Abstract—Inactive constraints do not contribute to the solution but increase the problem size and burden the numerical computations. We present a novel strategy for handling inactive constraints efficiently by systematically removing the inactive constraints and redundant constraint sets under a mesh refinement framework. The method is tailored for interior point-based solvers, which are known to be very sensitive to the choice of initial points in terms of feasibility. In the example problem shown, the proposed scheme achieves more than 40% reduction in computation time.

Index Terms—constrained control, optimal control, nonlinear predictive control

I. INTRODUCTION

Optimal control has been a very popular method for a wide range of applications, thanks to its capability to handle various types of constraints systematically. When formulating the optimal control problem (OCP), it is common practice to impose a large number of equality and inequality constraints to ensure all mission specifications are fulfilled. However, for the solution obtained, it is often the case that only a small subset of all the imposed constraints will actually be active. Furthermore, even for the ones in this small subset, the duration for which each constraint is active is generally much shorter than the time dimension of the OCP.

In numerical optimal control, the OCPs are transcribed into sparse nonlinear programming (NLP) problems and solved thereafter. The main computation overheads for NLP iterations are associated with solving the linear systems formulated based on the optimality conditions. Since the computational complexity is directly related to the dimension of the linear system, and the size of the system is mainly governed by the number of constraints, there exist significant computational benefits to not include the inactive constraints in the problem formulation.

One possibility is to only include the constraints that are determined to be active. Based on this, an external strategy for the implementation of path constraints with active-set based NLP solvers has been proposed [1]. The idea is to first solve the unconstrained problem and determine which constraints are likely to be active based on constraint violations. These constraints are then added back in the OCP and the problem is repetitively solved until all initial sets of constraints are satisfied. However, a fundamental problem arises when trying to implement the same idea on interior point method (IPM) based solvers, since good performance of the IPM solvers hinges on the initial point to be feasible, or at least close to feasible [2].

Another option is to remove constraints that are determined to be inactive. Removal of constraints for model predictive control (MPC) has been studied to accelerate computations for standard linear MPC [3], tube-based robust linear MPC [4] and recently nonlinear MPC [5], with computational benefits clearly demonstrated. However, all of them requires problem reformulation based on the quadratic regulation cost formulation, making their application specifically for receding horizon control of regulation tasks.

In this paper, we demonstrate an external constraint handling strategy that is tailored to IPM-based NLP solvers for solving a wide range of general purpose OCPs. Implemented together with mesh refinement schemes, constraints that do not contribute to the solution are systematically removed in the problem formulation. In this process, special attention has been paid to ensure the feasibility of the initial point. As a result, significant computational savings can potentially be achieved. To assist the discussions, Section III gives a high level introduction to numerical optimal control with direct collocation method. This is followed by a detailed illustration of the proposed external constraint handling strategy in Section IV. A real-world example is presented in Section IV to demonstrate the computational benefits.

II. NUMERICAL OPTIMAL CONTROL

Generally speaking, optimization-based control requires the solution of OCPs expressed in the general Bolza form:

\[
\min_{x,u,p,t_0,t_f} \Phi(x(t_0),x(t_f),t_f,p) + \int_{t_0}^{t_f} L(x(t),u(t),t,p) dt
\]  

subject to

\[
\dot{x}(t) = f(x(t),u(t),t,p), \forall t \in [t_0,t_f] \quad (1b)
\]
\[
c(x(t),u(t),t,p) \leq 0, \forall t \in [t_0,t_f] \quad (1c)
\]
\[
\phi(x(t_0),t_0,x(t_f),t_f,p) = 0, \quad (1d)
\]

with \(x(t) \in \mathbb{R}^n\) the state of the system, \(u(t) \in \mathbb{R}^m\) the control input, \(p \in \mathbb{R}^s\) static parameters, \(t_0 \in \mathbb{R}\) and \(t_f \in \mathbb{R}\) the initial and terminal time. \(\Phi\) is the Mayer cost functional \((\Phi : \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R} \times \mathbb{R}^s \to \mathbb{R})\), \(L\) is the Lagrange cost functional \((L : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R} \times \mathbb{R}^s \to \mathbb{R})\), \(f\) is the dynamic...
constraint \((f : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R} \times \mathbb{R}^s \to \mathbb{R}^n)\), \(c\) is the path constraint \((c : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R} \times \mathbb{R}^s \to \mathbb{R}^{ns})\) and \(\phi\) is the boundary condition.

