FINAL DEGREE PROJECT

Study: Industrial Engineering

Title: # Dynamics modelling of an Under Water Vehicle

Document: Memory and Annexes

Student: Anna Tibau Ragolta

Tutor: David Lane and Yvan Petillot

Department: Computing and Electrical Engineering

Date: February-July 2006

1. INTRODUCTION

1.1 Antecedents

Heriot Watt University of Edinburgh is going to take part on the *Student Autonomous Underwater Competition – Europe (SAUC-E)* in August 2006. A team of students from different disciplines has been building and developing a robot called Nessie during this and last year. At this time, the structure and hardware of the robot are almost finished and the team is entering in the mission-planning period. It means that they are working altogether using the software they have developed to resolve the missions of the competition. Simplifying, the mechanism consists in get environment information as position of the robot and the target, to process it and send movement information to the propulsion system. In February 2006, I joined the team to help with the control system by studying the robot's dynamic and finding the mathematic model of movement as it can be use to set the PID parameters more accurately.

1.2 Objective

The main objective of the project is to find the dynamic equation of the robot, dynamic model, by using the present knowledge on this area. With it, the behaviour of the robot will be easier to understand and movement tests will be available by computer without the need of the robot, what is a way to save time, batteries, money and the robot from water inside itself. Because the competition day is approaching, if there is time enough, a second part in this project will be setting a control system for Nessie by using the model.

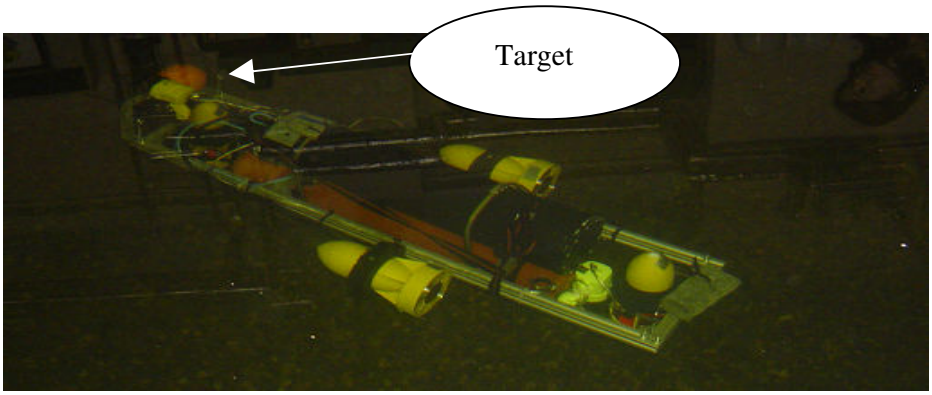
1.3 Specifications

The robot can move in four degrees of freedom (surge, heave, pitch and yaw) through the four propellers it has. Roll and Sway movements are not wanted but also not accessible to control, as it does not have propellers for them. However, the robot structure guaranties the stability in these two degrees of freedom. Pitch movement is also not wanted, so I will not consider it for the equation as well.

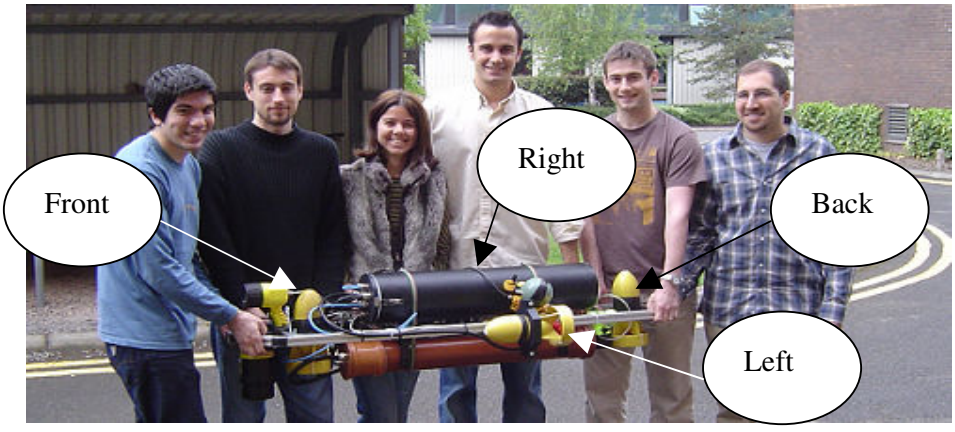
By simplifying it, it is easy to see that there are only 3 degrees of freedom to take into account, so the model becomes simpler. These three DOF's are totally controllable by the propulsion system, that consist in two propellers to move in surge direction and two more to move in the heave. The robot will be moving on a tank without water currents.

2. DESCRIPTION OF NESSIE

Nessie is an Autonomous Underwater Vehicle (AUV) that is created by a team of students in the Heriot Watt University to compete in the Student Autonomous Underwater Competition, Europe (SAUC-E) in August 2006.
There are several missions to achieve, an all them need an accurate control of the movement.
For instance, one of the missions consists in to locate a mid-water target and contact it with the nose of the vehicle as shown below:



This task requires a very good control of the vehicle, since the vehicle is autonomous, he positions himself and the targets by reading the information from his sensors, and he moves by giving the quantity he “thinks” necessary to his motion system.
For the motion system, Nessie dispose of four thrusters, one on the front and one on the back for immersion and surfacing, and one on the left and one on the right for move forward, backward and turn in yaw.



An Insertion Navigation System (INS), a Sound Navigation and Ranging (SONAR), two webcams and an altimeter, compose the environment information system.

3. DESCRIPTION OF THE DYNAMIC EQUATION FOR AN AUV

This equation represents the fundamental principle of dynamics; witch says that the force applied to a body is proportional to the acceleration the body gets. The proportional constant is the mass of the body, and it can be expressed as the following way:

$$F = m \cdot a \qquad \text{(Eq. 1)}$$

But the robot is moving in a environment that dissipates energy because is causing a resistance to the movement. This is the principle of action-reaction, also known as Newton's third low. The water is making forces against the movement of the robot. Then we have to rest these forces to the thrusters force, and if it is still positive, there will be acceleration and the robot will move.

Because the robot is submerged in the water, an other factor is making forces to the robot, is the principle of Archimedes, which states that a body immersed in a fluid is buoyed up by a force equal to the weight of the displaced fluid. Therefore, it will affect the heave degree of freedom.

Once we have a mass with acceleration, the principle of Galileo states that every system that is put under a force, it tends to get acceleration and change his velocity offering a resistance or inertia to it.

An other thing to take into account is that when the movement is linear, the variables are force and linear acceleration, and the constant is mass. But when the movement is angular, then the variables are torque force and angular acceleration, and the constant is the moment of inertia.
Thee degrees of freedom are linear and the tree left are angular.

With all this knowledge is time to write the equation of movement.

$$M \cdot \dot{v} + D \cdot v + B \cdot v \cdot |v| + C(v) \cdot v + g(\eta) = \tau \qquad \text{(Eq.2)}$$

Where M is a 6x6 matrix than contains masses and moments of inertia.
D is a matrix that contains the values of the linear drag coefficients. It is the reaction against the movement that the water does. B is a matrix that contains the values of the

quadratic drag coefficients, that represents the reaction against the movement that the water does due to the viscosity. C is a matrix that contains the values of the centripetal terms due a mass being in movement. g is a matrix with the values of the gravity and buoyancy terms, and τ is a vector that contains the force and torques that the propellers give.

Once we have all these values, we will obtain the model of the system (Nessie under the water) that will represent exactly the real behaviour of it.

The main task of this project is to find these values. For this, it is needed to do tests to get data, and through them, identify each value. It all is explained and done in the ANNEXES.

4. DESCRIPTION OF THE IDENTIFICATION METHOD

To find the parameters of the equation, I will use an identification method. As I said, these parameters are the result of physic laws and are difficult to calculate by following physical rules.
An identification method consists of finding the values of the matrices by studying the system as a black box from which we know the input and output values at any time.
If we know the behaviour of the robot, the outputs agreed with the inputs, we can also find a model that behaves the same way.
There are several identification methods to use, but the one I will use is a method created by Antonio Tiano, from the University of Pavia (Italy), that has been used to find the dynamic equation of other robots with good results.
This method uses the mathematic concept of the Least Square (LS) algorithm.

To do the identification, the data we need to know from the robot is the velocity and the force of the thrusters at any time. With this two data, the method will give the values of the matrices from the model.

To get these data we have to do some tests in the tank. These tests have to be uncoupled, that means that in each tests there is just one degree of freedom with movement.

The identification method it is all explained in the ANNEX 2

5. DESCRIPTION OF THE EXPERIMENTS

Identification of the thrusters

This test consists in to find the relation between the angular speed of the thrusters and the force they give.
The methodology used is very simple. The robot has to be stopped in the water, which means without speed, to be able to neglect the drags and inertias. Then, with a dynamometer and a rope, we can measure the force that the robot gives when increasing the angular speed of the thrusters.

This simple drawing illustrates how the test is.

Fig.1 Thrusters identification

These tests were done on the small tank of the Ocean Laboratory in Heriot Watt.
The results of these tests can be found in the ANNEX 4.

Identification of the parameters of the equation

The objective of there tests is to find the values of the equation matrices. To use the identification algorithm it is needed to know the speed of the robot and the angular speed of the thrusters at any time.
These tests need a big pool, since we want to reach the maximum speed of the thrusters and robot.
A small program in C++ has been done to store these values while the robot is moving, since the robot uses a Linux operating system. It can be found in the ANNEX 11, but to be understood it is need to see the programs of the other members of the team that have to run at the same time.

Pictures from the tests day in the pool can be found in ANNEX 12.

6. DESCRIPTION OF THE RESULT

Finally and after 5-month study, the mathematic equations of Nessie obtained in each uncoupled degree of freedom are the following:

Surge forward:

$$33{,}97 \cdot \dot{u} + 24 \cdot u + 7 \cdot |u| \cdot u = 0{,}0179 \cdot U_1 + 0{,}0179 \cdot U_2 \quad \text{(Eq.3)}$$

Surge backward:

$$33{,}97 \cdot \dot{u} + 24 \cdot u + 7 \cdot |u| \cdot u = 0{,}0016 \cdot U_1 + 0{,}0016 \cdot U_2 \quad \text{(Eq.4)}$$

Heave up:

$$-88 \cdot \dot{w} + 3{,}44 \cdot w = 0{,}0042 \cdot U_3 + 0{,}0042 \cdot U_4 \quad \text{(Eq.5)}$$

Heave down:

$$-88 \cdot \dot{w} + 3{,}44 \cdot w = 0{,}018 \cdot U_3 + 0{,}018 \cdot U_4 \quad \text{(Eq.6)}$$

Yaw right:

$$-723{,}7 \cdot \dot{r} + 170 \cdot r = 0{,}0179 \cdot 0{,}225 \cdot U_1 + 0{,}0016 \cdot 0{,}225 \cdot U_2 \quad \text{(Eq.7)}$$

Yaw left:

$$-723{,}7 \cdot \dot{r} + 170 \cdot r = 0{,}0016 \cdot 0{,}225 \cdot U_1 + 0{,}0179 \cdot 0{,}225 \cdot U_2 \quad \text{(Eq.8)}$$

Where the inputs U, are the square of the speed in revolutions per second (rps) on each thruster.

8. CONCLUSIONS

After applying the identification algorithm to the data and transform the parameters to the real constants of the equations, is easy to see that most of them are incoherent. The sum of the mass and the added mass cannot be negative, as it happens in equations 5, 6, 7 and 8. In addition, the linear speed constants in equations 7 and 8 seam too big. Nevertheless, I did not have time to compare to model with the real robot to assure that the equation is incoherent.

If they are, one of the main reasons is that the sensors in Nessie aren't very good and accurate. The INS has been used to measure the yaw, and through this, calculate the yaw speed. For surge and heave, I had to use the altimeter and integrate the data to find the speed, because with the INS there was a very big error between the real speed and the measured.

The altimeter was also giving wrong measurements sometimes, and the data are distorted because of this.
I could probably solve this problem by doing several tests and making an average, but unfortunately, a place to do the tests was not available since few days before having to present this project, so there was no time.

If with better data, the results are still incoherent, then the problem is in the identification algorithm program. The method have been used before with satisfactory results, so it have to work for Nesse as well.

Although I'm not happy with the results, I'm happy with all the knowledge I have leaned working on this project. I have helped the AUV team with the buoyancy, the suitable centre of gravity to turn about and do not lose the target location while moving. In addition, the identification of the thrusters has been useful to be able to turn about the centre of gravity. Moreover, I hope the new team will be able to find a good model of the new Heriot Watt AUV from the information of this project.

8. LIST OF CONTENTS

MEMORY

ANNEXES

9. BIBLIOGRAPHY

M.Carreras, A.Tiano, A.El-Fakdi, P.Ridao. *On the identification of non linear models of unmanned underwater vehicles (2002).*

Giovanni Indiveri. *Modelling and Identification of Underwater Robotic Systems. (1998).*

Thor I. Fossen. *Guidance and Control of Ocean Vehicles (1994).*

A. Meystel. *Autonomous Mobile Robots (1991)*

Benjamin C.Kuo. *Digital Control Systems (1992)*

Katsuhiko Ogata. *Modern Control Engineering (1997)*

Gene F. Franklin. *Digital Control of Dynamic Systems (1998)*

10. GLOSSARY

AUV: Autonomous Underwater Vehicle

OSL: Ocean Laboratory of Heriot Watt University

DOF: Degree of Freedom

COG: Centre of Gravity

COB: Centre of Buoyancy

PID: Proportional, Integrative, Derivative

ANNEXES

ANNEX.1 VECTORS NOTATION

To make it easier to understand, I will use the known Notation for marine vehicles by the Society of Naval Architects and Marine Engineers (SNAME):

DOF	Description	Name	Forces & Moments	Linear & angular v.	Position & Euler angles
1	Motion in x-direction	Surge	X	u	x
2	Motion in y-direction	Sway	Y	v	y
3	Motion in z-direction	Heave	Z	w	z
4	Rotation about the x-axis	Roll	K	p	Ø
5	Rotation about the y-axis	Pitch	M	q	θ
6	Rotation about the z-axis	Yaw	N	r	ω

U **:** Thrusters angular speed

τ **:** Thrusters torque

F **:** Thrusters force

Thruster 1: Left Thruster 3: Front

Thruster 2: Right Thruster 4: Back

Where X, Y and Z axes of the robot coordinate frame are the ones shown in the picture bellow:

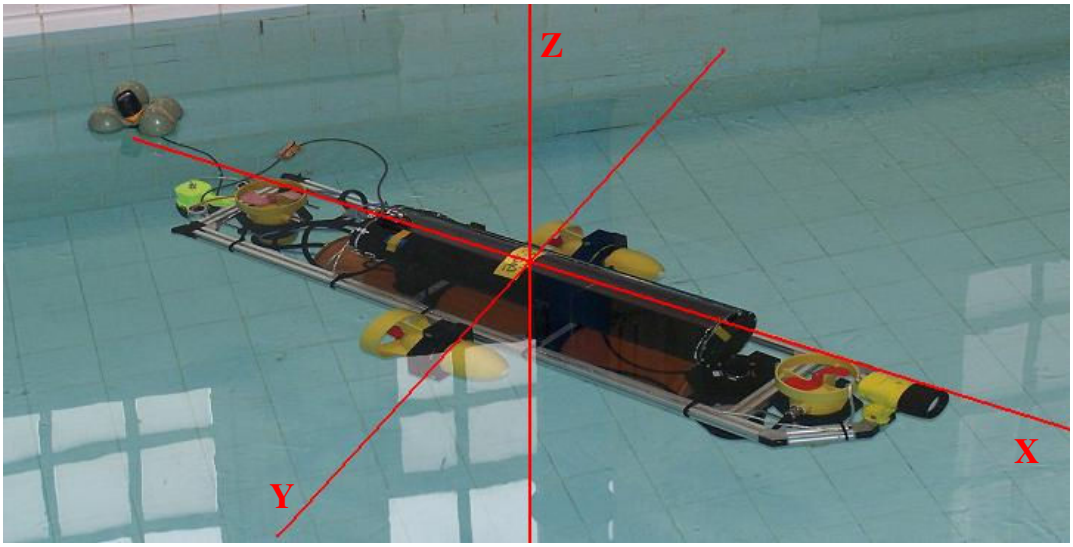


Fig.2 Axes of the robot

ANNEX.2 THE IDENTIFICATION METHOD'S THEORY

2.1 Introduction

To use this method we need the equation of movement written for each single degree of freedom. The tests will be uncoupled, that means that when we study the surge, we will just have surge movement, and there will be many simplifications in each single equation.

In the following equations, v means linear speed (for surge, heave and sway) and w angular speed (for roll, pitch and yaw).

Surge:

$$m \cdot \dot{v} + D \cdot v + B \cdot v^2 = F \quad \text{(Eq.9)}$$

Sway:

$$m \cdot \dot{v} + D \cdot v + B \cdot v^2 = F \quad \text{(Eq10)}$$

Heave:

$$m \cdot \dot{v} + D \cdot v + B \cdot v^2 = F \quad \text{(Eq.11)}$$

Roll:

$$I \cdot \dot{\omega} + D \cdot \omega + B \cdot \omega^2 + C(\omega) \cdot \omega + g(\eta) = \tau \quad \text{(Eq.12)}$$

Pitch:

$$I \cdot \dot{\omega} + D \cdot \omega + B \cdot \omega^2 + C(\omega) \cdot \omega + g(\eta) = \tau \quad \text{(Eq.13)}$$

Yaw:

$$I \cdot \dot{\omega} + D \cdot \omega + B \cdot \omega^2 + C(\omega) \cdot \omega = \tau \quad \text{(Eq.14)}$$

I have just taken into account the gravity and buoyancy term only in the DOF where it affects.
In addition, the centripetal terms in some cases are zero as well because there is no movement in the other degrees of freedom.

As I said before, because of Nessie's kind of movement, the only degrees of freedom that have to be studied are surge, heave and pitch. Therefore, let us study these tree ones.

2.2 Identification of surge parameters

I will describe here all the method for the surge. The other movements will be the same but with its characteristics. Let us write the equation (Eq.3) again:

$$m \cdot \dot{v} + D \cdot v + B \cdot v^2 = F \qquad \text{(Eq.3)}$$

We can write it as:

$$\dot{v} = \frac{F}{m} - \frac{D}{m} \cdot v - \frac{B \cdot |v|}{m} \cdot v \qquad \text{(Eq.15)}$$

Then simplify it as:

$$\dot{v} = \alpha \cdot F - \beta \cdot v - \delta \cdot |v| \cdot v \qquad \text{(Eq.16)}$$

Where:

$$\alpha = \frac{1}{m} \qquad \beta = \frac{D}{m} \qquad \delta = \frac{B}{m} \qquad \text{(Eq.17) (Eq.18) (Eq.19)}$$

Then we see that the acceleration is function of two variables, the velocity and the force, and it can be written as:

$$\dot{v} = \Phi(v(t), F(t))\Theta \qquad \text{(Eq.20)}$$

Where:

Φ A matrix contains values that only depend on the state and control vectors.

Θ $(\alpha\ \beta\ \gamma\ \delta)^T$ A constant vector with the parameters that characterized the dynamic of the system.

We can estimate the value of the Θ by minimizing the scalar cost of the J (Θ) function expressed as shown below:

$$J(\Theta) = \frac{1}{N} \sum_{k=1}^{N} \varepsilon^T(t_k) \cdot W^{-1}(t_k) \cdot \varepsilon(t_k) \qquad \text{(Eq.21)}$$

$W^{-1}(t_k)$ A weight matrix that takes into account the fidelity of the measures

$\varepsilon(t_k)$ The estimation error

Where the estimation error is the difference between the real velocity value and the estimated velocity value.

$$\varepsilon(t_k) = v(t_k) - \tilde{v}(t_k) \quad \text{(Eq.22)}$$

The interesting thing remains on the predicted error, because to predict the speed one-step forward, we need to find a model of the movement.
Then, minimizing the cost function, we will minimize the predicted error as well, and this will let us find the more accurate model of the movement.

For the predicted speed, one-step forward we use the integral between now and one-step backward of the Eq.20 as shown bellow:

$$v(t_k) - v(t_{k-1}) = \left[\int_{t_{k-1}}^{t_k} \Phi v(s), F(s)).ds \right] \cdot \Theta = F \cdot \Theta \quad \text{(Eq.23)}$$

Then the predicted velocity is given by the expression:

$$\tilde{v}(t_k) = v(t_{k-1}) + F\Theta \quad \text{(Eq.24)}$$

And we can rewrite the error as:

$$\varepsilon(t_k) = v(t_k) - v(t_{k-1}) - F\Theta \quad \text{(Eq.25)}$$

We can see that the only data we need to measure is time, velocity and force at every time instant.
Once we have the expression of the prediction error, we can apply the LS algorithm to minimize the function cost.

The cost function for N experiments is written in a matrix form as:

$$\Theta(N) = (F^T(N) \cdot W^{-1}(N) \cdot F(N))^{-1} \cdot F^T(N) \cdot W^{-1}(N) \cdot Y(N) \quad \text{(Eq.26)}$$

Where F will contains all the integrals, Y will contains all the rest ($v(t_k) - v(t_{k-1})$), and W will contains all the weights according to the test values fidelity.

2.3 Identification of heave parameters

Let us do the same with the heave. This time the gravity and buoyancy terms have to be taken into account

Let us rewrite the equation 9.

$$m \cdot \dot{v} + D \cdot v + B \cdot v^2 = F \qquad \text{(Eq.9)}$$

We can simplify it as we did with the surge:

$$\dot{v} = \alpha \cdot F - \beta \cdot v - \delta \cdot |v| \cdot v \qquad \text{(Eq.27)}$$

It is again the same equation as equation 10, so the identification method will be applied exactly as we did with surge.

2.4 Identification of yaw parameters

Now let us do the same with the equation for yaw. If we write the equation 14 again:

$$I \cdot \dot{\omega} + D \cdot \omega + B \cdot \omega^2 + C(\omega) \cdot \omega = \tau \qquad \text{(Eq.14)}$$

And we simplify it:

$$\dot{\omega} = \alpha \cdot \tau - \beta \cdot \omega - \delta \cdot |\omega| \cdot \omega \qquad \text{(Eq.28)}$$

Where now the parameters are:

$$\alpha = \frac{1}{I} \qquad \beta = \frac{D + C(\omega)}{I} \qquad \delta = \frac{B}{I} \qquad \text{(Eq.29) (Eq.30) (Eq.31)}$$

Applying the identification method the same way and taking into account the transformations, we will obtain the parameters for yaw.

ANNEX.3 THE DYNAMIC EQUATION IN THE MATRIX FORM

3.1 The entire equation

After explaining all the physics, we know how the dynamic equation of Nessie is.

$$M \cdot \dot{v} + D \cdot v + B \cdot v \cdot |v| + C(v) \cdot v + g(\eta) = \tau$$

Let us develop it for the six degrees of freedom and see what is inside each matrix.

$$\left(\begin{pmatrix} m & 0 & 0 & 0 & 0 & 0 \\ 0 & m & 0 & 0 & 0 & 0 \\ 0 & 0 & m & 0 & 0 & 0 \\ 0 & 0 & 0 & I_X & -I_{XY} & -I_{XZ} \\ 0 & 0 & 0 & -I_{YX} & I_Y & -I_{YZ} \\ 0 & 0 & 0 & -I_{ZX} & -I_{ZY} & I_Z \end{pmatrix} + \begin{pmatrix} -X_m & 0 & 0 & 0 & 0 & 0 \\ 0 & -Y_m & 0 & 0 & 0 & 0 \\ 0 & 0 & -Z_m & 0 & 0 & 0 \\ 0 & 0 & 0 & -K_m & 0 & 0 \\ 0 & 0 & 0 & 0 & -M_m & 0 \\ 0 & 0 & 0 & 0 & 0 & -N_m \end{pmatrix} \right) \cdot \begin{pmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \\ \dot{p} \\ \dot{q} \\ \dot{r} \end{pmatrix} + \begin{pmatrix} X_D & 0 & 0 & 0 & 0 & 0 \\ 0 & Y_D & 0 & 0 & 0 & 0 \\ 0 & 0 & Z_D & 0 & 0 & 0 \\ 0 & 0 & 0 & K_D & 0 & 0 \\ 0 & 0 & 0 & 0 & M_D & 0 \\ 0 & 0 & 0 & 0 & 0 & N_D \end{pmatrix} \cdot \begin{pmatrix} u \\ v \\ w \\ p \\ q \\ r \end{pmatrix} +$$

