Information Theoretic Model Predictive Control on Jump Diffusion Processes

Ziyi Wang\textsuperscript{1}, Grady Williams\textsuperscript{1} and Evangelos Theodorou\textsuperscript{1}

Abstract—In this paper we present an information theoretic approach to stochastic optimal control problems for systems with compound Poisson noise. We generalize previous work on information theoretic path integral control to discontinuous dynamics with compound Poisson noise and develop an iterative model predictive control (MPC) algorithm using importance sampling. The proposed algorithm is parallelizable and when implemented on a Graphical Processing Unit (GPU) can run in real time. We test the performance of the proposed algorithm in simulation for two control tasks using a cartpole system and a quadrotor. Our simulations demonstrate improved performance of the new scheme and indicate the importance of incorporating the statistical characteristics of stochastic disturbances in the computation of the stochastic optimal control policies.

I. INTRODUCTION

Despite the maturity of the field of stochastic optimal control theory, the majority of the theoretical work in this sub-field of control mainly considers stochastic systems with Gaussian stochastic disturbances. This observation is also valid if one considers the lack of scalable and real time algorithms for control of high dimensional stochastic systems with disturbances that are far from being Gaussian and zero mean. Motivated by this lack of computational methods for stochastic control in this paper we derive a novel algorithm for control of systems with generalized stochastic disturbances represented as jump diffusions processes.

Our analysis is based on a variational inference formulation of stochastic optimal control. In particular, the approach we take in this paper is based on the path integral control framework, which originated from Kappen and Theodorou’s work \cite{Kappen2005} \cite{Theodorou2009}. An iterative path integral control method, developed by Williams \cite{Williams2010}, has been implemented for autonomous racing. This method uses the information theoretic notions of free energy and relative entropy, and obtains optimal control policy distribution through minimization of the Kullback-Leibler divergence (KL-Divergence) between a control induced probability measure and the optimal control policy induced probability measure. This approach allows for a solution to the stochastic optimal control problem using an importance sampling scheme. The algorithm developed from this approach can utilize the parallel computing capabilities of the GPU, which means a large number of sampling trajectories can be propagated simultaneously and the algorithm can be implemented in real time.

In this paper we consider dynamics with Gaussian and the more general compound Poisson noise. Compound Poisson process, also known as the marked-jump process, is a doubly stochastic process where the stochasticity arises from both the jump time and amplitude \cite{Sato1999}. For simplicity, we use the term jump noise for compound Poisson noise. Processes with jumps have been widely used to describe the random evolution of, e.g., neuron dynamics \cite{Gerstner2002}, of soil moisture dynamics \cite{Olbrich2001}, or of financial figures such as stock prices, market indices, and interest rates \cite{Ruf2003}. For the application on dynamical systems, including jump terms in the dynamics can better capture the discontinuities in the process when e.g., an aircraft flies in gust or a vehicle drives on rough road. Therefore, it is important that methods that deal with jump terms in the dynamics are developed.

Motivated by the broad applicability of jump-diffusion processes we develop a method for controlling the discontinuous dynamics with jump noise and present an iterative MPC algorithm that can be parallelized using a GPU. We implement the algorithm in simulation on a cartpole and quadrotor system with Gaussian and jump noise added, and we compare the performance of the algorithm against the path integral control based information theoretic MPC algorithm \cite{Williams2010} that doesn’t account for the jump noise.

The rest of this paper is organized as follows: in section II we provide the problem formulation. In section III we introduce the information theoretic approach to the stochastic optimal control problem. Then we provide the MPC algorithm in section IV. The simulation results are included in section V. Finally, we conclude this paper and discuss future research directions.

II. PROBLEM FORMULATION

Consider a stochastic system with state \( x_t \in \mathbb{R}^n \) and control \( u_t \in \mathbb{R}^m \) at time \( t \). We assume the dynamics also has additive noise from Brownian motion \( dw \in \mathbb{R}^p \) and marked-jump process \( dP \in \mathbb{R}^q \) with constant jump rate. We define \( U \in \mathbb{R}^{m \times T} \) as the control sequence and \( X \in \mathbb{R}^{n \times T} \) as the state trajectory over the time horizon \( T \). We can formulate our stochastic optimal control problem as:

\[
U^* = \arg \min_{U \in \mathcal{U}} \mathbb{E}_Q \left[ \phi(x_T, T) + \int_{t_0}^T L(x_t, u_t, t) \, dt \right] 
\]  

