Oprogramowanie Naukowo-Techniczne sp. z o.o.

MATLAB EXPO

Warsaw, 04.06.2024 r.

Developing vehicle autonomy with different control strategies

Paweł Siatka, Junior Application Engineer, ONT

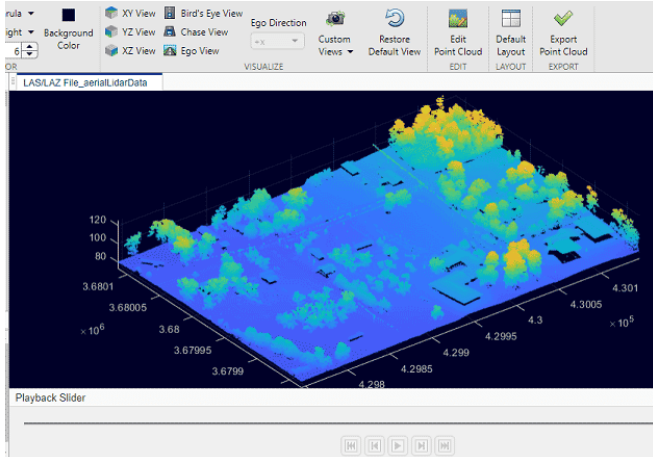
MathWorks Authorized Reseller

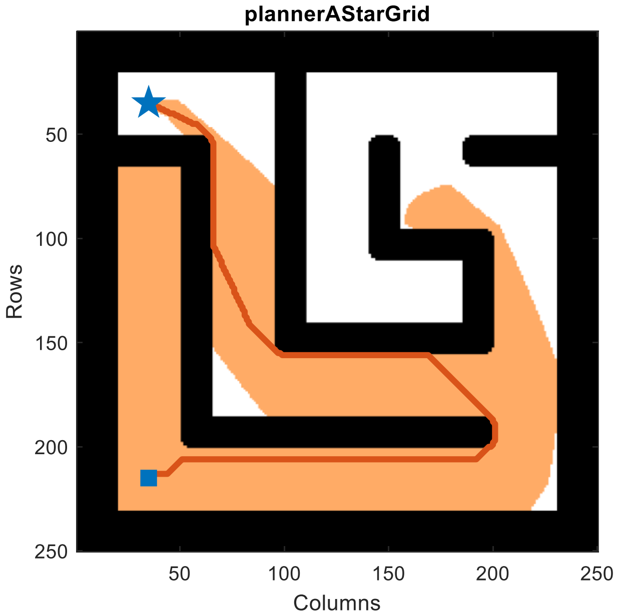
Autonomy

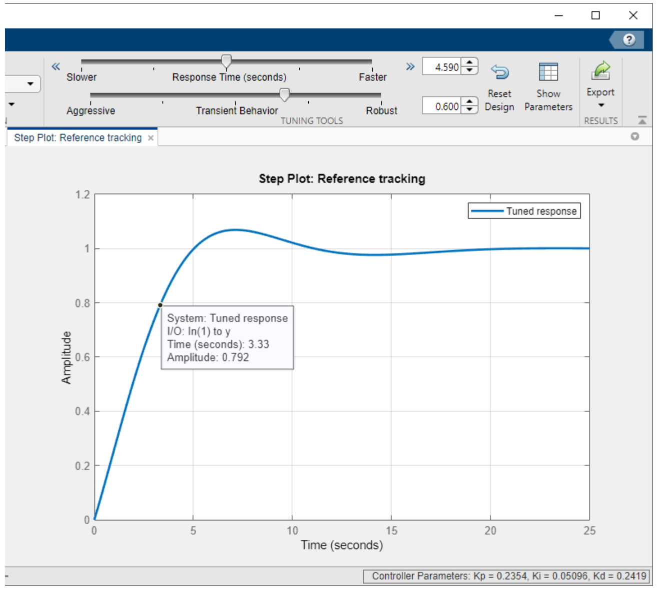
Perception & Localization

Plan & Decide

Control





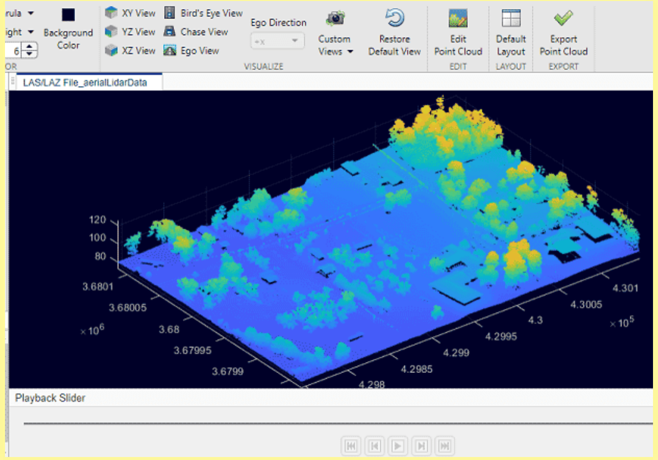


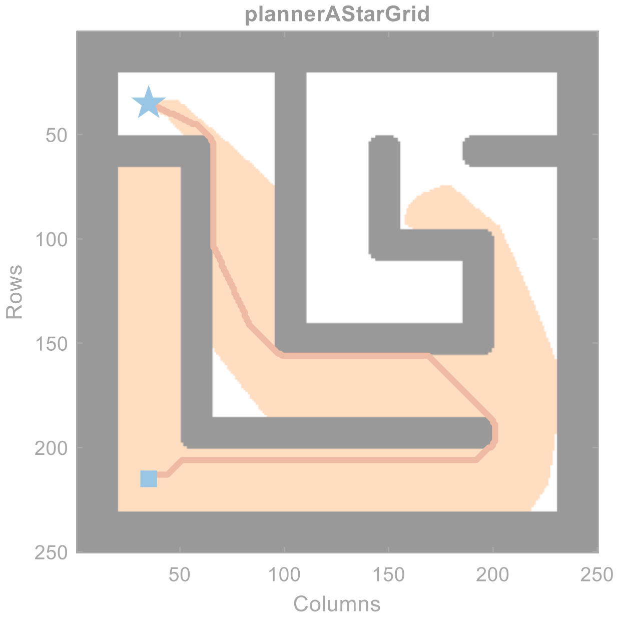
Autonomy

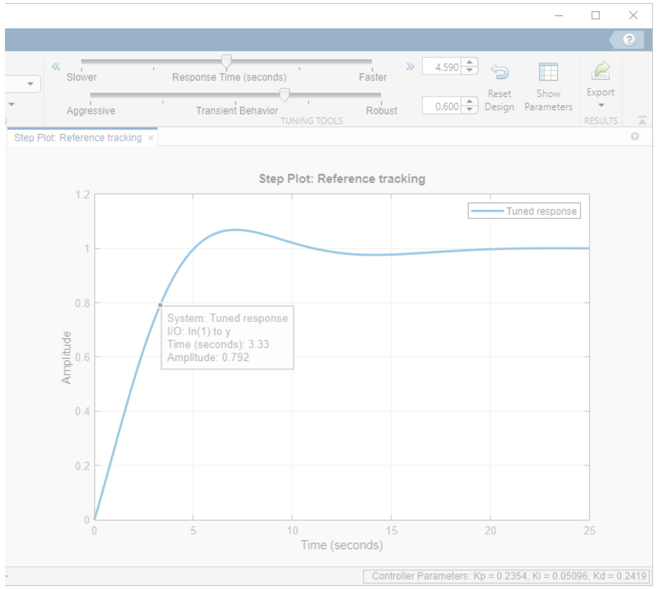
Perception & Localization

Plan & Decide

Control

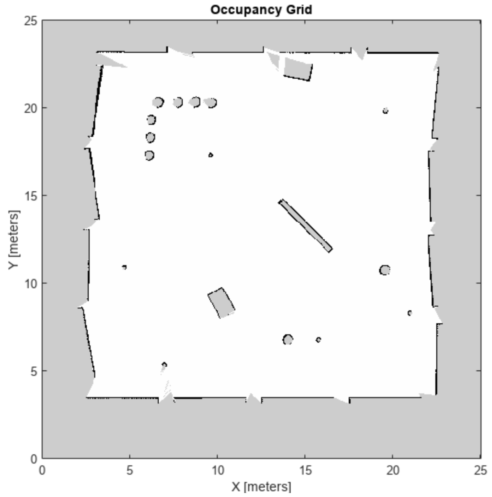


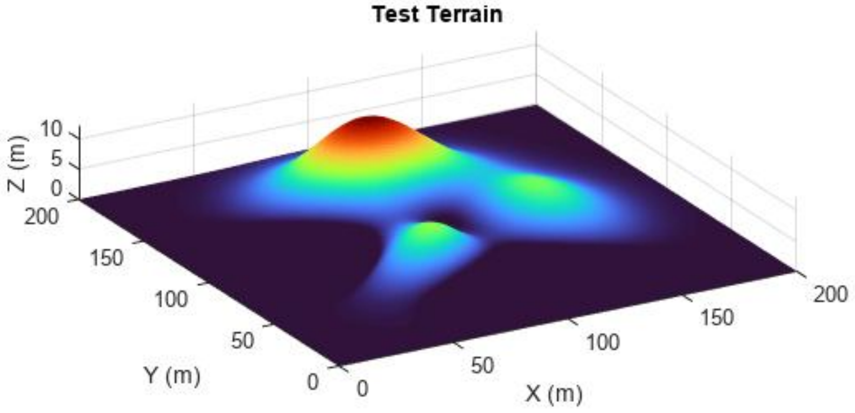


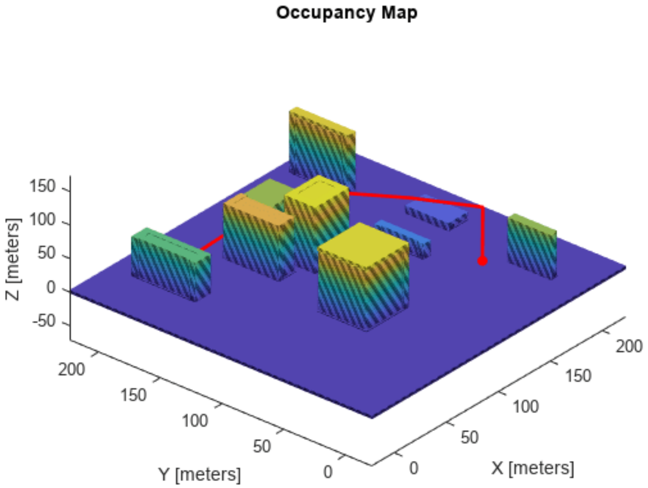


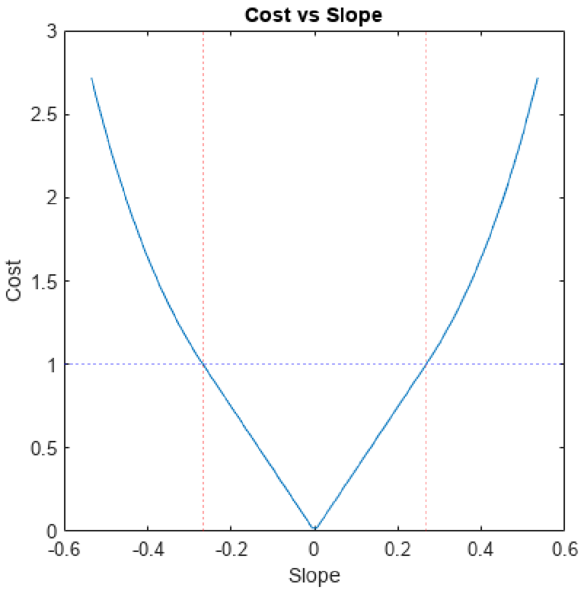
Mapping the environment

- 2D occupancy map
- 3D occupancy map
- 2.5D cost (elevation) map



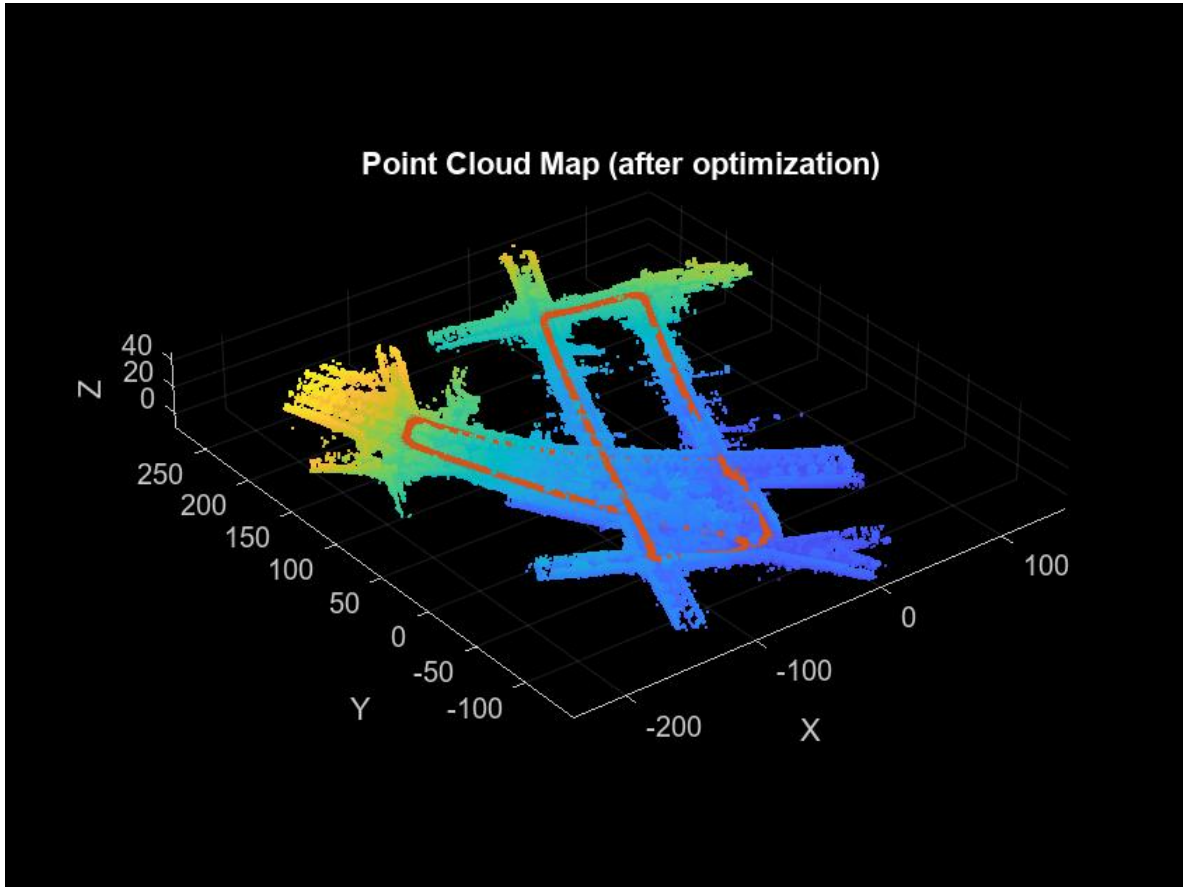




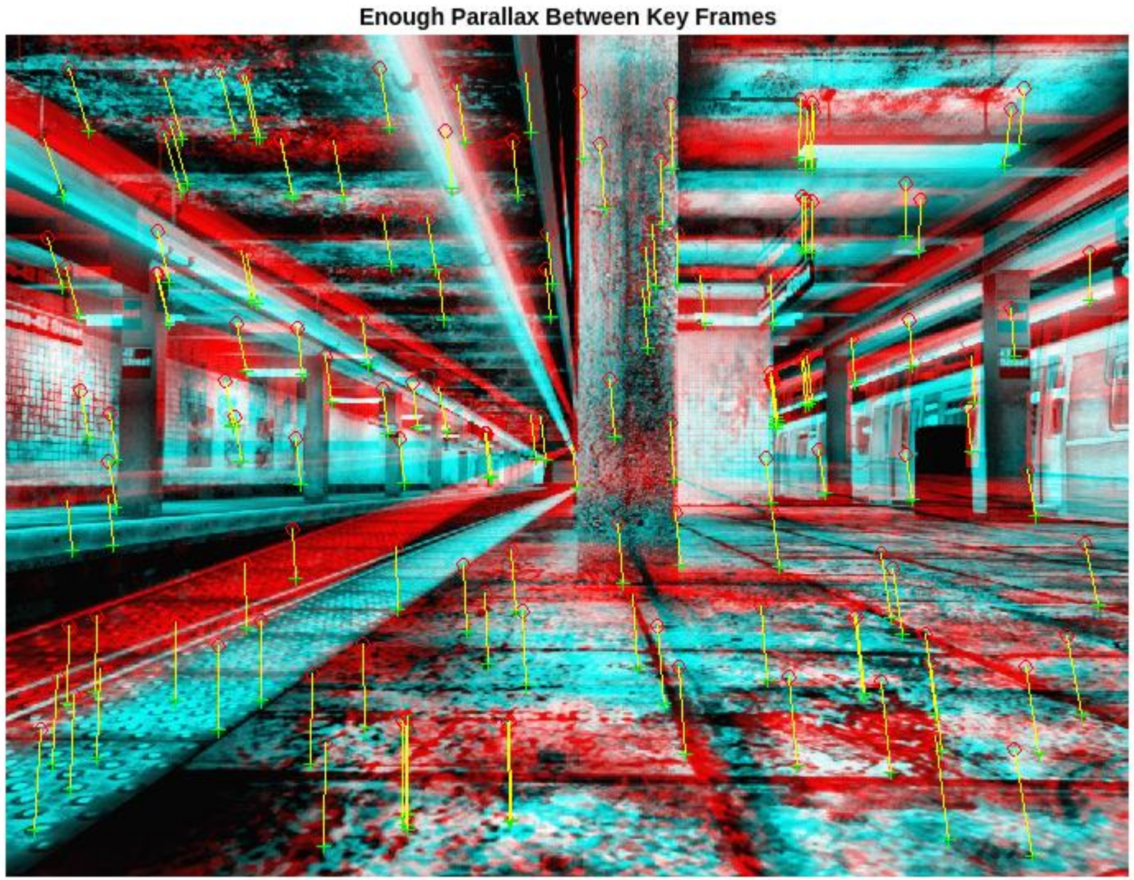


Mapping the environment

Lidar SLAM

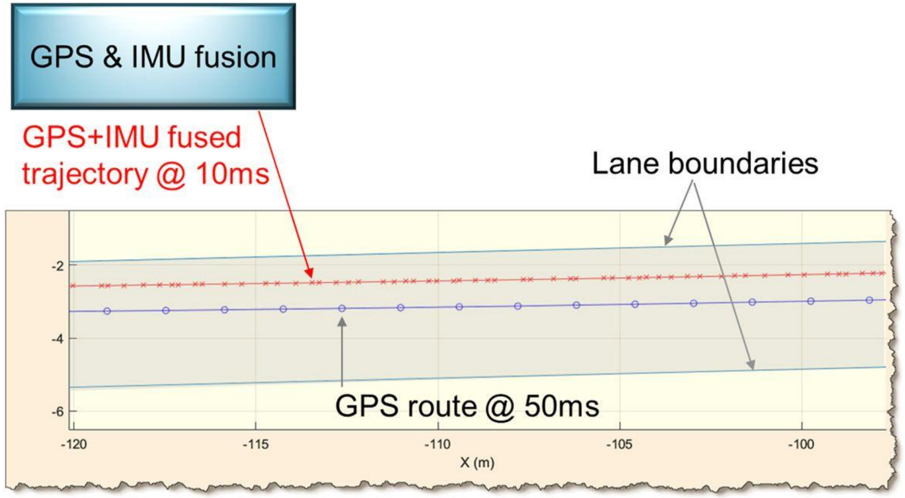


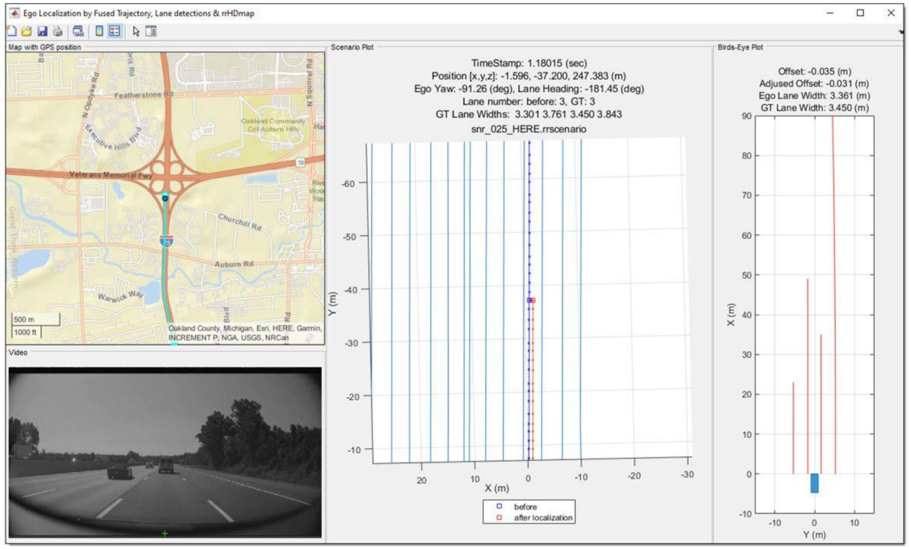
Structure from motion



Localization

- GPS
- IMU (acc + gyro + mag)
- SLAM
- Vision
- Dead reckoning



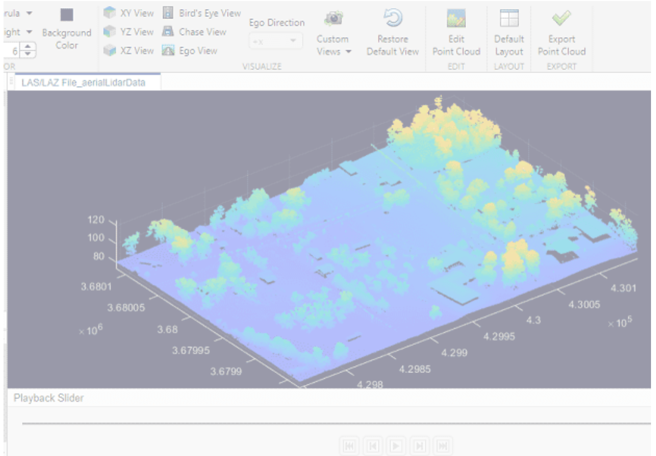


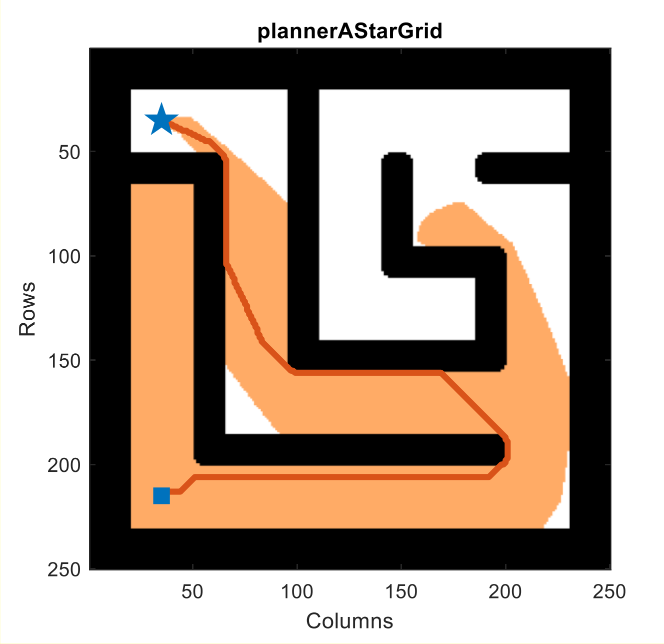
Autonomy

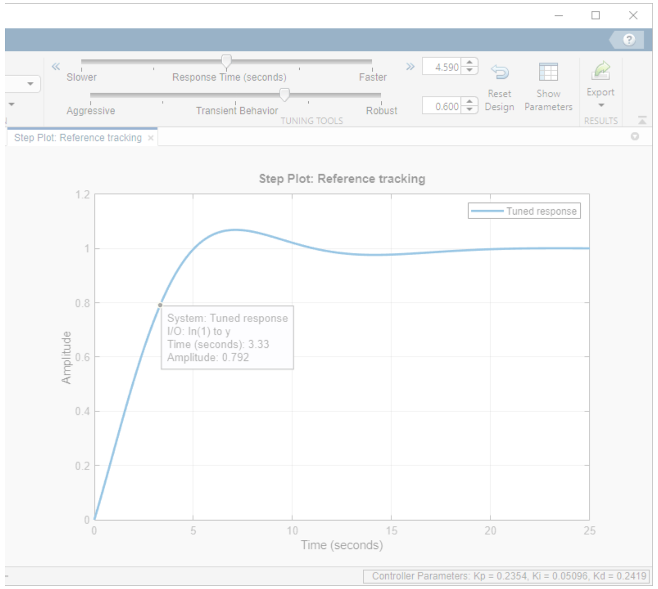
Perception & Localization

Plan & Decide

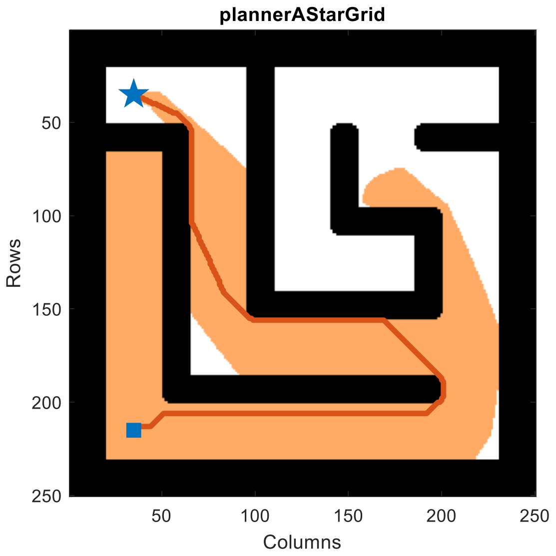
Control

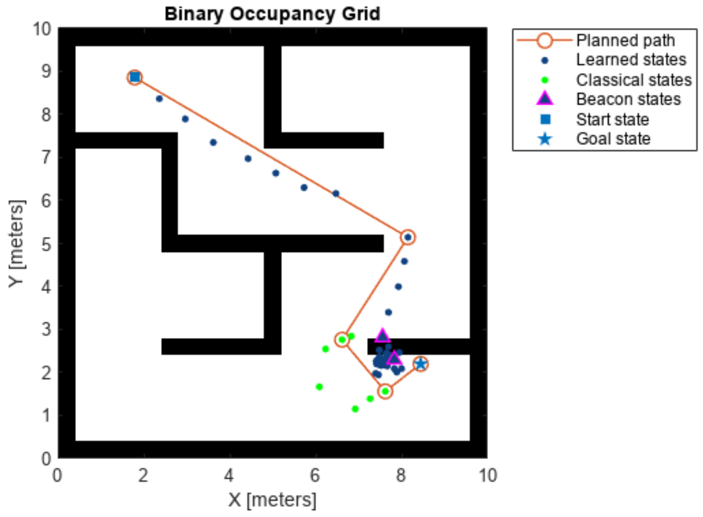


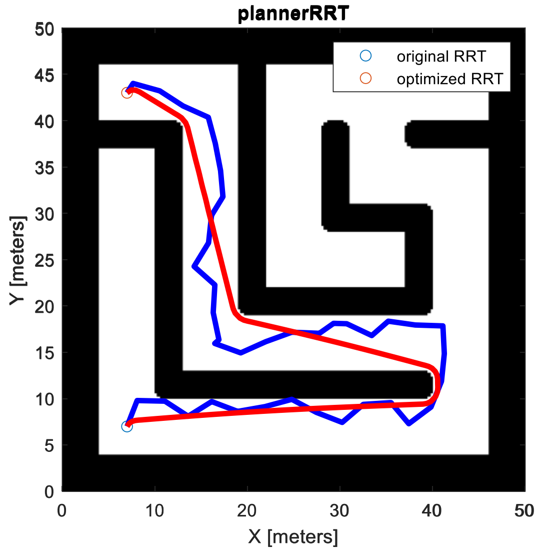


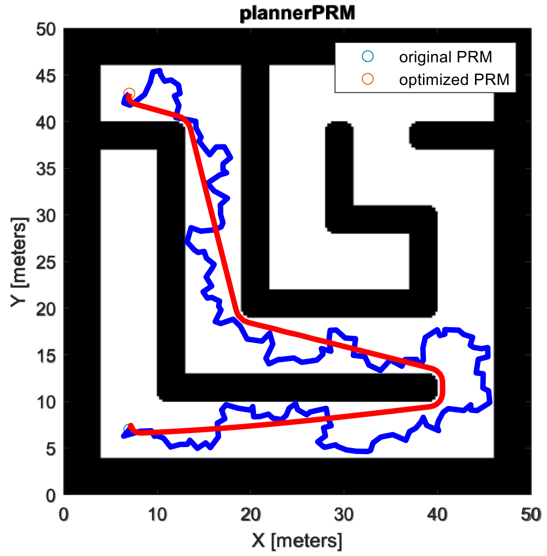


Path planning





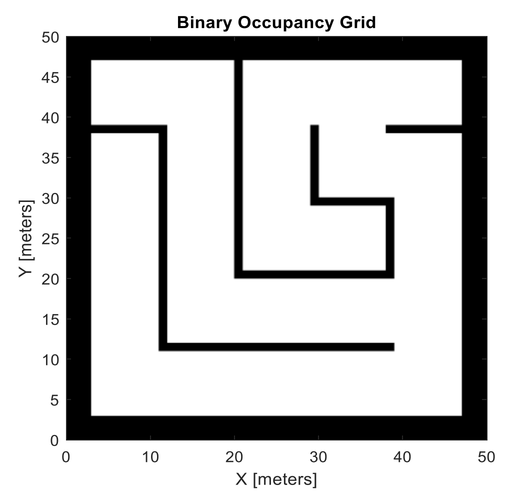




Path planning

Create a map of the environment

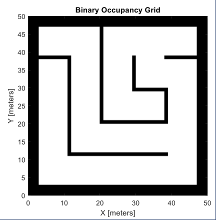
```
occMap = mapMaze(20,5);
```



Path planning

Create a map
of the environme

```
occMap = mapMaze(20,
```



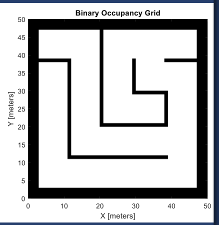
Create state space and space validator

```
stateSpace = ...
    stateSpaceSE2([0 50; 0 50; -pi pi]);
stateValidator = ...
    validatorOccupancyMap(stateSpace, "Map",occMap);
stateValidator.ValidationDistance = 1;
```

