Protocolo de pruebas para la identificación de ASV geneseas

Gerónimo González Marino

21 de julio de 2025

1. Modelo dinámico

La ecuación dinámica que a la que se quiere aproximar los parámetros es:

$$\frac{M}{T_{|\overline{V}|\overline{V}}} * \begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{r} \end{bmatrix} + \frac{C}{T_{r_{|\overline{V}|\overline{V}}}} * \begin{bmatrix} u \\ v \\ r \end{bmatrix} + \frac{D}{T_{r_{|\overline{V}|\overline{V}}}} * \begin{bmatrix} u \\ v \\ r \end{bmatrix} = \begin{bmatrix} |\overline{V_r}|\overline{V_r} + \alpha|\overline{V_l}|\overline{V_l} \\ 0 \\ |\overline{V_r}|\overline{V_r} - \alpha|\overline{V_l}|\overline{V_l} \end{bmatrix} \tag{1}$$

Suponiendo que en las condiciones de funcionamiento del robot las variables se encuentran completamente desacopladas, las ecuaciones se simplifican en:

$$(m - X_{\dot{u}})\dot{u} - X_u u - X_{u|u|} u|u| = |\overline{V_r}|\,\overline{V_r} + \alpha|\overline{V_l}|\,\overline{V_l} \tag{2}$$

$$(m - Y_{\dot{v}})\dot{v} - Y_v v - Y_{v|v|} v|v| = 0 \tag{3}$$

$$(I_z - N_{\dot{r}})\dot{r} - N_r r - N_{rr} r|r| = |\overline{V_r}|\,\overline{V_r} - \alpha|\overline{V_l}|\,\overline{V_l} \tag{4}$$

2. Pruebas a realizar

1. En dirección de u:

   a) Escalones a velocidad constante (suponiendo $\alpha = 1$). El comando usado es:

   ```
   ros2 launch my_publisher_pkg publisher_launch.py \
   mode:=derecho \
   thrust:=1.0 \
   alpha:=1.0
   ```

2. En dirección de ψ:

   a) Escalones a velocidad angular constante. El comando usado es:

```
ros2 launch my_publisher_pkg curva_launch.py \
mode:=curva \
left_thrust:=0.5 \
right_thrust:=-0.5 \
alpha:=1.0
```

3. Parámetros acoplados

   a) A velocidades constantes. Curvas a velocidad angula y lineal constantes. El comando usado es:

```
ros2 launch my_publisher_pkg curva_launch.py \
mode:=curva \
left_thrust:=0.5 \
right_thrust:=0.4 \
alpha:=1.0
```

   b) Zig-Zag. Onda cuadrada con velocidad lineal. El comando usado es:

```
ros2 launch my_publisher_pkg zigzag_wave_launch.py \
valor_alto:=1.0 \
valor_bajo:=0.50 \
periodo:=5.0 \
alpha:=1.0
```

4. Diferencia de empuje entere motores. El comando usado es:

```
ros2 launch my_publisher_pkg pi_control_launch.py \
kp:=1.0 ki:=0.07 kd:=0.0 \
right_thrust:=0.6 \
output_limits:="[-1.0, 1.0]" \
sample_time:=0.1 \
realimentacion:=1.0 \
offset_motor:=0.6
```

5. Señal PRBS con modulación de amplitud:

```
ros2 launch my_publisher_pkg prbs_launch.py \
amplitudes:="[0.4,0.5,0.6]" \
switch_interval:=4.0
```

6. Por si sobra tiempo:

   a) Onda cuadrada en u (suponiendo $\alpha = 1$). El comando usado es:

```
ros2 launch my_publisher_pkg square_wave_launch.py \
valor_alto:=0.50 \
valor_bajo:=-0.50 \
periodo:=5.0 \
alpha:=0.5
```

b) Onda cuadrada en ψ (suponiendo $\alpha = 1$). El comando usado es:

```
ros2 launch my_publisher_pkg zigzag_wave_launch.py \
valor_alto:=0.50 \
valor_bajo:=-0.50 \
periodo:=5.0 \
alpha:=0.5
```

2.1. Objetivo de las pruebas

1*a* Escalones a velocidad constante. De esta prueba se podrían obtener la parte parámetros de la dinámica y del estado estacionario. En el estado estacionario se podrían aproximar los valores de Dl y Dnl a partir de al menos dos datos de velocidad. En la dinámica del escalón se podría estimar la masa y masa aumentada.

2*a* Escalones a velocidad constante. De esta prueba se podrían obtener la parte parámetros de la dinámica y del estado estacionario. En el estado estacionario se podrían aproximar los valores de Dl y Dnl a partir de al menos dos datos de velocidad. En la dinámica del escalón se podría estimar la inercia e inercia aumentada.

3*a* Velocidad constante u y ψ. Determinar el acoplamiento del damping o coriolis

3*b* Zig-Zag con velocidad en u. Seria representativo para una validación

4 Diferencia de empuje de motores. Determinar el coeficiente α de empuje entre motores.

6*a* Onda cuadrada. De esta prueba interesaría principalmente la dinámica frente al escalón, muy similar a la anterior.

6*b*Onda cuadrada. De esta prueba interesaría principalmente la dinámica frente al escalón, muy similar a la anterior.

3. Útiles pruebas

3.1. Rutas

El nodo que levanta todos los sensores se ejecuta corriendo

```
cd ros2_ws
source install/setup.bash
ros2 launch base telemetria.launch.py
```

(Puede haber errores de tipeo)

Los launchs de las pruebas de este informe están en el espacio de trabajo $prueba_ws$

```
    cd prueba_ws
    source install/setup.bash
```

3.2. Abrir arduino a por ssh

```
export DISPLAY=$DISPLAY
export XAUTHORITY=$HOME/.Xauthority
```

3.3. Abrir WinSCP

```
wine ~/.wine/drive_c/Program\ Files\ \(x86\)/WinSCP/WinSCP.exe
```

3.4. Publicar en un tópico

RECORDAR QUE HAY QUE PUBLICAR EN LOS 4 MOTORES AL MISMO TIEMPO

```
ros2 topic pub /geneseas/motor_c std_msgs/msg/Float32 "data: 0.5"
```

3.5. Exportar roscore

NO PUDE HACERLO FUNCIONAR CON LA PC DEL ROBOT

```
source /opt/ros/jazzy/setup.bash
export ROS_DOMAIN_ID=0
export RMW_IMPLEMENTATION=rmw_fastrtps_cpp
```

4. Bitácora de cambios y revisiones

1. revisión de motores
   motor L al revés.
   motor c al revés.
   motor R bien
   Motor D no se el sentido

2. triggers:
   Motor C con publicando 0.05 no se mueve
   Mootr D esta bien
   Motor L publicando 0.06 no se mueve
   Motor R publicando -0.05 no se mueve

3. Utrasonidos
   ultrasonido 2 no anda (aveces tira mediciones)

MODIFICACIONES CÓDIGO ARDUINO (MOTORES) - Lineas modificadas para el ajuste del 0 de la PWM de los motores

COMIENZO DEL CÓDIGO

// Offsets individuales para corregir el centro float offsetL = 0.06; float offsetR = -0.05; float offsetC = 0.05; float offsetD = 0.0

En la parte del automatico

```
else if(automatico) //automatic control
{
 data1_ch=motorValues[0] - offsetL;
 data2_ch=motorValues[1] - offsetR;
 data3_ch=motorValues[2] - offsetC;
 data4_ch=motorValues[3] - offsetD;
```