**USE**

**Parameters**: dt=1, Length=200, Kp=0.8, Ki=0.0, Kd=0.02

**Basic: (same)**

get\_error(target\_distance:Real) : Real =

self.robot.get\_distance(self.robot.target) - target\_distance

**Robust: (same)**

get\_error(target\_distance:Real) : Real =

self.robot.get\_distance(self.robot.target) - target\_distance

- self.robot.sensor\_accuracy \* 40

**Probabilistic**

get\_error(target\_distance:Real) : Real =

let distance:Real = self.robot.get\_distance(self.robot.target) in

distance - target\_distance

- self.robot.sensor\_accuracy \* tolerance(self.confidence)

**Uncertain**

get\_error(target\_distance:UReal) : UReal =

let distance:Real = self.robot.get\_distance(self.robot.target) in

UReal(distance,

self.robot.sensor\_accuracy \*tolerance(self.confidence))

- target\_distance

- self.robot.sensor\_accuracy \* tolerance(self.confidence)

**URobot (similar to the one in Python)**

get\_error(target\_distance:UReal) : UReal =

let distance:Real = self.robot.get\_distance(self.robot.target) in

UReal(distance, self.robot.sensor\_accuracy)

- target\_distance

- self.robot.sensor\_accuracy

(in this latter case, the tolerance() is assumed to be 1.0, and thus it can be removed from all expressions)

**To execute the system:**

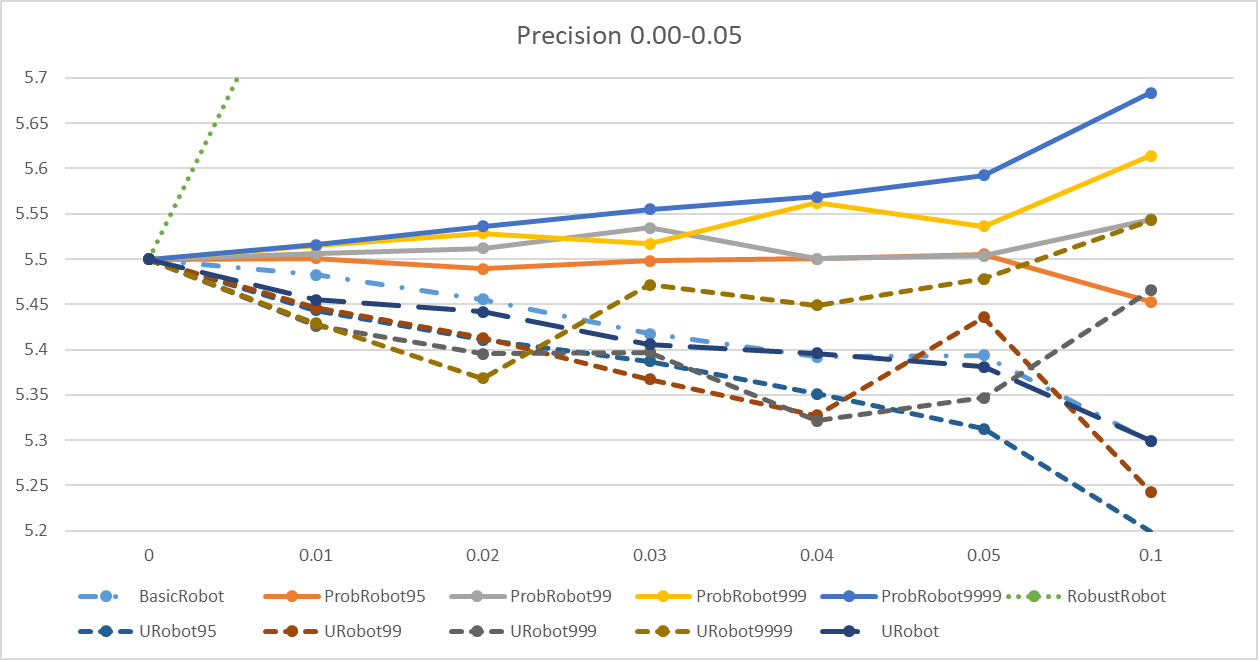
1) Launch USE

2) Using the graphical interface: File->Open specificacion and select the file ChasingRobot.use

3) On the command interface, input “open ChasingRobot.soil”

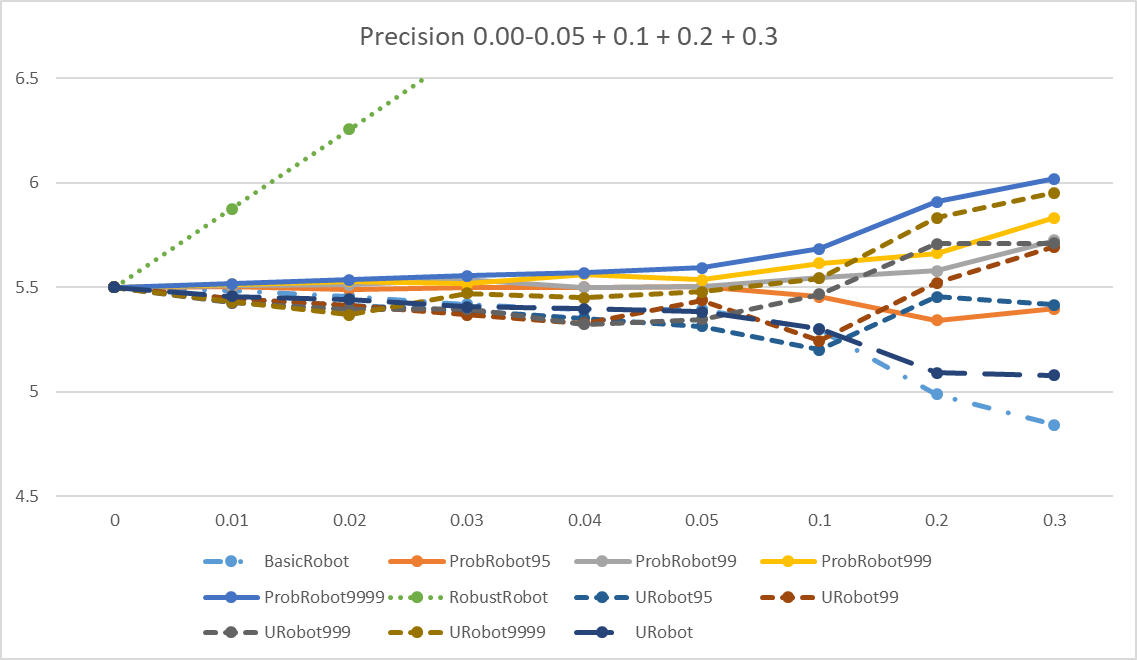
RESULTS

**Precisions from 0.0 to 0.05**



(Robust goes up to 7.4)

**Precisions from 0.0 to 0.05 plus 1.0, 2.0 and 3.0**



(Robust goes up to 11)

**Python**

In probabilistic controller

def get\_error(self, target\_distance: float) -> ufloat:

distance = self.robot.get\_distance(self.target)

return ufloat(distance, abs(Mobile.sensor\_accuracy))

In Stochastic controller:

def get\_error(self, target\_distance: float) -> float:

distance = self.robot.get\_distance(self.target)

udistance = ufloat(distance,self.robot.sensor\_accuracy)

return (udistance - target\_distance).nominal\_value - self.robot.sensor\_accuracy

The results are interesting:

**Executions:**

averages, mins = multiple\_run(10, 30, 0.2)

with parameters

rc = controller(robot, 1.59, 0.75, 0.51)

Results:

