Counter-Example Guided Imitation Learning of Feedback Controllers From Temporal Logic Specification

Thao Dang[1], **Alexandre Donzé**[2], Inzemamul Haque[3], Nikolaos Kekatos[4], Indranil Saha[3]

[1] Univ. Grenoble Alpes, CNRS, Grenoble INP, VERIMAG, Grenoble, France,
[2] Decyphir SAS, Moirans, France
[3] Department of Computer Science and Engineering, IIT Kanpur, India,
[4] Aristotle University of Thessaloniki, Thessaloniki, Greece

March 7^{th}, 2024

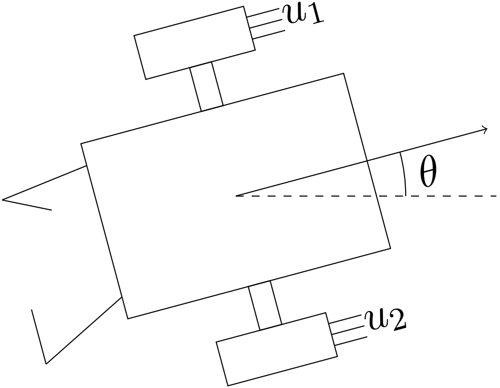
Counter-Example Guided Imitation Learning of Feedback Controllers From Temporal Logic Specification

Thao Dang[1], **Alexandre Donzé**[2], Inzemamul Haque[3], Nikolaos Kekatos[4], Indranil Saha[3]

[1] Univ. Grenoble Alpes, CNRS, Grenoble INP, VERIMAG, Grenoble, France,
[2] Decyphir SAS, Moirans, France
[3] Department of Computer Science and Engineering, IIT Kanpur, India,
[4] Aristotle University of Thessaloniki, Thessaloniki, Greece

March 7^{th}, 2024

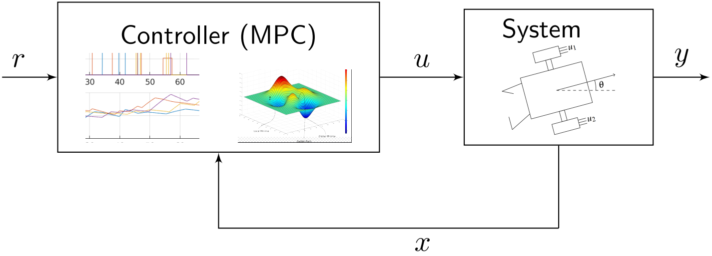
Control Problem



Flying Robot with two thrusters

Goal Go from x_0 to origin

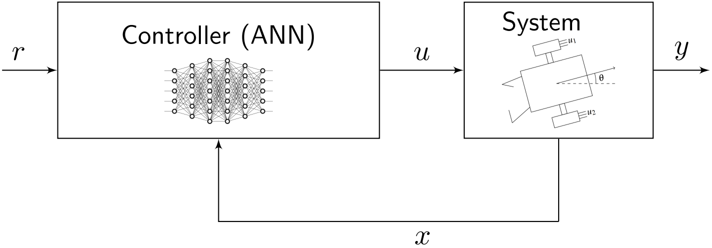
State Feedback Control



Issues

- MPC solves a NL optimization problem at each step
- Costly for online use
- Can we replace it with a NN (fast inference)?

Neural Network Control System

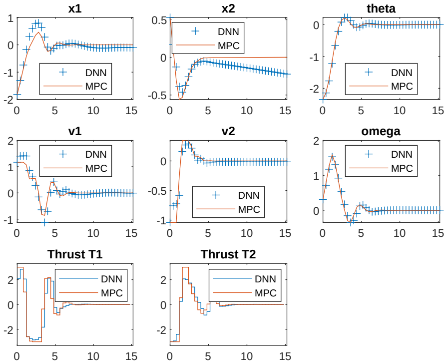


Supervised learning

- Simulate with MPC
- collect set of input/output for the NN

$$\mathcal{D} = \{(x, u), u = MPC(x)\}$$

Behavior Cloning Validation



Not So Good

Visible drift in x2, need more data ?

