Efficient Prioritization in Explicit Adaptive NMPC through Reachable-Space Search

Vishnu R. Desaraju and Nathan Michael
Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, 15213

This paper presents a computationally tractable explicit nonlinear model predictive control (NMPC) strategy that models and adapts to changes in plant dynamics. Explicit NMPC techniques enumerate all controllers derived from an NMPC formulation. However, the resulting database is exponential in the number of constraints and prohibitive for fast, online queries. Therefore, we propose to construct a database that is restricted to operation within the system’s reachable set under NMPC. To identify this reduced controller set, we construct a randomized search tree to explore the set of trajectories within the reachable set and apply the Experience-driven Predictive Control (EPC) algorithm to construct the database incrementally during the search. Additionally, we model transitions between controllers as a Markov chain with transition probabilities that inform a partial ordering on successors for each controller, thus enabling efficient search of the database at runtime. The resulting Explicit EPC algorithm thus consists of two phases: 1) offline generation of a simplified controller database and 2) efficient online application of the stored controllers. A set of simulation studies, including attitude control of a quadrotor micro air vehicle, demonstrate that the proposed approach enables the use of explicit adaptive NMPC for problems that would otherwise yield prohibitively large databases. We also present experimental evaluation of the proposed Explicit EPC algorithm implemented onboard a nano-quadrotor, thereby demonstrating that this approach enables adaptive NMPC on severely compute-constrained platforms.

I. Introduction

In order for autonomous systems to be deployed in complex and unknown settings, they must be able move safely and reliably, even as their dynamics change due to interactions with the environment. This ability fundamentally relies on the availability of feedback controllers that can ensure safety through constraint satisfaction and accurate trajectory tracking in the presence of dynamic model uncertainty. Reliable operation is further complicated by the limited onboard computation available to many systems and the fast timescales, in some cases on the order of milliseconds, in which the controller must produce an action in order to ensure system stability and safety. These restrictions necessitate the development of efficient control techniques that retain the safety and performance properties of more computationally complex approaches. Therefore, we propose an adaptive feedback control strategy that leverages an efficient, offline computed database of controllers to ensure accurate trajectory tracking and constraint satisfaction. We seek to develop an explicit nonlinear model predictive controller (NMPC) that leverages reachable-space search to overcome the exponential database growth of other explicit NMPC techniques. Furthermore, we represent the resulting controller database as a Markov chain that enables efficient online queries of the database, yielding a controller that is amenable to use on computationally constrained systems.

While some reactive feedback controllers enable constraint satisfaction, we seek an optimal control formulation to maximize system performance and guarantee constraint satisfaction. Preserving both optimality and constraint satisfaction in a feedback controller for a nonlinear system yields an NMPC formulation. NMPC employs a model of the nonlinear dynamics to predict system evolution over a finite horizon and optimize the control commands with respect to performance objectives and constraints. However, the use of a nonlinear model yields a computationally complex formulation as a nonlinear program.
Fast NMPC solution techniques that seek to address this complexity comprise three main categories: fast online optimization, explicit NMPC, and semi-explicit approaches. Online linear MPC formulations compute solutions at high rates through the use of efficient convex optimization techniques. Consequently, many fast NMPC techniques rely on convex approximations, such as sequential quadratic programming or iterative Linear Quadratic Regulator formulations, to achieve fast solution times. However, these techniques assume the optimization problem can be solved quickly and reliably at runtime and thus may not be feasible for systems with severe computational constraints. Additionally, the reliance on online optimization may raise software certifiability concerns for some applications.

In contrast, explicit MPC and NMPC approaches eliminate the need for online optimization by constructing a database of locally-optimal controllers derived via a receding horizon optimal control formulation. At runtime, these methods query the database to identify and apply the appropriate controller. However, explicit MPC techniques are known to scale poorly due to the exponential growth in the controller set with the number of constraints, leading to a prohibitively expensive database search. Consequently, numerous approximate explicit MPC strategies seek to improve the efficiency of database queries through the introduction of search trees connecting partitions of a reduced state-space. Alternatively, some approaches leverage function approximation techniques to compute a continuous mapping from state to control output that replaces the controller database and can be combined with a partitioning strategy to yield a hierarchical model. Another strategy is to restrict the size of the database by selecting a subset of the most commonly used controllers with interpolation between them at runtime.

Explicit MPC is part of a broader class of methodologies that generate a set of control policies during an offline training phase to enable high-rate online control. For example, the LQR-Tree algorithm leverages offline sums-of-squares optimization to construct sequences of controllers that are applied online to stabilize the system along specific trajectories. Other techniques, such as MPC-guided policy search and Learning Global Optima, employ demonstrations of optimal behavior to train a learner (e.g., $k$-NN or neural network model) that seeks to approximate the optimal control policy at a given query point. However, these techniques are limited by the number of optimal training examples that can be generated and therefore often focus on learning specific behaviors to avoid the same exponential growth in the number of policies as in explicit MPC.

Semi-explicit MPC and NMPC techniques combine a controller database with online optimization to balance the strengths and weaknesses of each. The Partial Enumeration and Nonlinear Partial Enumeration techniques incrementally construct a controller database by solving the MPC optimization problem online. If a locally-optimal controller is not found in the current database, the solution is introduced into the database for future use, thereby reducing the occurrence of online optimization computations. The Experience-driven Predictive Control (EPC) algorithm extends this idea by introducing online plant model adaptation to further simplify the required optimization. Due to their incremental and need-based optimization formulations, these approaches yield smaller databases than explicit MPC. While this improves computational tractability, the online optimization required to populate the database again raises certifiability concerns and may still be prohibitive for some severely constrained systems.

While explicit MPC seeks to enumerate all controllers and semi-explicit techniques learn from real experience gained through operation, many learning-based techniques employ synthetic experiences to generate or refine their policies. Action model-based approaches use prior experiences to construct a model that mimics the behavior of the world. The models aim to find optimal future actions via dynamic programming or stochastic shortest path that can then be used to provide synthetic experiences. Depending on the underlying learner, these experiences can be used to perform additional refinement (e.g., additional value iteration steps in a Q-learning framework) or to guide the learner toward better solutions. Other approaches aim to improve learning speed and performance by introducing “imagined” experiences to the training data provided to the learner. For example, a recurrent neural network trained on actual experiences can be used to predict comparable but imaginary scenarios to serve as more general training data for the original learner.

Learning is also often applied to simplify the dynamics model. Adaptive or learning-base MPC techniques leverage an online dynamics estimator or model learner to capture nonlinearities, exogenous perturbations, and other components that are difficult to model. While most of these techniques incorporate the changing dynamics model into an online optimization, the EPC algorithm computes a database of controllers that are parameterized by the system dynamics. It also leverages Locally Weighted Projection Regression to refine the dynamics model online and, as a result, compute controllers that adapt to the evolving system.
model.

In this work we propose the Explicit EPC algorithm, an explicit adaptive NMPC approach that builds upon the techniques employed in semi-explicit formulations by learning from synthetic experiences. One of the key observations from semi-explicit MPC is that, in practice, the system only uses a small number of the potential controllers that an explicit approach would compute. Therefore, we propose to compute a limited database consisting of the controllers required to track feasible trajectories, e.g., trajectories that lie within the reachable space of the system controlled by NMPC. To do so, we construct a randomized search tree through the reachable space to generate synthetic experiences and apply the EPC algorithm to incrementally construct a controller database from these experiences. We model transitions between controllers in the database as a Markov chain and use empirical transition probabilities to specify a partial ordering on successors for each controller. This approach improves the efficiency of database queries and permits further reductions of the database size, thereby enabling use on computationally constrained platforms.

While the reachable-space search shares similarities with the LQR-Tree algorithm, the proposed Explicit EPC algorithm addresses the problem of constrained optimal control for arbitrary (feasible) trajectory tracking rather than the generation of stabilizing trajectory or controller sequences. Alessio, et al.

Explicit Nonlinear Model Predictive Control (NMPC) constructs a database of locally optimal affine feedback controllers that solve a given NMPC problem without the need for online optimization. The Explicit EPC algorithm proposed in this section combines the Experience-driven Predictive Control (EPC) algorithm with a randomized search tree to generate synthetic experiences that can populate a database of local feedback controllers. This offline-constructed database can then be queried during online operation for high-rate control that is equivalent to solving the NMPC problem but requires no online optimization. Section A below summarizes the EPC algorithm, while Sect. B-D detail the contributions of this work.

II. Approach

Explicit Nonlinear Model Predictive Control (NMPC) constructs a database of locally optimal affine feedback controllers that solve a given NMPC problem without the need for online optimization. The Explicit EPC algorithm proposed in this section combines the Experience-driven Predictive Control (EPC) algorithm with a randomized search tree to generate synthetic experiences that can populate a database of local feedback controllers. This offline-constructed database can then be queried during online operation for high-rate control that is equivalent to solving the NMPC problem but requires no online optimization. Section A below summarizes the EPC algorithm, while Sect. B-D detail the contributions of this work.

A. Predictive Control Formulation

We consider the adaptive NMPC formulation used in EPC, summarized below, that leverages an online-learned model of plant dynamics to simplify the optimization problem. EPC employs techniques such as Locally Weighted Projection Regression (LWPR) to learn perturbations to the dynamics model online. Given a state-control pair, \((x, u)\), the model learner returns the anticipated error, \(\hat{p}\), between the predicted and actual next state. Combining \(\hat{p}\) with a first order Taylor-series approximation of the nonlinear equations of motion (computed about the current state \(x_0\) and a nominal control input \(u_r\) derived from the trajectory) yields an affine model of the system’s dynamics that evolves over time,

\[
\tilde{x}_{k+1} = A\tilde{x}_k + B\tilde{u}_k + c + \hat{p} \\
= A\tilde{x}_k + B\tilde{u}_k + \tilde{c},
\]

where \(\tilde{x}_k = x_k - x_0\), \(\tilde{u}_k = u_k - u_r\), and \(\tilde{c} = c + \hat{p}\).

