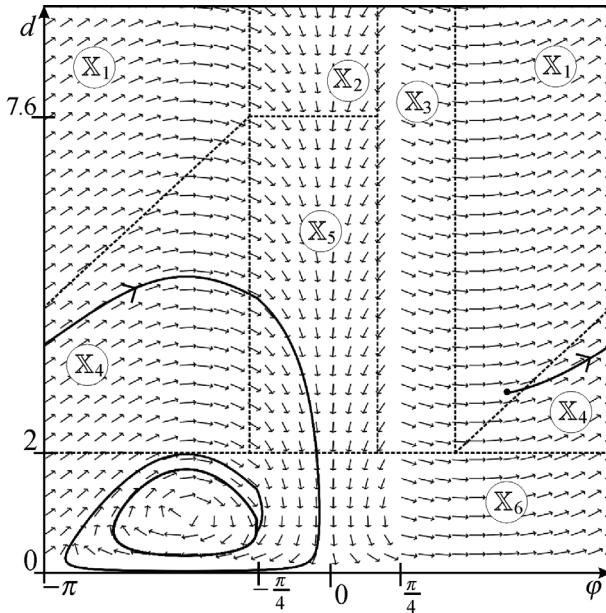


Figure 3.22. Champ de vecteurs associé à notre système

Sur cette même figure, l'espace d'état est découpé en six zones. En effet, du fait du recollement de la ligne $\varphi = \pi$ avec la ligne $\varphi = -\pi$, l'espace d'état a une nature cylindrique et on a six zones, alors que l'on en visualise huit sur la figure.

Relation de succession : on définit la relation, notée \hookrightarrow entre zones \mathbb{A}, \mathbb{B} de l'espace d'état comme suit :

$$(\mathbb{A} \hookrightarrow \mathbb{B}) \Leftrightarrow \exists \mathbf{x}_0 \in \mathbb{A} \in \phi(\eta(\mathbf{x}_0), \mathbf{x}_0) \in \mathbb{B}$$

où $\eta(\mathbf{x}_0)$ est le temps auquel le système sort de \mathbb{A} .

Convention : s'il existe $\mathbf{x}_0 \in \mathbb{A}$, tel que $\forall t > 0, \phi(t, \mathbf{x}_0) \subset \mathbb{A}$ alors $\eta(\mathbf{x}_0) = \infty$. Ainsi, $\phi(\eta(\mathbf{x}_0), \mathbf{x}_0) \in \mathbb{A}$ et donc on aura $(\mathbb{A} \hookrightarrow \mathbb{A})$.

1) Dessiner le graphe associé à cette relation.

2) En déduire un surensemble de l'espace d'état dans lequel le système restera piégé.

Exercice 3.5. Robot voilier

On considère le voilier décrit par les équations d'état suivantes :

$$\begin{cases} \dot{x} &= v \cos \theta + p_1 a \cos \psi \\ \dot{y} &= v \sin \theta + p_1 a \sin \psi \\ \dot{\theta} &= \omega \\ \dot{v} &= \frac{f_s \sin \delta_s - f_r \sin u_1 - p_2 v^2}{p_9} \\ \dot{\omega} &= \frac{f_s (p_6 - p_7 \cos \delta_s) - p_8 f_r \cos u_1 - p_3 \omega v}{p_{10}} \\ f_s &= p_4 \|\mathbf{w}_{\text{ap}}\| \sin (\delta_s - \psi_{\text{ap}}) \\ f_r &= p_5 v \sin u_1 \\ \sigma &= \cos \psi_{\text{ap}} + \cos u_2 \\ \delta_s &= \begin{cases} \pi + \psi_{\text{ap}} & \text{si } \sigma \leq 0 \\ -\text{sign}(\sin \psi_{\text{ap}}) \cdot u_2 & \text{sinon} \end{cases} \\ \mathbf{w}_{\text{ap}} &= \begin{pmatrix} a \cos (\psi - \theta) - v \\ a \sin (\psi - \theta) \end{pmatrix} \\ \psi_{\text{ap}} &= \text{angle } \mathbf{w}_{\text{ap}} \end{cases}$$

où (x, y, θ) correspond à la pose du bateau, v est sa vitesse d'avance, ω est sa vitesse angulaire, f_s (« *s* » comme *sail*) est la force du vent sur la voile, f_r (« *r* » comme *rudder*) est la force de l'eau sur le gouvernail, δ_s est l'angle de la voile, a est la vitesse du vent vrai, ψ est l'angle du vent vrai (voir figure 3.9) et \mathbf{w}_{ap} est le vecteur vent apparent. La quantité σ est un indicateur de tension de l'écoute. Ainsi, si $\sigma \leq 0$, l'écoute est lâche et la voile est en drapeau et si $\sigma \geq 0$, l'écoute est tendue et la voile est gonflée par le vent. Dans ces équations, les p_i sont des paramètres de conception du voilier. On prendra les valeurs suivantes, données en unités internationales : $p_1 = 0.1$ (coefficients de dérive), $p_2 = 1$ (coefficients de trainée), $p_3 = 6\,000$ (frottement angulaire de la coque sur l'eau), $p_4 = 1\,000$ (portance de la voile), $p_5 = 2\,000$ (portance du gouvernail), $p_6 = 1$ (position du centre de poussée du vent sur la voile), $p_7 = 1$ (position du mat), $p_8 = 2$ (position du gouvernail), $p_9 = 300$ (masse du voilier) et $p_{10} = 10\,000$ (moment d'inertie du voilier).

1) Simuler le bateau sous MATLAB.

2) Implémenter le régulateur proposé à la section 3.3. On prendra les paramètres suivants $\zeta = \frac{\pi}{4}$ pour l'angle de près, $r = 10$ m pour le rayon du couloir, $\delta_r^{\max} = 1$ rad pour l'angle maximal du gouvernail et $\beta = \frac{\pi}{4}$ pour l'angle de voile en vent de travers.

Exercice 3.6. Drone volant

On considère un drone volant [BEA 12] du type de celui représenté sur la figure 3.23a. Il s'agit d'un avion de 1 kg entièrement autonome. Un modèle possible pour décrire sa dynamique, très fortement inspiré de celui du *Faser Ultra Stick* [KLE 06] de la figure 3.23b, est donné par :

$$\begin{pmatrix} \dot{\mathbf{p}} \\ \dot{\psi} \\ \dot{\theta} \\ \dot{\varphi} \\ \dot{\mathbf{v}} \\ \dot{\omega} \end{pmatrix} = \begin{pmatrix} \mathbf{R}_{\text{euler}}(\varphi, \theta, \psi) \cdot \mathbf{v} \\ 0 & \frac{\sin \varphi}{\cos \theta} & \frac{\cos \varphi}{\cos \theta} \\ 0 & \cos \varphi & -\sin \varphi \\ 1 \tan \theta \sin \varphi & \tan \theta \cos \varphi & \\ 9.81 \cdot \begin{pmatrix} -\sin \theta \\ \cos \theta \sin \varphi \\ \cos \theta \cos \varphi \end{pmatrix} + \mathbf{f}_a + \begin{pmatrix} u_1 \\ 0 \\ 0 \end{pmatrix} - \omega \wedge \mathbf{v} \\ -\omega_3 \omega_2 - \frac{\|\mathbf{v}\|^2}{10} (\beta + 2u_3 + \frac{5\omega_1 - \omega_3}{\|\mathbf{v}\|}) \\ \omega_3 \omega_1 - \frac{100}{100} (1 + 20\alpha - 2u_3 + 30u_2 + \frac{300\omega_2}{\|\mathbf{v}\|}) \\ \frac{\omega_1 \omega_2}{10} + \frac{\|\mathbf{v}\|^2}{10} (\beta + \frac{u_3}{2} + \frac{\omega_1 - 2\omega_3}{2\|\mathbf{v}\|}) \end{pmatrix}$$

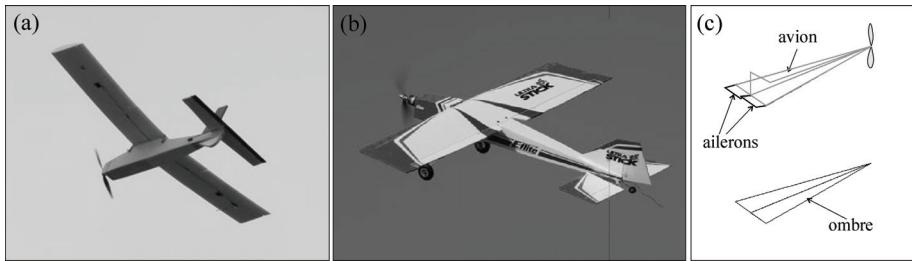


Figure 3.23. (a) Avion μ -STIC de l'ENSTA Bretagne ; (b) avion Faser Ultra Stick de l'Université du Minnesota ; (c) représentation graphique utilisée pour la simulation

avec :

$$\begin{aligned} \alpha &= \text{atan} \left(\frac{v_3}{v_2} \right), \quad \beta = \text{asin} \left(\frac{v_2}{\|\mathbf{v}\|} \right) \\ \mathbf{f}_a &= \frac{\|\mathbf{v}\|^2}{500} \begin{pmatrix} -\cos \alpha \cos \beta \cos \alpha \sin \beta & \sin \alpha \\ \sin \beta & \cos \beta & 0 \\ -\sin \alpha \cos \beta \sin \alpha \sin \beta & -\cos \alpha \end{pmatrix} \\ &\cdot \begin{pmatrix} 4 + (-0.3 + 10\alpha + \frac{10\omega_2}{\|\mathbf{v}\|} + 2u_3 + 0.3u_2)^2 + |u_2| + 3|u_3| \\ -50\beta + \frac{10\omega_3 - 3\omega_1}{\|\mathbf{v}\|} \\ 10 + 500\alpha + \frac{400\omega_2}{\|\mathbf{v}\|} + 50u_3 + 10u_2 \end{pmatrix} \end{aligned}$$

Dans ce modèle, toutes les quantités sont données en unités internationales. Le vecteur $\mathbf{p} = (x, y, z)$ représente la position de l'avion, avec l'axe des z orienté vers le centre de la terre. L'orientation du drone est représentée par les angles d'Euler (φ, θ, ψ) et la matrice d'Euler $\mathbf{R}_{\text{euler}}(\varphi, \theta, \psi)$ est donnée par la formule [1.7] page 20.

Le vecteur \mathbf{v} représente la vitesse de l'avion exprimée dans son propre repère. Le vecteur de rotation de l'avion est ici dénoté $\boldsymbol{\omega}$. Il est relié aux dérivées des angles d'Euler par la formule [1.11] de la page 23. Notons que cette même formule [1.11] nous donne les trois premières équations de notre modèle d'état. Les angles α et β sont correspondant aux angles d'attaque et de dérapage. Le vecteur \mathbf{f}_a correspond à l'accélération engendrée par les forces exercées par l'air. Une vision géométrique simplifiée du drone, donnée par la figure 3.23c, fait apparaître une hélice pour la propulsion, et deux ailerons pour la direction. Le vecteur des entrées $\mathbf{u} = (u_1, u_2, u_3)$ intervenant dans notre modèle d'état, fait apparaître l'accélération propulsive $u_1 \in [0, 10]$ (en ms^{-2}), la somme $u_2 \in [-0.6, 0.6]$ (en radian) des deux angles des ailerons et $u_3 \in [-0.3, 0.3]$ (en radian) le différentiel entre les deux ailerons.

1) Sous MATLAB, faire une simulation de cet avion. Pour le graphisme, on pourra s'inspirer de la figure 3.23c.

2) Proposer une loi de commande en cap, en assiette et en vitesse.

3) On souhaite amener le robot à se positionner sur un cercle de rayon $\bar{r} = 100$ m, centré en zéro à une altitude de 50 m et une vitesse de $\bar{v} = 15 \text{ ms}^{-1}$. Donner la loi de commande et illustrer le comportement associé du robot sous MATLAB.

3.5. Corrections

Correction de l'exercice 3.1 (robot char sur une ligne)

1) La simulation complète correspondant à cet exercice est donnée par le programme `tank_line.m`.

2) La commande en cap est donnée par `u=sawtooth(theta-bar-x(3))` ; L'utilisation de la fonction *dents de scie* (voir figure 3.2 page 101) `sawtooth()` permet de filtrer les sauts d'angles $\pm 2\pi$.

3) Pour le suivi, nous combinons une commande en cap avec une commande par champ de vecteurs. Le champ utilisé afin de rendre attractive la ligne est $\bar{\theta} = \varphi - \text{atan}(e)$ où e est l'écart algébrique à la ligne. Le programme de suivi est :

```
a=[-40 ; -4] ; b=[20 ; 6] ; % ligne
x=[-30 ; -10 ; pi ; 1] ; % x,y,theta,v
dt=0.1 ;
for t=0 :dt :50,
phi=angle(b-a) ; % angle de la ligne
m=[x(1) ; x(2)] ;
e=det([b-a,m-a])/norm(b-a) ; % écart a la ligne
```

```

thetabar=phi-atan(e); % cap a suivre
u=sawtooth(thetabar-x(3)); % commande en cap
x=x+f(x,u)*dt;
end;

```

Correction de l'exercice 3.2 (voiture de Van der Pol)

1) La simulation se fait par le script qui suit :

```

x=[0 ;0.1 ;pi/6 ;2 ;0.6] ; % x,y,theta,v,delta
dt=0.01 ; u=[0 0] ;
for t=0 :dt :100,
x=x+f(x,u)*dt ;
end ;

```

La fonction d'évolution est la suivante :

```

function xdot=f(x,u)
xdot=[x(4)*cos(x(5))*cos(x(3)) ;
x(4)*cos(x(5))*sin(x(3)) ;
x(4)*sin(x(5))/3 ;
u(1) ;u(2) ] ;

```

2) Il suffit de prendre :

$$\mathbf{u} = \rho(\mathbf{x}, \bar{\mathbf{u}}) = k \left(\bar{\mathbf{u}} - \begin{pmatrix} v \cos \delta \\ \frac{v \sin \delta}{L} \end{pmatrix} \right)$$

avec k grand. Ainsi, en suivant le même raisonnement que ceux élaborés pour l'ampli-op, on obtient que si le système ainsi bouclé est stable, on a :

$$\begin{pmatrix} \bar{u}_1 \\ \bar{u}_2 \end{pmatrix} \simeq \begin{pmatrix} v \cos \delta \\ \frac{v \sin \delta}{L} \end{pmatrix}$$

et le système bouclé devient le modèle char. Pour la simulation, on pourra prendre $k = 10$. Ainsi, le régulateur s'exprime par :

```
u=10*(ubar-[x(4)*cos(x(5)) ;x(4)*sin(x(5))/3]);
```

3) Pour \bar{u}_1 , il n'y a rien à faire. Pour \bar{u}_2 , on pourra prendre une commande proportionnelle en cap. La commande correspondant à cette deuxième boucle est :

$$\bar{\mathbf{u}} = \sigma(\mathbf{x}, \mathbf{w}) = \begin{cases} \bar{u}_1 = a_1 w_1 \\ \bar{u}_2 = a_2 \cdot \text{sawtooth}(w_2 - \theta) \end{cases}$$

avec par exemple $a_1 = 1$ et $a_2 = 5$. Les lignes de codes correspondantes sont :

```
ubar=[w(1) ;5*sawtooth(w(2)-x(3))] ;
```

4) On rajoute une troisième boucle donnée par :

$$\mathbf{w} = \tau(\mathbf{x}) = \begin{pmatrix} v_0 \\ \text{angle} \left(\begin{pmatrix} y \\ -(0.01 x^2 - 1) y - x \end{pmatrix} \right) \end{pmatrix}$$

où v_0 est la vitesse désirée pour la voiture et la fonction `angle` renvoie l'angle d'un vecteur (voir `angle.m`). On prendra par exemple $v_0 = 10 \text{ ms}^{-1}$. Le code complet (voir fichier `vanderpol.m`) correspondant à ces trois boucles est le suivant :

```
vdp=[x(2) ;-(0.01*x(1)^2-1)*x(2)-x(1)] ; % dynamique de Van der Pol
w=[10 ;angle(vdp)] ; % consigne en cap et vitesse (ici 10m/s)
ubar=[w(1) ;5*sawtooth(w(2)-x(3))] ;
u=10*(ubar-[x(4)*cos(x(5)) ;x(4)*sin(x(5)/3)]) ;
```

Le code pour dessiner le champ de vecteurs afin de vérifier le bon comportement de la voiture est :

```
function draw(x)
Lx=50 ;Ly=30 ;
Mx = -Lx :2 :Lx ; My = -Ly :2 :Ly ;
[X1,X2] = meshgrid(Mx,My) ;
VX=X2 ; VY=-(0.01*X1.^2-1).*X2-X1 ;
VX=VX./sqrt(VX.^2+VY.^2) ; VY=VY./sqrt(VX.^2+VY.^2) ;
quiver(Mx,My,VX,VY) ;
```

Correction de l'exercice 3.3 (ancrage)

1) Puisque $\dot{x}^2 + \dot{y}^2 = 1$, le robot ne peut pas s'arrêter.

2) Les équations d'état demandées sont les suivantes :

$$\begin{cases} \text{(i)} \quad \dot{\varphi} = \frac{\sin \varphi}{d} + u \\ \text{(ii)} \quad \dot{d} = -\cos \varphi \\ \text{(iii)} \quad \dot{\alpha} = -\frac{\sin \varphi}{d} \end{cases}$$

Les preuves pour (ii) et (iii) sont immédiates. Prouvons maintenant (i). D'après la figure 3.24, nous avons $\varphi - \theta + \alpha = \pi$. Soit, après dérivation :

$$\dot{\varphi} = -\dot{\alpha} + \dot{\theta} = \frac{\sin \varphi}{d} + u$$

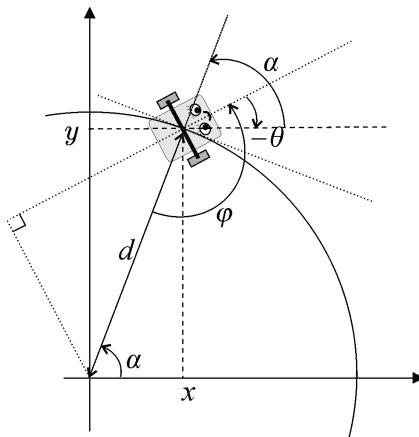


Figure 3.24. Utilisation de la symétrie cylindrique pour passer d'un système d'ordre 3 à un système d'ordre 2

3) Cette représentation nous permet de passer d'un système de dimension 3 à un autre système de dimension 2 (car α n'est plus nécessaire). Une représentation graphique du champ de vecteur devient alors possible (voir exercice 3.4).

4) L'idée de cette loi de commande est d'utiliser une régulation proportionnelle lorsque le robot pointe approximativement vers 0 (c'est-à-dire si $\cos \varphi > \frac{1}{\sqrt{2}}$) et de tourner à gauche si ça n'est pas le cas, afin de pointer vers zéro ultérieurement.

5) Le programme de simulation est rangé dans le fichier `anchor.m`.

Correction de l'exercice 3.4 (discréétisation de l'espace d'état)

Le graphe est représenté 3.25 sous deux formes différentes. L'état sera piégé sous la ligne en gras.

Sous une forme matricielle, le graphe est donné par :

$$\mathbf{G} = \begin{pmatrix} 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & [0, 1] \end{pmatrix}$$

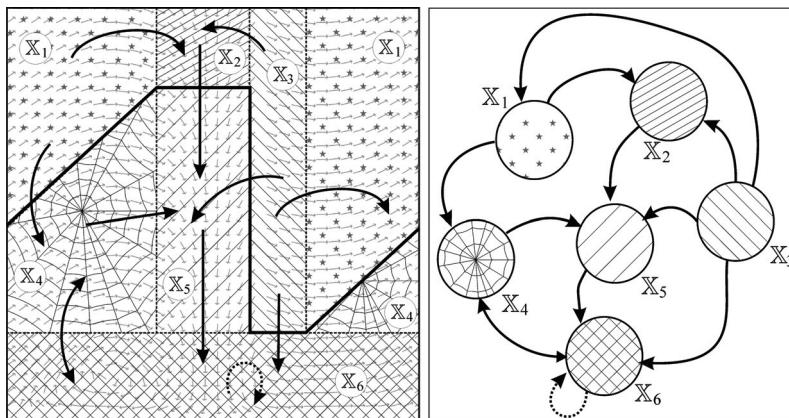


Figure 3.25. Discréétisation de la dynamique du système par une partition de l'espace d'état afin d'obtenir un graphe de transition

où l'intervalle booléen $[0, 1]$ signifie 0 ou 1. La clôture transitive est :

$$G^+ = G + G^2 + G^3 + \dots = \begin{pmatrix} 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{pmatrix}$$

La diagonale nous donne l'attracteur du graphe $X_\infty = X_4 \cup X_5 \cup X_6$. Ainsi, l'attracteur du système \mathbb{A} satisfait :

$$\mathbb{A} \subset X_4 \cup X_5 \cup X_6$$

Correction de l'exercice 3.5 (robot voilier)

1) Pour la simulation, on peut utiliser une méthode d'Euler, en prenant pour fonction d'évolution :

```
function xdot = f(x,u)
theta=x(3); v=x(4); w=x(5); deltar=u(1); deltasmax=u(2);
w_ap=[awind*cos(psi-theta)-v;awind*sin(psi-theta)];
psi_ap=angle(w_ap); a_ap=norm(w_ap);
sigma=cos(psi_ap)+cos(deltasmax);
if (sigma < 0), deltas=pi+psi_ap;
else deltas=-sign(sin(psi_ap))*deltasmax;
end;
```

```

fr = p5*v*sin(deltar); fs = p4*a_ap*sin(deltas-psi_ap);
dx=v*cos(theta)+p1*awind*cos(psi);
dy=v*sin(theta)+p1*awind*sin(psi);
dtheta=w;
dv=(1/p9)*(sin(deltas)*fs-sin(deltar)*fr-p2*v^2);
dw=(1/p10)*((p6-p7*cos(deltas))*fs-p8*cos(deltar)*fr-p3*w*v);
xdot=[dx ;dy ;dtheta ;dv ;dw];
end

```

2) Pour faire un suivi de ligne, on écrit le régulateur comme suit :

```

function [u,q] = control(x,q)
theta=x(3); r=10; zeta=pi/4; m=[x(1);x(2)];
e=det([(b-a)/norm(b-a),m-a]);
phi=angle(b-a);
if (abs(e) > r), q=sign(e); end;
thetabar=phi-atan(e/r);
if (cos(psi-thetabar)+cos(zeta) < 0)||((abs(e) <
r)&&(cos(psi-phi)+cos(zeta)<0))
thetabar=pi+psi-zeta*q;
end;
deltar=0.3*(sawtooth(theta-thetabar));
deltasmax=pi/4*(cos(psi-thetabar)+1);
u=[deltar;deltasmax];
end

```

Correction de l'exercice 3.6 (drone volant)

2) Pour la loi de commande, il nous faut régler la propulsion u_1 , l'assiette par u_2 et le cap par u_3 .

Propulsion : sur notre avion, la propulsion doit être dans l'intervalle $[0, 10]$ (en ms^{-2}). On prendra :

$$u_1 = 5(1 + \tanh(\bar{v} - \|\mathbf{v}\|))$$

où \mathbf{v} est le vecteur vitesse de l'avion et \bar{v} , la vitesse consigne. Si l'avion est à la bonne vitesse, on a une propulsion moyenne de 5 N. Sinon, grâce à la fonction de saturation tangente hyperbolique $\tanh(\cdot)$ (voir formule [3.1]), on aura toujours une propulsion dans l'intervalle $[0, 10]$.

Assiette : l'assiette sera réglée par la somme des angles des ailerons que l'on suppose dans l'intervalle $[-0.6, 0.6]$ rad. On prendra :

$$u_2 = -0.3 \cdot (\tanh(5(\bar{\theta} - \theta)) + |\sin \varphi|)$$

On a ainsi une commande proportionnelle sur l'assiette saturée par la fonction $tanh$. Le gain de 5 nous signale que la réaction devient significative à partir d'une erreur d'assiette égale à $\frac{1}{5}\text{rad} = 11\text{ deg}$. Pour une gîte φ assez forte, l'avion perd en altitude, ce qui justifie le terme en $|\sin \varphi|$.

Cap : sur notre modèle de l'avion, c'est la gîte φ qui permet un changement de cap de l'avion, comme en moto. Si l'on souhaite avoir un cap de $\bar{\psi}$, on pourra choisir la gîte suivante :

$$\bar{\varphi} = \tanh(5 \cdot \text{sawtooth}(\bar{\psi} - \psi))$$

En effet, on définit l'erreur en cap $\bar{\psi} - \psi$ que l'on filtre par la fonction à dents de scie *sawtooth* (voir figure 3.2 page 101). On multiplie par le gain 5 de la commande proportionnelle. Ce gain de 5 nous dit que la commande réagit de façon significative pour corriger le cap par la gîte lorsque $\text{sawtooth}(\bar{\psi} - \psi) = \frac{1}{5}$, c'est-à-dire lorsque $\bar{\psi} - \psi = \frac{1}{5}\text{rad} \simeq 11\text{ deg}$, ce qui semble raisonnable. On propose alors la commande :

$$u_3 = -0.3 \cdot \tanh(\bar{\varphi} - \varphi)$$

où la fonction *tanh* est à nouveau utilisée comme fonction de saturation. Ainsi, la consigne $\bar{\varphi}$ pour φ sera toujours dans l'intervalle $[-\frac{\pi}{4}, \frac{\pi}{4}]$.

Champ de vecteur : pour définir le comportement de l'avion afin qu'il puisse rejoindre son cercle, on affecte à chaque position de l'espace $\mathbf{p} = (p_x, p_y, p_z)$ une direction $\bar{\psi}$ et une assiette $\bar{\theta}$ donnée par :

$$\bar{\psi} = \text{angle}(\mathbf{p}) + \frac{\pi}{2} + \tanh \frac{\sqrt{p_1^2 + p_2^2} - \bar{r}}{50}$$

$$\bar{\theta} = -0.3 \cdot \tanh \frac{\bar{z} - z}{10}$$

Le champ de vecteur qui en résulte possède un cycle limite qui correspond à notre cercle de rayon \bar{r} . Pour le calcul de $\bar{\psi}$, l'expression de l'erreur nous indique une précision de l'ordre de 50 m. La composante $\tanh \frac{1}{50}(\sqrt{p_1^2 + p_2^2} - \bar{r})$ rend notre cercle attractif et celle en $\text{angle}(\mathbf{p}) + \frac{\pi}{2}$ nous crée le champ tournant. Pour l'expression de $\bar{\theta}$ nous avons une fonction de saturation qui nous assure une précision de 10 m en altitude et une assiette maximale de $0.2 \cdot \frac{\pi}{2} \simeq 0.31\text{ rad}$.

Récapitulatif : le régulateur pour notre avion aura pour entrées : les consignes $\bar{z}, \bar{r}, \bar{v}$, les angles d'Euler de l'avion, sa position \mathbf{p} et son vecteur vitesse. Il nous générera le vecteur de commande \mathbf{u} comme donné par l'algorithme :

Régulateur in : $\mathbf{p}, \mathbf{v}, \varphi, \theta, \psi, \bar{z}, \bar{r}, \bar{v}$; out : \mathbf{u}	
1	$\bar{\psi} = \text{angle}(\mathbf{p}) + \frac{\pi}{2} + \tanh \frac{\sqrt{p_1^2 + p_2^2 - \bar{r}}}{50}$
2	$\bar{\theta} = -0.3 \cdot \tanh \frac{\bar{z} - z}{10}$
3	$\bar{\varphi} = \tanh(5 \cdot \text{sawtooth}(\bar{\psi} - \psi))$
4	$\mathbf{u} = \begin{pmatrix} 5 (1 + \tanh(\bar{v} - \ \mathbf{v}\)) \\ -0.3 \cdot (\tanh(5(\bar{\theta} - \theta)) + \sin \varphi) \\ -0.3 \cdot \tanh(\bar{\varphi} - \varphi) \end{pmatrix}$

La figure 3.26 illustre le résultat de la simulation. Le programme correspondant est rangé dans le fichier `plane.m`.

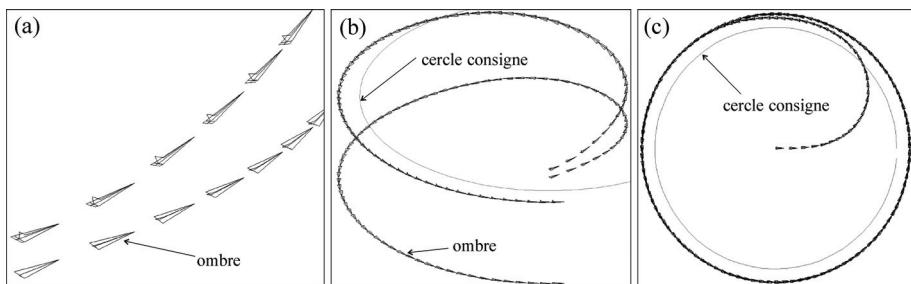


Figure 3.26. (a) Phase initiale : le robot s'élève et tourne sur sa gauche ; (b) le robot rejoint son cercle ; (c) vue de dessus : on remarque qu'il existe un biais entre le cercle consigne et le cercle réalisé

Chapitre 4

Guidage

Dans les chapitres précédents, nous avons étudié comment construire une loi de commande afin qu'un robot décrit par des équations d'état non linéaires (voir chapitre 2) ou pour lequel nous avons une idée de son comportement (voir chapitre 3). Le *guidage* se place à un plus haut niveau et s'intéresse à la consigne à donner au régulateur afin que le robot soit capable d'accomplir une mission donnée. Elle devra donc prendre en compte la connaissance de notre environnement, la présence d'obstacles, la rotundité de la terre, etc. Classiquement, le guidage s'applique à quatre types d'environnements : terrestre, marin, aérien et spatial. Ici, du fait des champs d'applications couverts dans ce livre, nous n'étudierons pas l'environnement spatial.

4.1. Guidage sur une sphère

Pour de grands trajets sur la surface de la terre, le repère cartésien, qui suppose une terre plane, ne peut plus être considéré. Nous devons alors repenser nos lois de commande en se repérant relativement à un système de coordonnées sphériques (aussi appelées *coordonnées géographiques*), qui tourne avec la terre. Notons ℓ_x la longitude et ℓ_y la latitude du point considéré. La transformation dans le système de coordonnées géographiques s'écrit :

$$\mathcal{T} : \begin{pmatrix} \ell_x \\ \ell_y \\ \rho \end{pmatrix} \rightarrow \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} \rho \cos \ell_y \cos \ell_x \\ \rho \cos \ell_y \sin \ell_x \\ \rho \sin \ell_y \end{pmatrix} \quad [4.1]$$

Lorsque $\rho = 6\ 370$ km, c'est que nous sommes à la surface de la terre que nous supposerons sphérique (voir figure 4.1a).

Considérons deux points **a**, **m** à la surface de la terre, comme illustré par la figure 4.1b, repérés par leurs coordonnées géographiques. Ici **a** représente par exemple un

point de référence à atteindre et \mathbf{m} est le centre de notre robot. En supposant que les deux points \mathbf{a} , \mathbf{m} ne soient pas trop éloignés (pas plus de 100 km), on peut alors raisonner dans le plan, à l'aide d'une carte locale, comme sur la figure 4.2a.

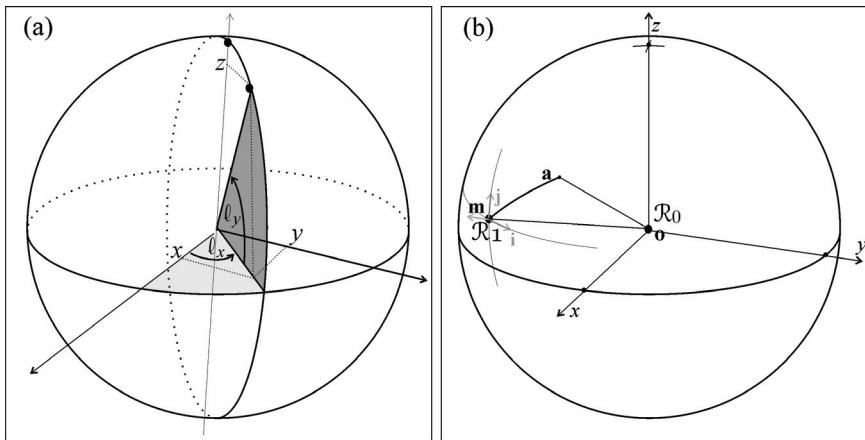


Figure 4.1. (a) Système de coordonnées géographiques ; (b) on cherche à exprimer \mathbf{a} dans la carte locale \mathcal{R}_1

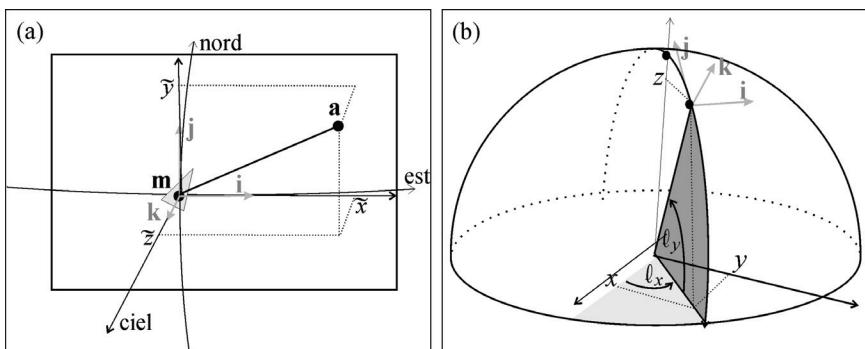


Figure 4.2. (a) La carte donne une vision locale et cartésienne autour du robot ; (b) un léger déplacement $d\ell_x$, $d\ell_y$, $d\rho$ engendre un déplacement dx , dy , dz dans la carte locale

Différencions la relation [4.1]. Nous obtenons :

$$\begin{pmatrix} dx \\ dy \\ dz \end{pmatrix} = \underbrace{\begin{pmatrix} -\rho \cos \ell_y \sin \ell_x & -\rho \sin \ell_y \cos \ell_x & \cos \ell_y \cos \ell_x \\ \rho \cos \ell_y \cos \ell_x & -\rho \sin \ell_y \sin \ell_x & \cos \ell_y \sin \ell_x \\ 0 & \rho \cos \ell_y & \sin \ell_y \end{pmatrix}}_{=J} \cdot \begin{pmatrix} d\ell_x \\ d\ell_y \\ d\rho \end{pmatrix}$$

Cette formule peut être utilisée pour trouver les coordonnées géographiques des directions cardinales, qui changent en fonction de l'endroit où le robot \mathbf{m} se trouve. Par exemple, le vecteur correspondant à la direction *Est* se trouve sur la première colonne de la matrice \mathbf{J} , la direction nord se trouve sur la deuxième colonne et l'altitude se lit sur la troisième colonne. Nous allons donc pouvoir construire un repère (Est-Nord-Altitude) \mathcal{R}_1 robot centré (en gris sur la figure 4.2a) qui correspond à la carte locale. La matrice de rotation correspondante est obtenue en normalisant chaque colonne de la matrice jacobienne \mathbf{J} . Nous obtenons :

$$\mathbf{R} = \begin{pmatrix} -\sin \ell_x - \sin \ell_y \cos \ell_x & \cos \ell_y \cos \ell_x \\ \cos \ell_x & -\sin \ell_y \sin \ell_x \\ 0 & \cos \ell_y \end{pmatrix} \quad [4.2]$$

La transformation qui permet de passer du repère géographique \mathcal{R}_0 à la carte locale \mathcal{R}_1 est :

$$\mathbf{v}_{|\mathcal{R}_1} = \mathbf{R}^T \cdot \mathbf{v}_{|\mathcal{R}_0} \quad [4.3]$$

Cette relation de changement de repère peut s'appliquer dans différents contextes comme par exemple pour mettre en cohérence des informations recueillies par deux robots différents.

EXEMPLE 4.1. – *Un robot situé en \mathbf{m} : (ℓ_x^m, ℓ_y^m) se déplace avec un vecteur vitesse \mathbf{v}^m relativement au sol fixe. Ce vecteur est exprimé dans la carte locale du robot. Cherchons avec quelle vitesse ce vecteur \mathbf{v}^m est perçu dans la carte locale d'un observateur localisé en \mathbf{a} : (ℓ_x^a, ℓ_y^a) . D'après [4.3], on a :*

$$\begin{cases} \mathbf{v}^m = \mathbf{R}^T(\ell_x^m, \ell_y^m) \cdot \mathbf{v}_{|\mathcal{R}_0} \\ \mathbf{v}^a = \mathbf{R}^T(\ell_x^a, \ell_y^a) \cdot \mathbf{v}_{|\mathcal{R}_0} \end{cases}$$

Donc :

$$\mathbf{v}^a = \mathbf{R}^T(\ell_x^a, \ell_y^a) \cdot \mathbf{R}(\ell_x^m, \ell_y^m) \cdot \mathbf{v}^m$$

Ce type de calcul est utile lorsque, par exemple, deux robots cherchent à se rencontrer.

Passage à une carte locale : prenons une carte locale \mathcal{R}_m centrée sur un point \mathbf{m} , (c'est-à-dire un repère positionné à la surface de la terre dont l'origine est \mathbf{m} et dont les directions sont est-nord-ciel). Cherchons maintenant à exprimer dans \mathcal{R}_m les coordonnées $(\tilde{x}, \tilde{y}, \tilde{z})$ d'un point \mathbf{a} repéré par ses coordonnées GPS (ℓ_x, ℓ_y, ρ) .

Nous pouvons passer des coordonnées géographiques aux coordonnées locales par la relation :

$$\begin{pmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{z} \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ \rho^m \end{pmatrix} \stackrel{[4.3]}{=} \underbrace{\begin{pmatrix} -\sin \ell_x^m & \cos \ell_x^m & 0 \\ -\cos \ell_x^m \sin \ell_y^m & -\sin \ell_x^m \sin \ell_y^m \cos \ell_y^m & \\ \cos \ell_x^m \cos \ell_y^m & \cos \ell_y^m \sin \ell_x^m & \sin \ell_y^m \end{pmatrix}}_{R^T(\ell_x^m, \ell_y^m)} \cdot \underbrace{\begin{pmatrix} \rho \cos \ell_y \cos \ell_x \\ \rho \cos \ell_y \sin \ell_x \\ \rho \sin \ell_y \end{pmatrix}}_{a|\mathcal{R}_0}$$

Lorsque $\ell_x^m \simeq \ell_x$ et $\ell_y^m \simeq \ell_y$, une approximation à l'ordre 1 s'obtient directement en s'aidant de la figure 4.2b :

$$\begin{pmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{z} \end{pmatrix} \simeq \begin{pmatrix} \rho \cos \ell_y \cdot (\ell_x - \ell_x^m) \\ \rho (\ell_y - \ell_y^m) \\ \rho - \rho^m \end{pmatrix} \quad [4.4]$$

En utilisant des relations trigonométriques, nous aurions aussi pu obtenir formellement ces mêmes résultats, mais de façon plus laborieuse. Par exemple, nous obtenons la dernière des trois équations par le raisonnement suivant :

$$\begin{aligned} \tilde{z} &= \rho (\cos \ell_x^m \cos \ell_y^m \cos \ell_y \cos \ell_x + \cos \ell_y^m \sin \ell_x^m \cos \ell_y \sin \ell_x + \sin \ell_y^m \sin \ell_y) - \rho^m \\ &= \rho (\cos \ell_y^m \cos \ell_y (\cos \ell_x^m \cos \ell_x + \sin \ell_x^m \sin \ell_x) + \sin \ell_y^m \sin \ell_y) - \rho^m \\ &= \rho (\cos \ell_y^m \cos \ell_y \underbrace{\cos (\ell_x^m - \ell_x)}_{\simeq 1} + \sin \ell_y^m \sin \ell_y - \rho^m) \simeq \rho \underbrace{\cos (\ell_y^m - \ell_y)}_{\simeq 1} - \rho^m \simeq \rho - \rho^m \end{aligned}$$

Notons que quand le robot bouge dans une zone de faible diamètre, on choisit parfois un point de référence qui n'est pas le centre **m** du robot, comme par exemple l'endroit d'où l'on lance notre robot.

4.2. Planification de trajectoires

En complète autonomie du robot, la trajectoire désirée doit être planifiée [LAV 06]. Très souvent, ces trajectoires sont des polynômes et ceci pour deux raisons. La première est que l'espace des polynômes possède une structure d'espace vectoriel et peut donc profiter de la puissance de l'algèbre linéaire. La seconde est qu'ils sont faciles à dériver, ce qui est utile pour la commande par bouclage linéarisant, car elle demande les dérivées successives des consignes.

4.2.1. Exemple simple

Illustrons sur l'exemple d'un robot de type char comment une telle planification peut être effectuée. Supposons qu'à l'instant initial $t = 0$, le robot se trouve au point

(x_0, y_0) et que nous souhaitions atteindre au temps t_1 le point (x_1, y_1) avec une vitesse égale à (v_x^1, v_y^1) . On se propose une trajectoire polynomiale de la forme :

$$x_d = a_x t^2 + b_x t + c_x$$

$$y_d = a_y t^2 + b_y t + c_y$$

Il nous faut résoudre le système d'équations :

$$\begin{aligned} c_x &= x_0, & c_y &= y_0 \\ a_x t_1^2 + b_x t_1 + c_x &= x_1 & a_y t_1^2 + b_y t_1 + c_y &= y_1 \\ 2a_x t_1 + b_x &= v_x^1, & 2a_y t_1 + b_y &= v_y^1 \end{aligned}$$

qui est linéaire. Nous obtenons aisément :

$$\begin{pmatrix} a_x \\ a_y \\ b_x \\ b_y \\ c_x \\ c_y \end{pmatrix} = \begin{pmatrix} \frac{1}{t_1^2} x_0 - \frac{1}{t_1^2} x_1 + \frac{1}{t_1} v_x^1 \\ \frac{1}{t_1^2} y_0 - \frac{1}{t_1^2} y_1 + \frac{1}{t_1} v_y^1 \\ -v_x^1 - \frac{2}{t_1} x_0 + \frac{2}{t_1} x_1 \\ -v_y^1 - \frac{2}{t_1} y_0 + \frac{2}{t_1} y_1 \\ x_0 \\ y_0 \end{pmatrix}$$

On a donc :

$$\begin{aligned} \dot{x}_d &= 2a_x t + b_x & \dot{y}_d &= 2a_y t + b_y \\ \ddot{x}_d &= 2a_x, & \ddot{y}_d &= 2a_y \end{aligned}$$

En insérant ces quantités dans une loi de commande obtenue par une méthode par bouclage linéarisant (comme par exemple celle donnée par l'équation [2.9]), nous obtenons une régulation qui réalise nos objectifs.

4.2.2. Polynômes de Bézier

On cherche ici à généraliser l'approche présentée dans la section précédente. A partir de points de contrôle $\mathbf{p}_0, \mathbf{p}_1, \dots, \mathbf{p}_n$, on peut générer un polynôme $\mathbf{f}(t)$ tel que $\mathbf{f}(0) = \mathbf{p}_0$, $\mathbf{f}(1) = \mathbf{p}_n$ et tel que pour $t \in [0, 1]$, le polynôme $\mathbf{f}(t)$ soit successivement attiré par les \mathbf{p}_i avec $i \in \{0, \dots, n\}$. Afin de bien comprendre le mécanisme de construction des polynômes de Bézier, prenons différents cas :

– cas $n = 1$. On prend l'interpolation linéaire classique :

$$\mathbf{f}(t) = (1 - t) \mathbf{p}_0 + t \mathbf{p}_1$$

Le point $\mathbf{f}(t)$ correspond à un barycentre entre les points de contrôle \mathbf{p}_0 et \mathbf{p}_1 , et les poids affectés à ces deux points changent dans le temps ;

– cas $n = 2$. On a maintenant trois points de contrôle $\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2$. On crée un point de contrôle auxiliaire \mathbf{p}_{01} qui se promène sur le segment $[\mathbf{p}_0, \mathbf{p}_1]$ et un autre \mathbf{p}_{12} qui associé au segment $[\mathbf{p}_1, \mathbf{p}_2]$. On prend :

$$\begin{aligned}\mathbf{f}(t) &= (1-t)\mathbf{p}_{01} + t\mathbf{p}_{12} \\ &= (1-t)\underbrace{((1-t)\mathbf{p}_0 + t\mathbf{p}_1)}_{\mathbf{p}_{01}} + t\underbrace{((1-t)\mathbf{p}_1 + t\mathbf{p}_2)}_{\mathbf{p}_{12}} \\ &= (1-t)^2\mathbf{p}_0 + 2(1-t)t\mathbf{p}_1 + t^2\mathbf{p}_2\end{aligned}$$

On obtient ainsi un polynôme d'ordre 2 ;

– cas $n = 3$. On applique le raisonnement précédent pour quatre points de contrôle. On obtient :

$$\begin{aligned}\mathbf{f}(t) &= (1-t)\mathbf{p}_{012} + t\mathbf{p}_{123} \\ &= (1-t)\underbrace{((1-t)\mathbf{p}_0 + t\mathbf{p}_{12})}_{\mathbf{p}_{012}} + t\underbrace{((1-t)\mathbf{p}_{12} + t\mathbf{p}_{23})}_{\mathbf{p}_{123}} \\ &= (1-t)^3\mathbf{p}_0 + 3(1-t)^2t\mathbf{p}_1 + 3(1-t)t^2\mathbf{p}_2 + t^3\mathbf{p}_3\end{aligned}$$

– pour n quelconque, on obtient :

$$\mathbf{f}(t) = \sum_{i=0}^n \underbrace{\frac{n!}{i!(n-i)!} (1-t)^{n-i} t^i}_{b_{i,n}(t)} \mathbf{p}_i$$

Les polynômes $b_{i,n}(t)$, appelés *polynômes de Bernstein*, forment une base de l'espace des polynômes de degré n . Lorsque nous montons en degré (c'est-à-dire en nombre de points de contrôle), on voit apparaître des instabilités numériques et des oscillations. C'est le phénomène de *Runge*. Pour les courbes complexes, avec des centaines de points de contrôle, on préfère alors les B-splines qui correspondent à une concaténation de courbes de Bézier d'ordre limité. La figure 4.3 illustre une telle concaténation où, pour chaque groupe de trois points, nous pourrons calculer un polynôme de Bézier de degré 2.

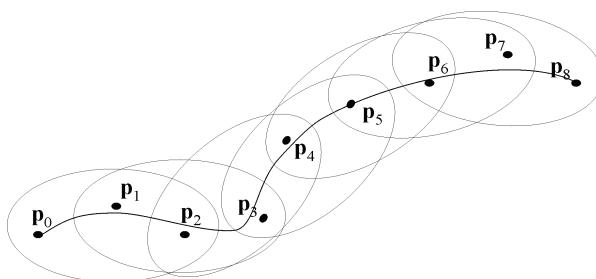


Figure 4.3. Illustration du principe des B-splines de degré 2

4.3. Diagramme de Voronoï

Considérons n points $\mathbf{p}_1, \dots, \mathbf{p}_n$. Contrairement aux sections précédentes, ici, les \mathbf{p}_i ne correspondent pas à des points de contrôle, mais à des points obstacles que l'on cherche à éviter. A chacun de ces points, on associe l'ensemble :

$$\mathbb{P}_i = \{ \mathbf{x} \in \mathbb{R}^d, \forall j, \|\mathbf{x} - \mathbf{p}_i\| \leq \|\mathbf{x} - \mathbf{p}_j\| \}$$

Pour tout i , cet ensemble est un polygone. La collection de ces \mathbb{P}_i s'appelle *diagramme de Voronoï*. La figure 4.4 représente un ensemble de points avec le diagramme de Voronoï associé. Si un environnement est formé d'obstacles, le robot devra tenter de planifier une trajectoire qui restera sur les frontières des \mathbb{P}_i .

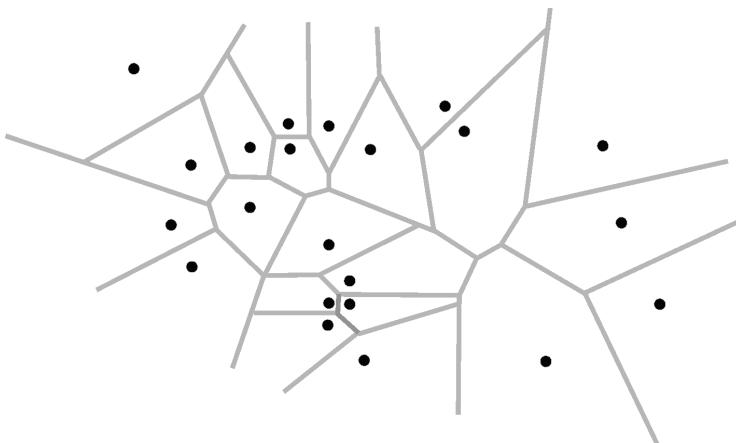


Figure 4.4. Diagramme de Voronoï

4.3.1. Triangulation de Delaunay

A partir de n points de l'espace, on peut utiliser les diagrammes de Voronoï pour effectuer une triangulation de l'espace. La triangulation correspondante, appelée *triangulation de Delaunay* permet de maximiser la quantité d'angles aigus et ainsi d'éviter les triangles allongés. Elle est obtenue en reliant par une arête les points voisins des régions correspondantes dans le diagramme de Voronoï. Dans une triangulation de Delaunay aucun triangle ne contient un autre point à l'intérieur de son cercle circonscrit. La figure 4.5 représente la triangulation de Delaunay associée au diagramme de Voronoï de la figure 4.4. Une triangulation de Delaunay est souvent utilisée en robotique pour représenter un espace comme par exemple la zone déjà explorée, la zone interdite, les lacs, etc. On associe souvent à chaque triangle une couleur suivant la caractéristique de l'espace auquel appartient le triangle (eau, terre, route, etc.).

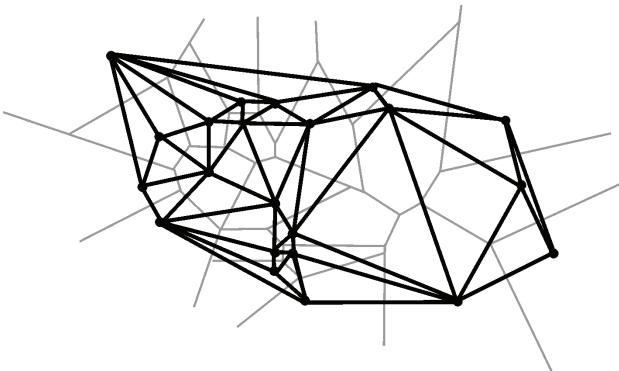


Figure 4.5. Triangulation de Delaunay

4.4. Méthode des champs de potentiels artificiels

Un robot mobile a besoin de se déplacer dans un environnement encombré par des obstacles mobiles ou non. La méthode des champs de potentiels artificiels [LAT 91] consiste à imaginer que le robot peut se comporter comme une particule électrique qui peut être attirée ou repoussée par des objets suivant le signe de leur charge. Il s'agit d'une approche réactive pour le guidage dans laquelle, la trajectoire n'est pas planifiée à l'avance. Alors qu'en physique nous avons la relation :

$$\mathbf{f} = -\text{grad}V(\mathbf{p})$$

où \mathbf{p} est la position d'une particule ponctuelle dans l'espace, V est le potentiel et \mathbf{f} la force engendrée sur la particule, en robotique mobile, nous aurons la même relation, mais \mathbf{p} sera la position du centre de robot, V sera un potentiel imaginé par le robot et \mathbf{f} sera le vecteur vitesse à suivre. Les champs de potentiel nous serviront à exprimer un comportement désiré pour un robot. Les obstacles seront représentés par des potentiels exerçant une force répulsive sur le robot et l'objectif à suivre devra exercer une force attractive. Dans une situation à plusieurs robots devant rester groupés tout en évitant les collisions, nous pourrons utiliser un potentiel répulsif en champ proche et attractif en champ lointain. D'une façon plus générale, les champs de vecteurs utilisés pourront ne pas dériver d'un potentiel, comme c'est le cas si l'on souhaite un comportement cyclique du robot. Le tableau qui suit donne quelques types de potentiels pouvant être utilisés.

Potentiel	$V(\mathbf{p})$	$-\text{grad}(V(\mathbf{p}))$
attractif conique	$\ \mathbf{p} - \hat{\mathbf{p}}\ $	$-\frac{\mathbf{p} - \hat{\mathbf{p}}}{\ \mathbf{p} - \hat{\mathbf{p}}\ }$
attractif quadratique	$\ \mathbf{p} - \hat{\mathbf{p}}\ ^2$	$-2(\mathbf{p} - \hat{\mathbf{p}})$
plan ou ligne attractive	$(\mathbf{p} - \hat{\mathbf{p}})^T \cdot \hat{\mathbf{n}} \hat{\mathbf{n}}^T \cdot (\mathbf{p} - \hat{\mathbf{p}})$	$-2 \hat{\mathbf{n}} \hat{\mathbf{n}}^T (\mathbf{p} - \hat{\mathbf{p}})$
répulsif	$\frac{1}{\ \mathbf{p} - \hat{\mathbf{q}}\ }$	$\frac{(\mathbf{p} - \hat{\mathbf{q}})}{\ \mathbf{p} - \hat{\mathbf{q}}\ ^3}$
uniforme	$-\hat{\mathbf{v}}^T \cdot \mathbf{p}$	$\hat{\mathbf{v}}$

Dans ce tableau, \hat{p} représente un point attractif, \hat{q} un point répulsif et \hat{v} une vitesse désirée pour le robot. Dans le cas du plan attractif, \hat{p} est un point du plan et \hat{n} est un vecteur orthonormal au plan. En additionnant plusieurs potentiels, on parvient à demander au robot (censé suivre la direction qui tend à faire diminuer le potentiel), à atteindre ses objectifs tout en s'éloignant des obstacles. La figure 4.6 représente trois champs de vecteurs dérivant de potentiels artificiels. Celui de gauche correspond à un champ uniforme, celui du centre à un potentiel répulsif qui s'additionne à un champ uniforme et celui de droite à la somme d'un potentiel attractif et d'un potentiel répulsif.

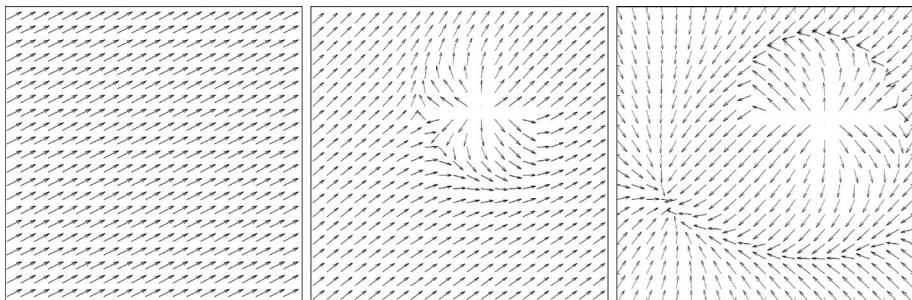


Figure 4.6. Champs de potentiels artificiels

4.5. Exercices

Exercice 4.1. Poursuite sur une sphère

On considère un robot \mathcal{R} se déplaçant à la surface d'une sphère pouvant être assimilée à la terre de rayon $\rho = 30$ m. Ce robot est repéré par sa longitude ℓ_x sa latitude ℓ_y et son cap ψ , relativement à l'est. Dans un repère local, les équations d'état du robot sont du type :

$$\begin{cases} \dot{x} = \cos \psi \\ \dot{y} = \sin \psi \\ \dot{\psi} = u \end{cases}$$

- 1) Donner les équations d'état dans le cas où le vecteur d'état est (ℓ_x, ℓ_y, ψ) .
- 2) Simuler avec graphisme 3D sous MATLAB ce système en évolution.
- 3) Un deuxième robot \mathcal{R}_a , décrit par les mêmes équations, se déplace de manière aléatoire sur la sphère (voir figure 4.7), proposer une loi de commande qui permette au robot \mathcal{R} de rejoindre le robot \mathcal{R}_a .

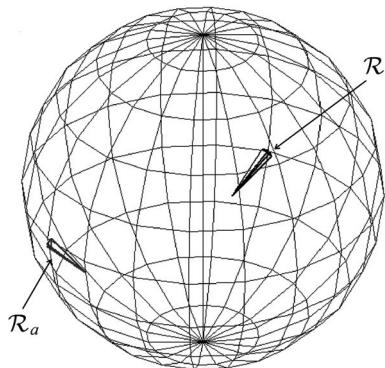


Figure 4.7. Sur la sphère, le robot \mathcal{R} poursuit le robot \mathcal{R}_a

Exercice 4.2. Planification d'une trajectoire

On considère une scène avec deux triangles comme sur la figure 4.8. Un robot décris par les équations d'état :

$$\begin{cases} \dot{x} = v \cos \theta \\ \dot{y} = v \sin \theta \\ \dot{\theta} = u_1 \\ \dot{v} = u_2 \end{cases}$$

avec pour état initial $(x, y, \theta, v) = (0, 0, 0, 1)$. Ce robot doit rejoindre le point de coordonnées $(8, 8)$.

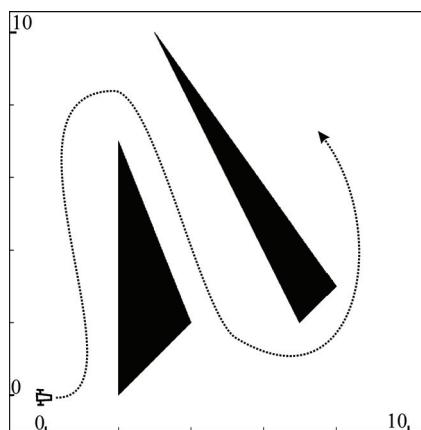


Figure 4.8. Le robot doit suivre une trajectoire sans percuter les obstacles

1) Sous MATLAB, trouver les points de contrôle pour un polynôme de Bézier permettant de relier la position initiale à la position désirée comme sur la figure.

2) En utilisant une technique par bouclage linéarisant, en déduire la loi de commande qui permet d'atteindre l'objectif en 50 sec.

Exercice 4.3. Tracé d'un diagramme de Voronoï

On considère les dix points de la figure 4.9. Tracer sur feuille le diagramme de Voronoï associé ainsi que la triangulation de Delaunay correspondante.

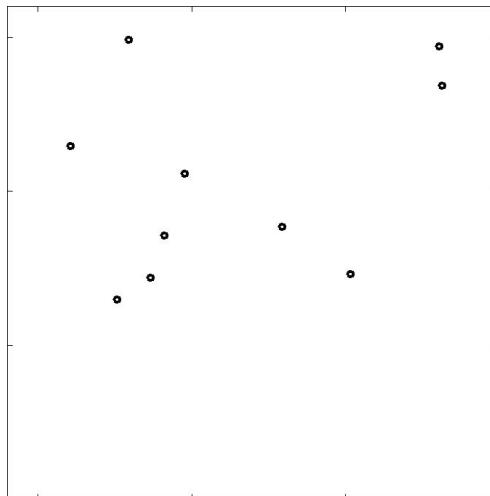


Figure 4.9. Dix points pour lesquels on souhaite construire un diagramme de Voronoï

Exercice 4.4. Calcul d'un diagramme de Voronoï

1) Montrer que si \mathbf{x} et \mathbf{y} sont deux vecteurs de \mathbb{R}^n , nous avons les équations dites de polarisation :

$$\{ \|\mathbf{x} - \mathbf{y}\|^2 = \|\mathbf{x}\|^2 + \|\mathbf{y}\|^2 - 2 \langle \mathbf{x}, \mathbf{y} \rangle$$

Pour cela, on développera l'expression du produit scalaire $\langle \mathbf{x} - \mathbf{y}, \mathbf{x} - \mathbf{y} \rangle$.

2) Soient $n + 1$ points $\mathbf{a}^1, \dots, \mathbf{a}^{n+1}$ dont la sphère circonscrite est notée \mathcal{S} . En s'aidant de la question précédente, donner une expression en fonction des \mathbf{a}^i du centre \mathbf{c} de \mathcal{S} et de son rayon r .

3) Soient trois points du plan $\mathbf{a}^1, \mathbf{a}^2, \mathbf{a}^3$. Quelles sont les conditions à vérifier pour que \mathbf{m} soit dans le cercle circonscrit au triangle $(\mathbf{a}^1, \mathbf{a}^2, \mathbf{a}^3)$?

4) Soient m points du plan $\mathbf{p}^1, \dots, \mathbf{p}^m$, une triangulation de Delaunay est une partition de l'espace en triangle $\mathcal{T}(k) = (\mathbf{a}^1(k), \mathbf{a}^2(k), \mathbf{a}^3(k))$, dont les sommets sont pris parmi les \mathbf{p}^i , telle que chaque cercle circonscrit $\mathcal{C}(k)$ à ce triangle $\mathcal{T}(k)$ ne contienne aucun des \mathbf{p}^i . Faire un programme MATLAB qui tire aléatoirement $m = 10$ points du plan et qui trace une triangulation de Delaunay. Quelle est la complexité de l'algorithme ?

5) A partir de la triangulation établie à la question précédente, construire le diagramme de Voronoï associé aux points $\mathbf{a}^1, \dots, \mathbf{a}^{n+1}$.

Exercice 4.5. Commande en cap d'une voiture de Dubins

Les résultats de cet exercice seront utilisés dans l'exercice 4.6 pour le calcul des chemins de Dubins. Une voiture de Dubins est décrite par les équations d'état :

$$\begin{cases} \dot{x} = \cos \theta \\ \dot{y} = \sin \theta \\ \dot{\theta} = u \end{cases}$$

θ est le cap du robot, (x, y) les coordonnées de son centre. Ce robot doit s'aligner avec un cap consigne $\bar{\theta}$.

1) On suppose que l'entrée $u \in [-1, 1]$. Donner une expression analytique de l'angle erreur $\delta \in [-\pi, \pi]$ en fonction de θ et $\bar{\theta}$ qui indique de quel angle le robot doit tourner pour rejoindre sa consigne le plus rapidement possible. Il faut prendre en compte que l'expression de δ doit être périodique relativement à θ et $\bar{\theta}$. En effet, des angles de $-\pi, \pi$ ou 3π doivent être considérés comme équivalents. Donner la loi de commande associée et simuler la loi de commande sous MATLAB.

2) Faire de même sachant que le robot n'a le droit de tourner que sur sa gauche (dans le sens direct trigonométrique), c'est-à-dire $u \in [0, 1]$. Notons que maintenant, $\delta \in [0, 2\pi]$.

3) Traiter maintenant le cas où le robot n'a le droit de tourner que sur sa droite, c'est-à-dire $u \in [-1, 0]$.

Exercice 4.6. Chemins de Dubins

Comme à l'exercice précédent, on considère un robot se déplaçant sur un plan et décrit par :

$$\begin{cases} \dot{x} = \cos \theta \\ \dot{y} = \sin \theta \\ \dot{\theta} = u \end{cases}$$

où θ est le cap du robot et (x, y) les coordonnées du centre. Son vecteur d'état est donné par $\mathbf{x} = (x, y, \theta)$ et son entrée u est contrainte à rester dans l'intervalle $[-u_{\max}, u_{\max}]$. Il s'agit ici de la voiture de Dubins [DUB 57] correspondant à un véhicule mobile non holonome le plus simple qui soit. Malgré sa simplicité, il illustre de nombreuses difficultés pouvant apparaître dans le contexte des robots non holonomes.

1) Calculer le rayon de courbure maximal r pouvant être effectué par robot.

2) Dubins a démontré que pour aller d'une configuration $\mathbf{a} = (x_a, y_a, \theta_a)$ à une configuration $\mathbf{b} = (x_b, y_b, \theta_b)$ pas trop rapprochées (c'est-à-dire séparées d'une distance supérieure à $4r$) la stratégie en temps minimal consiste toujours à (1) tourner à fond dans un sens (c'est-à-dire $u = \pm u_{\max}$) ; (2) aller tout droit ; (3) puis à nouveau tourner à fond. Le chemin correspondant à une telle manœuvre est appelé *chemin de Dubins*, est donc composé d'un arc de commencement, un segment puis d'un arc de terminaison. Il y a quatre possibilités de faire un chemin de Dubins : LSL, LSR, RSL, RSR, où L signifie gauche (*left*), R signifie droite (*right*) et S signifie tout droit (*straight*). Donner une configuration pour \mathbf{a} et \mathbf{b} pour laquelle aucun des quatre chemins ne correspond à une stratégie optimale (donc forcément avec \mathbf{a} et \mathbf{b} sont assez proches). On choisira une situation où la stratégie optimale est de type RLR.

3) Dans le cas d'une stratégie RSL, comme illustrée sur la figure 4.10, calculer en fonction de \mathbf{a} et \mathbf{b} la longueur L du chemin de Dubins.

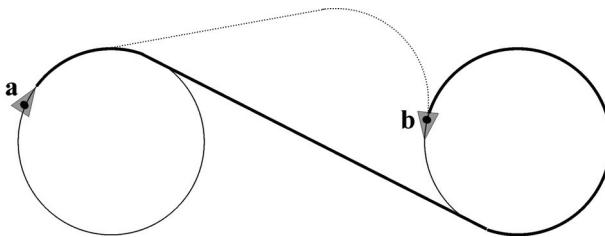


Figure 4.10. En gras, chemin de Dubins de type RSL (Right-Straight-Left) menant de \mathbf{a} à \mathbf{b} ; en pointillés, un chemin de Dubins du type RLR

4) Dans le cas d'une stratégie LSL, calculer en fonction de \mathbf{a} et \mathbf{b} la longueur L du chemin de Dubins.

5) En utilisant la symétrie axiale du problème, en déduire le calcul de L dans le cas des stratégies RSR et LSR. Proposer alors une fonction MATLAB qui calcule L dans toutes les situations. On fera pour cela intervenir les deux booléens ε_a ε_b qui

sont égaux à 1 si l'arc correspondant est dans le sens direct (c'est-à-dire à gauche) et -1 dans le cas contraire.

6) Déduire des questions précédentes un programme MATLAB qui calcule le chemin de longueur minimale pour une voiture de Dubins.

Exercice 4.7. Potentiels artificiels

Un robot situé en $\mathbf{p} = (x, y)$ doit rejoindre une cible de mouvement inconnu dont on connaît la position $\hat{\mathbf{p}}$ et la vitesse $\hat{\mathbf{v}}$ à l'instant présent. Ce couple $(\hat{\mathbf{p}}, \hat{\mathbf{v}})$ peut correspondre par exemple à une consigne donnée par un opérateur humain. Un obstacle fixe situé à la position $\hat{\mathbf{q}}$ est à éviter. On modélise le comportement souhaité pour notre robot par le potentiel :

$$V(\mathbf{p}) = -\hat{\mathbf{v}}^T \cdot \mathbf{p} + \|\mathbf{p} - \hat{\mathbf{p}}\|^2 + \frac{1}{\|\mathbf{p} - \hat{\mathbf{q}}\|}$$

où le potentiel $-\hat{\mathbf{v}}^T \cdot \mathbf{p}$ représente la consigne de vitesse, le potentiel $\|\mathbf{p} - \hat{\mathbf{p}}\|^2$ rend attractif la position cible $\hat{\mathbf{p}}$ et le potentiel $\frac{1}{\|\mathbf{p} - \hat{\mathbf{q}}\|}$ rend répulsif l'obstacle $\hat{\mathbf{q}}$.

1) Calculer le gradient du potentiel $V(\mathbf{p})$ et en déduire le vecteur vitesse consigne $\mathbf{w}(\mathbf{p}, t)$ à appliquer à notre robot de façon à ce qu'il réponde correctement à ce potentiel.

2) On suppose que notre robot obéit aux équations d'état suivantes :

$$\begin{cases} \dot{x} = v \cos \theta \\ \dot{y} = v \sin \theta \\ \dot{v} = u_1 \\ \dot{\theta} = u_2 \end{cases}$$

Proposer la loi de commande correspondant au champ de potentiel désiré. On suivra le principe rappelé sur la figure 4.11. Tout d'abord, on décompose le robot (en gris sur la figure) en une chaîne de deux blocs. Le premier bloc forme les vitesses à partir des actionneurs et le deuxième construit le vecteur position $\mathbf{p} = (x, y)$. Ensuite, on calcule l'inverse à gauche du premier bloc afin de se retrouver avec un système du type :

$$\begin{cases} \dot{x} = \bar{v} \cos \bar{\theta} \\ \dot{y} = \bar{v} \sin \bar{\theta} \end{cases}$$

Cette inversion approximative se fera par une simple commande proportionnelle. Puis, on génère la nouvelle entrée $(\bar{v}, \bar{\theta})$ par le potentiel à satisfaire. On illustrera le comportement de ce robot sous MATLAB avec une cible $\hat{\mathbf{p}} = (t, t)$ et un obstacle immobile placé en $\hat{\mathbf{q}} = (4, 5)$.

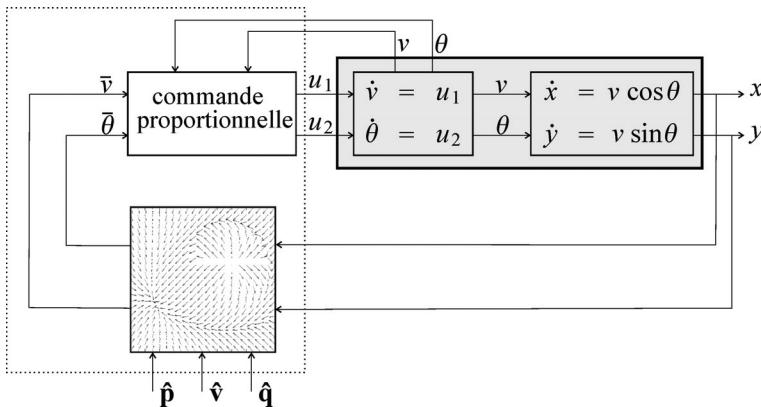


Figure 4.11. Régulateur (en pointillés) obtenu par la méthode des potentiels

3) On cherche maintenant à suivre la cible \hat{p} avec un obstacle mobile en \hat{q} avec :

$$\hat{p} = \begin{pmatrix} \cos \frac{t}{10} \\ 2 \sin \frac{t}{10} \end{pmatrix} \text{ et } \hat{q} = \begin{pmatrix} 2 \cos \frac{t}{5} \\ 2 \sin \frac{t}{5} \end{pmatrix}$$

Adapter les paramètres du potentiel pour arriver à suivre la cible sans percuter l'obstacle. Illustrer le comportement du robot ainsi régulé sous MATLAB.

4.6. Corrections

Correction de l'exercice 4.1 (poursuite sur une sphère)

1) On a, d'après [4.4], l'approximation à l'ordre 1 suivante :

$$\begin{pmatrix} dx \\ dy \end{pmatrix} \simeq \begin{pmatrix} \rho \cos \ell_y d\ell_x \\ \rho d\ell_y \end{pmatrix}$$

Donc, comme $\dot{x} = \cos \psi$ et $\dot{y} = \sin \psi$, on a :

$$\begin{cases} \dot{\ell}_x = \frac{\cos \psi}{\rho \cos \ell_y} \\ \dot{\ell}_y = \frac{\sin \psi}{\rho} \\ \dot{\theta} = u \end{cases}$$

2) Sous MATLAB, on trace les méridiens et les parallèles pour représenter la sphère, comme suit :

```
function draw_earth()
M=[]; a=pi/10;
Ly=-pi/2:a:pi/2; Lx=0:a:2*pi;
for ly=Ly, for lx=Lx, M=[M,T(lx,ly,rho)]; end; end;
for lx=Lx, for ly=Ly, M=[M,T(lx,ly,rho)]; end; end;
plot3(M(1,:),M(2,:),M(3,:));
end
```

La fonction de transformation T utilisée précédemment est donnée par :

```
function p=T(lx,ly,rho)
p=rho*[cos(ly)*cos(lx);cos(ly)*sin(lx);sin(ly)];
end
```

Pour tracer, le robot, on crée tout d'abord un motif en 3D, on lui fait ensuite subir un changement de repère qui l'amène dans la zone demandée (voir l'équation [4.2]) et enfin une transformation d'Euler pour lui donner la bonne orientation locale. Notons que dans notre cas, l'assiette θ et la gîte φ sont nulles, car le robot se déplace à l'horizontale. On obtient les instructions suivantes :

```
R1=[-sin(lx),-sin(ly)*cos(lx),cos(ly)*cos(lx);
cos(lx),-sin(ly)*sin(lx),cos(ly)*sin(lx);
0,cos(ly),sin(ly)];
E=eulermat(0,0,psi);
R=[R1*E,T(lx,ly,rho);0,0,0,1];
```

3) Le robot \mathcal{R} situé à en $\mathbf{m} : (\ell_x, \ell_y)$ avec un cap de ψ relativement à l'est doit rejoindre le robot \mathcal{R}_a situé au point $\mathbf{a} : (\ell_x^a, \ell_y^a)$. Si \mathbf{a} est dans un voisinage proche de \mathbf{m} (moins de 100 km), on peut calculer les coordonnées de \mathbf{a} dans une carte centrée sur \mathbf{m} . On obtient, d'après la formule [4.4] :

$$\begin{cases} \tilde{x}^a = \rho \cos \ell_y^a \cdot (\ell_x^a - \ell_x) \\ \tilde{y}^a = \rho (\ell_y^a - \ell_y) \end{cases}$$

Pour savoir si le robot \mathcal{R}_a est à gauche où à droite de \mathcal{R} , on regarde le sinus de l'angle entre les deux vecteurs $(\cos \psi, \sin \psi)$ et $(\tilde{x}^a, \tilde{y}^a)$. On prendra donc la commande proportionnelle :

$$u = \det \begin{pmatrix} \cos \psi \cos \ell_y \cdot (\ell_x^a - \ell_x) \\ \sin \psi \ell_y^a - \ell_y \end{pmatrix}$$

Cette loi de commande qui se base sur une représentation locale de la position de la cible \mathcal{R}_a fonctionne correctement même lorsque \mathcal{R}_a est éloigné de \mathcal{R} . Le programme donné dans le fichier `earth.m` montre le bon comportement de cette loi de commande.

Correction de l'exercice 4.2 (planification d'une trajectoire)

1) A partir des points de contrôle, la fonction consigne associée est donnée par :

```
function w=setpoint(t)
w=0 ;
for i=0 :n, w=w+b(i,n,t)*P( :,i+1) ; end ;
end
```

Les coefficients sont donnés par la fonction :

```
function y=b(i,n,t)
y=prod(1 :n)/(prod(1 :i)*(prod(1 :n-i)))*(1-t)^(n-i)*t^i ;
end
```

On joue aisément sur les points de contrôle pour obtenir une trajectoire qui convient. On prend par exemple les points de contrôle \mathbf{p}_i donnés dans la matrice :

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 2 & 3 & 4 & 5 & 5 & 7 & 8 & 10 & 9 & 8 \\ 1 & 4 & 7 & 9 & 10 & 8 & 6 & 4 & 1 & 0 & 0 & 1 & 4 & 8 \end{pmatrix}$$

et qui sont représentés sur la figure 4.12 à gauche.

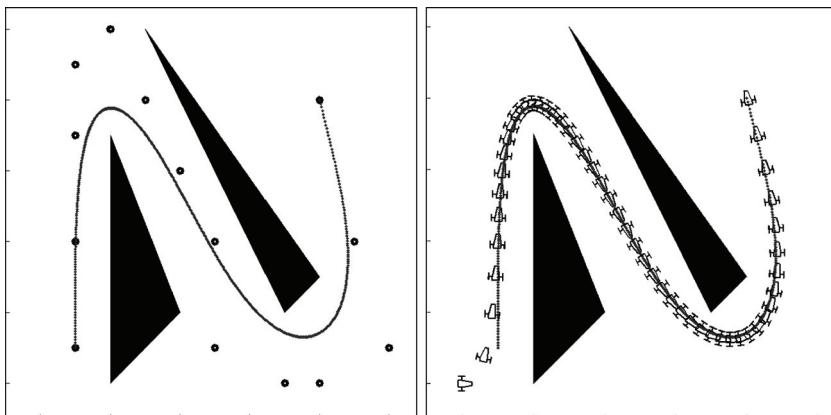


Figure 4.12. A gauche : points de contrôle avec le polynôme de Bézier associé ; à droite : trajectoire pour le robot qui suit le polynôme en évitant les obstacles

2) Nous pouvons prendre le régulateur de la section 2.4.1 page 61. Pour atteindre l'objectif en $t_{\max} = 50$ sec, nous devons prendre la consigne :

$$\mathbf{w}(t) = \sum_{i=0}^n b_{i,n}\left(\frac{t}{50}\right) \mathbf{p}_i$$

Pour appliquer notre régulateur, nous avons besoin de la dérivée $\dot{\mathbf{w}}(t)$ de la consigne $\mathbf{w}(t)$. Elle est donnée par :

$$\dot{\mathbf{w}}(t) = \frac{1}{50} \sum_{i=0}^n \dot{b}_{i,n}\left(\frac{t}{50}\right) \mathbf{p}_i$$

avec :

$$\dot{b}_{i,n}(t) = \frac{n!}{i!(n-i)!} \left(i(1-t)^{n-i} t^{i-1} - (n-i)(1-t)^{n-i-1} t^i \right)$$

Nous devons faire attention de bien prendre en compte le cas particulier $i = 0$ pour lequel $\dot{b}_{i,n}(t) = -n(1-t)^{n-1}$ et le cas $i = n$ pour lequel $\dot{b}_{n,n}(t) = nt^{n-1}$. La trajectoire résultante est représentée sur la figure 4.12 à droite. Le programme MATLAB illustrant cette loi de commande est donné dans le fichier `bezier.m`.

Correction de l'exercice 4.3 (tracé d'un diagramme de Voronoï)

Sur la figure 4.13, la triangulation de Delaunay est représentée par les traits en gras. Le diagramme de Voronoï est représenté par les traits fins.

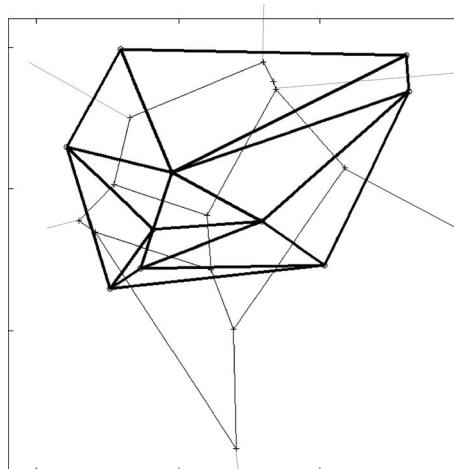


Figure 4.13. Diagramme de Voronoï

La figure 4.14 montre que le cercle circonscrit à chaque triangle n'enferme aucun autre point.

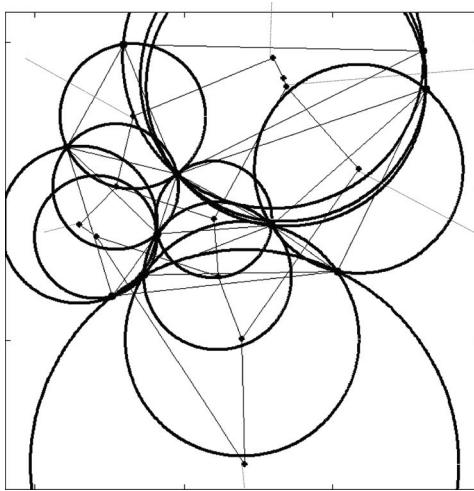


Figure 4.14. Cercles circonscrits aux triangles de Delaunay

Correction de l'exercice 4.4 (calcul d'un diagramme de Voronoï)

1) Rappelons que le produit scalaire est une forme bilinéaire et donc nous avons :

$$\|\mathbf{x} - \mathbf{y}\|^2 = \langle \mathbf{x} - \mathbf{y}, \mathbf{x} - \mathbf{y} \rangle = \langle \mathbf{x}, \mathbf{x} \rangle + \langle \mathbf{y}, \mathbf{y} \rangle - \langle \mathbf{x}, \mathbf{y} \rangle - \langle \mathbf{y}, \mathbf{x} \rangle = \|\mathbf{x}\|^2 + \|\mathbf{y}\|^2 - 2 \langle \mathbf{x}, \mathbf{y} \rangle$$

2) Soit \mathbf{c} le centre de la sphère \mathcal{S} . Pour $i \in \{1, \dots, n+1\}$, nous avons :

$$\|\mathbf{a}^i - \mathbf{c}\|^2 = r^2$$

Soit, en utilisant l'équation de polarisation :

$$\|\mathbf{a}^i\|^2 + \|\mathbf{c}\|^2 - 2 \langle \mathbf{a}^i, \mathbf{c} \rangle - r^2 = 0$$

Bien évidemment, nous avons $n+1$ inconnues qui sont r et \mathbf{c} . Afin d'obtenir un système linéaire en ces inconnues, posons :

$$c_{n+1} = \|\mathbf{c}\|^2 - r^2$$

Ainsi :

$$2 \langle \mathbf{a}^i, \mathbf{c} \rangle - c_{n+1} = \|\mathbf{a}^i\|^2$$

Nous avons donc :

$$\begin{pmatrix} 2a_1^1 & \dots & 2a_n^1 & -1 \\ \vdots & & \vdots & \vdots \\ 2a_1^{n+1} & \dots & 2a_n^{n+1} & -1 \end{pmatrix} \begin{pmatrix} c_1 \\ \vdots \\ c_n \\ c_{n+1} \end{pmatrix} = \begin{pmatrix} \|\mathbf{a}^1\|^2 \\ \vdots \\ \|\mathbf{a}^{n+1}\|^2 \end{pmatrix}$$

Par résolution du système linéaire, on obtient le centre du cercle circonscrit $\mathbf{c} = (c_1, \dots, c_n)$ ainsi que son centre $r = \sqrt{\|\mathbf{c}\|^2 - c_{n+1}}$.

3) Le point \mathbf{m} est dans le cercle circonscrit au triangle $(\mathbf{a}^1, \mathbf{a}^2, \mathbf{a}^3)$ si :

$$\|\mathbf{m} - \mathbf{c}\| \leq r$$

avec :

$$\begin{pmatrix} c_1 \\ c_2 \\ c_3 \end{pmatrix} = \begin{pmatrix} 2a_1^1 & 2a_2^1 & -1 \\ 2a_1^2 & 2a_2^2 & -1 \\ 2a_1^3 & 2a_2^3 & -1 \end{pmatrix}^{-1} \begin{pmatrix} \|\mathbf{a}^1\|^2 \\ \|\mathbf{a}^2\|^2 \\ \|\mathbf{a}^3\|^2 \end{pmatrix}$$

$$\mathbf{c} = \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} \text{ et } r = \sqrt{\|\mathbf{c}\|^2 - c_3}$$

4) Le programme est le suivant :

```
m=10;
P = 10*rand(2,m); % choix des points
C=[]; % liste des centres des cercles
K=[]; % liste des triangles
for i=1 :m-2
for j=i+1 :1 :m-1
for k=j+1 :1 :m
A=P( :, [i,j,k]); c=[2*A',[ -1;-1;-1]]\sum(A.^2,1)';
r=sqrt(norm(c(1 :2))^2-c(3));
ok=true;
for q=1 :m
ok=ok&&(q==i||q==j||q==k||norm(P( :,q)-c(1 :2))>r);
end
if ok,
C=[C,c]; % on mémorise les centres
K=[K,[i ;j ;k]]; % on mémorise les triangles
end, end, end;
```

La complexité de l'algorithme est en m^4 où m est le nombre de points considérés. Il existe des algorithmes en $m \log m$ comme l'*algorithme de sweep line* ou *algorithme de Fortune*.

4) Le programme suivant trace le diagramme de Voronoï. Le principe consiste à relier les centres des cercles circoncrits des triangles voisins :

```

for i=1 :size(K,2), for j=1 :size(K,2), % on prend tous les
triangles deux à deux
if sum(ismember(K( :,i),K( :,j)))==2, % s'ils sont voisins
plot(C(1,[i j]),C(2,[i j])); % on les relie
end, end, end

```

Correction de l'exercice 4.5 (commande en cap d'une voiture de Dubins)

1) On rappelle (voir formule [3.2]) l'expression de la fonction à dents de scie :

$$\text{sawtooth}(\theta) = \text{mod}(\theta + \pi, 2\pi) - \pi$$

Posons $\tilde{\theta} = \bar{\theta} - \theta$. Pour avoir $\delta \in [-\pi, \pi]$, on prendra $\delta = \text{sawtooth}(\tilde{\theta})$. Notons que si $\tilde{\theta} \in [-\pi, \pi]$ alors $\delta = \tilde{\theta}$. Pour avoir $u \in [-1, 1]$, on peut prendre une loi de commande proportionnelle du type :

$$u = \frac{\delta}{\pi} = \frac{1}{\pi} \text{sawtooth}(\tilde{\theta})$$

2) On crée sous MATLAB la fonction à dent de scie directionnelle :

```

function y = sawtooth(x,d)
y=d*pi+mod(x+pi-d*pi,2*pi)-pi;
end

```

Cette fonction retourne l'écart à gauche ($d = 1$), à droite ($d = -1$) ou au plus court ($d = 0$) comme illustré par la figure 4.15. Le cas au plus court correspond à la fonction à dents de scie usuelle. Pour nous, $d = 1$ et donc $\delta = \text{sawtooth}(\tilde{\theta}, 1)$. Pour avoir $u \in [0, 1]$, on doit prendre :

$$u = \frac{\delta}{2\pi} = \frac{\text{sawtooth}(\tilde{\theta}, 1)}{2\pi}$$

3) Maintenant, $d = -1$. On prend $\delta = \text{sawtooth}(\tilde{\theta}, -1) - \pi$. Pour avoir $u \in [-1, 0]$, on prend :

$$u = \frac{\delta}{2\pi} = \frac{\text{sawtooth}(\tilde{\theta}, -1)}{2\pi}$$

Le programme associé à cet exercice est donné dans `dubins.m`.

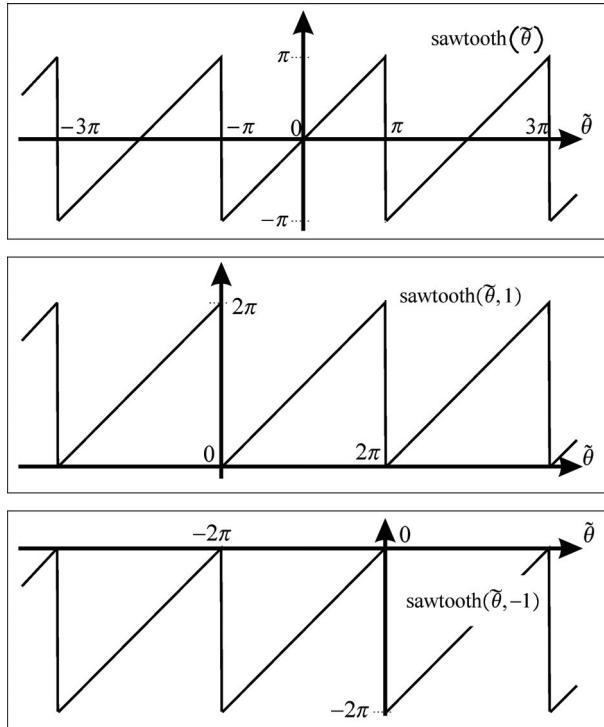


Figure 4.15. Fonctions dents de scie pour effectuer une différence angulaire et se retrouver dans l'intervalle souhaité

Correction de l'exercice 4.6 (chemins de Dubins)

1) Pour u constant, nous avons $\theta = ut + \theta_0$. Et donc $\dot{x}(t) = \cos(ut + \theta_0)$ et $\dot{y}(t) = \sin(ut + \theta_0)$. Soit, après intégration : $x(t) = x(0) + \frac{1}{u} \sin(ut + \theta_0)$ et $y(t) = y(0) - \frac{1}{u} \cos(ut + \theta_0)$. Le rayon de courbure associé est $r = \frac{1}{u}$. Puisque $u \in [-u_{\max}, u_{\max}]$, nous avons $r \in \frac{1}{[-u_{\max}, u_{\max}]} = [-\infty, -\frac{1}{u_{\max}}] \cup [\frac{1}{u_{\max}}, \infty]$.

2) On peut prendre **b** à une distance de $3.9r$ à droite de **a**. La stratégie optimale est de type RLR.

3) Comme illustré par la figure 4.17, les vecteurs $\mathbf{d}_a - \mathbf{c}_a$ et sont orthogonaux. Ainsi, en utilisant le théorème de Pythagore, la demi-longueur ℓ du segment du chemin de Dubins est donnée par :

$$\ell = \sqrt{\left(\frac{\|\mathbf{c}_b - \mathbf{c}_a\|}{2}\right)^2 - r^2}$$

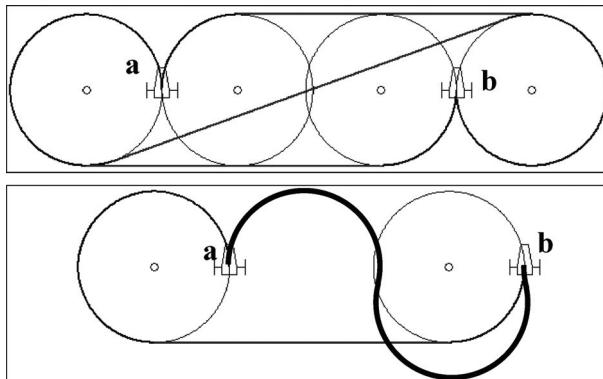


Figure 4.16. Haut : les quatre chemins de Dubins superposés de type RSR, LSR, RSL et LSL. Bas : le meilleur des quatre chemins avec en plus la stratégie RLR (en gras) qui se trouve être la meilleure de toute.

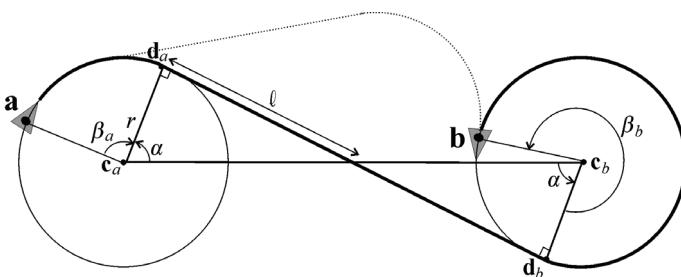


Figure 4.17. Le calcul de la longueur du chemin de Dubins RSL repose sur le fait que $d_a - c_a \perp d_b - d_a \perp d_b - c_b$

Si cette racine carrée n'est pas définie (c'est-à-dire si $\|\mathbf{c}_b - \mathbf{c}_a\| \leq 2r$), alors ce chemin n'est pas valide et ne pourra pas correspondre à un chemin de Dubins. En fait, c'est dans cette situation qu'une stratégie de type RLR, pourrait être optimale.

Dans le cas où $\|\mathbf{c}_b - \mathbf{c}_a\| \geq 2r$, le calcul de la longueur du chemin se fait comme suit :

$$\mathbf{c}_a = \mathbf{a} - r \begin{pmatrix} -\sin \theta_a \\ \cos \theta_a \end{pmatrix}; \mathbf{c}_b = \mathbf{b} + r \begin{pmatrix} -\sin \theta_b \\ \cos \theta_b \end{pmatrix}$$

$$\ell = \sqrt{\frac{1}{4} \|\mathbf{c}_b - \mathbf{c}_a\|^2 - r^2}. \text{ Si } \ell \text{ n'est pas réel, retourner } L = \infty \text{ (échec)}$$

$$\alpha = \text{atan2}(\ell, r)$$

$$\mathbf{d}_a = \mathbf{c}_a + \frac{r}{\|\mathbf{c}_b - \mathbf{c}_a\|} \begin{pmatrix} \cos \alpha & -\sin \alpha \\ \sin \alpha & \cos \alpha \end{pmatrix} (\mathbf{c}_b - \mathbf{c}_a)$$

$$\mathbf{d}_b = \mathbf{c}_b + \mathbf{c}_a - \mathbf{d}_a$$

$$\beta_a = \text{sawtooth}(\text{angle}(\mathbf{a} - \mathbf{c}_a, \mathbf{d}_a - \mathbf{c}_a), -1)$$

$$\beta_b = \text{sawtooth}(\text{angle}(\mathbf{d}_b - \mathbf{c}_b, \mathbf{b} - \mathbf{c}_b), 1)$$

$$L = r |\beta_a| + r |\beta_b| + 2\ell$$

La fonction *angle* rangée dans *angle.m*. Il s'agit de la fonction récursive suivante :

```
function theta = angle(u,v)
if (exist('v')==0), theta=atan2(u(2),u(1));
else theta=sawtooth(angle(v)-angle(u)); end;
end
```

4) Le calcul de la longueur du chemin se fait comme suit :

$$\mathbf{c}_a = \mathbf{a} + r \begin{pmatrix} -\sin \theta_a \\ \cos \theta_a \end{pmatrix}; \mathbf{c}_b = \mathbf{b} + r \begin{pmatrix} -\sin \theta_b \\ \cos \theta_b \end{pmatrix}$$

$$\ell = \frac{1}{2} \|\mathbf{c}_b - \mathbf{c}_a\|; \alpha = -\frac{\pi}{2}$$

$$\mathbf{d}_a = \mathbf{c}_a + \frac{r}{\|\mathbf{c}_b - \mathbf{c}_a\|} \begin{pmatrix} \cos \alpha & -\sin \alpha \\ \sin \alpha & \cos \alpha \end{pmatrix} (\mathbf{c}_b - \mathbf{c}_a)$$

$$\mathbf{d}_b = \mathbf{c}_b - \mathbf{c}_a + \mathbf{d}_a$$

$$\beta_a = \text{sawtooth}(\text{angle}(\mathbf{a} - \mathbf{c}_a, \mathbf{d}_a - \mathbf{c}_a), 1)$$

$$\beta_b = \text{sawtooth}(\text{angle}(\mathbf{d}_b - \mathbf{c}_b, \mathbf{b} - \mathbf{c}_b), 1)$$

$$L = r |\beta_a| + r |\beta_b| + 2\ell$$

5) La fonction MATLAB suivante calcule *L* dans toutes les situations :

```
function L = path(a,b,r,epsa,epsb)
ca=a(1 :2)+epsa*r*[-sin(a(3));cos(a(3))];
cb=b(1 :2)+epsb*r*[-sin(b(3));cos(b(3))];
if (epsa*epsb== -1),
ell2=0.25*norm(cb-ca)^2-r^2;
if ell2<0, L=inf; return; end;
ell=sqrt(ell2);
alpha=-epsa*atan2(ell,r);
else ell=0.5*norm(cb-ca); alpha=-epsa*pi/2;
end
```

```

R=[cos(alpha),-sin(alpha);sin(alpha),cos(alpha)];
da=ca+(r/norm(cb-ca))*R*(cb-ca);
db=cb+epsa*epsb*(da-ca);
betaa=sawtooth(angle(a(1:2)-ca,da-ca),epsa);
betab=sawtooth(angle(db-cb,b(1:2)-cb),epsb);
L=r*(abs(betaa)+abs(betab))+2*ell;
end

```

5) Le programme MATLAB qui suit calcule le chemin de Dubins à temps minimal :

```

L1=path(a,b,r,-1,-1); %RSR
L2=path(a,b,r,-1, 1); % RSL
L3=path(a,b,r, 1,-1); % LSR
L4=path(a,b,r, 1, 1); %LSL
L=min([L1,L2,L3,L4]);
if (L==L1) path(a,b,r,-1,-1); end;
if (L==L2) path(a,b,r,-1, 1); end;
if (L==L3) path(a,b,r, 1,-1); end;
if (L==L4) path(a,b,r, 1, 1); end;

```

Ce programme teste les quatre possibilités de chemin et prend le meilleur. Pour $r = 10$, $\mathbf{a} = (-20, 10, -3)$ et $\mathbf{b} = (20, -10, 2)$, on obtient les résultats de la figure 4.18. Pour $\mathbf{a} = (-3, 1, 0.9)$ et $\mathbf{b} = (2, -1, 1)$ on obtient ceux de la figure 4.19. Le programme est donné dans le fichier `dubinspath.m`.

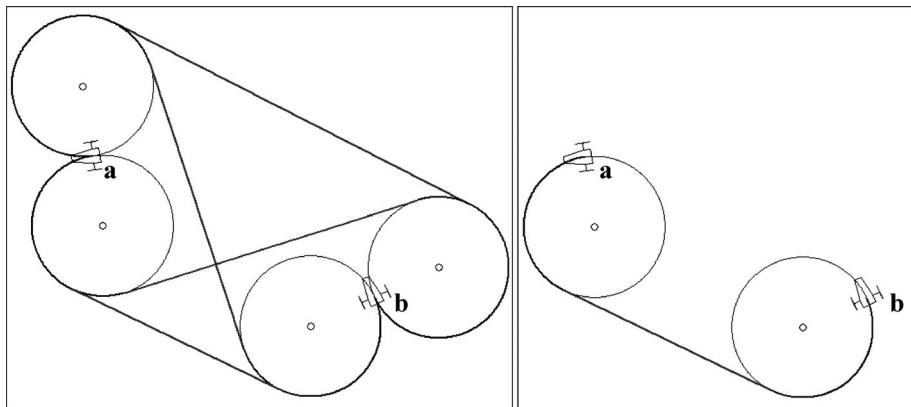


Figure 4.18. A gauche : les quatre chemins de Dubins possibles ;
à droite : le meilleur des quatre chemins

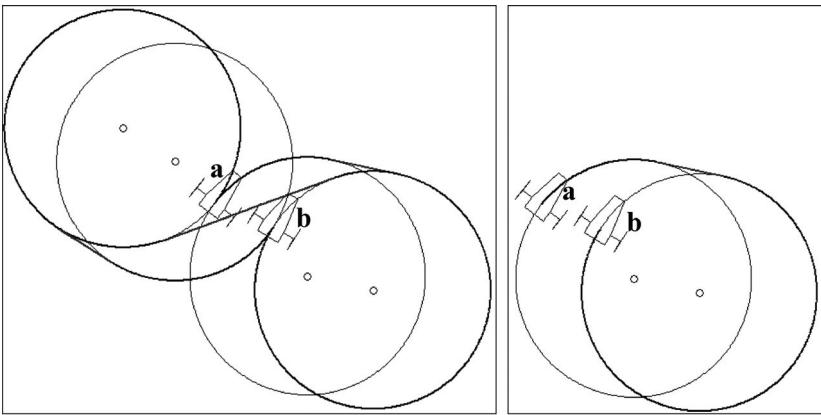


Figure 4.19. Une autre situation où le robot est obligé de faire une manœuvre assez complexe pour bouger très légèrement

Correction de l'exercice 4.7 (potentiels artificiels)

1) On a :

$$\frac{dV}{d\mathbf{p}}(\mathbf{p}) = -\hat{\mathbf{v}}^T + 2(\mathbf{p} - \hat{\mathbf{p}})^T - \frac{(\mathbf{p} - \hat{\mathbf{q}})^T}{\|\mathbf{p} - \hat{\mathbf{q}}\|^3}$$

Du fait que $\mathbf{w} = -\text{grad}(V(\mathbf{p}))$, on en déduit que :

$$\mathbf{w}(\mathbf{p}, t) = -\text{grad } \mathbf{V}(\mathbf{p}) = -\left(\frac{dV}{d\mathbf{p}}(\mathbf{p})\right)^T = \hat{\mathbf{v}} - 2(\mathbf{p} - \hat{\mathbf{p}}) + \frac{(\mathbf{p} - \hat{\mathbf{q}})}{\|\mathbf{p} - \hat{\mathbf{q}}\|^3}$$

2) Pour la commande, on pourra prendre une commande proportionnelle en vitesse et en cap, ce qui nous donne :

$$\mathbf{u} = \begin{pmatrix} \|\mathbf{w}(\mathbf{p}, t)\| - v \\ \text{sawtooth}(\text{angle}(\mathbf{w}(\mathbf{p}, t)) - \theta) \end{pmatrix}$$

où *sawtooth* est la fonction *dents de scie* qui évite les sauts de 2π . Du fait que la cible est $\hat{\mathbf{p}} = (t, t)$, la vitesse cible doit être prise à $\hat{\mathbf{v}} = (1, 1)$.

3) On prend le champ de vecteur :

$$\mathbf{w} = \bar{\mathbf{v}} - \frac{\mathbf{p} - \hat{\mathbf{p}}}{2} + \frac{\mathbf{p} - \hat{\mathbf{q}}}{\|\mathbf{p} - \hat{\mathbf{q}}\|^3} \text{ avec } \bar{\mathbf{v}} = \frac{d\hat{\mathbf{p}}}{dt} = \frac{1}{10} \begin{pmatrix} -\sin \frac{t}{10} \\ 2 \cos \frac{t}{10} \end{pmatrix}$$

qui donne un comportement acceptable. On a juste réduit le terme proportionnel (qui correspond à l'attractivité de la cible) afin de pas être trop sensible aux variations du champ de vecteurs autour de la cible ce qui engendrait des changements de cap violents et non souhaitables. Le programme est donné dans le fichier `potential.m`.

Chapitre 5

Localisation instantanée

La localisation consiste à retrouver la position du robot (c'est-à-dire les coordonnées de son centre ainsi que son orientation), ou plus généralement, tous les degrés de liberté du robot. Ce problème est rencontré lors de la navigation où l'on a besoin d'estimer la position, l'orientation et la vitesse du robot. Le problème de localisation est souvent considéré comme un cas particulier de l'estimation d'état qui sera traitée dans les chapitres suivants. Cependant, dans le cas où un modèle d'état précis du robot n'est pas disponible, une localisation instantanée reste souvent possible et peut suffire pour prendre une décision. Prenons, par exemple la situation où nous sommes sur un bateau et nous détectons un phare dont la position absolue ainsi que sa hauteur sont connues. En mesurant la hauteur du phare perçue, son angle relativement au bateau, et à l'aide d'une boussole, nous pouvons en déduire la position du bateau, et ceci, sans disposer d'un modèle d'état pour le bateau. La localisation instantanée, ou *sans modèle*, est une approche pour la localisation qui n'utilise pas l'équation d'évolution du robot, c'est-à-dire qu'elle ne cherche pas à rendre cohérentes les mesures à travers le temps. Cette localisation consiste principalement à résoudre des équations de nature géométrique qui sont souvent non linéaires. Les variables en jeu peuvent être des variables de position ou des variables cinématiques comme des vitesses ou des accélérations. Comme ces méthodes de localisation sont spécifiques et assez éloignées des méthodes d'estimation d'état, nous allons y consacrer un chapitre à part. Après avoir introduit les principaux capteurs utilisés pour la localisation, nous allons présenter la localisation goniométrique (qui utilise les angles de perception des amers par le robot) puis la multilatération qui utilise les distances entre le robot et les amers.

5.1. Capteurs

Les robots sont équipés de nombreux capteurs qui sont utilisés pour leur localisation. Nous allons maintenant en présenter quelques-uns.

Odomètres : les robots à roues sont généralement équipés d'odomètres qui mesurent les déplacements angulaires des roues. A partir des seuls odomètres, il est possible de calculer une estimation de la position du robot. La précision d'une telle localisation est très mauvaise du fait de l'intégration systématique de l'erreur d'estimation. On dit que l'estimation dérive.

Loch-Doppler : ce type de capteur, principalement utilisé en robotique sous-marine, permet de calculer la vitesse du robot. Un Loch-Doppler envoie des ultrasons qui sont réfléchis au fond de l'océan. Puisque le fond est immobile, le capteur est capable d'estimer la vitesse du robot en utilisant l'effet Doppler avec une très bonne précision (de l'ordre de 0.1 m/s).

Accéléromètres : ces capteurs fournissent une mesure de l'accélération instantanée en translation. Le principe d'un accéléromètre à un axe est illustré sur la figure 5.1. Généralement, trois accéléromètres sont utilisés par le robot. Du fait de la gravité, la valeur mesurée selon l'axe vertical doit être compensée.

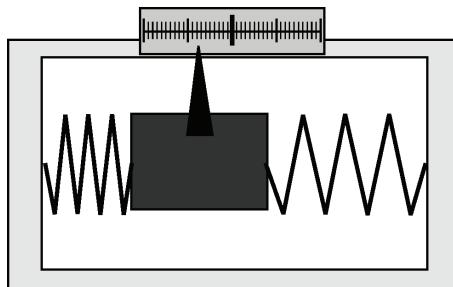


Figure 5.1. Principe d'un accéléromètre

Gyromètre ou gyroscope : ces capteurs fournissent une mesure de la vitesse instantanée de rotation. Il existe trois types de gyromètres : le gyromètre à effet Coriolis, le gyroscope mécanique et le gyroscope optique. Le principe du gyromètre à effet Coriolis est illustré par la figure 5.2 à gauche. Une tige verticale posée sur un disque horizontal vibre de gauche à droite. Si le disque tourne, du fait de la force de Coriolis, on ressent une vibration angulaire suivant l'axe de la barre dont l'amplitude permet de retrouver la vitesse de rotation du disque. Si le disque ne tourne pas, on ne ressent une vibration de translation, mais cette dernière n'est pas angulaire. Les gyromètres piézoélectriques, très utilisés pour la robotique à faible coût, forment une sous-classe des gyroscopes à effet Coriolis. Ils exploitent la variation d'amplitude d'un oscillateur piézoélectrique, induite par la force de Coriolis due au mouvement de rotation appliquée au capteur. Les gyroscopes mécaniques utilisent le fait qu'un corps en rotation tend à conserver son axe de rotation si aucun couple ne lui est appliqué. Un exemple bien connu est le gyroscope à cardans inventé par Foucault et représenté sur la figure 5.2 à droite. Un volant d'inertie au centre tourne à grande vitesse. Si

socle du gyroscope bouge, les deux angles des cardans ψ, θ changeront, mais pas l'axe de rotation du volant. A partir $\psi, \theta, \dot{\psi}, \dot{\theta}$, on peut retrouver la vitesse de rotation du socle (qui est fixé sur le robot). Si l'axe de rotation du volant est bien initialisé, et dans un cas parfait où aucun couple n'est exercé sur ce volant, un tel système pourrait théoriquement nous donner l'orientation du robot. Malheureusement, il existe toujours une petite dérive et seule, la vitesse de rotation peut être donnée de façon fiable et sans dérive.