(1)

where \( \mathcal{U} \) is the set of admissible control sequences, and the expectation is taken with respect to the probability measure \( Q \) induced by the controlled dynamics:

\[
dx_t = F(x_t, u_t, t) \, dt + B(x_t, t) \, dw^{(1)} + H(x_t, Q, t) \, dP^{(1)} 
\]  

(2)
with \( \mathbb{E}[d\mathbf{P}^{(1)}] = \nu^{(1)} dt \) and \( \nu^{(1)} \) is the jump rate. We assume zero mean normal distribution for the mark distribution, \( \phi_Q(q; t) \sim N(0, \sigma_q^2) \). For the cost function we consider a state-dependent cost and a quadratic control cost:

\[
\mathcal{L}(x_t, u_t, t) = g(x_t, t) + \frac{1}{2} u_t^T R(x_t, t) u_t
\]  

(3)

We consider dynamics affine in control:

\[
\mathcal{F}(x_t, u_t, t) = f(x_t, t) + G(x_t, t) u_t
\]  

(4)

The proof for existence and uniqueness of solution to the problem we are considering can be found in [8].

### III. INFORMATION THEORETIC APPROACH

In this section we present the derivation of our sampling based stochastic trajectory optimization method for jump diffusion processes using an information theoretic approach.

Before we start the derivation we need to introduce two quantities from information theory that are the foundation to our derivation. First we define the \textit{Free Energy} of a system as:

\[
\mathcal{F}(S(X)) = \log \left( \mathbb{E} \left[ \exp \left( - \frac{1}{\lambda} S(X) \right) \right] \right)
\]  

(5)

where \( \lambda \in \mathbb{R}^+ \) is called the inverse temperature, and \( S(X) \) is the state-dependent cost of a trajectory, \( S(X) = \phi(x_T, T) + \int_0^T g(x_t, t) dt \). The expectation is taken with respect to \( \mathbb{P} \), which is the probability measure induced by the unconstrained dynamics:

\[
dx_t = f(x_t, t) dt + B(x_t, t) dw^{(0)} + H(x_t, \mathbb{Q}, t) d\mathbf{P}^{(0)}
\]  

(6)

with \( \mathbb{E}[d\mathbf{P}^{(0)}] = \nu^{(0)} dt \) and the same mark distribution as the controlled dynamics. Next let \( \mathbb{M}, \mathbb{N} \) be two probability distributions that are absolutely continuous with each other. Then the \textit{KL-Divergence} between them is:

\[
\mathcal{D}_{KL}(\mathbb{M} \parallel \mathbb{N}) = \mathbb{E}_{\mathbb{M}} \left[ \log \left( \frac{d\mathbb{M}}{d\mathbb{N}} \right) \right]
\]  

(7)

The KL-Divergence provides a measure of how one probability distribution diverges from a second and can be roughly thought of as the distance between two probability distributions, although it is not symmetric. The KL-Divergence is useful for defining optimization objectives.

Now suppose probability distributions \( \mathbb{Q} \) and \( \mathbb{P} \) as defined previously are absolutely continuous with each other, we can make the following observation:

\[
\mathcal{F}(S(X)) \geq \mathbb{E}_\mathbb{Q} \left[ \log \left( \exp \left( - \frac{1}{\lambda} S(X) \right) \right) \right]
\]  

(9)

The right hand side can be simplified as:

\[
\text{RHS} = - \frac{1}{\lambda} \mathbb{E}_\mathbb{Q} \left[ S(X) - \lambda \log \left( \frac{d\mathbb{P}}{d\mathbb{Q}} \right) \right]
\]  

(10)

Substituting the terms back to (9) and multiplying both sides by \(-\lambda\):

\[
-\lambda \mathcal{F}(S(X)) \leq \mathbb{E}_\mathbb{Q}[S(X)] + \lambda \mathcal{D}_{KL}(\mathbb{Q} \parallel \mathbb{P})
\]  

(11)

