Technical Report

Hard Wall Stochastic Control based on Hallucination-EM and Power-EP

Max Welling[*]
Bren School of Computer and Information Science
University of California Irvine
Irvine, USA
welling@ics.uci.edu

Abstract

We study stochastic control problems in the presence of hard wall constraints. Walls are incorporated in the dynamics of the agent by restricting its domain and hence perturbing the noise process close to the walls. A novel penalty term is introduced for bouncing off a wall. To efficiently search for a good policy we propose the "hallucination expectation maximization" algorithm which iteratively maps the problem onto a non-Gaussian dynamical system. Hallucination weights anaesthetize the agent to render its local decisions optimal for the global planning problem. The E-step of HEM is solved using power-EP.

1 Introduction

Stochastic optimal control has a very long history in engineering, and its principles are widely applied in fields as diverse as biology, neuroscience, artificial intelligence and so on. Perhaps surprisingly, little seems known about control problems in the presence of hard, reflective walls which are prevalent in urban environments relevant to robotics. The goal of this contribution is to study the interaction between control, uncertainty and hard walls. Can a robot device a policy that exploits walls to its benefit? Can we learn policies that avoid walls because the cost of bumping into one is high? Figure 1 shows an example of an agent driving into a wall in an attempt to reduce the uncertainty in its position in order to hit the goal more precisely. The algorithm we propose can robustly learn a policy for this type of problem in a matter of minutes.

Stochastic control can be viewed as subfield of reinforcement learning. In that context it is relevant that we study problems with continuous location and control variables. Although we are interested in continuous time, the non-Gaussian nature of the problems we study has so far prevented us formulating the algorithm in this setting. Unlike most RL problems we assume that the environment is known to the agent, which simplifies it to a planning problem. We also opted to use a Markov decision process setting (MDP) where we know our precise location at all times[1] Stochastic or completely unknown environments and noisy

[*]On sabbatical at Radboud University Nijmegen, Dept. Biophysics, Netherlands.

[1]Incidentally, since our policy is in fact independent of $\mathbf{x}$, it doesn't matter whether the agent

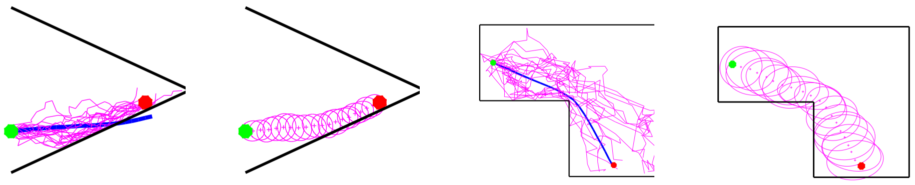

Figure 1: **Left:** Agent driving against a wall in an attempt to reduce its uncertainty in space in order to better locate the target. The blue solid line represents the concatenation of the control vectors $\mathbf{b}_{1:T}$. Magenta lines represent 20 samples from the policy driven, wall corrected dynamics of the agent. Note that the agent never actually drives inside the wall because the policy is continuously corrected by the wall constraints. **Middle Left:** Gaussian marginals of the "reward corrected dynamical system" computed by power-EP. **Middle Right & Right:** Agent balancing the cost of hitting the wall with the cost of control and finding the target with high noise dynamics (longer paths are more expensive).

measurements result in complications that we wanted to separate from our current focus, namely the presence of walls.

2 Stochastic Control

We will be interested in control problems in discrete time, $t = 1 : T$ and continuous space, $\mathbf{x}_t \in \mathbb{R}^d$, $\forall t$ with d referring to the dimensionality of space. We will denote with $P(\mathbf{x}_{1:T}) = P(\mathbf{x}_1) \prod_{t=1}^{T-1} P(\mathbf{x}_{t+1}|\mathbf{x}_t, \boldsymbol{\varphi}_t(\mathbf{x}_t))$ the controlled dynamics of the agent where $P(\mathbf{x}_1)$ is the initial distribution at $t = 1$ and $\boldsymbol{\varphi}_t(\mathbf{x}_t)$ is the control signal at time t. We will call the set $\{\boldsymbol{\varphi}_t,\ t = 1 : T\}$ a policy for the agent. The optimal policy is the policy that results in a maximal reward on average. A reward $R_s(\mathbf{x}_s)$, labeled with s, can be a function of a subset of $\mathbf{x}_{1:T}$, i.e. $\mathbf{x}_s \subseteqq \mathbf{x}_{1:T}$. Of particular interest are a goal reward, $R_T(\mathbf{x}_T)$, defined only at $t = T$, and control "rewards" which are independent of $\mathbf{x}$. It would be more appropriate to speak of control penalties which will always be interpreted as negative rewards. Putting this together, we arrive at the following objective function for the agent to maximize,

$$\mathcal{R}(\boldsymbol{\varphi}) = \int \mathcal{D}\mathbf{x} \left[P(\mathbf{x}_1) \prod_{t=1}^{T-1} P(\mathbf{x}_{t+1}|\mathbf{x}_t, \boldsymbol{\varphi}_t) \right] \left(\sum_{s=1}^{S} R_s(\mathbf{x}_s) \right) = \sum_s \mathbb{E}\left[R_s(\mathbf{x}_s)\right]_{P_s} \quad (1)$$

$$(2)$$

In the above equation we have defined $\int \mathcal{D}\mathbf{x}$ to be the integral over all possible paths $\mathbf{x}_1, .., \mathbf{x}_T$ and P_s to be the marginal of $P_{1:T}$ over the subset $\mathbf{x}_s$.

For simplicity we have chosen the functional form of the control to be independent of $\mathbf{x}$: $\mathbf{x}_{t+1} = \mathbf{x}_t + \mathbf{b}_t + \varepsilon_t$ resulting in $\boldsymbol{\varphi}(\mathbf{x}_t) = \mathbf{b}_t$. The goal reward will be modeled as a tight Gaussian blob around the goal location,

$$R_T^{\text{goal}}(\mathbf{x}_T) = J^{\text{goal}}\ \exp\left(-\frac{1}{2\sigma_G^2} (\mathbf{x}_T - \boldsymbol{\mu}_G)^2 \right) \quad (3)$$

Control rewards (negative penalties) are assumed to have the following form,

$$R_t^{\text{control}}(\boldsymbol{\varphi}_t) = \begin{cases} -\frac{1}{2\sigma_b^2} ||\mathbf{b}_t||_{L_2}^2 & \text{if } ||\mathbf{b}_t||_{L_2} \le \theta \\ -\infty & \text{if } ||\mathbf{b}_t||_{L_2} > \theta \end{cases} \quad (4)$$

knows its precise location or not.

The quadratic penalty is trivially generalized to an arbitrary functional form (as long as it is differentiable). The reason we choose this type of truncated penalty is twofold. Firstly, it models our inability to apply control above a certain threshold. Secondly, we can now restrict the domain of $\mathbf{b}$ to be those values for which the reward is finite. On this domain the reward is lower bounded, and by adding this lower bound it can be defined positive. Clearly, adding a constant reward will not have any impact on the optimal policy, even though it will have an impact on convergence properties of the algorithm we describe below.

3 Hard Wall Dynamics

A standard assumption in continuous control problems is gaussian noise for the dynamics of the agent. Our main motivation is to develop a framework to study continuous control problems in the presence of "hard walls". Since one feature of a hard wall is that the probability of being inside the wall is zero, this Gaussian noise assumption can no longer be maintained. This leads us to a more general definition of stochastic dynamics given by the general form,

$$P(\mathbf{x}_{t+1}|\mathbf{x}_t, \boldsymbol{\varphi}_t) = \frac{\phi_t(\mathbf{x}_{t+1}, \mathbf{x}_t, \boldsymbol{\varphi}_t)}{z_{t+1}(\mathbf{x}_t)}; \qquad z_{t+1}(\mathbf{x}_t) = \int \mathrm{d}\mathbf{x}_{t+1}\ \phi_t(\mathbf{x}_{t+1}, \mathbf{x}_t, \boldsymbol{\varphi}_t) \tag{5}$$

For a hard wall, we will assume that far away from a wall the agent still moves under Gaussian dynamics, but that close to the wall the Gaussian noise distribution will be truncated to lie within the allowed domain, i.e.

$$\phi_t(\mathbf{x}_{t+1}, \mathbf{x}_t, \boldsymbol{\varphi}_t) = \mathcal{N}(\mathbf{x}_{t+1}|\mathbf{x}_t + \boldsymbol{\varphi}_t, \Sigma_{\text{dyn}})\ \psi(\mathbf{x}_{t+1}, \theta) \tag{6}$$

where $\psi(\mathbf{x}, \theta)$ is a "Heaviside" function that assumes the value $\psi = 1$ in the allowed domain and $\psi = 0$ everywhere else. The parameters θ specify the allowed domain. In the following we will refer to this as "domain restricted dynamics".

To better understand this dynamics we imagine a forward simulation from it. At each time step we first propose a new location for the agent sampled from $\mathcal{N}(\mathbf{x}_{t+1}|\mathbf{x}_t + \boldsymbol{\varphi}_t, \Sigma_{\text{dyn}})$. This sample is accepted if it falls in the allowed domain but is rejected if it falls outside the allowed domain. In that case we simply try again until we succeed in sampling within the allowed domain. This process is similar but not equivalent to reflective wall dynamics where a sample outside the allowed domain is reflected instead of resampled. If we assume very small time increments and scale the noise variance down as $\sigma_d^2 \rightarrow \sigma_d^2/\delta t$ then in the case of isotropic noise the two processes become increasingly similar as $\delta t \rightarrow 0$. To see this intuitively one observes that for very small noise variance the only discrepancy between the two types of dynamics will occur very close to the wall position. At the wall position however, reflection (described by a mixture of two Gaussians with one Gaussian located symmetrically behind the wall), and truncation are identical. This is no longer true for finite time increments. In this case we can think of truncation as an approximation to reflective boundaries (at least for isotropic or wall-aligned noise). Note however that for finite time increments truncation is much easier to handle than reflection for instance close to *corners* where the noise distribution for reflection may not even have a (known) analytic solution.

3.1 Cost of Bumping into a Wall

By incorporating the hard walls into the dynamics, bumping into a wall is free of cost. We wish to include explicit penalty terms for hitting the wall. If the cost of hitting a wall is high compared to other costs the agent will devise a policy that steers free of the walls.

The expected penalty for an encounter with the wall at time t will be proportional to the probability of hitting the wall at that time. Reversely, the expected reward will be proportional to the probability of missing it, which is given by,

$$P_t^{\text{miss}} = \int d\mathbf{x}_{t-1} d\mathbf{x}_t \; P_{t-1}(\mathbf{x}_{t-1}) \, \mathcal{N}(\mathbf{x}_t | \mathbf{x}_{t-1} + \boldsymbol{\varphi}_{t-1}, \Sigma_{\text{dyn}}) \; \psi(\mathbf{x}_t) = \mathbb{E}\left[z_t(\mathbf{x}_{t-1})\right]_{P_{t-1}} \tag{7}$$

In words, the probability of missing the wall is given by the probability of being at position $\mathbf{x}_t$ at time t and then jumping to a location $\mathbf{x}_{t+1}$ at time $t+1$ inside the allowed domain using unconstrained dynamics. This leads to the following definition of the total expected reward,

$$\mathcal{R}^{\text{wall}} = \sum_{t=2}^{T} J^{\text{wall}} \; P_t^{\text{miss}} = \sum_{t=1}^{T} J^{\text{wall}} \; \mathbb{E}\left[z_t(x_{t-1})\right]_{P_{t-1}} \tag{8}$$

In comparing this expression with Eqn.1 we find that the reward-function at time t is thus given by,

$$R_{t-1}^{\text{wall}}(x_{t-1}) = J^{\text{wall}} \; z_t(x_{t-1}) \tag{9}$$

For convenience, this reward function is defined one time step before the agent would actually physically hit the wall.

4 Hallucination Expectation Maximization

We will now consider the problem of learning an optimal policy, i.e. a policy that would maximize the total expected reward $\mathcal{R}$ of eqn. 1. Our approach will be to map the problem iteratively to a dynamical system at which point the problem reduces to an instance of EM.

We will use the following relation, $\sum_i y_i \geq \prod_i y_i^{\gamma_i} \, \gamma_i^{-\gamma_i}$ for weights $\boldsymbol{\gamma}$ that satisfy $\sum_i \gamma_i = 1$ and $y_i \geq 0$. Equality is attained if we choose $\gamma_i \propto y_i$. Applying this to the sum of rewards in Eqn.1, we find,

$$\mathcal{R}(\boldsymbol{\varphi}) \geq \mathcal{B}(\boldsymbol{\varphi}, \boldsymbol{\gamma}) = \int \mathcal{D}\mathbf{x} \left[P(\mathbf{x}_1) \prod_{t=1}^{T-1} P(\mathbf{x}_{t+1} | \mathbf{x}_t, \boldsymbol{\varphi}_t) \right] \left(\prod_{s=1}^{S} R_s(\mathbf{x}_s)^{\gamma_s} \; \gamma_s^{-\gamma_s} \right) \tag{10}$$

This bound is not expected to become tight because unlike y_i, the rewards $R_s(\mathbf{x}_s)$ are functions of $\mathbf{x}$ (over which we average) and equality cannot be simultaneously achieved for all combinations of $\mathbf{x}$. The approximation is expected to be accurate when the rewards $R_s(\mathbf{x}_s)$ do not vary significantly within the high probability regions of the marginals $P(\mathbf{x}_s)$.

In practice, the normalized hallucination weights $\boldsymbol{\gamma}$ become too small for the rewards to express themselves. We suspect this is caused by the gap between the bound and the true objective. Fortunately, this issue can be corrected by applying another bound in the reverse direction,

$$\mathcal{B}(\boldsymbol{\varphi}, \boldsymbol{\gamma}) \leq \mathbb{E}\left[\prod_{s=1}^{S} R_s(\mathbf{x}_s)^{\beta \gamma_s} \right]^{\frac{1}{\beta}} \prod_s \gamma_s^{-\gamma_s} \tag{11}$$

as long as $R_t \geq 0$ everywhere. The impact of this "counter-bound" on the algorithm described below will be that we now normalize the weights to β instead of 1, effectively working as a "gain-factor" on those weights. For convenience we will incorporate the gain-factor into the weights by redefining $\gamma_s \rightarrow \beta \gamma_s$. The value of β is used as a hand-adjustable parameter in this paper (typically set to 10).

It is natural to think of the of the terms $R_s^{\gamma_s}$ as potentials for a dynamical system which we will call "*reward corrected dynamics*". This is in particular true when the labels s indicate time t,

$$P_{1:T}(\boldsymbol{\gamma}, \boldsymbol{\varphi}) = \frac{1}{\mathcal{Z}} P(\mathbf{x}_1) \prod_{t=1}^{T-1} P(\mathbf{x}_{t+1}|\mathbf{x}_t, \boldsymbol{\varphi}_t) \prod_{t=1}^{T} R_t^{\gamma_t}(\mathbf{x}_t) \tag{12}$$

again as long as $R_t \geq 0$ everywhere.

The weights $\boldsymbol{\gamma}$ have an interesting interpretation. Imagine that in order to get to a goal at time T one needs to sacrifice reward at earlier times. The weights handle this situation by down-weighting (and hence sacrificing) the rewards during the initial part of the journey and up-weighting the rewards during the last part. One could say that the agent has access to β units of a hallucinating drug, the intake of which s/he has to distribute over T time slices. By using the drug mostly at the time of sacrifice (in order to collect the loot at later times) the agent can subdue the incurred pain. The important consequence is that the agent can make decisions *locally in time*, i.e. no look ahead is necessary. Given $\boldsymbol{\gamma}$, the decision process can now be modeled with Markovian dynamics which is expressed mathematically by the fact that $\mathcal{B}$ (for fixed $\boldsymbol{\gamma}$) is equivalent to the partition function of the Markov process $P_{1:T}(\boldsymbol{\gamma}, \boldsymbol{\varphi})$.

Based on $\log \mathcal{B}(\boldsymbol{\gamma}, \boldsymbol{\varphi})$ as the objective function we propose the following iterative policy optimization algorithm, which we call "Hallucination Expectation Maximization ", or HEM for short:

Initialization: Initialize weights $\{\gamma_t,\ t = 1:T\}$ and policy parameters $\{\mathbf{b}_t,\ t = 1:T\}$.

E-Step: Compute the marginals $P_t^{\gamma}(\mathbf{x}_t)$ and $P_{t+1,t}^{\gamma}(\mathbf{x}_{t+1}, \mathbf{x}_t)$ of the reward corrected dynamical system $P_{1:T}(\boldsymbol{\gamma}, \boldsymbol{\varphi})$, Eqn.12.

M-step: Update the policy $\boldsymbol{\varphi}$ using gradient ascent, $\boldsymbol{\varphi}_t \leftarrow \boldsymbol{\varphi}_t + \eta \nabla_{\boldsymbol{\varphi}_t} \log \mathcal{B}(\boldsymbol{\gamma}, \boldsymbol{\varphi})$, where

$$\begin{aligned}\nabla_{\boldsymbol{\varphi}_t} \log \mathcal{B}(\boldsymbol{\gamma}, \boldsymbol{\varphi}) = {} & \mathbb{E}[\nabla_{\boldsymbol{\varphi}_t} \log P(\mathbf{x}_{t+1}|\mathbf{x}_t, \boldsymbol{\varphi}_t)]_{P_{t+1,t}^{\gamma}} - \mathbb{E}[\nabla_{\boldsymbol{\varphi}_t} \log P(\mathbf{x}_{t+1}|\mathbf{x}_t, \boldsymbol{\varphi}_t)]_{P_{t+1|t} P_t^{\gamma}} \\ & + \gamma_t \nabla_{\boldsymbol{\varphi}_t} \mathbb{E}[\log R_t(\mathbf{x}_t)]_{P_t^{\gamma}}\end{aligned} \tag{13}$$

Observe the average over $P_{t+1|t} P_t^{\gamma}$ which means that at time t we use the marginal distribution derived from the reward corrected dynamics, but then take one step of domain restricted dynamics. Explicitly, for $\mathbf{b}_t$ we use,

$$\nabla_{\mathbf{b}_t} \log P(\mathbf{x}_{t+1}|\mathbf{x}_t, \boldsymbol{\varphi}_t) = \mathbf{x}_{t+1} - \mathbf{x}_t - \mathbf{b}_t \tag{14}$$

The terms $\nabla_{\boldsymbol{\varphi}_t} \mathbb{E}[\log R_t(\mathbf{x}_t)]_{P_t^{\gamma}}$ are easily derived for the $\mathbf{x}$-independent control penalties defined in section 2. The penalty of hitting a wall will add a term, $\gamma_t^{\text{wall}}\ \mathbb{E}[\nabla_{\boldsymbol{\varphi}_t} \log P(\mathbf{x}_{t+1}|\mathbf{x}_t, \boldsymbol{\varphi}_t)]_{P_{t+1|t} P_t^{\gamma}}$, which proportional to the second term of 13.

H-Step: Update hallucination-weights (or "when do I eat my magic mushrooms"),

$$\gamma_t \leftarrow \frac{1}{Z_t} \exp\left(\mathbb{E}[\log R_t(\mathbf{x}_t)]_{P_t^{\gamma}}\right), \qquad \gamma_t \leftarrow \beta \gamma_t \tag{15}$$

We claim that this algorithm is guaranteed to increase $\mathcal{B}$ at every step of learning. See appendix A for a sketch of proof.

4.1 Solving the E-step with Power-EP

The greatest challenge for the successful execution of HEM is an adequate approximation of the E-step since it requires inference in a nonlinear dynamical system with non-Gaussian noise. Before we move on to describe our approach, we first discuss what does *not* work.

One possibility would be to apply a particle filter to sample forward and compute importance weights in a backwards pass (see e.g. [3]). This method is doomed for the current application because the goal reward applies the dominant correction to the dynamics. Hence, most particles sampled in the forward pass will be located very far away from the region where the smoothed distribution has significant mass (unless the policy is already very good) resulting in very high variance. What is needed is an iterative procedure that moves itself into high posterior probability regions of space. An MCMC algorithm for the entire chain, using e.g. Hybrid Monte Carlo to deform paths, is feasible but expected to mix slowly due to the strong correlations over time. Perhaps a variational method should help (see e.g. [1])? Unfortunately, variational approximations minimize the KL-divergence in such a way that the approximate posterior can not have probability mass in regions where the true density has not. The presence of walls will thus cause the variance to shrink to 0 for any variational posterior with infinite tails (such as a Gaussian). Fortunately, expectation propagation [4] does not share this degeneracy and a Gaussian with finite variance can be used to approximate the posterior distribution between walls. Unfortunately, in practice, the plain-vanilla EP updates become instable. However, a recent extension of EP, known as power-EP (PEP) [5, 8] with $\alpha = 0$ (exactly halfway between variational ($\alpha = -1$) and EP ($\alpha = 1$) is both stable and non-degenerate.

Perhaps equally important as stability and non-degeneracy is the fact that PEP allows us to guide the inference process towards the high probability regions of the posterior. This is achieved in two ways: 1) "*self-annealing*", where we initialize messages with very high variance and use damped PEP updates to slowly decrease the variance towards the final solution. For reasons of stability, we only turn on the impact of the wall constraints once the marginals have annealed themselves roughly inside the allowed domain. 2) placement of landmarks. We use Gaussian potentials at times and locations of our choice that will initially force the solution to pass near that landmark. Once the solution has stabilized we remove these landmarks by decreasing their precision to zero. A particularly useful landmark is placed at the goal location. One could claim that it is "unfair" to provide the algorithm with the location of the goal. However, we emphasize that we are interested in the effects of the wall corrected noise process rather than in the process of finding a good path from start to goal. One can imagine running a specialized optimization algorithm to find the optimal noise free solution and use this to set the landmarks.

