

Minerva Access is the Institutional Repository of The University of Melbourne

Author/s:

Lee, K;Moase, WH;Manzie, C

Title:

Mesh adaptation in direct collocated nonlinear model predictive control

Date:

2018-10-01

Citation:

Lee, K., Moase, W. H. & Manzie, C. (2018). Mesh adaptation in direct collocated nonlinear model predictive control. *International Journal of Robust and Nonlinear Control*, 28 (15), pp.4624-4634. <https://doi.org/10.1002/rnc.4235>.

Persistent Link:

<https://hdl.handle.net/11343/284056>

Mesh adaptation in direct collocated nonlinear model predictive control

K. Lee*, W. H. Moase and C. Manzie

The University of Melbourne, Victoria, Australia

SUMMARY

Direct methods are often deployed to solve nonlinear model predictive control problems where the optimal control problem is first transcribed into a nonlinear program and then solved to obtain the control input. This makes the computational cost of direct methods nontrivial, however efficiencies can be gained by utilising adaptation methods during transcription. Goal-oriented *a priori* error estimation is used as an adaptation strategy. Unlike other strategies, the refinement is directly related to the cost function. Therefore refinement only occurs only where it is needed with respect to the cost function. Two examples are presented and an improvement of up to 50% in the computational time is observed with no degradation in closed loop performance.

Copyright © 2017 John Wiley & Sons, Ltd.

Received ...

KEY WORDS: nonlinear model predictive control; adaptive mesh generation; direct collocation

1. INTRODUCTION

Nonlinear model predictive control (NMPC) is a feedback control method based on iteratively solving an optimal control problem (OCP) over a finite horizon utilising nonlinear process models and/or enforcing nonlinear constraints [1]. Linear model predictive control (MPC), where only linear models and constraints can be implemented, has proven popular in the past [2], in particular because the resulting optimal control problem can be solved efficiently online. However, due to increasing demands in process productivity, stricter environmental regulations, economic needs, and so on, this motivates the use of NMPC over MPC even though there are higher computational requirements.

A class of solution methods for nonlinear model predictive control are based on direct methods where the OCP is transcribed into a nonlinear program (NLP). Transcription via direct methods is categorised into single shooting [3], multiple shooting [4] and collocation methods [5]. These

This is the author manuscript accepted for publication and has undergone full peer review but has not been through the copyediting, typesetting, pagination and proofreading process, which may lead to differences between this version and the Version of Record. Please cite this article as doi: 10.1002/rnc.4235

*Correspondence to: Department of Mechanical Engineering, The University of Melbourne, Victoria, Australia. E-mail: kuanl@student.unimelb.edu.au

Contract/grant sponsor: Australian Research Council's Linkage Project; contract/grant number: LP110200025

Copyright © 2017 John Wiley & Sons, Ltd.

Prepared using rncauth.cls [Version: 2010/03/27 v2.00]

This article is protected by copyright. All rights reserved.

approaches differ in the way that the state and input variables are discretised and also in the way the optimisation is formulated.

In single shooting, the state variables are considered to be dependent variables of the input decision variables and the nonlinear process model is numerically integrated over the entire time period. In multiple shooting, the state variables are split into subintervals and then numerically integrated within the interval. The state trajectories are optimised separately and additional equality constraints are imposed to enforce continuity across the subintervals. In direct collocation, both the control input and the state are discretised simultaneously and treated as decision variables. This leads to a sparse finite dimensional optimisation problem where the constraints of the NLP consist of the constraints of the control problem along with the discretised system dynamics in the form of equality constraints. At each time step of the NMPC, an NLP is solved and only the first control input of the solution is applied to the control actuator.

Direct methods involve approximation of the control problem through discretisation of the control and/or state variables over the time horizon. The choice of discretisation is driven by keeping the error uniformly small and keeping the computational and memory costs of the controller hardware low. One method that addresses both these issues is adaptive refinement. Adaptive refinement seeks to refine the discretisation mesh in regions where the accuracy of the solution is demanded and to coarsen the mesh where it is not.

A number of adaptive refinement methods have been proposed for OCPs. *h*-refinement refers to adaptation of the mesh segment widths. These methods include minimising the maximum integration error [6] and the use of density functions based on control and state histories from previous mesh refinement iterations [7]. In [8], error estimates for primal and adjoint variables are developed and are used for both the refinement and stopping criteria.

In contrast, *hp*-refinement refers to simultaneous adaptation of both the mesh segment widths and the polynomial degree of the approximating functions. These include pseudospectral methods [9] and adaptive wavelet methods [10]. In [11] the decay rates of Legendre polynomial coefficients as an estimate of the state approximation error.

Adaptive mesh refinement for MPC strategies have also been reported. In [12], the work presented in [8] for OCPs is extended to MPC. Where in [13], sensitivity-based error estimates are applied to the control inputs to obtain locally refined meshes.

In consideration of NMPC as opposed to OCPs, the aforementioned adaptive refinement methods face an additional constraint. That is, that NMPC is a discrete control method and requires a fundamental time step on which control actions and sensor measurements are taken. The selection of the fundamental time step depends on a number of factors including the plant, sensor and actuator dynamics and the available computational resources of the controller hardware. This means that arbitrary refinement of the control input is not always possible.

The refinement method is based on goal-oriented mesh adaptation [14, 15] which has been extensively used in mesh refinement of partial differential equations particularly in the computational fluid dynamics (CFD) community. A unique aspect of this method compared to previous work is that the mesh is adapted with respect to the cost function and not the state or input dynamics. This leads to refinement dictated by the cost function in the control problem rather than being driven by the system dynamics.

In Section 2, the direct collocation method is presented with the modifications necessary for mesh adaptation. Section 3 introduces goal oriented error estimation which is the criteria used by the algorithm. The algorithm is outlined in Section 4 and a simple two-state system and a missile pitch axis control problem are presented showing the computational advantages of using this scheme in Section 5.

2. DIRECT COLLOCATION

The NMPC can be implemented using the direct collocation algorithm. For reference, it is derived as follows. Consider a continuous dynamical system of the form,

$$\dot{x} = f(x, u, t), \quad (1)$$

with state and input constraints of the form,

$$x \in \mathbb{X} \quad (2a)$$

$$u \in \mathbb{U} \quad (2b)$$

where $\mathbb{X}$ and $\mathbb{U}$ represent the admissible sets of the states and inputs respectively. Associated with this dynamical system is a continuous cost function,

$$J_c = \phi[x(t_f), t_f] + \int_{t_0}^{t_f} L[x(t), u(t), t] dt. \quad (3)$$

Assumption 1

The following assumptions are placed on the dynamic system:

1. $f : \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}^n$ is a twice continuously differentiable function on $\mathbb{X} \times \mathbb{U}$.
2. $\mathbb{X} \subseteq \mathbb{R}^n$ is closed and simply connected set.
3. $\mathbb{U} \subseteq \mathbb{R}^m$ is a compact and convex set.
4. A steady state equilibrium $f(x^*, u^*) = 0$ exists, and this is desired operating condition to which the NMPC regulator will drive the system to. Furthermore the point (x^*, u^*) is in $\mathbb{X} \times \mathbb{U}$.
5. In order to guarantee stability, some form of terminal constraint is also applied to the NMPC.

The control input and state variables are then discretised and the continuous time system and cost function are reformulated as an NLP.

Assumption 2

The fundamental time steps of the NMPC are invariant. That is to say, the discretised time intervals over which the control inputs remain constant. It follows that the refinement method is only applied within each fundamental time step and it is only useful to refine the state variables.

So that Assumption 2 holds, let the prediction horizon $T_f := t_f - t_0$ be discretised using two indices $j \in \{0, 1, 2, \dots, M\}$ and $k \in \{0, 1, 2, \dots, N\}$, where j is the control index and k is the state

index. Time along the prediction horizon is then given by $t_j = [t_0, t_1, \dots, t_M]$, where, $t_0 < t_1 < \dots < t_M$ and $t_k = [t_0, t_1, \dots, t_N]$, where, $t_0 < t_1 < \dots < t_N$. Also $t_f = t_M = t_N$. Let $\tau \in \mathbb{Z}_{>0}^N$ be a vector where each element represents the current state discretisation level for the time interval $t_j - t_{j-1}$. The relationship between the two indices are such that for $t_{j-1} < t_k < t_j$,

$$k = \begin{cases} 0, & \text{if } j = 0 \\ m + \sum_{l=1}^{j-1} \tau_l, & \text{otherwise} \end{cases} \quad (4)$$

where $m \in \{0, 1, \dots, \tau_j\}$. An example of τ , x_k and u_j for a dynamic system is illustrated in Figure 1. As can be seen in the figure, the control index j is divided over the prediction horizon and the state index k is determined by the value of τ as well as the control index.

Remark 1

Note that the control interval need not be evenly spaced (only invariant) and so schemes that involve move blocking can also be utilised.

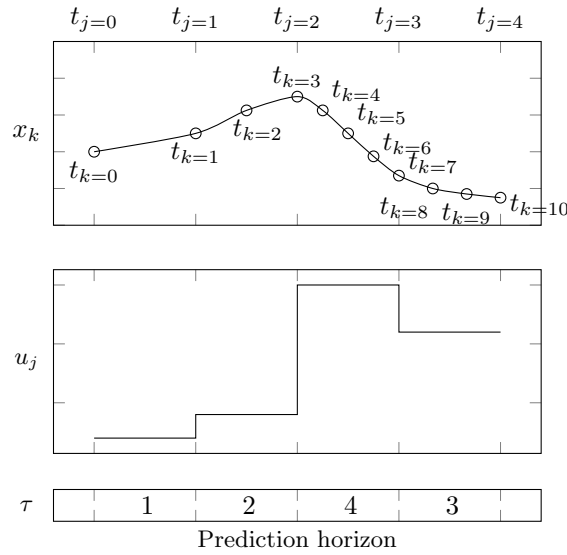


Figure 1. Refinement relationship between τ and the states and inputs

Suppose a collocation method is used to approximate the dynamical system (1) by a polynomial of degree s over the interval $[t_{k-1}, t_k]$. Let $0 \leq c_1 \leq c_2 \leq \dots \leq c_s \leq 1$,

$$p(t_{k-1}) = x_{k-1} \quad (5)$$

$$p'(t) = f(t, p(t)) \quad (6)$$

where, $t = t_{k-1} + c_q(t_k - t_{k-1})$ for $q = 1, \dots, s$ and $t_k = t_{k-1} + h$.