To find the KL-Divergence between \( \mathbb{Q} \) and \( \mathbb{P} \), we need \( \frac{d\mathbb{Q}}{d\mathbb{P}} \), which can be found using Girsanov’s theorem [4]:

\[
\frac{d\mathbb{Q}}{d\mathbb{P}} = \exp \left( \frac{1}{2} \int_{t_0}^T u_t^T G(x_t, t) \Sigma(x_t, t)^{-1} G(x_t, t) u_t dt \right)
\]  

\[+ \int_{t_0}^T u_t^T G(x_t, t)^{-1} B(x_t, t) dw^{(1)}
\]  

\[\int_{t_0}^T \left( (\gamma^J - 1) \nu^{(0)} \right) dt \cdot \prod_{k=1}^{\gamma^J} \frac{P^{(0)(t)}}{P^{(t)}}, \gamma^J(T_k) \gamma^M(Q_k, T_k)
\]  

(12)

where \( \Sigma(x_t, t) = B(x_t, t)B(x_t, t)^T \), \( \gamma^J(t) \) is the ratio of jump rates in the two dynamics, \( \int_{t_0}^T \nu^{(1)} dt = \int_{t_0}^T \gamma^J \nu^{(0)} dt \), and \( \gamma^M(q; t) \) is the scaling between the mark distributions, \( \int_Q \phi^{(1)}_Q(q; t) dq = \int_Q \gamma^M(q; t) \phi^{(0)}_Q(q; t) dq = 1 \).

Here we consider the case where the change of measure only includes changes in drift, and the jump rates and mark distributions are the same. Therefore, both \( \gamma^J \) and \( \gamma^M \) have the value 1, and the last two terms can be dropped. Additionally, since \( dw^{(1)} \) is a Brownian motion with respect to \( \mathbb{Q} \), we get \( \mathbb{E}_\mathbb{Q} \left[ \int_{t_0}^T dw^{(1)} \right] = 0 \). The KL-Divergence then simplifies to:

\[
\mathcal{D}_{KL}(\mathbb{Q} \parallel \mathbb{P}) = \mathbb{E}_\mathbb{Q} \left[ \frac{1}{2} \int_{t_0}^T u_t^T G(x_t, t)^T \Sigma(x_t, t)^{-1} G(x_t, t) u_t dt \right]
\]  

(13)

Using this result, if we assume the control cost matrix has the form:

\[
R(x_t, t) = \lambda G(x_t, t)^T \Sigma(x_t, t)^{-1} G(x_t, t)
\]  

(14)

we get the following form on the right hand side of equation (11):

\[
\mathbb{E}_\mathbb{Q}[S(X)] + \lambda \mathcal{D}_{KL}(\mathbb{Q} \parallel \mathbb{P}) = \mathbb{E}_\mathbb{Q} \left[ S(X) + \frac{1}{2} \int_{t_0}^T u_t^T R(x_t, t) u_t dt \right]
\]  

(15)
Note that this is equivalent to the cost function in (1). With this we have shown that the negative free energy serves as the lower bound for our stochastic optimal control problem, and we can rewrite (11) as a minimization problem:

\[ -\lambda F(S(X)) = \inf_Q \left[ E_Q[S(X)] + \lambda D_{KL}(Q \parallel P) \right] \tag{16} \]

In this minimization problem we have a state cost and a control cost in the form of KL-Divergence, which penalizes deviation from the uncontrolled distribution.

We now define the optimal measure that achieves the lower bound as:

\[ \frac{dQ^*}{dP} = \exp\left( -\frac{1}{\lambda} S(X) \right) \frac{dP}{E_P[\exp(-\frac{1}{\lambda} S(X))]}. \tag{17} \]

This result can be easily verified by plugging it into (11) and is derived in [9]. With this we can solve the minimization problem defined by (16) by moving the probability distribution \( Q \) induced by some control as close to the optimal distribution as possible. The distance can be represented by the KL-Divergence between the two distributions and the problem becomes:

\[ U^* = \arg \min_{U \in \mathcal{U}} D_{KL}(Q^* \parallel Q) \tag{18} \]

**A. KL-Divergence Minimization**

Applying the definition of KL-Divergence we have:

\[ D_{KL}(Q^* \parallel Q) = E_{Q^*} \left[ \log \frac{dQ^*}{dQ} \right] = E_Q^* \left[ \log \frac{dP}{dQ} \right] \tag{19} \]

