Model Based Predictive Control for a Smart Conveyor Belt System

David Reixach-Pérez
Master Degree in Automatic Control and Robotics
ETSEIB - Universitat Politècnica de Catalunya, Barcelonatech
david.reixach@estudiant.upc.edu

Abstract—**This paper proposes a MPC solution for a logistic system modeled as a 2-DOF manipulator. A formulation and a non-linear model is derived. A linearized approach for this model is stated. A multilayer control architecture is considered. LMPC and NMPC are considered for solving such problem, formulating the corresponding OCPs. Simulation for both propositions is performed and results are presented and discussed. Some conclusions are given regarding the work done.**

Index Terms—**Model Predictive Control; LMPC; NMPC; Multilayer Control Architecture; Kinematic Model; Dynamic Model; Simulation;**

I. Introduction

Recent development in robotics and control science provide new possibilities for handling classical logistic problems. In this work we propose a control solution for a smart conveyor belt for packages. The smart conveyor belt is a discretized finite surface for which linear velocity -magnitude and direction- can be set at each point independently. Understood as a roller conveyor, the velocity is transmitted to the package by means of friction. Each point represent a roller, for which rotating speed and orientation on the surface can be controlled.

We understand the control problem as a multilayer case. First, the package is modeled as a two-degree-of-freedom robot. A model based controller obtains a desired linear and angular velocity for its point within the surface, what with the use of the conveyor inverse kinematics model produces the desired linear velocity for each discretized point, that can be easily translated to rotating speed and angular position. Then a second control layer would ensure the actuators following its reference, for example independent PID controllers for each actuator.

Developing a dynamic model for our manipulator would be a task of a high complexity. We would have to include the dynamic behavior of the actuators and build a model that would depend on the position of the package as forces are only transmitted along its contact surface. However, a simplest yet powerful approach could be to embed actuators effect on the package as a dragging force and torque. This could provide a richer controller (e.g. friction force limitations). In this case study but, we exclusively relay on the kinematic model for the system as our scope is to follow a given trajectory minimizing tracking error and control effort.

Our objective is to implement a Model Predictive Control. In order to achieve fast convergence for the Optimal Control Problem (OCP) we first consider a linearized model and we propose a Linear MPC (LMPC). Also, due to its Non-linear nature we will consider a Non-linear MPC (NMPC) and will proceed with a fair comparison between methods. The document is structured as follows, in Section II the mathematical model is presented, in Section III the design of the control problem is stated, in Section IV experimentation and results are explained and in Section V conclusions are given.

II. Problem Formulation

In Fig. 1 a representation of our problem is shown. The formulation is split between two models, the kinematic model of the box understood as a 2-Degree-of-Freedom robot controlled by its linear and angular velocity; and the inverse kinematics model of the surface, in order to compute the velocity vector field needed to produce the desired control effect to the box.

The first model will be used to generate a MPC controller that ensures tracking a reference (position and orientation) and the second one to generate the desired reference trajectories to the actuators.

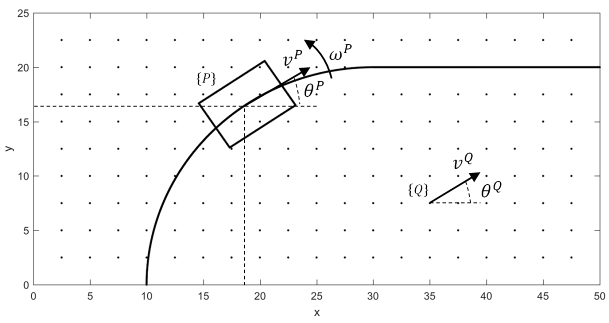


Figure 1: Problem formulation. P refers to box variables (top layer states and control actions) and Q to actuators state.

A. Non-linear model

The kinematic model for the package simplified as a 2-DOF robot, according to [1] is given by:

$$\dot{X} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} v(t)cos(\theta(t)) \\ v(t)sin(\theta(t)) \\ \omega(t) \end{bmatrix} \tag{1}$$

And it can be easily discretized:

$$\begin{bmatrix} x(k+1) \\ y(k+1) \\ \theta(k+1) \end{bmatrix} = \begin{bmatrix} x(k) + v(k)T_s cos(\theta(k)) \\ y(k) + v(k)T_s sin(\theta(k)) \\ \theta(k) + T_s\omega(k) \end{bmatrix} \quad (2)$$

Our Non-linear discrete-time approach assumes model time-invariance at each step, the full state is measured and no presence of disturbances.

B. Linear model

The kinematic model previously obtained can be linearized at instant (k), giving to the next expression:

$$\tilde{x}(k+1) = \mathbf{A}(k)\tilde{x}(k) + \mathbf{B}(k)\tilde{u}(k) \quad (3)$$

with

$$\mathbf{A}(k) \triangleq \begin{bmatrix} 1 & 0 & -v(k)T_s sin(\theta(k)) \\ 0 & 1 & v(k)T_s cos(\theta(k)) \\ 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{B}(k) \triangleq \begin{bmatrix} T_s cos(\theta(k)) & 0 \\ T_s sin(\theta(k)) & 0 \\ 0 & T_s \end{bmatrix}$$

$$\tilde{x} \triangleq x - x_k$$

$$\tilde{u} \triangleq u - u_k$$

Also, over the assumptions taken for the previous model, a linear behavior is assumed, what is a valid approximation close to the linearizing point, but not much further away.

C. Inverse Kinematics

Understanding the kinematics of the 2-DOF robot as the composition of its linear and angular velocity and mapping them through its ICR, the velocity vector for a point Q can be obtained with respect to the velocity of P and their relative position:

$$(v_x^Q, v_y^Q) = (v_x^P, v_y^P) + (P_y - Q_y, -P_x + Q_x) \cdot \dot{\theta}^P \quad (4)$$

III. CONTROLLERS DESIGN

The MPC algorithm idea is to obtain the control input that plugged to a modeled system generates an optimum behavior along a sequence, according to a given law. The MPC computes at each sampling time the optimal control sequence for a prediction horizon (H_p), the first control signal is applied to the plant and the control loop is repeated again, shifting the prediction horizon accordingly. In Fig. 2 an application of the algorithm specific to our case can be seen.

In the next subsections we present the formulation of the OCP, for the LMPC and the NMPC.

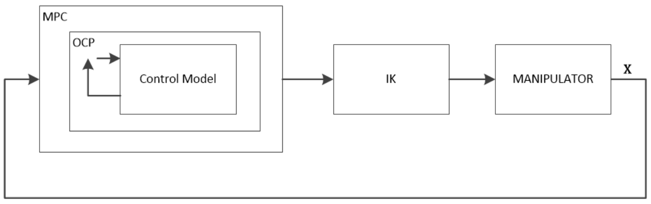


Figure 2: Schema of the top control layer (MPC and Inverse Kinematics module). A bottom layer is considered embedded inside our plant block.

A. LMPC

The OCP for the LMPC simulation comes from the linearized model we obtained in the previous section. The optimization law is a weighted minimization of the 2-norm of both the error of the states w.r.t. the reference and the control action.

Eq. 5 defines the optimization function, the dynamic model together with the control limitations constraint the problem. As it is seen, for each prediction horizon, the dynamic model is linearized at the last obtained point (from the previous iteration), (x_r, u_r), which do not correspond to the initial states for the current optimization.

$$\min_{\tilde{U}(k)} \left(\sum_{k=1}^{H_p} \tilde{e}(k)^T Q \tilde{e}(k) + \sum_{k=0}^{H_p-1} \tilde{u}(k)^T R \tilde{u}(k) \right) \quad (5)$$

s.t.

$$\tilde{x}(k+1) = \mathbf{A}_r\tilde{x}(k) + \mathbf{B}_r\tilde{u}(k) \qquad \forall k \in [0, Hp]$$

$$\mathbf{G}\tilde{u}(k) \le \mathbf{b} \qquad \forall k \in [0, Hp]$$

$$\tilde{x}(k) \in \Re^3 \qquad \forall k \in [0, Hp]$$

$$\tilde{u}(k) \in \Re^2 \qquad \forall k \in [0, Hp]$$

with

$$\mathbf{G} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ -1 & 0 \\ 0 & -1 \end{bmatrix}$$

$$\mathbf{b} = \begin{bmatrix} \overline{v} \\ \overline{\omega} \\ -\overline{v} \\ -\overline{\omega} \end{bmatrix}$$

$$\tilde{U}(k) \triangleq \{\tilde{u}(k), \tilde{u}(k+1), ..., \tilde{u}(k+H_p-1)\}$$

$$\tilde{e}(k) \triangleq \tilde{x}_R(k) - \tilde{x}(k)$$

$$\tilde{x}(k) \triangleq x(k) - x_r$$

$$\tilde{x}_R(k) \triangleq x_R(k) - x_r$$

$$\tilde{u}(k) \triangleq u(k) - u_r$$

After some testing we obtain the matrices we use as weighting factors:

$$Q = \begin{bmatrix} 10^3 & 0 & 0 \\ 0 & 10^3 & 0 \\ 0 & 0 & 10^3 \end{bmatrix}$$

$$R = \begin{bmatrix} 0.1 & 0 \\ 0 & 10^{-3} \end{bmatrix}$$

B. NMPC

The OCP for the NMPC simulation comes directly from Eq. 1 as the problem is formulated in continuous time. The optimization law, in this case formulated in continuous time (integral), is the same as the LMPC OCP. Its complete formulation is proposed below:

$$\min_{u(t), t\in[0,T_p]} \int_0^{T_p} (e(t)^T Q e(t) + u(t)^T R u(t)) dt \qquad (6)$$

s.t.

$$\dot{X} = \begin{bmatrix} v(t)cos(\theta(t)) \\ v(t)sin(\theta(t)) \\ \omega(t) \end{bmatrix}$$

$$\mathbf{G}u(t) \leq \mathbf{b} \qquad \forall t \in [0, Tp]$$

$$X(t) \in \Re^3 \qquad \forall t \in [0, Tp]$$

$$u(t) \in \Re^2 \qquad \forall t \in [0, Tp]$$

with:

$$X(t) \triangleq \begin{bmatrix} x(t) & y(t) & \theta(t) \end{bmatrix}^T$$

$$u(t) \triangleq \begin{bmatrix} v(t) & \omega(t) \end{bmatrix}^T$$

$$e(t) \triangleq X_R(t) - X(t)$$

weighting factors (also after some testing):

$$Q = \begin{bmatrix} 100 & 0 & 0 \\ 0 & 100 & 0 \\ 0 & 0 & 80 \end{bmatrix}$$

$$R = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

IV. Simulation Results

A. Experiment design

We develop a framework in MATLAB® for both experiments (LMPC and NMPC). In the first case, the MPC simulation is build making use of Yalmip Toolbox [2]. In the second case we use ACADO Toolkit [3]. In both experiments we simulate the system with the non-linear model.

The design of the experiment includes a reference trajectory to follow and the definition of the initial states. In Fig. 1 the reference trajectory for the linear position is seen. The angular position is defined as the tangent along the curve (the system has only 2-DOF). The trajectory is discretized in 800 points, what according to the sampling time chosen, T_s, is expected to be covered in 40s. The initial conditions for the states are assigned to the initial reference point, the initial conditions for the control actions are set to 0.

Name	Parameter	Value
Sample time	T_s	0.05 s
Disturbance variance	σ^2_W	0.0011
Prediction horizon	H_p	5
v bound	$\overline{v}$	10 u/s
ω bound	$\overline{\omega}$	4 rad/s

Table I: Simulation and Model Parameters

At the simulation stage, we include a low disturbance modeled as a noise. See the reformulation of the model below and other simulation parameters in Table I.

$$X(k+1) = f_m(X(k), u(k)) + \mathbf{E}w \qquad (7)$$

where f_m corresponds to the simulation model, and with:

$$\mathbf{E} = \begin{bmatrix} 1 \\ 1 \\ 0.1 \end{bmatrix}$$

$$w \sim \mathcal{N}(0, \sigma^2_W)$$

B. Results for the LMPC

Fig. 3. Show the simulation results for the LMPC. Regarding the error for the states, we mainly see an accumulation of the error for the y state. State x follows reliably the reference and state θ results in a chattering behavior, probably is the most affected state by the perturbation, with increasing displacement interval but approximately following its reference trajectory.

Regarding the control actions, in both signals high chattering is observed, v gets often saturated, while ω remains into the bounds.

C. Results for the NMPC [1]

Fig. 4. Show the simulation results for the NMPC. Regarding the error for the states, despite a bit chattering, position states (x, y) show a good tracking of the reference. State θ shows a highly oscillating behavior, misleading.

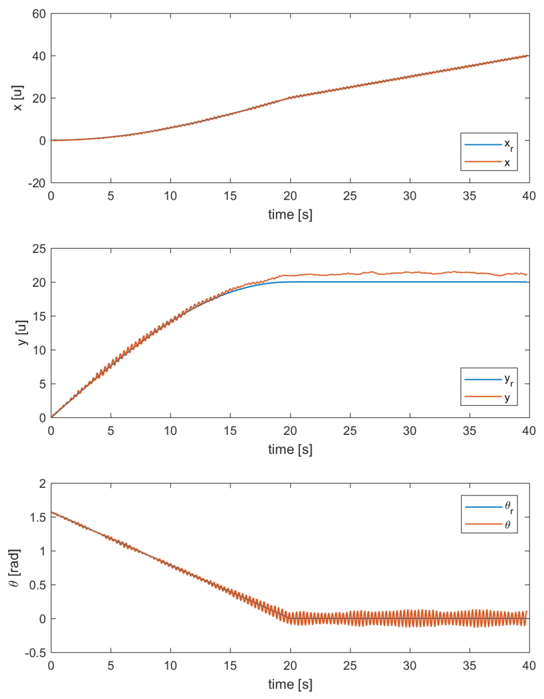
Regarding the control actions, in both signals they show a clear oscillating behavior, despite their frequency is much lower than in the linear case. v remains into the bounds, while ω gets closer to them, in some cases, saturation appears.

D. Results comparison

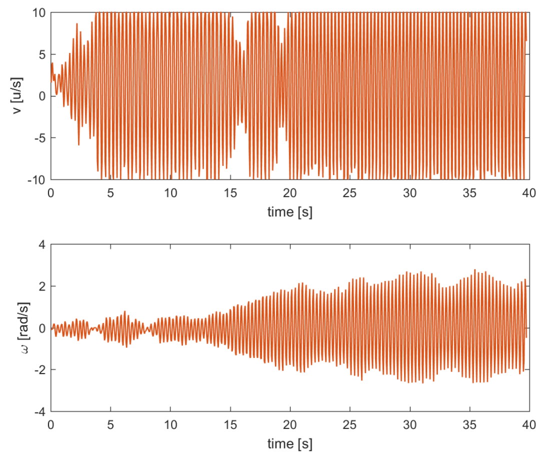
For both methods we evaluate, as performance indexes, the 2-norm for the control actions and the states-reference error component independently. In table II their evaluation is shown. Main differences are the index for the ω control action and the θ state. Also the x states index shows great difference between the controllers.

The difference in the control action mainly happens because, while being both oscillating signals, the second

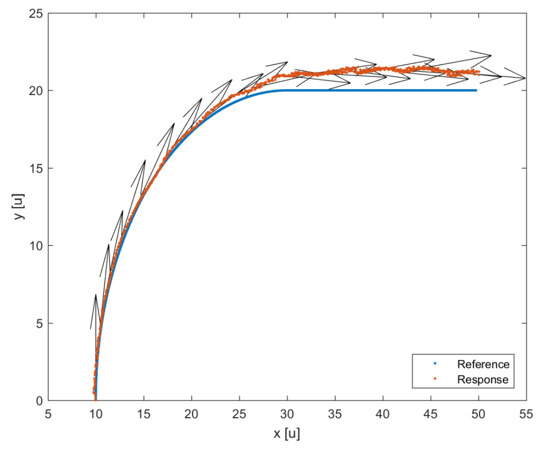
[1] Due to a programming error, the presented results for the NPMC Simulation do not include the defined perturbation.



(a) System's state behavior compared to its reference.

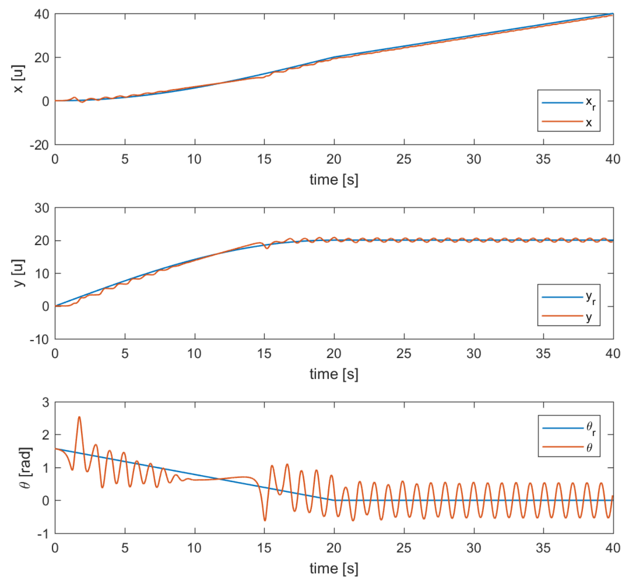


(b) Control action evolution.

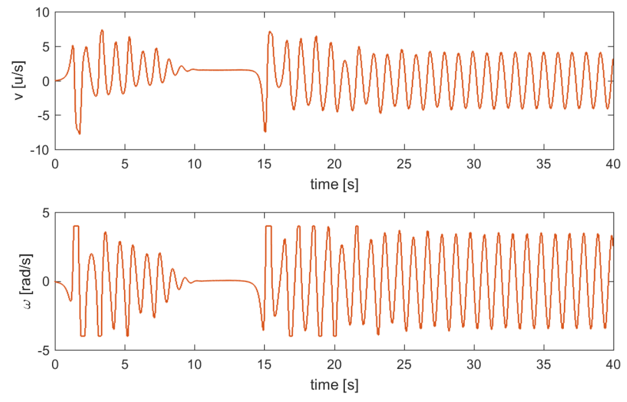


(c) Overall response of the system over the surface. Arrows at the given points correspond to orientation.

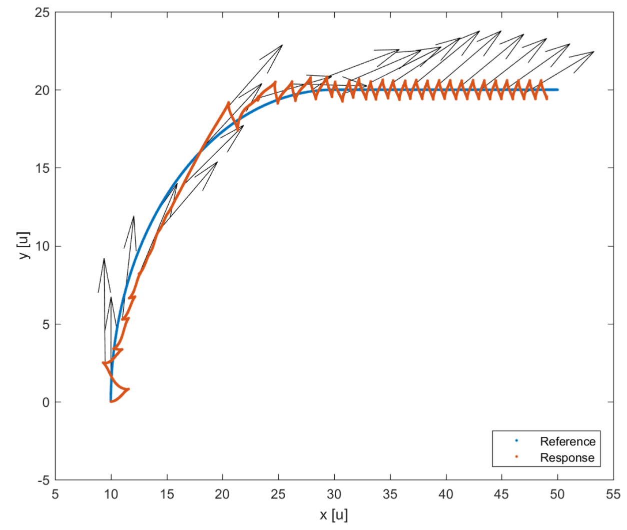
Figure 3: LMPC Simulation: Compound of the obtained responses.



(a) System's state behavior compared to its reference.



(b) Control action evolution.



(c) Overall response of the system over the surface. Arrows at the given points correspond to orientation.

Figure 4: NMPC Simulation: Compound of the obtained responses.

Perf. Index	LMPC	NMPC
$\|\|v\|\|_2$	201.11	261.41
$\|\|\omega\|\|_2$	24.87	196.10
$\|\|e_x\|\|_2$	9.75	77.61
$\|\|e_y\|\|_2$	41.43	56.18
$\|\|e_\theta\|\|_2$	1.24	37.03

Table II: Performance indexes evaluation for both simulations.

controller produces a more smooth, hence slow, sequence which results in a greater energy consumption. The control action computed by the LMPC, while being more rough and chattering, is quicker, also producing more frequent but smaller errors. What is seen in the difference for the θ state index.

For the same reason, the difference between the indexes for x state is produced. The LMPC is able to follow reliably the reference trajectory (with frequent errors but small), while the NMPC produce higher tracking error. We do not observe important differences between the indexes for y state, because in this case the LMPC accumulates a high error.

E. Inverse Kinematics Module

According to our control architecture (Fig. 2), once the MPC obtains a control action, the inverse kinematics module, defined by Eq. 4 should obtain actuators position for the whole discretized surface. In order to check our kinematic definition, in Fig. 5 a computed result for a given instant control action is represented.

Notice that in order to only actuate over the needed actuators for a given reference, thus setting free the rest of the surface for other packages, and accounting on a delay for the actuators a state prediction for a future horizon would have to be implemented. If using MPC, its computed trajectory for the prediction horizon at each state could be used as an estimator.

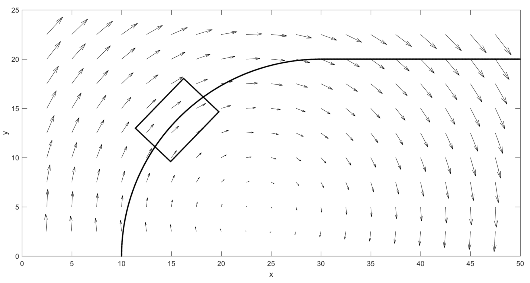


Figure 5: Example of the Inverse Kinematic module for a given instant. It computes the velocity vector that ensures the box, as a rigid body, draws the desired trajectory.

V. Conclusions

In this work a Model Based Predictive Control solution has been proposed and simulated for a Smart conveyor belt logistic system. The system modeled as a 2-DOF robot has been implemented to follow a plane trajectory.

The OCPs of the controllers considered, have been formulated optimizing reference error and control action 2-norms. This approach shows that does not prevent the control action from oscillating. A well-known improvement could be done, not weighting the control action 2-norm, but the variation of this. This has been proved that reduces control action oscillation, what at least is a result to be improved before a real implementation is considered.

Also, considering the dynamic model for the robot, as proposed in section I could produce better results. Despite these considerations, this scenario may not be the best case for a MPC. Other control architectures, also based in a dynamic model, such a classical manipulator controller with pre-computed torque feed-forward compensation could be studied.

References

[1] G. Campion, G. Bastin, and B. Dandrea-Novel, "Structural properties and classification of kinematic and dynamic models of wheeled mobile robots," *IEEE transactions on robotics and automation*, vol. 12, no. 1, pp. 47–62, 1996.

[2] J. Lofberg, "Yalmip : a toolbox for modeling and optimization in matlab," in *2004 IEEE International Conference on Robotics and Automation (IEEE Cat. No.04CH37508)*, Sept 2004, pp. 284–289.

[3] B. Houska, H. 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.