State transition:

I implemented the state transition probability function the way I did, since I believe it captures all possible scenarios, and raises errors if the input makes no sense. A naive implementation might fail silently, ie. give erronous values without alerting the caller, thus not being very fault tolerant.

Mapping:

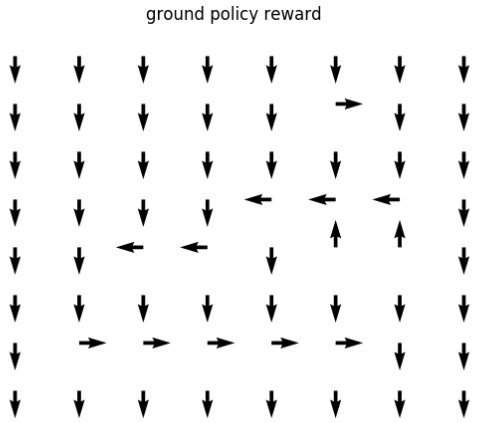
In my implementation a 1D mapping of the value function would make plotting the value function again unecessarily complicated. I therefore chose to let the value function map a 2D position to a value. That is, it maps the state, position in maze, to a value. The state itself might be considere 1D. The same is done for the policy function.

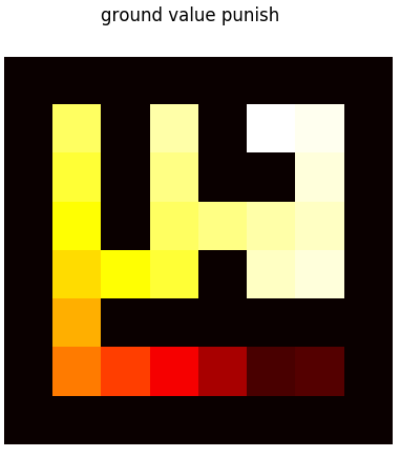
Convergence:

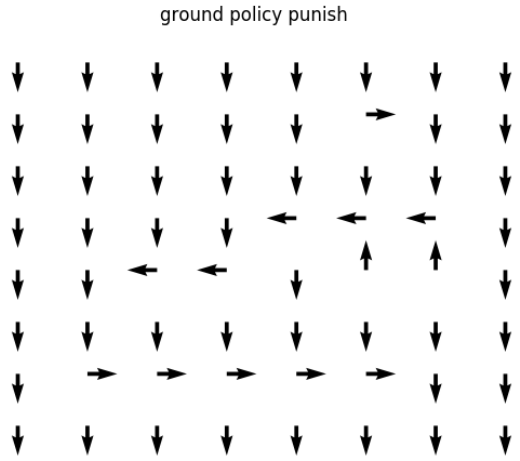
For value iteration we look at the change in the value function between iterations. When the change is below some threshold, the algorithm returns the result.

For policy iteration, the algorithm has converged when the policy is stable, ie. no change between iterations.

Ground Truth:







With discount 0.9, the punish and reward systems ended up with the same policy. The value functions were different of course.
When the discounting factor is sufficiently small, the DP algorithms terminate prematurely, possibly due to the way I've implemented the convergence check, or rounding errors for floating points. The policy is therefore only optimal near the goal, due to the few iterations. I have not looked into this yet.

I don't understand how it makes sense to compare results with different discounts factor values with the ground truth, which is valid when the discount is 0.9.

The outer loop of the VI algo need more iterations than the PI algorithm to obtain the same value function accuracy. However the PI algo has a lot of work in the inner loops in the policy evaluation, which is almost identical to VI. It would depend on what sort of optimization has been done to the algorithms' implementation. I find that VI was easier to implement, but many are of the opinion that PI is generally superior.