We already have \( \frac{dQ^*}{dP} \) from its definition. For \( \frac{dP}{dQ} \), we can use Girsanov’s theorem:

\[ \frac{dP}{dQ} = \exp \left( \frac{1}{2} \int_{t_0}^{T} u_t^T G(x_t, t)^T \Sigma(x_t, t)^{-1} G(x_t, t) u_t dt - \int_{t_0}^{T} u_t^T G(x_t, t)^T \Sigma(x_t, t)^{-1} B(x_t, t) d\epsilon(t) \right) \tag{20} \]

Setting the terms inside the exponential as \( D(X, U) \) and plugging the results back in (19) we have:

\[ D_{KL}(Q^* \parallel Q) = E_Q^* \left[ -\frac{1}{\lambda} S(X) - \log(\mathbb{E}_P[\exp(-\frac{1}{\lambda} S(X))]) + D(X, U) \right] \tag{21} \]

Since \( S(X) \) is not dependent on the control we can drop the first two terms from the minimization. Now we discretize the control as step function \( u_j = u_j \) if \( j \Delta t \leq t < (j + 1) \Delta t \) with \( j = \{0, 1, \ldots, N - 1\} \). Then we have:

\[ D(X, U) = \sum_{j=0}^{N-1} \left( \frac{1}{2} u_j^T \int_{t_j}^{t_{j+1}} G(x_t, t) dt u_j - u_j^T \int_{t_j}^{t_{j+1}} B(x_t, t) d\epsilon(t) \right) \tag{22} \]

where

\[ G(x_t, t) = G(x_t, t)^T \Sigma(x_t, t)^{-1} G(x_t, t) \tag{23} \]

\[ B(x_t, t) = G(x_t, t)^T \Sigma(x_t, t)^{-1} B(x_t, t) \tag{24} \]

\[ N = T/\Delta t \tag{25} \]

Note that each \( u_j \) does not depend on the trajectory taken, so we can taken them out of the expectation:

\[ E_Q^* \left[ D(X, U) \right] = \sum_{j=0}^{N-1} \left( \frac{1}{2} u_j^T E_Q^* \left[ \int_{t_j}^{t_{j+1}} G(x_t, t) dt \right] u_j - u_j^T E_Q^* \left[ \int_{t_j}^{t_{j+1}} B(x_t, t) d\epsilon(t) \right] \right) \tag{26} \]

We can approximate the two integrals for small enough \( \Delta t \) as:

\[ \int_{t_j}^{t_{j+1}} G(x_t, t) dt \approx G(x_{t_j}, t_j) \Delta t \tag{27} \]

\[ \int_{t_j}^{t_{j+1}} B(x_t, t) d\epsilon(t) \approx B(x_{t_j}, t_j) \epsilon_j(0) \sqrt{\Delta t} \tag{28} \]

where \( \epsilon_j(0) \) is a vector with standard normal variable in each entry, \( \epsilon_j(0) \sim \mathcal{N}(0, \Sigma_w) \). Then we can find \( u_j^* \) by taking the gradient with respect to \( u_j \), setting it to zero and solving for \( u_j \). The optimal control is found as:

\[ u_j^* = \frac{1}{\Delta t} E_Q^* \left[ G(x_{t_j}, t_j) \right]^{-1} E_Q^* \left[ B(x_{t_j}, t_j) \epsilon_j(0) \sqrt{\Delta t} \right] \tag{29} \]

**B. Importance Sampling**

We have obtained the optimal control in the form of expectation with respect to the optimal distribution. We can’t sample from the optimal distribution, but we can sample from the uncontrolled distribution \( P \) to approximate the controls. Therefore, we need to change the expectation through multiplying by \( \frac{dP}{dQ} \) and using the Radon-Nikodym derivative \( \frac{dQ}{dP} \):

\[ u_j^* = \frac{1}{\Delta t} E_P \left[ \exp(-\frac{1}{\lambda} S(X)) \right]^{-1} E_P \left[ \exp(-\frac{1}{\lambda} S(X)) B(x_{t_j}, t_j) \epsilon_j(0) \sqrt{\Delta t} \right] \tag{30} \]

The equation can be further simplified since \( G(x_{t_j}, t_j) \) and \( B(x_{t_j}, t_j) \) are deterministic at time \( t_j \):
\[ u_j^* = \frac{1}{\Delta t} \mathcal{G}(x_{t_j}, t_j)^{-1} B(x_{t_j}, t_j) E_{\mathcal{Q}} \left[ \frac{\exp(-\frac{1}{2} \tilde{S}(X)) e_j(0) \sqrt{\Delta t}}{E_{\mathcal{Q}}[\exp(-\frac{1}{2} \tilde{S}(X))] } \right] \]

