Simen Bootsma
12-03-2019

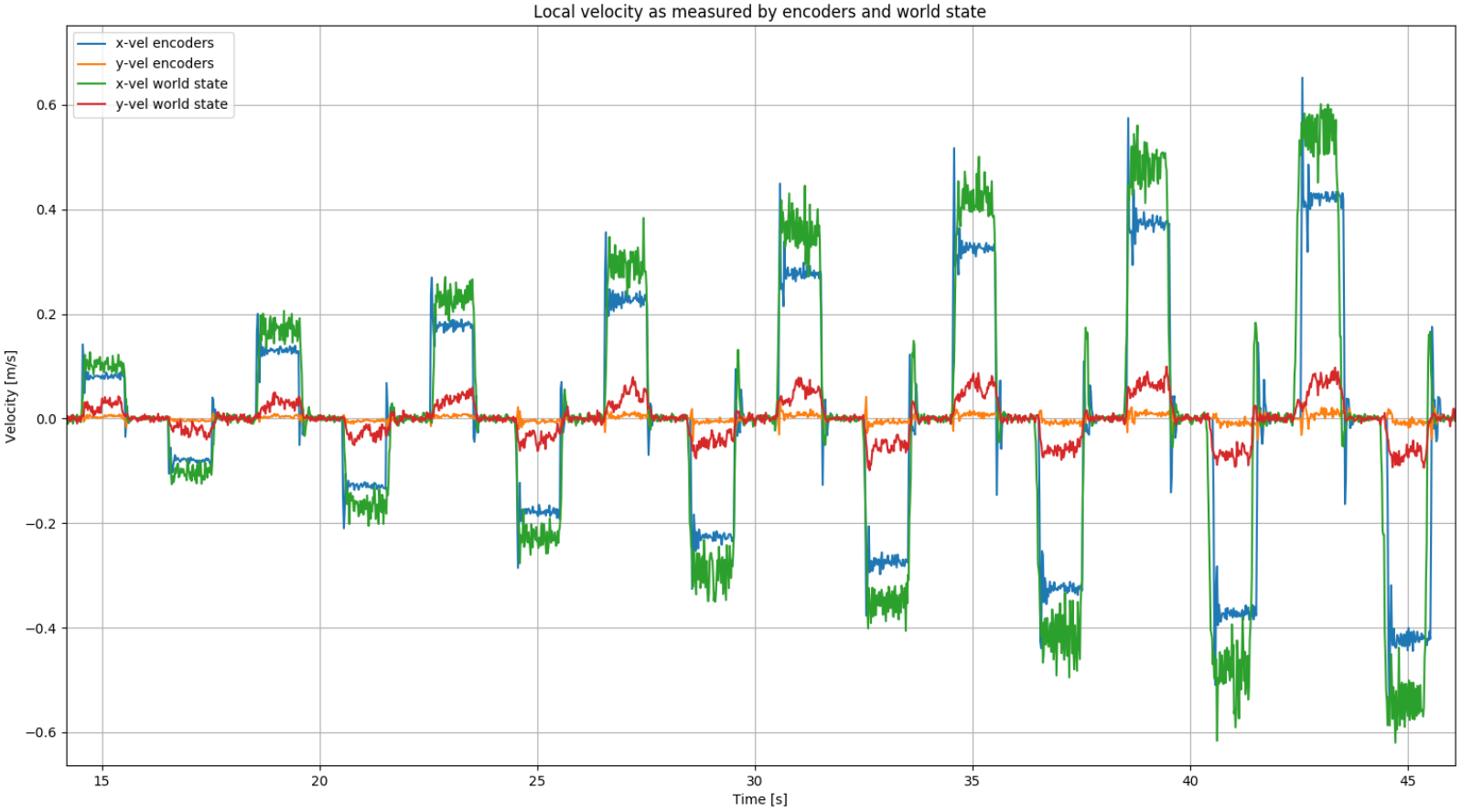
Velocity measurement

World State vs. Wheel Encoders

Update:
The problem described here has been solved. The transformation matrices *body2Wheels* and *wheels2Body* were used as if the velocities were forces acting on the ground. The correct matrices have been given in the Conventions file on drive.