This affine model enables the \(N\)-step receding-horizon control problem to be formulated as a quadratic program (QP) rather than as a nonlinear program,

\[
\begin{align*}
\text{argmin}_{\tilde{u}_k} & \quad \frac{1}{2} (\tilde{x}_{k+1} - \tilde{r}_{k+1})^T Q_{k+1} (\tilde{x}_{k+1} - \tilde{r}_{k+1}) + \frac{1}{2} (\tilde{u}_k - \tilde{u}_p)^T R_k (\tilde{u}_k - \tilde{u}_p) \\
\text{s.t.} & \quad \tilde{x}_{k+1} = A\tilde{x}_k + B\tilde{u}_k + \tilde{c} \\
& \quad G_{x_{k+1}} \tilde{x}_{k+1} \leq g_{x_{k+1}} \\
& \quad G_{u_k} \tilde{u}_k \leq g_{u_k} \\
& \quad \forall \ k = \{0, \ldots, N - 1\}
\end{align*}
\]
where \( \{r_1, \ldots, r_N\} \) denote \( N \) reference states along the current trajectory, \( \bar{r}_k = r_k - x_0 \), and \( \bar{u}_p \) is the compensatory control derived from the LWPR output.\(^{23}\) To simplify notation, we define \( x = [\bar{x}_1^T, \ldots, \bar{x}_N^T]^T \), \( r = [\bar{r}_1^T, \ldots, \bar{r}_N^T]^T \), \( u = [u_0^T, \ldots, u_{N-1}^T]^T \), \( u_p = [u_p^T, \ldots, u_p^T]^T \),

\[
\mathcal{B} = \begin{bmatrix} B & 0 & \ldots & 0 \\ AB & B & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ A^{N-1}B & A^{N-2}B & \ldots & B \end{bmatrix}, \quad c = \begin{bmatrix} \hat{c} \\ (A + I) \hat{c} \\ \vdots \\ \sum_{i=0}^{N-1} A^i \hat{c} \end{bmatrix}
\]

\[
\mathcal{Q} = \text{diag}(Q_1, \ldots, Q_N), \quad \mathcal{R} = \text{diag}(R_0, \ldots, R_{N-1}), \quad \mathcal{G}_x = \text{diag}(G_{x_1}, \ldots, G_{x_N}),
\]

\[
\mathcal{G}_u = \text{diag}(G_{u_0}, \ldots, G_{u_{N-1}}), \quad g_x = [g_{x_1}^T, \ldots, g_{x_N}^T]^T, \quad \text{and} \quad g_u = [g_{u_0}^T, \ldots, g_{u_{N-1}}^T]^T.
\]

Since \( \bar{x}_0 = 0 \), the QP (1) can we rewritten as

\[
\arg\min_u \frac{1}{2} (x - r)^T \mathcal{Q} (x - r) + \frac{1}{2} (u - u_p)^T \mathcal{R} (u - u_p)
\]

s.t. \( x = Bu + c \)

\[
\mathcal{G}_x x \leq g_x \\
\mathcal{G}_u u \leq g_u
\]

Substituting the system dynamics into the cost and constraints and dropping constant terms in the cost function yields an equivalent QP in terms of \( u \),

\[
\arg\min_u \frac{1}{2} u^T \mathcal{H} u + h^T u
\]

s.t. \( \Gamma u \leq \gamma \)

where \( \mathcal{H} = \mathcal{B}^T \mathcal{Q} \mathcal{B} + \mathcal{R} \), \( h = \mathcal{B}^T \mathcal{Q} (c - r) - \mathcal{R} u_p \),

\[
\Gamma = \begin{bmatrix} \mathcal{G}_x \mathcal{B} \\ \mathcal{G}_u \end{bmatrix}, \quad \text{and} \quad \gamma = \begin{bmatrix} g_x - \mathcal{G}_x c \\ g_u \end{bmatrix}
\]

As in other explicit and semi-explicit MPC techniques, we apply a standard result from multi-parametric QPs to derive optimal, local controllers from the Karush-Kuhn-Tucker (KKT) conditions for optimality,\(^{5}\)

\[
\mathcal{H} u + h + \Gamma^T \lambda = 0 \quad \Lambda (\Gamma u - \gamma) = 0,
\]

where \( \lambda \) is the vector of Lagrange multipliers and \( \Lambda = \text{diag}(\lambda) \). This allows \( u \) and \( \lambda \) to be reconstructed by solving a linear system derived from (2),

\[
\begin{bmatrix} \mathcal{H} & \Gamma_a^T \\ \Gamma_a & \Lambda \end{bmatrix} \begin{bmatrix} u_a \\ \lambda_a \end{bmatrix} = \begin{bmatrix} -h \\ \gamma_a \end{bmatrix}
\]

where the subscript \( a \) denotes rows corresponding to the active constraints (i.e., with \( \lambda > 0 \)) for a given solution. Assuming a linearly independent set of active constraints,\(^{33}\) the resulting \( u \) is affine in the predicted state error \( r \),

\[
u = \mathcal{E}_3 r - \left( \mathcal{E}_5 c - \mathcal{E}_4 \mathcal{R} u_p + \mathcal{E}_3 \begin{bmatrix} g_x^+ - \mathcal{G}_x c \\ g_x^- + \mathcal{G}_x c \\ g_u^+ \\ g_u^- \end{bmatrix}_a \right)
\]
where \( E_1 = \Gamma a H^{-1} \), \( E_2 = -(E_1 \Gamma a)^{-1} \), \( E_3 = E_1^T E_2 \), \( E_4 = H^{-1} + E_3 E_1 \), and \( E_5 = E_4 B^T Q \). Moreover, since the coefficients in (3) are all functions of \( A, B, \) and \( \hat{c} \), the overall control law \( \kappa(x_0, r_1, \ldots, r_N) \) can be written in terms of a parameterized feedback gain matrix \( K \) and feedforward vector \( k_f \).