(31)

Note that the expectations are taken with respect to the uncontrolled dynamics. This is not ideal since it means waiting for random Gaussian and jump noise to generate a meaningful trajectory. Therefore, we need to change the sampling distribution to the control induced distribution. In addition, we can also change the sampling variance to \( \Sigma^{(1)} = c \Sigma^{(0)} \) to increase the state space explored. To perform importance sampling we multiply by \( \frac{d\mathcal{P}}{d\mathcal{Q}} \) and change from the zero mean \( \epsilon_j^{(0)} \sqrt{\Delta t} \) to the non zero mean \( \mathcal{G}(x) u_j \Delta t + \epsilon_j^{(1)} \sqrt{\Delta t} \):

\[ u_j^* = u_j + \frac{1}{\Delta t} \mathcal{G}(x_{t_j}, t_j)^{-1} B(x_{t_j}, t_j) \cdot E_{\mathcal{Q}} \left[ \exp\left(-\frac{1}{2} \tilde{S}(X)\right) e_j(1) \sqrt{\Delta t} \right] \cdot \frac{d\mathcal{P}}{d\mathcal{Q}} \]

(32)

We can use Girsanov’s theorem again to get \( \frac{d\mathcal{P}}{d\mathcal{Q}} \):

\[
\frac{d\mathcal{P}}{d\mathcal{Q}} = \exp \left( -\frac{1}{2} \sum_{j=0}^{N-1} \left( u_j^T \mathcal{G}(x_{t_j})^T \Sigma^{-1} \mathcal{G}(x_{t_j}) u_j \Delta t \\
+ u_j^T \mathcal{G}(x_{t_j})^T \Sigma^{-1} B(x_{t_j}) \epsilon_j^{(1)} \sqrt{\Delta t} \\
+ (1-c^{-1}) \epsilon_j^{(1)T} B(x_{t_j})^T \Sigma^{-1} B(x_{t_j}) \epsilon_j^{(1)} \Delta t \right) \right) \]

(33)

The last terms comes from the change of sampling variance and the detailed derivation can be found in [10]. The addition of these terms can be added into the state cost:

\[ \tilde{S}(X) = \phi(x_{t_N}, t_N) + \sum_{j=0}^{N-1} \tilde{q}(x_{t_j}, u_j, t_j) \Delta t \]

(34)

\[
\tilde{q}(x_{t_j}, u_j, t_j) = q(x_{t_j}, t_j) + \frac{1}{2} u_j^T R u_j + \lambda u_j^T B \frac{\epsilon_j^{(1)}}{\sqrt{\Delta t}} + \frac{1}{2} \lambda (1-c^{-1}) \epsilon_j^{(1)T} B(x_{t_j})^{T} \Sigma^{-1} B(x_{t_j}) \epsilon_j^{(1)} / \Delta t \]

(35)

With the new state cost we can obtain the final expression of optimal control update rule:

\[ u_j^* = u_j + \mathcal{G}(x_{t_j}, t_j)^{-1} B(x_{t_j}, t_j) \left( E_{\mathcal{Q}}[\exp(-\frac{1}{2} \tilde{S}(X)) e_j^{(1)}] \right) \]

The term inside the square brackets is approximated as:

\[ \frac{\sum_{k=1}^{K} \exp(-\frac{1}{2} \tilde{S}(X_k)) e_k^{(1)}}{\sum_{k=1}^{K} \exp(-\frac{1}{2} \tilde{S}(X_k))} \]

(37)

using \( K \) sample trajectories.