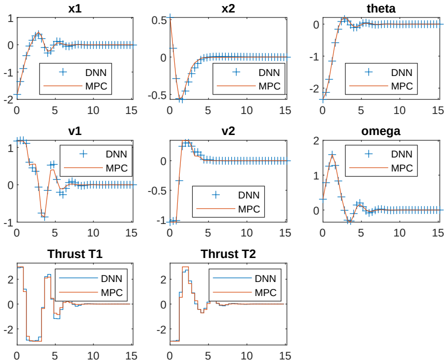
Data Aggregation Approach (DAgger)

$\mathcal{N}_0 \leftarrow \emptyset$, $\mathcal{D}_0 \leftarrow \emptyset$, $k \leftarrow 1$
repeat
$\quad \mathcal{D}_k \leftarrow \texttt{getNewData}\left(\mathcal{N}_{k-1}, \mathcal{D}_{k-1}\right)$
$\quad \mathcal{N}_k \leftarrow \texttt{Train}\left(\mathcal{D}_k\right)$
$\quad k \leftarrow k+1$
until $k > k_{max}$
return "Best" $\mathcal{N}_i$ for $i \in 1, \ldots, k_{max}$

Main Idea

New data is obtained by simulating with MPC from states visited by $\mathcal{N}_{k-1}$

DAgger Results

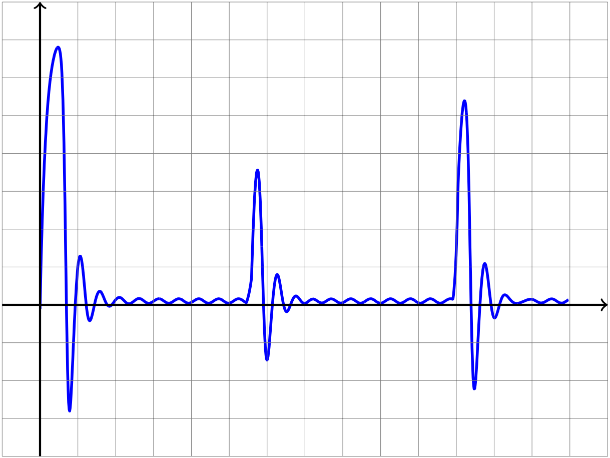


Looks better but...

- Can we trust one simulation to evaluate $\mathcal{N}_i$?
- Did we need 140 iterations ?

Outline

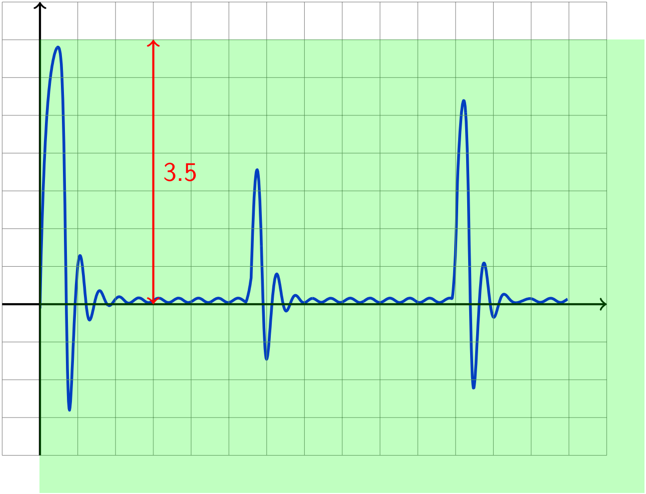
Temporal Logics Requirements (STL)



Temporal Logics Requirements (STL)

The signal is never above 3.5

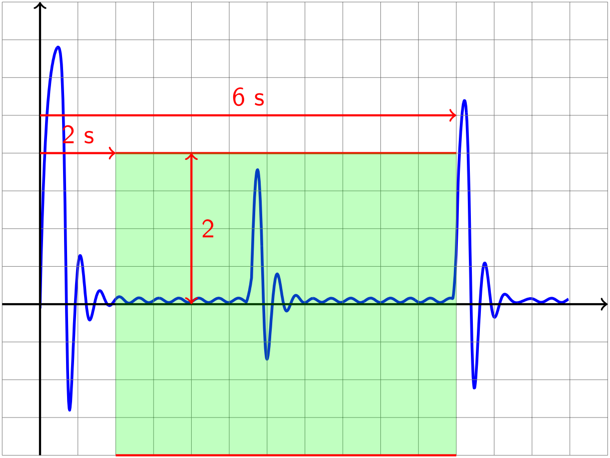
$$\varphi := \text{ alw } (x[t] < 3.5)$$



