

Non-linear Reference Tracking via Model Predictive Control and Extended Kalman Filter

Modelling and Control of Cyber-Physical Systems II
University of Trieste (UniTS)

Marco Tallone

January 2025

Abstract

This report presents the implementation of a Model Predictive Control (MPC) algorithm and an Extended Kalman Filter (EKF) to perform reference tracking of non-linear dynamical systems. The implemented algorithms have been tested and assessed on different non-linear systems and multiple trajectories. Results confirm the effectiveness of the proposed MPC approach in tracking the desired trajectories, even in the presence of noise and disturbances through the use of the EKF.

1 Introduction

Model Predictive Control (MPC) is a control strategy that has been widely adopted for the control of dynamical systems. The MPC approach is based on the solution of an optimization problem over a finite horizon, which allows for the consideration of constraints on the system states and inputs. When used for reference tracking, the MPC algorithm computes the optimal control input by minimizing a cost function that penalizes the deviation of the system states from the desired trajectory.

However, real-world systems are often affected by noise and disturbances, which can lead to undesired results with the adoption of an MPC controller. Moreover, whenever the complete state of a system cannot be fully measured, the use of state estimators is required to infer the unmeasured components and apply the MPC algorithm. To address this issues, the Extended Kalman Filter (EKF) can be used to estimate the states of a non-linear system by fusing the available measurements with the system dynamics and incorporating information about the process and measurement noise.

This report presents the implementation of a Model Predictive Control algorithm for non-linear dynamical systems based on successive linearization and discretization of the dynamics around operating points, as well as the development of an Extended Kalman Filter to estimate the states of the system in the presence of noise.

The objectives of this project are both to study the effectiveness of the non-linear MPC algorithm applied for tracking reference trajectories under different circumstances and also to reproduce and compare the obtained results with the ones presented by *Kunz, Huck and Summers* [2] in their work on the tracking of a helicopter model.

The following report is structured as follows. Section 2 introduces the non-linear models used for the simulations with their dynamics, while Section 3 explains how feasible trajectories can be generated for such models. Section 4 then describes the linear time varying (LTV) approximation of the models and how such approximation is used in the MPC algorithm. In Section ?? the addition of the Extended Kalman Filter is instead presented. Finally, Section ?? presents the results of the simulations followed by some final consideration in the last section.

From a technical point of view all the algorithms and the models for the dynamical systems have been developed in **MATLAB**, version **R2024b** [1]. The code has been structured in a modular way to allow for easy testing and comparison of different algorithms and models. However, with the aim of maintaining a clear and concise report of the work, the following sections will mostly present the results of this study and further concepts from a theoretical point of view, while implementation details and practical usage instructions will be available on the author's GitHub repository [4].

2 Non-linear Models

In order to test and asses the performance of the implemented MPC and EKF algorithms, two systems of different complexity have been considered. In particular, these are:

- A **Unicycle model**
- A **Helicopter model**

Both systems are mathematically described by non-linear dynamics.

2.1 Unicycle Model

The first system is a relatively simple unicycle model described by the non-linear dynamics in equations (2.1). This consists of a rigid body moving on a plane with two parallel wheels of radius r separated by a distance L .

$$\begin{cases} \dot{x} = \frac{r}{2}(\omega_1 + \omega_2) \cos(\theta) \\ \dot{y} = \frac{r}{2}(\omega_1 + \omega_2) \sin(\theta) \\ \dot{\theta} = \frac{r}{L}(\omega_1 - \omega_2) \end{cases} \quad (2.1)$$

The state of the system is hence defined by the Cartesian coordinates x and y of the center of mass of the unicycle and the heading angle θ . The inputs are instead the angular velocities of the two wheels ω_1 and ω_2 .

$$\mathbf{x}(t) = [x(t) \quad y(t) \quad \theta(t)]^T \quad \text{and} \quad \mathbf{u}(t) = [\omega_1(t) \quad \omega_2(t)]^T$$

2.2 Helicopter Model

The second system represents instead a miniature helicopter model taken from the paper by *Kunz, Huck and Summers* [2]. As explained by the authors, the dynamics can be described by the Newton-Euler laws of mechanics for single rigid bodies. However, a simplification has been made in favor of an easier implementation of the MPC algorithm for this model. In particular, introducing the assumption that the pitch and roll inputs directly act on the translational accelerations, it's possible to omit the pitch and roll angular states of the helicopter body. The result is the non-linear dynamics in equations (2.2).

