MIR PROGRAM

LINEAR MULTIVARIABLE CONTROL COURSE AS PART OF THE MUNDUS MIR MASTER

GUIDE TO PRATICAL WORK

ORIGIN	DOCUMENT N°	ISSUE
TOU	###-#-##-###	00

NB. OF PAGES	WRITTEN BY: DHAISNE Aurélien	COPY
9	DATE: 02/06/2020	N° /

DOCUMENT FOLLOW-UP

Issue	Date	Author	PURPOSE OF THE MODIFICATION
00	02/06/2020	ADH	Creation

	Name/Title	Date	Signature
Written by:	DHAISNE Aurélien Engineer at ECA Robotics		
Checked by:	OTT Hervé Head of the "scientific cell"		
Approved by:	FIDANI Alain Research and innovation director		

Internal distribution :
Fidani Alain; Ott Hervé ; Bolin Noë
External distribution :
Mr. Boizot Nicolas and the students of the MIR program

CONTENTS

1. GENERALITY 4

1.1. PURPOSE 4
1.2. DOCUMENTS OF REFERENCE 4
1.3. ACRONYMS 4
1.4. REQUIRED 4

2. GUIDELINE 5

2.1. MODEL IDENTIFICATION 5
2.1.1. Help on identification 5
2.2. OPTIMAL CONTROL 6
2.2.1. Option 1: Vertical plane, in depth command: 6
2.2.2. Option 2: Vertical plane, in pitch command: .... **Erreur ! Signet non défini.**
2.2.3. Option 3: Horizontal plane, in heading command: 6
2.2.4. Option 4: Horizontal plane, in yawrate command: .... **Erreur ! Signet non défini.**
2.2.5. Steps to take, to set up the control law: 6

LIST OF FIGURES

FIGURE 1 : PHOTO OF AN A18 4

1. GENERALITY

1.1. PURPOSE

ECARobotics created underwater drones called AUV. The purpose of these drones is to independently scan areas. The AUV, equipped with a sonar, has the function of detecting underwater mines. Indeed, mines are low-cost and effective solutions to destroy a military ship or a submarine.

The problem is that an AUV is equipped with a sonar and that this sensor requires high criteria of stability on all levels (vertical plan, horizontal plan, roll,…). With this guide, a simplified simulator of an AUV will be provided. The control command used in this simulator is a PID and this control lacks robustness and stability. The aim of this practical work will be to write an optimal control law on one of the planes.

Sonar can be deployed on different carriers and in this practical work we will work on the case of the “A18M”, just one of the AUV.

[Link of a video, online, of an “A18M”]

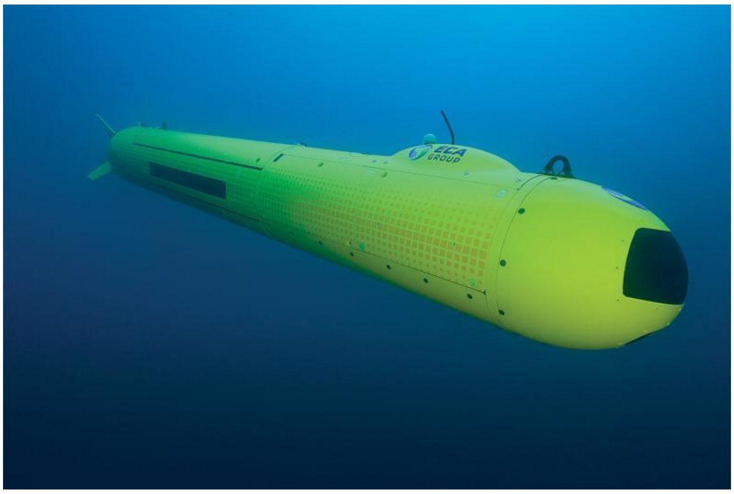

Figure 1 : Photo of an A18

1.2. DOCUMENTS OF REFERENCE

	Reference	Date	Title
[DR01]		02/06/2020	AUV Model.pdf

1.3. ACRONYMS

AUV	Autonomous Underwater Vehicle
LQG	Linear Quadratic Regulator
LQR	Linear Quadratic Gaussian
PID	Proportional Integral Derivative

1.4. REQUIRED

- Simulink
- Matlab
- Control System Toolbox in Matlab (or a substitute to solve The Riccati equation).

