Lecture 15: Decision-Making

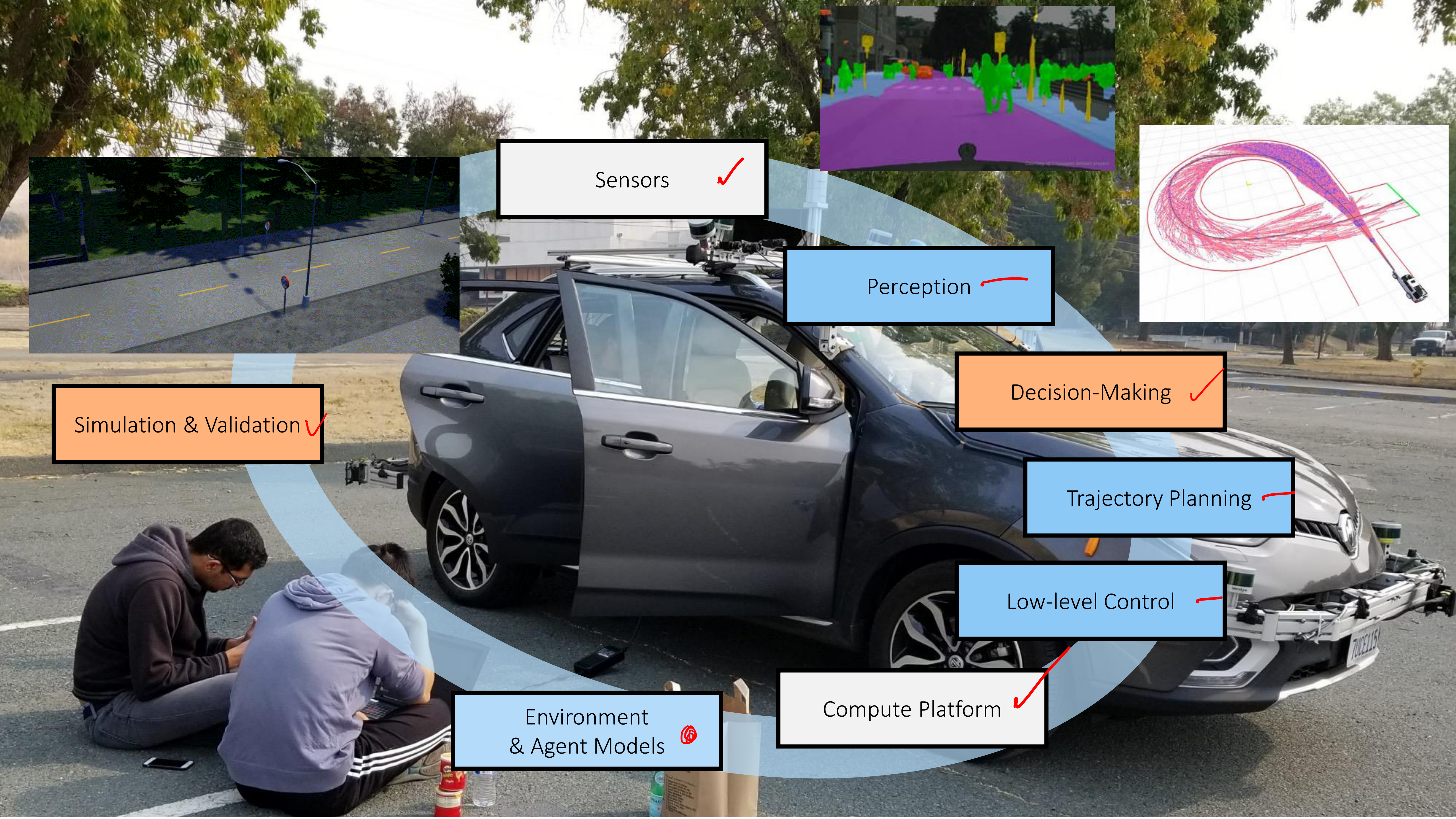
Professor Katie Driggs-Campbell

March 30, 2021

ECE484: Principles of Safe Autonomy

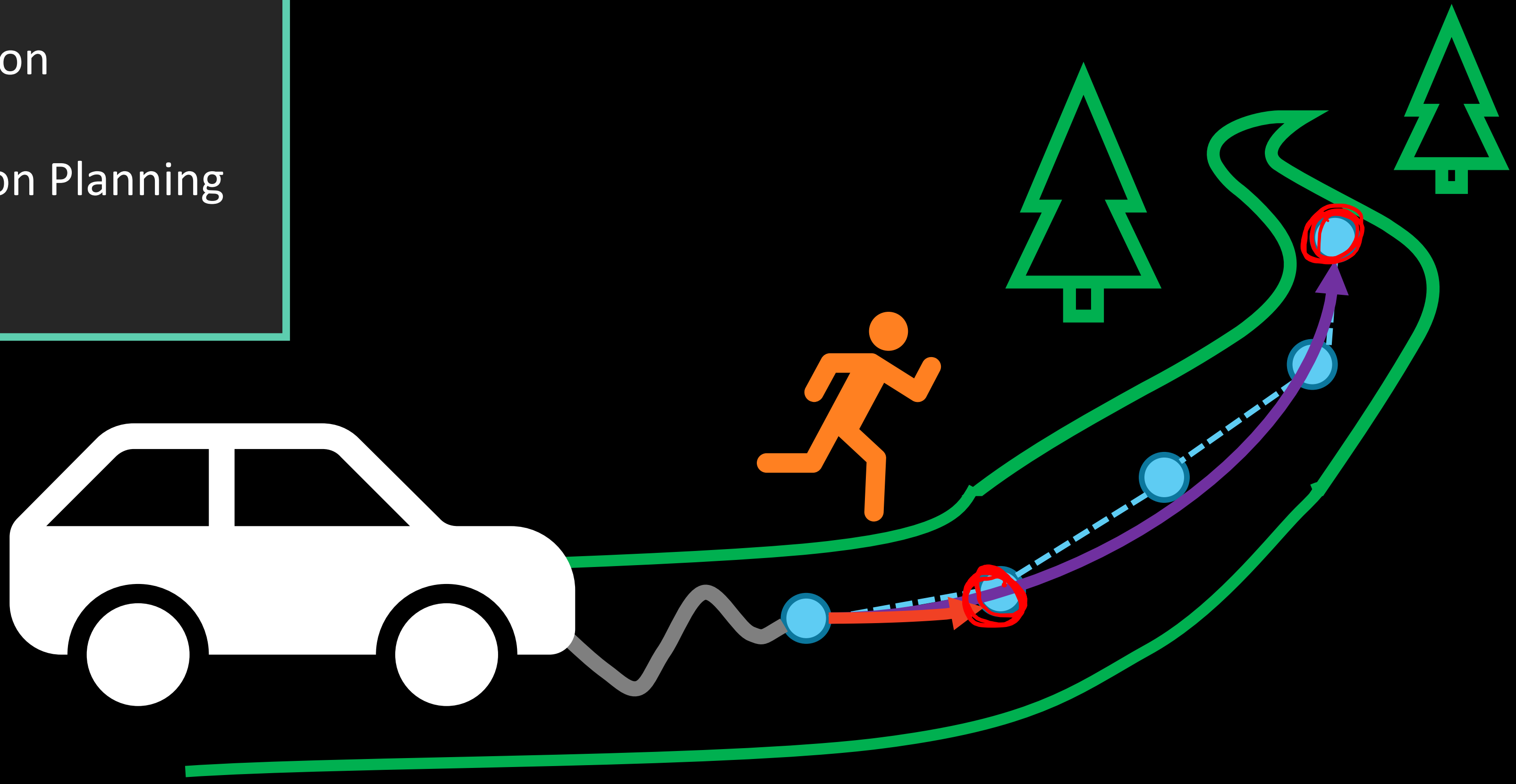
Administrivia

- Get started on MP3 and your project!
- Will go over oral exam protocol on Thursday
- Upcoming guest lecture attendance is “worth” double
- Participation grades (10% of total grade):
  - $P \approx \frac{1}{3}\{\text{attendance}\} + \frac{1}{3}\{\text{guest lecture participation}\} + \frac{1}{3}\{\text{team assessment}\}$
  - Stop by OH or make appointment to check attendance grade
  - For guest lecture participation, you can either send in questions beforehand (via Google forms to be posted on discord) or ask during class
  - For team assessment, we will post a Google form for collecting feedback on your teammates

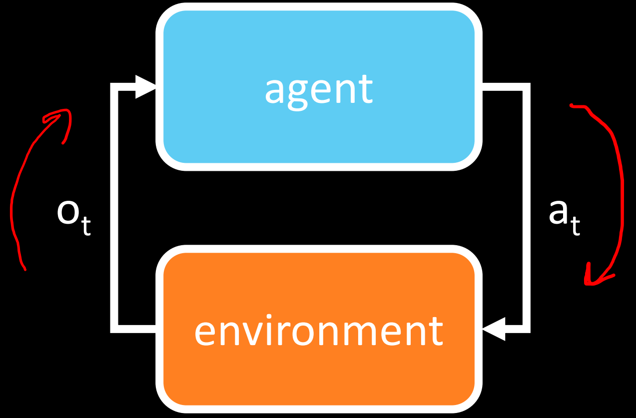


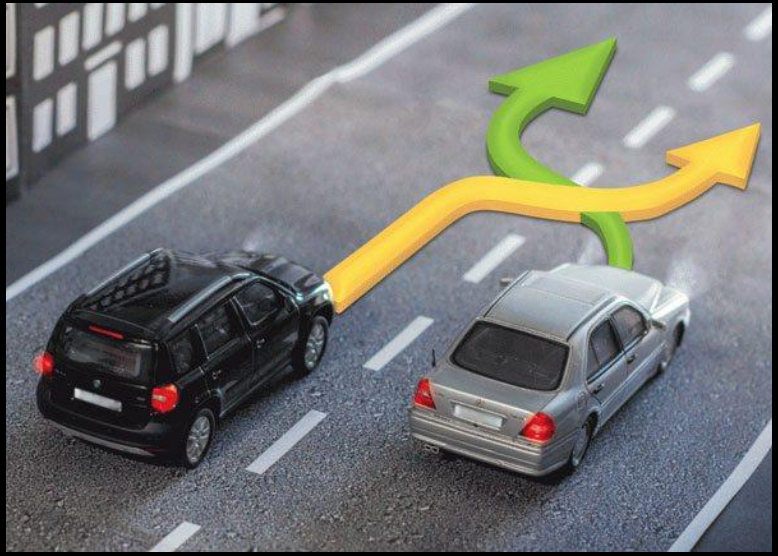
Sensors
Perception
Decision-Making
Trajectory Planning
Low-level Control
Compute Platform
Environment
& Agent Models
Simulation & Validation

- Vehicle Modeling
- Localization
- Detection & Recognition
- Control
- Trajectory/Path/Motion Planning
- ***Decision Making***
- Final topic: Safety!



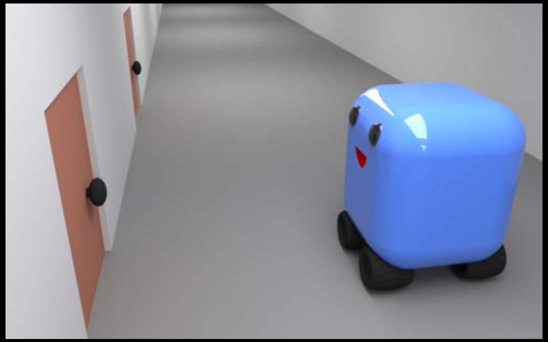
High-Level Decision-Making

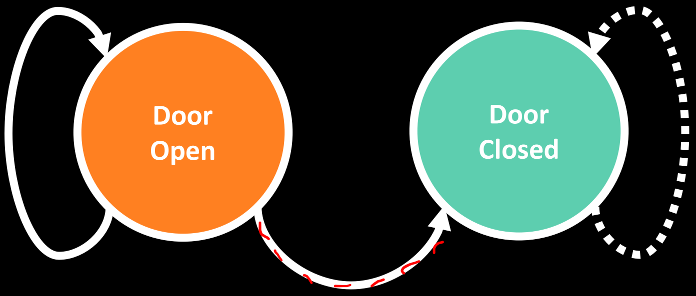


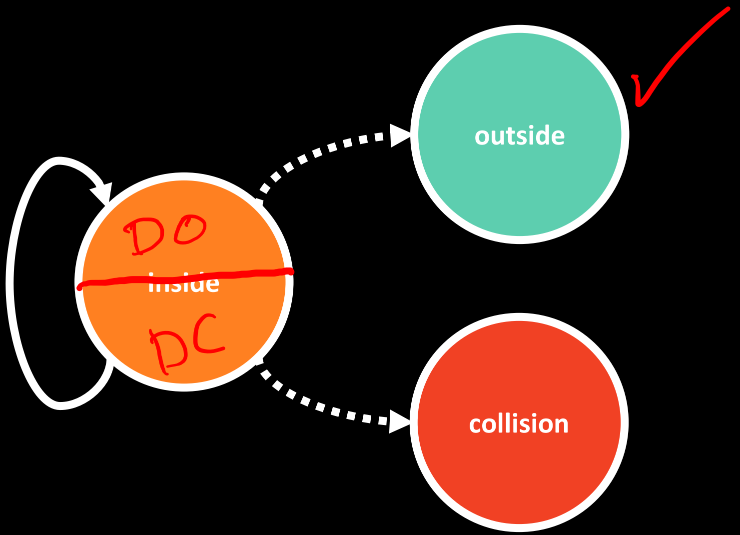


From Filtering to Decision-Making

Recall: Filtering allows us to recursively update our belief about some state







Decision-making helps us reason about what actions we should take

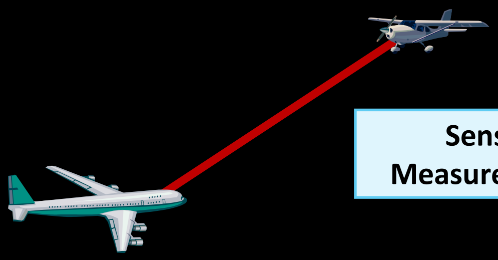
Today’s Plan

- Introduction to decision-making
- Markov Decision Processes
- MDP Policies and Value Iteration
- Simple Example
  - → Will post worked out complicated example

Today’s Plan

- Introduction to decision-making
- Markov Decision Processes
- MDP Policies and Value Iteration
- Simple Example

Traffic Alert and Collision Avoidance System (TCAS)



Sensor Measurements

```
IF (ITF.A LT G.ZTHR)
 THEN IF(ABS(ITF.VMD) LT G.ZTHR)
  THEN SET ZHIT;
  ELSE CLEAR ZHIT;
 ELSE IF (ITF.ADOT GE P.ZDTHR)
  THEN CLEAR ZHIT
  ELSE
   ITF.TAUV = -ITF.A/ITF.ADOT;
   IF (ITF.TAUV LT TVTHR AND
    ((ABS(ITF.VMD) LT G.ZTHR) OR
    (ITF.TAUV LT ITF.TRTRU))
    THEN SET ZHIT
    ELSE CLEAR ZHIT
IF (ZHIT EQ $TRUE AND
 ABS(ITF.ZDINT) GT P.MAXZDINT
 THEN CLEAR ZHIT
```

