Available online at www.sciencedirect.com

ScienceDirect

IFAC Papers Online CONFERENCE PAPER ARCHIVE

IFAC PapersOnLine 51-20 (2018) 343–348

Fast NMPC with Mixed-Integer Controls Using Quasi-Translations ⋆

Anson Maitland *,1 **John McPhee** *

* *Department of Systems Design Engineering, University of Waterloo, Waterloo, Canada*

Abstract: We present a computationally simple strategy to solve model predictive control problems with mixed-integer controls. We restrict the set of possible discrete control sequences considered at each time step by using parameterized quasi-translations, which relate discrete controls over consecutive timesteps. The parameters can be tuned to balance controller performance, chatter and turnaround time. Using quasi-translations we design two NMPCs for autonomous vehicle control with discrete gear input and exclusively acting pedal inputs. We compare these to an NMPC using a representative 'out-of-the-box' MINLP solver. The quasi-translation controllers run orders of magnitude faster with minimal degradation in tracking performance. This is an exciting development for integer controls that begs future work.

Keywords: Mixed-Integer control, Predictive control, Nonlinear control, Discontinuous control, Autonomous vehicles

1. INTRODUCTION

Discrete inputs, ranging from simple on/off buttons to complex commands expressed in a natural language, are ubiquitous among systems in our world. These inputs share the common feature that they belong to countable sets, *i.e.* there is a bijective mapping between them and a subset of the integers, and thus they are commonly deemed "integer controls". Being able to precisely control such systems presents a challenge since the lack of smooth inputs contradicts the underlying assumption of continuity present in optimal control theory.

A typical Nonlinear Model Predictive Control (NMPC) problem is formulated by the direct transcription of a continuous time optimal control problem (Rao (2009); Biegler (2007); Kameswaran and Biegler (2006)). The receding horizon principle of NMPC generates a feedback control strategy in which a parameterized, finite dimensional optimization problem is solved repeatedly over sequential timesteps. The finite problem of a NMPC (which approximates the original over a finite horizon), often takes the following form

$$\begin{aligned} \min_{\mathbf{X},\mathbf{U}} \left\{ \mathcal{J}(\mathbf{X},\mathbf{U}) = \sum_{k=1}^{H} \ell_k(\mathbf{X},\mathbf{U}) \right\} \\ \text{subject to} \quad \mathbf{X} \in \mathcal{X}(\mathbf{U};\mathbf{x}_0) \\ \mathbf{U} \in \mathcal{U} \end{aligned} \tag{1}$$

where $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_H]$ and $\mathbf{U} = [\mathbf{u}_0, \ldots, \mathbf{u}_{H-1}]$ are the states and controls over the horizon, respectively, H is the length of the prediction horizon, $\mathbf{x} \in \mathbb{R}^n$ and $\mathbf{u} \in \mathbb{R}^m$ are the plant state and control inputs, respectively, ℓ_k are the stage costs, $\mathcal{X} \subset \mathbb{R}^{nH}$ captures the state constraints including the model dynamics given by some discrete time model $\mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k)$ with initial state $\mathbf{x}_0$, and $\mathcal{U} \subset \mathbb{R}^{mH}$ captures the control constraints.

Problem (1) has mixed-integer controls when a component of the control input $\mathbf{u}$ belongs to a countable set. When this is the case we can reformulate the controls such that $\mathcal{U}$ admits a decomposition of the form $\mathcal{U} = \mathcal{U}_c \times \mathcal{U}_d$ where $\mathcal{U}_c \subset \mathbb{R}^{(m-d)H}$ and $\mathcal{U}_d \subset \mathbb{Z}^{dH}$ with $d \geq 1$. Furthermore, some constraints on continuous controls can be re-posed to include a discrete control, *e.g.* the constraint that the throttle and brake input of a vehicle act exclusively, see Section 4.

A naive approach to solving Problem (1) is to fix the integer controls for every possible option, solve the resulting NLPs and select the one with lowest cost. However, the resulting computational effort grows exponentially with the horizon length and is thus prohibitive for even moderately sized problems. More commonly the approach used in NMPC applications is to formulate the finite horizon optimization problem as a Mixed-Integer Nonlinear Program (MINLP). This approach is straightforward but doesn't resolve the underlying difficulty as it simply passes the complexity begat by integer controls to the MINLP solver. MINLPs are known to be NP-Hard (in fact, MILPs and NLPs are already NP-Hard) and can even be undecidable (Garey and Johnson (2002); Murty and Kabadi (1987); Jeroslow (1973)). Due to this difficulty, numerous MINLP solution methods exist, see Grossmann (2002); Schlüter et al. (2012). These solvers typically operate by reformulating and decomposing the MINLP into simpler subproblems like MILPs or NLPs. For example, techniques such as Branch and Bound (BB) and Cutting Plane methods perform a tree search of the integer controls space in order to decrease the computational cost (Borchers and Mitchell (1994); Nowak (2006); Gerdts (2005)). However these solvers can suffer from an exponential growth in solution complexity if all branches of the tree need to be

⋆ This work was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) and Maplesoft™.

[1] corresponding author, email: anson.maitland@uwaterloo.ca

10.1016/j.ifacol.2018.11.057

visited. Further, they often rely on heuristics which greatly effects the success of the method.

We introduce a new approach tailored for Mixed-Integer NMPC applications that has bounded complexity using Quasi-Translations (QTs). We apply this approach successfully to an autonomous vehicle control problem that has two discrete inputs: a binary pedal input, to ensure the exclusive action of the throttle or brake, and a gear selection input. Previous work on mixed-integer optimal control of vehicles has only considered the gear selection as an integer control, *e.g.* Hellström et al. (2009) developed a cruise-controller with discrete gear input using dynamic programming combined with preprocessing, Terwen et al. (2004) developed a predictive gear selection and cruise controller for a heavy truck using a BB method and Kirches et al. (2010), Gerdts (2006) and Gerdts (2005) consider the optimal control of a vehicle with discrete gear input using a convexification and relaxation method, a variable time transformation approach, and a BB method, respectively.

The paper is organized as follows: in Sections 2 and 3 we introduce QTs and a greedy search method for online optimization and in Section 4 we demonstrate our new strategy on a sample autonomous vehicle problem and compare it with a representative MINLP solver.

2. INTRODUCING QUASI-TRANSLATIONS FOR MIXED-INTEGER NMPC

The development of QTs for Mixed-Integer NMPC is guided by two assumptions:

(1) our future predictions are *almost* perfect, and
(2) the present is inevitable.

We translate these assumptions into constraints on integer controls via the following argument. Suppose we wish to control a system with a single binary input using NMPC. At any given timestep our NMPC returns an optimal binary input sequence of length H (the size of the prediction horizon). If our future predictions were perfect then exactly that sequence of inputs will be implemented over the next H steps. In particular, any switches in control input will be realized as predicted H timesteps previously. So from one timestep to the next we would see any switches predicted in the horizon advance one step closer to being realized. Since our future predictions are not exactly perfect, the timing of the switches in the solution from one timestep to the next should not differ by more than some upper bound. Our second assumption tells us that new switches in the control sequence can only be introduced at the end of the horizon.

The effect of these assumptions is twofold. First, we can dramatically reduce the set of integer control sequences under consideration and thus bound the computational cost of the NMPC. And second, we can control the chatter of the integer controls to restrict undesirable behaviour. Loosely speaking, a QT of an integer control sequence is any other integer control sequence that shares the same approximate switch timings and for which any switches not present in the original are near the end of the sequence.

During the online operation of the MPC the set of QTs of the integer controls of the previous timestep are used as candidates. The task of choosing an initial integer control sequence is not addressed here. One possibility is to utilize precomputed MINLP solutions for a variety of realistic initial conditions as potential warm-start values.

2.1 Background

For purposes of exposition we consider purely binary controls in the following. This is not as restrictive as it may first seem since all integer controls can be rewritten using a combination of binary ones (Kirches (2011)).

Let us denote a binary sequence by $\mathbf{b} = [b_1, \dots, b_H] \in \{0,1\}^H$. A *switch* occurs when consecutive elements differ. The number of switches s a binary sequence has is $s(\mathbf{b}) = \sum_{k=1}^{H-1} |b_{k+1} - b_k|$. A consecutive subsequence of elements denoted by $\mathbf{b}_{[i,i+N]} = [b_i, b_{i+1}, \dots, b_{i+N}]$ such that

$$[s(\mathbf{b}_{[i,i+N]}) = 0] \wedge [i > 1 \Rightarrow s(\mathbf{b}_{[i-1,i+N]}) = 1] \wedge [i + N < H \Rightarrow s(\mathbf{b}_{[i-1,i+N+1]}) = 1]$$

is called a *block*. When both $i > 1$ and $i+N < H$ the block is called an *internal block*. We call b_i the *front*, b_{i+N} the *end* and $N+1$ the *length* of the block. Thus it is clear to see that the number of blocks in a sequence $\mathbf{b}$ is $s(\mathbf{b}) + 1$ and the number of internal blocks is $\max\{s(\mathbf{b}) - 1, 0\}$. We denote the set of block fronts by $F(\mathbf{b}) = \{i \in \{2, \dots, H\} | b_i \neq b_{i-1}\}$ and the set of block ends by $E(\mathbf{b}) = \{i \in \{1, \dots, H-1\} | b_i \neq b_{i+1}\}$. So $s(\mathbf{b}) = |F(\mathbf{b})| = |E(\mathbf{b})|$. Further, the length of the smallest internal block is given by

$$l(\mathbf{b}) = \begin{cases} 0 & \text{if } |E(\mathbf{b})| \leq 1 \\ \min\{e_{i+1} - e_i | e_i \in E(\mathbf{b})\} & \text{otherwise} \end{cases}.$$

As an example, consider $\mathbf{b} = [1, 1, 0, 0, 1, 1, 1]$ then

$$s(\mathbf{b}) = 2, E(\mathbf{b}) = \{2, 4\}, F(\mathbf{b}) = \{3, 5\}, l(\mathbf{b}) = 2.$$

The above definitions can be straightforwardly generalized to arbitrary integer sequences.

To control the switching frequency of our controller we consider only those sequences with a maximum number of switches $0 \leq \overline{s} \leq H - 1$ and a minimum internal block length $1 \leq \underline{l} \leq H - 2$. Using these parameters we can define the set

$$\mathcal{B}(\overline{s}, \underline{l}) = \{\mathbf{b} \in \{0,1\}^H | [s(\mathbf{b}) \leq \overline{s}] \wedge [l(\mathbf{b}) \geq \underline{l}]\}.$$

Next we introduce two types of QTs. There are potentially many others to be explored but we limit our discussion to the following.

2.2 Crab-Walk Quasi-Translations

In this strategy the set of allowable sequences at a given timestep are constrained to be stiff translations of the previous timestep. Thus the length of the internal blocks are fixed and hence the blocks move along the prediction horizon over multiple timesteps akin to a crab walking.

We define the set of Crab-Walk QTs (CWQTs) of a sequence $\mathbf{b}$ by

$$\mathcal{C}(\mathbf{b}; \overline{s}, \underline{l}, \overline{r}) = \mathbf{b} \cup \left(\mathcal{C}^f(\mathbf{b}; \overline{r}) \cup \mathcal{C}^b(\mathbf{b}; \overline{r})\right) \cap \mathcal{B}(\overline{s}, \underline{l})$$

where $\overline{r} \geq 1$ is an upper limit on the number of steps taken,

$$\mathcal{C}^f(\mathbf{b}; \overline{r}) = \{\mathbf{b}' \in \{0,1\}^H | [b'_{[1,H-r]} = b_{[r+1,H]}] \wedge [r \leq \overline{r}]\}$$

is the set of allowed forward translations and

$$\mathcal{C}^b(\mathbf{b};\overline{r}) = \{\mathbf{b}' \in \{0,1\}^H | [b'_1 = \ldots = b'_r = b_1] \wedge [b'_{[r+1,H]} = b_{[1,H-r]}] \wedge [r \leq \overline{r}]\}$$

is the set of allowed backward translations. Note that in $\mathcal{C}^b$ we restrict elements at the beginning of the sequence so no new switches occur, unlike $\mathcal{C}^f$ where we allow new elements in the tail to be arbitrary. *e.g.* if $\mathbf{b} = [1,1,0,0,1,1,1]$ then

$$\begin{aligned}\mathcal{C}(\mathbf{b};2,1,2) = \{&[0,0,1,1,1,0,0],\\ &[0,0,1,1,1,1,0],\\ &[0,0,1,1,1,1,1],\\ &[1,0,0,1,1,1,1],\\ &[\mathbf{1},\mathbf{1},\mathbf{0},\mathbf{0},\mathbf{1},\mathbf{1},\mathbf{1}],\\ &[1,1,1,0,0,1,1],\\ &[1,1,1,1,0,0,1]\}.\end{aligned}$$

The parameters $\overline{s}, \underline{l}, \overline{r}$ allow us to limit the size of $\mathcal{C}$. By a simple counting argument we can provide an upper bound that is independent of H

$$|\mathcal{C}(\mathbf{b};\overline{s},\underline{l},\overline{r})| \leq |\mathcal{C}(\mathbf{b};H-1,1,\overline{r})| \leq 2^{\overline{r}+1} + \overline{r} - 1.$$

2.3 Inchworm Quasi-Translations

In this strategy a subset of the ends or fronts of blocks have their values swapped so that internal blocks are allowed to grow and shrink. In this way the internal blocks can move along the horizon over multiple timesteps like an inchworm.

The set of Inchworm QTs (IQTs) of a sequence $\mathbf{b}$ is defined as

$$\mathcal{I}(\mathbf{b};\overline{s},\underline{l},\overline{p}) = \mathbf{b} \cup \left(\mathcal{I}^f(\mathbf{b};\overline{p}) \cup \mathcal{I}^b(\mathbf{b};\overline{p})\right) \cap \mathcal{B}(\overline{s},\underline{l})$$

where $\overline{p} \geq 1$ is an upper limit on the number of elements that can be swapped,

$$\mathcal{I}^f(\mathbf{b};\overline{p}) = \{\mathbf{b}' \in \{0,1\}^H | [b'_i = b_{i+1}, i \in D] \wedge [b'_i = b_i, i \notin D \cup \{H\}], D \subseteq E(\mathbf{b}), 0 \leq |D| \leq \overline{p}\} \setminus \mathbf{b}$$

is the set of allowed forward translations and

$$\mathcal{I}^b(\mathbf{b};\overline{p}) = \{\mathbf{b}' \in \{0,1\}^H | [b'_i = b_{i-1}, i \in D] \wedge [b'_i = b_i, i \notin D], D \subseteq F(\mathbf{b}), 1 \leq |D| \leq \overline{p}\}$$

is the set of allowed backwards translations. Similar to the crab-walk strategy only the forward set allows new switches to be introduced and so switches must propagate over timesteps from the tail to the beginning of the sequence.*e.g.* if $\mathbf{b} = [1,1,0,0,1,1,1]$ then

$$\begin{aligned}\mathcal{I}(\mathbf{b};2,1,2) = \{&[1,0,0,0,1,1,1],\\ &[1,0,0,1,1,1,1],\\ &[1,1,0,1,1,1,1],\\ &[\mathbf{1},\mathbf{1},\mathbf{0},\mathbf{0},\mathbf{1},\mathbf{1},\mathbf{1}],\\ &[1,1,0,0,0,1,1],\\ &[1,1,1,0,0,1,1],\\ &[1,1,1,0,1,1,1]\}.\end{aligned}$$

Similar to the crab-walk strategy we can provide an upper bound on $\mathcal{I}$ which is independent of H

$$|\mathcal{I}(\mathbf{b};\overline{s},\underline{l},\overline{p})| \leq |\mathcal{I}(\mathbf{b};\overline{s},1,\overline{s})| \leq 3 \times 2^{\overline{s}} - 2.$$

Both strategies realize our guiding assumptions by careful selection of forward and backward sets. The parameters allow us to control the size of the set of possibilities and to control the chatter frequency by limiting the rate of translation and time between switches.

3. GREEDY QUASI-TRANSLATION OPTIMIZATION

Using QT strategies we can restrict the potential integer control sequences to a much smaller set of possibilities. However, a full enumeration approach of this smaller set may still be prohibitive since the size of the set of QTs scales exponentially with the parameters. Following the intuition of our first assumption further, we present a greedy search algorithm to find an approximate optimal QT. The greedy aspect of the search method results in some candidate solutions being missed. This algorithm scales logarithmically with the size of the set of potential QTs which drastically cuts down on the computation time. Further, if there are multiple integer controls, these search algorithms can be interleaved so that the additional cost is only additive.

In the following we present an algorithm for each of the QTs introduced. Both of these utilize a greedy scheme with a recursive sub-routine that simultaneously optimizes the discrete and continuous inputs. The search algorithms proceed by looking for decreases in the objective function by making sequential single-element QTs beginning at the first element and then progressing to the end of the horizon. Once an improvement is found, the algorithms proceed within the forward or backward set in which the improvement was first found. The greedy CWQT algorithm acts by sliding internal blocks forward or backwards and the greedy IQT algorithm acts by swapping the ends or fronts of blocks.

We denote the NLP routine that solves Problem (1) with fixed integer inputs $\mathbf{b}$ and returns the objective function value by $\Gamma(\mathbf{b})$. We define $\Gamma(\emptyset) = \infty$. Evaluating Γ is by far the most computationally expensive sub-routine as it solves an NLP problem and thus its calls should be minimized. Algorithm 1 finds an approximate CWQT optimum with worst-case performance $(2\overline{r}+1) \times O(\Gamma)$ and Algorithm 2 finds an approximate IQT optimum and has worst-case performance $(2\overline{s}+1) \times O(\Gamma)$.

4. AUTONOMOUS VEHICLE EXAMPLE

We demonstrate the use of QTs for NMPC in the control of a 7 state, 4 input nonlinear vehicle model. The control vector is $\mathbf{u} = (w_\delta \ \theta \ b \ \mu)^T$ whose components are steering angle rate $w_\delta \in [-0.5, 0.5]$ (rad/s), normalized pedal value $\theta \in [0,1]$, pedal selection $b \in \{0,1\}$ (0 for brake, 1 for throttle) and gear selection $\mu \in \{1,2,3,4,5\}$. The model states of interest are the absolute position (x,y) (m), speed v (m/s) and yaw angle ψ (rad). The full state vector is $\mathbf{x} = (x, y, v, \beta, \psi, w_z, \delta)^T$ where the unmentioned states are side slip angle β, rate of change of yaw w_z and steering angle δ. The vehicle model is a bicycle model with Pacejka tires (Pacejka and Bakker (1992)) and contains a rudimentary mapping to relate the throttle ϕ and gear inputs to wheel torque. More model details including the governing equations and parameters can be found in Gerdts (2005); Kirches et al. (2010). The referenced model takes brake force $F_B \in [0, 15000]$ (N) and throttle $\phi \in [0,1]$

Algorithm 1 Greedy Search for Optimal $\mathbf{b}' \in \mathcal{C}(\mathbf{b}; \overline{s}, \underline{l}, \overline{r})$

function GREEDYCRABWALK($\mathbf{b}$)
  $C \leftarrow \Gamma(\mathbf{b})$
  return STEP($\mathbf{b}, C,$ `forward`$, 0$)
end function

function STEP($\mathbf{b}, C,$ `direction`, `count`)
  if `count` $\geq \overline{r}$ **then**
    return b
  end if
  if `direction` is `forward` **then**
    $\mathbf{b}' \leftarrow \{[b_{[2,H]}, b_H]\} \cap \mathcal{B}(\overline{s}, \underline{l})$
    $\mathbf{b}'' \leftarrow \{[b_{[2,H]}, \neg b_H]\} \cap \mathcal{B}(\overline{s}, \underline{l})$
  else
    $\mathbf{b}' \leftarrow \{[b_1, b_{[1,H-1]}]\} \cap \mathcal{B}(\overline{s}, \underline{l})$
    $\mathbf{b}'' \leftarrow \emptyset$
  end if
  $C' \leftarrow \Gamma(\mathbf{b}')$
  $C'' \leftarrow \Gamma(\mathbf{b}'')$
  if $\min\{C, C', C''\} = C$ **then**
    if `count` $= 0$ **then**
      return STEP($\mathbf{b}, C,$ `backward`, `count`)
    else
      return b
    end if
  else if $\min\{C, C', C''\} = C'$ **then**
    return STEP($\mathbf{b}', C',$ `direction`, `count` $+ 1$)
  else
    return STEP($\mathbf{b}'', C'',$ `direction`, `count` $+ 1$)
  end if
end function

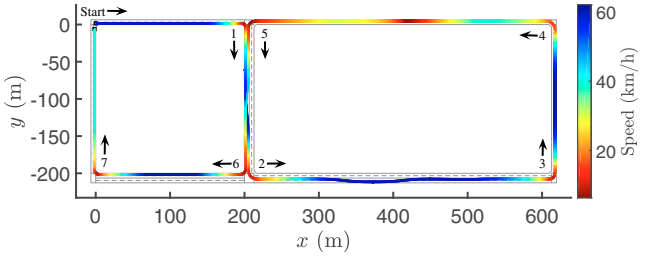

Fig. 1. The reference path follows a figure eight that begins and ends near $(0, 0)$. The corners are labelled in the order they are encountered. Between corners 1 & 2 there is a lane change to the left, between corners 2 & 3 there is a double lane change, between corners 4 & 5 there is a slow down maneuver and between corners 5 & 6 there is a lane change to the right.

as inputs, which we find from the following mappings of pedal value and selection $F_B = 15000(1 - b)\theta$ and $\phi = b\theta$.

The driving scenario is set up to mimic moderately aggressive urban driving and is 238.5s (≈ 4 min) in duration. The reference path and speed is plotted in Fig. 1.

The goal of the controller is to follow a reference trajectory while minimizing the inputs and their rates of action. We formulate the problem using a control parameterization (or a single-shooting approach) that eliminates the states as degrees of freedom in the resulting optimization problem. The cost function is given by

Algorithm 2 Greedy Search for Optimal $\mathbf{b}' \in \mathcal{I}(\mathbf{b}; \overline{s}, \underline{l}, \overline{p})$

function GREEDYINCHWORM($\mathbf{b}$)
  $E \leftarrow E(\mathbf{b}) \cup \{H\}$
  $F \leftarrow F(\mathbf{b})$
  $C \leftarrow \Gamma(\mathbf{b})$
  return INCH($\mathbf{b}, C, E, F, 0$)
end function

function INCH($\mathbf{b}, C, E, F,$ `count`)
  if $E = \{H\}$ and `count` $\geq \overline{p}$ **then**
    `count` $\leftarrow$ `count` $- 1$
  end if
  if $E \cup F = \emptyset$ or `count` $\geq \overline{p}$ **then**
    return b
  end if
  $m \leftarrow \min\{E \cup F\}$
  if $m = H$ **then**
    $\mathbf{b}' \leftarrow \{[b_{[1,H-1]}, \neg b_H]\} \cap \mathcal{B}(\overline{s}, \underline{l})$
  else if $m \in E$ **then**
    $\mathbf{b}' \leftarrow \{[b_{[1,m-1]}, b_{m+1}, b_{[m+1,H]}]\} \cap \mathcal{B}(\overline{s}, \underline{l})$
  else
    $\mathbf{b}' \leftarrow \{[b_{[1,m-1]}, b_{m-1}, b_{[m+1,H]}]\} \cap \mathcal{B}(\overline{s}, \underline{l})$
  end if
  $C' \leftarrow \Gamma(\mathbf{b}')$
  if $C \geq C'$ and $m \in E$ **then**
    return INCH($\mathbf{b}', C', E \setminus \{m\}, \emptyset,$ `count` $+ 1$)
  else if $C \geq C'$ and $m \in F$ **then**
    return INCH($\mathbf{b}', C', \emptyset, F \setminus \{m\},$ `count` $+ 1$)
  else if $C < C'$ and ($E = \emptyset$ or $F = \emptyset$) **then**
    return b
  else
    return INCH($\mathbf{b}, C, E \setminus \{m\}, F \setminus \{m\},$ `count`)
  end if
end function

$$\mathcal{J}(\mathbf{U}) = \sum_{k=1}^{H} \|\mathbf{x}_k - \mathbf{x}_k^r\|_{\mathbf{Q}}^2 + \|\mathbf{v}_{k-1}\|_{\mathbf{R}}^2 + \|\Delta\mathbf{v}_k\|_{\mathbf{R}'}^2 \qquad (2)$$

where $\mathbf{U} = [\mathbf{u}_0, \mathbf{u}_1, \ldots, \mathbf{u}_{H-1}]$ are the control inputs over the horizon, $H = 10$ is the number of timesteps in the horizon, $\mathbf{x}$ is the state of the system, $\mathbf{x}^r$ is the reference trajectory, $\mathbf{v} = (w_\delta \;\; F_B \;\; \phi)^T$ are the continuous inputs we wish to minimize and $\Delta\mathbf{v}_k = \mathbf{v}_k - \mathbf{v}_{k-1}$ (with $\Delta\mathbf{v}_H = 0$) are the input rates. The weighting matrices are $\mathbf{Q} = \text{diag}\left(10^4 \;\; 10^4 \;\; 2 \times 10^4 \;\; 0 \;\; 0 \;\; 100 \;\; 0\right)$, $\mathbf{R} = \text{diag}\left(500 \;\; 0 \;\; 200\right)$ and $\mathbf{R}' = \text{diag}\left(200 \;\; 10^4 \;\; 10^4\right)$. The states over the horizon are defined recursively using the explicit Euler method $\mathbf{x}_{k+1} = \mathbf{x}_k + \Delta t f(\mathbf{x}_k, \mathbf{u}_k)$ where $\Delta t = 0.05$s is the fixed timestep and f is the model's vector field.

In our fast NLP method we enforced the control constraints by including a penalty function of the form

$$\Theta(\mathbf{U}) = \sum_{k=1}^{H} \|\rho_p(\mathbf{u}_{k-1})\|_{\mathbf{P}}^2$$

where the half-penalty function ρ_p, for $p \in \mathbb{N}$, is given by

$$\rho_p(z) = \left(\frac{2z - (z^{\max} + z^{\min})}{z^{\max} - z^{\min}}\right)^p$$

which enforces constraints of the form $z \in [z^{\min}, z^{\max}]$ in the limit $p \to \infty$. We evaluate ρ_p entry-wise for vector-valued arguments. We set $p = 4$ and $\mathbf{P} = \text{diag}\left(10 \;\; 100 \;\; 0 \;\; 0\right)$ in our simulations. The resulting online NLP is an uncon-

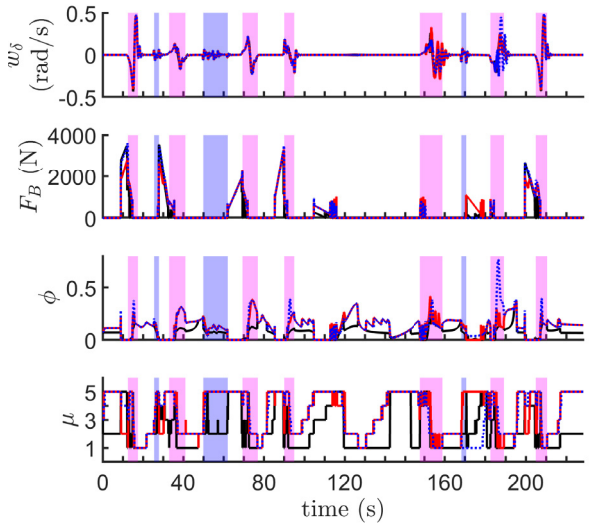

Fig. 2. Control inputs for the urban driving scenario. The solid black, solid red and dotted blue lines are the BONMIN-BB, CWQT and IQT solutions.The pink and blue patches highlight periods of turning and lane changing, respectively.

strained problem of minimal dimension. Our fast solver takes advantage of this and is simply comprised of a fixed number (5) of Newton steps applied to the objective function $\mathcal{J}(\mathbf{U}) + \Theta(\mathbf{U})$ where we use the solution of the previous timestep as the initial guess for the current step.

We compare our results to the solution found by the Basic Open-source Nonlinear Mixed Integer programming (BONMIN) solver (Bonami et al. (2008)). This solver is chosen as a representative of existing MINLP methods. At each timestep BONMIN was applied to $\mathcal{J}$ with bounds on $\mathbf{u}$ and linear inequalities enforcing sequential gear changes as constraints. We also used a fourth-degree polynomial interpolation of the transmission relation $i_g(\mu), \mu \in [1, 5]$ for the relaxed subproblems. BONMIN was implemented with all default parameters as a NLP based BB algorithm, which we denote BONMIN-BB. Other BONMIN algorithms, such as Outer-Approximation, were tried with default parameters but they all failed at some stage of the simulation with an exitflag indicative of a solver error or unbounded subproblem.