\[
\kappa(x_0, r_1, \ldots, r_N) = K(A, B, \hat{c})r + k_f(A, B, \hat{c}).
\]  

(4)

The KKT condition checks (used to determine whether a previously computed controller is locally optimal) have a similar structure. The active Lagrange multipliers, \( \lambda \), also take a similar form to the control law,

\[
\lambda = -E_a r + \left( E_6 c - E_4^T \mathcal{R} u_p + E_2 \begin{bmatrix} g_x^+ - G_x c \\ -g_x + G_x c \\ g_u \\ -g_u \end{bmatrix} \right)
\]

(5)

where \( E_6 = E_4^T B^T Q \).

Therefore, given the current state, references, affine dynamics model, and set of active constraints, we can compute and validate the optimal affine feedback law via (3) and (5) and thus construct a database of controllers simply by storing the appropriate sets of active constraints and reconstructing the control output online. This online reconstruction of the control and KKT matrices also ensures that the computed control response is locally optimal even as the system dynamics evolve.

B. Controller Search via Randomized Trajectories

Explicit MPC techniques employ formulations similar to those detailed in Sect. A to exhaustively enumerate all controllers for a given MPC formulation (i.e., all possible combinations of active constraints). However, this enumeration results in a controller database that is exponentially large in the number of constraints. We therefore seek to construct a database containing only the controllers required by the system to traverse trajectories in the reachable set with time horizon \( T \),

\[
\mathcal{R}(x_0, u, T) = \left\{ \int_0^T f(x_0, u) dt \forall u \in \mathcal{U} \right\}
\]

where \( f(\cdot) \) denotes the nonlinear dynamics model, \( \mathcal{U} \) is the set of controls permitted by (1), and the trajectories are reasonably tracked by the system. As the control input at any point is of the form \( u = K r + k_f \), the reachable set under (1) is given by \( \mathcal{R}_T = \mathcal{R}(x_0, K r + k_f, T) \forall r \in \mathcal{X}^N \), where \( \mathcal{X} \) is the set of states permitted by (1). Therefore, we pursue a database that consists of local, affine controllers required to track trajectories in \( \mathcal{R}_T \) by searching over feasible trajectories in \( \mathcal{R}_T \).

Due to the high dimensionality of the search space, we choose to randomly sample trajectories in \( \mathcal{R}_T \) using the Closed-loop Rapidly-exploring Random Tree (CL-RRT) algorithm.\(^{34}\) CL-RRT grows a tree of dynamically feasible trajectories through \( \mathcal{X} \) by randomly sampling points in the reference space of the feedback controller (e.g., \( r \in \mathcal{X}^N \) for (1)), finding the closest node in the current tree, and forward-simulating the closed-loop system dynamics from that node to track the reference.\(^{35}\) Consequently, the resulting tree of trajectories can be interpreted as a sampling-based approximation of \( \mathcal{R}_T \).\(^{36}\) As CL-RRT is probabilistically complete,\(^{37,38}\) in the limit the tree of trajectories will be arbitrarily dense in any part of \( \mathcal{R}_T \) with a diverse set of trajectories.\(^{b}\) Moreover, if the system is differentially flat, the samples may be drawn from the lower-dimensional space of flat outputs, \( \mathcal{S} \subset \mathcal{X} \), and used to compute smooth polynomial trajectories, \( s(t) \in \mathcal{S} \cap \mathcal{R}_T \), that provide full state references \( r = [s(t_0), s(t_1), \ldots, s(t_{N-1})] \) to the controller.\(^{35}\) This formulation allows us to solve (1) at any point along any branch of the tree and generate a set of affine feedback controllers, as in Sect. A. We can also assess if a given set of controllers is sufficient to cover a variety of relevant operating domains (i.e., at each simulation step of each branch).