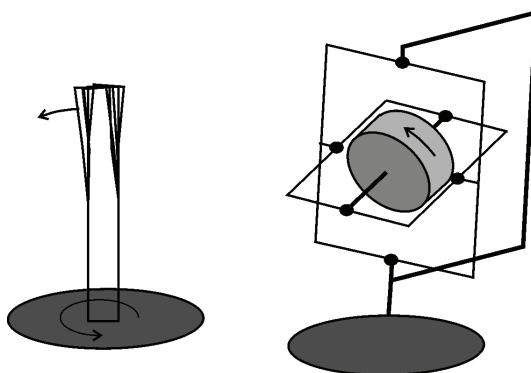


Figure 5.2. Gyroscope à effet Coriolis et gyroscope à cardans

Les gyroscopes optiques, plus récents, peuvent être aussi précis que les gyroscopes mécaniques. Ils utilisent l'effet Sagnac (pour un trajet optique circulaire, le temps que met la lumière pour faire un tour complet dépend du sens de parcours) et ont une précision de l'ordre de 0.001 deg/s. Le principe est illustré par la figure 5.3. Sur la figure (a), le laser part de sa source lumineuse représentée par le disque noir. Sur la figure (b), le séparateur de faisceau crée deux autres faisceaux qui parcourent dans un sens différent la boucle optique. Après différents rebonds sur les trois miroirs représentés en gris, les deux faisceaux se croisent. Puisque les deux faisceaux s'intersectent sur la gauche, le gyromètre est en train de tourner dans le sens indirect trigonométrique (c). Les faisceaux sont à nouveau séparés sur la figure (d). Les deux faisceaux qui arrivent sur le récepteur ne sont pas en phase. Leur déphasage permet de retrouver la vitesse de rotation du gyromètre, qui est fixé sur le robot.

Centrale inertielle : une telle centrale (aussi appelée IMU : *inertial measurement unit*) associe gyromètre et accéléromètre afin d'augmenter la précision de l'estimation. Les plus récents fusionnent d'autres types d'informations comme la vitesse estimée ou même la prise en compte de la rotation de la terre. Par exemple, la centrale Octans III de la société IXBLUE utilise l'effet Sagnac et la rotation de la terre pour en déduire la direction de l'axe nord-sud de la terre dans le repère du robot. La connaissance de cette direction nous donne deux équations impliquant les angles d'Euler du robot (voir section 1.2) qui sont la gîte ϕ , l'assiette θ et le cap ψ du robot, exprimé dans

un repère local. Grâce à l'accéléromètre inclus dans la centrale, il est possible d'en déduire le vecteur de gravité et de générer ainsi une équation supplémentaire qui permettra de calculer les trois angles d'Euler. Notons que les accéléromètres nous donnent aussi les accélérations suivant toutes les directions (le *cavalement* dans le sens du robot, le *pilonnement* (dans le sens de la verticale), l'*embardée* pour les accélérations latérales). La connaissance du vecteur de gravité et de l'axe de rotation par la centrale permet théoriquement, par un simple produit scalaire, de retrouver la latitude du robot. Cependant, la précision obtenue est trop faible pour être prise en compte dans la localisation.

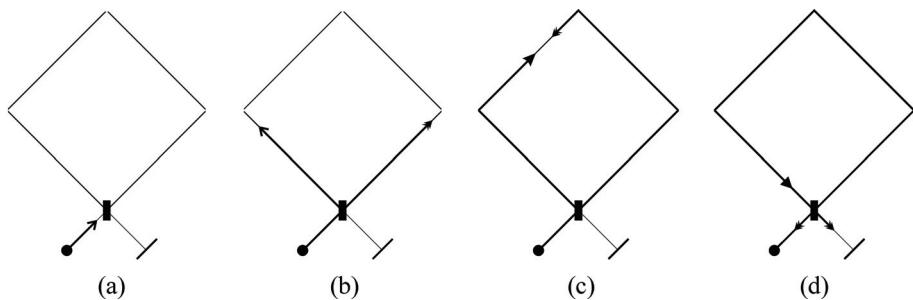


Figure 5.3. Principe de l'effet Sagnac pour les gyroscopes optiques

Baromètre : il mesure la pression. Dans le cas des robots sous-marins, il permet d'en déduire la profondeur du robot avec une précision de 1 cm. Pour les robots volants à l'intérieur des bâtiments, le baromètre permet de mesurer l'altitude avec une précision de l'ordre du mètre.

GPS (Global Positioning System) : le GNSS (*Global Navigation Satellite System*) est un système de navigation par satellites fournissant un service de géolocalisation couvrant le monde entier. De nos jours, sont opérationnels le système américain NAVSTAR (*NAvigation System by Timing And Ranging*) et le système russe GLONASS. Deux autres systèmes sont en cours de développement : le système chinois *Compass* et le système européen *Galileo*. En pratique, nos robots mobiles utiliseront le système américain, opérationnel depuis 1995, que nous appellerons GPS. Conçu à la base pour des applications exclusivement militaires et la précision des applications civiles était limitée à une centaine de mètres par la mise en place d'une dégradation volontaire de la qualité des signaux civils. La désactivation de cette dégradation en 2000 a permis de faire passer la précision à une dizaine de mètres. Du fait que les ondes électromagnétiques (ici autour de 1.2 MHz) ne se propagent pas sous l'eau ou à travers les murs, le GPS n'est opérationnel ni dans les bâtiments, ni dans l'eau. Ainsi, pendant une expérience de plongée, un robot sous-marin n'est capable de se localiser par GPS qu'au moment où il plonge et au moment où il réapparaît à la surface. Lorsqu'une station géoréférencée se trouve à proximité du robot et l'informe

des erreurs de distance calculée pour chaque satellite, une localisation avec une précision de ± 1 m est possible. Ce mode de fonctionnement forme ce que l'on appelle le GPS différentiel ou DGPS. Enfin, en utilisant la phase, il est même possible d'aller à une précision centimétrique. C'est le principe du *GPS cinématique*. Une présentation détaillée et pédagogique du GPS peut être trouvée dans la thèse de Vincent Drevelle [DRE 11]. En pratique, un GPS, nous donne une longitude ℓ_x et une latitude ℓ_y et il est souvent confortable de la convertir en coordonnées cartésiennes dans un repère local ($\mathbf{o}, \mathbf{i}, \mathbf{j}, \mathbf{k}$) fixé dans la zone où évolue le robot. Notons ℓ_x^0 et ℓ_y^0 la longitude et la latitude exprimée en radian de l'origine \mathbf{o} de ce repère. Nous supposerons que le vecteur \mathbf{i} indique le nord, \mathbf{j} indique l'est et \mathbf{k} est orienté vers le centre de la terre. Soit $\mathbf{p} = (p_x, p_y, p_z)$ les coordonnées du robot exprimées dans le repère $(\mathbf{o}, \mathbf{i}, \mathbf{j}, \mathbf{k})$. A partir de la latitude et de la longitude données par le GPS, nous pouvons en déduire les deux premières coordonnées du robot, exprimées en mètres dans le repère local, en utilisant la relation suivante (voir l'équation [4.4], page 138) :

$$\begin{pmatrix} p_x \\ p_y \end{pmatrix} = \rho \begin{pmatrix} 0 & 1 \\ \cos(\ell_y) & 0 \end{pmatrix} \begin{pmatrix} \ell_x - \ell_x^0 \\ \ell_y - \ell_y^0 \end{pmatrix} = \begin{pmatrix} \rho (\ell_y - \ell_y^0) \\ \rho \cos(\ell_y) (\ell_x - \ell_x^0) \end{pmatrix}$$

où ρ correspond à la distance entre \mathbf{o} et le centre de la terre ($\rho \simeq 6\,371$ km, si \mathbf{o} est pas trop loin du niveau de la mer). Cette formule est valable sur toute la terre, si nous supposons que cette dernière est sphérique et si le robot est dans le voisinage de l'origine \mathbf{o} (disons à une distance inférieure à 100 km). Pour comprendre cette formule, il faut noter que $\rho \cos(\ell_y)$ correspond à la distance entre \mathbf{o} et l'axe de rotation de la terre. Ainsi, si un robot se déplace sur une parallèle de latitude ℓ_y en modifiant sa longitude d'un angle $\alpha > 0$, il aura parcouru $\alpha \rho \cos(\ell_y)$ mètres. De même, si ce robot se déplace sur un méridien d'un angle de β en latitude, il aura parcouru $\beta \rho$ mètres.

Radar ou sonar : le robot émet des ondes électromagnétiques ou ultrasonores. Il en récupère l'écho et reconstruit une image qu'il interprète afin de cartographier son environnement. Le radar est principalement utilisé pour les robots de surface ou volant. Le sonar est utilisé comme télémètre à faible coût pour les robots roulants et pour la robotique sous-marine.

Caméras : les caméras sont des capteurs à bas coûts qui sont utilisés pour reconnaître de objets. Pour la localisation, elles sont utilisées comme goniomètres, c'est-à-dire qu'elles permettent de retrouver des angles relativement à des amers que l'on va alors utiliser pour se localiser. Les caméras sont aussi utilisées pour la reconnaissance d'objet et ceci même dans un contexte sous-marin [BAZ 08].

5.2. Localisation goniométrique

5.2.1. Formulation du problème

Il s'agit de se localiser à partir d'angles mesurés entre le robot et des amers dont la position en fonction du temps est connue. On considère le robot de la figure 5.4

se déplaçant sur un plan. On appelle *gisement* l'angle α_i entre l'axe du robot et le vecteur pointant sur l'amer. Ces angles ont pu être obtenus, par exemple, à l'aide d'une caméra.

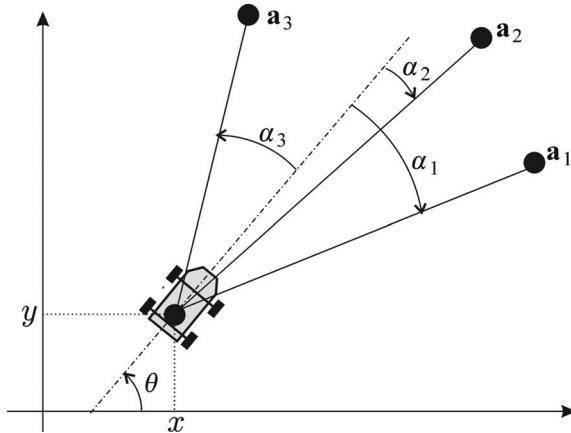


Figure 5.4. Un robot, se déplaçant sur un plan, mesure des angles pour se localiser

On rappelle que deux vecteurs \mathbf{u}, \mathbf{v} de \mathbb{R}^2 sont colinéaires si leur déterminant est nul, c'est-à-dire si $\det(\mathbf{u}, \mathbf{v}) = 0$. Ainsi, pour chaque amer, nous avons la relation :

$$\det \left(\begin{pmatrix} x_i - x \\ y_i - y \end{pmatrix}, \begin{pmatrix} \cos(\theta + \alpha_i) \\ \sin(\theta + \alpha_i) \end{pmatrix} \right) = 0$$

c'est-à-dire :

$$(x_i - x) \sin(\theta + \alpha_i) - (y_i - y) \cos(\theta + \alpha_i) = 0 \quad [5.1]$$

où (x_i, y_i) sont les coordonnées de l'amer \mathbf{a}_i et θ est le cap du robot.

5.2.2. Arcs capables

THÉORÈME 5.1.– *e l'angle inscrit : soit un triangle \mathbf{abm} comme représenté sur la figure 5.5. Notons \mathbf{c} le centre du cercle circonscrit à ce triangle (c'est-à-dire que \mathbf{c} est à l'intersection des trois médiatrices). Posons $\alpha = \widehat{\mathbf{amc}}$, $\beta = \widehat{\mathbf{cmb}}$, $\gamma = \widehat{\mathbf{acb}}$. On a la relation angulaire :*

$$\gamma = 2(\alpha + \beta)$$

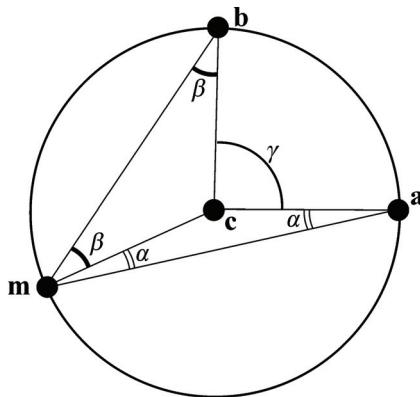


Figure 5.5. Illustration du théorème de l'angle inscrit

PREUVE.– Tout d'abord, les deux triangles amc et cmb sont isocèles. Donc on retrouve les angles α et β comme indiqué sur la figure. On faisant le tour du point c nous obtenons :

$$\gamma + (\pi - 2\beta) + (\pi - 2\alpha) = 2\pi$$

Soit $\gamma = 2\alpha + 2\beta$. ■

Une conséquence de ce théorème est que si m bouge sur le cercle, l'angle $\alpha + \beta$ ne bougera pas.

Arcs capables : soient deux points a_1 et a_2 . L'ensemble des points m tel que l'angle $\widehat{a_1ma_2}$ est égal à α est un arc de cercle, appelé *arc capable*. On peut le démontrer à partir des relations [5.1] ou à partir du théorème de l'angle inscrit. Se localiser par goniométrie se ramène souvent à intersecter des arcs. La figure 5.6 illustre cette notion d'arc capable.

5.2.3. Triangulation statique d'un robot plan

5.2.3.1. Deux amers et une boussole

Dans le cas où l'on dispose de deux amers et d'une boussole, nous avons, d'après [5.1], les deux relations :

$$\begin{cases} (x_1 - x) \sin(\theta + \alpha_1) - (y_1 - y) \cos(\theta + \alpha_1) = 0 \\ (x_2 - x) \sin(\theta + \alpha_2) - (y_2 - y) \cos(\theta + \alpha_2) = 0 \end{cases}$$

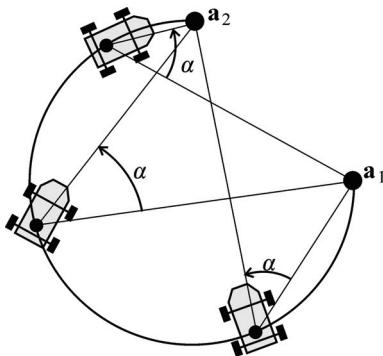


Figure 5.6. Les trois voitures perçoivent les amers avec le même angle

c'est-à-dire :

$$\underbrace{\begin{pmatrix} \sin(\theta + \alpha_1) - \cos(\theta + \alpha_1) \\ \sin(\theta + \alpha_2) - \cos(\theta + \alpha_2) \end{pmatrix}}_{\mathbf{A}(\theta, \alpha_1, \alpha_2)} \begin{pmatrix} x \\ y \end{pmatrix} = \underbrace{\begin{pmatrix} x_1 \sin(\theta + \alpha_1) - y_1 \cos(\theta + \alpha_1) \\ x_2 \sin(\theta + \alpha_2) - y_2 \cos(\theta + \alpha_2) \end{pmatrix}}_{\mathbf{b}(\theta, \alpha_1, \alpha_2, x_1, y_1, x_2, y_2)}$$

soit :

$$\begin{pmatrix} x \\ y \end{pmatrix} = \mathbf{A}^{-1}(\theta, \alpha_1, \alpha_2) \cdot \mathbf{b}(\theta, \alpha_1, \alpha_2, x_1, y_1, x_2, y_2)$$

Le problème de localisation est donc un problème linéaire qui peut se résoudre analytiquement. On a un problème d'identifiabilité si la matrice à inverser a un déterminant nul, c'est-à-dire :

$$\begin{aligned} \sin(\theta + \alpha_1) \cos(\theta + \alpha_2) - \cos(\theta + \alpha_1) \sin(\theta + \alpha_2) &= 0 \\ \Leftrightarrow \tan(\theta + \alpha_2) &= \tan(\theta + \alpha_1) \\ \Leftrightarrow \theta + \alpha_2 &= \theta + \alpha_1 + k\pi, \quad k \in \mathbb{N} \\ \Leftrightarrow \alpha_2 &= \alpha_1 + k\pi, \quad k \in \mathbb{N} \end{aligned}$$

Ce qui correspond à une situation où les deux amers et le robot sont alignés.

5.2.3.2. Trois amers

Dans le cas où nous ne disposons plus de boussole, il nous faut au moins trois amers. Il nous faut alors résoudre le système de trois équations à trois inconnues :

$$(x_i - x) \sin(\theta + \alpha_i) - (y_i - y) \cos(\theta + \alpha_i) = 0, \quad i \in \{1, 2, 3\}$$

On peut montrer que ce système a toujours une solution unique, sauf si le robot est situé sur le cercle passant par les trois amers. En effet, dans un tel cas, les arcs capables se superposent.

5.2.4. Triangulation dynamique

5.2.4.1. Un amer, une boussole, des odomètres

Dans le cas de l'observation d'état dynamique, on cherche une relation liant la position du robot aux dérivées des grandeurs mesurées. Pour la localisation, nous allons supposer qu'un seul amer est disponible. Nous allons utiliser les équations :

$$\begin{cases} \dot{x} = v \cos \theta \\ \dot{y} = v \sin \theta \end{cases} \quad [5.2]$$

où v représente la vitesse du robot mesurée par l'odomètre et θ son cap mesuré par la boussole. Ces équations, qui ont une nature cinématique, ne sont pas censées décrire le comportement d'un robot particulier dans un but de pouvoir le contrôler. Les entrées v et θ ne sont pas nécessairement les entrées réelles du système sur lesquelles nous pourrions agir. Ces équations doivent être comprises comme une simple relation différentielle entre des variables d'un robot plan. En dérivant la relation [5.1], on obtient :

$$\begin{aligned} & (\dot{x}_i - \dot{x}) \sin(\theta + \alpha_i) + (x_i - x) (\dot{\theta} + \dot{\alpha}_i) \cos(\theta + \alpha_i) \\ & - (\dot{y}_i - \dot{y}) \cos(\theta + \alpha_i) + (y_i - y) (\dot{\theta} + \dot{\alpha}_i) \sin(\theta + \alpha_i) = 0 \end{aligned} \quad [5.3]$$

Prenons les relations [5.1] et [5.3] pour $i = 1$. En isolant x et y , nous obtenons :

$$\begin{aligned} \begin{pmatrix} x \\ y \end{pmatrix} &= \begin{pmatrix} \sin(\theta + \alpha_1) & \cos(\theta + \alpha_1) \\ -\cos(\theta + \alpha_1) & \sin(\theta + \alpha_1) \end{pmatrix} \\ &\cdot \begin{pmatrix} -y_1 \\ x_1 - \frac{y_1 - v \sin \theta}{\dot{\theta} + \dot{\alpha}_1} \end{pmatrix} \begin{pmatrix} \cos(\theta + \alpha_1) \\ \sin(\theta + \alpha_1) \end{pmatrix} \end{aligned} \quad [5.4]$$

Cette relation peut nous permettre de nous localiser à l'aide d'un seul amer, fixe où mobile, et d'autres capteurs proprioceptifs. Par exemple, dans le cas où nous disposons d'une boussole, d'odomètres (pour un robot roulant), nous sommes capables de mesurer le cap θ par la boussole, la vitesse v et $\dot{\theta}$ par les odomètres. La relation [5.4] nous permet alors de calculer la position x et y à l'instant considéré.

5.2.4.2. Un amer sans boussole

Dans la situation où la boussole n'est pas présente, il nous manque une équation. Il nous faut soit rajouter un deuxième amer, soit dériver à nouveau. Restons avec un seul amer et dérivons la relation [5.3], nous obtenons :

$$\begin{aligned} & (\ddot{x}_1 - \ddot{x}) \sin(\theta + \alpha_1) - (\ddot{y}_1 - \ddot{y}) \cos(\theta + \alpha_1) \\ & + (x_1 - x) (\ddot{\theta} + \ddot{\alpha}_1) \cos(\theta + \alpha_1) + (y_1 - y) (\ddot{\theta} + \ddot{\alpha}_1) \sin(\theta + \alpha_1) \\ & + 2(\dot{x}_1 - \dot{x})(\dot{\theta} + \dot{\alpha}_1) \cos(\theta + \alpha_1) + 2(\dot{y}_1 - \dot{y})(\dot{\theta} + \dot{\alpha}_1) \sin(\theta + \alpha_1) \\ & - (x_1 - x)(\dot{\theta} + \dot{\alpha}_1)^2 \sin(\theta + \alpha_1) + (y_1 - y)(\dot{\theta} + \dot{\alpha}_1)^2 \cos(\theta + \alpha_1) = 0 \end{aligned}$$

De plus :

$$\begin{cases} \ddot{x} = \dot{v} \cos \theta - v \dot{\theta} \sin \theta \\ \ddot{y} = \dot{v} \sin \theta + v \dot{\theta} \cos \theta \end{cases}$$

Nous obtenons donc un système de trois équations à trois inconnues x, y, θ :

$$\begin{aligned} (x_1 - x) \sin(\theta + \alpha_1) - (y_1 - y) \cos(\theta + \alpha_1) &= 0 \\ (\dot{x}_1 - v \cos \theta) \sin(\theta + \alpha_1) + (x_1 - x) (\dot{\theta} + \dot{\alpha}_1) \cos(\theta + \alpha_1) \\ - (\dot{y}_1 - v \sin \theta) \cos(\theta + \alpha_1) + (y_1 - y) (\dot{\theta} + \dot{\alpha}_1) \sin(\theta + \alpha_1) &= 0 \\ (\ddot{x}_1 - \dot{v} \cos \theta + v \dot{\theta} \sin \theta) \sin(\theta + \alpha_1) - (\ddot{y}_1 - \dot{v} \sin \theta - v \dot{\theta} \cos \theta) \cos(\theta + \alpha_1) \\ + (x_1 - x) (\ddot{\theta} + \ddot{\alpha}_1) \cos(\theta + \alpha_1) + (y_1 - y) (\ddot{\theta} + \ddot{\alpha}_1) \sin(\theta + \alpha_1) \\ + 2(\dot{x}_1 - v \cos \theta) (\dot{\theta} + \dot{\alpha}_1) \cos(\theta + \alpha_1) + 2(\dot{y}_1 - v \sin \theta) (\dot{\theta} + \dot{\alpha}_1) \sin(\theta + \alpha_1) \\ - (x_1 - x) (\dot{\theta} + \dot{\alpha}_1)^2 \sin(\theta + \alpha_1) + (y_1 - y) (\dot{\theta} + \dot{\alpha}_1)^2 \cos(\theta + \alpha_1) &= 0 \end{aligned}$$

Les grandeurs $x_1, y_1, \dot{x}_1, \dot{y}_1, \ddot{x}_1, \ddot{y}_1$ sont calculées à partir de la trajectoire de l'amer 1, dont on connaît une expression analytique. Les grandeurs $\alpha_1, \dot{\alpha}_1, \ddot{\alpha}_1$ sont mesurées, par hypothèse. Les grandeurs $\dot{\theta}, \ddot{\theta}$ peuvent être obtenues à partir d'un gyromètre. La vitesse v peut être mesurée par les odomètres. On comprend bien que ce système n'est pas facile à résoudre de façon analytique et n'admet pas toujours une solution unique. Par exemple, si l'amer est fixe, par symétrie de rotation, il est clair que nous n'arriverons pas à retrouver l'angle θ . Dans un tel cas, il nous faudra au moins deux amers pour pouvoir se localiser.

5.3. Multilatération

La *multilatération* est une technique de localisation fondée sur la mesure de la différence de distances entre le robot et les amers. En effet, dans de nombreuses situations (comme c'est le cas pour la localisation GPS), les horloges entre les amers et le robot ne sont pas synchronisées et on ne mesure donc pas directement la distance absolue entre les amers et le robot (par temps vol d'une onde hertzienne ou acoustique), mais la différence entre ces distances. Nous allons ici donner le principe de cette technique.

Quatre amers envoient en même temps un signal bref à l'instant t_0 qui se propage avec une vitesse c . Chaque signal émis contient un identifiant pour l'amer, la position de l'amer et le temps d'émission t_0 . Le robot (qui ne dispose pas d'horloge précise, mais uniquement d'un chronomètre précis) reçoit les quatre signaux aux instants t_i .

Il en déduit aisément les décalages entre les temps de réception $\tau_2 = t_2 - t_1$, $\tau_3 = t_3 - t_1$, $\tau_4 = t_4 - t_1$ (voir figure 5.7). On obtient ainsi les quatre équations :

$$\begin{aligned}\sqrt{(x - x_1)^2 + (y - y_1)^2 + (z - z_1)^2} &= c(t_1 - t_0) \\ \sqrt{(x - x_2)^2 + (y - y_2)^2 + (z - z_2)^2} &= c(\tau_2 + t_1 - t_0) \\ \sqrt{(x - x_3)^2 + (y - y_3)^2 + (z - z_3)^2} &= c(\tau_3 + t_1 - t_0) \\ \sqrt{(x - x_4)^2 + (y - y_4)^2 + (z - z_4)^2} &= c(\tau_4 + t_1 - t_0)\end{aligned}$$

où les paramètres, connus avec une bonne précision sont c , t_0 , $x_1, y_1, z_1, \dots, x_4, y_4, z_4$, τ_2 , τ_3 , τ_4 . Les quatre inconnues sont x, y, z, t_1 . Une résolution de ce système permet à la fois au système de se localiser, mais aussi de recaler son horloge (par l'intermédiaire de t_1). Dans le cas du GPS, les amers sont mobiles. Ils utilisent un principe similaire pour se localiser et se synchroniser, à partir d'amers fixes situés au sol.

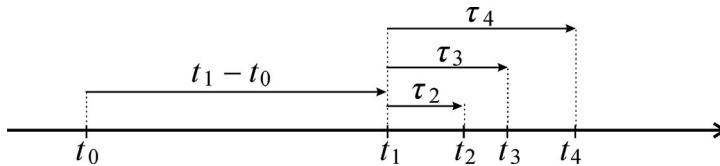


Figure 5.7. Le temps d'émission t_0 et les décalages des temps d'arrivée τ_2, τ_3, τ_4 sont connus

5.4. Exercices

Exercice 5.1. Estimation d'état instantanée

La localisation consiste à trouver la position et l'orientation du robot. Ce problème peut parfois se ramener à un problème d'estimation d'état, si nous disposons d'un modèle d'état pour notre robot. Dans cet exercice, nous allons donner une méthode parfois utilisée pour l'estimation d'état des systèmes non linéaires. Considérons le tricycle de la section 2.5 page 64 et décrit par les équations d'état :

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{v} \\ \dot{\delta} \end{pmatrix} = \begin{pmatrix} v \cos \delta \cos \theta \\ v \cos \delta \sin \theta \\ v \sin \delta \\ u_1 \\ u_2 \end{pmatrix}$$

On mesure les positions x et y avec une si bonne précision que nous pouvons supposer $\dot{x}, \dot{y}, \ddot{x}, \ddot{y}$ connus. Exprimer les autres variables d'état θ, v, δ en fonction de $x, y, \dot{x}, \dot{y}, \ddot{x}, \ddot{y}$.

Exercice 5.2. Localisation par lidar

On cherche ici à développer une méthode rapide de localisation d'un robot par un télémètre laser rotatif, ou lidar (*light radar*), de type Hokuyo, dans une pièce rectangulaire dont on ne connaît ni la longueur, ni la largeur.

1) Soient $\mathbf{a}_1, \dots, \mathbf{a}_{n_p}$ des points de \mathbb{R}^2 censés appartenir à une même droite. Trouver par une méthode des moindres-carrés cette droite. On représentera cette droite sous forme normale :

$$x \cos \alpha + y \sin \alpha = d, \text{ avec } d \geq 0$$

où α, d sont les paramètres de la droite.

2) On considère cent mesures θ_i d'un même angle θ d'un robot par cent boussoles posées sur le robots dont trente d'entre elles sont défectueuses. Estimer θ par la méthode de la médiane. Pour cela, on définira la fonction de *désambiguation* :

$$\mathbf{f} : \begin{cases} \mathbb{R} \rightarrow \mathbb{R}^2 \\ \theta \rightarrow \begin{pmatrix} \cos \theta \\ \sin \theta \end{pmatrix} \end{cases}$$

Une telle fonction, qui doit être continue, a pour objectif d'associer à deux angles équivalents (c'est-à-dire égaux à 2π près), une unique image, c'est-à-dire :

$$\theta_1 \sim \theta_2 \Leftrightarrow f(\theta_1) = f(\theta_2)$$

3) Le lidar du robot qui voit suivant un angle de 180° nous donne 512 points censés appartenir au rectangle représentant notre pièce. Ces points sont rangés dans le fichiers `lidar_data.mat`. On les prend par groupes de dix (soit 51 groupes). On cherche la droite qui passe au mieux par chaque groupe (par la méthode des moindres-carrés) et on ne garde que les groupes dont le résidu est faible. On obtient ainsi m droites, représentées par n points de la forme (α_i, d_i) , $i \in \{1, \dots, m\}$ dans l'espace dit de *Hough*. Un couple (α_i, d_i) est appelé *alignement*. On utilisant un estimateur de la médiane, trouver les quatre directions possibles pour notre pièce (sachant qu'elle est rectangulaire). Pourquoi cet estimateur de la médiane est-il considéré comme robuste ?

4) Comment filtrer les angles α_i des alignements ?

5) Comment filtrer les d_i ?

6) En déduire une méthode pour localiser le robot.

Exercice 5.3. Localisation goniométrique instantanée

On considère un robot bateau \mathcal{R} décrit par les équations d'état :

$$\begin{cases} \dot{x} = v \cos \theta \\ \dot{y} = v \sin \theta \\ \dot{\theta} = u_1 \\ \dot{v} = u_2 \end{cases}$$

où v est sa vitesse du robot, θ son orientation et (x, y) les coordonnées de son centre. Son vecteur d'état est donné par $\mathbf{x} = (x, y, \theta, v)$. Dans l'environnement du robot se trouve un amer (un phare par exemple) $\mathbf{m} = (x_m, y_m)$ ponctuel dont on connaît la position (voir figure 5.8). Le robot \mathcal{R} est équipé de cinq capteurs : une caméra omnidirectionnelle (qui lui permet de mesurer l'angle de gisement α de la direction de l'amer), des odomètres pour mesurer sa vitesse, une boussole qui lui donne son cap θ , un accéléromètre pour u_2 et un gyromètre pour u_1 .

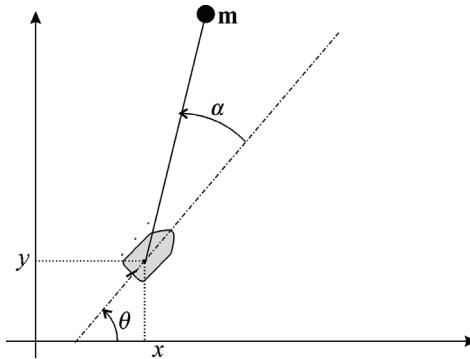


Figure 5.8. Localisation goniométrique

1) Simuler ce système dans une situation où le robot bouge autour de \mathbf{m} afin de générer les signaux $\alpha, \dot{\alpha}, \theta, v, u_1, u_2$. Pour la génération de α , on utilisera la fonction arc-tangente à deux arguments atan2 . Comme nous l'avons déjà vu à l'équation [1.8] page 20, la fonction $\text{atan2}(b, a)$ renvoie l'argument du vecteur de coordonnées (a, b) . Pour la génération de $\dot{\alpha}$ on pourra utiliser le fait que :

$$\frac{\partial \text{atan2}(b, a)}{\partial a} = -\frac{b}{a^2 + b^2} \quad \text{et} \quad \frac{\partial \text{atan2}(b, a)}{\partial b} = \frac{a}{a^2 + b^2}$$

2) Concevoir un système localisation instantanée pour \mathcal{R} . Vérifier le système de localisation par une simulation. Rajouter un petit bruit blanc gaussien sur vos mesures pour tester la robustesse de la localisation.

3) Le robot \mathcal{R} ne dispose plus d'odomètres. Comment adapter l'approche précédente pour permettre une localisation.

4) Toujours dans le cas sans odomètres, et en supposant que \dot{c} n'est pas disponible, utiliser un filtre de Kalman pour la localisation.

5) On rajoute un deuxième robot \mathcal{R}_b sur la scène qui est du même type que le premier robot (que nous appelons maintenant \mathcal{R}_a) sauf qu'il ne possède pas d'odomètre contrairement à \mathcal{R}_a . Le robot \mathcal{R}_b peut voir \mathcal{R}_a (voir figure 5.9) ce qui lui donne un angle β_b . Les deux robots peuvent communiquer par WIFI. Concevoir un système de localisation pour le robot \mathcal{R}_b qui lui permet aussi de retrouver sa vitesse.

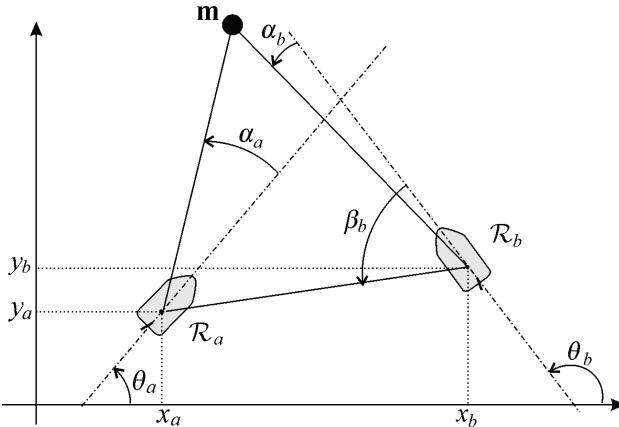


Figure 5.9. Les robots \mathcal{R}_a et \mathcal{R}_b peuvent se voir, ce qui va les aider dans la localisation

6) Le robot \mathcal{R}_a peut voir \mathcal{R}_b , ce qui lui donne une nouvelle mesure angulaire β_a . En déduire une méthode de localisation pour \mathcal{R}_a et \mathcal{R}_b plus fiable (c'est-à-dire qu'elle possède moins de singularités) que celle développée à la question 1.

Exercice 5.4. Localisation par mesure de distances

On considère un robot décrit par les équations d'état :

$$\begin{cases} \dot{x} = v \cos \theta \\ \dot{y} = v \sin \theta \\ \dot{\theta} = u_1 \\ \dot{v} = u_2 \end{cases}$$

où v est sa vitesse du robot, θ son orientation et (x, y) les coordonnées de son centre c. Son vecteur d'état est donné par $\mathbf{x} = (x, y, \theta, v)$.

Dans l'environnement du robot se trouve un amer m ponctuel fixe dont on connaît la position (voir figure 5.10). A chaque instant, le robot mesure la distance d entre son

centre \mathbf{c} et l'amer. Si l'amer et le robot ont tous les deux des horloges synchronisées, un tel système de mesure de distance peut se faire par la mesure du temps de vol d'un son entre l'amer et le robot. De plus, en utilisant l'effet Doppler, le robot est aussi capable de mesurer \dot{d} avec une très bonne précision. En plus du microphone qui lui permet de mesurer d et \dot{d} , le robot est équipé d'odomètres pour mesurer sa vitesse v , une boussole qui lui donne son cap θ . Dans cet exercice, on cherche à construire un système de localisation instantané afin de déterminer sa position (x, y) à partir de d, \dot{d}, θ, v .

- 1) Pour un vecteur (d, \dot{d}, θ, v) donné, il peut exister plusieurs configurations possibles pour le robot. Dessiner sur le dessin, toutes les configurations pour le robot compatibles avec celle qui est représentée.
- 2) On considère le repère \mathcal{R}_1 centré en \mathbf{m} représenté sur la figure. Donner l'expression des coordonnées (x_1, y_1) du centre \mathbf{c} du robot en fonction de x, y, θ, x_m, y_m . Il s'agit en fait d'une équation de changement de repère.
- 3) Exprimer x_1 et y_1 en fonction de d, \dot{d}, v .
- 4) En déduire la ou les positions du robot (x, y) en fonction de $(d, \dot{d}, \theta, v, x_m, y_m)$. Quelles sont les singularités de ce système de localisation ? Dans quels cas a-t-on une seule solution ?

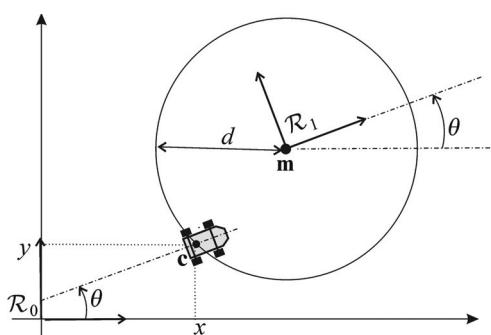


Figure 5.10. Le robot mesure la distance à l'amer

5.5. Corrections

Correction de l'exercice 5.1 (estimation d'état instantanée)

On a :

$$\theta = \arctan \left(\frac{\dot{y}}{\dot{x}} \right)$$

En dérivant cette équation, on obtient :

$$\dot{\theta} = \frac{1}{\left(\frac{\dot{y}}{\dot{x}}\right)^2 + 1} \left(\frac{\ddot{y}\dot{x} - \ddot{x}\dot{y}}{\dot{x}^2} \right)$$

De plus, d'après les équations d'état du tricycle :

$$\begin{cases} v \cos \delta = \frac{\dot{x}}{\cos \theta} \\ v \sin \delta = \dot{\theta} \end{cases}$$

En isolant v et δ , nous obtenons :

$$\begin{cases} v = \sqrt{(v \cos \delta)^2 + (v \sin \delta)^2} = \sqrt{\frac{\dot{x}^2}{\cos^2 \theta} + \dot{\theta}^2} \\ \delta = \arctan \left(\frac{\dot{\theta}}{\frac{\dot{x}}{\cos(\theta)}} \right) \end{cases}$$

Ainsi, le vecteur d'état s'exprime comme une fonction analytique de la sortie (x, y) et de ses dérivées :

$$\begin{pmatrix} x \\ y \\ \theta \\ v \\ \delta \end{pmatrix} = \begin{pmatrix} x \\ y \\ \arctan \left(\frac{\dot{y}}{\dot{x}} \right) \\ \sqrt{\frac{\dot{x}^2}{\cos^2 \left(\arctan \left(\frac{\dot{y}}{\dot{x}} \right) \right)} + \frac{1}{\left(\left(\frac{\dot{y}}{\dot{x}} \right)^2 + 1 \right)^2} \left(\frac{\ddot{y}\dot{x} - \ddot{x}\dot{y}}{\dot{x}^2} \right)^2} \\ \arctan \left(\frac{\frac{1}{\left(\frac{\dot{y}}{\dot{x}} \right)^2 + 1} \left(\frac{\ddot{y}\dot{x} - \ddot{x}\dot{y}}{\dot{x}^2} \right)}{\frac{\dot{x}}{\cos \left(\arctan \left(\frac{\dot{y}}{\dot{x}} \right) \right)}} \right) \end{pmatrix}$$

On a ainsi un observateur d'état que l'on peut qualifier de *quasi statique* car, contrairement aux observateurs classiques, il ne demande pas l'intégration d'équations différentielles (ils ne sont donc pas dynamiques). Obtenir un tel observateur peut se faire pour une large classe de systèmes non linéaires englobant la classe des systèmes plats [LAR 03].

Correction de l'exercice 5.2 (localisation par lidar)

1) Pour les équations de droite, il existe la forme explicite $y = ax + b$, la forme standard $ax + by + c = 0$, la forme normale $x \cos \alpha + y \sin \alpha = d$, et d'autres formes dont nous ne parlerons pas. La forme explicite ne peut représenter les droites verticales et donc cette forme possède des singularités. La forme standard n'admet pas de singularité mais le modèle associé est non identifiable (car des jeux de paramètres différents peuvent conduire à la même droite). La forme normale est identifiable, sans singularité, mais elle est non linéaire. Prenons la forme $p_1x + p_2y = 1$ (singulière si la droite passe par 0, ce qui ne sera pas le cas pour notre robot). Nous avons :

$$p_1x_i + p_2y_i = 1$$

c'est-à-dire, en considérant qu'un petit bruit est possible dans les mesures :

$$\underbrace{\begin{pmatrix} x_1 & y_1 \\ \vdots & \vdots \\ x_n & y_n \end{pmatrix}}_{\mathbf{A}} \begin{pmatrix} p_1 \\ p_2 \end{pmatrix} \simeq \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}$$

Ainsi, on prend pour estimateur :

$$\hat{\mathbf{p}} = \underbrace{(\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T}_{\mathbf{K}} \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}$$

Or, nous voulons une équation du type :

$$x \frac{\cos \hat{\alpha}}{\hat{d}} + y \frac{\sin \hat{\alpha}}{\hat{d}} = 1, \text{ avec } \hat{d} \geq 0$$

Nous prendrons donc :

$$\hat{d} = \frac{1}{\|\hat{\mathbf{p}}\|} \text{ et } \hat{\alpha} = \text{angle}(\hat{\mathbf{p}})$$

Afin de vérifier nos calculs, générerons sous MATLAB 1 000 points formant un nuage de points approximativement alignés.

```
N=1000 ; xi=5*randn(N,1) ; yi=-2*xi+1+randn(N,1) ;
plot(xi,yi,'+') ;
```

L'estimation de la droite se fait comme suit :

```
A=[xi,yi] ; y=ones(N,1) ; K=inv(A'*A)*A' ; phat=K*y ;
dhat=1/norm(phat) ;
alphahat=angle(phat) ; yhat=A*phat ; residu=norm(yhat-y) ;
```

Afin de tracer la droite estimée, on lance les instructions suivantes :

```
x=-15 :0.1 :15; y=(-phat(1)*x+1)/phat(2); plot(x,y,'.');
```

2) On prend la médiane \hat{x} des x_i et \hat{y} des y_i . L'estimateur d'angle est $\hat{\theta} = \arg(\hat{x}, \hat{y})$. Afin d'illustrer la robustesse de l'estimateur, générerons un jeu de données sous MATLAB :

```
theta=1; N=100; b=0.1*randn(N,1)+round(randn(N,1));
thetai=theta+b;
```

L'estimateur est le suivant :

```
xi=cos(thetai); yi=sin(thetai); xhat=median(xi);
yhat=median(yi);
theta_hat=atan2(yhat,xhat); hold on; plot(xi,yi,'+');
plot(xhat,yhat,'*');
```

3) Deux angles α_1 et α_2 sont équivalents si :

$$\begin{aligned} \alpha_2 &= \alpha_1 + \frac{k\pi}{2}, & k \in \mathbb{Z} \\ \Leftrightarrow 4\alpha_2 &= 4\alpha_1 + 2k\pi, \\ \Leftrightarrow \begin{cases} \cos 4\alpha_2 = \cos 4\alpha_1 \\ \sin 4\alpha_2 = \sin 4\alpha_1 \end{cases} \end{aligned}$$

avec $k \in \mathbb{Z}$. A chaque alignement (α_i, d_i) , on génère les m points du cercle unité :

$$\begin{pmatrix} x_i \\ y_i \end{pmatrix} = \begin{pmatrix} \cos 4\alpha_i \\ \sin 4\alpha_i \end{pmatrix}$$

qui représente la fonction de désambigüation. On prend la médiane \hat{x} des x_i et la médiane \hat{y} des y_i . L'argument du vecteur (\hat{x}, \hat{y}) nous donne l'angle $4\hat{\alpha}$ de la pièce (dans le repère du robot) à un quart de tour près. On prendra donc :

$$\hat{\alpha} = \frac{1}{4} \operatorname{atan2}(\hat{y}, \hat{x}) \in \left[-\frac{\pi}{4}, \frac{\pi}{4} \right]$$

Il est bien sûr important de supprimer à ce niveau les outliers d'orientation, c'est-à-dire les alignements (α_i, d_i) tels que la quantité $|\hat{x} - x_i| + |\hat{y} - y_i|$ ne soit pas négligeable. Afin d'illustrer le principe, on génère N angles α_i comme suit :

```
alpha=1; N=100; b=0.01*randn(N,1)+pi/2*round(10*randn(N,1));
alphai=alpha+b;
```

L'estimateur obtenu est :

```
xi=cos(4*alphai); yi=sin(4*alphai); xhat=median(xi);
yhat=median(yi);
alpha_hat=atan2(yhat,xhat)/4; plot(xi,yi,'+');
plot(cos(alpha_hat),sin(alpha_hat),'*');
```

4) Pour chaque α_i de l'alignement (α_i, d_i) , on calcule le numéro du mur :

$$k_i = \text{mod} \left(\text{round} \left(\frac{\alpha_i - \hat{\alpha}}{\pi/4} \right), 4 \right) \in \{0, 1, 2, 3\}$$

L'angle filtré est alors donné par :

$$\alpha_i = \hat{\alpha} + k_i \frac{\pi}{4}$$

5) Pour chaque $k \in \{0, 1, 2, 3\}$, on calcule (par l'estimateur de la médiane) la distance :

$$\hat{\delta}_k = \text{mediane} \{d_i \text{ avec } i \text{ tel que } k_i = k\}$$

6) On définit η_k comme le nombre d'alignements compatibles avec le mur k , c'est-à-dire $\eta_k = \text{card}(\{i \mid k_i = k\})$. On a donc à notre connaissance deux ou trois distances $\hat{\delta}_k$ correspondant à l'orientation $\hat{\alpha} + k_i \frac{\pi}{4}$ et dont l'indice de confiance est donnée par η_k . Pour casser la symétrie, il nous faut des informations complémentaires (comme les longueurs et largeurs ℓ_x et ℓ_y du rectangle, une information de cap). La figure 5.11 donne une illustration du principe de localisation du robot à partir des mesures télémétriques.

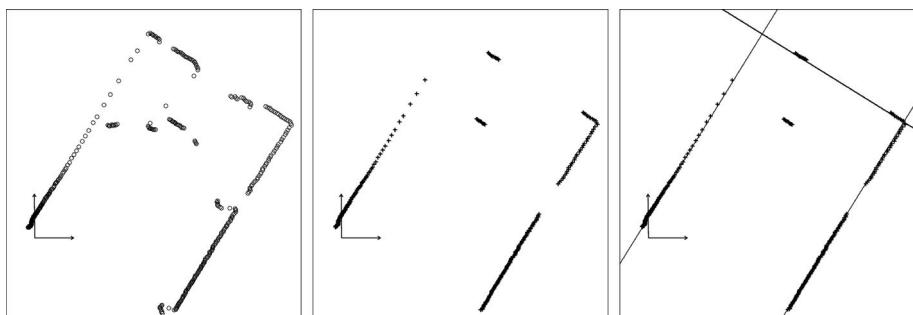


Figure 5.11. A gauche : données brutes du lidar ; au milieu : données associées aux alignements dont le résidu est faible ; à droite : murs reconstitués

Correction de l'exercice 5.3 (localisation goniométrique instantanée)

1) Pour la simulation, on a besoin de α et $\dot{\alpha}$. Tout d'abord, on a :

$$\alpha = -\theta + \text{atan}(y_m - y, x_m - x)$$

En dérivant cette relation, on obtient :

$$\begin{aligned}\dot{\alpha} &= -\dot{\theta} + \frac{d}{dt} \text{atan}(y_m - y, x_m - x) \\ &= -\dot{\theta} + \left(\frac{-(y_m - y)}{(x_m - x)^2 + (y_m - y)^2} \frac{(x_m - x)}{(x_m - x)^2 + (y_m - y)^2} \right) \left(\begin{array}{l} \dot{x}_m - \dot{x} \\ \dot{y}_m - \dot{y} \end{array} \right)\end{aligned}$$

Et donc, puisque $\dot{x} = v \cos \theta$, $\dot{y} = v \sin \theta$ et $\dot{\theta} = u_1$, on a :

$$\dot{\alpha} = -u_1 + \frac{(x_m - x)(\dot{y}_m - v \sin \theta) - (y_m - y)(\dot{x}_m - v \cos \theta)}{(x_m - x)^2 + (y_m - y)^2}$$

Le code MATLAB de la fonction d'évolution est :

```
function y = g(x,u,m,mdot)
st=sin(x(3));ct=cos(x(3));
v=m-[x(1:2)]; alpha=-x(3)+angle(v)
alphadot=-u(1)+(1/norm(v)^2)*(v(1)*(mdot(2)-x(4)*st)-v(2)
*(mdot(1)-x(4)*ct));
y=[alpha;alphadot;x(3);x(4)];
end
```

2) Nous avons la relation :

$$\det \left(\begin{pmatrix} x_m - x \\ y_m - y \end{pmatrix}, \begin{pmatrix} \cos(\theta + \alpha) \\ \sin(\theta + \alpha) \end{pmatrix} \right) = 0$$

c'est-à-dire :

$$(x_m - x) \sin(\theta + \alpha) - (y_m - y) \cos(\theta + \alpha) = 0$$

En dérivant cette relation, on obtient :

$$\begin{aligned}(\dot{x}_m - \dot{x}) \sin(\theta + \alpha) + (x_m - x) (\dot{\theta} + \dot{\alpha}) \cos(\theta + \alpha) \\ - (\dot{y}_m - \dot{y}) \cos(\theta + \alpha) + (y_m - y) (\dot{\theta} + \dot{\alpha}) \sin(\theta + \alpha) = 0\end{aligned}$$

Prenons les relations ci-dessus. En isolant x et y , nous obtenons :

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} \sin(\theta + \alpha) & \cos(\theta + \alpha) \\ -\cos(\theta + \alpha) & \sin(\theta + \alpha) \end{pmatrix} \left(x_m - \frac{-y_m}{\dot{\theta} + \dot{\alpha}} y_m + \frac{\dot{x}_m - v \cos \theta}{\dot{\theta} + \dot{\alpha}} \right) \begin{pmatrix} \cos(\theta + \alpha) \\ \sin(\theta + \alpha) \end{pmatrix}$$

Cette relation peut nous permettre de nous localiser à l'aide d'un seul amer, fixe où mobile, et d'autres capteurs proprioceptifs. Par exemple, dans le cas où nous disposons d'une boussole et d'odomètres (pour un robot roulant), nous sommes capables de mesurer le cap θ_a par la boussole, la vitesse v_a et $\dot{\theta}_a$ par les odomètres. Le code est donné dans le programme `locboat.m`. Le script MATLAB correspondant est :

```
function phat = loc(u,y,m,mdot)
alpha_m=y(1); dalpha_m=y(2); theta=y(3); v=y(4);
beta=theta+alpha_m; dbeta=(u(1)+dalpha_m);
A1=[sin(beta),cos(beta);-cos(beta),sin(beta)];
A2=[-m(2),m(1);m(1)-(mdot(2)-v*sin(theta))/dbeta,m(2)+(mdot(1)
-v*cos(theta))/dbeta];
phat=A1*A2*[cos(beta);sin(beta)];
end
```

3) Il faudrait calculer l'expression de $\ddot{\alpha}$ pour avoir l'équation manquante.

4) L'idée est de mettre le problème sous la forme d'un problème d'estimation d'état linéaire. Pour l'équation d'observation, on a :

$$\begin{aligned} (x_m - x) \sin(\theta + \alpha) - (y_m - y) \cos(\theta + \alpha) &= 0 \\ \Leftrightarrow \underbrace{x_m \sin(\theta + \alpha) - y_m \cos(\theta + \alpha)}_{=z : \text{mesuré}} &= (\sin(\theta + \alpha) - \cos(\theta + \alpha)) \begin{pmatrix} x \\ y \end{pmatrix} \end{aligned}$$

Et donc l'équation d'état qui décrit notre système est :

$$\begin{cases} \begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{v} \end{pmatrix} = \begin{pmatrix} 0 & 0 & \cos \theta \\ 0 & 0 & \sin \theta \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \\ v \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 1 \end{pmatrix} \mathbf{u} \\ z = (\sin(\theta + \alpha) - \cos(\theta + \alpha)) \begin{pmatrix} x \\ y \\ v \end{pmatrix} \end{cases}$$

On peut donc appliquer un filtre de Kalman (voir chapitre 7).