In practice, most optimal control problems formulated as (1) need to be solved with numerical schemes. Among various options, the direct collocation method has become very popular for solving practical optimal control problems.

### A. Direct collocation methods

Direct collocation methods can be categorized into fixed-order \(h\) methods [6], and variable-order \(p/hp\) methods [7], [8]. Here, we only provide a high level overview. For a mesh of size \(N = \sum_{k=1}^{K} N(k)\), the states can be approximated as

\[
x^{(k)}(\tau) \approx X^{(k)}(\tau) := \sum_{j=1}^{N(k)} A_j^{(k)} B_j^{(k)}(\tau),
\]

with mesh interval \(k \in \{1, \ldots, K\}, N(k)\) denoting the number of collocation points for interval \(k\), and \(B_j^{(k)}(\cdot)\) are basis functions. For typical \(h\) methods, \(\tau \in \mathbb{R}^N\) takes values on the interval \([0, 1]\) representing \([t_0, t_f]\), and \(B_j^{(k)}(\cdot)\) are elementary B-splines of various orders. For \(p/hp\) methods, \(\tau \in [-1, 1]\) and \(B_j^{(k)}(\cdot)\) are Lagrange interpolating polynomials.

We write \(X_j^{(k)}\) as the approximated states at collocation point \(\tau_j^{(k)}\), and the input are approximated analogously with \(U_j^{(k)}\).

Consequently, the OCP (1) can be approximated by

\[
\min_{X, U, p, t_0, t_f} \Phi(X_j^{(1)}, t_0, X_f^{(k)}, t_f, p) + \sum_{k=1}^{K} \sum_{i=1}^{N(k)} w_i^{(k)} L(X_i^{(k)}, t_i^{(k)}, \tau_i^{(k)}, t_0, t_f, p) = 0 \tag{2a}
\]

for \(i = 1, \ldots, N(k)\) and \(k = 1, \ldots, K\), subject to,

\[
\sum_{j=1}^{N(k)} A_{ij}^{(k)} X_j^{(k)} + D_{ij}^{(k)} f(X_i^{(k)}, U_i^{(k)}, \tau_i^{(k)}, t_0, t_f, p) = 0 \tag{2b}
\]

\[
c(X_i^{(k)}, U_i^{(k)}, \tau_i^{(k)}, t_0, t_f, p) \leq 0 \tag{2c}
\]

\[
\phi(X_i^{(k)}, t_0, X_f^{(k)}, t_f, p) = 0 \tag{2d}
\]

where \(w_i^{(k)}\) are the quadrature weights for the respective discretization method chosen, \(A\) is the numerical differentiation matrix with \(A_{ij}\) the element \((i, j)\) of the matrix, and \(D\) a constant matrix. The discretized problem can then be solved with off-the-shelf NLP solvers.

The NLP solver generates a discretized solution \(Z := (X, U, p, \tau, t_0, t_f)\) as sampled data points. Interpolating splines may be used to construct an approximation of the continuous-time optimal trajectory \(\tilde{z}(t) := (\tilde{z}(t), \tilde{u}(t), t, p)\). The quality of the interpolated solution needs to be assured through error analysis, assessing the level of accuracy and constraint satisfaction. When necessary, appropriate modifications must be made to the discretization mesh, until the solutions obtained with the new mesh fulfilling all predefined error tolerance levels (e.g. the absolute local error \(\eta_{tol}\) and the absolute local constraint violation \(\varepsilon_{c, tol}\)). This process is called mesh refinement (MR).

### III. EXTERNAL CONSTRAINT HANDLING WITH INTERIOR POINT METHOD NLP SOLVERS

#### A. Active and inactive constraints

A constraint (1c) is considered active if its presence influences the solution \(z^*(t) := (x^*(t), u^*(t), p^*, t_0^*, t_f^*)\). Vice versa, a constraint is inactive if it can be removed without affecting the solution. To clarify, consider a simplified problem:

\[
y^* \in \arg \min_y \Phi(y) \text{ subject to } c(y) \leq 0,
\]

where we need to identify conditions such that constraints can be determined to be active. The most obvious criteria is when the solution \(y^*\) is at the boundary of \(c_i(y) \leq 0\), i.e. \(c_i(y^*) = 0\). Additionally, express the Lagrangian as \(\mathcal{L} := \Phi(y) + \lambda^T c(y)\) and formulate the necessary optimality conditions (Karush-Kuhn-Tucker (KKT) conditions):

\[
\frac{\partial \Phi(y)}{\partial y} \bigg|_{y=y^*} + \lambda \frac{\partial c_i(y)}{\partial y} \bigg|_{y=y^*} = 0,
\]

\[
c(y^*) \leq 0, \quad \lambda \geq 0, \quad \lambda \circ c(y^*) = 0,
\]

with \(\circ\) the Hadamard product. From the included complementarity slackness condition, we know that for strictly positive Lagrange multipliers \((\lambda_i > 0)\), the corresponding solution will have \(c_i(y^*) = 0\), meaning the constraint is active.

#### B. Identify active constraints in numerical optimal control

In practice, additional challenges arise when solving the discretized problem (2) numerically. The first issue is the availability of the information: the NLP solver will only return the values of the discretized state \(X\), input \(U\), and Lagrange multipliers \(\Lambda\) at collocation points. Secondly, limited machine precision in numerical computations means that even when the corresponding constraints are inactive, the multiplier values are rarely truly equal to zero.

To better estimate the constraint activation status in-between collocation points, a criteria can be introduced based on the interpolated continuous trajectory \(\tilde{z}\). A constraint is considered to be potentially active if the the magnitude differences between the actual constraint \(c_i(\tilde{z}(t))\) and the user-defined constraint bounds are smaller than the specified constraint violation tolerance \(\varepsilon_{c, tol}\).

The multiplier information can then be used together to enforce the selection. To identify the regions where the constraints are likely to be active, the numerical multiplier data \(\Lambda\) are first normalized between 0 and 1 for each constraint \(c_i(Z) \leq 0\). Signal processing algorithms can be used to identify different intervals where the behaviour of Lagrange multipliers have significant changes. Figure 1 demonstrates the use of the MATLAB findchangepts function.

For each identified interval \(T_i\), the mean value of the normalized multipliers \((\bar{\Lambda}_{T_i})\) is calculated and compared based on the following criteria

\[
\begin{cases}
\bar{\Lambda}_{T_i} \geq \zeta & \text{constraint potentially active in interval } T_i \\
\text{otherwise} & \text{constraint potentially inactive in interval } T_i
\end{cases}
\]
with $\zeta$ a threshold parameter.

To sum up, the following definition is specified to determine whether the constraints are potentially active or potentially inactive at different collocation points of a numerical optimal control scheme.

**Definition 1:** A constraint $c_i(x_i, U_i, \tau_i, t_0, t_f, p) \leq 0$ is called potentially active at time instance $t_i := t_i(\tau_i, t_0, t_f)$ if one of the following criteria is met:

- Inbetween adjacent collocation points ($t \in [t_{i-1}, t_{i+1}]$), $c_i(\tilde{x}(t), \tilde{u}(t), t, p) \geq -\epsilon_{\text{atol}}$ holds, with $\epsilon_{\text{atol}} > 0$.
- $\Delta \tau_i \geq \zeta$, with $t_i \in \mathcal{T}_i$.

A constraint is called potentially inactive at time instance $t_i$ if otherwise.

In addition to identifying the time instances at which certain constraints may be potentially inactive, it is also preferable to determine the sets of constraints that never become active at all times.