Resolution Advisory

Surveillance

Advisory Logic

Display

```
PROCESS Reversal_modeling;

    Default modeled separation for current RA is 0 if current RA is negative;
    Set own altitude and own rate to own tracked altitude and own tracked rate;

    IF (own does not follow his RAs)
        THEN  Model separation achieved assuming RA not followed;
                IF (current RA is a climb RA)
                    THEN  CLEAR flag indicating the sense of the RA after a reversal;
                    ELSE  SET flag indicating the sense of the RA after a reversal;
                IF (modeled separation achieved by continuing current RA greater than 1.2 *
                    P.CROSSTHR)
                    THEN  CLEAR reversal flag in ITF
        ELSE
        <Begin own is assumed to follow its RA>
                IF (current RA is positive)
                    THEN  model response to current RA;
                          <model maximum displayable rate for climb if current rate exceeds
                          maximum displayable rate or minimum displayable rate for descent if
                          current rate is less than minimum displayable rate>

                          IF (tracked response lags modeled response in RA direction AND
                            time since RA less than a parameter time AND
                            own's rate has not changed by more than P.MODEL_ZD since the
                             RA was first issued)
                             THEN  set own altitude and own rate to modeled altitude and rate
                                   for use in reversal modeling;
                          Model separation achieved by continuing current RA;
                Set delay time to greater of pilot delay time remaining for last advisory against a
                    new threat, and the pilot quick reaction time;

                IF (considering a reversal from a descend RA to a climb RA)
                    THEN  set own goal rate to greater of own tracked rate (or maximum
                          displayable rate, whichever is less) and nominal climb rate;
                    ELSE  IF (own too close to ground to descend)
                             THEN  set own goal rate to zero;
                             ELSE  set own goal rate to lesser of own tracked rate (or minimum
                                   displayable rate, whichever is greater) and nominal descent
                                   rate;
                IF (vertical chase, low VMD geometry was not the reason for considering
                    reversal)
                    THEN  IF (intruder causing crossing OR (intruder level AND own crossing
                             from above) OR intruder rate and own modeled rate are opposite in sign)
                             THEN  use outer rate bound to model intruder;
                             ELSE  use inner rate bound to model intruder;
                    ELSE  use intruder's tracked vertical rate to model intruder;
                CALL MODEL_SEP
                    IN (delay, goal rate, own altitude, own rate, acceleration response, sense after
                        reversal, intruder altitude, modeled intruder rate, ITF entry)
                    OUT (predicted separation for sense reversal);

                IF (Predicted separation for sense reversal is not positive OR
                   modeled separation achieved by continuing current RA GE G.ALIM)
                    THEN  CLEAR reversal flag in ITF;
        <End own is assumed to follow its RA>

END Reversal_modeling;
```



```
PROCESS Reversal_modeling;

    NOMINAL_SEP = 0;
    Z = G.ZOWN;
    ZD = G.ZDOWN;
    DELAY = 0;

    IF (G.OWN_FOLLOW EQ FALSE)
        THEN  CALL MODEL_SEP
                  IN (DELAY, ZD, Z, ZD, P.VACCEL, OWNTENT(7), ITF.ZINT, ITF.ZDINT, ITF entry)
                  OUT (NOMINAL_SEP);
              IF (OWNTENT(7) EQ $TRUE)
                  THEN  NEW_SENSE = $FALSE;
                  ELSE  NEW_SENSE = $TRUE;
              IF (NOMINAL_SEP GT 1.2 * P.CROSSTHR)
                  THEN  CLEAR ITF.REVERSE;
        ELSE
        <Begin own is assumed to follow its RA>
            IF (OWNTENT(5,6) EQ '00')
                THEN  DELAY = MAX(P.TV1 - (G.TCUR - G.TPOSRA), 0);
                      IF (OWNTENT(7) EQ $FALSE)
                          THEN  ZDGOAL = MAX(MIN(G.ZDOWN, P.MAXDRATE), P.CLMRT);
                          ELSE  ZDGOAL = MIN(MAX(G.ZDOWN, P.MINDRATE), P.DESRT);
                      CALL PROJECT_VERTICAL_GIVEN_ZDGOAL
                          IN ((G.TCUR - G.TPOSRA), G.ZTV, G.ZDTV, ZDGOAL, P.TV1, P.VACCEL)
                          OUT (ZPROJ, ZDPROJ);
                      IF  (((OWNTENT(7) EQ $FALSE AND ZPROJ GT G.ZOWN AND
                          (G.ZDOWN GE G.ZDTV - P.MODEL_ZD)) OR
                          (OWNTENT(7) EQ $TRUE AND ZPROJ LT G.ZOWN AND
                          (G.ZDOWN LE G.ZDTV + P.MODEL_ZD))) AND
                          G.TCUR – G.TPOSRA LT P.MODEL_T)
                          THEN  Z = ZPROJ;
                                ZD = ZDPROJ;
                      CALL MODEL_SEP
                          IN (DELAY, ZDGOAL, Z, ZD, P.VACCEL, OWNTENT(7),
                              ITF.ZINT, ITF.ZDINT, ITF entry)
                          OUT (NOMINAL_SEP);
            IF (OWNTENT(7) EQ $TRUE)
                THEN  NEW_SENSE = $FALSE;
                ELSE  NEW_SENSE = $TRUE;
            DELAY = MAX(P.TV1 - (G.TCUR - G.TLASTNEWRA), P.QUIKREAC);

            IF (NEW_SENSE EQ $FALSE)
                THEN  ZDGOAL = MAX(P.CLMRT, MIN(G.ZDOWN, P.MAXDRATE));
                ELSE  IF (G.NODESCENT EQ $TRUE)
                          THEN  ZDGOAL = 0;
                          ELSE  ZDGOAL = MIN(P.DESRT, MAX(G.ZDOWN,          P.MINDRATE));
            IF (G.REV_CONSDRD EQ FALSE)
                THEN  IF ((ITF.INT_CROSS EQ $TRUE) OR (ITF.ZDINT EQ 0 AND
                          ITF.RZ GT 0) OR (ITF.ZDINT * G.ZDMODEL LT 0))
                          THEN  MZDINT = ITF.ZDOUTR;
                          ELSE  MZDINT = ITF.ZDINR;
                ELSE  MZDINT = ITF.ZDINT;

            CALL MODEL_SEP
                IN (DELAY, ZDGOAL, Z, ZD, P.RACCEL, NEW_SENSE, ITF.ZINT, MZDINT, ITF entry)
                OUT (ZMP);

            IF (ZMP LE 0 OR NOMINAL_SEP GE G.ALIM)
                THEN  CLEAR ITF.REVERSE;
        <End own is assumed to follow its RA>
END Reversal_modeling;
```

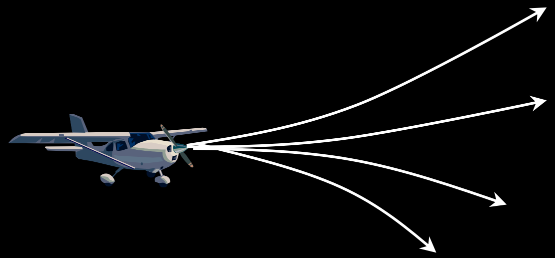

Traffic Alert and Collision Avoidance System (TCAS) RTCA DO-185B (1799 total pages / 440 is pseudocode)

Why is it hard?

State Uncertainty

Imperfect sensor information leads to uncertainty in position and velocity of aircraft

Dynamic Uncertainty



Variability in pilot behavior makes it difficult to predict future trajectories of aircraft

Multiple Objectives

System must carefully balance both safety and operational considerations

Decision-Making Methods

1. Explicit programming
   - Ex: if/then statements

   → Heavy burden on designer

Heuristic Method for Lane Changing: MOBIL

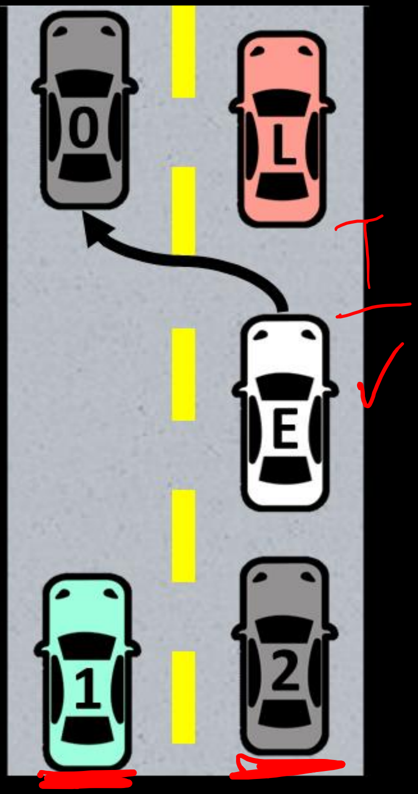
- Safety criterion:

$$\tilde{a}_E \geq -b_{safe}$$

- Decision rule:

$$\tilde{a}_E - a_E + p(\tilde{a}_1 - a_1 + \tilde{a}_2 - a_2) > \Delta a_{th}$$

- Politeness factor, p: 0.35
- Safe braking limit, b_{safe}: $2\, {}^{m}/_{s^2}$
- Acceleration threshold: $0.1\, {}^{m}/_{s^2}$
- Look-ahead horizon: $30m$



Decision-Making Methods

1. Explicit programming
   - Ex: if/then statements

   → Heavy burden on designer
2. Supervised learning
   - Ex: imitation learning

   → Generalizing is often a challenge
3. Optimization / optimal control
   - Ex: MPC

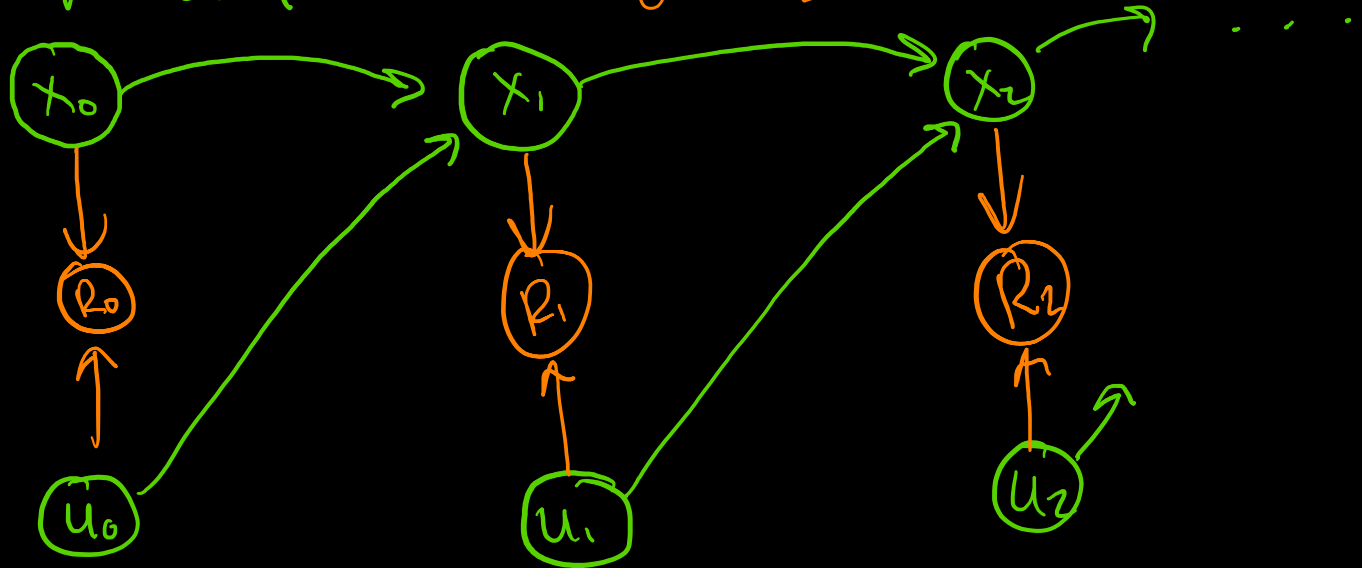
   → Requires a high-fidelity model and lots of computation
4. Planning
   - Given a stochastic model, how to algorithmically determine best policy?
5. Reinforcement Learning
   - If model is unknown (or very complex), learn policy through experience

Today’s Plan

- Introduction to decision-making
- Markov Decision Processes
- MDP Policies and Value Iteration
- Simple Example

Markov Decision Processes (MDPs)

$$(\underbrace{S, A, T}_{\text{model}}, \underbrace{\gamma, R}_{\text{design}})$$

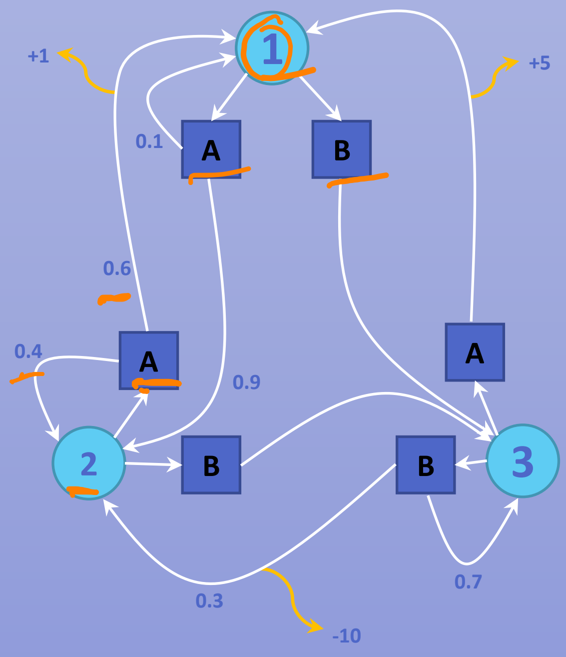


Uncertainty in Motion

- **Markov Decision Processes (MDPs)** model the AV and environment assuming full observability
  - $P(z|x)$: *deterministic* and bijective ✓
  - $P(x'|x,u)$: may be nondeterministic
  - Must incorporate uncertainty into the planner and generate actions for each state
- A policy for action selection is defined for all states

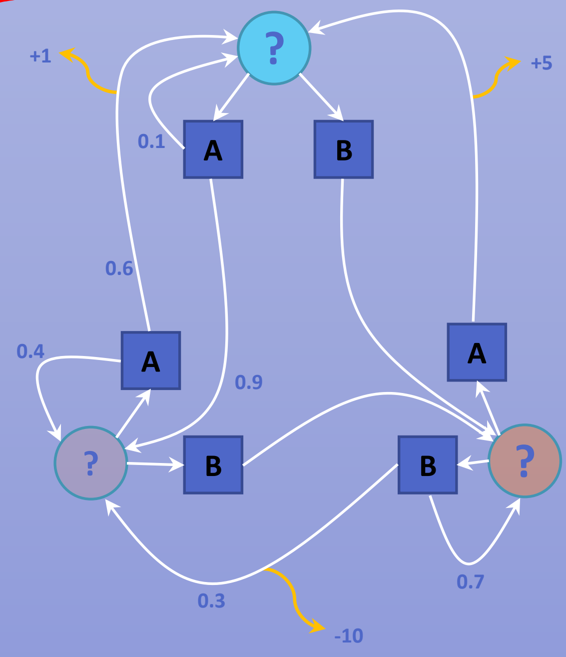
Markov Models

Markov Decision Processes (MDP)



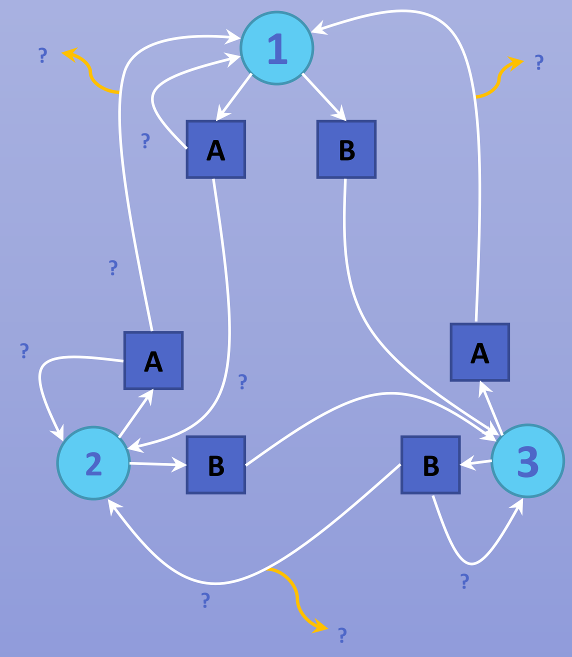
Uncertainty in effects of actions

Partially Observable MDP



Uncertainty in current state

Reinforcement Learning



Uncertainty in model

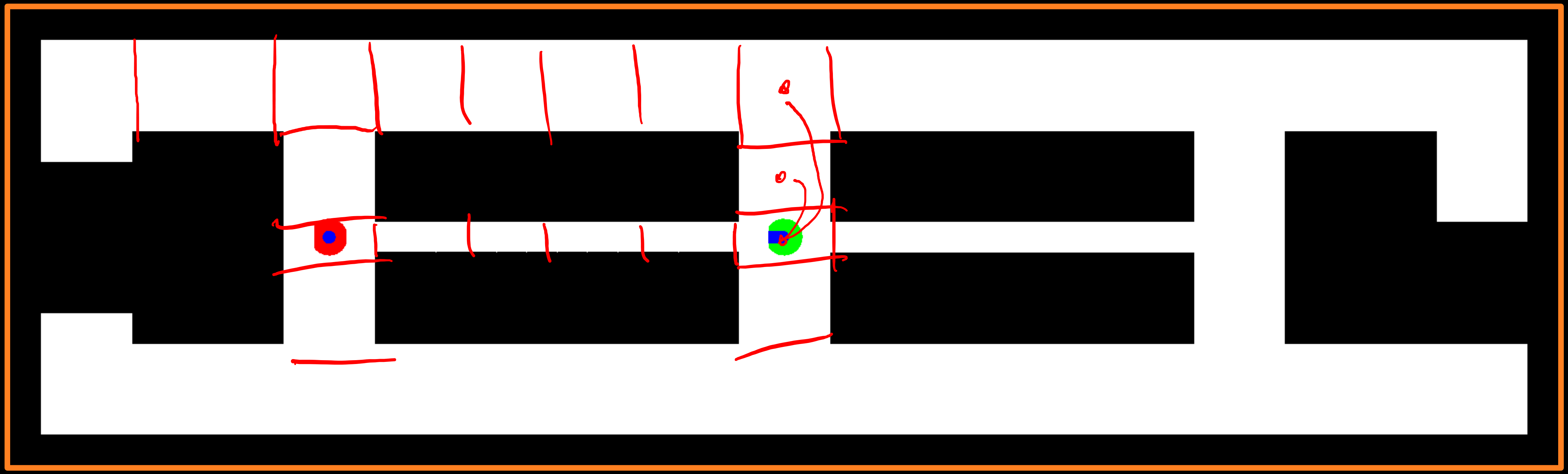
Markov Assumptions and Common Violations

Markov Assumption postulates that past and future data are independent if you know the current state.

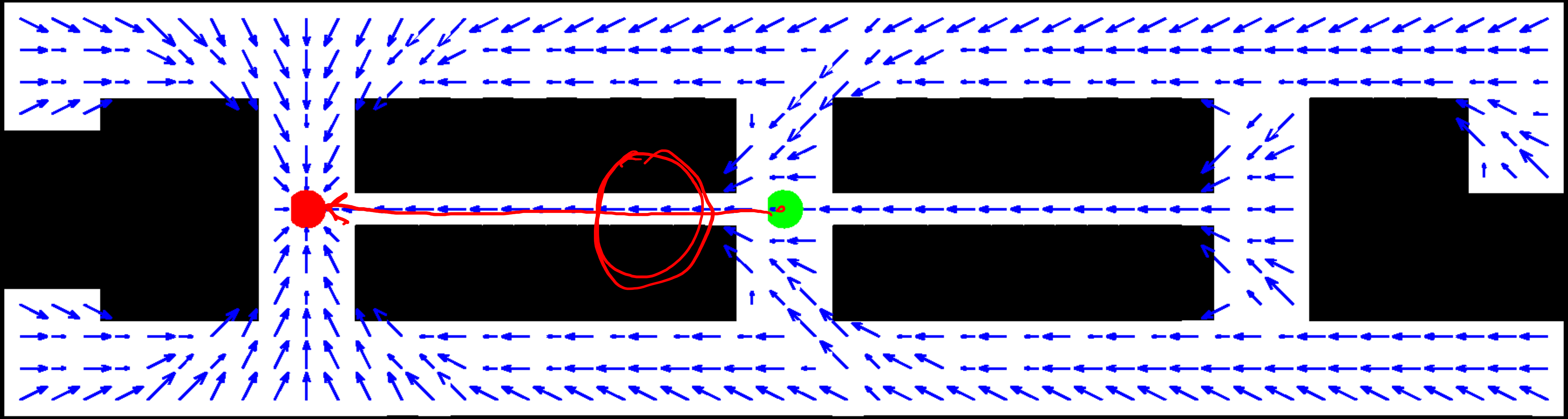
What are some common violations?

- Unmodeled dynamics in the environment not included in state
  - E.g., moving people and their effects on sensor measurements in localization
- Inaccuracies in the probabilistic model
  - E.g., error in the map of a localizing agent or incorrect model dynamics
- Approximation errors when using approximate representations
  - E.g., discretization errors from grids, Gaussian assumptions
- Variables in control scheme that influence multiple controls
  - E.g., the goal or target location will influence an entire sequence of control commands

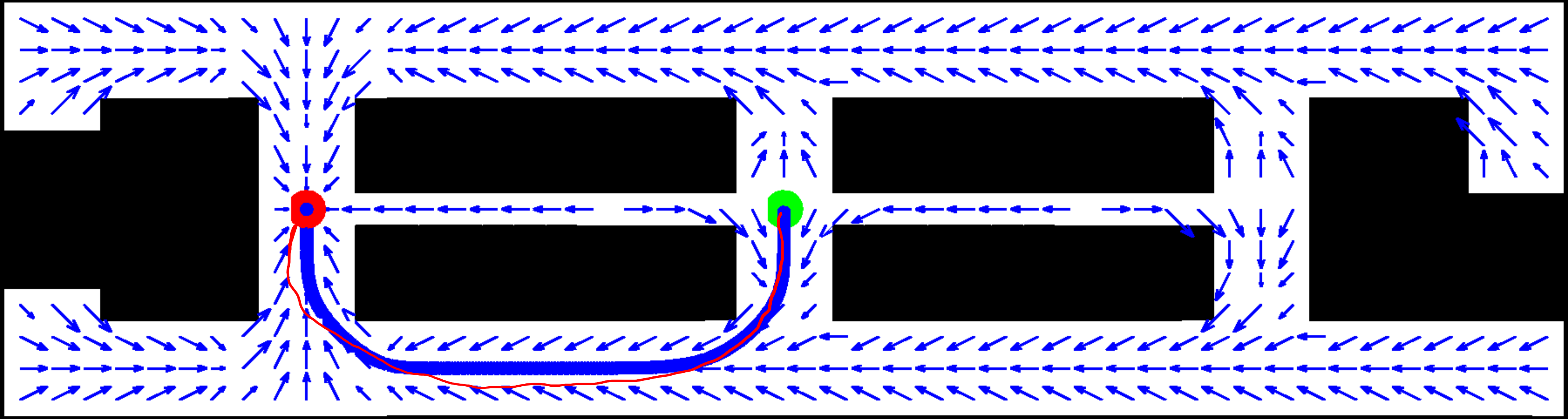
Grid World Example



Deterministic actions



Nondeterministic actions



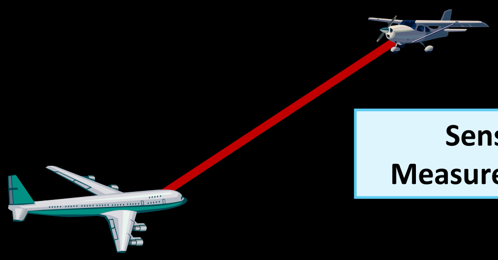
Defining Values

- Actions are driven by goals
  - E.g., reach destination, stay in lane
- Often, we want to reach goal while optimizing some cost
  - E.g., minimize time / energy consumption, obstacle avoidance
- We express both costs and goals in a single function, called the payoff function

ex)

$$r(x,u) = \begin{cases} 100 & \text{if reach goal} \\ -10 & \text{if collision} \\ \underline{-1} & \text{ow} \end{cases}$$

Traffic Alert and Collision Avoidance System (TCAS)



Sensor Measurements

```
IF (ITF.A LT G.ZTHR)
 THEN IF(ABS(ITF.VMD) LT G.ZTHR)
  THEN SET ZHIT;
  ELSE CLEAR ZHIT;
 ELSE IF (ITF.ADOT GE P.ZDTHR)
  THEN CLEAR ZHIT
  ELSE
   ITF.TAUV = -ITF.A/ITF.ADOT;
   IF (ITF.TAUV LT TVTHR AND
    ((ABS(ITF.VMD) LT G.ZTHR) OR
    (ITF.TAUV LT ITF.TRTRU))
    THEN SET ZHIT
    ELSE CLEAR ZHIT
IF (ZHIT EQ $TRUE AND
 ABS(ITF.ZDINT) GT P.MAXZDINT
 THEN CLEAR ZHIT
```

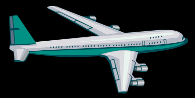
Resolution Advisory

Surveillance

Advisory Logic

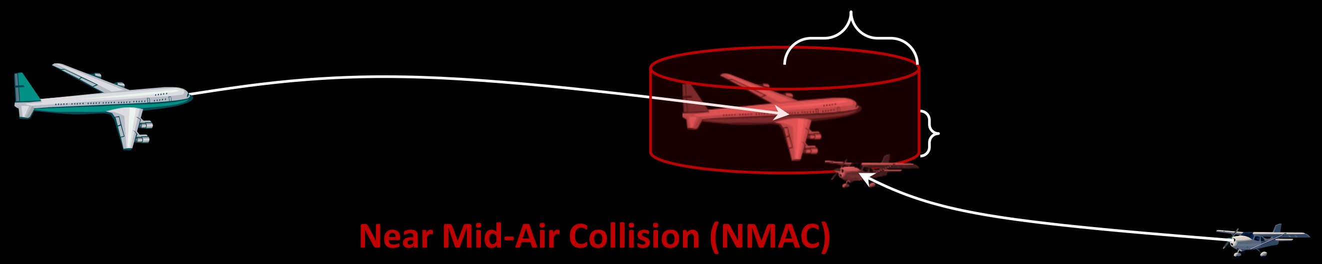
Display

ACAS X: Simplified MDP



State space	Action space
• Relative altitude	• Clear of conflict
• Own vertical rate	• Climb > 1500 ft/min
• Intruder vertical rate	• Climb > 2500 ft/min
• Time to lateral NMAC	• Descend > 1500 ft/min
• State of advisory	• Descend > 2500 ft/min
Dynamic model	**Reward model**
• Head-on, constant closure	• NMAC (-1)
• Random vertical acceleration	• Alert (-0.01)
• Pilot response delay (5 s)	• Reversal (-0.01)
• Pilot response strength (1/4 g)	• Strengthen (-0.009)
• State of advisory	• Clear of conflict (0.0001)

ACAS X: Simplified MDP

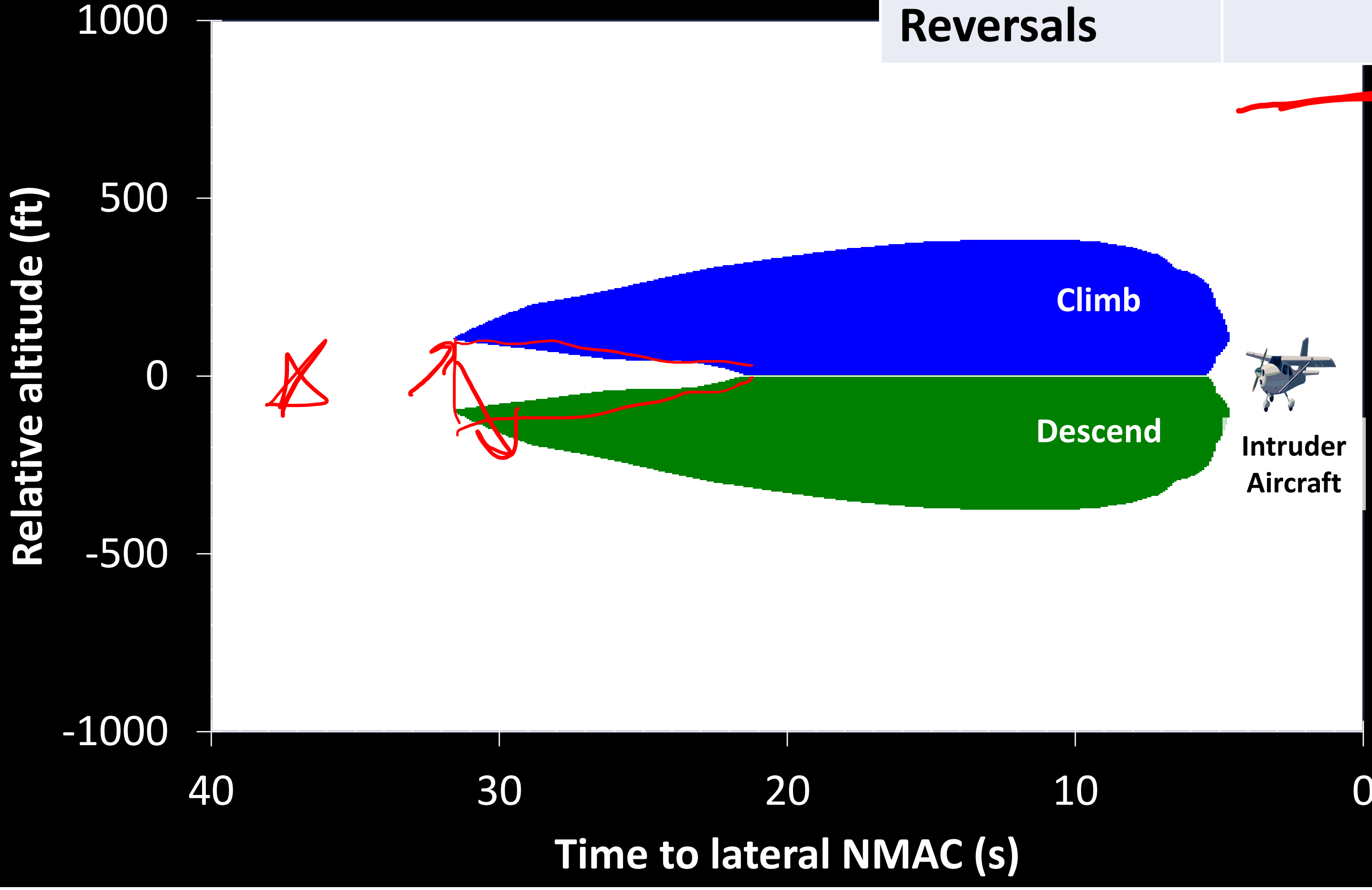


State space	Action space
• Relative altitude • Own vertical rate • Intruder vertical rate • Time to lateral NMAC • State of advisory	• Clear of conflict • Climb > 1500 ft/min • Climb > 2500 ft/min • Descend > 1500 ft/min • Descend > 2500 ft/min
Dynamic model	**Reward model**
• Head-on, constant closure • Random vertical acceleration • Pilot response delay (5 s) • Pilot response strength (1/4 g) • State of advisory	• NMAC (-1) • Alert (-0.01) • Reversal (-0.01) • Strengthen (-0.009) • Clear of conflict (0.0001)

Optimized Logic

Both Own and Intruder Level

Metric	TCAS	ACAS X
NMACs	169	3
Alerts	994,317	690,406
Strengthens	40,470	92,946
Reversals	197,315	9,569



Today’s Plan

- Introduction to decision-making
- Markov Decision Processes
- MDP Policies and Value Iteration
- Simple Example

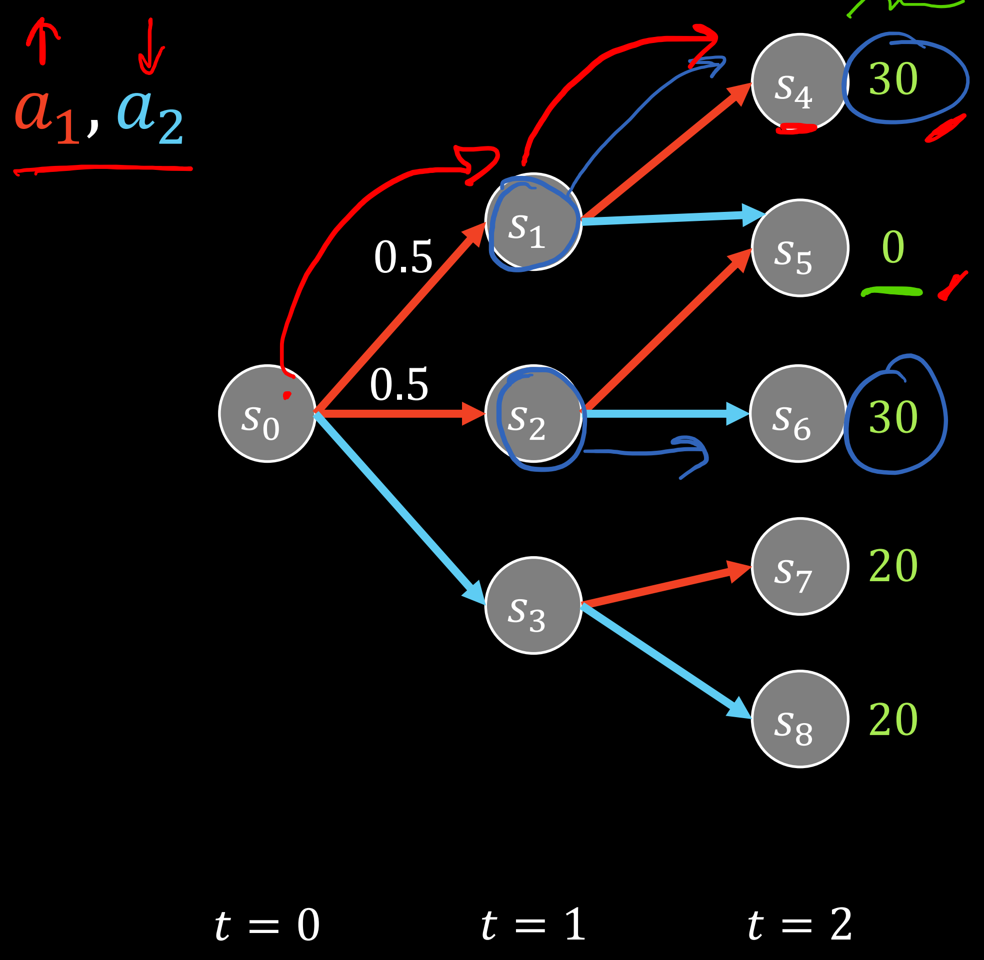
Decision-Making Policies

- We want to devise a scheme that generates actions to optimize the future payoff *in expectation*
- Policy: $\pi : x_t \rightarrow u_t$
  - Maps states to actions
  - Can be low-level reactive algorithm or a long-term, high-level planner
  - May or may not be deterministic
- Typically, we want a policy that optimizes future payoff, considering optimal actions over a planning (time) horizon

Open vs. Closed Loop Planning

- <u>Closed-Loop Planning:</u> accounts for future information in planning. This creates a reactive plan (policy) that can react to different outcomes over time
- <u>Open-Loop Planning:</u> path panning algorithms develop a static sequence of actions

Open Loop vs. Closed Loop Planning



open-loop plans: expected R

$U(a_1, a_1) = 0.5 \cdot 30 + 0.5 \cdot 0 = 15$

$U(a_1, a_2) = 15$

$U(a_2, a_1) = 20$

$U(a_2, a_2) = 20$

MDP Policies

- Policies map states to actions

$$\pi : x \rightarrow u$$

- We want to find a policy that maximizes future pay off
  - Suppose $T = 1$: $\pi_1(x) = \text{argmax}_u \; r(x,u)$
- We write the Value Function for given π:

$$V_1(x) = \gamma \max_u r(x,u)$$

- Generally, we want to find the sequence of actions that optimize the *expected cumulative discounted future payoff*

Expected Cumulative Payoff

payoff

$$R_T = \mathbb{E}\left[\sum_{\tau=0}^{T} \gamma^{\tau} r_{t+\tau}\right]$$

discount factor

1. Greedy case: $T = 1$
   → Optimize next payoff
2. Finite Horizon: $1 \leq T < \infty, (\gamma \leq 1)$
   → Optimize R_T for set time window
3. Infinite Horizon: $T = \infty, (\gamma < 1)$
   → Optimize R_∞ for all time
   If $|r| \leq r_{max}$, discounting guarantees R_∞ is finite

$$R_\infty \leq r_{max} + \gamma r_{max} + \gamma^2 r_{max} + \cdots = \frac{r_{max}}{1-\gamma}$$

Value Functions

Recall: $V_1(x) = \gamma \max_u r(x,u)$

For longer time horizons, we define V(x) recursively:

T = 2: pick an action that max sum V_1 + 1-step payoff

$$\pi_2(x) = \underset{u}{\text{argmax}} \left[r(x,u) + \int V_1(x') p(x'|x,u) dx' \right]$$

$$V_2(x) = \gamma \max_u \left[\qquad \right]$$

For finite T:

$$\pi_T(x) = \underset{u}{\text{argmax}} \left[r(x,u) + \int V_{T-1}(x') p(x'|x,u) dx' \right]$$

$$V_T(x) = \gamma \max_u \left[\qquad \right]$$

Value Functions

- In the infinite time horizon, we tend to reach equilibrium:

$$V_\infty(x) = \gamma \max_u \left[r(x,u) + \int V_\infty(x') p(x' | x,u) \, dx' \right]$$

- This is the *Bellman Equation*
  - Satisfying this is necessary and sufficient for an optimal policy

Computing the (Approximate) Value Function

- Initial guess for $\hat{V}$
  - $\hat{V}(x) \leftarrow r_{min}, \forall x$
- Successively update for increasing horizons
  - $\hat{V}(x) \leftarrow \gamma \max_{u} \left[r(x,u) + \int \hat{V}(x')p(x'|x,u)dx' \right]$
- Value iteration converges if $\gamma < 1$
- Given estimate $\hat{V}(x)$, policy is found:
  - $\pi(x) = \mathrm{argmax}_u \left[r(x,u) + \int \hat{V}(x')p(x'|x,u)dx' \right]$
- Often, we use the discrete version:
  - $\pi(x) = \mathrm{argmax}_u \left[r(x,u) + \sum_x' \hat{V}(x')p(x'|x,u) \right]$

Today's Plan

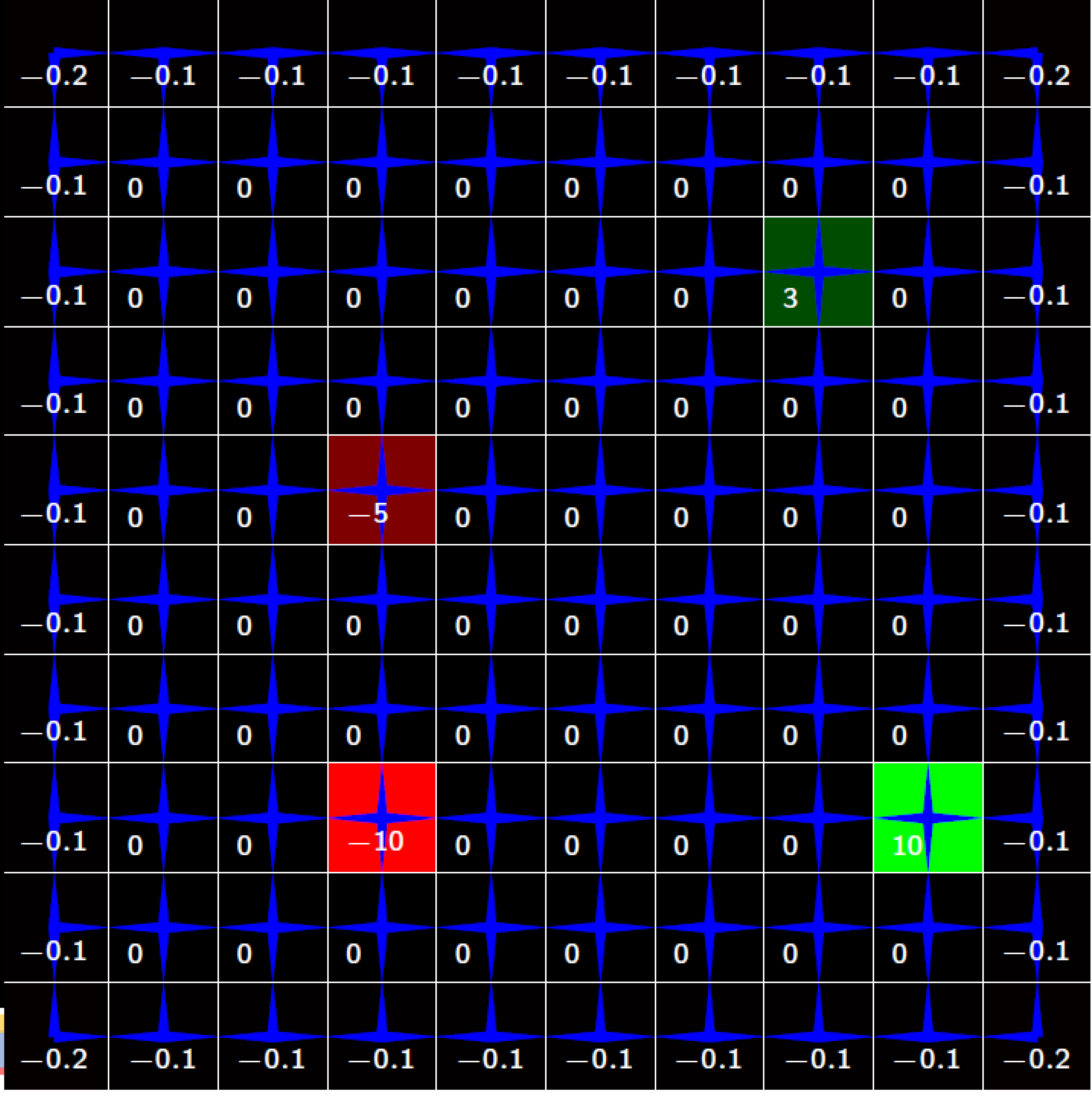
- Introduction to decision-making
- Markov Decision Processes
- MDP Policies and Value Iteration
- Simple Example

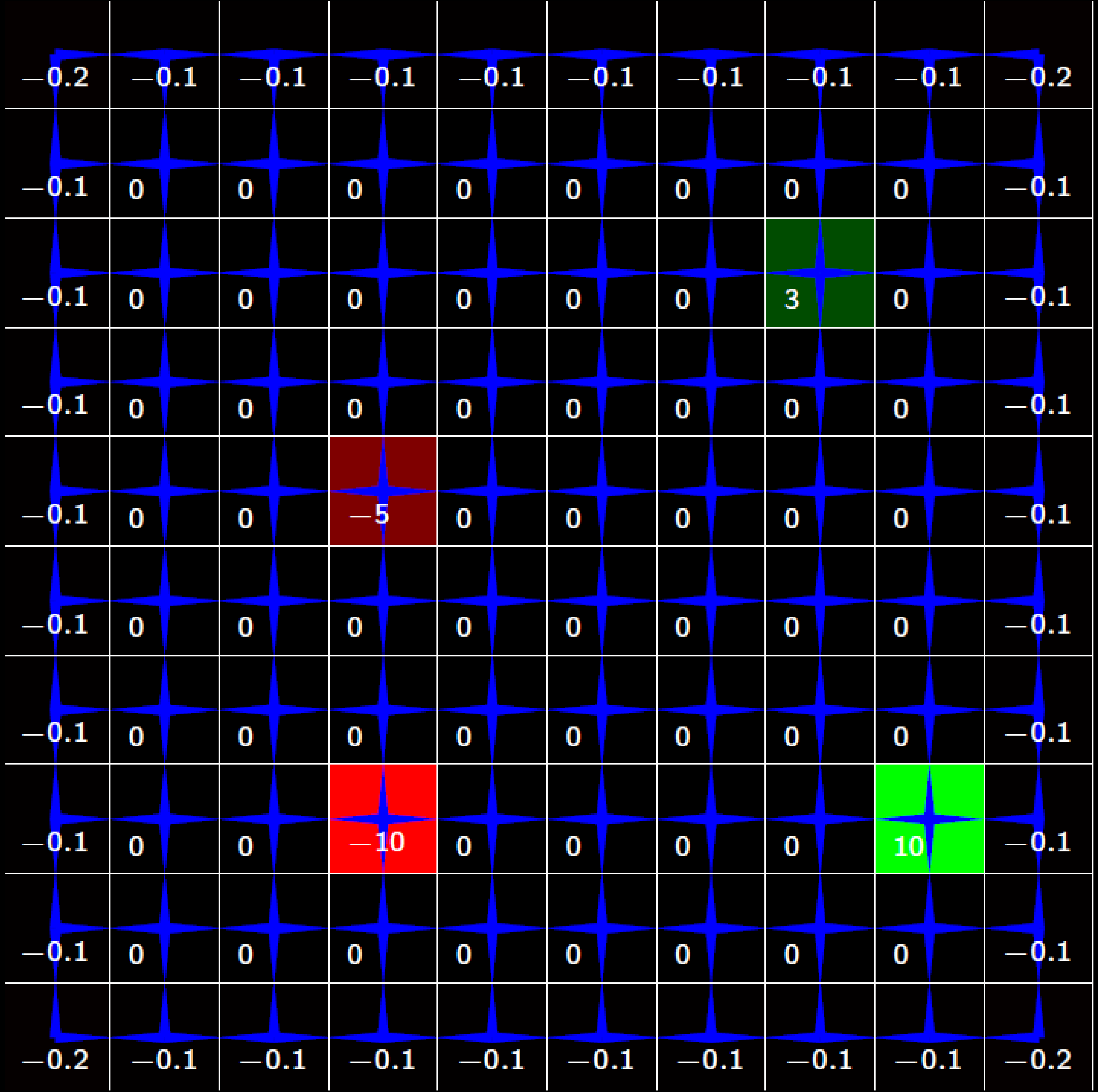
Example: Create an MDP

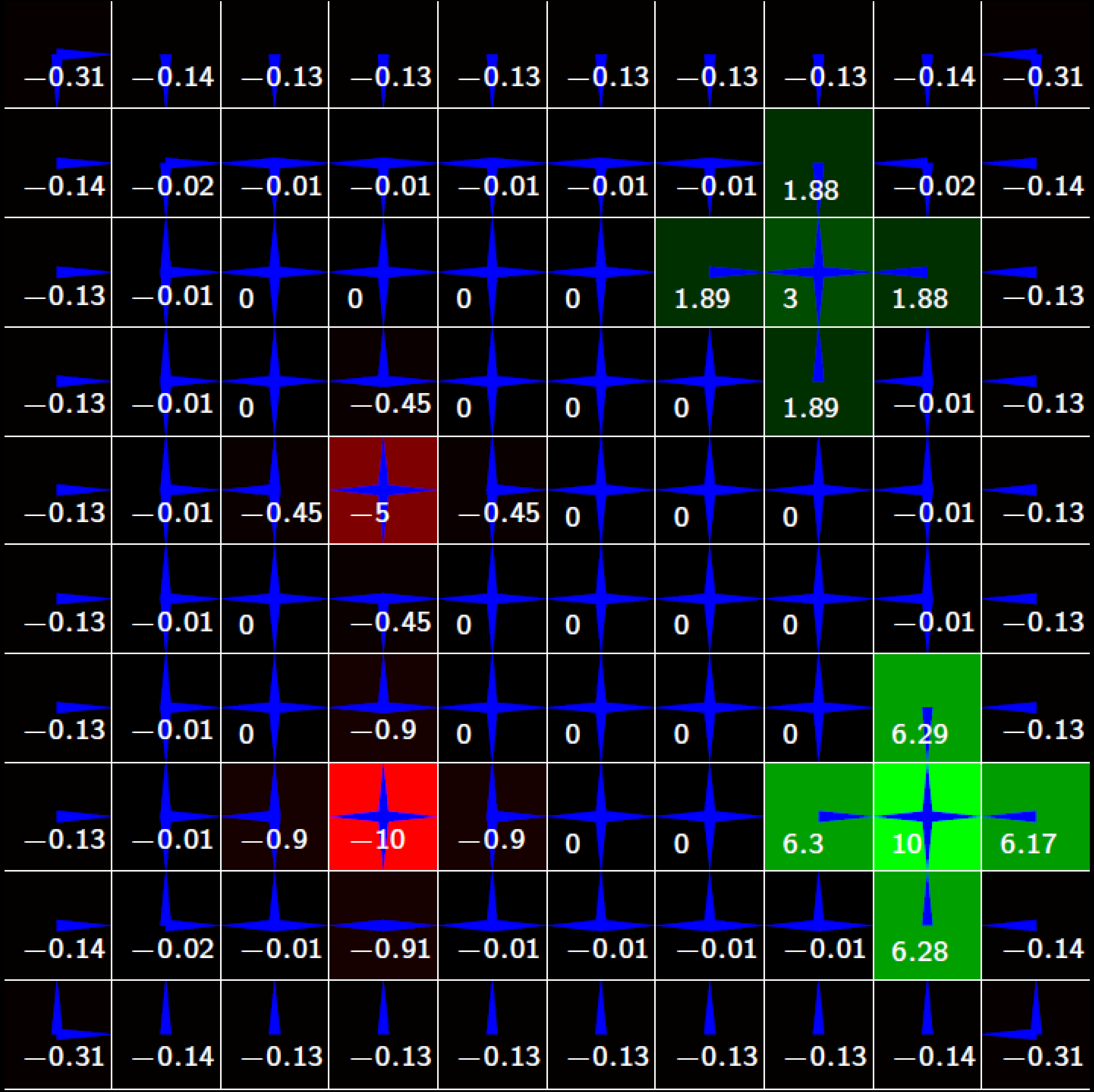
Example: Value Iteration

Grid world example

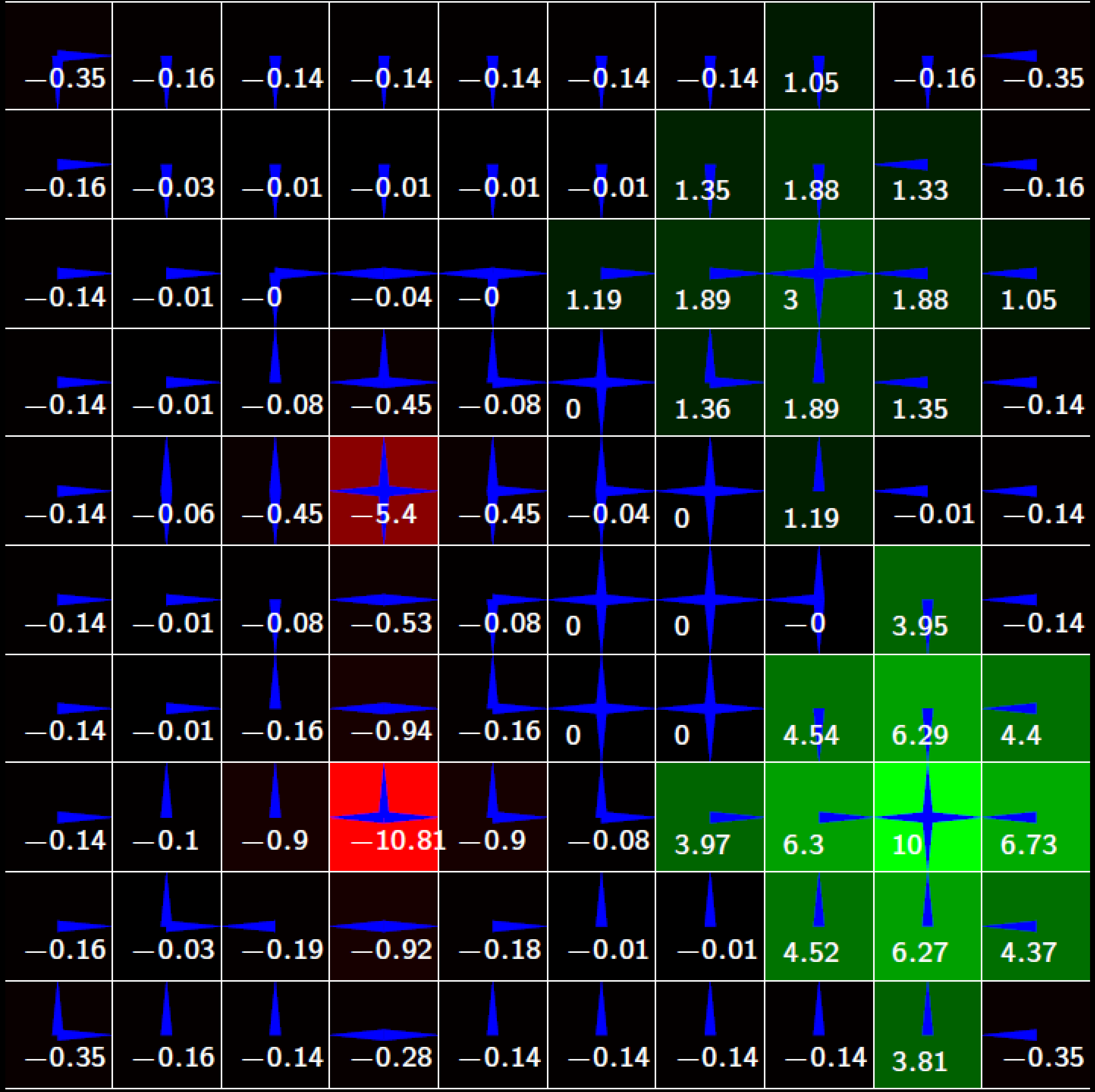
- States: cells in 10 × 10 grid
- Actions: up, down, left, right
- Transition model: 0.7 chance of moving in intended direction, uniform in other directions
- Reward:
  - two states with cost
  - two terminal states with rewards
  - −1 for wall crash
- Discount is 0.9



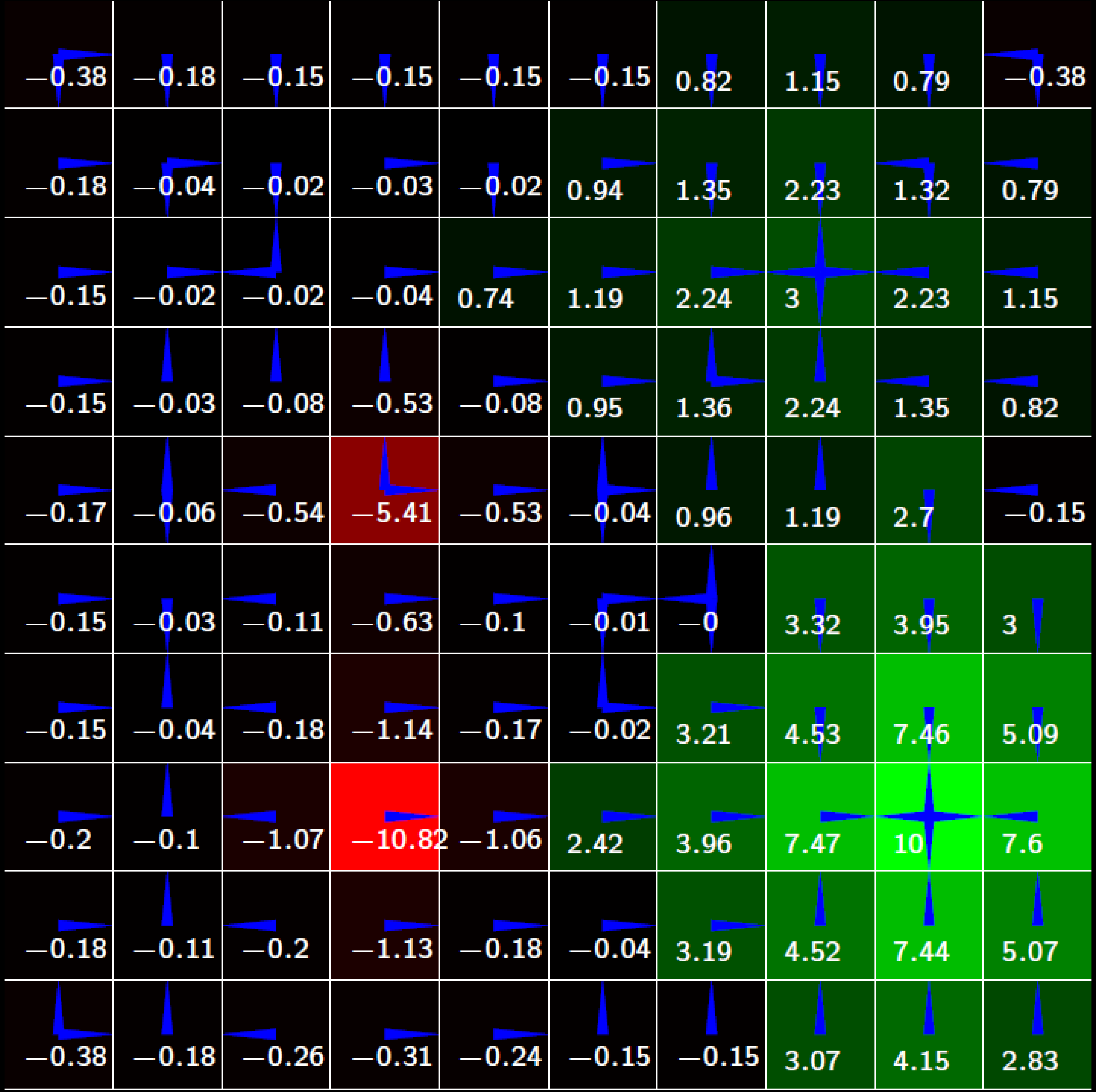

−0.2 −0.1 −0.1 −0.1 −0.1 −0.1 −0.1 −0.1 −0.1 −0.2
−0.1 0 0 0 0 0 0 0 0 −0.1
−0.1 0 0 0 0 0 0 3 0 −0.1
−0.1 0 0 0 0 0 0 0 0 −0.1
−0.1 0 0 −5 0 0 0 0 0 −0.1
−0.1 0 0 0 0 0 0 0 0 −0.1
−0.1 0 0 0 0 0 0 0 0 −0.1
−0.1 0 0 −10 0 0 0 0 10 −0.1
−0.1 0 0 0 0 0 0 0 0 −0.1
−0.2 −0.1 −0.1 −0.1 −0.1 −0.1 −0.1 −0.1 −0.1 −0.2



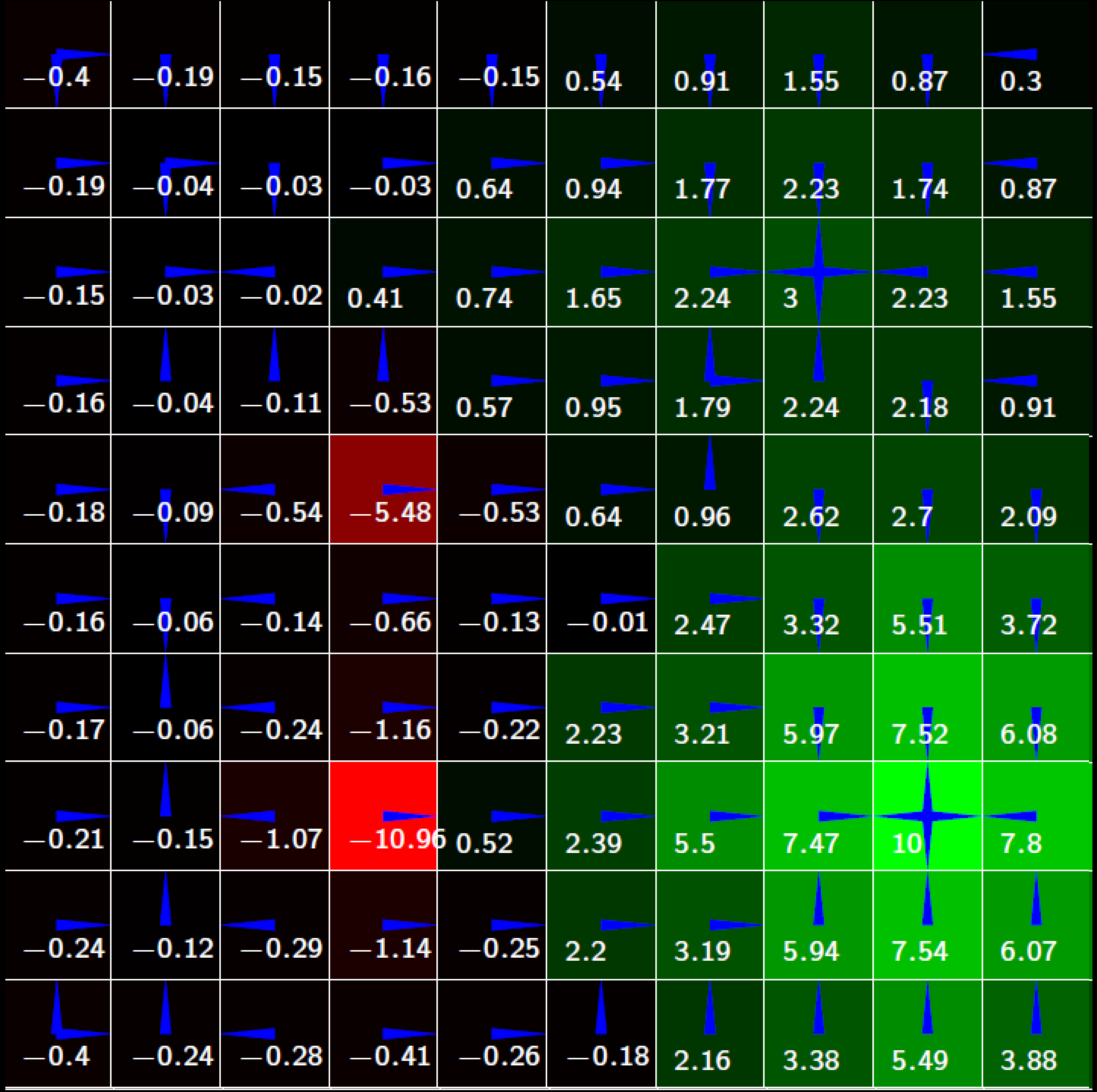
−0.31 −0.14 −0.13 −0.13 −0.13 −0.13 −0.13 −0.13 −0.14 −0.31
−0.14 −0.02 −0.01 −0.01 −0.01 −0.01 −0.01 1.88 −0.02 −0.14
−0.13 −0.01 0 0 0 0 1.89 3 1.88 −0.13
−0.13 −0.01 0 −0.45 0 0 0 1.89 −0.01 −0.13
−0.13 −0.01 −0.45 −5 −0.45 0 0 0 −0.01 −0.13
−0.13 −0.01 0 −0.45 0 0 0 0 −0.01 −0.13
−0.13 −0.01 0 −0.9 0 0 0 0 6.29 −0.13
−0.13 −0.01 −0.9 −10 −0.9 0 0 6.3 10 6.17
−0.14 −0.02 −0.01 −0.91 −0.01 −0.01 −0.01 −0.01 6.28 −0.14
−0.31 −0.14 −0.13 −0.13 −0.13 −0.13 −0.13 −0.13 −0.14 −0.31

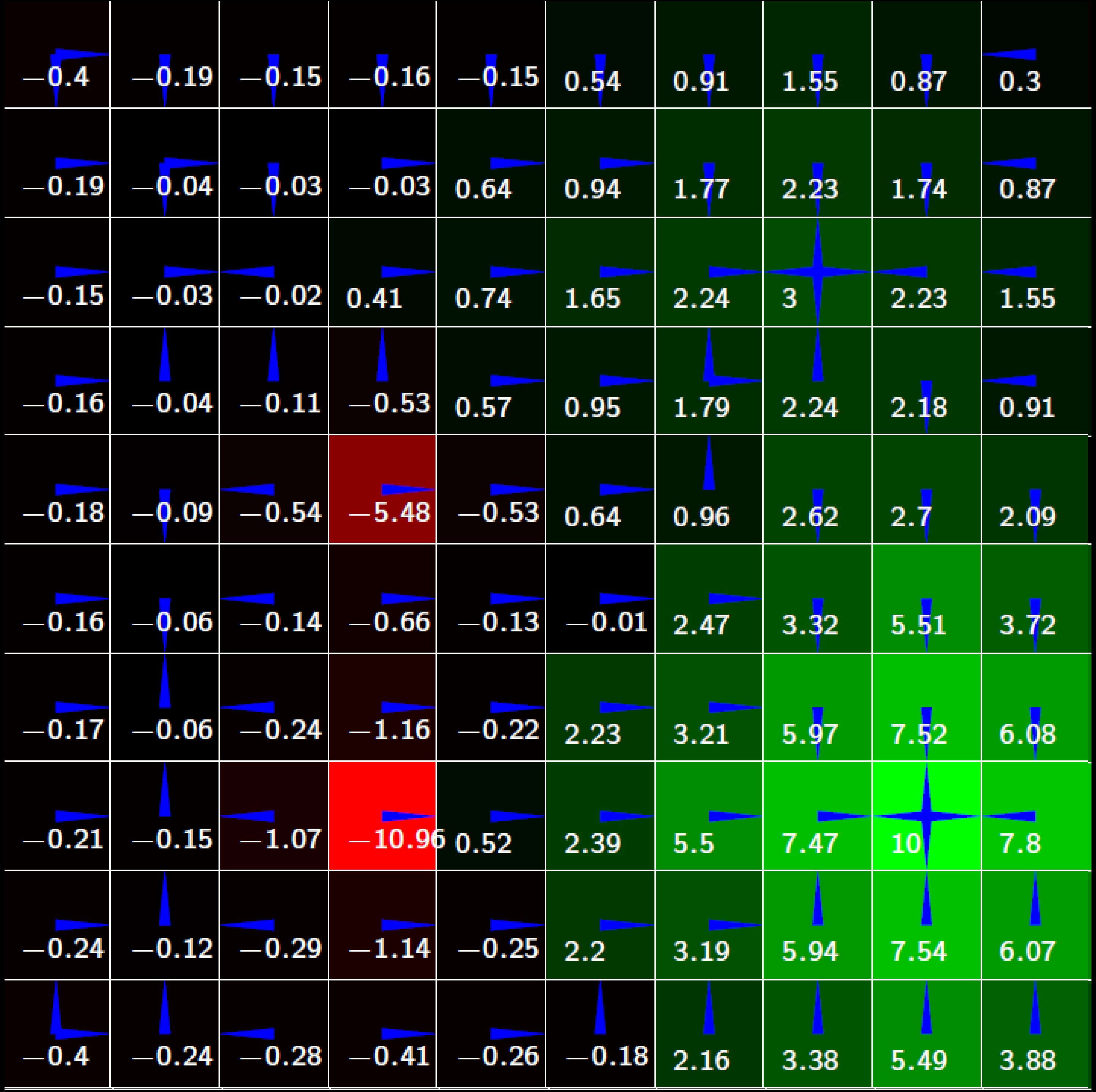


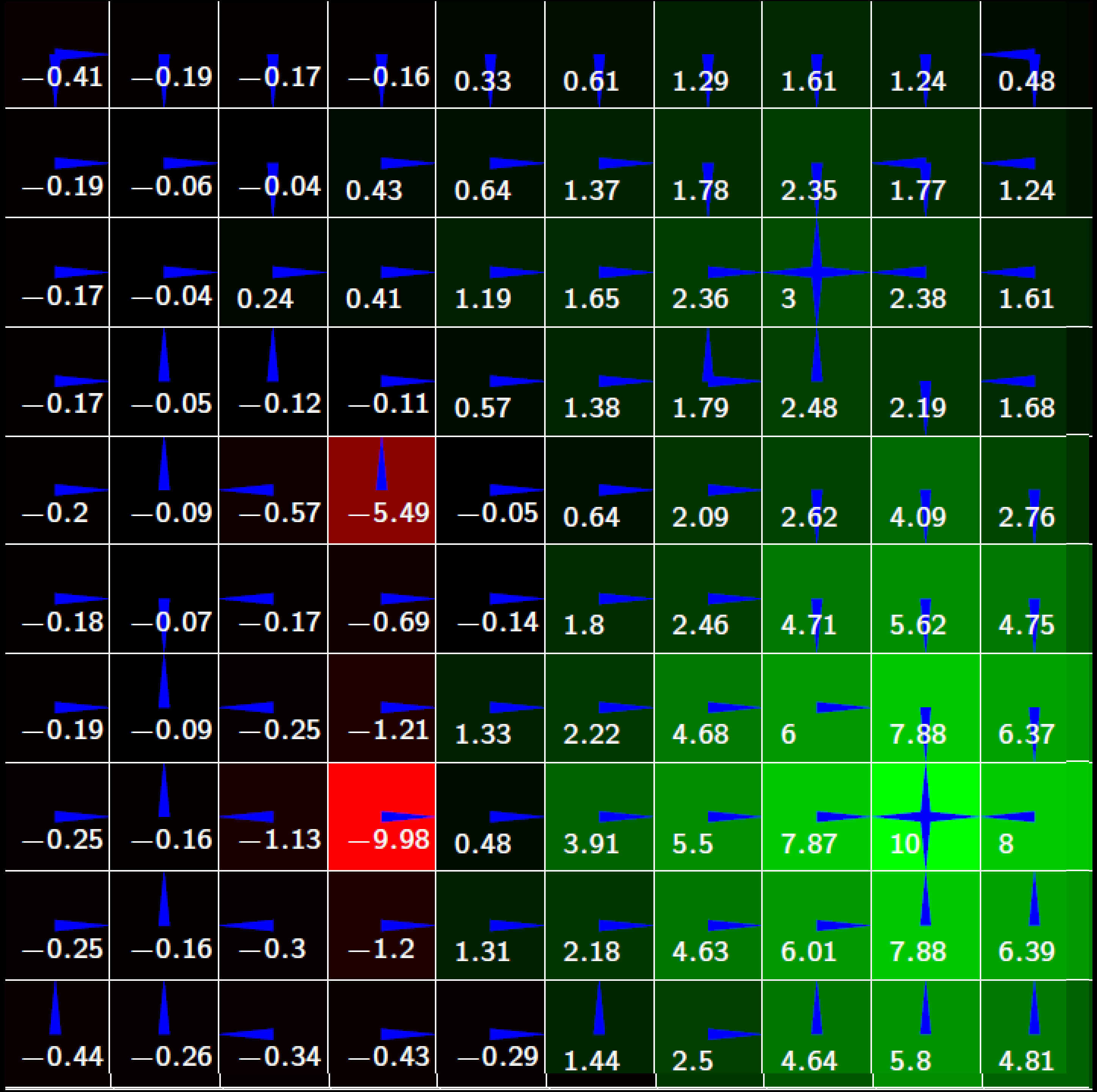
−0.35 −0.16 −0.14 −0.14 −0.14 −0.14 −0.14 1.05 −0.16 −0.35
−0.16 −0.03 −0.01 −0.01 −0.01 −0.01 1.35 1.88 1.33 −0.16
−0.14 −0.01 −0 −0.04 −0 1.19 1.89 3 1.88 1.05
−0.14 −0.01 −0.08 −0.45 −0.08 0 1.36 1.89 1.35 −0.14
−0.14 −0.06 −0.45 −5.4 −0.45 −0.04 0 1.19 −0.01 −0.14
−0.14 −0.01 −0.08 −0.53 −0.08 0 0 −0 3.95 −0.14
−0.14 −0.01 −0.16 −0.94 −0.16 0 0 4.54 6.29 4.4
−0.14 −0.1 −0.9 −10.81 −0.9 −0.08 3.97 6.3 10 6.73
−0.16 −0.03 −0.19 −0.92 −0.18 −0.01 −0.01 4.52 6.27 4.37
−0.35 −0.16 −0.14 −0.28 −0.14 −0.14 −0.14 −0.14 3.81 −0.35

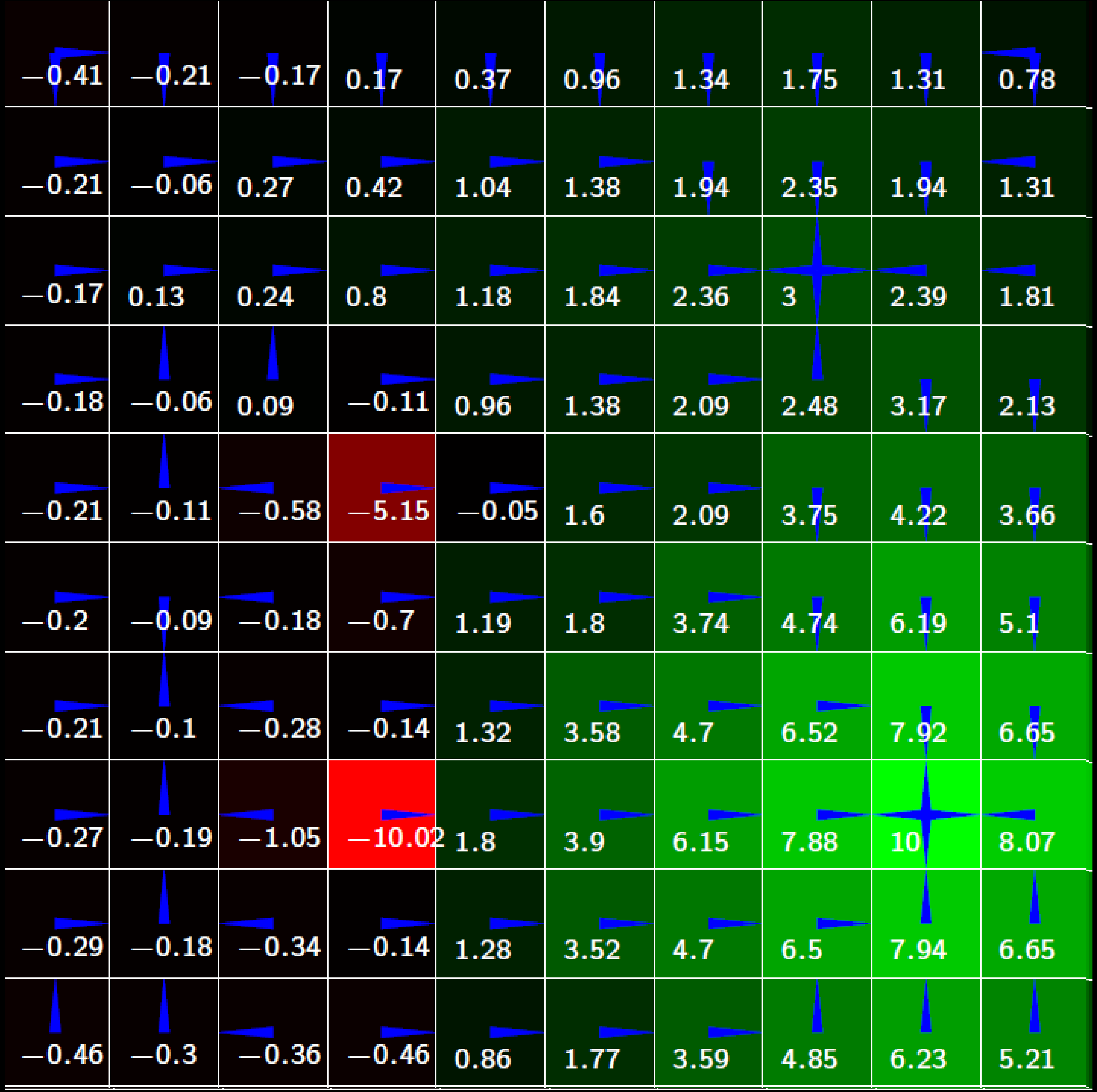


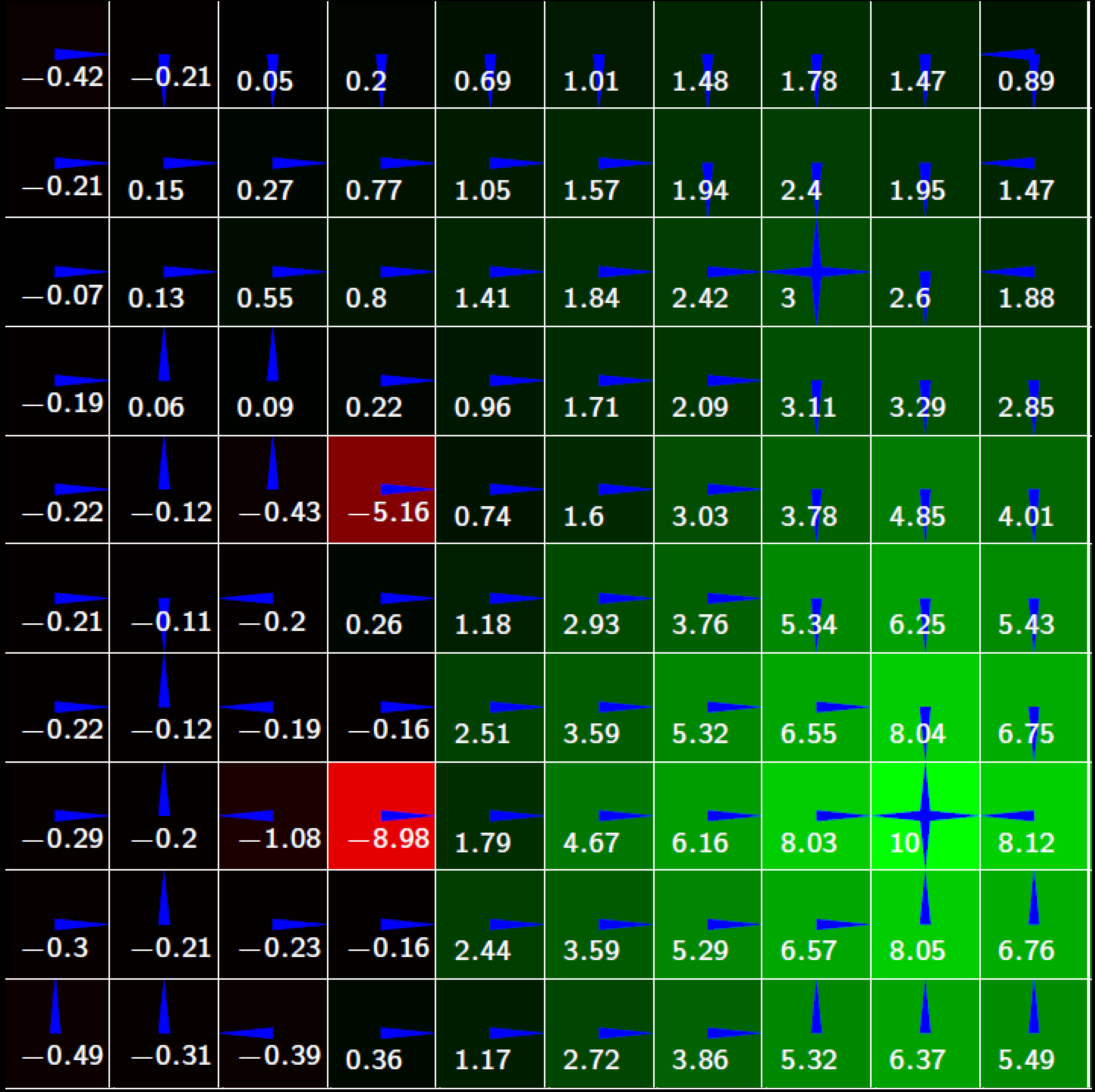
-0.38 -0.18 -0.15 -0.15 -0.15 -0.15 0.82 1.15 0.79 -0.38
-0.18 -0.04 -0.02 -0.03 -0.02 0.94 1.35 2.23 1.32 0.79
-0.15 -0.02 -0.02 -0.04 0.74 1.19 2.24 3 2.23 1.15
-0.15 -0.03 -0.08 -0.53 -0.08 0.95 1.36 2.24 1.35 0.82
-0.17 -0.06 -0.54 -5.41 -0.53 -0.04 0.96 1.19 2.7 -0.15
-0.15 -0.03 -0.11 -0.63 -0.1 -0.01 -0 3.32 3.95 3
-0.15 -0.04 -0.18 -1.14 -0.17 -0.02 3.21 4.53 7.46 5.09
-0.2 -0.1 -1.07 -10.82 -1.06 2.42 3.96 7.47 10 7.6
-0.18 -0.11 -0.2 -1.13 -0.18 -0.04 3.19 4.52 7.44 5.07
-0.38 -0.18 -0.26 -0.31 -0.24 -0.15 -0.15 3.07 4.15 2.83