$$\begin{pmatrix} X_B \cdot |u| & 0 & 0 & 0 & 0 & 0 \\ 0 & Y_B \cdot |v| & 0 & 0 & 0 & 0 \\ 0 & 0 & Z_B \cdot |w| & 0 & 0 & 0 \\ 0 & 0 & 0 & K_B \cdot |p| & 0 & 0 \\ 0 & 0 & 0 & 0 & M_B \cdot |q| & 0 \\ 0 & 0 & 0 & 0 & 0 & N_B \cdot |r| \end{pmatrix} \cdot \begin{pmatrix} u \\ v \\ w \\ p \\ q \\ r \end{pmatrix} + \left(\begin{pmatrix} 0 & 0 & 0 & 0 & m \cdot w & -m \cdot v \\ 0 & 0 & 0 & -mw & 0 & m \cdot u \\ 0 & 0 & 0 & -mv & m \cdot u & 0 \\ 0 & mw & -mv & 0 & -I_{YZ} \cdot q - I_{XZ} \cdot p + I_Z \cdot r & I_{YZ} \cdot r + I_{XZ} \cdot p - I_Y \cdot q \\ -mw & 0 & mu & I_{YZ} \cdot q + I_{XZ} \cdot p - I_Z \cdot r & 0 & -I_{XZ} \cdot r - I_{XY} \cdot q + I_X \cdot p \\ mv & -mu & 0 & -I_{YZ} \cdot r - I_{XZ} \cdot p + I_Y \cdot q & I_{XZ} \cdot r + I_{XY} \cdot q - I_X \cdot p & 0 \end{pmatrix} \right) +$$

$$\begin{pmatrix} 0 & 0 & 0 & 0 & -Z_m \cdot w & Y_m \cdot v \\ 0 & 0 & 0 & Z_m \cdot w & 0 & -X_m \cdot u \\ 0 & 0 & 0 & -Y_m \cdot v & X_m \cdot u & 0 \\ 0 & -Z_m \cdot w & Y_m \cdot v & 0 & -N_m \cdot r & M_m \cdot q \\ Z_m \cdot w & 0 & -X_m \cdot u & N_m \cdot r & 0 & -K_m \cdot p \\ -Y_m \cdot v & X_m \cdot u & 0 & -M_m \cdot q & K_m \cdot p & 0 \end{pmatrix} \cdot \begin{pmatrix} u \\ v \\ w \\ p \\ q \\ r \end{pmatrix} + \begin{pmatrix} -s\theta \cdot B + s\theta \cdot W \\ +c\theta s\phi \cdot B - c\theta s\phi \cdot W \\ +c\theta s\phi \cdot B - c\theta s\phi \cdot W \\ +y_b \cdot c\theta c\phi \cdot B - z_b \cdot c\theta s\phi \cdot B \\ -z_b \cdot s\theta \cdot B - x_b \cdot c\phi c\phi \cdot B \\ x_b \cdot c\theta s\phi \cdot B + y_b \cdot s\theta \cdot B \end{pmatrix} =$$

$$\begin{pmatrix} C_{T1} & C_{T2} & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & C_{T3} & C_{T4} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & -r_{X3} \cdot C_{T3} & -r_{X4} \cdot C_{T4} \\ r_{Y1} \cdot C_{T1} & r_{Y2} \cdot C_{T2} & 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} u1 \\ u2 \\ u3 \\ u4 \end{pmatrix}$$

We can see that the matrix M is the sum of two matrices. The first one contains the values of the masses and inertias of the robot, and the second one contains the values of the mass and inertia of the added mass.

The added mass is the effect of the water that goes together with the robot while it is moving. It makes a half sphere of water above the area of the robot that is in the front face of the movement.

The same happens with the C matrix. One of the matrices contains the centripetal terms due the masses and inertias of the robot, and the other contains the ones due the added masses.

3.2 The simplified equation for Nessie

Due his structure, Nessie only moves in surge, heave and yaw. So all the values from sway, roll and pitch can be simplified, and also the values that are multiplied by sway, roll and pitch velocities and accelerations.

In the gravity and buoyancy matrix, we can simplify all the components that have sine of a roll or pitch angle.

Nessie has neutral buoyancy, so the buoyancy terms can be also eliminated.

Let us write the simplified equation.

$$\left(\begin{pmatrix} m & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & m & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & I_Z \end{pmatrix} + \begin{pmatrix} -X_m & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -Z_m & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -N_m \end{pmatrix}\right) \cdot \begin{pmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \\ \dot{p} \\ \dot{q} \\ \dot{r} \end{pmatrix} + \begin{pmatrix} X_D & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & Z_D & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & N_D \end{pmatrix} \cdot \begin{pmatrix} u \\ v \\ w \\ p \\ q \\ r \end{pmatrix} +$$

$$\begin{pmatrix} X_B \cdot |u| & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & Z_B \cdot |w| & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & N_B \cdot |r| \end{pmatrix} \cdot \begin{pmatrix} u \\ v \\ w \\ p \\ q \\ r \end{pmatrix} + \left(\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}\right) \cdot \begin{pmatrix} u \\ v \\ w \\ p \\ q \\ r \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} =$$

$$\begin{pmatrix} C_{T1} & C_{T2} & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & C_{T3} & C_{T4} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ r_{Y1} \cdot C_{T1} & r_{Y2} \cdot C_{T2} & 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} u1 \\ u2 \\ u3 \\ u4 \end{pmatrix}$$

It is much simple than before…

3.3 The equation for each single degree of freedom

Surge:

$$(m - X_m) \cdot \dot{u} + X_D \cdot u + X_B \cdot |u| \cdot u = C_{T1} \cdot U_1 + C_{T2} \cdot U_2 \qquad \text{(Eq.32)}$$

Heave:

$$(m - Z_m) \cdot \dot{w} + Z_D \cdot w + Z_B \cdot |w| \cdot w = C_{T3} \cdot U_3 + C_{T4} \cdot U_4 \qquad \text{(Eq.33)}$$

Yaw:

$$(I_z - N_m) \cdot \dot{r} + N_D \cdot r + N_B \cdot |r| \cdot r = C_{T1} \cdot r_1 \cdot U_1 + C_{T2} \cdot r_2 \cdot U_2 \qquad \text{(Eq.34)}$$

ANNEX.4 IDENTIFICATION OF THE THRUSTERS

4.1 Introduction

For the identification method, we need to know the force that the thrusters give. Each thruster has an angular speed sensor. It is known that de relation between the square of the angular speed of the thuster and the force it gives is linear:

$$F = C \cdot U^2 \qquad \text{(Eq.35)}$$

Finding the C constant of each thruster, we will be able to control the force that they give. These tests are done with dynamometers. We have to ensure that the vehicle don't move to discard any drag coefficient.

As we can see in the equation below, without acceleration and speed, and neglecting the buoyancy terms, the only forces we have are the ones from the thusters:

$$M \cdot \dot{\nu} + D \cdot \nu + B \cdot \nu \cdot |\nu| + C(v) \cdot \nu + g(\eta) = \tau \qquad \text{(Eq.36)}$$

4.2 Identification results

Identification of Thrusters 1 and 2 forward:

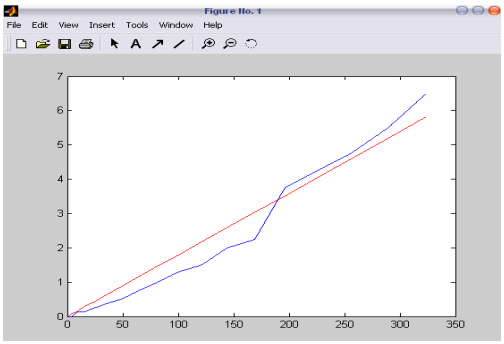


Fig.1 Square of the angular speed vs. N of force

In blue, the data from the tests

In red, the linear regression

Identification of Thrusters 1 and 2 backward:

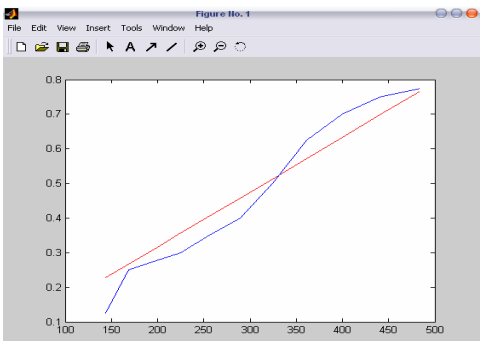


Fig.2 Square of angular speed vs. N of force

In blue, the data from the tests

In red, the linear regression

CfT1= 0.0179

CfT2= 0.0179

$$F_{T1and2\,forward} = 0.0179 \cdot U^2$$

(U in revolutions per second)

CbT1: 0.0016

CbT2: 0.0016

$$F_{T1andT2backward} = 0.0016 \cdot U^2$$

(U in revolutions per second)

Identification of Thrusters 3 and 4 up:

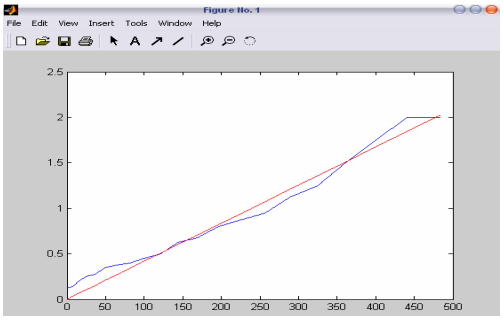


Fig.3 Square of the angular speed vs. N of force

In blue, the data from the tests

In red, the linear regression

CdT3=0.0042

CdT4=0.0042

$$F_{T3and4down} = 0.0042 \cdot U^2$$

(U in revolutions per second)

Identification of Thrusters 3 and 4 down:

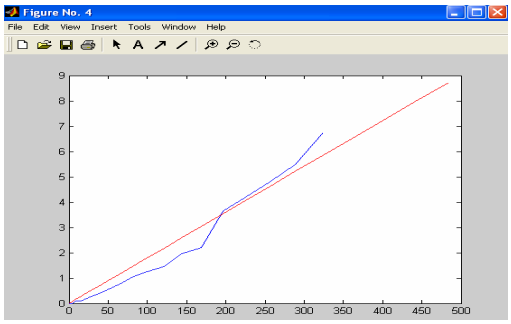


Fig.4 Square of angular speed vs. N of force

In blue, the data from the tests

In red, the linear regression

CuT3=0.0180

CuT4=0.0180

$$F_{T3and4down} = 0.0180 \cdot U^2$$

(U in revolutions per second)

To know the torques that the thrusters 1 and 2 do, we only have to multiply the force by the distance to the COG.

$r_{y1} = 22{,}5cm \quad r_{y2} = 22{,}5cm$

4.3 Table to turn the robot around the centre of gravity

To turn around the COG, the force that the two thrusters give, have to be the same but opposed.

$$0.0179 \cdot U^2 = -0.0016 \cdot U^2$$

Table of the speed on each thruster in rps

U1	U2 theoric	U2
1	3,365939	3
2	6,731878	7
3	10,09782	10
4	13,46376	13
5	16,8297	17
6	20,19563	20
7	23,56157	No possible

The one that turns backward limits the angular speed of the other motor.

Full speed turning will be 6rps on the motor that turns forward and 20rps on the motor that turns backward.

ANNEX.5 CALCULUS OF COG AND COB

Centre of Gravity

		Propeller Left	Support PL	frame	MT	Suuport MT1	Suuport MT2	BT	Support BT1	Support BT2	Propeller Front	Support PF	Propeller Back	Support PB	Sonar	Support sonar	Propeller Right	Suport PR	Altimeter	webcamF	webcamB	Drop1	Drop2	TOTAL
	MOMENT	**Distance**																						
	Weight	73	75	68	70	63	93	79	63	93	25	25	149	149	8,5	8,5	73	75	97	14	132	124	124	1681
Propeller Left	1,5	109,5																						
Support PLeft	0,4		30																					
frame	7,9			537,2																				
MT	9,22				645,4																			
Suuport MT1	0,135					8,505																		
Suuport MT2	0,135						12,555																	
BT	11,13							879,27																
Support BT1	0,11								6,93															
Support BT2	0,11									10,23														
Propeller Front	1,5										37,5													
Support PFront	0,14											3,5												
Propeller Back	1,5												223,5											
Support PBack	0,14													20,86										
Sonar	3,2														27,2									
Support sonar	0,38															3,23								
Propeller Right	1,5																109,5							
Suport PRight	0,4																	30						
Altimeter	1,1																		106,7					
webcam F	0,375																			5,25				
webcam B	0,375																				49,5			
Drop1	0,3																					37,2		
Drop2	0,3																						37,2	
																								2930,73
TOTAL	41,85																							
Distance	to COG	-2,9706	-4,9706	2,02939	0,02939	7,02939	-22,971	-8,9706	7,02939	-22,971	45,0294	45,0294	-78,971	-78,971	61,53	61,5294	-2,9706	-4,9706	-26,971	56,0294	-61,971	-54	-54	

Mass = 41,85 Kg

COG = 70,0294 cm from the front of the robot.

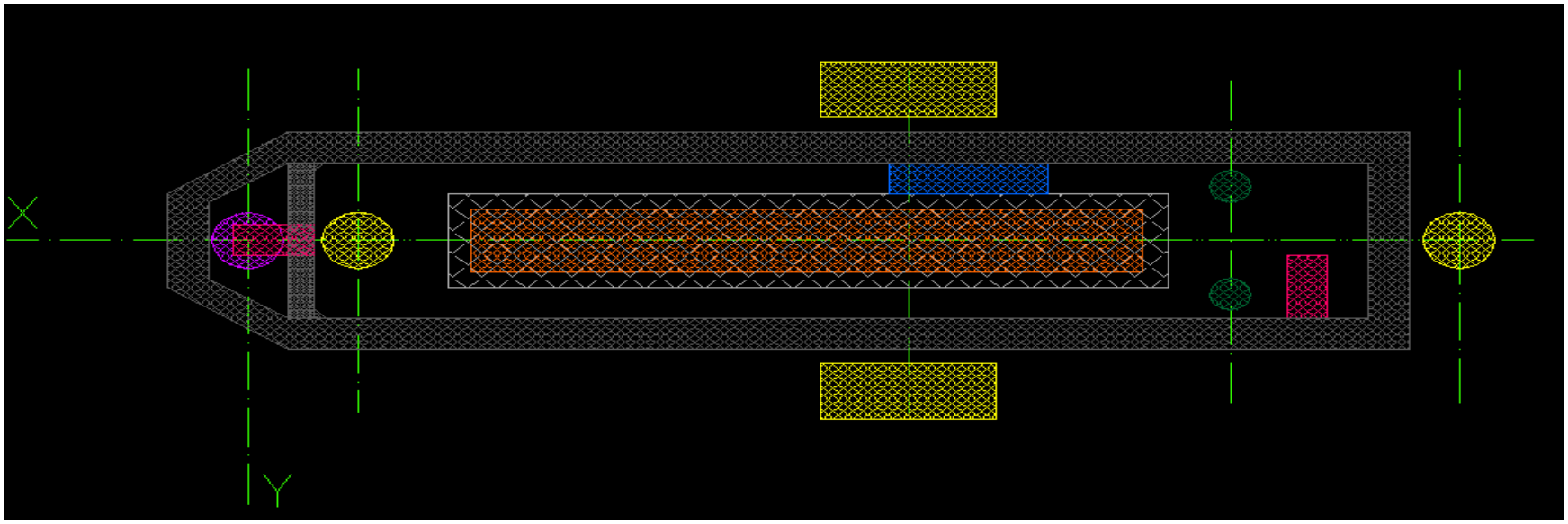
Centre of Buoyancy

		Propeller Left	Support PL	frame	MT	Suuport MT1	Suuport MT2	BT	Support BT1	Support BT2	Propeller Front	Support PF	Propeller Back	Support PB	Sonar	Support sonar	Propeller Right	Suport PR	Altimeter	webcamF	webcamB	Drop1	Drop2	TOTAL
	MOMENT Volumn (L)	**Distance**																						
		73	75	68	70	63	93	79	63	93	25	25	149	149	8,5	8,5	73	75	97	14	132	124	124	1681
Propeller L	1,4	102,2																						
Support PL	0,4		30																					
frame	1,6			108,8																				
MT	16,143				1130,01																			
Suuport MT1	0,1					6,3																		
Suuport MT2	0,1						9,3																	
BT	7,887							623,073																
Support BT1	0,15								9,45															
Support BT2	0,15									13,95														
Propeller F	1,4										35													
Support PF	0,1											2,5												
Propeller B	1,4												208,6											
Support PB	0,1													14,9										
Sonar	1,4														11,9									
Support sonar	0,325															2,7625								
Propeller R	1,4																102,2							
Suport PR	0,4																	30						
Altimeter	0,3927																		38,0919					
webcam F	0,6																			8,4				
Webcam B	0,6																				79,2			
Drop1	0,08																					9,92		
Drop2	0,08																						9,92	
																								2487
TOTAL	36,2077																							
Distance	to COB	-4,3009	-6,3009	0,69913	-1,3009	5,69913	-24,301	-10,301	5,69913	-24,301	43,6991	43,6991	-80,301	-80,301	60,2	60,1991	-4,3009	-6,3009	-28,301	54,6991	-63,301	-55,3	-55,3	

Volume = 36,2077 L

COG = 68,6991 cm from the front of the robot.

