SAMPLED–DATA MODEL PREDICTIVE CONTROL: ADAPTIVE TIME–MESH REFINEMENT ALGORITHMS AND GUARANTEES OF STABILITY

LUÍS TIAGO PAIVA AND FERNANDO A. C. C. FONTES
Systec–ISR, Faculdade de Engenharia, Universidade do Porto
4200-465, Porto, Portugal

ABSTRACT. This article addresses the problem of controlling a constrained, continuous–time, nonlinear system through Model Predictive Control (MPC). In particular, we focus on methods to efficiently and accurately solve the underlying optimal control problem (OCP). In the numerical solution of a nonlinear OCP, some form of discretization must be used at some stage. There are, however, benefits in postponing the discretization process and maintain a continuous-time model until a later stage. This is because that way we can exploit additional freedom to select the number and the location of the discretization node points. We propose an adaptive time–mesh refinement (AMR) algorithm that iteratively finds an adequate time–mesh satisfying a pre–defined bound on the local error estimate of the obtained trajectories. The algorithm provides a time–dependent stopping criterion, enabling us to impose higher accuracy in the initial parts of the receding horizon, which are more relevant to MPC. Additionally, we analyze the conditions to guarantee closed–loop stability of the MPC framework using the AMR algorithm. The numerical results show that the proposed AMR strategy can obtain solutions as fast as methods using a coarse equidistant–spaced mesh and, on the other hand, as accurate as methods using a fine equidistant–spaced mesh. Therefore, the OCP can be solved, and the MPC law obtained, faster and/or more accurately than with discrete-time MPC schemes using equidistant–spaced meshes.

1. Introduction. In this work, we address the problem of constructing a sampled-data feedback control law to stabilize a nonlinear continuous–time system using a Model Predictive Control (MPC) technique. We propose a method, an adaptive time–mesh refinement algorithm, to solve the optimal control problems involved in computing the sampled-data MPC law. We show that the proposed approach can improve the computational efficiency and accuracy while preserving the guarantees of stability.

A sampled-data feedback system is obtained whenever the variables in a plant evolve in continuous–time and are controlled using a digital device. This is the most frequent situation in applications with some degree of complexity requiring an advanced control, which is not easy or practical to implement in an analogue device. The resulting sampled-data systems are often addressed, in practice and in the literature, using a simplified discrete–time model. Regarding the MPC literature, in the last decade, frameworks using discrete–time models clearly dominate...
(see e.g. [10, 12, 31]). However, some of the earlier fundamental theoretical works on nonlinear MPC (e.g. [8, 17, 34, 35]) use continuous-time models. In the implementation of an MPC scheme, there is the need to perform, eventually, some form of discretization, or at least a finite parameterization, to solve the optimal control problems (OCP’s) involved. Nevertheless, we argue that there are several advantages in maintaining a continuous–time model until later stages. That is, the continuous–time model is fed into the optimizer, which is responsible for the discretization occurring afterwards. The benefits of using an MPC framework in a sampled–data setting can be of an analytical nature or of a numerical nature and are discussed below.

Regarding the benefits of analytical nature, we highlight the following:

• the plant model is usually obtained in continuous–time: when using a discrete–time approximation from the beginning, it might be difficult to analyze the errors introduced in the discrete approximation;

• discrete–time models generally have equidistant sampling instants: this might not be adequate for some nonlinear systems that have multiple time-scales of the dynamics, systems that can rapidly change behavior in short time intervals, or even when there is a discontinuity at some critical instant (e.g. impulsive control, [23], or discontinuous feedbacks [18]);

• in path–following problems, the speed at which a reference trajectory is followed has to be adapted in real-time: this procedure is more complex in discrete–time ([41, 42]) than in continuous–time ([7, 14, 44]);

• the inter-sampling behavior of a continuous–time plant is not appropriately described by a discrete–time model: as a consequence, we might have hidden oscillations or constraint violations [24] that are not detected by the model.

The benefits of numerical nature are the focus of this article. They are essentially due to the fact that when solving OCP’s, it is possible to devise numerical procedures that are more efficient using continuous–time models than using discrete–time models. One reason is that with continuous–time models we can exploit additional freedom in placing node points, manipulating the accuracy of the trajectory on several different time subintervals. Another benefit relates to the flexibility in manipulating the horizon length. It is well–known to the MPC community that increasing the prediction horizon might benefit, or even be necessary to guarantee, stability or recursive feasibility. Nevertheless, it is often reported that the computational time increases with the horizon length. While this is true for discrete–time systems with constant inter–sampling time, it is not necessarily the case when continuous–time is used. It is worth noting that some of the authors of works that vastly contributed to spread the use of discrete–time MPC ([28, 33, 43]) have also explored advantages of frameworks with continuous–time models. In [39], it is shown that a continuous–time method is significantly more efficient than standard discrete–time methods when solving constrained linear quadratic problems. In [27], Grüne et. al. provide an MPC scheme with a sampled–data implementation of the control law with the aim to minimize sampling errors. In [26], the authors propose a multi–step MPC scheme, which despite of being a discrete–time scheme, allows the optimization and feedback to be updated at different time instants. It is shown in [26] that such technique can maintain stability and quantifiable robust performance estimates while keeping the computational requirements low. When the optimal control problems maintain a continuous-time model, the numerical solution can use Adaptive Time-Mesh Refinement (AMR) algorithms, which have shown significant
computational time savings when compared with solving the OCP with similar accuracy on equidistant time meshes (as the ones used in the discrete-time models). (For a description of AMR algorithms see [4, 38, 40, 48] and the references therein.) Preliminary results on the application of the AMR technique to MPC, have been reported in [36, 37].

We argue that this is the right time to start using more often MPC in the sampled-data setting, even in applications. Many theoretical questions of using sampled-data systems are well documented in the literature ([15, 19, 22, 27, 32]). Also, there are now several readily available software packages to solve nonlinear OCP's ([5, 13, 29]), which greatly facilitate the implementation of sampled-data nonlinear MPC schemes. Moreover, the choice of the discretization inter-sampling intervals can be automated, by pre-defining some error criteria, if adaptive time-mesh refinement algorithms such as the one described here are used.

In this paper, we consider an adaptive time-mesh refinement (AMR) algorithm to solve a nonlinear OCP ([38]) and extend it to include time-dependent levels of refinement, so that it can be applied in an MPC setting. When using direct methods to solve the OCP, an essential first step is to select an appropriate mesh to discretize the time interval and the dynamic equation. This selection — the number and location of the mesh node points —, has implications on the accuracy of the solution and on the efficiency of the method. Adaptive mesh-refinement algorithms increase iteratively the number of node points in the time subintervals for which the accuracy of the solution must be improved to attain a desired accuracy. The developed algorithm iteratively computes an adequate time-mesh that, in the end, satisfies a pre-defined bound on the local discretization error estimate of the obtained trajectories.

In an MPC context, the prediction can be interpreted in the sense of planning. When we make plans for the future, we establish planning strategies depending on the prediction horizon. When we plan our – professional or private – schedule, we plan with more detail and on a finer time scale in the shorter term: we might do a detailed hourly plan for the next day, a daily plan for the following week, and a draft of a monthly plan for the next year. Combining this idea with the refinement strategy, we obtain an adaptive time-mesh refinement algorithm which generates meshes with higher concentration of node points in the beginning of the prediction horizon and less concentration of node points in the end of the same interval. This way, we are bringing about the idea of having more node points where they are needed while keeping the overall number low. This is an important feature, since we want to increase the accuracy of the solution while maintaining a modest computational time.

The main features in the adaptation to an MPC context of the refinement strategy developed for an OCP are (a) the pre-defined levels of refinement are now time-dependent, which requires a generalization of the initially developed procedure; (b) since similar (time-shifted) OCP’s are repeatedly solved, the developed procedure projects in the new time-mesh the solution of a previous OCP, to create a warm start; (c) in addition, we show that the MPC with the AMR algorithm preserves stability, provided that the design parameters of the controller satisfy a given stability condition.

This paper is organized as follows. In Section 2, we characterize the system model, the model assumptions, and the sampled-data feedback control setting. Section 3 introduces the optimal control problem used, while Section 4 discusses
its numerical solution methods. In Section 5, we describe the adaptive time–mesh refinement (AMR) algorithm to solve OCP’s and extend it to time–dependent levels of refinement. In Section 6, we provide the MPC framework and combine it with the extended AMR algorithm. In Section 7, we analyze the stability of the AMR–MPC scheme. In Section 8, we illustrate the application of the AMR–MPC scheme to a car–like system involving parking maneuvers and we report numerical results attesting the performance of the methodology developed. Finally, some conclusions are drawn in Section 9.

Nomenclature.

| System data | Design parameters | Optimal Control and MPC frameworks |
|-------------|-------------------|-----------------------------------|
| \( f : \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}^n \) | \( L : \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R} \) | \( \mathcal{P}(x_0) \) Optimal Control Problem (OCP) starting at \( x_0 \) |
| initial state measured at instant \( t_0 \) | \( G : \mathbb{R}^n \rightarrow \mathbb{R} \) | \( \mathcal{A}_0 \subset \mathbb{R}^n \) set of admissible initial states |
| \( x_0 \in X_0 \) | \( T \in \mathbb{R}_+ \) | \( \mathcal{J} \) cost functional of the OCP |
| \( x_0 \in \mathbb{R}^n \) | \( X_f \subset \mathbb{R}^n \) | \( V \) value function of the OCP |
| state sampled at instant \( t_k \) | \( \mathcal{X} \subset \mathbb{R}^n \) | \( (\hat{x}, \hat{u}) \) optimal trajectory–control pair solving the OCP |
| \( X_0 \subset \mathbb{R}^n \) | \( U \subset \mathbb{R}^m \) input constraints set |
| set of possible initial states | \( \kappa : \mathbb{R}_+ \times \mathbb{R}^n \rightarrow \mathbb{R}^m \) | \( (\bar{x}, \bar{u}) \) admissible trajectory–control pair for the OCP |
| \( X \subset \mathbb{R}^n \) | \( \kappa : \mathbb{R}_+ \times \mathbb{R}^n \rightarrow \mathbb{R}^m \) | \( (x^*, u^*) \) closed–loop trajectory–control pair, numerically computed with MPC |
| state constraints set | \( \mathcal{F} : 0, 1, 2, \ldots, K \) | \( \mathcal{A}(x_0) \) set of admissible trajectory–control pairs satisfying the constraints of \( \mathcal{P}(x_0) \) |
| \( \tau_i \) | \( t \) real time |
| mesh refining interval | \( t_k \) sampling instant |
| \( \Pi = \{ t_k \}_{k \geq 0} \) sequence of sampling instants |
| \( j \) refinement level index | \( s \) simulation time used in the prediction model and OCP |
| \( i : 0, 1, 2, \ldots \) mesh node index | \( s_i \) mesh node instant |
| \( \tau_i \) extreme of the mesh refining interval |
| \( j \) refinement level index | \( s_i \) mesh node instant |
| \( \tau_i \) extreme of the mesh refining interval |
| \( \epsilon(s) \) local discretization error estimate at time \( s \) |
| \( \epsilon_j \) level of refinement \( j \) |
| \( \epsilon_{\text{max}} \) stopping criterion for the refinement procedure |
| \( r \) refinement iteration index |
| \( \pi_r = \{ s_i \}_{i=0, 1, \ldots, N_r} \) optimization mesh at \( r^{\text{th}} \) refinement iteration |
| \( U(\pi_0, [t_1, t_2]) \) set of control functions from \( [t_1, t_2] \) to \( U \), piecewise constant on \( \pi_0 \). |
2. The addressed system. In this section, we describe the system model, the sample-data feedback control and the model assumptions.

We consider a dynamical system for which the evolution of the state after some time $t_0$ is described by the following model:

$$\dot{x}(t) = f(x(t), u(t)) \quad \text{a.e. } t \geq t_0, \quad (1a)$$

$$x(t_0) = x_0. \quad (1b)$$

The state and input constraints are expressed as

$$x(t) \in X \quad \forall t \geq t_0, \quad (2a)$$

$$u(t) \in U \quad \text{a.e. } t \geq t_0, \quad (2b)$$

where $X \subset \mathbb{R}^n$ is the state constraint set, and $U \subset \mathbb{R}^m$ is the control constraint set. The initial state $x_0$, measured at the time instant $t_0$, is selected from the set of possible initial states $X_0 \subset X$, $x_0 \in X_0$.

The trajectory solution to (1) is here denoted by the function $t \mapsto x(t; x_0, u)$, when is convenient to identify explicitly the dependence of the trajectory with respect to a particular initial state or control function.

We assume that the system satisfies the following set of hypotheses.

**H1:** The sets $X$, $X_0$ and $U$ are compact, contain the origin and $f(0,0) = 0$.

**H2:** The system (1)-(2) can be driven asymptotically to the origin, from $X_0$, by some open-loop control. That is, for any $x_0 \in X_0$, there exists a control function $u$ with $u(t) \in U$, such that $x(t; x_0, u) \to 0$ as $t \to \infty$.

**H3:** The function $f$ is continuous, and $x \mapsto f(x, u)$ is Lipschitz continuous for any $u \in U$.

It should be noted that these hypotheses depend only on the data of the given system. Later, we establish additional conditions depending on the design parameters to achieve certain desired properties.

2.1. Sampled-data feedback and stabilization. Our objective is to drive the state of the system to a desired equilibrium point, here considered without loss of generality to be the origin. This goal is to be achieved using sampled-data (SD) feedback control. In sampled-data (SD) feedback control (e.g. [9]), at any given time, the control is not a function of the state at that time, rather it is a function of the state at the last sampling instant. More precisely, consider a sequence of sampling instants $\Pi := \{t_k\}_{k \in \mathbb{N}}$ with a constant sampling period $\delta > 0$ and $t_{i+1} = t_i + \delta$ for all $k \in \mathbb{N}$. Consider also a feedback law $\kappa : \mathbb{R}_+ \times \mathbb{R}^n \to \mathbb{R}^m$ and a function $t \mapsto [t]_\Pi$ that extracts from $\Pi$ the last sampling instant before $t$, that is

$$[t]_\Pi := \max_k \{t_k \in \Pi : t_k \leq t\}.$$

The control obtained via sampled-data feedback is defined for any $t \geq t_0$ by

$$u(t) := \kappa(t, x([t]_\Pi)).$$

The solution of system (1)-(2) under this sampled-data feedback, starting from some initial state $x_0$ is denoted by the function $t \mapsto x(t; x_0, \kappa(t, x([t]_\Pi)))$, or simply by $t \mapsto x(t; x_0, \kappa)$. The MPC technique, described below, combines naturally with this sampled-data feedback concept (see [16, 18]).

To meet the goal of driving the state of the system to the origin through a sampled-data feedback, we must have open-loop controllability as is imposed in
2340  LUÍS TIAGO PAIVA AND FERNANDO A. C. C. FONTES

H2. The converse, whether asymptotical controllability implies sampled-feedback stabilization, is less obvious and was shown in [9] (in that result, though, somewhat different notions of controllability, stabilizability and sampled-data feedback were used).

3. The optimal control problem. The Model Predictive Control strategy involves solving a sequence of open-loop Optimal Control Problems (OCP) at each sampling instant $t_k$, every time using the currently measured state of the plant $x_k$. We consider the following OCP starting from an initial state $x_k \in X_0$:

$$P(x_k) : \text{Minimise } \int_0^T L(x(s), u(s)) \, ds + G(x(T))$$

subject to

$$\dot{x}(s) = f(x(s), u(s)) \quad \text{a.e. } s \in [0,T],$$

$$x(0) = x_k,$$

$$x(T) \in X_f,$$

$$x(s) \in X \quad \forall s \in [0,T],$$

$$u(s) \in U \quad \text{a.e. } s \in [0,T].$$

Here, in addition to the data already specified in (1)–(2), we use a fixed horizon $T$ in $\mathbb{R}_+$, the running cost function $L : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$, the terminal cost function $G : \mathbb{R}^n \to \mathbb{R}$, and the terminal state constraint set $X_f \subset \mathbb{R}^n$. The quantities $T, L, G$ and $X_f$ are, in general, not given with the system plant (1) and can therefore be selected by the control engineer to determine the characteristics of the controller and the corresponding behavior of the controlled system. Hence, we call design parameters to such quantities.

We note that since the OCP is time-invariant, it is indifferent to consider and solve the problem in the interval $[t_k, t_k + T]$ or $[0, T]$, as long as the initial state $x_k$ is the correct one. For a time-varying formulation of this problem see e.g. [17]. For a general reference on optimal control see e.g. [25, 46].

Regarding the domain of the optimization problem, we might conceptually consider that the control functions $u$ are selected from the space of measurable functions. However, practicality imposes that these problems have to be solved numerically and so the domain of the optimization problem has to be finitely parameterizable (i.e. the control functions have to be representable in a digital computer using a finite number of parameters). The parameterization used might be based on piecewise constant control, which is the most common choice (e.g. [32, 38]), and also is the one used here. Alternatively, the parameterization can be based on polynomials or splines ([40]), based on bang-bang controls ([19]), or any other representation of the control functions by a finite set of parameters.

4. Numerical solution of optimal control problems. The solution of nonlinear optimal control problems, in most cases, cannot be explicitly characterized analytically, but can only be obtained using numerical methods. Numerical methods to solve these problems include dynamic programming-based algorithms (solving the Hamilton–Jacobi–Bellman equation), indirect methods (applying the maximum principle to convert the problem into a boundary value problem), and direct methods (discretizing and solving the resulting static nonlinear programming problem).
To solve the OCP, we employ direct methods [3], in which the model is initially discretized in a given time–mesh, the controls are parameterized by piecewise constant functions, and then the problem is transcribed into a finite-dimensional nonlinear programming problem (NLP). The resulting NLP can be solved using one of the several widely available optimization solvers.

We stress that even when using a piecewise constant control parameterization, the intervals of constant control (defining what we call the time–mesh of the optimization solver) do not have to coincide with the inter-sampling intervals (defining the instants when each MPC iteration starts). Here, the optimization mesh \( \pi_0 = \{ s_i \}_{i \in \{0,1,...,N_0\} } \) defines the node points where the control function can change value; it contains the sampling instants in \( \Pi \) in the interval \([0,T]\) and possibility some other node points, \( i.e. \pi_0 \supset \Pi \cap [0,T] \).

We consider here that in the OCP above, we are seeking a minimizer by selecting a control function among the set \( U \) of functions from \([0,T] \) to \( \mathbb{R}^m \) taking values on \( U \) which are piecewise constant on \( \pi_0 \) and right–continuous functions. The domain of the optimization problem is the set of admissible pairs \( (x,u) \) comprising a control function \( u \in U \) and the corresponding state trajectory satisfying the constraints \((3b)-(3f)\), defined to be

\[ A(x_k) := \{ (x,u) : u \in U, \text{ equations } (3b)-(3f) \text{ are satisfied for } P(x_k) \} . \]

If there exists an admissible process, \( i.e. \) if the above set is non-empty, we say that the OCP is feasible. It is convenient to define also the set of admissible initial states to be

\[ A_0 := \{ x_k \in X : P(x_k) \text{ is feasible} \} . \]

We say that a pair \( (\bar{x}, \bar{u}) \in A(x_k) \) is a minimizer or an optimal solution to \( P(x_k) \) if for all \( (x,u) \in A(x_k) \) we have

\[ \int_0^T L(\bar{x}(s), \bar{u}(s)) \, ds + G(\bar{x}(T)) \leq \int_0^T L(x(s), u(s)) \, ds + G(x(T)) \]  \hfill (4)

We note that the set \( U \) depends on the optimization mesh \( \pi_0 \) and so does the set \( A(x_k) \), the set \( A_0 \), as well as the minimum value on the left-hand side of \((4)\). We use the notation \( U(\pi_0) \) to make this dependence explicit when useful, and the notation \( U(\pi_0, [t_1,t_2]) \) when the control functions to consider have the domain restricted to some subinterval \([t_1,t_2]\). The difference between the minimum value of the problem when the controls are selected from \( U(\pi_0) \) and when they are selected from the space of measurable functions is here denoted by optimality gap of \( \pi_0 \).

The conditions to guarantee the existence of an optimal solution to the discretized optimal control problem are the same as the ones to guarantee the existence of an optimal solution to the equivalent NLP which, by the Weierstrass extreme value theorem, are simply the continuity of the cost functions \( L \) and \( G \) and compactness and non-emptiness of the domain.

The fact that the ordinary differential equation \((1a)\) is discretized and then integrated numerically leads to a discretization error in the trajectory. We note that this is distinct and does not include the optimality gap due to the use of piecewise constant control functions. Given two consecutive mesh nodes \( s_i, s_{i+1} \), the local discretization error, here denoted by \( \varepsilon(s_{i+1}) \), is the difference between the computed solution and the exact solution of the differential equation on \([s_i, s_{i+1}]\) when the two solutions coincide at \( s_i \). The local discretization error is a measure used to guide the refinement in adaptive mesh-refinement (AMR) procedures (cf.
5. Adaptive Mesh Refinement (AMR) for OCP. When numerically solving an OCP, most frequently, regular time meshes having equidistant spacing are used in the discretization procedure. However, in some cases, these meshes are not the most adequate to deal with nonlinear behaviors. One way to improve the accuracy of the results, while maintaining a reasonable computational time and memory requirement, is to construct a mesh having different time steps. The best location for the smaller step sizes is, in general, not known a priori, so the mesh is refined iteratively. In a mesh–refinement procedure the problem is solved, typically, in an initial coarse uniform mesh in order to capture the basic structure of the solution and of the local discretization error. Then, this initial mesh is repeatedly refined according to a chosen strategy until some stopping criterion is attained. Several mesh refinement methods employing direct collocation methods have been described in the recent years ([3, 4, 40, 48]). The refinement method used here

(a) permits several levels of refinement, obtaining a multi–level time–mesh in a single iteration;
(b) it also permits different refinement criteria – the local discretization error of the primal variables, the discretization error of the dual variables and a combination of both criteria can be used in the refinement procedure;
(c) it considers distinct criteria for refining the mesh and for stopping the refinement procedure – the refinement strategy can be driven by the information given by the dual variables and it can be stopped according to the information given by the primal variables.

As described in [38], there are advantages in choosing the error of the adjoint multipliers as a refinement criterion, namely in lowering the computational time, because the adjoint multipliers are solution to a linear differential equation system, which is easily solved with high accuracy. To decrease the overall computational time, the solution computed in the previous iteration is used as a warm start in the next one, which proved to be of major importance to improve computational efficiency. This adaptive strategy leads to results with higher accuracy and yet with lower overall computational time, when compared to results obtained by meshes having equidistant spacing, as is the case when using discrete–time models from the beginning.

5.1. The AMR algorithm. We describe here a specific multi-level AMR algorithm proposed in [38], where we refer to for further details. The adaptive time–mesh refinement procedure starts by discretizing the time interval \([0, T]\) in an initial coarse mesh

\[ \pi_0 = \{s_i\}_{i=0,1,...,N_0} \subset [0, T] \]

where the NLP problem associated to the OCP is obtained and solved in order to capture the main structure of the solution and to estimate the discretization error. To simplify presentation, we consider that this mesh is used both to define the piecewise constant control intervals and to discretize the differential equation (1a).

In the second step, we need to identify the time intervals that verify the refinement criteria. That is, we partition the time domain into a set of intervals for which the discretization error estimate is above or below a certain threshold \(\varepsilon^{\max}\).
Defining the refining intervals

\[
S_i = [\tau_{i-1}, \tau_i], \quad i = 1, \ldots, \mathcal{I} - 1
\]

\[
S_{\mathcal{I}} = [\tau_{\mathcal{I}-1}, \tau_{\mathcal{I}}],
\]

we identify the interval extremals \( \tau_0, \ldots, \tau_{\mathcal{I}} \) and whether each interval is to be refined. The interval extremals are in \( \pi_0 \) and have the property that \( \tau_0 < \tau_1 < \tau_2 < \ldots < \tau_{\mathcal{I}} \).

In the third step, the intervals \( S_i \) are refined taking under consideration different levels of refinement, \( i.e. \), they are divided into smaller intervals according to user-defined levels of refinement \( \varepsilon_1, \varepsilon_2, \ldots, \varepsilon_J \) with \( \varepsilon_1 = \varepsilon^{\max} \). We consider that an interval \( S_{i,j} \) is in the \( j^{th} \) level of refinement if

\[
S_{i,j} = \{ s \in S_i : \varepsilon(s) \in [\varepsilon_j, \varepsilon_{j+1}] \}, \quad j = 1, \ldots, \mathcal{J}.
\]

where \( \varepsilon(s) \) is an estimate of the discretization error.

The algorithm, while constructing the next mesh \( \pi_1 \), adds more node-points to intervals in higher refinement levels and it adds less node points to those in lower refinement levels (see Figure 1), following a block-structured adaptive mesh refinement strategy where no nodes are removed, just added, \( i.e. \pi_1 \supset \pi_0 \). By defining several levels of refinement in a single iteration, we generate a multi-block time-mesh.

In the example of the figure, we are subdividing each interval in the first level into \( \mathcal{N} = 2 \) subintervals (adding \( \mathcal{N} - 1 \) additional nodes to each interval). Each time interval in refining level \( j \) is subdivided into \( \mathcal{N}^j \) subintervals (by adding \( \mathcal{N}^j - 1 \) additional nodes). The procedure is then repeated for iteration \( r = 1, 2, \ldots, \) until a stopping criterion is achieved \( (\varepsilon(s) \leq \varepsilon^{\max} \) for all \( s \in [0, T] \)).

**Figure 1.** Illustration of the multi-level adaptive time-mesh refinement strategy

**Refinement and stopping criteria.** In order to proceed with the mesh refinement strategy, we have to define some refinement criteria and a stopping criterion.

For the stopping criterion, we consider a threshold for the discretization error of the trajectory (primal variables) \( (\varepsilon_x) \), and the algorithm computes its maximum
estimate value $\| (\varepsilon_x) \|_\infty$. For the refinement criteria, we could also consider thresholds for $(\varepsilon_x)$ for different time instants, as is done in e.g. [4]. However, as shown in [38], similar error information can be extracted from the dual variables and such information can be computed much faster. Therefore, we set as refinement criteria levels of discretization error $(\varepsilon_q)$ for the adjoint multipliers.

The adjoint multipliers satisfy a differential equation system which is defined by the Maximum Principle. Such differential equation is linear and can therefore be easily solved in a fast way and with high accuracy. In order to select which time intervals should be further refined at each refinement step, the algorithm computes the local error of the multipliers for each time instant $s$:

$$\varepsilon_q(s) = \| q_{\text{MP}}(s) - q_{\text{NLP}}(s) \|,$$

where $q_{\text{MP}}$ is a solution to the differential equation system given by the Maximum Principle [21, 46], and $q_{\text{NLP}}$ is the multiplier numerically obtained by solving the associated nonlinear programming problem (NLP) using a numeric solver. We may consider that this algorithm is somehow a hybrid between a direct and indirect method, since it involves an initial discretization as in a direct method, but it is driven by information obtained by the maximum principle, as in an indirect method. We note that, currently, with the existing automatic differentiation software, obtaining the adjoint equation is an easy process that can be automated (see e.g. CasADi [2] or ICLOCS2 [1] for optimal control software with algorithmic differentiation).

5.2. Extended AMR algorithm. We extend the algorithm described in section 5.1, by considering time–dependent refinement thresholds $(\varepsilon_{\text{max}}(s))$ for the mesh refinement algorithm. This procedure adds more node points and thereby obtains a more accurate solution in the time intervals that correspond to tighter refinement thresholds, see Fig. 2. The time–dependent refinement thresholds can be defined within subintervals of the interval $[t_k, t_k + T]$ as follows

$$\varepsilon_{\text{max}}(s) = \begin{cases} 
\varepsilon_1, & s \in [t_k, t_k + \beta_1 T] \\
\varepsilon_2, & s \in [t_k + \beta_1 T, t_k + \beta_2 T] \\
\vdots \\
\varepsilon_J, & s \in [t_k + \beta_{J-1} T, t_k + T]
\end{cases}$$

where $0 < \beta_1 < \beta_2 < \ldots < \beta_{J-1} < 1$.

For example, we can define a time–dependent refinement thresholds with 3 levels:

$$\varepsilon_{\text{max}}(s) = \varepsilon_j = 10^{j-1} \varepsilon_1, \quad s \in [t_k + \beta_j T, t_k + \beta_{j+1} T]$$

for $j = 1, \ldots, 3$ with $\varepsilon_1 = 10^{-5}$. Considering $\beta_1 = 10\%$, $\beta_2 = 30\%$, we obtain:

$$\varepsilon_{\text{max}}(s) = \begin{cases} 
10^{-5}, & s \in [t_k, t_k + 0.1 T] \\
10^{-4}, & s \in [t_k + 0.1 T, t_k + 0.3 T] \\
10^{-3}, & s \in [t_k + 0.3 T, t_k + T]
\end{cases}$$

6. Model predictive control. The MPC algorithm constructs a sampled–data feedback control by concatenating the initial parts of the optimal control functions obtained as solution to each problem $P(x_k)$ along each $t_k \in \Pi$. Starting at $t_k = t_0$, the following four steps are implemented at each MPC iteration $k$.

1. Measure the current state of the plant $x_k = x(t_k)$:
2. Solve the OCP $\mathcal{P}(x_k)$ obtaining the optimal control function $\bar{u} : [0, T] \rightarrow \mathbb{R}^m$ as well as the corresponding trajectory $\bar{x} : [0, T] \rightarrow \mathbb{R}^n$.

3. Define and apply to the plant the control $u^*(t_k + t) := \bar{u}(t)$ in the interval $t \in [0, \delta]$ (the remaining part of the control function $\bar{u}(t), t > \delta$ is disregarded);

4. Repeat for the next sampling time instant $t_k \leftarrow t_k + \delta$ and the next MPC iteration $k \leftarrow k + 1$.

The pair $(x^*, u^*)$, defined for $t \in [t_0, +\infty)$, denotes the sampled-data control resulting from applying the MPC algorithm and the corresponding state trajectory.

6.1. Model predictive control coupled with the extended AMR algorithm.

The algorithm extension described in Section 5.2 is of relevance in the MPC context, since it is desirable to have a solution with higher accuracy in the initial part of the horizon.

When applying the MPC procedure to solve an OCP, at each time instant $t_k$ we compute the solution in the interval $[t_k, t_k + T]$ but we just implement the open-loop control in its initial part, until $t_k + \delta$. Therefore, taking under consideration the planning strategy discussed above, it would be an improvement if we have an adaptive time-mesh able to cope with this feature, i.e., a time-mesh that is highly refined in the initial part of the time horizon and it is coarser in the final remaining part of the same horizon. Then, we would implement the control on the time interval $[t_k, t_k + \delta]$ computed with high accuracy in a refined mesh. For the remaining time interval we have a less accurate estimate of the solution. In the example bellow, with $T = 10$ and $\delta = 2$, we might have

$$\epsilon^\text{max}(s) = \begin{cases} 10^{-5}, & s \in [t_k, t_k + 0.2T] \\ 5 \times 10^{-5}, & s \in [t_k + 0.2T, t_k + 0.4T] \\ 10^{-4}, & s \in [t_k + 0.4T, t_k + T] \end{cases}$$

Following the described strategy, we obtain an adaptive time-mesh refinement algorithm which generates meshes with higher concentration of node points in the
beginning of the interval \([t_k, t_k + T]\) and less concentration of node points towards the end of the same interval, enforcing the idea of having more nodes point where they are needed and keeping a low overall number of node points. This is an important issue because we want to increase the accuracy of the solution without compromising low CPU times.

We start the MPC procedure in the usual way but considering an extended adaptive mesh refinement strategy to solve the OCP. We discretize the time interval \([0, T]\) and we solve our OCP in open-loop. Then, we implement the control in the first sampling interval. When starting the next MPC step, we apply the time-mesh refinement strategy in order to find the best mesh suited to the solve the OCP in the second sampling interval. We repeat this technique until the MPC procedure ends. In the MPC algorithm, step 2 is modified as follows:

2. (a) Select the intervals \(S_{i,j}\) to be refined according to the time-dependent refinement thresholds \(\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_J\) and generate a new time-mesh;
   (b) Solve the OCP \(P(x_k)\) obtaining the optimal control function \(\bar{u} : [0, T] \to \mathbb{R}^n\) as well as the corresponding trajectory \(\bar{x} : [0, T] \to \mathbb{R}^n\), in the new time-mesh;

This AMR-MPC algorithm is illustrated in Fig. 3.

![Figure 3. Illustration of the extended time-mesh refinement algorithm for MPC](image-url)

**Warm start.** Since we are solving a sequence of open-loop OCP’s, to decrease the CPU time when going from the problem in \([t_k, t_k + T]\) to the problem in \([t_k + \delta, t_k + T + \delta]\), the solution of the previous is used as a warm start for the problem. To create this warm start, the solution obtained in \([t_k, t_k + T]\) is projected in the new mesh in \([t_k + \delta, t_k + T + \delta]\). This action proved to be vital in the decreasing of the overall computational time. In particular, we notice that the number of iterations of the NLP solver remains within the same order of magnitude along the procedure, even when the number of nodes is significantly increased.
Another strategy that could be pursued to further accelerate obtaining the next solution would be to use additional pre-computed information from the optimizer (e.g. Hessian matrices, gradients); see [11].

7. Stability of the AMR–MPC. In this section, we provide a stability guarantee for the AMR–MPC strategy. The main result asserts that the system resulting from the application of the MPC algorithm is stable, as long as the design parameters for the AMR–MPC strategy. The main result asserts that the system resulting from Stability of the AMR–MPC.

Sampled–Data MPC: AMR Algorithms and Stability 2347

SC: Sufficient condition for Stability.

The design parameters \( T, L, G \) and \( X_f \) satisfy:

1. The set \( X_f \) is a subset of \( X \), is closed, and contains the origin. The function \( G \) is Lipschitz continuous and positive definite. The function \( L \) is continuous and there exists a function \( M : \mathbb{R}^n \to \mathbb{R}_+ \) which is continuous, positive definite and radially unbounded, such that \( L(x,u) \geq M(x) \) for all \( u \in U \).

2. The horizon \( T \) is such that \( X_0 \) is contained in \( A_0 \), when controls from \( U(\pi_0) \) are used.

3. There exists a control law \( k_f : [0, \delta] \times X_f \to \mathbb{R}^m \), such that for all \( x_f \in X_f \),

\[
G(x(\delta;x_f,k_f)) - G(x_f) \leq -\int_0^\delta L(x(s;x_f,k_f),k_f)ds, \quad (SCa)
\]

\[
x(\delta;x_f,k_f) \in X_f,
\]

\[
x(t;x_f,k_f) \in X, \quad \text{all } s \in [0,\delta]. \quad (SCb)
\]

and

\[
k_f(\cdot,x_f) \in U(\pi_0,[0,\delta]). \quad (SCc)
\]

Remark 1. Condition 1 on the objective functions is nonrestrictive for stabilizing and tracking problems. A typical choice for \( L \) is a quadratic cost. For the terminal set \( X_f \), some possible choices are the set \( X \), a subspace of \( X \), or an ellipsoid centred at the origin. This last choice is convenient for systems for which the linearization is stabilizable (see [8]).

In case the aim is the convergence to a target point \( x^\# \) different from the origin, in which the pair \((x^#,u^#)\) is an equilibrium pair (i.e. \( f(x^#,u^#) = 0 \)), then condition 1 should be adapted as follows: The set \( X_f \) is a subset of \( X \), is closed, and contains \( x^# \). The function \( G \) is Lipschitz continuous and \( x \to G(x-x^#) \) is positive definite. The function \( L \) is continuous and there exists a function \( M : \mathbb{R}^n \to \mathbb{R}_+ \) which is continuous, radially unbounded, with \( x \to M(x-x^#) \) positive definite such that \( L(x-x^#,u-u^#) \geq M(x-x^#) \) for all \( u \in U \).

Condition 2 guarantees the feasibility of the first OCP which implies the feasibility of all subsequent problems. It is often ensured by selecting a sufficient large horizon \( T \).

Condition (SCa) is similar to a decrease condition of a Lyapunov function. (Note that the pointwise version of this condition \( \nabla G(x(s;x_f,k_f)) \cdot f(x(s;x_f,k_f),k_f) \leq -L(x(s;x_f,k_f),k_f) \) for a.e. \( s \in [0,\delta] \), immediately implies (SCa) after integration on \([0,\delta])\). However, it can be much easier to verify on a smaller set \( X_f \) than globally.

Condition (SCb) is an invariance condition. After the end of the horizon, the trajectory under the feedback \( \kappa_f \) has to remain within \( X \) and return to \( X_f \) after time
δ. Alternatively, if the set X f is invariant under κ f, i.e., x(s; x f, k) \in X, for all s \in [0, δ], then this condition can replace both conditions in (SCb) since X f \subset X.

Condition (SCc) specifies the admissible controls. Note that from the definition of U it implies that k f(s, x f) \in U a.e. s \in [0, δ].

The stability result making use of this condition follows.

**Theorem 7.1.** Assume the system satisfies hypotheses H1-H3. If the design parameters T, L, G and X f satisfy the stability condition SC, then applying the AMR-MPC strategy starting from any x 0 \in X 0 and with some initial mesh π 0 ⊃ Π we have:

1. all optimal control problems involved in the AMR-MPC strategy, P(x k) for all k \geq 0, are feasible and have a minimum.
2. the closed-loop trajectory x* is asymptotically attracted to the origin, that is x*(t) \to 0 as t \to +\infty.

**Remark 2.** We note that this result is just a nominal stability result and, as such, the trajectory x* is the closed-loop trajectory obtained numerically by applying the AMR-MPC strategy. That is, it is assumed that the evolution of the plant trajectory follows exactly the behavior described by the model and is computed using the described numerical scheme.

### 7.1. Proof of the stability result.
This theorem is based, with changes related to AMR, on arguments to establish stability of sampled-data MPC in [8] and [17, Thm. 3]. In what follows we establish, respectively, recursive feasibility, existence of an optimal solution, uniform decrease of the value function, and asymptotical convergence.

**Lemma 7.2** (Recursive feasibility). Let x 0 \in X 0 and x k = x(kδ) for k \in \mathbb{N} when the nominal model is followed and the AMR-MPC strategy is applied starting with initial mesh π 0 ⊃ (Π \cap [0, T]). Then, P(x k) is feasible for all k \in \mathbb{N}. Moreover, an optimal solution to P(x k) exists.

**Proof.** For the initial MPC iteration, k = 0, t = 0, the initial state x(0) = x 0 is selected from X 0 and X 0 \subset A 0 when π 0 is used. By definition of A 0, problem P(x 0) is feasible.

Let the pair (\bar{x}, \bar{u}), defined for s \in [0, T], be an optimal solution to P(x 0). Then, assuming the nominal model is followed, we have that, for iteration k = 1, t = δ, the new initial state is x 1 = \bar{x}(δ).

We now show that P(x 1) is also feasible. This is accomplished by defining a pair (\bar{x}, \bar{u}), also in the interval s \in [0, T], which is guaranteed to be admissible if constructed as an extension of (\bar{x}, \bar{u}) to the interval t \in [t_k + T, t_k + T + δ] as follows.

\[
\tilde{u}(s) = \begin{cases} 
\bar{u}(s + \delta) & \text{if } s \in [0, T - \delta) \\
\kappa_f(s - T + \delta; \bar{x}(T)) & \text{if } s \in [T - \delta, T] 
\end{cases}
\]

\[
\tilde{x}(s) = \begin{cases} 
\bar{x}(s + \delta) & \text{if } s \in [0, T - \delta) \\
x(s - T + \delta; \bar{x}(T), \kappa_f(s - T + \delta, \bar{x}(T))) & \text{if } s \in [T - \delta, T],
\end{cases}
\]

where κ f is the control law in SC. See Fig. 4.

The AMR procedure is such that the time-mesh nodes in [δ, T] of a certain iteration k are maintained in the next iteration k + 1 in [0, T - δ] (i.e. no mesh nodes are removed in our AMR and all meshes contain the sampling instants in the interval [0, T], π r ⊃ ... ⊃ π 0 ⊃ (Π \cap [0, T])). The control \tilde{u} can be constructed in the new mesh. Also, from (SCc) we have that \tilde{u}(s) \in U for s \in [T - δ, T]. Therefore, \tilde{u}
Figure 4. Construction of the (extended) admissible control \( \tilde{u} \) with \( \Pi = \{ t_k \}_{k \in \mathbb{N}}, t_k = k\delta \), and with \( \pi_r = \{ s_i \}_{i \in 0, 1, \ldots, N_r}, s_i = i\delta/2 \).

is an admissible control. Moreover, (SCb) guarantees that \( \tilde{x}(s) \in X_f \) and \( \tilde{x}(s) \in X \) for \( s \in [T - \delta, T] \). Hence, the pair \( (\tilde{x}, \tilde{u}) \) is admissible for \( \mathcal{P}(x_1) \). Problem \( \mathcal{P}(x_1) \) is feasible.

Since the problems are time-invariant, our construction ensures that after a feasible problem \( \mathcal{P}(x_k) \) at MPC iteration \( k \), it follows another feasible OCP \( \mathcal{P}(x_{k+1}) \) at iteration \( k + 1 \). Using an induction argument, we show that all OCP’s for \( k \geq 0 \) are feasible.

To show the existence of an optimal solution to the discretized \( \mathcal{P}(x_k) \), we have to show existence of an optimal solution of the equivalent NLP. That is, by the Weierstrass extreme value theorem, the minimum exists if in the NLP (i) the objective function is continuous; (ii) the domain is nonempty; (iii) the domain is compact. Condition (i) is guaranteed by the continuity of \( L \) and \( G \). Condition (ii) holds from the feasibility of the OCP established above. Condition (iii) is implied by the compactness of \( U, X \) and that its images by a continuous function \( f \) are also compact. Therefore, the NLP and thereby the OCP has a minimum.

A key step to show the convergence of the trajectory to the origin is to establish that the value function of the optimal control problems decreases uniformly along the sequence \( \{ x_k \}_{k \in \mathbb{N}} \). This, in turn, can be shown using SC, mainly with conditions (SCa),(SCb), and (SCc).
We define the cost functional and value function of our OCP as follows. The cost functional is defined to be
\[ J(x_0, u) := \int_0^T L(x(s; x_0, u), u) ds + G(x(T; x_0, u)), \]
if the pair \((x(\cdot; x_0, u), u)\) is admissible. Otherwise
\[ J(x_0, u) := +\infty. \]

Having guaranteed the existence of an optimal solution (of the discretized OCP), the value function is
\[ V(x_0) = \min_u J(x_0, u) = J(x_0, \bar{u}). \]

We have the following lemma.

**Lemma 7.3 (Uniform decrease of the value function).** For all \(k \in \mathbb{N}\)
\[ V(x_{k+1}) - V(x_k) \leq -\int_{t_k}^{t_{k+\delta}} M(x^*(s)) ds. \]

**Proof.** The value function at \(x_0\) is
\[ V(x_0) = \int_0^T L(\bar{x}(s)), \bar{u}(s)) ds + G(\bar{x}(T)). \]
Since the pair \((\bar{x}, \bar{u})\) is admissible for \(P(x_1)\), but not necessarily optimal, we have
\[ V(x_1) \leq J(x_1, \bar{u}). \]

We also have
\[ J(x_1, \bar{u}) = \int_0^T L(\bar{x}(t)), \bar{u}(t)) dt + G(\bar{x}(T)) \]
\[ = \int_0^T L(\bar{x}(t)), \bar{u}(t)) dt + \int_0^{T+\delta} L(x(t; \bar{x}(T), \kappa_f)), \kappa_f(t - T, \bar{x}(T))) dt \]
\[ + G(x(\delta; \bar{x}(T), \kappa_f). \]

Computing the difference, we obtain
\[ V(x_1) - V(x_0) \leq J(x_1, \bar{u}) - V(x_0) \]
\[ \leq -\int_0^{\delta} L(\bar{x}(t)), \bar{u}(t)) dt + \int_0^{T+\delta} L(x(t; \bar{x}(T), \kappa_f(t - T, \bar{x}(T)))) \]
\[ + G(x(\delta; \bar{x}(T), \kappa_f) - G(\bar{x}(T)) \]
\[ \leq -\int_0^{\delta} L(\bar{x}(t)), \bar{u}(t)) dt, \]
where in the last inequality we have used (SCa). Finally, recalling the condition on \(M\) in \(SC\) we have
\[ V(x_1) - V(x_0) \leq -\int_0^{\delta} M(\bar{x}(t)) dt. \]

Since the OCP’s are time-invariant and \(x^*\) is the concatenation of the initial part of the optimal trajectories of the sequence of OCP’s, we have
\[ V(x_{k+1}) - V(x_k) \leq -\int_0^{\delta} M(x^*(t_k + s)) ds \]
\[ \square \]
From the previous lemma, we can then write that for any $k \geq 0$

$$0 \leq V(x_k) \leq V(x_0) - \int_{t_0}^{t_k} M(x^*(t)) dt.$$ 

Since $V(x_0)$ is finite, we conclude that the function $k \mapsto V(x_k)$ is bounded and then that $k \mapsto \int_{t_0}^{t_k} M(x^*(k\delta)) dt$ is also bounded. Therefore $k \mapsto x^*(k\delta)$ is bounded and, since $f$ is continuous and takes values on bounded sets of $(x, u)$, $t \mapsto x^*(t)$ and $t \mapsto x^*(t)$ are also bounded. All the conditions to apply Barbalat’s lemma ([45, 20]) are met, yielding that the trajectory asymptotically converges to the origin.

Note that we have just confirmed what is usually known in the literature as attractiveness (i.e. $x^*(t) \to 0$ as $t \to +\infty$). Stability in the sense of Lyapunov is not established here. It would be achieved if $V$ is continuous, for example. However, in the application explored below, a car-like system with minimum turning radius $R$, it is not possible to attain stability in the sense of Lyapunov. To see this, note that when the system is at the target point, an arbitrarily small perturbation in $y$ would require a significantly long path (with length bounded below) to return to the target.

8. Application. To test and to validate the proposed algorithm, a problem involving parking manoeuvres in a constrained space is solved. Consider a car-like system with a minimum turning radius $r_{\min}$. At a given time instant $s$, we define the state $x(s) = (x(s), y(s), \psi(s), v(s))$ and the control $u(s) = (a(s), c(s))$ where $(x(s), y(s))$ is the position of the midpoint of the axle connecting the rear wheels, $\psi(s)$ is the heading angle, $v(s)$ is the speed, $a(s)$ is the acceleration and $c(s)$ is the curvature (see Figure 5). The objective is to drive a car-like vehicle from an initial position $x_0$ to a target point $x^\#$ in a constrained environment. This constrained environment, implemented as a pathwise state constraint, represents a parking lot where the vehicle is parked and the target area is the exit (see Figure 6).

We have chosen this application to show how MPC can overcome nonholonomy challenges ([6, 30]) since it involves planning, not just reactive control; it can generate required nonlinear, discontinuous feedback; and it illustrates that the sampled-data MPC framework combines well with sampling-feedbacks [18]. MPC can also overcome constraints challenges since it is known to be a (if not the main) technique to deal appropriately with constraints.

The optimal control problem (P_CP) for this system on $s \in [0, T]$ is written as follows:

$$\text{Minimise} \int_0^T L(x(s), u(s)) \, ds + G(x(T))$$

subject to

(i) dynamic constraints

$$\dot{x}(s) = v(s) \cos(\psi(s)) \quad \text{a.e. } s \in [0, T]$$
$$\dot{y}(s) = v(s) \sin(\psi(s)) \quad \text{a.e. } s \in [0, T]$$
$$\dot{\psi}(s) = v(s) c(s) \quad \text{a.e. } s \in [0, T]$$
$$\dot{v}(s) = a(s) \quad \text{a.e. } s \in [0, T]$$
(ii) input constraints
\[ a_{\text{min}} \leq a(s) \leq a_{\text{max}} \quad \text{a.e. } s \in [0, T] \]
\[ -1/r_{\text{min}} \leq c(s) \leq 1/r_{\text{min}} \quad \text{a.e. } s \in [0, T] \]

(iii) the initial state constraint
\[ x(0) = x_0 = (x_0, y_0, \psi_0, v_0) \quad (10) \]

(iv) the terminal constraint
\[ x(T) \in X_f, \quad (11) \]

and

(v) the path–wise state constraint
\[ x(s) \in X \quad \forall s \in [0, T]. \quad (12) \]

The state constraint set \( X \) is defined to be the set of states \( x(s) \) satisfying
\[
\begin{cases}
-2M \leq y(s) \leq M & \text{if } x(s) \in [x_0, x^\dagger] \\
-b(s, x) \leq y(s) \leq b(s, x) & \text{if } x(s) \in [x^\dagger, x^\#] \\
-m \leq y(t) \leq m & \text{if } x(s) \in [x^\dagger, x_f]
\end{cases}
\quad (13)
\]

where
\[ b(s, x) = y^\dagger - \sqrt{\rho^2 - (x(s) - x^\dagger)^2}, \quad \rho = |x^\dagger - x^\#|. \]

and \( x^\# = (x^\#, y^\#, \psi^\#, v^\#) \) is the target point, which is an equilibrium state with \( u = 0 \).

The design parameters \( L, G, X_f \) and \( T \) are selected in such a way that the stability condition \( SC \) is satisfied, guaranteeing convergence to the equilibrium point \( x^\# \).

Note that we are not changing the coordinates to make the target state coincide with the origin (see Remark 1).

The terminal set \( X_f \) is the set of points \((x, y)\) in the \( x \) axis, heading towards \( x^\# \) with positive speed (see Figure 6), defined to be
\[ X_f = \{(x, y, \psi, v) : x - x^\# \leq 0 \land y - y^\# = 0 \land \psi = 0 \land v \geq 0\}. \]
The running cost is simply a quadratic cost

\[ L(x(t), u(t)) = (x - x^#)^2 + (y - y^#)^2 + (\psi - \psi^#)^2 + (v - v^#)^2 + \alpha(a^2 + c^2), \]

with the control effort weight \( \alpha \geq 0 \). The terminal cost is the cost to reach the target point in time \( \bar{t} \) from a point in \( X_f \) maintaining constant speed (of at least \( v_{\text{min}} > 0 \) and zero curvature, that is

\[ G(x(T)) = \int_0^\bar{t} L(x(s), u(s)) \, ds, \]

for which an explicit formula is

\[ G(x(T)) = G(x_T, y_T, \psi_T, v_T) = \frac{|x_T - x^#|}{\max\{v_T, v_{\text{min}}\}} \left( \frac{1}{3} (x_T - x^#)^2 + (v_T - v^#)^2 \right). \]

This formula is obtained noting that with \( u(s) = (a(s), c(s)) = (0, 0) \), the time to reach the origin is

\[ \bar{t} = \frac{x_f - x_T}{v_T}, \]

and the trajectory is for \( t \in [0, \bar{t}] \)

\[ v(s) = v_T \]
\[ \psi(s) = 0 \]
\[ y(s) = y_f \]
\[ x(s) = x_T + v_T s. \]
Finally, selecting the horizon $T$ to satisfy
\[
T \geq \max \left\{ \sqrt{\frac{4\pi r_{\min}}{a_{\max}}}, \sqrt{\frac{4M}{a_{\max}}} \right\},
\]
it can be verified that the stability condition $\text{SC}$ holds (see [17, Section 4.3] for details of the verification for a similar system).

8.1. **Numerical implementation.** To test the proposed algorithm, we have implemented it in MATLAB R2016a combined with optimal control solver ICLOCS – version 0.1b ([1]) using IPOPT as the NLP solver ([47]). The problems were solved in a computer with an Intel™ Core® i7-4770K CPU @3.50 GHz.

We consider the target point \( x^# = (4, 0, 0, 0) \),

the input constraints
\[
-5 \leq a(s) \leq 5 \quad \forall s \in [0, 10]
\]
\[-0.9 \leq c(s) \leq 0.9 \quad \forall s \in [0, 10]
\]
the end–point constraints
\[
x(0) = (0.5, -3, -\pi/2, 0) \quad (14)
\]
\[X_f = \{(x, 0, 0, v) : x - 4 \leq 0, v \geq 0\} \quad (15)
\]
the pathwise state constraint
\[
\begin{cases}
-4 \leq y(s) \leq 2 & \text{if } x(s) \in [0, 2] \\
-b(s, x(s)) \leq y(s) \leq b(s, x(s)) & \text{if } x(s) \in [2, 3] \\
-0.5 \leq y(s) \leq 0.5 & \text{if } x(s) \in [3, 4]
\end{cases} \quad (16)
\]
where
\[
\rho = 1, \quad b(s, x(s)) = -1.5 - \sqrt{1 - (x(s) - 3)^2}.
\]
We consider the horizon $T = 10$, the sampling period $\delta = 2$ s, and $v_{\text{min}} = 0.1$.

8.2. **Numerical results for the AMR–OCP.** We start by discussing the results obtained when solving optimal control problem (P$_{\text{CP}}$) using the AMR procedure.

We consider as stopping criterion the bound on the estimate of the trajectory discretization error
\[
\varepsilon_{\text{x}}^{\text{max}} = 1 \times 10^{-5},
\]
and as refinement criteria for the different levels, we consider the vector
\[
\varepsilon_{\text{x}} = [1, 5, 10, 10^2] \varepsilon_{\text{x}}^{\text{max}}.
\]
By the AMR algorithm, each time interval of refining level $j$, is subdivided into $N_j$ subintervals by adding $N_j^2 - 1$ additional nodes. We consider $N = 3$. The problem is solved in the following three meshes:

\( a) \) $\pi_0$ the initial coarse mesh with 21 equidistant nodes;
\( b) \) $\pi_{\text{AMR}}$ obtained by the multi–level time–mesh refinement strategy;
\( c) \) $\pi_F$ considering equidistant nodes with the lowest time step of $\pi_{\text{AMR}}$. 
The problem is first solved using the initial coarse mesh $\pi_0$. In Fig. 7, we can see the path obtained in this initial solution. The car starts at position $(0.5, -3)$, heading downwards. It moves in reverse until reaching the point of maximum $y$, then moves forward turning left until attaining the target.

When we analyze the discretization error, see Fig. 8, we verify that it is above $\varepsilon_1 = \varepsilon_{\text{max}}^x$ between the 2nd and 8th nodes, and it is slightly above $\varepsilon_3$ at node 6. Therefore, we apply refining level 3 in the two intervals adjacent to node 6, dividing each into $N^3 = 27$ subintervals and we apply refining level 1 in the remaining 4 intervals between the 2nd and 8th node, diving each one of those into 3 subintervals. In total, the refining procedure adds 60 new nodes, $4*(N-1)+2*(N^3-1)$, forming a new mesh $\pi_1$ with 82 nodes (one additional node is added by the procedure to compensate for truncating each interval size to maximum precision) and with the smallest subinterval of dimension 1/54 seconds.

Table 1 shows for each mesh the number of nodes, the smallest time step of each mesh, the number of iterations needed to solve the NLP problem, the maximum absolute local errors of the trajectory, and CPU times for solving the OCP problem and for computing the local error as well.

The OCP is then solved in the new mesh $\pi_1$. Despite the significant increase in the number of mesh nodes, the optimizer takes about the same number iterations (coincidentally exactly the same) and similar time to obtain a solution when compared to the first run using $\pi_0$ (see Table 1). This is because the previous solution was used as a warm start of the optimizer.

The discretization error estimate is again analyzed and it is verified that it is already below the threshold $\varepsilon_{\text{max}}^x$. Therefore, there is no need for further refining iterations and the AMR algorithms terminates with final mesh $\pi_{\text{AMR}}$ equal to $\pi_1$. In Table 1, the times and number of iterations of the optimizer for $\pi_{\text{AMR}}$ are reported as the sum of the values obtained in all previous iterations.

Fig. 9 depicts the optimal path computed in the final mesh $\pi_{\text{AMR}}$ and Fig. 10 plots the trajectory and the associated control. We can see that the $\pi_{\text{AMR}}$ mesh has higher resolution precisely in the time subintervals where the discretization error was higher in the first step. These time intervals coincide with the ones where the speed attains a higher absolute value.

Table 1. Results for problem (P_{CP}) solved in each time-mesh.

| $\pi_j$ | $N_j$ | $\Delta t_j$ | $I_j$ | $\|x^{(j)}\|_{\infty}$ | CPU time (s) |
|---------|-------|--------------|-------|-----------------|--------------|
| $\pi_0$ | 21    | 0.5          | 42    | 1.0016E-4       | 0.9816       |
| $\pi_1$ | 82    | 1/54         | 42    | 3.3801E-7       | 0.7061       |
| $\pi_{\text{AMR}}$ | 82 | 1/54 | 84 | 3.3801E-7 | 1.6877 |
| $\pi_F$ | 541   | 1/54         | 403   | 4.0358E-7       | 13.2473      |

To compare the performance of AMR strategy for a similar level of accuracy, we tried to solve the OCP in a mesh $\pi_F$ which is an equidistant mesh with nodes spaced with the smallest time step of $\pi_{\text{AMR}}$, which is 1/54 seconds. The mesh obtained this way has 541 nodes. With this mesh, we could not obtain a solution from the optimizer within the maximum of 5000 iterations of the NLP when cold started.
Figure 7. Optimal path computed in the initial coarse mesh.

(with no meaningful initial guess) with the initial state \( x(0) = (0.5, -3, -\pi/2, 0) \). We considered then the problem with initial state \( x(0) = (0.5, -3, \pi/4, 0.6) \), which is considerably simpler problem since the car starts already moving towards the origin. The results from this run are reported in the last line of Table 1.

Despite the fact that a simpler problem is being solved and as, as expected, the maximum absolute local errors are of the same order of magnitude, it took considerably more time and more NLP iterations to obtain a solution in the equidistant fine mesh \( \pi_F \). The mesh \( \pi_{AMR} \) has about 15% of the nodes of \( \pi_F \), and computing the solution using \( \pi_{AMR} \) takes only about 12% of the time needed to get a solution using \( \pi_F \), causing significant savings in memory and computational cost.

Moreover, the fact that the AMR algorithm was able to solve a problem that could not be solved in an equidistant fine mesh shows evidence of its robustness.

8.3. Numerical results for the AMR–MPC. To implement the AMR–MPC scheme, we consider a sampling period of \( \delta = 2 \) seconds and we consider \( N = 2 \) subintervals in each refinement. We also set the discretization error estimate threshold to

\[ \varepsilon_{x}^{\text{max}} = 1 \times 10^{-5}, \]
and we consider three levels of refinement (the tightest level on the initial 20% part of the horizon, coinciding with the sampling period, the second level on the following 20%, and the third level on the remaining part of the horizon) with the following limits

\[ \bar{\varepsilon}_x(t) = \begin{cases} 
\varepsilon_{max}, & t \in [t_k, t_k + 0.2T] \\
5 \times \varepsilon_{max}, & t \in [t_k + 0.2T, t_k + 0.4T] \\
10 \times \varepsilon_{max}, & t \in [t_k + 0.4T, t_k + T] 
\end{cases} \]

Before entering the MPC loop, the OCP is solved in the initial coarse mesh \( \pi_0 \). For the solution obtained in such mesh, we plot the discretization error against the new time–varying threshold, Fig. 11, and we can see where the refinement is expected to occur in the first MPC iterations.

We apply the AMR–MPC scheme. Table 2 shows for each MPC iteration and each AMR mesh the information about the number of nodes, the smallest time step, the number of iterations needed to solve the NLP problem, the maximum absolute local errors of the trajectory, and computational times for solving the OCP problem and for computing the local error. We observe that the MPC iterations are considerably faster than the initial cold–started OCP. The refinement procedure is only activated in the first MPC iteration. The second MPC iteration still uses part of a refined mesh, but after that the algorithm detects that there is no need to refine and uses a coarse mesh. All error estimates are well below the pre–defined threshold \( \varepsilon_{max} \). Figure 12 shows that the path resulting from the AMR–MPC scheme is equal or very similar to the path obtained when solving the open–loop OCP.

We also observe in Fig. 13 that the solution to the several OCP’s overlap (different colors are used for the solution to each OCP; the slight difference in the interval from 2 to 4 seconds, corresponding to the 2nd MPC step, is due to different refinement thresholds between the 1st and 2nd MPC step). This is expected by the choice of objective functions we have done. Since the terminal cost is the cost to go...
till the target and after that the cost is zero, we have that

\[ W(x(T)) = \int_{T}^{T+i} L(x(s), u(s)) \, ds = \int_{T}^{\infty} L(x(s), u(s)) \, ds. \]

Hence, the solution to each OCP coincides in its horizon with the solution to the OCP with infinite horizon objective function with the terminal constraint set imposed also at \( t = T \)

\[ \int_{0}^{\infty} L(x(s), u(s)) \, ds. \]

9. **Conclusions.** In this article, we develop an extended adaptive time–mesh refinement (AMR) algorithm to solve open-loop optimal control problems (OCP). The algorithm provides local mesh resolution refining only where it is required, thereby guaranteeing accuracy of the procedure in an efficient way. In the extension proposed, we consider a time–dependent stopping criterion for the mesh refinement algorithm with different levels. The extended algorithm is particularly useful to solve the OCP’s involved in MPC schemes. Such use is explored in this article. We propose an MPC framework incorporating the extended AMR algorithm. We show that stability of the system resulting from applying the AMR–MPC strategy can be guaranteed.
The extended AMR and the AMR–MPC algorithms are tested in a car-like system with limited turning radius, performing manoeuvres in a constrained space. In the end, the OCP’s are solved within MPC with an adapted mesh with local mesh resolution, which has less nodes in the overall procedure, yet having maximum absolute local error of the same order of magnitude when compared against a refined mesh with equidistant–spacing. The fast response of the algorithm, even with tight accuracy limits, shows its ability to solve complex problems in real–time. The application demonstrates the advantage of the proposed adaptive mesh strategy, which leads to results obtained approximately as fast as the ones given.
by a coarse equidistant–spacing mesh and as accurate as the ones given by a fine equidistant–spacing mesh.

This framework allows to use continuous–time plant models directly: the discretization procedure is automated and there is no need to select a priori an adequate discretization time–step.

Another advantage of the use of the AMR–MPC is that even if the optimization procedure is forced to stop in an early stage, as might be the case in real–time, it can still provide a solution, although it might be a less accurate one.

The AMR–MPC strategy proposed here might also be useful in situations that are not explored here but are worthy of further research. These include systems with
multiple time-scales in the dynamics, impulsive systems, or the stability analysis for MPC with economic cost functions.

Acknowledgments. In this research, we acknowledge the support of FEDER/-COMPETE2020/NORTE2020/POCI/PIDDAC/MCTES/FCT funds through grants SFRH/BPD/126683/2016, UID/IEEA/00147/006933–SYSTEC, PTDC-EEI-AUT-2933-2014|16858–TOCCATA, and POCI-01-0145-FEDER-031447|FCT–UPWIND.

REFERENCES

[1] ICLOCS2: A MATLAB toolbox for optimization based control, URL http://www.ee.ic.ac.uk/ICLOCS/.

[2] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings and M. Diehl, CasADi – A software framework for nonlinear optimization and optimal control, *Mathematical Programming Computation*, (2018), 1–36.

[3] J. T. Betts, *Practical Methods for Optimal Control Using Nonlinear Programming*, SIAM, 2001.