\(^{a}\)Assuming \( c \) constraints of the form \( a_{\text{min}} \leq a \leq a_{\text{max}} \), there may be up to \( 3^N c \) controllers.\(^{21}\)

\(^{b}\)We intentionally do not use RRT here as it includes an additional rewiring step that effectively homogenizes the branches in the tree to obtain asymptotic optimality. In contrast, standard RRT and CL-RRT enable branches to grow in arbitrary directions. This results in a greater diversity of trajectory segments and therefore improved coverage of \( \mathcal{R}_T \).
C. Controller Database Generation via Synthetic Experience

As these reachable space trajectories are generated via random sampling and closed-loop simulation, they also constitute a diverse set of synthetic experiences for the closed-loop system. We therefore leverage these synthetic experiences and propose the following approach to generate a database, $\mathcal{M}$, of affine feedback controllers (illustrated in Fig. 1a). As described in Alg. 1, the database is initialized to be empty, and we introduce a transition frequency matrix, $\Phi$, that records the number of transitions between each pair of controllers added to $\mathcal{M}$. We initialize the search tree with a nominal state, $x_0 \in \mathcal{X}$, and its corresponding flat outputs, $s_0 \in \mathcal{S}$. For each random sample $s'$ generated, we compute $s(t)$ as detailed in Sect. B. The system dynamics are forward-simulated along $s(t)$ using EPC with the current database of controllers (the simulation step size, $\Delta t$, matches the desired control rate) and $\Phi$ is incremented accordingly. If the appropriate controller is not found in the database, EPC solves the QP (1) and adds the resulting controller to the database (increasing the size of $\Phi$ as well).

The forward-simulation terminates upon reaching a predefined maximum segment duration, $T$, and a new node is added to the tree containing the end state and index of the final controller used. Following Reist et al., we terminate growing of the tree after we have generated $M$ consecutive samples with resulting trajectories covered by the controllers in $\mathcal{M}$, leading to no change in $\mathcal{M}$. After the database is complete, we can define an order-1 Markov chain that represents the transitions between controllers, with empirical transition probabilities defined by $\overline{\Phi} = \Phi$ with normalized rows. Sorting the outgoing transitions from each state of the Markov chain according to probability of occurrence yields a (strict) partial ordering, $\Omega$, of the controllers. This ordering informs the online query process detailed in Sect. D.

The proposed Markov chain representation enables two additional simplifications of the controller database. The first is model reduction via elimination of low-occupancy states to reduce storage requirements and query times, which is essential for deployment on systems with severe computational constraints. This is achieved by retaining only the $K$ most frequently visited states defined by the $K$ columns of $\Phi$ with the highest sums, or via more advanced approaches such as conservation of transition rates. The second simplification (illustrated in Fig. 1b) reduces the number of transitions out of each state so as to limit the query time per control iteration by retaining the $K$ highest probability transitions out of each state such that the sum of the retained transitions satisfies a threshold, $\rho$, or by leveraging more advanced techniques, such as entropy-based methods. Combinations of these two simplifications may also be applied, by preserving a limited set of states $\mathcal{M}'$ such that $\overline{\Phi}_{ij} \leq \epsilon \forall m_i \in \mathcal{M}', m_j \in \mathcal{M} \cap \mathcal{M}'$ over the next $N$ queries.

D. Online Database Query

Once the controller database, $\mathcal{M}$, is populated, it enables high-rate, online control that recovers the functionality of (1). As described in Alg. 2, we query $\mathcal{M}$ in each control iteration to identify the appropriate...
Algorithm 1 Controller Database Generation

1: \( \mathcal{M} \leftarrow \emptyset, \Phi \leftarrow 0, k \leftarrow 0 \)
2: Add root node \( n_0 = (x_0, s_0) \) to tree \( T \)
3: while \( k < M \) do
4: \( \) Generate sample \( s' \in S \)
5: \( \) Select parent node \( n^* = \arg\min_{n \in T} \| s_n - s' \| \)
6: \( \) Compute spline \( s(t) \) from \( s_n^* \) to \( s' \)
7: \( t_{\text{sim}} \leftarrow 0, x_{\text{sim}} \leftarrow x_n^*, j \leftarrow i_n^* \)
8: while \( t_{\text{sim}} < T \) do
9: \( \) Get \( r \) from \( s(t) \)
10: \( \) for each element \( m_i \in \mathcal{M} \) do
11: \( \) if \( x_{\text{sim}}, r \) satisfy KKT criteria (2) then
12: \( \) \( u \leftarrow \kappa_i(x_{\text{sim}}, r), \) controller\_found \( \leftarrow \) true
13: \( \) \( \Phi_{ij} \leftarrow \Phi_{ij} + 1, j \leftarrow i, k \leftarrow k + 1 \)
14: \( \) break
15: \( \) end if
16: \( \) end for
17: \( \) if controller\_found is false then
18: \( \) Solve QP (1) to generate new controller, \( \kappa_{\text{new}} = (K, k_{\text{ff}}) \)
19: \( \) Add new element \( m_{\text{new}} \) containing \( \kappa_{\text{new}} \) to \( \mathcal{M} \)
20: \( \) \( u \leftarrow \kappa_{\text{new}}(x_{\text{sim}}, r), j \leftarrow |\mathcal{M}|, k \leftarrow 0 \)
21: \( \) end if
22: \( t_{\text{sim}} \leftarrow t_{\text{sim}} + \Delta t_{\text{sim}}, x_{\text{sim}} \leftarrow x_{\text{sim}} + \int_0^{\Delta t_{\text{sim}}} f(x, u) d\tau \)
23: \( \) end while
24: \( \) Add new node \( n = (x_{\text{sim}}, s_{\text{sim}}, j) \) to \( T \)
25: \( \) end while
26: Compute ordering \( \Omega \) based on transition frequencies in \( \Phi \)