Location of all Nessie elements



In yellow, the four thrusters

In pink, the two cameras

In purple, the sonar

In orange, the battery tube

In grey, the frame

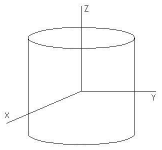
In blue, the altimeter

In light gray, the main tube

ANNEX.6 CALCULUS OF THE INERTIA I_Z

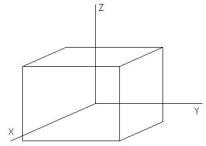
6.1 Theory

Inertia of a cylinder:



$$I_x = M\left(\frac{R^2}{4} + \frac{h^2}{12}\right) \qquad I_y = M\left(\frac{R^2}{4} + \frac{h^2}{12}\right) \qquad I_z = M\left(\frac{R^2}{2}\right)$$

Inertia of a box:



$$I_x = M\left(h^2 + b^2\right) \qquad I_y = M\left(a^2 + h^2\right) \qquad I_z = M\left(a^2 + b^2\right)$$

Steiner's theorem:

$$I_o = I + M \cdot d^2$$

6.2 Calculus of the inertia

COMPONENT	Weight (Kg)	Form	R	h	a	b	Iz	dx	Izo
Propeller 1	1,9	cilinder	0,0564	0,18			0,006640956	-2,971	16,773228
frame	7,9	box		0,01	0,5	0,32	2,78396	2,0294	35,31953
MT	9,49	cilinder	0,083	0,755			0,467138923	0,0294	0,4753365
BT	11,35	cilinder	0,056	0,83			0,660482983	-8,971	914,01577
Propeller 2	1,64	cilinder	0,0515	0,18			0,005515423	45,029	3325,345
Propeller 4	1,64	cilinder	0,0515	0,18			0,005515423	-78,97	10227,631
Sonar	3,58	cilinder	0,055	0,18			0,012373375	61,529	13553,412
Propeller 3	1,9	cilinder	0,0564	0,18			0,006640956	-2,971	16,773228
Altimeter	1,1	cilinder	0,025	0,2			0,003838542	-26,97	800,15898
webcam F	0,375	box		0,05	0,15	0,08	0,0108375	56,029	1177,2456
Webcam R	0,375	box		0,05	0,15	0,08	0,0108375	-61,97	1440,1445
Drop1	0,21	cilinder	0,0175	0,07			0,000101828	-53,97	611,6937
Drop2	0,21	cilinder	0,0175	0,07			0,000101828	-53,97	611,6937

$$I_Z = 3{,}273068\ \mathrm{Kg/m^2}$$

ANNEX.7 IDENTIFICATION OF THE PARAMETERS

7.1 Identification of surge parameters

-Taking into account the quadratic drag term:

$\alpha = 0.0294$

$\beta = 0.7058$

$\delta = 0.2222$

With the equations 17, 18 and 19 we can transform the parameters to the real values of the equation:

$$\alpha = \frac{1}{m - X_m} = 0.0294 \qquad \beta = \frac{X_D}{m - X_m} = 0.7058 \qquad \delta = \frac{X_B}{m - X_m} = 0.2222$$

$X_m = 7{,}83Kg$

$X_D = 24\frac{N}{m/s}$

$X_B = 7\frac{N}{m/s^2}$

-Neglecting the quadratic drag term:

$\alpha = 0.0093$

$\beta = 1.2549 \quad X_m = -65.67Kg$

$X_D = 135\frac{N}{m/s}$

7.2 Identification of yaw parameters

-Taking into account the quadratic drag term:

$\alpha = 0.0458$

$\beta = 0.3054$

$\delta = -1.0291$

$N_m = 20Kg$

$N_D = 6.66\frac{N}{m/s}$

$N_B = -22.46\frac{N}{m/s^2}$

-Neglecting the quadratic drag terms

$\alpha = \text{-}0.0113$

$\beta = 0.0389$

$N_m = 130Kg$

$N_D = 3.44\frac{N}{m/s}$

7.3 Identification of heave parameters

-Taking into account the quadratic drag term:

$\alpha = 0.0013$

$\beta = \text{-}0.0018$

$\delta = 0.3006$

$Z_m = 766\frac{Kg}{m^2}$

$Z_D = -1.38\frac{N}{rad/s}$

$Z_B = 230\frac{N}{rad/s^2}$

-Neglecting the quadratic drag terms

$\alpha = 0.0013$

$\beta = 0.2220$

$Z_m = 727\frac{Kg}{m^2}$

$Z_D = 170\frac{N}{rad/s}$

ANNEX.8 THE RESULTS

Since it moves at high speed in surge, I will consider the quadratic term in this degree of freedom.

For yaw and heave, I will neglect it.

Results of the identification:

$$X_m = 7.83Kg$$

$$Z_m = 130Kg$$

$$N_m = 727\frac{Kg}{m^2}$$

$$X_D = 24\frac{N}{m/s}$$

$$Z_D = 3.44\frac{N}{m/s}$$

$$N_D = 170\frac{N}{rad/s}$$

$$X_B = 7\frac{N}{m/s^2}$$

$$C_{T1}andC_{T2\,forward} = 0.0179\frac{N}{rps^2}$$

$$C_{T1}andC_{T2backward} = 0.0016\frac{N}{rps^2}$$

$$C_{T3}andC_{T4\,forward} = 0.018\frac{N}{rps^2}$$

$$C_{T3}andC_{T4backward} = 0.0042\frac{N}{rps^2}$$

The final matrix equation:

$$\left(\begin{pmatrix} 41{,}8 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 41{,}8 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 3{,}273 \end{pmatrix} + \begin{pmatrix} -7{,}83 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -130 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -727 \end{pmatrix}\right) \cdot \begin{pmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \\ \dot{p} \\ \dot{q} \\ \dot{r} \end{pmatrix} + \begin{pmatrix} 24 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 3{,}44 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 170 \end{pmatrix} \cdot \begin{pmatrix} u \\ v \\ w \\ p \\ q \\ r \end{pmatrix} +$$

$$\begin{pmatrix} 7\cdot|u| & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} u \\ v \\ w \\ p \\ q \\ r \end{pmatrix} + \left(\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}\right) \cdot \begin{pmatrix} u \\ v \\ w \\ p \\ q \\ r \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} =$$

$$\begin{pmatrix} C_{T1} & C_{T2} & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & C_{T3} & C_{T4} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0,225 \cdot C_{T1} & 0,225 \cdot C_{T2} & 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} u1 \\ u2 \\ u3 \\ u4 \end{pmatrix}$$

Where CT3=CT4=C forward or backward depending if the robot moves forward or backward.

To turn right: CT1 → forward and CT2 → backward

To turn left: CT1 → backward and CT2 → forward

The final equations for each degree of freedom:

Surge forward:

$$33,97 \cdot \dot{u} + 24 \cdot u + 7 \cdot |u| \cdot u = 0,0179 \cdot U_1 + 0,0179 \cdot U_2 \quad \text{(Eq.36)}$$

Surge backward:

$$33,97 \cdot \dot{u} + 24 \cdot u + 7 \cdot |u| \cdot u = 0,0016 \cdot U_1 + 0,0016 \cdot U_2 \quad \text{(Eq.37)}$$

Heave up:

$$-88 \cdot \dot{w} + 3,44 \cdot w = 0,0042 \cdot U_3 + 0,0042 \cdot U_4 \quad \text{(Eq.38)}$$

Heave down:

$$-88 \cdot \dot{w} + 3,44 \cdot w = 0,018 \cdot U_3 + 0,018 \cdot U_4 \quad \text{(Eq.39)}$$

Yaw right:

$$-723,7 \cdot \dot{r} + 170 \cdot r = 0,0179 \cdot 0,225 \cdot U_1 + 0,0016 \cdot 0,225 \cdot U_2 \quad \text{(Eq.40)}$$

Yaw left:

$$-723,7 \cdot \dot{r} + 170 \cdot r = 0,0016 \cdot 0,225 \cdot U_1 + 0,0179 \cdot 0,225 \cdot U_2 \quad \text{(Eq.41)}$$

ANNEX.9 CONTROL SYSTEM

9.1 Introduction

The dynamic of Nessie inside the water is a nonlinear system. It means that the principle of superposition can't be applied. The PID controllers are to control linear systems.
Actually, it is studied that most linear systems are only linear in a limited operating range. Dynamic systems can normally be treated as linear systems at low speed. Therefore, we can find an equivalent linear system in place of the nonlinear for Nessie's normal operating range to be able to calculate his control. For this, we have to see if the system operates around an equilibrium point. If it is, and the signals involved are small signals, then it is possible to approximate the nonlinear system by a linear system.

9.2 Linearization of the nonlinear Nessie's mathematic model

The real equation of Nessie's dynamic is differential, quadratic. To find the equivalent linear system is not an easy task.

