- **MOTION_HANDLER.PDL**

We inserted the priority to the motion handler PRIORITY=4.
We commented and not used the semaphore signal WAIT sem_exec.

When the sensor tracking is enabled we added the features

```
$SENSOR_TRK(TRUE, 1)
$SENSOR_TYPE := 10  --10 is for the absolute mode, 6 is for the relative one
$SENSOR_ENBL := TRUE
$SENSOR_CNVRSN[1] := 1   -- [bit/mm]
$SENSOR_CNVRSN[2] := 1   -- [bit/mm]
$SENSOR_CNVRSN[3] := 1   -- [bit/mm]
$SENSOR_CNVRSN[4] := 1   -- [bit/deg]
$SENSOR_CNVRSN[5] := 1   -- [bit/deg]
$SENSOR_CNVRSN[6] := 1   -- [bit/deg]
--
$SENSOR_GAIN[1] := 100   -- fattore di guadagno X [percentuale]
$SENSOR_GAIN[2] := 100   -- fattore di guadagno Y [percentuale]
$SENSOR_GAIN[3] := 100   -- fattore di guadagno Z [percentuale]
$SENSOR_GAIN[4] := 100
$SENSOR_GAIN[5] := 100
$SENSOR_GAIN[6] := 100
--
$SENSOR_OFST_LIM[1] := 500  -- max linear translation limit [mm]
$SENSOR_OFST_LIM[2] := 800   -- max rotation [deg]
vb_first_sns_trk := FALSE
```

We commented

```
--SENSOR_SET_DATA(sar_cartesian_correction, 1)
--SENSOR_GET_OFST(sar_ofst, 1)
--SIGNAL sem_exec
```

In order to reduce data traffic to avoid the undesired delay during sensor tracking control modality.

Very important is the insert of DELAY 1 (1ms) in order to guide the mainloop at 1ms and not as fast as it can.

- **MOTION_SERVER.PDL**

We inserted the priority to the motion server PRIORITY=5.

We read the message type and its ID with one single READ command. READ vi_server_netlun (t_msg.id::4, t_msg.message_type::1). This is in order to avoid errors.
We commented the semaphore --WAIT sem_exec as we have done before for the motion handler.
We added a checkpoint in order to be sure that the message is new and we need to read it, IF t_msg.id> prev_t_msg_id THEN
READ
vi_server_netlun(t_msg.sensor_tracking_correction[1]::4,t_msg.sensor_tracking_correction[2]::4,t_msg.sensor_tracking_correction[3]::4,t_msg.sensor_tracking_correction[4]::4,t_msg.sensor_tracking_correction[5]::4, t_msg.sensor_tracking_correction[6]::4) in a single shot.
We update here the sensor correction data with SENSOR_SET_DATA(sar_cartesian_correction, 1).

- **CONTROLLERS.yaml**

We had some difficulties changing the parameter ee_vel_limit online, so, in order to use the absolute modality we changed the parameter to 5.0. In this way we have no undesired saturations.
We changed the state_publish_rate to 50 Hz (controller for sensor tracking).

- **STATE_SERVER.PDL**

We changed the loop frequency to 50 Hz instead of 25Hz.
In order to give the possibility to design a control, we have in feedback the real cartesian and joint positions. With:
rr_ARM_rPOS() for the cartesian ones and t_jntp for the joint ones.

- **SENSOR_TRACKING_CONTROLLER.CPP**

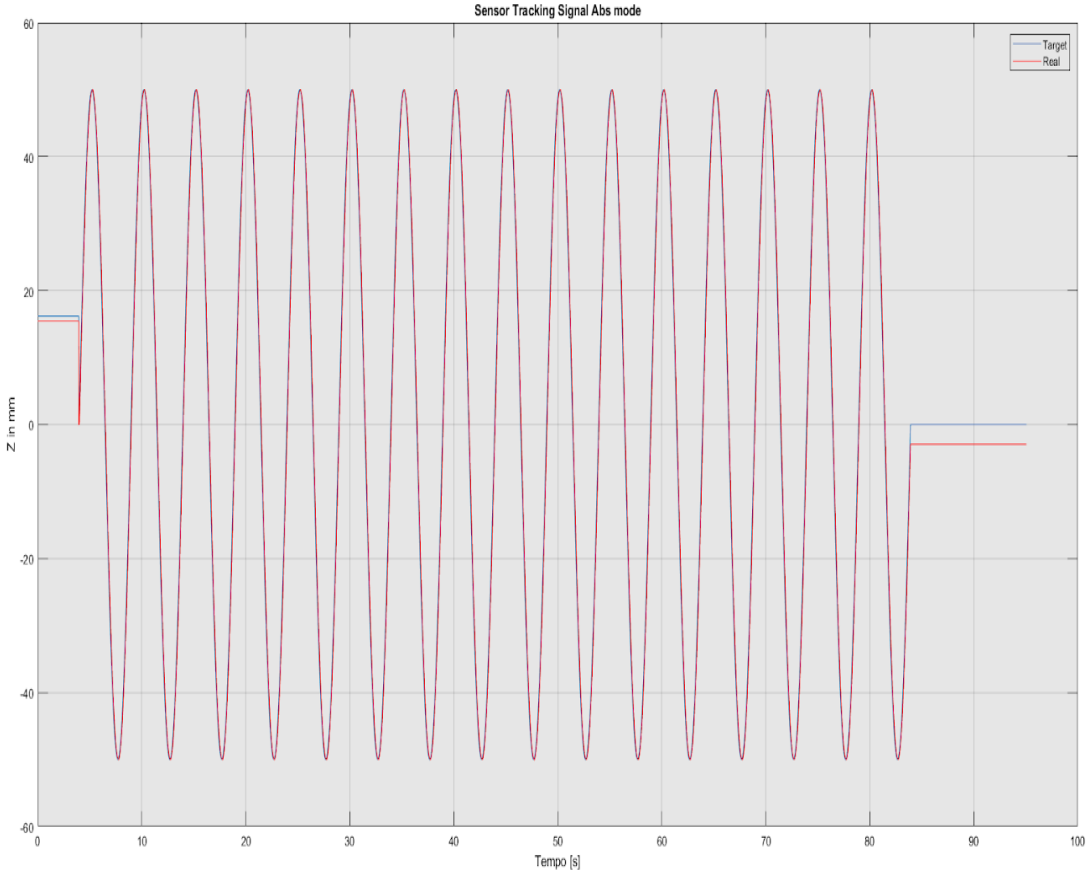
We modified the sensor tracking controller (ROS side) in order to receive in input the **arm_cmd_vel** that are **positions** for the absolute modality of the sensor tracking type **10** or **arm_cmd_vel** are **velocities** for the relative modality (**6**). The second input is the sensor tracking type (sns_trk_type).