2. Guideline

Imagine that your vehicle is operating with PIDs. Its behavior is "good" but good not enough for the new sensor that your company wants to put on. The new behavior is much stricter and requires more complex control. You have more choices in mind: LQR command, LQG, robust, adaptive, ... you are running out of time and the simplest and fastest solution is to develop an LQR command.

The purpose of this practical work is to show the entire process of creating a control command law from a model. The model is developed in [DR01]. The first step will be to identify the missing parameters of the model using the data provided. Then, it will be necessary to create a control-command law adapted to the specified need.

2.1. MODEL IDENTIFICATION

All parameters are provided except the damping parameters in the horizontal and the vertical plan, as: $K_{sh}, CY_{uv}, CY_{ur}, CN_{uv}, CN_{ur}, CZ_{uw}, CZ_{uq}, CM_{uw}, CM_{uq}$.

To identify them, you have data from some missions:

- **Speed step**: On this mission, the AUV makes pure advance speed steps (called "u") with all other speeds are so small than we consider them as null. These data are particularly useful for identifying K_{sh}.
- **Pitch step**: On this mission, the AUV makes pitch steps (called "θ") that implies speeds steps as "w" and "q", all other speeds are so small than we consider them as null. These data are particularly useful for identifying parameters as CZ and CM.
- **Dieudonnée**: On this mission, the AUV makes virtual helms steps (called "A") that implies speeds steps as "v" and "r", all other speeds are so small than we consider them as null. These data are particularly useful for identifying parameters as CY and CN.

2.1.1. ²Help on identification

The identification of $K_{sh}, CZ_{uw}, CZ_{uq}, CM_{uw}, CM_{uq}$ are simple, you can use the identification method of your choice. (Example: using least square via the Matlab function fminsearch or using y = Ax => A = x\y).

Identified $CY_{uv}, CY_{ur}, CN_{uv}, CN_{ur}$ is more complicated because you have several local minima and using least squares does not work unless you have the correct initial values. A tip is to use another identification method or a less square under conditions.
Help:

- You can us fmincon if you have the optimization toolbox if not you have "fminsearchcon" at your disposal (opensource code).
- $CY_{uv} \sim CZ_{uw}$ (at ±20%) because of the symmetry
- $CY_{ur} \sim -CZ_{uq}$ (at ±2.5%) because of the symmetry
- $CN_{uv} \sim -CM_{uw}$ (at ±10%) because of the symmetry
- $CN_{ur} \sim CM_{uq}$ (at ±10%) because of the symmetry

After identification, run a simulation with your parameters and check that the simulation is stable. If this is the case, check that your simulation gives a result consistent with the mission data provided.

Before going to the next step, ask for the teacher's validation

2.2. OPTIMAL CONTROL

To begin, a PID control has already been implemented on the 4 planes: advance, roll, horizontal and vertical. The PID commands are not stable enough to allow the AUV to properly identify mines, so you have to switch to a more stable and robust command: the optimal command.

Optimal control are too simple for advance and too complicated for roll, you can choose to make one of a control either in the vertical plane, or in the horizontal plane.

The criteria to minimize are also to be chosen by yourself. To help you, here are some macroscopic behaviors that we want to have:

2.2.1. Option 1: Vertical plane, in depth command

- We want to follow a depth command without having too many pitch oscillations (low “q”).
- We want with a very low or even zero final error, if possible, and above all not to oscillate around the command.
- We, also, do not want to have an oscillating lift speed (low “w”).
- We do not want too big an overshoot (maximum 2m).

2.2.2. Option 2: Horizontal plane, in yawrate command:

- This command does not exist, you have to create it. Help will be provided to integrate it into the simulator and the control configuration file.
- We want with a very low or even zero final error, if possible, and above all not to oscillate around the command when the yawrate is stabilize.
- No overshoot allowed.