Introduction
The way we do velocity measurement currently, is different on the PC and on the robots. On the PC, the velocities we work with are calculated in the world state and they are based on the positions of the robots given by ssl-vision. However, on the robot we determine the velocity using the data from the wheel encoders and a Kalman Filter. If you know how fast the wheels are turning, you can determine the velocity of the robots in both directions. Work is being done on incorporating the acceleration given by the XSens in this filter, but at the moment that is not accurate enough yet.

I let the robot drive forwards and backwards at increasing speeds and recorded the local velocity on the robot and the global velocity on the PC simultaneously. I put the results in the same graph, shown below.

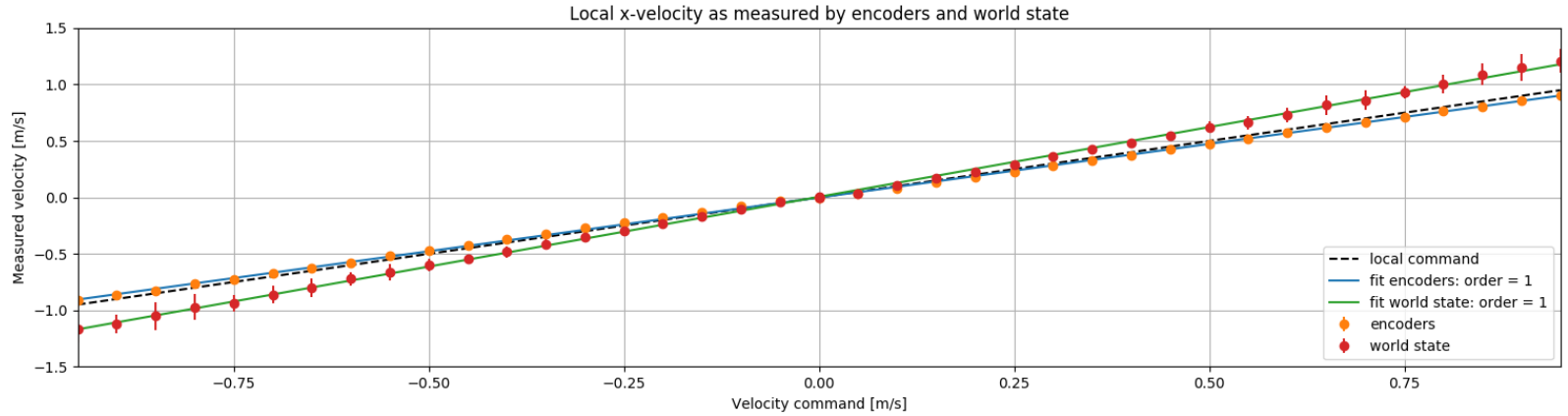


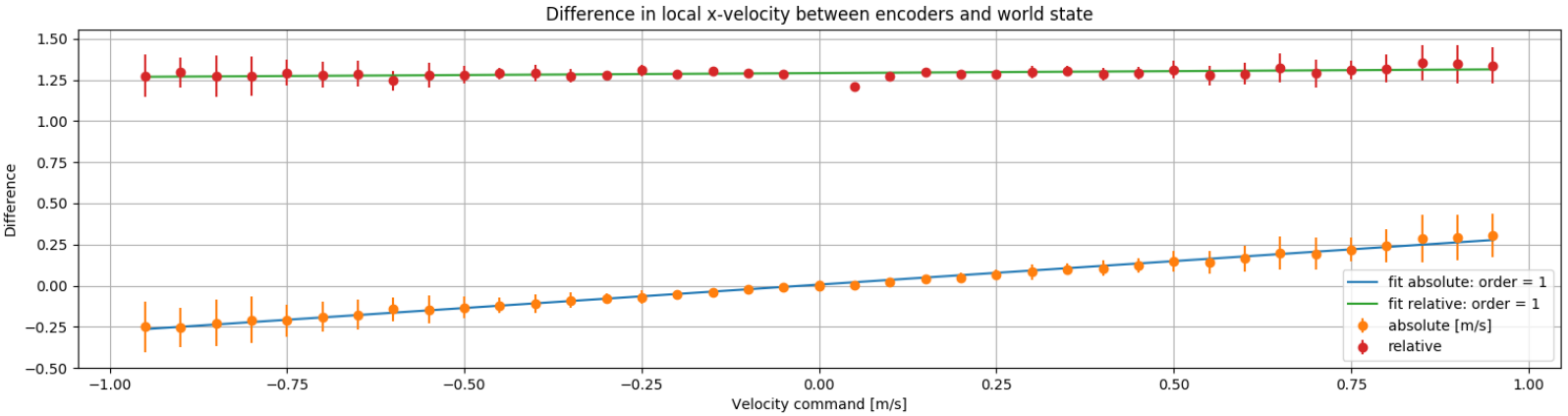
Analysis

It is clear that the world state measures a higher velocity than the encoders for all speeds. Now the question becomes: which one of the two is right? To determine this I taped a measuring unit on the field and let the robot drive for 2 seconds at 0.3 m/s. According