For example, an implicit (backward) Euler approximation written in collocation form has degree $s = 1$, and,

$$p(t) = f(t_k, p(t_k))(t - t_{k-1}) + p(t_{k-1}) \quad (7)$$

The collocation approximations within each k -indexed time interval in the NLP are then,

$$R(x, u) = \begin{Bmatrix} x_0 - x_{initial} \\ x_1 - p(t_1) \\ \vdots \\ x_k - p(t_k) \\ \vdots \\ x_N - p(t_N) \end{Bmatrix} = 0 \quad (8)$$

Note that the state discretisation method chosen above follows the h -refinement methods for mesh adaptation. The alternative, and equally valid, hp -refinement methods could also be employed by altering the degree of the collocation polynomial $p(\cdot)$.

The state constraints are discretised with the new index k as follows,

$$x_k \in \mathbb{X} \quad (9)$$

The cost function is also similarly discretised,

$$J = \phi(x_N, t_N) + \sum_{k=1}^N L(x_k, u_k, t_k) \Delta t. \quad (10)$$

Remark 2

Discretisation of the state constraints in this way means that there are only guarantees for the constraints at the sampling time and not at the intersampling times. For practical reasons, the possible (albeit mild) violations at intersampling times are ignored.

3. GOAL ORIENTED ERROR ESTIMATION

A necessary step in adaptive mesh refinement is to establish a method of estimating the error in the approximation of the dynamical system (1). Specifically, an estimate of the error is required for each discretisation time interval in the prediction horizon so that the refinement algorithm can selectively refine (coarsen) only in the time intervals where the error is deemed to be high (low).

There are numerous error indicators that can be used to determine the error of the computed solution. For goal oriented error estimation, the error indicator relates the approximation error directly to the cost function. This is particularly useful in the context of NMPC as the error indicator can be used to target the refinement of the approximate solution only where it has an appreciable impact on the cost function.

There are two main methods in goal oriented error estimation, namely, the continuous method [15] and the discrete method [16]. In the continuous method, a continuous adjoint system is derived from the continuous differential equations. The continuous adjoint system is then discretised and the error estimates are derived. The order of operation is reversed for the discrete method, where the continuous differential equations are first discretised and only then is a discrete adjoint system derived. While the output for both methods are the same, there are formulation, numerical and

computational pros and cons for each method [17]. The proposed approach in this paper follows the discrete method.

Firstly, let C and F denote some arbitrary coarse and fine discretisation respectively. Suppose the optimal state trajectory x and input u are generated by the NLP solver over the coarse and fine discretisations and are given by $\{x_C, u_C\}$ and $\{x_F, u_F\}$ respectively.

Assumption 3

$\|x_C - x_F\|_2 < \epsilon_x$ and $\|u_C - u_F\|_2 < \epsilon_u$.

Remark 3

The aim is to avoid utilising the (computationally expensive) NLP solver over the fine discretisation. Assumption 3 means that the solution to the NLP utilising the coarse discretisation cannot be too different from the fine discretisation, so that:

1. The system states calculated using the coarse discretisation can be interpolated onto the fine discretisation.
2. The optimal input trajectory calculated using the coarse discretisation can be used directly on the fine discretisation.

Note that Assumption 3 is similarly made in [6].

Denote the cost function (10) for the fine discretisation as $J_F(x_F, u)$ and the residual (8) as $R_F(x_F, u)$.

Now define $x_F^C := I_F^C x_C$ as the solution of the coarse discretisation that has been interpolated onto the fine discretisation, where I_F^C is the interpolation operator.

Consider Taylor series expansion of the cost function for the fine discretisation,

$$J_F(x_F, u) = J_F(x_F^C, u) + \left. \frac{\partial J_F}{\partial x_F} \right|_{x_F^C} (x_F - x_F^C) + \mathcal{O}((x_F - x_F^C)^2) \quad (11)$$

Similarly, the Taylor series expansion of the residual is,

$$R_F(x_F, u) = R_F(x_F^C, u) + \left. \frac{\partial R_F}{\partial x_F} \right|_{x_F^C} (x_F - x_F^C) + \mathcal{O}((x_F - x_F^C)^2) = 0 \quad (12)$$

Assumption 4

There is a feasible solution to the NLP, then this implies that an inverse of $\left. \frac{\partial R_F}{\partial x_F} \right|_{x_F^C}$ exists

Remark 4

NLPs are typically solved numerically using Newton-like algorithms. A necessary condition of the convergence of these algorithms is that the Jacobian of the function at the solution is nonsingular [18]. Therefore if the NLP converges, then the Jacobian in Assumption 4 is non-singular.

