ISSN 1751-8644
doi: 0000000000
www.ietdl.org

A Shifting Strategy for Efficient Block-based Nonlinear Model Predictive Control Using Real-Time Iterations

Oscar Gonzalez[1], *Anthony Rossiter*[2*]

[1] *ACSE, The University of Sheffield, Sheffield, UK*
[2] *ACSE, The University of Sheffield, Sheffield, UK*
* *E-mail: ojgonzalezvillarreal1@sheffield.ac.uk*

Abstract: Nonlinear Model Predictive Control (NMPC) requires the use of efficient solutions and strategies for its implementation in fast/real-time systems. A popular approach for this is the Real Time Iteration (RTI) Scheme which uses a shifting strategy, namely the Initial Value Embedding (IVE), that shifts the solution from one sampling time to the next. However, this strategy together with other efficient strategies such as Move Blocking, present a recursive feasibility problem. This paper proposes a novel modified shifting strategy which preserve both recursive feasibility and stability properties, as well as achieves a significant reduction in the computational burden associated with the optimisation. The proposed approach is validated through a simulation of an inverted pendulum where it clearly outperforms other standard solutions in terms of performance and recursive feasibility properties. Additionally, the approach was tested on two computing platforms: a laptop with an i7 processor and a Beaglebone Blue Linux-based computer for robotic systems, where computational gains compared to existing approaches are shown to be as high as 100 times faster.

1 Introduction

Nonlinear Model Predictive Control (NMPC) is an advanced nonlinear optimisation method for Optimal Feedback Control that uses a mathematical model of a dynamical system to predict and optimise its future performance. Its popularity comes from its ability to handle constraints explicitly, as well as complex multivariable nonlinear dynamic systems [1]. One of the main challenges is that the required optimisation represents a significant computational burden, which has limited its application to relatively slow systems such as chemical reactors [2]. However, given the improvements in electronics in the last two decades, its application to fast real-time systems is now looking more feasible [3].

A key requirement for the implementation in real-time of NMPC is the use of efficient solutions. Efficiency may come in many different forms, from approximate solutions and inexact mathematical representations to tailored coding, special hardware such as Field Programmable Gate Arrays (FPGA) [4] and other strategies.

One of the most popular NMPC approaches nowadays is the Real-Time Iteration (RTI) Scheme proposed in [5] where a set of strategies are used to achieve real-time performance, namely: the Initial Value Embedding (IVE), the approximated single SQP solution, and the computations separation, offering solutions in the microseconds range [6–8] whilst preserving stability guarantees [9]. A tutorial-like paper of the latter is presented in [10]. Among other solutions based on inexact and approximate solutions, authors from [11] propose an inexact mathematical representation which avoids having time-varying matrices and preserves stability and recursive feasibility guarantees for a non-negligible region of the state space. In [12], a similar approach based on adjoints and inexact Jacobians is presented. In [13], the authors propose an inexact updating scheme that allows a reduction in the number of sensitivity updates required at each sampling time, by using a Curvature-like Measure of Nonlinearity (CMoN). Authors from [14] propose a partially tightened NMPC that uses a Riccati-like recursive equation combined with an interior-point like method that uses logarithmic barriers to remove the constraints at later stages of the prediction horizon. Finally, a common method for achieving faster computation times is by reducing the numbers of degrees of freedom; this can be done using input-parameterised solutions such as Move Blocking and Laguerre Polynomials [15] where an input-structure is embedded into the decision variables. Authors from [16] proposed methods for solving the blocked optimisation using highly parallelizable algorithms, which allow for even faster computation times. However, most works in this area, including [3, 16–18] have used blocking for NMPC with no regard to the recursive feasibility problem it presents.

In the case of tailored coding, a wide variety of toolkits exist that facilitate the implementation of NMPC. One of the most popular is the ACADO toolkit [19], an open source code capable of exporting efficient automatically generated code. Authors from [20] provide a tutorial-like paper for this toolkit. In [21], a toolkit named VIATOC that also exports automatically generated code is presented. Other toolkits such as CasADI and GRAMPC are also discussed in [21]. Another important area of research is the development of efficient QP solvers. Several solvers are available nowadays such as qpOASES, FORCES, CVXGEN and qpDUNES [20, 22, 23]. Furthermore, the solution of NMPC problems can be done using simultaneous or sequential approaches as discussed in [24] resulting in sparse or condensed QPs. Authors from [7] concluded that using condensed QPs is computationally faster for small to medium sized optimisations, whereas sparse solutions have better performance for medium to large. However, one of the main benefits of simultaneous approaches is that they present better stability characteristics for unstable systems [8]. Finally, different methods can be used for discretising an optimal control problem (OCP) such as direct single/multiple-shooting and direct collocation [6].

This paper focuses on a single-shooting condensed/sequential NMPC approach. It uses the RTI Scheme as a base line methodology in which the proposed shifting strategy is embedded, thus re-formulating the optimisation. Although the proposed methodology is formulated using this specific approach, it is possible to apply it using other approaches such as multiple-shooting (sequential or simultaneous), as well as in combination with other solutions and methods such as [16] or [25]. The proposed approach uses concepts from the block based solution for linear MPC presented in [26] which are extended to an NMPC formulation and merged with the RTI Scheme. The key contribution of this paper, which differs from both aforementioned works, is the modified shifting strategy.

By conceptualizing the optimisation in an absolute-time-frame, this allows the reduction of both the numbers of degrees of freedom and constraints, whilst also preserving recursive feasibility guarantees and stability properties. The proposed approach is shown to give computational benefits up to 100 times faster than the standard RTI-NMPC solution.

This paper is organized as follows: Section 2 presents a detailed description of the modeling, prediction and optimisation methods to be used such as the RTI Scheme and the block-based solutions. Section 3 develops the main ideas of the proposed shifting strategy presented in this paper and gives a simple overall generic example. Section 4 gives clear insight into important coding aspects for implementing the proposed approach and an example code applied to the benchmark problem of section 5 is provided in [27] and [28]. Section 5 presents a benchmark of this method applied to a fully nonlinear inverted pendulum model and focuses mainly on the overall performance and recursive feasibility properties of the different strategies that were tested. Additionally, it presents the computation times of the proposed approach implemented in two different systems: a laptop with an i7 processor, and the aforementioned Beaglebone Blue Linux-board [29]. Finally, section 6 contains conclusions, summarises the contribution of the paper and describes future work.

2 Nonlinear Model Predictive Control

2.1 Modeling

Throughout this paper, discrete-time nonlinear dynamics of the following form will be considered:

$$\begin{aligned} x_{k+1|k} &= f(x_{k|k}, u_{k|k}) \\ y_{k|k} &= g(x_{k|k}, u_{k|k}) \end{aligned} \tag{1}$$

where x_k are the states, u_k are the controls or inputs of the system and y_k are the outputs, with n_x, n_u and n_y the number of states, inputs and outputs, respectively. The notation $k+1|k$ reads "value at $k+1$ predicted at sample time k", and will only be used in full when needed for clarity.

Remark 1. *Continuous-time models, can be discretized using any integration method to reduce the infinite Optimal Control Problem (OCP) to an approximate but tractable and finite Nonlinear Problem (NLP). This allows simulating the system forward using the future nominal input trajectory, and linearising along the resulting trajectory.*

2.2 Prediction

By expanding a Taylor series up to first order terms, the system (1) can be approximated by:

$$\begin{aligned} x_{k+1} &= f(\bar{x}_k, \bar{u}_k) + \tfrac{\partial f(\bar{x}_k, \bar{u}_k)}{\partial \bar{x}_k}\delta x_k + \tfrac{\partial f(\bar{x}_k, \bar{u}_k)}{\partial \bar{u}_k}\delta u_k \\ &= \bar{x}_{k+1} + A_k \delta x_k + B_k \delta u_k \end{aligned} \tag{2}$$

where $\delta x_k = x_k - \bar{x}_k$ and $\delta u_k = u_k - \bar{u}_k$ represent the state and input deviations from the nominal points at time step $t = k$, respectively, and $A_k = \frac{\partial f(x_k, u_k)}{\partial x_k}$ and $B_k = \frac{\partial f(x_k, u_k)}{\partial u_k}$ represent the partial derivatives of the system dynamics. Notice the deviation $\delta x_{k+1} = x_{k+1} - \bar{x}_{k+1}$ at time step $t = k+1$ can be approximated by:

$$\delta x_{k+1} = A_k \delta x_k + B_k \delta u_k \tag{3}$$

Given that the nominal point $\bar{x}_{k+1}$ and linearisation matrices (A_k, B_k) of (2) depend parametrically on $\bar{x}_k$ and $\bar{u}_k$, and that at a given sampling time $t = k$ the value of $\bar{x}_k$ is already given either by measurements or by state estimation, the value of $\bar{x}_{k+1}$ can only be obtained by assuming (or guessing) a value for $\bar{u}_k$. If values are assumed/guessed for the future nominal input trajectory $\bar{U} = \begin{bmatrix} \bar{u}_k^T & \bar{u}_{k+1}^T & \cdots & \bar{u}_{k+N_p-1}^T \end{bmatrix}^T$, this allows the computation of the predicted nominal state trajectory $\bar{X} = \begin{bmatrix} \bar{x}_{k+1}^T & \bar{x}_{k+2}^T & \cdots & \bar{x}_{k+N_p}^T \end{bmatrix}^T$, and linearisation matrices A_k and B_k at future time steps $t = k+1, k+2, \cdots, k+N_p$, where N_p is known as the prediction horizon. This prediction assumption is known as single-shooting [20]. Common MPC strategies such as GPC, also use a control horizon N_u where the inputs after $k + N_u - 1$ are enforced to be the same [30]. Let us consider $N_u = N_p$ for now as this is just a special case of the blocked solution presented in section 2.5.

Once $\bar{X}$ is obtained using $\bar{U}$, the prediction equation (3) can be shifted forward:

$$\delta x_{k+2} = A_{k+1}\delta x_{k+1} + B_{k+1}\delta u_{k+1} \tag{4}$$

Thus, substituting equation (3) into (4) gives:

$$\begin{aligned} \delta x_{k+2} &= A_{k+1}(A_k \delta x_k + B_k \delta u_k) + B_{k+1}\delta u_{k+1} \\ &= A_{k+1}A_k \delta x_k + A_{k+1}B_k \delta u_k + B_{k+1}\delta u_{k+1} \end{aligned} \tag{5}$$

By repeating the above process recursively for N_p steps and considering only the output of the system, the predicted deviations from the nominal output trajectory can be represented in condensed form by:

$$\delta \hat{Y} = G\delta x_k + H\delta \hat{U} \tag{6}$$

where $\delta \hat{Y} = \hat{Y} - \bar{Y} = \begin{bmatrix} \delta y_{k+1}^T & \delta y_{k+2}^T & \cdots & \delta y_{k+N_p}^T \end{bmatrix}^T$ are the output deviations, $\delta \hat{U} = \hat{U} - \bar{U} = \begin{bmatrix} \delta u_k^T & \delta u_{k+1}^T & \cdots & \delta u_{k+N_p-1}^T \end{bmatrix}^T$ are the input deviations, and matrices G and H are given by:

$$G = \begin{bmatrix} C_1 A_0 \\ C_2 A_1 A_0 \\ \vdots \\ C_{N_p} A_{N_p-1} \cdots A_1 A_0 \end{bmatrix} \tag{7}$$

$$H = \begin{bmatrix} C_1 B_0 & O & \cdots & \cdots \\ C_2 A_1 B_0 & C_2 B_1 & O & \cdots \\ C_3 A_2 A_1 B_0 & C_3 A_2 B_1 & C_3 B_2 & \cdots \\ \vdots & \vdots & \vdots & \ddots \\ C_{N_p} A_{N_p-1} \cdots A_1 B_0 & C_{N_p} A_{N_p-1} \cdots A_2 B_1 & \cdots & \cdots \end{bmatrix} \tag{8}$$

where G has dimensions of $N_p n_y \times n_x$, H has dimensions of $N_p n_y \times N_p n_u$, $C_k = \frac{\partial g(\bar{x}_k, \bar{u}_k)}{\partial \bar{x}_k}$ is the partial derivative w.r.t the nominal state in (1), and O represents a matrix of zeros with the same dimensions of $C_k B_k$. Notice the k notation in A_k and B_k has been dropped for simplicity.

2.3 Optimisation

Once the prediction is formulated, a quadratic cost function penalizing the predicted errors between the reference trajectory Y_r and the predicted output trajectory $\hat{Y}$, as well as penalizing the input trajectory $\hat{U}$, can be formulated as:

$$J = \frac{1}{2}(Y_r - \hat{Y})^T Q (Y_r - \hat{Y}) + \frac{1}{2}\hat{U}^T R \hat{U} \tag{9}$$

where Q is a positive semi-definite matrix penalizing the predicted errors with dimensions $n_y N_p \times n_y N_p$ and R is a positive definite matrix penalizing input deviations with dimension $n_u N_p \times n_u N_p$. The latter represents an unbiased performance index for the inverted pendulum system when no disturbances are present, given it stabilizes at $u_k = 0$. If other types of system are used, the cost function (9) must be reformulated slightly, e.g. using unbiased costs [31].

In the following, two types of solutions will be formulated: the first one based on deviations of the nominal input trajectory $\delta\hat{U}$ and the second one based on the input trajectory $\hat{U}$ directly. Although both solutions give the same result, the inequality constraints are expressed differently. Let us first reformulate cost function (9) by expressing it in the standard QP form:

$$J = \tfrac{1}{2}z^T E z + f^T z \quad s.t \quad Mz \leq \gamma \tag{10}$$

where z is the decision variable to be optimised depending on which formulation (deviations or absolute) is chosen, E is a symmetric matrix formally known as the Hessian with dimensions $N_E = N_p n_u \times N_p n_u$, f is a column-vector with dimension $N_p n_u$ typically referred as the linear term, and M and γ are a matrix and vector respectively related to the inequality constraints. Equality constraints can be implemented by selecting the upper and lower limits of the inequality constraints to be the same.

2.3.1 Deviations Formulation: Substituting expression (6) and the definitions of $\hat{Y}$ and $\hat{U}$ into cost function (9) gives:

$$J = \frac{1}{2}(Y_r - \bar{Y} - G\delta x_k - H\delta\hat{U})^T Q (Y_r - \bar{Y} - G\delta x_k - H\delta\hat{U}) + \frac{1}{2}(\bar{U} + \delta\hat{U})^T R(\bar{U} + \delta\hat{U}) \tag{11}$$

By optimising w.r.t $\delta\hat{U}$, the optimisation has the standard QP form (10) where $E = H^T QH + R$ and $f = -(H^T Q(Y_r - \bar{Y} - G\delta x_k) - R\bar{U})$. Input and output inequality constraints are expressed relative to the nominal trajectory as:

$$M = \begin{bmatrix} I \\ -I \\ H \\ -H \end{bmatrix}; \qquad \gamma = \begin{bmatrix} U_{max} - \bar{U} \\ -(U_{min} - \bar{U}) \\ Y_{max} - \bar{Y} - G\delta x_k \\ -(Y_{min} - \bar{Y} - G\delta x_k) \end{bmatrix} \tag{12}$$

2.3.2 Absolute Formulation: Similarly, substituting expression (6) and the definition of $\hat{Y}$ and $\delta\hat{U}$ into cost function (9) gives:

$$J = \frac{1}{2}(Y_r - \bar{Y} - G\delta x_k - H\hat{U} + H\bar{U})^T Q (Y_r - \bar{Y} - G\delta x_k - H\hat{U} + H\bar{U}) + \frac{1}{2}\hat{U}^T R\hat{U} \tag{13}$$

Once again, by optimising w.r.t $\hat{U}$, the optimisation has the standard QP form (10) where $E = H^T QH + R$ and $f = -H^T Q(Y_r - \bar{Y} - G\delta x_k + H\bar{U})$. The input and output constraints are expressed as follows:

$$M = \begin{bmatrix} I \\ -I \\ H \\ -H \end{bmatrix}; \quad \gamma = \begin{bmatrix} U_{max} \\ -U_{min} \\ Y_{max} - \bar{Y} - G\delta x_k + H\bar{U} \\ -(Y_{min} - \bar{Y} - G\delta x_k + H\bar{U}) \end{bmatrix} \tag{14}$$

Having defined the QP problem, any QP solver such as qpOASES [22, 23] or quadprog MATLAB function can be used to find the solution. In this paper, the Hildreth's primal-dual quadratic programming procedure presented in [32] was used for the simulations of section 5.2 given its simplicity and "hot-starting" capabilities. A fixed amount of iterations were done to obtain predictable timings and accurate comparisons between the different approaches.

Remark 2. *For rigorous closed-loop stability guarantees, suitable terminal costs or zero-terminal constraints must be added by modifying the relevant matrices appropriately [6, 20, 33].*

2.4 Real Time Iteration Scheme

The Real Time Iteration (RTI) Scheme is a method developed by [5] for Nonlinear Optimisation in Optimal Feedback Control that is capable of giving real-time performance based on strategies summarised in the following subsections.

2.4.1 Initial Value Embedding: Initial Value Embedding (IVE) uses the solution found in the previous step in a shifted version, typically duplicating the last input variable $u_{k+N_p|k+1} = u_{k+N_p-1|k}$, to obtain the nominal trajectory over which the formulation will linearise and optimise. Additionally, in the case of QPs with "hot-start" capabilities such as active-set, it also uses a shifted version of the Lagrange multipliers λ found in the previous optimisation.

2.4.2 Single SQP: One can further reduce the computational burden and achieve predictable timings, by performing only a single SQP, ie. only linearise the optimisation once instead of re-linearising over and over until convergence. This is reasonable given that the optimisation is "hot-started" from the previous solution, which is expected to be close to the optimal solution, provided no significant disturbances have entered the system. Additionally, because the problem is forced to finish solving the linearised QP rather fast to give a quick feedback correction, the number of iterations or allowed time for solving the QP must be limited. In general, the solution of the problem is not given exactly but as an approximation that is expected to decrease the cost J at each iteration. Moreover, one must be satisfied with finding a local minimum and the solution can be subject to small approximation errors given only one re-linearization is done.

2.4.3 Computation Separation: Computation Separation is arguably the most important strategy. It separates the computations into feedback and preparation phases. A timing diagram illustrating this can be found in [10, 34].

(a) Preparation Phase

The preparation phase uses a predicted state $\hat{\bar{x}}_{k|k-1}$ as a starting point obtained from the last nominal input trajectory in its shifted version to linearise and prepare a QP. The standard RTI Scheme only performs the aforementioned tasks and solves the QP in the feedback phase, however, in this work a small modification is used where the QP is iterated during preparation assuming $\delta x_k = 0$ to find the vector of Lagrange Multipliers λ which is then used to compute the solution as given in equations (15) or (16).

(b) Feedback Phase

Once the state measurement becomes available, the feedback phase quickly delivers an approximate solution by calculating the predicted state deviation $\delta x_k = x_k - \hat{\bar{x}}_{k|k-1}$ and computing the "feedback phase" parts of equations (15) or (16), depending on which type of solution is being used. This allows the optimisation to have robustness against noise, disturbances and uncertainty. Because the state deviation δx_k has an effect, not only on the linear term f, but also in constraint vector γ (see eqns.(11,12,13,14)), the standard RTI Scheme recomputes them before solving the QP.

To further elaborate on the strategy of computation separation, notice that the value of δx_k in cost functions (11,13) only makes sense to be used in the context of the RTI Scheme, in particular, in the feedback phase. Assuming the vector of Lagrange Multipliers (λ), has been found by an appropriate QP in the preparation phase, the solution for both types (deviations and absolute), can be expressed as the summation of the QP result which is calculated in the preparation phase, and the effect of δx_k which is calculated in the feedback phase of the RTI. From the QP procedure presented in [32] it can be shown that the both solutions are given by the expressions below:

i) The deviations solution is given as:

$$\hat{U} = \bar{U} - E^{-1}\left[\underbrace{\overbrace{-(H^T Q(Y_r - \bar{Y}) - R\bar{U})}^{\text{Unconstrained}} \overbrace{+M^T\lambda}^{\text{Constrained}}}_{\text{Preparation Phase}} \underbrace{+H^T QG\delta x_k}_{\text{Feedback Phase}}\right] \tag{15}$$

ii) The absolute solution is given as:

$$\hat{U} = -E^{-1}\left[\underbrace{\overbrace{-H^T Q(Y_r - \bar{Y} + H\bar{U})}^{\text{Unconstrained}} \overbrace{+M^T\lambda}^{\text{Constrained}}}_{\text{Preparation Phase}} \underbrace{+H^T QG\delta x_k}_{\text{Feedback Phase}}\right] \tag{16}$$

A general drawback of NMPC methods based on the RTI Scheme is that the predictions can be subject to approximation errors given small deviation models are used and only one SQP iteration is done, ie. re-linearizing the system only once.

2.5 Blocked Solutions

A popular method for reducing the computational burden further is by using blocked solutions, where the inputs or decision variables are blocked in sections and assumed to have the same value [16, 17, 26, 35]. This allows a reduction in the number of degrees of freedom and consequently the optimisation time. To achieve this, an equality constraint of block size N_B is embedded into the optimisation as $u_k = u_{k+1} = \cdots = u_{k+N_B-1}$ across all the prediction horizon. The latter can be represented by an input structure of the form:

$$\delta\hat{U} = \mathbb{N}\delta\hat{\mathbb{U}} \tag{17}$$

$$\hat{U} = \mathbb{N}\hat{\mathbb{U}} \tag{18}$$

where $\hat{\mathbb{U}}$ (or $\delta\hat{\mathbb{U}}$) is the blocked decision variable and $\mathbb{N}$ is the blocking matrix defined as:

$$\mathbb{N} = \begin{bmatrix} \mathbb{I} & \mathbb{O} & \cdots & \mathbb{O} \\ \mathbb{O} & \mathbb{I} & \cdots & \mathbb{O} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbb{O} & \mathbb{O} & \cdots & \mathbb{I} \end{bmatrix} \tag{19}$$

with dimensions $N_p n_u \times \lceil \frac{N_p}{N_B} \rceil n_u$ where the operator $\lceil x \rceil$ rounds the result towards infinity, $\mathbb{I}$ is a matrix containing N_B vertically blocked identity matrices of n_u dimension and $\mathbb{O}$ is a matrix of zeros of the same dimension. Obviously with $N_B = 1$, the blocking structure is just an identity and represents the same QP as (11) or (13). Substituting (17) and (18) into cost functions (11) and (13) respectively, leads to:

i) Blocked Deviations Formulation

$$J = \frac{1}{2}(Y_r - \bar{Y} - G\delta x_k - H\mathbb{N}\delta\hat{\mathbb{U}})^T Q (Y_r - \bar{Y} - G\delta x_k - H\mathbb{N}\delta\hat{\mathbb{U}}) + \frac{1}{2}(\bar{U} + \delta\hat{\mathbb{U}})^T R (\bar{U} + \delta\hat{\mathbb{U}}) \tag{20}$$

ii) Blocked Absolute Formulation

$$J = \frac{1}{2}(Y_r - \bar{Y} - G\delta x_k - H\mathbb{N}\hat{\mathbb{U}} + H\bar{U})^T Q (Y_r - \bar{Y} - G\delta x_k - H\mathbb{N}\hat{\mathbb{U}} + H\bar{U}) + \frac{1}{2}(\mathbb{N}\hat{\mathbb{U}})^T R (\mathbb{N}\hat{\mathbb{U}}) \tag{21}$$

Optimising w.r.t the decision variables, ($\delta\hat{\mathbb{U}}$) and ($\hat{\mathbb{U}}$), results in the modified Hessian (22) for both modified cost functions, (20) and (21), respectively.

$$E_{\mathbb{N}} = \mathbb{N}^T (H^T Q H + R)\mathbb{N} = \mathbb{N}^T E \mathbb{N} \tag{22}$$

which has reduced dimensions of $N_{E_{\mathbb{N}}} = \lceil \frac{N_p}{N_B} \rceil n_u \times \lceil \frac{N_p}{N_B} \rceil n_u$, and the linear terms can be found to be, respectively:

i) Blocked Deviations Linear Term

$$f_{\mathbb{N}} = -\mathbb{N}^T (H^T Q (Y_r - \bar{Y} - G\delta x_k) - R\bar{U}) = -\mathbb{N} f \tag{23}$$

ii) Blocked Absolute Linear Term

$$f_{\mathbb{N}} = -\mathbb{N}^T H^T Q (Y_r - \bar{Y} - G\delta x_k + H\bar{U}) = -\mathbb{N} f \tag{24}$$

Finally, the constraint matrix M is modified to

$$M_{\mathbb{N}} = \begin{bmatrix} \mathbb{N}^T & -\mathbb{N}^T & (H\mathbb{N})^T & -(H\mathbb{N})^T \end{bmatrix}^T \tag{25}$$

whereas the constraint vectors γ remain the same.

The modified Hessian $E_{\mathbb{N}}$, linear term $f_{\mathbb{N}}$ and constraint matrix $M_{\mathbb{N}}$ can be used to compress/decompress a pre-prepared QP, eg. to use the strategy with a given toolkit such as ACADO.

Remark 3. *$\mathbb{N}$ has N_B vertically blocked identity matrices, so the number of constraints related to the input can be reduced provided the respective rows in γ of a given vertically blocked section are equal. This is not the case when when the solutions are based on deviations to a shifted blocked input trajectory (IVE strategy of RTI). In fact, unlike the unblocked case where both formulations (absolute or deviations) result in exactly the same solution, in this case, they will have different solutions because it is conceptually different to embed a blocked structure into either the deviations or absolute variables. However, if the shifting strategy proposed in this paper is used, it will be seen that they give the same results as a direct consequence of performing a consistent optimisation.*

After the optimisation is solved, definitions (17) and (18) can be used to recover the solution in the original variables. This results in solutions (15) and (16) presented in the section 2.4 to change to:

i) Blocked Deviations Solution

$$\hat{U} = \bar{U} - \mathbb{N}E_{\mathbb{N}}^{-1}\mathbb{N}^T \left[\underbrace{\overbrace{-(H^T Q (Y_r - \bar{Y}) - R\bar{U})}^{\text{Unconstrained}} \overbrace{+M^T\lambda}^{\text{Constrained}}}_{\text{Preparation Phase}} \underbrace{+H^T Q G \delta x_k}_{\text{Feedback Phase}} \right] \tag{26}$$

ii) Blocked Absolute Solution

$$\hat{U} = -\mathbb{N}E_{\mathbb{N}}^{-1}\mathbb{N}^T \left[\underbrace{\overbrace{-H^T Q (Y_r - \bar{Y} + H\bar{U})}^{\text{Unconstrained}} \overbrace{+M^T\lambda}^{\text{Constrained}}}_{\text{Preparation Phase}} \underbrace{+H^T Q G \delta x_k}_{\text{Feedback Phase}} \right] \tag{27}$$

One of the advantages of blocking is that the problem or system itself may required control actions in the future, and not all congested in the beginning of the horizon as with standard GPC approaches. This benefit can be seen in figure 1 where the predicted optimal trajectory of both approaches is given for the inverted pendulum problem presented in section 5 and compared to the one using the full decision vector. Notice although they all present differences in the input solution, the solutions of position and angle trajectories are nearly indistinguishable for the blocked and full solutions, whereas the GPC solution clearly results in a different trajectory. As expected, the predicted costs of all cases, full decision, blocked and standard GPC were $J_{full} = 1872$, $J_{blk} = 1878$ and $J_{std-gpc} = 1994$ respectively, which clearly shows the superiority of blocking over the standard GPC approach. Obviously, the GPC solution would adjust the input as the horizon is moved forward (receding horizon) and might be able to perform similarly in closed-loop. However, it is the inconsistency/ill-posedness of the problem within each prediction that may negatively affect the overall closed loop solution in the long term, especially when constraints come into play. This is discussed in section 3.2.

3 Shifting Strategy

This section presents the shifting strategy proposed in this paper which represents the main contribution and has the main goal of achieving faster computation times whilst preserving the stability and recursive feasibility properties of the standard RTI Scheme.

3.1 Sub-strategies

The three sub-strategies summarised next are used in the proposed approach. These sub-strategies are explained further in the following subsections and an overall example is given in section 3.6.

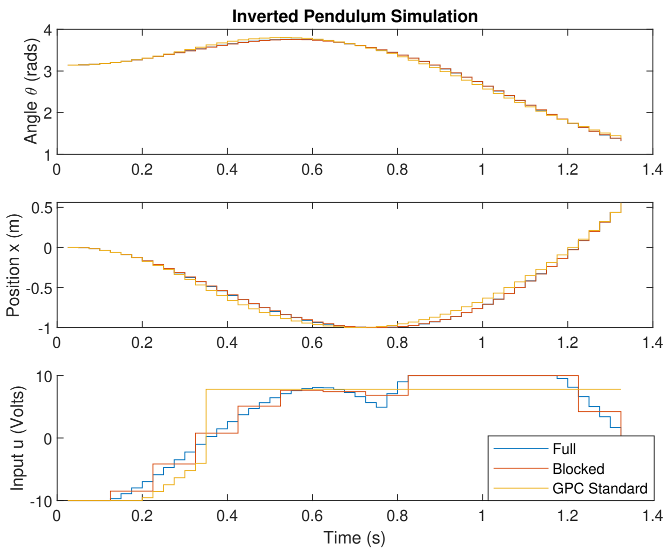

Fig. 1: Blocking vs GPC Standard Comparison with $N_B = 4$

3.1.1 Reducing the Number of Degrees Of Freedom: The method uses an embedded input-structure, in this case blocked, for reducing the degrees of freedom of the optimisation. Other input-structures such as Laguerre or Kautz Polynomials [15] were not considered but represent potential alternatives, although they would present different recursive feasibility properties and overall performance in the general case.

3.1.2 Reducing the Number of Shooting Points: It selects a reduced number of points of interest which sometimes are referred to as shooting points. These shooting points represent the constraints and future errors that are included in the optimisation, and are not necessarily at every sampling time but rather spread across the prediction horizon.

3.1.3 Absolute-Time-Frame Shifting: It shifts the shooting points and blocked-inputs structure in an absolute-time-frame, rather than in a relative-time-frame (as implemented in standard NMPC/RTI methods) to maintain consistency along the optimisation.

3.2 Consistent Optimisation and Recursive Feasibility

One of the most important properties to maintain in an optimisation is recursive feasibility [36]. This property is present in an optimisation *if and only if* for a given feasible solution at time $t = k$, all subsequent solutions at future times $t = k+1, k+2, \cdots, k+\infty$, remain feasible [26, 35]. At this point, it is emphasized that recursive feasibility is not related to how an initial feasible solution is found but rather, maintaining feasibility. In the case of a RTI Scheme, it is typically assumed that the solution is initially close to an optimal and feasible solution for nominal stability [9], thus initial feasibility is implied. Moreover, strong recursive feasibility guarantees can be given when the optimisation explicitly includes the solution from the previous sampling time as a possible solution of the current optimisation, also known as the tail [30, 31]. This is a direct result of performing a consistent optimisation where at each time, the latter improves or “builds on top of” the previous solution. This property is not naturally present in blocked solutions and instead, at each sampling time, the optimisation is forced to disregard the previous solution and find a new one which is the main reason they may lack recursive feasibility guarantees [15]. In other words, at each sampling time the optimiser makes a plan which is then immediately forced to disregard at the next sampling time.

3.3 Shifting Strategy applied to Blocked Solutions

To solve the aforementioned problems, we propose the following blocked solution which shifts the blocks in an absolute-time-frame to maintain consistency in the "breaking points" of the blocked input. A similar approach is presented in [26], however, an important difference with the proposed approach in this paper is that they do not apply it in the context of the RTI where the IVE approach is used, nor conceptualize it in an absolute-time-frame. Moreover, they do not formulate it in the context of NMPC for both types of solutions given above. Additionally, they use a time varying horizon whereas in the proposed approach, the horizon is maintained constant through the use of the ideal horizon. Finally, although the approach has similarities with lifted systems [37], it has important differences given both measurements and control actions are available at the all times and the strategy is applied to reduce computation times whilst maintaining consistency and recursive feasibility.

In simple terms, the proposed approach applies a set of blocking structures sequentially which guarantees that the previous solution (ie. the tail) is always included in the optimisation. By using this method, the solution based on deviations can now be applied consistently as it now represents the exact same solution given by the absolute blocked formulation; this will be seen later in the results section 5.2.

Definition 1. Shifting Blocked Sections
The proposed strategy can be formally represented with the input equalities (28) and (29) below, defined for time steps $[k, k+N_B-1]$ with an horizon N_p, "resetting" at time step $k+N_B$ and repeating infinitely.

$$u_{k+i+(n-1)N_B|k+j} = u_{k+nN_B-1|k+j} \tag{28}$$

$$\forall j = [0, N_B-1]; \quad \forall n = [1, \lceil \tfrac{N_p-N_B+j}{N_B} \rceil]$$

$$\begin{cases} \forall i = [j, N_B-1] & \text{if} \quad n = 1 \\ \forall i = [0, N_B-1] & \text{if} \quad 1 < n < \lceil \tfrac{N_p-N_B+j}{N_B} \rceil + 1 \end{cases}$$

and for the last block ($n = \lceil \frac{N_p-N_B+j}{N_B} \rceil + 1$),

$$u_{k+i+(n-1)N_B|k+j} = u_{k+i_{max}+(n-1)N_B|k+j} \tag{29}$$

$$\forall j = [0, N_B-1]; \quad \forall i = [0, i_{max}]$$

$$\text{with} \quad i_{max} = N_p + j - 1 - \lceil \tfrac{N_p-N_B+j}{N_B} \rceil N_B$$

where j is the time step, $j = 0$ giving the standard blocking structure, n is the number of blocked sections, $n = 1$ being the first one, and i is related to the size of the given blocked section. Notice the latter changes as $j \to N_B - 1$ and the number of blocks n depends on the selected prediction horizon. This will be explored further in the following subsections.

As a quick example of definition 1, consider a simple SISO optimisation with prediction horizon $N_p = 4$ and block size $N_B = 2$. The proposed strategy would apply the two following blocking matrices $\mathbb{N}_1$ and $\mathbb{N}_2$ sequentially, in order, and repeating infinitely ($\mathbb{N}_1, \mathbb{N}_2, \mathbb{N}_1, \dots$).

$$\mathbb{N}_1 = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{bmatrix} \qquad \mathbb{N}_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{30}$$

Lemma 1. N_B Unique Blocking Matrices
When using definition 1 with a blocking size of N_B, there will always be exactly N_B unique blocking matrices ($\mathbb{N}$) that include the tail, maintaining consistency over time, hence preserving recursive feasibility.

Proof: Consider only the first blocked section $n = 1$ of a given blocking structure. By applying (28) to maintain consistency at any time step $k + j$ within that first block, the latter gives:

$$\begin{gathered} u_{k+i|k+j} = u_{k+N_B-1|k+j} \\ \forall\, j = [0, N_B - 1]; \quad \forall\, i = [j, N_B - 1] \end{gathered} \tag{31}$$

Thus, the number of equalities i, ie. the size of the first blocked section $N_{B_1} = N_B - j$, decreases as $j \to N_B - 1$ leading to N_B sizes for the latter and consequently N_B unique blocking structures $\mathbb{N}$. Conceptually, the latter shrinks until reaching its limit and "resetting" its size to N_B as in $\mathbb{N}_1$ of example (30). □

3.3.1 The Ideal Prediction Horizon: Notice in example (30), the dimension of the resulting Hessian $E_{\mathbb{N}}$ would vary from $N_{E_{\mathbb{N}}} = 2 \times 2$ to $N_{E_{\mathbb{N}}} = 3 \times 3$. This is undesirable behavior given that to achieve better computing performance, dynamic memory allocation should be avoided. A work around to this problem is to use an ideal prediction horizon which allows implementation of the proposed shifting strategy without modifying the dimension of the Hessian.

Lemma 2. The Ideal Prediction Horizon
When using definition 1 *to maintain recursive feasibility, the selected horizon N_p must be an integer multiple of the block size plus 1 to keep the Hessian dimension $N_{E_{\mathbb{N}}}$ constant for any block size N_B:*

Proof: The expected size of the Hessian $N_{E_{\mathbb{N}}}$ for any block size N_B is given by:

$$N_{E_{\mathbb{N}}} = \lceil \frac{N_p - N_{B_1}}{N_B} \rceil + 1 \tag{32}$$

where N_{B_1} is the length of the first blocked section ($n = 1$ of 28) of a given blocking structure. For constant dimensions, the following must hold:

$$\begin{gathered} \lceil \tfrac{N_p - N_B}{N_B} \rceil + 1 = \lceil \tfrac{N_p - i}{N_B} \rceil + 1 \\ \forall i = [1, N_B] \end{gathered} \tag{33}$$

The latter can only be satisfied for all i and any N_B by using $N_p = nN_B + 1$, where n is an integer number. Substituting in (33) gives:

$$\lceil \frac{nN_B + 1 - N_B}{N_B} \rceil + 1 = \lceil \frac{nN_B + 1 - i}{N_B} \rceil + 1 \tag{34}$$

After some algebraic manipulation:

$$n - 1 + \lceil \frac{1}{N_B} \rceil = n + \lceil \frac{1 - i}{N_B} \rceil \tag{35}$$

because $\lceil \frac{1}{N_B} \rceil = 1$ and $\lceil \frac{1-i}{N_B} \rceil = 0$ for all i, equation (35) holds. □

Algorithm 1. The Ideal Prediction Horizon
The following steps are advised for selecting the ideal prediction horizon.

1. *Select the desired block size N_B.*
2. *Select a desired horizon $N_{p_{des}} > N_B$.*
3. *The closest ideal prediction horizon is then given by $N_p = \lceil \frac{N_{p_{des}}}{N_B} \rceil N_B + 1$ or $N_p = \lfloor \frac{N_{p_{des}}}{N_B} \rfloor N_B + 1$. For better stability properties, the upper one is suggested.*

Theorem 1. Shifting Strategy applied to Blocked Solutions - Recursive Feasibility
When using Definition 1 *with Lemma* 1 *and* 2, *recursive feasibility guarantees are recovered by always including the tail.*

Proof: Considering an optimal feasible solution with the blocked structure given by Definition 1 at time k ($j = 0$) with block size N_B:

$$u_{k+i+(n-1)N_B|k} = u_{k+nN_B-1|k} \tag{36}$$

$$\forall n = [1, \lceil \frac{N_p}{N_B} \rceil - 1]; \quad \forall i = [0, N_B - 1]$$

and the last block containing a single input $u_{k+N_p-1|k}$. The tail of the solution will be automatically included at the next time step $k + 1$ ($j = 1$) giving:

$$u_{k+i+(n-1)N_B|k+1} = u_{k+nN_B-1|k+1} \tag{37}$$

$$\forall n = [1, \lceil \frac{N_p}{N_B} \rceil - 1]; \quad \forall i = [1, N_B - 1]$$

and the last block containing two blocked inputs, $u_{k+N_p-1|k+1} = u_{k+N_p|k+1}$. The same is true $\forall\, j = [0, N_B - 1]$. □

3.3.2 Breaking Points: Shifting Lagrange Multipliers: An important concept in the proposed strategy is that of the "breaking points". As shown in Lemma 1, the first blocked section size N_{B_1} shrinks until it reaches its limit. When this happens, we refer to it as the "breaking point" and the optimisation must apply a blocking matrix $\mathbb{N}$ where the first blocked section "resets" and has the original block size $N_{B_1} = N_B$ such as $\mathbb{N}_1$ in (30). This is relevant when performing a constrained optimisation in the context of the RTI Scheme for hot-started solutions, as an active-set guess can be provided.

Theorem 2. Breaking Points - Shifting Lagrange Multipliers
When using definition 1 *with Lemma* 2, *the Lagrange Multipliers* (λ) *active-set guess for hot started solutions must be shifted only when the optimisation reaches the "breaking point" to maintain consistency.*

Proof: Consider the following unblocked Lagrange Multipliers related to positive input constraints ($M \le u_{max}$):

$$\lambda = \left[\lambda_{k|k}, \lambda_{k+1|k}, \cdots, \lambda_{k+N_p-1|k}\right] \tag{38}$$

The usual shifting strategy used by the RTI is:

$$\begin{gathered} \lambda_{k+i|k+i} > 0 \quad \text{active if} \quad \lambda_{k+i|k+i-1} > 0 \\ \lambda_{k+i|k+i} = 0 \quad \text{inactive if} \quad \lambda_{k+i|k+i-1} = 0 \\ \forall i = [0, N_p - 1] \end{gathered} \tag{39}$$

Now for simplicity, consider an ideal prediction horizon $N_p = N_B + 1$. When the proposed shifting input structure given by definition 1 is used, it requires only two lambdas $\left[\lambda_{1|k+j}, \lambda_{2|k+j}\right]$ for the two blocked sections where:

$$\begin{gathered} \lambda_{1|k+j} = \lambda_{k+i|k+j} = \lambda_{k+N_B-1|k+j} \\ \forall\, j = [0, N_B - 1]; \quad \forall\, i = [j, N_B - 1] \end{gathered} \tag{40}$$

for the first blocked section with initial block size N_B, and

$$\begin{gathered} \lambda_{2|k+j} = \lambda_{k+N_B+i|k+j} = \lambda_{k+N_B+j|k+j} \\ \forall j = [0, N_B - 1]; \quad \forall i = [0, j] \end{gathered} \tag{41}$$

for the second blocked section. Applying the RTI IVE shifting (39) combined with equalities (40) and (41), and considering the "breaking point" of definition (1) happens at $k + N_B$ gives:

$$\begin{gathered} \lambda_{1|k+N_B} > 0 \quad \text{active if} \quad \lambda_{2|k+N_B-1} > 0 \\ \lambda_{1|k+N_B} = 0 \quad \text{inactive if} \quad \lambda_{2|k+N_B-1} = 0 \end{gathered} \tag{42}$$

at that time step. The same holds for the rest of the blocks where:

$$\begin{gathered} \lambda_{i|k+N_B} > 0 \quad \text{active if} \quad \lambda_{i+1|k+N_B-1} > 0 \\ \lambda_{i|k+N_B} = 0 \quad \text{inactive if} \quad \lambda_{i+1|k+N_B-1} = 0 \\ \forall i = [1, N_{E_{\mathbb{N}}}] \end{gathered} \tag{43}$$

□

3.4 Shifting Strategy applied to Shooting Points

The concept of a reduced number of points of interest is already used in the shooting methods, e.g. [3, 18], however, they do not apply the shifting strategy proposed in this paper where the points are kept in an absolute-time-frame. The proposed approach of this paper builds on top of the conceptual "breaking points" aforementioned for the input-blocking and selects a subset of future errors and output constraints directly at the end of each blocked input for the optimisation.

Definition 2. Shifting Shooting Points
The proposed approach can be formally represented by selecting a subset of references, outputs and output constraints given by:

$$\begin{gathered} e_{k+nN_B|k+j} = r_{k+nN_B|k+j} - \hat{y}_{k+nN_B|k+j} \\ y_{min} \leq \hat{y}_{k+nN_B|k+j} \leq y_{max} \end{gathered} \tag{44}$$

$$\forall n = [1, N_{E_{\mathbb{N}}} - 1]; \quad \forall j = [0, N_B - 1]$$

where j is the time step, n is related to number of shooting points, and the last point of the prediction horizon:

$$\begin{gathered} e_{k+N_p+j|k+j} = r_{k+N_p+j|k+j} - \hat{y}_{k+N_p+j|k+j} \\ y_{min} \leq \hat{y}_{k+N_p+j|k+j} \leq y_{max} \end{gathered} \tag{45}$$

is always included. To select the points at the end of each blocked input, the time step j must be "in phase" with the time step j used by Definition 1.

Remark 4. *Selecting a given subset of output errors or output constraints can be achieved by selecting the respective rows of matrices $H, G, Y_r, \bar{Y}, M, \gamma$.*

Theorem 3. Shooting Points - Recursive Feasibility
The tail of the optimisation is automatically included by using Definition 2, giving recursive feasibility guarantees.

Proof: Consider an optimal solution for the shooting points:

$$\begin{aligned} \hat{r} &= \left[r_{k+N_B|k}, r_{k+2N_B|k}, \cdots, r_{k+(N_{E_{\mathbb{N}}}-1)N_B|k}, r_{k+N_p|k}\right] \\ \hat{Y} &= \left[\hat{y}_{k+N_B|k}, \hat{y}_{k+2N_B|k}, \cdots, \hat{y}_{k+(N_{E_{\mathbb{N}}}-1)N_B|k}, \hat{y}_{k+N_p|k}\right] \end{aligned} \tag{46}$$

that satisfies the constraints $Y_{min} \leq \hat{Y} \leq Y_{max}$, ie. is feasible at time k. By using (44) and (45) for time step $k+1$ $(j=1)$, the optimisation will keep looking at the same output errors and constraints at all the points (ie. the tail), except the last one.

$$\begin{aligned} \hat{r} &= \left[r_{k+N_B|k+1}, \cdots, r_{k+(N_{E_{\mathbb{N}}}-1)N_B|k+1}, r_{k+N_p+1|k+1}\right] \\ \hat{Y} &= \left[\hat{y}_{k+N_B|k+1}, \cdots, \hat{y}_{k+(N_{E_{\mathbb{N}}}-1)N_B|k+1}, \hat{y}_{k+N_p+1|k+1}\right] \end{aligned} \tag{47}$$

The only difference from the aforementioned variables to be used for the optimisation is the error $e_{k+N_p+1|k+1} = r_{k+N_B+1|k+1} - \hat{y}_{k+N_p+1|k+1}$. If there were no reference changes and a sufficiently big horizon is used, this error would make negligible change to the cost J and therefore the optimisation would be able to follow the plan obtained at the previous time step k, provided the previous decisions can be replicated (ie. the tail of the decision variables is available). It is noted that a rigorous guarantee requires invariant sets/terminal modes [31]. Nonetheless, works such as [2, 7, 8, 10, 20] have shown excellent performance without them, both in real systems and simulations. Moreover, because only one linearisation of the optimisation is performed in the RTI Scheme, the constraint satisfaction will be subject to the accuracy of the linearisation process. This is a common problem for any RTI Scheme variation.

Although this proof is derived by selecting shooting points at the conceptual "breaking points", it is a general result and holds, independently of whether blocked approaches were used or not. In other words, if a non-blocked input-parameterisation was used, it would still guarantee recursive feasibility for the shooting points provided the tail of the decision variables is always available. Without the proposed shifting strategy, no recursive feasibility guarantee can be given without the use of soft-constraints, ie. slack variables, which would relax the feasibility problem entirely. □

Theorem 4. Shooting Points - Shifting Lagrange Multipliers
As in Theorem 2, the Lagrange Multipliers (λ) must be shifted only when the optimisation reaches the "breaking points" to maintain consistency. This holds even for non-blocked solution. This proof is similar to Theorem 2 thus is omitted.

3.5 Stability, Optimality and Convergence

As discussed in remark 2, the stability of this scheme may be guaranteed with the use of zero-terminal constraints [6], or suitable terminal weights such as infinite horizon costing [20] which would make the resulting closed-loop sequence of costs to have Lyapunov stability. Indeed, the typical zero-terminal constraint proof can be seen directly in the assumption of theorem's 3 proof where new information would add negligible terms to the cost and remain feasible. In the case of infinite horizon costing, a local LQR control law may be used to stabilize the system in the terminal region after N_u control actions as in [20], however in some cases, the state may not be able to get inside the terminal region in one SQP iteration as performed by the RTI Scheme. Moreover, the required assumptions of the RTI Scheme discussed in [10] must be considered to achieve global optimality, including that the optimisation is initialized at the global optimum, and that there are no abrupt reference or state jumps. Finally, as per all SQP methods, the convergence of the numerical solution may be subject to appropriate step-size selection (typically full for RTI [10]), and the accuracy of the linearization process.

3.6 An Example of The Overall Shifting Strategy

To understand the overall strategy, consider the simple generic example given in (30) with the ideal prediction horizon $N_p = 5$, and the same block size $N_B = 2$. By dropping the absolute notation $k+i|k+j$ and applying the overall shifting strategy, the selected shooting points and constraints for both blocking structures expressed relative to the "current" time step, would lead to considering cost function (9) subject to the two set of variables and constraints defined in table 1, used in sequence and repeating infinitely $(1_{st}, 2_{nd}, 1_{st}, ...)$ for the optimisation. The optimisation can then be prepared by selecting appropriate matrices and vectors for Y_r, G, H, M, γ.

Remark 5. *Notice the entire future input trajectory is constrained in the optimisation given the blocking input-structure used, however, the output constraints only constrain the shooting points. This will be seen in the results section 5.2.*

To give a comprehensive visualization of the strategy, figure 2 shows the predicted trajectories in *the relative time frame* of three subsequent optimisation problems for the inverted pendulum problem presented in section 5 when the strategy is using a block size of $N_B = 4$. It is noticeable how the shooting points at times approximately $0.2 \leq t \leq 0.4$ (s) are kept at the constraint limits. Moreover, notice how most of the time, the shooting points and the blocked inputs are moving left horizontally indicating that the resulting optimal of all three optimisations, in absolute time, are nearly identical.

4 Efficient Coding

This section presents the algorithms and computational savings used to achieve efficient coding of the entire NMPC optimisation presented in section 2 together with the proposed Shifting Strategy of this paper presented in section 3.

Seq.	1_{st}	2_{nd}
Refs.	$Y_r = \begin{bmatrix} r_{k+2} \\ r_{k+4} \\ r_{k+5} \end{bmatrix}$	$Y_r = \begin{bmatrix} r_{k+1} \\ r_{k+3} \\ r_{k+5} \end{bmatrix}$
Outputs	$\hat{Y} = \begin{bmatrix} y_{k+2} \\ y_{k+4} \\ y_{k+5} \end{bmatrix}$	$\hat{Y} = \begin{bmatrix} y_{k+1} \\ y_{k+3} \\ y_{k+5} \end{bmatrix}$
Input Structure	$\hat{U} = \mathbb{N}_1 \hat{\mathbb{U}}$ or $\delta\hat{U} = \mathbb{N}_1 \delta\hat{\mathbb{U}}$ $\mathbb{N}_1 = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$\hat{U} = \mathbb{N}_2 \hat{\mathbb{U}}$ or $\delta\hat{U} = \mathbb{N}_2 \delta\hat{\mathbb{U}}$ $\mathbb{N}_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix}$
Constr.	$u_- \leq \begin{bmatrix} u_k \\ u_{k+2} \\ u_{k+4} \end{bmatrix} \leq u_+$ $y_- \leq \begin{bmatrix} y_{k+2} \\ y_{k+4} \\ y_{k+5} \end{bmatrix} \leq y_+$	$u_- \leq \begin{bmatrix} u_k \\ u_{k+1} \\ u_{k+3} \end{bmatrix} \leq u_+$ $y_- \leq \begin{bmatrix} y_{k+1} \\ y_{k+3} \\ y_{k+5} \end{bmatrix} \leq y_+$

Table 1 Shifting Strategy Example

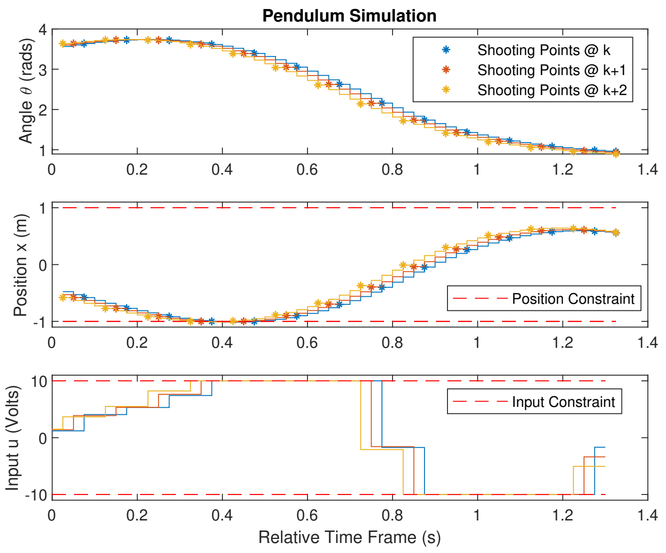

Fig. 2: Shifting Example with $N_B = 4$ starting from $j = 1$ of Definition 1, in relative time frames.

Algorithm 2. *Efficient H and G Computation*
In order to fill the H and G matrices efficiently, the following algorithm uses dummy variables T_G and T_H to calculate the required rows of the matrices recursively. Assuming $[T_G]_0 = [A_0]$ and $[T_H]_0 = [B_0]$ as starting points, it can be shown that all subsequent sub-matrices required by the H and G matrices are determined by:

$$[T_G]_k = A_k \, [T_G]_{k-1} \qquad [T_H]_k = \begin{bmatrix} A_k \, [T_H]_{k-1} & B_k \end{bmatrix} \qquad (48)$$
$$\forall k = [1, \cdots, N_p - 1]$$

The values are then assigned into the corresponding rows and columns of H and G, and only the rows of the "shooting points" are stored as discussed in section 3. In case the output has a nonlinear relation to the state, matrices T_G and T_H must first be multiplied (separately) with the appropriate C_k (see equations (7, 8)).

Once these matrices are filled, a significant number of computations can be avoided by taking advantage of the nature of the operations as in [7]. Furthermore, memory used should be pre-allocated avoiding dynamic memory allocation at all cost. Finally, it is important to NEVER repeat the same operation twice. Below we present a list of the computational savings that can be achieved.

1. The value of $H\mathbb{N}$ can be obtained by the summation of the respective columns in the H matrix. We will refer to this operation as $H\mathbb{N} = H_{\mathbb{N}}$.
2. The computation of the modified Hessian can be done as $E_{\mathbb{N}} = H_{\mathbb{N}}^T Q H_{\mathbb{N}} + \mathbb{N}^T R \mathbb{N}$.
3. The operation $\mathbb{N}^T R \mathbb{N}$ corresponds to the summation of penalization values of the corresponding blocked inputs. This values can be gathered in a vector $r_{\mathbb{N}}$ and added directly to the diagonal.
4. The operation $\mathbb{N}^T R \bar{U}$ represents the summation of the inputs in the block multiplied by the respective penalization.
5. Assuming Q is diagonal, the values on the diagonal (q) can be multiplied individually to the rows of $H_{\mathbb{N}}$ in $QH_{\mathbb{N}}$ operation of the Hessian [7]. We will refer to this operation as $H_{Q\mathbb{N}}$.
6. If Q is diagonal or symmetric, $Q^T = Q$, therefore $\mathbb{N}^T H^T Q = H_{Q\mathbb{N}}^T$.
7. Given the Hessian is symmetric, only the lower (or upper) triangular values need be calculated; the rest can be duplicated [7].
8. Given the Hessian is symmetric, a Cholesky decomposition is used to calculate the inverse of the Hessian efficiently by calculating only lower (or upper) triangular values and duplicating the rest.
9. An efficient version of Hildreth's QP provided [32] was developed, avoiding repeated computations by storing relevant results required by the optimisation.
10. The recovery of the original solution (full sized vector) from the blocked solution is done programatically, rather than through equations (18) or (17).

Based on the ideas presented in this section, the proposed strategy can significantly reduce the memory required for the optimisation. In particular, this allows the reduction of matrix $E \to E_{\mathbb{N}}$, gradient $f \to f_{\mathbb{N}}$, constraint matrix $M \to M_{\mathbb{N}}$, prediction matrix $H \to H_{\mathbb{N}}$, as well as constraint vector γ, nominal output vector $\bar{Y}$ and state-to-output prediction matrix G by selecting only the rows related to the shooting points. However, if the methodology is meant to be used to compress/decompress a given QP, as explained in section 2.5, the optimisation could add up to half the memory (depending on block size - half at $N_B = 2$) for storing the compressed QP matrices ($E_{\mathbb{N}}, f_{\mathbb{N}}, M_{\mathbb{N}}, \gamma$).

The ideas and computational savings gathered up to this section are implemented in the MATLAB and C++ codes given in [27] and [28] for the benchmark presented in the following section.

5 The Inverted Pendulum - A benchmark

The inverted pendulum is a nonlinear system widely used by academics, known to present several control challenges such as nonlinear and non-minimum phase dynamics, physical constraints and under-actuation (multiple outputs - single input), where the task is to drive the pendulum to its upright position, and simultaneously control its position in a rail. This makes it an interesting and challenging benchmark for NMPC.

The application of NMPC to a real inverted pendulum was succesfully achieved in [34] using modest hardware. An impressive application to a real triple inverted pendulum is presented in [38] where a nonlinear optimisation using a collocation points is to calculate the solution offline, however, they don't apply it in a receding horizon context nor do they apply it using the RTI scheme. Other authors have used it extensively for benchmark simulations such as [1, 6, 10, 13, 14, 20].

This section presents a benchmark of the proposed approach in a non-linear inverted pendulum for comparing the performance of the different strategies presented in this paper. The results show that by applying the proposed approach, the system achieves better performance and presents better recursive feasibility properties, which is a consequence of a consistent optimisation. Furthermore, it will be seen that the proposed approach can have computational gains up to 100 times faster on an i7 laptop, and up to 70 times faster on a relatively low power Linux-based embedded platform such as the aforementioned Beaglebone Blue, which would otherwise render the application of NMPC to this system unfeasible.

5.1 System Modeling

Several variations of the mathematical model of an inverted pendulum have been used, some of which are more complex than others. For our simulation, we used the mathematical model presented in [33] which contains two main non-linearities, namely, the gravitational effect, $g\sin(\theta)$, and the non-linear torque-relationship $\cos(\theta)u$ of the bar-link with the input u (or car acceleration $\ddot{p}=u$). The model is given by the following differential equations.

$$\begin{bmatrix}\ddot{\theta}\\ \ddot{p}\end{bmatrix}=\begin{bmatrix}-b\dot{\theta}+g\sin\theta+\cos\theta u\\ u\end{bmatrix} \tag{49}$$

Assuming the state $x=\begin{bmatrix}\theta & \dot{\theta} & p & \dot{p}\end{bmatrix}^T$, the system was simulated using a forward Euler integration method. Considering θ and p as the relevant outputs leads to the following linearisation matrices of the state space model (3):

$$A_k=\begin{bmatrix}1 & T & 0 & 0\\ T\alpha_k & (1-Tb) & 0 & 0\\ 0 & 0 & 1 & T\\ 0 & 0 & 0 & 1\end{bmatrix} \quad B_k=\begin{bmatrix}0\\ Tc\theta_k\\ 0\\ T\end{bmatrix} \quad C_k=\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 0 & 1 & 0\end{bmatrix} \tag{50}$$

where T is the sampling time, $c\theta_k=\cos\theta_k$, $s\theta_k=\sin\theta_k$ and $\alpha_k=gc\theta_k-s\theta_k u_k$. The parameters used were $b=0.3$ and $g=9.81$ (m/s^2), and the simulation was done using a sampling time of $T=0.025$ (s). Only the position of the car and the angle were considered as outputs ($n_y=2$) and the constraints of the system were considered as $-10\leq u\leq 10$ (m/s^2) and $-1\leq p\leq 1$ (m).

5.2 Simulations

All the simulation tests were done in the nominal case (no noise, no disturbances, no uncertainty) given that the prime interest is in the “inner” recursive feasibility, stability properties and computational efficiency; disturbance rejection and noise cancellation can be addressed separately using offset-free optimisations [18, 39] and observer/estimator design or filters [30–32, 40] respectively. The simulation was initialized with the state $\boldsymbol{x}=\begin{bmatrix}\pi & 0 & 0 & 0\end{bmatrix}$, and was run for $T_s=8$ seconds, allowing the system to swing up in "one shot" or "two shots" (see Figure 3). For the initial guess, a future nominal input trajectory of zeros $\bar{U}=\mathbb{O}$ was used which represents the free response of the system, a condition from which any optimisation could be initialised.

A desired prediction horizon of $N_{p_{des}}=50$ $(T_p=1.25$ $(s))$ was selected and the ideal prediction horizon was then acquired depending on the selected block size (N_B). For reference, the ideal horizon is displayed next to the block size in parenthesis in all the tables. Regarding the tuning parameters, the optimisation was done using $Q=I$ and an input penalization of $R=0.1I$. Moreover, a terminal cost (last 2 values in Q diagonal) of $Q_f=500$ was used for both, angle and position, as a "soft" zero-terminal constraint to improve the stability characteristics of the underlying optimisation.

5.2.1 Performance and Recursive Feasibility Comparison:

To assess the performance and recursive feasibility properties of the proposed approach, the system was tested with four possible types of solution (deviations/absolute with/without the proposed shifting strategy) and for different block sizes (N_B). Moreover, to compare the performance, two QP solvers were used in this comparison, namely the MATLAB R2018a quadprog function using the interior-point method and Hildreth’s QP presented in [32], which is an active-set primal-dual type of QP that allows for hot-starting the solution (initial guess for λ). In the former, the solution did not have a limit in iterations or time (solved to optimallity), and the latter performed a fixed number of 20 iterations (approximated solution) when the unconstrained solution did not satisfy the constraints. In the particular case of non-shifted solution based on deviations, the constraints remained in the full sized vector. Additionally, in the case of the MATLAB quadprog function, every time it returned an infeasibility flag, it was counted and the previous solution was used explicitly; this represents essentially open-loop control (no feedback), thus is a risk. The number of infeasibilities presented in a given type of solution is shown in brackets in table 2. Finally, the solution of the optimisation was always saturated to respect the input constraints regardless of the result from the QP.

Table 2 gathers the comparison of the costs for all the different types of solutions where $J_{Dev-Shift}$ represents the cost of the deviations solution with the proposed shifting strategy, $J_{Abs-Shift}$ the cost of the absolute solution, and so on. For reference, costs less than 2000 swung up the system in "one shot", costs greater than 2000 but less than 3000 swung up the system in "two shots", and costs above 3000 means the optimisation was not able to stabilize the system (see Figure 3). The following summarises the main results from Table 2.

1. In both QPs (quadprog and Hildreth’s), the results from the proposed deviation and absolute formulations with the shifting strategy are exactly the same (columns 1-2, and columns 5-6 equal). This is a direct result of what has been said repeatedly throughout the paper: consistency. Moreover, no infeasibilities were recorded for both types of solutions when using quadprog.
2. The solution giving deviations without shifting presented a significant number of infeasibilities (239), and worse performance when solving to optimality (quadprog) than when using an approximate solution (Hildreth’s). This is a direct result of inconsistency combined with the fact that Hildreth’s does not check for infeasibility and therefore feedback is always applied.
3. Although there were some differences in the results between both QP’s for the proposed approach (ie. columns 1-2 $\neq$ columns 5-6), most likely given that Hildreth’s QP did not converge to the solution in the 20 iterations (slow convergence rate of λ [32]), they gave similar results for all the cases (columns 1-2 $\approx$ 5-6).
4. Overall, the best "Total" cost is given by the proposed shifting strategy and the absolute non-shifted formulation gave the worse results.
5. For block sizes $N_B=10, 11$, non of the solutions was able to swing up the system in "one shot". This is unsurprising and linked to the obvious observation that there are sensible block sizes.
6. In both QP’s, suboptimalities $\Delta J=(J_{N_B}/J_1-1)\times 100<13.26\%$ where obtained through all the "one shot" solutions which give acceptable performance such as the ones given in both figures (3) and (4).
7. Notice block sizes $N_B=2, 7$ presented even better performance than the original full size vector $N_B=1$. This is because in the linearisation process, the optimisation might take a different "branch" of the solution that improved further AFTER re-linearisation. Moreover, allowing the intermediate constraints to be violated may relax the solution and lead to better performance at the cost of having to accept the violations. Finally, the optimisation is done in a finite horizon where both block sizes have slightly longer prediction horizon which could result in better overall predictions.

Figure 3 shows an example response with block size $N_B=6$ where it can be seen that the optimisation was able to swing up and stabilize the system in "one shot" using the proposed shifting strategy (both deviations/absolute giving same result). In contrast, it took "two shots" for the non-shifted solution based on deviations and the optimisation failed completely in the case of the absolute non-shifted solution.

Another value that was compared was the summation of the absolute violation to the position constraints for different block sizes (N_B). In other words:

$$\sum_{\forall k} v_k \qquad \text{where} \qquad v_k=\begin{cases}|p_k|-1 & \text{if} \quad |p_k|>1\\ 0 & \text{else}\end{cases} \tag{51}$$

Table 2 Cost comparison for different block sizes N_B using Shifting and Non-Shifting Strategies, and using quadprog MATLAB function and Hildreth's QP for solving the optimisation

QP	quadprog				Hildreth's			
N_B (N_P)	$J_{Dev-Shift}$	$J_{Abs-Shift}$	J_{Dev}	J_{Abs}	$J_{Dev-Shift}$	$J_{Abs-Shift}$	J_{Dev}	J_{Abs}
1 (50)	1101	1101	1101	1101	1102	1102	1102	1102
2 (51)	1099	1099	1105 [17]	1119[8]	1102	1102	1108	1130
3 (52)	1104	1104	2732 [18]	1239	1103	1103	1114	1238
4 (53)	1138	1138	2758 [32]	2475	1140	1140	1135	2787
5 (51)	1194	1194	1224 [14]	4118	1244	1244	1218	3952
6 (55)	1168	1168	2436 [18]	4002	1177	1177	1237	3715
7 (57)	1098	1098	2438 [26]	2197	1098	1098	1245	2201
8 (57)	1183	1183	2533 [38]	2173	1192	1192	1285	2293
9 (55)	1207	1207	2361 [20]	3514	1220	1220	2697	2230
10 (51)	2524	2524	2544 [27]	3475	2555	2555	2448	3474
11 (56)	2776	2776	2284 [15]	3448	2714	2714	4392	3447
12 (61)	1244	1244	4432 [14]	3397	1247	1247	2203	3399
Total	16835	16835	27948 [239]	32258 [8]	16894	16894	21185	30969

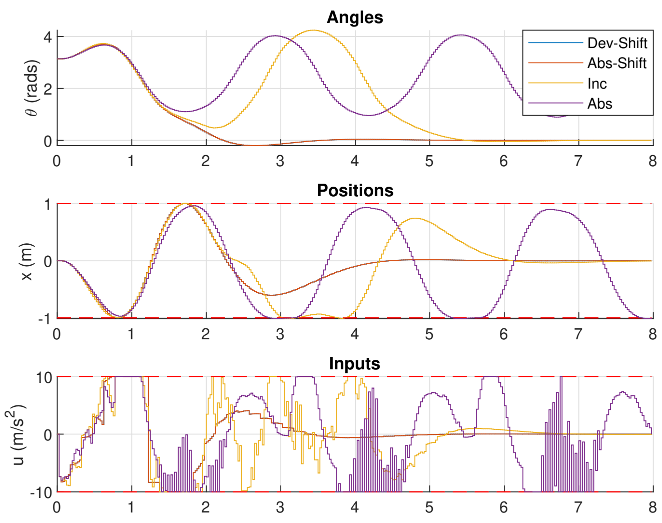

Fig. 3: Example Performance Comparison with $N_B = 6$ using quadprog

The results of this are gathered in Table 3 where $V_{T-Shift}$ represents the total violation of the proposed shifting strategy (deviations and absolute are the same), and V_{Dev} and V_{Abs} the total violation of the deviation and absolute non-shifting solutions, respectively. Additionally, given that the proposed shifting strategy is only supposed to enforce the constraints in the shooting points, the summation of the constraint violation at this particular points was stored separately and is represented by $V_{S-Shift}$ in the table. The following summarize the main results from Table 3.

1. In the full optimisation case (quadprog), there were no violations of the constraints at the shooting points ($V_{S-Shift} = 0$) when using the proposed strategy.
2. In the approximated optimisation case (Hildreth's), only 3 significantly small (1mm) violations occurred on the shooting points. Notice as the block size increases, the number of constraints in the optimisation are reduced. This ultimately allows the QP to find the active set in less iterations, resulting in no violations on the shooting points at bigger block sizes whilst performing slightly better because of the relaxation of the intermediate constraints.
3. Although the non-shifted solutions gave presumably "good" results for the Hildreth's case, they present significant cost suboptimalities. Moreover, the non-shifted deviation solution requires the full decision vector (not the blocked vector) to be constrained, thus removing part of the computational benefit.

To illustrate the concept of satisfying the constraints in the shooting points, Figure 4 shows the response of the system with block size $N_B = 12$ where it can clearly be seen that the solution satisfies the constraints (at the very limits) at the shooting points, which in this case are at times $t = [0.925, 1.225] = [3N_B + 1, 4N_B + 1]T$. The "extra step" in both shooting points is due to the computation separation strategy of the RTI which uses a predicted state, thus always optimising relative to "one step ahead" and applying the feedback phase in the next sampling time when the measurement of the state is available. This can clearly be seen in the input response where the first decision is at $t = 0.025\ (s)$ instead of $t = 0\ (s)$. Another important thing to notice is that the solution clearly exhibits the blocking structure, in particular, after $t > 3$ when the system is stabilised within the prediction horizon. Finally, a particular drawback of the proposed approach is that it only guarantees satisfying constraints at the shooting points by fixating all attention to them, therefore a small slack is required to protect the non-shooting points from small constraint violations. The selection of the slack size itself is a non-trivial task but could be selected based on Monte Carlo simulations, analyzing the system from a variety of conditions, and obviously would be increasing as the block size increases. In general, this is a problem from which most direct methods suffer because of the discretisation of the problem and thus is not unique to the proposal in this paper.

5.2.2 Computation Time Comparison: A core topic of interest is the computation time benefits achieved by the proposed strategy. Notice that the latter benefits from two main parts: (i) the first being the reduction in the number of degrees of freedom and points of interest which by itself would lend to faster unconstrained solutions, and (ii) the second being the reduction of the number of constraints which would otherwise lack the recursive feasibility guarantees if they were not reduced in the absolute-time-frame used by the proposed approach.

In order to properly test the computation times, the strategy was tailored to be executed with a given block size N_B and its ideal prediction horizon N_p which define the sizes of the matrices to be used. Each block size was programmed separately as a MATLAB function, and MATLAB C Coder was used to produce tailored C++ code to perform the optimisation. Regarding the QP iterations, the interest was on how fast the prepared QPs could be run, therefore the optimisation performed exactly 20 steps of Hildreth's QP when the unconstrained solution did not satisfy the constraints, independently of whether the active set was found or not. This is a realistic scenario given a situation may arise in a real system where the optimisation can only run for a given maximum number of iterations. Three execution times were of interest namely:

1. Preparation: The total time required to compute QP matrices $E_{\mathbb{N}}, f_{\mathbb{N}}, M_{\mathbb{N}}, \gamma$; referred to as QP_p in Table 4.
2. Unconstrained Solution: The preparation time plus obtaining the unconstrained solution and computing the feedback gain $K_x = E_{\mathbb{N}}^{-1} H_{\mathbb{N}Q}^T G$ to be used in the feedback phase; referred to as QP_u in Table 4.

Table 3 Constraint violation comparison for different block sizes N_B, using Shifting and Non-Shifting Strategies, and quadprog MATLAB function and Hildreth's QP for solving the optimisation

QP	quadprog				Hildreth's			
N_B (N_p)	$V_{T-Shift}$	$V_{S-Shift}$	V_{Dev}	V_{Abs}	$V_{T-Shift}$	$V_{S-Shift}$	V_{Dev}	V_{Abs}
1 (50)	0	0	0	0	0	0	0	0
2 (51)	0.002	0	0.004	0.006	0.002	0.001	0.002	0
3 (52)	0.009	0	0.017	0	0.008	0	0.010	0
4 (53)	0.002	0	0.064	0	0.002	0.001	0.012	0
5 (51)	0	0	0.044	0.106	0	0	0.030	0.040
6 (55)	0.003	0	0.094	0.178	0.003	0	0.040	0.301
7 (57)	0.171	0	0.300	0	0.171	0	0.089	0
8 (57)	0.114	0	0.423	0	0.108	0.001	0.021	0.094
9 (55)	0	0	0.406	0.059	0	0	0.100	0.008
10 (51)	0.032	0	0.686	0	0.026	0	0.334	0
11 (56)	0.069	0	0.313	0	0.156	0	0.214	0
12 (61)	1.055	0	0.986	0	1.033	0	0.055	0
Total	1.457	0	3.337	0.349	1.511	0.003	0.908	0.444

Table 4 Comparison of computation times (in microseconds) for different block sizes N_B in 32/64 bit formats on 2 different systems: Ubuntu 18.04 (Intel i7-5700HQ 64-bit @ 3.5 GHz) & Beaglebone Blue running RT Debian (ARM Cortex-A8 32-bit @ 1 GHz)

Sys	Ubuntu 18.04 (i7-5700HQ @ 3.5 GHz)						Beaglebone Blue (ARM Cortex-A8 @ 1 GHz)					
bits	64-bits			32-bits			64-bits			32-bits		
N_B (N_p)	QP_p	QP_u	QP_c	QP_p	QP_u	QP_c	QP_p	QP_u	QP_c	QP_p	QP_u	QP_c
1 (50)	165	276	2646	144	248	2291	6393	14326	108567	6689	14879	119865
2 (51)	41	60	427	35	53	415	2239	3459	19800	2144	3359	21586
3 (52)	26	34	185	22	30	172	1756	2204	8568	1598	2043	9211
4 (53)	21	25	85	18	22	77	1628	1863	5250	1456	1680	5100
5 (51)	17	19	57	15	17	52	1440	1575	3496	1372	1493	3493
6 (55)	19	21	52	16	17	46	1619	1729	3252	1390	1487	3077
7 (57)	19	21	45	17	18	41	1706	1794	2962	1655	1734	2996
8 (57)	18	19	39	16	17	35	1693	1761	2634	1426	1492	2439
9 (55)	17	18	33	15	15	29	1572	1625	2306	1447	1495	2235
10 (51)	14	15	26	12	13	23	1312	1352	1860	1146	1182	1717
11 (56)	17	18	29	14	15	26	1552	1593	2105	1366	1403	1938
12 (61)	20	20	32	17	18	29	1819	1861	2357	1602	1639	2172
Gain	12	18	102	12	19	100	5	11	58	6	13	70

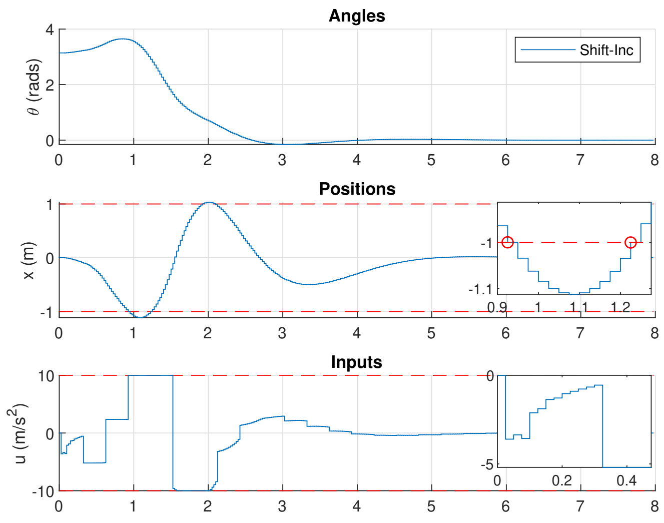

Fig. 4: Shifting Strategy Response with $N_B = 12$ using Hildreth's QP

3. Constrained Solution: The unconstrained solution time plus the time required to perform exactly 20 QP iterations of Hildreth's QP; referred to as QP_c in Table 4.

The resulting C++ code can be found in [28] and was tested on two different systems: a laptop running Ubuntu 18.04 with an i7-5700HQ 64-bits processor running @ 3.5 GHz with 12 GB of DDR3 RAM @ 1.6 GHz; and a Beaglebone Blue embedded platform running Real-Time (RT) Debian with an ARM Cortex-A8 32-bits processor running @ 1GHz. Given that the latter is a 32-bit system, the code was also produced and tested in 32-bits format on both systems. The simulation was done 1000 times and the minimum execution times for all cases were stored. This represents the fastest computation time that the approaches could obtain if a real time OS was used given the exact same computations/QP iterations are performed. The resulting computing times are gathered in Table 4 and the following summarises the main results.

1. For the constrained solution, the optimisation was able to get computation times QP_c up to 102 times faster in the laptop when using 64-bits, and 100 times faster when using 32-bits format. In the case of the embedded system (Beaglebone Blue), the optimisation was able to get computation time QP_c up to 70 times faster when using 32-bits, and only 58 times faster when using 64-bits.
2. Gains of up to 19 and 13 times faster were observed for the unconstrained solution, and up to 12 and 6 times faster for the preparation in the laptop and the embedded system respectively.
3. The fastest execution time in all the cases is obtained with block size $N_B = 10$. This is because this block size, has the smallest ideal prediction horizon ($N_p = 51$) with the smallest Hessian size of 6×6 ($\lceil \frac{51}{10} \rceil$) and only 24 constraints (6×4). In contrast, the optimisation with block size $N_B = 1$ has a Hessian size of 50×50 and 200 constraints (50×4).
4. The block size of $N_B = 1$ would render the application unfeasible in the embedded system (Beaglebone Blue) given that the optimisation wouldn't be able to finish within a sampling time ($T = 0.025$ (s)). However, in all the cases the computation time quickly drops to less than 8% with a block size of $N_B = 3$.

6 Conclusion

This paper presents a novel shifting strategy based on efficient blocked solutions for NMPC combined with the RTI Scheme. The proposed strategy uses a set of blocking structures which if applied sequentially automatically include the tail of the solution hence preserving recursive feasibility guarantees whilst reducing the degrees of freedom and the input-related constraints. Additionally, the proposed approach uses a reduced amount of points of interest, sometimes called shooting points, which represent output errors and output constraints that must be satisfied, and a stability and recursive feasibility guarantee is presented for the infinite horizon case, or for when the system includes special terminal conditions such as zero-terminal constraints or infinite horizon costing. Finally, it presents a set of algorithms and computational savings that can be used to code the proposed approach efficiently.

The overall resulting strategy is tested using an inverted pendulum as a benchmark where the proposed approach clearly outperforms the standard solutions in recursive feasibility properties and general performance, giving suboptimallities $\Delta J < 13\%$ and fully satisfying the constraints at the shooting points when a full optimisation is performed. Finally, the computational benefits of the proposed approach were evaluated in two physical systems: an i7 laptop and a Beaglebone Blue embedded system, where computation times up to 100 and 70 times faster were possible.

Future work related to the proposed strategy will be to merge the approach with the ACADO toolkit allowing it to use the efficient sensitivity generation and integration methods it contains, as well as a variety of QP solvers, and ultimately, the automatic code generation for its implementation in real robotic and mechatronic systems such as UAVs using, for example, the Beaglebone Blue Linux-based computing platform.

7 Acknowledgments

This work was funded by CONACyT, Mexico.

8 References

1 R. Quirynen, S. Gros, and M. Diehl, “Efficient NMPC for nonlinear models with linear subsystems,” *IEEE Conference on Decision and Control*, pp. 5101–5106, 2013.

2 H. Seki, S. Ooyama, and M. Ogawa, “Nonlinear Model Predictive Control Using Sucessive Linearization - Application to Chemical Reactors,” *Trans. of the Society of Instrument and Control Engineers*, vol. E-3, no. 1, pp. 66–72, 2004.

3 A. Zannelli, G. Horn, G. Frison, and M. Diehl, “Nonlinear Model Predictive Control of a Human-sized Quadrotor,” *European Control Conference*, vol. 16, no. 1, pp. 41–50, 2018.

4 Q. Mei, F. Xu, H. Chen, Z. Li, and Y. Hu, “Fast Model Predictive Control Based on Multiscale System Theory,” *World Congress on Intelligent Control and Automation*, pp. 2517–2522, 2016.

5 M. Diehl, H. G. Bock, and J. P. Schlöder, “A Real-Time Iteration Scheme for Nonlinear Optimization in Optimal Feedback Control,” *SIAM Journal on Control and Optimization*, vol. 43, no. 5, pp. 1714–1736, 2005.

6 B. Houska, H. J. Ferreau, and M. Diehl, “An auto-generated real-time iteration algorithm for nonlinear MPC in the microsecond range,” *Automatica*, vol. 47, no. 10, pp. 2279–2285, 2011.

7 M. Vukov, A. Domahidi, H. J. Ferreau, M. Morari, and M. Diehl, “Auto-generated algorithms for nonlinear model predictive control on long and on short horizons,” *IEEE Conference on Decision and Control*, pp. 5113–5118, 2013.

8 S. Gros, R. Quirynen, and M. Diehl, “Aircraft control based on fast non-linear MPC & multiple-shooting,” *IEEE Conference on Decision and Control*, no. 1, pp. 1142–1147, 2012.

9 M. Diehl, R. Findeisen, F. Allgöwer, H. G. Bock, and J. P. Schlöder, “Nominal stability of real-time iteration scheme for nonlinear model predictive control,” *IEE Proceedings-Control Theory and Applications*, vol. 152, no. 3, pp. 296–308, 2005.

10 S. Gros, M. Zanon, R. Quirynen, A. Bemporad, and M. Diehl, “From linear to nonlinear MPC: bridging the gap via the real-time iteration,” *International Journal of Control*, vol. 7179, no. November, pp. 1–19, 2016.

11 F. Guarantees, R. Quirynen, M. Diehl, and Q. Moritz, “An Efficient Inexact NMPC Scheme with Stability and Feasibility Guarantees,” *IFAC-PapersOnLine*, vol. 49, no. 18, pp. 53–58, 2016.

12 L. Wirsching, J. Albersmeyer, P. Kühl, M. Diehl, and H. Bock, “An Adjoint-based Numerical Method for Fast Nonlinear Model Predictive Control,” *IFAC Proceedings Volumes*, vol. 41, no. 2, pp. 1934–1939, 2008.

13 Y. Chen, D. Cuccato, M. Bruschetta, and A. Beghi, “An Inexact Sensitivity Updating Scheme for Fast Nonlinear Model Predictive Control based on a Curvature-like Measure of Nonlinearity,” *IEEE Conference on Decision and Control*, pp. 4382–4387, 2017.

14 A. Zanelli, R. Quirynen, G. Frison, and M. Diehl, “A Partially Tightened Real-Time Iteration Scheme for Nonlinear Model Predictive Control,” *IEEE Conference on Decision and Control*, vol. 1, no. 1, 2017.

15 J. Rossiter, L. Wang, and G. Valencia-Palomo, “Efficient algorithms for trading off feasibility and performance in predictive control,” *International Journal of Control*, vol. 83, pp. 789–797, April 2010.

16 D. Kouzoupis, R. Quirynen, B. Houska, and M. Diehl, “A block based ALADIN scheme for highly parallelizable direct Optimal Control,” *American Control Conference*, vol. 2016-, pp. 1124–1129, 2016.

17 D. Kouzoupis, R. Quirynen, J. V. Frasch, and M. Diehl, “Block Condensing for Fast Nonlinear MPC with the Dual Newton Strategy,” *IFAC-PapersOnLine*, vol. 48, no. 23, pp. 26–31, 2015.

18 R. Huang, L. T. Biegler, and S. C. Patwardhan, “Fast Offset-Free Nonlinear Model Predictive Control Based on Moving Horizon Estimation,” *Industrial & Engineering Chemistry Research*, vol. 49, no. 17, pp. 7882–7890, 2010.

19 B. Houska, H. J. Ferreau, and M. Diehl, “ACADO toolkit-An open-source framework for automatic control and dynamic optimization,” *Opt. Control Applications and Methods*, vol. 32, no. 3, pp. 298–312, 2011.

20 B. Houska, H. J. Ferreau, and M. Diehl, “Autogenerating microsecond solvers for nonlinear MPC: A tutorial using ACADO integrators,” *Optimal Control Applications and Methods*, vol. 36, pp. 685–704, 2015.

21 J. Kalmari, J. Backman, and A. Visala, “A toolkit for nonlinear model predictive control using gradient projection and code generation,” *Control Engineering Practice*, vol. 39, pp. 56–66, 2015.

22 H. J. Ferreau, C. Kirches, A. Potschka, H. G. Bock, and M. Diehl, “qpOASES: a parametric active-set algorithm for quadratic programming,” *Mathematical Programming Computation*, vol. 6, no. 4, pp. 327–363, 2014.

23 M. D. H.J.Ferreau, H.G. Bock, “An online active set strategy to overcome the limitations of Explicit MPC,” *International Journal of Robust and Nonlinear Control*, vol. 18, no. October 2014, pp. 816–830, 2008.

24 R. Quirynen, M. Vukov, and M. Diehl, “Multiple Shooting in a Microsecond,” in *Multiple Shooting and Time Domain Decomposition Methods*, vol. 9, pp. 183–202, Springer, Cham, 2015.

25 C. Shen, B. Buckham, and Y. Shi, “Modified C/GMRES Algorithm for Fast Nonlinear Model Predictive Tracking Control of AUVs,” *IEEE Transactions on Control Systems Technology*, vol. 25, no. 5, pp. 1896–1904, 2017.

26 R. Cagienard, P. Grieder, E. C. Kerrigan, and M. Morari, “Move blocking strategies in receding horizon control,” *Journal of Process Control*, vol. 17, no. 6, pp. 563–570, 2007.

27 O. J. G. Villarreal, “MATLAB Code for "A Shifting Strategy for Efficient Block-based Nonlinear Model Predictive Control Using Real Time Iterations.” `https://doi.org/10.24433/CO.0e3ceef1-9d0d-4bbc-b3ff-9fe6213ebc12`, 2018.

28 O. J. G. Villarreal, “C++ Code for "A Shifting Strategy for Efficient Block-based Nonlinear Model Predictive Control Using Real Time Iterations.” `https://doi.org/10.24433/CO.5d68cc1d-237e-4440-8e0e-dad909605e3f`, 2018.

29 Beagleboard.org, “Beaglebone Blue.” `https://beagleboard.org/blue`, 2017.

30 J. A. Rossiter, *Model-based predictive control : a practical approach.* Control series, Boca Raton: CRC Press, 2003.

31 J. A. Rossiter, *A first course in predictive control.* Boca Raton: CRC Press, Taylor & Francis, 2nd ed., 2018.

32 L. Wang, *Model Predictive Control System Design and Implementation Using Matlab.* Springer, 2009.

33 M. Alamir, “Fast NMPC: A reality-steered paradigm: Key properties of fast NMPC algorithms,” *European Control Conference*, no. 4, pp. 2472–2477, 2014.

34 A. Mills, A. Wills, and B. Ninness, “Nonlinear model predictive control of an inverted pendulum,” *American Control Conference*, pp. 2335–2340, 2009.

35 R. C. Shekhar and C. Manzie, “Optimal move blocking strategies for model predictive control,” *Automatica*, vol. 61, pp. 27–34, 2015.

36 C. J. Ong, Z. Wang, and M. Dehghan, “Model Predictive Control for Switching Systems With Dwell-Time Restriction,” *IEEE Transactions on Automatic Control*, vol. 61, no. 12, pp. 4189–4195, 2016.

37 J. A. Rossiter, J. Sheng, T. Chen, and S. L. Shah, “Interpretations of and options in dual-rate predictive control,” *Journal of Process Control*, vol. 15, no. 2, pp. 135–148, 2005.

38 T. Glück, A. Eder, and A. Kugi, “Swing-up control of a triple pendulum on a cart with experimental validation,” *Automatica*, vol. 49, no. 3, pp. 801–808, 2013.

39 J. Huusom, N. Poulsen, S. Jorgensen, and J. Jorgensen, “Tuning of methods for offset free MPC based on ARX model representations,” *American Control Conference*, pp. 2355–2360, 2010.

40 E. F. Camacho, *Model predictive control.* Advanced textbooks in control and signal processing, London ; New York: Springer, 2nd ed. ed., 2003.