Optimal Control : A Review

Hanady Gebran, Zehua Zhang, Xingyu Song and Yaw Obeng Okofo Dartey
Supervisor: Riddhiman Laha, M.Sc.
Technical University Munich

Abstract

Optimal Control is a widely used controlling method. It is a process of determining control and states for a system over a period of time to minimise a predefined performance. In this report, four commonly used optimal control methods will be introduced. The literature review and applications of these four methods are also covered.

1. Introduction

In classical control system design, multiple methods of analysis, like PID controller, aim at trying and employing repeatedly different parameters to fit the system. As the criterion for whether the design is acceptable, some indications in the time and frequency domain such as rising time, settling time, overshoot, and bandwidth are calculated and considered. However, in the control theory problems, the complex MIMO system (multiple-input, multiple-output system) requires that different performance criteria must be satisfied. For example, during the control design of UAV (unmanned aerial vehicle), fast take-off speed, accurate trajectory tracking, and less fuel consumption are all desired results. This is a difficult problem for traditional control methods. Therefore, optimal control methods are proposed to solve the optimization problem of such complex systems [1].

The goal of optimal control theory is to determine the control signals that will cause a process to satisfy the physical constraints and at the same time minimize (or maximize) some performance criteria [1]. The cost function is used to measure performance. In other words, a suitable criterion needs to be chosen and then minimizes the value of the corresponding cost function [2]. The specific formula of performance J is:

$$J = \int_{t_0}^{t} F[x(t), u(t), t]dt \tag{1}$$

where $x(t)$ is the state of the system, $u(t)$ is the input, t_0 is the initial time and t is the end time. Take the UAV, which is mentioned above, as an example, if the design goal is to reach the target height as quickly as possible. The system state is the position, velocity, and angular velocity of the UAV, the control variable is the motor speed, and the optimal performance is to make the time shortest. In this situation, the cost function can be expressed as:

$$J = \int_{t_0}^{t} 1dt \tag{2}$$

Another common problem is that the energy consumption is desired to be the least when the target height can be reached within a certain period. The states and inputs remain unchanged, and the optimized cost function becomes:

$$J = \frac{1}{2}\int_{t_0}^{t} u(t)^2 dt \tag{3}$$

The design of optimal control depends upon the choice of J. After selecting a suitable performance index, the controlling part in the system can be determined by many methods. In the next four chapters, four major methods (of optimal control will be introduced, including the basic principles, related literature, and future outlook.

2. Methods and Literature Review

2.1. Linear Quadratic Regulator

2.1.1 Definition

If the system dynamics can be represented by a set of linear differential equations, and the cost function is quadratic, such problems are called linear quadratic (LQ) problems, and the controller is called linear quadratic controller (LQR). Back to the UAV problem mentioned in chapter 1, when the UAV is expected to fly along the preset path accurately and consume little energy, the positional state error should be small and the input amount should also be small at the same time. Since the value may be positive and negative, the squared value $x^T x$ is chosen to evaluate the error and $u^T u$ is for the input energy [3]. Now the problem is

changed to minimize these two squares, which can be expressed like below:

$$J = \int_{t_0}^{t} (x^T x + u^T u) dt \tag{4}$$

A major feature of the LQR is that this method can deal with multiple inputs and multiple outputs problems. For example, when considering the problem of reaching the target height, the error of the position state is expected to be minimized, and the requirement for the accuracy of the position in the Z-axis direction is higher than that of the X-axis and the Y-axis. In other words, some states and actuators are more cared for than others. This can be achieved by adding two matrices to the equation: Q and R. Both matrices are positive definite. If a certain value in the Q matrix is set to be very large, a slight change in the corresponding state will lead to a significant change in the J value, which is equivalent to "penalizing" the state. Therefore, the key to designing the LQR controller is to select the appropriate Q matrix and R matrix to constrain the system to obtain the desired control result [3]. The equation of J with Q and R matrices is:

$$J = \int_{t_0}^{t} (x^T Q x + u^T R u) dt \tag{5}$$

After selecting the Q and R matrices, the feedback can be calculated by:

$$K = R^{-1} B^T P \tag{6}$$

Among them, the value of P is obtained by the ricatti formula:

$$PA + A^T P - PBR^{-1} B^T P = -Q \tag{7}$$

It should be noted that P and K are time-variant. Only when t in equation (5) approaches infinity, P can be a constant value calculated by equation (7) [3]. The mathematical calculations involved will not be described in detail.
Figure 1 shows the workflow of the LQR designing. The

Figure 1. The flow of designing the LQR controller

choice of Q matrix and R matrix depends on the experience of engineers. In practical applications, the weights of Q and R are usually initialized as identity matrices I. Then change the corresponding element values in Q and R according to the acceptable error of different variables. If the units of the variables are different, it is reasonable to compare the numerical relationship by setting a reference unit. For example, in a system, there are two states: the position and the angle. The error of the position state is expected to be no more than 1 m, and the angle error is no more than 1/60 rad. If meter and rad are selected as the reference units, it is obvious that the requirements for angular accuracy are stricter, and the "penalty" for this error should be larger. In other words, the corresponding element value in Q should be larger. The ratio of the elements for the position and the angle in the Q matrix can be set to 1:3600 (the square is because the square part $x^T Q x$ in J) [4]. After selecting the Q and R matrices, output the results, and then adjust Q and R according to the results. These tasks are all done manually. In recent years, more and more methods for automatically obtaining Q and R matrices have been proposed, which will be introduced at the end of the next section.

2.1.2 Development of LQR method

As the name suggests, the LQR controller is mainly used to deal with linear problems. When the system equations can be transformed in linear form, LQR can be used to design the control model and calculate the feedback through the Riccati equation. An example of robot control by using LQR will be given in this section. Secondly, how to use LQR control in nonlinear problems will also be introduced. Finally, with the development of advanced algorithms such as metaheuristic algorithm, there are more and more researches on combining advanced algorithms and LQR methods to automatically obtain Q and R matrices, which will be introduced at the end of this section.

Linear Problem LQR optimization control can play an important role in controlling the robots by providing all states of the system for feedback, including velocity and position. Abdolshah et al. [5] design an LQR controller for a planar parallel robot to reach a good path tracing with a torque smaller than 3.5 N. The structure of the robot is shown as Figure 2 (a). The robot has three DOF (degrees of freedom). There are four cables at the lower end of the plane which is respectively connected with a circular actuator. The other end of the cable is directly connected to the motor through a pulley. The force analysis of the robot is shown in Figure 2 (b) [6]. Since this chapter focuses on the design of the LQR controller, the detailed force analysis calculation and derivation of the state formula will be ignored. For the robot system, a state equation as follows will

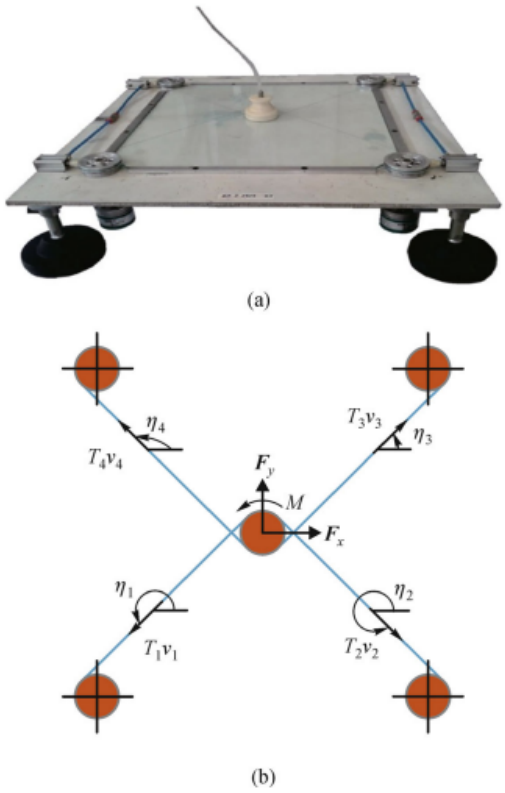

Figure 2. The Feriba-3 cable-driven parallel robot from [5]. (a)Overview; (b)statcs diagram

be obtained:

$$\begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \\ \dot{q}_4 \\ \dot{q}_5 \\ \dot{q}_6 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \\ q_5 \\ q_6 \end{bmatrix}$$

$$+ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{m} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{m} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{I} \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ F_X \\ F_Y \\ M \end{bmatrix} \quad (8)$$

where six states from q_1 to q_6 are the positions, velocities, and angular velocities of the robot in the X and Y directions. F_X, F_Y, and M are the forces applied on the robot, which are considered as the input variable [5].

Because the desired action is to minimize the tracing error (output) and limit the torque and the force (input), the performance index is designed as below:

$$J = \int_0 \infty [y^T(t) \cdot Q \cdot y(t) + W^T(t) \cdot L \cdot W(t)] dt \quad (9)$$

where Q is the weight matrix for the output and L is for the input. After trying different values for the Q and the L matrix, the elements are selected as follows:

$$Q = \begin{bmatrix} 2.5 \times 10^6 & 0 & 0 & 1 & 0 & 0 \\ 0 & 2.5 \times 10^6 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1.8 \times 10^3 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1.6 \times 10^3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1.6 \times 10^3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 3 \end{bmatrix} \quad (10)$$

$$L = I \quad (11)$$

The trajectory is a circular path. The final result is shown in Figure 3. The system with the LQR controller fits the reference path much better than the open-loop. And all the forces are smaller than 2 N.

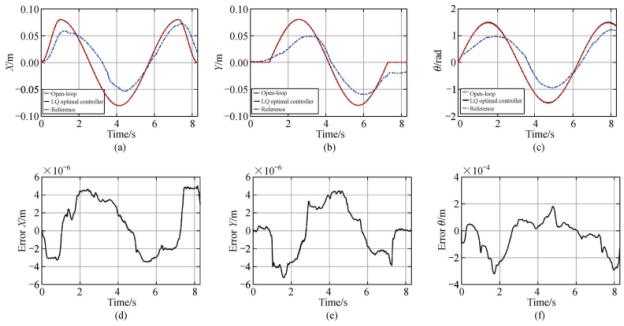

Figure 3. Comparison of the open-loop and the LQR controller of the circular trajectory from [5]. (a)X-axis (b)Y-axis (c)rotation angle (d)-(f) the errors

Nonlinear System The LQR controller is only suitable for linear systems, which means that the state equation of the system should be able to be transformed into a linear form. In practical problems, many models cannot be simply described by linear systems. But this does not mean that the LQR controller cannot handle these nonlinear problems. The local linearization method can be used to transform nonlinear systems into linear systems on a local scale [3].

Kazeminasab et al. [7] tried to solve the nonlinear problem with the LQR controller. This research aims at an in-pipe robot, which is used to detect the quality of drinking water in WDS (Water distribution System). The geometrical representation of the robot is shown in figure 4. Due to the

pressure caused by the water flow in the pipe and the uncertainty of the pipe shape, the motion control of the in-pipe robot in the pipe becomes a challenge. The authors propose a motion control algorithm that can stabilize the robot while tracking the desired velocity. As shown in the figure

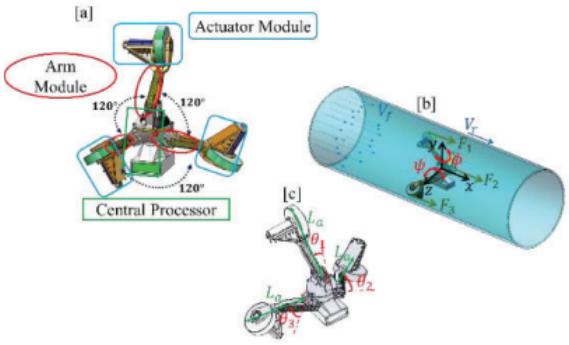

Figure 4. Robot modelling from [7]. (a)CAD design (b)Free body diagram in the pipe (c)geometrical representation

4 (b), the power of the in-pipe robot is provided by three actuators, and the forces applied are F_1, F_2, and F_3. In addition, there is another dominant force applied to the robot: the drag force F_d. The drag force produced by the fluid is typically nonlinear, and its formula is as follows [8]:

$$F_d = \frac{1}{2}\rho C_d A(V_r - V_f)^2 \tag{12}$$

where ρ is water density, C_d is the robot's drag coefficient and A is the frontal area of the in-pipe robot facing the flow. V_r is the velocity of the robot in the X-axis, and V_f is the velocity of the flow. The overall motion equation of the robot can be described as [7]:

$$F_1 + F_2 + F_3 + \frac{1}{2}\rho C_d A(V_r - V_f)^2 = m\ddot{x} \tag{13}$$

Since the control purpose of the in-pipe robot is to make the robot track the target velocity, V_r in the formula is one of the states in the system. Equation (13) is a quadratic equation about the state, not linear. The authors also discuss the equation of the torque in the paper. In this section, only the dynamical equation (13) is used as an example to introduce the application of local linearization.

F_1, F_2, and F_3 are produced from the actuators, which are the input of the system. Therefore, these three forces can be represented by vector u. Then the dynamical equation can be defined as $\dot{x}_s = F(x_s, u)$. The local linearization method is to linearize the equation around the equilibrium point with Taylor series. The equilibrium point of the state and the input can be easily obtained: $x_s^e = [0 \quad 0 \quad 0 \quad 0]^T$ and $u_e = [0 \quad 0 \quad 0]^T$, and the state equation at this point is like [7]:

$$\dot{x}_s = F(x_s, u) = F(x_s^e + \delta x_s, u_e + \delta u) \tag{14}$$

According to the Taylor series, it can be:

$$\delta\dot{x}_s = \frac{\partial F}{\partial x_s}\delta x_s + \frac{\partial F}{\partial u}\delta u + O(x_s) + O(u) \tag{15}$$

where $O(x_s)$ and $O(u)$ are terms with higher order in Taylor expansion, which can be dismissed. The rest of the equation is like:

$$\delta\dot{x}_s = \frac{\partial F}{\partial x_s}\delta x_s + \frac{\partial F}{\partial u}\delta u = A\delta x_s + B\delta u \tag{16}$$

Then the partial derivatives can be used as the A and B matrices in the equation of state. The nonlinear dynamical equations can be transformed into local linear. When the linearization is successfully achieved, the next steps are the same as the example in the previous section. By selecting the appropriate Q and R matrices, the LQR controller is implemented to track the target speed and maintain the stability of the robot. Figure 5 and 6 show the final result, it takes less than 0.5 seconds for the robot to reach the desired velocity and the deviation angle can be stabilized in 1 second. It should be noticed that the author made a small mistake in the legend. The desired velocity of the yellow line in figure 5 should be 0.5.

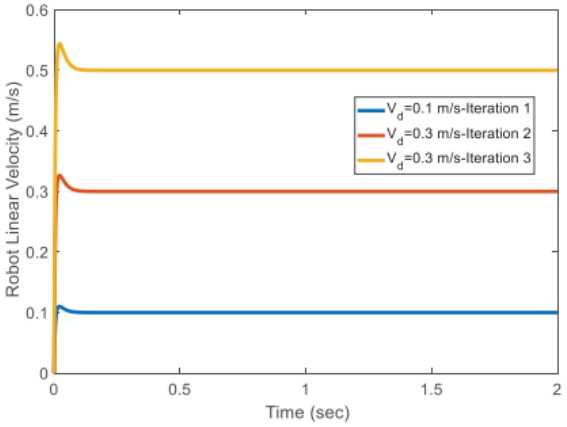

Figure 5. Linear velocity of the robot from [7]

Combined with Advanced Algorithm The applications of the LQR controller in the linear and nonlinear system are introduced before. A remained problem is that the selection of Q and R matrices must be done manually. It is very empiric and time-consuming. More and more engineers started

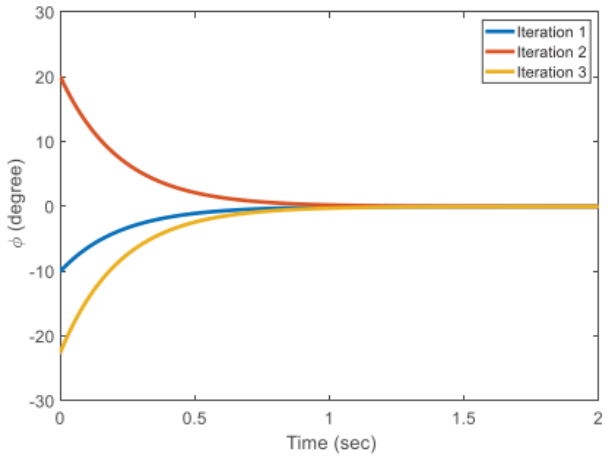

Figure 6. Performance of the controller in stabilizing $\phi$from [7]

to find a method to implement the automation of the parameter selection step.

Howimanporn et al. [9] proposed to combine LQR controller with PSO (particle swarm optimization) method. PSO is an evolutionary algorithm that was developed by Kennedy and Eberhart [10]. This algorithm optimizes problems based on iterative cooperation and competition among individuals by imitating the behavior of individuals in a group. The authors use the LQR controller to stabilize the inverted pendulum system and use PSO to determine the optimal gain under the condition of the LQR controller cost function. The diagram of the system is shown in figure 7.

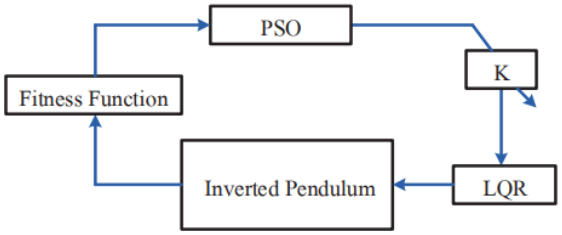

Figure 7. Block diagram of PSO based LQR on inverted pendulum system from [9].

The inverted pendulum model is a very classic control problem. Therefore, the dynamical equation of the inverted pendulum will not be presented here. PSO algorithm is adopted to search for the optimal gain of the LQR controller. The generation is set to 100 iterations and the population size is initialized as 40. The state-space contains two states: the position and the velocity. The velocity in the next iteration can be calculated by [9]:

$$v_{ij}^{k+1} = wv_{ij}^{k} + c_1 r_1 (pbest_{ij}^{k} - x_{ij}^{k}) + c_2 r_2 (gbest_{ij}^{k} - x_{ij}^{k}) \tag{17}$$

where v_{ij}^{k+1} is the current velocity of particle generation and v_{ij}^{k} is the velocity in the previous generation. w is the factor changes iteratively. r_1 and r_2 are generated randomly to simulate the random influence from the environment. c_1 and c_2 are preset constants. $pbest_{ij}^{k}$ is the individual best solution and $gbest_{ij}^{k}$ is the global best solution. The individual best solution should be set according to the limitation and target of each particle, and determine the global solution for the population [9]. x_{ij}^{k} is the previous position, and the current position can be calculated by:

$$x_{ij}^{k+1} = x_{ij}^{k} + v_{ij}^{k+1} \tag{18}$$

After designing the PSO algorithm and choosing the parameter of Q and R matrices, calculate the result and compare the global best with the target. If the termination criterion is met, the global best solution is the optimal gain. If no, back to equation (17) and (18) to calculate in the next generation. Figure 8 shows the final result, it can be seen that after 30 - 40 iterations, the value of the cost function reaches a good result [9].

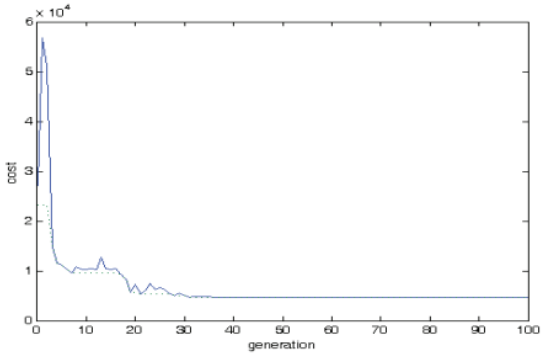

Figure 8. Value of global best of performance in the cost function from [9].

2.2. Model Predictive Control

2.2.1 Introduction

Model predictive control (MPC) is an advanced method of process control based on the principle of predictions to a finite horizon at each sampling time starting from the current state. Since the state of the system is updated during each sampling period, a new optimization problem must be solved recursively. It uses a dynamic model of the system

to make predictions about the system's future behavior, and optimize the forecast to produce the best decision while satisfying a set of constraints. Therefore, the model, which is intended to represent the behavior of complex dynamical systems, plays an important role in MPC.

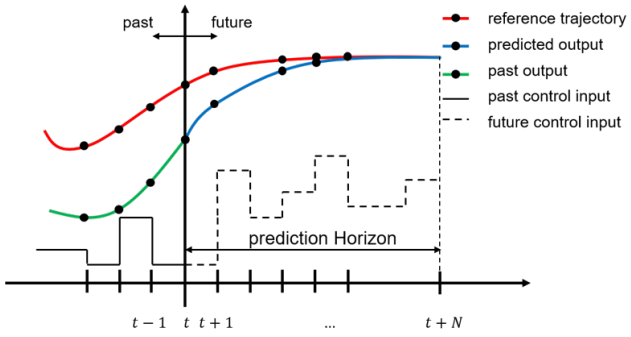

Figure 9. Model Predictive Control Scheme from Varma et al. [11]

The MPC strategy is illustrated in 9. The future outputs for a determined horizon N, called the prediction horizon, are predicted at each sampling time using the dynamic model of the system. The past trend for the outputs y up to t and inputs u up to t are known. The controller is then to find the future trend for the control inputs which keep the predicted outputs as close as possible to the reference trajectory. This step usually takes the form of a quadratic cost function of the errors between the predicted output and the reference trajectory. The control inputs are obtained through iteration. For linear systems, if a linear or quadratic objective function is considered, the resulting optimization is a linear or a quadratic programming problem, respectively. In contrast, for nonlinear systems, the optimization becomes a nonlinear programming problem which is nonconvex in the majority of cases. This MPC scheme is implemented using the block diagram shown below

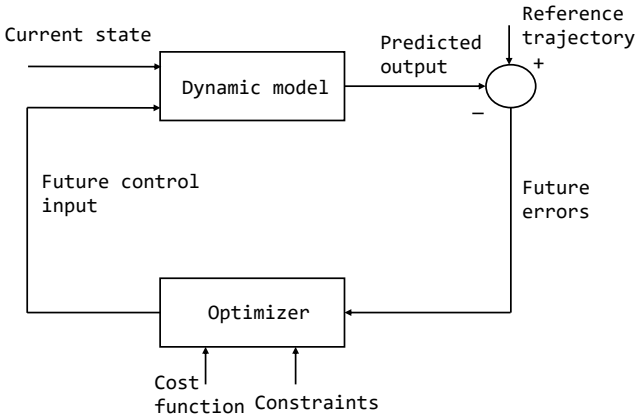

Figure 10. Model Predictive Control Block Diagram

As shown in the figure 10, MPC uses a dynamic model to predict the future system outputs, based on past and current values and on the proposed optimal future control actions. With the reference trajectory, the difference between the reference trajectory and the predicted output is fed back to the optimizer, which minimizes the quadratic cost function and considers the constraints to determine the controller outputs. The controller outputs are implemented in real-time and then the process is repeated every sampling time with actual process data.

2.2.2 MPC Method

MPC is used to control a process while satisfying a set of constraints and uses a model of the plant to make predictions about future plant outputs. In Alothman et al.[12], a nonlinear MPC controller is developed to control the suspended payload position carried with two quadrotors by cables. The dynamic model is derived using the Euler-Lagrange equation. In Varma et al.[11], the authors compares different controllers' performances for trajectory tracking in autonomous driving. The dynamic model of vehicle is derived using Newton-Euler Method. In Chen et al.[13], the authors design input convex networks to obtain accurate models of complex physical systems. Then optimal controllers can be achieved via solving a convex model predictive control problem.

Model of Plant Predictive models are the mathematical model which describe the behaviour of the real system. The two most commonly used mathematical modeling methods are Newton-Euler and Euler-Lagrangian.

Euler-Lagrange Method In Alothman et al.[12], the dynamics of quadrotor-payload system with 13 degrees of

freedom are derived from Euler-Lagrange method which is energy-based approach:

$$L(q, \dot{q}) = T(q, \dot{q}) - U(q) \tag{19}$$

where $L(q, \dot{q})$ is the Lagrangian, $T(q, \dot{q})$ is kinetic energy and $U(q)$ is potential energy. Then the Euler-Lagrange equation is

$$\frac{d}{dt}(\frac{\partial L}{\partial \dot{q}}) - \frac{\partial L}{\partial q} = Q \tag{20}$$

The generalised force Q defined here is based on the choice of the generalised coordinates q and the external conservative force F_i.

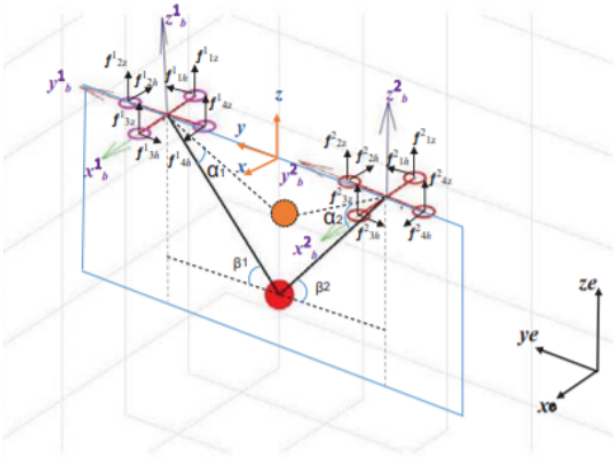

Figure 11. Two quadrotors carrying a payload

Taking the generalised forces and equation (19) into equation (20), the Euler-Lagrange equation based on M matrix and the system model function f can be rewritten in

$$M\ddot{q} = f(q, \dot{q}) \tag{21}$$

And the nonlinear discrete dynamic model is:

$$\mathbf{x}_{k+1} = f(\mathbf{x}_k, u_k) \tag{22}$$

*Newton-Euler Method In Varma et al. [11] and Chen et al. [14], the vehicle model applied in both papers is the vehicle bicycle kinematic model and the vehicle dynamics is derived from Newton-Euler method. The updated vehicle state after one sampling time is:

$$\begin{aligned} v_1 &= v_0 + aT_r \\ \theta_1 &= \theta_0 + kl \\ x_1 &= x_0 + \frac{\sin(\theta_0 + kl) - \sin(\theta_0}{k} \\ y_1 &= y_0 + \frac{\cos(\theta_0 + kl) - \cos(\theta_0}{k} \end{aligned} \tag{23}$$

where T_r is sampling time and the distance the vehicle moves in one sampling time is $l = v_0 T_r + \frac{1}{2} a T_r^2$, and the curvature is $k = \frac{\tan\delta}{L}$. The state vector includes $x = [x, y, v, \theta]$ which is x-y position and velocity, and heading angle. The control input includes $u = [a, \delta]$ which is the acceleration and the steering angle. L is the wheel base. The vehicle dynamic equation can be rewritten as:

$$x_{k+1} = f(x_k, u_k) \tag{24}$$

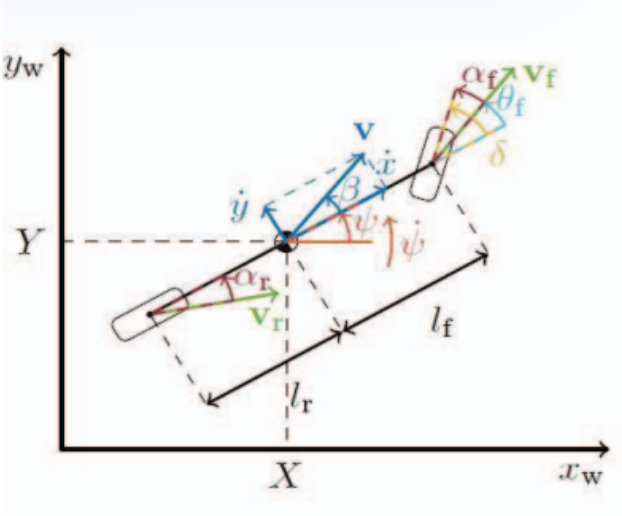

Figure 12. Bicycle Model Co-Ordinamte System

Cost function Once the model is built, the formulation of the MPC problem still requires the cost function to be minimized and the constraints to be imposed. The general cost function is concerned with the penalties for states and control inputs. Each cost is calculated for all stages of the prediction and added together. The cost function in Chen et al. [14] has the following form:

$$\begin{aligned} J = \quad & (x_N - x_{ref,N})^T Q_f (x_N - x_{ref,N}) \\ & + \sum_{i=0}^{N-1} ((x_i - x_{ref,i})^T Q (x_i - x_{ref,i}) \\ & + (u_i - u_{ref,i})^T R (u_i - u_{ref,i})) \end{aligned} \tag{25}$$

where the terminal state and its desired state are denoted by x_N and $x_{ref,N}$, respectively, and the reference state is denoted by $x_{ref,i}$. The prediction horizon is denoted by N. Q_f and Q are presented by positive semidefinite state matrices and R is presented by positive definite matrix. Regardless of the UAV system of Alothman et al.[12] or the vehicle of Chen et al. [14], the cost penalty for the errors between reference states and actual states is for trajectory

tracking, and the cost penalty for the control inputs is to penalize rapid change of control inputs which is always related to safety and comfort.

Constraints Additionally, operational constraints must be taken into account during the minimization of the cost function. In reality, all systems are subject to constraints. There are a lot of reasons causing limits in a system, such as safety, regulations, or environmental conditions. In most MPC problem formulations, only two types of constraints considered are states and control inputs constraints. The constraints on the inputs are in general hard constraints which impose lower and upper bounds on these variables, that is:

$$u(k) \in \mathcal{U} := u \in R^m : u_{min} \leq u \leq u_{max}, \forall k \in [0, N-1] \tag{26}$$

The states constraints are in general of the form:

$$x(k) \in \mathcal{X} := x \in R^m : x_{min} \leq x \leq x_{max}, \forall k \in [0, N-1] \tag{27}$$

Other constraints can also be introduced for specific goals such as collision avoidance goals in Chen et al. [14]:

$$d(x_k, O_j^k) > 0, k = 1, 2, ..., N. J = 1, 2, ..., M \tag{28}$$

There are m obstacles and O_j^k is the space occupied by the jth obstacle at time step k, $d(x_k, O_j^k)$ is the distance from the host vehicle to the jth obstacle at time step k.

Optimal Control Problem With a dynamics model, cost function and constraints on states and control inputs, a discrete time version of the constrained optimal control problem (OCP) is shown:

$$\min_u J \tag{29a}$$

$$\text{s.t.} \quad x_{k+1} = F(x(k), u(k)) \tag{29b}$$

$$x(0) = x_0 \tag{29c}$$

$$x(k) \in \mathcal{X} \tag{29d}$$

$$u(k) \in \mathcal{U} \tag{29e}$$

2.2.3 Comparison between LQR and MPC

The research from Varma et al. [11] compares different controllers' performances for trajectory tracking in autonomous driving. First is longitudinal speed tracking, the graph below shows the tracking performance for a reference speed of 40 km/hr

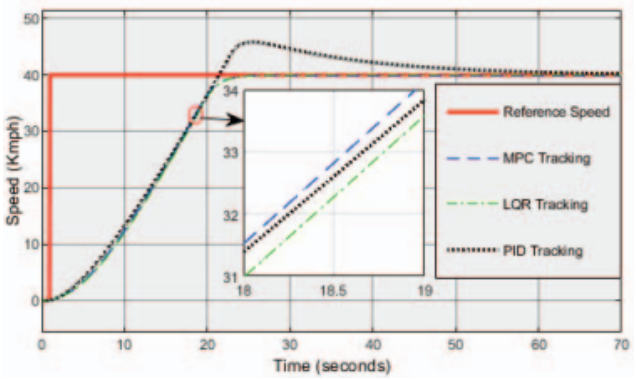

Figure 13. Longitudinal Speed Tracking @40 km/hr from Varma et al. [11]

Rise time for MPC < LQR < PID and overshoot is also very negligible for LQR and MPC, so it is obvious that MPC performs better for Longitudinal Speed control. For lateral steering control, the lateral position errors are analyzed and compared for different controllers:

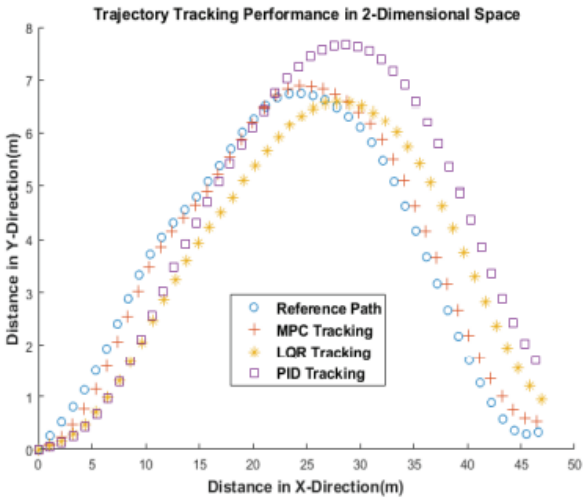

Figure 14. Lateral Trajectory Tracking from Varma et al. [11]

Here it is also obviously that MPC performs better than LQR.

2.2.4 Advantages and Disadvantages of MPC

The main advantage of MPC is its flexibility in achieving complex goals and imposing constraints. However, it also has some disadvantages such as high computational load in the control process. The main weakness of MPC is the performance highly depends on the detailed dynamics models.

Many parameters are needed for describing the model. In these papers, the controlled plants are quadrotors and vehicles, and the vehicles are simplified into bicycle models. None of these models are really complex and do not have a large number of parameters. What if the plant is far more complex than vehicles and quadrotors and cannot be simplified? It could take months or years to compute the corresponding dynamic models. To solve this problem, the paper Chen et.al [13] try to parameterize the complex system dynamics using deep neural networks to capturing complex relationship. The overall methodology is shown below

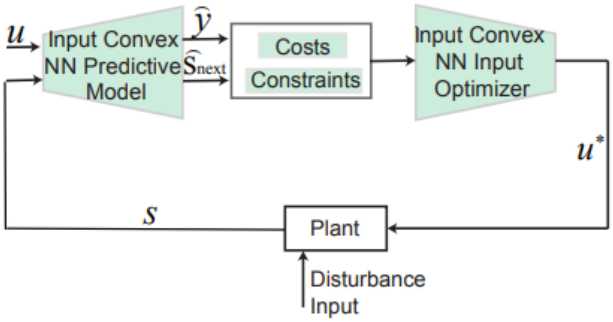

Figure 15. Optimal Control via Neural Networks from Chen et.al [13]

Firstly, it ultilizes an input convex network model to learn the system dynamics. Then to compute the best control decisions via solving a convex model predictive control problem. Comparing with previous controlling problem, here only the dynamics model is changed by using neural networks instead of Newton-Euler or Euler-Lagrangian method.

2.3. Data-Enabled Predictive Control

2.3.1 Context [15]

Traditionally, control methods have been based on physical dynamic models, which have been expected to describe the true underlying behavior of the system in a reasonably accurate representation. These descriptions, in complex dynamic systems, are often highly elusive or nonexistent, hence the importance of using data-driven approaches. Data-driven models are based on data measurements of the real system and require minimal prior knowledge of the system. However, they require new control strategies as classical analysis and synthesis tools cannot handle probabilistic models.

The extensive field of data-driven control methods is categorized into two areas: indirect data-driven control approaches, consisting of sequential system identification and model-based control, and direct data-driven control approaches, which seek an optimal decision suitable for the input data. These approaches both have a rich history, and they have received a renewed interest due to new methods and widespread interest in machine learning. The advantages and disadvantages of both paradigms have been commonly elaborated: the modeling and identification, in particular, is cumbersome, and the modeling results are often not useful for control, since operators generally prefer the end-to-end approaches. Whereas direct data-driven control solve these problems by learning control policies directly from data, but existing methods often are not suitable for real-time and safety-critical applications, due to lack of certificates among other reasons.

Outline: in this section, the major focus is to solve the control problem without the intermediate step of identifying the system or estimating its state: the aim is a direct end-to-end solution and is based on the DeePC algorithm. First, a simple control problem, that will provide context for explanations, will be stated. Afterwards, an introduction to the bevahior approach is provided as it is a fundamental base for the DeePC logic. The system is later formulated for an MPC-based method, before being expressed in the DeePC framework. Some practical examples and enhancements of DeePC will be presented, as well as some outlook considerations.

2.3.2 Literature review summary

The fundamental lemma of Willems et al. [16] shows that all trajectories of a linear system can be obtained from a given finite number of trajectories, this result is of practical importance in the control of unstable systems by data. In Coulson et al. [17], The authors consider the problem of optimal path following for unknown systems. They present a new data-driven predictive control (DeePC) algorithm that computes optimal and safe control policies using real-time feedback to drive the unknown system along a desired trajectory while satisfying system constraints and relying on Willems' lemma. In Alpago et al. [18], the authors extend DeePC to make use of additional available input-output data for reducing the effect of noise. In Elokda et al. [19], the authors study the application of DeePC for position control of real-world nano-quadcopters. In Huang et al.[20], the authors use DeePC in voltage source converter (VSC)-based high-voltage direct current (HVDC) stations to achieve safe and optimal control over a wide area for damping power system oscillations.

2.3.3 Problem statement [17][19]

Consider the discrete-time system given by

$$\begin{cases} x(t+1) = Ax(t) + Bu(t) \\ y(t) = Cx(t) + Du(t) \end{cases} \tag{30}$$

where $A \in R^{n\times n}, B \in R^{n\times m}, C \in R^{p\times n}, D \in R^{p\times m}$, and $x(t) \in R^n, u(t) \in R^m, y(t) \in R^p$ are respectively the state, control input, and output of the system at time $t \in Z_{\geq 0}$. Given a desired reference trajectory $r = (r_0, r_1, \ldots) \in (R^p)^Z \geq 0$, input constraint set $\mathcal{U} \subseteq R^m$, output constraint set $\mathcal{Y} \subseteq R^p$,

The objective is to apply control entries in order for the system output to follow the reference trajectory r while satisfying constraints and optimizing a cost function. In the case where the model of the system is known, i.e. the matrices A, B, C and D are known, the problem can be approached using the MPC (see section 2.2). Unless stated, the theoretical problem considered in this section (2.3) is this path-following problem in the case where the model of the system is unknown, but where input/output data samples are available. [17]

2.3.4 Non-parametric system representation [16][17]

The approach taken by the current state of the art is to use a behavior viewpoint to be able to substitute the need for a model or system identification process. The behavioral approach of systems theory starts from a deep but simple idea: a dynamic system is a set of trajectories that the system can express.This is done by defining Linear time-invariant (LTI) systems as trajectory sets and constructing any controllable trajectory of an LTI system using a finite number of data samples generated by a sufficiently rich input signal. To do this, they introduce the Hankel matrix which is itself a non-parametric predictive model based on raw data and which implicitly estimates the state of an LTI system.

Considering a Linear time-invariant system:

$$B(A,B,C,D) = \Big\{ \mathrm{col}(u,y) \in \left(R^{m+p}\right)^{Z_{\geq 0}} \mid \exists x \in (R^n)^Z_{\geq 0} \text{ s.t. } \sigma x = Ax + Bu, y = Cx + Du \Big\}. \tag{31}$$

Definition: A dynamical system is a 3-tuple $(Z_{\geq 0}, W, B)$ where $Z_{\geq 0}$ is the discrete-time axis, W is a signal space, and $B \subseteq W^Z \geq 0$ is the behaviour.

Definition: Let $L, T \in Z > 0$ such that $T \geq L$. The signal $u = \mathrm{col}\,(u_1, \ldots, u_T) \in R^{Tm}$ is persistently exciting of order L if the Hankel matrix

$$H_L(u) := \begin{pmatrix} u_1 & \cdots & u_{T-L+1} \\ \vdots & \ddots & \vdots \\ u_L & \cdots & u_T \end{pmatrix} \tag{32}$$

is of full row rank.

The term persistently exciting is intended to characterize an input signal sufficiently rich to excite the system and generate an output sequence representative of the system's behavior

Each column of the Hankel matrix contains an input/output trajectory of length L and by linearity of the system, each linear combination of the matrix column is also a trajectory of the LTI. In other words, the result of multiplying the Hankel matrix by any real vector g is a trajectory expressible by the system. Furthermore, for Linear time-invariant systems under certain necessary and sufficient conditions based on the rank of the Hankel matrix of the data, this matrix can generate all the trajectories and thus describe the system completely:

Theorem: Let $T_\mathrm{d}, L \in Z_{>0}$. Let $(u_\mathrm{d}, y_\mathrm{d}) = \{(u_\mathrm{d}(i), y_\mathrm{d}(i))\}_{i=1}^{T_\mathrm{d}}$ be a trajectory of (30) of length T_d such that $\{u_\mathrm{d}(i)\}_{i=1}^{T_\mathrm{d}}$ is persistently exciting of order $L+n$. Then $(u, y) = \{(u(i), y(i))\}_{i=1}^{L}$ is a trajectory of if and only if there exists $g \in R^{T_\mathrm{d}-L+1}$ such that

$$\begin{pmatrix} H_L(u_\mathrm{d}) \\ H_L(y_\mathrm{d}) \end{pmatrix} g = \begin{pmatrix} u \\ y \end{pmatrix}. \tag{33}$$

2.3.5 MPC problem formulation[17]

Particular attention has been devoted to the problem of optimal trajectory tracking, widely studied in model-based control. The model predictive control and estimation algorithm for trajectory tracking when the system model is known is given by considering the following optimization problem:

$$\begin{aligned} \text{minimize}_{u,x,y} \quad & \textstyle\sum_{k=0}^{N-1} \left(\|y_k - r_{t+k}\|_Q^2 + \|u_k\|_R^2 \right) \\ \text{subject to} \quad & x_{k+1} = Ax_k + Bu_k, \forall k \in \{0, \ldots, N-1\} \\ & y_k = Cx_k + Du_k, \forall k \in \{0, \ldots, N-1\} \\ & x_0 = \hat{x}(t) \\ & u_k \in \mathcal{U}, \forall k \in \{0, \ldots, N-1\} \\ & y_k \in \mathcal{Y}, \forall k \in \{0, \ldots, N-1\} \end{aligned} \tag{34}$$

where $N \in Z_{>0}$ is the time horizon, $u = (u_0, \ldots, u_{N-1})$, $x = (x_0, \ldots, x_N), y = (y_0, \ldots, y_{N-1})$ are the decision variables, and $r_{t+k} \in R^p$ is the desired reference at time $t+k$, where $t \in Z \geq 0$ is the time at which the optimization problem should be solved. The norm $\|u_k\|_R$ denotes the quadratic form $u_k^T R u_k$ (similarly for $\|\cdot\|_Q$), where $R \in R^{m\times m}$ is the control cost matrix and $Q \in R^{p\times p}$ is the output cost matrix. The estimated state at time t is denoted by $x(t)$ and the predicted state and output at time $t+k$ are denoted by x_k and y_k, respectively. If the entire state is measured then $\hat{x}(t) = x(t)$.

2.3.6 MPC Algorithm[17]

The classical MPC algorithm involves solving optimization problem (34) in a receding horizon manner.

Algorithm MPC

Input: (A, B, C, D), reference trajectory r, past input/output data (u, y), constraint sets $\mathcal{U}$ and $\mathcal{Y}$, and performance matrices Q and R
1) Generate state estimate $\hat{x}(t)$ using past input/output data.
2) Solve (34) for $u^\star = \left(u_0^\star, \ldots, u_{N-1}^\star\right)$.
3) Apply inputs $(u(t), \ldots, u(t+s)) = (u_0^\star, \ldots, u_s^\star)$ for some $s \leq N-1$.
4) Set t to $t+s$ and update past input/output data.
5) Iterate through Steps 1,4

2.3.7 DeePC problem formulation for deterministic LTI systems [17][19]

Problem formulation Let $T_{\mathrm{d}}, T_{\text{ini}} \in Z_{>0}$ be the length of data collection and the time horizon used for initial condition estimation, respectively. Suppose $(u_{\mathrm{d}}, y_{\mathrm{d}}) = \{(u_{\mathrm{d}}(i), y_{\mathrm{d}}(i))\}_{i=1}^{T_{\mathrm{d}}}$ is a sequence of input/output measurements collected during an offline procedure. Suppose further that the input $\{u_{\mathrm{d}}(i)\}_{i=1}^{T_{\mathrm{d}}}$ is persistently exciting of order $T_{\text{ini}} + T_{\mathrm{f}} + n$. The input/output measurements are partitioned into Hankel matrices

$$\begin{pmatrix} U_{\mathrm{p}} \\ U_{\mathrm{f}} \end{pmatrix} := H_{T_{\text{in}}+T_{\mathrm{f}}}(u_{\mathrm{d}}), \begin{pmatrix} Y_{\mathrm{p}} \\ Y_{\mathrm{f}} \end{pmatrix} := H_{T_{\text{ini}}+T_{\mathrm{f}}}(y_{\mathrm{d}}), \tag{35}$$

where U_{p} consists of the first T_{ini} block rows of $H_{T_{\text{ini}}+T_{\mathrm{f}}}(u_{\mathrm{d}})$ and U_{f} consists of the last T_{f} block rows of $H_{T_{\text{ini}}+T_{\mathrm{f}}}(u_{\mathrm{d}})$ (similarly for Y_{p} and Y_{f}). The data in U_{p} and Y_{p} will be used in conjunction with past data to perform implicit initial condition estimation, and the data in U_{f} and Y_{f} will be used to predict future trajectories.

