2016 IEEE International Conference on Robotics and Automation (ICRA)
Stockholm, Sweden, May 16-21, 2016

Fast Nonlinear Model Predictive Control via Partial Enumeration

Vishnu R. Desaraju and Nathan Michael

Abstract— **In this work, we consider the problem of fast, accurate control of a robot with constrained dynamics. We present a new nonlinear model predictive control (MPC) technique, Nonlinear Partial Enumeration (NPE), that combines online and offline computation in a nonlinear version of the partial enumeration method for MPC, thereby dramatically decreasing the compute time per control iteration. We apply NPE to the problem of MAV flight and demonstrate through a set of simulation trials that NPE outperforms other fast control methodologies during aggressive motion and enables the system to learn a reusable set of local feedback controllers that enable more efficient operation over time.**

I. Introduction

For robots operating in challenging, real-world environments, the ability to move accurately and reliably is a fundamental capability, necessitating accurate control strategies that can operate at the timescales required for highly dynamic systems. As a result, in this work we aim to develop an intelligent, high-rate, feedback control strategy that enables accurate and increasingly efficient operation.

Traditional feedback control strategies can run at very high rates due to their simplicity. However, to achieve this level of simplicity, these approaches are purely reactive and typically do not account for system limitations (e.g., actuator constraints). As a result, they can lead to degraded performance in more challenging settings outside their nominal operating regime. Conversely, optimal control techniques do explicitly look ahead and consider such constraints but incur a computational penalty. Model predictive control (MPC) seeks a middle ground by casting the control problem as a finite horizon constrained optimization. MPC ensures the generated commands obey actuator and operating limits by optimizing over the predicted evolution of the system dynamics.

While MPC was traditionally applied to systems with slow timescales, recent advances have made this a viable control strategy for systems with timescales on the order of milliseconds, such as agile autonomous robots. Many of these fast MPC techniques can be classified as either online or offline approaches. Online methods seek to reduce the solution time through standard optimization techniques, including warm-starting the solution using a previous solution [1], exploiting problem structure, and trading speed for optimality [2], but the dependence on optimization in an essential control loop can lead to reliability and certifiability concerns. Offline approaches, referred to as explicit MPC, precompute a simple control law (e.g., piecewise-affine [3] or via function approximation [4]) that reproduces the MPC solution in different regions of the state-space, thereby avoiding online optimization entirely. While these controllers are very fast and easily certifiable, they are known to scale poorly even for low-dimensional problems [4, 5]. Partial Enumeration (PE) MPC [5] seeks a balance between the online and offline methods, using infrequent online optimization to update a bounded explicit MPC mapping.

However, these fast MPC approaches are typically limited to linear systems, and although they can be applied to nonlinear systems via linearization, the accuracy of the resulting motion is heavily dependent on the fidelity of the prediction model used. This naturally motivates the use of nonlinear MPC (NMPC), as the nonlinear motion model will predict system evolution more accurately than any linear approximation about a nominal operating point. Many NMPC approaches use sequential quadratic programming (SQP), and as a result, share similarities with other techniques based on local quadratic approximations, such as DDP [6], LQR-Trees [7], and ILQR [8]. However, online optimization using the nonlinear model comes at the expense of added computational complexity, while offline approaches suffer from the same limitations as explicit MPC, in addition to increased offline computation [9].

Therefore, in this work we present Nonlinear Partial Enumeration (NPE), a NMPC technique that combines online and offline computation to yield a nonlinear version of Partial Enumeration MPC, thereby dramatically decreasing the solution time per NMPC iteration and making it viable for use on systems with dynamics that evolve on the order of milliseconds. The proposed approach leverages a parallelized structure, ensuring that a feasible solution is returned at the required rate while employing slower optimization techniques to learn local control laws that capture the functionality of standard NMPC. Using the problem of aggressive MAV flight as a guiding example, we leverage this formulation to demonstrate through a set of simulation trials the functionality and performance of NPE, as well as its ability to enable online learning of reusable local feedback control laws.

II. Nonlinear Partial Enumeration

In this section, we present a novel nonlinear extension of the Partial Enumeration (PE) technique [5] to construct online a piecewise-affine control law as the solution to a NMPC problem. Just as linear PE leverages explicit MPC techniques based on multi-parametric quadratic programming (mp-QP), we employ ideas from explicit NMPC to combine solutions

The authors are with the Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213, USA. {rajeswar, nmichael}@cmu.edu
We gratefully acknowledge the support of ARL grant W911NF-08-2-0004

to a nonlinear program (NLP) with local mp-QPs [9], thereby reducing the number of NLPs that must be solved online.

We first formulate a finite horizon NLP to compute the control sequence $\{\mathbf{u}_1, \dots, \mathbf{u}_N\}$ given the current state $\mathbf{x}_0$ and references $\mathbf{r}_1, \dots, \mathbf{r}_N$ (e.g., from a desired trajectory),

$$\begin{aligned} \underset{\mathbf{u}_k}{\operatorname{argmin}} & \sum_{k=0}^{N-1} J(\mathbf{x}_{k+1}, \mathbf{r}_{k+1}, \mathbf{u}_k) \\ \text{s.t.} \quad & \dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}) \\ & g(\mathbf{x}_{k+1}, \mathbf{u}_k) \leq 0 \; \forall k = 0, \dots, N-1 \end{aligned} \tag{1}$$

where the differential equation constraint is enforced via numerical integration. The resulting optimal control sequence must satisfy the first-order KKT conditions

$$\nabla_{\mathbf{u}} L(\mathbf{x}, \mathbf{r}, \mathbf{u}, \boldsymbol{\lambda}) = \mathbf{0} \tag{2}$$

$$\boldsymbol{\Lambda} g(\mathbf{x}, \mathbf{u}) = \mathbf{0} \tag{3}$$

$$\boldsymbol{\lambda} \geq \mathbf{0} \tag{4}$$

$$g(\mathbf{x}, \mathbf{u}) \leq \mathbf{0} \tag{5}$$

where $\boldsymbol{\lambda}$ is a vector of Lagrange multipliers, $L(\mathbf{x}, \mathbf{r}, \mathbf{u}, \boldsymbol{\lambda}) = J(\mathbf{x}, \mathbf{r}, \mathbf{u}) + \boldsymbol{\lambda}^{\mathrm{T}} g(\mathbf{x}, \mathbf{u})$ is the corresponding Lagrangian, and $\boldsymbol{\Lambda} = \operatorname{diag}(\boldsymbol{\lambda})$. In a standard online NMPC framework, the first element of this sequence would be applied and the problem re-solved from the updated state.

Instead, given a sequence of control inputs $\{\mathbf{u}_k^*\}$ that are the solution to (1) at $\mathbf{x}^*$, we define difference variables $\bar{\mathbf{x}} = \mathbf{x} - \mathbf{x}^*$, $\bar{\mathbf{r}} = \mathbf{r} - \mathbf{x}^*$, and $\bar{\mathbf{u}} = \mathbf{u} - \mathbf{u}^*$ and formulate the local QP as

$$\begin{aligned} \underset{\mathbf{u}_k}{\operatorname{argmin}} & \sum_{k=0}^{N-1} \frac{1}{2} (\bar{\mathbf{x}}_{k+1} - \bar{\mathbf{r}}_{k+1})^{\mathrm{T}} \mathbf{Q} (\bar{\mathbf{x}}_{k+1} - \bar{\mathbf{r}}_{k+1}) + \frac{1}{2} \bar{\mathbf{u}}_k^{\mathrm{T}} \mathbf{R} \bar{\mathbf{u}}_k \\ \text{s.t.} \quad & \bar{\mathbf{x}}_{k+1} = \mathbf{A} \bar{\mathbf{x}}_k + \mathbf{B} \bar{\mathbf{u}}_k \\ & \mathbf{G}_{\mathbf{x}} \bar{\mathbf{x}}_{k+1} \leq \mathbf{g}_{\mathbf{x}} \\ & \mathbf{G}_{\mathbf{u}} \bar{\mathbf{u}}_k \leq \mathbf{g}_{\mathbf{u}} \\ & \forall \; k = 0, \dots, N-1 \end{aligned} \tag{6}$$

where $\mathbf{Q}$ and $\mathbf{R}$ are given by the Hessian of $J(\mathbf{x}, \mathbf{r}, \mathbf{u})$ and $\mathbf{A}, \mathbf{B}, \mathbf{G}_{\mathbf{x}}, \mathbf{G}_{\mathbf{u}}, \mathbf{g}_{\mathbf{x}}, \mathbf{g}_{\mathbf{u}}$ are given by the linearization of $f(\mathbf{x}, \mathbf{u})$ and $g(\mathbf{x}, \mathbf{u})$ about $\{\mathbf{u}_k^*\}$ and $\mathbf{x}^*$. To simplify the formulation, we note that the linearized dynamics over N steps can be rewritten as $\boldsymbol{x} = \boldsymbol{\mathcal{A}} \bar{\mathbf{x}}_0 + \boldsymbol{\mathcal{B}} \boldsymbol{u}$, where

$$\boldsymbol{x} = \begin{bmatrix} \bar{\mathbf{x}}_1 \\ \bar{\mathbf{x}}_2 \\ \vdots \\ \bar{\mathbf{x}}_N \end{bmatrix} \quad \boldsymbol{u} = \begin{bmatrix} \bar{\mathbf{u}}_0 \\ \bar{\mathbf{u}}_1 \\ \vdots \\ \bar{\mathbf{u}}_{N-1} \end{bmatrix}$$

$$\boldsymbol{\mathcal{A}} = \begin{bmatrix} \mathbf{A} \\ \mathbf{A}^2 \\ \vdots \\ \mathbf{A}^N \end{bmatrix} \quad \boldsymbol{\mathcal{B}} = \begin{bmatrix} \mathbf{B} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{AB} & \mathbf{B} & \dots & \mathbf{0} \\ \vdots & \vdots & \ddots & \\ \mathbf{A}^{N-1}\mathbf{B} & \mathbf{A}^{N-2}\mathbf{B} & \dots & \mathbf{B} \end{bmatrix}$$

Additionally, let $\boldsymbol{\mathcal{H}} = \boldsymbol{\mathcal{B}}^{\mathrm{T}} \boldsymbol{\mathcal{Q}} \boldsymbol{\mathcal{B}} + \boldsymbol{\mathcal{R}}$, where $\boldsymbol{\mathcal{Q}} = \operatorname{diag}(\mathbf{Q}, \dots, \mathbf{Q})$ and $\boldsymbol{\mathcal{R}} = \operatorname{diag}(\mathbf{R}, \dots, \mathbf{R})$ of the appropriate dimensions, and $\boldsymbol{h} = \boldsymbol{\mathcal{B}}^{\mathrm{T}} \boldsymbol{\mathcal{Q}} (\boldsymbol{\mathcal{A}} \bar{\mathbf{x}}_0 - \boldsymbol{r})$, where $\boldsymbol{r}$ is defined analogously to $\boldsymbol{x}$. Similarly, let $\boldsymbol{\mathcal{G}}_{\mathbf{x}} = \operatorname{diag}(\mathbf{G}_{\mathbf{x}}, \dots, \mathbf{G}_{\mathbf{x}})$, $\boldsymbol{\mathcal{G}}_{\mathbf{u}} = \operatorname{diag}(\mathbf{G}_{\mathbf{u}}, \dots, \mathbf{G}_{\mathbf{u}})$, $\boldsymbol{g}_{\mathbf{x}} = \left[\, \mathbf{g}_{\mathbf{x}}^{\mathrm{T}}, \dots, \mathbf{g}_{\mathbf{x}}^{\mathrm{T}} \right]^{\mathrm{T}}$, $\boldsymbol{g}_{\mathbf{u}} = \left[\, \mathbf{g}_{\mathbf{u}}^{\mathrm{T}}, \dots, \mathbf{g}_{\mathbf{u}}^{\mathrm{T}} \right]^{\mathrm{T}}$, and

$$\boldsymbol{\Gamma} = \begin{bmatrix} \boldsymbol{\mathcal{G}}_{\mathbf{x}} \boldsymbol{\mathcal{B}} \\ \boldsymbol{\mathcal{G}}_{\mathbf{u}} \end{bmatrix} \quad \boldsymbol{\gamma} = \begin{bmatrix} \boldsymbol{g}_{\mathbf{x}} - \boldsymbol{\mathcal{G}}_{\mathbf{x}} \boldsymbol{\mathcal{A}} \bar{\mathbf{x}}_0 \\ \boldsymbol{g}_{\mathbf{u}} \end{bmatrix}$$

We can then rewrite (6) in an equivalent form (dropping constant terms in the cost function) as

$$\begin{aligned} \underset{\boldsymbol{u}}{\operatorname{argmin}} & \; \frac{1}{2} \boldsymbol{u}^{\mathrm{T}} \boldsymbol{\mathcal{H}} \boldsymbol{u} + \boldsymbol{h}^{\mathrm{T}} \boldsymbol{u} \\ \text{s.t.} \quad & \boldsymbol{\Gamma} \boldsymbol{u} \leq \boldsymbol{\gamma} \end{aligned} \tag{7}$$

This form facilitates writing the KKT conditions (2) and (3) for the local QP as

$$\begin{aligned} \boldsymbol{\mathcal{H}} \boldsymbol{u} + \boldsymbol{h} + \boldsymbol{\Gamma}^{\mathrm{T}} \boldsymbol{\lambda} &= \mathbf{0} \\ \boldsymbol{\Lambda} (\boldsymbol{\Gamma} \boldsymbol{u} - \boldsymbol{\gamma}) &= \mathbf{0} \end{aligned} \tag{8}$$

If we only consider the active constraints (i.e., with $\lambda > 0$) for a given solution, we can reconstruct $\boldsymbol{u}$ and $\boldsymbol{\lambda}$ by solving a linear system derived from (8), where the subscript a indicates rows corresponding to the active constraints

$$\begin{bmatrix} \boldsymbol{\mathcal{H}} & \boldsymbol{\Gamma}_a^{\mathrm{T}} \\ \boldsymbol{\Gamma}_a & \mathbf{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{u} \\ \boldsymbol{\lambda}_a \end{bmatrix} = \begin{bmatrix} -\boldsymbol{h} \\ \boldsymbol{\gamma}_a \end{bmatrix} \tag{9}$$

Assuming the active constraints are linearly independent (Bemporad, et al. [3] suggest alternatives if this assumption fails), the resulting local QP control law $\boldsymbol{u}^*$ is affine in $\bar{\mathbf{x}}_0$ and $\boldsymbol{r}$. If we denote the NLP solution as $\boldsymbol{u}_{\mathrm{NL}}$, the overall control law $\kappa(\mathbf{x}_0, \boldsymbol{r})$ then consists of an affine feedback term computed via the local QP and a constant feedforward term determined by the NLP, which we can combine into gain matrices $\mathbf{K}_1$ and $\mathbf{K}_2$ and a feedforward vector $\mathbf{k}_{\mathrm{ff}}$

$$\begin{aligned} \kappa(\bar{\mathbf{x}}_0, \boldsymbol{r}) &= \boldsymbol{u}^*(\bar{\mathbf{x}}_0, \boldsymbol{r}) + \boldsymbol{u}_{\mathrm{NL}} \\ &= \mathbf{K}_1 \bar{\mathbf{x}}_0 + \mathbf{K}_2 \boldsymbol{r} + \mathbf{k}_{\mathrm{ff}} \end{aligned} \tag{10}$$

However, since the feedback term is derived from a local approximation, we must determine the region of validity of this solution. A similar region is computed in mp-QP explicit NMPC [9] by formulating additional optimization problems to find hard bounds on the suboptimality of $\boldsymbol{u}^*$ relative to the NLP solution. However, this is intractable for online computation. We instead follow PE and determine the region of validity by first checking the remaining KKT conditions (4) and (5) for the local QP. We can then further restrict the region of validity via commonly used measures of suboptimality, such as the KKT tolerance criteria used to determine when to terminate iterations in sequential quadratic programming [10]

$$\begin{aligned} |\nabla_u L(\mathbf{x}, \mathbf{r}, \mathbf{u}, \boldsymbol{\lambda})| &\leq \epsilon \\ -\boldsymbol{\delta} \leq \boldsymbol{\Lambda} g(\mathbf{x}, \mathbf{u}) &\leq \boldsymbol{\delta} \end{aligned} \tag{11}$$

where ϵ and $\boldsymbol{\delta}$ are predefined tolerance parameters.

Introducing these nonlinear KKT criteria enables us to extend the state of the art for fast NMPC by defining a Nonlinear Partial Enumeration (NPE) strategy, as described in Algorithm 1. As in linear PE, we aim to construct a mapping $\mathcal{M}$ from regions of the state space to local, affine

Algorithm 1 Nonlinear Partial Enumeration

1: $\mathcal{M} \leftarrow \emptyset$ or $\mathcal{M}_{\text{prior}}$
2: `solution_found` $\leftarrow$ false
3: `nco_running` $\leftarrow$ false
4: **while** control is enabled **do**
5: $\quad \mathbf{x}_0 \leftarrow$ current system state
6: $\quad \boldsymbol{r} \leftarrow$ current reference sequence
7: $\quad$ **for** each element $m_i \in \mathcal{M}$ **do**
8: $\quad\quad$ Compute $\boldsymbol{u}, \boldsymbol{\lambda}$ via (9)
9: $\quad\quad$ **if** $\mathbf{x}_0, \boldsymbol{r}$ satisfy KKT criteria (4), (5) applied to (7) and (11) **then**
10: $\quad\quad\quad$ `importance`$_i \leftarrow$ current time, sort $\mathcal{M}$
11: $\quad\quad\quad$ `solution_found` $\leftarrow$ true
12: $\quad\quad\quad$ Apply affine control law (10) from m_i
13: $\quad\quad\quad$ **break**
14: $\quad\quad$ **end if**
15: $\quad$ **end for**
16: $\quad$ **if** `solution_found` is false **then**
17: $\quad\quad$ **if** `nco_running` is false **then**
18: $\quad\quad\quad$ Start Alg. 2 (parallel thread)
19: $\quad\quad$ **end if**
20: $\quad\quad$ Apply intermediate control via linear MPC (13)
21: $\quad$ **end if**
22: **end while**

Algorithm 2 NPE: New Controller Optimization

1: `nco_running` $\leftarrow$ true
2: $\boldsymbol{u}_{\text{NL}} \leftarrow$ Solution to NLP (1) at $\mathbf{x}_0$
3: $(\mathbf{K}_1, \mathbf{K}_2, \mathbf{k}_{\text{ff}}) \leftarrow$ Local QP (7) solution about $\mathbf{x}_0, \boldsymbol{u}_{\text{NL}}$
4: **if** NLP and local QP solutions are found **then**
5: $\quad$ **if** $|\mathcal{M}| >$ max table size **then**
6: $\quad\quad$ Remove element from $\mathcal{M}$ with minimum `importance`
7: $\quad$ **end if**
8: $\quad$ Add new element $m_{\text{new}} =$ $(\mathbf{x}_0, \mathbf{K}_1, \mathbf{K}_2, \mathbf{k}_{\text{ff}}, \texttt{importance})$ to $\mathcal{M}$
9: **end if**
10: `nco_running` $\leftarrow$ false

controllers. Each element in $\mathcal{M}$ is defined by a nominal state, an affine controller, and an `importance` score that is used to order the elements. Intuitively, the system is not expected to transition between regions frequently, so we choose to order the elements by when they were last used (Pannocchia et al. [5] discuss other strategies). $\mathcal{M}$ can either be initialized as an empty set or with information from previous runs to reduce the need for online optimization. In each control iteration, we first evaluate the KKT criteria at the current state and reference for each element in $\mathcal{M}$ (lines 7-9). If any element satisfies the criteria, we update its `importance` value to the current time (line 10) and apply the corresponding affine controller (line 12). In this case, no online optimization is required to generate a locally optimal feedback control law.

If none of the elements satisfy the criteria, we use a parallelized approach to compute and add a new element to $\mathcal{M}$, without blocking the main control loop. As described in Alg. 2, we solve the NLP and local QP (lines 2-3), and in line 8 add the corresponding element to $\mathcal{M}$ (as defined in (10)). To control the amount of time spent querying the mapping, we can restrict its size and, if necessary, remove the lowest-`importance` element prior to adding m_{new}, as shown in lines 5-6.

While the new element of $\mathcal{M}$ is being computed, we use an intermediate controller to quickly compute suboptimal commands that ensure stability and constraint satisfaction (Alg. 1, line 20). The intermediate controller is formulated as a linear MPC with a shorter horizon $\tilde{N}$ and soft constraints:

$$\begin{aligned} \underset{\mathbf{u}_k, \mathbf{c}_k}{\operatorname{argmin}} \sum_{k=0}^{\tilde{N}-1} & \frac{1}{2}(\bar{\mathbf{x}}_{k+1} - \bar{\mathbf{r}}_{k+1})^{\mathrm{T}} \mathbf{Q} (\bar{\mathbf{x}}_{k+1} - \bar{\mathbf{r}}_{k+1}) \\ & + \frac{1}{2}\bar{\mathbf{u}}_k^{\mathrm{T}} \mathbf{R} \bar{\mathbf{u}}_k + \frac{1}{2}\mathbf{c}_k^{\mathrm{T}} \mathbf{P} \mathbf{c}_k \\ \text{s.t.} \quad & \bar{\mathbf{x}}_{k+1} = \mathbf{A}\bar{\mathbf{x}}_k + \mathbf{B}\bar{\mathbf{u}}_k \\ & \mathbf{G}_{\mathbf{x}} \bar{\mathbf{x}}_{k+1} - \mathbf{c}_k \leq \mathbf{g}_{\mathbf{x}} \\ & \mathbf{G}_{\mathbf{u}} \bar{\mathbf{u}}_k \leq \mathbf{g}_{\mathbf{u}} \\ & \forall\ k = 0, \ldots, N-1 \end{aligned} \tag{12}$$

The bounds on the control inputs are enforced as hard constraints to ensure the resulting commands are feasible, while slack variables $\mathbf{c}_k$ are added to the state constraints to allow violations with some cost penalty $\mathbf{P}$. The slack variables are unconstrained to ensure existence of a solution.

As in the local QP, this can be re-written such that $\mathbf{u}_k$ and $\mathbf{c}_k$ are the only decision variables,

$$\begin{aligned} \underset{\boldsymbol{u}, \boldsymbol{c}}{\operatorname{argmin}} \ & \frac{1}{2}\boldsymbol{u}^{\mathrm{T}} \boldsymbol{\mathcal{H}} \boldsymbol{u} + \boldsymbol{h}^{\mathrm{T}} \boldsymbol{u} \ + \frac{1}{2}\boldsymbol{c}^{\mathrm{T}} \boldsymbol{\mathcal{P}} \boldsymbol{c} \\ \text{s.t.} \quad & \boldsymbol{\Gamma} \boldsymbol{u} - \boldsymbol{c} \leq \boldsymbol{\gamma} \end{aligned} \tag{13}$$

where $\boldsymbol{\mathcal{P}}$ and $\boldsymbol{c}$ aggregate $\mathbf{P}$ and $\mathbf{c}_k$, respectively.

As this process iterates, $\mathcal{M}$ will be populated by the most useful elements, reducing the dependence on the intermediate controller. The combination of controllers queried from $\mathcal{M}$ and the intermediate controller ensures the existence of a locally optimal feedback controller at every iteration. Since the computationally expensive components of the algorithm are run in parallel, NPE will compute high-rate, stabilizing commands at all times, thereby enabling fast, nearly optimization-free but minimally-suboptimal control that improves over time.

III. Application to MAV flight

To illustrate the performance of our proposed NPE approach, we consider the specific case of a quadrotor micro aerial vehicle (MAV) tracking aggressive trajectories. The dynamics of a quadrotor can be modeled as a 12 dimensional nonlinear system whose state $\mathbf{x} = \begin{bmatrix}\mathbf{p}^{\mathrm{T}} & \mathbf{v}^{\mathrm{T}} & \boldsymbol{\xi}^{\mathrm{T}} & \boldsymbol{\omega}^{\mathrm{T}}\end{bmatrix}^{\mathrm{T}}$ consists of position ($\mathbf{p}$), velocity ($\mathbf{v}$), attitude ($\boldsymbol{\xi} = \begin{bmatrix}\phi & \theta & \psi\end{bmatrix}^{\mathrm{T}}$), and angular velocity ($\boldsymbol{\omega}$). Attitude is represented by roll (ϕ), pitch (θ), and yaw (ψ) angles following the ZYX convention. The control input $\mathbf{u} = \begin{bmatrix}F & \boldsymbol{\tau}^{\mathrm{T}}\end{bmatrix}^{\mathrm{T}}$ consists

of thrust along the $+z$ body axis (F) and moments about each of the 3 body axes ($\boldsymbol{\tau} = \begin{bmatrix}\tau_\phi & \tau_\theta & \tau_\psi\end{bmatrix}^{\mathrm{T}}$). The system's time evolution is governed by $\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u})$, where

$$f(\mathbf{x}, \mathbf{u}) = \begin{cases} \dot{\mathbf{p}} & = \mathbf{v} \\ \dot{\mathbf{v}} & = \frac{1}{m} F \mathbf{R}_{\boldsymbol{\xi}} \mathbf{e}_3 - g\mathbf{e}_3 \\ \dot{\boldsymbol{\xi}} & = \mathbf{S}_{\boldsymbol{\xi}} \boldsymbol{\omega} \\ \dot{\boldsymbol{\omega}} & = \mathbf{J}^{-1}\left(\boldsymbol{\tau} - \boldsymbol{\omega} \times \mathbf{J}\boldsymbol{\omega}\right) \end{cases} \tag{14}$$

The constants g, m, and $\mathbf{J}$ denote gravity, vehicle mass, and inertia, respectively. The vector $\mathbf{e}_3$ is the third column of the 3×3 identity matrix, $\mathbf{R}_{\boldsymbol{\xi}}$ denotes the rotation matrix formed from the ZYX Euler angles $\boldsymbol{\xi}$ that takes vectors from body frame to world frame, and $\mathbf{S}_{\boldsymbol{\xi}}$ is the inverse of the Jacobian that relates ZYX Euler angle rates to angular velocities [11].

To formulate the NMPC problem, we select a standard linear-quadratic cost function

$$J(\mathbf{x}, \mathbf{r}, \mathbf{u}) = \frac{1}{2}(\mathbf{x} - \mathbf{r})^{\mathrm{T}} \mathbf{Q} (\mathbf{x} - \mathbf{r}) + \frac{1}{2}\mathbf{u}^{\mathrm{T}} \mathbf{R} \mathbf{u}$$

and define $g(\mathbf{x}, \mathbf{u})$ to enforce the following constraints

$$\begin{aligned} \mathbf{v}_{\min} \leq \mathbf{v} \leq \mathbf{v}_{\max} \\ \boldsymbol{\xi}_{\min} \leq \boldsymbol{\xi} \leq \boldsymbol{\xi}_{\max} \\ \mathbf{u}_{\min} \leq \mathbf{u} \leq \mathbf{u}_{\max} \end{aligned}$$

By using (14) as the dynamics constraint in (1), the controller will directly compute force and moment commands from $\mathbf{r}$, without the need for intermediate commands often seen in quadrotor control [11].

IV. Results

To assess the performance of our proposed NPE algorithm, we conducted a series of quadrotor flights using a high-fidelity simulation environment. The simulator and controller are built around ROS [12], and the controller uses the NLopt [13] library to solve the NLP and qpOASES [14] for the local QP. The simulation is run on a 2.9 GHz Intel mobile processor. In practice, quadrotor attitude controllers are often run at high rates (e.g., greater than 200 Hz) for reliable stabilization. Since our formulation directly computes the forces and moments (as an attitude controller would, see Sect. III), we require the controller to return solutions at 200 Hz. Given the relative degree of the quadrotor dynamics, we choose a ten-step prediction horizon with a step size of 20ms, thereby allowing the controls to have a non-trivial effect on position states over the course of the predicted motion.

We first consider a scenario (shown in Fig. 1) in which the quadrotor must track a linear trajectory that requires increasing speeds every lap (ranging from 0.6 m/s to 3.0 m/s). We compare the performance of NPE against a proportional-derivative (PD) controller and linear MPC. The linear MPC follows the formulation in (13) and uses the same parameters as in NPE. It uses a model of the system dynamics that is linearized about the nominal hover state and commands. These controllers are chosen as they can achieve the 200 Hz update requirement. A standard NMPC implementation is three orders of magnitude slower due to the NLP solver (see

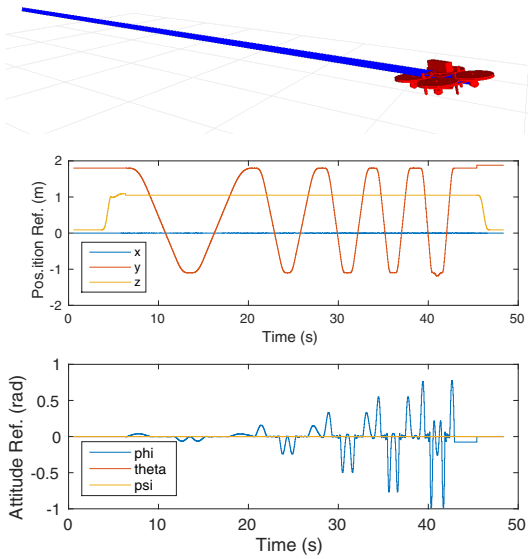


Fig. 1: Reference trajectory for the first test scenario

Table I), and therefore is not viable for comparison in this scenario.

As Fig. 2 illustrates, PD control is able to track the trajectory well at lower speeds but degrades at higher speeds due to overshoot and the resulting large oscillations. Linear MPC does not exhibit this overshoot, due to its predictive model. However, it does suffer from sustained tracking error and large roll angles due to the choice of linearization point. Linearizing about a non-zero roll command can actually eliminate this error in one direction, but consequently increases the error during the return lap. Since NPE uses the NLP to provide a feed-forward term when computing a new controller, the linearization point for the local QP and resulting feedback controllers is chosen intelligently, resulting in reduced tracking error and less severe roll angles.

To better illustrate NPE's ability to learn, use, and reuse controllers, we consider another scenario in which the quadrotor is repeatedly commanded to take-off and land aggressively, i.e., by commanding a 1.5m step change in the desired altitude. NPE is initialized with an empty set of controllers, and Fig. 3 shows the evolution of the control strategy over repeated takeoff and landing sequences. The first column illustrates the dependence on the intermediate controller, while the remaining columns correspond to the online computed controllers as they are added to the mapping $\mathcal{M}$. The first local feedback controller is computed about a common operating mode (near hover, away from all constraint boundaries) and is analogous to a finite-horizon LQR solution. Consequently, it is applicable to non-aggressive portions of the test scenario, as is shown by the high usage times in the second column, but still requires substantial use of the intermediate controller for aggressive motion (nearly a 2:1 ratio for usage duration). However, as additional, specialized controllers are computed, NPE's reliance on the intermediate controller decreases until the system operates

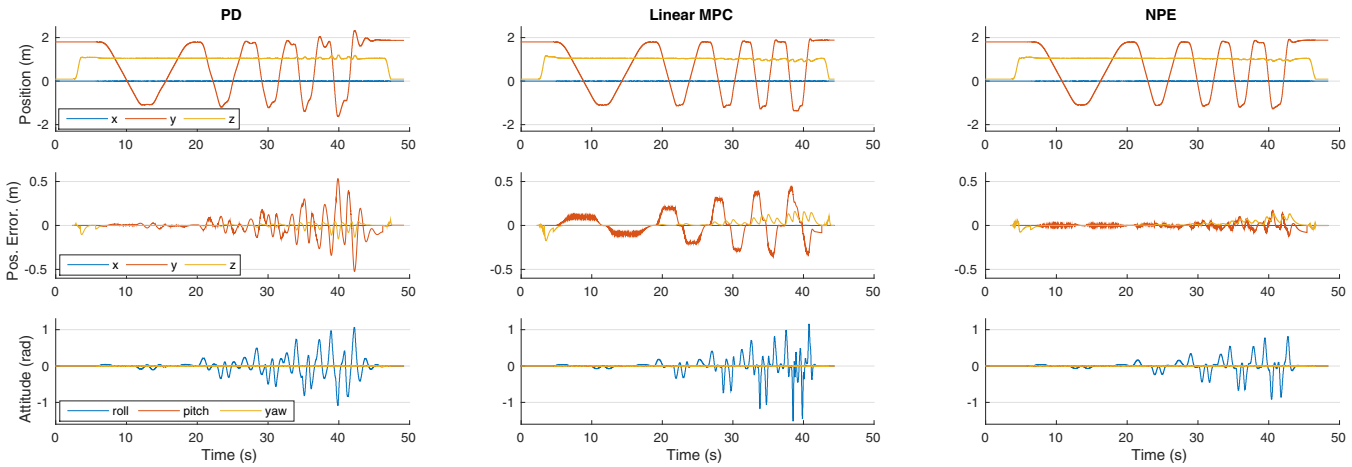


Fig. 2: Comparison of position, trajectory tracking error, and attitude for the three controllers considered (PD, Linear MPC, and NPE). NPE yields substantially improved tracking performance with reduced overshoot and oscillations.

solely using the learned local feedback control laws.

Although this takeoff-hover-land maneuver only activates the constraints on z-velocity and thrust, as shown in Fig. 4, NPE computed 36 different controllers corresponding to combinations of these constraints over the prediction horizon. More diverse maneuvers will activate far more constraints, especially due to the coupling in the nonlinear dynamics, further emphasizing the need for a bounded set of candidate controllers. Figure 4 also shows that NPE largely satisfies these constraints. The minimal violations observed are due to unmodeled dynamics (such as the motor time constant) resulting small prediction errors. However, this effect is independent of the NPE formulation and further illustrates the importance of model fidelity in predictive control.

These learned controllers can be reused in subsequent trials, enabling the NPE-controlled system to leverage previous computation to operate more efficiently. Figure 5 shows the controller usage for another set of takeoff and landing sequences where NPE is initialized with the set of controllers learned in the previous trials. As expected, the system immediately leverages the previously computed controllers, and as a result, the intermediate controller is never applied. Figure 6 shows an overlay of the transitions between controllers for these takeoff and landing sequences, illustrating a consistent behavior in terms of controller switching. The slight variations in the switches can be attributed to the overlap between the valid regions for the learned controllers. This is an effect of the nonlinear dynamics model and KKT tolerance criteria, and removing these overlaps and any redundant controllers is an avenue for future investigation.

One of the key performance metrics for NPE is solution speed. The low computational cost of NPE is illustrated in Table I, which provides statistics on the compute times per component of NPE (controller query, intermediate controller, NLP, local QP, and adding a new controller) for the scenario shown in Fig. 3. The controller query and intermediate controller easily achieve the requisite 200 Hz, while the more computationally expensive components are run in parallel with decreasing frequency. The first row of the table also shows that the NLP is only solved 36 times, which is in stark contrast to standard NMPC solutions that would require solving the NLP in each of the 20807 control iterations. This demonstrates that NPE is an effective real-time model predictive control methodology for nonlinear dynamic systems, such as a quadrotors, where linearity assumptions are degraded during aggressive motions.

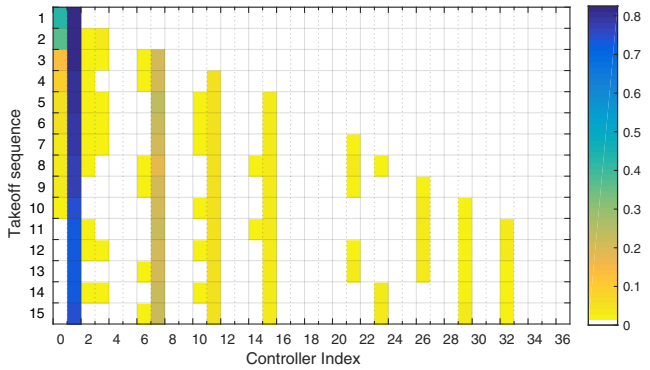
V. Conclusions and Future Work

In this work, we have presented Nonlinear Partial Enumeration (NPE) as a fast solution strategy for nonlinear model predictive control (NMPC). NPE extends the linear PE algorithm to use a nonlinear model for more accurate motion prediction and, with minimal online optimization, produces a piecewise-affine controller covering the relevant regions of the state-space. Through a set of simulation studies focused on aggressive trajectory control for a quadrotor micro aerial vehicle, we have demonstrated that NPE outperforms other fast control methodologies and enables reuse of learned controllers in subsequent flights, thereby reducing the overhead associated with re-solving the optimizations online.

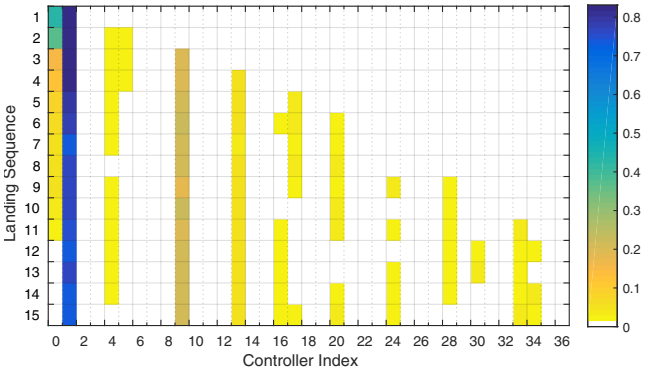
Finally, while the preliminary results we have presented demonstrate the core functionality of NPE, the natural next step is to evaluate performance through experimental trials in challenging, real-world environments to demonstrate its utility as an intelligent, high-rate, feedback control strategy that enables accurate and reliable operation. We will also investigate strategies for mitigating the effects of unmodeled dynamics on system performance via extensions to the NPE formulation.

TABLE I: Solution times for NPE, including the number of control iterations over which the statistics are computed.

	Query	Interm.	NLP	Local QP	Add Element
Iterations	20807	1061	36	36	36
Mean (ms)	1.107	0.923	1412.6	6.232	1.404
Std. Dev. (ms)	0.678	0.781	1063.5	1.310	0.827



(a) Controller usage over multiple takeoff sequences



(b) Controller usage over multiple landing sequences

Fig. 3: Total time a controller is applied (in seconds, indicated by color) during a sequence of similar actions. The first column (index 0) corresponds to the intermediate controller, while index 1 corresponds to the first computed controller.