Temporal Logics Requirements (STL)

Between 2s and 6s the signal is between -2 and 2

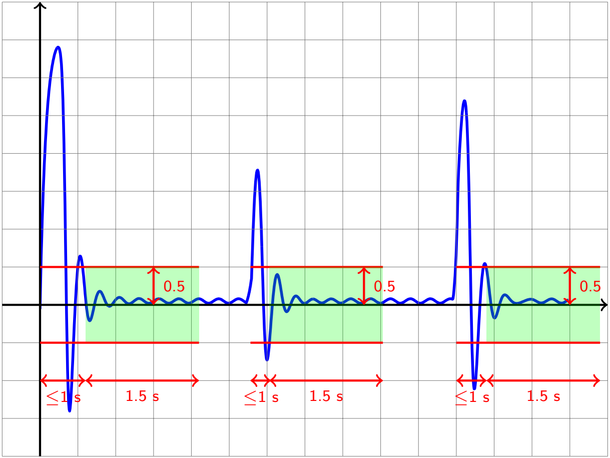
$$\varphi := \ \mathsf{alw}_{[2,6]} \ \ (|x[t]| < 2)$$



Temporal Logics Requirements (STL)

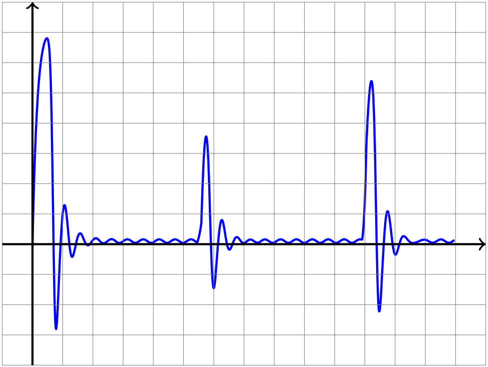
Always $|x|>0.5 \Rightarrow$ *after 1 s,* $|x|$ *settles under 0.5 for 1.5 s*

$$\varphi := \text{ alw}(x[t] > .5 \rightarrow \text{ ev}_{[0,.6]} \;\; (\text{ alw}_{[0,1.5]}\; x[t] < 0.5))$$



Parametric-STL Formulas

STL formula where numeric constants are left unspecified.

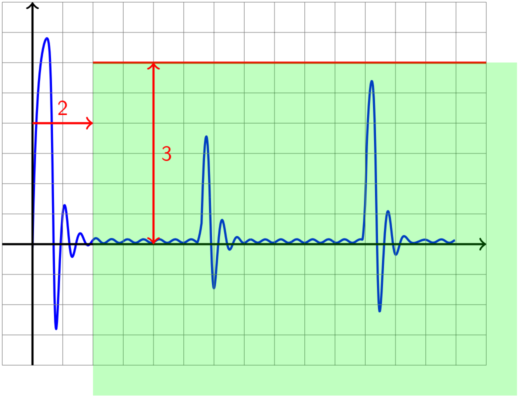


Parametric-STL Formulas

STL formula where numeric constants are left unspecified.

"After 2s, the signal is never above 3"

$$\varphi := \ \mathsf{ev}_{[2,\infty]} \ \ (x[t] < 3)$$

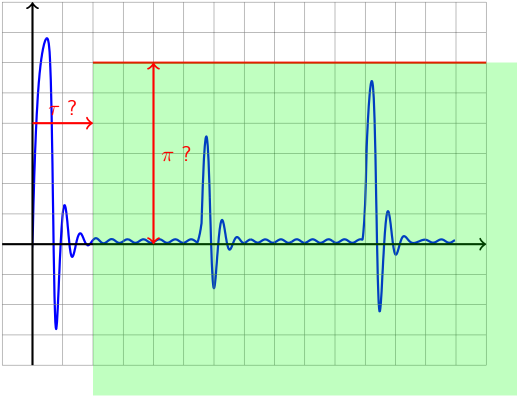


Parametric-STL Formulas

STL formula where numeric constants are left unspecified.