Algorithm 1 MPPI Control on Jump Diffusion

Given:

- \( K \): Number of samples;
- \( N \): Number of timesteps;
- \( (u_0, u_1, \cdots, u_{N-1}) \): Initial control sequence;
- \( x_0 \): Initial states;
- \( \Delta t, f, G, B, H \): System/sampling dynamics;
- \( \phi, q, \lambda, R \): Cost function parameters;
- \( c, \Sigma_w, \Sigma_e, \nu \): Noise parameters
- \( u_{init} \): Value for new control initialization;

while task not completed do

for \( k = 0 \) to \( K-1 \) do

Update \( x_0 \):

Sample \( \epsilon_k = (\epsilon_0^k, \cdots, \epsilon_{N-1}^k) \in \mathcal{N}(0, c \Sigma_w) \);

for \( i = 0 \) to \( N-1 \) do

if \( p < \nu \Delta t \) then

\[ x_{i+1} = x_i + (f + Gu_i) \Delta t + Be_i \sqrt{\Delta t}; \]

else

\[ x_{i+1} = x_i + (f + Gu_i) \Delta t + Be_i \sqrt{\Delta t}; \]

end if

end for

end for

for \( i = 0 \) to \( N-2 \) do

\[ u_i = u_{i+1} + \frac{\sum_{k=1}^{K} \exp\left(-\frac{1}{2} \tilde{S}(X_k)\right) \epsilon_k^{(1)}}{\sum_{k=1}^{K} \exp\left(-\frac{1}{2} \tilde{S}(X_k)\right)}; \]

end for

Execute control policy \( u_0 \);

end for

for \( i = 0 \) to \( N-2 \) do

\[ u_i = u_{i+1}; \]

end for

\[ u_{N-1} = u_{init}; \]

end while

IV. Model Predictive Control Algorithm

With equation (36) and (37), we have an iterative update law for the optimal control policy at each timestep. This allows for the algorithm to be implemented in a MPC fashion. In the MPC setting, after the optimal control sequence is obtained, only the first control action is executed and re-optimization occurs from the new initial states. Since the path integral control framework gives the entire optimal control sequence, we can keep the un-executed control sequence to further start the optimization for the next iteration. In addition, the final control action in the sequence is initialized to a prescribed value for every iteration. This is very important for increasing the performance of the algorithm as we are reusing information from previous optimization iterations.

Another key aspect of the algorithm is that the most computationally involved parts, namely trajectory propagation and cost computation for each sampled trajectory, can be...
done in parallel on a GPU. Parallel computation allows us to sample thousands of trajectories at the same time rather than in sequence, with the computation time for each trajectory of less than 20 milliseconds in our simulation, which is smaller than our $\Delta t$ of 20 milliseconds for the choice of 50 Hz control frequency. This means that the algorithm can be implemented in real time. The description of the MPPI algorithm is given in Algorithm I. To implement the algorithm on GPU, simply assign one thread for the sampling of each trajectory.

In addition, we assume that both noises affect the states through the control channels in our simulation. In this case, $\mathbf{B}$ and $\mathbf{H}$ matrices are the same as $\mathbf{G}$, and the matrix transform $\mathbf{G}^{-1}\mathbf{B}$, which maps from state space to control space, go to identity. For the simulation of jump noise, we assume that it comes from a single source and $d\mathbf{P} \in \mathbb{R}$. We use the Zero-One Jump Law [7] for checking whether jump occurs at each timestep. The Zero-One Jump Law states that if the product of jump rate and time interval $\nu\Delta t$ is small enough, the probability of more than one jump occurring at each timestep can be neglected. Therefore, we use a jump timer $\nu$, which is sampled from a uniform distribution, and check against the probability of a single jump occurring to determine whether jump occurs at each timestep. When a jump occurs, a zero mean Gaussian vector determines the magnitude of jump noise in each control channel.

V. SIMULATION RESULTS

To test the capability of MPPI algorithm for jump diffusion processes, we apply it to a cart pole and a quadrotor model with artificially introduced jump noise. We also apply the MPPI algorithm that doesn’t include jump noise on the same systems and compare the results. To avoid confusion, we refer to the MPPI algorithm with jump noise in sampling presented in this paper as the new MPPI algorithm, and we refer to the MPPI algorithm without jump noise in sampling as the old MPPI algorithm.

A. Cart Pole

We applied the new and old MPPI algorithm on a standard cart pole system with artificially introduced Gaussian and jump noise. The task is to swing up and stabilize the cart pole. We used 1000 trajectories during sampling and ran each algorithm for 100 trials. We tested the robustness of both algorithms by varying the jump amplitude and rate while keeping Gaussian noise the same. In Table. I, we demonstrate the simulation results. We found that the new MPPI algorithm has a higher success rate in stabilizing the cart pole. Specifically, with only small jump noise, both algorithms managed to balance the cart pole. As we increased the jump amplitude, both algorithms started to fail, but the new algorithm has a higher success rate of stabilizing the pole than the old algorithm. For a fixed jump amplitude, increasing the jump rate results in lower success rates in both algorithms and vice versa.