$$\begin{cases} \dot{x}_I = \cos(\psi)\dot{x}_B - \sin(\psi)\dot{y}_B \\ \dot{y}_I = \sin(\psi)\dot{x}_B + \cos(\psi)\dot{y}_B \\ \dot{z}_I = \dot{z}_B \\ \ddot{x}_B = b_x u_x + k_x \dot{x}_B + \dot{\psi} \dot{y}_B \\ \ddot{y}_B = b_y u_y + k_y \dot{y}_B - \dot{\psi} \dot{x}_B \\ \dot{z}_B = b_z u_z - g \\ \dot{\psi} = \dot{\psi} \\ \ddot{\psi} = b_\psi u_\psi + k_\psi \dot{\psi} \end{cases} \quad (2.2)$$

where x_I, y_I, z_I denote the position of the helicopter in the inertial frame, $\dot{x}_B, \dot{y}_B, \dot{z}_B$ the velocities of the helicopter in the body frame, ψ the yaw heading angle and $\dot{\psi}$ its rotational velocity in yaw. The control inputs for pitch, roll, thrust and yaw are instead denoted by u_x, u_y, u_z, u_ψ .

Due to implementation necessities, the only difference with respect to the original paper is the absence of the augmented integral states which were used to reduce steady-state error. The rest of the setup is consistent with the original work.

In this case we hence obtained a model having 8 state components and 4 control inputs formalized as follows.

$$\begin{aligned} \mathbf{x}(t) &= [x_I(t) \quad y_I(t) \quad z_I(t) \quad \dot{x}_B(t) \quad \dot{y}_B(t) \quad \dot{z}_B(t) \quad \psi(t) \quad \dot{\psi}(t)]^T \\ &\text{and} \\ \mathbf{u}(t) &= [u_x(t) \quad u_y(t) \quad u_z(t) \quad u_\psi(t)]^T \end{aligned}$$

3 Trajectory Generation

Prior to the implementation of the MPC algorithm for reference tracking, it's compulsory to properly define the reference trajectories that the system should follow. In particular, any desired trajectory should be feasible for the given system. Formally, given a continuous-time reference trajectory $[\mathbf{x}_{\text{ref}}(t) \quad \mathbf{u}_{\text{ref}}(t)]^T$, in order for it to be feasible, it must satisfy the differential equations of the system dynamics within the given constraints:

$$\begin{aligned} \dot{\mathbf{x}}_{\text{ref}}(t) &= \mathbf{f}(\mathbf{x}_{\text{ref}}(t), \mathbf{u}_{\text{ref}}(t)) \\ \mathbf{x}_{\text{ref}} &\in \mathcal{X}, \quad \mathbf{u}_{\text{ref}} \in \mathcal{U} \end{aligned} \quad (3.1)$$

where \mathcal{X} and \mathcal{U} are the sets of feasible states and inputs for the system while the function $\mathbf{f}(\cdot, \cdot) : \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}^n$ describes the non-linear state dynamics.

The approaches considered in this study for the feasible trajectory generation task rely on the definition of *differential flatness* introduced by Murray [3]. Formally, a non-linear system is *differentially flat* if there exists a function $\Gamma(\cdot)$ such that

$$\mathbf{z} = \Gamma(\mathbf{x}, \mathbf{u}, \dot{\mathbf{u}}, \dots, \mathbf{u}^{(p)})$$

where \mathbf{z} are the so-called *flat outputs*. For a differentially flat system, feasible trajectories can be written as functions of the flat outputs \mathbf{z} and their derivatives. Hence, the general idea is to find a mapping

$$\begin{aligned} \mathbf{x} &= \mathbf{x}(\mathbf{z}, \dot{\mathbf{z}}, \dots, \mathbf{z}^{(q)}) \\ \mathbf{u} &= \mathbf{u}(\mathbf{z}, \dot{\mathbf{z}}, \dots, \mathbf{z}^{(q)}) \end{aligned}$$

between the original states and the flat outputs such that all states and inputs can be determined from these outputs without integration. Then, a feasible trajectory can simply be obtained by only providing the reference values for the flat outputs (and possibly their derivatives), since the remaining reference states and associated inputs can be recovered from the mapping.