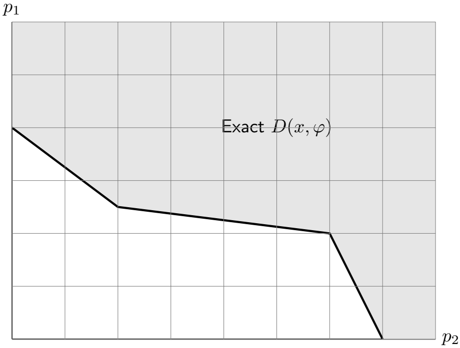
"After τ s, the signal is never above π"

$$\varphi := \mathsf{ev}_{[\tau,\infty]} \ (x[t] < \pi)$$



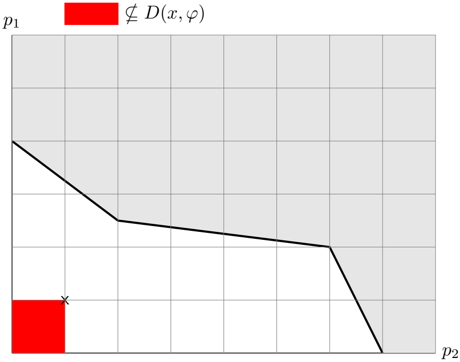
PSTL Validity Domains

- The validity domain D of φ and x is the set of valuations v s.t. $x \models \varphi(v)$
- In case of monoticity, ∂D has the structure of a Pareto front



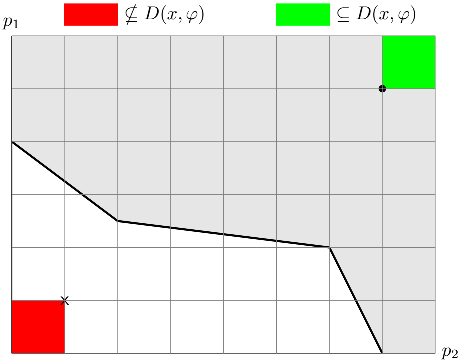
PSTL Validity Domains

- The validity domain D of φ and x is the set of valuations v s.t. $x \models \varphi(v)$
- In case of monoticity, ∂D has the structure of a Pareto front



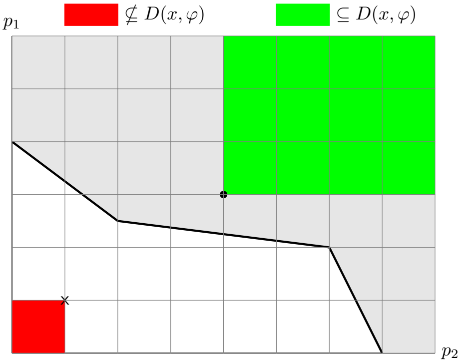
PSTL Validity Domains

- The validity domain D of φ and x is the set of valuations v s.t. $x \models \varphi(v)$
- In case of monoticity, ∂D has the structure of a Pareto front



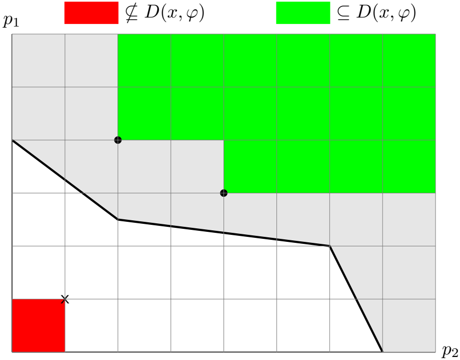
PSTL Validity Domains

- The validity domain D of φ and x is the set of valuations v s.t. $x \models \varphi(v)$
- In case of monoticity, ∂D has the structure of a Pareto front



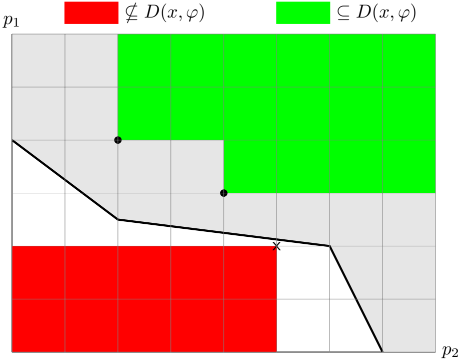
PSTL Validity Domains

- The validity domain D of φ and x is the set of valuations v s.t. $x \models \varphi(v)$
- In case of monoticity, ∂D has the structure of a Pareto front



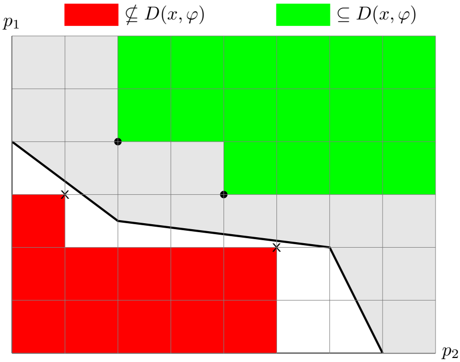
PSTL Validity Domains

- The validity domain D of φ and x is the set of valuations v s.t. $x \models \varphi(v)$
- In case of monoticity, ∂D has the structure of a Pareto front



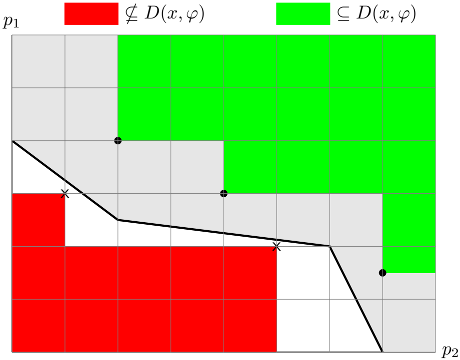
PSTL Validity Domains

- The validity domain D of φ and x is the set of valuations v s.t. $x \models \varphi(v)$
- In case of monoticity, ∂D has the structure of a Pareto front

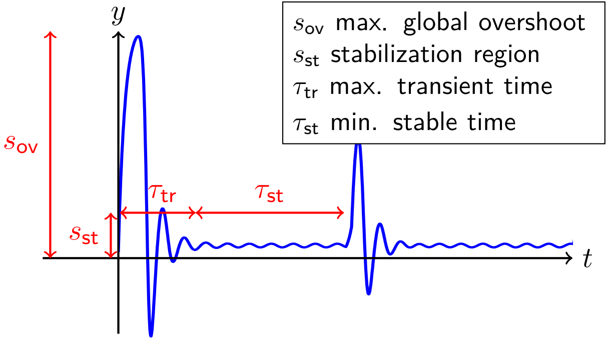


PSTL Validity Domains

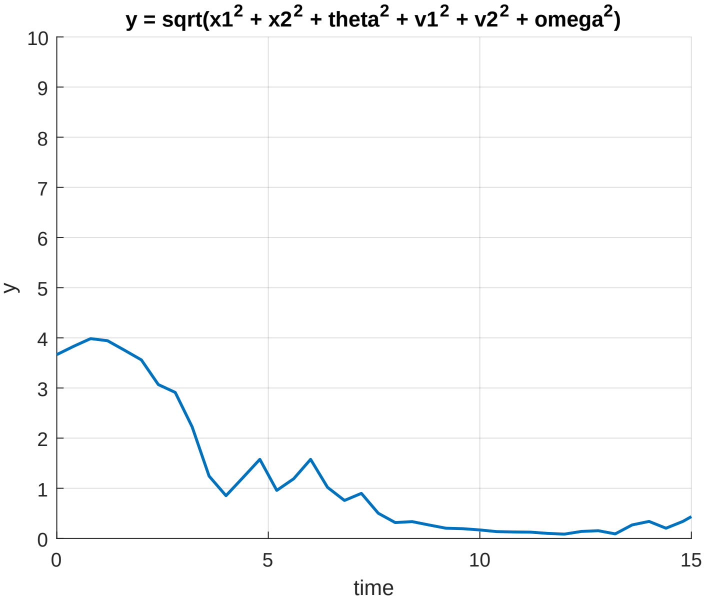
- The validity domain D of φ and x is the set of valuations v s.t. $x \models \varphi(v)$
- In case of monoticity, ∂D has the structure of a Pareto front