to the robot, it reached a velocity of exactly 0.3 m/s and should therefore have moved 60 cm forward. However, I measured that the robot had moved 70 cm in 2 seconds, resulting in a velocity of 0.35 m/s. This is exactly the world state measured it to be. Therefore, the world state is right.

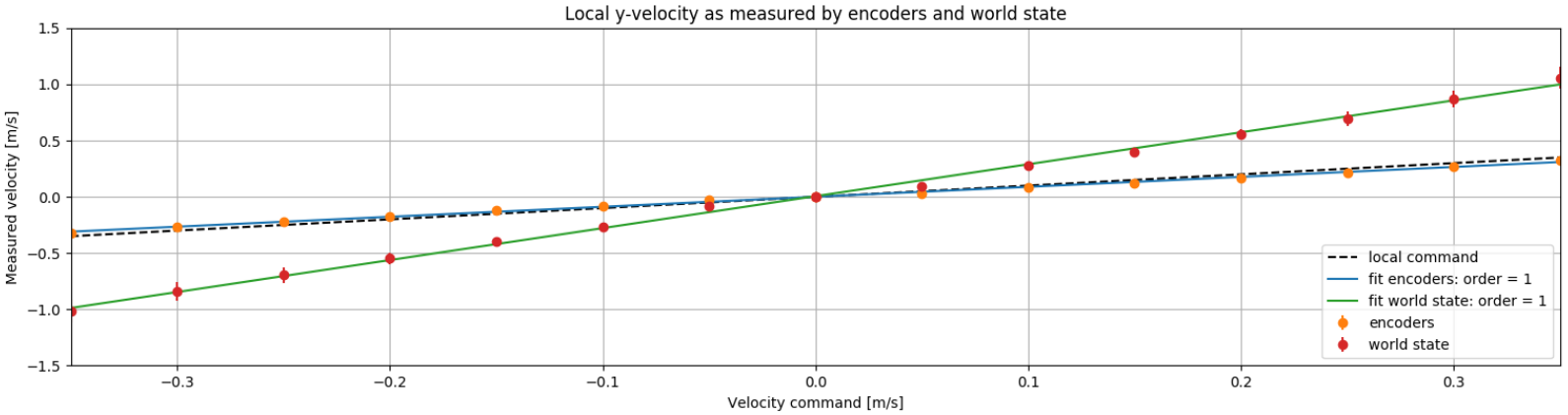
To able to account for it, we need to know how the difference behaves. Whether it is a constant difference or it changes with increasing velocity. In order to analyze this, I let the robot drive back and forth with increasing velocity again. At the times of constant velocity, I took the average and the standard deviation. That results in a data point for every given velocity command with a certain standard deviation for both the world state and the wheel encoders. The result is given in the graph below.

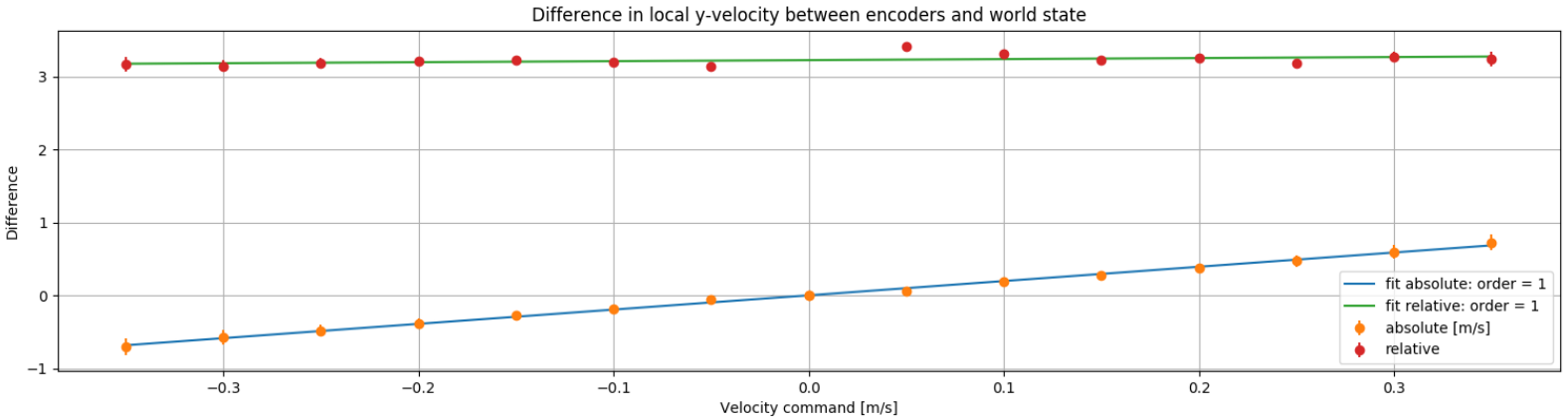
In this figure, the black dashed line (local command) represents the line y = x, so that is the velocity the robot is controlled at. From the top graph becomes clear that the difference progresses linearly and from the bottom graph becomes clear that the difference is relative,