Algorithm 2 Controller Database Query

1: Inputs: \( \mathcal{M} \) and \( \Omega \) from Alg. 1, Current \( x, r \), Previous controller index \( j^* \)
2: for each element \( m_i \in \mathcal{M} \) ordered by \( \Omega_{j^*} \), do
3: \( \) if \( x, r \) satisfy KKT criteria (2) then
4: \( \) \( u \leftarrow \kappa_i(x, r) \)
5: \( \) \( \) return \( u \)
6: \( \) \( \) end if
7: \( \) \( \) end for
7: return \( u \) \( \leftarrow \) safety controller

controller \( \kappa_i \). However, as each control iteration represents a state transition in the Markov chain (including self-loops), we identify the next transition by iterating through \( \mathcal{M} \) according to the order specified by entry \( j^* \) in \( \Omega \), where \( j^* \) is the index of the previous controller applied. This ordering aims to reduce the number of controllers that must be evaluated (since evaluating the KKT conditions is more expensive than a simple lookup table query), thereby improving the query efficiency relative to orderings based on measures of controller utility.

Although the termination condition in Alg. 1 ensures a high probability of coverage, it cannot guarantee perfect coverage. Therefore, if no controller in the database is suitable, we apply a safety controller (e.g., a short horizon MPC with soft constraints\(^{22}\) or any other globally stabilizing controller) until the system returns to the region covered by \( \mathcal{M} \).

Finally, the controllers applied online retain key properties of EPC.\(^{23}\) As \( \kappa \) is parameterized by the system dynamics (Sect. A), it can be combined with an online model learner, such as LWPR, for online adaptation to changing plant dynamics undergoing exogenous disturbances. It also has similar stability properties in that the formulation in (1) permits the inclusion of several standard techniques to ensure stability of the underlying MPC formulation, and the explicit controller database exactly reproduces the results of this stabilizing MPC. Transitions to the safety controller can also be shown to preserve stability by enforcing an
average dwell time requirement.\textsuperscript{41}

III. Simulation Results

To demonstrate the performance of the proposed Explicit EPC algorithm, we seek to demonstrate the following results through a series of simulation trials: \textbf{R1}: severely reduced controller database size, \textbf{R2}: database coverage of controllers required at run-time. \textbf{R3}: improved computational efficiency via the Markov chain-based ordering and simplification, \textbf{R4}: stable control performance, \textbf{R5}: constraint satisfaction, \textbf{R6}: real-time computation of control commands, and \textbf{R7}: adaptation performance. We consider two applications: 1) pendulum control to demonstrate basic functionality of the proposed approach and 2) quadrotor attitude control to demonstrate the value of Markov chain simplification in achieving the control rates required for stability. The offline computed controller database for each application is evaluated through a series of real-time simulations that require the system to track a variety of random, smooth trajectories. All simulations are implemented in C++ via ROS\textsuperscript{42} and are run on an 2.9 GHz Intel mobile processor.

A. Pendulum Control

The first system is a simple pendulum modeled by two states (angle and angular velocity) and one control input (torque). The corresponding MPC formulation has a 10-step prediction horizon and enforces constraints on angular velocity and torque.

1. Database Computation

To generate the controller database offline, we apply Alg. 1 with a termination threshold, $M$, of 50000 consecutive queries, and as Fig. 2 illustrates, the database satisfies this threshold after computing 268 controllers. This number is in stark contrast to the $3^{20} \approx 3.49 \times 10^{9}$ possible controllers that a standard explicit MPC would aim to enumerate (R1). Figure 3 shows the Markov chain transition probabilities derived from the search process. Table 1 summarizes the database generation results.

2. Database Query Performance

To evaluate Alg. 2 with this controller database, the pendulum is commanded to track a series of random, smooth trajectories over the course of a 600 s trial. As Table 2 shows, the limited number of controllers in the database permit tracking a variety of trajectories with negligible use of the safety controller. This demonstrates that the proposed reachable-space search does cover the space of control scenarios that the system may encounter during operation (R2). Additionally, repeating the same trial with a simple search ordering (as employed by EPC) yields substantially longer query times than the proposed ordering based on Markov chain transition probabilities (R3). Figures 4 and 5 illustrate the control performance provided by this database in terms of trajectory tracking error (R7) and constraint satisfaction (R5).