[4] J. T. Betts and W. P. Huffman, Mesh refinement in direct transcription methods for optimal control, *Optimal Control Applications and Methods*, 19 (1998), 1–21.
Figure 13. Trajectory and control resulting from the AMR–MPC scheme

[5] J. Frederic Bonnans, D. Giorgi, V. Grelard, B. Heymann, S. Maindrault, P. Martinon, O. Tis- 
sot and J. Liu, Bocop – A Collection of Examples, Technical report, INRIA, 2017, URL http://www.bocop.org.
[6] R. W. Brockett, Asymptotic stability and feedback stabilization, in Differential Geometric
Control Theory (eds. R. W. Brockett, R. S. Millman and H. S. Sussmann), Birkhouser, Boston,
27 (1983), 181–191.
[7] A. Caldeira and F. Fontes, Model predictive control for path-following of nonholonomic sys-

tems, in Proceedings of the 10th Portuguese Conference in Automatic Control (ed. IFAC),
2010, 374–379.
[8] H. Chen and F. Allgöwer, A quasi-infinite horizon nonlinear model predictive control scheme
with guaranteed stability, Automatica, 34 (1998), 1205–1217.
[9] F. H. Clarke, Y. S. Ledyaev, E. D. Sontag and A. I. Subbotin, Asymptotic controllability implies feedback stabilization, IEEE Transactions on Automatic Control, 42 (1997), 1394–1407.

[10] D. M. de la Peña and D. Limón (eds.), IFAC-PapersOnLine | 5th IFAC Conference on Nonlinear Model Predictive Control | NMPC 2015 - Seville, Spain, 17–20 September 2015 | ScienceDirect.com, vol. 48, 2015, URL http://www.sciencedirect.com/journal/ifac-papersonline/vol/48/issue/23.

[11] M. Diehl, H. G. Bock, J. P. Schlöder, R. Findeisen, R. Nagy and F. Allgöwer, Real-time optimization and nonlinear model predictive control of processes governed by differential-algebraic equations, Journal of Process Control, 12 (2002), 577–585.

[12] D. Dochain, D. Henrion and D. Peaucelle (eds.), IFAC-PapersOnLine | 20th IFAC World Congress | ScienceDirect.com, vol. 50, 2017, URL https://www.sciencedirect.com/journal/ifac-papersonline/vol/50/issue/1.

[13] P. Falugi, E. Kerrigan and E. van Wyk, Imperial college london optimal control software: User guide, 2010, URL http://www.ee.ic.ac.uk/ICLOCS/user_guide.pdf, Imperial College London, London, England.

[14] F. A. C. C. Fontes, Discontinuous feedback stabilization using nonlinear model predictive controllers, in Proceedings of CDC 2000 – 39th IEEE Conference on Decision and Control, vol. 5, IEEE, Sydney, Australia, 2000, 4969–4971.

[15] F. A. C. C. Fontes, A general framework to design stabilizing nonlinear model predictive controllers, Systems & Control Letters, 42 (2001), 127–143.

[16] F. A. C. C. Fontes, Discontinuous feedbacks, discontinuous optimal controls, and continuous-time model predictive control, International Journal of Robust and Nonlinear Control, 13 (2003), 191–209.

[17] F. A. C. C. Fontes and L. Magni, Min-max model predictive control of nonlinear systems using discontinuous feedbacks, IEEE Transactions on Automatic Control, 48 (2003), 1750–1755.

[18] F. A. C. C. Fontes and L. Magni, A generalization of Barbalat’s lemma with applications to robust model predictive control, in Proceedings of Sixteenth International Symposium on Mathematical Theory of Networks and Systems (MTNS2004), Leuven, Belgium July 5-9, vol. 4, 2004.

[19] F. A. C. C. Fontes and H. Frankowska, Normality and nondegeneracy for optimal control problems with state constraints, Journal of Optimization Theory and Applications, 166 (2015), 115–136.

[20] F. A. C. C. Fontes, L. Magni and E. Gyurkovics, Sampled-Data Model Predictive Control for Nonlinear Time-Varying Systems: Stability and Robustness, in Assessment and Future Directions of Nonlinear Model Predictive Control (eds. D.-I. R. F. e. F. Allgöwer and P. D. L. T. Biegler), no. 358 in Lecture Notes in Control and Information Sciences, Springer Berlin Heidelberg, 2007, 115–129.

[21] F. A. C. C. Fontes and F. L. Pereira, Model predictive control of impulsive dynamical systems, in Nonlinear Model Predictive Control, 45 (2012), 305–310.

[22] F. A. C. C. Fontes and L. T. Paiva, Guaranteed constraint satisfaction in continuous-time control problems, IEEE Control Systems Letters, 3 (2019), 13–18.

[23] M. Gerds, Optimal Control of ODEs and DAEs, De Gruyter, Berlin, Boston, 2012, URL https://www.degruyter.com/view/product/119403.

[24] L. Grüne and V. G. Palma, Robustness of performance and stability for multistep and updated multistep MPC schemes, Discrete and Continuous Dynamical Systems, 35 (2015), 4385–4414.

[25] L. Grüne, D. Nesci and J. Pannek, Model predictive control for nonlinear sampled-data systems, in Assessment and Future Directions of Nonlinear Model Predictive Control (NMPC05) (ed. R. F. e. F. Allgöwer L. Biegler), vol. 358 of Lecture Notes in Control and Information Sciences, Springer Verlag, Heidelberg, 358 (2007), 105–113.

[26] L. Grüne and J. Pannek, Nonlinear Model Predictive Control, Springer, 2011.
[29] B. Houska, H. J. Ferreau and M. Diehl, ACADO toolkit—An open-source framework for automatic control and dynamic optimization, *Optimal Control Applications and Methods*, 32 (2011), 298–312.

[30] I. Kolmanovsky and N. McClamroch, Developments in nonholonomic control problems, *IEEE Control Systems*, 15 (1995), 20–36.

[31] M. Lazar, F. Allgower, P. M. Van den Hof and B. Cott (eds.), *4th IFAC Conference on Nonlinear Model Predictive Control, NMPC 12*, IFAC Proceedings Volumes (IFAC-PapersOnline), IFAC, Noordwijkerhout; Netherlands, 2012.

[32] L. Magni and R. Scattolini, Model predictive control of continuous-time nonlinear systems with piecewise constant control, *IEEE Transactions on Automatic Control*, 49 (2004), 900–906.

[33] D. Q. Mayne, J. B. Rawlings, C. V. Rao and P. O. M. Scokaert, Constrained model predictive control: Stability and optimality, *Automatica*, 36 (2000), 789–814.

[34] D. Mayne and H. Michalska, Receding horizon control of nonlinear systems, *IEEE Transactions on Automatic Control*, 35 (1990), 814–824.

[35] H. Michalska and D. Mayne, Robust receding horizon control of constrained nonlinear systems, *IEEE Transactions on Automatic Control*, 38 (1993), 1623–1633.

[36] L. T. Paiva and F. A. C. C. Fontes, A sufficient condition for stability of sampled-data model predictive control using adaptive time–mesh refinement, in *Proceedings of NMPC 2018–6th IFAC International Conference on Nonlinear Model Predictive Control*, Madison, WI, USA, August 2018 (ed. IFAC), 51 (2018), 104–109.

[37] L. T. Paiva and F. A. Fontes, Sampled-data model predictive control using adaptive time–mesh refinement algorithms, in *CONTROLO 2016: Proceedings of the 12th Portuguese Conference on Automatic Control*, 402 (2016), 143–153.

[38] L. T. Paiva and F. A. C. C. Fontes, Adaptive time-mesh refinement in optimal control problems with state constraints, *Discrete and Continuous Dynamical Systems*, 35 (2015), 4553–4572.

[39] G. Pannocchia, J. Rawlings, D. Mayne and G. Mancuso, Whither Discrete Time Model Predictive Control?, *IEEE Transactions on Automatic Control*, 60 (2015), 246–252.

[40] M. A. Patterson, W. W. Hager and A. V. Rao, A ph mesh refinement method for optimal control, *Optimal Control Applications and Methods*, 36 (2015), 398–421.

[41] I. Prodan, S. Olaru, F. A. C. C. Fontes, F. L. Pereira, J. B. d. Sousa, C. S. Maniu and S.-I. Niculescu, Predictive control for path-following, from trajectory generation to the parametrization of the discrete tracking sequences, in *Developments in Model-Based Optimization and Control*, Lecture Notes in Control and Information Sciences, Springer, Cham, 2015, 161–181.

[42] I. Prodan, S. Olaru, F. A. Fontes, C. Stoica and S.-I. Niculescu, A predictive control-based algorithm for path following of autonomous aerial vehicles, in *Control Applications (CCA), 2013 IEEE International Conference on*, IEEE, 2013, 1042–1047.

[43] J. B. Rawlings and D. Q. Mayne, *Model Predictive Control: Theory and Design*, Nob Hill Pub., 2009.

[44] A. Rucco, A. P. Aguiar, F. A. Fontes, F. L. Pereira and J. B. de Sousa, A model predictive control-based architecture for cooperative path-following of multiple unmanned aerial vehicles, in *Developments in Model-Based Optimization and Control*, Springer, 464 (2015), 141–160.

[45] J.-J. E. Slotine and W. Li, *Applied nonlinear control*, Prentice Hall, New York, 1991.

[46] R. B. Vinter, *Optimal Control*, Birkhauser, Boston, 2000.

[47] A. Wächter and L. T. Biegler, On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming, *Mathematical Programming*, 106 (2006), 25–57.

[48] Y. Zhao and P. Tsiotras, Density functions for mesh refinement in numerical optimal control, *Journal of Guidance, Control, and Dynamics*, 34 (2011), 271–277.

Received January 2018; revised January 2019.

*E-mail address: ltpaiva@fe.up.pt*  
*E-mail address: faf@fe.up.pt*