therefore there is a constant factor between the world state velocity and the wheel encoders velocity. This factor is equal to 1.29 ± 0.03.
To investigate this behaviour a little more, I tried the same thing for moving in its local y-direction (sideways). The result is given below.





It becomes clear that the same behaviour can be observed in the sideways direction as in the forwards direction. However, the factor of difference is a lot higher. It is 3.23 ± 0.05 instead of 1.29 ± 0.03.

After that result, I also decided to do the experiment in the intermediate direction: at an angle of 45 degrees. I looked at the results for the x-direction, y-direction and the absolute velocity. Since they are a lot of graphs, I put them in an appendix. In the local velocity graph with the raw data can be seen that the velocity is never constant for velocities below 0.1 m/s, therefore I decided not to include them in the analysis. The same behaviour was observed again and the factors also were different this time. They have been shown in the table below.

Velocity type	x-direction	y-direction	Absolute ($\sqrt{{v_x}^2 + {v_y}^2}$)
Factor	1.497 +- 0.001	3.248 +- 0.012	2.395 +- 0.008

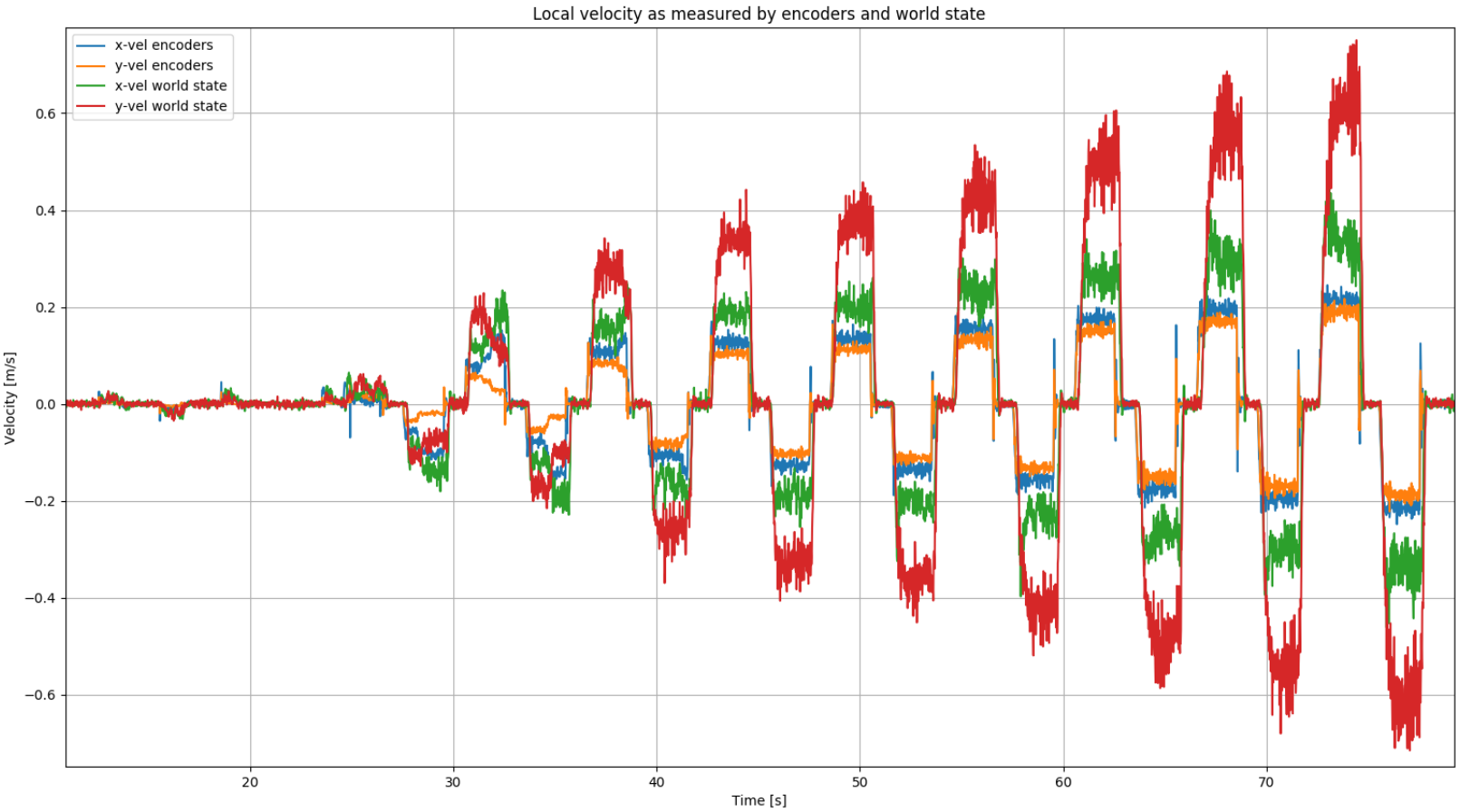
Conclusion

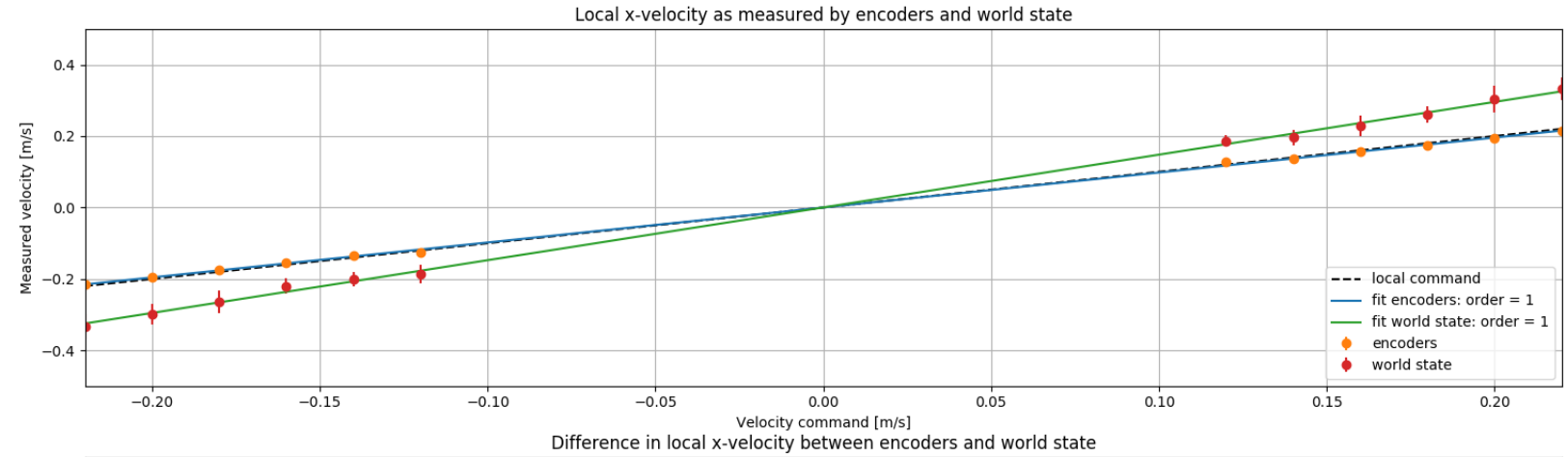
The velocity as measured by the wheel encoders is a certain factor too low as compared to the world state. This factor is different in the local x- and y-velocity, but seems to stay the same for moving in an arbitrary direction. The values are approximately 1.3 for the x-direction and 3.2 for the y-direction.

