# Adaptive Value Iteration

William S. Moses

6.832 Project Presentations
May 17, 2018

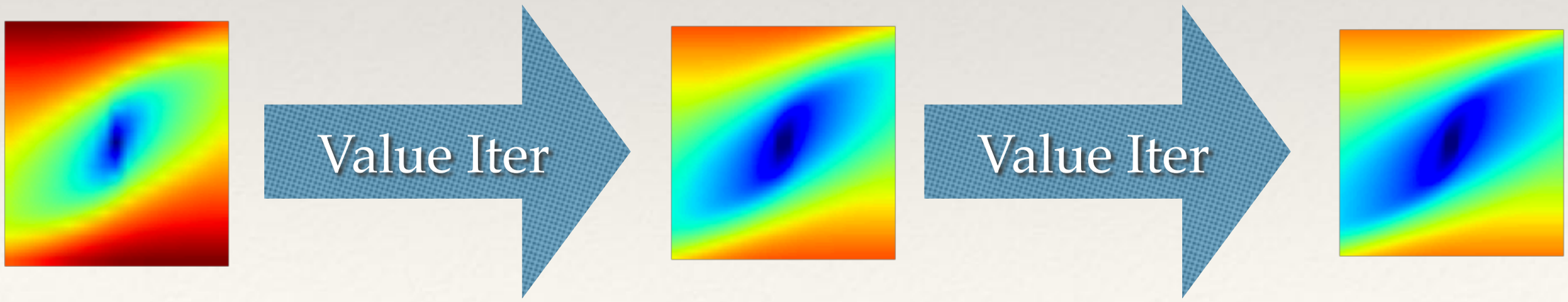
CSAIL
MIT COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE LABORATORY

# Value Iteration is Slow!

Idea!

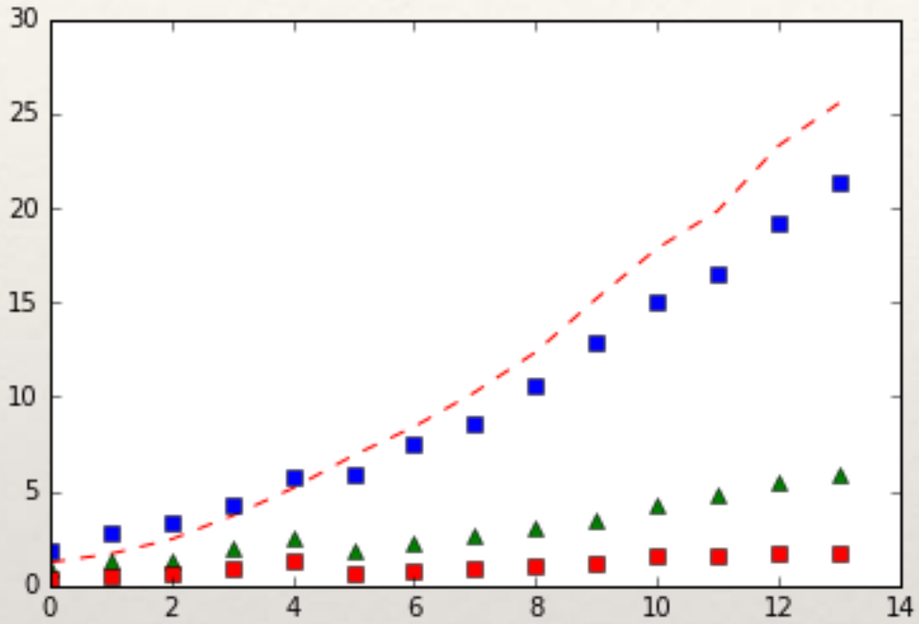
Re-use computation from lower resolution, and build up to high resolution!

Interpolate lower resolution up and use it instead of starting 0 state.



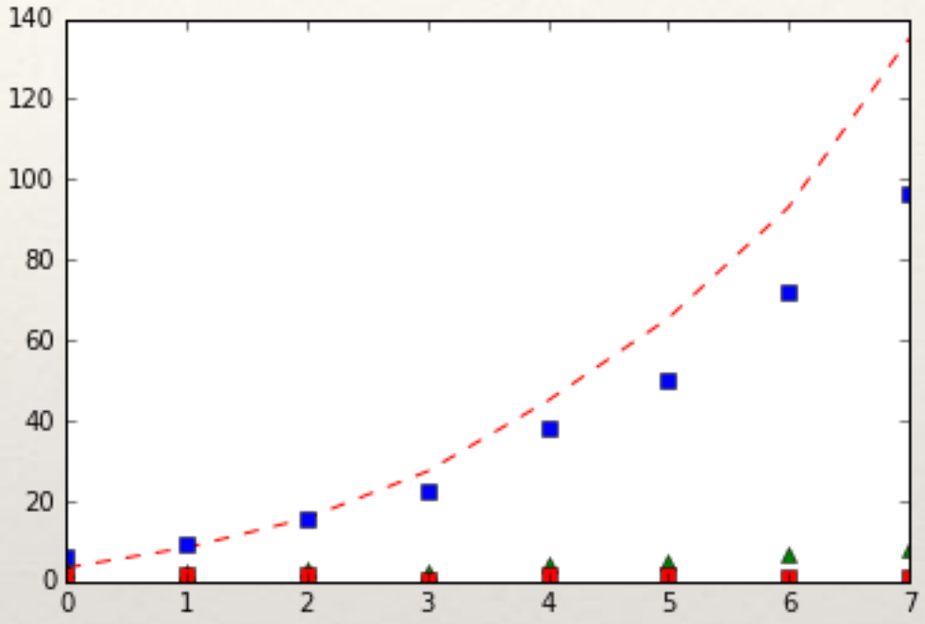
# Quadratic regulator

1-dimension




1/4: 12.9%
1/2: 34.0%
Full: 100.0%

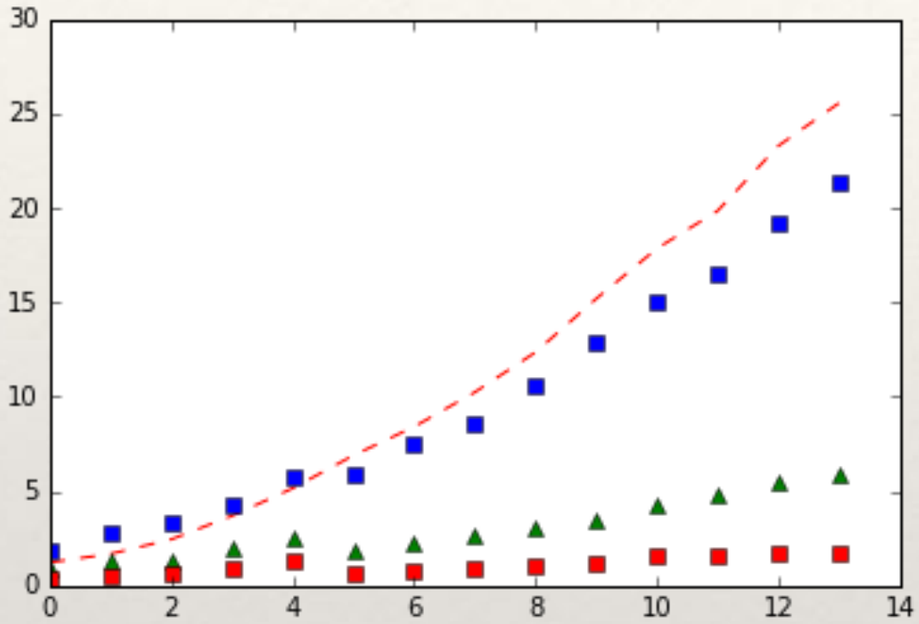
2-dimension




1/4: 4.5%
1/2: 14.0%
Full: 94.5%

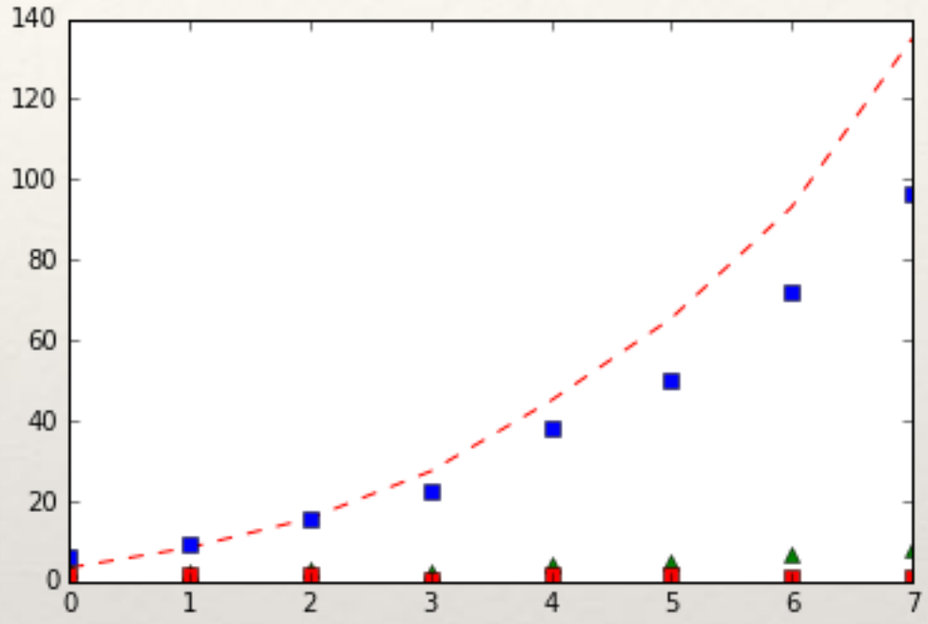
# Quadratic regulator

1-dimension




1/4: 12.9%
1/2: 34.0%
Full: 100.0%

2-dimension

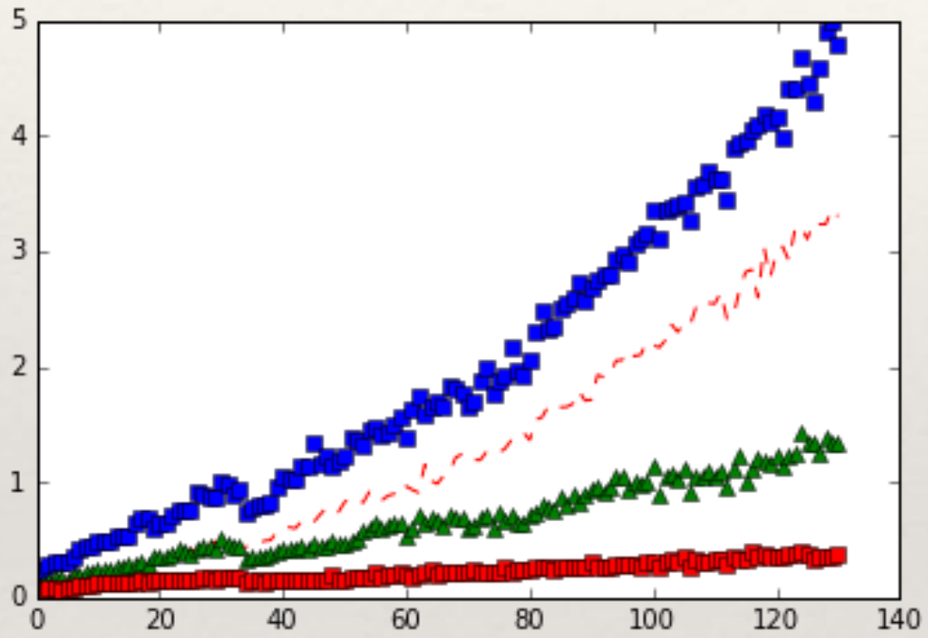


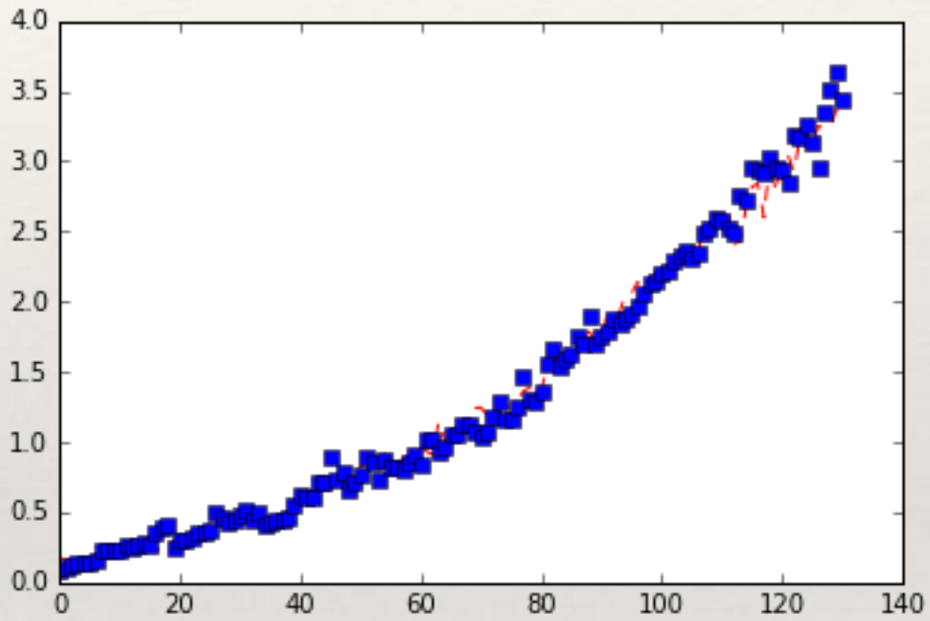

1/4: 4.5%
1/2: 14.0%
Full: 94.5%

**Can progressively add resolution, without adding time (and often save time)**

**High-dimensions => More Savings!**

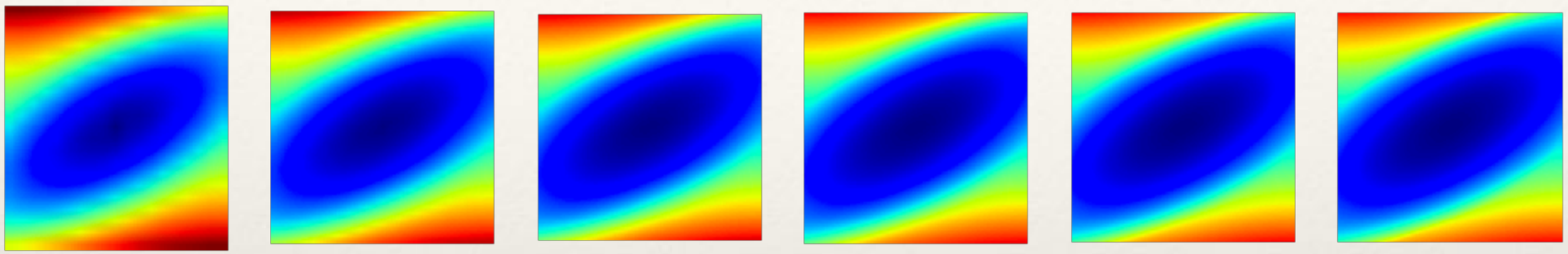
# Min-time (1D)

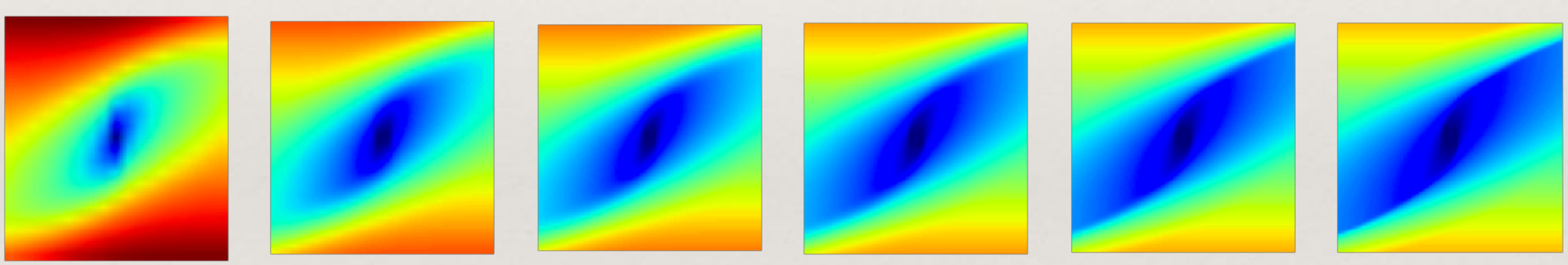



**Sometimes resolution scaling HURTS performance!**

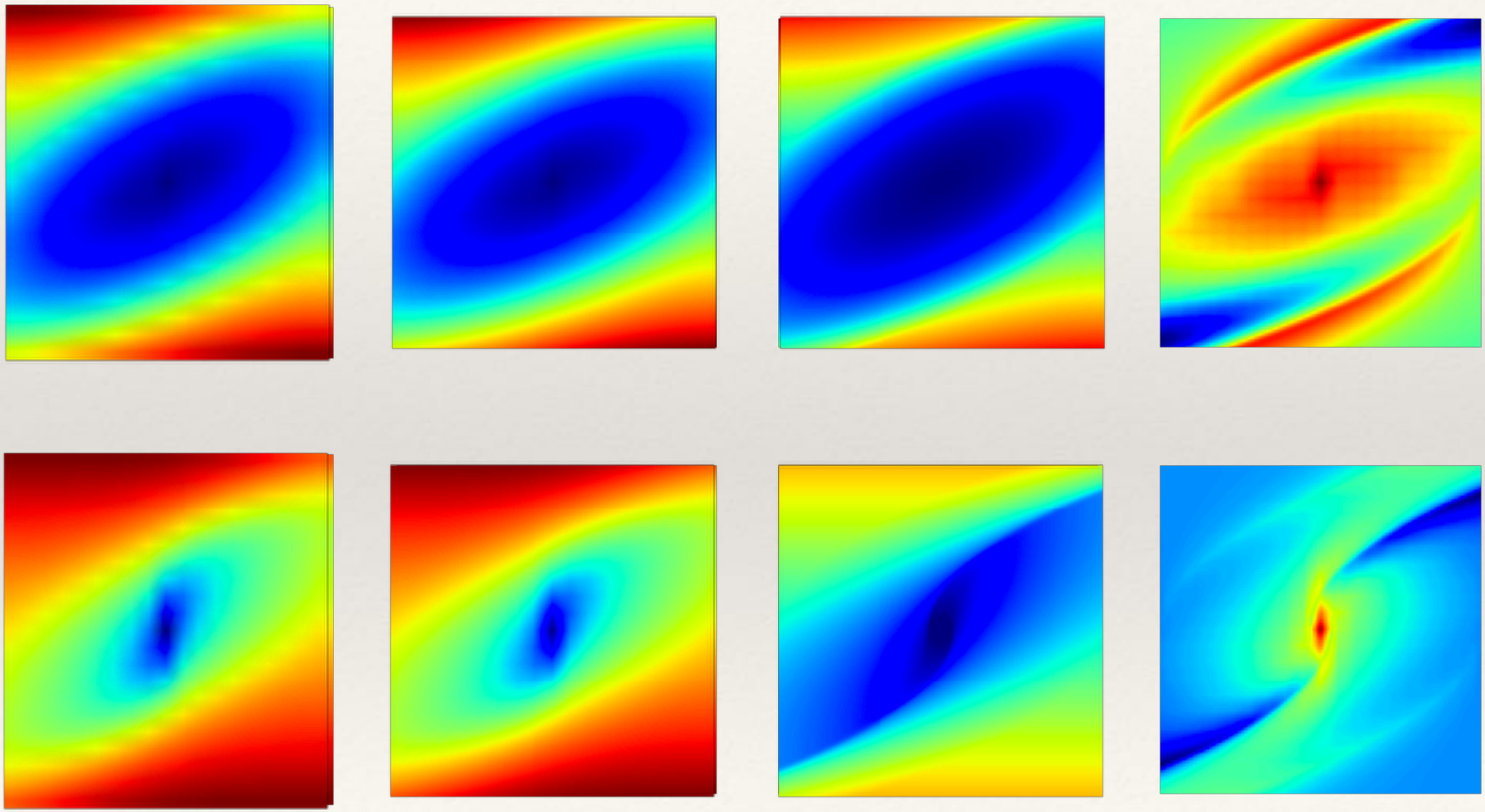
# Quadratic Cost vs Min Time





**Upscaling the quadratic regulator causes minor changes, whereas upscaling min-time causes global changes (and thus must globally recompute)**

# Quadratic Cost vs Min Time



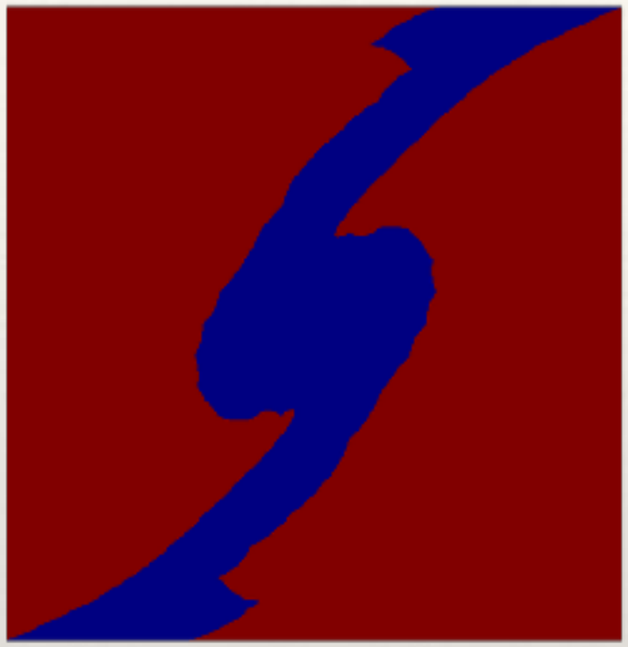
# Use High-Resolution Sparingly

Run high-resolution only on areas where needed

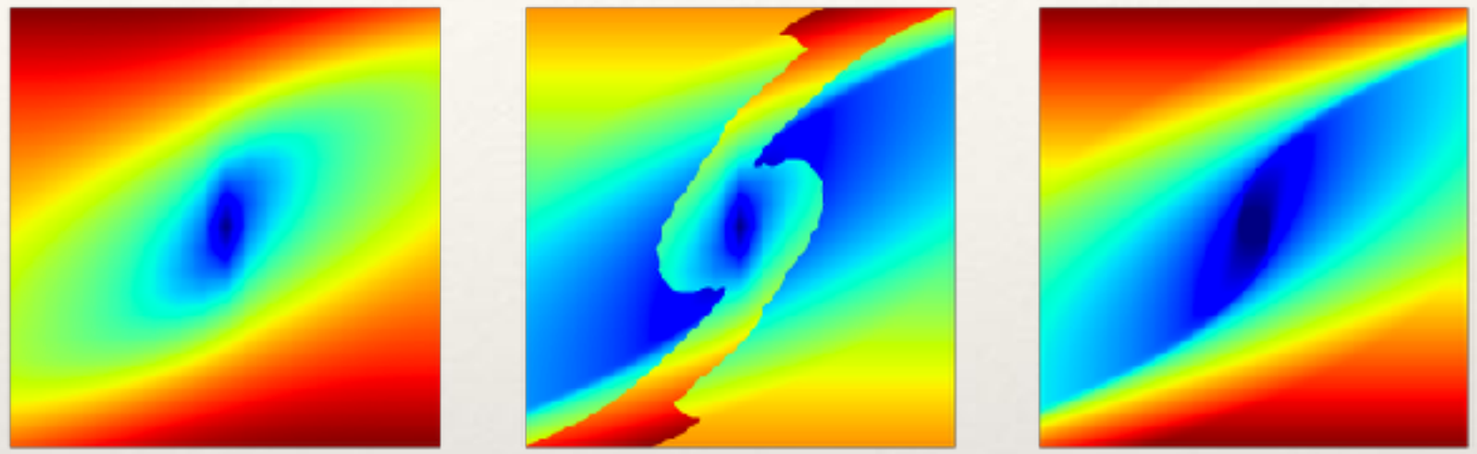
Plug the results of the high resolution, back into the low resolution Value Iteration.

Find places to use high-res, by calculating next resolution and looking at error.

# Bounded-Error Filter

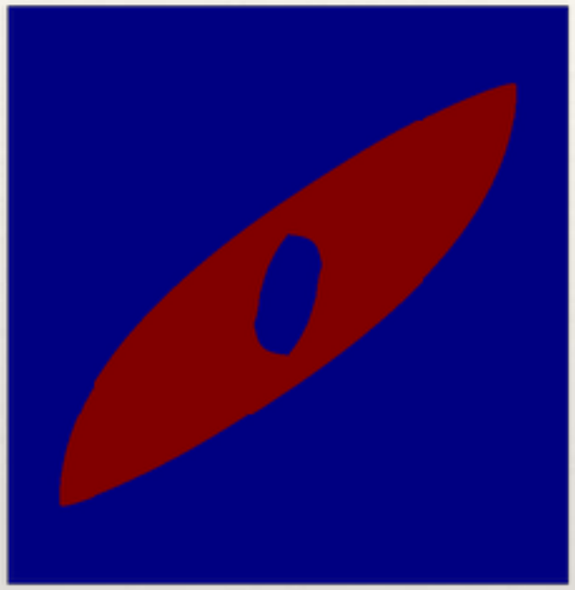


Bounded Error Filter

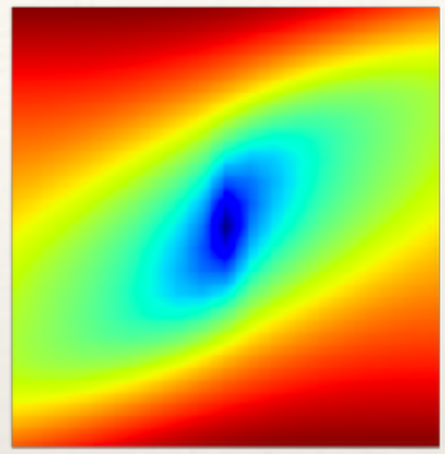


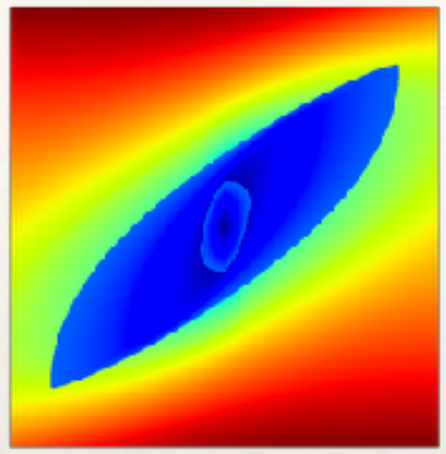
**Requires high-res for almost everything!**

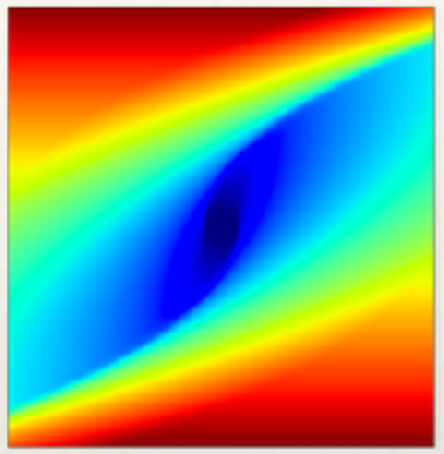
# Cost-Threshold Filter

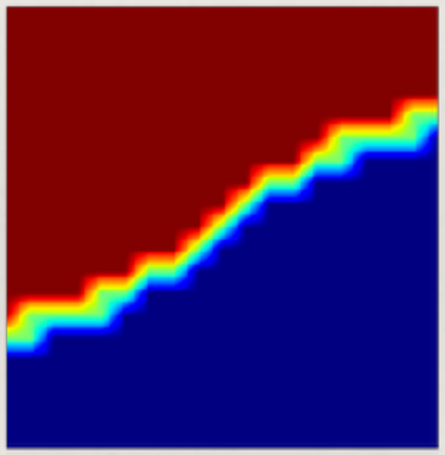


Cost Threshold Filter

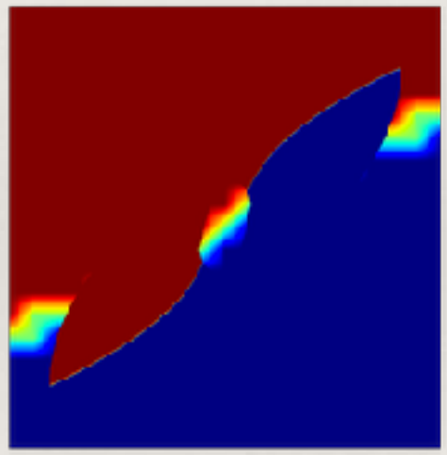




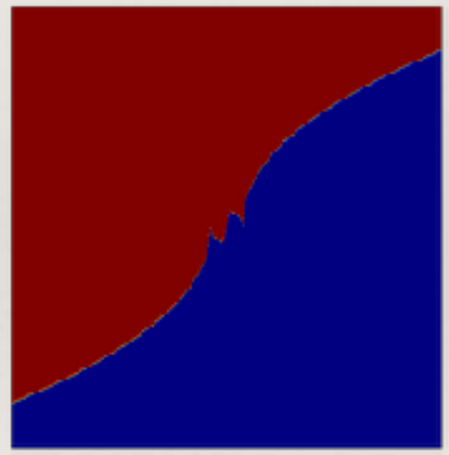




Low Res



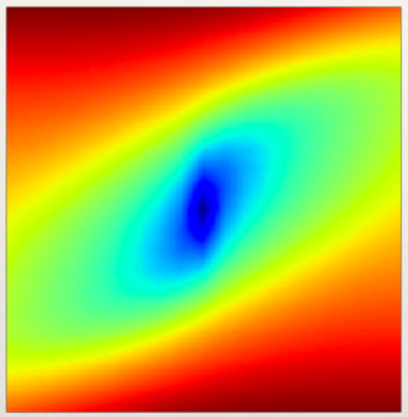
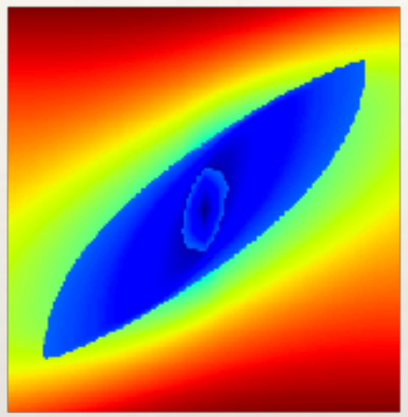
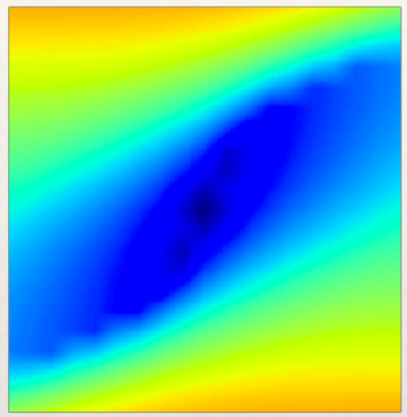
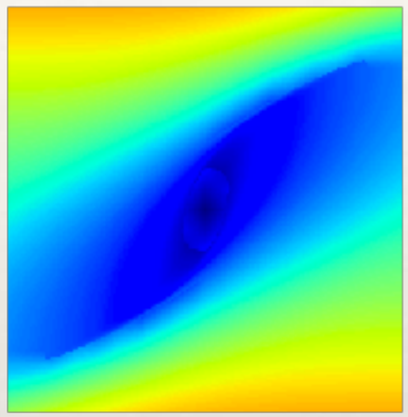
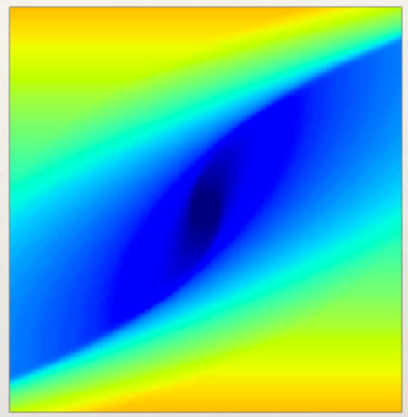
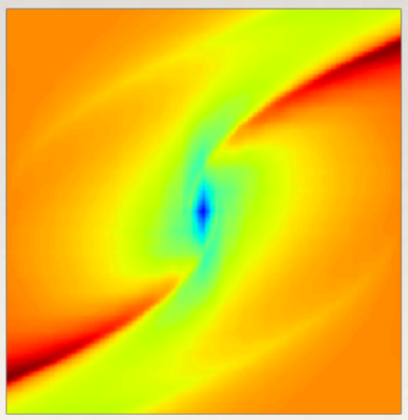
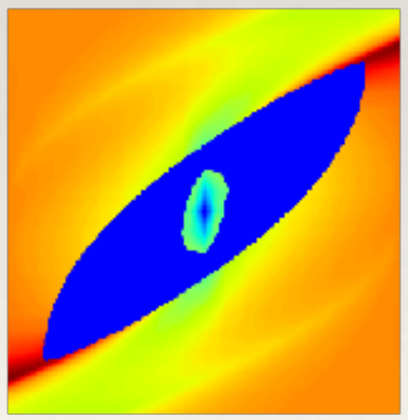
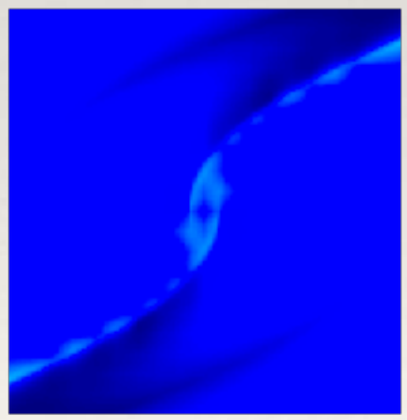
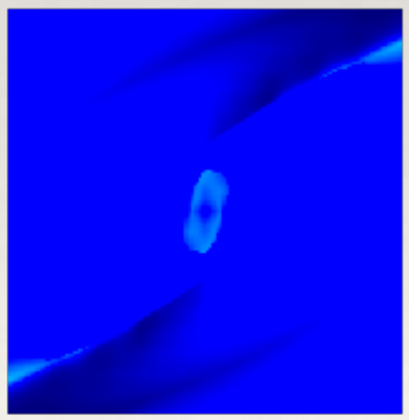
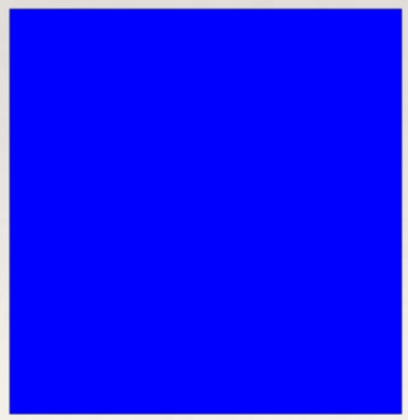
Mixed Res



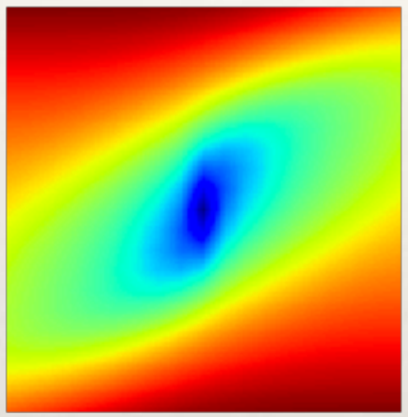
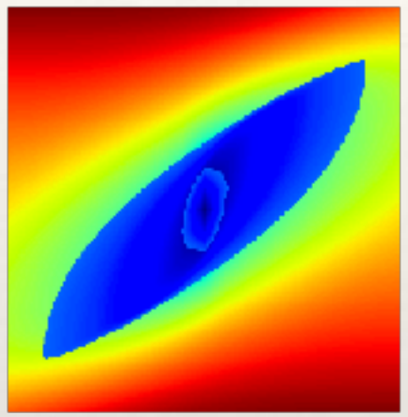
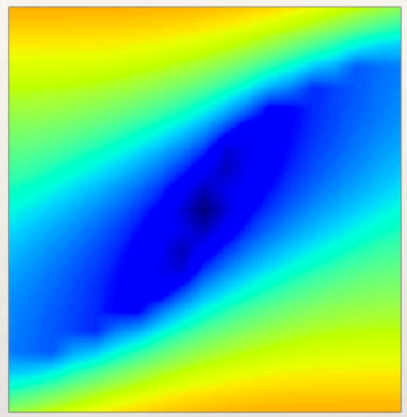
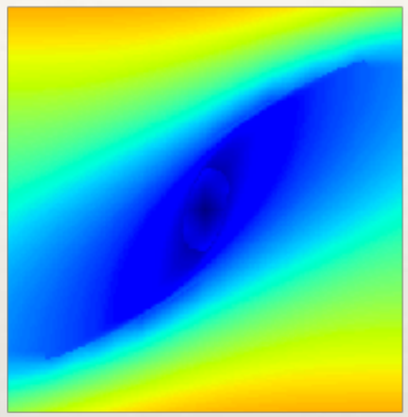
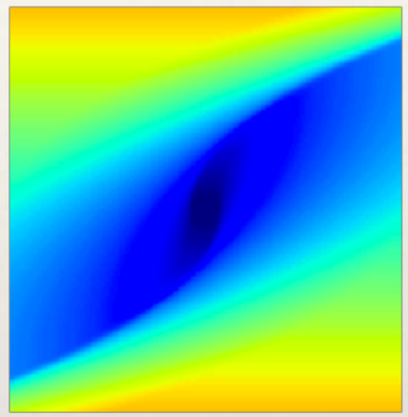
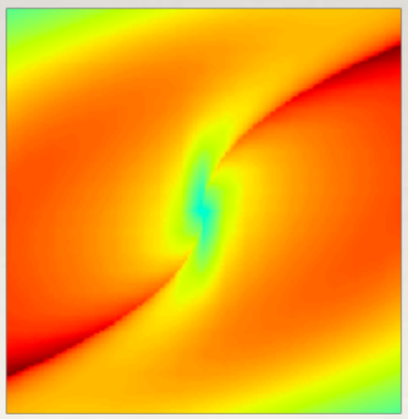
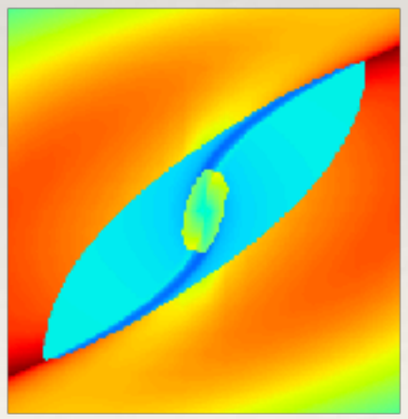
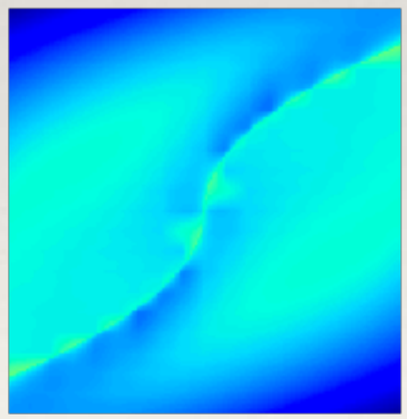
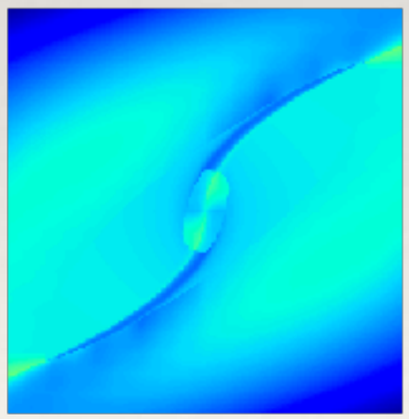
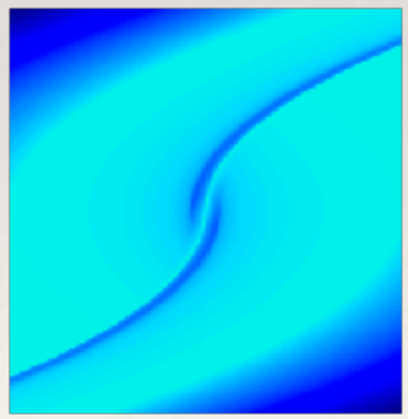
High Res

**Only compute the shortest path in detail!**

# Full Algorithm

| | Low Res | + Graft Hi Res | + Rerun low res | + Regraft | High Res |
|---|---|---|---|---|---|
| |  |  |  |  |  |
| Time: | 0.229 | 1.69 | 0.085 | 0.0 | 8.543 |
| Hi-Res Diff: |  |  |  |  |  |
| MSE: | 3.5 | 2.8 | 0.33 | 0.34 | 0.35 |

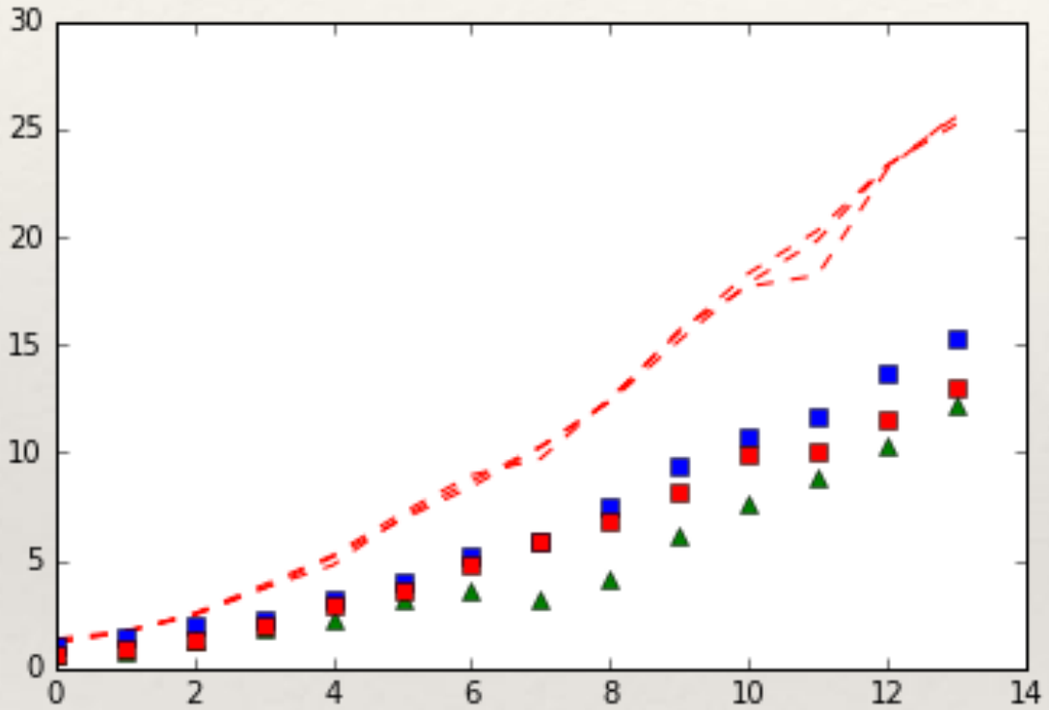
# Full Algorithm

| | Low Res | + Graft Hi Res | + Rerun low res | + Regraft | High Res |
|---|---|---|---|---|---|
| |  |  |  |  |  |
| Time: | 0.229 | 1.69 | 0.085 | 0.0 | 8.543 |
| True Diff: |  |  |  |  |  |
| MSE: | 3.5 | 2.8 | 0.33 | 0.34 | 0.35 |

# Conclusions

- You CAN progressively scan resolution, where there are non-global changes.
- Savings scale (getting more efficient) with dimensions.
- Grafting higher-resolution pieces allows progressive scan to work with global changes — and lets lower resolution have the same accuracy as high resolution, without a fully dense grid!
- Future Work: Automate detection/grafting pipeline
- Parallelizing Drake's Value Iteration gives ~7.67x speedup on 9 cores
- Extra 20% boost by using Tapir Parallel Compiler

# Backup

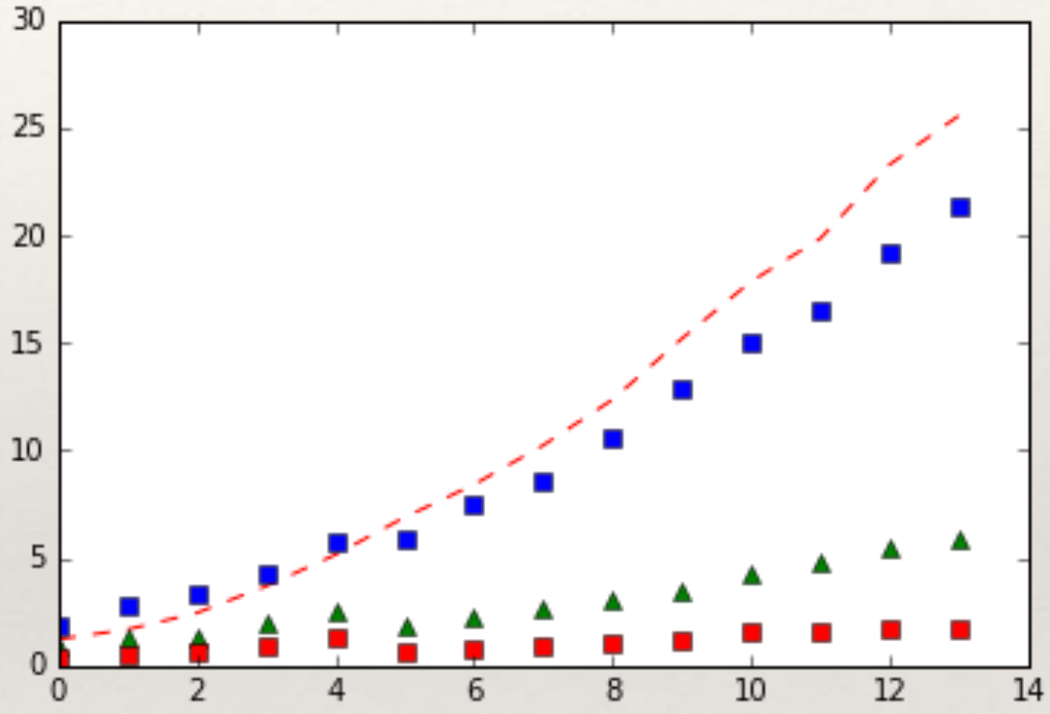



1d scaling

Blue: 2x,

Red: 1.3x,

Green 1.1x




Cumulative Cost scaling