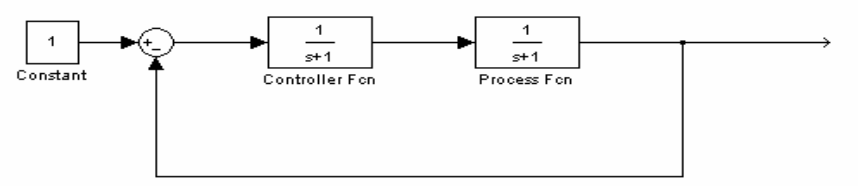
Let us assume that at low speed, the quadratic drag terms can be neglected, so I can use the simplified equations.

9.3 Control for yaw

9.3.1. Continuous control for yow theory

We can control the yaw with a PID feedback control since we can get the yaw at any time from the INS.

The feedback loop control will be something like the following, but with the corresponding Controller and Process Functions.



The equation of the system for yaw degree of freedom and neglecting the quadratic terms was:

$$(I_z - N_m) \cdot \dot{r} + N_D \cdot r = C_{T1} \cdot r_1 \cdot U_1 + C_{T2} \cdot r_2 \cdot U_2$$

To make it easier to understand let us write it like the following:

$$y(t) = A \cdot \dot{x}(t) + B \cdot \ddot{x}(t)$$

$$y(t) = C_{T1} \cdot r_1 \cdot U_1 + C_{T2} \cdot r_2 \cdot U_2 = (2 \cdot C_{T1} \cdot r_1) \cdot U_1$$

$$A = \frac{N_D}{(2 \cdot C_{T1} \cdot r_1)}$$

$$B = \frac{(I_z - N_m)}{(2 \cdot C_{T1} \cdot r_1)}$$

$$x(t) : position$$

$$y(t) : ThusterRps$$

The Laplace transformation for this equation is:

$$Y(s) = A \cdot S \cdot X(s) + B \cdot S^2 \cdot X(s)$$

The system or process transfer function P:

$$P(s) = \frac{X(s)}{Y(s)} = \frac{1}{(B \cdot S^2 + A \cdot S)}$$

Then, calling the transfer function of the PID controler as C, the transfer function of the feedback loop is:

$$H(s) = \frac{C(s) \cdot P(s)}{1 + C(s) \cdot P(s)}$$

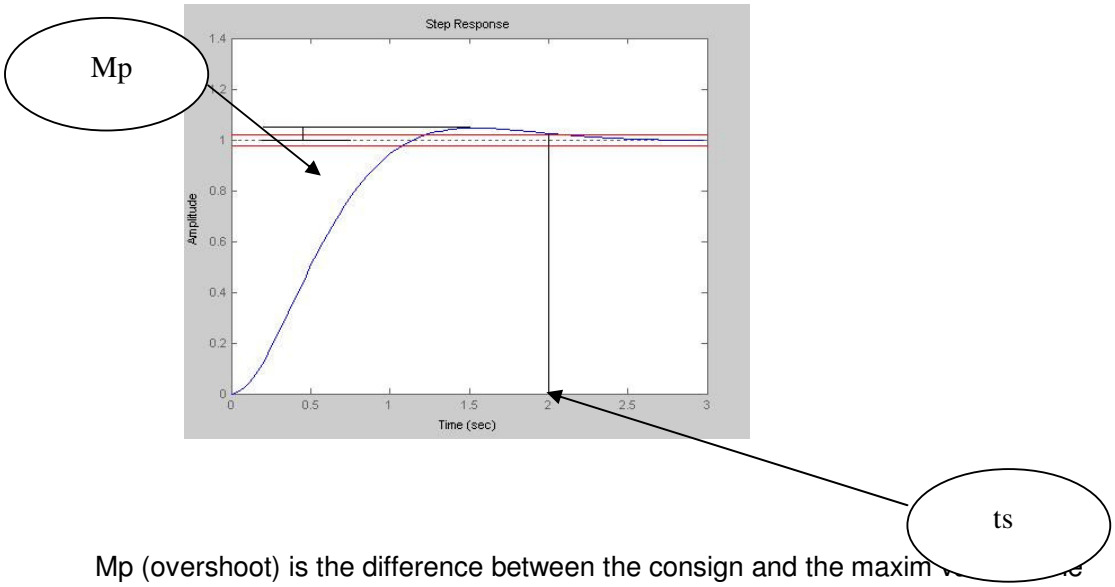
If we know witch behaviour we want our robot to have in front a consigned yaw position (H(s)), we can find the suitable controller with the direct method, where:

$$C(s) = \frac{1}{P(s)} \cdot \frac{H(s)}{1 - H(s)}$$

In annex 10 I have done a program with Matlab that returns the transfer function of the Controller by entering the Process transfer function and the desired transfer function of the loop. Where H have the following form:

$$H(s) = \frac{\omega_n^{\,2}}{S^2 + 2 \cdot \xi \cdot \omega_n \cdot S + \omega_n^{\,2}}$$

And we can model it as we want by entering witch overshoot (Mp) and settling time (ts) we desire, where:



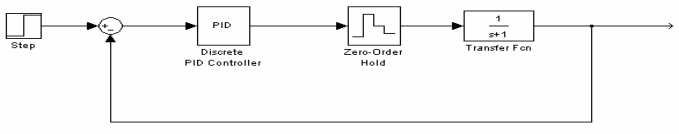
Mp (overshoot) is the difference between the consign and the maxim[illegible] output

ts (settling time) is the time that the system takes to get the consign value with a error of the 2%.

9.3.2. Discrete control for yow theory

When the system to control is a digital system, it is more efficient to use discrete PID controllers.

Now the feedback loop will be as the following:



The sampling time is 0.1 s

The D/A used is a zero order hold

Using the same method as before, the equation of the controller D will be:

$$D(z) = \frac{1}{P(z)} \cdot \frac{F(z)}{1 - F(z)}$$

$P(z)$ Is the discrete transfer function of the process

$F(z)$ Is the transfer function of the closed loop, and have to be chosen according to the behaviour we want it to have

To have zero error in front a step input, F(z) can be chosen as:

$$F(z) = \frac{1}{z}$$

9.4 Control for heave

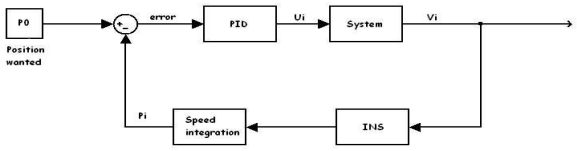
9.4.1 Control for heave theory

The same theory as for yaw (9.3.1, 9.3.2) can be applied.

9.5 Control for surge

9.5 Control for surge theory

There in no sensor to know the position in surge



To control this movement we will do the following:

1. We know the initial distance (***d***) between the robot and the target by the sonar.

It means → The integral of the velocity respect to time have to be ***d***

2. The initial velocity of the robot is zero.

3. We want the robot to arrive to the target at zero velocity.

4. If the robot moves at constant velocity, the graphic of the velocity versus time is like the following:

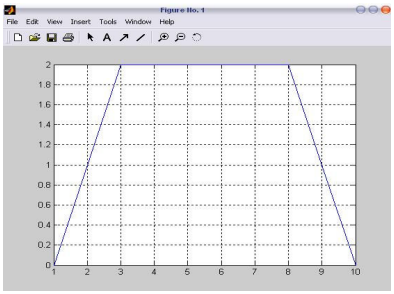


Image 1

5. To move at a wanted velocity, we need to know witch force give to the thrusters, or what is the same, how many revolutions per second.

6. The relation between the thrusters force and the velocity of the robot is not linear at all, but we can try to find an equivalent linear system for a limited operating range where the Force and the velocity have a linear dependency.

$$v = C \cdot F$$

7. If the velocity that the robot moves is significantly low and constant, we can neglect the two ramps in figure 1. Integrating the equation and making it be equal to ***d***, we will know how many time we have to apply F force to the thrusters to arrive to the target.

$$\int_0^t v(t) = \int_0^t V_{const} = d \quad \rightarrow \quad t = \frac{d}{V_{const}}$$

We will have to apply constant revolutions per second during t seconds

ANNEX.10 PROGRAMS WITH MATLAB

10.1 Program to find the thrusters constants

```
load thrusters;

Cf=regress(FfN',rps1q');
Cb=regress(FbN',rps2q');
Cu=regress(FuN',rps3q');
Cd=regress(FdN',rps1q');

x=[1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22]
y1=Cf.*x;
y2=Cb.*x;
y3=Cu.*x;
y4=Cd.*x;

fig
plot(rps1q,FfN);
hold
plot(x,y1);

fig
plot(rps2q,FbN);
hold
plot(x,y2);

fig
plot(rps3q,FuN);
hold
plot(x,y3);

fig
plot(rps1q,FdN);
hold
plot(x,y4);
```

10.2 Program to read the data for surge

```
function [time,y,u]=read_data;

% position and time;

load forward;

surgedata=forward(:,11);
T=length(surgedata);

%filter for surge, median every 3 data

R=fix(T/3);

for p=1:R;

   surgefilted(p)=(surgedata((3*p))+surgedata((3*p)-1)+surgedata((3*p)-2))/3;
   p=p+1;
```

```
end

T1=length(surgefilted);
t=[0:0.3:(T-1)/10];

% surge speed;


k=2;

for k=2:R;

   surgerate(k)=(surgefilted(k)-surgefilted(k-1))/0.6;
   k=k+1;
end

%force


pRdata=(forward(:,1))/10;  %angular speed of right thuster
pLdata=(forward(:,2)/10);  %angular speed of left thuster

CR=0.0179  %constant of propeller right going forward
CL=0.0179 %constant of propeller left going forward


f=(2*CR*(pRdata.*pRdata)); %force of the 2 thusters

%filter for force

T=length(f);
R=fix(T/3);

for p=1:R;

   forcefilted(p)=(f((3*p))+f((3*p)-1)+f((3*p)-2))/3;
   p=p+1;
end
```

10.3 Program to find the controllers for yaw, surge and heave

```
% DIRECT METHOD TO FIND THE PID CONTROLER
function[C]=set_controler;
%Desired behaviour

ts=input('Settling time (ts)=>');

xi=0.7;
wn=4/(ts*xi);
num=[wn^2];
den=[1 2*wn*xi wn^2];
H=tf(num,den);
denp=[10 23];
nump=[1];
P=tf(nump,denp);
Cc=(1/P)*(H/(1-H));

C=minreal(Cc);
```

ANNEX.11 PROGRAMS WITH C++ TO STORE DATA

```
/**STORE DATA --->RECIEVES OCEANSHELL MESSAGES FROM INS AND AUTOPILOT

*/

#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <iostream>
#include <pthread.h>
#include <fstream>


// OceanLIB include
#include <OceanLIB/Accessing/olibINIFile.h>
#include <OceanLIB/Conversion/olibScreenCoordConverter.h>

// OceanSHELL includes
#include <OceanSHELL/oshGlobals.h>
#include <OceanSHELL/oshMsg.h>
#include <OceanSHELL/oshMsg_Generic.h>
#include <OceanSHELL/oshMsgQ.h>


// Messages includes

#include "../MissionPlanning/Messages.h"
//#include "../Navigation/INSMsg.h"

using namespace std;

int32 port;  //port for OceanSHELL messages

// Motor encoders data
float speedMotorRight;
float speedMotorLeft;
float speedMotorFront;
float speedMotorBack;

//INS Data
float speedx=0.0;
float speedy=0.0;
float speedz=0.0;
float speedroll=0.0;
float speedpitch=0.0;
float speedyaw=0.0;
float altitude=0.0;
float yaw=0.0;

void *Input(void *parameter){

        oshMsgQ msgQ(port,port);

        oshMsg_Generic receivedMsg;
        DATAACQUISITIONMsg insMessage;
        AutopilotMsg autopilot;
```

```
	while(1)
	{
		cout<<"Waiting"<<endl;
		msgQ.Recv(receivedMsg);

		switch(receivedMsg.GetMsgID())
		{
			case DATAACQUISITION_MSG:
				insMessage <= receivedMsg;


				speedx=insMessage.GetvelX();
				speedy=insMessage.GetvelY();
				speedz=insMessage.GetvelZ();
				speedroll=insMessage.GetgyrX();
				speedpitch=insMessage.GetgyrY();
				speedyaw=insMessage.GetgyrZ();
				yaw=insMessage.GetYawNow();
				altitude=insMessage.GetAltitudeNow();
				break;

			case AUTOPILOT_MSG:
				autopilot <= receivedMsg;

				speedMotorRight=autopilot.GetSpeedRight();
				speedMotorLeft=autopilot.GetSpeedLeft();
				speedMotorFront=autopilot.GetSpeedFront();
				speedMotorBack=autopilot.GetSpeedBack();


				break;


			default:;

		}
	}
}


int main(int argc, char *argv[])
{

	pthread_t threaddata;

	ofstream store;
	char *returnvalue = NULL;
	int error;
	int finish=0;
	int i=0;
	int j=0;


	if (argc != 2)
	{
		cout << "Use: ./StoreData <port>" << endl;
		exit(1);
	}

	// Port number
```

```
	port = atoi(argv[1]);

	error = pthread_create (&threaddata, NULL,Input, NULL); /**MISSIONMANAGER IS
WORKING WITH WORLDMODEL*/

	if (error != 0)
	{
		perror ("We can't create thread underwebcam");
		exit (-1);
	}

	oshMsgQ msgQ(port,port);

	store.open("Data.txt");
		store << "rps MR"  << "\t";
		store << "rps ML"  << "\t";
		store << "rps MF"  << "\t";
		store << "rps MB"  << "\t";
		store << "speedx"  << "\t";
		store << "speedy"  << "\t";
		store << "speedz"  << "\t";
		store << "spRoll" << "\t";
		store << "spPitch"  << "\t";
		store << "spYaw"  << "\t";
		store << "alti"  << "\t";
		store << "yaw"  << "\t";
		store<<endl;


	while(i<100000){	/**--------------------------------------MISSION---------------------------------*/
		usleep(100000);

		store << speedMotorRight << "\t";
		store << speedMotorLeft << "\t";
		store << speedMotorFront << "\t";
		store << speedMotorBack << "\t";
		store << speedx << "\t";
		store<< speedy << "\t";
		store<< speedz << "\t";
		store<< speedroll << "\t";
		store<<speedpitch << "\t";
		store<<speedyaw << "\t";
		store<<altitude << "\t";
		store<< yaw << "\t";
		store<<endl;

		cout<<"Row stored"<<endl;

		i++;

	}

	store.close();

	return EXIT_SUCCESS;
}
```

ANNEX.12 IMAGES FROM THE TESTS DAY

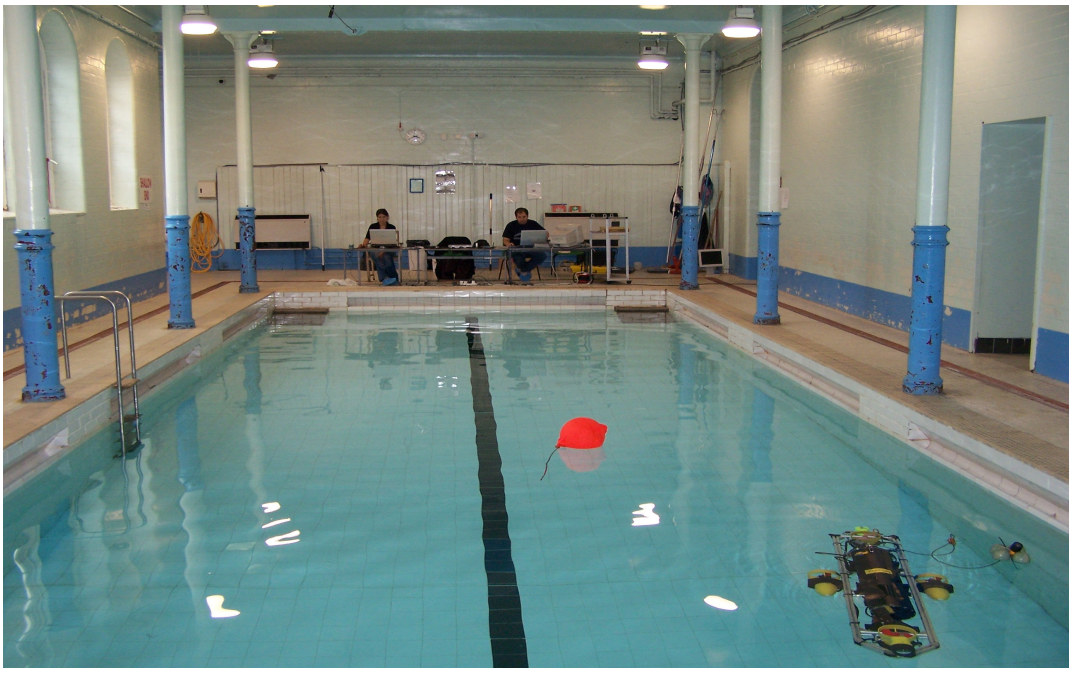


Image of the pool where we where testing the robot and doing the first missions

Image of the team running a mission in Nessie.