Using Maple, the objective function, its gradient and Hessian were generated and optimized as standalone functions. These functions were then converted to MEX functions that we then used in our fast NLP method and we passed to BONMIN. All simulations were run using MATLAB 2017b on a desktop with an Intel(R) Core(TM) i7-4790 CPU. BONMIN was run on MATLAB using the OPTI Toolbox v2.27 (Currie et al. (2012)).

We select one parameterization of each of the QT strategies to compare with the BONMIN-BB solution. The first is a CWQT with parameters $\overline{s}_b = 1, \overline{r}_b = 4, \overline{s}_\mu = 3, \underline{l}_\mu = 1, \overline{r}_\mu = 3$ and the second is an IQT with parameters $\overline{s}_b = \overline{p}_b = 1, \overline{s}_\mu = \overline{p}_\mu = 3, \underline{l}_\mu = 3$. The subscripts denote which integer control the parameters are determining. In Figures 2 and 3 we show an overview of the simulations over the entire scenario. Firstly, we note the BONMIN-BB solution is remarkably similar in the switching times of the

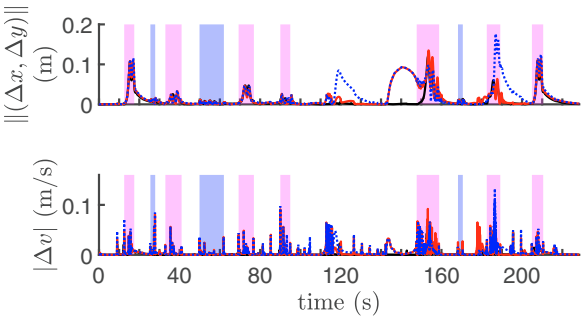

Fig. 3. Position and velocity tracking errors.

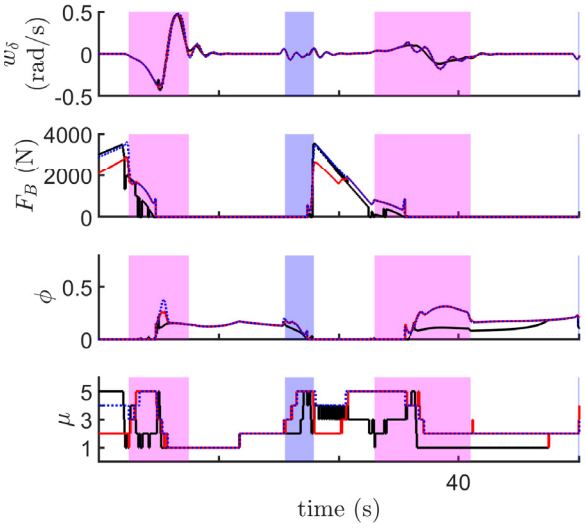

Fig. 4. Control inputs close-up over the 10-50s timespan.

throttle/brake input to the QT strategies. However, the gear selection differs more dramatically between solutions. This is to be expected since the gear selection μ is not controlled explicitly by our choice of objective function but is implicitly influenced by the desire to minimize ϕ. Greatest chatter of the BONMIN-BB solution occurs about the throttle/brake switches. We can see that in all solutions the throttle to brake switches mostly occur going into a corner and the brake to throttle switches mostly occur halfway through a corner, much like a human driver. The position error of all solutions spike during cornering.

In Figure 4 we zoom in on the control inputs over the 10-50s period. From this figure we can more clearly see a number of results. Firstly, the BONMIN-BB solution switches more often than the QT solutions; however it is more efficient as it uses less total throttle input. This is partly due to the controller tuning which placed much greater cost on tracking error than control effort. Around the 120-150s period we see that the BONMIN-BB solution is able to rapidly shift down and up during the straight slow-down maneuver and thus be more efficient in its use of throttle. However, the BONMIN-BB solution at some points switches gears 19 times (the maximum possible) within a one second span; meanwhile the CWQT and IQT solutions switch gears at most 7 and 3 times within a one second span, respectively. The QT strategy reduces the chattering behaviour at the expense of efficiency.

Table 1. NMPC Accuracy

	mean $\|(\Delta x, \Delta y)\|$ (cm)	mean $\|\Delta v\|$ (cm/s)
BONMIN-BB	0.90	0.16
CWQT	1.61	0.49
IQT	2.15	0.57

Table 2. NMPC Computation Times

	Total (s)	Max Turnaround (s)
BONMIN-BB	6098.1 [-]	115.6 [-]
CWQT	18.8 [> 300×]	0.0084 [> 13000×]
IQT	12.1 [> 500×]	0.0057 [> 20000×]

Further, looking at Figure 3 we can see that the BONMIN-BB solution has better tracking accuracy. The BONMIN-BB solution is never more than 11.94 cm from its target position. In comparison, the IQT trajectory (which performs worst) deviates as much as 17.64 cm from target. However, relative to the lane width (3.2 m) this is only a 2% increase in position error. Similarly, BONMIN-BB is at most 2.5% better in terms of relative velocity error. These results are expected since BONMIN-BB relies on the IPOPT solver for its NLP subproblems, which we expect to find better solutions than our fast NLP solver that uses a fixed number of Newton iterations. We summarize these results in Table 1.

In Table 2 we summarize the computation time results along with the relative increase in computing speed (shown in square brackets) for the entire scenario and maximum turnaround time. This is where there is greatest difference in controller behaviour. The QT approaches run orders of magnitude faster than BONMIN-BB and run faster than real-time on our architecture. Most crucially we can see the importance of being able to bound the MINLP solution cost since existing solution methods may require an exponential effort, dramatically increasing turnaround times. Overall, given the relatively small decrease in tracking performance, these timing results are excellent.

5. CONCLUSION

We introduced a new and fast approach to design an NMPC with mixed-integer controls. On a sample autonomous vehicle control scenario, these methods ran orders of magnitude faster than a representative, 'out-of-the-box' MINLP solver with minimal degradation in performance. The QT approach takes advantage of the sequential nature of NMPC to reduce chatter and accelerate turnaround times.

In future we would like to conduct a more thorough comparison to fast MINLP methods for Mixed-Integer NMPC (*e.g.* the methods of Kirches et al. (2010), Bock et al. (2018)) as well as an investigation of the role of the QT parameters on performance. Questions of controller stability and robustness when using this approach remain unexamined.

REFERENCES

Biegler, L.T. (2007). An overview of simultaneous strategies for dynamic optimization. *Chemical Engineering and Processing: Process Intensification*, 46(11), 1043–1053.

Bock, H.G., Kirches, C., Meyer, A., and Potschka, A. (2018). Numerical solution of optimal control problems with explicit and implicit switches. *Optimization Methods and Software*, 33(3), 450–474.

Bonami, P., Biegler, L.T., Conn, A.R., Cornuéjols, G., Grossmann, I.E., Laird, C.D., Lee, J., Lodi, A., Margot, F., Sawaya, N., et al. (2008). An algorithmic framework for convex mixed integer nonlinear programs. *Discrete Optimization*, 5(2), 186–204.

Borchers, B. and Mitchell, J.E. (1994). An improved branch and bound algorithm for mixed integer nonlinear programs. *Computers & Operations Research*, 21(4), 359–367.

Currie, J., Wilson, D.I., et al. (2012). Opti: lowering the barrier between open source optimizers and the industrial matlab user. *Foundations of computer-aided process operations*, 24, 32.

Garey, M.R. and Johnson, D.S. (2002). *Computers and intractability*, volume 29. wh freeman New York.

Gerdts, M. (2005). Solving mixed-integer optimal control problems by branch&bound: a case study from automobile test-driving with gear shift. *Optimal Control Applications and Methods*, 26(1), 1–18.

Gerdts, M. (2006). A variable time transformation method for mixed-integer optimal control problems. *Optimal Control Applications and Methods*, 27(3), 169–182.

Grossmann, I.E. (2002). Review of nonlinear mixed-integer and disjunctive programming techniques. *Optimization and engineering*, 3(3), 227–252.

Hellström, E., Ivarsson, M., Åslund, J., and Nielsen, L. (2009). Look-ahead control for heavy trucks to minimize trip time and fuel consumption. *Control Engineering Practice*, 17(2), 245–254.

Jeroslow, R. (1973). There cannot be any algorithm for integer programming with quadratic constraints. *Operations Research*, 21(1), 221–224.

Kameswaran, S. and Biegler, L.T. (2006). Simultaneous dynamic optimization strategies: Recent advances and challenges. *Computers & Chemical Engineering*, 30(10), 1560–1575.

Kirches, C. (2011). *Fast numerical methods for mixed-integer nonlinear model-predictive control*. Springer.

Kirches, C., Sager, S., Bock, H.G., and Schlöder, J.P. (2010). Time-optimal control of automobile test drives with gear shifts. *Optimal Control Applications and Methods*, 31(2), 137–153.

Murty, K.G. and Kabadi, S.N. (1987). Some np-complete problems in quadratic and nonlinear programming. *Mathematical programming*, 39(2), 117–129.

Nowak, I. (2006). *Relaxation and decomposition methods for mixed integer nonlinear programming*, volume 152. Springer Science & Business Media.

Pacejka, H.B. and Bakker, E. (1992). The magic formula tyre model. *Vehicle System Dynamics*, 21(S1), 1–18.

Rao, A.V. (2009). A survey of numerical methods for optimal control. *Advances in the Astronautical Sciences*, 135(1), 497–528.

Schlüter, M., Gerdts, M., and Rückmann, J.J. (2012). A numerical study of midaco on 100 minlp benchmarks. *Optimization*, 61(7), 873–900.

Terwen, S., Back, M., and Krebs, V. (2004). Predictive powertrain control for heavy duty trucks. *IFAC Proceedings Volumes*, 37(22), 105–110.