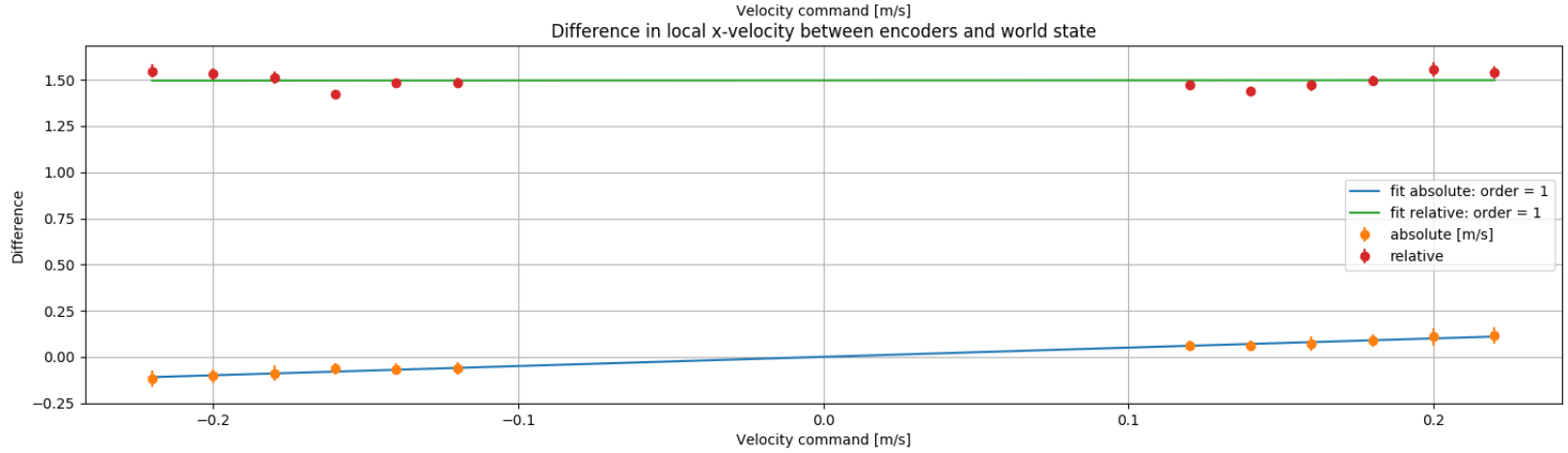
Since the world state is measured to be correct, there must be a miscalculation or mistake in the measurement somewhere in the robot code. It would be very nice if we could point out where it goes wrong, but at the moment I cannot find it. Therefore I have to say with pain in my heart, it might be best to just add a “magical constant” that compensates for this difference. If there is time left to find out why there is a difference, please do so.