Path planning

Create a ma
of the environn

```
occMap = mapMaze(2
```



Create a planner

```
planRRT = plannerRRT(stateSpace, ...
    stateValidator, "MaxConnectionDistance", 3, ...
    "MaxNumTreeNodes", 1e3);
```

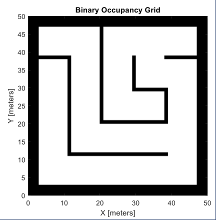
```
planPRM = plannerPRM(stateSpace, ...
    stateValidator, "MaxConnectionDistance", 3, ...
    "MaxNumNodes", 8e2);
```

```
planAStarGrid = plannerAStarGrid(occMap);
```

Path planning

Create a map
of the environmen

```
occMap = mapMaze(20,5)
```



Plan the path

```
[pathRRT, solRRT] = plan(planRRT, start, stop);
```

```
[pathPRM, solPRM] = plan(planPRM, start, stop);
```

```
plan(planAStarGrid, ...
    [50-start(2) start(1)]*resolution, ...
    [50-stop(2) stop(1)]*resolution);
```

ate a planner

```
rRRT(stateSpace, ...
or, "MaxConnectionDistance", 3, ...
odes", 1e3);
```

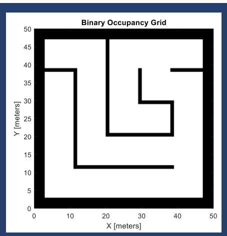
```
rPRM(stateSpace, ...
or, "MaxConnectionDistance", 3, ...
", 8e2);
```

```
plannerAStarGrid(occMap);
```

Path planning

Create a map
of the environme

```
occMap = mapMaze(20,
```



Visualize*

```
show(occMap);
hold on
scatter(start(1), start(2));
scatter(stop(1), stop(2));
plot(solRRT.TreeData(:,1), ...
    solRRT.TreeData(:,2),"Color","blue","LineWidth",1)
plot(pathRRT.States(:,1), ...
    pathRRT.States(:,2), "Color","red","LineWidth",3);
```

```
figure
show(planAStarGrid)
title("plannerAStarGrid");
```

reate a planner

```
nnerRRT(stateSpace, ...
dator, "MaxConnectionDistance", 3, ...
eeNodes", 1e3);
```

```
nnerPRM(stateSpace, ...
dator, "MaxConnectionDistance", 3, ...
des", 8e2);
```

```
= plannerAStarGrid(occMap);
```

Plan the path

```
lRRT] = plan(planRRT, start, stop);
```

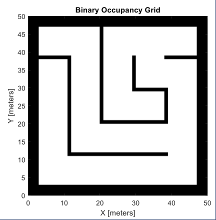
```
lPRM] = plan(planPRM, start, stop);
```

```
arGrid, ...
t(2) start(1)]*resolution, ...
(2) stop(1)]*resolution);
```

Path planning

Create a map
of the environme

```
occMap = mapMaze(20,
```



Optimize the path

```
optPathRRT = optimizePath(pathRRT.States(:,1:2), occMap);
```

```
optPathPRM = optimizePath(pathPRM.States(:,1:2), occMap);
```

reate a planner

```
nnerRRT(stateSpace, ...
dator, "MaxConnectionDistance", 3, ...
eeNodes", 1e3);
```

```
nnerPRM(stateSpace, ...
dator, "MaxConnectionDistance", 3, ...
des", 8e2);
```

```
= plannerAStarGrid(occMap);
```

Plan the path

```
lRRT] = plan(planRRT, start, stop);
```

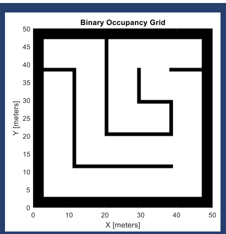
```
lPRM] = plan(planPRM, start, stop);
```

```
tarGrid, ...
t(2) start(1)]*resolution, ...
(2) stop(1)]*resolution);
```

Path planning

Create a map of the environment

```
occMap = mapMaze(20,5);
```



Create state space and space validator

```
stateSpace = ...
    stateSpaceSE2([0 50; 0 50; -pi pi]);
stateValidator = ...
    validatorOccupancyMap(stateSpace, "Map",occMap);
stateValidator.ValidationDistance = 1;
```

Create a planner

```
planRRT = plannerRRT(stateSpace, ...
    stateValidator, "MaxConnectionDistance", 3, ...
    "MaxNumTreeNodes", 1e3);
```

```
planPRM = plannerPRM(stateSpace, ...
    stateValidator, "MaxConnectionDistance", 3, ...
    "MaxNumNodes", 8e2);
```

```
planAStarGrid = plannerAStarGrid(occMap);
```

Plan the path

```
[pathRRT, solRRT] = plan(planRRT, start, stop);
```

```
[pathPRM, solPRM] = plan(planPRM, start, stop);
```

```
plan(planAStarGrid, ...
    [50-start(2) start(1)]*resolution, ...
    [50-stop(2) stop(1)]*resolution);
```

Visualize*

```
show(occMap);
hold on
scatter(start(1), start(2));
scatter(stop(1), stop(2));
plot(solRRT.TreeData(:,1), ...
    solRRT.TreeData(:,2),"Color","blue","LineWidth",1)
plot(pathRRT.States(:,1), ...
    pathRRT.States(:,2), "Color","red","LineWidth",3);
```

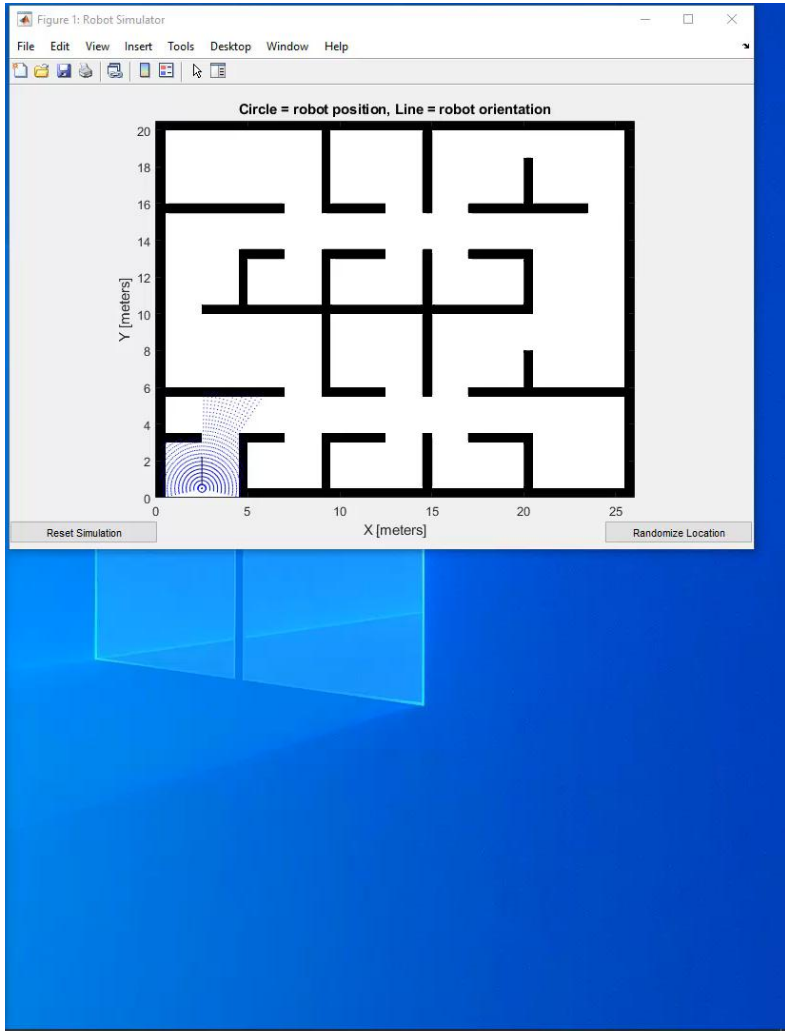
```
figure
show(planAStarGrid)
title("plannerAStarGrid");
```

Optimize the path

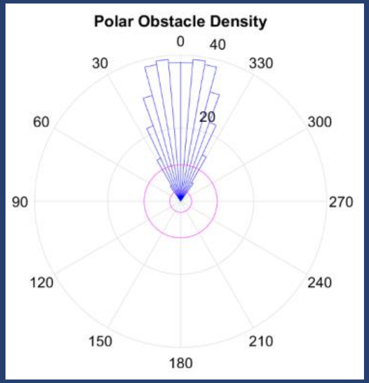
```
optPathRRT = optimizePath(pathRRT.States(:,1:2), occMap);
```

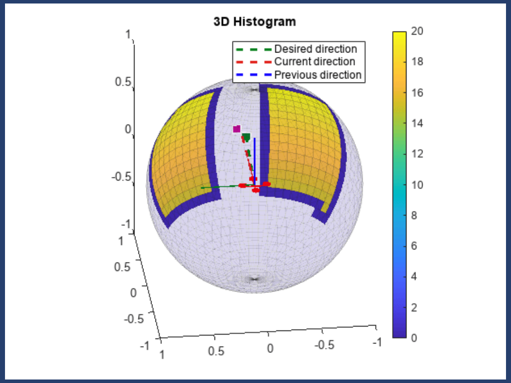
```
optPathPRM = optimizePath(pathPRM.States(:,1:2), occMap);
```