Let $(u_{\text{ini}}, y_{\text{ini}}) = \{(u_{\text{ini}}(t+i), y_{\text{ini}}(t+i))\}_{i=-T_{\text{inin}}}^{-1}$ be the T_{ini} most recent past input/output measurements from the system. By Theorem described by the equation (33), $(u, y) = \{u(t+i), y(t+i)\}_{i=0}^{T_i-1}$ is a possible future trajectory of (30) if and only if there exists $g \in R_{\mathrm{d}}^{T_{\mathrm{d}}-T_{\text{ini}}-T_{\mathrm{f}}+1}$ satisfying

$$\begin{pmatrix} U_{\mathrm{p}} \\ Y_{\mathrm{p}} \\ U_{\mathrm{f}} \\ Y_{\mathrm{f}} \end{pmatrix} g = \begin{pmatrix} u_{\text{ini}} \\ y_{\text{ini}} \\ u \\ y \end{pmatrix} \tag{36}$$

Each column of the Hankel matrix is a trajectory of the system (motion primitive), and any new trajectory (right side of (36)) can be computed by a linear combination of these motion primitives. Hence, given an input sequence u to be applied to the system, the first three block equations of (36) can be solved for g, and the corresponding output sequence is given by $y = Y_{\mathrm{f}} g$. The first two blocks of equations in (36) are used to implicitly fix the initial condition from which the future trajectory begins. In order to uniquely define the initial condition from which the future trajectory starts, they define $T_{textini} \geq \ell$, where ℓ is the delay of the system (i.e., the length of time in the past required to uniquely identify the current state of the system by back-propagation of the dynamics (30)). This in turn implies that the predicted trajectory given by $y = Y_{\mathrm{f}}\mathrm{g}$ is unique.

The Hankel matrix in (36) performs both state estimation and prediction simultaneously, and thus can be used as a predictive model for the system (30). Substituting (36) for the unknown dynamics in the optimization problem yields the following data-driven optimization problem, which calculates the optimal control inputs without knowing the system model:

Given a time horizon $N \in Z > 0$, a reference trajectory $r = (r_0, r_1, \ldots) \in (R^p)^{Z_{\geq 0}}$, past input/output data $\mathrm{col}\,(u_{\text{ini}}, y_{\text{ini}}) \in B_{T_{\text{ini}}}$, input constraint set $\mathcal{U} \subseteq R^m$, output constraint set $\mathcal{Y} \subseteq R^p$, state cost matrix $Q \in R^{p\times p}$, and control cost matrix $R \in R^{m\times m}$, the DeePC formulation is as follows:

$$\begin{aligned} \underset{g,u,y}{\text{minimize}} \quad & \textstyle\sum_{k=0}^{N-1} \left(\|y_k - r_{t+k}\|_Q^2 + \|u_k\|_R^2 \right) \\ \text{subject to} \quad & \begin{pmatrix} U_{\mathrm{p}} \\ Y_{\mathrm{p}} \\ U_{\mathrm{f}} \\ Y_{\mathrm{f}} \end{pmatrix} g = \begin{pmatrix} u_{\text{ini}} \\ y_{\text{ini}} \\ u \\ y \end{pmatrix}, \\ & u_k \in \mathcal{U}, \forall k \in \{0, \ldots, N-1\}, \\ & y_k \in \mathcal{Y}, \forall k \in \{0, \ldots, N-1\}. \end{aligned} \tag{37}$$

2.3.8 DeePC algorithm for deterministic LTI systems [17][19]

Algorithm DeePC

Input: $\mathrm{col}\left(u^{\mathrm{d}}, y^{\mathrm{d}}\right) \in B_T$, reference trajectory $r \in R^{Np}$, past input/output data $\mathrm{col}\,(u_{\text{ini}}\,, y_{\text{ini}}\,) \in B_{T_{\text{ini}}}$, constraint sets $\mathcal{U}$ and $\mathcal{Y}$, and performance matrices Q and R
1) Solve 37 for $g^\star$.
2) Compute the optimal input sequence $u^\star = U_{\mathrm{f}} g^\star$.
3) Apply input $(u(t), \ldots, u(t+s)) = (u_0^\star, \ldots, u_s^\star)$ for some $s \leq N-1$.
4) Set t to $t+s$ and update u_{ini} and y_{ini} to the T_{ini} most recent input/output measurements.
5) Iterate through Steps $1, 4$

2.3.9 Equivalence of DeePC and MPC for deterministic LTI systems [17]

After introducing this new algorithm 2.3.8, it seems reasonable to ask about its performance. For this purpose, DeePC has been compared to MPC in several settings, the first one

being the deterministic LTI. Two aspects considered are the comparative complexity and the trajectories obtained after applying the control obtained by the 2 algorithms.

First, MPC has a considerable computational complexity disadvantage because of the iterative computations at each time step while a significant feature of the DeePC algorithm is its simplicity. DeePC algorithm performs the system identification, state estimation and trajectory prediction with a single linear equation, which results in a quadratic program with $T - T_{\text{ini}} - N + 1$ number of decision variables where $T \in Z_{>0}$ is the amount of data collected.

Second, regarding trajectories, it can be shown that Algorithm 2.3.6 and Algorithm 2.3.8 give equivalent trajectories in closed loop under certain assumptions.

Corollary : (Equivalent Closed Loop Behaviour): Consider a controllable LTI system $B \in \mathcal{L}^{m+p}$ with minimal input/ouput/state representation $B(A, B, C, D)$ given as in 30 . Consider the MPC and DeePC optimization problems 34 and 37 with $Q \succeq 0, R \succ 0$, and $\mathcal{U}, \mathcal{Y}$. Under persistently exciting conditions (obtained by collecting a certain necessary amount of data samples). Algorithm MPC and Algorithm DeePC result in equivalent closed loop behaviour, i.e., the optimal control sequence $u^\star$ and corresponding system output $y^\star$ at every iteration is identical

2.3.10 Regularized DeePC Algorithm [17][18]

Consider now the nonlinear discrete-time system given by

$$\begin{cases} x(t+1) = f(x(t), u(t)) \\ y(t) = h(x(t), u(t), \eta(t)) \end{cases} \tag{38}$$

where $\eta(t) \in R^p$ is white measurement noise, and $f : R^n \times R^m \to R^n$ and $h : R^n \times R^m \times R^p \to R^p$ are not necessarily linear. To apply the DeePC some modifications are made to 37 and lead to the following regularized optimization problem:

$$\begin{aligned} \underset{g,u,y,\sigma_y}{\text{minimize}} \quad & \sum_{k=0}^{N-1} \left(\|y_k - r_{t+k}\|_Q^2 + \|u_k\|_R^2 \right) \\ & + \lambda_g \|g\|_1 + \lambda_y \|\sigma_y\|_1 \\ \text{subject to} \quad & \begin{pmatrix} \widehat{U}_{\text{p}} \\ \widehat{Y}_{\text{p}} \\ \widehat{U}_{\text{f}} \\ \widehat{Y}_{\text{f}} \end{pmatrix} g = \begin{pmatrix} u_{\text{ini}} \\ y_{\text{ini}} \\ u \\ y \end{pmatrix} + \begin{pmatrix} 0 \\ \sigma_y \\ 0 \\ 0 \end{pmatrix}, \\ & u_k \in \mathcal{U}, \forall k \in \{0, \dots, N-1\}, \\ & y_k \in \mathcal{Y}, \forall k \in \{0, \dots, N-1\}, \end{aligned}$$

where $\sigma_y \in R^{T_{\text{ini}} p}$ is an auxiliary slack variable, $\lambda_y, \lambda_g \in R_{>0}$ are regularization parameters, and $\text{col}\left(\widehat{U}_{\text{p}}, \widehat{Y}_{\text{p}}, \widehat{U}_{\text{f}}, \widehat{Y}_{\text{f}}\right)$ is a low-rank matrix approximation of $\text{col}\left(U_{\text{p}}, Y_{\text{p}}, U_{\text{f}}, Y_{\text{f}}\right)$.

Slack variable When the output measurements are corrupted by noise, the constraint equation in 37 can become inconsistent. Therefore, in 2.3.10, by including the slack variable σ_y in the constraint, the feasibility of the constraint is guaranteed at all times. The slack variable is penalized with a weighted penalty function to a norm.

