SCISPACE
formerly Typeset

Open access • Proceedings Article • DOI:10.23919/ECC51009.2020.9143732

Real-time Robot Arm Motion Planning and Control with Nonlinear Model Predictive Control using Augmented Lagrangian on a First-Order Solver — Source link

Ajay Suresha Sathya, Joris Gillis, Goele Pipeleers, Jan Swevers

Institutions: Katholieke Universiteit Leuven

Published on: 12 May 2020 - European Control Conference

Topics: Solver, Nonlinear programming, Augmented Lagrangian method, Robotic arm and Sequential quadratic programming

Related papers:

- Real-Time Collision- Free Trajectory Optimization of Robot Manipulators via Semi-Infinite Parameter Optimization
- Nonlinear Predictive Control With End Point Constraints
- Constrained Motion Cueing for Driving Simulators Using a Real-Time Nonlinear MPC Scheme
- Structure-control dynamic design of parallel robots for end-effector trajectory tracking and singularity avoidance
- Trajectory tracking control of a 6-degree-of-freedom robot arm using nonlinear optimization

Share this paper:

View more about this paper here: https://typeset.io/papers/real-time-robot-arm-motion-planning-and-control-with-3lf6tbqpob

Real-time Robot Arm Motion Planning and Control with Nonlinear Model Predictive Control using Augmented Lagrangian on a First-Order Solver

Ajay Suresha Sathya[1], Joris Gillis[1], Goele Pipeleers[1] and Jan Swevers[1]

***Abstract*— In this work we implement motion planning and control of a robot arm with nonlinear model predictive control using the optimization algorithm PANOC. PANOC is a first order nonlinear optimization solver, with convergence guarantees, that is matrix-free unlike the popular sequential quadratic programming and nonlinear interior-point methods. We extend this solver to deal with hard constraints using an augmented Lagrangian method. This is used to implement a multiple-shooting MPC algorithm with collision avoidance capabilities on a robot arm. The computational time is benchmarked against other nonlinear optimization solvers. The algorithm is validated with simulations.**

I. INTRODUCTION

A. Background

Motion planning with obstacle avoidance for a robot manipulator is an important problem. It has applications in several cluttered environments where a robot needs to reach a goal position, such as an automatic warehouse, factory assembly shop or even a personal home chore robot. With humans increasingly sharing the workspace of the robot and considering the imperfections of the perception algorithms, it is vital that our motion planning and control algorithm is highly reactive to changes in the environment.

Manipulator motion planning with obstacle avoidance has been an active area of research and there are several paradigms for approaching this problem. Some early approaches included using an artificial potential field to repel the manipulator from the obstacle while attracting it towards the goal region at the same time [1]. Such methods are prone to get stuck in a local minima easily and follow very slow motions near the obstacles. Sampling based graph search methods such as Rapidly-exploring Random Trees (RRT) and Probabilisitic RoadMaps (PRM) [2], [3] provide probabilistic completeness guarantees, that is, the probability of finding a successful motion plan (provided that one exists) increases with the search time and approaches 1. Karaman and Frazzoli [4] also demonstrate that such planners are asymptotically optimal for a class of methods. But these methods suffer from the drawback of requiring an extra smoothing step to compute the trajectory before executing it. Optimization-based methods [5], [6] avoid this problem by directly computing a locally optimal, smooth path in the joint space. These methods are also faster than the sampling-based methods, but do not possess any completeness guarantees and might fail to find a solution, even if one exists, if the solver is stuck in a region of local infeasibility.

The potential field methods are fast and reactive but are suboptimal while the sampling and optimization-based methods are optimal but are not fast enough to react to changes in the environment in real-time as it requires replanning. Moreover both the sampling-based and the optimization-based motion planners compute only a joint path and therefore require a separate joint path following controller. Model Predictive Control (MPC) is a powerful strategy to address the aforementioned drawbacks. With MPC, we combine the task of motion planning and path following and we also obtain the reactivity because of the feedback given to the MPC controller.

The optimal control problem for robot manipulator trajectory generation requires the solution of a nonlinear program (NLP). Two of the standard methods for solving the NLPs are Interior-Point (IP) [7] methods and the Sequential Quadratic Programming methods (SQP) [8]. But these methods invoke a QP solver for each iteration which requires generating and solving a linear system of the KKT matrix. In this work, we employ a recently proposed first order solver proximal averaged Newton-type method for optimal control (PANOC) [9] to solve the problem. PANOC is a matrix-free method for non-convex problems that only requires vector-vector operations, that is shown to exhibit fast convergence [10], [11].

B. Contributions

The NLP formulation for robot motion control using MPC imposes constraints and costs (such as tracking error) on the end effector frame of the robot. This involves constraints and penalties of the robot kinematics which is a highly nonlinear function for a seven degree-of-freedom (dof) robot. In a single-shooting approach, such penalties at an instant are nonlinear functions of all the preceding input actions. This leads to a problem that is, too ill-conditioned to robustly achieve a feasible solution. While, in a multiple-shooting approach, constraints and penalties on the end-effector frame are a function of only the joint state at that instant and is not directly a function of the control actions. This decoupling

*The authors gratefully acknowledge support from Flanders Make SBO project - MULTIROB. Flanders Make is the Flemish strategic research centre for the manufacturing industry. This work also benefits from OPTEC: KU Leuven-BOF PFV/10/002 Centre of Excellence: Optimization in Engineering (OPTEC), and projects G0C4515N and G0A6917N of the Research Foundation - Flanders (FWO - Flanders).

[1]The authors are with MECO Research Team, Department of Mechanical Engineering, KU Leuven, C300 BE-3001, Belgium DMMS Lab, Flanders Make, Leuven, Belgium `firstname.lastname@kuleuven.be`

leads to an optimization problem that is better conditioned and therefore, a multiple-shooting approach is preferred.

These multiple-shooting constraints must be satisfied to high accuracy for a computed trajectory to be meaningful. But vanilla PANOC algorithm can only deal with hard constraints that lead to feasible sets that permit an efficient projection operation. In this work, we implement an augmented Lagrangian method on top of the vanilla PANOC algorithm to enforce both equality constraints and inequality constraints. This method is also used to enforce obstacle avoidance constraints.

The method is shown to be fast by benchmarking it against other solvers. The reactivity of the solver is studied on a case where the obstacle is moving and the robot needs to reactively avoid the obstacle.

II. NMPC FOR POINT-TO-POINT TRAJECTORY GENERATION

A. Problem Statement

Let $q \in \mathbb{R}^n$ denote the joint angles, where n refers to the number of degrees-of-freedom (DOF) of the robotic arm. Let T_{EE} refer to the End-Effector (EE) frame of the manipulator with respect to the inertial frame. The relationship between T_{EE} and the joint angles q is generally a nonlinear function and is denoted as follows:

$$T_{\mathrm{EE}} = f_{\mathrm{kin}}(q) \tag{1}$$

Let $\dot{q}$ refer to the joint velocities. Let the state of the control system denoted by x be defined as $x = [q, \dot{q}]^T$. Assuming an acceleration-resolved robot controller, let the control action $u = \ddot{q}$. The dynamical system is a double-integrator system and is given as follows:

$$\dot{x} = f_c(x,u) = \begin{bmatrix} \dot{q} \\ u \end{bmatrix} \tag{2}$$

This continuous-state system is discretized, assuming a piecewise constant control action, to obtain the discrete-time system with a sampling time of t_{s}. expressed as follows:

$$x_{k+1} = f_k(x_k,u_k) = x_k + \begin{bmatrix} \dot{q}t_{\mathrm{s}} + \frac{1}{2}ut_{\mathrm{s}}^2 \\ ut_{\mathrm{s}} \end{bmatrix} \tag{3}$$

The controller aims to compute a trajectory that takes the robot manipulator from a starting pose q_0 to a desired end-effector frame T_{goal}. The trajectory must prevent collision of the manipulator with obstacles that are denoted as $O_{kj} \subset \mathbb{R}^3$ for the jth obstacle at kth instant. These objects are described by open sets (possibly non-convex):

$$O_{kj}^{\mathrm{obs}} \subset \mathbb{R}^3 \tag{4}$$

The geometry of the manipulator is modelled as a union of convex objects and described as:

$$O_{kl}^{\mathrm{rob}}(x_k) \subset \mathbb{R}^3 \tag{5}$$

The distance between the manipulator object and environmental obstacle is defined as:

$$\mathrm{dist}(O_{kj}^{\mathrm{obs}}, O_{kl}^{\mathrm{rob}}(x_k)) = \min(\|p-q\|^2, p \in O_{kj}^{\mathrm{obs}}, q \in O_{kl}^{\mathrm{rob}}(x_k)) \tag{6}$$

The states x_k and the control actions u_k also have a feasible set denoted as:

$$x_k \in X_k \quad u_k \in U_k \tag{7}$$

where X_k and U_k are assumed to be closed compact and convex sets projecting onto which, is computationally inexpensive, such as box, balls or hyperplanes. In this work, X_k and U_k refer to the box-constraints corresponding to joint position and velocity limits and the joint acceleration limits respectively.

B. Nonlinear model predictive control

The nonlinear model predictive control for the trajectory planning problem can be formulated in the following form

$$\begin{aligned}
\textbf{minimize} \quad & \ell_N(x_N) + \sum_{k=0}^{N-1} \ell_k(x_k,u_k), && \text{(8a)}\\
\textbf{subject to} \quad & x_0 = x_{\mathrm{start}}, \quad f_{\mathrm{kin}}(q_N) = T_{\mathrm{goal}} && \text{(8b)}\\
& x_{k+1} = f_k(x_k,u_k), k \in \mathbb{N}_{[0,N-1]}, && \text{(8c)}\\
& u_k \in U_k, k \in \mathbb{N}_{[0,N-1]}, && \text{(8d)}\\
& x_k \in X_k, k \in \mathbb{N}_{[0,N]}, && \text{(8e)}\\
& \mathrm{dist}(O_{kj}^{\mathrm{obs}}, O_{kl}^{\mathrm{rob}}(x_k)) \geq 0, && \\
& k \in \mathbb{N}_{[0,N-1]}, l \in N_{[1,n]}, j \in N_{[1,n_{\mathrm{obs}}]} && \text{(8f)}
\end{aligned}$$

Here $\ell_k(x_k,u_k): \mathbb{R}^{n_x \times n_u} \to \mathbb{R}$ refers to the stage costs that is taken to be reference tracking error. The reference tracking error is described as a quadratic error on the translation and the orientation terms and also a quadratic penalty on the control calculated as follows:

$$\begin{aligned}\ell_k(x_k,u_k) = \|u_k - u_{\mathrm{goal}}\|_{R_k}^2 + \|p_k - p_{\mathrm{goal}}\|_{Q_{\mathrm{pos},k}}^2 + \\ \|\mathrm{diag}(C_k^T C_{\mathrm{goal}}) - 1\|_{Q_{\mathrm{rot},k}}^2\end{aligned} \tag{9}$$

where p_k refers to the translational terms and C_k refers to the rotation matrix or the direction cosine matrix in homogeneous transformation matrix for the EE at the kth instant, $T_k(x_k) = f_{\mathrm{kin}}(q_k)$.

C. PANOC Algorithm

The main features of PANOC algorithm [9] is summarized here. Let $o(z): \mathbb{R}^{n_z} \to \mathbb{R}$ be a function that is $C_{L_l}^{1,1}$. Let Z denote the feasibility set of z. One can define a projected gradient step as:

$$z^{\nu+1} = \Pi_Z(z^\nu - \gamma \nabla o(z^\nu)) \tag{10}$$

where Π denotes a projection operation to the feasible set Z and (10) always leads to a decrease in cost function if $\gamma < \frac{1}{L_l}$. Reaching the accumulation point of (10) can be equivalently viewed as fixed-point iteration. A series of iterates $z^\nu, z^{\nu+1}, \ldots$ and so on are used to implement a quasi-Newton method like the limited-memory BFGS (L-BFGS) method [12]. The L-BFGS method exploits the curvature information of the fixed point residual to speed up convergence. Globalization is achieved by using Forward-Backward Envelope (FBE) as a merit function [13], [14]. This FBE is real-valued and continuous and is proved to have the same minima as the original problem (10). For each descent step in PANOC, a linesearch is performed to find a convex combination of the quasi-Newton step and the projected-gradient step that ensures a decrease in the FBE thus providing PANOC with global convergence properties. Since L_l is, in general, not known in advance for a given function, an initial value L_l^0 is chosen for the Lipschitz constant and backtracking is performed to increase the value of L_l whenever the assumed value of the constant is found to be too small in a local region.

Therefore, PANOC is a first-order matrix-free solver for nonconvex optimization problems with favourable convergence properties. It can easily deal with hard constraints that have a feasibility set that permits a computationally simple projection operation. If not, one can resort to relaxation of the hard constraint to a soft constraint and use the penalty method when accurate constraint satisfaction is not a key requirement (such as the case of obstacle avoidance where the obstacles are enlarged to provide a tolerance against the constraint violation that is inevitable with the penalty method [10]).

D. Augmented Lagrangian Formulation

In [10] and [11], where PANOC was used for the motion planning of a cart-and-trailer system and a quadcopter respectively, the dynamics constraints of the form in (8c) were enforced using single-shooting. For the current problem, which consists of penalty terms that are highly nonlinear functions of the joint states, a single-shooting formulation was found to be unsuitable. The ill-conditioning of the formulation slowed down convergence and was even prone to get trapped in local infeasibilities thus failing to find a trajectory to the goal position.

For a multiple shooting formulation, the system dynamics in (8f) is treated as constraints of the nonlinear program (NLP). Using PANOC algorithm, such hard constraints are typically solved using the quadratic penalty method [8]. In the quadratic penalty method, the sum of square of the constraint violation is multiplied by a factor and added to the original cost function. This factor is increased sequentially in an outer iteration step to satisfies the constraints more closely. It is important for the multiple shooting constraints to be satisfied accurately. This calls for a high factor in the penalty method, which can however cause ill-conditioning and convergence issues. With the Augmented Lagrangian method (ALM) [15] , we try to satisfy constraints more accurately without increasing the factor to a very high value. In an ALM method, the quadratic penalty term is added to the Lagragian, thus "augmenting" it. Let $x = \begin{bmatrix} x_1 & x_2 & \dots & x_N \end{bmatrix}$ and $u = \begin{bmatrix} u_1 & u_2 & \dots & u_{N-1} \end{bmatrix}$ Let us define the equality constraints as the residual from 8b and (8c) as $h(x,u)$ and the inequality constraints from equations (8f) as $g(x,u)$. Let the total total objective function from (8a) be defined as:

$$Q(x,u) = \ell_N(x_N) + \sum_{k=0}^{N-1} \ell_k(x_k,u_k) \tag{11}$$

The augmented Lagrangian formulation for this problem, following the notation from [15], is defined as:

$$\mathscr{L}(x,u,\lambda,\mu,c_1,c_2) = Q + \lambda^T h(x,u) + \frac{1}{2}c_1\|h(x,u)\|^2 + \mu^T g^+(x,\mu,u) + \frac{1}{2}c_2\|g^+(x,\mu,u)\|^2 \tag{12}$$

where $g^+(x,\mu,u) = \max\{g(x,u), -\frac{1}{c_2}\mu\}$. The algorithm for minimizing the augmented Lagrangian is shown below in Algorithm 1.

Algorithm 1 Augmented Lagrangian algorithm for problem (12)

```
Require: $x_0$, $u_0$, $\lambda_0$, $\mu_0$, $c_{1_0}$, $c_{2_0}$, max_inner_iter, max_outer_iter, con_tol,
 1: for $\nu = 0, 1 \dots,$ max_outer_iter do
 2:     Minimize $\mathscr{L}(x_\nu, u_\nu, \lambda_\nu, \mu_\nu, c_{1_\nu}, c_{2_\nu})$ in variables $u$ and $x$ with PANOC until max_inner_iter to obtain $u_{\nu+1}$ and $x_{\nu+1}$
 3:     if $\max(\|h(x_{\nu+1},u_{\nu+1})\|_\infty, \|g(x_{\nu+1},u_{\nu+1})\|_\infty) \leq$ con_tol then
 4:         STOP and return $x_\nu$, $u_\nu$, $\lambda_\nu$, $\mu_\nu$
 5:     else if $\max(\|h(x_{\nu+1},u_{\nu+1})\|_\infty, \|g(x_{\nu+1},u_{\nu+1})\|_\infty) \leq 0.75\max(\|h(x_\nu,u_\nu)\|_\infty, \|g(x_\nu,u_\nu)\|_\infty)$ then
 6:         $\lambda_{\nu+1} = \lambda_\nu + c_{1_\nu} h(x_{\nu+1},u_{\nu+1})$
 7:         $\mu_{\nu+1} = \mu_\nu + c_{2_\nu} g^+(x_{\nu+1},u_{\nu+1},\mu_\nu)$
 8:         $c_{2_{\nu+1}} = c_{2_\nu}, \quad c_{1_{\nu+1}} = c_{1_\nu}$
 9:     else
10:         $c_{2_{\nu+1}} = 2c_{2_\nu}, \quad c_{1_{\nu+1}} = 2c_{1_\nu}$
11:         $\mu_{\nu+1} = \mu_\nu, \quad \lambda_{\nu+1} = \lambda_\nu$
12: return $x_{\nu+1}, u_{\nu+1}, \mu_{\nu+1}, \lambda_{\nu+1}$
```

III. SIMULATIONS

The controller is validated on a KUKA LBR robot manipulator with 7 DOF shown in the Figure 1. The goal position for the EE is shown as a green cuboid under the robot end-effector. The obstacle to be avoided is modelled as a black coloured ball. The MPC algorithm that is explained in the section II is executed for this task. MPC sampling time is taken to be 0.2 seconds with a horizon of 20 steps, thus providing a prediction horizon of 4 seconds. The MPC controller itself is however run at a rate of 50Hz. The joint states are read (velocities are computed) from the robot once every 20 ms and is taken as the starting point for the MPC solver which recomputes the control action every 20ms. The MPC output is then applied to the simulated robot model to obtain the joint states and the velocity after 20 ms. To this value a Gaussian noise with standard deviation of 10^{-4}

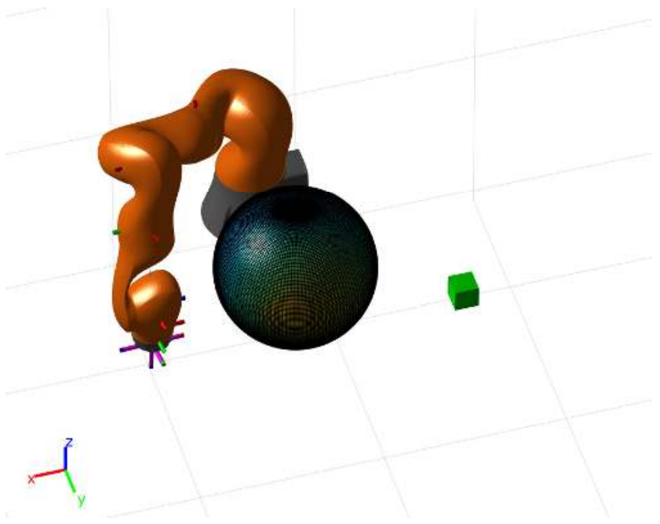


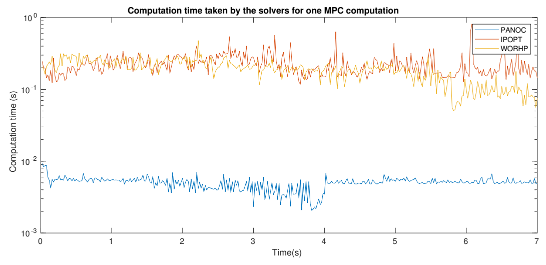
Fig. 1: The KUKA iiwa robot on which simulations are performed

is added before being fed back to the MPC controller to simulate the real world noise and disturbances.

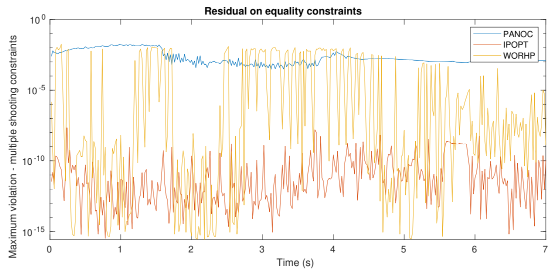
PANOC with ALM is benchmarked against two other algorithms: IPOPT [7], which is an interior-point solver and WORHP [16] which is an SQP solver. PANOC with ALM is allowed a maximum of 3 outer ALM iterations and 30 inner PANOC iterations. Optimization engine (OpEn) implementation of PANOC is used in this work. Interior-Point solver is called with the option of L-BFGS hessian approximation because it was found to be faster than the default BFGS approximation. A buffer size of 10 is used for the L-BFGS memory of both the PANOC and IPOPT solvers. WORHP was used with a full hessian computation. The maximum SQP iterations of WORHP is limited is the spirit of the real-time iteration scheme [17]. All the simulations were done on a system with Intel i7-8850H CPU @ 2.60GHz $\times$ 12 running an Ubuntu 16.04 operating system.

A. *Stationary Obstacle*

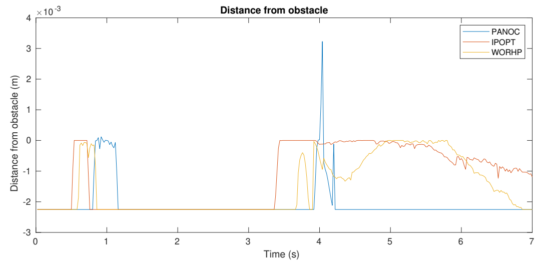
In this case the robot plans and executes a trajectory from the starting position shown in the Figure 1 to the green cuboid. The computation times required for different solvers is shown in the Figure 2a. One can see that PANOC is orders of magnitude faster than both WORHP and IPOPT solvers. Unlike PANOC or WORHP, the maximum number of iterations of IPOPT is not restricted, but is instead solved till it converged to a tolerance of 10^{-3}. This is because, IPOPT being a barrier method, is not suitable for real-time iteration. The maximum number of iterations in WORHP per MPC step was restricted to 6 because it was found that the WORHP solver needed atleast 6 SQP iterations to reach the goal location within the allocated time. WORHP was found to be as slow as IPOPT mainly because WORHP computed a full hessian which alone took about half of the total time taken by WORHP. Hence the time taken by WORHP can be reduced by almost half through appropriate code-generation methods for computing the Hessian so that solving the KKT



(a) Computation time for stationary obstacle avoidance case



(b) Maximum constraint violation of the multiple-shooting constraints in stationary obstacle case

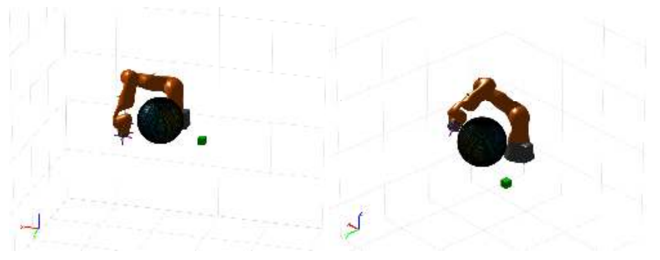


(c) Distance from the stationary obstacle

Fig. 2: Computation time and primal feasibility residuals for the motion planning for the stationary case

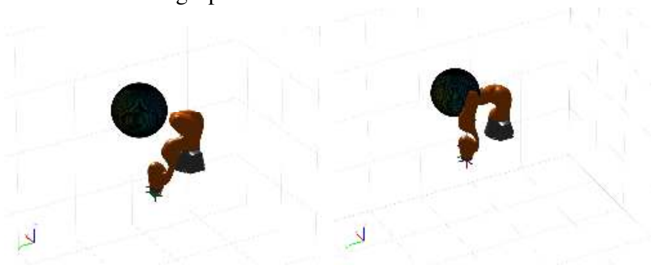
system becomes the bottle-neck, but it would still be an order of magnitude slower than PANOC. The mean, maximum and the standard deviations of the computation times can be seen in the table III-A. PANOC has a mean time of 4.98 ms with a standard deviation of 1 ms. Even the maximum time taken by PANOC is 9.27 ms and does not exceed 20 ms. Therefore, it is highly suitable for deployment on a robot with an MPC controller rate of 50 Hz. the other two solvers take over a 100 ms for each MPC computation and are hence not suitable for deployment on a real robot.

Figure 2b shows the maximum residual of the equality constraints for the three solvers. It is is very important for this residual to be low for the multiple-shooting constraints to lead to a meaningful trajectory. PANOC with ALM understandably performs the worst in this regard because it does not provide superlinear convergence to the optimal primal-dual point. But the important thing to note is that the constraint satisfaction is still sufficiently satisfied for robot MPC purposes with the violation being around 10^{-3}. While IPOPT consistently computes trajectories that closely satisfy the equality constraints, the performance of WORHP solver is more inconsistent in this regard and is sometimes worse than PANOC as well. This is likely due to 6 SQP iterations not



(a) The robot is in the starting position and needs to reach the goal pose denoted by the green box while avoiding the black coloured ball the is moving upwards.

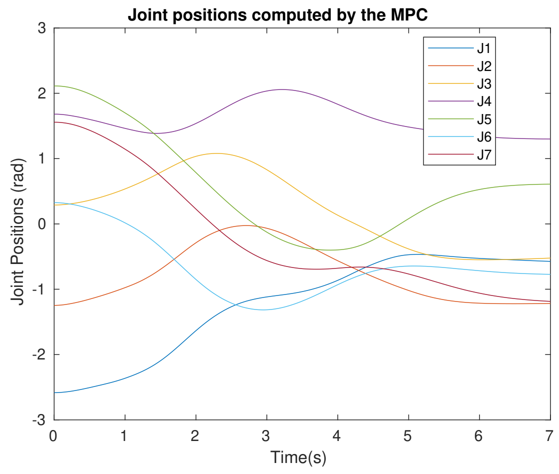
(b) The robot is crossing over the moving obstacle avoiding it



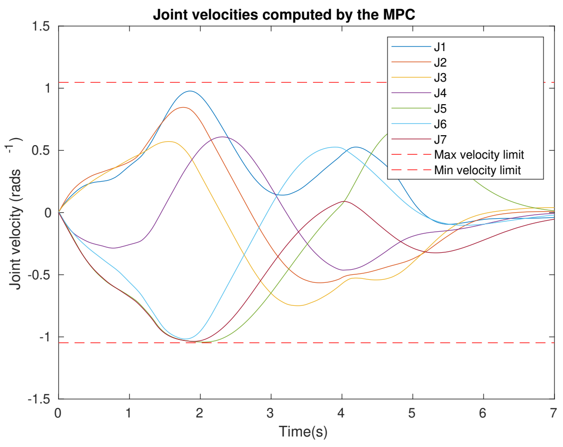
(c) The final pose of the robot reaches the goal position with succesfull obstacle avoidance.

(d) Collision is observed several times when the obstacle avoidance constraints are not implemented inclusing at the final goal pose

Fig. 3: Visualization of obstacle avoidance



(a) The joint position values of the trajectory followed by the robot



(b) The joint velocity values for the trajectory computed by the MPC

Fig. 4: Computed trajectories during obstacle avoidance

being sufficient to satisfy the constraints to a low tolerance all the time. Figure 2c compares the obstacle penetration of the robot motion from the three solvers. All the solvers achieve obstacle avoidance except for PANOC at a particular instance, but the violation is still a small value of 3mm and did not result in collision because the robot geometry is modelled slightly conservatively.

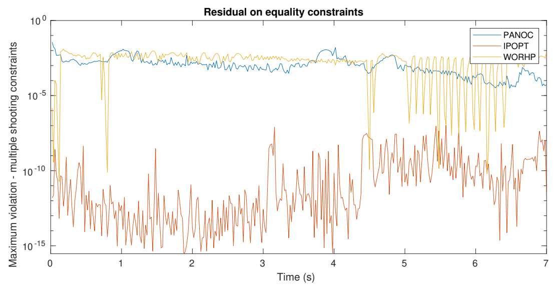
solvers	PANOC	IPOPT	WORHP
Mean time	4.98 ms	226.6 ms	189.7 ms
Standard deviation	1 ms	84.5 ms	64.8 ms
Max time	9.27 ms	800 ms	477 ms

B. Moving Obstacle

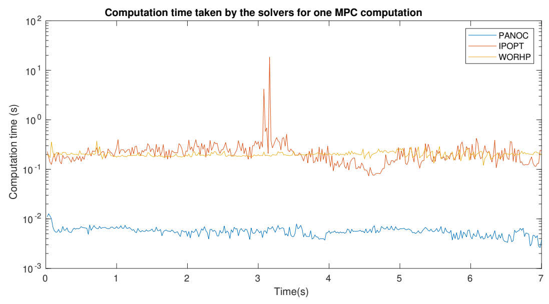
In this section we simulate a scenario where the obstacle is moving and actuate robot with MPC control actions from the different solvers. The ball is given a small velocity such that it moves in the upward direction. We can see the initial position of the robot and the obstacle in the figure 3a. In the figure 3b we see the robot crossing over the moving obstacle to reach the goal position in 3c. If the collision avoidance constraints are not added, the final goal position, that is reached is shown in the figure 3d which we can clearly see is a configuration in collision. In the figures 4a and 4b, are plotted the joint position and joint velocity values computed by MPC using PANOC with ALM. The smoothness of the control action despite the multiple shooting constraints not being satisfied to a very low tolerance is worthy of being noted.

We do not form a motion model of the obstacle. Forming such a motion model in many cases is not feasible, for example when humans and robots share the workspace, it is not always possible to predict the motion of humans. Instead we record and send the instantaneous position of the obstacle to the MPC solver, thus forcing the MPC solver to react to newly received obstacle position each time. We hypothesize that PANOC-ALM would be more reactive to the changing robot position and achieve better constraint satisfaction because it takes more inner iteration steps than an SQP method. This is indeed what is observed in 5a. One can see that PANOC regularly satisfies the equality constraints better than the WORHP solver. In fact for this example, it was found that robot simulated with control actions from WORHP MPC failed to reach the goal state within 7 seconds when the maximum allowed SQP iterations are 6. IPOPT still accurately satisfies the constraints because the number of iterations are not limited for this solver.

The computation times taken by the different solvers remain very similar as can be seen in 5b. This is reasonable because computationally, the problem remains the same. However



(a) Maximum constraint violation of the multiple-shooting constraints in moving obstacle case



(b) Computation time for moving obstacle avoidance case

Fig. 5: Computation time and primal feasibility residuals for the motion planning in the moving obstacle case

computation time of WORHP solver towards the end of the trajectory is worse compared to the stationary obstacle case because WORHP has not reached the goal pose yet which would have prevented WORHP from computing all 6 SQP iterations.

IV. CONCLUSIONS AND FUTURE WORK

PANOC with ALM is demonstrated to be orders of magnitude faster than IPOPT and WORHP. But the constraint satisfaction of the multiple shooting and other initial and terminal constraints is worse when the MPC is solved using PANOC. However, it was found to be still accurate enough to result in smooth trajectories that successfully reach the goal, when the robot is simulated with the joint acceleration inputs from the MPC. The PANOC solver was found to always compute the input action within 10 ms and is thus suitable to be used for an MPC controller that runs at 50 Hz.

However, with multiple shooting formulation, the obstacle avoidance constraints are enforced only at discrete points providing no guarantee of constraint satisfaction between these points. In future work we aim to address this problem by computing the swept volume of the robot to avoid the obstacles and also extend the obstacle shape to other convex primitives following the treatment in [5]. We also aim to extend the solver to deal with L1 norm of the distance error in order to obtain more time-optimal trajectories.

ACKNOWLEDGMENT

The authors would like to thank Ben Hermans for discussion on the implementation of the augmented Lagrangian method.

REFERENCES

[1] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *Proceedings. 1985 IEEE International Conference on Robotics and Automation*, vol. 2. IEEE, 1985, pp. 500–505.

[2] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[3] J. J. Kuffner and S. M. LaValle, "Rrt-connect: An efficient approach to single-query path planning," in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, vol. 2. IEEE, 2000, pp. 995–1001.

[4] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The international journal of robotics research*, vol. 30, no. 7, pp. 846–894, 2011.

[5] J. Schulman, J. Ho, A. X. Lee, I. Awwal, H. Bradlow, and P. Abbeel, "Finding locally optimal, collision-free trajectories with sequential convex optimization." in *Robotics: science and systems*, vol. 9, no. 1. Citeseer, 2013, pp. 1–10.

[6] M. Zucker, N. Ratliff, A. D. Dragan, M. Pivtoraiko, M. Klingensmith, C. M. Dellin, J. A. Bagnell, and S. S. Srinivasa, "Chomp: Covariant hamiltonian optimization for motion planning," *The International Journal of Robotics Research*, vol. 32, no. 9-10, pp. 1164–1193, 2013.

[7] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical programming*, vol. 106, no. 1, pp. 25–57, 2006.

[8] J. Nocedal and S. Wright, *Numerical optimization*. Springer Science & Business Media, 2006.

[9] L. Stella, A. Themelis, P. Sopasakis, and P. Patrinos, "A simple and efficient algorithm for nonlinear model predictive control," in *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*. IEEE, 2017, pp. 1939–1944.

[10] A. Sathya, P. Sopasakis, R. Van Parys, A. Themelis, G. Pipeleers, and P. Patrinos, "Embedded nonlinear model predictive control for obstacle avoidance using panoc," in *2018 European Control Conference (ECC)*. IEEE, 2018, pp. 1523–1528.

[11] E. Small, P. Sopasakis, E. Fresk, P. Patrinos, and G. Nikolakopoulos, "Aerial navigation in obstructed environments with embedded nonlinear model predictive control," in *2019 18th European Control Conference (ECC)*. IEEE, 2019, pp. 3556–3563.

[12] D. C. Liu and J. Nocedal, "On the limited memory bfgs method for large scale optimization," *Mathematical programming*, vol. 45, no. 1-3, pp. 503–528, 1989.

[13] A. Themelis, L. Stella, and P. Patrinos, "Forward-backward envelope for the sum of two nonconvex functions: Further properties and nonmonotone linesearch algorithms," *SIAM Journal on Optimization*, vol. 28, no. 3, pp. 2274–2303, 2018.

[14] L. Stella, A. Themelis, and P. Patrinos, "Forward–backward quasi-newton methods for nonsmooth optimization problems," *Computational Optimization and Applications*, vol. 67, no. 3, pp. 443–487, 2017.

[15] D. P. Bertsekas, *Constrained optimization and Lagrange multiplier methods*. Academic press, 2014.

[16] C. Büskens and D. Wassel, "The esa nlp solver worhp," in *Modeling and optimization in space engineering*. Springer, 2012, pp. 85–110.

[17] 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.