It's easy to show that, for the unicycle model described by equation (2.1), the Cartesian coordinates $z_1 = x$ and $z_2 = y$ are flat outputs. The remaining heading angle state¹ and the control inputs can in fact be obtained from these two outputs and their first order derivatives as shown in equations (3.2).

$$\theta = \arctan\left(\frac{\dot{z}_2}{\dot{z}_1}\right), \quad \omega_1 = \frac{2\sqrt{\dot{z}_1^2 + \dot{z}_2^2} + L\dot{\theta}}{2r}, \quad \omega_2 = \frac{2\sqrt{\dot{z}_1^2 + \dot{z}_2^2} - L\dot{\theta}}{2r} \quad (3.2)$$

For what concerns the helicopter model instead, a set of flat outputs is given by the state components

$$z_1 = x_I, \quad z_2 = y_I, \quad z_3 = z_I, \quad z_4 = \psi$$

which allow to obtain the remaining states and inputs according to equations (3.3).

$$\begin{aligned} \dot{x}_B &= \cos(z_4)\dot{z}_1 + \sin(z_4)\dot{z}_2 \\ \dot{y}_B &= -\sin(z_4)\dot{z}_1 + \cos(z_4)\dot{z}_2 \\ \dot{z}_B &= \dot{z}_3 \\ \dot{\psi} &= \dot{z}_4 \\ u_x &= \frac{1}{b_x} (\cos(z_4) [\ddot{z}_1 - k_x \dot{z}_1] + \sin(z_4) [\ddot{z}_2 - k_x \dot{z}_2]) \\ u_y &= \frac{1}{b_y} (\cos(z_4) [\ddot{z}_2 - k_y \dot{z}_2] + \sin(z_4) [-\ddot{z}_1 + k_y \dot{z}_1]) \\ u_z &= \frac{1}{b_z} (\ddot{z}_3 + g) \\ u_\psi &= \frac{1}{b_\psi} (\ddot{z}_4 - k_\psi \dot{z}_4) \end{aligned} \quad (3.3)$$

After having identified the flat outputs and the required mappings to reconstruct state and inputs components, the generation of the feasible trajectory can then be conducted in two ways.

¹In the implemented models, the $\text{atan2}(\cdot, \cdot)$ function has been used instead of the normal arctangent to obtain the correct heading angle in the range $[-\pi, \pi]$.

The first approach, which is the one used in this study, consists in providing an analytical expression $\mathbf{z}(t)$ for the evolution of the flat outputs over time and then sampling the resulting trajectory at discrete time steps $\{t_k\}_{k=1}^{N_{guide}}$ to obtain a finite set of N_{guide} reference points $\{(\mathbf{x}_{ref,k}, \mathbf{u}_{ref,k})\}_{k=1}^{N_{guide}}$. In particular, two trajectories have been generated with this technique:

- a **circular** trajectory of radius $r_0 = 0.5$ whose flat outputs are defined respectively by the parametric equations:

$$\begin{cases} z_1(t) = r_0 \cos(t) \\ z_2(t) = r_0 \sin(t) \end{cases} \quad \begin{cases} z_1(t) = r_0 \cos(t) \\ z_2(t) = r_0 \sin(t) \\ z_3(t) = 0 \\ z_4(t) = \text{atan2}(\dot{z}_2(t), \dot{z}_1(t)) \end{cases}$$

for the unicycle model

for the helicopter model

- a **lemniscate** (or “figure-eight”) trajectory with parameter $a = 1$ described by:

$$\begin{cases} z_1(t) = \frac{a\sqrt{2}\cos(t)}{\sin(t)^2+1} \\ z_2(t) = \frac{a\sqrt{2}\cos(t)\sin(t)}{\sin(t)^2+1} \end{cases} \quad \begin{cases} z_1(t) = \frac{a\sqrt{2}\cos(t)}{\sin(t)^2+1} \\ z_2(t) = \frac{a\sqrt{2}\cos(t)\sin(t)}{\sin(t)^2+1} \\ z_3(t) = 0 \\ z_4(t) = \text{atan2}(\dot{z}_2(t), \dot{z}_1(t)) \end{cases}$$

for the unicycle model

for the helicopter model

In the conducted experiments, the models performed a full revolution² around the respective trajectories in a determined amount of time. Specifically, due to the discretization with sampling time T_s later introduced in Section 4, the total time to complete the trajectory has been set to $T_{end} = N_{guide} \cdot T_s$ for simplicity. In this way, at each iteration, the discretized system has one time-step to reach the next reference point. Moreover, notice that the discretization assumes a constant input applied to the system at each time-step. Hence, a possibility would be to just sample the reference inputs at the beginning of each time interval $\mathbf{u}_{ref,k} = \mathbf{u}_{ref}(t_k)$. However, due to the continuous-time nature of the inputs obtained with this approach, a more realistic approximation of the system’s behavior can be obtained by averaging the values of the inputs over each time interval:

$$\mathbf{u}_{ref,k} = \frac{1}{N_{samples}} \sum_{i=1}^{N_{samples}} \mathbf{u}_{ref} \left(t_k + \frac{i \cdot T_s}{N_{samples}} \right)$$

where $N_{samples}$ is the number of sub-samples taken in each time interval to compute the average, which depends on the specific trajectory and model considered.

When an analytical expression for the flat outputs is not available, the approach proposed by Murray [3] can be used to generate feasible trajectories. In this case the values of the flat outputs (and possibly their derivatives up to a determined order) for a minimal³ set of representative points of the desired trajectory are needed. Then, between any two given points, the flat outputs can be parametrized as a linear combination of a finite set of basis functions. The coefficients of this linear combination are determined by simply solving a linear system of equations involving the known values of the flat outputs and their derivatives at the endpoints of the interval. This results in a parametrized analytical expression for the flat outputs between the given points, which can then be resampled to obtain a higher number of reference points.

Therefore, this approach presents the huge advantage of being able to generate feasible and smooth point-to-point trajectories for any arbitrary set of initial reference points.

In practice, however, to generate realistic inputs for most systems, high order derivatives of the basis functions involved in the parametrization are often required, especially when the desired trajectories present a high number of sharp turns or sudden changes in direction. This turns out to be unpractical since the known values for the flat outputs derivatives of the same order must be provided at least for the initial set of points. Hence, although implemented, this method has not been used in this study, but further information can be found in Appendix ?? or in the original work by Murray [3].

²However, the implementation easily allows to perform more than one lap around the given shapes.

³At least 2 points.

4 MPC Problem Formulation

The Model Predictive Control (MPC) algorithm is based on the solution of an optimization problem over a finite horizon of length N . In general, the formulation of such optimization problem requires a discrete-time representation of the model dynamics. Moreover, in order to cast the MPC problem as a Quadratic Program (QP) one and hence solve it efficiently, the model must also be linear. The models introduced in Section 2 are not only defined in continuous-time, but also highly non-linear. To overcome this issues, a Linear Time Varying (LTV) approximation of the dynamics is here presented. Then, two formulations of the MPC algorithm as a QP problem are introduced.

4.1 Linear Time Varying (LTV) Model

The LTV model is obtained by successive linearization and discretization of the non-linear dynamics $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u})$ around a set of nominal state and input trajectories sampled at a fixed regular time intervals T_s :

$$\begin{aligned}\bar{\mathbf{x}}_k &= \bar{\mathbf{x}}(kT_s), \quad k = k_0, \dots, k_0 + N + 1 \\ \bar{\mathbf{u}}_k &= \bar{\mathbf{u}}(kT_s), \quad k = k_0, \dots, k_0 + N\end{aligned}$$

Hence, the system is first linearized around the nominal states and inputs as follows:

$$\begin{aligned}\delta \mathbf{x}(t) &= \mathbf{A}_k \delta \mathbf{x}(t) + \mathbf{B}_k \delta \mathbf{u}(t) \\ \mathbf{A}_k &= \left. \frac{\partial \mathbf{f}}{\partial \mathbf{x}} \right|_{\bar{\mathbf{x}}_k, \bar{\mathbf{u}}_k}, \quad \mathbf{B}_k = \left. \frac{\partial \mathbf{f}}{\partial \mathbf{u}} \right|_{\bar{\mathbf{x}}_k, \bar{\mathbf{u}}_k} \\ \delta \mathbf{x}(t) &= \mathbf{x}(t) - \bar{\mathbf{x}}_k, \quad \delta \mathbf{u}(t) = \mathbf{u}(t) - \bar{\mathbf{u}}_k\end{aligned} \tag{4.1}$$

with $kT_s \leq t < (k+1)T_s$ for $k = k_0, \dots, k_0 + N$. Then, the obtained linear system is discretized using first order forward Euler method:

$$\begin{aligned}\delta \mathbf{x}_{k+1} &= \mathbf{A}_{d,k} \delta \mathbf{x}_k + \mathbf{B}_{d,k} \delta \mathbf{u}_k \\ \mathbf{A}_{d,k} &= \mathbb{I} + T_s \mathbf{A}_k, \quad \mathbf{B}_{d,k} = T_s \mathbf{B}_k \\ \delta \mathbf{x}_k &= \mathbf{x}_k - \bar{\mathbf{x}}_k, \quad \delta \mathbf{u}_k = \mathbf{u}_k - \bar{\mathbf{u}}_k\end{aligned} \tag{4.2}$$