**Definition 2:** A constraint $c_i(x_i, U_i, \tau_i, t_0, t_f, p) \leq 0$ is called potentially redundant if for all $t_i := t_i(\tau_i, t_0, t_f)$ with $i = 1, \ldots, N$, the constraints $c_i(x_i, U_i, \tau_i, t_0, t_f, p) \leq 0$ are potentially inactive. Otherwise, this set of constraints is potentially enforced.

If the potentially inactive constraints and potentially redundant constraint sets are truly inactive constraints, they can be removed from the OCP formulation without affecting the solutions.

**C. Initialization for interior point method**

The interior point method (IPM) for solving NLPs was introduced in the early 1960s [9]–[11], and has become very popular in numerical optimal control. The fundamental idea is to augment the objective function with barrier functions of constraints in order to enforce their satisfaction. As a result, potential solutions will iterate only in the feasible region following the so-called central path, resulting in a very efficient algorithm.

Standard interior point methods are very sensitive to the choice of a starting point. To ensure the initial guess to be strictly feasible with respect to constraints, various initialization methods were developed (e.g. [12] and the collective study in [6]) and implemented in modern IPM solvers, generally requiring the solution of a sparse linear least squares problem.

To ensure reliable and efficient computation of the initialization algorithm as well as the subsequent NLP iterations, several criteria [2] can be formulated regarding ideal initial guesses with IPM:

- it should satisfy or be close to primal and dual feasibility
- it should be close to the central path
- it should be as close to optimality as possible

Because of these characteristics, external constraint handling schemes developed for active-set based SQP solvers, such as [1], are not suitable for IPM-based solvers. By first solving the unconstrained problem and gradually adding constraints based on the constraint violation error, the solution of previous solves will all be infeasible for the new OCP formulation, and the solution may undergo drastic changes as well. For IPM-based NLP solvers, this would lead to a higher computational overhead for initialization, as well as higher chances for the iterations to frequently enter the slow and unreliable feasibility restoration phase.

**D. Proposed Scheme for Constraint Handling**

Based on the criteria presented in Section III-B and the characteristics of IPM as discussed in Section III-C, a strategy of efficiently handling constraints in OCPs solved with IPM-based NLP solvers is developed, with the workflow presented in Figure 2. The approach is called external, since the modifications to the OCP are made at the mesh refinement iteration level, instead of during the NLP iterations.

The unmodified OCP is first solved on the initial coarse mesh. Even with all constraints equations included, the
computation time will still be quite low at this stage due to the small problem size. Once the solution is obtained, based on Definitions 1 and 2 potentially inactive constraints and potentially redundant constraint sets can be identified. Constraints that are potentially redundant can be excluded from the OCP formulation. Furthermore, if the problem has fixed terminal time, i.e. the time instance corresponding to a mesh point will not change, then potentially inactive constraints in the potentially enforced constraint sets may also be removed.

Here, special attention must be made to constraints and constraint sets that were determined to be potentially inactive or redundant in the previous solves. If they never become active or enforced again, the constraint removal process may just continuous until the mesh refinement process is converged. However, there will be the chance that, after refining the mesh, certain constraints and constraint sets removed earlier may become potentially active or enforced again, and need to be added back to the OCP. If this happens, the previous solution will no longer be a feasible initial guess for the new problem formulation. To assist the subsequent solve of NLPs, the following auxiliary feasibility problem (AFP) can be solved beforehand:

\[
J^* := \min_{\tilde{X}, \tilde{U}, p, t_0, t_f} \sum_{l=1}^{n_g} \tilde{s}_l \tag{3a}
\]

subject to, for \(i = 1, \ldots, N^{(k)}\) and \(k = 1, \ldots, K\),

\[
\sum_{j=1}^{N} A_{ij}^{(k)} X_j^{(k)} + B_{ij}^{(k)} f(X_i^{(k)}, U_i^{(k)}, \tau_i^{(k)}, t_0, t_f, p) = 0 \tag{3b}
\]

\[
e(X_i^{(1)}, U_i^{(1)}, \tau_i^{(1)}, t_0, t_f) \leq s, \text{ with } s \geq 0 \tag{3c}
\]

\[
\phi(X_i^{(1)}, t_0, X_K^{(1)}, t_f) \geq 0 \tag{3d}
\]

with \(s \in \mathbb{R}^{n_g}\) slack variables. The initial guess for the AFP will be \(\tilde{Z} := (\tilde{X}, \tilde{U}, p, \tilde{\tau}, t_0, t_f)\), the values of the interpolated solution \(\tilde{z}\) at the collocation points of the refined mesh.

E. Properties of the external constraint handling strategy

Proof of feasibility and optimality invariance for the removal of inactive constraints and redundant constraint sets on a given discretization mesh can be made straightforward. However, with the size of the mesh changing throughout the refinement process, the analysis of errors, and identification of constraint activation status (Section III-B) are all subject to considerable uncertainties. If a constraint or constraint set has been erroneously identified as inactive or redundant in earlier iterations, it must be included again in the subsequent OCP formulation (in a way similar to [1]) to ensure the solution of the modified OCP is equivalent to the unmodified problem. The challenge is to ensure that the subsequent OCP solve can be supplied with a feasible initial guess close to the potential solution, to assist the IPM-based NLP solvers.

The introduction of the AFP is the answer to this challenge, with its solution guaranteed to be a feasible point for the corresponding original OCP, thus suitable to be used as a initial point.

**Proposition 1:** If the original OCP (2) has feasible points, then a solution to the auxiliary feasibility problem (3) will be a feasible point of (2) on the same discretization mesh, and (3) will have corresponding objective value \(J^* = 0\).

**Proof:** If \(Z := (X, U, \tau, t_0, t_f)\) is a feasible point of (2), then (2a) must hold. With \(s \geq 0\), the solution for the AFP (3) will be the situation where \(\sum s = 0\), and (2c) guarantees the existence of such a solution.

Now, for the very same reason, we need to obtain a suitable initial guess for the slack variables \(s\) in the AFP. One possible way is by calculating the constraint violation errors of the interpolated solutions on the refined mesh.

**Proposition 2:** Define \(\tilde{s} \in \mathbb{R}^{n_g}\) as the absolute local constraint violation error \(\epsilon_c(t)\) calculated at the collocation points of the refined mesh, with the updated initial guess \(\tilde{Z} := (\tilde{X}, \tilde{U}, p, \tilde{\tau}, t_0, t_f)\). For any set \(\{s \in \mathbb{R}^{n_g} | \tilde{s}_i \geq \max_{i=1, \ldots, N}\{\tilde{s}_{i,l}\}, l = 1, \ldots, n_g\}\) implemented as the initial guess for \(s\), the AFP (3) will have a strictly feasible initial point with respect to the constraints (3c).

**Proof:** \(\tilde{s}_{i,l} = \max_s([-c_l(X_i, U_i, p, \tau_i, t_0, t_f)])\) by definition.

- if \(c_l(X_i, U_i, \tau_i, t_0, t_f) < 0\), i.e. the constraint is satisfied and the solution is not on the boundary, then \(\tilde{s}_{i,l} = 0\), thus \(c_l(X_i, U_i, \tau_i, t_0, t_f) < \tilde{s}_{i,l}\) holds.
- if \(c_l(X_i, U_i, \tau_i, t_0, t_f) = 0\) (constraint satisfied and solution is on the boundary) or \(c_l(X_i, U_i, \tau_i, t_0, t_f) > 0\) (constraint violation occurs), then \(\tilde{s}_{i,l} = \max_{i=1, \ldots, N}\{\tilde{s}_{i,l}\}\), the inequality \(c_l(X_i, U_i, \tau_i, t_0, t_f) \leq \tilde{s}_{i,l}\) will always be true. From \(\tilde{s}_i = \max_{i=1, \ldots, N}\{\tilde{s}_{i,l}\}\), it can then be concluded that \(c_l(X_i, U_i, \tau_i, t_0, t_f, p) \leq \tilde{s}_{i,l}\) holds for all \(i = 1, \ldots, N\) and \(l = 1, \ldots, n_g\).

Thus, except for the initial solve, all subsequent solves of OCPs and AFPs will have feasible initial guesses.

**Proposition 3:** If the unmodified OCP has feasible points, and the initial solve of the discretized OCP has been successful, then all subsequent solves of OCPs and AFPs with mesh refinement schemes and the proposed external constraint handling method will have a feasible initial point with respect to the constraints (2c) or (3c).

**Proof:** For any interpolated solution \(\tilde{Z} := (\tilde{X}, \tilde{U}, p, \tilde{\tau}, t_0, t_f)\) on the new mesh, if (2c) is not satisfied, then the corresponding AFP will be solved and Proposition 2 ensures that AFP will have a feasible initial guess. Next from Proposition 1 the solution of the AFP will be a feasible initial guess for the subsequent OCP solve.

F. A practically more efficient alternative implementation

Although in the extreme format, the proposed external constraint handling scheme can guarantee feasible initial points under conditions stated in Proposition 3. Nevertheless, it is not efficient in practice. The frequent solve of AFPs are not only time consuming, but also often are not necessary.
Note again the conditions regarding ideal initial guesses for IPM methods. The requirement on primal and dual feasibility is actually not to satisfy, but rather being close to fulfillment. In fact, the computational performance of a modern IPM solver using near feasible guesses are very much comparable to feasible initial points. In addition, constraint satisfaction for variable simple bounds can be computationally much easier to achieve by the NLP solver, thus no need to enforce it through the solve of an AFP.

Another aspect is that, since constraint addition is a more complicated process than constraint removal, it would be much more preferable to erroneously identify an inactive constraint as active, than the opposite situation. Therefore, to compensate for potential inaccuracies, it is often a good idea in practice to enlarge the intervals with potential constraint activation by an interval size \( \beta \), with \( \beta \) either fixed or adaptively changing during the mesh refinement process. This adaptive change also guarantees the convergence of the overall scheme, i.e. in the worst case, \( \beta \) can be sufficiently large to impose the constraints for the whole trajectory, with the original problem recovered.

Thus, a practically more efficient version of the external handling scheme can be formulated, by restricting the conditions for solving the AFP to only the following cases:

- when a potentially redundant path constraint set turns into a potentially enforced path constraint set, or
- when the start/end points for the potential constraint activation intervals has been reduced/enlarged by an amount larger than the size of the buffer zones introduced by \( \beta \).

IV. EXAMPLE

To demonstrate the computational benefits of the proposed external constraint handling scheme, here we illustrate an example problem that is relative large in time dimension. The task involves the finding of a fuel-optimal flight path of a commercial aircraft (Fokker 50). In the vicinity of possible flight routes, air traffic control authorities have identified five non-flight zones (NFZ) that the aircraft needs to avoid.

From simple flight mechanics with a flat earth assumption, both the longitudinal and lateral motion of the aircraft can be described by the following dynamic equations

\[
\begin{align*}
\dot{h}(t) &= v_T(t) \sin(\gamma(t)) \\
POS_N(t) &= v_T(t) \cos(\gamma(t)) \cos(\chi(t)) \\
POS_E(t) &= v_T(t) \cos(\gamma(t)) \sin(\chi(t)) \\
\dot{v}_T(t) &= \frac{1}{m(t)}(T(v_C(t), h(t), \Gamma(t)) - D(v_T(t), h(t), \alpha(t)) - m(t)g \sin(\gamma(t))) \\
\dot{\gamma}(t) &= \frac{1}{m(t)v_T(t)}(L(v_T(t), h(t), \alpha(t)) \cos(\phi(t)) - m(t)g \cos(\gamma(t))) \\
\dot{\chi}(t) &= \frac{L(v_T(t), h(t), \alpha(t)) \sin(\phi(t))}{\cos(\gamma(t))v_T(t) m(t)} \\
\dot{m}(t) &= FF(h(t), v_C(t), \Gamma(t))
\end{align*}
\]

with \( h \) the altitude [m], \( POS_N \) the north position [m], \( POS_E \) the east position [m], \( v_T \) the true airspeed [m/s], \( \gamma \) the flight path angle [rad], \( \chi \) the tracking angle [rad], and \( m \) the mass [kg]. \( T, L, D \) are the thrust, lift and drag forces, and \( FF \) is the fuel flow model. The tabular data require the input as the calibrated airspeed \( v_C \), which can related to \( v_T \) via a conversion using atmospheric parameters.

