Estimation and control using sampling-based Bayesian reinforcement learning

Patrick Slade¹, Zachary N. Sunberg², Mykel J. Kochenderfer²

¹Mechanical Engineering Department, Stanford University, Stanford, USA
²Aeronautics and Astronautics Department, Stanford University, Stanford, USA
E-mail: patslade@stanford.edu

Abstract: Real-world autonomous systems operate under uncertainty about both their pose and dynamics. Autonomous control systems must simultaneously perform estimation and control tasks to maintain robustness to changing dynamics or modelling errors. However, information gathering actions often conflict with optimal actions for reaching control objectives, requiring a trade-off between exploration and exploitation. The specific problem setting considered here is for discrete-time non-linear systems, with process noise, input-constraints, and parameter uncertainty. This study frames this problem as a Bayes-adaptive Markov decision process and solves it online using Monte Carlo tree search with an unscented Kalman filter to account for process noise and parameter uncertainty. This method is compared with certainty equivalent model predictive control and a tree search method that approximates the QMDP solution, providing insight into when information gathering is useful. Discrete time simulations characterise performance over a range of process noise and bounds on unknown parameters. An offline optimisation method is used to select the Monte Carlo tree search parameters without hand-tuning. In lieu of recursive feasibility guarantees, a probabilistic bounding heuristic is offered that increases the probability of keeping the state within a desired region.

1 Introduction
Planning for localisation and manipulation tasks requires an accurate dynamics model of the system [1–3]. Often the dynamics are only partially known. For example, order-fulfilment robots manipulate containers with varying loads in warehouses [4], autonomous vehicles encounter changing environments [5–7], and nursing robotic systems interact with people [8]. Payload shifts, environment conditions, and human decisions act as time-varying parameters that dramatically alter system dynamics. Motion determined by physical laws also often have unknown parameters, such as friction or inertial properties. In these cases, the robot must estimate the system dynamics from measurements to achieve its goals. The task of estimation can conflict with the original goal. The robot must trade-off exploration to gain better understanding of the dynamics and exploitation to obtain rewards from the current knowledge.

Many approaches attempt to handle this trade-off between exploration and exploitation known as the ‘dual control’ problem [9]. When the belief, action, and state spaces are continuous, the exact solution is generally intractable. Many approximate solutions are used, such as adaptive control, stochastic optimal control, and sliding mode control [10–12]. Adaptive controllers often sequentially perform an estimation task and then the control task. The agent may be able to improve performance by using the estimation actions to also begin accomplishing the control task. A popular solution is certainty equivalent control; a robot plans by assuming its dynamics model is exact [13]. The approach of model predictive control (MPC) is to update the dynamics model after every observation and compute a new plan to a fixed horizon that is optimal for the updated most likely model [13, 14]. Variants of MPC extend to non-linear systems [15, 16] and account for uncertainty by propagating worst-case outcomes [17–19] or using probabilistic constraints [20]. Equations from the unscented transform have been incorporated as MPC constraints in an attempt to improve state estimation [21–23]. However, these methods cannot guarantee stability for systems with time-varying parameters, those following a non-Gaussian noise distribution, or significant outliers which would cause unbounded changes to the system which could not be corrected by an input-constrained control law [21, 24, 25].

Reinforcement learning is another popular approach, where the underlying planning problem is a Markov decision process (MDP) with some unknown probability distributions governing transitions [26]. Agents interact with the environment to accrue rewards and may learn the transition probabilities if it helps with the task. If the prior distribution of these transition probabilities is known, the policy that will collect the most reward in expectation is found by solving a Bayes-adaptive MDP [27]. Bayes-adaptive MDPs are typically computationally intractable [27]. However, many approximate solution methods are available, particularly when the problem is recast as a partially observable MDP (POMDP). If the problem has discrete state and action spaces, both offline [28] and online [29] POMDP methods have been adapted to RL, and others [30] can be easily adapted. Several methods for continuous-state problems have also been proposed. For systems where the uncertainty is approximately Gaussian, a value-iteration based method that uses an augmented reward function explicitly penalising uncertainty has been demonstrated [31]. Many generic optimisation methods such as gradient descent and nature inspired meta-heuristic algorithms [32, 33] have been applied to solve MDPs, but these require a policy parameterisation that is often domain specific and difficult to specify before finding a good policy. Using deep neural networks as generic parameterisation structures and optimising with stochastic gradient descent has performed well on a wide range of problems [34], but this approach requires a very large amount of data and is not useful for online planning in new environments.