To derive the PEP updates we first break up the probability distribution Eqn.12 into a product of factors (ignoring multiplicative constants such as control penalties):

$$P_{1:T}(\mathbf{x}_{1:T}) \propto \tag{16}$$
$$[P(\mathbf{x}_1)] \left[\prod_{t=1}^{T-1} N(\mathbf{x}_{t+1}|\mathbf{x}_t, \boldsymbol{\varphi}_t)\right] \left[\prod_{t=1}^{T} \psi(\mathbf{x}_t, \theta)\right] \left[\prod_{t=1}^{T-1} (z_{t+1}(\mathbf{x}_t, \boldsymbol{\varphi}_t))^{\gamma_t^{\text{wall}}-1}\right] \left[\left(R_T^{\text{goal}}(\mathbf{x}_T)\right)^{\gamma_G}\right]$$

In PEP we need to raise the interactions to a power $1/\wp$ which is related to α as $\alpha = (2/\wp) - 1$ [5]. We chose $\wp = 2 \rightarrow \alpha = 0$. We maintain Gaussian messages to approximate each single site potential (including potential landmarks during the initialization) and two Gaussian messages for each Gaussian interaction term $\mathcal{N}(\mathbf{x}_{t+1}|\mathbf{x}_t) \approx m_t(\mathbf{x}_t) m_{t+1}(\mathbf{x}_{t+1})$. The Gaussian terms $P(\mathbf{x}_1)$ and $R_T^{\text{goal}}(\mathbf{x}_T)$, as well as the landmarks are Gaussian and hence equal to their messages. At all times the product of the messages will be equal to the marginal distribution, $P_t^{\gamma}(\mathbf{x}_t) = \prod_a m_a(\mathbf{x}_t)$ where a labels all messages that depend on $\mathbf{x}_t$.

The remaining three terms are updated jointly and require sampling. The details of that update are provided in the algorithm box in appendix[2] B. We observe two facts about this update. Firstly, by changing to relative coordinates we have managed to analytically

[2]The update is valid for a slightly more general policy $\boldsymbol{\varphi}_t(\mathbf{x}_t) = A_t \mathbf{x}_t + \mathbf{b}_t$.

integrate out half of the number of dimensions. This is beneficial for both speed and accuracy of the resulting algorithm. Secondly, processing the term $z_{t+1}(\mathbf{x}_t)^{\gamma_t^{\text{wall}}-1}$ in lines block 8 has computational complexity $\mathcal{O}(N_{\text{sample}}^2)$which is in line with particle smoothing algorithms.

5 Experiments

In the following we will discuss some illustrative examples of control problems. In all our problems the agents have to reach a goal location modeled with a Gaussian function Eqn.3 where we have used $\sigma_G = 0.1$. The same variance was used for $P(\mathbf{x}_1)$. The strength of the goal reward, J^{goal} was usually large (say 1000). In the first experiment, shown in Figure 1 (left two Figures), the agent has to find the goal (green dot) in the presence of significant noise. Black lines represent the wall. The obtained policy is interesting in the fact that it uses the wall in order to reduce the uncertainty in its location. This can clearly be seen from the samples drawn from the policy driven dynamics. Due to its reduced uncertainty it can collect a higher reward from the target location. The only reason it did not hit the wall more violently is the fact that it has to pay a relatively high price for hitting it. In the second figure we show the resulting marginals obtained from PEP. Note that these reflect the various rewards and penalties which means that their variance is usually smaller than for the forward simulation.

The second example (illustrated in the right two plots of Figure 1) concerns an agent that has to reach a goal located around a corner. In doing so it has to negotiate control costs and the cost of hitting the wall. Since we choose the latter relatively high it converged on a policy that steers clear of the wall on average. Relaxing it resulted in a policy that “hugged the wall” before clearing the corner.

In a third example, illustrated in Figure 2 we implemented two agents that interact through a joint reward given by $R_{abt}^{\text{int}}(\mathbf{x}_{at}, \mathbf{x}_{bt}) = (\beta + ||\mathbf{x}_{at} - \mathbf{x}_{bt}||)^{\alpha}$. The agents follow independent dynamics. The reward corrected dynamics however is a coupled system of two agents interacting through this reward, see Figure 2. The problem can be solved efficiently using PEP again, but now also exchanging messages between the agents. Note that in this case goal, control, wall rewards/penalties need to be traded off with proximity penalties.

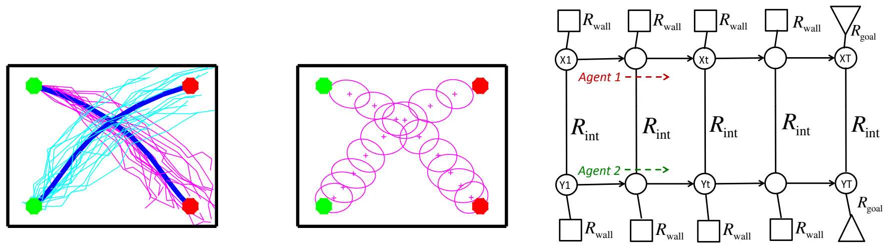

Figure 2: **Left Two Figures:** Two agents with a penalty for being within each others vicinity. The agents take a small de-tour to avoid each other’s company. **Right Figure** Chain graph representation for the two-agent, reward corrected dynamical system.

6 Conclusions

We have presented an three-step algorithm for solving stochastic control problems. The computational bottleneck of this algorithm, namely inference in a non-Gaussian continuous dynamical system was tackled with power-EP. Through a number of illustrative examples we have shown the algorithm holds promise for more complex applications. We emphasize that our contribution goes beyond simple obstacle avoidance as evidenced by the discovery of subtle policies that *exploit* the wall (rather than avoid it) in order to reduce its uncertainty.

We were not the first to propose algorithms based on EM. For instance, in [2] and [7] algorithms are proposed for reinforcement learning. Our problem setting is different in the sense that we solve control problems over a fixed time interval with goal states only at the last time point. This, unlike the infinite horizon discounted reward problems, leads to time-dependent control policies. The use of hallucination weights and wall hitting policies is also new to the best of our knowledge.

The presented algorithm was tested on a problem setting that was stripped of many important complexities. For instance, we used a very simple $\mathbf{x}$-independent control signal. We implemented and experimented with a linear policies of the form $\boldsymbol{\varphi}_t(\mathbf{x}_t) = A_t\mathbf{x}_t + \mathbf{b}_t$ but the resulting behavior did not contribute to clarifying the interaction of reward, noise and control. The framework is however easily extended to $\mathbf{x}$-dependent policies by a parametric function $\boldsymbol{\varphi}_t(\mathbf{x}_t) = f(\mathbf{x}_t, \theta_t)$ and learning the parameters θ. The required joint statistics over $\mathbf{x}_t, \mathbf{x}_{t+1}$ can be obtained within EP.

Acknowledgements

This work was supported by the Dutch National Science Foundation (NWO).

A Proof of HEM Convergence

First write $\log \mathcal{B}(\boldsymbol{\gamma}, \boldsymbol{\varphi}) = \log \mathcal{Z}(\boldsymbol{\gamma}, \boldsymbol{\varphi}) + \mathcal{H}(\boldsymbol{\gamma})$ with $\mathcal{Z}$ the partition function of $P_{1:T}(\boldsymbol{\gamma}, \boldsymbol{\varphi})$ (Eqn.12) and $\mathcal{H}(\boldsymbol{\gamma})$ the entropy of $\boldsymbol{\gamma}$: $\mathcal{H}(\boldsymbol{\gamma}) = -\sum_t \gamma_t \log \gamma_t$. The term $\log \mathcal{Z}$ can be further bounded as a negative variational free energy $\mathcal{Z}(\boldsymbol{\gamma}, \boldsymbol{\varphi}) \geq -\mathcal{F}(Q, \boldsymbol{\gamma}, \boldsymbol{\varphi})$ with $-\mathcal{F}(Q, \tilde{P}(\boldsymbol{\gamma}, \boldsymbol{\varphi})) = \int Q \log \tilde{P} - Q \log Q$ [6] where $\tilde{P}$ is the unnormalized part of the probability of Eqn.12. It is easy to verify that for $Q = P$ we have that $\log \mathcal{Z} = -\mathcal{F}$ and hence that $\log \mathcal{B} = \log \mathcal{Z} + \mathcal{H}$. The HEM algorithm can thus be interpreted as coordinate descent on $-\mathcal{F}(Q, \boldsymbol{\gamma}, \boldsymbol{\varphi}) + \mathcal{H}(\boldsymbol{\gamma})$ where the E-step computes all marginals $Q_t(\mathbf{x}_t) = P_t(\mathbf{x}_t)$ simultaneously given $\boldsymbol{\varphi}, \boldsymbol{\gamma}$ and the H and M steps maximize $\boldsymbol{\gamma}$ and $\boldsymbol{\varphi}$ respectively.

B PEP Edge Update

References

[1] Cdric Archambeau, Manfred Opper, Yuan Shen, Dan Cornford, and John Shawe-Taylor. Variational inference for diffusion processes. In *Advances in Neural Information Processing Systems 20*, pages 17–24. 2007.

[2] P. Dayan and G.E.Hinton. Using expectation-maximization for reinforcement learning. *Neural Computation*, 9(2):271–278, 1997.

[3] N. de Freitas A. Doucet S. Maskell M. Klaas, M. Briers and D. Lang. Fast particle smoothing: If i had a million particles. In *Proceedings of the 21st International Conference on Machine Learning*, 2006.

[4] T. Minka. Expectation propagation for approximate Bayesian inference. In *Proc. of the Conf. on Uncertainty in Artificial Intelligence*, pages 362–369, 2001.