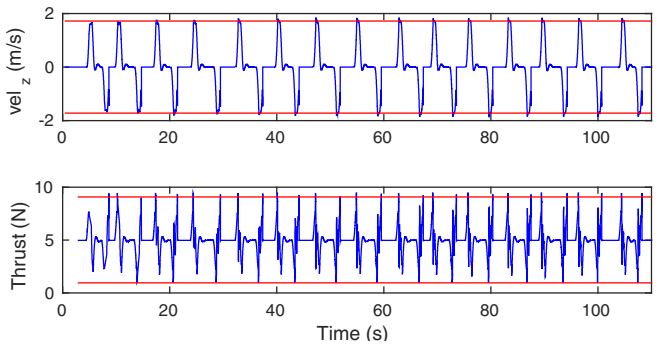
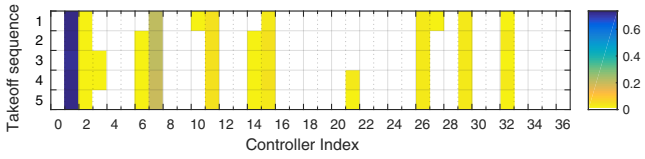
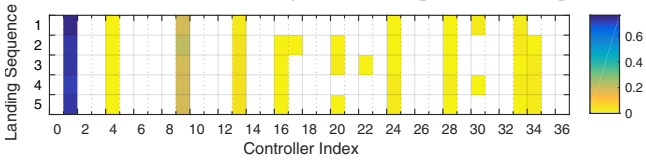


Fig. 4: Vehicle velocity (along the world z-axis) and commanded thrust over repeated takeoff-hover-land sequences. Red lines indicate constraints enforced in NPE.

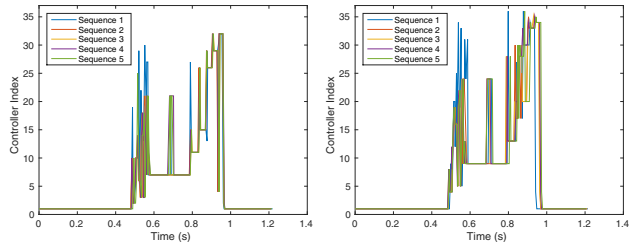


(a) Learned controller usage over multiple takeoff sequences



(b) Learned controller usage over multiple landing sequences

Fig. 5: Total controller application time for a sequence of actions using previously computed controllers. The first column shows the intermediate controller is never used.



(a) (b)

Fig. 6: Overlay of controller switches illustrating similarity across trials.

References

[1] H. J. Ferreau, H. G. Bock, and M. Diehl, "An online active set strategy to overcome the limitations of explicit MPC," *International Journal of Robust and Nonlinear Control*, vol. 18, no. 8, pp. 816–830, May 2008.

[2] Y. Wang and S. Boyd, "Fast Model Predictive Control Using Online Optimization," *IEEE Trans. Control Syst. Technol.*, vol. 18, no. 2, pp. 267–278, Mar. 2010.

[3] A. Bemporad, V. D. M. Morari, and E. N. Pistikopoulos, "The explicit linear quadratic regulator for constrained systems," *Automatica*, vol. 38, pp. 3–20, 2002.

[4] A. Domahidi, M. N. Zeilinger, M. Morari, and C. N. Jones, "Learning a Feasible and Stabilizing Explicit Model Predictive Control Law by Robust Optimization," in *Proc. of the IEEE Conf. on Decision and Control*. IEEE, Dec. 2011, pp. 513–519.

[5] G. Pannocchia, J. B. Rawlings, and S. J. Wright, "Fast, large-scale model predictive control by partial enumeration," *Automatica*, vol. 43, no. 5, pp. 852–860, 2007.

[6] D. H. Jacobson and D. Q. Mayne, *Differential dynamic programming*, ser. Mod. Analytic Comput. Methods Sci. Math. North-Holland, 1970.

[7] R. Tedrake, "LQR-Trees: Feedback motion planning on sparse randomized trees," in *Proc. of Robot.: Sci. and Syst.* Seattle, USA: IEEE, June 2009.

[8] W. Li and E. Todorov, "Iterative linear quadratic regulator design for nonlinear biological movement systems," in *Intl. Conf. on Informatics in Control, Autom. and Robot.*, Setubal, Portugal, June 2004, pp. 222–229.

[9] A. Grancharova and T. Johansen, *Explicit Nonlinear Model Predictive Control*, 2012, vol. 429.

[10] F. Debrouwere, M. Vukov, R. Quirynen, M. Diehl, and J. Swevers, "Experimental validation of combined nonlinear optimal control and estimation of an overhead crane," in *Proc. of the Intl. Fed. of Autom. Control*, Cape Town, South Africa, Aug. 2014, pp. 9617–9622.

[11] N. Michael, D. Mellinger, Q. Lindsey, and V. Kumar, "Experimental evaluation of multirobot aerial control algorithms," *IEEE Robotics & Automation Magazine*, Sept. 2010.

[12] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, no. 3.2, 2009, p. 5.

[13] S. G. Johnson, "The NLopt nonlinear-optimization package," 2014.

[14] H. Ferreau, C. Kirches, A. Potschka, H. Bock, and M. Diehl, "qpOASES: A parametric active-set algorithm for quadratic programming," *Mathematical Programming Computation*, vol. 6, no. 4, pp. 327–363, 2014.