In Fig. 1 we compare the response of the two algorithms where MPPI without jump fails. The pole angle plot and the Poisson noise plot show that the old MPPI algorithm failed after a noise spike and had to restabilize the pole. On the other hand, the new algorithm experienced noise spikes of similar magnitude and maintained balance. The cart position plot shows that the new algorithm managed to maintain balance efficiently around the origin.

B. Quadrotor

We also applied the new and old MPPI algorithm on a quadrotor system in simulation. The task is to fly from an initial position to a target position. Since it is a more complex system we increased the number of sampling trajectories to 3000 and ran each algorithm for 100 trials. Again we varied the jump amplitude and rate while keeping Gaussian noise the same. In Table. II we list the simulation results. Similar to the cartpole simulation, we found that the new MPPI algorithm has a higher success rate in completing the task. Specifically, with only small jump noise, both algorithms can carry out the task perfectly. As we increased the jump noise amplitude, the failure rate of the old MPPI algorithm increased while the new algorithm maintained perfect task completion rate. Additionally, for a jump amplitude large enough that the old MPPI algorithm has a non zero failure rate, increasing the jump rate further increases the failure rate of the old MPPI algorithm and vice versa.
TABLE II
SUCCESS RATES OF THE NEW AND OLD MPPI ALGORITHM APPLIED ON A QUADROTOR WITH JUMP NOISE FOR 100 TRIALS WITH 3000 SAMPLING TRAJECTORIES.

| Jump noise $\nu$, $\sigma_j^2$ | New MPPI | Old MPPI |
|-------------------------------|-----------|-----------|
| $\nu = 0.2$, $\sigma_j^2 = 5$ | 100%      | 100%      |
| $\nu = 0.2$, $\sigma_j^2 = 10$ | 100%      | 98%       |
| $\nu = 0.2$, $\sigma_j^2 = 20$ | 100%      | 97%       |
| $\nu = 0.2$, $\sigma_j^2 = 30$ | 100%      | 87%       |
| $\nu = 0.1$, $\sigma_j^2 = 20$ | 100%      | 98%       |
| $\nu = 0.5$, $\sigma_j^2 = 20$ | 100%      | 91%       |

In Fig. 2 to 4, we compare the response of the two algorithms for one test case ($\nu = 0.2$, $\sigma_j^2 = 20$). We plot the mean and 95% confidence region of the responses. The x and y position plots show that the mean of trajectories resulted from both algorithms follow a similar path to the target, but the variance of trajectories resulted from the old MPPI algorithm is much larger. From the z position plot, we observe that the variance of trajectories resulted from the old MPPI algorithm is significantly larger than the new MPPI algorithm since there are three crash runs. For the other plots, we find similar or smaller variance for both algorithms with more sampling trajectories. Similar levels of reduction in variance is also seen in the case with low jump magnitude (Plots not shown here). The results suggest that increasing the number of sampling trajectories correspond to algorithms with high jump noise amplitude. Doubling the sampling trajectories resulted in one less crash run for the old MPPI algorithm, while the new MPPI algorithm maintained perfect success rate. From the x and y position plots, we observe that the variance for both algorithms are smaller than the case with fewer sampling trajectories. There is one region in the z position plot where the variance for the old MPPI algorithm increases significantly due to the crash runs. For the other plots, we find similar or smaller variance for both algorithms with more sampling trajectories.

Fig. 2. Comparison of MPPI algorithm on a quadrotor with (green) and without (blue) jump noise ($\sigma_j^2=20$, $\nu=0.2$) with 3000 trajectories in sampling. On the top left is x position. On the top right is y position. On the bottom left is z position. On the bottom right is roll angle. The line represents the mean and shaded area represents the 95% confidence region (mean plus and minus two times standard deviation). The red line indicates the target position.

Fig. 3. Comparison of MPPI algorithm on a quadrotor with (green) and without (blue) jump noise ($\sigma_j^2=20$, $\nu=0.2$) with 3000 trajectories in sampling. On the top left is x velocity. On the top right is y velocity. On the bottom left is z velocity. On the bottom right is roll velocity. The line represents the mean and shaded area represents the 95% confidence region (mean plus and minus two times standard deviation). The red line indicates the target position.