5) Nous avons deux nouvelles équations disponibles pour \mathcal{R}_b :

$$\begin{cases} (x_m - x_b) \sin(\theta_b + \alpha_b) - (y_m - y_b) \cos(\theta_b + \alpha_b) = 0 \\ (x_a - x_b) \sin(\theta_b + \beta_b) - (y_a - y_b) \cos(\theta_b + \beta_b) = 0 \end{cases}$$

qui nous permettent de retrouver (x_b, y_b) :

$$\begin{pmatrix} x_b \\ y_b \end{pmatrix} = \begin{pmatrix} -\sin(\theta_b + \alpha_b) \cos(\theta_b + \alpha_b) \\ -\sin(\theta_b + \beta_b) \cos(\theta_b + \beta_b) \end{pmatrix}^{-1} \begin{pmatrix} y_m \cos(\theta_b + \alpha_b) - x_m \sin(\theta_b + \alpha_b) \\ y_a \cos(\theta_b + \beta_b) - x_a \sin(\theta_b + \beta_b) \end{pmatrix}$$

6) Nous avons une nouvelle équation :

$$(x_a - x_b) \sin(\theta_a + \beta_a) - (y_a - y_b) \cos(\theta_a + \beta_a) = 0$$

où β_a est l'angle avec lequel \mathcal{R}_a voit \mathcal{R}_b . On a donc cinq équations à quatre inconnues :

$$\begin{aligned} (x_m - x_a) \sin(\theta_a + \alpha_a) - (y_m - y_a) \cos(\theta_a + \alpha_a) &= 0 \\ (\dot{x}_m - v_a \cos \theta_a) \sin(\theta_a + \alpha_a) + (x_m - x_a) (\dot{\theta}_a + \dot{\alpha}_a) \cos(\theta_a + \alpha_a) & \\ - (\dot{y}_m - v_a \sin \theta_a) \cos(\theta_a + \alpha_a) + (y_m - y_a) (\dot{\theta}_a + \dot{\alpha}_a) \sin(\theta_a + \alpha_a) &= 0 \\ (x_m - x_b) \sin(\theta_b + \alpha_b) - (y_m - y_b) \cos(\theta_b + \alpha_b) &= 0 \\ (x_a - x_b) \sin(\theta_b + \beta_b) - (y_a - y_b) \cos(\theta_b + \beta_b) &= 0 \\ (x_a - x_b) \sin(\theta_a + \beta_a) - (y_a - y_b) \cos(\theta_a + \beta_a) &= 0 \end{aligned}$$

Ces équations sont linéaires en (x_a, y_a, x_b, y_b) et on peut donc les résoudre par la formule des moindres-carrés.

Correction de l'exercice 5.4 (localisation par mesure de distance)

1) Il existe deux solutions : celle déjà représentée sur la figure 5.10 et sa symétrique par rapport à la droite passant par \mathbf{m} et de vecteur directeur $\mathbf{u} = (\cos \theta, \sin \theta)$.

2) On a :

$$\begin{pmatrix} x_1 \\ y_1 \end{pmatrix} = \begin{pmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{pmatrix} \begin{pmatrix} x - x_m \\ y - y_m \end{pmatrix}$$

3) On a :

$$(x - x_m)^2 + (y - y_m)^2 = d^2$$

En dérivant, on obtient $2\dot{x}(x - x_m) + 2\dot{y}(y - y_m) = 2d\dot{d}$, c'est-à-dire :

$$(v \cos \theta)(x - x_m) + (v \sin \theta)(y - y_m) = d\dot{d}$$

Dans le repère \mathcal{R}_1 ces deux équations s'écrivent :

$$\begin{cases} x_1^2 + y_1^2 = d^2 \\ vx_1 = d\dot{d} \end{cases}$$

La résolution de ce système est aisée. On obtient :

$$\begin{cases} x_1 = \frac{d\dot{d}}{v} \\ y_1 = \varepsilon d \sqrt{1 - \frac{\dot{d}^2}{v^2}} \end{cases}$$

avec $\varepsilon = \pm 1$. On retrouve bien deux solutions symétriques.

4) L'équation de localisation est :

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{pmatrix} \left(\varepsilon d \sqrt{1 - \frac{\dot{d}^2}{v^2}} \right) + \begin{pmatrix} x_m \\ y_m \end{pmatrix}$$

Si $v = 0$ on ne pourra pas se localiser. On a donc une singularité. On a une solution unique si $1 - \frac{\dot{d}^2}{v^2} = 0$, c'est-à-dire si $\dot{d} = \pm v$, ce qui signifie que le robot se dirige dans la direction de l'amer ou dans le sens opposé. Enfin, nous n'aurons jamais $1 - \frac{\dot{d}^2}{v^2} < 0$ (sauf s'il existe un problème dans les capteurs) car cela signifierait une absence de solution. Cette condition est logique car on ne peut se rapprocher de l'amer plus vite que sa propre vitesse.

Chapitre 6

Identification

Le but de l'identification est d'estimer, avec une certaine précision, des quantités non mesurées à partir d'autres grandeurs, qui elles, sont mesurées. Dans le cas particulier, où la quantité à estimer est le vecteur d'état d'un système linéaire invariant, les observateurs d'état par placement de pôles (ou de Luenberger) peuvent être considérés comme un outil efficace pour l'identification. Dans ce chapitre, nous allons présenter quelques notions de base sur l'estimation, dans le but d'introduire le filtrage de Kalman au chapitre suivant. Rapidement, ce filtrage peut être vu comme un observateur d'état pour des systèmes linéaires dynamique à coefficients variant dans le temps. Mais, contrairement aux observateurs plus classiques utilisant une approche par placement de pôles, le filtrage de Kalman utilise les propriétés probabilistes des signaux. Ici, nous allons considérer le cas statique (par opposition à dynamique). Les inconnues à estimer sont toutes rangées dans un vecteur de paramètres \mathbf{p} et les mesures sont rangées dans un vecteur de mesures \mathbf{y} . Pour effectuer cette estimation, nous allons principalement nous intéresser à l'approche dite des *moindres-carrés* qui cherche à trouver le vecteur \mathbf{p} qui minimise la somme des carrés des erreurs.

6.1. Fonctions quadratiques

Dans le cas où la dépendance entre les vecteurs \mathbf{p} et \mathbf{y} est linéaire, la méthode des moindres-carrés revient à minimiser une fonction quadratique. Ce paragraphe rappelle quelques notions attachées à ces fonctions d'une nature particulière.

6.1.1. Définition

Une fonction quadratique $f : \mathbb{R}^n \rightarrow \mathbb{R}$ est une fonction de la forme :

$$f(\mathbf{x}) = \mathbf{x}^T \cdot \mathbf{Q} \cdot \mathbf{x} + \mathbf{L} \mathbf{x} + c$$

où \mathbf{Q} est une matrice symétrique. Cette définition est équivalente à dire que $f(\mathbf{x})$ est une combinaison linéaire d'une constante c , des x_i , de leur carré x_i^2 et des produits croisés $x_i x_j$ où $i \neq j$. Par exemple, la fonction $f(x_1, x_2) = 2x_1^2 - 6x_1x_2 + x_2^2 - 2x_1 + x_2 + 1$ est une fonction quadratique. On a :

$$f(\mathbf{x}) = (x_1 \ x_2) \begin{pmatrix} 2 & -3 \\ -3 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + (-2 \ 1) \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + 1 \quad [6.1]$$

Nous allons montrer dans la suite que la fonction dérivée de f au point \mathbf{x} est une fonction affine. Pour notre exemple, la dérivée de f au point \mathbf{x} est donnée par :

$$\frac{df}{d\mathbf{x}}(\mathbf{x}) = \begin{pmatrix} \frac{\partial f}{\partial x_1}(\mathbf{x}) & \frac{\partial f}{\partial x_2}(\mathbf{x}) \end{pmatrix}$$

avec $\frac{\partial f}{\partial x_1}(\mathbf{x}) = 4x_1 - 6x_2 - 2$ et $\frac{\partial f}{\partial x_2}(\mathbf{x}) = -6x_1 + 2x_2 + 1$, c'est-à-dire :

$$\frac{df}{d\mathbf{x}}(\mathbf{x}) = (4x_1 - 6x_2 - 2 \ ; \ -6x_1 + 2x_2 + 1)$$

Il s'agit d'une fonction affine en \mathbf{x} . La fonction, $\mathbf{x} \mapsto \mathbf{x}^T \mathbf{Q} \mathbf{x}$ qui compose $f(\mathbf{x})$ ne comporte que des termes en $x_i x_j$ et en x_i^2 . Une telle fonction est appelée *forme quadratique*.

6.1.2. Dérivée d'une forme quadratique

Considérons la forme quadratique :

$$f(\mathbf{x}) = \mathbf{x}^T \mathbf{Q} \mathbf{x}$$

Le développement de Taylor à l'ordre 1 de f au point \mathbf{x} au voisinage de \mathbf{x} , nous donne :

$$f(\mathbf{x} + \delta\mathbf{x}) = f(\mathbf{x}) + \frac{df}{d\mathbf{x}}(\mathbf{x}) \cdot \delta\mathbf{x} + o(||\delta\mathbf{x}||)$$

où $o(||\delta\mathbf{x}||)$ signifie négligeable devant $||\delta\mathbf{x}||$, lorsque $\delta\mathbf{x}$ est infiniment petit. Bien sur, ici, $\frac{df}{d\mathbf{x}}(\mathbf{x})$ sera représentée par une matrice $1 \times n$ car, tout comme la fonction que l'on linéarise, elle va de \mathbb{R}^n vers \mathbb{R} . Or :

$$\begin{aligned} f(\mathbf{x} + \delta\mathbf{x}) &= (\mathbf{x} + \delta\mathbf{x})^T \cdot \mathbf{Q} \cdot (\mathbf{x} + \delta\mathbf{x}) \\ &= \mathbf{x}^T \cdot \mathbf{Q} \cdot \mathbf{x} + \mathbf{x}^T \cdot \mathbf{Q} \cdot \delta\mathbf{x} + \delta\mathbf{x}^T \cdot \mathbf{Q} \cdot \mathbf{x} + \delta\mathbf{x}^T \cdot \mathbf{Q} \cdot \delta\mathbf{x} \\ &= \mathbf{x}^T \cdot \mathbf{Q} \cdot \mathbf{x} + 2\mathbf{x}^T \cdot \mathbf{Q} \cdot \delta\mathbf{x} + o(||\delta\mathbf{x}||) \end{aligned}$$

car \mathbf{Q} est symétrique et $\delta\mathbf{x}^T \cdot \mathbf{Q} \cdot \delta\mathbf{x} = o(||\delta\mathbf{x}||)$. Par unicité du développement de Taylor, et d'après les expressions pour $f(\mathbf{x} + \delta\mathbf{x})$, nous avons :

$$\frac{df}{d\mathbf{x}}(\mathbf{x}) = \left(\frac{\partial f}{\partial x_1}(\mathbf{x}), \dots, \frac{\partial f}{\partial x_n}(\mathbf{x}) \right) = 2\mathbf{x}^T \mathbf{Q}$$

Par exemple, la dérivée de la fonction quadratique [6.1] est :

$$2(x_1 \ x_2) \begin{pmatrix} 2 & -3 \\ -3 & 1 \end{pmatrix} + (-2 \ 1) = (4x_1 - 6x_2 - 2 \ -6x_1 + 2x_2 + 1)$$

6.1.3. Valeurs propres d'une fonction quadratique

Ce sont les valeurs propres de \mathbf{Q} . Les valeurs propres sont toutes réelles et les vecteurs propres sont tous deux à deux orthogonaux. Les courbes de niveaux d'une fonction quadratique $f(\mathbf{x}) = \alpha$ sont de la forme :

$$\mathbf{x}^T \mathbf{Q} \mathbf{x} + \mathbf{L} \mathbf{x} = \alpha - c$$

et sont appelées *quadriques*. Ce sont des ellipsoïdes si toutes les valeurs propres sont de même signe ou des hyperboloïdes si elles ont des signes distincts. Si toutes les valeurs propres de \mathbf{Q} sont positives, on dit que la forme quadratique $\mathbf{x}^T \mathbf{Q} \mathbf{x}$ est positive. Si elles sont toutes non nulles, on dit que la forme quadratique est définie. Si elles sont toutes strictement positives, on dira que la forme quadratique est définie positive. La fonction quadratique f admet un et un seul minimum si et seulement si sa forme quadratique associée est définie positive.

6.1.4. Minimisation d'une fonction quadratique

Si $f(\mathbf{x}) = \mathbf{x}^T \mathbf{Q} \mathbf{x} + \mathbf{L} \mathbf{x} + c$ admet un et un seul minimiseur \mathbf{x}^* si \mathbf{Q} est définie positive. Dans ce cas :

$$\frac{df}{d\mathbf{x}}(\mathbf{x}^*) = \mathbf{0}$$

Donc $2\mathbf{x}^{*T} \mathbf{Q} + \mathbf{L} = \mathbf{0}$, c'est-à-dire :

$$\mathbf{x}^* = -\frac{1}{2} \mathbf{Q}^{-1} \mathbf{L}^T$$

Son minimum est donné par :

$$\begin{aligned} f(\mathbf{x}^*) &= \left(-\frac{1}{2} \mathbf{Q}^{-1} \mathbf{L}^T \right)^T \mathbf{Q} \left(-\frac{1}{2} \mathbf{Q}^{-1} \mathbf{L}^T \right) + \mathbf{L} \left(-\frac{1}{2} \mathbf{Q}^{-1} \mathbf{L}^T \right) + c \\ &= \frac{1}{4} \mathbf{L} \mathbf{Q}^{-1} \mathbf{L}^T - \frac{1}{2} \mathbf{L} \mathbf{Q}^{-1} \mathbf{L}^T + c \\ &= -\frac{1}{4} \mathbf{L} \mathbf{Q}^{-1} \mathbf{L}^T + c \end{aligned}$$

EXEMPLE 6.1.– *La fonction quadratique :*

$$f(\mathbf{x}) = (x_1 \ x_2) \begin{pmatrix} 2 & -1 \\ -1 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + (3 \ 4) \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + 5$$

admet un minimum car la matrice de sa forme quadratique \mathbf{Q} est définie positive (ses valeurs propres $\frac{3}{2} \pm \frac{1}{2}\sqrt{5}$ sont toutes les deux positives). La fonction possède pour minimiseur le vecteur :

$$\mathbf{x}^* = -\frac{1}{2} \begin{pmatrix} 2 & -1 \\ -1 & 1 \end{pmatrix}^{-1} \begin{pmatrix} 3 \\ 4 \end{pmatrix} = \begin{pmatrix} -\frac{7}{2} \\ -\frac{11}{2} \end{pmatrix}$$

Son minimum est :

$$f(\mathbf{x}^*) = -\frac{1}{4} (3 \ 4) \begin{pmatrix} 2 & -1 \\ -1 & 1 \end{pmatrix}^{-1} \begin{pmatrix} 3 \\ 4 \end{pmatrix} + 5 = -\frac{45}{4}$$

EXEMPLE 6.2.– *La fonction $f(x) = 3x^2 + 6x + 7$ admet un minimum car la matrice de sa forme quadratique (qui correspond ici au scalaire 3) est définie positive (car $3 > 0$). Son minimiseur est le scalaire :*

$$x^* = -\frac{1}{2} \cdot \frac{1}{3} \cdot 6 = -1$$

et son minimum est $f(x^*) = 3 - 6 + 7 = 4$.

6.2. Méthode des moindres-carrés

Estimer, c'est obtenir un ordre de grandeur sur certaines quantités d'un système à partir de mesures d'autres quantités de ce même système. Le problème d'estimation que nous allons considérer dans ce chapitre est le suivant. Soit un système sur lequel on a effectué des mesures $\mathbf{y} = (y_1, \dots, y_p)$ et un modèle $\mathcal{M}(\mathbf{p})$ dépendant d'un vecteur de paramètres \mathbf{p} . Il nous faut estimer \mathbf{p} tel que les sorties $\mathbf{f}(\mathbf{p})$ générées par $\mathcal{M}(\mathbf{p})$ ressemblent le plus possible à \mathbf{y} .

6.2.1. Cas linéaire

Supposons que le vecteur des sorties puisse se mettre sous la forme :

$$\mathbf{f}(\mathbf{p}) = \mathbf{M}\mathbf{p}$$

Le modèle est alors qualifié de *linéaire par rapport aux paramètres*. Nous voudrions avoir :

$$\mathbf{f}(\mathbf{p}) = \mathbf{y}$$

mais cela n'est généralement pas possible à cause de la présence du bruit et du fait que le nombre de mesures est généralement supérieur au nombre de paramètres (c'est-à-dire $\dim(\mathbf{y}) > \dim(\mathbf{p})$). Nous allons donc chercher le meilleur \mathbf{p} , c'est-à-dire celui qui minimise le critère, dit des *moindres-carrés* :

$$j(\mathbf{p}) = \|\mathbf{f}(\mathbf{p}) - \mathbf{y}\|^2$$

Nous avons :

$$\begin{aligned} j(\mathbf{p}) &= \|\mathbf{f}(\mathbf{p}) - \mathbf{y}\|^2 = \|\mathbf{M}\mathbf{p} - \mathbf{y}\|^2 \\ &= (\mathbf{M}\mathbf{p} - \mathbf{y})^T (\mathbf{M}\mathbf{p} - \mathbf{y}) = (\mathbf{p}^T \mathbf{M}^T - \mathbf{y}^T) (\mathbf{M}\mathbf{p} - \mathbf{y}) \\ &= \mathbf{p}^T \mathbf{M}^T \mathbf{M}\mathbf{p} - \mathbf{p}^T \mathbf{M}^T \mathbf{y} - \mathbf{y}^T \mathbf{M}\mathbf{p} + \mathbf{y}^T \mathbf{y} \\ &= \mathbf{p}^T \mathbf{M}^T \mathbf{M}\mathbf{p} - 2\mathbf{y}^T \mathbf{M}\mathbf{p} + \mathbf{y}^T \mathbf{y} \end{aligned}$$

Or $\mathbf{M}^T \mathbf{M}$ est symétrique (car $(\mathbf{M}^T \mathbf{M})^T = \mathbf{M}^T \mathbf{M}$). On a donc une fonction quadratique. De plus, $\mathbf{M}^T \mathbf{M}$ a toutes ses valeurs propres positives ou nulles. Le minimiseur $\hat{\mathbf{p}}$ s'obtient comme suit :

$$\begin{aligned} \frac{dj}{d\mathbf{p}}(\hat{\mathbf{p}}) = \mathbf{0} &\Leftrightarrow 2\hat{\mathbf{p}}^T \mathbf{M}^T \mathbf{M} - 2\mathbf{y}^T \mathbf{M} = \mathbf{0} \Leftrightarrow \hat{\mathbf{p}}^T \mathbf{M}^T \mathbf{M} = \mathbf{y}^T \mathbf{M} \\ &\Leftrightarrow \mathbf{M}^T \mathbf{M} \hat{\mathbf{p}} = \mathbf{M}^T \mathbf{y} \quad \Leftrightarrow \hat{\mathbf{p}} = (\mathbf{M}^T \mathbf{M})^{-1} \mathbf{M}^T \mathbf{y} \end{aligned}$$

La matrice :

$$\mathbf{K} = (\mathbf{M}^T \mathbf{M})^{-1} \mathbf{M}^T$$

s'appelle *inverse généralisée* de la matrice rectangulaire \mathbf{M} . Le vecteur $\hat{\mathbf{p}}$ est appelé *l'estimé* au sens des moindres-carrés. La fonction :

$$\mathbf{y} \mapsto \mathbf{K}\mathbf{y}$$

est appelée *estimateur*. Notons que cet estimateur est linéaire car la fonction modèle f est, elle aussi, linéaire. Le vecteur :

$$\hat{\mathbf{y}} = \mathbf{M}\hat{\mathbf{p}} = \mathbf{M}\mathbf{K}\mathbf{y}$$

est le vecteur des *mesures filtrées* et la quantité :

$$\mathbf{r} = \hat{\mathbf{y}} - \mathbf{y} = (\mathbf{M}\mathbf{K} - \mathbf{I})\mathbf{y}$$

est appelée *vecteur des résidus*. La norme de ce vecteur représente la distance entre y et l'hyper plan $\mathbf{f}(\mathbb{R}^n)$. Si cette norme est grande, c'est souvent que l'on a une erreur de modèle ou des données aberrantes.

6.2.2. Cas non linéaire

Si \mathbf{y} est le vecteur des mesures et si $\mathbf{f}(\mathbf{p})$ est la sortie générée par le modèle, alors l'estimée au sens des moindres-carrés est définie par :

$$\hat{\mathbf{p}} = \arg \min_{\mathbf{p} \in \mathbb{R}^n} \|\mathbf{f}(\mathbf{p}) - \mathbf{y}\|^2$$

Lorsque $\mathbf{f}(\mathbf{p})$ est linéaire par rapport à \mathbf{p} , c'est-à-dire $\mathbf{f}(\mathbf{p}) = \mathbf{M}\mathbf{p}$ alors le vecteur des paramètres $\hat{\mathbf{p}}$ estimé au sens des moindres-carrés est $\hat{\mathbf{p}} = (\mathbf{M}^T \mathbf{M})^{-1} \mathbf{M} \mathbf{y}$ et le vecteur des mesures filtrées est $\hat{\mathbf{y}} = \mathbf{M} (\mathbf{M}^T \mathbf{M})^{-1} \mathbf{M} \mathbf{y}$. En général, et même lorsque $\mathbf{f}(\mathbf{p})$ est non linéaire, on peut faire l'interprétation géométrique suivante (voir figure 6.1) :

- le vecteur des mesures filtrées $\hat{\mathbf{y}}$ représente la projection de \mathbf{y} sur l'ensemble $\mathbf{f}(\mathbb{R}^n)$;
- le vecteur estimé au sens des moindres-carrés $\hat{\mathbf{p}}$ représente l'image inverse par $\mathbf{f}(\cdot)$ du vecteur des mesures filtrées $\hat{\mathbf{y}}$.

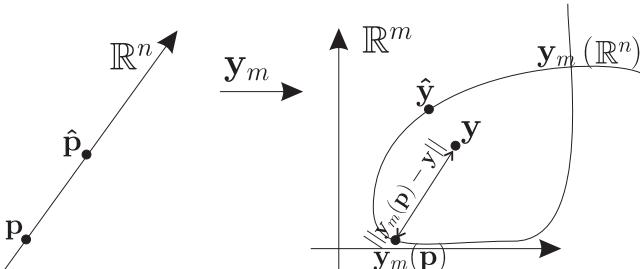


Figure 6.1. Illustration de la méthode des moindres-carrés dans le cas non linéaire

Lorsque $\mathbf{f}(\mathbf{p})$ est non linéaire, on peut utiliser un algorithme d'optimisation local pour espérer obtenir $\hat{\mathbf{p}}$. L'algorithme ci-après propose une version simple d'une telle méthode optimisation :

Algorithme MINIMISE(input : p)
1 $j^+ = j(\mathbf{p})$
2 Prendre aléatoirement un déplacement δ
3 $\mathbf{q} = \mathbf{p} + \delta$
4 if $j(\mathbf{q}) < j^+$ then $\{\mathbf{p} = \mathbf{q}; j^+ = j(\mathbf{q})\}$;
5 go to 2

Cet algorithme converge vers un optimum local du critère $j(\mathbf{p}) = \|\mathbf{f}(\mathbf{p}) - \mathbf{y}\|^2$. La quantité δ est le pas qui représente un petit vecteur tiré aléatoirement dans \mathbb{R}^n . Dans le cas de la méthode du *recuit simulé*, le pas diminue au fur et à mesure des itérations en fonction d'un paramètre appelé *température* qui décroît dans le temps.

6.3. Exercices

Exercice 6.1. Représentation d'une fonction quadratique

On considère la fonction quadratique $f(x, y) = x \cdot y$.

- 1) Donner le gradient de f au point (x_0, y_0) .
- 2) Mettre f sous la forme $(x \ y) \cdot \mathbf{Q} \cdot (x \ y)^T + \mathbf{L} (x \ y)^T + c$, où \mathbf{Q} est une matrice symétrique. Vérifier que le gradient trouvé au 1) est donné par $2 (x \ y) \mathbf{Q}$. Tracer sous MATLAB le champ de vecteurs associé à ce gradient à l'aide de l'instruction `quiver`. Interpréter.
- 3) En utilisant l'instruction `contour` de MATLAB, tracer les courbes de niveaux de f puis tracer le graphe de f . La fonction f admet-elle un minimum ?
- 4) Reprendre cet exercice avec la fonction $g(x, y) = 2x^2 + xy + 4y^2 + y - x + 3$.

Exercice 6.2. Identification d'une parabole

On cherche à trouver une parabole $p_1 t^2 + p_2 t + p_3$ qui passe par n points donnés par :

t	-3	-1	0	2	3	6
y	17	3	1	5	11	46

- 1) Donner au sens des moindres-carrés une estimation des paramètres p_1, p_2, p_3 .
- 2) Quelles sont les mesures filtrées correspondantes ? Donner le vecteur des résidus.

Exercice 6.3. Identification des paramètres d'un moteur à courant continu

La vitesse angulaire Ω d'un moteur à courant continu en régime permanent dépend linéairement de la tension d'alimentation U et du couple résistant T_r :

$$\Omega = p_1 U + p_2 T_r$$

On effectue une série d'expériences sur un moteur particulier. On mesure :

U (V)	4	10	10	13	15
T_r (Nm)	0	1	5	5	3
Ω (rad/sec)	5	10	8	14	17

- 1) Donner au sens des moindres-carrés une estimation des paramètres p_1, p_2 . Donner les mesures filtrées et le vecteur des résidus correspondants.

2) En déduire une estimation de la vitesse angulaire du moteur $U = 20 \text{ V}$ et $T_r = 10 \text{ Nm}$.

Exercice 6.4. Estimation d'une fonction de transfert

On considère le système décrit par les équations de récurrence :

$$y(k) + a_1 y(k-1) + a_0 y(k-2) = b_1 u(k-1) + b_0 u(k-2)$$

Sur ce système, on effectue des mesures bruitées de l'entrée $u(k)$ et la sortie $y(k)$ pour k variant de 0 à 7. On obtient :

k	0	1	2	3	4	5	6	7
$u(k)$	1	-1	1	-1	1	-1	1	-1
$y(k)$	0	-1	-2	3	7	11	16	36

Estimer le vecteur des paramètres $\mathbf{p} = (a_1, a_0, b_1, b_0)$ par la méthode des moindres-carrés. Conclure.

Exercice 6.5. Méthode de Monté-Carlo

On considère le système à temps discret donné par sa représentation d'état :

$$\begin{cases} \mathbf{x}(k+1) = \begin{pmatrix} 1 & 0 \\ a & 0.9 \end{pmatrix} \mathbf{x}(k) + \begin{pmatrix} b \\ 1-b \end{pmatrix} u(k) \\ y(k) = \begin{pmatrix} 1 & 1 \end{pmatrix} \mathbf{x}(k) \end{cases}$$

où a, b sont deux paramètres à estimer. L'état initial est donné par $\mathbf{x}(0) = (0, 0)$ et $u(k) = 1$. On recueille six mesures :

$$(y(1), \dots, y(6)) = (0 \ 1 \ 2.65 \ 4.885 \ 7.646 \ 10.882)$$

Notons que ces valeurs ont été obtenues pour les valeurs $a^* = 0.9$ et $b^* = 0.75$, mais nous ne sommes pas censés le savoir. Nous allons juste supposer que $a \in [0, 2]$ et $b \in [0, 2]$.

1) Proposer un programme MATLAB qui estime les paramètres a et b par une méthode de Monté-Carlo. Pour cela, nous allons générer par tirage aléatoire uniforme un nuage de vecteurs $\mathbf{p} = (a, b)$. Puis, par simulation des équations d'état, nous allons calculer pour tous les \mathbf{p} , les sorties $y_m(\mathbf{p}, k)$ correspondantes. Nous allons alors afficher sur l'écran les vecteurs \mathbf{p} tels que pour chaque $k \in \{1, \dots, 6\}$, $|y_m(k) - y(k)| < \varepsilon$, où ε est un petit nombre positif.

2) Calculer en fonction de a et b la fonction de transfert du système.

3) Supposons que les vraies valeurs $a^* = 0.9$ et $b^* = 0.75$ pour a et b sont connues. Calculer l'ensemble de tous les couples (a, b) qui génèrent la même fonction

de transfert que le couple (a^*, b^*) . En déduire une interprétation des résultats obtenus à la question 1).

Exercice 6.6. Localisation par recuit simulé

Le problème de localisation que nous allons considérer est inspiré de [KIE 01]. Le robot, représenté sur la figure 6.2, est équipé de huit télémètres laser capables de mesurer sa distance aux parois pour des angles égaux à $\frac{k\pi}{4}$, $k \in \{0, \dots, 7\}$. Nous supposons que les obstacles sont constitués de n segments $[\mathbf{a}_i \mathbf{b}_i]$, $i = 1, \dots, n$, où les coordonnées de \mathbf{a}_i et \mathbf{b}_i sont connues. Les huit distances sont rangées dans le vecteur \mathbf{y} et le problème de localisation revient à estimer la position et l'orientation du robot à partir de \mathbf{y} .

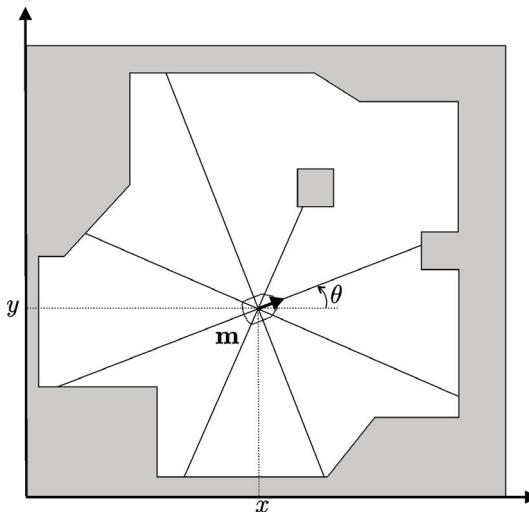


Figure 6.2. Robot équipé de huit télémètres cherchant à se localiser

1) Soient \mathbf{m} , \mathbf{a} , \mathbf{b} trois points de \mathbb{R}^2 et \vec{u} un vecteur unitaire. Démontrer que la demi-droite $\mathcal{E}(\mathbf{m}, \vec{u})$ intersecte le segment $[\mathbf{a}\mathbf{b}]$ si et seulement si :

$$\begin{cases} \det(\mathbf{a} - \mathbf{m}, \vec{u}) \cdot \det(\mathbf{b} - \mathbf{m}, \vec{u}) \leq 0 \\ \det(\mathbf{a} - \mathbf{m}, \mathbf{b} - \mathbf{a}) \cdot \det(\vec{u}, \mathbf{b} - \mathbf{a}) \geq 0 \end{cases}$$

Si cette condition est vérifiée, montrer que la distance de \mathbf{m} à $[\mathbf{a}\mathbf{b}]$ suivant \vec{u} est :

$$d = \frac{\det(\mathbf{a} - \mathbf{m}, \mathbf{b} - \mathbf{a})}{\det(\vec{u}, \mathbf{b} - \mathbf{a})}$$

2) Développer un simulateur $\mathbf{f}(\mathbf{p})$ calculant les distances directionnelles entre la pose $\mathbf{p} = (x, y, \theta)$ et les murs.

3) En utilisant une méthode d'optimisation globale du type `recuit simulé`, concevoir un programme MATLAB qui donne une estimation $\hat{\mathbf{p}}$ au sens des moindres-carrés de la pose \mathbf{p} à partir de \mathbf{y} . Pour les segments $[\mathbf{a}_i, \mathbf{b}_i]$ de la pièce et pour le vecteur des distances mesurées, on prendra les quantités qui suivent :

```
A=[0 7 7 9 9 7 7 4 2 0 5 6 6 5; 0 0 2 2 4 4 7 7 5 5 2 2 3 3];
B=[7 7 9 9 7 7 4 2 0 0 6 6 5 5; 0 2 2 4 4 7 7 5 5 0 2 3 3 2];
y=[6.4 ;3.6 ;2.3 ;2.1 ;1.7 ;1.6 ;3.0 ;3.1];
```

6.4. Corrections

Correction de l'exercice 6.1 (représentation d'une fonction quadratique)

1) Le gradient de $f(x, y) = x \cdot y$. au point (x_0, y_0) est :

$$\frac{df}{d(x, y)}(x_0, y_0) = (y_0 \ x_0)$$

2) On a :

$$f(x, y) = (x \ y) \begin{pmatrix} 0 & \frac{1}{2} \\ \frac{1}{2} & 0 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

Le gradient est donné par :

$$\frac{df}{d(x, y)}(x, y) = 2(x \ y) \mathbf{Q} = 2(x \ y) \begin{pmatrix} 0 & \frac{1}{2} \\ \frac{1}{2} & 0 \end{pmatrix} = (y \ x)$$

On retrouve bien le résultat trouvé au 1). Le champ de vecteurs associé à ce gradient s'obtient en tapant :

```
Mx = -1 :0.1 :1; My = -1 :0.1 :1;
[X, Y] = meshgrid(Mx, My); GX=Y; GY=X;
quiver(Mx, My, GX, GY);
```

On obtient la figure 6.3 à gauche.

3) On forme la fonction f en tapant $Z = X.*Y$. Le contour s'obtient en tapant `contour3(X, Y, Z, 20)` puis le graphe de f par `surface(X, Y, Z)`. On obtient respectivement la figure du centre et celle de droite. La fonction n'admet pas de minimum. Les instructions associées à cette question sont rangées dans le script `quadra.m`.

4) Pour la fonction $g(x, y) = 2x^2 + xy + 4y^2 + y - x + 3$, on obtient :

$$g(x, y) = (x \ y) \begin{pmatrix} 2 & \frac{1}{2} \\ \frac{1}{2} & 4 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + (-1 \ 1) \begin{pmatrix} x \\ y \end{pmatrix} + 3$$

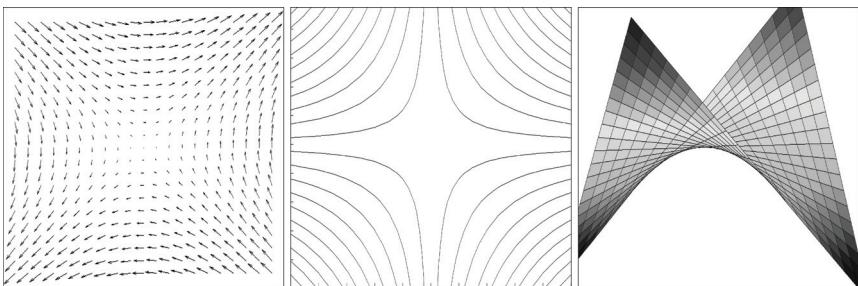


Figure 6.3. Représentation de la fonction $x \cdot y$ sur le pavé $[-1, 1]^2$;
 à gauche : champ de vecteur du gradient ; au centre : isocontours de la fonction ;
 à droite : vue 3D du graphe

Puisque les valeurs propres $3 + \frac{1}{2}\sqrt{5}, 3 - \frac{1}{2}\sqrt{5}$ de la matrice sont strictement positives, la forme quadratique associée à g est définie positive et donc f admet un minimum. Pour le calculer, il faut résoudre :

$$\frac{dg}{d(x, y)} = 2(x \ y) \begin{pmatrix} 2 & \frac{1}{2} \\ \frac{1}{2} & 4 \end{pmatrix} + (-1 \ 1) = (0 \ 0)$$

On obtient :

$$\begin{pmatrix} x \\ y \end{pmatrix} = -\frac{1}{2} \begin{pmatrix} 2 & \frac{1}{2} \\ \frac{1}{2} & 4 \end{pmatrix}^{-1} \begin{pmatrix} -1 \\ 1 \end{pmatrix} = \frac{1}{31} \cdot \begin{pmatrix} 9 \\ -5 \end{pmatrix}$$

En traçant les graphiques de la même façon que pour f (voir figure 6.4) on peut vérifier que $g(x, y)$ admet un minimum et que les courbes de niveaux sont des ellipses.

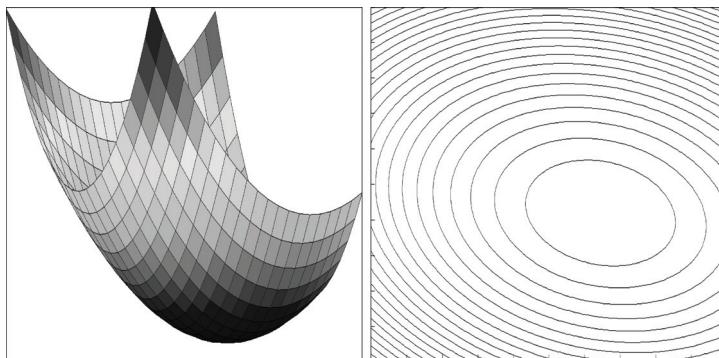


Figure 6.4. A gauche : graphe de la fonction $g(x, y)$; à droite : isocontours.
 L'espace de recherche pour x et y correspond à l'intervalle $[-1, 1]$

Correction de l'exercice 6.2 (identification d'une parabole)

REMARQUE 6.1.– Notons tout d'abord que pour obtenir ces mesures, nous avons pris $p_1^* = \sqrt{2}$, $p_2^* = -1$, $p_3^* = 1$, pour en déduire les mesures non bruitées :

$$\mathbf{y}^* = (16.72, 3.41, 1, 4.65, 10.73, 45.91)$$

Ensuite, nous avons tronqué à l'entier le plus proche. La ligne d'instruction sous MATLAB est donnée par :

```
t=[-3 ; -1 ; 0 ; 2 ; 3 ; 6], y1=sqrt(2)*t.^2-t+1, y=round(y1)
```

Le vecteur des mesures est donc $\mathbf{y} = (17, 3, 1, 5, 11, 46)$, comme donné dans l'énoncé, et le vecteur des paramètres à estimer est $\mathbf{p} = (p_1, p_2, p_3)$. Bien évidemment ce processus de génération des données n'est pas connu et nous ne sommes pas censés l'utiliser lors de la résolution de l'exercice.

1) La sortie modèle est :

$$\mathbf{f}(\mathbf{p}) = \begin{pmatrix} f_1(\mathbf{p}) \\ f_2(\mathbf{p}) \\ f_3(\mathbf{p}) \\ f_4(\mathbf{p}) \\ f_5(\mathbf{p}) \\ f_6(\mathbf{p}) \end{pmatrix} = \begin{pmatrix} 9p_1 - 3p_2 + p_3 \\ p_1 - p_2 + p_3 \\ 0p_1 - 0p_2 + p_3 \\ 4p_1 + 2p_2 + p_3 \\ 9p_1 + 3p_2 + p_3 \\ 36p_1 + 6p_2 + p_3 \end{pmatrix} = \begin{pmatrix} 9 & -3 & 1 \\ 1 & -1 & 1 \\ 0 & 0 & 1 \\ 4 & 2 & 1 \\ 9 & 3 & 1 \\ 36 & 6 & 1 \end{pmatrix} \begin{pmatrix} p_1 \\ p_2 \\ p_3 \end{pmatrix}$$

Le vecteur estimé au sens des moindres-carrés est donc :

$$\hat{\mathbf{p}} = (\mathbf{M}^T \mathbf{M})^{-1} \mathbf{M}^T \mathbf{y} = \begin{pmatrix} 1.41 \\ -0.98 \\ 1.06 \end{pmatrix}$$

2) Les mesures filtrées sont :

$$\hat{\mathbf{y}} = \mathbf{f}(\hat{\mathbf{p}}) = \mathbf{M} \hat{\mathbf{p}} = (16.76, 3.46, 1.06, 4.76, 10.84, 46.11)$$

et le vecteur des résidus est :

$$\mathbf{r} = \hat{\mathbf{y}} - \mathbf{y} = (-0.24, 0.46, 0.06, -0.24, -0.15, 0.11)$$

Le script MATLAB correspondant à cet exercice est rangé dans le fichier `parab.m`.

Correction de l'exercice 6.3 (identification des paramètres d'un moteur à courant continu)

1) On a :

$$\mathbf{f}(\mathbf{p}) = \mathbf{M} \cdot \mathbf{p}$$

avec :

$$\mathbf{M} = \begin{pmatrix} 4 & 0 \\ 10 & 1 \\ 10 & 5 \\ 13 & 5 \\ 15 & 3 \end{pmatrix}, \mathbf{p} = \begin{pmatrix} p_1 \\ p_2 \end{pmatrix} \text{ et } \mathbf{y} = \begin{pmatrix} 5 \\ 10 \\ 8 \\ 14 \\ 17 \end{pmatrix}$$

Donc :

$$\hat{\mathbf{p}} = (\mathbf{M}^T \mathbf{M})^{-1} \mathbf{M}^T \mathbf{y} = \begin{pmatrix} 1.188 \\ -0.516 \end{pmatrix}$$

Le vecteur des mesures filtrées est :

$$\hat{\mathbf{y}} = \mathbf{M} \cdot \hat{\mathbf{p}} = (4.75, 11.36, 9.3, 12.86, 16.27)$$

et le vecteur des résidus est :

$$\mathbf{r} = \hat{\mathbf{y}} - \mathbf{y} = (-0.25, 1.36, 1.3, -1.14, -0.73)$$

2) Pour $U = 20 \text{ V}$ et $T_r = 10 \text{ Nm}$, on a :

$$\hat{\Omega} = (U \ T_r) \cdot \hat{\mathbf{p}} = (20 \ 10) \begin{pmatrix} 1.188 \\ -0.516 \end{pmatrix} = 18.6 \text{ rad/sec}$$

Correction de l'exercice 6.4 (estimation d'une fonction de transfert)

Les équations de récurrence pour $k = 2$ jusqu'à 7 sont :

$$y(2) = -a_1 y(1) - a_0 y(0) + b_1 u(1) + b_0 u(0)$$

$$y(3) = -a_1 y(2) - a_0 y(1) + b_1 u(2) + b_0 u(1)$$

⋮

$$y(7) = -a_1 y(6) - a_0 y(5) + b_1 u(6) + b_0 u(5)$$

Donc la matrice \mathbf{M} est donnée par :

$$\mathbf{M} = \begin{pmatrix} -y(1) & -y(0) & u(1) & u(0) \\ -y(2) & -y(1) & u(2) & u(1) \\ -y(3) & -y(2) & u(3) & u(2) \\ -y(4) & -y(3) & u(4) & u(3) \\ -y(5) & -y(4) & u(5) & u(4) \\ -y(6) & -y(5) & u(6) & u(5) \end{pmatrix} = \begin{pmatrix} 1 & 0 & -1 & 1 \\ 2 & 1 & 1 & -1 \\ -3 & 2 & -1 & 1 \\ -7 & -3 & 1 & -1 \\ -11 & -7 & -1 & 1 \\ -16 & -11 & 1 & -1 \end{pmatrix} = \begin{pmatrix} \mathbf{m}_2^T \\ \mathbf{m}_3^T \\ \mathbf{m}_4^T \\ \mathbf{m}_5^T \\ \mathbf{m}_6^T \\ \mathbf{m}_7^T \end{pmatrix}$$

Le vecteur :

$$\mathbf{m}^T(k) = (-y(k-1), -y(k-2), u(k-1), u(k-2))$$

contient l'information reliant la $k^{\text{ième}}$ mesure à l'inconnue \mathbf{p} . Il est appelé *régresseur*. Comme ces k équations ne sont pas tout-à-fait satisfaites car les $u(i)$ et les $y(i)$ ne sont connus que de façon approximative, on cherche à trouver le vecteur $\hat{\mathbf{p}}$ qui minimise le critère :

$$j(\mathbf{p}) = \|\mathbf{M}\mathbf{p} - \mathbf{y}\|^2$$

avec :

$$\mathbf{y} = \begin{pmatrix} y(2) \\ \vdots \\ y(7) \end{pmatrix} = \begin{pmatrix} -2 \\ 3 \\ 7 \\ 11 \\ 16 \\ 36 \end{pmatrix} \quad \text{et} \quad \mathbf{p} = \begin{pmatrix} a_1 \\ a_0 \\ b_1 \\ b_0 \end{pmatrix}$$

Nous devrions avoir $\hat{\mathbf{p}} = (\mathbf{M}^T \mathbf{M})^{-1} \mathbf{M}^T \mathbf{y}$. Or, ici la matrice \mathbf{M} est de rang 3. En effet, du fait de la forme particulière du signal $u(k)$, les deux dernières colonnes de \mathbf{M} sont dépendantes. Dans notre cas, nous ne pouvons donc pas identifier \mathbf{p} . Cette situation demeure toutefois atypique et il faudrait choisir une autre entrée pour pouvoir rendre le problème identifiable.

Correction de l'exercice 6.5 (méthode de Monté-Carlo)

1) Pour le pavé $[0, 2] \times [0, 2]$ et pour $\varepsilon = 0.3$, l'algorithme MATLAB (rangé dans `montecarlo.m`) qui suit permet de caractériser l'ensemble de vraisemblance recherché :

```

y=[0 ;1 ;2.5 ;4.1 ;5.8 ;7.5] ;
for i=1 :10000,
a=2*rand(1) ; b=2*rand(1) ; x=[0 ;0] ; ym=0*y ;
A=[1,0 ;a,0.3] ;B=[b ;1-b] ;C=[1 1] ;
for k=1 :6, x1=A*x+B ;ym(k)=C*x ;x=x1 ; end
if norm(ym-y,inf)<0.3, plot(a,b ,'+black') ; else
plot(a,b ,'.blue') ; end ;
end

```

On obtient alors l'ensemble de solutions représenté par la figure 6.5. On remarque qu'il existe un continuum de vecteurs vraisemblables.

2) La fonction de transfert du système est :

$$(1 \ 1) \left(s\mathbf{I} - \begin{pmatrix} 1 & 0 \\ a & 0.9 \end{pmatrix} \right)^{-1} \begin{pmatrix} b \\ 1-b \end{pmatrix} = \frac{10s + b(1 + 10a) - 10}{(10s - 9)(s - 1)}$$

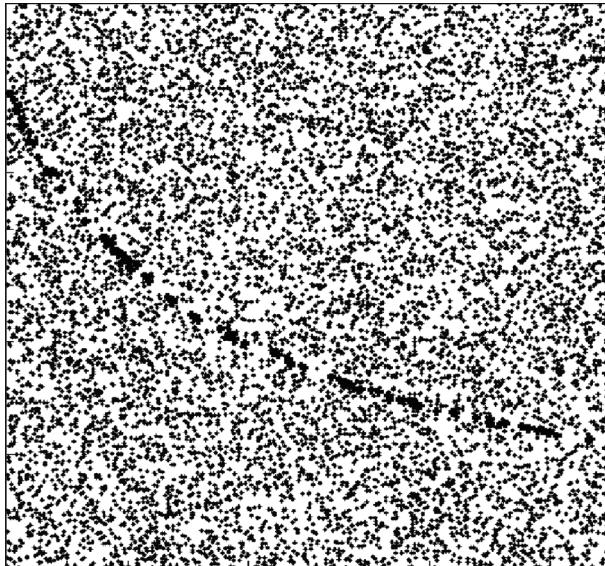


Figure 6.5. Méthode de Monté Carlo pour l'estimation des paramètres a et b

3) Si l'on sait que $a^* = 0.9$ et $b^* = 0.75$ sont les vrais paramètres, alors, pour tout couple (a, b) tel que :

$$\frac{10s + b(1 + 10a) - 10}{(10s - 9)(s - 1)} = \frac{10s + b^*(1 + 10a^*) - 10}{(10s - 9)(s - 1)}$$

on aura la même fonction de transfert. Cette condition se traduit par :

$$b(1 + 10a) = 0.75(1 + 9) = 7.5$$

ou encore :

$$b = \frac{7.5}{1 + 10a}$$

Dans de telles situations, où différentes valeurs pour le vecteur de paramètres produisent un même comportement, on dit que le modèle est *non identifiable*.

Correction de l'exercice 6.6 (localisation par recuit simulé)

1) Pour comprendre la preuve qui va suivre on rappelle que (i) $\det(\vec{u}, \vec{v}) > 0$ si \vec{v} est sur la gauche de \vec{u} , (ii) $\det(\vec{u}, \vec{v}) < 0$ si \vec{v} est sur la droite de \vec{u} et (iii) $\det(\vec{u}, \vec{v}) = 0$ si \vec{u} et \vec{v} sont colinéaires. Par exemple, sur la figure 6.6,

$\det(\mathbf{a} - \mathbf{m}, \vec{\mathbf{u}}) > 0$ et $\det(\mathbf{b} - \mathbf{m}, \vec{\mathbf{u}}) < 0$. Rappelons aussi que le déterminant est une forme multilinéaire, c'est-à-dire :

$$\begin{aligned}\det(a\mathbf{u} + b\mathbf{v}, c\mathbf{x} + d\mathbf{y}) &= a \det(\mathbf{u}, c\mathbf{x} + d\mathbf{y}) + b \det(\mathbf{v}, c\mathbf{x} + d\mathbf{y}) \\ &= ac \det(\mathbf{u}, \mathbf{x}) + bc \det(\mathbf{v}, \mathbf{x}) + ad \det(\mathbf{u}, \mathbf{y}) + bd \det(\mathbf{v}, \mathbf{y})\end{aligned}$$

PREUVE.— La ligne $\mathcal{D}(\mathbf{m}, \vec{\mathbf{u}})$ passant par \mathbf{m} et portée par $\vec{\mathbf{u}}$ sépare le plan en deux demi-plans : celui qui satisfait $\det(\mathbf{z} - \mathbf{m}, \vec{\mathbf{u}}) \geq 0$ et l'autre qui vérifie $\det(\mathbf{z} - \mathbf{m}, \vec{\mathbf{u}}) \leq 0$. Cette ligne coupe le segment $[\mathbf{ab}]$ si \mathbf{a} et \mathbf{b} ne sont pas dans le même demi-plan (voir la figure 6.6), c'est-à-dire si $\det(\mathbf{a} - \mathbf{m}, \vec{\mathbf{u}}) \cdot \det(\mathbf{b} - \mathbf{m}, \vec{\mathbf{u}}) \leq 0$.

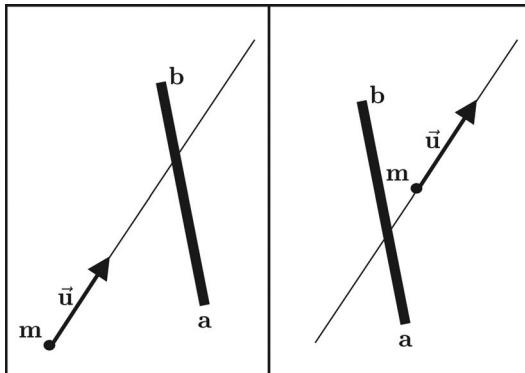


Figure 6.6. La droite $\mathcal{D}(\mathbf{m}, \vec{\mathbf{u}})$ coupe le segment $[\mathbf{ab}]$

Dans la figure de gauche, la demi-droite $\mathcal{E}(\mathbf{m}, \vec{\mathbf{u}})$ coupe le segment $[\mathbf{ab}]$, ce qui n'est pas le cas dans la figure de droite. L'inégalité n'est donc pas suffisante pour affirmer que $\mathcal{E}(\mathbf{m}, \vec{\mathbf{u}})$ coupe $[\mathbf{ab}]$. Supposons que $\det(\mathbf{a} - \mathbf{m}, \vec{\mathbf{u}}) \cdot \det(\mathbf{b} - \mathbf{m}, \vec{\mathbf{u}}) \leq 0$ (c'est-à-dire $\mathcal{D}(\mathbf{m}, \vec{\mathbf{u}})$ coupe $[\mathbf{ab}]$). Les points de $\mathcal{E}(\mathbf{m}, \vec{\mathbf{u}})$ satisfont $\mathbf{z} = \mathbf{m} + \alpha \vec{\mathbf{u}}$, $\alpha \geq 0$. Le point \mathbf{z} appartient au segment $[\mathbf{ab}]$ si $\mathbf{m} + \alpha \vec{\mathbf{u}} = \mathbf{a}$ et $\mathbf{b} - \mathbf{a}$ sont colinéaires, c'est-à-dire lorsque α satisfait $\det(\mathbf{m} + \alpha \vec{\mathbf{u}} - \mathbf{a}, \mathbf{b} - \mathbf{a}) = 0$ (voir figure 6.7).

Puisque le déterminant est une forme multilinéaire nous avons :

$$\det(\mathbf{m} - \mathbf{a}, \mathbf{b} - \mathbf{a}) + \alpha \det(\vec{\mathbf{u}}, \mathbf{b} - \mathbf{a}) = 0$$

C'est-à-dire :

$$\alpha = \frac{\det(\mathbf{a} - \mathbf{m}, \mathbf{b} - \mathbf{a})}{\det(\vec{\mathbf{u}}, \mathbf{b} - \mathbf{a})}$$

Si $\alpha \geq 0$, alors α représente la distance d de \mathbf{m} au segment suivant $\vec{\mathbf{u}}$. Si $\alpha < 0$, alors le rayon du télémètre n'atteindra jamais le segment. La condition $\alpha \geq 0$ correspond à la seconde inéquation qu'il nous fallait démontrer.

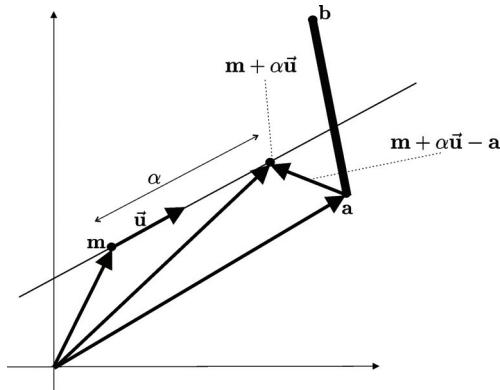


Figure 6.7. Si le point $\mathbf{m} + \alpha \vec{\mathbf{u}}$ se retrouve sur le segment $[\mathbf{a}, \mathbf{b}]$ alors α correspond à la distance directionnelle

2) Dans ce qui suit, les coordonnées du centre \mathbf{m} du robot sont notées (x, y) et $\vec{\mathbf{u}}$ représente un vecteur unitaire correspondant à la direction du laser. Pour le $k^{\text{ième}}$ capteur, nous avons :

$$\vec{\mathbf{u}} = \begin{pmatrix} \cos\left(\frac{k\pi}{4} + \theta\right) \\ \sin\left(\frac{k\pi}{4} + \theta\right) \end{pmatrix}, \quad k \in \{0, \dots, 7\}$$

Pour avoir une expression de $\mathbf{f}(\mathbf{p})$, il nous faut calculer les distances retournées par les télémètres. En utilisant le théorème montré à la question 1), nous pouvons en déduire le simulateur $\mathbf{f}(\mathbf{p})$ qui suit :

```

input : (x, y, theta)
for i = 1 to 8
     $\vec{\mathbf{u}} := \left( \cos\left(\frac{(i-1)\pi}{4} + \theta\right); \sin\left(\frac{(i-1)\pi}{4} + \theta\right) \right);$ 
     $\mathbf{m} := (x \ y)^T; \quad \ell_i := \infty;$ 
    for j = 1 to n
         $\alpha := \frac{\det(\mathbf{a}_j - \mathbf{m}, \mathbf{b}_j - \mathbf{a}_j)}{\det(\vec{\mathbf{u}}, \mathbf{b}_j - \mathbf{a}_j)};$ 
        if  $(\det(\mathbf{a}_j - \mathbf{m}, \vec{\mathbf{u}}) \cdot \det(\mathbf{b}_j - \mathbf{m}, \vec{\mathbf{u}}) \leq 0)$  and  $(\alpha \geq 0)$ 
            then  $\ell_i := \min(\ell_i, \alpha);$ 
    next j
next i;
return  $(\ell_1, \dots, \ell_8)$ 

```

La fonction MATLAB qui suit correspond à ce simulateur :

```

function y=f(p)
y=inf(8,1);
for i=1 :8,

```

```

u=[cos((i-1)*pi/4+p(3));sin((i-1)*pi/4+p(3))] ;
m=[p(1);p(2)] ;
for j=1:length(A),
a=A(:,j);b=B(:,j);
if det([a-m u])*det([b-m u]) <= 0
alpha=-det([b-a m-a])/det([b-a u]);
if alpha >= 0, y(i)=min(alpha,y(i)); end;
end; end; end

```

3) Le programme MATLAB suivant, rangé dans `anneal.m`, propose une minimisation du critère $j(\mathbf{p}) = \|\mathbf{f}(\mathbf{p}) - \mathbf{y}\|$ par la méthode du recuit simulé :

```

function j1=j(p), j1=norm(y-f(p)); end % fonction à minimiser
A=[0 7 7 9 9 7 7 4 2 0 5 6 6 5; 0 0 2 2 4 4 7 7 5 5 2 2 3 3];
B=[7 7 9 9 7 7 4 2 0 0 6 6 5 5; 0 2 2 4 4 7 7 5 5 0 2 3 3 2];
y=[6.4;3.6;2.3;2.1;1.7;1.6;3.0;3.1];
p0=[0;0;0];
T=10;
while (T>0.01)
p=p0+T*randn(3,1); draw(p,y);
if j(p)<j(p0), p0=p; end;
T=0.99*T;
end

```

Ce programme effectue une recherche aléatoire et conserve le paramètre \mathbf{p} qui est le meilleur jusqu'à présent. La variable T est la température qui donne le pas du déplacement de la recherche. Cette température diminue de façon exponentielle en fonction du temps. L'algorithme nous génère la solution représentée par la figure 6.8.

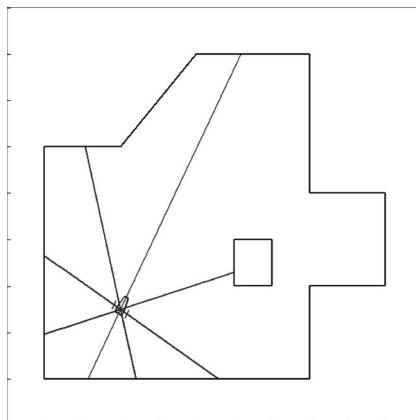


Figure 6.8. La position du robot retrouvée à partir des huit distances données par les télémètres laser

Chapitre 7

Filtre de Kalman

Dans les chapitres 2 et 3, nous avons proposé des outils pour la commande non linéaire des robots. Pour cela, nous avions supposé que le vecteur d'état était parfaitement connu. Or, cela n'est pas le cas en pratique. Ce vecteur doit être estimé à partir des mesures capteurs. Dans le cas où les seules variables inconnues sont associées à la position du robot, le chapitre 5 donne des pistes pour les retrouver. Dans le cas plus général, le *filtrage* ou *observation d'état* cherche à reconstruire ce vecteur d'état au mieux à partir de toutes les données mesurées sur le robot à travers le temps en prenant en compte les équations d'état. Le but de ce chapitre est de montrer comment une telle reconstruction peut être faite, dans un contexte stochastique où le système à observer est supposé linéaire. C'est l'objectif du filtre de Kalman [KAL 60] qui sera développé dans ce chapitre. Le filtre de Kalman est utilisé dans de nombreuses applications de robotique mobile, bien que les robots considérés soient fortement non linéaires. Pour de telles applications, les conditions initiales sont supposées relativement bien connues afin de permettre une linéarisation fiable.

7.1. Matrices de covariance

Le filtre de Kalman repose principalement sur la notion de matrice de covariance qu'il est important de maîtriser pour comprendre la synthèse et l'utilisation de l'observateur. Cette section rappelle les notions fondamentales sur les matrices de covariance.

7.1.1. Définitions et interprétations

Considérons deux vecteurs aléatoires $\mathbf{x} \in \mathbb{R}^n$ et $\mathbf{y} \in \mathbb{R}^m$. Les espérances mathématiques de \mathbf{x} et \mathbf{y} sont dénotées $\bar{\mathbf{x}} = E(\mathbf{x})$, $\bar{\mathbf{y}} = E(\mathbf{y})$. Définissons les

variations de \mathbf{x} et \mathbf{y} par $\tilde{\mathbf{x}} = \mathbf{x} - \bar{\mathbf{x}}$ et $\tilde{\mathbf{y}} = \mathbf{y} - \bar{\mathbf{y}}$. La *matrice de covariance* est donnée par :

$$\boldsymbol{\Gamma}_{\mathbf{x}\mathbf{y}} = E(\tilde{\mathbf{x}} \cdot \tilde{\mathbf{y}}^T) = E((\mathbf{x} - \bar{\mathbf{x}})(\mathbf{y} - \bar{\mathbf{y}})^T)$$

La matrice de covariance pour \mathbf{x} est définie par :

$$\boldsymbol{\Gamma}_{\mathbf{x}} = \boldsymbol{\Gamma}_{\mathbf{x}\mathbf{x}} = E(\tilde{\mathbf{x}} \cdot \tilde{\mathbf{x}}^T) = E((\mathbf{x} - \bar{\mathbf{x}})(\mathbf{x} - \bar{\mathbf{x}})^T)$$

Celle pour \mathbf{y} est :

$$\boldsymbol{\Gamma}_{\mathbf{y}} = \boldsymbol{\Gamma}_{\mathbf{y}\mathbf{y}} = E(\tilde{\mathbf{y}} \cdot \tilde{\mathbf{y}}^T) = E((\mathbf{y} - \bar{\mathbf{y}})(\mathbf{y} - \bar{\mathbf{y}})^T)$$

Notons que $\mathbf{x}, \mathbf{y}, \tilde{\mathbf{x}}, \tilde{\mathbf{y}}$ sont des vecteurs aléatoires alors que $\bar{\mathbf{x}}, \bar{\mathbf{y}}, \boldsymbol{\Gamma}_{\mathbf{x}}, \boldsymbol{\Gamma}_{\mathbf{y}}, \boldsymbol{\Gamma}_{\mathbf{x}\mathbf{y}}$ sont déterministes. Une matrice de covariance $\boldsymbol{\Gamma}_{\mathbf{x}}$ d'un vecteur aléatoire \mathbf{x} est toujours définie positive (nous écrirons $\boldsymbol{\Gamma}_{\mathbf{x}} \succ \mathbf{0}$), sauf cas dégénéré. Sur la machine, un vecteur aléatoire peut être représenté par un nuage de points associés à des réalisations. Considérons le programme MATLAB suivant :

```
x=2+randn(1000,1); e=randn(1000,1); y=2*x.^2+e; plot(x,y);
xbar=mean(x); ybar=mean(y); xtilde=x-xbar; ytilde=y-ybar;
plot(xtilde,ytilde);
Gx=mean(xtilde.^2); Gy=mean(ytilde.^2); Gxy=mean(xtilde.*ytilde);
```

Nous obtenons la figure 7.1 qui nous donne une représentation des variables aléatoires x, y (figure de gauche) et \tilde{x}, \tilde{y} (figure de droite). Le programme nous donne aussi les estimations :

$$\bar{x} \simeq 1.99, \bar{y} \simeq 9.983, \Gamma_x \simeq 1.003, \Gamma_y \simeq 74.03, \Gamma_{xy} \simeq 8.082$$

où $\bar{x}, \bar{y}, \Gamma_x, \Gamma_y, \Gamma_{xy}$ correspondent à $xbar, ybar, Gx, Gy, Gxy$.

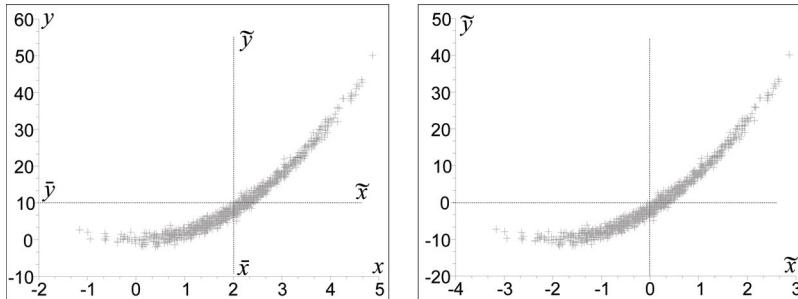


Figure 7.1. Un nuage de points pour représenter une paire de deux variables aléatoires

Deux vecteurs aléatoires x et y sont linéairement indépendants (ou non corrélés ou orthogonaux) si $\Gamma_{xy} = 0$. Sur la figure 7.2, les deux nuages de points correspondent à des variables non corrélées. Seule la figure de droite correspond à des variables indépendantes.

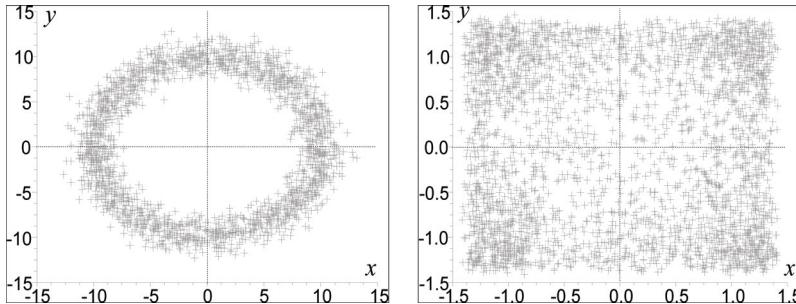


Figure 7.2. A gauche : variables (x, y) dépendantes mais non corrélées ; à droite : variables indépendantes

La figure de gauche a été générée par :

```
rho=10+randn(2000,1); theta=2*pi*rand(2000,1); x=rho.*sin(theta);  
y=rho.*cos(theta);
```

Et la figure de droite a été générée comme suit :

```
x=atan(2*randn(3000,1)); y=atan(2*randn(3000,1));
```

Blancheur : un vecteur aléatoire x est dit *blanc* si toutes ses composantes x_i sont indépendantes entre elles. Dans un tel cas, la matrice de covariance Γ_x de x est diagonale.

7.1.2. Propriétés

Une matrice de covariance est symétrique et positive, c'est-à-dire que toutes ses valeurs propres sont réelles et positives. L'ensemble de toutes les matrices de covariance de $\mathbb{R}^{n \times n}$ sera noté $\mathcal{S}^+(\mathbb{R}^n)$.

Décomposition : toute matrice symétrique Γ est diagonalisable et possède une base de vecteurs propres orthonormaux. On peut donc écrire :

$$\Gamma = \mathbf{R} \cdot \mathbf{D} \cdot \mathbf{R}^{-1}$$

où \mathbf{R} est une matrice de rotation (c'est-à-dire $\mathbf{R}^T \mathbf{R} = \mathbf{I}$ et $\det \mathbf{R} = 1$). La matrice \mathbf{R} correspond aux vecteurs propres et \mathbf{D} est une matrice diagonale dont les éléments sont les valeurs propres. Pour les matrices de $\mathcal{S}^+(\mathbb{R}^n)$, ces valeurs propres sont positives.

Racine carrée : toute matrice Γ de $\mathcal{S}^+(\mathbb{R}^n)$ admet une unique racine carrée dans $\mathcal{S}^+(\mathbb{R}^n)$. Cette dernière sera notée $\Gamma^{\frac{1}{2}}$. D'après le théorème de correspondance des valeurs propres, les valeurs propres de $\Gamma^{\frac{1}{2}}$ sont les racines carrées de celles des valeurs propres de Γ .

EXEMPLE 7.1.– Considérons le script MATLAB suivant :

```
A=rand(3,3); S1=A*A'; [R,D]=eig(S1); S2=R*D*R'; A2=sqrtm(S2); S3=A2*A2'.
```

La matrice D est diagonale et la matrice R est une matrice de rotation qui contient les vecteurs propres de $S1$. Les trois matrices $S1$, $S2$ et $S3$ sont égales. Cela n'est pas le cas pour les matrices A et $A2$ car seulement $A2$ est symétrique. Ici, `sqrt` renvoie la racine carrée de $S2$ et donc $A2$ est une matrice de covariance.

Ordre : si Γ_1 et Γ_2 appartiennent à $\mathcal{S}^+(\mathbb{R}^n)$, alors $\Gamma = \Gamma_1 + \Gamma_2$ appartient aussi à $\mathcal{S}^+(\mathbb{R}^n)$. Cela revient à dire que $\mathcal{S}^+(\mathbb{R}^n)$ est un cône convexe de $\mathbb{R}^{n \times n}$. Définissons la relation d'ordre :

$$\Gamma_1 \leq \Gamma_2 \Leftrightarrow \Gamma_2 - \Gamma_1 \in \mathcal{S}^+(\mathbb{R}^n)$$

On vérifie qu'elle est bien réflexive, antisymétrique et transitive. Si $\Gamma_1 \leq \Gamma_2$ alors l'ellipsoïde de confiance (voir le paragraphe qui suit) de niveau a de Γ_1 est inclus dans celle correspondant à Γ_2 . Plus une matrice de covariance est petite (au sens de cette relation d'ordre) et plus elle est précise. On dira qu'elle est meilleure ou plus précise.

7.1.3. Ellipsoïde de confiance

Un vecteur aléatoire \mathbf{x} de \mathbb{R}^n peut être caractérisé par la paire $(\bar{\mathbf{x}}, \Gamma_{\mathbf{x}})$, à laquelle peut être associée un ellipsoïde de \mathbb{R}^n qui est censé enfermer les valeurs consistantes pour \mathbf{x} . En pratique, pour des raisons purement graphiques, nous sommes souvent intéressés par seulement deux composantes $\mathbf{w} = (x_i, x_j)$ de \mathbf{x} (l'écran d'un ordinateur est en effet bidimensionnel). La moyenne $\bar{\mathbf{w}}$ peut être déduite directement de $\bar{\mathbf{x}}$ par extraction des $i^{\text{ème}}$ et $j^{\text{ème}}$ composantes. La matrice de covariance $\Gamma_{\mathbf{w}} \in \mathcal{S}^+(\mathbb{R}^2)$ peut aussi être obtenue à partir de $\Gamma_{\mathbf{x}} \in \mathcal{S}^+(\mathbb{R}^n)$ par extraction des $i^{\text{ème}}$ et $j^{\text{ème}}$ lignes et colonnes. L'*ellipsoïde de confiance* associée à \mathbf{w} est décrite par l'inégalité :

$$\mathcal{E}_{\mathbf{w}} : (\mathbf{w} - \bar{\mathbf{w}})^T \Gamma_{\mathbf{w}}^{-1} (\mathbf{w} - \bar{\mathbf{w}}) \leq a^2$$

où a est un réel positif arbitraire. Dans si \mathbf{w} est un vecteur aléatoire gaussien, cet ellipsoïde correspond à une courbe de niveau de la densité de probabilité pour \mathbf{w} .

Puisque $\Gamma_w^{-1} \succ 0$, elle possède une racine carrée $\Gamma_w^{-\frac{1}{2}}$ qui est aussi définie positive. Nous pouvons donc écrire :

$$\begin{aligned}
 \mathcal{E}_w &= \left\{ w \mid (w - \bar{w})^T \Gamma_w^{-\frac{1}{2}} \cdot \Gamma_w^{-\frac{1}{2}} (w - \bar{w}) \leq a^2 \right\} \\
 &= \left\{ w \mid \left\| \frac{1}{a} \cdot \Gamma_w^{-\frac{1}{2}} (w - \bar{w}) \right\| \leq 1 \right\} \\
 &= \left\{ w \mid \frac{1}{a} \cdot \Gamma_w^{-\frac{1}{2}} (w - \bar{w}) \in \mathcal{U} \right\}, \text{ où } \mathcal{U} \text{ est le disque unité} \\
 &= \left\{ w \mid w \in \bar{w} + a \Gamma_w^{\frac{1}{2}} \mathcal{U} \right\} \\
 &= \bar{w} + a \Gamma_w^{\frac{1}{2}} \mathcal{U}
 \end{aligned}$$

L'ellipsoïde \mathcal{E}_w peut donc être définie comme l'image du disque unité par la fonction affine $w(s) = \bar{w} + a \cdot \Gamma_w^{\frac{1}{2}} s$, comme illustré par la figure 7.3.

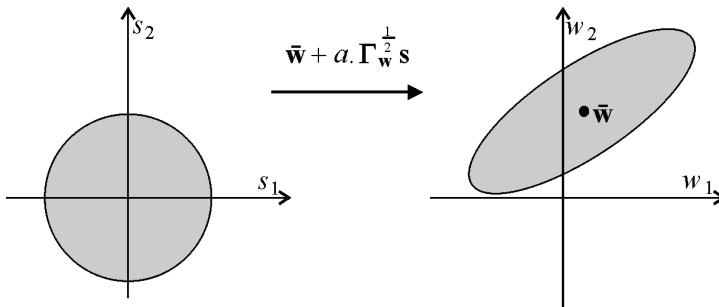


Figure 7.3. Un ellipsoïde de confiance est l'image du cercle unité par une fonction affine

Rappelons que pour un vecteur aléatoire gaussien normé centré s , la variable aléatoire $z = s^T s$ suit une loi du χ^2 . En dimension 2, cette densité de probabilité est donnée par :

$$\pi_z(z) = \begin{cases} \frac{1}{2} \exp\left(-\frac{z}{2}\right) & \text{si } z \geq 0 \\ 0 & \text{sinon} \end{cases}$$

Ainsi, pour un $a > 0$ donné, nous avons :

$$\begin{aligned}
 \eta &\stackrel{\text{def}}{=} \text{prob} (||s|| \leq a) = \text{prob} (s^T s \leq a^2) = \text{prob} (z \leq a^2) \\
 &= \int_0^{a^2} \frac{1}{2} \exp\left(-\frac{z}{2}\right) dz = 1 - e^{-\frac{1}{2}a^2}
 \end{aligned}$$

Et donc :

$$a = \sqrt{-2 \ln(1 - \eta)}$$

Cette relation nous permet de calculer le seuil a qu'il nous faut choisir pour avoir une probabilité d'appartenance à l'ellipsoïde de η . Mais attention cette interprétation probabiliste n'a du sens que dans le cas gaussien. La fonction MATLAB suivante nous permet de tracer \mathcal{E}_w pour une probabilité η donnée :

```
function draw_ellipse(wbar,Gw,eta);
s=0 :0.01 :2*pi;
w=wbar*ones(size(s))+sqrtm(-2*log(1-eta)*Gw)*[cos(s);sin(s)];
plot(w(1, :),w(2, :));
```

7.1.4. Génération de vecteurs aléatoires gaussiens

Si nous générions n nombres aléatoires gaussiens centrés, nous obtenons la réalisation d'un vecteur aléatoire dont le centre est $\bar{\mathbf{x}} = \mathbf{0}$ et dont la matrice de covariance Γ_x est l'identité. Dans cette section, nous montrons comment, à partir d'un générateur de nombres aléatoires gaussiens centrés, qui nous permet de réaliser \mathbf{x} , il est possible d'obtenir un vecteur aléatoire gaussien \mathbf{y} de dimension n d'espérance et de matrice de covariance Γ_y . Le principe de cette génération repose sur le théorème suivant.

THÉORÈME 7.1.– *Si \mathbf{x} , α et \mathbf{y} sont trois vecteurs aléatoires liés par la relation $\mathbf{y} = \mathbf{Ax} + \alpha + \mathbf{b}$ (où \mathbf{A} et \mathbf{b} sont déterministes). Supposons que \mathbf{x} , α soient indépendants et que α soit centré, nous avons :*

$$\begin{aligned}\bar{\mathbf{y}} &= \mathbf{A}\bar{\mathbf{x}} + \mathbf{b} \\ \Gamma_y &= \mathbf{A} \cdot \Gamma_x \cdot \mathbf{A}^T + \Gamma_\alpha\end{aligned}\quad [7.1]$$

PREUVE.– Nous avons :

$$\bar{\mathbf{y}} = E(\mathbf{Ax} + \alpha + \mathbf{b}) = \mathbf{A}E(\mathbf{x}) + E(\alpha) + \mathbf{b} = \mathbf{A}\bar{\mathbf{x}} + \mathbf{b}$$

De plus :

$$\begin{aligned}\Gamma_y &= E((\mathbf{y} - \bar{\mathbf{y}})(\mathbf{y} - \bar{\mathbf{y}})^T) \\ &= E((\mathbf{Ax} + \alpha + \mathbf{b} - \mathbf{A}\bar{\mathbf{x}} - \mathbf{b})(\mathbf{Ax} + \alpha + \mathbf{b} - \mathbf{A}\bar{\mathbf{x}} - \mathbf{b})^T) \\ &= E((\mathbf{A}\tilde{\mathbf{x}} + \alpha) \cdot (\mathbf{A}\tilde{\mathbf{x}} + \alpha)^T) \\ &= \mathbf{A} \cdot \underbrace{E(\tilde{\mathbf{x}} \cdot \tilde{\mathbf{x}}^T)}_{=\Gamma_x} \cdot \mathbf{A}^T + \mathbf{A} \cdot \underbrace{E(\tilde{\mathbf{x}} \cdot \alpha^T)}_{=0} + \underbrace{E(\alpha \cdot \tilde{\mathbf{x}}^T)}_{=0} \cdot \mathbf{A}^T + \underbrace{E(\alpha \cdot \alpha^T)}_{=\Gamma_\alpha} \\ &= \mathbf{A} \cdot \Gamma_x \cdot \mathbf{A}^T + \Gamma_\alpha\end{aligned}$$

ce qui termine la preuve. ■

Ainsi, si \mathbf{x} est un vecteur aléatoire gaussien, blanc et unitaire centré (c'est-à-dire $\bar{\mathbf{x}} = \mathbf{0}$ et $\mathbf{\Gamma}_x = \mathbf{I}$), le vecteur aléatoire $\mathbf{y} = \mathbf{\Gamma}_y^{\frac{1}{2}}\mathbf{x} + \bar{\mathbf{y}}$ aura une espérance de $\bar{\mathbf{y}}$ et une matrice de covariance égale à $\mathbf{\Gamma}_y$ (voir figure 7.4). Pour générer un vecteur aléatoire gaussien de matrice de covariance $\mathbf{\Gamma}_y$ et d'espérance $\bar{\mathbf{y}}$, nous allons utiliser cette propriété. Ainsi, la figure 7.4 à droite a été obtenue par le script :

```
n=1000 ; Gy=[3,1 ; 1,3] ; ybar=[2 ; 3] ; x=randn(2,n) ;
y=ybar*ones(1,n)+sqrtm(Gy)*x; plot(y(1, :),y(2, :),'.')
```

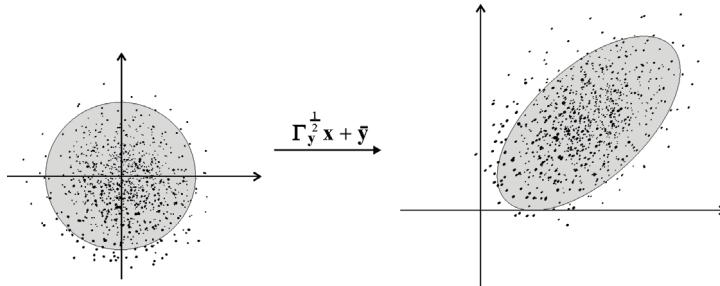


Figure 7.4. Le vecteur aléatoire gaussien $\mathbf{y} : (\bar{\mathbf{y}}, \mathbf{\Gamma}_y)$ est l'image par une application affine d'un vecteur aléatoire \mathbf{x} gaussien blanc et unitaire

7.2. Estimateur orthogonal non biaisé

Considérons deux vecteurs aléatoires $\mathbf{x} \in \mathbb{R}^n$ et $\mathbf{y} \in \mathbb{R}^m$. Le vecteur \mathbf{y} correspond au vecteur des mesures, qui est pour l'instant un vecteur aléatoire qui sera disponible qu'une fois la campagne de mesures effectuée. Le vecteur aléatoire \mathbf{x} est celui qu'il nous faut estimer. Un *estimateur* est une fonction $\phi(\mathbf{y})$ censée nous donner une estimation de \mathbf{x} connaissant la mesure \mathbf{y} . La figure 7.5 montre un estimateur non linéaire correspondant à $E(x|\mathbf{y})$.

Or, l'obtention d'une expression analytique pour un tel estimateur n'est généralement pas facile et on préfère se limiter aux estimateurs linéaires. Un *estimateur linéaire* est une fonction de $\mathbb{R}^m \rightarrow \mathbb{R}^n$ linéaire de la forme :

$$\hat{\mathbf{x}} = \mathbf{K}\mathbf{y} + \mathbf{b} \quad [7.2]$$

où $\mathbf{K} \in \mathbb{R}^{n \times m}$ et $\mathbf{b} \in \mathbb{R}^n$. Dans cette section, nous allons proposer une méthode capable de trouver un *bon* \mathbf{K} et un *bon* \mathbf{b} à partir de la seule connaissance des moments du premier ordre $\bar{\mathbf{x}}, \bar{\mathbf{y}}$ et des moments du second ordre $\mathbf{\Gamma}_x, \mathbf{\Gamma}_y, \mathbf{\Gamma}_{xy}$. L'*erreur d'estimation* est :

$$\varepsilon = \hat{\mathbf{x}} - \mathbf{x}$$

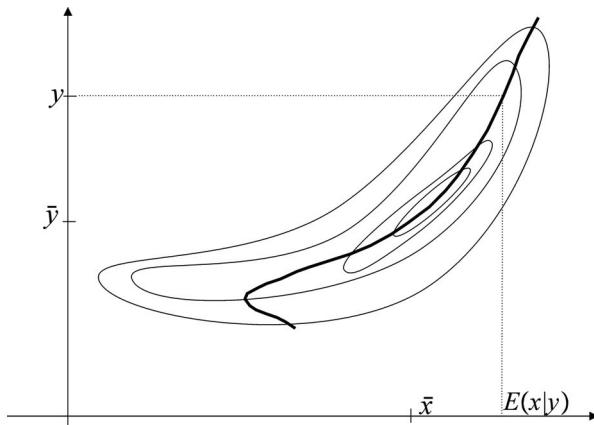
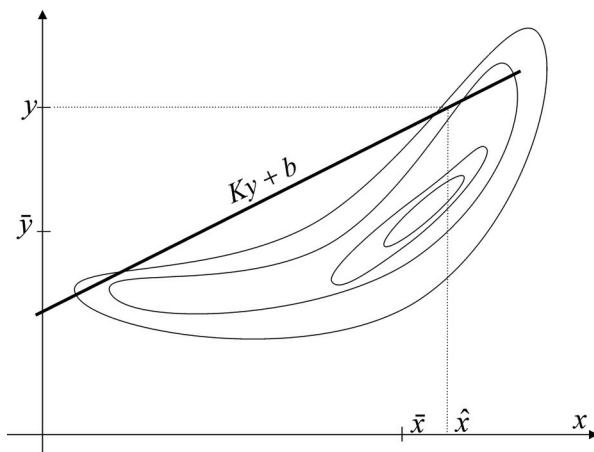
Figure 7.5. Estimateur non linéaire $E(x|y)$ 

Figure 7.6. Estimateur linéaire biaisé

L'estimateur est dit *non biaisé* si $E(\varepsilon) = \mathbf{0}$. Il est *orthogonal* si $E(\varepsilon \tilde{y}^T) = \mathbf{0}$. Ce nom provient du fait que l'espace des variables aléatoires de \mathbb{R} peut être muni d'un produit scalaire défini par $\langle a, b \rangle = E((a - \bar{a})(b - \bar{b}))$ et que si ce produit scalaire est nul les deux variables aléatoires a et b sont dites orthogonales. Dans le cas vectoriel (qui est celui de notre paragraphe puisque ε et \tilde{y} sont des vecteurs), on dit que les deux vecteurs aléatoires \mathbf{a} et \mathbf{b} sont orthogonaux si leurs composantes le sont, c'est-à-dire $E((a_i - \bar{a}_i)(b_j - \bar{b}_j)) = 0$ pour tout (i, j) ou de façon équivalente $E((\mathbf{a} - \bar{\mathbf{a}})(\mathbf{b} - \bar{\mathbf{b}})^T) = \mathbf{0}$. La figure 7.6 représente les courbes de niveau d'une loi de probabilité pour le couple (x, y) . La droite illustre un estimateur linéaire. Tirs

au hasard un couple (x, y) tout en respectant sa loi de probabilité. Il est clair que la probabilité d'être en dessous de la droite est forte, c'est-à-dire que la probabilité d'avoir $\hat{x} < x$ est grande ou encore que $E(\varepsilon) < 0$. L'estimateur est donc biaisé. La figure 7.7 représente quatre estimateurs linéaires différents. Pour l'estimateur (a), $E(\varepsilon) < 0$ et pour l'estimateur (c), $E(\varepsilon) > 0$. Pour les estimateurs (b) et (d), $E(\varepsilon) = 0$ et donc les deux estimateurs sont non biaisés. Pourtant, il est clair que l'estimateur (b) est meilleur. Ce qui différencie les deux, c'est l'orthogonalité. Pour (d), on a $E(\varepsilon\tilde{y}) < 0$, (si $\tilde{y} > 0$, ε tend à être négatif alors que si $\tilde{y} < 0$, ε tend à être positif).

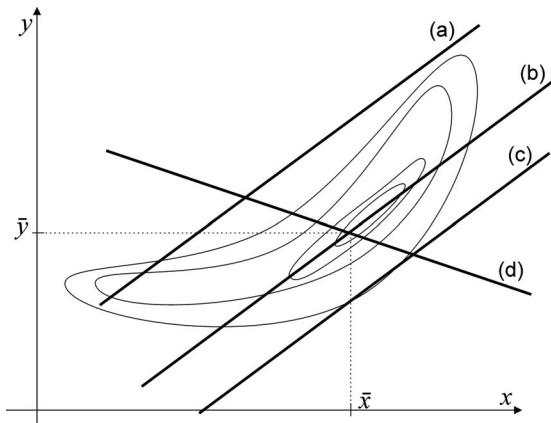


Figure 7.7. Parmi ces quatre estimateurs linéaires, l'estimateur (b), qui est non biaisé et orthogonal, semble être le meilleur

THÉORÈME 7.2.– Soient deux vecteurs aléatoires \mathbf{x} et \mathbf{y} . Il existe un unique estimateur orthogonal non biaisé. Il est donné par :

$$\hat{\mathbf{x}} = \bar{\mathbf{x}} + \mathbf{K} \cdot (\mathbf{y} - \bar{\mathbf{y}}) \quad [7.3]$$

où :

$$\mathbf{K} = \mathbf{\Gamma}_{\mathbf{xy}} \mathbf{\Gamma}_{\mathbf{y}}^{-1} \quad [7.4]$$

est appelé gain de Kalman.

EXEMPLE 7.2.– Considérons à nouveau l'exemple de la section 7.1.1. Nous obtenons :

$$\hat{x} = \bar{x} + \Gamma_{xy} \Gamma_y^{-1} \cdot (y - \bar{y}) = 2 + 0.1 \cdot (y - 10)$$

L'estimateur correspondant est illustré par la figure 7.8.

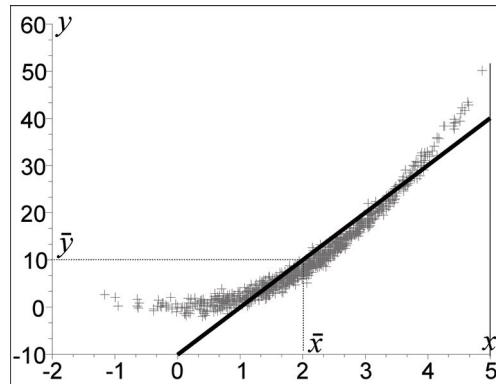


Figure 7.8. Estimateur orthogonal linéaire et non biaisé

PREUVE DU THÉORÈME.– Nous avons :

$$E(\varepsilon) = E(\hat{\mathbf{x}} - \mathbf{x}) \stackrel{[7.2]}{=} E(\mathbf{K}\mathbf{y} + \mathbf{b} - \mathbf{x}) = \mathbf{K}E(\mathbf{y}) + \mathbf{b} - E(\mathbf{x}) = \mathbf{K}\bar{\mathbf{y}} + \mathbf{b} - \bar{\mathbf{x}}$$

L'estimateur est non biaisé si $E(\varepsilon) = \mathbf{0}$, c'est-à-dire :

$$\mathbf{b} = \bar{\mathbf{x}} - \mathbf{K}\bar{\mathbf{y}} \quad [7.5]$$

ce qui nous donne [7.3]. Dans ce cas :

$$\varepsilon = \hat{\mathbf{x}} - \mathbf{x} \stackrel{[7.3]}{=} \bar{\mathbf{x}} + \mathbf{K} \cdot (\mathbf{y} - \bar{\mathbf{y}}) - \mathbf{x} = \mathbf{K}\tilde{\mathbf{y}} - \tilde{\mathbf{x}} \quad [7.6]$$

L'estimateur est orthogonal si :

$$\begin{aligned} E(\varepsilon \cdot \tilde{\mathbf{y}}^T) = \mathbf{0} &\stackrel{[7.6]}{\Leftrightarrow} E((\mathbf{K}\tilde{\mathbf{y}} - \tilde{\mathbf{x}}) \cdot \tilde{\mathbf{y}}^T) = \mathbf{0} \\ &\Leftrightarrow E(\mathbf{K}\tilde{\mathbf{y}}\tilde{\mathbf{y}}^T - \tilde{\mathbf{x}}\tilde{\mathbf{y}}^T) = \mathbf{0} \\ &\Leftrightarrow \mathbf{K}\Gamma_y - \Gamma_{xy} = \mathbf{0} \\ &\Leftrightarrow \mathbf{K} = \Gamma_{xy} \cdot \Gamma_y^{-1} \end{aligned}$$

ce qui termine la preuve. ■

THÉORÈME 7.3.– *La matrice de covariance de l'erreur de l'estimateur linéaire orthogonal non biaisé est :*

$$\Gamma_\varepsilon = \Gamma_x - \mathbf{K} \cdot \Gamma_{yx} \quad [7.7]$$

PREUVE.– La matrice de covariance de ε dans le cas non biaisé s'écrit :

$$\begin{aligned} \Gamma_\varepsilon &= E(\varepsilon \cdot \varepsilon^T) \stackrel{[7.6]}{=} E((\mathbf{K}\tilde{\mathbf{y}} - \tilde{\mathbf{x}}) \cdot (\mathbf{K}\tilde{\mathbf{y}} - \tilde{\mathbf{x}})^T) \\ &= E((\mathbf{K}\tilde{\mathbf{y}} - \tilde{\mathbf{x}}) \cdot (\tilde{\mathbf{y}}^T \mathbf{K}^T - \tilde{\mathbf{x}}^T)) \\ &= E(\mathbf{K}\tilde{\mathbf{y}}\tilde{\mathbf{y}}^T \mathbf{K}^T - \tilde{\mathbf{x}}\tilde{\mathbf{y}}^T \mathbf{K}^T - \mathbf{K}\tilde{\mathbf{y}}\tilde{\mathbf{x}}^T + \tilde{\mathbf{x}}\tilde{\mathbf{x}}^T) \end{aligned}$$

Soit, en utilisant la linéarité de l'opérateur espérance :

$$\Gamma_\varepsilon = (\mathbf{K}\Gamma_y - \Gamma_{xy})\mathbf{K}^T - \mathbf{K}\Gamma_{yx} + \Gamma_x \quad [7.8]$$

Or, d'après [7.4], dans le cas orthogonal $\mathbf{K}\Gamma_y - \Gamma_{xy} = 0$, ce qui termine la preuve. ■

Nous allons maintenant donner un théorème qui montre que l'estimateur linéaire orthogonal non biaisé est le meilleur parmi tous les estimateurs non biaisés. Pour comprendre cette notion de *meilleur*, il faut se souvenir des inégalités sur les matrices de covariance (voir section 7.1), qui dit que $\Gamma_1 \leq \Gamma_2$ si et seulement si $\Delta = \Gamma_2 - \Gamma_1$ est une matrice de covariance.

THÉORÈME 7.4.— *Il n'existe aucun estimateur linéaire non biaisé qui permette d'obtenir une matrice de covariance sur l'erreur Γ_ε plus petite que celle donnée par l'estimateur orthogonal.*

PREUVE.— Toute matrice \mathbf{K} possible pour notre estimateur linéaire non biaisé s'écrit sous la forme $\mathbf{K} = \mathbf{K}_0 + \Delta$ avec $\mathbf{K}_0 = \Gamma_{xy}\Gamma_y^{-1}$ et Δ étant une matrice arbitraire. La matrice de covariance pour l'erreur est d'après (7.8) :

$$\begin{aligned} \Gamma_\varepsilon &= ((\mathbf{K}_0 + \Delta)\Gamma_y - \Gamma_{xy})(\mathbf{K}_0 + \Delta)^T - (\mathbf{K}_0 + \Delta)\Gamma_{yx} + \Gamma_x \\ &= (\mathbf{K}_0 + \Delta)(\underbrace{\Gamma_y\mathbf{K}_0^T + \Gamma_y\Delta^T}_{=\Gamma_{yx}}) - (\underbrace{\Gamma_{xy}\mathbf{K}_0^T + \Gamma_{xy}\Delta^T}_{=\mathbf{K}_0\Gamma_{yx}}) - (\mathbf{K}_0\Gamma_{yx} + \Delta\Gamma_{yx}) + \Gamma_x \\ &= \mathbf{K}_0\Gamma_{yx} + \Delta\Gamma_{yx} + \underbrace{\mathbf{K}_0\Gamma_y\Delta^T}_{=\Gamma_{xy}} + \Delta\Gamma_y\Delta^T - \mathbf{K}_0\Gamma_{yx} - \Gamma_{xy}\Delta^T - \mathbf{K}_0\Gamma_{yx} - \Delta\Gamma_{yx} + \Gamma_x \\ &= -\mathbf{K}_0\Gamma_{yx} + \Delta\Gamma_y\Delta^T + \Gamma_x \end{aligned}$$

Puisque $\Delta\Gamma_y\Delta^T$ est toujours symétrique positive, la matrice de covariance Γ_ε est minimale pour $\Delta = 0$, c'est-à-dire pour $\mathbf{K} = \Gamma_{xy}\Gamma_y^{-1}$, ce qui correspond à l'estimateur non biaisé orthogonal. ■

7.3. Application à l'estimation linéaire

Supposons que \mathbf{x} et \mathbf{y} soient reliés par la relation :

$$\mathbf{y} = \mathbf{Cx} + \beta$$

où β est vecteur aléatoire centré, non corrélé avec \mathbf{x} . La matrice de covariance de \mathbf{x} et β sont notées Γ_x et Γ_β . Utilisons les résultats établis à la section précédente afin de

trouver le meilleur estimateur linéaire non biaisé pour \mathbf{x} (voir [WAL 14] pour plus de détails sur l'estimation linéaire). Nous avons :

$$\begin{aligned}\bar{\mathbf{y}} &= \mathbf{C}\bar{\mathbf{x}} + \bar{\beta} = \mathbf{C}\bar{\mathbf{x}} \\ \mathbf{\Gamma}_y &\stackrel{[7.1]}{=} \mathbf{C}\mathbf{\Gamma}_x\mathbf{C}^T + \mathbf{\Gamma}_\beta \\ \mathbf{\Gamma}_{xy} &= E(\tilde{\mathbf{x}} \cdot \tilde{\mathbf{y}}^T) = E\left(\tilde{\mathbf{x}} \cdot \left(\mathbf{C}\tilde{\mathbf{x}} + \tilde{\beta}\right)^T\right) = E\left(\tilde{\mathbf{x}} \cdot \tilde{\mathbf{x}}^T\mathbf{C}^T + \tilde{\mathbf{x}} \cdot \tilde{\beta}^T\right) \quad [7.9] \\ &= E(\tilde{\mathbf{x}} \cdot \tilde{\mathbf{x}}^T)\mathbf{C}^T + \underbrace{E\left(\tilde{\mathbf{x}} \cdot \tilde{\beta}^T\right)}_{=0} = \mathbf{\Gamma}_x\mathbf{C}^T\end{aligned}$$

Par conséquence, le meilleur estimateur non biaisé pour \mathbf{x} et la matrice de covariance sur l'erreur peuvent être obtenus à partir de $\mathbf{\Gamma}_x$, $\mathbf{\Gamma}_\beta$, \mathbf{C} , $\bar{\mathbf{x}}$ en utilisant les formules suivantes :

$$\begin{aligned}(i) \quad \hat{\mathbf{x}} &\stackrel{[7.3]}{=} \bar{\mathbf{x}} + \mathbf{K}\tilde{\mathbf{y}} && \text{(estimée)} \\ (ii) \quad \mathbf{\Gamma}_\varepsilon &\stackrel{[7.7]}{=} \mathbf{\Gamma}_x - \mathbf{K}\mathbf{C}\mathbf{\Gamma}_x && \text{(covariance de l'erreur)} \\ (iii) \quad \tilde{\mathbf{y}} &\stackrel{[7.9]}{=} \mathbf{y} - \mathbf{C}\bar{\mathbf{x}} && \text{(innovation)} \\ (iv) \quad \mathbf{\Gamma}_y &\stackrel{[7.9]}{=} \mathbf{C}\mathbf{\Gamma}_x\mathbf{C}^T + \mathbf{\Gamma}_\beta && \text{(covariance de l'innovation)} \\ (v) \quad \mathbf{K} &\stackrel{[7.4,7.9]}{=} \mathbf{\Gamma}_x\mathbf{C}^T\mathbf{\Gamma}_y^{-1} && \text{(gain Kalman)}\end{aligned} \quad [7.10]$$

REMARQUE 7.1. – La figure 7.5 illustre une situation où il est intéressant de ne pas prendre un estimateur linéaire. Ici, l'estimateur choisi correspond à $\hat{x} = E(x|y)$. Dans le cas particulier où le couple (\mathbf{x}, \mathbf{y}) est gaussien, l'estimateur $\hat{\mathbf{x}} = E(\mathbf{x}|y)$ correspond à l'estimateur orthogonal non biaisé. Dans ce cas, nous avons d'après [7.10] :

$$\begin{aligned}E(\mathbf{x}|y) &= \bar{\mathbf{x}} + \mathbf{\Gamma}_{xy}\mathbf{\Gamma}_y^{-1}(\mathbf{y} - \bar{\mathbf{y}}) \\ E(\varepsilon \cdot \varepsilon^T | y) &= E\left((\hat{\mathbf{x}} - \mathbf{x})(\hat{\mathbf{x}} - \mathbf{x})^T | y\right) = \mathbf{\Gamma}_x - \mathbf{\Gamma}_{xy}\mathbf{\Gamma}_y^{-1}\mathbf{\Gamma}_{yx}\end{aligned}$$

7.4. Filtre de Kalman

Ce paragraphe présente le filtre de Kalman (voir [DEL 93] pour plus d'informations). Considérons le système décrit par les équations d'état :

$$\begin{cases} \mathbf{x}_{k+1} = \mathbf{A}_k \mathbf{x}_k + \mathbf{u}_k + \alpha_k \\ \mathbf{y}_k = \mathbf{C}_k \mathbf{x}_k + \beta_k \end{cases}$$

où α_k et β_k sont des signaux aléatoires gaussiens indépendants entre eux et blancs dans le temps. Par blanc dans le temps, nous entendons que les vecteurs α_{k_1} et α_{k_2} (ou bien β_{k_1} et β_{k_2}) sont indépendants entre eux si $k_1 \neq k_2$. Le filtre de Kalman alterne deux phases : la *correction* et la *prédiction*. Pour comprendre le mécanisme d'un tel filtre, plaçons-nous à l'instant k et supposons que nous ayons déjà traité les

mesures $\mathbf{y}_0, \mathbf{y}_1, \dots, \mathbf{y}_{k-1}$. A ce stade, le vecteur d'état est un vecteur aléatoire que nous allons noter $\mathbf{x}_{k|k-1}$ (car nous nous trouvons à l'instant k et que les mesures ont été traitées jusqu'à l'instant $k-1$). Ce vecteur aléatoire est représenté par une estimée notée $\hat{\mathbf{x}}_{k|k-1}$ et une matrice de covariance $\Gamma_{k|k-1}$.

Correction : récoltons la mesure \mathbf{y}_k . Le vecteur aléatoire représentant l'état est désormais $\mathbf{x}_{k|k}$ qui est différent de $\mathbf{x}_{k|k-1}$ car $\mathbf{x}_{k|k}$ a connaissance de la mesure \mathbf{y}_k . L'espérance $\hat{\mathbf{x}}_{k|k}$ et la matrice de covariance $\Gamma_{k|k}$ associées à $\mathbf{x}_{k|k}$ sont données par les équations [7.10]. Nous avons donc :

- (i) $\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k \cdot \tilde{\mathbf{y}}_k$ (estimée corrigée)
 - (ii) $\Gamma_{k|k} = \Gamma_{k|k-1} - \mathbf{K}_k \cdot \mathbf{C}_k \Gamma_{k|k-1}$ (covariance corrigée)
 - (iii) $\tilde{\mathbf{y}}_k = \mathbf{y}_k - \mathbf{C}_k \hat{\mathbf{x}}_{k|k-1}$ (innovation)
 - (iv) $\mathbf{S}_k = \mathbf{C}_k \Gamma_{k|k-1} \mathbf{C}_k^T + \Gamma_{\beta_k}$ (covariance de l'innovation)
 - (v) $\mathbf{K}_k = \Gamma_{k|k-1} \mathbf{C}_k^T \mathbf{S}_k^{-1}$ (gain de Kalman)
- [7.11]

Prédiction : sachant les mesures $\mathbf{y}_0, \mathbf{y}_1, \dots, \mathbf{y}_k$, le vecteur aléatoire représentant l'état est désormais $\mathbf{x}_{k+1|k}$. Calculons son espérance $\hat{\mathbf{x}}_{k+1|k}$ et sa matrice de covariance $\Gamma_{k+1|k}$. Puisque :

$$\mathbf{x}_{k+1} = \mathbf{A}_k \mathbf{x}_k + \mathbf{u}_k + \alpha_k$$

nous avons, d'après [7.1] :

$$\hat{\mathbf{x}}_{k+1|k} = \mathbf{A}_k \hat{\mathbf{x}}_{k|k} + \mathbf{u}_k \quad [7.12]$$

et :

$$\Gamma_{k+1|k} = \mathbf{A}_k \cdot \Gamma_{k|k} \cdot \mathbf{A}_k^T + \Gamma_{\alpha_k} \quad [7.13]$$

Filtre de Kalman : le filtre de Kalman complet est donné par les équations suivantes :

- $\hat{\mathbf{x}}_{k+1|k} \stackrel{[7.12]}{=} \mathbf{A}_k \hat{\mathbf{x}}_{k|k} + \mathbf{u}_k$ (estimée prédictive)
- $\Gamma_{k+1|k} \stackrel{[7.13]}{=} \mathbf{A}_k \cdot \Gamma_{k|k} \cdot \mathbf{A}_k^T + \Gamma_{\alpha_k}$ (covariance prédictive)
- $\hat{\mathbf{x}}_{k|k} \stackrel{[7.10,i]}{=} \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k \cdot \tilde{\mathbf{y}}_k$ (estimée corrigée)
- $\Gamma_{k|k} \stackrel{[7.10,ii]}{=} (\mathbf{I} - \mathbf{K}_k \mathbf{C}_k) \Gamma_{k|k-1}$ (covariance corrigée)
- $\tilde{\mathbf{y}}_k \stackrel{[7.10,iii]}{=} \mathbf{y}_k - \mathbf{C}_k \hat{\mathbf{x}}_{k|k-1}$ (innovation)
- $\mathbf{S}_k \stackrel{[7.10,iv]}{=} \mathbf{C}_k \Gamma_{k|k-1} \mathbf{C}_k^T + \Gamma_{\beta_k}$ (covariance de l'innovation)
- $\mathbf{K}_k \stackrel{[7.10,v]}{=} \Gamma_{k|k-1} \mathbf{C}_k^T \mathbf{S}_k^{-1}$ (gain de Kalman)

La figure 7.9 illustre le fait que le filtre de Kalman mémorise le vecteur $\hat{\mathbf{x}}_{k+1|k}$ et la matrice $\Gamma_{k+1|k}$. Ses entrées sont $\mathbf{y}_k, \mathbf{u}_k, \mathbf{A}_k, \mathbf{C}_k, \Gamma_{\alpha_k}$ et Γ_{β_k} . Les quantités $\hat{\mathbf{x}}_{k|k}, \tilde{\mathbf{y}}_k, \mathbf{S}_k, \mathbf{K}_k$ sont des variables auxiliaires.

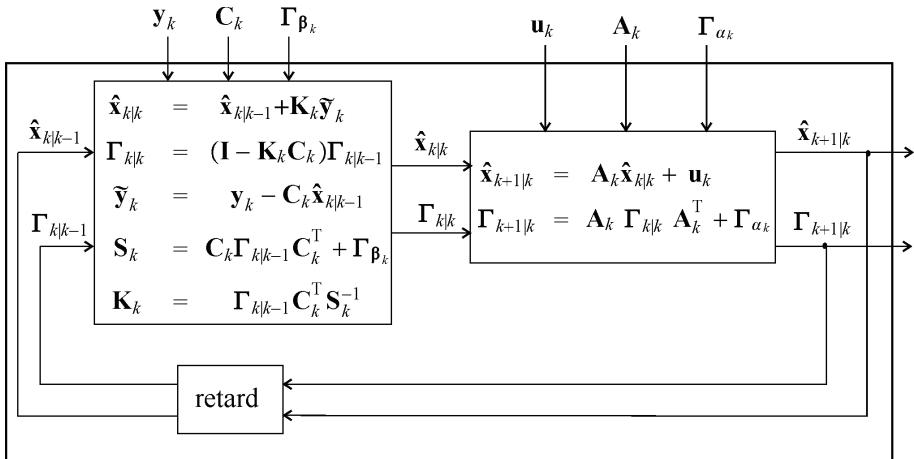


Figure 7.9. Le filtre de Kalman est composé d'un correcteur suivi d'un prédicteur

La fonction MATLAB qui suit implémente le filtre de Kalman. Dans ce programme, nous avons les correspondances suivantes : $x0 \leftrightarrow \hat{x}_{k|k-1}$, $G1 \leftrightarrow \Gamma_{k|k-1}$, $x1 \leftrightarrow \hat{x}_{k+1|k}$, $G1 \leftrightarrow \Gamma_{k+1|k}$, $xup \leftrightarrow \hat{x}_{k|k}$, $Gup \leftrightarrow \Gamma_{k|k}$ (le terme up signifie *update* ou correction).

```
function [x1,G1]=kalman(x0,G0,u,y,Galpha,Gbeta,A,C);
S=C*G0*C'+Gbeta;
K=G0*C'*inv(S);
ytilde=y-C*x0;
xup=x0+K*ytilde;
Gup=G0-K*C*G0;
x1=A*xup + u;
G1=A*Gup*A'+Galpha;
end
```

REMARQUE 7.2.- La covariance de l'innovation S_k peut parfois, du fait de problèmes numériques, perdre le caractère défini positive. Si de tels problèmes surviennent, il est préférable de remplacer l'équation pour covariance corrigée par :

$$\Gamma_{k|k} = \sqrt{(\mathbf{I} - \mathbf{K}_k \mathbf{C}_k) \Gamma_{k|k-1} \Gamma_{k|k-1}^T (\mathbf{I} - \mathbf{K}_k \mathbf{C}_k)^T}$$

qui elle, sera toujours définie positive, et ceci même si la matrice $\Gamma_{k|k-1}$ ne l'est pas. Les équations du filtre de Kalman seront alors plus stables dans le sens où une légère erreur sur le caractère défini positif des matrices de covariance s'efface à l'itération suivante.

REMARQUE 7.3.- Lorsque aucune mesure n'existe, le filtre de Kalman fonctionne en mode *prédicteur*. Afin de pouvoir utiliser la fonction *kalman*, y , Γ_{β} , C doivent

devenir des quantités vides. Elles doivent cependant posséder les bonnes dimensions afin de permettre, sous MATLAB, de faire les opérations matricielles. L'appel se fera alors comme suit :

```
[xhat, Gx]=kalman(xhat, Gx, u, eye(0, 1), Galpha, eye(0, 0), A,
eye(0, length(x)))
```

7.5. Lisseur de Kalman

Le filtre de Kalman est causal. Cela signifie que l'estimée $\hat{x}_{k|k-1}$ ne prend en compte que le passé strict. Le *lissage* consiste en une estimation de l'état alors que toutes les mesures (futures, présentes et passées) sont à notre disposition. Notons N , le temps k maximal. Ce temps correspond par exemple à la date terminale d'une mission effectuée par un robot et pour lequel, on cherche à estimer sa trajectoire. Pour effectuer un lissage, il suffit de relancer un filtre de Kalman dans le sens rétrograde et de fusionner pour chaque k l'information provenant du futur avec celle provenant du passé. Une version optimisée appelée *lisseur de Kalman* peut alors être obtenue en rajoutant aux équations du filtre de Kalman les équations suivantes :

$$\begin{aligned} \mathbf{J}_k &= \mathbf{\Gamma}_{k|k} \cdot \mathbf{A}_k^T \cdot \mathbf{\Gamma}_{k+1|k}^{-1} \\ \hat{\mathbf{x}}_{k|N} &= \hat{\mathbf{x}}_{k|k} + \mathbf{J}_k (\hat{\mathbf{x}}_{k+1|N} - \hat{\mathbf{x}}_{k+1|k}) \\ \mathbf{\Gamma}_{k|N} &= \mathbf{\Gamma}_{k|k} + \mathbf{J}_k (\mathbf{\Gamma}_{k+1|N} - \mathbf{\Gamma}_{k+1|k}) \mathbf{J}_k^T \end{aligned} \quad [7.14]$$

Pour pouvoir effectuer un lissage, on doit tout d'abord lancer le filtre de Kalman pour k allant de 0 à N puis lancer de façon rétrograde les équations [7.14] pour k allant de N à 0. Notons que toutes les grandeurs $\hat{\mathbf{x}}_{k+1|k}$, $\hat{\mathbf{x}}_{k|k}$, $\mathbf{\Gamma}_{k+1|k}$, $\mathbf{\Gamma}_{k|k}$, $\hat{\mathbf{x}}_{k|N}$ sont stockées dans les listes $\mathbf{x_forw\{k\}}$, $\mathbf{x_up\{k\}}$, $\mathbf{G_forw\{k\}}$, $\mathbf{G_up\{k\}}$, $\mathbf{x_back\{k\}}$, $\mathbf{G_back\{k\}}$, où *forw*, *up*, *back* signifient *forward*, *update*, *backward*. Les quantités \mathbf{u}_k , $\mathbf{\Gamma}_\alpha(k)$, \mathbf{y}_k , $\mathbf{\Gamma}_\beta(k)$, \mathbf{A}_k sont, elles aussi, stockées dans des listes. La partie directe (ou *forward*) du lisseur, correspondant au filtre de Kalman, est donnée par le script :

```
x_forw{1} = ... ; G_forw{1} = ... ; % initialisation
for k=1 :kmax,
[x_forw{k+1}, G_forw{k+1}, x_up{k}, G_up{k}]
=kalman(x_forw{k}, G_forw{k}, u{k}, y{k}, Galpha{k}, Gbeta{k}, A{k}, C{k}) ;
end ;
```

Quant à la partie rétrograde (ou *backward*) du lisseur, elle est donnée par :

```
x_back{kmax} = x_up{kmax} ;
G_back{kmax} = G_up{kmax} ;
for k=kmax-1 :-1 :1,
```

```

J=G_up{k}*A'/G_forw{k+1} ;
x_back{k}=x_up{k}+J*(x_back{k+1}-x_forw{k+1}) ;
G_back{k}=G_up{k}+J*(G_back{k+1}-G_forw{k+1})*J' ;
end ;

```

Notons que du fait de la spécificité de MATLAB, qui indice les listes à partir de 1, la condition initiale correspond à $k = 1$.

7.6. Exercices

Exercice 7.1. Distribution gaussienne

La distribution de probabilité d'un vecteur aléatoire gaussien \mathbf{x} est entièrement caractérisée par son espérance $\bar{\mathbf{x}}$ et sa matrice de covariance $\mathbf{\Gamma}_x$. Plus précisément, elle est donnée par :

$$\pi_x(\mathbf{x}) = \frac{1}{\sqrt{(2\pi)^n \det(\mathbf{\Gamma}_x)}} \cdot \exp\left(-\frac{1}{2} (\mathbf{x} - \bar{\mathbf{x}})^T \mathbf{\Gamma}_x^{-1} (\mathbf{x} - \bar{\mathbf{x}})\right)$$

1) Tracer le graphe et les courbes de niveaux de π_x avec :

$$\bar{\mathbf{x}} = \begin{pmatrix} 1 \\ 2 \end{pmatrix} \text{ et } \mathbf{\Gamma}_x = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

2) On définit le vecteur aléatoire :

$$\mathbf{y} = \begin{pmatrix} \cos \frac{\pi}{6} & -\sin \frac{\pi}{6} \\ \sin \frac{\pi}{6} & \cos \frac{\pi}{6} \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & 3 \end{pmatrix} \mathbf{x} + \begin{pmatrix} 2 \\ -5 \end{pmatrix}$$

Tracer le graphe et les courbes de niveaux de π_y . Interpréter.

Exercice 7.2. Ellipses de confiance

Sous MATLAB, on génère six matrices de covariance comme suit :

```

A1=[1 0 ;0 3] ; A2=[cos(pi/4) -sin(pi/4) ;sin(pi/4) cos(pi/4)] ;
G1=eye(2,2) ; G2=3*eye(2,2) ; G3=A1*G2*A1'+G1 ; G4=A2*G3*A2' ;
G5=G4+G3 ; G6=A2*G5*A2' ;

```

Ici, A2 correspond à une matrice de rotation d'angle $\frac{\pi}{4}$. Ensuite, on trace les six ellipsoïdes de confiance à 90 % associées à ces matrices en les centrant autour de 0. On obtient la figure 7.10.

1) Indiquer sur la figure à quelle matrice de covariance est associée chacune des ellipsoïdes.

2) Vérifier le résultat en regénérant ces ellipsoïdes sous MATLAB.

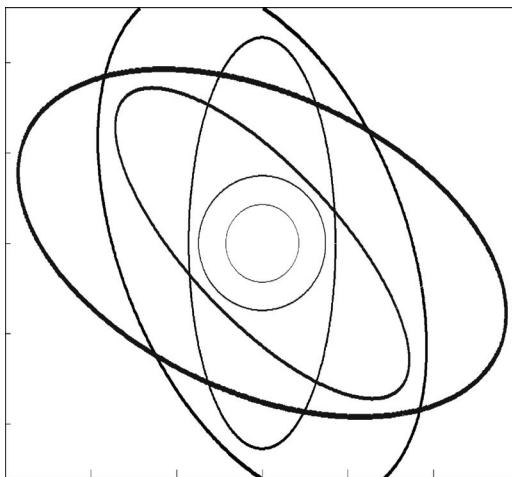


Figure 7.10. Ellipsoïdes de confiance associées aux six matrices de covariance

Exercice 7.3. Ellipsoïde de confiance : prédiction

1) Sous MATLAB, générer un nuage de $n = 1\ 000$ points représentant un vecteur aléatoire gaussien centré de \mathbb{R}^2 et dont la matrice de covariance est l'identité. Déduire du nuage précédent un nuage pour le vecteur aléatoire \mathbf{x} tel que :

$$\bar{\mathbf{x}} = \begin{pmatrix} 1 \\ 2 \end{pmatrix} \text{ et } \Gamma_{\mathbf{x}} = \begin{pmatrix} 3 & 1 \\ 1 & 3 \end{pmatrix}$$

On rangera les nuages dans une matrice à deux lignes et n colonnes.

2) Tracer les ellipsoïdes de confiance pour les probabilités $\eta \in \{0.9, 0.99, 0.999\}$.

3) Retrouver une estimation de $\bar{\mathbf{x}}$ et $\Gamma_{\mathbf{x}}$ à partir du nuage pour \mathbf{x} .

4) Cette distribution représente la connaissance que l'on a sur les conditions initiales d'un système (un robot par exemple) décrit par des équations d'état de la forme :

$$\dot{\mathbf{x}} = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \mathbf{x} + \begin{pmatrix} 2 \\ 3 \end{pmatrix} u$$

où l'entrée $u(t) = \sin(t)$ est connue. Faire un programme qui illustre l'évolution de ce nuage de particules avec le temps. On prendra pour période d'échantillonnage $\delta = 0.01$ sec.

5) Représenter cette évolution uniquement avec un calcul sur les ellipsoïdes de confiance.

Exercice 7.4. Ellipsoïde de confiance : correction

1) Tout comme dans l'exercice précédent, générer un nuage gaussien de $n = 1\,000$ points associé au vecteur aléatoire \mathbf{x} avec :

$$\bar{\mathbf{x}} = \begin{pmatrix} 1 \\ 2 \end{pmatrix} \text{ et } \Gamma_{\mathbf{x}} = \begin{pmatrix} 3 & 1 \\ 1 & 3 \end{pmatrix}$$

On rangera les nuages dans une matrice à deux lignes et n colonnes. Vérifier la cohérence du générateur aléatoire en le comparant avec le générateur `mvnrnd` de MATLAB (où *mvn* signifie *multivariate normal distribution*).

2) Trouver un estimateur linéaire non biaisé et orthogonal qui permet de retrouver x_1 à partir de x_2 . Tracer l'estimateur.

3) Reprendre cette même question, mais pour retrouver x_2 à partir de x_1 . Tracer l'estimateur et interpréter la différence avec la question précédente.

Exercice 7.5. Propagation de matrices de covariance

On considère trois vecteurs aléatoires $\mathbf{a}, \mathbf{b}, \mathbf{c}$ centrés et de matrices de covariance égales à l'identité. Ces trois vecteurs sont indépendants entre eux. Soient \mathbf{x}, \mathbf{y} les deux vecteurs aléatoires définis comme suit :

$$\begin{aligned} \mathbf{x} &= \mathbf{A} \mathbf{a} - \mathbf{b} \\ \mathbf{y} &= \mathbf{C} \mathbf{x} + \mathbf{c} \end{aligned}$$

où \mathbf{A}, \mathbf{C} sont des matrices connues.

1) Donner l'expression des espérances mathématiques $\bar{\mathbf{x}}, \bar{\mathbf{y}}$ et des matrices de covariance $\Gamma_{\mathbf{x}}, \Gamma_{\mathbf{y}}$ de ces deux vecteurs, en fonction de \mathbf{A} et \mathbf{C} .

2) On forme le vecteur $\mathbf{v} = (\mathbf{x}, \mathbf{y})$. Calculer l'espérance mathématique $\bar{\mathbf{v}}$ et la matrice de covariance $\Gamma_{\mathbf{v}}$ pour \mathbf{v} .

3) Déduire de la question précédente la matrice de covariance du vecteur aléatoire $\mathbf{z} = \mathbf{y} - \mathbf{x}$. On suppose bien sûr que \mathbf{x} et \mathbf{y} ont la même dimension.

4) On fait une mesure \mathbf{y} , c'est-à-dire que désormais, le vecteur aléatoire \mathbf{y} devient déterministe et est connu parfaitement. Donner une estimation $\hat{\mathbf{x}}$ pour \mathbf{x} en utilisant un estimateur linéaire, non biaisé et orthogonal.

Exercice 7.6. Bruit brownien

On considère un signal aléatoire stationnaire, discréteisé, blanc et centré. Ce signal est noté $x(t_k)$ avec $k \in \mathbb{N}$. Plus précisément, pour chaque $t_k = k\delta$ les variables

aléatoires $x(t_k)$ de variance σ_x^2 sont indépendantes entre elles. Un bruit brownien est défini comme l'intégrale d'un bruit blanc. Dans notre cas, nous formons le bruit brownien comme suit :

$$y(t_k) = \delta \cdot \sum_{i=0}^k x(t_k)$$

1) Calculer en fonction du temps, la variance $\sigma_y^2(t_k)$ du signal $y(t_k)$. Comment l'écart-type $\sigma_y(t_k)$ évolue-t-il en fonction de δ et en fonction de t_k ? Interpréter. Valider le résultat par une simulation sous MATLAB.

2) On fait tendre δ vers 0. Quel écart-type σ_x doit-on choisir en fonction de δ afin que les variances $\sigma_y^2(t_k)$ ne changent pas? Illustrer avec un programme MATLAB qui génère des bruits browniens $y(t)$ insensibles aux changements de périodes d'échantillonnage.

Exercice 7.7. Estimateur linéaire pour la résolution de trois équations

L'estimateur linéaire peut être utilisé pour résoudre des problèmes pouvant se traduire par des équations linéaires. Considérons à titre d'illustration le système :

$$\begin{cases} 2x_1 + 3x_2 = 8 \\ 3x_1 + 2x_2 = 7 \\ x_1 - x_2 = 0 \end{cases}$$

Comme l'on a plus d'équations que d'inconnues, l'estimateur linéaire doit faire une sorte de compromis entre toutes ces équations. Supposons que les erreurs ε_i sur la $i^{\text{ème}}$ équation sont centrées et de variances : $\sigma_1^2 = 1$, $\sigma_2^2 = 4$ et $\sigma_3^2 = 4$. Résoudre le système en utilisant un estimateur linéaire et donner la matrice de covariance associée.

Exercice 7.8. Estimateur linéaire pour l'estimation des paramètres d'un moteur électrique

Considérons un moteur à courant continu dont les paramètres avaient été estimés avec une méthode des moindres-carrés (voir exercice 6.3). Rappelons que dans cet exemple, la vitesse angulaire Ω d'un moteur à courant continu vérifie la relation :

$$\Omega = x_1 U + x_2 T_r$$

où U est la tension d'entrée, T_r est le couple résistant et $\mathbf{x} = (x_1, x_2)$ est le vecteur des paramètres qu'il nous faut estimer. Le tableau suivant rappelle les mesures prélevées sur le moteur pour différentes conditions expérimentales :

$U(\text{V})$	4	10	10	13	15
$T_r(\text{Nm})$	0	1	5	5	3
$\Omega(\text{rad/sec})$	5	10	8	14	17

Nous supposons que la variance de l'erreur de mesure vaut 9 et ne dépend pas des conditions expérimentales. De plus, supposons que nous sachions *a priori* que $x_1 \simeq 1$ et $x_2 \simeq -1$ avec une variance de 4. Estimer les paramètres du moteur et donner la matrice de covariance associée.

Exercice 7.9. Trochoïde

1) Une masse ponctuelle (posée sur une roue) se déplace suivant une trochoïde de la forme :

$$\begin{cases} x(t) = p_1 t - p_2 \sin t \\ y(t) = p_1 - p_2 \cos t \end{cases}$$

où x correspond à l'abscisse et y à l'altitude de la masse. On mesure y pour différents instants t :

t (sec)	1	2	3	7
y (m)	0.38	3.25	4.97	-0.26

Les erreurs de mesure ont un écart-type de 10 cm. En utilisant un filtre orthogonal non biaisé, calculer une estimation pour p_1 et p_2 .

2) Tracer sous MATLAB la trajectoire estimée de la masse.

Exercice 7.10. Filtre de Kalman pour la résolution de trois équations

Considérons à nouveau les équations linéaires considérées à l'exercice 7.7 :

$$\begin{cases} 2x_1 + 3x_2 = 8 + \beta_1 \\ 3x_1 + 2x_2 = 7 + \beta_2 \\ x_1 - x_2 = 0 + \beta_3 \end{cases}$$

où $\beta_1, \beta_2, \beta_3$ sont trois variables aléatoires indépendantes, centrées et de variances respectives 1, 4, 4.

1) En appelant le filtre de Kalman trois fois, résoudre sous MATLAB ce système. On donnera une estimation de la solution et la matrice de covariance de l'erreur.

2) Tracer les ellipsoïdes de confiance associés à chaque appel.

3) Comparer avec les résultats obtenus à l'exercice 7.7.

Exercice 7.11. Filtre de Kalman sur trois pas

Considérons le système à temps discret :

$$\begin{cases} \mathbf{x}_{k+1} = \mathbf{A}_k \mathbf{x}_k + \mathbf{u}_k + \alpha_k \\ y_k = \mathbf{C}_k \mathbf{x}_k + \beta_k \end{cases}$$

avec $k \in \{0, 1, 2\}$. Les valeurs pour les quantités \mathbf{A}_k , \mathbf{C}_k , \mathbf{u}_k , y_k sont données par :

k	\mathbf{A}_k	\mathbf{u}_k	\mathbf{C}_k	y_k
0	$\begin{pmatrix} 0.5 & 0 \\ 0 & 1 \end{pmatrix}$	$\begin{pmatrix} 8 \\ 16 \end{pmatrix}$	$\begin{pmatrix} 1 & 1 \end{pmatrix}$	7
1	$\begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix}$	$\begin{pmatrix} -6 \\ -18 \end{pmatrix}$	$\begin{pmatrix} 1 & 1 \end{pmatrix}$	30
2	$\begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix}$	$\begin{pmatrix} 32 \\ -8 \end{pmatrix}$	$\begin{pmatrix} 1 & 1 \end{pmatrix}$	-6

Supposons que les signaux α_k et β_k soient blancs gaussiens avec une matrice de covariance unitaire, c'est-à-dire :

$$\Gamma_\alpha = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \text{ et } \Gamma_\beta = 1$$

Le vecteur d'état initial est inconnu et est représenté par une estimation $\hat{\mathbf{x}}_{0|-1}$ et une matrice de covariance $\Gamma_{0|-1}$. Nous prendrons :

$$\hat{\mathbf{x}}_{0|-1} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \quad \Gamma_{0|-1} = \begin{pmatrix} 100 & 0 \\ 0 & 100 \end{pmatrix}$$

Sous MATLAB, tracer les ellipsoïdes de confiance de centre $\hat{\mathbf{x}}_{k|k}$ et de matrice de covariance $\Gamma_{k|k}$ obtenues par le filtre de Kalman.

Exercice 7.12. Estimation des paramètres d'un moteur électrique

Considérons à nouveau le moteur à courant continu de vitesse angulaire Ω (voir les exercices 6.3 à 7.8). Nous avons :

$$\Omega = x_1 U + x_2 T_r$$

où U est la tension d'entrée, T_r est le couple résistant et $\mathbf{x} = (x_1, x_2)$ est le vecteur des paramètres à estimer. Le tableau suivant rappelle les mesures collectées pour différentes conditions expérimentales :

k	0	1	2	3	4
$U(\text{V})$	4	10	10	13	15
$T_r(\text{Nm})$	0	1	5	5	3
$\Omega(\text{rad/sec})$	5	10	11	14	17

Nous supposons toujours que la variance pour l'erreur de mesure vaille 9 et que $x_1 \simeq 1$ et $x_2 \simeq -1$ avec une variance de 4. En utilisant le filtre de Kalman, calculer une estimation des paramètres x_1, x_2 et donner la matrice de covariance associée.

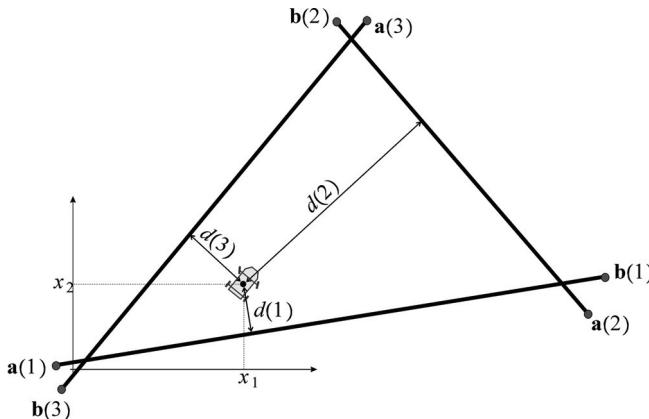


Figure 7.11. Le robot doit se localiser à partir de la mesure de sa distance aux trois murs

Exercice 7.13. Localisation à partir de la mesure des distances aux murs

On considère un robot supposé ponctuel positionné en $\mathbf{x} = (x_1, x_2)$. Il mesure sa distance à trois murs, comme sur la figure 7.11. Le $i^{\text{ème}}$ mur correspond à une droite repérée par deux points $\mathbf{a}(i)$ et $\mathbf{b}(i)$. La distance au $i^{\text{ème}}$ mur est :

$$d(i) = \det(\mathbf{u}(i), \mathbf{x} - \mathbf{a}(i)) + \beta_i$$

avec $\mathbf{u}(i) = \frac{\mathbf{b}(i) - \mathbf{a}(i)}{\|\mathbf{b}(i) - \mathbf{a}(i)\|}$. Chaque distance est mesurée avec une erreur β_i centrée de variance 1 et toutes les erreurs sont indépendantes entre elles. Avant toute prise de mesures, le robot pense qu'il est à la position $\bar{\mathbf{x}} = (1, 2)$ avec une matrice de covariance associée donnée par $100 \cdot \mathbf{I}$ où \mathbf{I} est la matrice identité.

1) Donner, en fonction des $\mathbf{a}(i)$, $\mathbf{b}(i)$, $d(i)$, une estimation de la position du robot ainsi que la matrice de covariance pour l'erreur. On pourra utiliser pour cela l'expression de l'estimateur linéaire orthogonal non biaisé ou bien de façon équivalente l'expression du filtre de Kalman en mode correction.

2) Les coordonnées des points ainsi que les distances sont données par :

i	1	2	3
$\mathbf{a}(i)$	$\begin{pmatrix} 2 \\ 1 \end{pmatrix}$	$\begin{pmatrix} 15 \\ 5 \end{pmatrix}$	$\begin{pmatrix} 3 \\ 12 \end{pmatrix}$
$\mathbf{b}(i)$	$\begin{pmatrix} 15 \\ 5 \end{pmatrix}$	$\begin{pmatrix} 3 \\ 12 \end{pmatrix}$	$\begin{pmatrix} 2 \\ 1 \end{pmatrix}$
$d(i)$	2	5	4

Concevoir un programme MATLAB qui nous donne l'estimation demandée.

Exercice 7.14. Estimation de la température

La température dans une pièce est censée vérifier (après discréétisation temporelle) l'équation d'état :

$$\begin{cases} x_{k+1} = x_k + \alpha_k \\ y_k = x_k + \beta_k \end{cases}$$

Nous supposons que le bruit d'état α_k et le bruit de mesure β_k sont indépendants et gaussiens de covariance $\Gamma_\alpha = 4$ et $\Gamma_\beta = 3$.

1) Donner l'expression du filtre de Kalman permettant d'estimer la température x_k à partir de la mesure y_k . En déduire une expression de $\hat{x}_{k+1|k}$ et $\Gamma_{k+1|k}$ en fonction de $\hat{x}_{k|k-1}$, $\Gamma_{k|k-1}$, y_k .

2) Pour k assez grand, on peut supposer que $\Gamma_{k+1|k} = \Gamma_{k|k-1} = \Gamma_\infty$. On obtient alors le filtre de Kalman dit *asymptotique*. Donner l'expression du filtre de Kalman asymptotique. Comment caractériser la précision de ce filtre ?

3) On revient dans le cas non asymptotique, mais on suppose maintenant que $\Gamma_{\alpha_k} = 0$. Quelle est la valeur de Γ_∞ ? Interpréter.

Exercice 7.15. Marcheur aveugle

On considère un marcheur aveugle se déplaçant sur une droite horizontale. Son déplacement est décrit par l'équation d'état discréétisée :

$$\begin{cases} x_1(k+1) = x_1(k) + x_2(k) \cdot u(k) \\ x_2(k+1) = x_2(k) + \alpha_2(k) \end{cases}$$

où $x_1(k)$ est la position du marcheur, $x_2(k)$ est la longueur d'un pas (appelé le *facteur d'échelle*) et $u(k)$ le nombre de pas par unité de temps. On mesure uniquement la quantité $u(k)$. Ainsi, à chaque unité de temps, le marcheur se déplace de la distance $x_2(k)u(k)$. A l'instant initial, on sait que x_1 est nul et que x_2 est proche de 1. On représentera $x_2(0)$ par une distribution gaussienne de moyenne 1 et d'écart-type 0.02. Le facteur d'échelle x_2 évolue de façon lente par l'intermédiaire de $\alpha_2(k)$ que l'on supposera centré, blanc et d'écart-type 0.01.

1) On applique une entrée $u(k) = 1$ pour $k = 0, \dots, 9$ et $u(k) = -1$ pour $k = 10, \dots, 19$. Faire un programme MATLAB qui implémente un filtre de Kalman prédicteur capable d'estimer la position $x_1(k)$.

2) Tracer les ellipsoïdes de confiance associées à la probabilité $\eta = 0.99$. Comment évolue l'incertitude pour x_1 en fonction de k ?

3) Tracer, en fonction de k , le déterminant de la matrice de covariance Γ_x . Interpréter.

Exercice 7.16. Pendule simple

On considère le pendule de la figure 7.12. L'entrée de ce système est le couple u exercé sur le pendule.

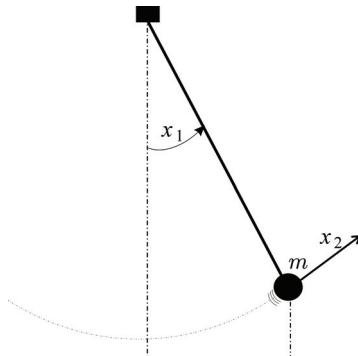


Figure 7.12. Pendule simple de vecteur d'état $\mathbf{x} = (x_1, x_2)$

Sa représentation d'état est supposée être :

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} x_2 \\ -\sin x_1 + u \end{pmatrix}$$

Il s'agit bien sûr d'un modèle normalisé où les coefficients (masse, gravité, longueur) ont tous été fixés à 1.

1) On souhaite que la position $x_1(t)$ du pendule soit égale à une consigne $w(t)$ pouvant varier dans le temps. En utilisant une méthode de linéarisation par bouclage, proposer un régulateur par retour d'état tel que l'erreur $e = w - x_1$ converge vers 0 en $\exp(-t)$ (ce qui signifie que l'on place les pôles en -1). Il faudra pour cela écrire l'expression de u en fonction de \mathbf{x} , w , \dot{w} , \ddot{w} .

2) On souhaite que l'angle x_1 du pendule soit égal à $\sin t$ une fois le régime transitoire passé. Quelle expression de la commande $u(t)$ nous faudra-t-il choisir ? Il faudra donner l'expression de u en fonction de \mathbf{x} et de t .

3) Pour implémenter la commande proposée à la question précédente, il est nécessaire que tout l'état soit disponible. Or, nous disposons uniquement d'un gyromètre positionné sur l'axe du pendule qui nous délivre une mesure de x_2 toutes les $\delta = 0.01$ sec, avec une erreur gaussienne blanche d'écart-type 0.1rad/sec . Il nous faut donc reconstruire l'état \mathbf{x} du pendule pour implémenter notre commande. Ceci peut se faire grâce à un observateur d'état. Ici, nous proposons d'utiliser un filtre de Kalman pour réaliser cet observateur. Auparavant, il faut linéariser ce système

autour du point $\mathbf{x} = (0, 0)$ et le discréteriser avec une période d'échantillonnage de δ . Donner les arguments $(y_k, u_k, \Gamma_{\alpha_k}, \Gamma_{\beta_k}, \mathbf{A}_k, \mathbf{C}_k)$ à fournir au filtre de Kalman à chaque itération k pour que ce dernier nous génère une estimée $\hat{\mathbf{x}}$ de notre état.

4) Proposer sous la forme d'un diagramme de type schéma bloc, l'ensemble du système, en faisant apparaître les entrées et les sorties de chaque sous-système (pendule, observateur et régulateur).

Exercice 7.17. Estimation d'état pour le pendule inversé

On considère un pendule inversé dont les équations d'état sont données par :

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{pmatrix} = \begin{pmatrix} x_3 \\ x_4 \\ \frac{m \sin x_2 (g \cos x_2 - \ell x_4^2) + u}{M + m \sin^2 x_2} \\ \frac{\sin x_2 ((M+m)g - m \ell x_4^2 \cos x_2) + \cos x_2 u}{\ell (M + m \sin^2 x_2)} \end{pmatrix} \text{ et } y = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

Ici, nous avons pris pour vecteur d'état $\mathbf{x} = (x, \theta, \dot{x}, \dot{\theta})$, où l'entrée u est la force exercée sur le chariot de masse M , x est la position du chariot et θ est l'angle entre le pendule et la verticale. Ici, nous supposons que seuls la position du chariot x et l'angle θ du pendule sont mesurés.

1) Linéariser ce système autour de l'état $\mathbf{x} = \mathbf{0}$.

2) Proposer un régulateur par retour d'état de la forme $u = -\mathbf{K} \cdot \mathbf{x} + h w$ qui stabilise le système. On utilisera pour cela une méthode par placement de pôles (instruction `place` de MATLAB). On prendra les pôles tous égaux à -2 . Pour le précompensateur h , on prendra une consigne w qui correspond à la position désirée pour le chariot. On rappelle [JAU 14] qu'il faut prendre :

$$h = -(\mathbf{E} \cdot (\mathbf{A} - \mathbf{B} \cdot \mathbf{K})^{-1} \cdot \mathbf{B})^{-1}$$

où \mathbf{E} est la matrice de consigne donnée par :

$$\mathbf{E} = (1 \ 0 \ 0 \ 0)$$

Simuler le système contrôlé par ce retour d'état.

3) Pour effectuer un retour de sortie, nous avons besoin d'une estimation $\hat{\mathbf{x}}$ du vecteur d'état \mathbf{x} . Pour cela, nous allons utiliser un filtre de Kalman (voir figure 7.13).

Faire une discréterisation du système avec un pas de $dt = 0.01$ sec puis proposer un filtre de Kalman pour faire l'observation de l'état.

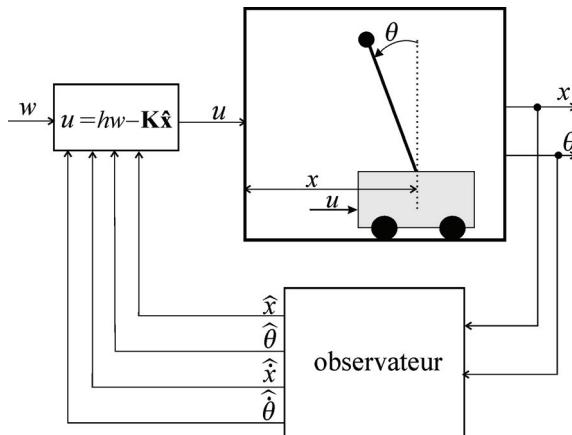


Figure 7.13. Filtre de Kalman utilisé pour estimer l'état du pendule inversé

4) Implémenter sous MATLAB ce filtre. On prendra un bruit d'état blanc centré gaussien de variance $\sigma_x^2 = dt \cdot 0.01$ pour les variables x, θ . Pour le bruit de mesure, nous prendrons un bruit blanc gaussien centré et de variance $\sigma_y^2 = 0.01^2$. Etudier la robustesse de l'observateur lorsque l'on augmente de bruit de mesure. Tracer, sur une figure séparée, les ellipses de confiance dans l'espace (x, θ) et vérifier si la valeur vraie (x^*, θ^*) est dans l'ellipse.

5) Un Kalman étendu peut être obtenu en remplaçant, dans l'étape de prédiction du filtre de Kalman, l'instruction :

$$\hat{\mathbf{x}}_{k+1|k} = \mathbf{A}_k \hat{\mathbf{x}}_{k|k} + \mathbf{B}_k \mathbf{u}_k$$

par :

$$\hat{\mathbf{x}}_{k+1|k} = \hat{\mathbf{x}}_{k|k} + \mathbf{f}(\hat{\mathbf{x}}_{k|k}, \mathbf{u}_k) \cdot dt$$

On a donc ici remplacé la prédiction faite sur le modèle linéarisé par une prédiction faite sur le modèle non linéaire initial plus proche de la réalité. Proposer une implémentation de ce filtre de Kalman étendu.

Exercice 7.18. Localisation à l'estime

La *localisation à l'estime* (ou *dead reckoning*) correspond au problème de localisation dans le cas où seuls des capteurs proprioceptifs sont disponibles. Ce type de navigation était utilisé par les navigateurs anciens qui cherchaient à se localiser les longues traversées. Ils parvenaient très approximativement à se localiser en mesurant le cap du bateau, la vitesse à différents instants puis en intégrant sur tout le trajet déjà effectué toutes les variations de position correspondantes. Dans un contexte plus

général, nous pouvons considérer qu'utiliser un observateur d'état en mode prédition et sans correction (dans le cas particulier où l'état est la position du robot) correspond à de la navigation à l'estime. Prenons le robot représenté sur la figure 7.14 et dont les équations d'état sont :

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{v} \\ \dot{\delta} \end{pmatrix} = \begin{pmatrix} v \cos \delta \cos \theta \\ v \cos \delta \sin \theta \\ \frac{v \sin \delta}{3} + \alpha_\theta \\ u_1 + \alpha_v \\ u_2 + \alpha_\delta \end{pmatrix}$$

où $\alpha_\theta, \alpha_v, \alpha_\delta$ sont des bruits blancs gaussiens à temps continu et indépendants entre eux. Plus rigoureusement, il s'agit de distributions aléatoires de puissance infinie, mais une fois ces distributions discrétisées, les difficultés mathématiques disparaissent.

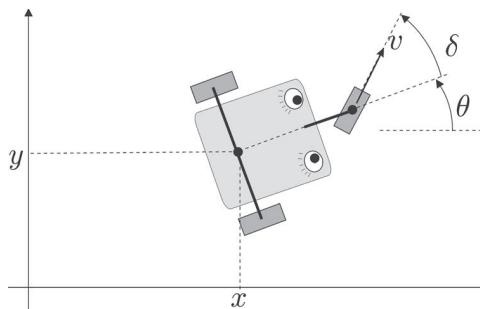


Figure 7.14. Tricycle qui cherche à se localiser à l'estime

Le robot est muni d'une boussole qui renvoie θ avec une bonne précision et d'un capteur d'angle qui renvoie l'angle δ de la roue avant.

- 1) Discréteriser ce système avec une méthode d'Euler. Simuler ce système sous MATLAB pour une entrée $\mathbf{u}(t)$ et un vecteur initial arbitraires. Pour la variance des bruits discrétisés $\alpha_\theta, \alpha_v, \alpha_\delta$ nous prendrons $0.01 \cdot dt$, où dt est le pas de discrétisation.
- 2) Exprimer ce problème de localisation sous une forme linéaire et discrétisée.
- 3) A l'aide d'un filtre de Kalman, prédire la position du robot ainsi que la matrice de covariance associée.
- 4) Que devient le programme de localisation si l'on suppose que grâce à des odomètres, le robot est capable de mesurer sa vitesse v avec une variance de 0.01 ?

Exercice 7.19. Localisation goniométrique

On considère à nouveau un robot de type voiture décrit par les équations d'état :

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \\ \dot{x}_5 \end{pmatrix} = \begin{pmatrix} x_4 \cos x_5 \cos x_3 \\ x_4 \cos x_5 \sin x_3 \\ \frac{x_4 \sin x_5}{3} \\ u_1 \\ u_2 \end{pmatrix}$$

Le vecteur (x_1, x_2) représente les coordonnées du centre du robot, x_3 est le cap du robot, x_4 sa vitesse et x_5 l'angle des roues avant. Le robot est entouré d'amers ponctuels $\mathbf{m}(1), \mathbf{m}(2), \dots$ dont les positions sont connues. Le robot perçoit ces amers $\mathbf{m}(i)$ seulement si la distance est suffisamment faible (inférieure à 15 m). Dans un tel cas, le robot mesure l'angle δ_i avec une grande précision. Nous supposerons aussi, qu'à chaque instant, le robot connaît les angles x_3 et x_5 sans erreur. Enfin, il mesure sa vitesse x_4 avec une erreur de variance 1. La figure 7.15 illustre une situation où deux amers $\mathbf{m}(1)$ et $\mathbf{m}(2)$ sont vus par le robot.

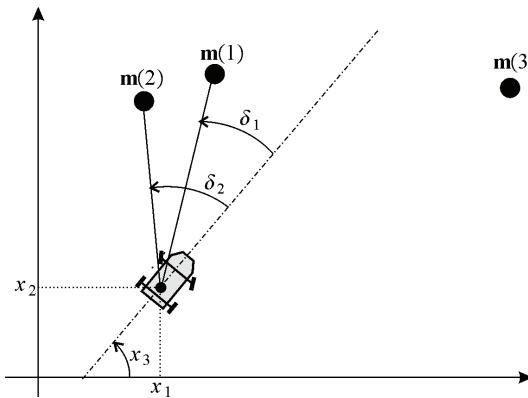


Figure 7.15. Localisation goniométrique

Afin de se localiser, on souhaite utiliser un filtre de Kalman. Pour cela, il nous faut des équations linéaires, ce qui n'est pas le cas ici. Puisque x_3 et x_5 sont connues, la non linéarité peut se fondre dans une dépendance temporelle. Posons pour cela $\mathbf{z} = (x_1, x_2, x_4)$.

1) Montrer que \mathbf{z} vérifie une équation d'évolution d'état linéaire. Donner l'équation d'observation associée.

2) Donner une discréttisation pour l'évolution de \mathbf{z} afin de pouvoir alimenter un filtre de Kalman.

3) Implémenter sous MATLAB un simulateur avec le robot entouré des quatre amers suivantes :

$$\mathbf{a}(1) = \begin{pmatrix} 0 \\ 25 \end{pmatrix}, \mathbf{a}(2) = \begin{pmatrix} 15 \\ 30 \end{pmatrix}, \mathbf{a}(3) = \begin{pmatrix} 30 \\ 15 \end{pmatrix}, \mathbf{a}(4) = \begin{pmatrix} 15 \\ 20 \end{pmatrix}$$

comme indiqué précédemment, le robot perçoit de façon goniométrique seulement les amers qui lui sont proches.

4) Implémenter un filtre de Kalman pour la localisation. L'état initial sera supposé inconnu.

5) On a maintenant deux robots \mathcal{R}_a et \mathcal{R}_b qui peuvent communiquer en wifi, tout en mesurant les angles des amers (voir figure 7.16). Lorsque les distances sont faibles (c'est-à-dire inférieures à 20 m), grâce à des caméras, les robots peuvent mesurer les angles φ_a et φ_b avec une bonne précision (voir figure). Proposer un filtre de Kalman centralisé pour la localisation des deux robots.

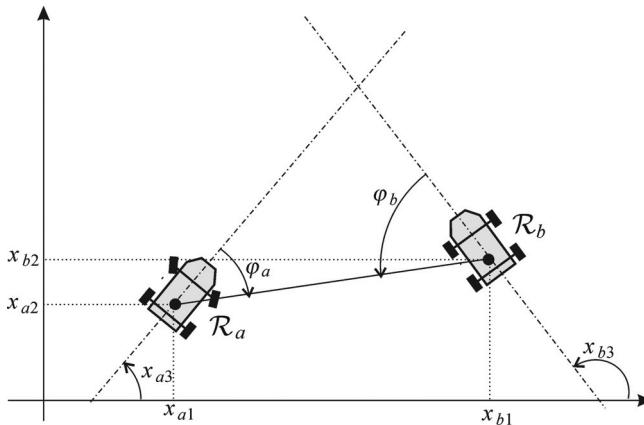


Figure 7.16. Localisation goniométrique pour deux robots communiquant

Exercice 7.20. Suivi d'un bateau par deux radars

Le mouvement d'un bateau que l'on cherche à suivre est décrit par les équations d'état :

$$\begin{cases} p_x(k+1) = p_x(k) + dt \cdot v_x(k) \\ v_x(k+1) = v_x(k) - dt \cdot v_x(k) + \alpha_x(k) \\ p_y(k+1) = p_y(k) + dt \cdot v_y(k) \\ v_y(k+1) = v_y(k) - dt \cdot v_y(k) + \alpha_y(k) \end{cases}$$

où $dt = 0.01$ et α_x et α_y sont des bruits blancs gaussiens de matrice de variance dt . Le vecteur d'état est donc $\mathbf{x} = (p_x, v_x, p_y, v_y)$.

- 1) Faire un programme MATLAB qui simule le bateau.
- 2) Deux radars positionnés en **a** : $(a_x, a_y) = (0, 0)$ et **b** : $(b_x, b_y) = (1, 0)$ mesurent le carré de la distance au bateau. L'équation d'observation est :
$$\mathbf{y}_k = \underbrace{\left(\frac{(p_x(k) - a_x)^2 + (p_y(k) - a_y)^2}{(p_x(k) - b_x)^2 + (p_y(k) - b_y)^2} \right)}_{\mathbf{g}(\mathbf{x}_k)} + \beta_k$$

où $\beta_1(k)$ et $\beta_2(k)$ sont des bruits blancs gaussiens indépendants unitaires. Adapter la simulation de façon à visualiser les radars et générer le vecteur de mesures $\mathbf{y}(k)$.
- 3) Effectuer une linéarisation de cette équation d'observation autour de l'estimation courante $\hat{\mathbf{x}}_k$ du vecteur d'état \mathbf{x}_k . En déduire une équation de la forme $\mathbf{z}_k = \mathbf{C}_k \mathbf{x}_k$ où $\mathbf{z}_k = \mathbf{h}(\mathbf{y}_k, \hat{\mathbf{x}}_k)$ prend le rôle de la mesure prélevée à l'instant k .
- 4) Implémenter un filtre de Kalman qui permette une localisation du bateau.

Exercice 7.21. Localisation d'un robot dans une piscine

On considère un robot sous-marin se déplaçant dans une piscine rectangulaire de longueur $2R_y$ et de largeur $2R_y$. Un sonar positionné juste en dessous du robot tourne avec une vitesse angulaire constante. La profondeur z s'obtient facilement avec un capteur de pression et nous supposerons donc cette grandeur connue. Le robot est supposé être lesté de telle façon que les angles de gîte et d'assiette puissent être considérés comme nuls. Dans notre contexte, se localiser signifie donc estimer les coordonnées (x, y) du robot. L'origine du repère sera pris au centre de la piscine. Pour cette localisation, nous supposons que nous mesurons l'angle α du sonar relativement au corps du robot, l'angle de cap θ par une boussole et les accélérations tangentielles a_T et normales a_N par l'intermédiaire d'accéléromètres. Le sonar nous renvoie toutes les 0.1s la longueur ℓ du faisceau sonar. La figure 7.17 représente la longueur $\ell(t)$ du faisceau sonar, obtenue par simulation, lorsque le sonar effectue sept tours sur lui-même, alors que le robot se déplace.

- 1) A partir du signal récupéré par le sonar, proposer une méthode robuste pour détecter les minimums locaux, qui correspondent à la situation où le sonar pointe perpendiculairement à un des quatre murs de la piscine.
- 2) Posons $v_x = \dot{x}$, $v_y = \dot{y}$. Montrer que nous pouvons écrire les relations d'état liants les mesures entre elles :

$$\begin{cases} \dot{x} = v_x \\ \dot{y} = v_y \\ \dot{v}_x = a_T \cos \theta - a_N \sin \theta \\ \dot{v}_y = a_T \sin \theta + a_N \cos \theta \end{cases}$$

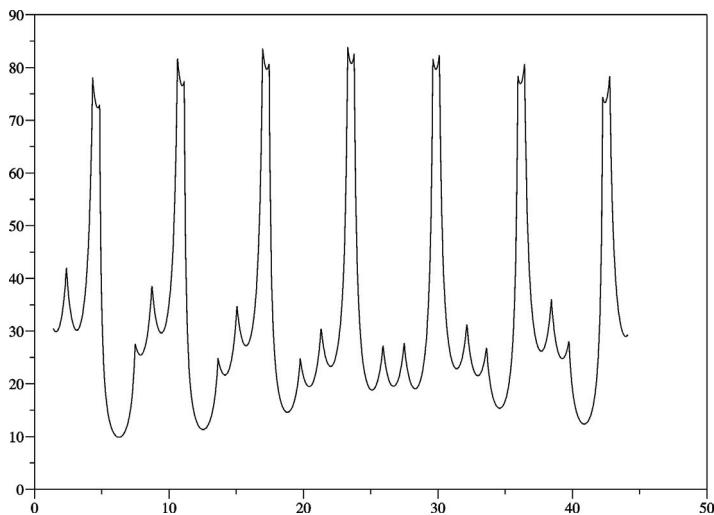


Figure 7.17. Mesures télémétriques recueillies par le robot

3) Nous supposons maintenant que nous ayons recueilli (soit par simulation, soit par une expérience réelle) pour chaque $t \in [0, t_{\max}]$, les accélérations (a_T, a_N) , l'angle de cap θ et l'angle du sonar α . Proposer une méthode récursive, basée sur le filtre de Kalman, pour localiser le robot.

Exercice 7.22. SLAM

Le *Redermor* (robot sous-marin du GESMA, Brest) a fait une mission de deux heures dans la baie de Douarnenez (voir figure 7.18). Tout au long de sa mission, il a recueilli les données de sa centrale inertielle (qui nous donne les angles d'Euler ϕ, θ, ψ), de son Loch-Doppler (qui nous donne la vitesse du robot \mathbf{v}_r dans le repère du robot), de son capteur de pression (qui nous donne la profondeur p_z du robot) et de son capteur d'altitude (écho-sondeur qui nous donne l'altitude a), avec une période d'échantillonnage de $dt = 0.1$ sec. Ces données sont dans le fichier `slam_data.txt`. Ce fichier est composé de 59 996 lignes (une ligne par instant d'échantillonnage) et de neuf colonnes correspondant respectivement à :

$$(t, \varphi, \theta, \psi, v_x, v_y, v_z, p_z, a)$$

où p_z est la profondeur du robot et a est son altitude (c'est-à-dire sa distance au fond).

1) Sachant que le robot est parti d'une position $\mathbf{p} = (0, 0, 0)$, l'instant $t = 0$, et en utilisant une méthode d'Euler, en déduire une estimation de la trajectoire. On utilisera pour cela l'équation d'état :

$$\dot{\mathbf{p}}(t) = \mathbf{R}(\varphi(t), \theta(t), \psi(t)) \cdot \mathbf{v}_r(t)$$

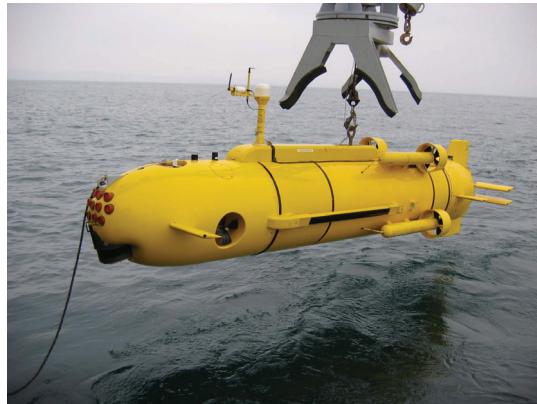


Figure 7.18. Le Redermor, construit par le GESMA (Groupe d'Etude Sous-Marine de l'Atlantique), juste avant d'être mis à l'eau

avec $\mathbf{R}(\varphi, \theta, \psi)$ étant la matrice d'Euler (voir [1.7]) dont nous rappelons ici l'expression :

$$\mathbf{R}(\varphi, \theta, \psi) = \begin{pmatrix} \cos \psi & -\sin \psi & 0 \\ \sin \psi & \cos \psi & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos \theta & 0 & \sin \theta \\ 0 & 1 & 0 \\ -\sin \theta & 0 & \cos \theta \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos \varphi & -\sin \varphi \\ 0 & \sin \varphi & \cos \varphi \end{pmatrix}$$

2) Les angles ψ, θ, φ sont mesurés avec un écart-type de $(2 \times 10^{-4}, 2 \times 10^{-4}, 5 \times 10^{-3})$. Les composantes de \mathbf{v}_r sont mesurées avec un écart-type de $\sigma_v = 1 \text{ ms}^{-1}$. On peut supposer que le robot satisfait l'équation :

$$\mathbf{p}_{k+1} = \mathbf{p}_k + (dt \cdot \mathbf{R}(k)) \cdot \bar{\mathbf{v}}_r(k) + \alpha_k$$

où α_k est un bruit blanc et $\bar{\mathbf{v}}_r(k)$ est une mesure de la vitesse moyenne sur la période d'échantillonnage correspondante. Montrer qu'une matrice de covariance réaliste pour α_k est :

$$\Gamma_\alpha = dt^2 \sigma_v^2 \cdot \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

3) En utilisant le filtre de Kalman comme un prédicteur, calculer la précision avec laquelle le robot connaît sa position pour chaque instant $t = k \cdot dt$. Donner, en fonction de t , l'écart-type de l'erreur sur la position. Que vaut-il au bout d'une heure ? Au bout de deux heures ? Vérifier expérimentalement le calcul sous MATLAB en implémentant un prédicteur de Kalman.

4) Lors de sa mission, le robot peut percevoir avec son sonar latéral quelques amers (ici des mines). Lorsque le robot perçoit un amer, ce dernier est sur sa droite et dans

un plan perpendiculaire au robot. Le fond de la mer sera supposé plat et horizontal. Le tableau suivant nous donne les instants de détection, les numéros i des amers et la distance r_i entre le robot et l'amer :

t	1 054	1 092	1 374	1 748	3 038	3 688	4 024	4 817	5 172	5 232	5 279	5 688
i	1	2	1	0	1	5	4	3	3	4	5	1
$r_i(t)$	52.42	12.47	54.40	52.68	27.73	26.98	37.90	36.71	37.37	31.03	33.51	15.05

Le SLAM (*Simultaneous Localization And Mapping*) cherche à utiliser ces détections répétées afin d'améliorer la précision de l'estimation sur la trajectoire. Pour cela, on forme un grand vecteur d'état \mathbf{x} , de dimension $3 + 2 \cdot 6 = 15$ qui contient la position du robot \mathbf{p} ainsi que le vecteur \mathbf{q} de dimension 12 contenant les coordonnées (en x et y) des six amers. Notons que comme les amers sont immobiles, on a $\dot{\mathbf{q}} = \mathbf{0}$. Donner la fonction MATLAB $[y, C, Gbeta] = g(k)$ correspondant à l'observation. Cette fonction renvoie le vecteur de mesure y , la matrice $C(k)$ et la matrice de covariance du bruit de mesure. Pour l'écart-type du bruit de mesure β_k nous prendrons 0.1 pour la profondeur et 1 pour la distance robot-amer.