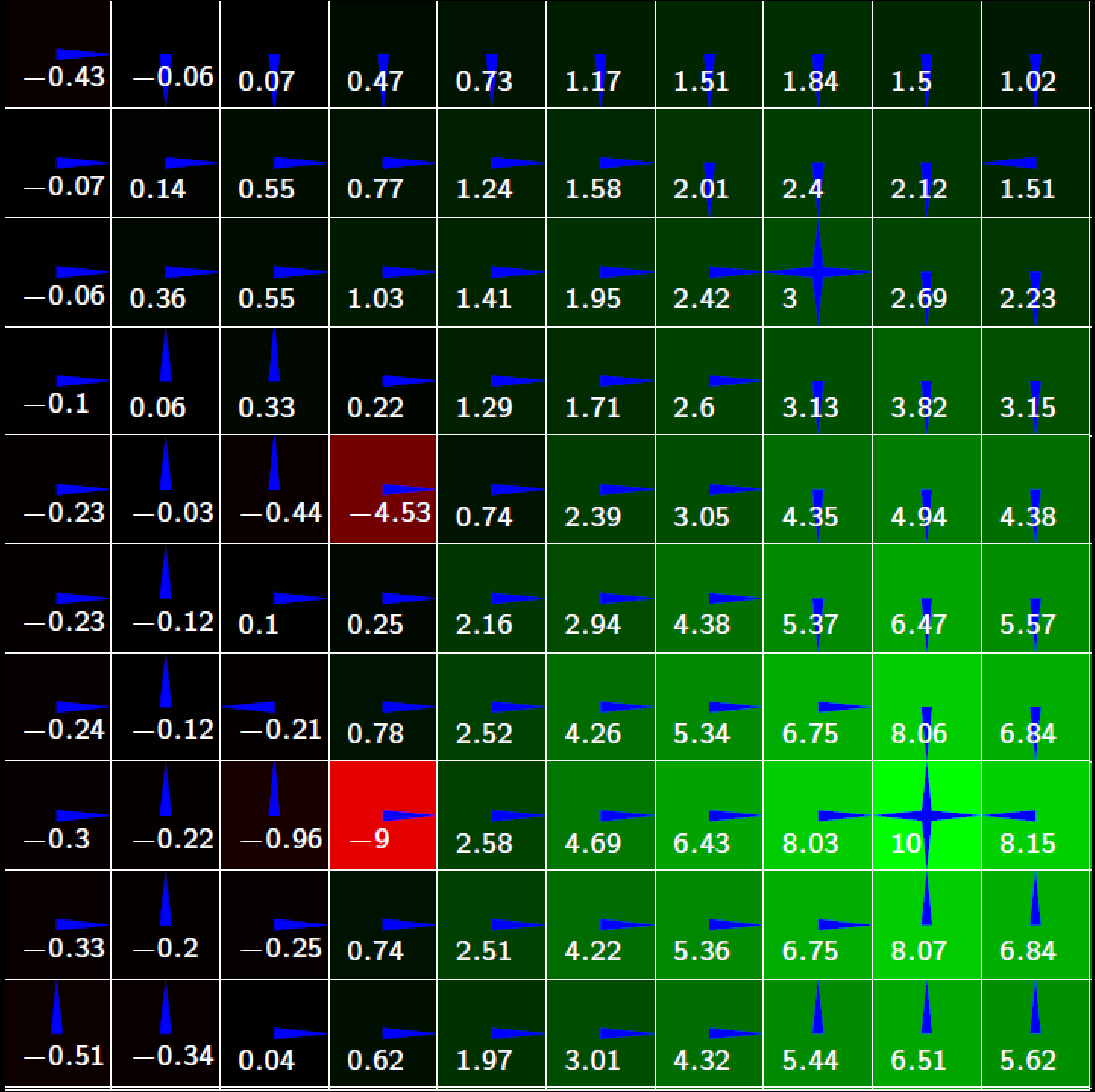

−0.4 −0.19 −0.15 −0.16 −0.15 0.54 0.91 1.55 0.87 0.3
−0.19 −0.04 −0.03 −0.03 0.64 0.94 1.77 2.23 1.74 0.87
−0.15 −0.03 −0.02 0.41 0.74 1.65 2.24 3 2.23 1.55
−0.16 −0.04 −0.11 −0.53 0.57 0.95 1.79 2.24 2.18 0.91
−0.18 −0.09 −0.54 −5.48 −0.53 0.64 0.96 2.62 2.7 2.09
−0.16 −0.06 −0.14 −0.66 −0.13 −0.01 2.47 3.32 5.51 3.72
−0.17 −0.06 −0.24 −1.16 −0.22 2.23 3.21 5.97 7.52 6.08
−0.21 −0.15 −1.07 −10.96 0.52 2.39 5.5 7.47 10 7.8
−0.24 −0.12 −0.29 −1.14 −0.25 2.2 3.19 5.94 7.54 6.07
−0.4 −0.24 −0.28 −0.41 −0.26 −0.18 2.16 3.38 5.49 3.88


−0.4 −0.19 −0.15 −0.16 −0.15 0.54 0.91 1.55 0.87 0.3
−0.19 −0.04 −0.03 −0.03 0.64 0.94 1.77 2.23 1.74 0.87
−0.15 −0.03 −0.02 0.41 0.74 1.65 2.24 3 2.23 1.55
−0.16 −0.04 −0.11 −0.53 0.57 0.95 1.79 2.24 2.18 0.91
−0.18 −0.09 −0.54 −5.48 −0.53 0.64 0.96 2.62 2.7 2.09
−0.16 −0.06 −0.14 −0.66 −0.13 −0.01 2.47 3.32 5.51 3.72
−0.17 −0.06 −0.24 −1.16 −0.22 2.23 3.21 5.97 7.52 6.08
−0.21 −0.15 −1.07 −10.96 0.52 2.39 5.5 7.47 10 7.8
−0.24 −0.12 −0.29 −1.14 −0.25 2.2 3.19 5.94 7.54 6.07
−0.4 −0.24 −0.28 −0.41 −0.26 −0.18 2.16 3.38 5.49 3.88

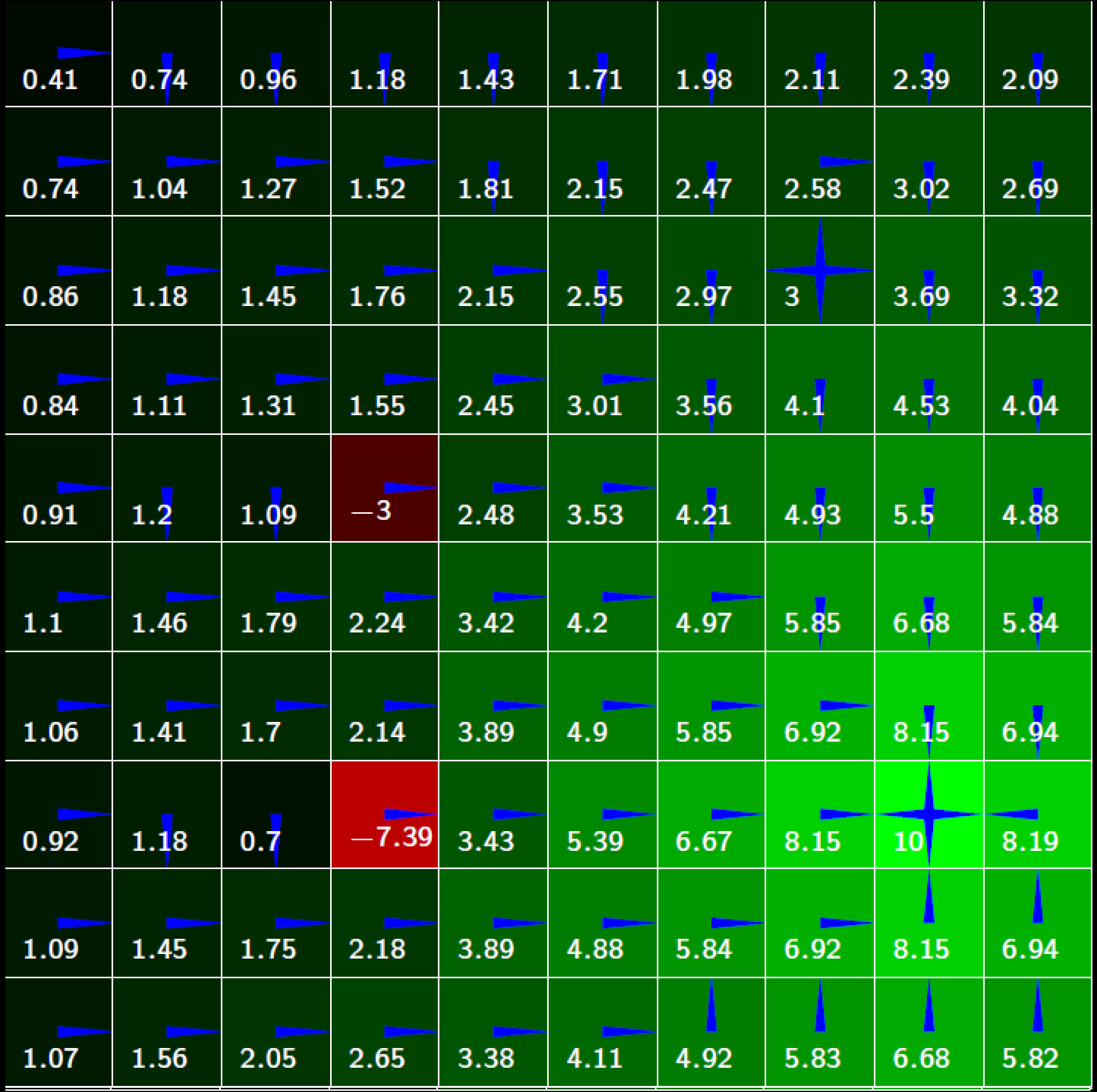

−0.41 −0.19 −0.17 −0.16 0.33 0.61 1.29 1.61 1.24 0.48
−0.19 −0.06 −0.04 0.43 0.64 1.37 1.78 2.35 1.77 1.24
−0.17 −0.04 0.24 0.41 1.19 1.65 2.36 3 2.38 1.61
−0.17 −0.05 −0.12 −0.11 0.57 1.38 1.79 2.48 2.19 1.68
−0.2 −0.09 −0.57 −5.49 −0.05 0.64 2.09 2.62 4.09 2.76
−0.18 −0.07 −0.17 −0.69 −0.14 1.8 2.46 4.71 5.62 4.75
−0.19 −0.09 −0.25 −1.21 1.33 2.22 4.68 6 7.88 6.37
−0.25 −0.16 −1.13 −9.98 0.48 3.91 5.5 7.87 10 8
−0.25 −0.16 −0.3 −1.2 1.31 2.18 4.63 6.01 7.88 6.39
−0.44 −0.26 −0.34 −0.43 −0.29 1.44 2.5 4.64 5.8 4.81


−0.41 −0.21 −0.17 0.17 0.37 0.96 1.34 1.75 1.31 0.78
−0.21 −0.06 0.27 0.42 1.04 1.38 1.94 2.35 1.94 1.31
−0.17 0.13 0.24 0.8 1.18 1.84 2.36 3 2.39 1.81
−0.18 −0.06 0.09 −0.11 0.96 1.38 2.09 2.48 3.17 2.13
−0.21 −0.11 −0.58 −5.15 −0.05 1.6 2.09 3.75 4.22 3.66
−0.2 −0.09 −0.18 −0.7 1.19 1.8 3.74 4.74 6.19 5.1
−0.21 −0.1 −0.28 −0.14 1.32 3.58 4.7 6.52 7.92 6.65
−0.27 −0.19 −1.05 −10.02 1.8 3.9 6.15 7.88 10 8.07
−0.29 −0.18 −0.34 −0.14 1.28 3.52 4.7 6.5 7.94 6.65
−0.46 −0.3 −0.36 −0.46 0.86 1.77 3.59 4.85 6.23 5.21


-0.42 -0.21 0.05 0.2 0.69 1.01 1.48 1.78 1.47 0.89
-0.21 0.15 0.27 0.77 1.05 1.57 1.94 2.4 1.95 1.47
-0.07 0.13 0.55 0.8 1.41 1.84 2.42 3 2.6 1.88
-0.19 0.06 0.09 0.22 0.96 1.71 2.09 3.11 3.29 2.85
-0.22 -0.12 -0.43 -5.16 0.74 1.6 3.03 3.78 4.85 4.01
-0.21 -0.11 -0.2 0.26 1.18 2.93 3.76 5.34 6.25 5.43
-0.22 -0.12 -0.19 -0.16 2.51 3.59 5.32 6.55 8.04 6.75
-0.29 -0.2 -1.08 -8.98 1.79 4.67 6.16 8.03 10 8.12
-0.3 -0.21 -0.23 -0.16 2.44 3.59 5.29 6.57 8.05 6.76
-0.49 -0.31 -0.39 0.36 1.17 2.72 3.86 5.32 6.37 5.49