Fig. 4. Comparison of MPPI algorithm on a quadrotor with (green) and without (blue) jump noise ($\sigma_j^2=20$, $\nu=0.2$) with 3000 trajectories in sampling. On the top left is pitch angle. On the top right is yaw angle. On the bottom left is pitch velocity. On the bottom right is yaw velocity. The line represents the mean and shaded area represents the 95% confidence region (mean plus and minus two times standard deviation). The red line indicates the target position.
a decrease in variance in generated trajectories. The decrease results from better approximation of the expectation with more samples.

In Fig. 8 we compare the total variance (sum of variance in all states over the entire time horizon for all trajectories) of both algorithms with two jump noise levels using 3000 and 6000 sampling trajectories. We find that with low jump noise amplitude, the old MPPI algorithm results in slightly lower variance than the new MPPI algorithm. The new MPPI algorithm tends to generate trajectories that oscillate around the target location since the dynamics is perturbed more during sampling. For the case of high jump noise amplitude, the difference in variance between the two algorithms is significantly reduced with increased sampling trajectories. This is due to the benefit of exploring more of the state space by including jump noise is reduced with increased sampling trajectories.

VI. CONCLUSIONS

We presented an information theoretic model predictive control algorithm that obtains the optimal control through sampling with Gaussian and compound Poisson noise. We applied the proposed algorithm on cart pole and quadrotor systems with artificially introduced compound Poisson noise and compared its performance to the previously developed algorithm that doesn’t include Poisson noise in sampling. We demonstrated superior performance of our new algorithm than the old algorithm under large Poisson noise level and comparable performance under low Poisson noise level.
results suggest that it is important to consider the statistical characteristics of stochastic disturbances in the computation of the optimal control policies.

In addition these results motivate our research in three main directions. The first direction is to develop adaptive stochastic control algorithms for systems that undergo stochastic disturbances such as the ones considered in this paper. We believe that these adaptive stochastic controllers will surpass the performance of existing algorithms used for terrestrial agility and high speed navigation in uncertain environments. This research direction will involve the development of novel inference algorithms for learning dynamics of stochastic systems with compound Poisson disturbance in real time.

The second research direction includes the improvement of the proposed algorithm by incorporating second order optimization methods such as momentum-based optimization techniques. We expect that these techniques will find better minima despite the increased variance in the cost function that arises from the Poisson noise in the dynamics.

Finally, the third direction includes theoretical analysis on the stability properties of the proposed algorithm. This analysis will consist of guarantees and performance bounds as a function of sampled trajectories and levels of instability of the stochastic system in consideration.

REFERENCES

[1] H. J. Kappen, Linear theory for control of nonlinear stochastic systems, Phys Rev Lett, vol. 95, p. 200201, 2005, journal Article United States.
[2] E. Theodorou, J. Buchli, and S. Schaal, A generalized path integral approach to reinforcement learning, Journal of Machine Learning Research, no. 11, pp. 31373181, 2010.
[3] G. Williams, P. Drews, B. Goldfain, J. M. Rehg, and E. A. Theodorou, Aggressive driving with model predictive path integral control, in 2016 IEEE International Conference on Robotics and Automation (ICRA), May 2016, pp. 14331440.
[4] F. B. Hanson, Applied Stochastic Processes and Control for JumpDiffusions. SIAM, 2007.
[5] Giraudo, M. T. and Sacerdote, L. Jump-diffusion processes as models for neuronal activity. Biosystems 40, 7582 (1997).
[6] Daly, E. and Porporato, A. Probabilistic dynamics of some jump-diffusion systems. Phys. Rev. E 73, 026108 (2006).
[7] Cont, R. and Tankov, P. Financial modelling with jump processes (Chapman and Hall/CRC, Boca Raton, FL, 2004).
[8] B. Øksendal and A. Sulem. Applied Stochastic Control of Jump Diffusions. Universitext. Springer-Verlag, Berlin, 2004.
[9] E. A. Theodorou and E. Todorov, Relative entropy and free energy dualities: Connections to path integral and kl control, in Decision and Control (CDC), 2012 IEEE 51st Annual Conference on. IEEE, 2012, pp. 14661473.
[10] G. Williams, A. Aldrich, and E. Theodorou, Model predictive path integral control using covariance variable importance sampling, preprint, http://arxiv.org/abs/1509.01149v2