Collision Avoidance

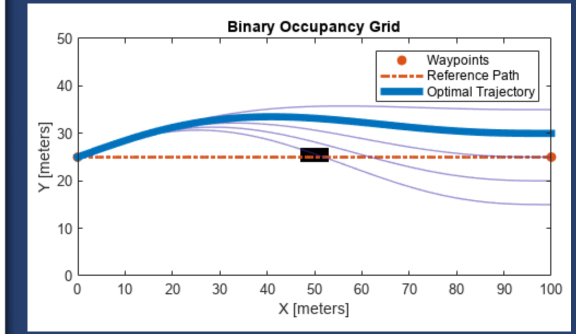


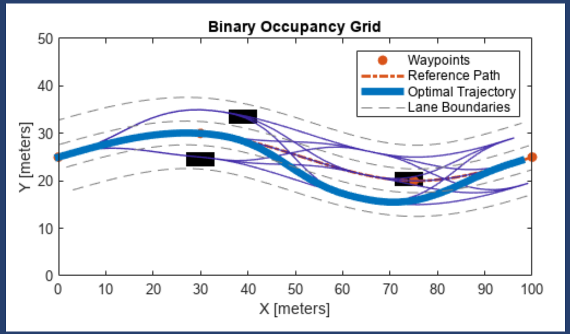
Vector Field Histogram





Optimal trajectory in Frenet coorinated system



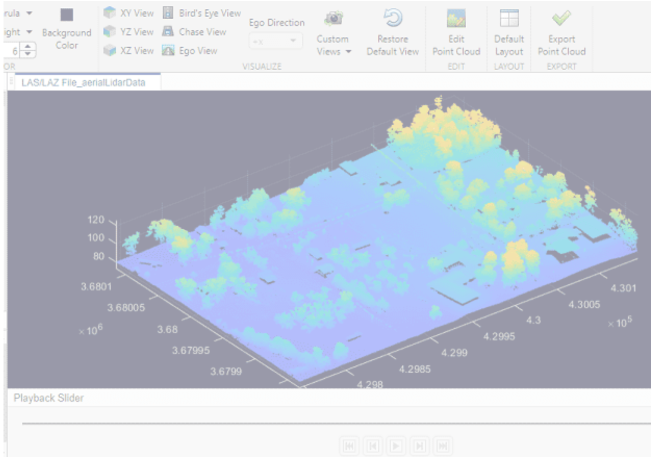


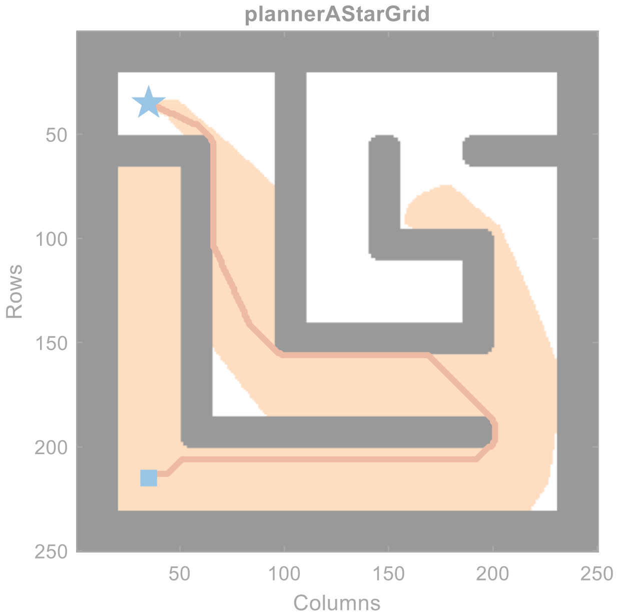
Autonomy

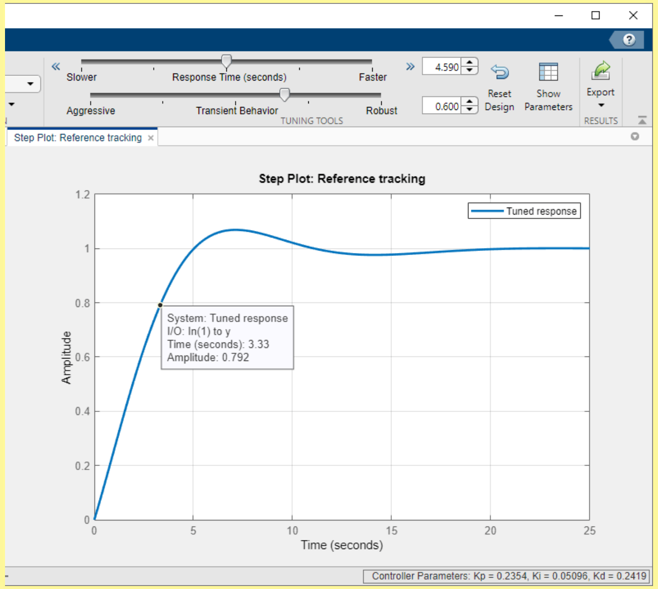
Perception & Localization

Plan & Decide

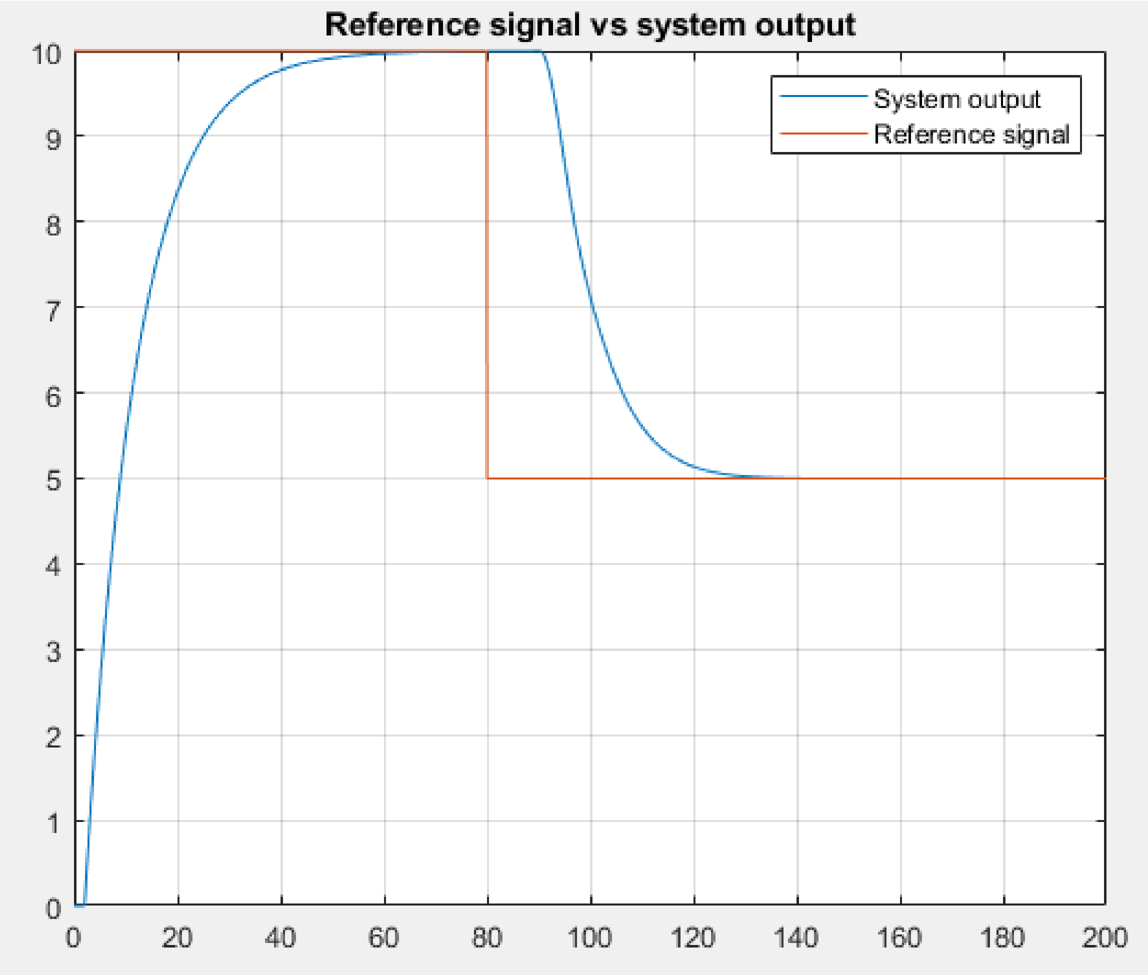
Control

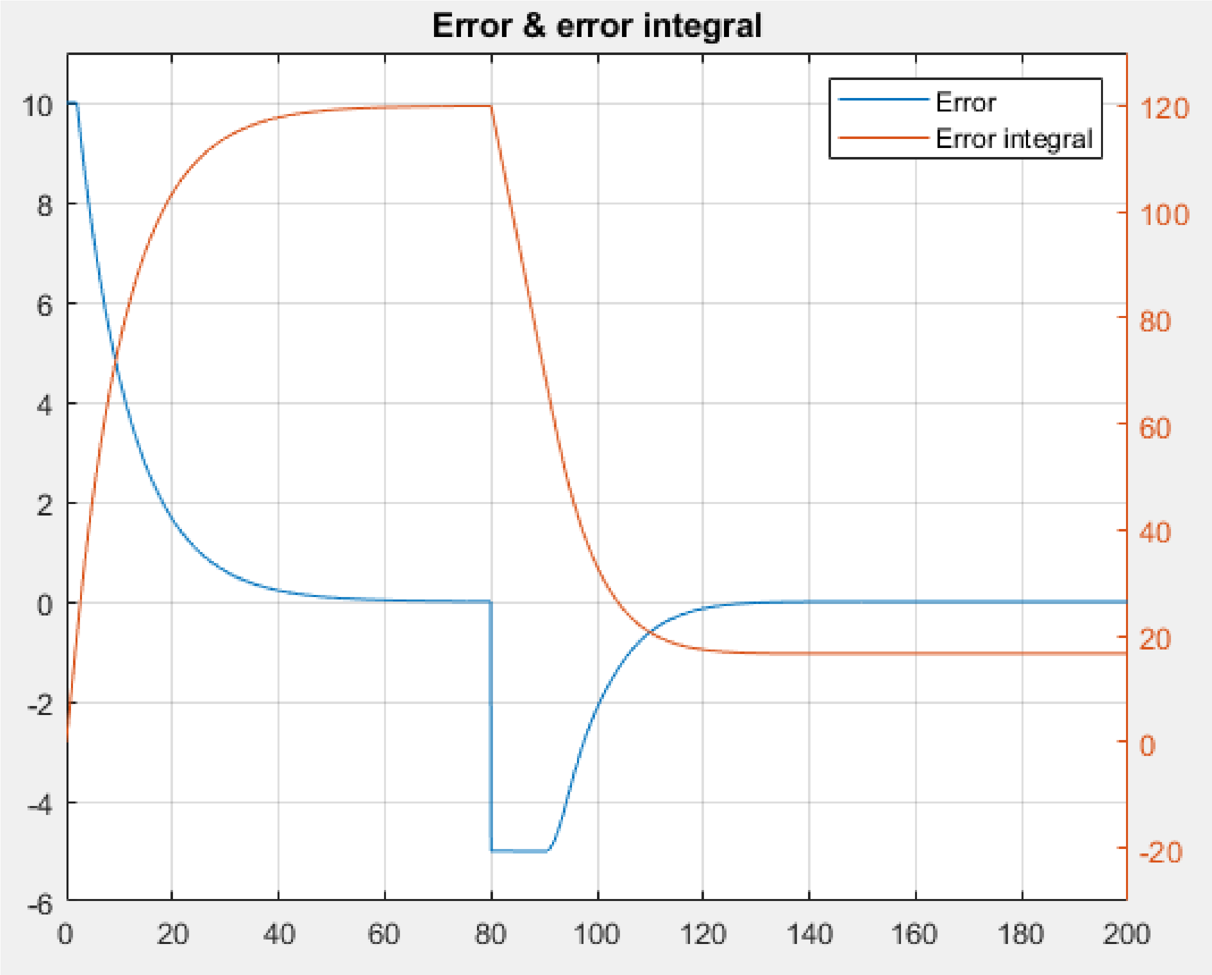






Controllers





Types of controllers

- PID
- Active Disturbance Rejection Controller
- Model Predictive Control
- Reinforced Learning Controller
- Fuzzy Logic Controller

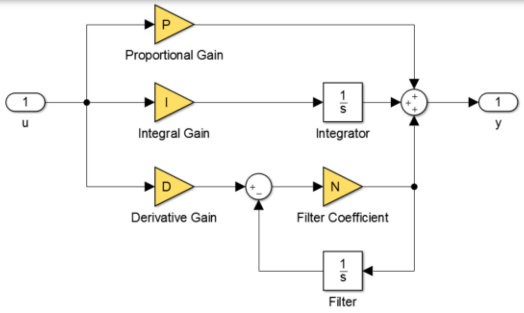
Adaptive Control

Optimal Control

PID Controller

HOW

- P term increases controller speed
- I term removes steady state error
- D term minimizes the overshoot and oscilations



WHY

+ Simple design
+ Able to satisfy most control problems
+ Autotuning with MATLAB

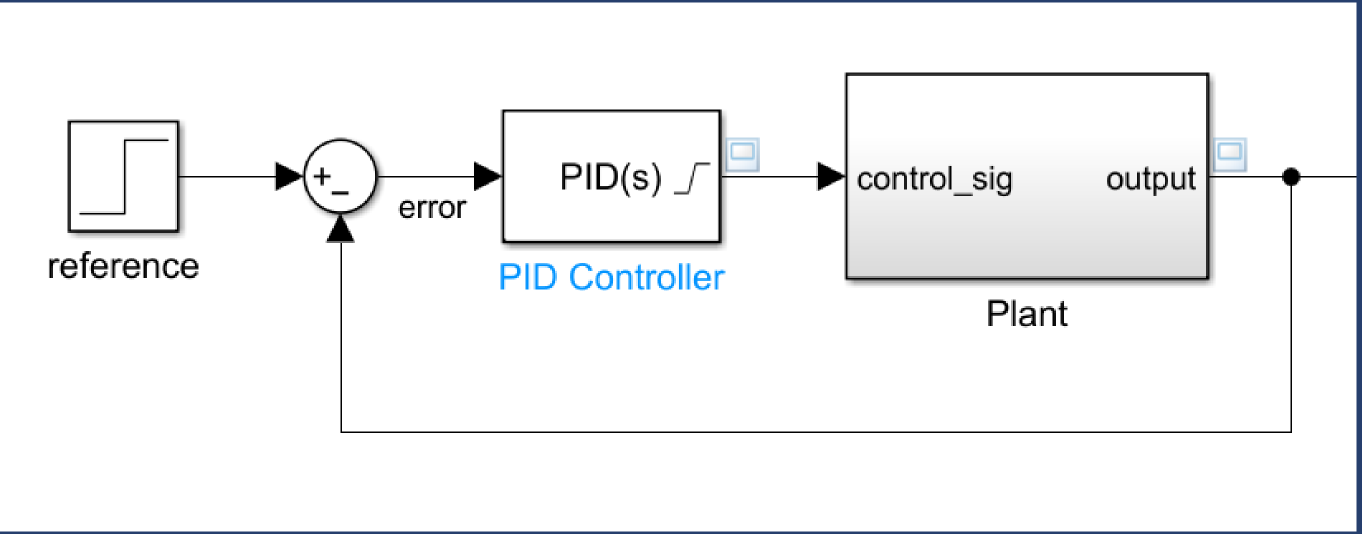
- Works best with SISO LTI systems
- Doesn't handle disturbances and noise well