5) En utilisant un filtre de Kalman, retrouver la position des amers avec l'incertitude associée. Montrer comment le robot a pu recalculer sa position.

6) Utiliser le lisseur de Kalman pour améliorer la précision sur la position des amers en prenant en compte le passé, mais aussi le futur.

Exercice 7.23. SLAM a priori

Un robot sous-marin, possède un système de navigation (centrale inertuelle et Loch-Doppler) qui lui donne sa position en x, y avec une dérive de 100 mètres par heure. Cela signifie que si le robot connaît sa position avec une précision de r mètres à l'instant t , alors une heure avant et une heure après, il connaît sa position avec une erreur inférieure à $r + 100$ m.

A l'instant initial, notre robot se localise par GPS avec une précision de 10 m et plonge, dans une zone où le fond est plat et se promène pendant huit heures à profondeur constante (qu'il peut mesurer avec un capteur de pression). Lorsqu'il remonte, il se localise à nouveau par GPS, toujours avec une précision de 10 m. Toutes les heures, le robot passe juste au-dessus d'un amer remarquable (petit rocher avec une forme spéciale par exemple) que l'on peut détecter à l'aide d'une caméra placée sous le robot. Le tableau suivant indique le numéro de l'amer détecté en fonction du temps exprimé en heure :

t(heure)	1	2	3	4	5	6	7
amer	1	2	1	3	2	1	4

De ce tableau, on peut déduire que le robot rencontre quatre amers remarquables en tout et qu'il rencontre trois fois l'amer 1, aux temps $t = 1H$, $t = 3H$ et $t = 6H$. Avec quelle précision est-il capable de localiser les amers 1, 2, 3 et 4 ?

7.7. Corrections

Correction de l'exercice 7.1 (distribution gaussienne)

1) Pour tracer le graphe de π_x , on écrit les lignes suivantes :

```
Mx1 = -5 :0.1 :5; Mx2 = -5 :0.1 :5; [X1,X2] = meshgrid(Mx1,Mx2);
xbar=[1 ;2] ; Gx=[1 0 ;0 1] ; invGx=inv(Gx)
dX1=X1-xbar(1) ; dX2=X2-xbar(2) ;
Q=invGx(1,1)*(dX1.^2)+2*(invGx(1,2)*dX1.*dX2)+(invGx(2,2)*dX2.^2);
Z=(1/(2*pi*sqrt(det(Gx)))*exp(-(1/2)*Q) ;
contour3(X1,X2,Z,20,'black') ;
surface(X1,X2,Z) ;
```

On obtient alors les résultats illustrés par les figures 7.19 (le script est rangé dans gauss.m).

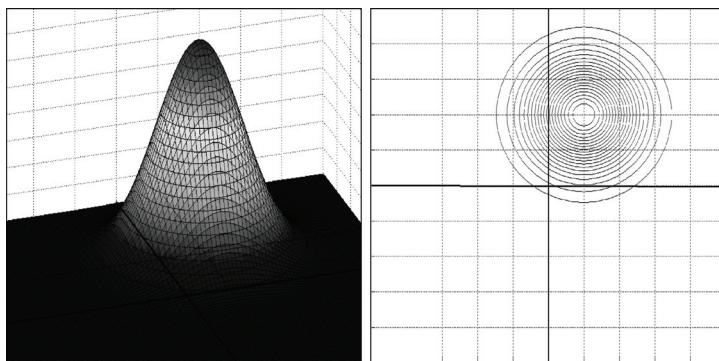


Figure 7.19. Densité de probabilité d'un vecteur aléatoire gaussien normé, mais non centré

2) On obtient les caractéristiques de la loi π_y grâce aux relations :

$$\begin{aligned}\bar{y} &= A\bar{x} + b \\ \Gamma_y &= A \cdot \Gamma_x \cdot A^T\end{aligned}$$

Sous MATLAB cela donne :

```
A=[cos(pi/6) -sin(pi/6) ;sin(pi/6) cos(pi/6)]*[1 0 ;0,3] ;
Gy=A*Gx*A' ; ybar=A*xbar+[2 ;-5] ;
```

La figure 7.20 illustre le résultat obtenu.

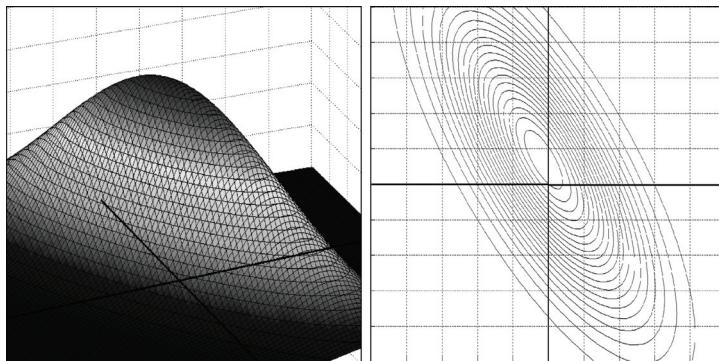


Figure 7.20. Densité du vecteur aléatoire y après transformation de x par une application linéaire

On remarque que, conformément à la relation linéaire liant x et y , π_y peut être obtenue à partir de π_x par une dilatation de 2 suivant x_1 suivie d'une rotation de $\frac{\pi}{6}$ puis de la translation. Le caractère gaussien est conservé.

Correction de l'exercice 7.2 (ellipses de confiance)

Les matrices $G1, \dots, G6$, sont dessinées sur la figure 7.21. Du fait des instructions $G1=eye(2,2)$ et $G2=3*eye(2,2)$, les ellipses associées à $G1$ et $G2$ sont des cercles et $G2$ s'obtient à partir de $G1$ par une homothétie de rapport $\sqrt{3}$. Puisque $G3=A1*G2*A1' + G1$ avec $A1=[1 \ 0 \ ; \ 0 \ 3]$ on peut obtenir $G3$ par une dilatation suivant x_2 de rapport 3 puis un gonflement de l'ellipse obtenue, du fait d'ajout de $G1$. Pour obtenir $G4$, on fait subir à $G3$ une rotation d'angle $\frac{\pi}{4}$ (car $G4=A2*G3*A2'$). Comme $G5=G4+G3$, $G5$ enferme les ellipses $G4$ et $G3$. A nouveau $G6$ est obtenue par une rotation d'angle $\frac{\pi}{4}$ de $G5$. Le programme associé est rangé dans `sixcov.m`.

Correction de l'exercice 7.3 (ellipsoïde de confiance : prédiction)

1) $n=1000$; $Gx=[3,1;1,3]$; $xbar=[2;3]$; $b=randn(2,n)$;
 $x=xbar*ones(1,n)+sqrtm(Gx)*b$; $plot(x(1, :), x(2, :), ' .')$;

2) Il faut faire les appels suivants :

`draw_ellipse(xbar,Gx,0.9)` ; `draw_ellipse(xbar,Gx,0.99)` ;
`draw_ellipse(xbar,Gx,0.999)` ;

3) $xhat=[mean(x(1, :)) ; mean(x(2, :))]$; $xtilde=x-xbar*ones(1,n)$;
 $g11=mean(xtilde(1, :).*xtilde(1, :))$; $g12=mean(xtilde(1, :).*xtilde(2, :))$;
 $g22=mean(xtilde(2, :).*xtilde(2, :))$; $Ghat=[g11,g12 ; g12,g22]$;

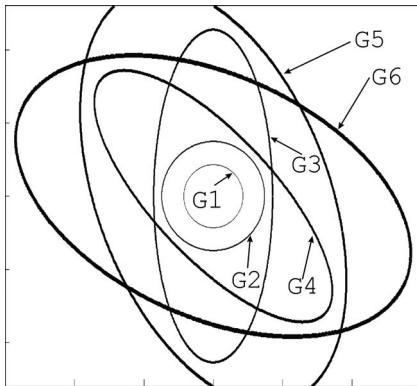


Figure 7.21. Ellipses de confiance de la plus fine à la plus épaisse

4) Le programme de simulation particulaire est :

```
dt=0.01 ; A=[0 1 ;-1 0] ; B=[2 ;3] ;
for t=0 :dt :5
Ad=eye(2,2)+dt*A ; ud=dt*sin(t)*B
x=Ad*x+ud*ones(1,n) ;
end
```

5) Par la formule d'Euler autour de $t = k\delta$, on a l'approximation :

$$\begin{aligned} \mathbf{x}(k+1) &= \mathbf{x}(k) + \delta \cdot \left(\begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \mathbf{x}(k) + \begin{pmatrix} 2 \\ 3 \end{pmatrix} u(t) \right) \\ &= \mathbf{A}\mathbf{x}(k) + \mathbf{B}u(k) \end{aligned}$$

avec :

$$\mathbf{A} = \begin{pmatrix} 1 & \delta \\ -\delta & 1 \end{pmatrix} \text{ et } \mathbf{B} = \begin{pmatrix} 2\delta \\ 3\delta \end{pmatrix}$$

Donc, la matrice de l'ellipsoïde se calcule comme suit :

$$\Gamma_{k+1} = \mathbf{A}\Gamma_k\mathbf{A}^T$$

et son centre $\hat{\mathbf{x}}$ par :

$$\hat{\mathbf{x}}(k+1) = \mathbf{A}\hat{\mathbf{x}}(k) + \mathbf{B}u(k)$$

Dans la boucle du programme, il nous faut donc rajouter les instructions suivantes : $xbar=Ad*xbar+ud$ et $Gx=Ad*Gx*Ad'$. Bien évidemment, ces instructions demandent un temps de calcul négligeable devant celui qu'il nous faut faire si l'on manipule

le nuage de particules (voir question 4). Dans un contexte linéaire gaussien, on manipulera donc directement les matrices de covariance et les moyennes, ce qui nous amènera au filtre de Kalman. Dans un contexte non linéaire, une manipulation du nuage de particules devra souvent être privilégiée. L'ensemble du programme associé à cet exercice est rangé dans `predicov.m`.

Correction de l'exercice 7.4 (ellipsoïde de confiance : correction)

```
1) n=1000 ; Gx=[3,1 ; 1,3] ; xbar=[1 ; 2] ; b=randn(2,n) ;
x=xbar*ones(1,n)+sqrtm(Gx)*b; plot(x(1, :),x(2, :),'.')
```

Pour générer ce même nuage plus rapidement, nous aurions pu écrire `x=mvnrnd(xbar,Gx,n)'`.

2) On a $\hat{x}_1 = \bar{x}_1 + K \cdot (x_2 - \bar{x}_2)$ avec $K = \Gamma_{x_1 x_2} \Gamma_{x_2}^{-1} = \frac{1}{3}$. Soit :

$$\hat{x}_1 = 1 + \frac{1}{3} \cdot (x_2 - 2)$$

3) On a $\hat{x}_2 = \bar{x}_2 + K \cdot (x_1 - \bar{x}_1)$ avec $K = \Gamma_{x_1 x_2} \Gamma_{x_1}^{-1} = \frac{1}{3}$. Soit :

$$\hat{x}_2 = 2 + \frac{1}{3} \cdot (x_1 - 1)$$

Pour bien comprendre la non-inversibilité des estimateurs, on les représente graphiquement comme sur la figure 7.22. On fait alors l'estimateur dans un sens puis dans l'autre. On ne revient pas au même point. Le programme associé à cet exercice est rangé dans `corrcoev.m`.

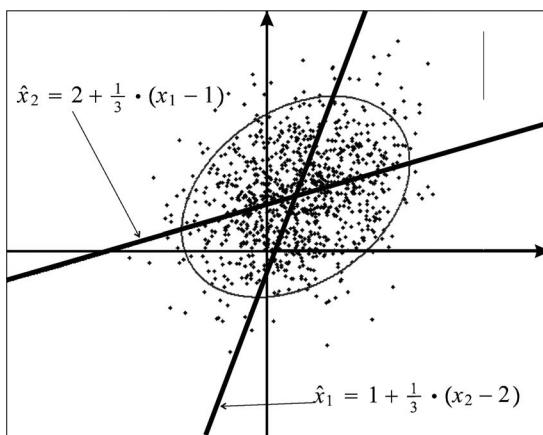


Figure 7.22. Les deux droites en gras représentent les deux estimateurs $\hat{x}_1 (x_2)$ et $\hat{x}_2 (x_1)$

Correction de l'exercice 7.5 (propagation de matrices de covariance)

1) On a :

$$\bar{\mathbf{x}} = E(\mathbf{x}) = E(\mathbf{A} \mathbf{a} - \mathbf{b}) = \mathbf{A} E(\mathbf{a}) - E(\mathbf{b}) = \mathbf{0}$$

car les variables aléatoires \mathbf{a} et \mathbf{b} sont centrées. De même :

$$\bar{y} = E(\mathbf{y}) = E(\mathbf{C} \mathbf{x} + \mathbf{c}) = \mathbf{C} E(\mathbf{x}) + E(\mathbf{c}) = \mathbf{0}$$

car \mathbf{c} est centrée. Pour les matrices de covariance, on a :

$$\begin{aligned}\Gamma_{\mathbf{x}} &= E((\mathbf{x} - \bar{\mathbf{x}})(\mathbf{x} - \bar{\mathbf{x}})^T) = E(\mathbf{x} \mathbf{x}^T) \\ &= E((\mathbf{A} \mathbf{a} - \mathbf{b})(\mathbf{A} \mathbf{a} - \mathbf{b})^T) \\ &= E(\mathbf{A} \mathbf{a} \mathbf{a}^T \mathbf{A}^T - \mathbf{A} \mathbf{a} \mathbf{b}^T - \mathbf{b} \mathbf{a}^T \mathbf{A}^T + \mathbf{b} \mathbf{b}^T) \\ &= \mathbf{A} \underbrace{E(\mathbf{a} \mathbf{a}^T)}_{=\mathbf{I}} \mathbf{A}^T - \mathbf{A} \underbrace{E(\mathbf{a} \mathbf{b}^T)}_{=0} - \underbrace{E(\mathbf{b} \mathbf{a}^T)}_{=0} \mathbf{A}^T + \underbrace{E(\mathbf{b} \mathbf{b}^T)}_{=\mathbf{I}} \\ &= \mathbf{A} \cdot \mathbf{A}^T + \mathbf{I}\end{aligned}$$

De même :

$$\Gamma_{\mathbf{y}} = \mathbf{C} \Gamma_{\mathbf{x}} \mathbf{C}^T + \Gamma_{\mathbf{c}} = \mathbf{C} (\mathbf{A} \mathbf{A}^T + \mathbf{I}) \mathbf{C}^T + \mathbf{I}$$

2) On a bien sûr $\bar{\mathbf{v}} = (\bar{\mathbf{x}}, \bar{y}) = \mathbf{0}$. On a :

$$\Gamma_{\mathbf{v}} = E(\mathbf{v} \mathbf{v}^T) = \begin{pmatrix} E(\mathbf{x} \mathbf{x}^T) & E(\mathbf{x} \mathbf{y}^T) \\ E(\mathbf{y} \mathbf{x}^T) & E(\mathbf{y} \mathbf{y}^T) \end{pmatrix}$$

Or :

$$\Gamma_{\mathbf{xy}} = E(\mathbf{x} \mathbf{y}^T) = E(\mathbf{x} (\mathbf{C} \mathbf{x} + \mathbf{c})^T) = E(\mathbf{x} \mathbf{x}^T \mathbf{C}^T + \mathbf{x} \mathbf{c}^T) = \Gamma_{\mathbf{x}} \mathbf{C}^T + \Gamma_{\mathbf{xc}} = \Gamma_{\mathbf{x}} \mathbf{C}^T$$

car \mathbf{x} et \mathbf{c} sont indépendants (et donc $\Gamma_{\mathbf{xc}} = \mathbf{0}$). Donc :

$$\Gamma_{\mathbf{v}} = \begin{pmatrix} \Gamma_{\mathbf{x}} & \Gamma_{\mathbf{x}} \mathbf{C}^T \\ \mathbf{C} \Gamma_{\mathbf{x}} & \Gamma_{\mathbf{y}} \end{pmatrix}$$

3) On a :

$$\mathbf{z} = (-\mathbf{I} \ \mathbf{I}) \cdot \mathbf{v}$$

Donc :

$$\begin{aligned}\Gamma_{\mathbf{z}} &= (-\mathbf{I} \ \mathbf{I}) \Gamma_{\mathbf{v}} \begin{pmatrix} -\mathbf{I} \\ \mathbf{I} \end{pmatrix} = (-\mathbf{I} \ \mathbf{I}) \begin{pmatrix} \Gamma_{\mathbf{x}} & \Gamma_{\mathbf{x}} \mathbf{C}^T \\ \mathbf{C} \Gamma_{\mathbf{x}} & \Gamma_{\mathbf{y}} \end{pmatrix} \begin{pmatrix} -\mathbf{I} \\ \mathbf{I} \end{pmatrix} \\ &= (-\mathbf{I} \ \mathbf{I}) \begin{pmatrix} -\Gamma_{\mathbf{x}} + \Gamma_{\mathbf{x}} \mathbf{C}^T \\ -\mathbf{C} \Gamma_{\mathbf{x}} + \Gamma_{\mathbf{y}} \end{pmatrix} = \Gamma_{\mathbf{x}} - \Gamma_{\mathbf{x}} \mathbf{C}^T - \mathbf{C} \Gamma_{\mathbf{x}} + \Gamma_{\mathbf{y}}\end{aligned}$$

4) D'après l'expression de l'estimateur linéaire, on a $\hat{\mathbf{x}} = \bar{\mathbf{x}} + \mathbf{K}\tilde{\mathbf{y}}$ avec $\mathbf{K} = \mathbf{\Gamma}_{\mathbf{xy}}\mathbf{\Gamma}_{\mathbf{y}}^{-1} = \mathbf{\Gamma}_{\mathbf{x}}\mathbf{C}^T\mathbf{\Gamma}_{\mathbf{y}}^{-1}$ et $\tilde{\mathbf{y}} = \mathbf{y} - \mathbf{C}\bar{\mathbf{x}}$. Donc :

$$\hat{\mathbf{x}} = \bar{\mathbf{x}} + (\mathbf{\Gamma}_{\mathbf{x}}\mathbf{C}^T\mathbf{\Gamma}_{\mathbf{y}}^{-1})(\mathbf{y} - \mathbf{C}\bar{\mathbf{x}}) = \mathbf{\Gamma}_{\mathbf{x}}\mathbf{C}^T\mathbf{\Gamma}_{\mathbf{y}}^{-1} \cdot \mathbf{y}$$

car $\bar{\mathbf{x}} = \mathbf{0}$.

Correction de l'exercice 7.6 (bruit brownien)

1) On rappelle tout d'abord que si a et b sont deux variables aléatoires centrées indépendantes de variance σ_a^2 et σ_b^2 , et si α est un réel déterministe, alors :

$$\sigma_{a+b}^2 = E((a+b)^2) = E(a^2 + b^2 + 2ab) = \sigma_a^2 + \sigma_b^2 + 2E(ab) = \sigma_a^2 + \sigma_b^2$$

$$\sigma_{\alpha a}^2 = E((\alpha a)^2) = \alpha^2 \sigma_a^2$$

Donc :

$$\sigma_y^2(t_k) = \delta^2 \cdot \sum_{j=0}^k \sigma_x^2(t_j) = \delta^2 \cdot k \sigma_x^2$$

Or $t_k = k\delta$ et donc $\sigma_y^2(t_k) = \delta t_k \sigma_x^2$. On remarque donc que l'écart-type $\sigma_y(t_k) = \sqrt{\delta} \sqrt{t_k} \sigma_x$ tend vers zéro lorsque $\delta \rightarrow 0$ et que cette erreur évolue en \sqrt{t} (effet de la marche aléatoire). Ce phénomène vient du fait que les erreurs se compensent et ceci d'autant plus que δ est petit. Afin d'illustrer ce phénomène sous MATLAB, on programme une fonction de simulation comme suit :

```
function [T,X,Y]=Simu(delta,sig_x,tmax)
T=0 :delta :tmax ;
kmax=length(T) ;
X=sig_x*randn(1,kmax) ; Y=0*X ;
for k=1 :kmax-1, Y(k+1)=Y(k)+delta*X(k) ; end
end
```

Puis on appelle cette fonction pour $\delta = 0.1$, $\delta = 0.01$ et $\delta = 0.001$. Le résultat obtenu est représenté sur la figure 7.23. Chaque figure correspond à $t \in [0, 100]$ et $y \in [-7, 7]$. On remarque que lorsque la période d'échantillonnage δ diminue, le bruit brownien devient plus faible par effet de compensation. Les figures ont été générées par le programme :

```
for i=1 :100,
delta=0.1; % 0.01, 0.001
[T,X,Y]=Simu(delta,1,100) ;
plot(T,Y) ;
end
```

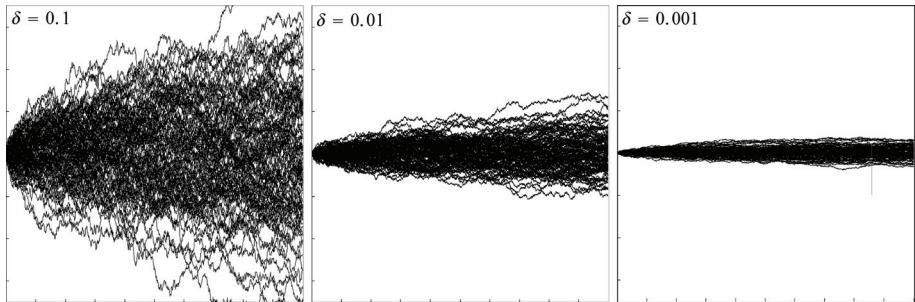


Figure 7.23. Lorsque δ s'approche de 0 diminue, le bruit brownien (intégrale du bruit blanc) devient plus faible

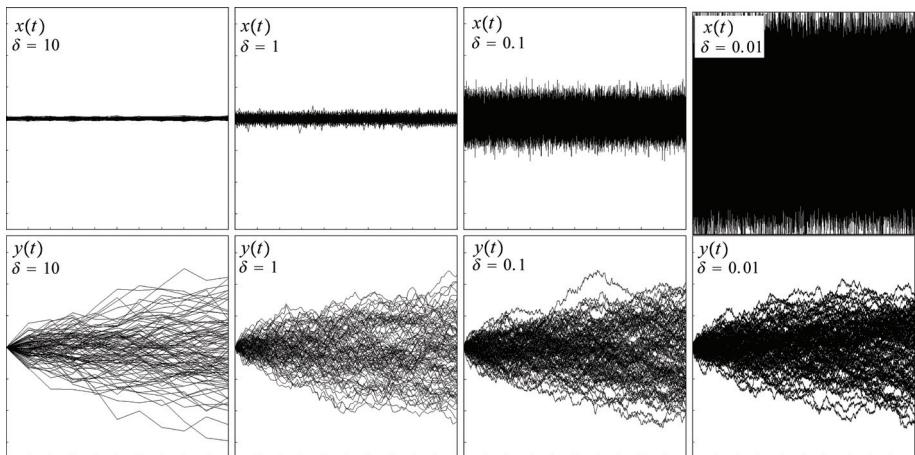


Figure 7.24. Bruit blanc avec le bruit brownien correspondant et pour différentes valeurs de δ

2) On a :

$$\sigma_x(\delta) = \frac{1}{\sqrt{\delta}} \cdot \frac{\sigma_y(t_k)}{\sqrt{t_k}}$$

Pour maintenir $\sigma_y(t_k)$ indépendant de δ , l'écart-type σ_x pour $x(t_k)$ doit donc augmenter à l'infini de façon à lutter contre l'effet de compensation. Lorsque l'on atteint la limite, c'est-à-dire $\delta = 0$, le signal $x(t_k)$ possède un écart-type infini. C'est pour cela qu'un signal aléatoire centré blanc et à temps continu (c'est-à-dire que $\delta = 0$) doit posséder une puissance infinie afin d'avoir une influence sur un intégrateur ou plus généralement sur un quelconque système physique. On observe ce phénomène en physique lors de l'étude du mouvement brownien des particules qui se déplacent dans un espace limité avec une vitesse infinie. Dans un tel cas, $y(t)$ correspond au déplacement de la particule et $x(t)$ à sa vitesse. La figure 7.24 donne pour différents

δ le bruit blanc avec son intégrale. Les échelles sont identiques à celles choisies pour la figure 7.23. On comprend que, lorsque δ tend vers 0, il faut augmenter la puissance du bruit blanc $x(t)$ pour obtenir un bruit brownien semblable. Les figures ont été générées par le programme suivant, qui se trouve rangé dans le fichier `brownien.m`:

```
for i=1 :100,
delta=10; % ou bien 1, 0.1, 0.01
[T,X,Y]=Simu(delta,0.2/sqrt(delta),100);
plot(T,X); plot(T,Y);
end
```

Correction de l'exercice 7.7 (estimateur linéaire pour la résolution de trois équations)

Nous traduisons le problème comme suit :

$$\underbrace{\begin{pmatrix} 8 \\ 7 \\ 0 \end{pmatrix}}_{\mathbf{y}} = \underbrace{\begin{pmatrix} 2 & 3 \\ 3 & 2 \\ 1 & -1 \end{pmatrix}}_{\mathbf{C}} \underbrace{\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}}_{\mathbf{x}} + \underbrace{\begin{pmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \end{pmatrix}}_{\boldsymbol{\beta}}$$

Nous prenons :

$$\bar{\mathbf{x}} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \mathbf{\Gamma}_x = \begin{pmatrix} 1000 & 0 \\ 0 & 1000 \end{pmatrix}, \mathbf{\Gamma}_{\beta} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 4 \end{pmatrix}$$

ce qui revient à dire qu'*a priori*, le vecteur \mathbf{x} est en gros dans l'intervalle $[-33, 33]$, que les β_i sont indépendants entre eux et la première équation est deux fois plus précise. Nous obtenons :

$$\begin{aligned} \tilde{\mathbf{y}} &= \mathbf{y} - \mathbf{C}\bar{\mathbf{x}} = \begin{pmatrix} 8 \\ 7 \\ 0 \end{pmatrix} \\ \mathbf{\Gamma}_y &= \mathbf{C}\mathbf{\Gamma}_x\mathbf{C}^T + \mathbf{\Gamma}_{\beta} = \begin{pmatrix} 13001 & 12000 & -1000 \\ 12000 & 13004 & 1000 \\ -1000 & 1000 & 2004 \end{pmatrix} \\ \mathbf{K} &= \mathbf{\Gamma}_x\mathbf{C}^T\mathbf{\Gamma}_y^{-1} = \begin{pmatrix} -0.09 & 0.288 & 0.311 \\ 0.355 & -0.155 & -0.24 \end{pmatrix} \\ \hat{\mathbf{x}} &= \bar{\mathbf{x}} + \mathbf{K}\tilde{\mathbf{y}} = \begin{pmatrix} 1.311 \\ 1.756 \end{pmatrix} \\ \mathbf{\Gamma}_{\varepsilon} &= \mathbf{\Gamma}_x - \mathbf{K}\mathbf{C}\mathbf{\Gamma}_x = \begin{pmatrix} 0.722 & -0.517 \\ -0.54 & 0.44 \end{pmatrix} \end{aligned}$$

Ainsi, nous pouvons représenter la solution de notre système linéaire par $\hat{\mathbf{x}}$ et la matrice $\mathbf{\Gamma}_{\varepsilon}$.

Correction de l'exercice 7.8 (estimateur linéaire pour l'estimation des paramètres d'un moteur électrique)

Appliquons les formules [7.10] avec :

$$\bar{x} = \begin{pmatrix} 1 \\ -1 \end{pmatrix}, \Gamma_x = \begin{pmatrix} 4 & 0 \\ 0 & 4 \end{pmatrix}; C = \begin{pmatrix} 4 & 0 \\ 10 & 1 \\ 10 & 5 \\ 13 & 5 \\ 15 & 3 \end{pmatrix}, \Gamma_\beta = \begin{pmatrix} 9 & 0 & 0 & 0 & 0 \\ 0 & 9 & 0 & 0 & 0 \\ 0 & 0 & 9 & 0 & 0 \\ 0 & 0 & 0 & 9 & 0 \\ 0 & 0 & 0 & 0 & 9 \end{pmatrix} \text{ et } y = \begin{pmatrix} 5 \\ 10 \\ 8 \\ 14 \\ 17 \end{pmatrix}$$

Le programme MATLAB correspondant est le suivant :

```
y=[5 ;10 ;8 ;14 ;17] ;
C=[4 0 ;10 1;10 5 ; 13 5 ;15 3] ;
xbar=[1 ;-1] ;
Gx=4*eye(2,2) ;
Gbeta=9*eye(5,5) ;
ytilde=y-C*xbar
Gy=C*Gx*C'+Gbeta ;
K=Gx*C'*inv(Gy) ;
xhat=xbar+K*ytilde
Ge=Gx-K*C*Gx
```

Nous obtenons :

$$\tilde{y} = y - C\bar{x} = \begin{pmatrix} 1 \\ 1 \\ 3 \\ 6 \\ 5 \end{pmatrix}$$

$$\Gamma_y = C\Gamma_x C^T + \Gamma_\beta = \begin{pmatrix} 73 & 160 & 160 & 208 & 240 \\ 160 & 413 & 420 & 540 & 612 \\ 160 & 420 & 509 & 620 & 660 \\ 208 & 540 & 620 & 785 & 840 \\ 240 & 612 & 660 & 840 & 945 \end{pmatrix}$$

$$K = \Gamma_x C^T \Gamma_y^{-1} = \begin{pmatrix} 0.027 & 0.0491 & -0.0247 & -0.0044 & 0.046 \\ -0.0739 & -0.118 & 0.148 & 0.092 & -0.077 \end{pmatrix}$$

$$\hat{x} = \bar{x} + K\tilde{y} = \begin{pmatrix} 1.2 \\ -0.58 \end{pmatrix}$$

$$\Gamma_\varepsilon = \Gamma_x - K C \Gamma_x = \begin{pmatrix} 0.062 & -0.166 \\ -0.179 & 0.593 \end{pmatrix}$$

Correction de l'exercice 7.9 (trochoïde)

1) On a :

$$\bar{\mathbf{p}} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \Gamma_{\mathbf{p}} = \begin{pmatrix} 10^2 & 0 \\ 0 & 10^2 \end{pmatrix}$$

$$\mathbf{y} = \begin{pmatrix} 0.38 \\ 3.25 \\ 4.97 \\ -0.26 \end{pmatrix}, \mathbf{C} = \begin{pmatrix} 1 - \cos(1) \\ 1 - \cos(2) \\ 1 - \cos(3) \\ 1 - \cos(7) \end{pmatrix}, \Gamma_{\beta} = \begin{pmatrix} 0.01 & 0 & 0 & 0 \\ 0 & 0.01 & 0 & 0 \\ 0 & 0 & 0.01 & 0 \\ 0 & 0 & 0 & 0.01 \end{pmatrix}$$

et on applique l'estimateur linéaire. Le script MATLAB correspondant est le suivant :

```
y=[0.38 ;3.25 ;4.97 ;-0.26] ;t=[1 ;2 ;3 ;7] ;
C=[ones(size(t)), -cos(t)] ;
pbar=[0 ;0] ;Gp=1000*eye(2,2) ;Gbeta=0.01*eye(4,4) ;
ytilde=y-C*pbar; Gy=C*Gp*C'+Gbeta ;
K=Gp*C'*inv(Gy) ;
phat=pbar+K*ytilde; Ge=Gp-K*C*Gp ;
```

On obtient :

$$\hat{\mathbf{p}} = \begin{pmatrix} 2.001 \\ 2.999 \end{pmatrix} \text{ et } \Gamma_{\varepsilon} = \begin{pmatrix} 0.0025 & -0.0001 \\ -0.0001 & 0.0050 \end{pmatrix}$$

2) Afin de tracer la trajectoire estimée de la masse, nous écrivons :

```
t1=0 :0.01 :20 ; x1=phat(1)*t1-phat(2)*sin(t1) ;
y1=phat(1)-phat(2)*cos(t1); plot(x1,y1) ;
```

On obtient alors la figure 7.25.

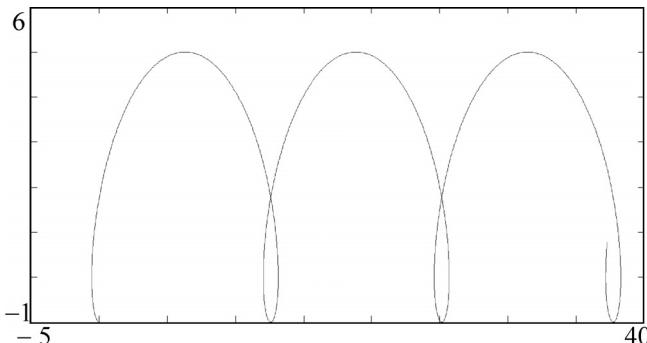


Figure 7.25. Trajectoire estimée pour la masse

Correction de l'exercice 7.10 (filtre de Kalman pour la résolution de trois équations)

1) Le script MATLAB demandé est le suivant :

```
Galpha=zeros(2,2); A=eye(2,2);
C0=[2 3]; C1=[3 2]; C2=[1 -1]; u=0;
xhat0=[0;0]; Gx0=1000*eye(2,2);
[xhat1,Gx1]=kalman(xhat0,Gx0,u,8,Galpha,1,A,C0)
[xhat2,Gx2]=kalman(xhat1,Gx1,u,7,Galpha,4,A,C1)
[xhat3,Gx3]=kalman(xhat2,Gx2,u,0,Galpha,4,A,C2)
```

2) Si l'on trace les trois matrices de covariance associées, on se rend compte qu'à chaque nouvelle équation, la matrice de covariance se contracte jusqu'à se concentrer autour d'un point qui correspond à la solution.

3) Comme l'on pouvait s'y attendre, les résultats obtenus sont identiques à ceux obtenus à l'exercice 7.7.

Correction de l'exercice 7.11 (filtre de Kalman sur trois pas)

La figure 7.26 donne les ellipsoïdes de confiance obtenues par le filtre de Kalman.

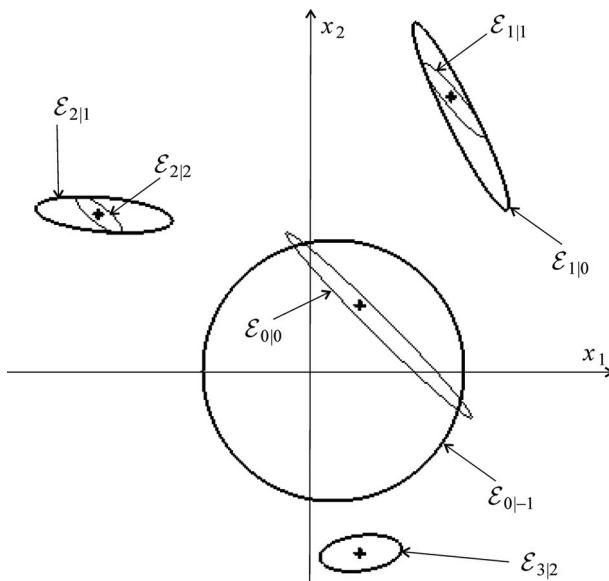


Figure 7.26. Illustration graphique du filtre de Kalman sur un exemple simple

Le programme MATLAB correspondant, rangé dans le fichier `kalm3steps.m`, est le suivant :

```

A0=[0.5 0 ;0 1] ; A1=[1 -1 ;1 1] ; A2=[1 -1 ;1 1] ;
u0=[8 ;16] ; u1=[-6 ;-18] ; u2=[32 ;-8] ;
C0=[1 1] ;C1=[1 1] ;C2=[1 1] ;
y0=7 ; y1=30 ; y2=-6 ;
Galpha=1*eye(2,2) ;
Gbeta=1*eye(1,1) ;
xhat0=[0 ;0] ; Gx0=100*eye(2,2) ;
[xhat1,Gx1]=kalman(xhat0,Gx0,u0,y0,Galpha,Gbeta,A0,C0) ;
[xhat2,Gx2]=kalman(xhat1,Gx1,u1,y1,Galpha,Gbeta,A1,C1) ;
[xhat3,Gx3]=kalman(xhat2,Gx2,u2,y2,Galpha,Gbeta,A2,C2) ;

```

Correction de l'exercice 7.12 (estimation des paramètres d'un moteur électrique)

Afin d'utiliser le filtre de Kalman, nous allons prendre les équations d'état :

$$\begin{cases} \mathbf{x}_{k+1} = \underbrace{\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}}_{\mathbf{A}_k} \mathbf{x}_k + \mathbf{u}_k + \alpha_k \\ y_k = \underbrace{\begin{pmatrix} U(k) & T_r(k) \end{pmatrix}}_{\mathbf{C}_k} \mathbf{x}_k + \beta_k \end{cases}$$

Le programme MATLAB associé est :

```

y=[5 ;10 ;11 ;14 ;17] ;
C=[4 0 ;10 1 ;10 5 ; 13 5 ;15 3]
xhat=[1 ;-1] ;Gx=4*eye(2,2) ;
Galpha=zeros(2,2) ; Gbeta=9 ;
A=eye(2,2) ; u=zeros(2,1) ;
for k=1 :5,
[xhat,Gx]=kalman(xhat,Gx,u,y(k),Galpha,Gbeta,A,C(k, :)) ;
end ;

```

Ce programme est rangé dans `kalmotor.m`. L'estimation obtenue est alors :

$$\bar{\mathbf{x}} = \begin{pmatrix} 1.13 \\ -0.14 \end{pmatrix} \text{ et } \boldsymbol{\Gamma}_{\mathbf{x}} = \begin{pmatrix} 0.06 & -0.17 \\ -0.17 & 0.6 \end{pmatrix}$$

Correction de l'exercice 7.13 (localisation à partir de la mesure des distances aux murs)

1) On a :

$$d(i) = -u_2(i) \cdot x_1 + u_1(i) \cdot x_2 + u_2(i) \cdot a_1(i) - u_1(i) \cdot a_2(i) + \beta_i$$

En posant $y_i = d(i) - \bar{d}(i)$ et $\bar{d}(i) = u_2(i) \cdot a_1(i) - u_1(i) \cdot a_2(i)$, on obtient :

$$y_i = (-u_2(i) u_1(i)) \mathbf{x} + \beta_i$$

Ainsi, on forme les quantités :

$$\mathbf{y} = \begin{pmatrix} d(1) - \bar{d}(i) \\ d(2) - \bar{d}(i) \\ d(3) - \bar{d}(i) \end{pmatrix}, \mathbf{C} = \begin{pmatrix} -u_2(1) u_1(1) \\ -u_2(2) u_1(2) \\ -u_2(3) u_1(3) \end{pmatrix}, \mathbf{\Gamma}_\beta = \mathbf{I}_3, \mathbf{\Gamma}_x = 100 \cdot \mathbf{I}_2, \bar{\mathbf{x}} = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$$

Puis on applique les équations de l'estimateur linéaire orthogonal :

$$\begin{aligned} \hat{\mathbf{x}} &= \bar{\mathbf{x}} + \mathbf{K} \cdot \tilde{\mathbf{y}} & \tilde{\mathbf{y}} &= \mathbf{y} - \mathbf{C} \bar{\mathbf{x}} & \mathbf{\Gamma}_y &= \mathbf{C} \mathbf{\Gamma}_x \mathbf{C}^T + \mathbf{\Gamma}_\beta \\ \mathbf{\Gamma}_\varepsilon &= (\mathbf{I} - \mathbf{K} \mathbf{C}) \mathbf{\Gamma}_x & \mathbf{K} &= \mathbf{\Gamma}_x \mathbf{C}^T \mathbf{\Gamma}_y^{-1} \end{aligned}$$

2) Le programme demandé est :

```
A=[2 15 3 ; 1 5 12] ; B=[15 3 2 ; 5 12 1] ; C=[] ; dbar=[] ;
for i=1 :3,
u=(B(:,i)-A(:,i))/norm(B(:,i)-A(:,i));
C=[C ; [-u(2),u(1)]] ; dbar=[dbar ; det([u,-A(:,i)])] ;
end
d=[2 ; 5 ; 4] ; y=d-dbar ;
x0=[1 ; 2] ; G0=100*eye(2,2) ; u=0 ; Galpha=0*G0 ; Gbeta=eye(3,3) ;
[x1,G1]=kalman(x0,G0,u,y,Galpha,Gbeta,eye(2,2),C) ;
```

L'estimation ainsi calculée est représentée sur la figure 7.27 par son ellipsoïde de confiance.

Correction de l'exercice 7.14 (estimation de la température)

1) Le filtre de Kalman est donné par :

$$\left\{ \begin{array}{l} \hat{x}_{k+1|k} = \hat{x}_{k|k} \\ \Gamma_{k+1|k} = \Gamma_{k|k} + \Gamma_\alpha \\ \hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k \tilde{y}_k \\ \Gamma_{k|k} = (1 - K_k) \Gamma_{k|k-1} \\ \tilde{y}_k = y_k - \hat{x}_{k|k-1} \\ S_k = \Gamma_{k|k-1} + \Gamma_\beta \\ K_k = \Gamma_{k|k-1} S_k^{-1} \end{array} \right.$$

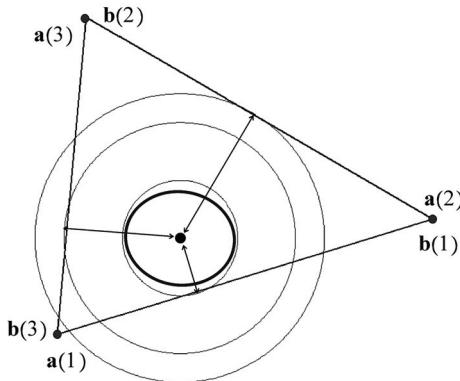


Figure 7.27. Ellipse de confiance (en gras) associée à la localisation du robot

c'est-à-dire :

$$\begin{cases} \hat{x}_{k+1|k} = \hat{x}_{k|k-1} + \frac{\Gamma_{k|k-1}}{\Gamma_{k|k-1} + \Gamma_\beta} (y_k - \hat{x}_{k|k-1}) = \hat{x}_{k|k-1} + \frac{\Gamma_{k|k-1}}{\Gamma_{k|k-1} + 3} (y_k - \hat{x}_{k|k-1}) \\ \Gamma_{k+1|k} = \left(1 - \frac{\Gamma_{k|k-1}}{\Gamma_{k|k-1} + \Gamma_\beta}\right) \Gamma_{k|k-1} + \Gamma_\alpha = \left(1 - \frac{\Gamma_{k|k-1}}{\Gamma_{k|k-1} + 3}\right) \Gamma_{k|k-1} + 4 \end{cases}$$

2) Pour $k \rightarrow \infty$, nous avons $\Gamma_{k+1|k} - \Gamma_{k|k-1} \rightarrow 0$, i.e., $\Gamma_{k+1|k} \rightarrow \Gamma_\infty$. Donc :

$$\Gamma_\infty = \left(1 - \frac{\Gamma_\infty}{\Gamma_\infty + \Gamma_\beta}\right) \Gamma_\infty + \Gamma_\alpha$$

c'est-à-dire :

$$\Gamma_\infty^2 - \Gamma_\alpha \Gamma_\infty - \Gamma_\alpha \Gamma_\beta = 0$$

Il existe une unique solution positive. Elle est donnée par :

$$\Gamma_\infty = \frac{\Gamma_\alpha + \sqrt{\Gamma_\alpha^2 + 4\Gamma_\alpha\Gamma_\beta}}{2} = \frac{4 + \sqrt{16 + 4 \cdot 4 \cdot 3}}{2} = 6$$

et donc le filtre asymptotique s'exprime par :

$$\hat{x}_{k+1} = \hat{x}_k + \frac{2}{3} (y_k - \hat{x}_k)$$

La précision de l'estimation est donnée par la variance $\Gamma_\infty = 6$. On aura la température avec une précision de $\pm\sqrt{6}$ deg.

3) Dans la situation où $\Gamma_{\alpha_k} = 0$, nous obtenons $\Gamma_\infty = 0$. Ce qui signifie qu'après une durée suffisamment longue, le filtre de Kalman non asymptotique renvoie la température exacte, sans aucune incertitude.

Correction de l'exercice 7.15 (marcheur aveugle)

1) On a une évolution linéaire en l'état puisque :

$$\mathbf{x}(k+1) = \begin{pmatrix} 1 & u(k) \\ 0 & 1 \end{pmatrix} \mathbf{x}(k)$$

On initialise le filtre par :

$$\hat{\mathbf{x}} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \text{ et } \Gamma_x = \begin{pmatrix} 0 & 0 \\ 0 & 0.02^2 \end{pmatrix}$$

Le programme qui simule notre système et estime l'état par un filtre de Kalman est le suivant (voir fichier `blindwalk.m`) :

```
xhat=[0 ;1] ; Gx=diag([0,0.02^2]) ;
Galpha=diag([0;0.01^2]) ;
x=[0 ;1+0.02*randn(1)] ;
for k=0 :19,
if (k<10), u=1; else u=-1; end
Ak=[1 u ;0 1] ;
[xhat,Gx]=kalman(xhat,Gx,0,eye(0,1),Galpha,eye(0,0),Ak,eye(0,2)) ;
alpha=mvnrnd([0;0],Galpha)' ;
x=Ak*x+alpha ;
end ;
```

2) Les ellipses sont données sur la figure 7.28 en haut. Comme l'on a aucune mesure extéroceptive, on remarque que ces ellipses grossissent. Pourtant, comme le montre l'écart-type en fonction du temps, la projection de ces ellipses suivant x_1 forment des intervalles dont la taille peut décroître. En effet, les incertitudes du facteur d'échelle accumulées à l'aller sont récupérées en partie au retour. Si je fais dix pas en avant puis dix pas en arrière tous de même longueur, je reviens à mon point de départ, et ceci quelle que soit la longueur de mes pas.

REMARQUE 7.4.– Sur un robot sous-marin du commerce qui navigue à l'estime (c'est-à-dire qui se localise par un prédicteur de Kalman), si les facteurs d'échelle sont mal connus, on observe souvent une incertitude qui diminue sur le trajet du retour. C'est le même phénomène que celui décrit dans cet exercice avec le marcheur.

Correction de l'exercice 7.16 (pendule simple)

1) On a :

$$\ddot{x}_1 = -\sin x_1 + u$$

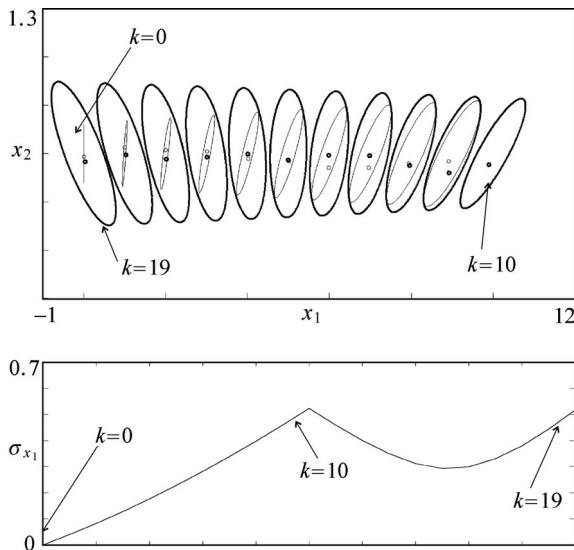


Figure 7.28. Haut : ellipses de confiance pour k allant de 0 à 19 ; bas : écart-type pour x_1

donc en prenant $u = \sin x_1 + v$, on obtient :

$$\ddot{x}_1 = v$$

On prend alors une commande proportionnelle et dérivée :

$$v = (w - x_1) + 2(\dot{w} - \dot{x}) + \ddot{w} = (w - x_1) + 2(\dot{w} - x_2) + \ddot{w}$$

Le régulateur est donc :

$$u = \sin x_1 + (w - x_1) + 2(\dot{w} - x_2) + \ddot{w}$$

2) On prend $w(t) = \sin t$. Ainsi, $\dot{w}(t) = \cos t$ et $\ddot{w} = -\sin t$. Par conséquence :

$$u = \sin x_1 + (\sin t - x_1) + 2(\cos t - x_2) - \sin t$$

3) En linéarisant le pendule, nous obtenons :

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ y \end{pmatrix} = \begin{pmatrix} x_2 \\ -\sin x_1 + u \\ x_2 \end{pmatrix} \simeq \begin{pmatrix} x_2 \\ -x_1 + u \\ x_2 \end{pmatrix}$$

Et donc, on peut dire au filtre de Kalman que le système qu'il observe est décrit par les équations :

$$\begin{aligned}\mathbf{x}_{k+1} &= \underbrace{\begin{pmatrix} 1 & \delta \\ -\delta & 1 \end{pmatrix}}_{\mathbf{A}_k} \mathbf{x}_k + \underbrace{\begin{pmatrix} 0 \\ \delta \end{pmatrix}}_{\mathbf{B}_k} u_k + \alpha_k \\ y_k &= \underbrace{\begin{pmatrix} 0 & 1 \end{pmatrix}}_{\mathbf{C}_k} \mathbf{x}_k + \beta_k\end{aligned}$$

Pour les matrices de covariance de l'état, nous devons prendre quelque chose de l'ordre de δ^2 et qui dépend de la précision de la prédition (perturbation extérieure comme le vent, approximation de linéarisation, frottement négligé, etc.). Nous pouvons prendre par exemple :

$$\Gamma_{\alpha_k} = \begin{pmatrix} \delta^2 & 0 \\ 0 & \delta^2 \end{pmatrix} \text{ et } \Gamma_{\beta_k} = 0.1^2$$

mais d'autres valeurs sont possibles. Le réglage des matrices de covariance d'un filtre de Kalman est un problème délicat qui se fait souvent après quelques expérimentations.