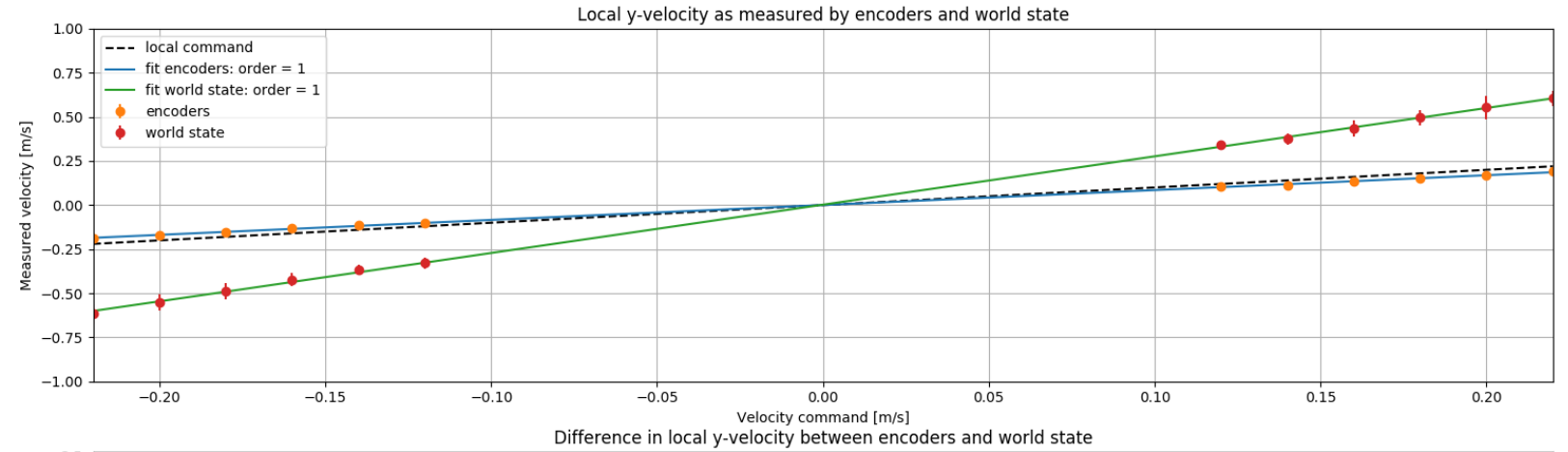
Appendix A

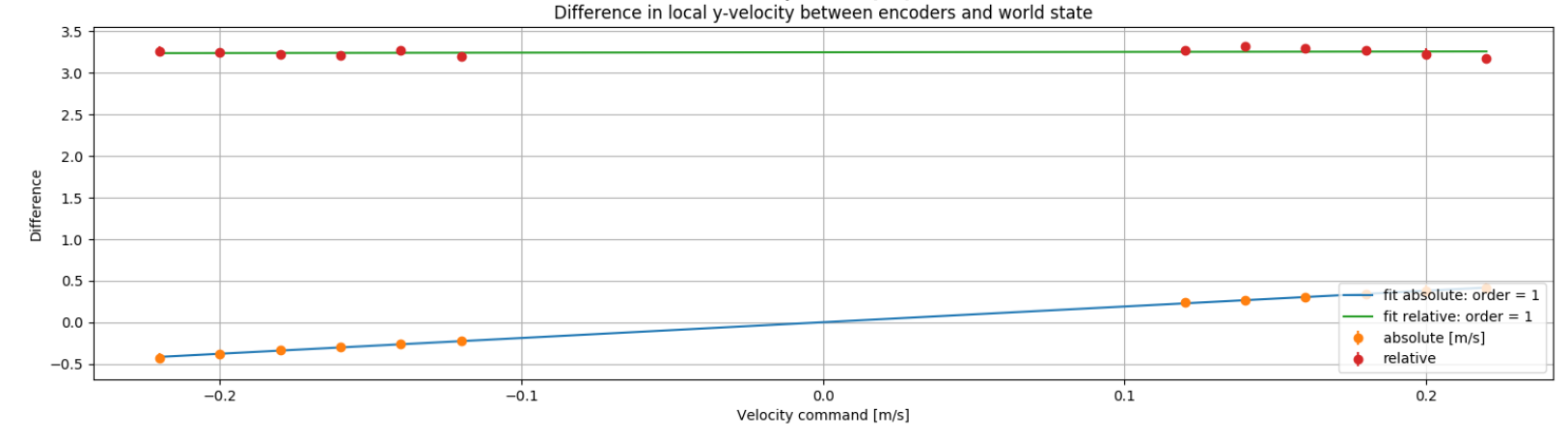
Results of movement in the 45 degree direction