In a preliminary version of this work, Monte Carlo tree search (MCTS) was used to solve an approximation of the problem with the tree search, implicitly balancing the exploration and exploitation that outperformed MPC in some non-linear system simulations [35]. MCTS has also been extended to handle POMDPs with non-Gaussian noise [36].

The contribution of this research is applying MCTS to the dual control problem, testing it against a baseline MPC approach, and proposing and demonstrating several approximations and techniques to improve performance. Specifically, we address problems expressed in discrete time with input constraints; time-
varying parameters; continuous state, action, and observation spaces; and process noise with a Gaussian distribution truncated to prevent non-physical parameter values. The continuous state and action spaces of the problem are handled using the double progressive widening (DPW) variant of MCTS [37].

Our previous work indicated MCTS was a viable solution for this dual control problem [35]. This paper performs systematic simulation experiments to determine the performance of MCTS, a tree search variant of QMDP (QMDP-TS), and benchmark MPC algorithms at varying levels of parameter uncertainty. A method for using an offline metaheuristic method, cross-entropy [38], is applied to optimise solver parameters without hand tuning. A stability analysis based on the convergence of physical parameter estimates is performed to understand how information gathering impacts system stability in an empirical setting. Finally, a sampling-based method for regulating the state to a bounded region with high probability is described and tested.

The organisation of this paper is as follows. Section 2 introduces the problems and solution methods, Sections 3 and 4 describe the problem and approach in detail, and Section 5 compares simulations between MCTS, QMDP-TS, and MPC.

2 Background

This section reviews sequential decision-making models, approximate solution methods for these models, the cross-entropy algorithm for tuning parameters in the solvers, and gives a brief introduction to the confidence regions used for probabilistic state heuristics.

2.1 MDPs, POMDPs, Bayes-adaptive MDPs

A MDP is a framework for sequential decision making where an agent moves stochastically between states over time. Rewards are accrued based on what states are reached. The agent influences the rewards and state trajectory by selecting actions each time step. An MDP is defined by the tuple \((\mathcal{X}, \mathcal{U}, T, R, \gamma)\), where:

- \(\mathcal{X}\) is the set of states,
- \(\mathcal{U}\) is the set of actions the agent may take,
- \(T(\mathcal{x}', \mathcal{x}, \mathcal{u})\) is the probability of transitioning to state \(\mathcal{x}'\) by taking action \(\mathcal{u}\) at state \(\mathcal{x}\),
- \(R(\mathcal{x}, \mathcal{u})\) is the reward (or cost) of taking action \(\mathcal{u}\) at state \(\mathcal{x}\),
- \(\gamma \in [0, 1]\) is the discount factor for future rewards.

The solution to an MDP is a policy, \(\pi(\mathcal{x}): \mathcal{X} \rightarrow \mathcal{U}\), that maps every state to an optimal action in order to accruze the largest expected rewards over a planning horizon. For discounted or finite horizon MDPs, an optimal policy satisfies the Bellman equation [13]. For small, discrete action and states spaces, the unique fixed point solution of the Bellman equation may be found efficiently with dynamic programming. Approximate dynamic programming may be used for problems with large or continuous state and action spaces [39].

A POMDP is an MDP where true state cannot be directly observed by the agent. Instead, the agent receives stochastic observations which have distributions conditioned on the state [27]. A belief state \(b\) is used to encode the probability of being in each state \(\mathcal{x}\). This belief is updated by the agent at each step depending on the previous action and observation. Since the agent may have any of many possible beliefs about its location in the state space, the belief space, \(\mathbb{B}\), typically has infinite cardinality, making POMDPs computationally intractable to solve [40].

A Bayes-adaptive MDP has only partially known transition probabilities. Initially, the agent is given only a prior distribution of these transition probabilities. As the agent interacts with the environment, it extracts information about the transition probabilities from the actions it takes and states it visits. A Bayes-adaptive MDP is the same as a POMDP when adding the unknown parameters that govern the transition probabilities to the state.

A POMDP is considered an MDP when the belief space is treated as the state space [41], also known as a belief MDP. The belief MDP defines the transition dynamics by a Bayesian belief update after an action is taken and a resulting observation is received. In a POMDP, the reward for a belief is the expectation of the state reward given that the state is distributed according to that belief [42]. Thus, uncertainty itself cannot be explicitly penalised. When the belief update is computationally tractable, approximate dynamic programming techniques designed for MDPs may be applied to POMDPs by using the corresponding belief MDP.

2.2 Monte Carlo tree search and QMDP tree search

MCTS is a sampling-based online approach for approximately solving MDPs which can be applied to POMDPs by using the associated belief MDP. MCTS uses a generative model \(G\) to generate a random state and reward \((\mathcal{x}', r)\) \(\sim G(\mathcal{x}, \mathcal{u}, w)\), where \(w\) is a random noise variable. It performs a forward search through the state space, using \(G\) to draw prospective trajectories and rewards. In MCTS, a tree is created with alternating layers of state nodes and action nodes [43]. A single iteration of MCTS consists of four stages: selection, expansion, rollout, and propagation [44]. By performing many iterations of this process, MCTS estimates the value at each node and chooses the action with the highest value.

QMDP is an offline approximation technique that accounts for one step of uncertainty [45]. This method performs well when the action choice cannot reduce the state uncertainty and thus information gathering is not important [27, 36]. The method is modified for online use by following same tree search structure as MCTS, referred to as QMDP-TS. The first step is solved exactly the same as in MCTS on the belief MDP, but all subsequent steps are treated as fully observable. Rather than computing beliefs for these steps, the mean is taken to be the true state and propagated exactly.

2.3 Unscented Kalman filter

In problems with linear Gaussian dynamics and observation functions, perfect Bayesian state estimation can be achieved with the Kalman filter. A Kalman filter is an iterative algorithm that can exactly update Gaussian beliefs over the state given the action taken, the observation received, and the transition and observation models. Variants of the unscented Kalman filter as well as sampling based estimators eliminate the underlying Gaussian model assumption [3]. For systems with non-linear dynamics, the extended Kalman filter approximates the state distribution with a Gaussian distribution. It propagates the prediction analytically using a linearisation of the system dynamics [46]. The true posterior mean and covariance of the transformed Gaussian distribution can accumulate large errors and possibly diverge depending on how well this linearisation matches the change in the system. A unscented Kalman filter (UKF) uses deterministic sampling to approximate the Gaussian distribution with a minimal set of carefully chosen points, achieving a second-order approximation (Taylor series expansion) when the points are propagated through any non-linearity [47].

Let \(\mathcal{x}_k, \mathcal{u}_k\), and \(\mathcal{o}_k\) be the state, action, and observation at time \(t = k\). For a system with non-linear-Gaussian dynamics, the transition and observation models can be expressed as

\[
x_{k + 1} = f(x_k, u_k) + w_k
\]

\[
\mathcal{y}_k = h(x_k, u_k) + v_k
\]

where \(f\) and \(h\) are non-linear functions, and \(w\) and \(v\) are normally distributed independent random variables for the process and measurement noise, respectively. The Gaussian belief has an estimate for the mean and covariance. The UKF updates the belief at each timestep by taking a sample of sigma-points, approximating new mean and covariance predictions. This gives a new estimate for the state and covariance based on the action taken. The UKF uses tunable parameters controlling the spread of sample points and knowledge of the distribution. Prior work has defined the optimal parameter values for estimating Gaussian distributions [46].
2.4 Cross-entropy

The cross-entropy method is an optimisation technique that iteratively updates a probability distribution that describes the input values that are likely to be optimal [38]. This probability distribution belongs to a heuristically chosen family of parameterised distributions, for example this work uses the multivariate normal family parameterised by the mean and covariance matrix. The update of the probability distribution is performed in two steps. First, a fixed number of samples are drawn from the distribution and evaluated with respect to the optimisation objective. Second, a smaller number of elite samples with the highest objective values are selected and the parameters of the distribution are fit to these samples, usually by maximising likelihood, yielding the probability distribution for the next generation. This process is continued for a specified number of iterations or until meeting a convergence threshold.

2.5 Confidence regions

Confidence regions are multivariate extensions of confidence intervals that can be used to bound the value that a random variable with probability greater than or equal to the confidence level, 1 − α. The specified statistical significance factor α is selected between 0 and 1, typically <0.1. The confidence region will be computed for a joint normal distribution as a function of the eigenvalues of the distribution’s covariance, \( \psi \), and the degrees of freedom \( \rho \).

Computing the eigenvalues satisfying (3) requires rescaling the eigenvalues of the distribution’s covariance, \([d_1, \ldots, d_p]\). The eigenvalue of the confidence region computed along the \(i\)th axis is

\[ \epsilon_i = \lambda_i \sqrt{\chi_i^2(\alpha)}. \]

(4)

The eigenvalues of the distribution’s covariance are replaced by the diagonal matrix \([\epsilon_1, \ldots, \epsilon_p]\) to form the confidence region ellipsoid.

3 Problem formulation

Consider a robot trying to control a system with linear-Gaussian dynamics. The transition at step \(k\) is described by

\[ x_{k+1} = f(x_k, \theta_k, u_k) + w_k \]

(5)

where \(x_k\) and \(u_k\) are the state and action, \(\theta_k\) is a vector of the unknown and time-varying parameters of the dynamics, \(w_k \sim \mathcal{N}(0, \Sigma^w)\) is the process noise, and \(f\) is a time-varying function that is linear with respect to \(x_k\) and \(u_k\)

\[ f(x_k, \theta_k, u_k) = A(\theta_k)x_k + B(\theta_k)u_k. \]

(6)

The observation model is described by the linear equation

\[ o_k = h(x_k, \theta_k, u_k) + v_k \]

(7)

with observation \(o_k\), the measurement noise \(v_k \sim \mathcal{N}(0, \Sigma^v)\), and a time-varying linear function \(h\)

\[ h(x_k, \theta_k, u_k) = C(\theta_k)x_k + D(\theta_k)u_k. \]

(8)

While \(f\) and \(h\) are physical equations known a priori, the parameters \(\theta_k\) are not known beforehand. They can be appended to the state vector to form a state-parameter vector or hyperstate [20]

\[ \xi_k = \begin{bmatrix} x_k \\ \theta_k \end{bmatrix}. \]

(9)

Thus, the system dynamics for \(\xi\) may be described by

\[ \xi_{k+1} = \begin{bmatrix} A(\theta_k) & 0 \\ 0 & I \end{bmatrix} \xi_k + \begin{bmatrix} B(\theta_k) \\ 0 \end{bmatrix} u_k + w_k \]

(10)

where \(w \sim \mathcal{N}(0, \text{diag}(\Sigma^w, \Sigma^w'))\) and \(\Sigma^w\) is a parameter drift matrix. We assume the parameter drift is equivalent to the process noise, simplifying the non-linear function to include additive process noise to the hyperstate

\[ \xi_{k+1} = f(\xi_k, u_k) + w_k. \]

(11)

The observation model is described by

\[ o_k = [C(\theta_k) & 0] \xi_k + D(\theta_k) u_k + v_k. \]

(12)

Using a UKF to describe the belief about the current hyperstate in the state-parameter space forms a belief MDP over all possible UKF states. This belief MDP is described by the tuple \((\mathcal{X}, \mathcal{U}, T, R)\), where:

- \(\mathcal{X}\) is the space of all possible beliefs. Since the belief maintained by the UKF is Gaussian, it can be described by the mean and covariance, \(b = \mathcal{N}(\xi, \Sigma^b)\).
- \(\mathcal{U}\) is all possible actions that the agent may take.
- \(T(b'[\theta, u])\) is a distribution over possible UKF states after a belief update. This distribution depends on the observation model. It is difficult to represent explicitly, so it is implicitly defined by the generative model, \(G\).
- \(R(x, u)\) is a reward function for a given state and action. It is constructed as desired for a given control task. In our work, we approximated \(R(x, u) = R(\xi, u)\), a linear reward for the estimated mean state and action.

The generative model for the UKF approximated belief MDP is

\[ b_{k+1} = G(b_k, u_k). \]

(13)

with \(G\) defined by the UKF update of the estimated mean and covariance with an observation sampled according to (12). The observation is given by the measurement update equations. Solving this belief MDP gives a policy that approximately maximises the sum of expected rewards over some planning horizon.

4 Approach

This section discusses using MCTS, QMDP-TS, and MPC with a UKF to control a system with unknown parameters.

4.1 Monte Carlo tree search and QMDP-TS

Our approach uses the upper confidence tree [43] with DPW [37] extensions of MCTS and QMDP-TS. The tree is built by repeatedly exploring the action node that maximises upper confidence estimate

\[ \text{UCB}(b, u) = \hat{Q}(b, u) + c \sqrt{\frac{\log N(b)}{N(b, u)}}. \]

(14)

where \(\hat{Q}(b, u)\) is an estimate of the state-action value function from rollout simulations and tree search, \(N(b, u)\) counts the times action \(u\) is taken from the hyperstate belief \(b\), and \(c\) is an exploration constant that balances exploration and exploitation as the tree expands.

DPW defines tree growth for large or continuous state and action spaces. To avoid a shallow search, the number of children of each state-action node \((b, u)\) is limited to
fixed horizon from the current state are found, and the first action from this sequence is taken [13].

Since there are no time-varying stochastic non-linear MPC algorithms that guarantee feasibility for parameters with Gaussian noise, linear MPC was selected for comparison [21, 24]. Linear MPC acts as a baseline to benchmark the performance between the optimal control and reinforcement learning algorithms. The important distinction between the described POMDP methods and MPC is that the POMDP methods reason about learning the parameters in the system. Certainty equivalent MPC assumes values for these parameters, typically the mean of the belief from an estimator such as a UKF.

### 4.3 Simultaneous estimation and control

This subsection describes the high level loop that controls the system by receiving observations and specifying actions. Any system described by a model in the form of (11) with an approximately Gaussian belief over the hyperstate can use MCTS, QMDP-TS, or MPC as the definition of a policy to choose a suitable control action. At every time step, this policy selects an action based on the belief. The state then evolves over time according to the dynamics and the action specified by the policy, and a new observation is generated. This observation and action are used to update the belief state with the UKF, improving the parameter estimate and providing the belief for the next action. The entire process is shown in Algorithm 1 (Fig. 2).

### 4.4 Probabilistic bounding heuristic

MPC is popular in part because of its ability to incorporate feasibility constraints. In particular, for many problems it can guarantee persistent feasibility, that is, that the state will stay in a specified region in all future steps [13]. For a system with time-varying and normally distributed parameters, the possible disturbances to the system are unbounded. Thus, no controller can guarantee feasibility. To approximate feasibility, a heuristic was developed to select appropriate actions at each state node to keep the norm of the next state within a desired bound. The heuristic consists of three steps: computing confidence regions, truncating samples, and checking the norm of the next state by propagating uniformly sampled actions. If this heuristic is followed, the norm of the next state is guaranteed to lie within the desired confidence region $\beta_u$, with probability equal to the confidence level. Similar heuristics that use confidence intervals and regions when estimating partially identified parameters have been studied in the past [50, 51].

The norm of the next state can be written as

$$\| x_{k+1} \| = \| f(x_k, \theta_k, u_k) + w_k \| .$$

An upper bound for this next state norm is computed by finding the probabilistic worst case given the process noise distribution and hyperstate distribution estimated by the UKF. Since the process noise is additive, it can be separated to create a conservative norm approximation

$$\| x_{k+1} \| \leq \beta(b_k, u_k),$$

defined as

$$\beta(b, u) = \beta_b(b, u) + \beta_w,$$

$$\beta_b(b, u) = \max_{z \in \Gamma(\mu, \Sigma, \alpha)} \| f(z, u) \|,$$

$$\beta_w = \max_{w \in \Gamma(0, \Sigma, \alpha)} \| w \|,$$

where $\Gamma(\mu, \Sigma, \alpha)$ is the confidence region ellipsoid defined in (3). Confidence regions are computed from the given distributions to find the components of the upper bound corresponding to the belief over the hyperstate $\beta_b$ and process noise $\beta_w$. This assumes

---

**Fig. 1** Illustration of a simple DPW tree shows state nodes as circles and action nodes as squares. In this example, the state is a multivariate normal where the darker colour and larger size of the orange gradient visually represents a larger covariance.

**Fig. 2** Algorithm 1: simultaneous estimation and control

$k \mathcal{N}(b, u)^\delta$,  

where $k$ and $\delta$ are parameter constants tuned to control the widening of the tree. With an increase in $\mathcal{N}(b, u)$ the number of children also grows, widening the tree. The number of actions explored at each state is controlled in the same way with an additional set of parameters.

Controlling the growth of both the state and action nodes allows the tree to balance the exploration of promising actions and exploitation of the current knowledge of the system. Actions that minimise parameter uncertainty, such as the initial blue action node in Fig. 1, may result in higher rewards for subsequent children nodes. The tree search will steer exploration in these most promising nodes to find the best policy. Thus, exploratory actions are selected when they improve the reward more than exploitive actions such as the initial red action node.

---

**Algorithm 1:** simultaneous estimation and control

1. for $t \in [0, T)$
2. $u_t \leftarrow \text{POLICY}(b_t)$
3. $\xi_{t+1} \leftarrow \text{DYNAMICS}(\xi_t, u_t)$
4. $o_t \leftarrow \text{RECEIVE MEASUREMENTS}(\xi_{t+1})$
5. $b_{t+1} \leftarrow \text{UKF}(b_t, o_t)$
5 Simulation and results

A model of a robot performing planar manipulation was used to test the simultaneous estimation and control capability of MCTS, QMDP-TS, and MPC.

5.1 Planar manipulation model

We consider an agent R pushing a box B in the plane, where the agent may apply an arbitrary force $F$ in the $x$ and $y$ directions, in addition to a torque $T$. This problem, along with relevant parameters and variables used to describe the system state, is illustrated in Fig. 4. The state-space form of the system uses the state vector $x_k = [p_{k, x}, p_{k, y}, v_{k, x}, v_{k, y}, \theta_{x}, \theta_{y}, \omega_{x}, \omega_{y}]^T$, corresponding to the linear and angular positions and velocities of $B$ in the global frame $N$. The linear and angular accelerations are given as $\dot{a}_k = [a_{k, x}, a_{k, y}, a_{k, \theta}]^T$. The system has time-varying, unknown parameters $b_k = [m_k, J_{x, k}, J_{y, k}, T_{x, k}, T_{y, k}]^T$ which represent the mass, linear friction, inertia, and distance from the centre of mass of the box to the robot with respect to the $x$ and $y$ directions. The unknown parameters are limited to enforce a lower bound that prevents non-physical values such as negative mass. This bound truncates the Gaussian distribution over the unknown parameters.

We can describe the dynamics of the system about its centre of mass, $B_{cm}$. These are given as

\begin{align}
F_{B, x} &= F_x - \mu v_x = ma_x \\
F_{B, y} &= F_y - \mu v_y = ma_y \\
T_B &= T + r_x \times F_y = J a_y.
\end{align}

The system in discrete time with step size of $\Delta t$ is

\begin{align}
\dot{v}_{k+1} &= \alpha_{k, \Delta t} + v_k \\
\dot{p}_{k+1} &= v_{k, \Delta t} + p_k.
\end{align}

Rewriting (21)–(23) in state-space form gives

\begin{align}
x_{k+1} &= f(x_k, u_k) = A x_k + B (b_k, p_k) u_k + w_k,
\end{align}

where

\begin{align}
A &= \begin{bmatrix}
    1 & 0 & 0 & 0 & \Delta t & 0 & 0 & 0 \\
    0 & 1 & 0 & 0 & 0 & \Delta t & 0 & 0 \\
    0 & 0 & 1 & 0 & 0 & 0 & \Delta t & 0 \\
    0 & 0 & 0 & 1 & 0 & 0 & 0 & \Delta t \\
\end{bmatrix}
\end{align}

\begin{align}
B &= \begin{bmatrix}
    0 \\
    0 \\
    \Delta t \\
    \Delta t \\
\end{bmatrix}
\end{align}

\begin{align}
B^{-1} &= \frac{\Delta t}{J_a} \begin{bmatrix}
    \cos(p_{0, k}) r_{y, k} + \sin(p_{0, k}) r_{x, k} \\
    \cos(p_{0, k}) r_{x, k} - \sin(p_{0, k}) r_{y, k} \\
\end{bmatrix}
\end{align}

\begin{align}
B^{2/2} &= \frac{\Delta t}{J_a} \begin{bmatrix}
    \cos(p_{0, k}) r_{y, k} + \sin(p_{0, k}) r_{x, k} \\
    \cos(p_{0, k}) r_{x, k} - \sin(p_{0, k}) r_{y, k} \\
\end{bmatrix}
\end{align}

\begin{align}
u_k &= \begin{bmatrix}
    F_{x, k} \\
    F_{y, k} \\
    T_{x, k} \\
\end{bmatrix}
\end{align}

with $m$ as the mass of $B$ and $J$ is $B$‘s moment of inertia.

For a robot with noisy sensors which measure its position, velocity, and acceleration in $N$, the observation model is

\begin{align}
y_k &= h(x_k, u_k) + v_k
\end{align}

where $y_k = [p_{k, x}, p_{k, y}, v_{k, x}, v_{k, y}, \theta_{x}, \theta_{y}, \omega_{x}, \omega_{y}]^T$.

The measurement functions are given by

\begin{align}
p &= r_m + r_0 \\
v &= v_m + v_{w, k} \times r_0
\end{align}
with an orientation in the positive normal distribution with a mean of 1 and variance of 0.5. The uncertain conditions. Measurement noise was not included to condition. The initial parameter distribution in all simulations is a parameter lower bound of 0.05. The linear interpolation between points do not reflect the effects of process noise and the lower bound on unknown parameters.

The reward function is linear with a weighted norm penalty.

\[
R(n, a, y, k) = \begin{bmatrix} R_{\text{pos}} & 0 \\ 0 & R_{\text{vel}} \end{bmatrix} \begin{bmatrix} x_k \end{bmatrix} + R_{\text{eff}} u_k .
\]  

(37)

The values for \( R_{\text{pos}}, R_{\text{vel}} \) and \( R_{\text{eff}} \) are \(-2.5, -50, \) and \(-0.3, \) respectively. These reward values were selected as they allowed the position and velocity to converge approximately to zero within half the simulation length for MPC in the lowest process noise configuration and largest lower bound of unknown parameters to get a conservative setting.

The MCTS and QMDP-TS implementations use a discount factor of 0.99. In the tree search, the next actions to be explored from a state node are selected by an epsilon greedy strategy. A random action is taken with probability 0.8, otherwise an MPC policy is computed using a state randomly sampled from the belief. This biases the actions to policies that generally perform well when accurate parameter estimates are available. The total number of nodes in the tree search is limited to 3000 for all simulations other than the bounded heuristic simulations which uses 300 total nodes. The bounding heuristic limited the maximum number of actions, \( n_a \) to 50. The MCTS with DPW implementation is from the POMDPs.jl package [52]. The optimisation in the MPC controller is solved with the Convex.jl package [53].

5.3 Cross-entropy for tuning solver parameters

The cross-entropy method is used offline to optimise the hyperparameters for the MCTS and QMDP-TS solvers. These hyperparameters were then held constant for the simulation experiments below. The optimised hyperparameters include the number of actions sampled at each state node, the number of states sampled at each action node, the depth of the tree, and the exploration constant. The parameters for the DPW search were constrained by fixing \( \delta = (1/30) \) so that only \( k \) is free to be optimised to control the number of child states in (15).

The initial parameter distribution is chosen heuristically with a mean of \( \mu_c = [20, 20, 10, 20] \) and covariance of \( \Sigma_c = \text{diag}(64, 64, 16, 81) \) for the respective hyperparameters. The samples from this distribution are rounded to the closest integer value with a minimum value enforced at 1. For the cross-entropy optimisation, the population size is 50, 10 elite samples are used to fit the next distribution, the iteration limit is 25, and the optimisation is ended when the maximum eigenvalue drops below 3.

The parameters for all test cases are the cross-entropy results for the condition with process noise variance 0.01. The resulting mean values for the MCTS hyperparameters were \( \mu = [22, 5, 12, 27] \). The planning horizon of MPC was selected to match the optimised MCTS depth of 12.

5.4 Performance

First, single trajectories from the MCTS and MPC solution methods are shown to illustrate the qualitative difference between their behaviour before statistical results are presented. A single simulation trajectory from the MCTS simulations is visualised in Fig. 5a, and a trajectory from the MPC simulations is shown in Fig. 5b. The MCTS true and estimated state trajectory match well. The MPC state estimate diverges from the true state as the number of steps increases. MCTS keeps the state closer to the origin by gathering information to achieve a better estimate.

Simulation results for a range of process noise levels and a fixed parameter lower bound of 0.0625 are shown in Fig. 6. This lower bound prevents any of the unknown parameter values from becoming smaller than 0.0625 due to the added process noise. The error bars represent the standard error of the mean. For all methods, total rewards generally decrease as process noise increases. The MPC oracle is an upper bound computed by allowing the agent to fully observe the time-varying parameter values. All methods other than the oracle achieve similar rewards for conditions with little process noise where the uncertainty is small enough that the model estimate does not require exploration to be accurate. MCTS and QMDP-TS outperform MPC by a large margin for higher process noise levels.
The reward for MCTS remains approximately the same across all significant effect on performance. A smaller lower bound allows QMDP-TS rewards decreasing to a lesser degree in Fig. 7. The useful information that maintained performance when the unknown parameter lower bound of 0.0625.

These smaller parameter values may result in a larger change in the process noise in the UKF to try to better maintain the consistency of the filter in the presence of significant non-linearity. Cautious MPC is expected to require less computation with every additional node. The trade-off between computation time and performance is a decision required by the control designer as it is specific to the problem being solved. However, MCTS and QMDP can perform early stopping to return the best action without completing the full tree search. In general, linear MPC is expected to require less computation than compared methods. The implementations were not optimised for speed and are thus not directly compared.

### 5.5 Probabilistic bounding heuristic

Simulation results using MCTS with and without the probabilistic bounding heuristic in Table 1 were tested for a process noise variation of 0.01, unknown parameter lower bound of 0.1, and confidence level of 0.95. The percentages denote the ratio of steps outside the desired bounds to total steps. Since these desired bounds are chosen by the control designer, several different bound levels were compared. The bounding heuristic decreased the percent out of bounds by a factor between 3 and 5. As expected with the conservative confidence region estimates, the state remains within the specified region more than 95% of the time when the bounding heuristic is used for two of the three desired bounds. For very tight bounds, where there are few possible actions to keep the system within the desired bounds, the bounding heuristic approaches the same percentage out of the desired bounds as standard MCTS. The total reward values decreased with smaller desired bounds due to the extra constraint of only selecting actions that would meet the required next state norm.

### 6 Conclusions

This paper considers the problem of controlling a robot while estimating unknown parameters, a common challenge of physical systems in uncertain environments. An online, sampling-based approach, MCTS, provides an approximate solution to this continuous control problem. Simulations of a 2D manipulation task show that this method effectively balances exploration and exploitation, improving performance as the lower bound on the unknown parameters decreases. This outperforms certainty equivalent MPC and QMDP-TS, an MCTS variant using a one-step lookahead to account for limited uncertainty. An offline optimisation method for automatically selecting tuning parameters and a heuristic for selecting actions to probabilistically bound the next state are also demonstrated. The new MCTS algorithm addresses the challenge of estimation and control for many real-world systems that are non-linear, have input constraints, can be modelled with discrete time steps, and have time-varying unknown parameters or any combinations of these attributes.
The authors thank Mac Schwager for discussions on background variance of 0.01 and unknown parameter lower bound of 0.05 each simulation step averaged over 100 simulations for a process noise.

---

**Table 1 Probabilistic bounding performance**

<table>
<thead>
<tr>
<th>MCTS (percent outside bounds)</th>
<th>2.5%</th>
<th>6.3%</th>
<th>27.3%</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCTS heuristic (percent outside bounds)</td>
<td>0.5%</td>
<td>2.2%</td>
<td>11.3%</td>
</tr>
<tr>
<td>Rewards with heuristic</td>
<td>−147.0</td>
<td>−153.3</td>
<td>181.6</td>
</tr>
</tbody>
</table>

---

**References**

5. Chow, Tori Fujinami, and Ian de Vlaming for their work on the probabilistic bounding heuristic as well as Vincent Chow, Tori Fujinami, and Ian de Vlaming for their work on the QMDP-TS implementation.

---

**Fig. 8** Mean absolute error of the unknown parameters and the rewards at each simulation step averaged over 100 simulations for a process noise variance of 0.01 and unknown parameter lower bound of 0.05.