WHEN

- Good initial choice in most cases

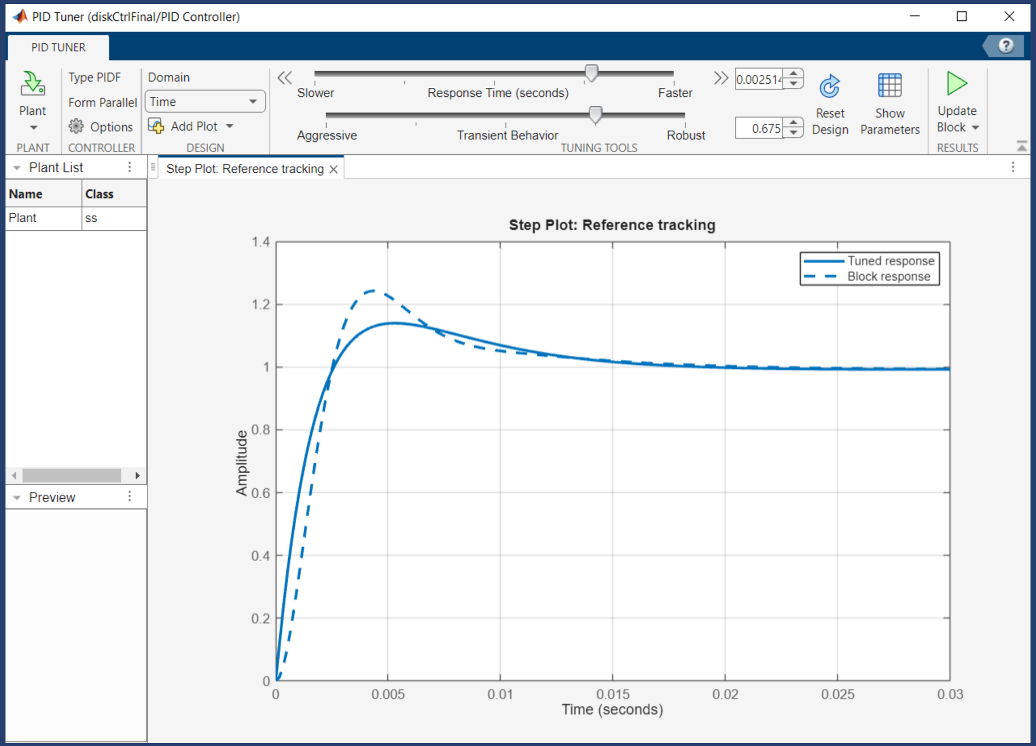
PID Controller

Insert the PID controller block

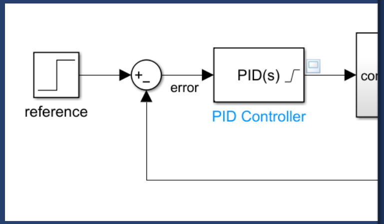


PID Controller

Tune the controller gains

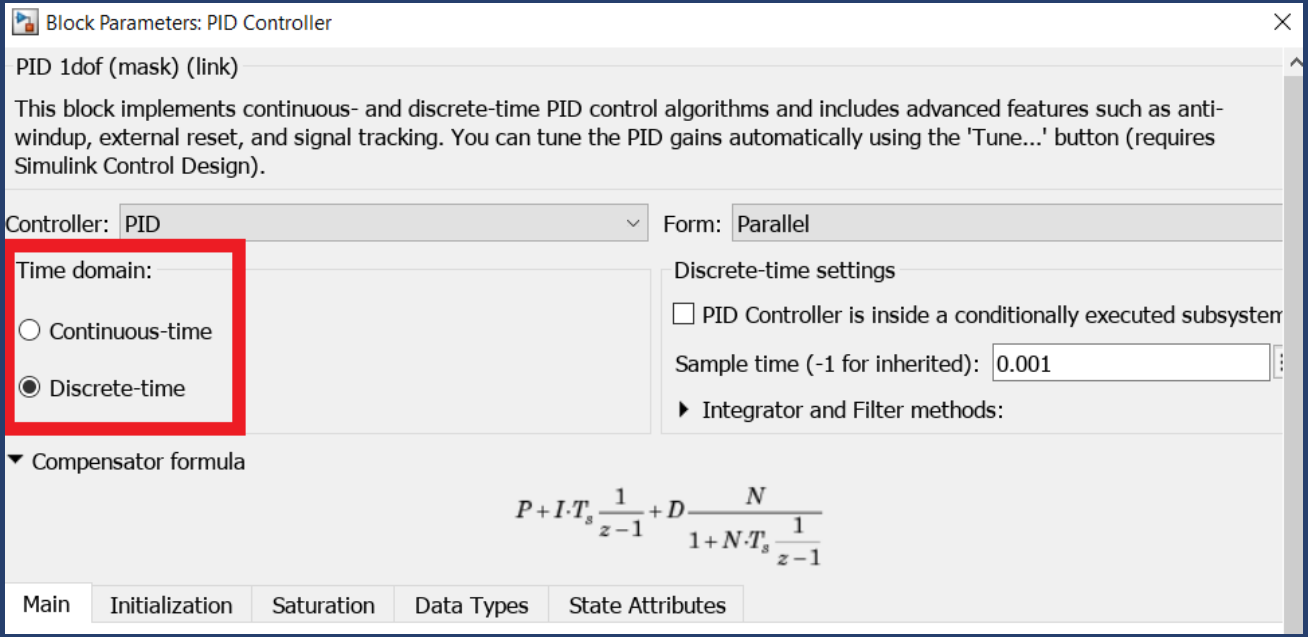


Insert the PID contr

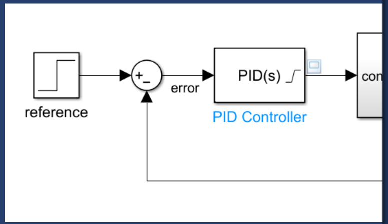


PID Controller

Discretize the tuned controller

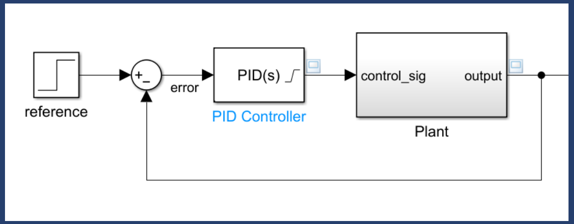


Insert the PID contr

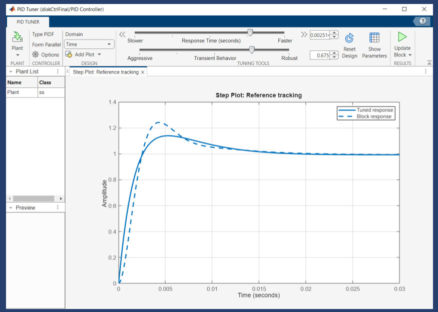


PID Controller

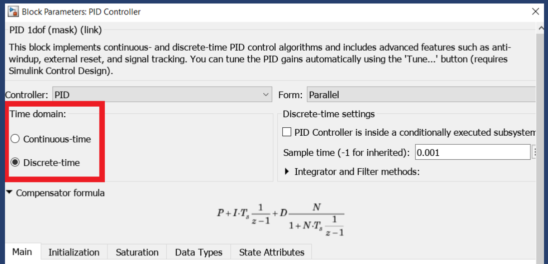
Insert the PID controller block



Tune the controller gains



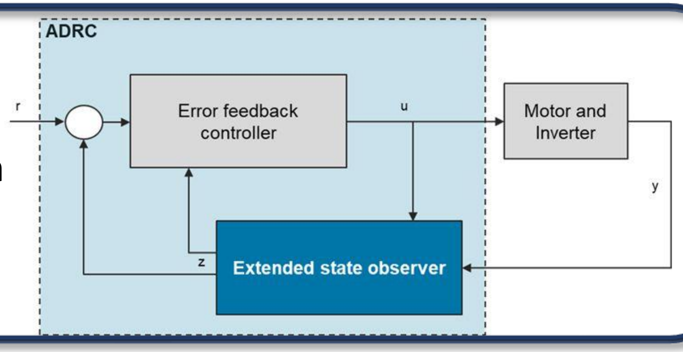
Discretize the tuned controller



Active Disturbance Rejection Control

HOW

- Extended State Observer is used to estimate uncertainties and disturbances
- Controller reduces the effect of that estimate on the known part of the system



WHY

+ Handles nonlinearities and disturbances
+ Requires only an approximate model
+ Can provide better performance than PID

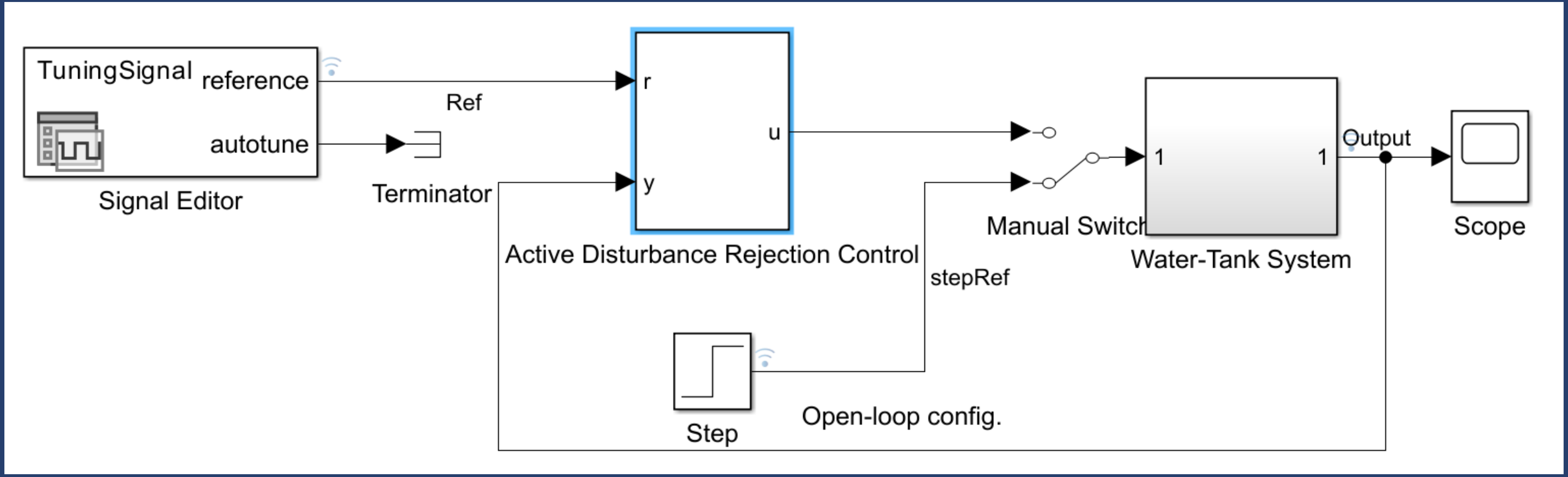
− Tuning isn't always easy
− Efficiency depends on ESO

WHEN

- Uncertain dynamics, unknown disturbances, time-varying parameters and approximate plant model