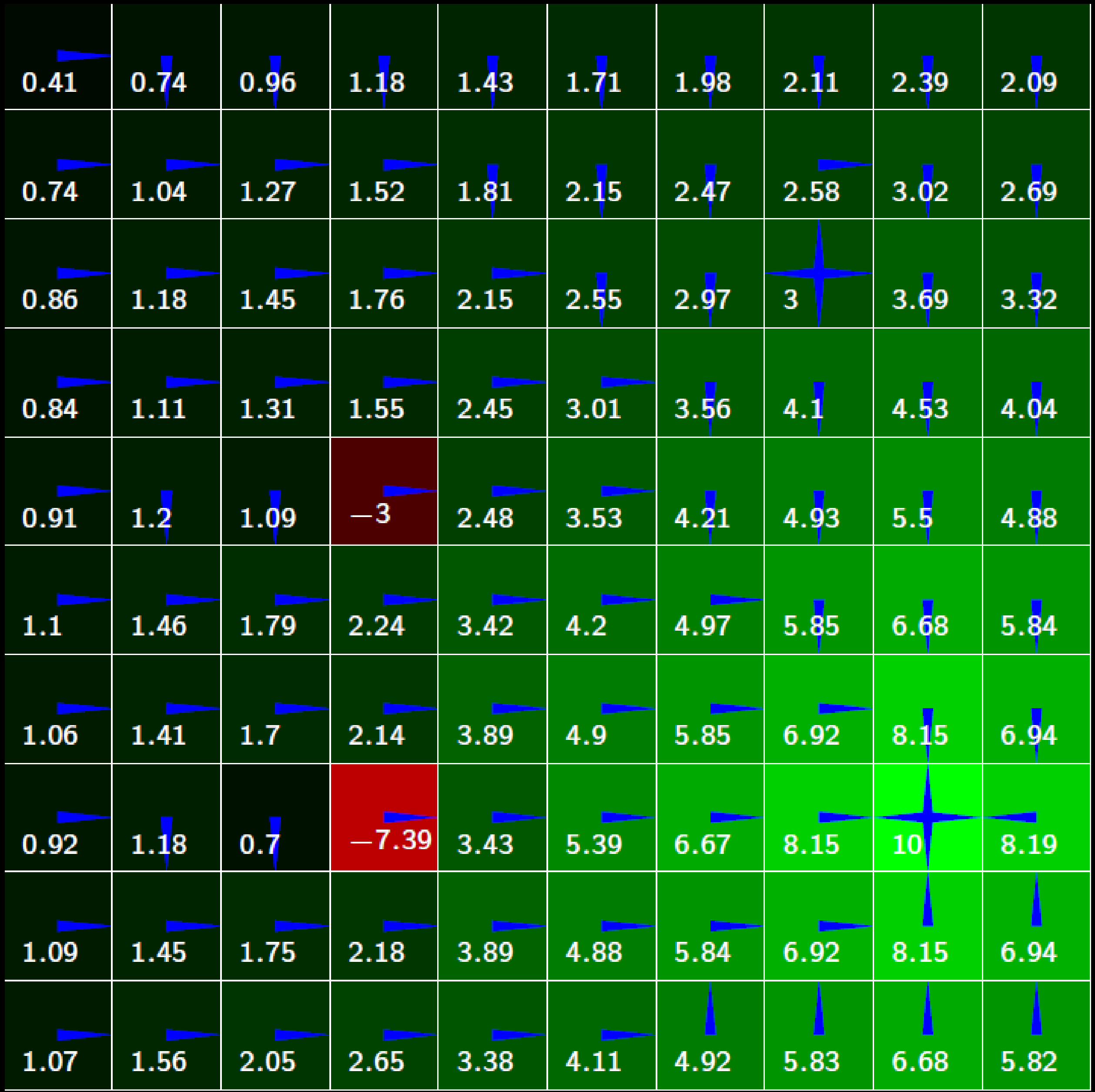


-0.43 -0.06 0.07 0.47 0.73 1.17 1.51 1.84 1.5 1.02
-0.07 0.14 0.55 0.77 1.24 1.58 2.01 2.4 2.12 1.51
-0.06 0.36 0.55 1.03 1.41 1.95 2.42 3 2.69 2.23
-0.1 0.06 0.33 0.22 1.29 1.71 2.6 3.13 3.82 3.15
-0.23 -0.03 -0.44 -4.53 0.74 2.39 3.05 4.35 4.94 4.38
-0.23 -0.12 0.1 0.25 2.16 2.94 4.38 5.37 6.47 5.57
-0.24 -0.12 -0.21 0.78 2.52 4.26 5.34 6.75 8.06 6.84
-0.3 -0.22 -0.96 -9 2.58 4.69 6.43 8.03 10 8.15
-0.33 -0.2 -0.25 0.74 2.51 4.22 5.36 6.75 8.07 6.84
-0.51 -0.34 0.04 0.62 1.97 3.01 4.32 5.44 6.51 5.62

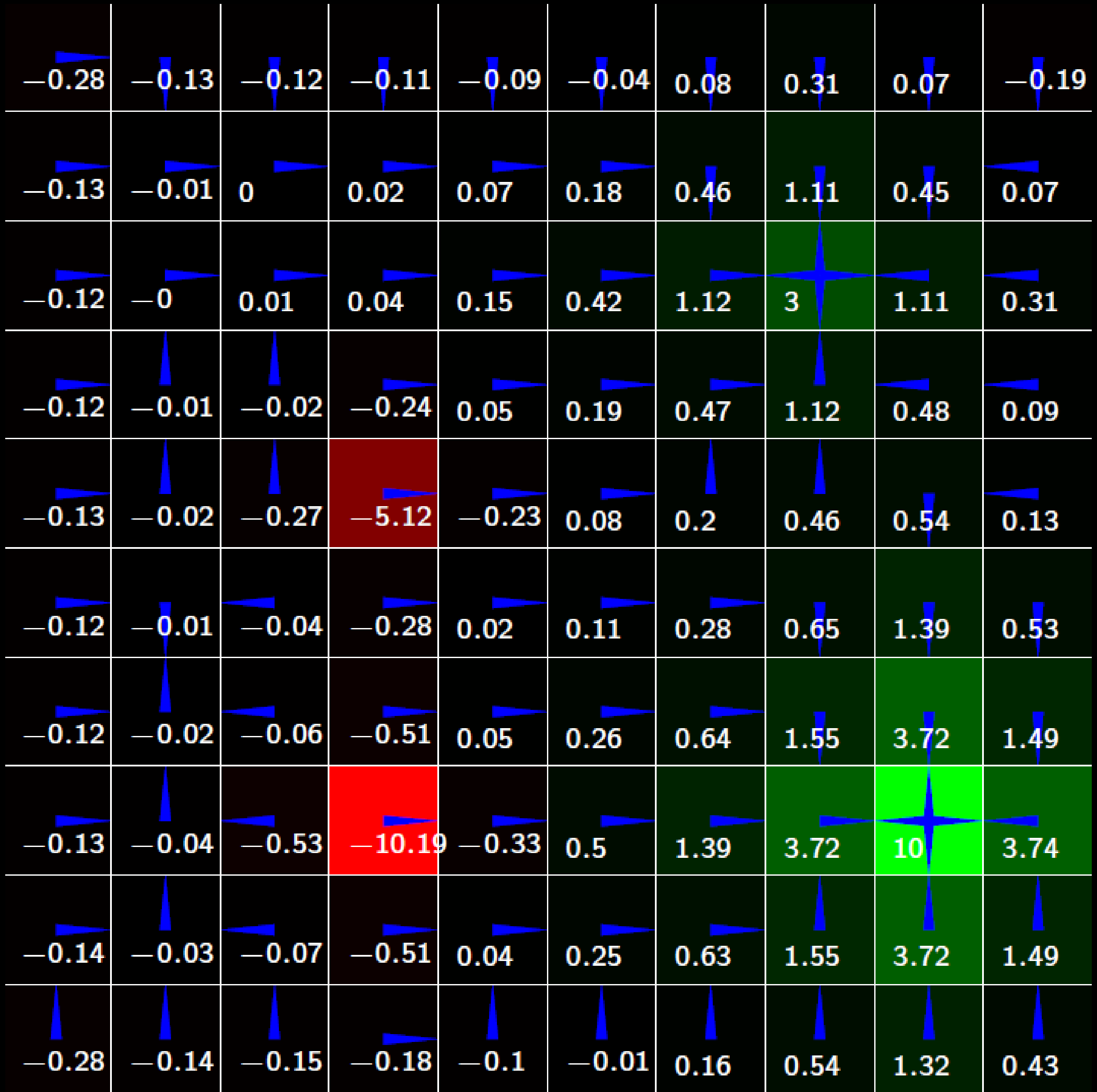

0.41 0.74 0.96 1.18 1.43 1.71 1.98 2.11 2.39 2.09
0.74 1.04 1.27 1.52 1.81 2.15 2.47 2.58 3.02 2.69
0.86 1.18 1.45 1.76 2.15 2.55 2.97 3 3.69 3.32
0.84 1.11 1.31 1.55 2.45 3.01 3.56 4.1 4.53 4.04
0.91 1.2 1.09 −3 2.48 3.53 4.21 4.93 5.5 4.88
1.1 1.46 1.79 2.24 3.42 4.2 4.97 5.85 6.68 5.84
1.06 1.41 1.7 2.14 3.89 4.9 5.85 6.92 8.15 6.94
0.92 1.18 0.7 −7.39 3.43 5.39 6.67 8.15 10 8.19
1.09 1.45 1.75 2.18 3.89 4.88 5.84 6.92 8.15 6.94
1.07 1.56 2.05 2.65 3.38 4.11 4.92 5.83 6.68 5.82

Converged!

$\gamma = 0.9$



$\gamma = 0.5$



Summary

- Discussed **decision-making (planning)** schemes and how they fit into the AV stack
- Defined the **MDP** model for decision-making, including **goals, costs, payoff,** and **policies**
- Defined **Expected Cumulative Payoff,** which plays a key role in **optimizing actions over planning horizons**
- Used **value iteration** to determine the “value” of a particular state, which helps us determine the best action to take considering future payoff
- We generally assumed the transition and reward function are known exactly – but what if we don’t have access to this information?
  - Will post notes on basic Q-learning for RL!

Extra Slides

Models of Optimal Behavior

- In the finite-horizon model, the agent should optimize expected reward for the next H steps: $E[\sum_{t=0}^{H} r_t]$
  - Continuously executing H-step optimal actions is known as receding horizon control
- In the infinite-horizon discounted model, agent should optimize: $E[\sum_{t=0}^{H} \gamma^t r_t]$
  - Discount factor is between 0 and 1, can be thought of as interest rate (reward now is worth more than reward later)
  - Keeps the utility of an infinite sequence finite

Challenges

- Value iteration and Policy iteration are both standard, and no agreement on which is better in theory
- In practice, value iteration is preferred over policy iteration as the latter requires solving linear equations, which scales ~cubically with the size of the state space
- Real-world applications face challenges:
    1. Curse of modeling: Where does the (probabilistic) environment model come from?
    2. Curse of dimensionality: Even if you have a model, computing and storing expectations over large state-spaces is impractical

Mods to Dynamic Programming

Structured Dynamic Programming

- $R(s, a)$ and $U(s)$ can also be represented using a decision tree
- Structured value iteration and structured policy iteration performs updates on leaves of the decision trees instead of all the states
- Structured dynamic programming algorithms improve efficiency by aggregating states, and additive decomposition of reward and value functions

Approximate Dynamic Programming

- For large or continuous spaces, ADP is concerned with finding approx. optimal policies
- This is an active area of research that is conceptually similar to reinforcement learning
- Some approximation methods are:
  - Local approximation relies on the idea that close states have similar values (builds on kNN)
  - Global approximation uses a fixed set of parameters to approximate the value function over the entire state space, generally based on linear regression

Online Methods

- Online methods compute optimal action from current state
  - Expand tree up to some horizon
  - States reachable from the current state is typically small compared to full state space
- Heuristics and branch-and-bound techniques allow search space to be pruned
- Monte Carlo methods provide approximate solutions

Forward Search

Provides optimal action from current state *s* up to depth *d*

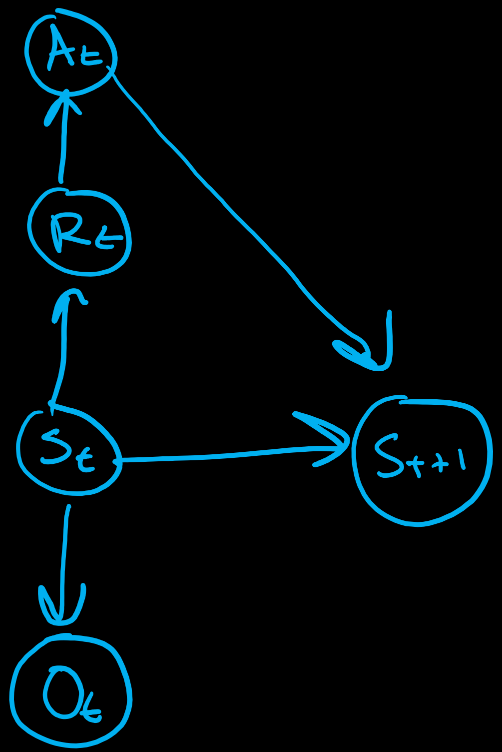
function $\textsc{SelectAction}(s, d)$
  if $d = 0$
    return $(\textsc{nil}, 0)$
  $(a^*, v^*) \leftarrow (\textsc{nil}, -\infty)$
  for $a \in A(s)$
    $q \leftarrow R(s, a)$
    for $s' \in S(s, a)$
      $(a', v') \leftarrow \textsc{SelectAction}(s', d - 1)$
      $q \leftarrow q + \gamma T(s' \mid s, a) v'$
    if $q > v^*$
      $(a^*, v^*) \leftarrow (a, q)$
  return (a^*, v^*)

Time complexity is $O((|S| \times |A|)^d)$

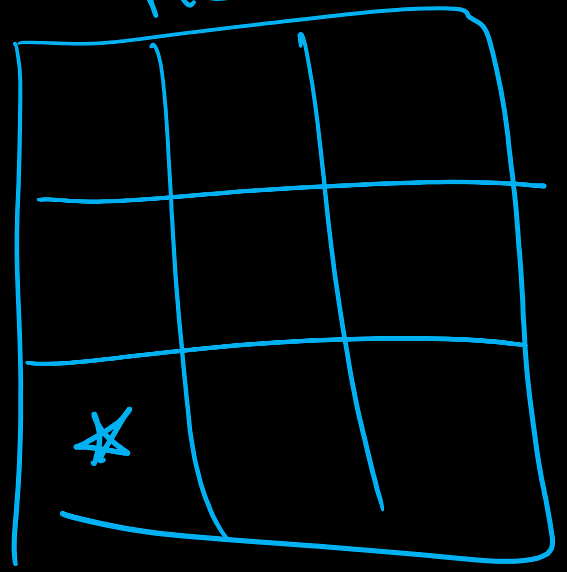
MDP Policy Summary

- MDPs represent sequential decision making problems using a transition and reward function
- Optimal policies can be found using dynamic programming
- Problems with large or continuous state spaces can be solved approximately using function approximation
- We generally assumed the transition and reward function are known exactly. On Wednesday, we'll relax this assumption.

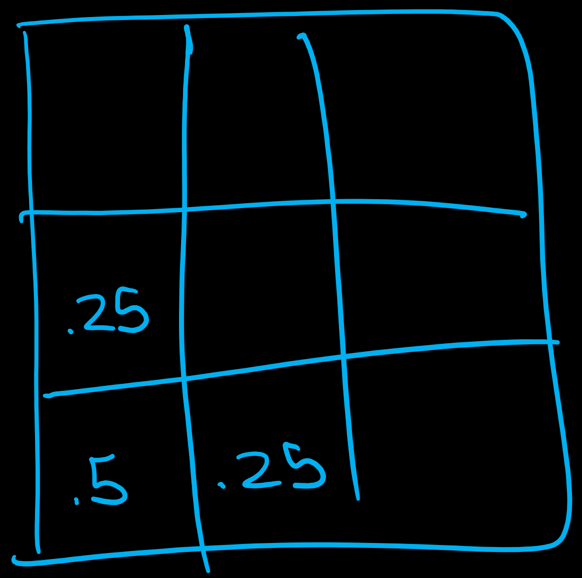
Partially Observable MDPs



MDP



belief state MDP



$$\pi : \underbrace{o_{0:t-1}, a_{0:t-1}} \rightarrow a_t$$

POMDP Executions

function POMDPPolicyExecution(π)

$b \leftarrow$ initial belief state

loop

Execute action $a = \pi(b)$

Observe o and reward r

$b \leftarrow$ UpdateBelief(b, a, o)

Alpha vectors

one step horizons

$$U^*(s) = \max_a R(s,a) \longrightarrow U^*(b) = \max_a \sum_s b(s) R(s,a)$$

an alpha vector α_a represents $R(\cdot, a)$

$\bar{b}$ is the vector of beliefs

$$U^*(b) = \max_a \left(\alpha_a^T \bar{b} \right)$$

alpha vector example

Imagine we have an exam tomorrow, but there is a non-negligible chance I forget about the exam. You can choose to either study or take the evening off.

- If you study and there is an exam, you ace it (R=100)
- If you study and there is no exam, you get nothing (R=0)
- If you relax and there is an exam, you fail and are stressed (R=-100)
- If you relax and there is no exam, you are very happy (R=100)

$$\bar{b} = \begin{pmatrix} P_E \\ 1-P_E \end{pmatrix} \qquad \alpha_{study} = \begin{pmatrix} 100 \\ 0 \end{pmatrix} \qquad \alpha_{relax} = \begin{pmatrix} -100 \\ 100 \end{pmatrix}$$

alpha vector example

$$\alpha_{study}\, b = \alpha_{relax}\, b$$

$$100 \cdot P_E = -100 P_E + 100(1 - P_E) \rightarrow P_E = .33$$

Why are POMDPs hard to solve?

- Combinatorial explosions!
  - H-step conditional plans: (|O|^h-1)/(|O|-1), so the number of policies is A^ (|O|^h-1)/(|O|-1)
  - For a two action two observation problem, there are 2^63 six step conditional plans
- Instead of solving exactly, we can approximate value iteration and/or solve offline
- There are many great solvers available