2.2.3. Steps to take, to set up the control law:

- Choose one of the options above.
- Linearize the non-linear model according to the criteria to minimize that you have chosen, if you can help yourself from the document in [DR01] (trick: Jacobian matrix).
- **Before going to the next step, ask for the teacher's validation**
- Compute state and the steady states for the command
- **Before going to the next step, ask for the teacher's validation**
- Have your linearization checked. Help will be provided to integrate your order into the simulator.
- Apply the correct weights so that the AUV has the desired behavior in your chosen option.

2.3. HELPS

2.3.1. Help on the inversion of the mass matrix

If we simplify the mass matrix of the vehicle by noting it:

$$M = \begin{bmatrix} m11 + X_{\dot{u}} & m12 & m13 & m14 & m15 & m16 \\ m12 & m22 + Y_{\dot{v}} & m23 & m24 & m25 & m26 \\ m13 & m23 & m33 + Z_{\dot{w}} & m34 & m35 & m36 \\ m14 & m24 & m34 & m44 + K_{\dot{p}} & m45 & m46 \\ m15 & m25 & m35 & m45 & m55 + M_{\dot{q}} & m56 \\ m16 & m26 & m36 & m46 & m56 & m66 + N_{\dot{r}} \end{bmatrix} = \begin{bmatrix} m11_t & m12 & m13 & m14 & m15 & m16 \\ m12 & m22_t & m23 & m24 & m25 & m26 \\ m13 & m23 & m33_t & m34 & m35 & m36 \\ m14 & m24 & m34 & m44_t & m45 & m46 \\ m15 & m25 & m35 & m45 & m55_t & m56 \\ m16 & m26 & m36 & m46 & m56 & m55_t \end{bmatrix}$$

<u>For the **horizontal** plane, I recommend you to simplify the equations by equated:</u>

$$Det_M = m11_t.m22_t.m66_t - m11_t.m26^2 - m12^2.m66_t + 2.m12.m16.m26 - m16^2.m22_t$$

$$Det_{11} = \frac{m22_t.m66_t - m26^2}{Det_M}$$

$$Det_{12} = -\frac{m12.m66_t - m16.m26}{Det_M}$$

$$Det_{16} = \frac{m12.m26 - m16.m22_t}{Det_M}$$

$$Det_{22} = \frac{m11_t.m66_t - m16^2}{Det_M}$$

$$Det_{26} = -\frac{m11_t.m26 - m12.m16}{Det_M}$$

$$Det_{66} = \frac{m11_t.m22_t - m12^2}{Det_M}$$

That imply, in horizontal plane

$$M^{-1} = \begin{bmatrix} Det_{11} & Det_{12} & 0 & 0 & 0 & Det_{16} \\ Det_{12} & Det_{22} & 0 & 0 & 0 & Det_{26} \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ Det_{16} & Det_{26} & 0 & 0 & 0 & Det_{66} \end{bmatrix} => \begin{bmatrix} Det_{11} & Det_{12} & Det_{16} \\ Det_{12} & Det_{22} & Det_{26} \\ Det_{16} & Det_{26} & Det_{66} \end{bmatrix} => \begin{bmatrix} Det_{11} & 0 & 0 \\ 0 & Det_{22} & 0 \\ 0 & 0 & Det_{66} \end{bmatrix}$$