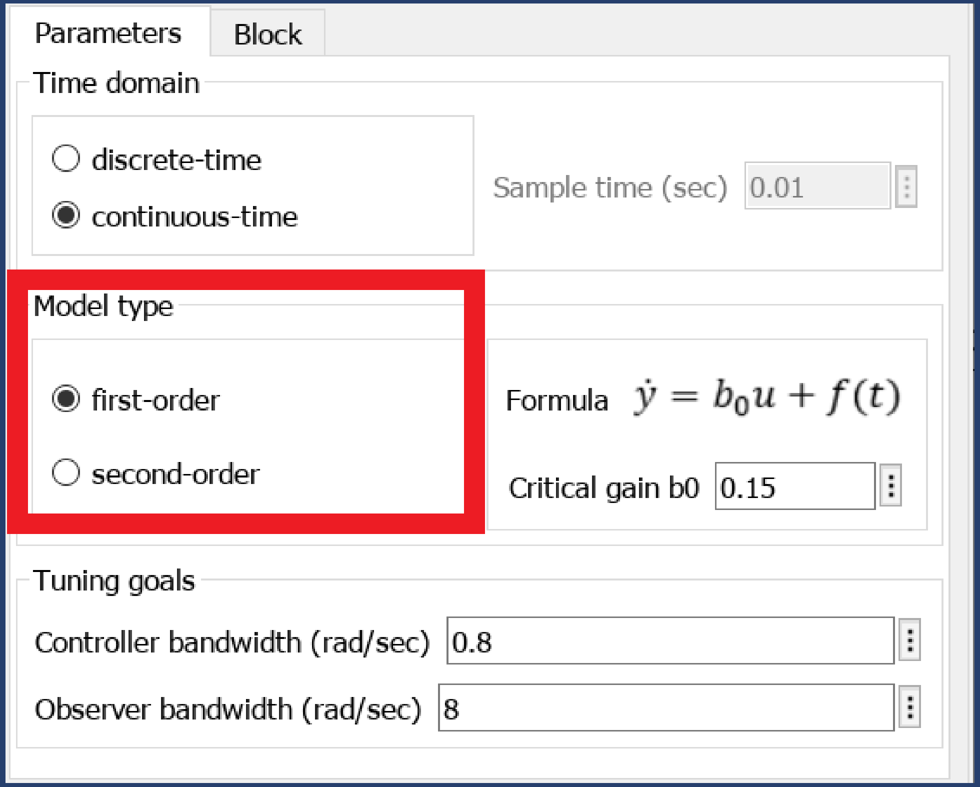
Active Disturbance Rejection Control

Insert the ADRC block



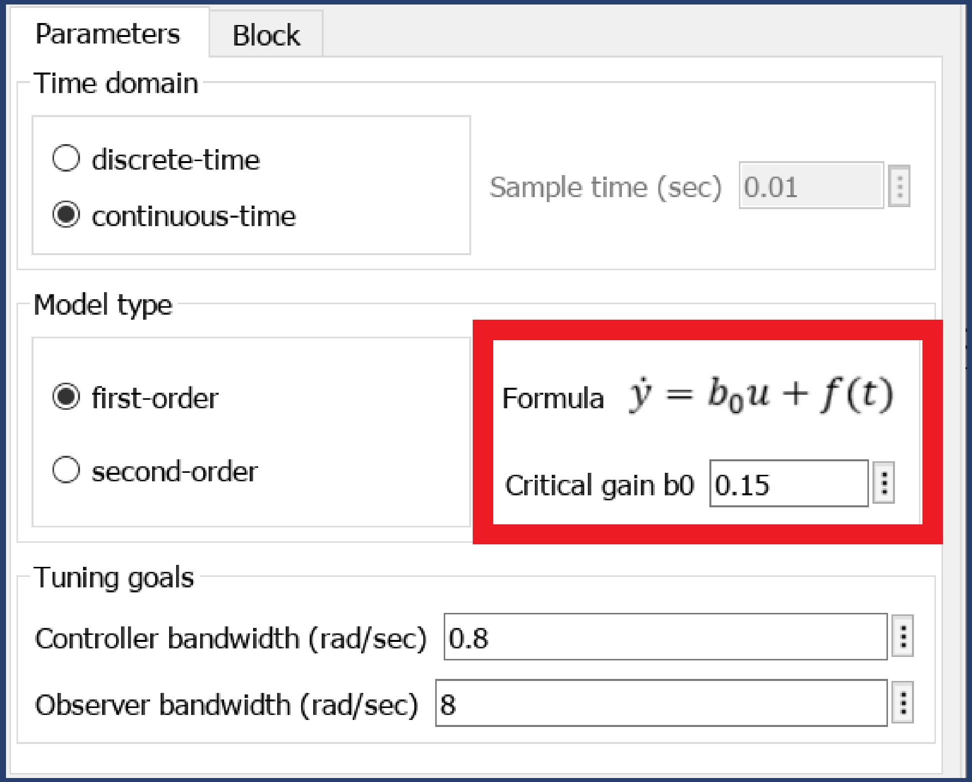
Active Disturbance Rejection Control

Select the controller order



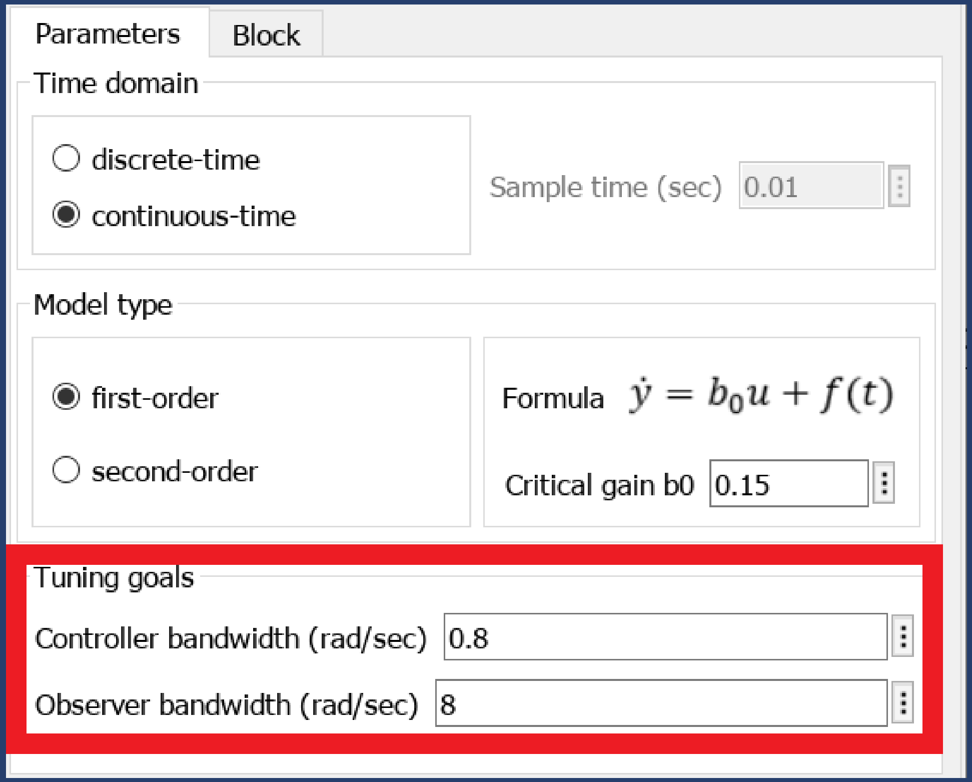
Active Disturbance Rejection Control

Provide a reasonable guess for critical gain



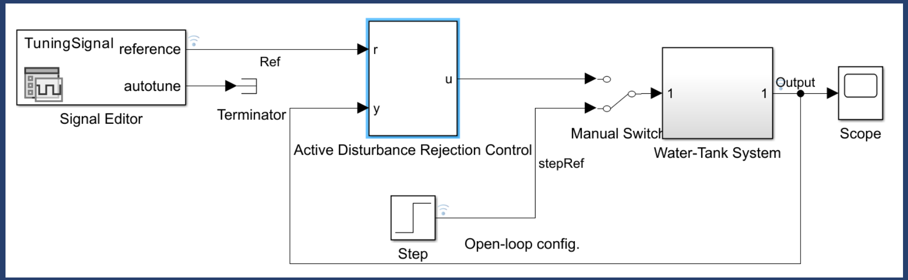
Active Disturbance Rejection Control

Set the controller and observer bandwidths

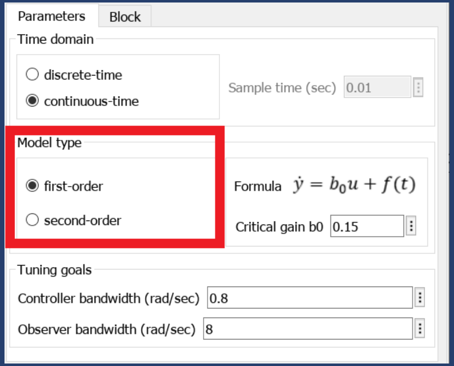


Active Disturbance Rejection Control

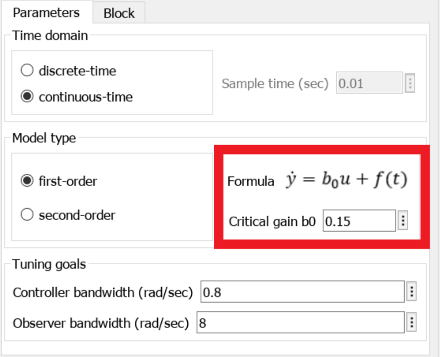
Insert the ADRC block



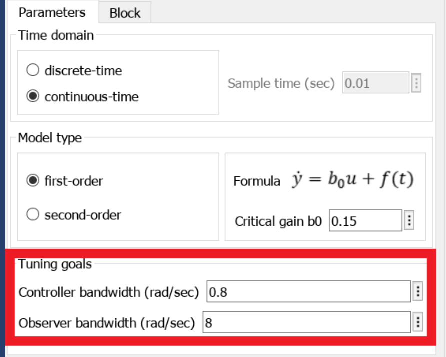
Select the controller order



Provide a reasonable guess for critical gain



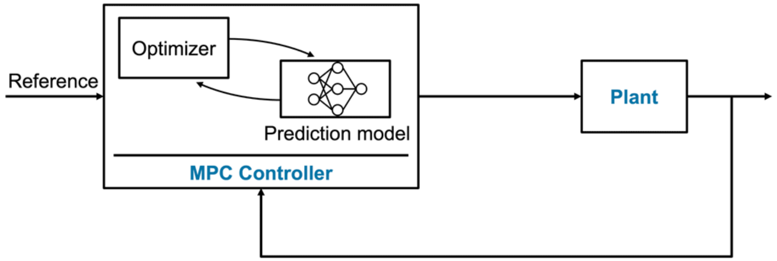
Set the controller and observer bandwidths



Model Predictive Control

HOW

- Internal model of the plant to predict its future state
- Iteratively calculates the optimal sequence of control commands
- The first few actions are used and the entire proces is reiterated



WHY

\+ Can take future actions into consideration

\+ More responsive

\+ Well suited to handle MIMO systems

\- The model accuracy heavily impacts performance

\- Resource heavy

WHEN

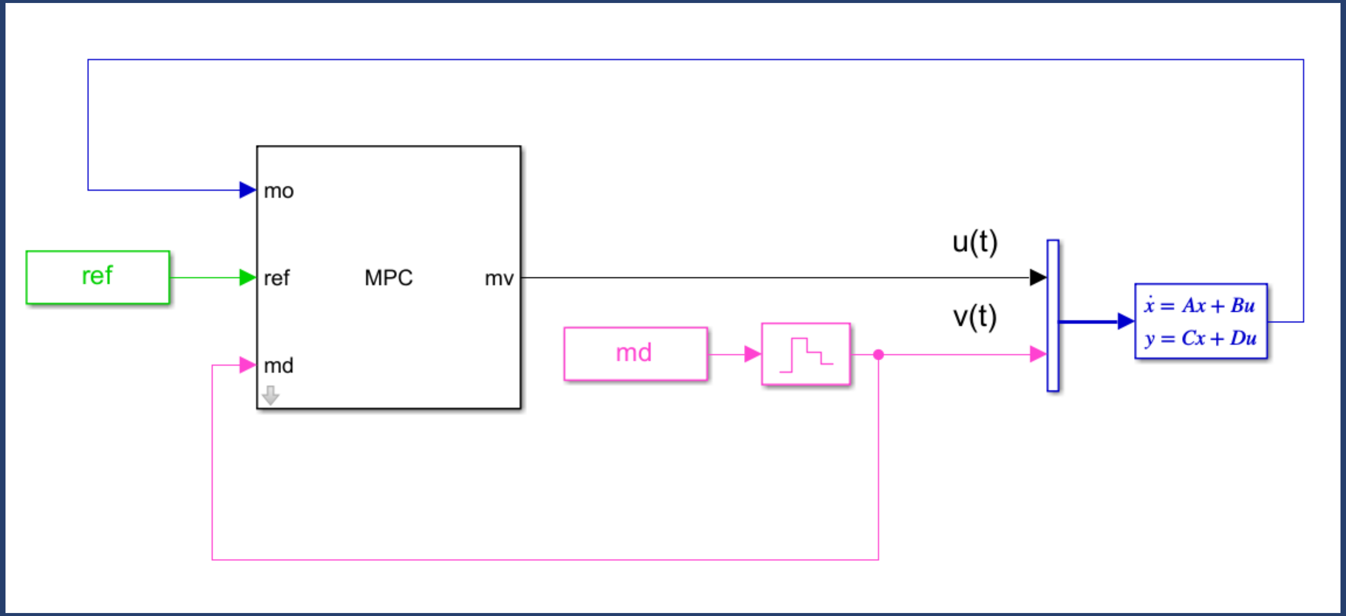
- Complex MIMO systems with constraints / operating limits

! Must be able to fit at least 10-20 samples in the span of the rise time

! We have an accurate model of the system

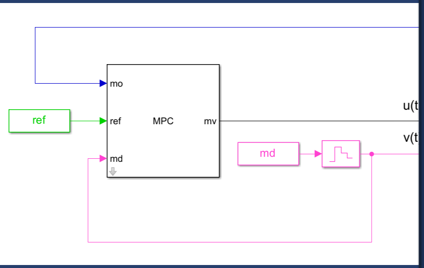
Model Predictive Control

Insert the MPC block

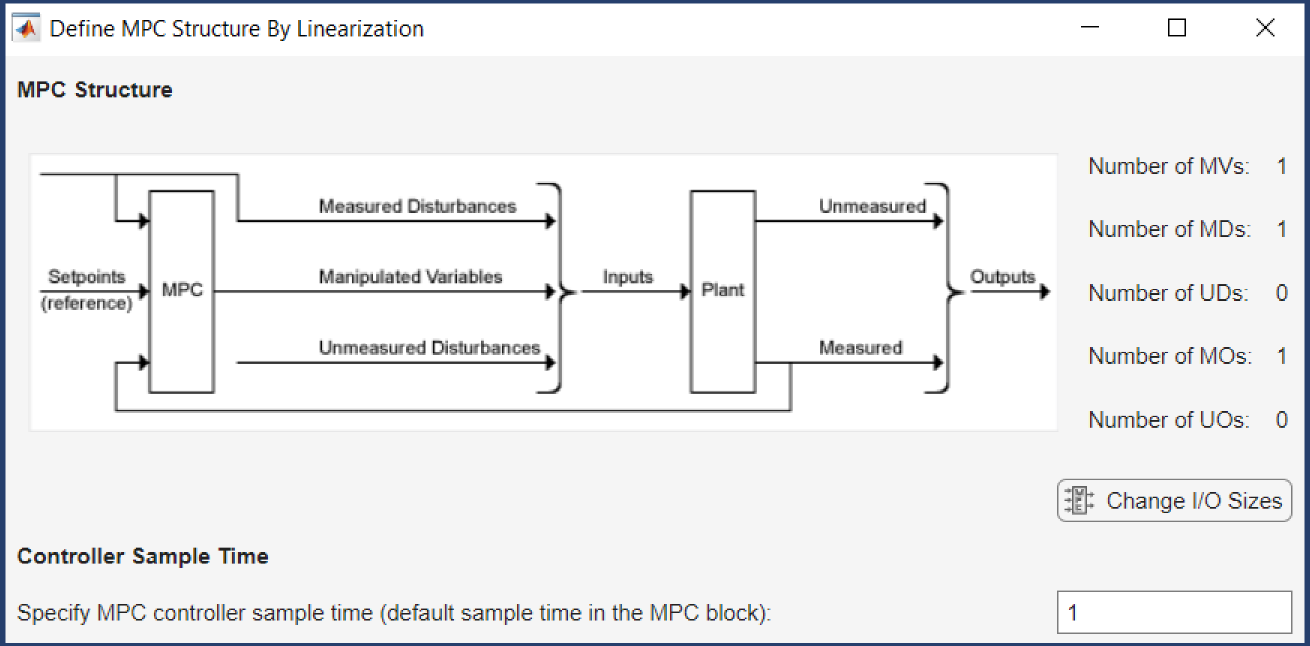


Model Predictive Control

Insert the MPC bl

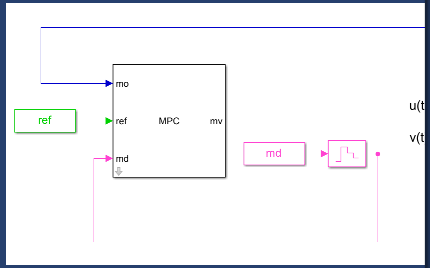


Choose structure and sampling time



Model Predictive Control

Insert the MPC bl



Define I/O signal constraints

Input and Output Channel Specifications

Plant Inputs

	Channel	Type	Name	Unit	Nominal Value	Scale Factor
1	u(1)	MV	MPC Controller		0	1
2	u(2)	MD	Zero-Order Hold1		0	1

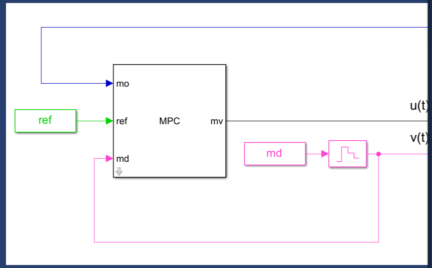
Plant Outputs

	Channel	Type	Name	Unit	Nominal Value	Scale Factor
1	y(1)	MO	State-Space1		0	1

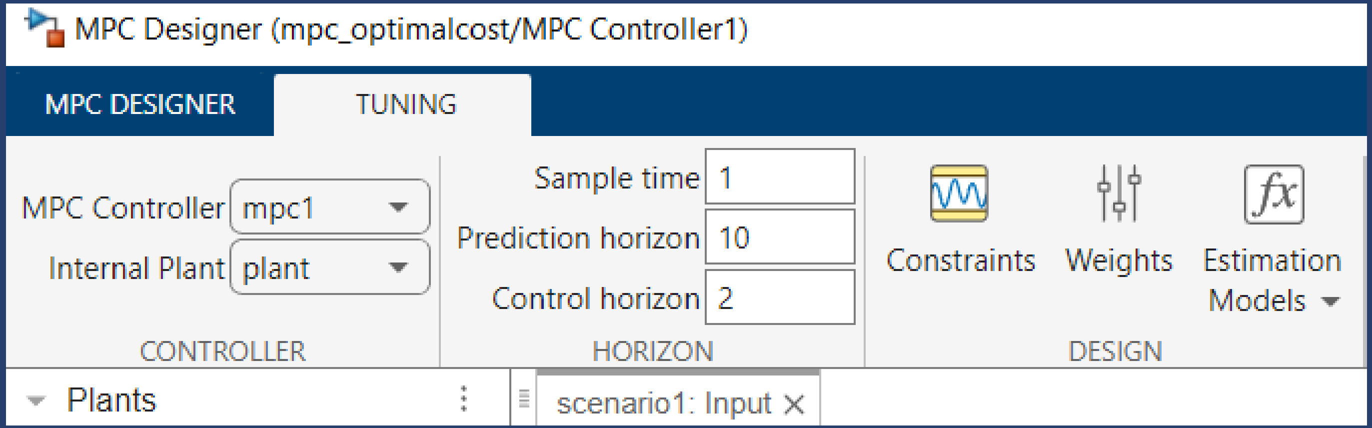
Help OK Cancel Apply

Model Predictive Control

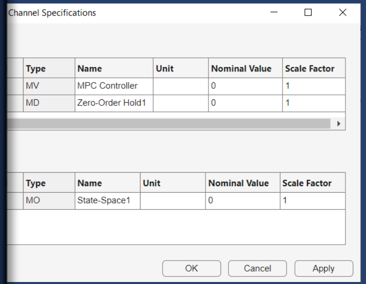
Insert the MPC bl



Choose prediction and control horizons

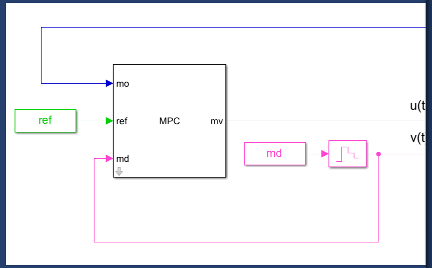


I/O signal constraints



Model Predictive Control

Insert the MPC bl



Select weights for the optimization function

Weights (mpc1)