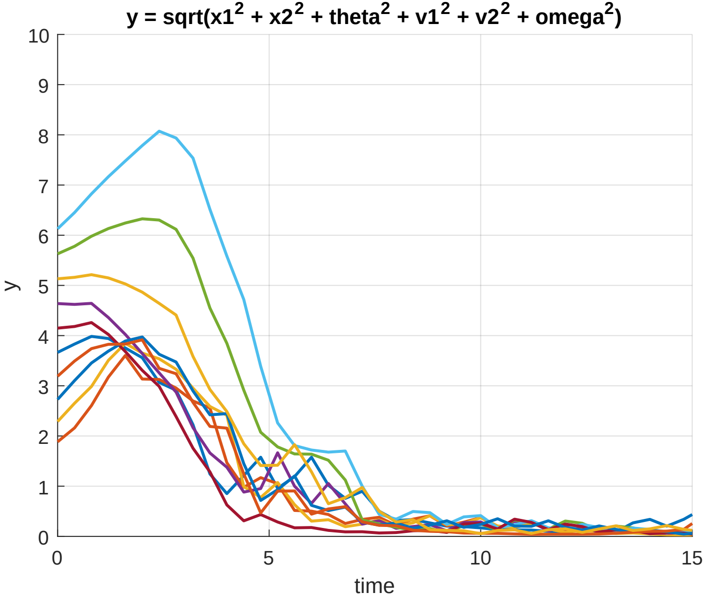
PSTL for Stabilization



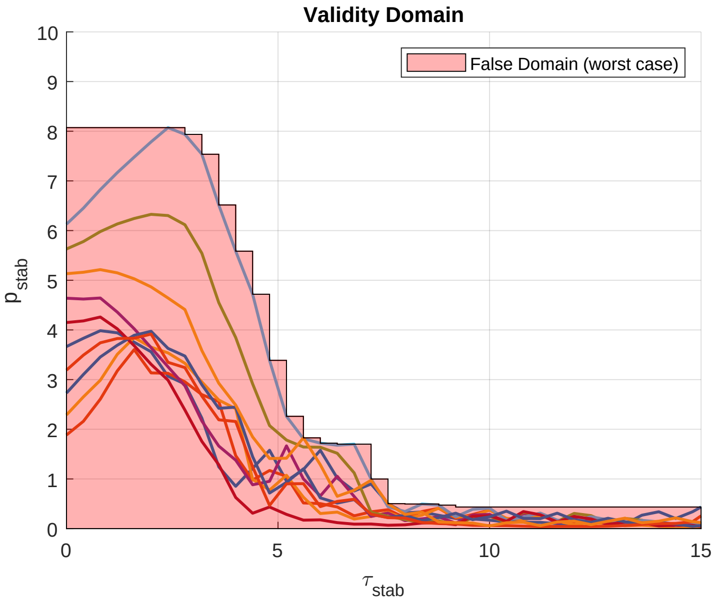
Flying Paretos



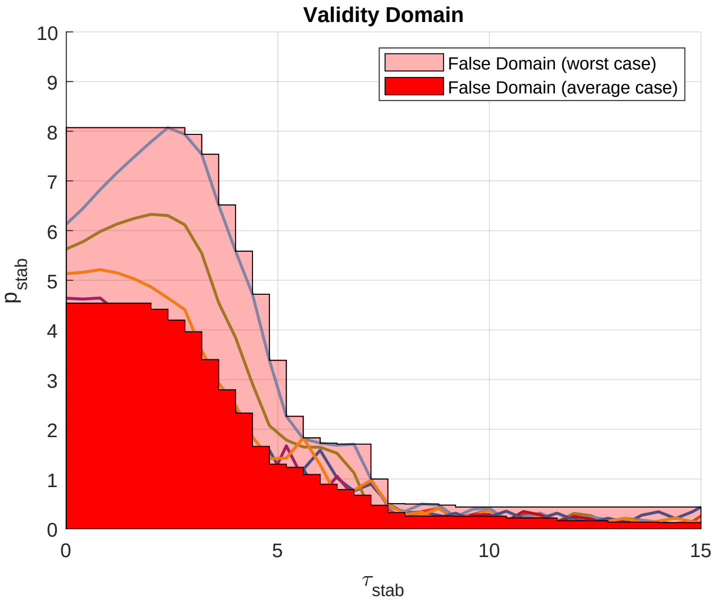
Flying Paretos



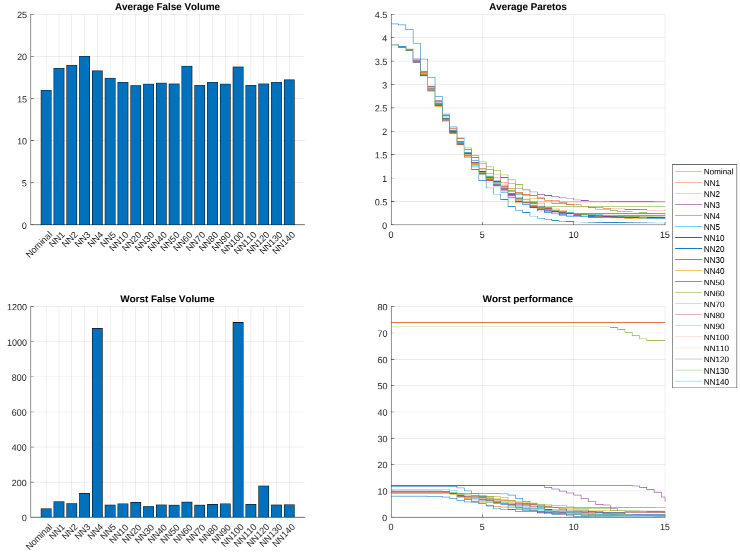
Flying Paretos



Flying Paretos



DAgger "Progress"



Outline

How can we improve the data collection process?

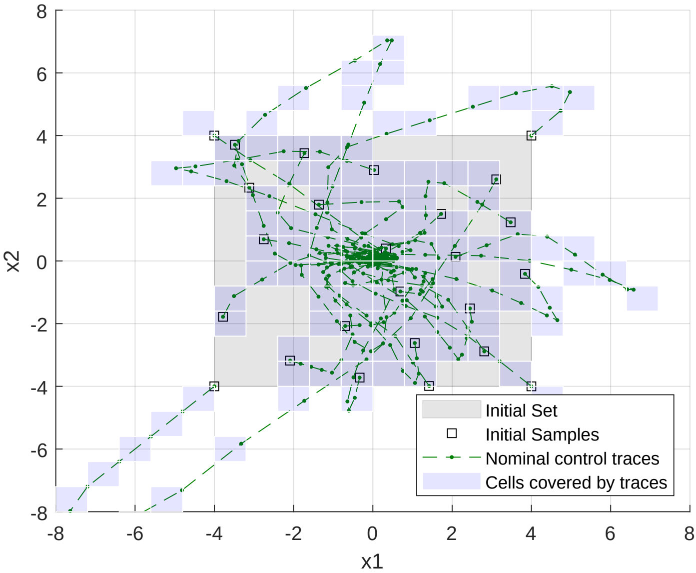
It's the data stupid

- Idea 1: Better control over coverage (sample efficiency)
- Idea 2: Generate data from counter-examples

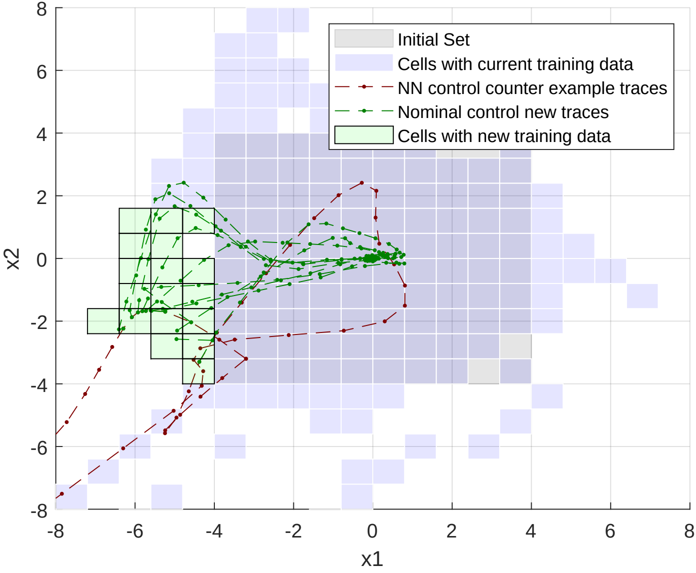
New Data Acquisition Procedure

```
procedure GETNEWDATA(N, D)
    if D is ∅ then                                        ▷ Sample traces from initial set
        InitSamples ← getInitSamples()
        InitTraces ← simNominal(InitSamples)
        D_new ← gridFilter(InitTraces)
        Status ← "New data available for training."
    else                                                  ▷ Search for counter-examples
        CexTraces ← falsify(N)
        if CexTraces ≠ ∅ then
            (D_new, Status) ← fixAndMerge(D, CexTraces)
        else                                              ▷ Success.
            D_new ← D
            Status ← "No counter-example found."
```