This can eventually be rewritten as:

$$\begin{aligned}\mathbf{x}_{k+1} &= \mathbf{A}_{d,k} \mathbf{x}_k + \mathbf{B}_{d,k} \mathbf{u}_k + \mathbf{d}_k \\ \mathbf{d}_k &= \bar{\mathbf{x}}_{k+1} - \mathbf{A}_{d,k} \bar{\mathbf{x}}_k - \mathbf{B}_{d,k} \bar{\mathbf{u}}_k\end{aligned} \tag{4.3}$$

Using this LTV model, the MPC controller is able to solve an optimization problem for each k^{th} time step in the prediction horizon N . In particular, the result of the optimization will be the optimal input sequence that minimizes the cost function in the prediction horizon, from which only the first input will actually be applied. Hence, in successive applications of the algorithm, a nominal input sequence for future linearizations can be obtained as the remaining part of the optimal input sequence⁴ while the nominal states can be obtained by applying the nominal inputs to the non-linear dynamics. For the first iteration of the MPC algorithm, the reference input sequence can be used even if it will result in an inaccurate linearization.

4.2 MPC Formulations

The Model Predictive Control problem with the LTV model approximation of the non-linear dynamics is formulated as the optimization problem 4.4, which is solved at every time instance t .

$$\begin{aligned}\min_{\mathbf{u}_{t \rightarrow t+N|t}} \quad & \sum_{k=t}^{t+N} (\Delta \mathbf{x}_k^T \mathbf{Q} \Delta \mathbf{x}_k + \Delta \mathbf{u}_k^T \mathbf{R} \Delta \mathbf{u}_k) \\ \text{s.t.} \quad & \mathbf{x}_{k+1|t} = \mathbf{A}_{d,k|t} \mathbf{x}_{k|t} + \mathbf{B}_{d,k|t} \mathbf{u}_{k|t} + \mathbf{d}_{k|t} \\ & \mathbf{d}_{k|t} = \bar{\mathbf{x}}_{k+1|t} - \mathbf{A}_{d,k|t} \bar{\mathbf{x}}_{k|t} - \mathbf{B}_{d,k|t} \bar{\mathbf{u}}_{k|t} \\ & \mathbf{x}_{k|t} \in \mathcal{X}, \quad k = t, \dots, t+N \\ & \mathbf{u}_{k|t} \in \mathcal{U}, \quad k = t, \dots, t+N+1\end{aligned} \tag{4.4}$$

⁴With the last input duplicated as N nominal inputs are needed.

Where $\mathbf{U}_{t \rightarrow t+N|t} = [\mathbf{u}_{t|t} \ \dots \ \mathbf{u}_{t+N|t}]^T$ is the input sequence to be optimized, while $\Delta \mathbf{x}_k = \mathbf{x}_{k|t} - \bar{\mathbf{x}}_k$ and $\Delta \mathbf{u}_k = \mathbf{u}_{k|t} - \bar{\mathbf{u}}_k$ are the differences between the predicted states and the reference trajectory. The matrices \mathbf{Q} and \mathbf{R} are the weighting matrices for the state and input errors, respectively. This general expression of the MPC problem can be further casted into a QP problems in two different formulations: the *dense* formulation and the *sparse* formulation.

References

- [1] The MathWorks Inc. MATLAB version: 24.2.0 (r2024b), 2024. URL: <https://www.mathworks.com>.
- [2] Konstantin Kunz, Stephan M. Huck, and Tyler H. Summers. Fast model predictive control of miniature helicopters. In *2013 European Control Conference (ECC)*, pages 1377–1382, 2013. doi:10.23919/ECC.2013.6669699.
- [3] R.M. Murray. *Optimization Based Control*. California Institute of Technology, 2023. URL: http://www.cds.caltech.edu/~murray/books/AM08/pdf/obc-complete_12Mar2023.pdf.
- [4] Marco Tallone. Non-Linear Tracking via MPC and EKF. URL: <https://github.com/marcotallone/mpc-tracking>.