Input Weights (dimensionless)

	Channel	Type	Weight	Rate Weight	Target
1	u(1)	MV	0	0.1	nominal

Output Weights (dimensionless)

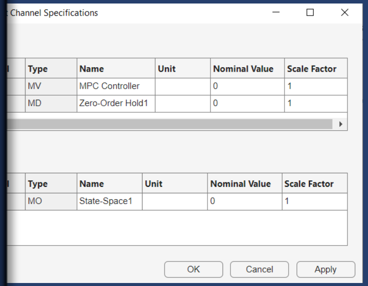
	Channel	Type	Weight
1	y(1)	MO	1

ECR Weight (dimensionless)

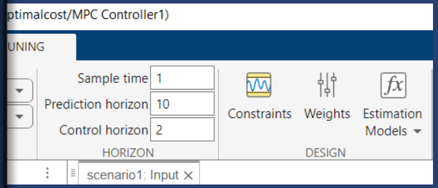
Weight on the slack variable: 100000

Help | OK | Cancel | Apply

I/O signal constraints

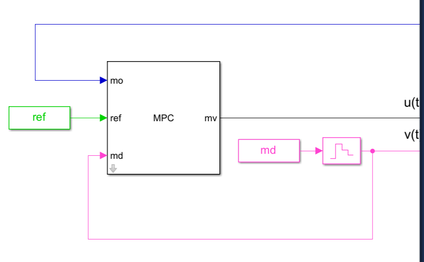


prediction and control horizons

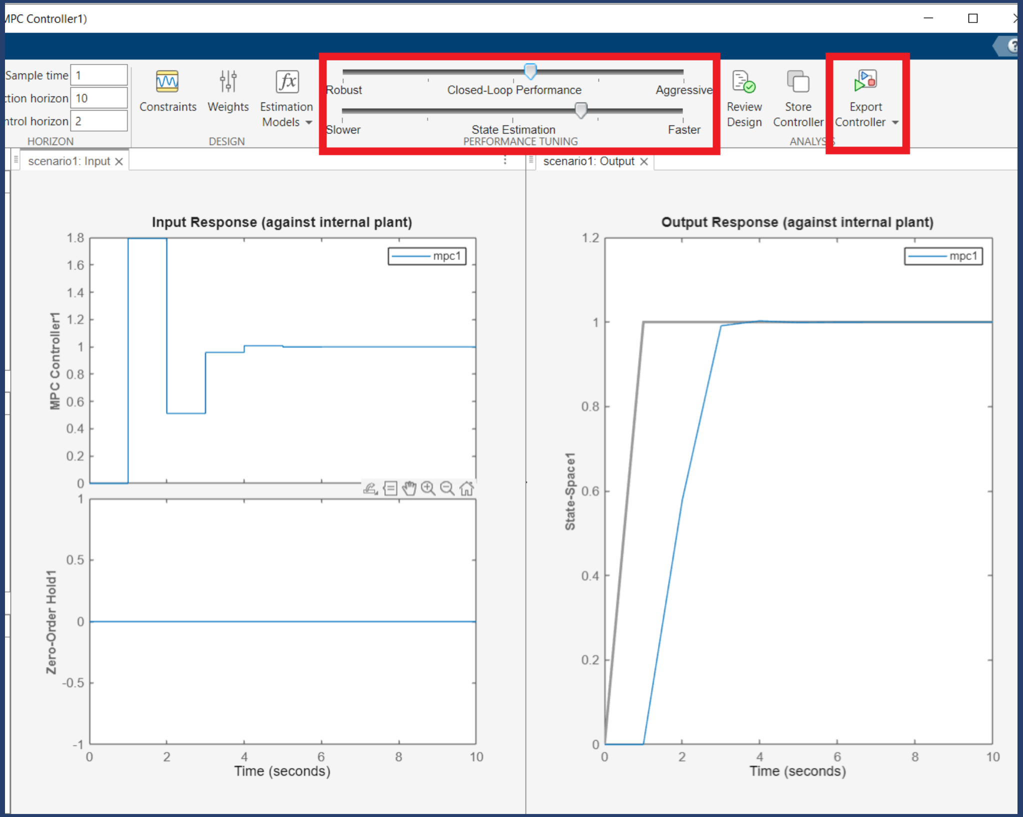


Model Predictive Control

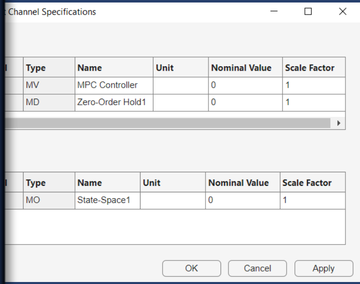
Insert the MPC bl



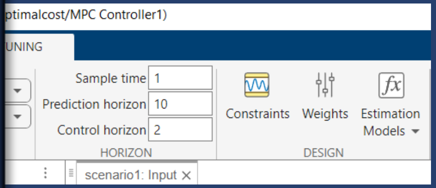
Tune the response



I/O signal constraints

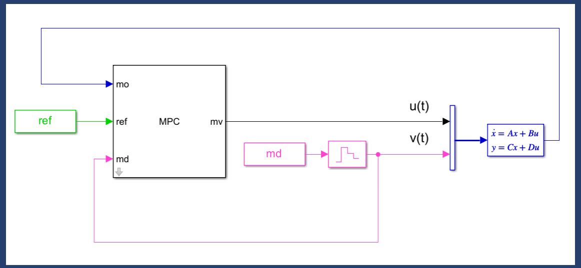


prediction and control horizons

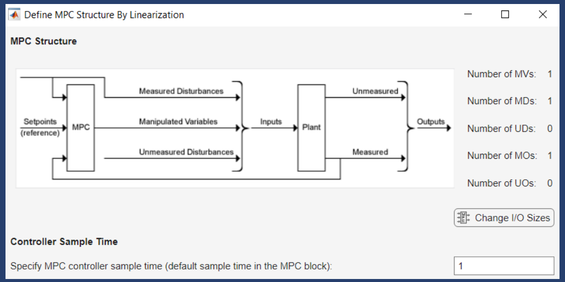


Model Predictive Control

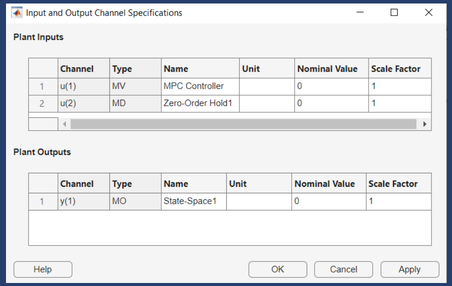
Insert the MPC block



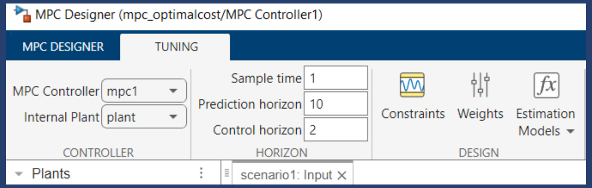
Choose structure and sampling time



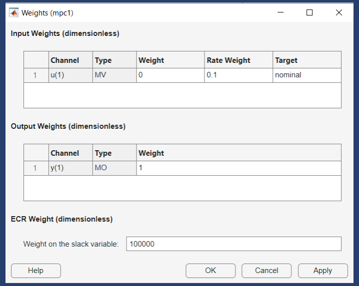
Define I/O signal constraints



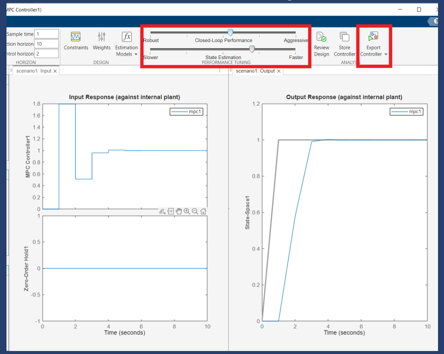
Choose prediction and control horizons



Select weights for the optimization function



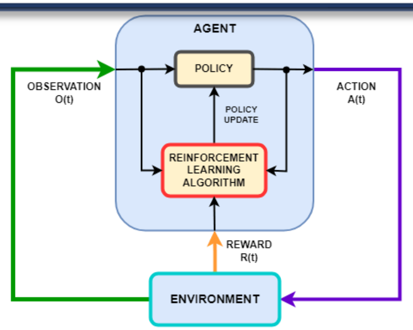
Tune the response



Reinforcement Learning

HOW

- Machine Learning agent interacting with the environment
- "Policy" - a Deep Neural Network
- Policy updated through rewards and punishments



WHY

\+ Can handle different kinds of outputs

\+ Well suited for MIMO systems

\+ Maximizes goal function

\- Training takes time and effort

\- Resource heavy

WHEN

- When it is difficult to characterize dynamics and operating conditions, and learning control policies directly from data is more practical

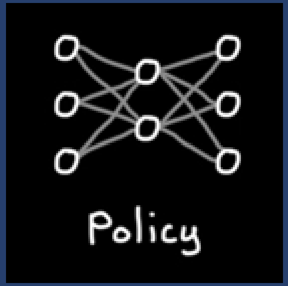
! The hardware platform can support it

Reinforcement Learning

Create environment and its interface

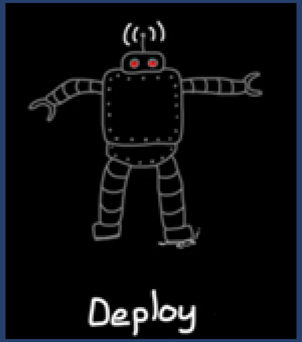
Create the reward signal

Create the agent and the critic neural networks



Train the neural networks

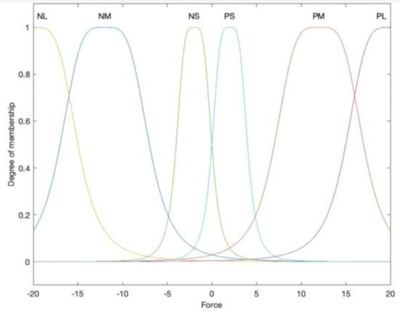
Deploy



Fuzzy Logic

HOW

- Defined by rules, membership functions and corresponding actions
- A weighted action is calculated based on membership functions



WHY

+ Works with hard-to-model systems
+ In some cases, might be more intuitive
+ Interpretable

− Relies on our knowledge of the system
− Might be difficult to tune

WHEN

- When it's easier to infer logical rules of control rather than a mathematical model

Fuzzy Logic

Create a Fuzzy Inference System object

```
cpFIS = mamfis(...
    'NumInputs', 1, 'NumInputMFs', 2,...
    'NumOutputs', 1, 'NumOutputMFs', 2,...
    'AddRule', 'none');
```

Fuzzy Logic

Create a Fuzzy Inferen
System object

```
cpFIS = mamfis(...
    'NumInputs', 1, 'NumInputMFs', 2
    'NumOutputs', 1, 'NumOutputMFs',
    'AddRule', 'none');
```

Define inputs parameters

```
cpFIS.Inputs(1).Name = 'Theta';
cpFIS.Inputs(1).Range = [-pi, pi];
cpFIS.Inputs(1).MembershipFunctions(1).Name = 'Negative';
cpFIS.Inputs(1).MembershipFunctions(1).Type = 'zmf';
cpFIS.Inputs(1).MembershipFunctions(1).Parameters = [-0.5 0.5];
cpFIS.Inputs(1).MembershipFunctions(2).Name = 'Positive';
cpFIS.Inputs(1).MembershipFunctions(2).Type = 'smf';
cpFIS.Inputs(1).MembershipFunctions(2).Parameters = [-0.5 0.5];
```

Fuzzy Logic

Create a Fuzzy Inferen
System object

```
cpFIS = mamfis(...
    'NumInputs', 1, 'NumInputMFs', 2
    'NumOutputs', 1, 'NumOutputMFs',
    'AddRule', 'none');
```

Define outputs parameters

```
cpFIS.Outputs(1).Name = 'Force';
cpFIS.Outputs(1).Range = [-20, 20];
cpFIS.Outputs(1).MembershipFunctions(1).Name = 'NM'; % Negative Medium
cpFIS.Outputs(1).MembershipFunctions(1).Type = 'gbellmf';
cpFIS.Outputs(1).MembershipFunctions(1).Parameters = [5, 2, -12];
cpFIS.Outputs(1).MembershipFunctions(2).Name = 'PM'; % Positive Medium
cpFIS.Outputs(1).MembershipFunctions(2).Type = 'gbellmf';
cpFIS.Outputs(1).MembershipFunctions(2).Parameters = [5, 2, 12];
```

Fuzzy Logic

Create a Fuzzy Inferen
System object

```
cpFIS = mamfis(...
    'NumInputs', 1, 'NumInputMFs', 2
    'NumOutputs', 1, 'NumOutputMFs',
    'AddRule', 'none');
```

Specify the rules

```
rules = [...
    "If Theta is Negative then Force is NM";...
    "If Theta is Positive then Force is PM"];

cpFIS = addRule(cpFIS, rules);
```

ine outputs parameters

```
ts(1).Name = 'Force';
ts(1).Range = [-20, 20];
ts(1).MembershipFunctions(1).Name = 'NM'; % Negative Medium
ts(1).MembershipFunctions(1).Type = 'gbellmf';
ts(1).MembershipFunctions(1).Parameters = [5, 2, -12];
ts(1).MembershipFunctions(2).Name = 'PM'; % Positive Medium
ts(1).MembershipFunctions(2).Type = 'gbellmf';
ts(1).MembershipFunctions(2).Parameters = [5, 2, 12];
```


