Model predictive control based reference generation for optimal proportional integral derivative control

Fatos Gashi[1], Khalil Abuibaid[2], Martin Ruskowski[1,2], and Achim Wagner[1]

***Abstract*— We introduce an alternative approach towards optimal proportional integral derivative (PID) control, consisting of model predictive control (MPC) based reference generation. To this end, we have integrated the reference as part of optimization variables of the resulting problem, where a deliberate sequence of errors is induced to obtain an optimal PID control action. In addition, the desired behavior of the PID controller is achieved without the need for internal modification of the PID gains. To better highlight the ability of coping with poor PID tuning, several test cases consisting of progressively degraded PID gains are presented. Validation of the proposed strategy is displayed by comprehensive simulations using two different plants.**

I. INTRODUCTION

Proportional integral derivative (PID) controllers are ubiquitous in various control structures. The main reason comes from their simplicity, ease of implementation, and robustness. Although many other controllers outperform PID, due to the low computational effort, they remain the most common controller used to control a plant or a process, particularly on field controllers or other embedded devices.

The major drawback of PID controllers comes from performance. Apart from the inability to deliver optimal results, tuning them is not a straightforward process. On the other hand, usage of optimal controllers, e.g., model predictive control (MPC) based controllers is very compelling, in particular for control of industrial processes [1]. However, deployment of the latter controller also poses some limitations, notably in a hard real-time setup. One of the fundamental restrictions of MPC controllers comes from high computational efforts. It is not very infrequent to encounter control structures with limited access on the controller, in particular with industrial robots. Usually, the user provides only the reference, or rather the desired points, which then are modified into a suitable reference and fed to the low-level controller, commonly a PID type. To ensure smooth motions, a typical approach is trajectory approximation using polynomials of certain order [2], e.g., quintic polynomial approximation. To better emphasize the proposed control strategy, in the remainder of the paper it is assumed that no additional interpolation or approximation of the reference takes place, unless it is stated otherwise. With the purpose of maximizing PID performance, several approaches were developed, in particular the ones integrating PID controller in an MPC framework. In [3], PID parameters are continuously tuned using the recursive least square method. Another interesting method is introduced in [4], where PID parameters are calculated at each sampling time. By doing so, PID parameters are treated as decision variables in the overall optimization problem. In addition to this, a correction term is added. Similarly, [5] treats the PID parameters as decision variables for the resulting MPC problem. Both of these methods update PID parameters using Levenberg-Marquardt algorithm, and also both cases result in an increase of problem dimensions. Another method examining MPC-based auto-tuning of PID gains is presented in [6]. In [7], a stochastic system is controlled by a PID controller where its performance index is based on survival information potential. Unsurprisingly, methods where PID gains were included as decision variables (e.g., [4] and [5]) delivered the best performance in terms of error rates. Nevertheless, these methods are the most difficult ones to implement in standard field controllers.

Although in a quite different context and setup, it is noteworthy the idea of adjusting reference presented in [8]. The methods that exploit modification of reference to achieve certain system behavior, particularly in the presence of state and control constraints, are known as reference governors (RG). Since RG methods are capable of dealing with constraints, they are commonly deployed either as a substitution for MPC [9]-[11], or embedded within the MPC formulation [12]-[17]. All of these methods pose similar limitation on the reference dynamics, which ultimately might affect the optimality of PID controller.

In contrast, this work takes a slightly different path and addresses the possibilities for an alternative approach towards obtaining optimal PID control action by intentionally causing a specific error such that PID controller, which is an error-based method, delivers optimal results. At the same time, it is an effort to improve the overall monolithic architecture of a PID controlled plant where a desired behavior of PID is achieved without direct intervention within the PID itself.

The rest of the paper is organized as follows. The PID model alongside with the used plant models are introduced in section II. The MPC formulation setups are thoroughly described in III. Simulation results are presented in section IV. Last but not least, in section V conclusions are briefly summarised.

II. SYSTEM MODEL

Apart from assisting to illustrate the idea, we opted to simulate both a linear and a nonlinear system with the sole

[1]The author is with the German Research Center for Artificial Intelligence (DFKI), Kaiserslautern, D-67663, Germany

[2]The author is with the Chair of Machine Tools and Control Systems, Department of Mechanical and Process Engineering, RPTU Kaiserslautern-Landau, Kaiserslautern, D-67663, Germany

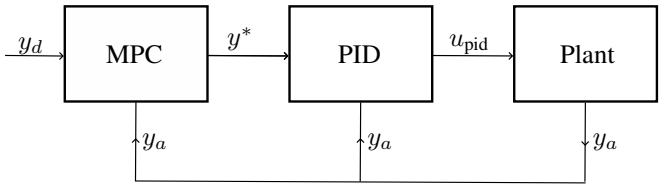

Fig. 1. Schematic illustration of common deployment of MPC controller in conjunction with a PID one. MPC block receives as input the desired set-point y_d for which the optimization takes place, then optimal reference y^* of measured variable is forwarded to the PID controller. The corresponding output of PID block takes into consideration difference between actual value of measured variable y_a and inputted reference y^*, value of which is ultimately used to control the plant.

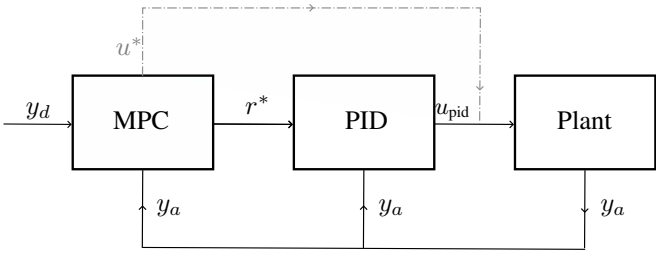

Fig. 2. Abstracted representation of the proposed method. Instead of MPC block providing optimal reference y^* for PID controller, it issues optimal reference r^* which will modify the error such that the equivalency between u^* and u_{pid} is enforced.

purpose of exposing the performance of suggested method in models where state variables have a simpler relationship, i.e., linear time-invariant (LTI), followed by one with a more complex relationship, often present in real-world scenarios.

A. PID Controller

An ordinary structure, surely with some level of abstraction, that makes use of MPC is illustrated in Fig. 1. The optimal output sequence of the MPC block, namely y^*, is exploited by PID in the form of a reference, based on which it will try to drive the system towards the desired point. In this configuration, MPC could be thought of as a higher layer of control, whereas PID is the lower one [3].

The underlying control law of a PID controller is described by [19]

$$u_{\text{pid}}(t) = u_0 + K_p e(t) + K_i \int_0^t e(\tilde{t})d\tilde{t} + K_d \dot{e}(t), \quad (1)$$

where u_0 is a constant, and K_p, K_i, K_d are the proportional, integral and derivative parameters of PID, respectively. Moreover, the error is defined as the difference between reference and measured process variable, in our case the actual output of the system, namely

$$e(t) = r(t) - y_a(t). \quad (2)$$

By using trapezoidal method for integration [19], the discrete-time version of (1) becomes

$$u_{\text{pid}}(k) = u(k-1) + K_1 e(k) + K_2 e(k-1) + K_3 e(k-2), \quad (3)$$

with

$$K_1 = K_p + \frac{K_i T_s}{2} + \frac{K_d}{T_s}, \;\; K_2 = -K_p + \frac{K_i T_s}{2} - \frac{2K_d}{T_s}, \quad (4)$$

and $\quad K_3 = \dfrac{K_d}{T_s}.$

In (4), sampling time T_s is the only varying parameter affecting error gains of (3). Since the focus of our approach is not on the possibility to dynamically reconfigure these parameters, they will remain fixed, value of which is found by any of the well-known methods of PID tuning [18].

In consideration of the foregoing, control input is highly dependent on actual error and its delayed versions, in this case up to the order of two (3). Thus, if one feeds an optimal reference for the controlled variable, most likely the PID controller will fail to deliver an optimal input sequence. Plain speaking, for the optimal output y^* there is a corresponding optimal input sequence u^*, which differs from u_{pid}.

To overcome such limitations, it is useful to include the dynamical constraints of PID controller (3) in the overall optimisation scheme. One of the possibilities would be to incorporate PID parameters as decision variables. Certainly, in a MPC framework this contradicts the above stated constraint on the inability to dynamically reconfigure PID parameters.

Alternatively, the only degree of freedom left on (3) is the reference $r(k)$. In order to obtain an optimal input sequence, the reference can be used as optimization variable, in other words, optimality is achieved by purposely inducing an error sequence such that the controlled process variable is driven to the desired point. By doing so, the optimal control sequence u^* is equal with the PID controller input, i.e., (3).

To further elaborate, the discrete time version of (2), which is a casual sequence, contains optimization variables $r(k)$, i.e.,

$$e(k) = r(k) - y_a(k), \quad (5)$$

and by substituting (5) into (3) we get the dependency between PID input and reference $r(k)$. With the proposed method, the analogous structure of Fig. 1 is shown in Fig. 2. The gray dashed branch is presented only to show equivalency between optimal control sequence u^* with u_{pid}.

B. LTI System - an academic example

The state evolution of discrete time (LTI) system under consideration is described by the following set of equations [19]

$$\mathbf{x}(k+1) = \mathbf{A}\mathbf{x}(k) + \mathbf{B}u(k), \quad (6)$$

with

$$\mathbf{A} = \begin{bmatrix} 0 & 1 \\ -0.74119 & 1.724 \end{bmatrix} \quad \text{and} \quad \mathbf{B} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad (7)$$

where the former one represents the state transition matrix and the later one the input matrix. Moreover, $\mathbf{x} \in \mathbb{R}^2$ and $u \in \mathbb{R}$.

The output of the system is defined by

$$y(k) = \mathbf{C}\mathbf{x}(k), \quad \mathbf{C} = \begin{bmatrix} 0.041 \\ 0.0453 \end{bmatrix}, \tag{8}$$

where $\mathbf{C}$ is the so-called output matrix.

C. Nonlinear system - Robotino

Robotino is a three-wheeled omnidirectional robot. From [20], the kinematic relationship between wheel angular velocities and global velocities is described by

$$\begin{bmatrix} \dot{x}(t) \\ \dot{y}(t) \\ \dot{\theta}(t) \end{bmatrix} = g\, \mathbf{R}(\theta)\, \mathbf{T}^{-1}(\beta_i) \begin{bmatrix} \omega_1(t) \\ \omega_2(t) \\ \omega_3(t) \end{bmatrix}, \tag{9}$$

where $\mathbf{x}^T(t) = \begin{bmatrix} \dot{x}(t) & \dot{y}(t) & \dot{\theta}(t) \end{bmatrix}$ is the robot's global velocity vector, g is the wheel radius, $\omega^T = \begin{bmatrix} \omega_1(t) & \omega_2(t) & \omega_3(t) \end{bmatrix}$ is a vector containing angular wheel velocities, β_i are the constant angles between wheels axis of rotations and $i \in \{1, 2, 3\}$ indicates the corresponding motor. The rotation matrix $\mathbf{R}(\theta)$ expresses orientation of the local frame with respect to global one, and is defined as

$$\mathbf{R}(\theta(t)) = \begin{bmatrix} \cos(\theta(t)) & -\sin(\theta(t)) & 0 \\ \sin(\theta(t) & \cos(\theta(t)) & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{10}$$

Similarly, the constant transformation from each wheel to the local frame is expressed by

$$\mathbf{T}(\beta_i) = \begin{bmatrix} -\sin(\beta_1) & \cos(\beta_1) & L \\ -\sin(\beta_2) & \cos(\beta_2) & L \\ -\sin(\beta_3) & \cos(\beta_3) & L \end{bmatrix}, \tag{11}$$

with L being the distance from each wheel and robot center.

The dynamic model of direct current (DC) motors used to drive the Robotino wheels is given by [21]

$$\begin{bmatrix} \dot{c}_1(t) \\ \dot{c}_2(t) \\ \dot{c}_3(t) \\ \dot{\omega}_1(t) \\ \dot{\omega}_2(t) \\ \dot{\omega}_3(t) \end{bmatrix} = \begin{bmatrix} -\frac{R}{L} & 0 & 0 & -\frac{k_m}{L} & 0 & 0 \\ 0 & -\frac{R}{L} & 0 & 0 & -\frac{k_m}{L} & 0 \\ 0 & 0 & -\frac{R}{L} & 0 & 0 & -\frac{k_m}{L} \\ \frac{k_t}{J} & 0 & 0 & -\frac{b}{J} & 0 & 0 \\ 0 & \frac{k_t}{J} & 0 & 0 & -\frac{b}{J} & 0 \\ 0 & 0 & \frac{k_t}{J} & 0 & 0 & -\frac{b}{J} \end{bmatrix} \begin{bmatrix} c_1(t) \\ c_2(t) \\ c_3(t) \\ \omega_1(t) \\ \omega_2(t) \\ \omega_3(t) \end{bmatrix} + \begin{bmatrix} \frac{1}{L} & 0 & 0 \\ 0 & \frac{1}{L} & 0 \\ 0 & 0 & \frac{1}{L} \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} u_1(t) \\ u_2(t) \\ u_3(t) \end{bmatrix}, \tag{12}$$

where $c_1(t)$, $c_2(t)$, and $c_3(t)$ are the respective motor currents, R is the resistance, L is the inductance, k_m is back electromotive force constant, k_t torque constant, and b viscosity friction coefficient. Moreover, the voltages (u_1, u_2, u_3) applied to DC motors are the means through which Robotino is being controlled.

The discrete counterpart of (12) is obtained by using truncated discretization method [22] of second degree (j=2),

$$\mathbf{A}_d = \sum_{i=0}^{j} \frac{T_s^i}{i!}\mathbf{A}^i, \text{ and } \mathbf{B}_d = \sum_{i=0}^{j-1} \frac{T_s^{i+1}}{i!}\mathbf{A}^i\mathbf{B}, \tag{13}$$

with $\mathbf{A}_d$ and B_d being the discrete state transition matrix and input matrix, respectively. For the kinematic equations (9), the well-known explicit Runge-Kutta method [23] of fourth order is used.

III. MPC FORMULATION

For the sake of simplicity we shall make use of the LTI system model, and the objective is to drive the systems output (8) to follow a given reference. The corresponding optimization problem can be formulated as

$$\min_{\mathbf{u},\mathbf{r}} \quad \sum_{k=1}^{N} \|y(k) - y_d(k)\|^2 \tag{14}$$

$$\text{s.t.} \quad \mathbf{x}(k+1) = \mathbf{A}\mathbf{x}(k) + \mathbf{B}u(k) \tag{14a}$$

$$\mathbf{x}_0 = \mathbf{x}(0) \tag{14b}$$

$$y(k) = g(\mathbf{x}(k), u(k)) \tag{14c}$$

$$u(k) = u(k-1) + K_1 e(k) + K_2 e(k-1) + K_3 e(k-2) \tag{14d}$$

$$\underline{\mathbf{x}} \leq \mathbf{x}(k) \leq \bar{\mathbf{x}} \tag{14e}$$

$$\underline{u} \leq u(k) \leq \bar{u}, \tag{14f}$$

where $k \in \{0, \ldots, N-1\}$ is the iteration index, N the prediction horizon, vectors $\mathbf{u} = \begin{bmatrix} u(0) & u(1) & \ldots & u(N-1) \end{bmatrix}$ and $\mathbf{r}^T = \begin{bmatrix} r(0) & r(1) & \ldots & r(N) \end{bmatrix}$ contain the corresponding decision variables, and lastly, y_d is the desired output. Though not explicitly shown, the decision variables of vector $\mathbf{r}$ are integrated in (14d). In addition, limits of state variables and control input are confined within (14e) and (14f), respectively.

In order to have better insight of the proposed method, in particular for comparison purposes, we have added two more existing approaches inspired from the work of [4] and [5], which will be presented in the sequel.

A. PID with fixed (optimal) gains during prediction horizon

Aside from classical methods of tuning parameters, there are existing strategies with PID parameters being varied over the time, i.e., auto tuning. From the perspective of MPC, this means that although K_p, K_i and K_d are held fix within the prediction horizon N, they are still part of decision variables, in other words the optimization problem (14) becomes

$$\min_{\mathbf{z} \in \mathcal{Z}} \quad J(\mathbf{z}) \tag{15}$$

where $\mathcal{Z}$ is a feasible nonempty set conform all of the constraints (14a)−(14d). Despite the fact that the objective function remains the same as in (14), the vector containing all decision variables, excluding $\mathbf{r}$, is appended with three more elements, namely

$$\mathbf{z}^T = \begin{bmatrix} \mathbf{u} & \mathbf{x} & K_p & K_i & K_d. \end{bmatrix}, \tag{16}$$

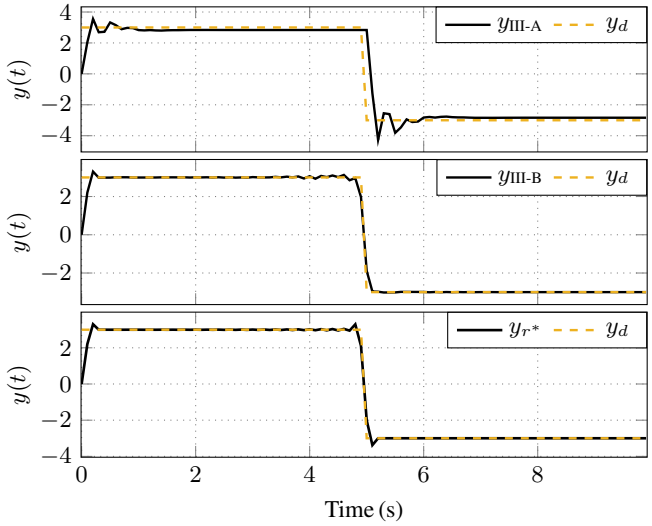

Fig. 3. LTI system outputs y for given desired reference y_d. The two upper plots correspond to formulation III-A and III-B, whereas the bottom plot corresponds to the proposed method. Length of the prediction horizon is N=100, sampling time is 0.1 s. PID gains obtained for III-A and III-Bn are part of the optimization problem, whereas for the proposed method the used PID gains are $K_p = 10.45$, $K_i = 10$ and $K_d = 10$.

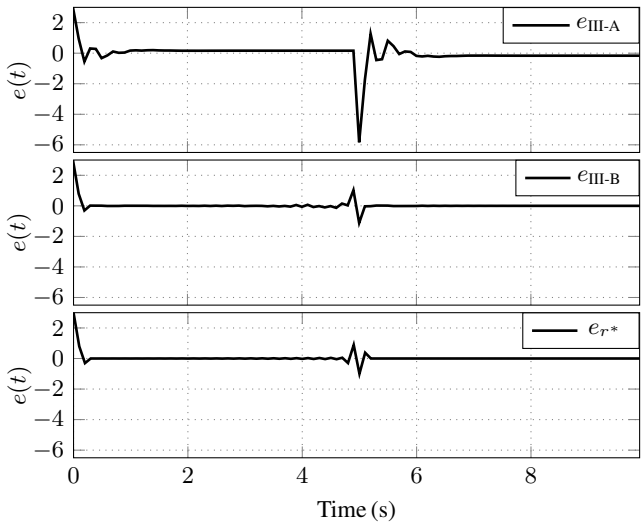

Fig. 4. Error e for the simulated LTI system. The top plot, which represents formulation III-A, is the one with worst performance, mainly due to the largest error values and inability to converge to zero. The middle and bottom plot manifest very similar behaviour in terms of reference following.

B. PID parameters as optimization variables

In contrast to the previous section, the PID parameters are varied along prediction horizon, in a similar fashion as $\mathbf{u}$. The resulting vector containing all decision variables is

$$\mathbf{z}^T = \begin{bmatrix} \mathbf{u} & \mathbf{x} & \mathbf{K}_p & \mathbf{K}_i & \mathbf{K}_d \end{bmatrix}, \tag{17}$$

with

$$\begin{aligned} \mathbf{K}_p^T &= \begin{bmatrix} K_p(0) & K_p(1) & \dots & K_p(N-1) \end{bmatrix} \\ \mathbf{K}_i^T &= \begin{bmatrix} K_i(0) & K_i(1) & \dots & K_i(N-1) \end{bmatrix} \\ \mathbf{K}_d^T &= \begin{bmatrix} K_d(0) & K_d(1) & \dots & K_d(N-1) \end{bmatrix}. \end{aligned} \tag{18}$$

By introducing PID parameters as decision variables during each time step within the receding horizon, the overall space of the feasible solution is increased due to added degrees of freedom coming from such parameters. Evidently, this requires fast dynamic reconfiguration of PID parameters, therefore it shall be used only to compare with the proposed approach. The rest of the optimisation problem remains the same as in (14).

IV. Simulation Results

In order to test the versatility and complexity reduction of the proposed method, we have devised several simulations. Due to poor performance, the formulation of III-A is not carried out for the Robotino case. Indeed there is room for improvement for this approach which may come from fine tuning, or slightly changing the formulation, however is not part of the scope for this work. Therefore, for the two simulated systems the emphasis will be on direct comparison of our proposed approach versus formulation from section III-B, with main focus on quality of solution (error rates), solving times, and complexity. To gain better insight on how increase of dimensions affects solving times, the simulations are carried out for prediction horizons of $N = 50$ up to $N = 250$, with increments of 25.

The MPC algorthitms are implementation in Python using CasADi framework [24], whereas the resulting optimization problems are solved using interior point optimizer (IPOPT) [25], with MA-57 from [26] as linear solver. The simulations are carried out on a standard Thinkpad machine with Intel Core i7-12800H processor, running on Ubuntu 22.04.3 LTS.

A. LTI System Results

The reference signal which is to be followed by LTI system output is of the form

$$y_d(t) = -3\,\mathrm{sgn}(t - t_0), \quad \mathrm{t} \geq 0, \tag{19}$$

where t_0 is the nearest integer of the half of signal duration. Sampling time is $T_s = 0.1$ seconds and all of the initial conditions are zero, namely

$$\mathbf{x}_0^T = \begin{bmatrix} 0 & 0 \end{bmatrix}. \tag{20}$$

The systems output is illustrated in Fig. 3 where the upper plot corresponds to III-A formulation, the middle plot to the formulation of III-B, and the bottom plot is formulated conform proposed method. The mismatches between desired systems output y and desired trajectory y_d around time $t = 5$ s are present in all plots of Fig. 3. As pointed out previously, the performance of III-A doesn't deliver satisfactory results. In terms of reference following, III-B and proposed method offer quite similar performance, depicted on the middle and bottom plot of Fig. 3.

Likewise, error plots of Fig. 4 offer similar insight, where most noticeable error is from the upper plot, which corresponds to III-A. Not only that it has larger values, but also the error fails to converge to zero on the steady state. The middle and bottom plot of Fig. 4 have similar performance

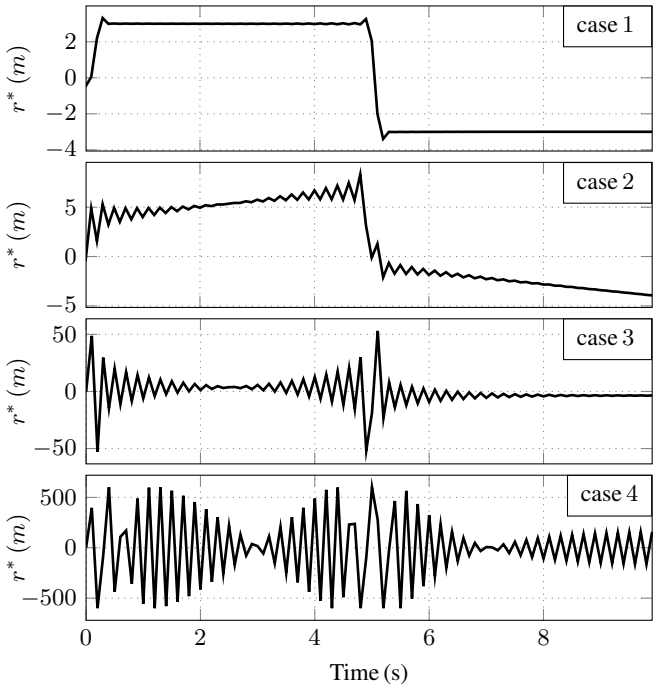

Fig. 5. Plots of optimal reference r^* for different for different PID gains, with horizon $N = 100$, sampling time $T_s = 0.1$. Each case corresponds to the PID gain values from Table I. Fast and high variations in the optimal reference can be an indication of ill-suited PID gain values.

TABLE I

PID GAINS.

Case	K_p	K_i	K_d	solving time (s)
1	100	100	100	0.004094387
2	0	0	1	0.003565975
3	1	0	0	0.003360563
4	0	1	100	0.005328638

and both of them peak around time $t = 5\,\text{s}$, which is the time that desired reference changes sign instantly.

With the intention of testing the robustness of our method with respect to PID gain variations, four cases are arranged. The exact values of PID gains are presented in Table I, with the respective plots of optimal reference displayed in Fig 5. Generally speaking, the cases where PID gains were nonzero resulted with smoother performance. Fluctuations of optimal reference increase drastically from case 1 up to case 4, with the latter one being the less suitable for real world systems. Judging by solving times for each of the cases, no specific comments can be made as all of them are more or less in the same order, thus no particular difficulty is manifested by the solver throughout the presented cases.

Another aspect to be considered is the computation time. In Table II are displayed computation times for both approaches and also for the same desired output as in (19). It is obvious the increase of computation time with increase of horizon length, however the increasing rate for III-B approach is higher than the proposed one. The ratio between computation times of III-B and proposed method are presented in the last column of Table II. For $N = 50$, the proposed method is approximately ten times (11.4) faster

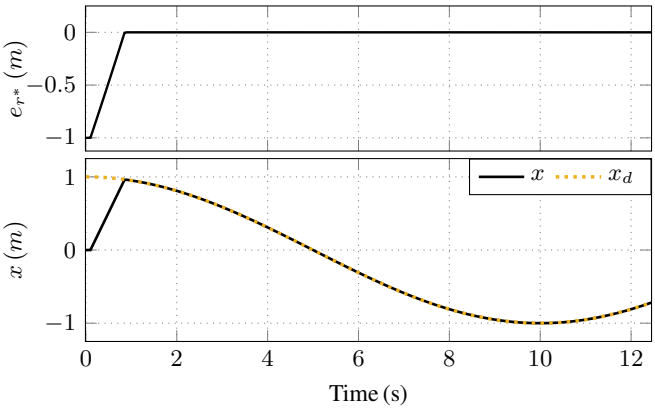

Fig. 6. Robotino trajectory along x-axis using the proposed approach, with prediction horizon $N = 250$ and PID gains $\mathbf{K} = \begin{bmatrix}10.45 & 10 & 2.5\end{bmatrix}$. Roughly, after a second the actual position of Robotino converges to the desired one.

TABLE II

LTI SYSTEM COMPUTATION TIMES.

Horizon (N)	Proposed method	III-B	Ratio
50	0.00270 (s)	0.03084 (s)	11.3937
75	0.00331 (s)	0.08040 (s)	24.2501
100	0.00404 (s)	0.16759 (s)	41.3868
125	0.00466 (s)	0.18969 (s)	40.6310
150	0.00525 (s)	0.35107 (s)	66.8326
175	0.00581 (s)	0.46025 (s)	79.1474
200	0.00529 (s)	0.64701 (s)	122.182
225	0.00715 (s)	0.50439 (s)	70.5418
250	0.00765 (s)	0.99705 (s)	130.208

than III-B, whereas for $N = 250$ this rate is approximately 130. This decrease in computation times can be attributed to the reduction of problem size. For LTI system, the reduction in the number of decision variables of proposed method in comparison to III-B, is approximately $\propto (2 \times (N-1))$.

In an MPC framework it is essential to have short solving times, in particular if it is intended for real-time control applications. If we consider the LTI system from II-B with sampling time $T_s = 0.1\,\text{s}$, based on solving times from Table II only first two prediction horizons $(50, 100)$ are feasible within an MPC setup as the solving times are less than sampling time T_s. On the contrary, all of the simulated prediction horizons of the proposed method from Table II are suitable for MPC setup as none of the solving times exceeds sampling time T_s.

B. Robotino simulation results

The aim is to make Robotino follow the desired trajectory which is given in the Euclidean space, and in our case is only along x-axis. The simulation setup consists of sampling time $T_s = 0.05$, and desired trajectory

$$\mathbf{x}_d = A\sin(2\pi f_0(T_s k)), \quad k \in \{0, 1, \dots\}, \tag{21}$$

with $A = 1$ and frequency $f_0 = \frac{1}{400}$ Hz. Further, Robotino's initial condition are all zero and together with above mentioned parameters, are kept unchanged throughout the entire simulations. In addition, the objective function is scaled

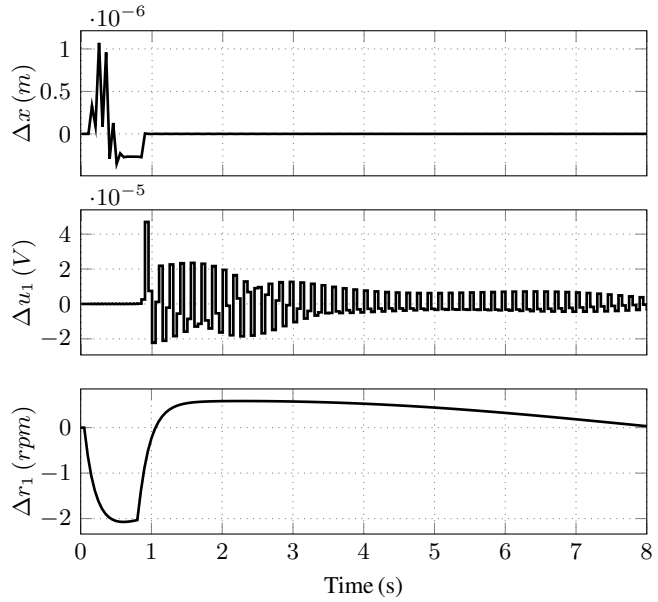

Fig. 7. Trajectory differences between proposed approach and III-B, with plots of Δx along x-axis and input values Δu_1. The plot of Δr_1 corresponds to the difference between optimal reference $\omega^*_{r_1}$ and optimal rotational speed of first motor ω^*_1.

TABLE III
NLP SOLVING TIMES.

Horizon (N)	Proposed method (r^*)	III-B	Ratio
50	0.0258 (s)	8.7922 (s)	340.783
75	0.0407 (s)	5.5750 (s)	136.977
100	0.0474 (s)	1.6440 (s)	34.683
125	0.0633 (s)	10.984 (s)	173.523
150	0.0792 (s)	8.9174 (s)	112.593
175	0.1000 (s)	5.7260 (s)	57.2600
200	0.1110 (s)	20.895 (s)	188.243
250	0.1350 (s)	30.013 (s)	222.318

with the matrix $\mathbf{Q} = \text{diag}([100, 10, 10])$ and PID gains $\mathbf{K} = \begin{bmatrix}10.45 & 2.5 & 1.43\end{bmatrix}$.

As depicted by Fig. 6, after nearly a second the error $e_{r^*}(t)$ converges to zero. Due to physical constraints, some time is required for Robotino to reach the desired reference. In regard to reference following, both approaches deliver almost the same results, which is better illustrated in the plots of Fig. 7. The difference between trajectories along x axis is very small (of the order 10^{-6}), and after one second Δx goes to zero. Similarly, differences on input Δu_1 experience small fluctuations, which are irrelevant due to small values (of order 10^{-5}). The bottom plot of Fig. 7 contains the difference between optimal reference and optimal rotation speed of first motor, namely $\Delta r = \omega^*_{r_1} - \omega^*_1$. While approaching to the end of simulation time, Δr as well approaches to zero. For this case also, values of Δr tend to be quite smooth, which is an indication of suitable PID gain values. Because of local frame choice, i.e., (11), and also desired path along x-axis, there is a symmetry between ω_1 and ω_3, thus only one of them is displayed. In regard to solving times, the same picture doesn't hold. Exhibited in Table III are solving times of the resultant nonlinear programming problem (NLP) with the same initial guess. While for the proposed method the solving time increases with the increase of horizon N, the same consistency isn't maintained in formulation III-B. Nonetheless, the proposed method resulted with shorter solving times in comparison to formulation III-B, as indicated by the last column of Table III which contains the solving time ratio between III-B and proposed method.

V. CONCLUSION AND FUTURE WORK

We present an alternative approach towards obtaining an optimal PID controller. For the proposed method, reference is incorporated in the optimization variables of the given optimal path following problem. The proposed controller is tested for two different systems, and compared against existing contemporary approaches. The simulations demonstrated a similar performance of the proposed approach in the context of path following problem, whilst reducing the solving times. A very compelling direction for future extensions would be further improvement of solving times so that validation of the presented approach takes place in a real plant.

ACKNOWLEDGMENT

This work was supported by the EU project EVENFLOW under Horizon Europe agreement No. 10107043, and also by the Federal Ministry for Economic Affairs and Climate Action (BMWK) of Germany within the scope of project TWIN4TRUCKS (FKZ: 13IK010F).

REFERENCES

[1] Vazquez, S., Rodriguez, J., Rivera, M., Franquelo, L. & Norambuena, M. Model Predictive Control for Power Converters and Drives: Advances and Trends. *IEEE Transactions On Industrial Electronics*. 64, 935-947 (2017)

[2] A. Tika, F. Gashi and N. Bajcinca, "Robot Online Task and Trajectory Planning using Mixed-Integer Model Predictive Control," 2022 European Control Conference (ECC), London, United Kingdom, 2022, pp. 2005-2011.

[3] Abdelrauf, A., Abdel-Geliel, M. & Zakzouk, E. Adaptive PID controller based on model predictive control. *2016 European Control Conference (ECC)*. pp. 746-751 (2016)

[4] Hong, X., Iplikci, S., Chen, S. & Warwick, K. B-Spline Neural Networks Based PID Controller for Hammerstein Systems. *Emerging Intelligent Computing Technology And Applications*. pp. 38-46 (2012)

[5] Cetin, M. & Iplikci, S. A novel auto-tuning PID control mechanism for nonlinear systems. *ISA Transactions*. 58 pp. 292-308 (2015).

[6] Na, M. Auto-tuned PID controller using a model predictive control method for the steam generator water level. *IEEE Transactions On Nuclear Science*. 48, 1664-1671 (2001)

[7] Zhang, J., Pu, J., Yin, X. & Ning, M. An improved approach to tuning MPC-PID controller parameters for non-Gaussian systems. *2017 Chinese Automation Congress (CAC)*. pp. 7040-7045 (2017)

[8] H. Hjalmarsson, M. Gevers, S. Gunnarsson and O. Lequin, "Iterative feedback tuning: theory and applications," in IEEE Control Systems Magazine, vol. 18, no. 4, pp. 26-41, Aug. 1998, doi: 10.1109/37.710876 .

[9] Bemporad, A. Reference governor for constrained nonlinear systems. *IEEE Transactions On Automatic Control*. **43**, 415-419 (1998)

[10] Gilbert, E., Kolmanovsky, I. & Tan, K. Nonlinear control of discrete-time linear systems with state and control constraints: a reference governor with global convergence properties. *Proceedings Of 1994 33rd IEEE Conference On Decision And Control*. **1** pp. 144-149 vol.1 (1994)

[11] Kolmanovsky, I., Garone, E. & Di Cairano, S. Reference and command governors: A tutorial on their theory and automotive applications. *2014 American Control Conference*. pp. 226-241 (2014)

[12] Bemporad, A., Casavola, A. & Mosca, E. Nonlinear control of constrained linear systems via predictive reference management. *IEEE Transactions On Automatic Control*. **42**, 340-349 (1997)
[13] Weiss, A., Baldwin, M., Erwin, R. & Kolmanovsky, I. Model Predictive Control for Spacecraft Rendezvous and Docking: Strategies for Handling Constraints and Case Studies. *IEEE Transactions On Control Systems Technology*. **23**, 1638-1647 (2015)
[14] Gupta, R., Kalabić, U., Di Cairano, S., Bloch, A. & Kolmanovsky, I. Constrained spacecraft attitude control on SO(3) using fast nonlinear model predictive control. *2015 American Control Conference (ACC)*. pp. 2980-2986 (2015)
[15] Leung, J., Permenter, F. & Kolmanovsky, I. A Computational Governor for Maintaining Feasibility and Low Computational Cost in Model Predictive Control. *IEEE Transactions On Automatic Control*. **69**, 2791-2806 (2024)
[16] Vrlić, M., Ritzberger, D. & Jakubek, S. Model-Predictive-Control-Based Reference Governor for Fuel Cells in Automotive Application Compared with Performance from a Real Vehicle. *Energies*. **14** pp. 2206 (2021,4)
[17] İşleyen, A., Wouw, N. & Arslan, Ö. From Low to High Order Motion Planners: Safe Robot Navigation Using Motion Prediction and Reference Governor. *IEEE Robotics And Automation Letters*. **7**, 9715-9722 (2022)
[18] Åström, K. & Hägglund, T. PID control. *IEEE Control Systems Magazine*. 1066 (2006)
[19] Kuo, B. Digital Control Systems. (Holt, Rinehart,1980)
[20] Eberhard, P. & Tang, Q. Sensor Data Fusion for the Localization and Position Control of One Kind of Omnidirectional Mobile Robots. *Multibody System Dynamics, Robotics And Control*. pp. 45-73 (2013)
[21] Tang, Q. & Eberhard, P. Cooperative Search by Combining Simulated and Real Robots in a Swarm under the View of Multibody System Dynamics. *Advances In Mechanical Engineering*. 2013 (2015,1)
[22] Rodríguez-Millán, J., Patete, A. & González, C. Picard Discretization of Nonlinear Systems: Symbolic or Numeric Implementation?. *Computer Aided Systems Theory – EUROCAST 2007*. pp. 121-129 (2007)
[23] Betts, J. Practical Methods for Optimal Control and Estimation Using Nonlinear Programming, Second Edition. (Society for Industrial,2010)
[24] Andersson, J., Gillis, J., Horn, G., Rawlings, J. & Diehl, M. CasADi – A software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*. 11, 1-36 (2019)
[25] Wächter, A. & Biegler, L. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*. 106, 25-57 (2006)
[26] Hopper, M. Harwell Subroutine Library. A Catalogue of Subroutines (1973). Supplement Number 2 (1973)