If Assumption 4 holds, define

$$\psi_F := \left. \frac{\partial J_F}{\partial x_F} \right|_{x_F^C} \left[\left. \frac{\partial R_F}{\partial x_F} \right|_{x_F^C} \right]^{-1}. \quad (13)$$

Ignoring higher order terms as $(x_F - x_F^C)$ is small and combining (11), (12) and (13),

$$J_F(x_F, u) = J_F(x_F^C, u) - \psi_F R_F(x_F^C, u) \quad (14)$$

Solving (13) requires evaluation at the fine discretisation level, but let

$$\psi_F = L_F^C \psi_C - \delta \psi_F, \quad (15)$$

where, $L_F^C \psi_C$ is an interpolated estimate of ψ_F and $\delta \psi_F$ is the error of ψ between the fine and coarse estimate. Rewriting (14) and combining with (15),

$$\begin{aligned} \Delta J_F &:= J_F(x_F, u) - J_F(x_F^C, u) \\ &= L_F^C \psi_C R_F(x_F^C, u) - \delta \psi_F R_F(x_F^C, u) \end{aligned} \quad (16)$$

Remark 5

Equation (16) is the error estimate of the cost function between the fine discretisation and the (coarse) interpolated discretisation. The implication of ΔJ_F is that the error between the coarse and fine discretisation can be made without having to calculate the cost function or solving the NLP at the level of the fine discretisation.

The first term of the right hand side of (16) is called the *computable correction*, so-called because its solution can be found without needing to solve on the fine discretisation level, and the second term is known as the *remaining error*.

It can be seen that,

$$\|L_F^C \psi_C R_F(x_F^C, u)\| \leq \sum_{k=0}^N \|(L_F^C \psi_C)_k (R_F(x_F^C, u))_k\| \quad (17)$$

where, $(L_F^C \psi_C)_k$ and $(R_F(x_F^C, u))_k$ are the k^{th} components of each of the respective vectors.

Let ε_j be the error estimate of the cost function for each discretised time interval $t_j - t_{j-1}$,

$$\varepsilon_j := \sum_k^{\tau_j} \|(L_F^C \psi_C)_k (R_F(x_F^C, u))_k\| \quad (18)$$

where, k is given by (4). Equation (18) estimates the discretisation error in each control time interval and can be used to alter the state discretisation level. Note that as in [19], the error estimate developed here is based on just the computable correction. It is also possible to estimate the remaining error, although doing so incurs a higher computational cost.

4. PROPOSED ALGORITHM

For each control iteration, the standard collocated NMPC algorithm consists of transcribing the optimal control problem into an NLP, solving the NLP and applying the first element of the input u as the control input to the plant. To reduce the computational burden of solving the NLP, the

proposed algorithm modifies the standard algorithm by adapting the number of states x_k at each control iteration. The adaptation is driven by the error estimate of the cost function, and this means the accuracy of the solution with respect to the cost function should not suffer when calculating a (potentially) faster and smaller NLP solution.

The following description outlines the main steps of the proposed algorithm. Refinement and coarsening thresholds ε_{lower} and ε_{upper} are defined respectively. When the error estimate ε_j in (18) exceeds either of these thresholds, then an adaptation rule is applied to change the discretisation level. In order to limit the level of coarsening and refinement which is determined by the vector τ , τ_{min} and τ_{max} are also defined respectively.

Algorithm 1

The steps of the algorithm are described as follows:

1. Set ε_{lower} , ε_{upper} , τ_{min} and τ_{max} .
2. Initialise each τ_j to $\frac{\tau_{max}}{2}$.
3. For each control iteration, $i < i_{max}$:
 - (a) Transcribe the optimal control problem defined by equations (1)–(3) into discretised form given by (8)–(10) using the current discretisation level given by τ .
 - (b) Solve the optimal control problem to produce the optimal input and state trajectories, $\{u_j, x_{C,k}\}$.
 - (c) Apply the first element of u_j as the control input to the plant.
 - (d) Assume that the state solution in each $t_j - t_{j-1}$ time interval is to be refined and produce an interpolated solution $\{u_j, x_{F,k}^C\}$ and τ_{int} .
 - (e) Use the interpolated solution $\{u_j, x_{F,k}^C\}$ to determine the error estimate ε_j using (18).
 - (f) Determine the new refinement level for each element where,

$$\tau_j = \begin{cases} \tau_j - 1, & \text{if } \varepsilon_j < \varepsilon_{lower} \text{ and } \tau_j > \tau_{min} \\ \tau_j + 1, & \text{if } \varepsilon_j > \varepsilon_{upper} \text{ and } \tau_j < \tau_{max} \\ \tau_j, & \text{otherwise} \end{cases} \quad (19)$$

- (g) Increment the control iteration i .

Steps (3a) to (3c) of Algorithm 1 form the standard steps in solving an NMPC problem. The refinement part is carried out in steps (3d) to (3f). In step (3d) the interpolated solution is found through interpolation methods and not by solving the optimal control problem at the refinement level of τ_{int} . The refinement strategy (19) in step (3f) increases or decreases the discretisation level by one, but alternative strategies such as doubling or halving the discretisation level or varying the level proportionately to the magnitude of the error could also be adopted. The tuning parameters ε_{lower} and ε_{upper} control the sensitivity of the refinement strategy with respect the error estimate. The parameters τ_{min} and τ_{max} act as hard stops for the refinement strategy to prevent runaway conditions.

Remark 6

One method to speed up the computation of the solution is to “warm start” the solution by initialising the current time step with the solution calculated at the previous time step. Additionally, the solution

at the previous time step can be shifted right during initialisation so that it better represents the solution at the current time. For a non-uniform mesh, some interpolation of the solution is required along with shifting. Similarly, the mesh refined at the previous time step can be used to “warm start” the mesh at the current time step. Indeed, to reduce the computational overhead the mesh is only refined once per control iteration.

The algorithm was implemented in the casADi framework [20]. This framework exposed many of the underlying NLP functions which proved useful in obtaining, for instance, the Jacobian $\left. \frac{\partial R_F}{\partial x_F} \right|_{x_F^C}$. The framework is also based on the *Python* programming language which makes debugging and development very rapid.

The advantages of this algorithm is that it has a small overhead when compared to the overall computational requirements of solving an NLP. However the adapted meshes are not optimal with respect to the cost function as the refinement algorithm is performed only once at every MPC step. Another disadvantage is that the refinement is only applied to the mesh associated with the system states, whereas in [12], both the states and control inputs can be refined, and, in [13] where the control inputs are explicitly refined.

5. EXAMPLES

5.1. Two-state system

The first example is a two-state nonlinear system taken from [21].

This algorithm is robust in the sense that it will perform no worse than the coarsest (initial) mesh. However, no attempt has been made to evaluate parasitic scenarios, for instance, the effects of adjacent time intervals with refinement levels τ_{min} and τ_{max} .

$$\frac{d}{dt} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1.2x_2 + 0.1 \sin(t) \\ 0.2 \sin(x_1) + u \end{bmatrix} \quad (20)$$

The cost function applied to this system is,

$$J = \sum_k^{\tau_j} (t_k - t_{k-1}) x_k^2 + \sum_j^M (t_j - t_{j-1}) u_j^2. \quad (21)$$

The cost function here penalises both the states and input and attempts to regulate the states to origin while minimising the input effort. The system is subjected to a total simulation time of $T = 12.5$ seconds. The NMPC controller uses a prediction horizon of $T_f = 6$ seconds, control discretisation of $N = 25$, a minimum discretisation level of $\tau_{min} = 1$, a maximum discretisation level of $\tau_{max} = 10$, $\varepsilon_{upper} = 1 \times 10^{-3}$ and $\varepsilon_{lower} = 5 \times 10^{-4}$.

Figure 2 shows the states and inputs against time and also x_2 against x_1 . It can be seen that the controlled system stabilises at a limit cycle around the origin. Figure 3 shows the discretisation level τ represented as a shade with prediction horizon along the horizontal axis and time along the vertical axis. Note that the mesh is adapted quite aggressively during the first 2 seconds of simulation, but

much less after this point. Also note that the higher level of discretisation at the end of the prediction horizon suggests that the adaptive mesh is accounting for the terminal limit cycle behaviour.

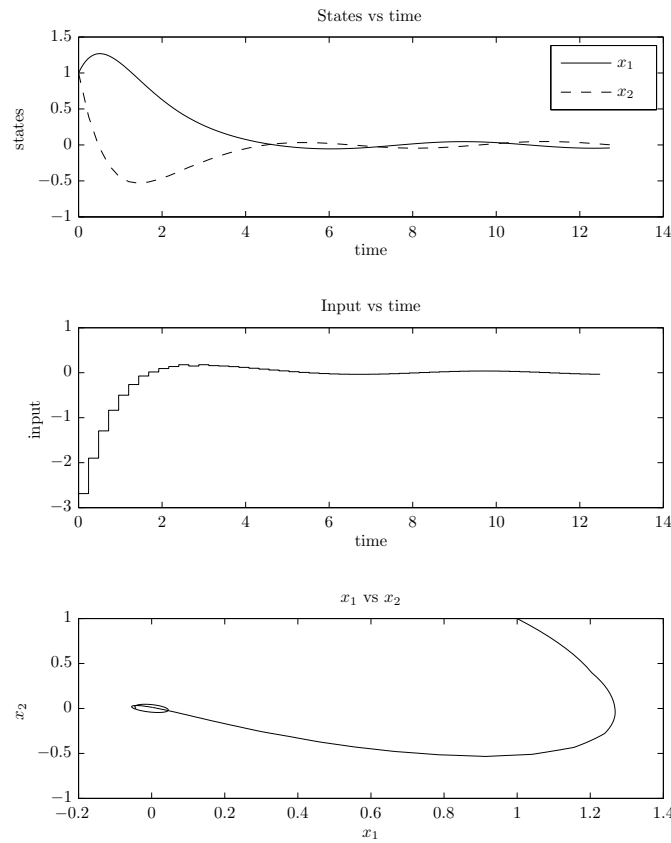


Figure 2. Two state system

In order to evaluate the computation load of the adaptive mesh algorithm, the computational time is measured. The algorithm is compared against an NMPC controller without adaptation with the discretisation level set to τ_{max} . As this is an indirect measurement of the computational load, ten separate runs are conducted and the average and standard deviation time of the runs are shown in Table I. Also shown is the open loop cost function evaluated using the closed loop data for a single run (although there were no differences in cost between the 10 runs). The table shows that

	Mean time (sec)	Standard dev. time (sec)	Cost function
With adaptation using Algorithm 1	8.68	0.22	6.41
With constant temporal resolution	17.46	0.51	6.41

Table I. Comparison of computational time and cost function evaluation for two state system (sec)

the average computational time for the adaptive mesh algorithm is approximately halved for the

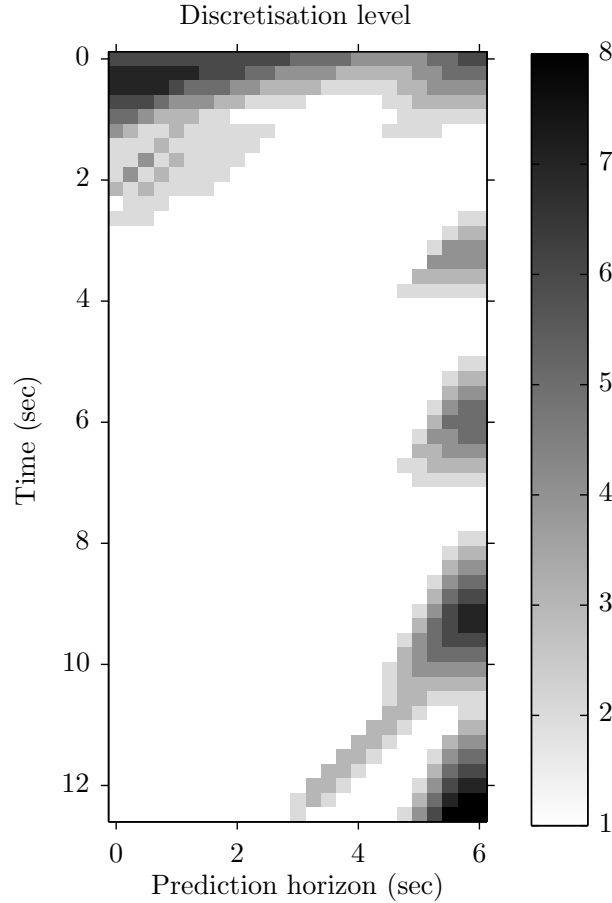


Figure 3. Two state system discretisation level. The horizontal axis shows the discretisation level for the prediction horizon and the vertical axis shows the evolution of the prediction horizon at each controller time step.

simple two state system. The variance of the computational time is also smaller with adaptation than without. The results also show that there are no differences in the cost function and this implies that adaptation has not adversely affected the performance of the controller. Note that without adaptation the controller cannot run in real-time.

5.2. Missile pitch axis system

The second example is a pitch axis missile model. It is based on a model given in [22] and has been modified so that the control input is the deflection rate $\dot{\delta}_c$ rather than the desired deflection. The pitch axis missile model is given by,

$$\frac{d}{dt} \begin{bmatrix} \alpha \\ q \\ \delta \\ \dot{\delta} \\ \delta_c \end{bmatrix} = \begin{bmatrix} K_a M C_n(\alpha, \delta) \cos(\alpha) + q \\ K_q M^2 C_m(\alpha, \delta) \\ \dot{\delta} \\ -\omega_a^2 \delta - 2\zeta \omega_a \dot{\delta} + \omega_a^2 \delta_c \\ u \end{bmatrix} \quad (22)$$

The missile's normal acceleration is given by,

$$\eta_r = K_z M^2 C_n(\alpha, \delta) \quad (23)$$

The cost function is,

$$J = \sum_k^{\tau_j} (t_k - t_{k-1}) (\beta_1(\alpha_k - \alpha_c)^2 + \beta_2(q_k - q_c)^2 + \beta_3(\delta_k - \delta_c)^2 + \beta_4\dot{\delta}_k^2) + \sum_j^M (t_j - t_{j-1}) (\beta_5(u_j - \delta_c)^2) \quad (24)$$

where the β 's are weights associated with each of the states. This cost function attempts to drive the missile to a target state represented by $[\alpha_c, q_c, \delta_c, 0, \delta_c]^T$ which has been computed offline as the equivalent target normal acceleration of $\eta_c = 10g$.

The total simulation time is $T = 10$ seconds with a prediction horizon $T_f = 10$ seconds and control discretisation of $N = 30$. A minimum discretisation level of $\tau_{min} = 1$ and a maximum discretisation level $\tau_{max} = 10$ is utilised. The error thresholds are set to $\varepsilon_{upper} = 2.5 \times 10^{-5}$ and $\varepsilon_{lower} = 5 \times 10^{-6}$.

It can be seen from Figure 4 that the desired normal acceleration is achieved using the NMPC controller. Furthermore the control input deflection rate also settles to zero once the steady state normal acceleration is reached. Figure 5 shows the discretisation level τ and it can be seen that aggressive discretisation occurs in the first 5 seconds of the controller where there is rapid actuation and state changes and much less discretisation is required once the missile is in steady state. As the error estimates are based on the cost function, this suggests that when the system is regulated to the desired set point then the mesh would naturally be coarsened and therefore reduce the computational load.

The computational load of the proposed algorithm for the missile example is also compared against an NMPC controller without adaptation. The average and standard deviation time of ten separate runs are shown in Table II. Also shown is the open loop cost function evaluated using the closed loop data for a single run. The table shows that the average computational time for the

	Mean time (sec)	Standard dev. time (sec)	Cost function
With adaptation using Algorithm 1	13.13	0.33	3.09
With constant temporal resolution	19.29	0.47	3.06

Table II. Comparison of computational time and cost function evaluation for missile example (sec)

adaptive mesh is two-thirds of the non-adapted time for the missile system. The variances again are lower in the adapted case over the the non-adapted algorithm and there are minimal differences in the cost function. Again, it can be seen that there is minimal adverse impact of the closed loop cost due to adaptation with the benefits of a shortened computational time.

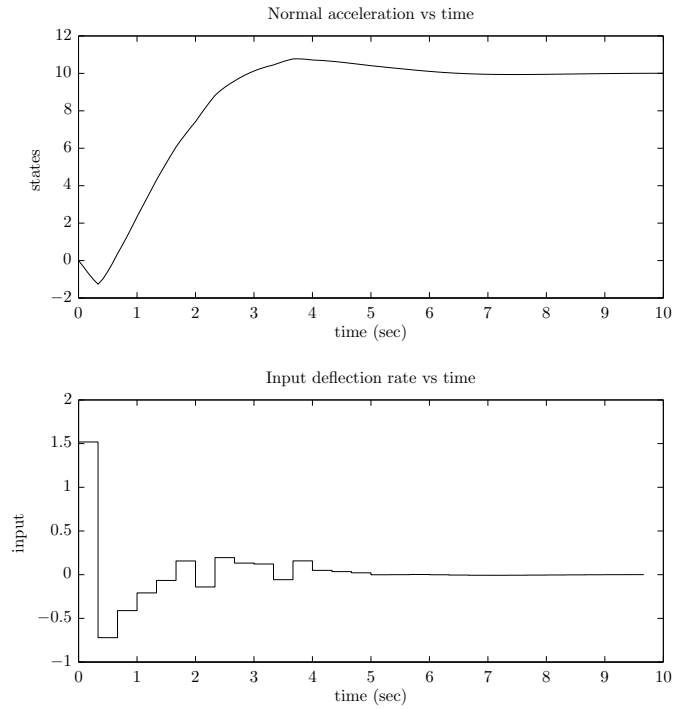


Figure 4. Pitch axis missile system

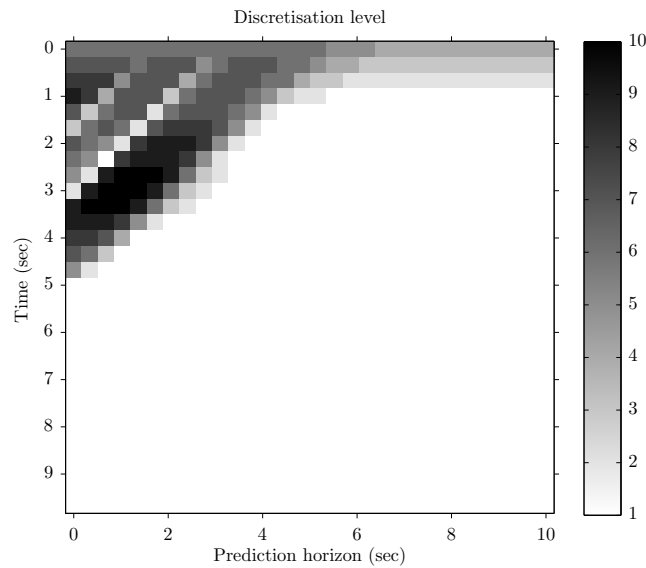


Figure 5. Missile system discretisation level. The horizontal axis shows the discretisation level for the prediction horizon and the vertical axis shows the evolution of the prediction horizon at each controller time step.

6. CONCLUSION

An adaptive mesh algorithm for direct collated non-linear model predictive control was presented. The algorithm locally refines or coarsens the mesh where necessary. The algorithm itself has low overheads and improvements of the computational time of up to 50% was observed in the given examples. This shows that a tangible benefit for real-world NMPC problems and the possibility to deploy the algorithm for real-time applications.

The refinement is based on goal-oriented error estimation which means the refinement of the control problem relates directly to the cost function rather than refinement based on the system dynamics or other heuristics. This method can potentially reduce much of the computational burden of NMPC, for instance, where some states have only a small or no contribution to the cost then the mesh would be appropriately coarsened in that area.