Correction de l'exercice 7.17 (estimation d'état pour le pendule inversé)

1) Linéarisons ce système autour de $\mathbf{x} = \mathbf{0}$ par la méthode par développement limité. Nous avons :

$$\left\{ \begin{aligned}\frac{m \sin x_2 (g \cos x_2 - \ell x_4^2) + u}{M + m \sin^2 x_2} &= \frac{m (x_2 + \varepsilon) (g (1 + \varepsilon) - \ell \varepsilon) + u}{M + m (x_2 + \varepsilon)^2} \\ &= \frac{m (x_2 + \varepsilon) (g + \varepsilon) + u}{M + \varepsilon} = \frac{mgx_2 + u}{M} + \varepsilon \\ \frac{\sin x_2 ((M+m)g - m\ell x_4^2 \cos x_2) + \cos x_2 u}{\ell (M + m \sin^2 x_2)} \\ &= \frac{(x_2 + \varepsilon) ((M+m)g - m\ell \varepsilon (1 + \varepsilon)) + (1 + \varepsilon) u}{\ell (M + m (x_2 + \varepsilon)^2)} = \frac{x_2 (M+m)g + u}{\ell M} + \varepsilon\end{aligned}\right.$$

Ainsi, nous obtenons le système linéarisé :

$$\left\{ \begin{aligned}\dot{\mathbf{x}} &= \underbrace{\begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & \frac{mg}{M} & 0 & 0 \\ 0 & \frac{(M+m)g}{M\ell} & 0 & 0 \end{pmatrix}}_{\mathbf{A}} \mathbf{x} + \underbrace{\begin{pmatrix} 0 \\ 0 \\ \frac{1}{M} \\ \frac{1}{M\ell} \end{pmatrix}}_{\mathbf{B}} u \\ y &= \underbrace{\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}}_{\mathbf{C}} \mathbf{x}\end{aligned}\right.$$

2) Le gain \mathbf{K} est obtenu par la résolution du système :

$$\det(\mathbf{A} - \mathbf{B}\mathbf{K}) = (s + 2)^4$$

Le résultat peut se calculer par l'instruction MATLAB :

```
K = place(A,B,[-2 -2.01 -2.02 -2.03]);
```

où nous avons pris soin d'éviter les pôles multiples, afin que l'instruction `place` fonctionne. Le précompensateur s'obtient par la relation :

$$h = -(\mathbf{E} \cdot (\mathbf{A} - \mathbf{B} \cdot \mathbf{K})^{-1} \cdot \mathbf{B})^{-1}$$

où :

$$\mathbf{E} = \begin{pmatrix} 1 & 0 & 0 & 0 \end{pmatrix}$$

Le code MATLAB correspondant est donc :

```
m=1;M=5;l=1;g=9.81;
A=[0 0 1 0;0 0 0 1;0 m*g/M 0 0;0 (M+m)*g/(l*M) 0 0];
B=[0 0;1/M;1/(l*M)]; C=[1 0 0 0;0 1 0 0];
E=[1 0 0 0];
K=place(A,B,[-2 -2.01 -2.02 -2.03]);
H=-inv(E*inv(A-B*K)*B);
```

Le code MATLAB pour la simulation est de la forme :

```
x=[0 ;0.02 ;0 ;0] ;
for t=0 :dt :30,
w=1; u=-K*x+H*w;
x=x+f(x,u)*dt ;
end
```

3) Une discréttisation par Euler nous donne :

$$\mathbf{x}(k+1) = (\mathbf{I} + dt \mathbf{A}) \cdot \mathbf{x}(k) + dt \mathbf{B} u(k) + \alpha(k)$$

où le vecteur $\alpha(k) \in \mathbb{R}^4$ est le bruit d'état qui prend en compte les erreurs de modélisation, l'erreur de discréttisation et l'erreur de linéarisation. L'équation d'observation est :

$$\mathbf{y}(k) = \underbrace{\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}}_{\mathbf{C}} \mathbf{x}(k) + \begin{pmatrix} \beta_1(k) \\ \beta_2(k) \end{pmatrix}$$

Le filtre de Kalman s'exprime sous la forme :

```
[xr,P]=kalman(xr,P,dt*B*u,y,Q_alpha,Q_beta,eye(4,4)+A*dt,C)
```

4) Le programme, aussi rangé dans le fichier `invpend.m`, est le suivant :

```

sigm2_y=dt*0.01;
Q_alpha=dt*0.0001*eye(4,4); % bruit d'état
Q_beta=(sigm2_y)*eye(2,2); % bruit de mesure
x=[0 ; 0.02 ; 0 ; 0]; % état initial pour le système
xr=[0 ; 0 ; 0 ; 0]; P=eye(4,4);% état initial pour le Kalman
for t=0 :dt :10,
w=1; % consigne
y=C*x+sqrt(sigm2_y)*randn(2,1);
u=-K*xr+H*w;
[xr,P]=kalman(xr,P,dt*B*u,y,Q_alpha,Q_beta,eye(4,4)+A*dt,C);
x=x+f(x,u)*dt;
end

```

Correction de l'exercice 7.18 (localisation à l'estime)

1) La difficulté réside dans le choix de la matrice de covariance et dans le tirage du bruit aléatoire. La simulation se fait par une simple méthode d'Euler :

```

x=[0 ; 0 ;pi/3 ;4 ;0.3];
for t=0 :dt :1,
Galpha=dt*diag([0 0 0.01 0.01 0.01]);
alpha=mvnrnd([0 0 0 0],Galpha)';
x=x+f(x,[0 ; 0])*dt+alpha;
end

```

La fonction d'évolution $f(x, u)$ est la suivante :

```

function xdot=f(x,u)
xdot=[x(4)*cos(x(5))*cos(x(3));
x(4)*cos(x(5))*sin(x(3));
x(4)*sin(x(5))/3; u(1);u(2)];
end;

```

2) Afin d'utiliser le filtre de Kalman pour prédire l'état du système, il nous faut décrire les dépendances entre les variables d'état par une équation d'état linéaire. Si nous posons $z = (x, y, v)$, nous obtenons :

$$\dot{z} = \begin{pmatrix} 0 & 0 & \cos \delta \cos \theta \\ 0 & 0 & \cos \delta \sin \theta \\ 0 & 0 & 0 \end{pmatrix} z + \begin{pmatrix} 0 \\ 0 \\ u_1 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ \alpha_2 \end{pmatrix}$$

Soit, après discréétisation par la méthode d'Euler :

$$\mathbf{z}_{k+1} = \underbrace{\begin{pmatrix} 1 & dt \cos \delta \cos \theta \\ 0 & 1 & dt \cos \delta \sin \theta \\ 0 & 0 & 1 \end{pmatrix}}_{=\mathbf{A}_k} \mathbf{z}_k + \underbrace{\begin{pmatrix} 0 \\ 0 \\ dt \cdot u_1(k) \end{pmatrix}}_{=\mathbf{u}_k} + \underbrace{\begin{pmatrix} 0 \\ 0 \\ dt \cdot \alpha_2 \end{pmatrix}}_{=\alpha_k}$$

3) Le programme, rangé dans le fichier `deadreckoning.m`, est :

```

x=[0 ;0 ;pi/3 ;4 ;0.3] ; Galphax=dt*diag([0 0 0.01 0.01 0.01]) ;
zhat=[x(1) ;x(2) ;x(4)] ; Gz=zeros(3,3) ; Galphaz=dt*diag([0.01
0.01 0.01]) ;
for t=0 :dt :10,
alphax=mvnrnd([0 0 0 0],Galphax)' ;
ux=[0 ;0] ; x=x+f(x,ux)*dt+alphax ;
uz=[0 ;0 ;dt*ux(1)] ;
y=[] ; C=[] ; Gbeta=[] ; % cas sans odomètre
Ak=[1 0 dt*cos(x(5))*cos(x(3)) ; 0 1 dt*cos(x(5))*sin(x(3)) ; 0 0
1] ;
[zhat,Gz]=kalman(zhat,Gz,uz,y,Galphaz,Gbeta,Ak,C) ;
end.

```

4) Si nous avions eu des odomètres capables de nous donner une estimation de la vitesse avec une variance de 0.01, nous aurions dû saisir concernant l'observation :

```
y=x(4)+mvnrnd(0,0.1) ; C=[0 0 1] ; Gbeta=0.1 ;
```

Correction de 7.19 (localisation goniométrique)

1) Nous avons :

$$\begin{pmatrix} \dot{z}_1 \\ \dot{z}_2 \\ \dot{z}_3 \end{pmatrix} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_4 \end{pmatrix} = \begin{pmatrix} x_4 \cos x_5 \cos x_3 \\ x_4 \cos x_5 \sin x_3 \\ u_1 \end{pmatrix} = \begin{pmatrix} 0 & 0 & \cos x_5 \cos x_3 \\ 0 & 0 & \cos x_5 \sin x_3 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} z_1 \\ z_2 \\ z_3 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ u_1 \end{pmatrix}$$

Lorsque le robot voit l'amer $\mathbf{m}(i) = (x_m(i), y_m(i))$ avec un angle δ_i , nous avons :

$$(x_m(i) - x_1) \sin(x_3 + \delta_i) - (y_m(i) - x_2) \cos(x_3 + \delta_i) = 0$$

c'est-à-dire :

$$\underbrace{-x_m(i) \sin(x_3 + \delta_i) + y_m(i) \cos(x_3 + \delta_i)}_{\text{connu}} = \underbrace{(-\sin(x_3 + \delta_i) \cos(x_3 + \delta_i))}_{\text{connu}} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \beta_i$$

où β_i est un bruit que nous pourrons supposer blanc gaussien et de variance 1. Ce bruit permet de prendre en compte les incertitudes sur les angles mesurés (principalement les δ_i). Si $\{i_1, i_2, \dots\}$ sont les numéros des amers vus par le robot, nous avons l'équation d'observation :

$$\mathbf{y}(k) = \underbrace{\begin{pmatrix} 0 & 0 & 1 \\ -\sin(x_3 + \delta_{i_1}) \cos(x_3 + \delta_{i_1}) & 0 & 0 \\ -\sin(x_3 + \delta_{i_2}) \cos(x_3 + \delta_{i_2}) & 0 & 0 \\ \vdots & \vdots & \vdots \end{pmatrix}}_{\mathbf{C}(k)} \cdot \mathbf{z}(k)$$

Notons que la dimension de \mathbf{y} dépend de k . La première équation est donnée par les odomètres qui nous donnent la vitesse. Les autres équations correspondent à la mesure goniométrique des amers.

2) Une discréétisation par Euler, nous donne :

$$\mathbf{z}(k+1) = \underbrace{\begin{pmatrix} 1 & 0 & dt \cdot \cos x_5 \cdot \cos x_3 \\ 0 & 1 & dt \cdot \cos x_5 \cdot \sin x_3 \\ 0 & 0 & 1 \end{pmatrix}}_{\mathbf{A}(k)} \cdot \mathbf{z}(k) + \begin{pmatrix} 0 \\ 0 \\ dt \cdot u_1 \end{pmatrix} + \alpha(k)$$

qui est linéaire.

3) Pour réaliser le simulateur, il nous faut reprendre le simulateur de l'exercice 7.19, pour lequel il nous faut rajouter une fonction d'observation. Afin, d'alimenter le filtre de Kalman, cette fonction doit retourner un vecteur de mesures \mathbf{y} , la matrice de covariance associée, mais aussi la matrice d'observation $\mathbf{C}(k)$. Cette fonction d'observation est :

```
function [y,Gbeta,Ck]=g(x)
Ck=[0,0,1] ; y=x(4) ; beta=1 ; %odomètres
for i=1 :length(landmarks),
a=landmarks( :,i) ;
da=a-x(1 :2) ;
delta=angle(da)-x(3) ;
if (norm(da)<15),
yi=-a(1)*sin(x(3)+delta)+a(2)*cos(x(3)+delta) ;
Cki=[-sin(x(3)+delta),cos(x(3)+delta),0] ;
y=[y ;yi] ; Ck=[Ck ;Cki] ; beta=[beta ;1] ;
end ;
end ;
```

```
Gbeta=diag(beta);
y=y+mvnrnd(zeros(size(y)),Gbeta)';
end
```

Les matrice des amers a été initialisée dans le programme principal comme suit :

```
landmarks=[0 15 30 15 ; 25 30 15 20];
```

4) Le programme principal incluant le simulateur et le filtre de Kalman est le suivant :

```
dt=0.05; u=[0;0];
x=[0;-20;pi/3;20;0.1]; % état initial
zhat=[0;0;0]; Gz=10^3*eye(3,3); % l'état initial est inconnu
Galpha=dt*0.001*eye(3,3); % bruit d'état
for t=0 :dt :10,
[y,Gbeta,Ck]=g(x);
Ak=eye(3,3)+dt*cos(x(5))*[0 0 cos(x(3)); 0 0 sin(x(3)); 0 0 0];
uk=dt*[0;0;u(1)];
[zhat,Gz]=kalman(zhat,Gz,uk,y,Galpha,Gbeta,Ak,Ck);
alphax=0*x; alphax([1;2;4])=mvnrnd(zeros(3,1),Galpha)';
x=x+f(x,u)*dt+alphax;
end;
```

5) Nous pouvons voir les deux robots comme un seul système dont le vecteur d'état est :

$$\mathbf{x} = (x_{a1}, x_{a2}, x_{a3}, x_{a4}, x_{a5}, x_{b1}, x_{b2}, x_{b3}, x_{b4}, x_{b5})$$

Le vecteur :

$$\mathbf{z} = (x_{a1}, x_{a2}, x_{a4}, x_{b1}, x_{b2}, x_{b4})$$

satisfait une équation d'évolution linéaire donnée par :

$$\underbrace{\begin{pmatrix} \dot{x}_{a1} \\ \dot{x}_{a2} \\ \dot{x}_{a4} \\ \dot{x}_{b1} \\ \dot{x}_{b1} \\ \dot{x}_{b1} \end{pmatrix}}_{=\dot{\mathbf{z}}} = \begin{pmatrix} 0 & 0 & \cos x_{a5} & \cos x_{a3} & 0 & 0 & 0 \\ 0 & 0 & \cos x_{a5} & \sin x_{a3} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \cos x_{b5} & \cos x_{b3} & 0 \\ 0 & 0 & 0 & 0 & \cos x_{b5} & \sin x_{b3} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \underbrace{\begin{pmatrix} x_{a1} \\ x_{a2} \\ x_{a4} \\ x_{b1} \\ x_{b1} \\ x_{b1} \end{pmatrix}}_{=\mathbf{z}} + \begin{pmatrix} 0 \\ 0 \\ 0 \\ u_{a1} \\ 0 \\ 0 \\ u_{b1} \end{pmatrix}$$

Lorsque les deux robots sont capables de se voir, nous avons la relation :

$$(x_{b1} - x_{a1}) \sin(x_{a3} + \varphi_a) - (x_{b2} - x_{a2}) \cos(x_{a3} + \varphi_a) = 0$$

c'est-à-dire :

$$0 = (-\sin(x_{a3} + \varphi_a) \cos(x_{a3} + \varphi_a) \ 0 \ \sin(x_{a3} + \varphi_a) \ -\cos(x_{a3} + \varphi_a) \ 0) \cdot \mathbf{z} + \beta(k)$$

où $\beta(k)$ correspond à un bruit de mesure que nous pourrons supposer gaussien et blanc. Le 0 à gauche de l'égalité correspond à la mesure. La fonction d'observation associée peut se coder de la façon suivante :

```
function [yab,Gab,Cab]=gab(xa,xb)
da=xb(1 :2)-xa(1 :2) ; phi=atan(da)-xa(3) ;
yab=[] ; Gab=[] ; Cab=[];
if (norm(da)<20),
Cab=[-sin(xa(3)+phi),cos(xa(3)+phi),0,sin(xa(3)+phi),-cos(xa(3)+phi),0] ;
Gab=1 ; yab=mvnrnd(0,Gab) ;
end ; end
```

A cette fonction d'observation interrobot, il nous faut rajouter la détection des amers par les robots déjà vue à la question précédente. On peut alors fusionner la fonction d'observation interrobots et celle des amers en une seule fonction comme suit :

```
function [y,Gbeta,Ck]=gall(xa,xb)
[ya,Ga,Cak]=g(xa) ; [yb,Gb,Cbk]=g(xb) ;
[yab,Gab,Cabk]=gab(xa,xb) ;
y=[ya ;yb ;yab] ;
Gbeta=blkdiag(Ga,Gb,Gab) ;
Ck=[blkdiag(Cak,Cbk) ; Cabk] ;
end
```

On peut alors utiliser un filtre de Kalman pour faire la localisation. Le script, rangé dans le fichier `gonio.m`, est donné par :

```
ua=[0 ;0] ; ub=[0 ;0] ; % entrée pour les robots A et B
xa=[-13 ;-22 ;pi/3 ;15 ;0.1] ; % état initial pour le robot A
xb=[20 ;-10 ;pi/3 ;18 ;0.2] ; % état initial pour le robot B
zhat=zeros(6,1) ; Gz=10^3*eye(6,6) ; % initialisation du filtre
Galphaa=dt*diag([0.1,0.1,0.5]) ; Galphab=Galphaa ;
Galpha=blkdiag(Galphaa,Galphab) ; % covariance pour le bruit
d'état
for t=0 :dt :10,
[y,Gbeta,Ck]=gall(xa,xb) ; % observation
Aak=[1 0 dt*cos(xa(5))*cos(xa(3)) ; 0 1 dt*cos(xa(5))*sin(xa(3)) ;
0 0 1] ;
Abk=[1 0 dt*cos(xb(5))*cos(xb(3)) ; 0 1 dt*cos(xb(5))*sin(xb(3)) ;
0 0 1] ;
Ak=blkdiag(Aak,Abk) ;
```

```

uk=dt*[0;0;ua(1);0;0;ub(1)];
[zhat,Gz]=kalman(zhat,Gz,uk,y,Galpha,Gbeta,Ak,Ck);
alphaa=0*xa; alphaa([1;2;4])=mvnrnd(zeros(3,1),Galphaaa)';
alphab=0*xb; alphab([1;2;4])=mvnrnd(zeros(3,1),Galphab)';
xa=xa+f(xa,ua)*dt+alphaa; xb=xb+f(xb,ub)*dt+alphab;
end;

```

Correction de l'exercice 7.20 (suivi d'un bateau par deux radars)

1) On retrouve le simulateur dans le programme qui sera donné à la correction de la question 4.

2) Puisque :

$$\mathbf{g}(\mathbf{x}) = \begin{pmatrix} (p_x - a_x)^2 + (p_y - a_y)^2 \\ (p_x - b_x)^2 + (p_y - b_y)^2 \end{pmatrix}$$

nous avons :

$$\frac{d\mathbf{g}}{d\mathbf{x}}(\hat{\mathbf{x}}) = \begin{pmatrix} 2(\hat{p}_x - a_x) & 0 & 2(\hat{p}_y - a_y) & 0 \\ 2(\hat{p}_x - b_x) & 0 & 2(\hat{p}_y - b_y) & 0 \end{pmatrix}$$

Et donc l'équation d'observation peut s'approximer par son équation tangente :

$$\mathbf{y} \simeq \mathbf{g}(\hat{\mathbf{x}}) + \frac{d\mathbf{g}}{d\mathbf{x}}(\hat{\mathbf{x}}) \cdot (\mathbf{x} - \hat{\mathbf{x}})$$

c'est-à-dire :

$$\underbrace{\mathbf{y} - \mathbf{g}(\hat{\mathbf{x}})}_{\mathbf{z}} + \underbrace{\frac{d\mathbf{g}}{d\mathbf{x}}(\hat{\mathbf{x}}) \cdot \hat{\mathbf{x}}}_{\mathbf{C}} \simeq \underbrace{\frac{d\mathbf{g}}{d\mathbf{x}}(\hat{\mathbf{x}})}_{\mathbf{C}} \cdot \mathbf{x}$$

3) Pour implémenter le filtre de Kalman, nous prenons :

$$\mathbf{A}_k = \begin{pmatrix} 1 & dt & 0 & 0 \\ 0 & 1 - dt & 0 & 0 \\ 0 & 0 & 1 & dt \\ 0 & 0 & 0 & 1 - dt \end{pmatrix} \text{ et } \mathbf{C}_k = \begin{pmatrix} 2(\hat{p}_x - a_x) & 0 & 2(\hat{p}_y - a_y) & 0 \\ 2(\hat{p}_x - b_x) & 0 & 2(\hat{p}_y - b_y) & 0 \end{pmatrix}$$

4) Le programme principal MATLAB correspondant, rangé dans le fichier `radar.m`, est le suivant :

```

dt=0.01;a=[0;0]; b=[1;0]; x=[0;0;2;0];
Ak=[1 dt 0 0; 0 (1-dt) 0 0; 0 0 1 dt; 0 0 0 (1-dt)];
Galpha=dt*diag([0;1;0;1]); Gbeta=eye(2,2);
xhat=[1;0;3;0]; Gx=10000*eye(4,4);

```

```

for t=0 :dt :10,
beta=mvnrnd([0 ;0],Gbeta)' ; y=g(x)+beta ;
Ck=2*[xhat(1)-a(1),0,xhat(3)-a(2),0 ;xhat(1)-b(1),0,xhat(3)-b(2),0] ;
zk=y-g(xhat)+Ck*xhat ;
[xhat,Gx]=kalman(xhat,Gx,0,zk,Galpha,Gbeta,Ak,Ck) ;
alpha=mvnrnd([0 ;0 ;0 ;0],Galpha)' ;
x=Ak*x+alpha ;
end

```

La fonction $g(x)$ est :

```

function y=g(x)
y=[norm(x([1,3])-a)^2;norm(x([1,3])-b)^2] ;
end

```

Correction de l'exercice 7.21 (localisation d'un robot dans une piscine)

1) Lorsque le sonar pointe sur un mur, la distance ℓ renvoyée satisfait :

$$a = \ell \cdot \cos \beta \quad [7.15]$$

où β est l'angle entre la normale au mur et le faisceau sonar et a est la distance entre le sonar et le mur. Nous allons supposer que le robot est immobile et que seul le sonar tourne (cela revient à supposer que les vitesses tangentielle v et angulaire $\dot{\theta}$ du robot sont négligeables devant la vitesse de rotation $\dot{\alpha}$ du sonar). Considérons tout d'abord, la situation où à l'instant t , le sonar est normal au mur. Si τ est un réel positif suffisamment petit, c'est-à-dire tel que à l'instant $t - \tau$ le sonar pointe toujours sur le même mur, nous devrions avoir, d'après la relation [7.15] :

$$a = \ell(t - \tau) \cdot \cos(-\dot{\alpha}\tau)$$

Rappelons que la vitesse de rotation du sonar $\dot{\alpha}$ est supposée connue et constante. Prenons $\tau = k\delta$, $k \in \{0, 1, 2, \dots, N-1\}$, où δ est la durée séparant deux pings du sonar et N est un entier tel que, à l'instant $t - N\delta$, le sonar pointe nécessairement sur le mur qui se trouve orthogonal au faisceau sonar à l'instant t . Posons :

$$\tilde{a}_k = \ell(t - k\delta) \cdot \cos(-k\delta\dot{\alpha})$$

La quantité \tilde{a}_k devrait correspondre à la distance a entre le robot et le mur pointé. Mais du fait de la présence d'un bruit de mesure, il est préférable d'obtenir une estimation \hat{a} de la distance a par une moyenne :

$$\hat{a} = \frac{1}{N} \sum_{k=0}^{N-1} \tilde{a}_k$$

Nous pouvons vérifier le fait que le sonar pointe bien perpendiculairement à un mur en vérifiant que la variance des \tilde{a}_k est faible, c'est-à-dire :

$$\frac{1}{N} \sum_{k=1}^{N-1} (\tilde{a}_k - \hat{a})^2 < \varepsilon$$

où ε est un seuil fixé proche de 0. Il s'agit du *test de la variance*. Or, en pratique il y a beaucoup de données aberrantes. Il nous faut donc modifier notre méthode. Une approche plus robuste que celle présentée précédemment, consiste à calculer la médiane plutôt que la moyenne. Pour cela, il faut trier les \tilde{a}_k par ordre croissant. Ensuite, on prend le milieu \bar{a} de la liste et on supprime de la liste les éléments de cette liste qui sont les plus éloignés de \bar{a} . On en supprime par exemple une moitié. Ils sont faciles à trouver car ils se trouvent soit en début, soit en fin de liste. On fait alors la moyenne des éléments restant pour obtenir \hat{a} . On fait alors le test de la variance sur les éléments restants pour vérifier que le sonar est normal à un mur. Notons que si le robot dispose d'une boussole fiable, le test de la variance devient inutile. En effet, la boussole nous donne θ et l'angle α est connu, ce qui nous permet de savoir si le sonar pointe ou non suivant la normale à un mur et aussi de quel mur il s'agit.

2) Pour les deux dernières équations, il suffit de remarquer que l'accélération absolue est obtenue à partir de l'accélération mesurée (a_T, a_N) par les accéléromètres par une simple rotation d'angle θ :

$$\begin{pmatrix} \ddot{x} \\ \ddot{y} \end{pmatrix} = \begin{pmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{pmatrix} \begin{pmatrix} a_T \\ a_N \end{pmatrix}$$

3) Afin de pouvoir utiliser un filtre de Kalman, il nous faut tout d'abord discréteriser le temps. Remarquons que, lorsque $\alpha + \theta$ est un multiple de $\frac{\pi}{2}$, le faisceau sonar est orienté face à un des quatre murs de la piscine (car cette dernière est supposée rectangulaire). Dans un tel cas, la longueur mesurée peut nous permettre de calculer soit x , soit y . Pour notre problème, le temps k discret s'incrémentera chaque fois que le faisceau sonar pointe face à un mur de la piscine, c'est-à-dire que $k = E(\frac{\alpha+\theta}{\pi/2})$, où E désigne la partie entière d'un nombre réel. Une discréttisation par Euler de nos équations d'état se traduit par :

$$\begin{cases} x(k+1) = x(k) + v_x(k) \cdot T(k) \\ y(k+1) = y(k) + v_y(k) \cdot T(k) \\ v_x(k+1) = v_x(k) + (a_T(k) \cos \theta(k) - a_N(k) \sin \theta(k)) \cdot T(k) \\ v_y(k+1) = v_y(k) + (a_T(k) \sin \theta(k) + a_N(k) \cos \theta(k)) \cdot T(k) \end{cases}$$

où $T(k)$ est le temps qui s'est écoulé entre deux incrémentations successives de k . Sous forme matricielle, ces équations d'état deviennent :

$$\begin{aligned}
 \mathbf{x}(k+1) &= \begin{pmatrix} 1 & 0 & T(k) & 0 \\ 0 & 1 & 0 & T(k) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \mathbf{x}(k) + \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ -T(k) \sin \theta(k) & T(k) \cos \theta(k) \\ T(k) \cos \theta(k) & T(k) \sin \theta(k) \end{pmatrix} \mathbf{u}(k) \\
 r(k) &= C(k) \cdot \mathbf{x}(k)
 \end{aligned}
 \tag{7.16}$$

avec :

$$\mathbf{x}(k) = (x(k), y(k), v_x(k), v_y(k))$$

$$\mathbf{u}(k) = (a_N(k), a_T(k))$$

En ce qui concerne l'équation de mesure, il nous faut distinguer quatre cas :

– cas 0, mur droit ($\theta(k) + \alpha(k) = 2k\pi$). Dans ce cas, nous avons une mesure sur $x : r(k) = x(k) \simeq R_x - d(k)$, où $d(k)$ est la distance retournée par le sonar. Ainsi, la matrice d'observation sera $C(k) = (1 \ 0 \ 0 \ 0)$;

– cas 1, mur fond ($\theta(k) + \alpha(k) = 2k\pi + \frac{\pi}{2}$). Dans ce cas, nous avons une mesure sur $y : r(k) = y(k) \simeq R_y - d(k)$ et donc $C(k) = (0 \ 1 \ 0 \ 0)$;

– cas 2, mur gauche ($\theta(k) + \alpha(k) = 2k\pi + \pi$). Nous avons à nouveau une mesure sur $x : r(k) = x(k) \simeq -R_x + d(k)$. La matrice d'observation sera $C(k) = (1 \ 0 \ 0 \ 0)$;

– cas 3, mur devant ($\theta(k) + \alpha(k) = 2k\pi + \frac{3\pi}{2}$). Nous avons une mesure sur $y : r(k) = y(k) \simeq -R_y + d(k)$. Ainsi $C(k) = (0 \ 1 \ 0 \ 0)$.

Si nous connaissons $\theta + \alpha$, pour savoir dans quel cas i nous sommes, il faut résoudre :

$$\exists k \in \mathbb{N}, \theta(k) + \alpha(k) = 2k\pi + \frac{i\pi}{2}$$

c'est-à-dire :

$$\exists k \in \mathbb{N}, \frac{2}{\pi}(\theta(k) + \alpha(k)) = i + 4k$$

Donc, pour avoir le cas i le plus probable, on calcule l'entier le plus proche de la quantité $\frac{2}{\pi}(\theta(k) + \alpha(k))$, et on regarde le reste de la division euclidienne de cet entier par 4. Sous MATLAB, cela se fait par le calcul suivant :

```
i=mod(round((thetak+alphak)*2/pi),4)
```

Notons que le système [7.16] n'a pas pour but de reproduire le comportement dynamique du système au niveau de la commande, mais plutôt de permettre l'utilisation d'un filtre de Kalman, dans le but d'estimer la position et la vitesse

du robot. Nous nous sommes arrangés pour avoir un système discret décrit par des équations d'état linéaires de la forme :

$$\begin{cases} \mathbf{x}(k+1) = \mathbf{A}(k)\mathbf{x}(k) + \mathbf{B}(k)\mathbf{u}(k) + \alpha(k) \\ r(k) = C(k)\mathbf{x}(k) + \beta(k) \end{cases}$$

Nous venons de rajouter deux signaux de bruit α et β qui seront supposés blancs, de matrice de covariance Γ_α et Γ_β (notons qu'ici, Γ_β est un scalaire). Ces deux matrices ont pour but de modéliser les incertitudes de modèle et les bruits de mesure. Un filtre de Kalman nous permettra alors de nous localiser. La figure 7.29 représente le robot à un instant t , le faisceau sonar associé et une suite d'ellipsoïdes de confiance engendrés par le filtre de Kalman. Le grand cercle représente l'ellipse de confiance initiale.

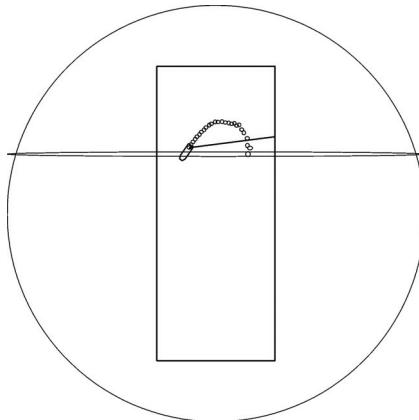


Figure 7.29. Localisation du robot sous-marin à l'aide du filtre de Kalman

Correction de l'exercice 7.22 (SLAM)

1) Le programme de simulation par une méthode d'Euler est le suivant :

```
M=load('slam_data.txt'); t=M( :,1); phi=M( :,2); theta=M( :,3);
psi=M( :,4);
vr=M( :,5 :7); depth=M( :,8); alt=M( :,9);
dt=0.1; kmax=length(M);
xhat=[0 ;0 ;0];
for k=1 :kmax,
    xhat=xhat+dt*eulermat(phi(k),theta(k),psi(k))*vr(k, :)';
end
```

2) Les angles d'Euler sont très bien connus grâce à la centrale inertie. Donc, on a :

$$\begin{aligned}\mathbf{p}_{k+1} &= \mathbf{p}_k + dt \cdot \mathbf{R}(\varphi_k, \theta_k, \psi_k) \cdot (\bar{\mathbf{v}}_r(k) + \alpha_v(k)) \\ &= \mathbf{p}_k + dt \cdot \underbrace{\mathbf{R}(\varphi_k, \theta_k, \psi_k) \cdot \bar{\mathbf{v}}_r(k)}_{\mathbf{u}_k} + \underbrace{dt \cdot \mathbf{R}(\varphi_k, \theta_k, \psi_k) \cdot \alpha_v(k)}_{=\alpha_k}\end{aligned}$$

On peut approximer α_k par un bruit blanc de matrice de covariance :

$$\begin{aligned}\boldsymbol{\Gamma}_\alpha &= dt^2 \cdot \mathbf{R}(\varphi_k, \theta_k, \psi_k) \boldsymbol{\Gamma}_{\alpha_v} \mathbf{R}^T(\varphi_k, \theta_k, \psi_k) \text{ car } \alpha_k = dt \cdot \mathbf{R}(\varphi_k, \theta_k, \psi_k) \cdot \alpha_v(k) \\ &= dt^2 \sigma_v^2 \mathbf{I} \quad \text{car } \mathbf{R} \mathbf{R}^T = \mathbf{I} \\ &= 10^{-2} \cdot \mathbf{I}\end{aligned}$$

Notons que nous n'avons pas pris en compte les erreurs de discréétisation et les erreurs sur les angles de la centrale.

3) Lorsque le filtre de Kalman est utilisé en mode prédicteur, on a :

$$\boldsymbol{\Gamma}_{k+1|k} = \mathbf{A}_k \cdot \boldsymbol{\Gamma}_{k|k} \cdot \mathbf{A}_k^T + \boldsymbol{\Gamma}_{\alpha_k} = \mathbf{A}_k \cdot \boldsymbol{\Gamma}_{k|k-1} \cdot \mathbf{A}_k^T + \boldsymbol{\Gamma}_\alpha$$

Comme aucune mesure n'existe en mode prédicteur, on a :

$$\boldsymbol{\Gamma}_{k|k} = \boldsymbol{\Gamma}_{k|k-1} = \boldsymbol{\Gamma}_k$$

et donc l'équation devient :

$$\boldsymbol{\Gamma}_{k+1} = \mathbf{A}_k \cdot \boldsymbol{\Gamma}_k \cdot \mathbf{A}_k^T + \boldsymbol{\Gamma}_\alpha$$

Ainsi :

$$\boldsymbol{\Gamma}_1 = \mathbf{A}_0 \cdot \boldsymbol{\Gamma}_0 \cdot \mathbf{A}_0^T + \boldsymbol{\Gamma}_\alpha$$

$$\boldsymbol{\Gamma}_2 = \mathbf{A}_1 \cdot \boldsymbol{\Gamma}_1 \cdot \mathbf{A}_1^T + \boldsymbol{\Gamma}_\alpha = \mathbf{A}_1 \mathbf{A}_0 \cdot \boldsymbol{\Gamma}_0 \cdot \mathbf{A}_0^T \mathbf{A}_1^T + \mathbf{A}_1 \boldsymbol{\Gamma}_\alpha \mathbf{A}_1^T + \boldsymbol{\Gamma}_\alpha$$

$$\boldsymbol{\Gamma}_3 = \mathbf{A}_2 \cdot \boldsymbol{\Gamma}_2 \cdot \mathbf{A}_2^T + \boldsymbol{\Gamma}_\alpha = \mathbf{A}_2 \mathbf{A}_1 \mathbf{A}_0 \cdot \boldsymbol{\Gamma}_0 \cdot \mathbf{A}_0^T \mathbf{A}_1^T \mathbf{A}_2^T + \mathbf{A}_2 \mathbf{A}_1 \boldsymbol{\Gamma}_\alpha \mathbf{A}_1^T \mathbf{A}_2^T + \mathbf{A}_2 \boldsymbol{\Gamma}_\alpha \mathbf{A}_2^T + \boldsymbol{\Gamma}_\alpha$$

Or, dans notre contexte, les matrices \mathbf{A}_i sont égales à l'identité. Donc on a :

$$\boldsymbol{\Gamma}_k = k \cdot \boldsymbol{\Gamma}_\alpha = k dt^2 \sigma_v^2 \mathbf{I}$$

On remarque que la matrice de covariance augmente donc de façon linéaire avec le temps k . L'écart-type augmente donc avec \sqrt{k} , ce qui est un phénomène connu de la marche aléatoire. Puisque $t = k \cdot dt$, la covariance de l'état prédit est :

$$\boldsymbol{\Gamma}_x(t) = \frac{t}{dt} \cdot \boldsymbol{\Gamma}_\alpha = \frac{t}{dt} dt^2 \sigma_v^2 \mathbf{I} = t \cdot dt \cdot \sigma_v^2 \cdot \mathbf{I}$$

ce qui correspond à un écart-type (ou une dérive) :

$$\sigma_x(t) = \sigma_v \sqrt{t \cdot dt} = 0.3\sqrt{t}$$

Au bout d'une heure, l'erreur est de $\sigma_x(3600) = 0.3 \cdot \sqrt{3600} = 18 \text{ m}$ et au bout de deux heures, $\sigma_x(2 \cdot 3600) = 0.3 \cdot \sqrt{2 \cdot 3600} = 25 \text{ m}$.

Le code correspondant au prédicteur est donné ci-après :

```
M=dlmread('slam_data.txt');
t=M(:,1); phi=M(:,2); theta=M(:,3); psi=M(:,4);
vr=M(:,5:7);
depth=M(:,8); alt=M(:,9);
dt=0.1; kmax=size(M,1);
xhat=zeros(3,1);
Gx=diag([0,0,0]);
Galpha=[0.01*eye(3,3)];
A=eye(3,3);
for k=1:kmax,
u=[dt*eulermat(phi(k),theta(k),psi(k))*vr(k,:)]';
[xhat,Gx]=kalman(xhat,Gx,u,eye(0,1),Galpha,eye(0,0),A,eye(0,3));
end;
```

On remarquera que comme l'on a pas de mesure, les quantités y, C, Γ_β sont vides. Il faut tout de même respecter les dimensions d'où les appels aux fonctions `eye(0,n)` dans les paramètres de la fonction `kalman`. Le résultat du prédicteur est illustré par la figure 7.30.

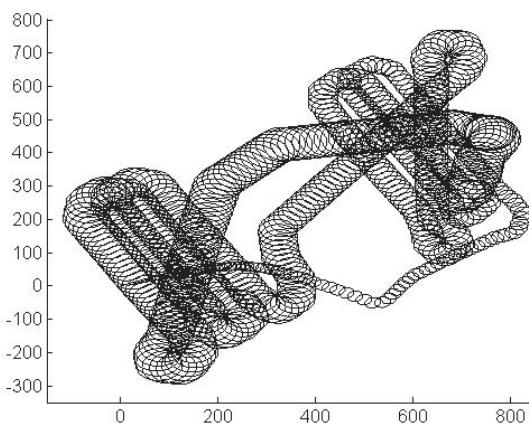


Figure 7.30. Trajectoire de confiance obtenue par le prédicteur

4) L'équation d'observation est donnée par :

$$\underbrace{\begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix}}_{\mathbf{y}_k} = \underbrace{\begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \dots & 0 & 0 \end{pmatrix}}_{\mathbf{C}(k)} \underbrace{\begin{pmatrix} \mathbf{p}_k \\ \mathbf{q}_k \end{pmatrix}}_{\mathbf{x}_k} + \beta_k$$

si aucun amer n'est détecté. Elle sera :

$$\underbrace{\begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix}}_{\mathbf{y}_k} = \underbrace{\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & \dots & 0 & 0 & -1 & 0 & 0 & 0 & \dots & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & \dots & 0 & 0 & 0 & -1 & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & \dots & 0 & 0 & 0 & 0 & 0 & 0 & \dots & 0 & 0 \end{pmatrix}}_{\mathbf{C}(k)} \underbrace{\begin{pmatrix} \mathbf{p}_k \\ \mathbf{q}_k \end{pmatrix}}_{\mathbf{x}_k} + \beta_k$$

dans le cas où le i ème amer \mathbf{m}_i est détecté. Dans ce deuxième cas, le sous-vecteur (y_1, y_2) représente les deux premières composantes du vecteur :

$$\mathbf{p} - \mathbf{m}_i = \mathbf{R}(k) \cdot \begin{pmatrix} 0 \\ -\sqrt{r_i^2(k) - a^2(k)} \\ -a(k) \end{pmatrix}$$

La composante y_3 correspond à la mesure de profondeur donnée par le capteur de pression. La fonction d'évolution est la suivante :

```
function [y,C,Gbeta]=g(k)
y=depth(k) ; C=zeros(1,nx) ; C(1,3)=1 ; Gbeta=0.01 ;
T=[10540,10920,13740,17480,30380,36880,40240,48170,51720,52320,52790,56880 ;
1,2,1,0,1,5,4,3,3,4,5,1 ;
52.42,12.47,54.40,52.68,27.73,26.98,37.90,36.71,37.37,31.03,33.51,15.05] ;
j=find(T(1, :)==k) ; % retourne le numero de la colonne
if (~isempty(j))
e=eulermat(phi(k),theta(k),psi(k))*[0 ; -sqrt(T(3,j)^2-(alt(k))^2) ; -alt(k)] ;
y=[e(1 :2) ; y] ; C=[zeros(2,nx) ; C] ;
m=T(2,j) ;
C(1,1)=1 ; C(1,3+2*m+1)=-1 ; C(2,2)=1 ; C(2,3+2*m+2)=-1 ;
Gbeta=0.01*eye(3,3) ;
end
```

Notons que la dimension des sorties dépend de la détection ou de la non-détection d'un amer.

5) L'évolution du système robot + amers peut être décrite par :

$$\underbrace{\begin{pmatrix} \mathbf{p}_{k+1} \\ \mathbf{q}_{k+1} \end{pmatrix}}_{\mathbf{x}_{k+1}} = \underbrace{\begin{pmatrix} \mathbf{I}_3 & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_{12} \end{pmatrix}}_{\mathbf{A}} \underbrace{\begin{pmatrix} \mathbf{p}_k \\ \mathbf{q}_k \end{pmatrix}}_{\mathbf{x}_k} + \underbrace{\begin{pmatrix} dt \cdot \mathbf{R}(k) \cdot \mathbf{v}_r \\ \mathbf{0}_{12 \times 1} \end{pmatrix}}_{\mathbf{u}_k} + \alpha_k$$

On utilise des listes pour mémoriser tous les résultats intermédiaires afin de pouvoir procéder au lissage (question suivante). On notera n_p la dimension de l'espace (ici 3), n_m le nombre d'amers et n_x la dimension du vecteur d'état. La trajectoire de confiance associée est dessinée sur la figure 7.31.

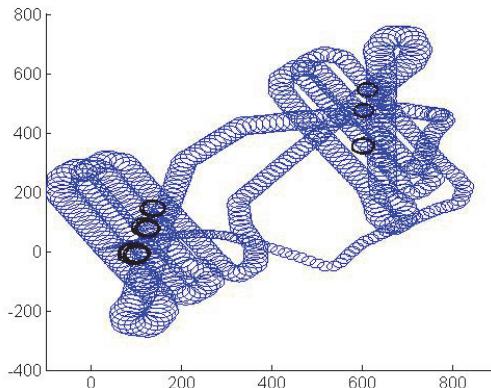


Figure 7.31. Trajectoire de confiance obtenue par le filtre de Kalman avec les ellipses pour les six amers

Le script correspondant au filtre de Kalman est :

```
np=3; nm=6; nx=np+2*nm;
M=dlmread('slam_data.txt');
t=M(:,1); phi=M(:,2); theta=M(:,3); psi=M(:,4);
vr=M(:,5:7);
depth=M(:,8); alt=M(:,9);
dt=0.1; kmax=size(M,1);
x_forw{1}=zeros(nx,1);
G_forw{1}=diag([1,1,1,1000000*ones(1,2*nm)]);
Galpha=[0.01*eye(np,np),zeros(np,2*nm);zeros(2*nm,nx)];
for k=1:kmax,
A=eye(nx,nx);
u{k}=[dt*euler(phi(k),theta(k),psi(k))*vr(k,:);zeros(2*nm,1)];
[y,C,Gbeta]=slam_g(k);
[x_forw{k+1},G_forw{k+1},xup{k},Gup{k}]=kalman(x_forw{k},
G_forw{k},u{k},y,Galpha,Gbeta,A,C);
end;
```

6) On rajoute à la suite du filtre de Kalman les instructions :

```
x_back{kmax}=xup{kmax};
G_back{kmax}=Gup{kmax};
```

```

for k=kmax-1 :-1 :1,
J=Gup{k} * A' / G_forw{k+1} ;
x_back{k}=xup{k}+J*(x_back{k+1}-x_forw{k+1}) ;
G_back{k}=Gup{k}+J*(G_back{k+1}-G_forw{k+1})*J' ;
end ;

```

La trajectoire associée est illustrée par la figure 7.32. On remarque que les ellipses de confiance sont plus petites qu'avec le filtre, surtout vers la fin de la mission. L'ensemble de tous les programmes associés à cet exercice sont rangés dans le fichier **slam.m**.

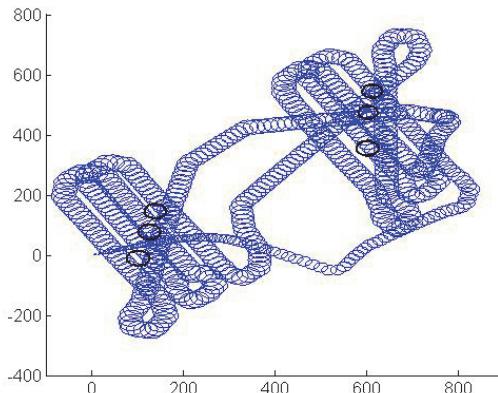


Figure 7.32. Trajectoire de confiance obtenue par le lisseur de Kalman avec les ellipses pour les six amers

Correction de l'exercice 7.23 (SLAM a priori)

Le raisonnement est donné dans le tableau qui suit :

t(H)	0	1	2	3	4	5	6	7	8
amer	0	1	2	1	3	2	1	4	0
(a)	10	110	210	310	410	510	610	710	810
(b)	10	710	610	510	410	310	210	110	10
(c)	10	110	210	310	410	310	210	110	10
(d)	10	110	210	110	310	210	110	110	10
(e)	10	110	210	110	210	210	110	110	10

Ligne (a) : précision obtenue en propageant dans le sens du temps ; ligne (b) : précision obtenue en rétropropageant dans le sens inverse du temps ; ligne (c) : minimum des lignes (a) et (b) ; ligne (d) : mise en correspondance entre les amers ; ligne (e) : propagation dans le sens direct et inverse du temps.

Bibliographie

- [BAZ 08] BAZEILLE S., Vision sous-marine monoculaire pour la reconnaissance d'objets, PhD thesis, Université de Bretagne Occidentale, 2008.
- [BEA 12] BEARD R., MCLAIN T., *Small Unmanned Aircraft, Theory and Practice*, Princeton University Press, 2012.
- [BOU 06] BOURLÈS H., *Systèmes linéaires ; de la modélisation à la commande*, Hermès-Lavoisier, Paris, 2006.
- [BOY 06] BOYER F., ALAMIR M., CHABLAT D., KHALIL W., LEROYER A., LEMOINE P., « Robot anguille sous-marin en 3d », *Techniques de l'Ingénieur*, 2006.
- [CHE 07] CHEVALLEREAU C., BESSONNET G., ABBA G., AOUSTIN Y., *Les robots marcheurs bipèdes ; Modélisation, conception, synthèse de la marche, commande*, Hermès-Lavoisier, Paris, 2007.
- [CRE 14] CREUZE V., « Robots marins et sous-marins ; perception, modélisation, commande », *Techniques de l'ingénieur*, 2014.
- [DEL 93] DE LARMINAT P., *Automatique, commande des systèmes linéaires*, Hermès, Paris, 1993.
- [DRE 11] DREVELLE V., Étude de méthodes ensemblistes robustes pour une localisation multisensorielle intègre, Application à la navigation des véhicules en milieu urbain, PhD dissertation, Université de Technologie de Compiègne, Compiègne, 2011.
- [DUB 57] DUBINS L.E., « On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents », *American Journal of Mathematics*, vol. 79, n° 3, p. 497-516, 1957.
- [FAN 01] FANTONI I., LOZANO R., *Non-linear control for underactuated mechanical systems*, Springer-Verlag, Londres, 2001.
- [FLI 13] FLIESS M., JOIN C., « Model-free control », *International Journal of Control*, vol. 86, n° 12, p. 2228-2252, 2013.
- [FOS 02] FOSSEN T., *Marine Control Systems : Guidance, Navigation and Control of Ships, Rigs and Underwater Vehicles*, Marine Cybernetics, Trondheim, 2002.

- [GOR 11] GORGUES T., MÉNAGE O., TERRE T., GAILLARD F., « An innovative approach of the surface layer sampling », *Journal des Sciences Halieutique et Aquatique*, vol. 4, p. 105-109, 2011.
- [GUI 11] GUILLOU G., Architecture multi-agents pour le pilotage automatique des voiliers de compétition et extensions algébriques des réseaux de Petri, PhD dissertation, Université de Bretagne, Brest, 2011.
- [HER 10] HERRERO P., JAULIN L., VEHÍ J., SAINZ M.A., « Guaranteed set-point computation with application to the control of a sailboat », *International Journal of Control Automation and Systems*, vol. 8, n° 1, p. 1-7, 2010.
- [JAU 04] JAULIN L., « Modélisation et commande d'un bateau à voile », *Conférence Internationale Francophone d'Automatique (CIFA'04)*, Douz, Tunisie, 2004.
- [JAU 05] JAULIN L., *Représentation d'état pour la modélisation et la commande des systèmes*, Hermès-Lavoisier, Paris, 2005.
- [JAU 10] JAULIN L., « Commande d'un skate-car par biomimétisme », *CIFA 2010*, Nancy, 2010.
- [JAU 12] JAULIN L., BARS F.L., « An Interval Approach for Stability Analysis ; Application to Sailboat Robotics », *IEEE Transaction on Robotics*, vol. 27, n° 5, 2012.
- [JAU 14] JAULIN L., *Automatique pour la robotique ; cours et exercices*, ISTE Editions, Paris, 2014.
- [KAL 60] KALMAN E.R., « Contributions to the theory of optimal control », *Bol. Soc. Mat. Mex.*, vol. 5, p. 102-119, 1960.
- [KIE 01] KIEFFER M., JAULIN L., WALTER E., MEIZEL D., « Localisation et suivi robustes d'un robot mobile grâce à l'analyse par intervalles », *Traitements du Signal*, vol. 17, n° 3, p. 207-219, 2001.
- [KLE 06] KLEIN E.M.V., *Aircraft System Identification : Theory And Practice*, American Institute of Aeronautics and Astronautics, 2006.
- [LAR 03] LAROCHE B., MARTIN P., PETIT N., Commande par platitude, Equations différentielles ordinaires et aux dérivées partielles, disponible à l'adresse : <http://cas.ensmp.fr/~petit/ensta/main.pdf>, 2003.
- [LAT 91] LATOMBE J., *Robot Motion Planning*, Kluwer Academic Publishers, Boston, MA, 1991.
- [LAU 01] LAUMOND J., *La robotique mobile*, Hermès, Paris, 2001.
- [LAV 06] LAVALLE S., *Planning algorithm*, Cambridge University Press, Cambridge, 2006.
- [ROM 12] ROMERO-RAMIREZ M., Contribution à la commande de voiliers robotisés, PhD dissertation, Université Pierre et Marie Curie, France, 2012.
- [WAL 14] WALTER E., *Numerical Methods and Optimization ; a Consumer Guide*, Springer, Londres, 2014.

Index

A

accéléromètre, 162
actionneur, 9
adjoint d'un vecteur de rotation, 16
aéroglissoir, 85
algorithme
 de Fortune, 154
 de *sweep line*, 154
ancrage, 122
angle(s), 113, 158
 d'Euler, 19
 de Cardan, 19
antidéplacement, 15
arc capable, 166, 167
assiette, 20
atan2, 20, 173
AUV, 33

B

baromètre, 164
bateau à voile, 67
biomimétisme, 105
blancheur, 205
bouclage linéarisant, 51, 55
brownien, 221

C

cap, 20
capteur, 9
carte locale, 136
centrale inertielle, 163
champ de potentiel artificiel, 142
char, 61
chemin de Dubins, 147
commande
 mimétique, 100
 non linéaire, 51
 proportionnelle, 76
 et dérivée, 52, 65
 sans modèle, 100
coordonnées
 géographiques, 135
 homogènes, 32
correction, 215
covariance, 204

D

dead reckoning, 19, 228, 254
degré relatif, 57
déplacement, 15
dérivée d'une forme quadratique, 186
DGPS, 165
diagramme de Voronoi, 141

drone volant, 125

dynamique

de l'erreur, 52

des zéros, 77

E

ellipsoïde de confiance, 206

ensemble des sorties acceptables, 59

estimateur, 189

biaisé, 209

linéaire, 209

orthogonal, 209

estimation

à l'estime, 19

linéaire, 213

F

facteur d'échelle, 225

filtrage, 203

filtre de Kalman, 214

fonction

à dents de scie, 101

quadratique, 185

forme quadratique, 185, 186

formule

de Rodrigues, 26

de Varignon, 26

G

gain de Kalman, 211

garde le près, 115

GESMA, 31

gisement, 166

gîte, 20

GNSS, 164

GPS, 164, 170

cinématique, 165

grand gain, 73, 76

graphe des dépendances

différentielles, 69, 85, 94

groupe spécial orthogonal, 15

guidage, 135

gyromètre, 162

gyroscope, 162

H, I

hovercraft, 85

identifiable, 199

identification, 185

identité de Jacobi, 25

IMU, 163

indépendance, 205

intelligence, 9

inverse généralisée, 57, 189

L

lacet, 20

lidar, 172

linéarisation par bouclage, 51, 55

 dynamique, 97

 statique, 51

linéarité par rapport aux paramètres, 188

lissage, 217

lisseur de Kalman, 217

localisation, 161, 193

 goniométrique, 165

Loch-Doppler, 162

M

manivelle, 78

marche aléatoire, 241, 264

matrice

 adjointe, 25

 cosinus directeur, 18

 de covariance, 204

 de rotation, 14

modélisation, 13

modèle

 cinématique, 22, 73, 76

 dynamique, 73

modulo, 101

moindres-carrés, 188

multilatération, 170

N, O

navigation, 161
 à l'estime, 228, 254
 nuage de points, 204
 odomètre, 162

P

paramétrisation de
 Denavit-Hartenberg, 34
 pendule simple, 54
 période de Schuler, 29
 phénomène de Runge, 140
 PID, 53, 57
 planification de trajectoires, 138
 polaire des vitesses, 68
 polynômes
 de Bézier, 139
 de Bernstein, 140
 prédiction, 215
 produit vectoriel, 16
 PWM, 108

R

rapport cyclique, 108
 recuit simulé, 190, 202
 Redermor, 31
 régulateur statique, 66, 72
 représentation d'état, 13
 résidu, 189
 retard différentiel, 58
 robot
 manipulateur, 34
 mobile, 9

patineur, 103
 mobile, 9
 roue, 35
 roulis, 20
 ROV, 102
 Runge Kutta, 29

S

SAUCISSE, 84
sawtooth, 101, 116, 127, 133, 155, 158, 160
 singularités, 59
skate car, 103
snakeboard, 108
 SNAME, 21
 Staubli, 34
 système
 plat, 82
 redondant, 57

T

tangage, 20
tanh, 100
 télémètre, 193
 triangulation, 167
 de Delaunay, 141

V

variance, 204
 variation, 204
 vecteur de rotation, 16
 voilier, 110
 de Dubins, 146, 147