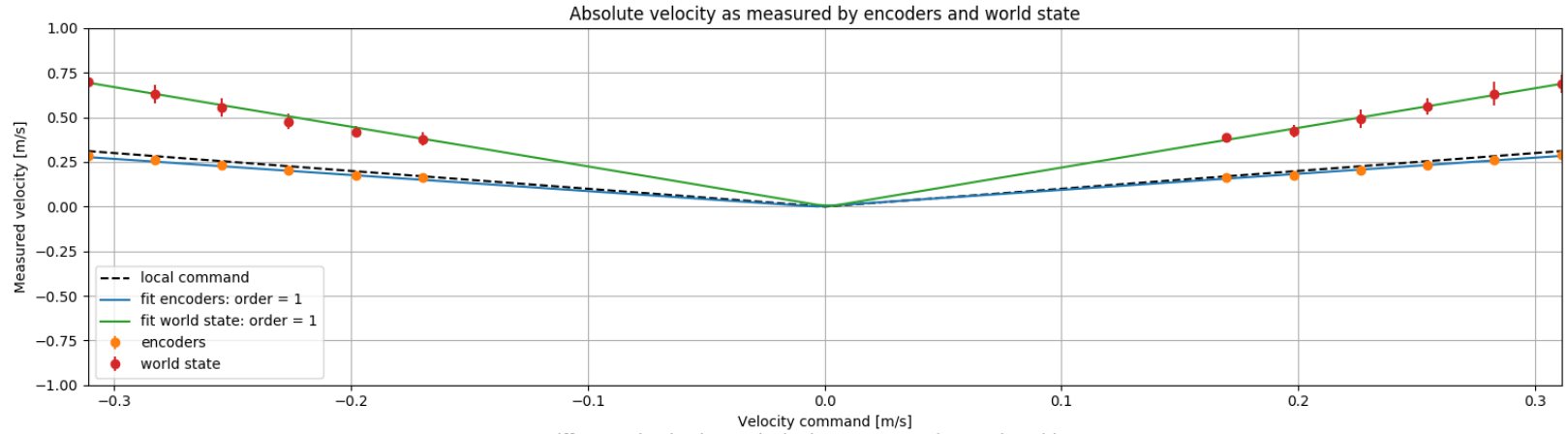


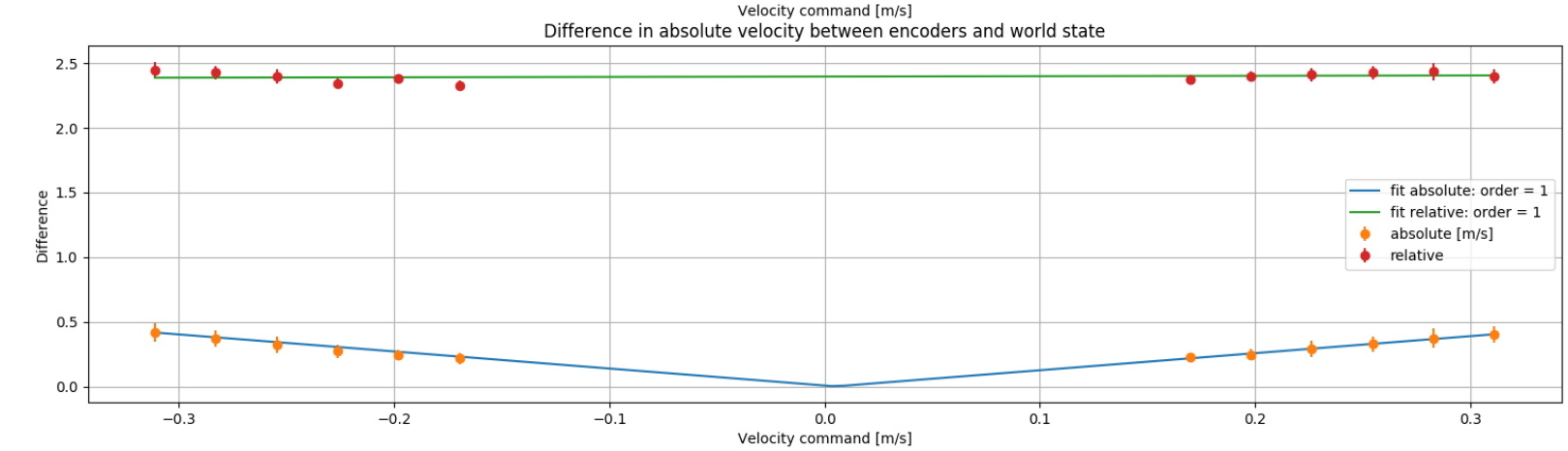

Local x-velocity as measured by encoders and world state
Measured velocity [m/s]
Velocity command [m/s]
local command
fit encoders: order = 1
fit world state: order = 1
encoders
world state


Difference in local x-velocity between encoders and world state
Difference
Velocity command [m/s]
fit absolute: order = 1
fit relative: order = 1
absolute [m/s]
relative


Local y-velocity as measured by encoders and world state
Measured velocity [m/s]
Velocity command [m/s]
local command
fit encoders: order = 1
fit world state: order = 1
encoders
world state


Difference in local y-velocity between encoders and world state
Difference
Velocity command [m/s]
fit absolute: order = 1
fit relative: order = 1
absolute [m/s]
relative


Absolute velocity as measured by encoders and world state
Measured velocity [m/s]
Velocity command [m/s]
local command
fit encoders: order = 1
fit world state: order = 1
encoders
world state


Difference in absolute velocity between encoders and world state
Difference
Velocity command [m/s]
fit absolute: order = 1
fit relative: order = 1
absolute [m/s]
relative