<table>
<thead>
<tr>
<th>Tree Branches</th>
<th>Simulation Steps</th>
<th>Control Database Size</th>
<th>Explicit MPC Bound</th>
</tr>
</thead>
<tbody>
<tr>
<td>560</td>
<td>415517</td>
<td>268</td>
<td>$3^{20}$</td>
</tr>
</tbody>
</table>

Table 1: Statistics for the pendulum control database computation

<table>
<thead>
<tr>
<th></th>
<th>Queries</th>
<th>Coverage</th>
<th>Mean Query Time</th>
<th>Safety Control Calls</th>
</tr>
</thead>
<tbody>
<tr>
<td>Markov chain ordering</td>
<td>119165</td>
<td>99.987%</td>
<td>0.1316 ms</td>
<td>15</td>
</tr>
<tr>
<td>Simple ordering</td>
<td>119373</td>
<td>99.975%</td>
<td>0.7778 ms</td>
<td>30</td>
</tr>
</tbody>
</table>

Table 2: Statistics for the 600 s pendulum database evaluation

B. Quadrotor Attitude Control

The second application we consider is attitude control of a small quadrotor micro air vehicle. Quadrotors are highly dynamic systems that require attitude controllers that run at high feedback rates (e.g., 200 Hz),
and as a result, are sensitive to computational delays in the controller. We employ a standard nonlinear model of the attitude dynamics\(^{43}\) modeled by six states (attitude and angular velocities) and three control inputs (torque about each axis). The corresponding MPC formulation has a 15-step prediction horizon and enforces constraints on roll, pitch, and the three torques.

1. **Database Computation**

As with the pendulum, we first apply Alg. 1 with a termination threshold, \(M\), of 50000 consecutive queries. As Fig. 7 illustrates, the database satisfies this threshold after computing 570 controllers. This number is again drastically smaller than the \(3^{75} \approx 6.08 \times 10^{35}\) possible controllers that a standard explicit MPC would aim to enumerate (R1). Figure 8 shows the Markov chain transition probabilities derived from the search process, and Table 3 summarizes the database generation results.

2. **Database Query Performance**

To evaluate online control using the database, the quadrotor is commanded to track a series of random, smooth trajectories for a 600 s trial (commanded trajectories are in position and yaw, but contain sufficiently aggressive sections to excite the attitude dynamics, and for simplicity, the translational dynamics are controlled via LQR). We consider both the search order based on Markov chain transition probabilities and a simple search order as employed by EPC.\(^{23}\) During the first 250 s of the trial, the proposed Markov chain-based ordering strategy yields faster solution times than the simple ordering (R3), as shown in Table 4. Furthermore, there are no calls to the safety controller in this time, demonstrating that the proposed reachable-space search does cover the space of control scenarios that the system may encounter during operation (R2).

However, we also observe that the database is still too large for reliable, realtime operation as an extended query causes the quadrotor to crash partway through the trial with either search ordering. This failure stems from an aggressive trajectory commanded approximately 250 s into the simulation that requires a controller not found in the database. Both ordering strategies are unable to detect this omission without searching the entire database, resulting in a delay of over 100 ms that causes the failure.

3. **Database Simplification**

We therefore simplify the Markov chain by preserving the highest-probability outgoing edges for each state such that they represent 99% of transitions in the original model. This simplification allows queries to ter-
Figure 3: Markov chain transition probabilities with states that correspond to the relevant database controller enumeration for the pendulum feedback control system.

Table 3: Statistics for the quadrotor attitude control database computation

<table>
<thead>
<tr>
<th>Tree Branches</th>
<th>Simulation Steps</th>
<th>Database Size</th>
<th>Explicit MPC Bound</th>
</tr>
</thead>
<tbody>
<tr>
<td>3000</td>
<td>602060</td>
<td>570</td>
<td>375</td>
</tr>
</tbody>
</table>

Table 4: Statistics for the 600 s quadrotor attitude control database evaluation. Entries in red indicate failures due to a prolonged database search before applying the safety controller.

<table>
<thead>
<tr>
<th>Method</th>
<th>Number of Queries</th>
<th>Percent Coverage</th>
<th>Mean Loop Time</th>
<th>Safety Control Calls</th>
</tr>
</thead>
<tbody>
<tr>
<td>EPC ordering</td>
<td>49166</td>
<td>100.00%</td>
<td>1.49 ms</td>
<td>Failed on first call</td>
</tr>
<tr>
<td>MC ordering</td>
<td>49074</td>
<td>100.00%</td>
<td>1.24 ms</td>
<td>Failed on first call</td>
</tr>
<tr>
<td>MC simplification</td>
<td>118956</td>
<td><strong>99.97%</strong></td>
<td><strong>1.45 ms</strong></td>
<td>34</td>
</tr>
</tbody>
</table>
IV. Experimental Evaluation

To assess the real-time control performance of the proposed efficient explicit formulation, we implement the controller database onboard a Bitcraze Crazyflie 2.0 quadrotor: a 32 g, severely compute-constrained platform with a 168 MHz ARM processor and 192 KB of SRAM. In this section, we consider the problem of controlling the translational dynamics (outer loop) of the quadrotor, rather than the attitude dynamics as in Sect. B. The database for the outer loop is computed offline via Alg. 1 and applied online via Alg. 2.

<table>
<thead>
<tr>
<th>Roll (rad)</th>
<th>Pitch (rad)</th>
<th>x-Position (m)</th>
<th>y-Position (m)</th>
</tr>
</thead>
<tbody>
<tr>
<td>No adaptation</td>
<td>0.0908 0.0909</td>
<td>0.0400 0.0360</td>
<td></td>
</tr>
<tr>
<td>With adaptation</td>
<td>0.0597 0.0585</td>
<td>0.0299 0.0271</td>
<td></td>
</tr>
</tbody>
</table>

Table 5: Comparison of mean tracking errors with and without adaptation
at a rate of 100 Hz. Following a standard cascaded control formulation,\textsuperscript{43} we apply a PD control law for the inner loop to track the outer loop outputs. Position state feedback is provided via a motion capture system at 100 Hz and broadcast wirelessly to the vehicle, while the onboard IMU provides feedback for 500 Hz attitude control.