Additionally, \( g = 9.81 \text{ m/s}^2 \) is the gravity acceleration. We have three control inputs, \( \phi \) the roll angle in [rad], \( \Gamma \) the throttle settings normalized between 0 and 1, and \( \alpha \) the angle of attack in [rad]. Further details of the aerodynamic modelling can be obtained from [13].

The avoidance of NFZs can be implemented with the following path constraints

\[
(POS_N(t) - POS_{N0})^2 + (POS_E(t) - POS_{E0})^2 \geq r_N^2
\]

with \( POS_{N0} \) and \( POS_{E0} \) the north and east position of the center of the non-flight zones, and \( r_N \) the radius.

The problem will have the boundary cost \( \Phi(\{x(t_0), x(t_f), \alpha(t_f), \Gamma(t_f)\} = -m(t_f) \) (maximize the mass at the end of the flight, with fixed \( t_f = 7475 \text{ s} \), subject to the dynamics and path constraints. Furthermore, variable simple bounds are imposed together with the boundary conditions.

The OCP is transcribed using the optimal control software ICLOCS2 [14] with Hermite-Simpson discretization, and solved with IPM-based NLP solver IPOPT [15] (version 3.12.4) compiled with linear solver MA57 [16]. All computation results shown were obtained on an Intel Core i7-4770 computer, running 64-bit Windows 10 with MATLAB 2017a.

Figure 3 illustrates the results of the example problem solved to a user-defined tolerance. In can be observed that only two out of five NFZ are actually active constraints to the problem, and they are only active in relatively short durations of the complete flight trajectory. Therefore, including all possible constraints into the OCP may not be favourable.

Using the proposed external constraint handling scheme, with a very conservative buffer interval setting of \( \beta = 0.1(t_f - t_0) = 747.5 \text{ s} \), the simulation history for constraint activation intervals implemented in the OCP are demonstrated in Table IV. It can be seen that in the initial solve.
(MR iteration 1), all constraint sets are enforced and all constraints are treated as potentially active. Based only on the solution from this coarse grid, the constraint handling method correctly identified that the constraint sets related to NFZ 2, 3 and 5 are all potentially redundant. It also determined that constraints in the constraint sets related to NFZ 1 are only potentially active near the end of the flight, whereas for NFZ 4 they are at the beginning of the mission.

In later iterations of the mesh refinement, these intervals generally stay the same, with a tendency to slightly and monotonically shrink as the information gets more accurate. One exception seen is the interval end time for NFZ 4, with minor fluctuations meaning some potentially inactive constraints need to be considered as potentially active at some iterations (e.g. iteration 2, 5 and 6). Nonetheless, since these changes are significantly smaller than the interval buffer $\beta$ introduced, they will have no influences on initial point feasibility for the OCP in subsequent iterations, and the solve of AFPs will not be required.

Table II compares the computational performance of the standard solve, as well as solves with external constraint handling function enabled. Even with a very conservative buffer interval setting, the total computation time saw a 40% reduction, while the number of mesh refinement iterations remain the same. More importantly, the resultant OCP problem and the discretization mesh after mesh refinement is much more efficient for re-computations, costing only half of the time compared to the standard solve. Therefore, the benefits of the proposed external constraint handling can be clearly seen for both off-line and real-time embedded applications.

V. Conclusions

A strategy has been developed to systematically identify and handle inactive constraints and redundant constraint sets for solving optimal control problems numerically together with mesh refinement schemes. Unlike previous work that would always result in infeasible initial guesses, the proposed scheme is capable of providing guarantees on the feasibility of initial points under mild conditions (the original OCP has feasible points, and the initial solve of the discretized OCP is successful), making it particularly suitable for OCP toolboxes that utilize IPM-based NLP solvers in lowering the computational cost.

REFERENCES