Fuzzy Logic

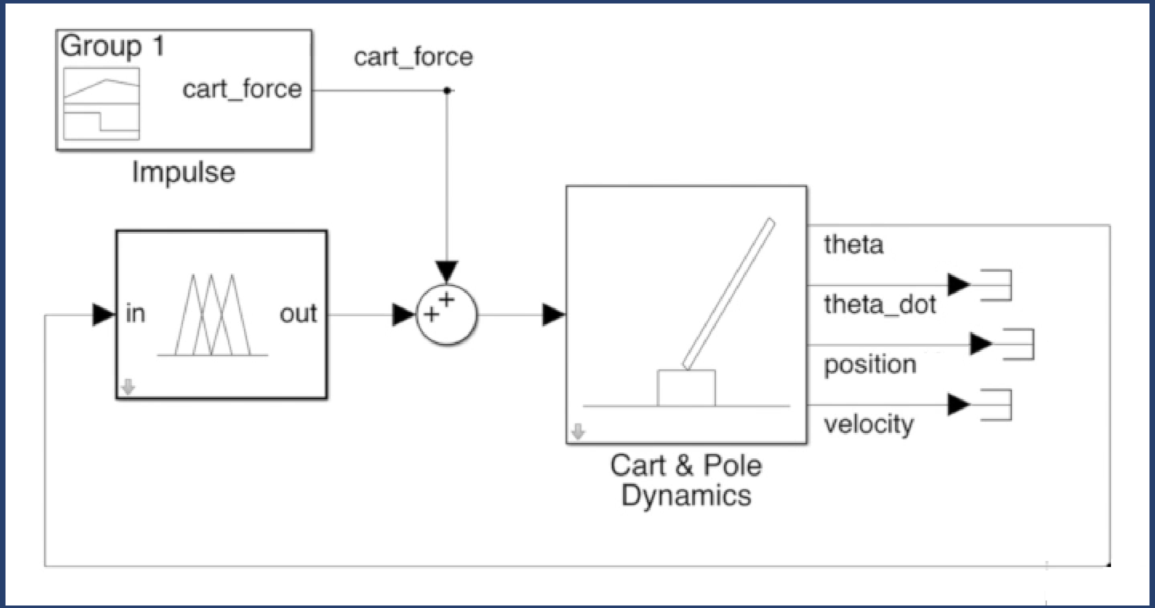
Create a Fuzzy Inferen
System object

```
cpFIS = mamfis(...
    'NumInputs', 1, 'NumInputMFs', 2
    'NumOutputs', 1, 'NumOutputMFs',
    'AddRule', 'none');
```

ine outputs parameters

```
ts(1).Name = 'Force';
ts(1).Range = [-20, 20];
ts(1).MembershipFunctions(1).Name = 'NM'; % Negative Medium
ts(1).MembershipFunctions(1).Type = 'gbellmf';
ts(1).MembershipFunctions(1).Parameters = [5, 2, -12];
ts(1).MembershipFunctions(2).Name = 'PM'; % Positive Medium
ts(1).MembershipFunctions(2).Type = 'gbellmf';
ts(1).MembershipFunctions(2).Parameters = [5, 2, 12];
```

Add a Fuzzy Logic Controller block



Fuzzy Logic

Create a Fuzzy Inference System object

```
cpFIS = mamfis(...
    'NumInputs', 1, 'NumInputMFs', 2,...
    'NumOutputs', 1, 'NumOutputMFs', 2,...
    'AddRule', 'none');
```

Define inputs parameters

```
cpFIS.Inputs(1).Name = 'Theta';
cpFIS.Inputs(1).Range = [-pi, pi];
cpFIS.Inputs(1).MembershipFunctions(1).Name = 'Negative';
cpFIS.Inputs(1).MembershipFunctions(1).Type = 'zmf';
cpFIS.Inputs(1).MembershipFunctions(1).Parameters = [-0.5 0.5];
cpFIS.Inputs(1).MembershipFunctions(2).Name = 'Positive';
cpFIS.Inputs(1).MembershipFunctions(2).Type = 'smf';
cpFIS.Inputs(1).MembershipFunctions(2).Parameters = [-0.5 0.5];
```

Define outputs parameters

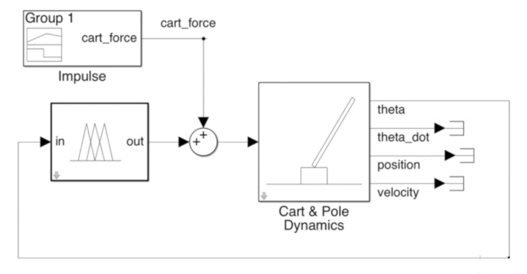
```
cpFIS.Outputs(1).Name = 'Force';
cpFIS.Outputs(1).Range = [-20, 20];
cpFIS.Outputs(1).MembershipFunctions(1).Name = 'NM'; % Negative Medium
cpFIS.Outputs(1).MembershipFunctions(1).Type = 'gbellmf';
cpFIS.Outputs(1).MembershipFunctions(1).Parameters = [5, 2, -12];
cpFIS.Outputs(1).MembershipFunctions(2).Name = 'PM'; % Positive Medium
cpFIS.Outputs(1).MembershipFunctions(2).Type = 'gbellmf';
cpFIS.Outputs(1).MembershipFunctions(2).Parameters = [5, 2, 12];
```

Specify the rules

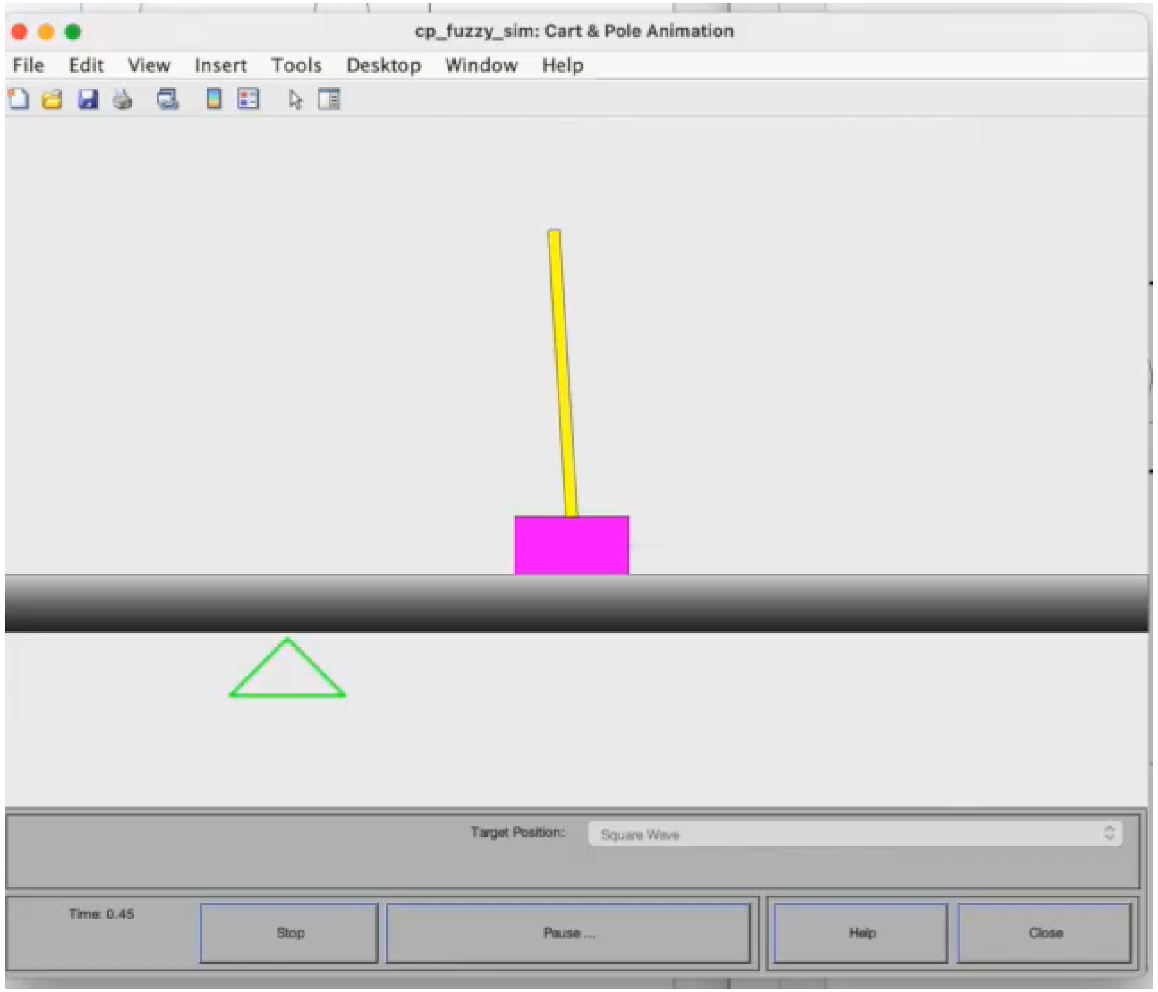
```
rules = [...
    "If Theta is Negative then Force is NM";...
    "If Theta is Positive then Force is PM"];

cpFIS = addRule(cpFIS, rules);
```

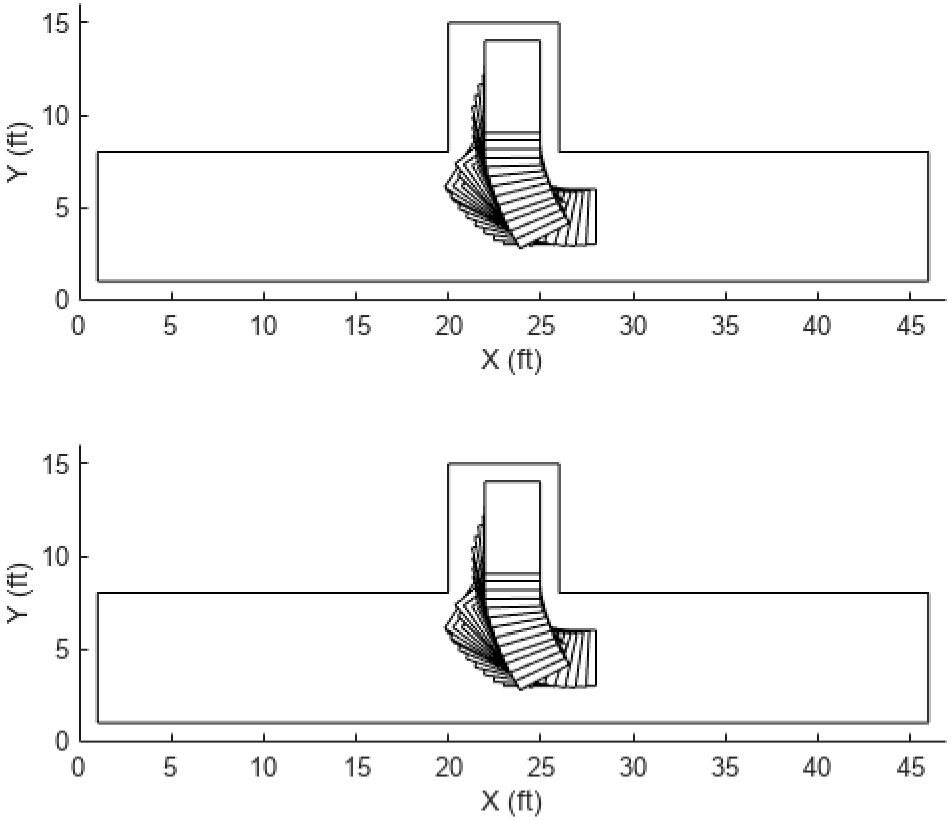
Add a fuzzy logic controller block



Fuzzy Logic



Autonomous Parking Using Fuzzy Inference System



Summary

★★★ - best ★ - worst

	PID	ADRC	MPC	Reinforcement Learning	Fuzzy Logic
Handling nonlinearities, disturbances etc.	★	★★	★★★	★★★	★★
Ease of implementation	★★★	★★★	★★	★	★★
Hardware resources required	★★★	★★	★	★	★★★
When to use	Often the best first choice	In the presence of uncertain dynamics, disturbances and without detailed model	Complex MIMO systems with constraints / operating limits	When it is difficult to characterize dynamics and operating conditions	In systems that are easy to understand but hard to model

Based on MathWorks white paper: https://www.mathworks.com/campaigns/offers/next/field-oriented-control-techniques-white-paper.html

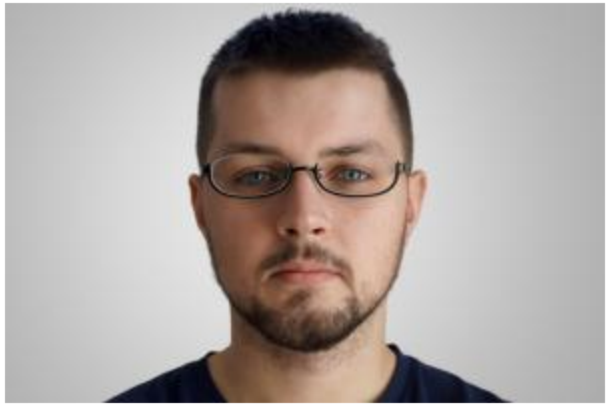
MATLAB EXPO

Warszawa | 4.06.2024

Paweł Siatka

Junior Application Engineer, ONT

pawel.siatka@ont.com.pl

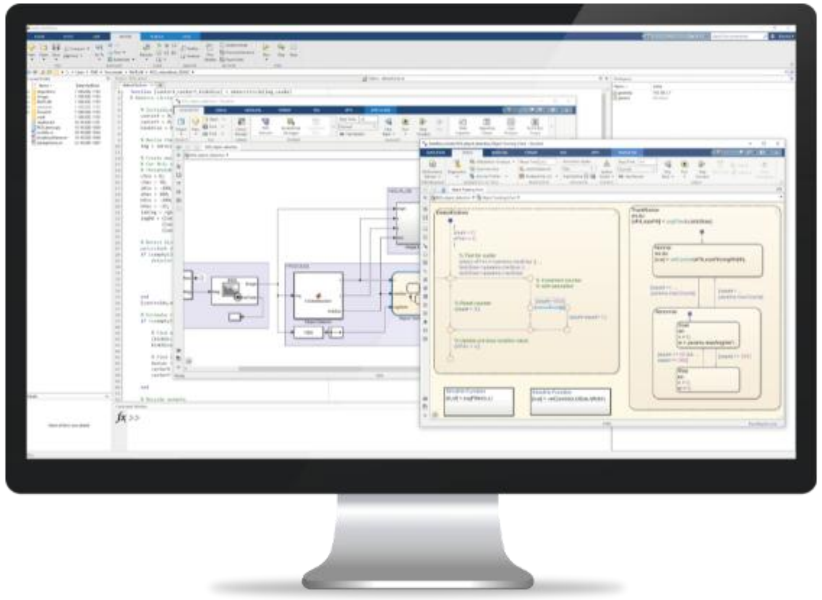


MathWorks®

Authorized Reseller

APPLICATIONS

- Robotics and Automation
- Computational Finance
- Autonomous Vehicles
- Electronics
- Artificial Intelligence
- Biomedical Engineering
- Systems Engineering and certification
- Power Electronics and Systems
- Communications and Radar Systems



Let's stay in touch

Oprogramowanie Naukowo-Techniczne sp. z o.o.
MATLAB and Simulink authorised reseller for Poland
ul. Pod Fortem 19, 31-302 Kraków, Poland | www.ont.com.pl