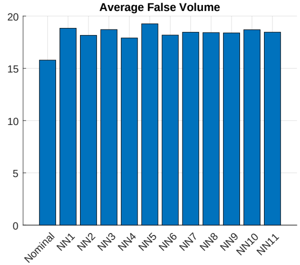
Grid Coverage - Initial Sampling

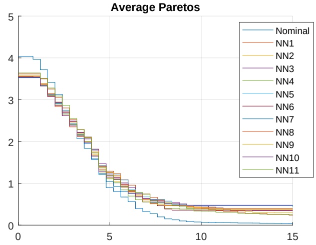


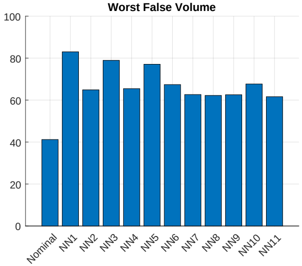
Counter-Example Data Extraction

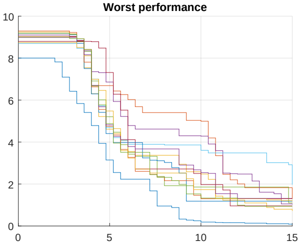


Results


Average False Volume
20
15
10
5
0
Nominal
NN1
NN2
NN3
NN4
NN5
NN6
NN7
NN8
NN9
NN10
NN11


Average Paretos
5
4
3
2
1
0
0
5
10
15
Nominal
NN1
NN2
NN3
NN4
NN5
NN6
NN7
NN8
NN9
NN10
NN11


Worst False Volume
100
80
60
40
20
0
Nominal
NN1
NN2
NN3
NN4
NN5
NN6
NN7
NN8
NN9
NN10
NN11


Worst performance
10
8
6
4
2
0
0
5
10
15

Outline

Discussion

More sensible stopping criterion and performance evaluation

- PSTL provides some control over what criterion to train for in priority
- Falsification stopping criterion gives some confidence over result

Discussion

More sensible stopping criterion and performance evaluation

- PSTL provides some control over what criterion to train for in priority
- Falsification stopping criterion gives some confidence over result

Approach "Probably" more sample efficient than vanilla DAgger

- "Converges" with less iteration and samples
- Evidence that performance is not affected by filtering (like quantization)

Discussion

More sensible stopping criterion and performance evaluation

- PSTL provides some control over what criterion to train for in priority
- Falsification stopping criterion gives some confidence over result

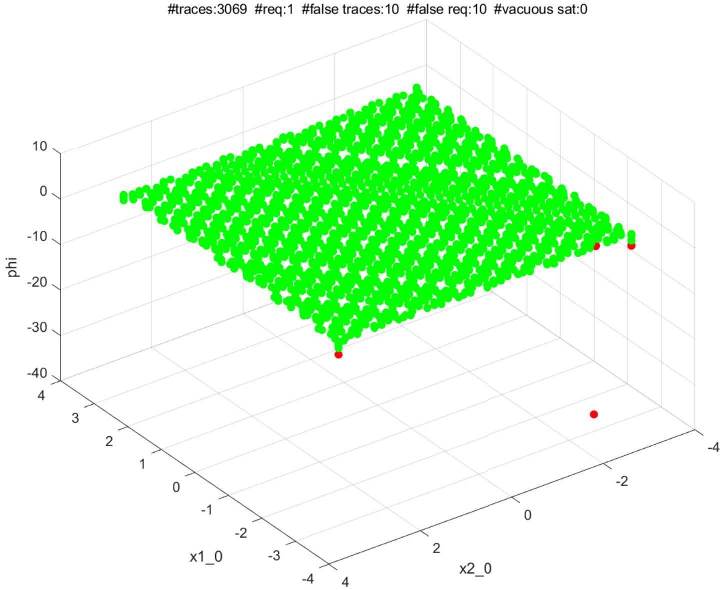
Approach "Probably" more sample efficient than vanilla DAgger

- "Converges" with less iteration and samples
- Evidence that performance is not affected by filtering (like quantization)

But...

- Reproducibility is an issue. High variability even with same training data
- Performance still comparable to original `final_policy` provided in Mathworks examples, though...

Mathworks' Policy Falsification



Personal Conclusion

- Flying robot actually very unstable,
- I now doubt it can be robustly solved with pure NN state feedback controller
- Our method seem to work well with more well-behaved problems,
- More experiments in the making