Power-EP Update

```
1.   Input:$\theta, \gamma, \wp, \kappa, A_t, \mathbf{b}_t, \Sigma_{\text{dyn}}, \Sigma_t, \Sigma_{t+1}, \mu_t, \mu_{t+1}, V_t, V_{t+1}, \nu_t, \nu_{t+1}, N_{\text{sample}}$
     Output:$\Sigma_{t,\text{new}}, \Sigma_{t+1,\text{new}}, \mu_{t,\text{new}}, \mu_{t+1,\text{new}}, V_{t,\text{new}}, V_{t+1,\text{new}}, \nu_{t,\text{new}}, \nu_{t+1,\text{new}}$
2.   Remove message from marginal
2A.     $\Sigma_{t\setminus t+1}^{-1} = \Sigma_t^{-1} - \frac{1}{\wp}V_t^{-1}, \quad \boldsymbol{\mu}_{t\setminus t+1} = \Sigma_{t\setminus t+1}(\Sigma_t^{-1}\boldsymbol{\mu}_t - \frac{1}{\wp}V_t^{-1}\boldsymbol{\nu}_t)$
2B.     $\Sigma_{t+1\setminus t}^{-1} = \Sigma_{t+1}^{-1} - \frac{1}{\wp}V_{t+1}^{-1}, \quad \boldsymbol{\mu}_{t+1\setminus t} = \Sigma_{t+1\setminus t}(\Sigma_{t+1}^{-1}\boldsymbol{\mu}_{t+1} - \frac{1}{\wp}V_{t+1}^{-1}\boldsymbol{\nu}_{t+1})$
3.   Transform to relative coordinates:
3A.     $\Sigma_+ = A_t\Sigma_{t\setminus t+1}A_t^T + \Sigma_{t+1\setminus t}, \quad \boldsymbol{\mu}_+ = \frac{1}{2}(A_t\boldsymbol{\mu}_{t\setminus t+1} + \boldsymbol{\mu}_{t+1\setminus t})$
3B.     $\Sigma_- = A_t\Sigma_{t\setminus t+1}A_t^T - \Sigma_{t+1\setminus t}, \quad \boldsymbol{\mu}_- = A_t\boldsymbol{\mu}_{t\setminus t+1} - \boldsymbol{\mu}_{t+1\setminus t}$
4.   Compute moments for $\mathbf{x}_-$ variables:
4A.     $\Gamma^{-1} = \Sigma_+^{-1} + \frac{1}{\wp}\Sigma_{\text{dyn}}^{-1}$
4B.     $\mathbb{E}[\mathbf{x}_-] = \Gamma(\Sigma_+^{-1}\boldsymbol{\mu}_- + \frac{1}{\wp}\Sigma_{\text{dyn}}^{-1}\mathbf{b}), \quad \mathbb{E}[\mathbf{x}_-\mathbf{x}_-^T] = \Gamma + \mathbb{E}[\mathbf{x}_-]\mathbb{E}[\mathbf{x}_-]^T$
5.   Compute moments for $\mathbf{x}_+$ variables:
5A.     $\Lambda = \frac{1}{2}\Sigma_-\Sigma_+^{-1}, \quad \boldsymbol{\delta} = \boldsymbol{\mu}_+ - \Lambda\boldsymbol{\mu}_-$
5B.     $\mathbb{E}[\mathbf{x}_+] = \boldsymbol{\delta} + \Lambda\mathbb{E}[\mathbf{x}_-]$
5C.     $\mathbb{E}[\mathbf{x}_+\mathbf{x}_+^T] = \frac{1}{4}(\Sigma_+ - \Sigma_-\Sigma_+^{-1}\Sigma_-) + \boldsymbol{\delta}\boldsymbol{\delta}^T + \boldsymbol{\delta}\mathbb{E}[\mathbf{x}_-]^T\Lambda^T + \Lambda\mathbb{E}[\mathbf{x}_-]\boldsymbol{\delta}^T + \Lambda\mathbb{E}[\mathbf{x}_-\mathbf{x}_-^T]\Lambda^T$
5D.     $\mathbb{E}[\mathbf{x}_+\mathbf{x}_-^T] = \boldsymbol{\delta}\mathbb{E}[\mathbf{x}_-]^T + \Lambda\mathbb{E}[\mathbf{x}_-\mathbf{x}_-^T], \quad \mathbb{E}[\mathbf{x}_-\mathbf{x}_+^T] = \mathbb{E}[\mathbf{x}_+\mathbf{x}_-^T]^T$
6.   Transform back to normal coordinates:
6A.     $\mathbb{E}[\mathbf{x}_t] = A_t^{-1}(\mathbb{E}[\mathbf{x}_+] + \frac{1}{2}\mathbb{E}[\mathbf{x}_-]), \quad \mathbb{E}[\mathbf{x}_{t+1}] = \mathbb{E}[\mathbf{x}_+] - \frac{1}{2}\mathbb{E}[\mathbf{x}_-]$
6B.     $\mathbb{E}[\mathbf{x}_t\mathbf{x}_t^T] = A_t^{-1}(\mathbb{E}[\mathbf{x}_+\mathbf{x}_+^T] + \frac{1}{4}\mathbb{E}[\mathbf{x}_-\mathbf{x}_-^T] + \frac{1}{2}(\mathbb{E}(\mathbf{x}_+\mathbf{x}_-^T] + \mathbb{E}(\mathbf{x}_-\mathbf{x}_+^T]))A_t^{-T}$
6C.     $\mathbb{E}[\mathbf{x}_{t+1}\mathbf{x}_{t+1}^T] = \mathbb{E}[\mathbf{x}_+\mathbf{x}_+^T] + \frac{1}{4}\mathbb{E}[\mathbf{x}_-\mathbf{x}_-^T] - \frac{1}{2}(\mathbb{E}(\mathbf{x}_+\mathbf{x}_-^T] + \mathbb{E}(\mathbf{x}_-\mathbf{x}_+^T])$
7.   Compute intermediate locations and covariances:
7A.     $\boldsymbol{\mu}_{t,i} = \mathbb{E}[\mathbf{x}_t], \quad \Sigma_{t,i} = \mathbb{E}[\mathbf{x}_t\mathbf{x}_t^T] - \boldsymbol{\mu}_{t,i}\boldsymbol{\mu}_{t,i}^T$
7B.     $\boldsymbol{\mu}_{t+1,i} = \mathbb{E}[\mathbf{x}_{t+1}], \quad \Sigma_{t+1,i} = \mathbb{E}[\mathbf{x}_{t+1}\mathbf{x}_{t+1}^T] - \boldsymbol{\mu}_{t+1,i}\boldsymbol{\mu}_{t+1,i}^T$
8.   Compute candidate locations at $t$:
8A.     Sample $Y_{t,s} \sim \mathcal{N}(\boldsymbol{\mu}_{t,i}; \Sigma_{t,i}),\ s = 1 : N_{\text{sam}}, \quad Y'_{t,r,s} \sim \mathcal{N}(A_tY_{t,s} + \mathbf{b}_t; \Sigma_{\text{dyn}}),\ r = 1 : N_{\text{sam}}, \forall s$
8B.     $W_{t,r,s} = \psi(Y'_{t,r,s}, \theta), \quad Z_{t,s} = \sum_r W_{t,r,s}/N_{\text{sam}}$
8C.     $W'_{t,s} = Z_{t,s}^{(\gamma-1)/\wp}, \quad W'_{t,s} = W'_{t,s}/(\sum_s W'_{t,s})$
8D.     $\boldsymbol{\mu}_{t,\text{cand}} = \sum_s W'_{t,s}Y_{t,s}, \quad \Sigma_{t,\text{cand}} = (\sum_s W'_{t,s}Y_{t,s}Y_{t,s}^T) - \boldsymbol{\mu}_{t,\text{cand}}\boldsymbol{\mu}_{t,\text{cand}}^T$
9.   Compute candidate locations at $t+1$:
9A.     Sample $Y_{t+1,s} \sim \mathcal{N}(\boldsymbol{\mu}_{t+1,i}; \Sigma_{t+1,i}),\ s = 1 : N_{\text{sam}}$
9B.     $W_{t+1,s} = \psi(Y_{t+1,s}, \theta), \quad W_{t+1,s} = W_{t+1,s}/(\sum_s W_{t+1,s})$
9C.     $\boldsymbol{\mu}_{t+1,\text{cand}} = \sum_s W_{t+1,s}Y_{t+1,s}, \quad \Sigma_{t+1,\text{cand}} = (\sum_s W_{t+1,s}Y_{t+1,s}Y_{t+1,s}^T) - \boldsymbol{\mu}_{t+1,\text{cand}}\boldsymbol{\mu}_{t+1,\text{cand}}^T$
10.  Compute new locations and covariances:
10A.    $\Sigma_{t,\text{new}}^{-1} = \wp(1-\gamma)\Sigma_{t,\text{cand}}^{-1} + (1 - \wp(1-\gamma))\Sigma_t^{-1}$
10B.    $\Sigma_{t+1,\text{new}}^{-1} = \wp(1-\gamma)\Sigma_{t+1,\text{cand}}^{-1} + (1 - \wp(1-\gamma))\Sigma_{t+1}^{-1}$
10C.    $\boldsymbol{\mu}_{t,\text{new}} = \Sigma_{t,\text{new}}(\wp(1-\gamma)\Sigma_{t,\text{cand}}^{-1}\boldsymbol{\mu}_{t,\text{cand}} + (1 - \wp(1-\gamma))\Sigma_t^{-1}\boldsymbol{\mu}_t)$
10D.    $\boldsymbol{\mu}_{t+1,\text{new}} = \Sigma_{t+1,\text{new}}(\wp(1-\gamma)\Sigma_{t+1,\text{cand}}^{-1}\boldsymbol{\mu}_{t+1,\text{cand}} + (1 - \wp(1-\gamma))\Sigma_{t+1}^{-1}\boldsymbol{\mu}_j)$
11.  Compute new messages:
11A.    $V_{t,\text{new}}^{-1} = V_t^{-1} + \Sigma_{t,\text{new}}^{-1} - \Sigma_t^{-1}$
11B.    $V_{t+1,\text{new}}^{-1} = V_{t+1}^{-1} + \Sigma_{t+1,\text{new}}^{-1} - \Sigma_{t+1}^{-1}$
11C.    $\nu_{t,\text{new}} = V_{t,\text{new}}(V_t^{-1}\boldsymbol{\nu}_t + \Sigma_{t,\text{new}}^{-1}\boldsymbol{\mu}_{t,\text{new}} - \Sigma_t^{-1}\boldsymbol{\mu}_t)$
11D.    $\nu_{t+1,\text{new}} = V_{t+1,\text{new}}(V_{t+1}^{-1}\boldsymbol{\nu}_{t+1} + \Sigma_{t+1,\text{new}}^{-1}\boldsymbol{\mu}_{t+1,\text{new}} - \Sigma_{t+1}^{-1}\boldsymbol{\mu}_{t+1})$
```

[5] T. Minka. Divergence measures and message passing. Technical report, Microsoft Research Technical Report (MSR-TR-2005-173), 2005.

[6] R.M. Neal and G.E. Hinton. A view of the EM algorithm that justifies incremental, sparse and other variants. 1999.

[7] Marc Toussaint, Stefan Harmeling, and Amos Storkey. Probabilistic inference for solving (po)mdps. Technical report, University of Edinburgh, School of Informatics, 2006. Research Report EDI-INF-RR-0934.

[8] Wim Wiegerinck and Tom Heskes. Fractional belief propagation. In *Neural Information Processing Systems 15*, pages 438–445. 2003.