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
```

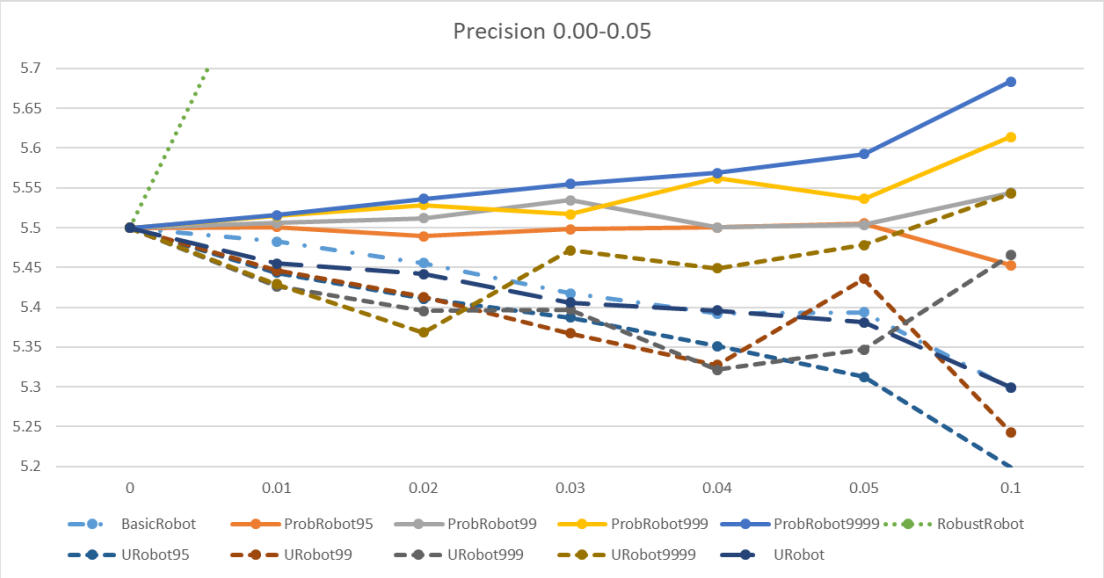
(in this latter case, the tolerance() is assumed to be 1.0, and thus it can be removed from all expressions)

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”

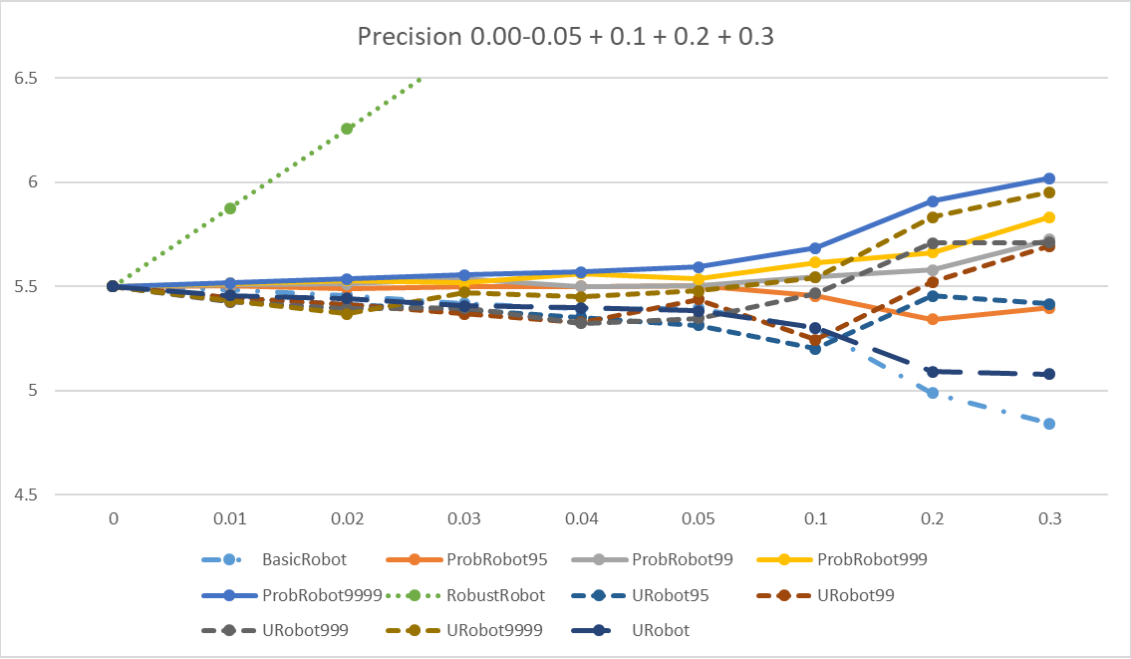
RESULTS

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: