On Model Predictive Path Following and Trajectory Tracking for Industrial Robots

Mathias Hauan Arbo*and, Esten Ingar Grøtli† and Jan Tommy Gravdahl*

*Department of Engineering Cybernetics
NTNU, Norwegian University of Science and Technology
†Mathematics and Cybernetics, SINTEF DIGITAL, Trondheim, Norway

Abstract—**In this article the model predictive path following controller and the model predictive trajectory tracking controller are compared for a robotic manipulator. We consider both the Runge-Kutta and collocation based discretization. We show how path-following can stop at obstructions in a way trajectory tracking cannot. We give simulations for a two-link manipulator, and discuss the real-time viability of our implementations.**

I. INTRODUCTION

Modern industrial robots weld, grind, screw, measure, film, paint, pick and place, and perform other tasks that require the robot to follow arbitrary paths in space. For a typical robot cell, we simplify our work as robotics engineers by having enclosed, structured workspaces with no obstructions. For dedicated large-scale production of a small set of products, this is easily achieved. Small-scale manufacturing has a large variety of products and each product series is produced in low-scale or on-demand. Flexible robot cells are moved around in cluttered workspaces and require rapid prototyping of the paths. This means the control strategies must handle obstructions without sacrificing quality of the product.

We consider model predictive control (MPC), as it handles constraints on states and rudimentary obstructions. This article considers known obstructions, and focuses on the difference between path-following and trajectory tracking.

In trajectory tracking model predictive control (TT-MPC), the robot is to follow a path with an explicit path-timing. The trajectory may even incorporate constraints on the torques or velocities of the robot through optimal control problem (OCP) approaches such as [1], where a time-optimal path timing law under constraints is generated. The path timing law specifies the relation between the desired path and time, while accounting for state constraints during the execution of the path. The OCP is solved under the assumption that the robot is moving along the path, and is an open-loop approach to the constraint handling. In [2] this was extended to also allow constraints on the acceleration and inertial forces at the end-effector.

In [3], a model predictive path-following controller (MPFC) that handles both path-timing and error from path in the same OCP is described. In [4], the MPFC is shown to converge to the path given terminal constraints without needing terminal penalties. In [5] the MPFC is implemented

on a KUKA LWR IV robot, without end penalty or a terminal constraint. This is done with the ACADO framework [6], which uses a sequential programming method (SQP), iteratively solving quadratic programs approximating the nonlinear program using the qpOASES active set solver.

In [7], a real-time MPFC scheme for contouring control of an x-y table is described. Here a linear time varying approximation of the dynamics is used to define a QP which is solved using an active-set solver. In [8], this is implemented on an x-y table, and the MPFC outperformed both a similarly implemented TT-MPC, and the industry standard cascaded PI controlled set-point controller operating at a higher sampling frequency.

In [9], an MPFC is applied to a tower crane. The OCP is solved using the gradient projection method, an indirect method where Pontryagin's Maximum Principle is solved for the OCP without inequality constraints on the states. Instead slack variables are introduced to implicitly handle the inequality constraints.

In this article we

- draw attention to differences in MPFC and TT-MPC behavior, with and without obstructions,
- compare the Runge-Kutta and collocation integration method for the two strategies,
- solve the nonlinear programs (NLP) for the control strategies using the interior point solver IPOPT,
- and discuss the framework for real-time applications.

II. THEORY

A. System

The robot is an n degrees-of-freedom system with the state-space representation

$$\dot{\boldsymbol{y}}(t) = \boldsymbol{x}(t) \tag{1a}$$

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f_x}(\boldsymbol{y}(t), \boldsymbol{x}(t), \boldsymbol{u}(t)) \tag{1b}$$

where $\boldsymbol{y} \in \mathbb{R}^n$ are the generalized coordinates, $\boldsymbol{x} \in \mathbb{R}^n$ are generalized velocities, and $\boldsymbol{u} \in \mathbb{R}$ are the inputs. The function $\boldsymbol{f_x}$ describes acceleration. We assume the robot has known forward kinematics, allowing us to define a point of interest $\boldsymbol{p}$ on the robot such that

$$\boldsymbol{p}(t) = \boldsymbol{f_p}(y(t)) \tag{2}$$

where $\boldsymbol{f_p}(\cdot) \in \mathbb{R}^{n_p}$ is found from the forward kinematics. We describe the $\mathcal{C}^1$ reference path as $\boldsymbol{\varrho}(\cdot) \in \mathbb{R}^{n_p}$, defined in the same frame as $\boldsymbol{p}$.

The path-timing variable s moves from 0 to s_f. The TT-MPC assumes $s = t$ for $t < s_f$ and $s = s_f$ otherwise. The MPFC controls s through the path-timing dynamics, which we model as a double integrator

$$\begin{bmatrix} \dot{s}(t) \\ \ddot{s}(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} s(t) \\ \dot{s}(t) \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} v(t) := \boldsymbol{f_s}(s, \dot{s}, v), \quad (3)$$

with piecewise constant input $v(\cdot) \in \mathbb{R}$. To ensure that we never move backwards along the path, and that we have a maximum along path speed, we constrain $\dot{s} \in [0, \dot{s}_u]$. For more information on choice of the path-timing dynamics, we refer the reader to [4] and [9].

For the MPFC we define the extended state $\boldsymbol{\xi} = [\boldsymbol{y}^T, \boldsymbol{x}^T, s, \dot{s}]^T$, input $\boldsymbol{w} = [\boldsymbol{u}^T, v]^T$, and dynamics

$$\dot{\boldsymbol{\xi}}(t) = \begin{bmatrix} \boldsymbol{x}(t) \\ \boldsymbol{f_x}(\boldsymbol{\xi}(t), \boldsymbol{w}(t)) \\ \boldsymbol{f_s}(\boldsymbol{\xi}(t), \boldsymbol{w}(t)) \end{bmatrix} = \boldsymbol{f_\xi}(\boldsymbol{\xi}(t), \boldsymbol{w}(t)). \quad (4)$$

Similarly, for the TT-MPC we define the extended state $\boldsymbol{\chi} = [\boldsymbol{y}^T, \boldsymbol{x}^T]$ with dynamics

$$\dot{\boldsymbol{\chi}}(t) = \begin{bmatrix} \boldsymbol{x}(t) \\ \boldsymbol{f_x}(\boldsymbol{\chi}(t), \boldsymbol{u}(t)) \end{bmatrix} = \boldsymbol{f_\chi}(\boldsymbol{\chi}(t), \boldsymbol{u}(t)). \quad (5)$$

We define the deviation from the path as

$$\boldsymbol{e}_{pf}(t) := \boldsymbol{f_p}(\boldsymbol{y}(t)) - \boldsymbol{\varrho}(s(t)). \quad (6)$$

for the MPFC, and

$$\boldsymbol{e}_{tt}(t) := \boldsymbol{f_p}(\boldsymbol{y}(t)) - \boldsymbol{\varrho}(t) \quad (7)$$

for the TT-MPC. For s to converge to s_f we also define the path-timing error

$$e_s(t) := s(t) - s_f. \quad (8)$$

B. Optimal Control Problem

With the previously defined dynamics and errors, we can describe the OCP for the MPFC as

$$\min_{\boldsymbol{e}_{pf}, \dot{\boldsymbol{e}}_{pf}, e_s, \boldsymbol{u}, v} \int_{t_k}^{t_k+T} J_{pf}(\tau, \bar{\boldsymbol{\xi}}(\tau), \bar{\boldsymbol{w}}(\tau)) \mathrm{d}\tau \quad (9a)$$

s.t.:

$$\dot{\bar{\boldsymbol{\xi}}}(\tau) = \boldsymbol{f_\xi}(\bar{\boldsymbol{\xi}}(\tau), \bar{\boldsymbol{w}}(\tau)) \quad (9b)$$
$$\bar{\boldsymbol{\xi}}(t_k) = \boldsymbol{\xi}(t_k) \quad (9c)$$
$$\bar{\boldsymbol{\xi}}(\tau) \in [\bar{\boldsymbol{\xi}}_l, \bar{\boldsymbol{\xi}}_u] \quad (9d)$$
$$\boldsymbol{h}_c(\bar{\boldsymbol{\xi}}(\tau)) \leq 0 \quad (9e)$$

and for the TT-MPC as

$$\min_{\boldsymbol{e}_{tt}, \dot{\boldsymbol{e}}_{tt}, e_s, \boldsymbol{u}} \int_{t_k}^{t_k+T} J_{tt}(\tau, \bar{\boldsymbol{\chi}}(\tau), \bar{\boldsymbol{u}}(\tau)) \mathrm{d}\tau \quad (10a)$$

s.t.:

$$\dot{\bar{\boldsymbol{\chi}}}(\tau) = \boldsymbol{f_\chi}(\bar{\boldsymbol{\chi}}(\tau), \bar{\boldsymbol{u}}(\tau)) \quad (10b)$$
$$\bar{\boldsymbol{\chi}}(t_k) = \boldsymbol{\chi}(t_k) \quad (10c)$$
$$\bar{\boldsymbol{\chi}}(\tau) \in [\bar{\boldsymbol{\chi}}_l, \bar{\boldsymbol{\chi}}_u] \quad (10d)$$
$$\boldsymbol{h}_c(\bar{\boldsymbol{\chi}}(\tau)) \leq 0 \quad (10e)$$

where subscript u refers to the upper bounds on the states, subscript l refers to the lower bound, and $\boldsymbol{h}_c$ describes other constraints such as obstacles in the path. The bar denotes internal states of the MPC.

The cost integrands are defined as

$$J_{pf}(\tau, \bar{\boldsymbol{\xi}}(\tau), \bar{\boldsymbol{w}}(\tau)) = \frac{1}{2} \bar{\boldsymbol{e}}_{pf}(\tau)^T \boldsymbol{Q} \bar{\boldsymbol{e}}_{pf} + \frac{1}{2} \dot{\bar{\boldsymbol{e}}}_{pf}(\tau)^T \boldsymbol{Q}_d \dot{\bar{\boldsymbol{e}}}_{pf}$$
$$+ \frac{1}{2} \bar{\boldsymbol{u}}(\tau)^T \boldsymbol{R} \bar{\boldsymbol{u}}(\tau)$$
$$+ \frac{1}{2} q \bar{e}_s(\tau)^2 + \frac{1}{2} r \bar{v}(\tau)^2 \quad (11)$$

for the MPFC and

$$J_{tt}(\tau, \bar{\boldsymbol{\chi}}(\tau), \bar{\boldsymbol{w}}(\tau)) = \frac{1}{2} \bar{\boldsymbol{e}}_{tt}(\tau)^T \boldsymbol{Q} \bar{\boldsymbol{e}}_{tt} + \frac{1}{2} \dot{\bar{\boldsymbol{e}}}_{tt}(\tau)^T \boldsymbol{Q}_d \dot{\bar{\boldsymbol{e}}}_{tt}$$
$$+ \frac{1}{2} \bar{\boldsymbol{u}}(\tau)^T \boldsymbol{R} \bar{\boldsymbol{u}}(\tau) \quad (12)$$

for the TT-MPC. The matrices $\boldsymbol{Q}$, $\boldsymbol{Q}_d$, and $\boldsymbol{R}$ are positive definite. The scalars q and r are positive. We have included the derivative of the path deviation to reduce oscillations.

Solving the OCPs can be done by an indirect approach using Pontryagin's Maximum Principle as done in [9], or a direct approach as done in [7]. We will apply the direct simultaneous approach, reformulating the OCP as an NLP by discretizing the problem. The direct simultaneous approach is most common in real-time applications, with existing software support such as ACADO [6] and CasADI [10].

C. Nonlinear Program and Interior Point

In this section we only give the discretization of (9) as the TT-MPC is similar and simpler. To discretize the OCP we use two different integration methods: the 4th order Runge-Kutta (RK4), and collocation based on Lagrange polynomials with d Legendre points.

The control input is a piecewise continuous function, constant on intervals of length δ_t, which is the length of our timesteps. This gives us a horizon of length $N_T = T/\delta_t$. With the simultaneous approach we use the integration method between each time step, and constrain the result and next state to be equal.

1) Runge-Kutta: Given a δ_t, RK4 [11] gives us the equation

$$\bar{\boldsymbol{\xi}}_{k+1} = \bar{\boldsymbol{\xi}}_k + \frac{\delta_t}{6}(k_1 + 2k_2 + 2k_3 + k_4) = F(\bar{\boldsymbol{\xi}}_k, \bar{\boldsymbol{w}}_k) \quad (13)$$

with

$$k_1 = \boldsymbol{f_\xi}(\bar{\boldsymbol{\xi}}_k, \bar{\boldsymbol{w}}_k), \tag{14a}$$

$$k_2 = \boldsymbol{f_\xi}\left(\bar{\boldsymbol{\xi}}_k + k_1\frac{\delta_t}{2}, \bar{\boldsymbol{w}}_k\right), \tag{14b}$$

$$k_3 = \boldsymbol{f_\xi}\left(\bar{\boldsymbol{\xi}}_k + k_2\frac{\delta_t}{2}, \bar{\boldsymbol{w}}_k\right), \tag{14c}$$

$$k_4 = \boldsymbol{f_\xi}\left(\bar{\boldsymbol{\xi}}_k + k_3\delta_t, \bar{\boldsymbol{w}}_k\right). \tag{14d}$$

The resulting NLP is then

$$\min_{\boldsymbol{q}} \quad \phi(\boldsymbol{q}) \tag{15a}$$

s.t.:

$$\boldsymbol{f}_e(\boldsymbol{q}) = \boldsymbol{0} \tag{15b}$$

$$\boldsymbol{h}_e(\boldsymbol{q}) \leq \boldsymbol{0}, \tag{15c}$$

where $\boldsymbol{q} = [\bar{\boldsymbol{\xi}}_k^T, \bar{\boldsymbol{w}}_k^T, \dots, \bar{\boldsymbol{\xi}}_{k+N_T-1}^T, \bar{\boldsymbol{w}}_{k+N_T-1}^T, \bar{\boldsymbol{\xi}}_{k+N_T}^T]^T$, the cost function is approximated with the rectangle method

$$\phi(\boldsymbol{q}) = \sum_{j=k}^{k+N_T} \delta_t J_{pf}(t_j, \bar{\boldsymbol{\xi}}_j, \bar{\boldsymbol{w}}_j), \tag{16}$$

and

$$\boldsymbol{f}_e(\boldsymbol{q}) = \begin{bmatrix} \bar{\boldsymbol{\xi}}_k - \boldsymbol{\xi}(t_k) \\ \bar{\boldsymbol{\xi}}_{k+1} - F(\bar{\boldsymbol{\xi}}_k, \bar{\boldsymbol{w}}_k) \\ \vdots \\ \bar{\boldsymbol{\xi}}_{k+N_T} - F(\bar{\boldsymbol{\xi}}_{k+N_T-1}, \bar{\boldsymbol{w}}_{k+N_T-1}) \end{bmatrix}. \tag{17}$$

The inequality constraints use (9d)-(9e) enforced on $\bar{\boldsymbol{\xi}}_i$ for $i = k, \dots, k + N_t$.

2) Collocation: For the collocation method, with d collocation points, we define $j = 0, \dots, d$ Lagrange polynomials

$$L_j(\tilde{\tau}) = \prod_{r=0, r\neq j}^{d} \frac{\tilde{\tau} - \theta_r}{\theta_j - \theta_r} \tag{18}$$

where $\tilde{\tau} \in [0, 1]$, θ_0 is 0, and the other θ_i are Legendre collocation points. The approximation of the state trajectory between t_k and t_{k+1} is then

$$\bar{\boldsymbol{\xi}}(\tau) = \sum_{j=0}^{d} L_j\left(\frac{\tau - t_k}{\delta_t}\right)\bar{\boldsymbol{\xi}}_{k,j}, \text{ for } \tau \in [t_k, t_{k+1}] \tag{19}$$

where $\bar{\boldsymbol{\xi}}_{k,j}$ are optimization variables included in $\boldsymbol{q}$. Requiring the simultaneous constraint to hold, and the derivatives to fit on the Legendre points, we have

$$\bar{\boldsymbol{\xi}}_{k+1,0} = \sum_{j=0}^{d} L_j(1)\bar{\boldsymbol{\xi}}_{k,j} \tag{20}$$

$$\boldsymbol{f_\xi}(\bar{\boldsymbol{\xi}}_{k,j}, \bar{\boldsymbol{w}}_k) - \frac{1}{\delta_t}\sum_{r=0}^{d} \dot{L}_r(\theta_j) = 0, \text{ for } j = 1, \dots, d \tag{21}$$

where $L_r(1)$ and $\dot{L}_r(\theta_j)$ are independent of t_k and are precomputed. This gives a similar structure to (15) with the cost function evaluated with $\bar{\boldsymbol{\xi}}_{i,0}$ for $i = k, \dots, k + N_t$. We have chosen to evaluate (9d) at all states $\bar{\boldsymbol{\xi}}_{k,j}$ and the

nonlinear inequality constraints, (9e), at $\bar{\boldsymbol{\xi}}_{k,0}$. This reduces the computational burden, but allows the collocation points $\bar{\boldsymbol{\xi}}_{k,j}$ between t_k and t_{k+1} to violate the nonlinear inequality constraints. The optimization vector is of the form

$$\boldsymbol{q} = [\bar{\boldsymbol{\xi}}_{k,0}^T, \bar{\boldsymbol{\xi}}_{k,1}^T, \dots, \bar{\boldsymbol{\xi}}_{k,d}^T, \bar{\boldsymbol{w}}_k^T, \\ \dots, \bar{\boldsymbol{\xi}}_{k+N_T-1,d}^T, \bar{\boldsymbol{w}}_{k+N_T-1}^T, \\ \bar{\boldsymbol{\xi}}_{k+N_T,0}^T, \dots, \bar{\boldsymbol{\xi}}_{k+N_T,d}^T]^T \tag{22}$$

and equality constraint function

$$\boldsymbol{f}_e(\boldsymbol{q}) = \begin{bmatrix} \bar{\boldsymbol{\xi}}_{k,0} - \boldsymbol{\xi}(t_k) \\ \bar{\boldsymbol{\xi}}_{k+1,0} - \sum_{r=0}^{d} L_r(1)\bar{\boldsymbol{\xi}}_{k,r} \\ \boldsymbol{f_\xi}(\bar{\boldsymbol{\xi}}_{k,0}, \bar{\boldsymbol{w}}_k) - \sum_{r=0}^{d} \dot{L}_r(\theta_0)\bar{\boldsymbol{\xi}}_{k,0} \\ \vdots \\ \boldsymbol{f_\xi}(\bar{\boldsymbol{\xi}}_{k,1}, \bar{\boldsymbol{w}}_k) - \sum_{r=0}^{d} \dot{L}_r(\theta_1)\bar{\boldsymbol{\xi}}_{k,d} \\ \vdots \end{bmatrix}. \tag{23}$$

3) Interior point solver: Primal Interior point methods consider NLPs of the form

$$\min_{\tilde{\boldsymbol{q}}} \quad \phi(\tilde{\boldsymbol{q}}) - \mu\sum_{i=0}^{\tilde{n}} \ln(\tilde{\boldsymbol{q}}_n) \tag{24a}$$

s.t.:

$$\boldsymbol{f}_e(\tilde{\boldsymbol{q}}) = \boldsymbol{0} \tag{24b}$$

where $\tilde{\boldsymbol{q}}$ includes slack variables to make $\boldsymbol{h}_e$ an equality constraint, and μ defines the steepness of the barrier associated with the slack variables. For large values of μ the $\ln$ term dominates and the solution tends to the center of the feasible region. As μ decreases, ϕ dominates and the solution moves towards the optimal solution. Solving for decreasing μ will converge to the solution of (15). Interior point methods are difficult to warm-start, as a too low μ may make certain slack variables prematurely small and cause slow convergence. In the timing tests, we have not used warm start as the initial states are random, but in the MPC implementation we use the previous $\bar{\boldsymbol{\xi}}$ predictions as an initial guess.

We will use the interior point solver IPOPT [12], a primal-dual interior point solver, solving (24) using the primal-dual equations, see section 3.1 in [12]. Interior point methods have consistent runtime with respect to problem size, allowing us to potentially include more states with little effect on runtime.

Convergence of the MPFC is ensured using terminal sets and penalties as constructed in [4] where an example is given for the same system as ours with different parameters. In this article we do not consider the terminal cost and penalty.

III. SIMULATION

We consider a two-link manipulator. This can easily be extended to 6 degrees-of-freedom, and results here are indicative of the larger systems as interior point methods are consistent with respect to the number of variables. The system was implemented using Python and the CasADi framework [10]. CasADi allows us to define symbolic expressions for the

various equations in (9), and evaluate the derivatives using algorithmic differentiation, e.g. for RK4, which may be difficult to do by hand. The framework supports IPOPT [12].
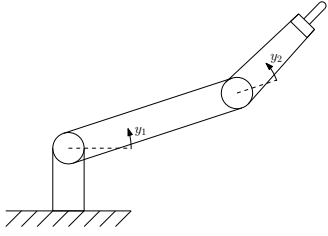
A. System

Fig. 1: Two-link manipulator with 2 revolute joints.

The robot has two links of length L_1 and L_2, with link masses M_1 and M_2 at L_{c1} and L_{c2}, and masses m_1 and m_2 at the joints. The point of interest is the tip of the end effector. The system is described by

$$\dot{\boldsymbol{y}} = \dot{\boldsymbol{x}} \tag{25a}$$

$$\dot{\boldsymbol{x}} = \boldsymbol{M}(\boldsymbol{y})^{-1}\left(\boldsymbol{u} - \boldsymbol{C}(\boldsymbol{y},\boldsymbol{x})\boldsymbol{x} - \boldsymbol{G}(\boldsymbol{y})\right) \tag{25b}$$

with

$$\boldsymbol{M}(\boldsymbol{y}) = \begin{bmatrix} a_1 + a_2\cos(y_2) & \frac{1}{2}a_2\cos(y_2) + a_3 \\ \frac{1}{2}a_2\cos(y_2) + a_3 & a_3 \end{bmatrix} \tag{26}$$

$$\boldsymbol{C}(\boldsymbol{y},\boldsymbol{x}) = \begin{bmatrix} -\frac{1}{2}a_2\sin(y_2)x_2 & -\frac{1}{2}\sin(y_2)(x_1 + x_2) \\ \frac{1}{2}a_2\sin(y_2)x_1 & 0 \end{bmatrix} \tag{27}$$

$$\boldsymbol{G}(\boldsymbol{y}) = \begin{bmatrix} g_1\cos(y_1) + g_2\cos(y_1 + y_2) \\ g_2\cos(y_1 + y_2) \end{bmatrix}, \tag{28}$$

where

$$a_1 = I_1 + I_2 + m_1 L_{c1}^2 + m_2(L_1^2 + L_2^2), \tag{29}$$

$$a_2 = 2m_2 L_{c2} L_1, \quad a_3 = m_2 L_{c2}^2 + I_2, \tag{30}$$

$$g_1 = (L_{c1}(m_1 + M_1) + L_1(m_2 + M_2))g, \tag{31}$$

$$g_2 = L_{c2}(m_2 + M_2)g. \tag{32}$$

For brevity we give $a_1, a_2, a_3, g_1, g_2 \in \mathbb{R}$ in Table I, for more information see [13]. The joint angles are defined as in Fig.1. The maximum torques are 30 Nm, the timesteps are $\delta_t = 0.01$ s, and if not otherwise specified the horizon is $T = 0.20$ s.

TABLE I: System parameters

Parameter	a_1	a_2	a_3	g_1	g_2
Value	0.5578	0.2263	0.0785	17.0694	4.3164

TABLE II: MPC Parameters

Parameter	$\boldsymbol{Q}$	$\boldsymbol{Q}_d$	$\boldsymbol{R}$	q	r
MPFC	$10^4\boldsymbol{I}_{2\times2}$	$10^1\boldsymbol{I}_{2\times2}$	$10^{-3}\boldsymbol{I}_{2\times2}$	1	10^{-3}
TT-MPC	$10^4\boldsymbol{I}_{2\times2}$	$10^1\boldsymbol{I}_{2\times2}$	$10^{-3}\boldsymbol{I}_{2\times2}$		

In order to study obstacle avoidance, we include obstacles o_i as bounding circles with known radius r_{o_i} and position $\boldsymbol{p}_{o_i}$. Their inequality equations are

$$\boldsymbol{h}_{o_i} = \left\| \boldsymbol{f_p}(\bar{\boldsymbol{\xi}}(t)) - \boldsymbol{p}_{o_i} \right\|^2 - r_{o_i}^2 > 0. \tag{33}$$

In actual applications a vision system would bound detected objects or people by a circle that the point of interest is not to enter. When present we consider two obstacles with: $r_{o_1} = 0.02$m, at $p_{o_1} = [0.55, 0.75]^T$, and $r_{o_2} = 0.04$m, at $p_{o_2} = [0.4, 0.4]^T$.

The reference path is a circle of radius 0.2 with center at $[0.55, 0.55]^T$.

B. Results

1) Moving to origin: In Fig.2 we see the Cartesian paths of the Runge-Kutta TT-MPC and MPFC. The black dot is $\varrho(0)$. For this simulation we used maximum joint speeds of 0.5π rad/s to exaggerate the differences. The set-point of the TT-MPC moves gradually while the TT-MPC is approaching the path. The MPFC on the other hand first approaches the origin of the path, then moves along it. If q is large compared to $\boldsymbol{Q}$, we will move along the path faster than to the path. The MPFC has come further with no difference in path deviation as $\dot{\bar{s}}$ is greater than the rate of t, allowing it to move faster along the path.

2) Obstacles: In Fig.3 two obstacles have been placed in the path, and the speed constraints are removed. Both MPFC and TT-MPC pass the first obstacle, but the MPFC stops at the second. The second obstacle is too large, and the horizon is not long enough to pass behind the obstacle. The MPFC will decrease $\dot{\bar{s}}$ to zero, see $t \approx 2$s in Fig.5, whereas the TT-MPC will have a gradually increasing cost as the trajectory set-point moves forward through the obstacle, forcing it around the object.

When obstructed, there was a difference between the two integration methods for the TT-MPC. We saw that the collocation method left the TT-MPC path closer to the obstacles, see Fig.4. The MPFC decreased $\dot{s}$ upon approaching the first object, at $t \approx 0.9$s in Fig.5, and was not affected by integration differences.

3) Timing: The simulations were performed on a Macbook Pro with a 2.5 Ghz i7 CPU. Using the compilation feature of CasADi we can create implementations that approach speeds needed for real-time systems. To compare the timings of the two integration methods we have performed a Monte-Carlo simulation of the MPFC with uniformly distributed initial positions in the upper right quadrant of the workspace. Box plot of the solver using RK4 for varying horizon lengths is given in Fig.6. In Fig.7 we give the same for the collocation method.

CasADi gives timing statistics of the solver. Upon inspection it appears that the collocation method has faster evaluation of the constraint functions, Hessian of the problem Lagrangian, and generally fewer iterations, but the increased optimization vector length makes the solver slower. In Table III we give typical timings of the solver. The cost function

and cost gradient are not included as they were the same and approximately 1 ms.
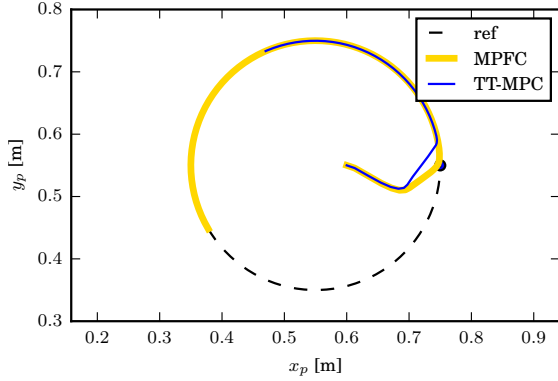

Fig. 2: TT-MPC and MPFC moving from the same start point towards the path. The blue dot is the start of the reference path.
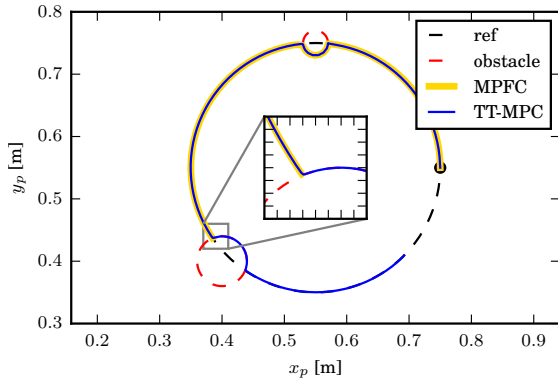

Fig. 3: TT-MPC and MPFC initialized close to the path origin. The path is obstructed by a small and a large object. Both controllers pass the first object, but the MPFC does not pass the second.

TABLE III: Typical timing statistics T = 0.6

	Constr.[1]	$\nabla_{\boldsymbol{q}}$Constr.	$\boldsymbol{H}(\mathcal{L})$[2]	Iter.	Solver tot.
MPFC-RK4	0.03 ms	0.16 ms	0.52 ms	27	0.045 s
MPFC-Col	0.03 ms	0.08 ms	0.15 ms	24	0.102 s
TT-MPC-RK4	0.03 ms	0.23 ms	0.66 ms	14	0.039 s
TT-MPC-Col	0.03 ms	0.07 ms	0.11 ms	14	0.046 s

IV. DISCUSSION AND FUTURE WORK

The collocation method has a sparse structure in the equality constraints, and relies on evaluation of $\boldsymbol{f_\xi}$, whereas the RK4 requires evaluation of its algorithimic differentiation.

[1]The constraint function include both the inequality constraints and the ODE dynamics.

[2]Hessian of the problem Lagrangian.
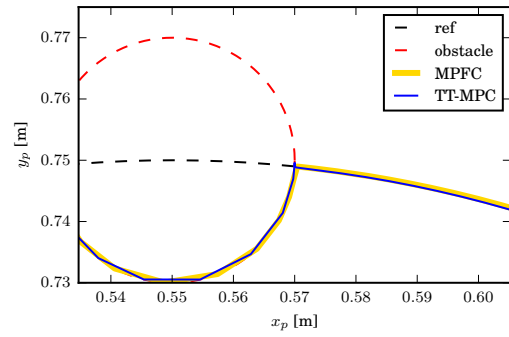
(a) Collocation method
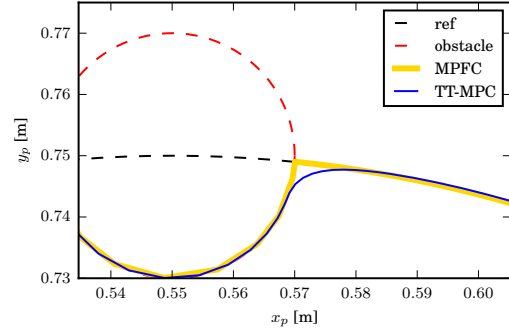
(b) RK4

Fig. 4: Difference between the two integration methods when the path is obstructed. Only evident in TT-MPC as the MPFC slows sufficiently down before the obstacle to not encounter the integration problems.
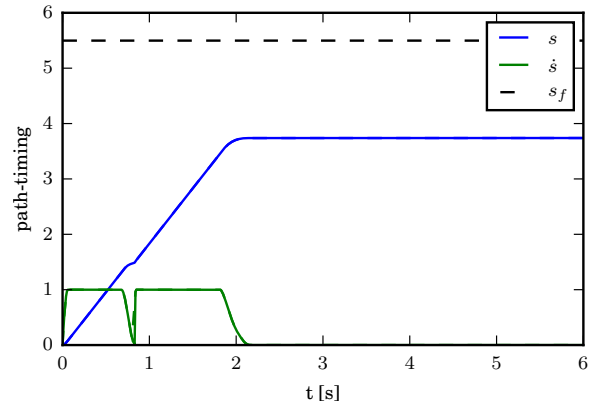

Fig. 5: Timing parameter s, blue line, and $\dot{s}$, green line, when MPFC follows the obstructed path. First object encountered at $t \approx 0.9$s, second object encountered at $t \approx 2$s.

This gives the collocation method faster function evaluations, but it did not appear to be sufficient to make the collocation method faster than RK4. The solver itself took more time with the increased number of states. For systems with complex dynamics the collocation method may be necessary. TT-MPC had fewer states, simpler dynamics and was faster
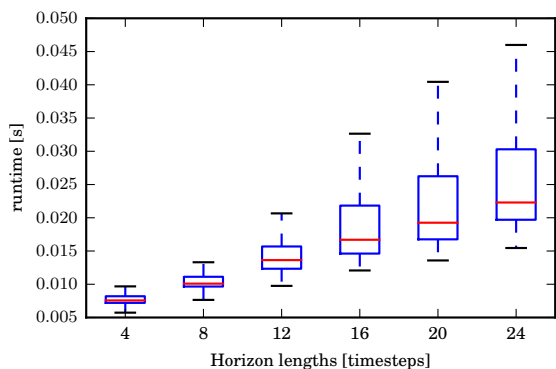
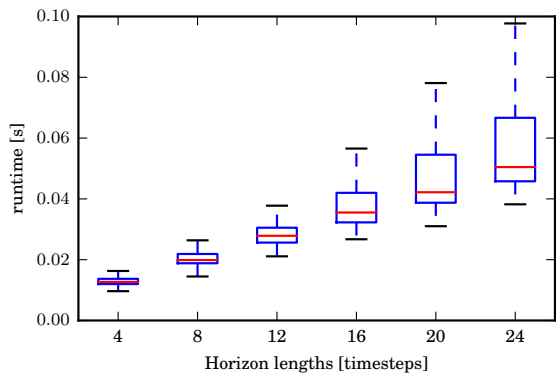Fig. 6: Boxplot of 2000 RK4 MPFC solver run times.

Fig. 7: Boxplot of 2000 collocation MPFC solver run times.

on the whole, with the same observations as for the MPFC regarding integration method.

The MPFC has the ability to stop along its path. For time-critical systems this is not desired, but for robots in open environments it can prove useful. It also suggests that when obstructed by an unknown object, it may push against the object with a constant force. In future experiments this will be investigated further. The TT-MPC will observe a growing difference in the current position and the desired position. With a known obstruction it will project the path onto the constraint attempting to minimize the path error. Suddenly removing the obstruction should result in the TT-MPC moving towards its current set point as fast as possible. With an unknown obstruction the TT-MPC may exert a gradually increasing force on the obstruction.

For the MPC to be real-time feasible, we require the solver to run faster than the control interval used. In this article we have considered a control interval of length $\delta_t = 0.01$ s. For low horizon lengths we are approaching such timing with the CasADi running in Python. Future work will extend this framework for a 6 degrees-of-freedom robot with a 3D path. The low horizon length needed to be able to achieve fast run time of the solver suggests that terminal constraints may be required in the final system.

The obstructions considered in this article were static and known apriori. Future work may include varying number of obstacles that enter the robot workspace.

V. CONCLUSION

The model predictive path-following controller gives rise to a set of new design opportunities. Of most value for obstructed environments is the fact that it may freely stop and resume along its path. The question is whether a constraint ends the path, as the path-following controller did, or whether the robot should move along the path projected onto the constraint, as the trajectory tracking controller did.

We also saw that the interior point method of IPOPT interfaced through CasADi in Python, approached timings we would desire in a real-time systems.

VI. ACKNOWLEDGEMENTS

REFERENCES

[1] D. Verscheure, B. Demeulenaere, J. Swevers, J. De Schutter, and M. Diehl, "Time-optimal path tracking for robots: A convex optimization approach," *IEEE Transactions on Automatic Control*, vol. 54, no. 10, pp. 2318–2327, 2009.

[2] F. Debrouwere, W. Van Loock, G. Pipeleers, M. Diehl, J. Swevers, and J. De Schutter, "Convex time-optimal robot path following with Cartesian acceleration and inertial force and torque constraints," *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, vol. 227, no. 10, pp. 724–732, nov 2013.

[3] T. Faulwasser, B. Kern, and R. Findeisen, "Model predictive path-following for constrained nonlinear systems," *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*, no. 3, pp. 8642–8647, 2009.

[4] T. Faulwasser and R. Findeisen, "Nonlinear Model Predictive Control for Constrained Output Path Following," *IEEE Transactions on Automatic Control*, vol. 9286, no. c, pp. 1–1, 2016.

[5] T. Faulwasser, T. Weber, P. Zometa, and R. Findeisen, "Implementation of Nonlinear Model Predictive Path-Following Control for an Industrial Robot," *IEEE Transactions on Control Systems Technology*, pp. 1–7, 2016.

[6] B. Houska, H. J. Ferreau, and M. Diehl, "ACADO Toolkit – An Open Source Framework for Automatic Control and Dynamic Optimization," *Optimal Control Applications and Methods*, vol. 32, no. 3, pp. 298–312, 2011.

[7] D. Lam, C. Manzie, and M. Good, "Model predictive contouring control," in *49th IEEE Conference on Decision and Control (CDC)*, vol. 86, no. 8. IEEE, dec 2010, pp. 6137–6142.

[8] ——, "Application of model predictive contouring control to an X-Y table," *IFAC Proceedings Volumes (IFAC-PapersOnline)*, vol. 18, pp. 10 325–10 330, 2011.

[9] M. Böck and A. Kugi, "Real-time nonlinear model predictive path-following control of a laboratory tower crane," *IEEE Transactions on Control Systems Technology*, vol. 22, no. 4, pp. 1461–1473, 2014.

[10] J. Andersson, "A General-Purpose Software Framework for Dynamic Optimization," PhD thesis, Arenberg Doctoral School, KU Leuven, Belgium, 2013.

[11] O. Egeland and J. T. Gravdahl, *Modeling and Simulation for Automatic Control*. Trondheim: Marine Cybernetics, 2003.

[12] A. Wächter and L. T. Biegler, *On the Implementation of Primal-Dual Interior Point Filter Line Search Algorithm for Large-Scale Nonlinear Programming*, 2006, vol. 106, no. 1.

[13] A. Astolfi, D. Karagiannis, and R. Ortega, *Nonlinear and Adaptive Control with Applications (Communications and Control Engineering)*, 2007.