For the **vertical** plane, I recommend you to simplify the equations by equated:

$$Det_M = m11_t.m33_t.m55_t - m11_t.m35^2 - m13^2.m55_t + 2.m13.m15.m35 - m15^2.m33_t$$

$$Det_{11} = \frac{m33_t.m55_t - m35^2}{Det_M}$$

$$Det_{13} = -\frac{m13.m55_t - m15.m35}{Det_M}$$

$$Det_{15} = \frac{m13.m35 - m15.m33_t}{Det_M}$$

$$Det_{33} = \frac{m11_t.m55_t - m15^2}{Det_M}$$

$$Det_{35} = -\frac{m11_t.m35 - m13.m15}{Det_M}$$

$$Det_{55} = \frac{m11_t.m33_t - m13^2}{Det_M}$$

That imply, in vertical plan

$$M^{-1} = \begin{bmatrix} Det_{11} & 0 & Det_{13} & 0 & Det_{15} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ Det_{13} & 0 & Det_{33} & 0 & Det_{35} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ Det_{15} & 0 & Det_{35} & 0 & Det_{55} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} => \begin{bmatrix} Det_{11} & Det_{13} & Det_{15} \\ Det_{13} & Det_{33} & Det_{35} \\ Det_{15} & Det_{35} & Det_{55} \end{bmatrix} => \begin{bmatrix} Det_{11} & 0 & 0 \\ 0 & Det_{33} & 0 \\ 0 & 0 & Det_{55} \end{bmatrix}$$

Trick for linearization of the model

We can easily consider an AUV as a symmetrical cylinder so: For i=1..6 and j =1..6 all mass $m_{ij} = 0$ and $m_{ii} \neq 0$. Use this hypothesis before the inversion of the matrix to win time and in complexity.

2.3.2. Verification by numerical Application

To verify is your A and B matrix of the state system are good, you can use this specific numerical application.

Here are some values to check your formulas. The values of the hydrodynamics and helms parameters are not physical at all, they are just there to help you check your results:

$$CYuv := -1 : CYur := 2 : CZuw := -3 : CZuq := -4 : CMuw := 5 : CMuq := -6 : CNuv := -7 : CNur := -8 :$$
$$CY := 0.1 : CZ := -0.2 : CM := -0.3 : CN := -0.4 :$$
$$\rho := 1026 : Sref := 0.385 : Lref := 5 : m := 1974.26 : zv := 0 : zg := 3e\text{-}2 :$$
$$q := q0 : W := 9.81 \cdot m : B := W : u0 := 2 : \theta := \theta0 : w := w0 :$$

$$CZ0 := -0.01 : CM0 := -0.002 : CY0 := 0 : CN0 := 0 :$$
$$Iy := 4173.5 : Iz := 4173.5 : Yv := 1845.9 : Zw := 1845.9 : Mq := 3388.7 : Nr := 3388.7 :$$

$$q0 := 0 : w0 := u0 \cdot \tan(\theta0) : r0 := eval\left(\frac{31.27 \cdot \text{Pi}}{180}\right) :$$

2.3.2.1. Option 1: Vertical plane, in depth command

$$BAR_0 = -0.02915087136$$
$$\theta_0 = -0.001389941910$$
$$\begin{bmatrix} \dot{w} \\ \dot{q} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} -0.3102042846 & -1.034427877 & X \\ 1.305869985 & -7.835198137 & -0.07683268846 \\ X & X & X \end{bmatrix} . \begin{bmatrix} w \\ q \\ \theta \end{bmatrix}$$
$$\begin{bmatrix} \dot{w} \\ \dot{q} \end{bmatrix} = \begin{bmatrix} -0.04136057128 \\ -0.1567043982 \end{bmatrix} . BAR$$

2.3.2.2. Option 2: Horizontal plane, in heading command

$$A_0 = -9.929936045$$
$$\theta_0 = -1.983804207$$
$$\begin{bmatrix} \dot{v} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} -0.1034014282 & 0.0004135952421 \\ -1.828217979 & -10.44695988 \end{bmatrix} . \begin{bmatrix} v \\ r \end{bmatrix}$$
$$\begin{bmatrix} \dot{v} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} 0.02068028564 \\ -0.2089391976 \end{bmatrix} . A$$

2.3.3. Coding

- LQR problem

If you have this error : `Function 'lqr' not supported for code generation.`
Add in your function this line : “`coder.extrinsic('lqr');`”

- lqr / lqrd / dlqr / …

You work in continuous form of the state representation, you have to linearize the matrices (A, B, …). Be careful on the differences between the matlab function : lqr, dlqr and lqrd.