- **TWIST_SIN_PUB.CPP**

This node simulates the commands of the sensor tracking. The parameters that must be selected are the axes according to the chosen movement, the sensor tracking modality, the amplitude in meters and the start button to start the movement.
During the absolute modality the user must pay attention to initial and final position because at end of the correction the robot returns to the original path in 2 seconds (if there is no path, it returns to the original position, the explanation is in COMAU motion manual).

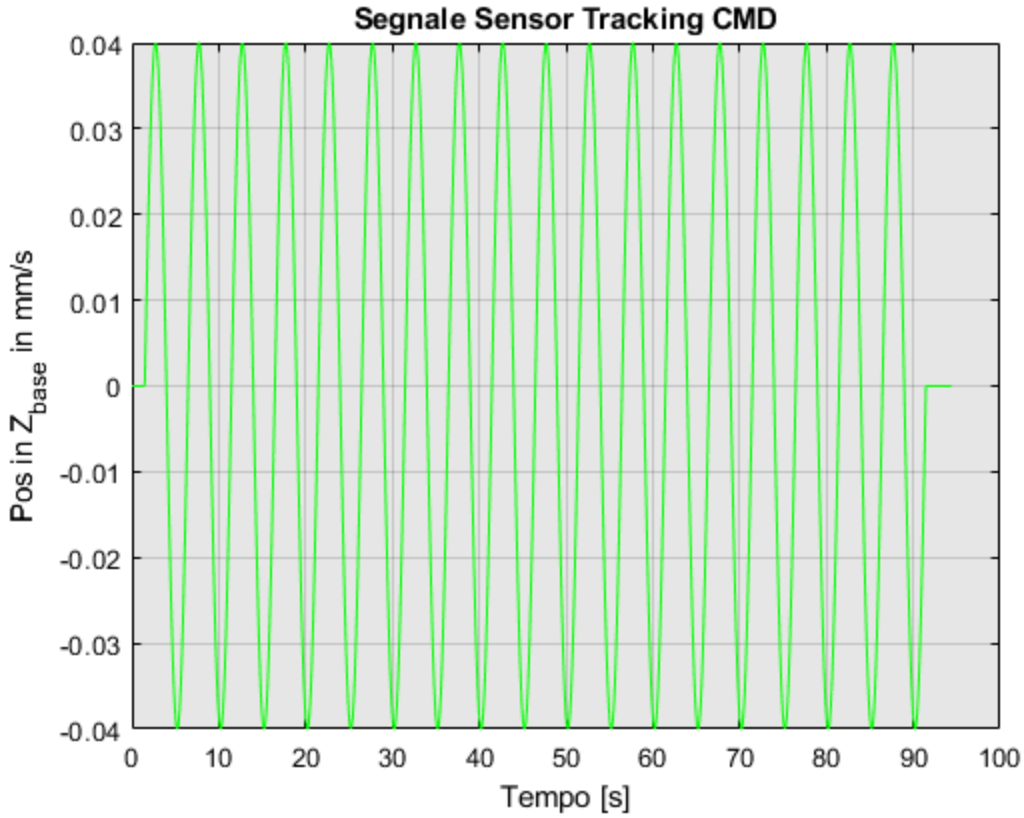
- **GENERAL CONSIDERATIONS**

We suggest you to use the absolute modality ($SENSOR_TYPE := 10 in PDL motion handler.PDL and in sns_trk_type topic ROS) because you can control the robot using positions directly and the actuation have low undesired effects due to leak of



communication. In fact in this figure we can see that the real cartesian movement, along z axis is very close to the target one.

If we consider the relative modality, due to the leak of communication, we have an error in the target position that affects the correct



execution of the movement.

In this figure we can see the commanded velocities along z axis, but at the end of the communication the movement becomes this one.