One-norm regularization on g The cost includes a penalty to on the norm of g, favoring sparsity and thus selecting only the most informative trajectories of noisy data to predict future behavior.

Low-rank approximation By approximating the low-rank matrix (via singular value decomposition (SVD)), the most dominant sub-behavior is considered, resulting in a data matrix describing the behavior of the nearest deterministic LTI system. In the case of noisy measurements, the SVD filters the noise. In the case of nonlinear dynamics, SVD results in a matrix describing an approximate LTI model, i.e., the most relevant infinitely dimensional elevator basis functions, whose dimension can be chosen by adjusting the SVD cutoff. The DeePC algorithm does not require that the matrix $\text{col}\left(\widehat{U}_{\text{p}}, \widehat{Y}_{\text{p}}, \widehat{U}_{\text{f}}, \widehat{Y}_{\text{f}}\right)$ have a Hankel structure.

Averaging Hankel Matrices: The output matrices Y_p and Y_f are constructed offline from the trajectory $y_{0,T-1}$. Suppose that multiple T-long output trajectories $y^{(1)}_{0,T-1}, \dots, y^{(N)}_{0,T-1}$ are available. Using those additional data online to improve the prediction will lead to an intractable optimization problem. However, the additional offline paths can be used to construct N different Hankel matrices $\mathcal{H}^{(1)}, \dots, \mathcal{H}^{(N)}$ defined analogously to 35, and average those matrices to obtain the Hankel matrix.

$$\overline{\mathcal{H}}_N := \frac{1}{N} \sum_{i=1}^{N} \mathcal{H}^{(i)}$$

The Law of Large Numbers guarantees that making use of additional data to mitigate the effect of noise in the data-driven model, hence reducing the risk of overfitting the data-driven model, hence reducing the risk of overfitting that would be present if the data was used directly in DeePC.

2.3.11 Applications [17][19][20]

Quadcopter In [17] and [19], the Regularized DeePC algorithm was applied to a high-fidelity nonlinear quadcopter model and compared to system identification (ID) followed by MPC using the identified model.

The states of the quadcopter model are given by the 3 spatial coordinates (x, y, z) and their velocities, and the

3 angular coordinates (α, β, γ) and their velocities, i.e., the state is $(x, y, z, \dot{x}, \dot{y}, \dot{z}, \alpha, \beta, \gamma, \dot{\alpha}, \dot{\beta}, \dot{\gamma})$. The inputs are given by the thrusts of the 4 rotors, (u_1, u_2, u_3, u_4).

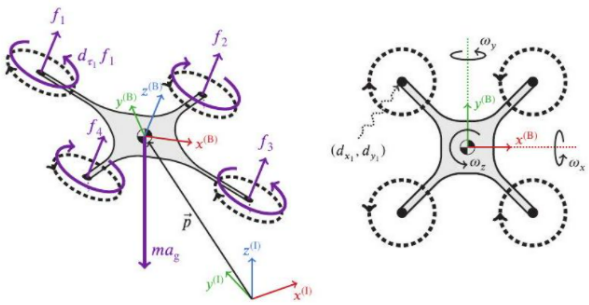

Figure 16. Perspective view (left) and top view (right) of the quadcopter model used for simulation from Elodka et.al [19]

Data were collected from the nonlinear model subjected to an additive white noise measure. They simulated the ID system followed by the MPC algorithm and the regularized DeePC algorithm on a simulation in which the quadcopter was commanded to perform a single-step trajectory in (x, y, z) coordinates. The duration of the constraint violations and their cost were calculated. This was repeated 30 times with different datasets to construct $\mathrm{col}\,(U_\mathrm{p}, Y_\mathrm{p}, U_\mathrm{f}, Y_\mathrm{f})$, and different random seeds for the measurement noise. The results are plotted in the histograms below and show that DeePC consistently outperforms the identification-based MPC in terms of cost (fig 18) and constraint satisfaction (fig 17).

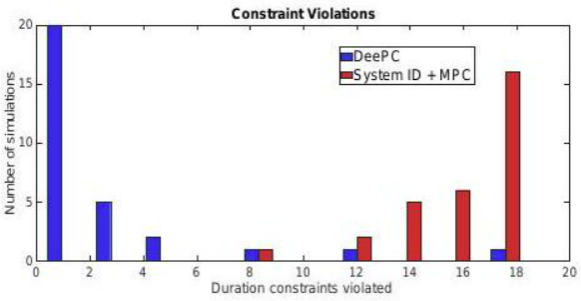

Figure 17. Constraints histogram satisfaction from Coulson et al. ([17])

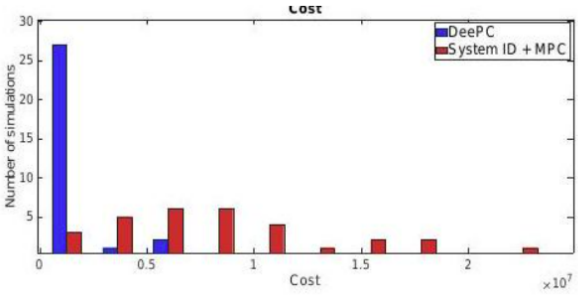

Figure 18. Cost histogram from Coulson et al. ([17])

Power System Oscillation Damping In [20], the DeePC algorithm was applied to VSC-HVDC stations to perform centralized optimal control over a large area in order to mitigate low frequency oscillations and compared to identification using a least-square multi-step prediction error method (PEM) followed by MPC using the idenified model.

In particular, the DeePC algorithm was employed in a VSC-HVDC link considering the dynamic interaction between two VSC-HVDC stations. The four-zone system has weakly damped interzone oscillations due to the fast exciters in the SGs and the long transmission lines. VSC-HVDC station 1 performs active power control to regulate the power flow of the DC link, and VSC-HVDC station 2 performs DC voltage control for the HVDC link. Both VSC-HVDC stations apply phase-locked loops to synchronize with the AC network and voltage control loops to regulate their terminal voltage. In general, the conventional control structures of VSC-HVDC stations, as shown in the figure 19, do not have enough control freedom to realize the oscillation damping functionality, so auxiliary control is required.

The simulations on DeePC and PEM-MPC were repeated 100 times with different data sets to construct the Hankel matrices. The histogram in Figure 20 displays the closed-loop costs from 10 s to 30 s. It shows that the DeePC consistently achieves better closed-loop performance than the PEM-MPC at certainty equivalence. This performance gap is due to the fact that the PEM-MPC uses a nominal model without any robustification. Both algorithms could be further improved, however, for fairness the comparison was between the baseline DeePC and the baseline PEM-MPC.

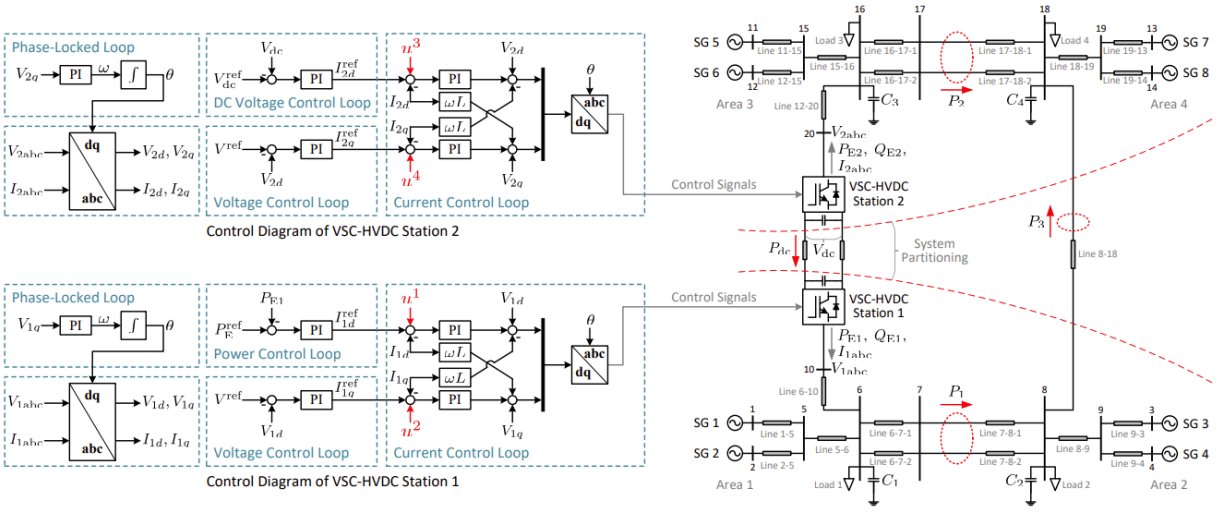

Figure 19. One-line diagram of a four-area test system with integration of an HVDC link from Huang et al. [20]

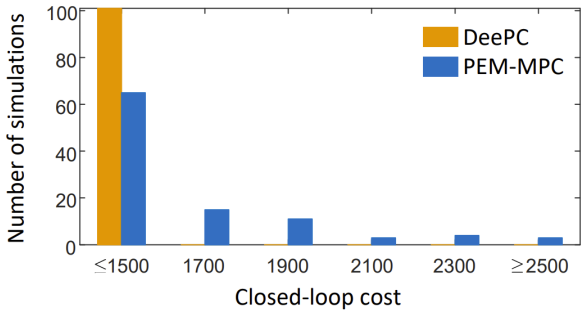

Figure 20. Cost comparison of DeePC and certainty-equivalence PEM-MPC in terms of closed-loop cost from 10s to 30s [20]

2.3.12 Outlook

An important consideration concerns the appropriate circumstances for using DeePC. The ID+MPC approach projects the problem onto an LTI framework and estimates the model in this space. It therefore leads to a smaller variance error if the system is LTI, but suffers from a large bias error if it is not. DeePC provides no projection or denoising, thus the bias error is lower but the variance error is higher in the LTI case (no free lunch, noisier). Consequently, although for a deterministic LTI the both algorithms are equivalent, for a stochastic LTI, ID+MPC is worth considering and for a non-linear system, DeePC is an advantageous alternative.