We generate the controller database with a horizon length of $N = 10$ control iterations to maintain tractability and with constraints on $x$-velocity, $y$-velocity, and the three control inputs. This yields a database with only 191 entries, as opposed to the $3^{50} \approx 7.18 \times 10^{23}$ total possible controllers (R1). Figure 12 illustrates the connectivity in the corresponding Markov chain before and after simplification. As the database only requires the active set and the simplified list of successors for each controller, the total memory required to store the database is only 3.9 KB. However, to reduce redundant online matrix operations, we precompute and cache some of the EPC matrices, raising the total memory usage to 8.8 KB.

To evaluate online control performance, the vehicle is commanded to track a linear trajectory, as shown in Fig. 13. Figure 14 depicts a set of transitions between controllers in the database as the vehicle executes this trajectory, and Fig. 15 shows the corresponding query times for the controller database. This trial demonstrates that the proposed approach is computationally tractable on a severely constrained platform and yields stable control performance (R4). Moreover, the total position control loop time, including the database query step, achieves a mean of 3.889 ms with a sufficiently low standard deviation to reliably achieve a 100 Hz update rate (R6). Table 6 summarizes the online control performance.

This realtime control performance stems from the Markov chain simplification. Each controller in the database has on average only 9 successors to evaluate (R3). Without the simplification, each controller would have 191 successors to evaluate and could cause the controller to block for nearly 100 ms, destabilizing the vehicle. Finally, even after the simplification, we observe minimal application of the safety controller (Table 6). As a result, the database yields 99.3% coverage of the set of controllers required to track this trajectory (R2).

Comparing Figs. 14 and 16 shows that the controller transitions occur as expected when the velocity approaches the constraint boundary. Moreover, these results illustrate that switching between controllers in
the simplified database largely yields constraint satisfaction (R5). Although we do observe very infrequent constraint violations in the experimental studies, this mirrors the performance of the EPC algorithm and may be mitigated by leveraging a robust version of EPC that addresses additional sources of uncertainty.44

V. Conclusions and Future Work

In this work, we have presented the Explicit EPC algorithm as a technique for constructing an explicit NMPC controller database that leverages reachable-space search to identify a limited set of controllers that cover nearly all potential operating domains. Transitions between controllers in the resulting database are modeled via a Markov chain that facilitates further simplification to enable high-rate constrained control. Furthermore, as the controllers in the database are parameterized by the system dynamics, they can be used in conjunction with an online model learner to enable constrained adaptive control. A set of simulation trials demonstrate the offline and online performance of the proposed approach applied to a simple pendulum and the high-rate, constrained, adaptive nonlinear control problem of quadrotor attitude control. Additionally, we demonstrate that this approach enables real-time predictive control on a nano quadrotor with severe computational constraints. Following these results, we intend to extend the formulation to take into account other sources of uncertainty, such as imperfect state information and communication latency, and refine the implementation onboard the small-scale quadrotor platform to further assess performance in real-world scenarios.

Acknowledgments

We gratefully acknowledge the support of ARL grant W911NF-08-2-0004 and thank Arjav Desai and Matt Collins for their assistance with the Crazyflie platform.

Table 6: Statistics for the Crazyflie evaluation trial.

<table>
<thead>
<tr>
<th>Num. Queries</th>
<th>Coverage</th>
<th>Query Time (Mean)</th>
<th>Query Time (Std.Dev)</th>
<th>Safety Control Calls</th>
</tr>
</thead>
<tbody>
<tr>
<td>8702</td>
<td>99.349%</td>
<td>3.889 ms</td>
<td>4.028 ms</td>
<td>57</td>
</tr>
</tbody>
</table>
Figure 7: Growth of the attitude controller database during the offline search.

References


Figure 8: Transition probabilities for the Markov chain with states corresponding to controllers in the database for the quadrotor attitude control scenario. The entries for the first 100 controllers are magnified for clarity.

Figure 9: Quadrotor attitude reference tracking performance via the control database.
Figure 10: Roll and pitch torque constraints are satisfied for the duration of the attitude control evaluation.

Figure 11: Roll and pitch torque commands satisfy constraints even in the presence of a 30% max torque disturbance.
Figure 12: Transition probability matrices (a) before and (b) after simplification of the Markov chain underlying the position controller in the Crazyflie flight experiments.
Figure 13: Snapshots of the Crazyflie tracking one lap of the linear trajectory used to evaluate real-time control feasibility.

Figure 14: The Crazyflie transitions between multiple controllers while executing the linear trajectory.

Figure 15: Controller database query times onboard the Crazyflie where even the spikes corresponding to the controller changes are below the 10 ms desired threshold.

Figure 16: Controller switches yield velocity constraint satisfaction with minimal violations (likely due to uncompensated sources of uncertainty).