A natural extension for this mesh adaptation algorithm is in moving horizon estimator (MHE) problems. Further work could also consider refinement for each state separately (i.e. refinement on separate channels) or by combining the control inputs into the adaptation.

REFERENCES

1. Grüne L, Pannek J. *Nonlinear Model Predictive Control: Theory and Algorithms*. Springer-Verlag: London, England, 2011.
2. Qin SJ, Badgwell TA. A survey of industrial model predictive control technology. *Control Engineering Practice* Jul 2003; **11**(7):733–764.
3. Sargent RWH, Sullivan GR. The development of an efficient optimal control package. *Proceedings of the 8th IFIP Conference on Optimization Techniques, Part 2*, Stoer J (ed.), Springer: Heidelberg, 1978.
4. Bock HG, Plitt KJ. A multiple shooting algorithm for direct solution of optimal control problems. *Proceedings 9th IFAC World Congress, Budapest*, Pergamon Press: Oxford, England, 1984.
5. Biegler LT. Solution of dynamic optimization problems by successive quadratic programming and orthogonal collocation. *Computers & Chemical Engineering* 1984; **8**(3–4):243–247.
6. Betts JT, Huffman WP. Mesh refinement in direct transcription methods for optimal control. *Optimal Control Applications & Methods* Jan-Feb 1998; **19**(1):1–21.
7. Zhao Y, Tsiotras P. Density functions for mesh refinement in numerical optimal control. *Journal of Guidance, Control and Dynamics* Jan-Feb 2011; **34**(1):271–277.
8. Paiva LT, Fontes FACC. Adaptive time–mesh refinement in optimal control problems with state constraints. *Discrete and Continuous Dynamic Systems - Series A* Sep 2016; **35**(9):4553–4572.
9. Darby CL, Hager WW, Rao AV. An *hp*-adaptive pseudospectral method for solving optimal control problems. *Optimal Control Applications & Methods* Jul-Aug 2010; **32**(4):476502.
10. Dahmen W, Kunoth A. Adaptive wavelet methods for linear-quadratic elliptic control problems. *SIAM Journal of Control Optimization* 2005; **43**(5):1640–1675.
11. Liu F, Hager WW, Rao AV. Adaptive mesh refinement method for optimal control using decay rates of Legendre polynomial coefficients. *IEEE Transactions on Control System Technology* Jun; **PP**:1–9.
12. Paiva LT, Fontes FACC. Sampled-data model predictive control using adaptive time-mesh refinement algorithms. *Lecture Notes in Electrical Engineering* Sep 2016; **402**:143–153.
13. Xiao L, Liu P, Liu X, Zhang Z, Wang Y, Yang C, Gui W, Chen X, Zhu B. Sensitivity-based adaptive mesh refinement collocation method for dynamic optimization of chemical and biochemical processes. *Bioprocess and Biosystems Engineering* Jun 2017; **40**:1375–1389.
14. Venditti DA, Darmofal DA. Adjoint error estimation and grid adaptation for functional outputs: Application to quasi-one-dimensional flow. *Journal of Computational Physics* Oct 2000; **164**(1):204–227.
15. Becker R, Rannacher R. *Acta Numerica*, vol. 10, chap. An optimal control approach to a posteriori error estimation in finite element methods. Cambridge University Press: Cambridge, England, 2001; 1–102.

16. Giles MB, Pierce NA. Adjoint equations in CFD: duality, boundary conditions and solution behaviour. *13th Computational Fluid Dynamics Conference, Snowmass Village, CO, USA*, AIAA: Reston, USA, 1997; 182–199.
17. Nadarajah S, Jameson A. A comparison of the continuous and discrete adjoint approach to automatic aerodynamic optimization. *38th Aerospace Sciences Meeting and Exhibit, Aerospace Sciences Meetings, Reno, NV, USA*, AIAA-2000-0667, AIAA: Reston, USA, 2000.
18. Chachuat B. Nonlinear and Dynamic Optimization: From Theory to Practice 2007. Url: <http://lawwww.epfl.ch/page4234.html>.
19. Müller JD, Giles MB. Solution adaptive mesh refinement using adjoint error analysis. *15th Computational Fluid Dynamics Conference, Anaheim, California, USA*, AIAA: Reston, USA.
20. Andersson J. A general purpose software framework for dynamic optimization. PhD thesis, Arenberg Doctoral School, KU Leuven, Department of Electrical Engineering (ESAT/SCD) and Optimization in Engineering Center, Kasteelpark Arenberg 10, 3001-Heverlee, Belgium Oct 2013.
21. Falugi P, Kerrigan E, van Wyk E. *Imperial College London Optimal Control Software User Guide (ICLOCS)*. Department of Electrical and Electronic Engineering, Imperial College London, London, England, United Kingdom May 2010.
22. Nichols RA, Reichert RT. Gain scheduling for H -infinity controllers: A flight control example. *IEEE Transactions on Control Systems Technology* Jun 1993; **1**(2):69–79.