Finally, the biggest drawbacks with DeePC lies within the case of collecting the data online or having too much data. Indeed, if we have too much data to use, we average the Hankel matrix, but this method strongly exploits the underlying linear structure of the problem (indeed, thanks to the linear structure and the overlay, we can average the Hankel matrices corresponding to different input trajectories and initial conditions). This attempt to exploit additional data to improve the performance of the algorithm when dealing with stochastic systems, without increasing the dimension of the optimization problem, seems to be lacking as more data should have a greater influence in a databased algorithm. Furthermore, for online deployment, DeePC implicitly estimates the same multi-step prediction model at each iteration, which adds significant additional computational cost to its online application.

2.4. Stochastic Optimal Control

The optimal control theory defines the sum of paths of a path cost and end cost. An optimised control sequence and trajectory move the system under investigation to the desired state. In these cases, the dynamics and environment of the system may depend explicitly on time in a finite fixed horizon optimal control system. Limited moving horizons have static dynamics and environments, making it a time-independent optimal control. Infinite horizons with average rewards and absorbing states are optimal control problem states fairly used.

When dealing with Optimal Control problems, some dis-

tinguishing factors should be made. Discrete-time vs continuous time should be identified, discrete state vs continuous state should also be distinguished. Lastly, observable and partially observable control problems can be characterised [21]. The linear quadratic Gaussian control model is a well-studied formulation in stochastic control. The model is linear, the objective function is the expected value of a quadratic form, and the disturbances are additive.

The best control solution for discrete-time centralized systems with additive uncertainty is the same as it would be without the additive disturbances. In relation to autonomous systems, this can be applied to environmental perturbations, unmodeled compliance, ground interactions and battery charge fluctuations [22].

For systems defined by stochastic differential equations, Markov chain approximation numerical methods [23] can be used to compute optimal value functions and control. An approximating process is generated by a controllable finite-state Markov chain, then the Bellman equation is solved to yield an approximating cost and control.

2.4.1 Framework for Autonmous Impedance Regulation of Robots

Researchers have been trying to understand how the human regulates its arm impedance during task performance, taking inspiration from human limb capabilities. [24] proposed a 2D stiffness model with the assumption that end arm stiffness has a linear relation to the magnitude of the joint torques. Achieving autonomous interaction control of robots needs the stiffness profile encoded offline and reproduced online. A more efficient way to get this done is through imitation learning. Here the use of Dynamic Movement Primitives (DMPs)[25] is widely used to encode behaviours in robotics and human motor control scenarios.

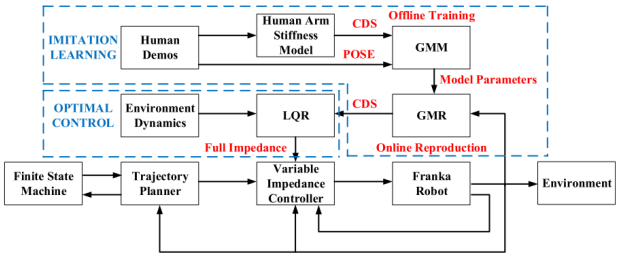

Figure 21. Architecture of impedence controlled by an imitation learning algorithm (GMM/GMR) and an optimal control method (LQR) [26]

Wu et al. proposed a framework that consolidates the benefits of optimal control and Imitation Learning to realize an optimal and adaptive interactive performance having a generalization potential. Despite limited literature on learning dynamics, it is common to see imitation learning is widely used in modelling task kinematics. However, a significant barrier is the challenge of physically meaningful dynamics from human demonstrations.

2.4.2 Optimal Control for Multiple Mobile Robots with Uncertainties [27]

A feedback control system has to be developed for an object to move in the neighbourhood of the calculated optimal program trajectory. This is done with the assumption that the control system reduces motion errors. The numerical model of an object not having a control system differs from the same thing with a control system. A direct approach to optimal control can be used in controlling four mobile robots that work interchangeably without collision. This involves reducing the infinite-dimensional optimization problem of optimal control to the finite-dimensional optimization problem of nonlinear programming.

Alternatively, an indirect method can be used to get this. This is known as the synthesized optimal control that provides an initial solution to the problem by synthesizing a key to stabilizing the system. In the initial step, it is essential to form a multidimensional function with a state vector and a couple of parametric argumentative vectors. The results showed a clear advantage of using synthesized optimal control being deployed under direct supervision. Regardless of the issues, this success was achieved because it occurs less sensitive to existing uncertainties.

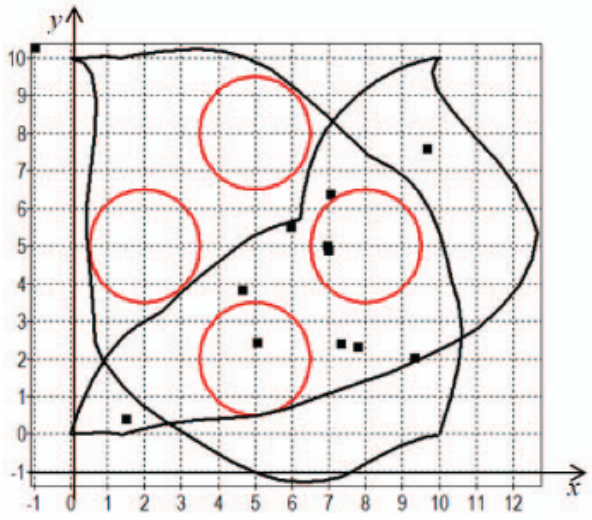

Figure 22. Synthesized approach optimal trajectory[27].

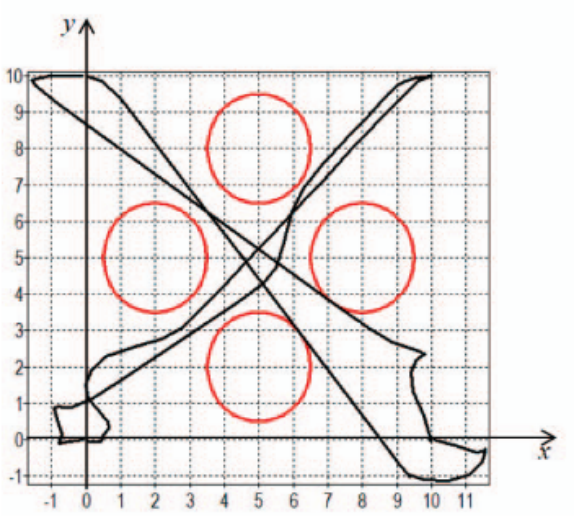

Figure 23. Direct approach optimal trajectory[27].

The consideration issue for a group of four mobile robots was addressed using two ways (synthesized and direct), and the behavior of the object model with the resulting controls was investigated in the presence of errors in the model and in the beginning circumstances, which were added as noise. Specified figures were employed as parameter and constant values, as well as a number f=of stabilization points for each robot, throughout the execution of the tests. In order to solve the challenge, the writers adopted a synthesized technique.

The control synthesis problem was initially handled using the symbolic regression approach of the network operator in order to produce a steady state for each robot. Because the network operator approach was created to solve the synthesis problem, it offers two distinct advantages.The synthesis technique outperforms the direct approach in complicated environments with dynamic and static phase limitations, as shown in the charts (Fig.1-2).

2.4.3 Data-driven Approach to Prediction and Optimal Bucket-filling Control in Autonomous Excavators

In construction, the social and ezcavator bucket relationship is non-linear making physcial modeling a challenge if the excavation steps are being automated. Sandzimier et al. proposed a data-driven statistical model based on measurements and laboratory experiments. This contributed to the development of an optimal control algorithm for switching through the excavation phases with ease.

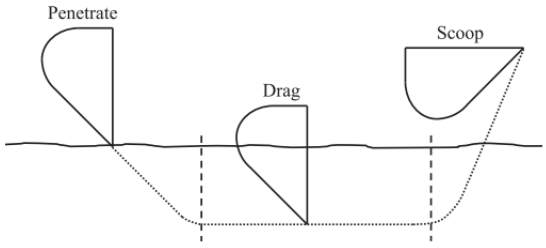

Figure 24. The three phases of an excavation cycle that can be automated are the bukcet penetrating the soil, dragging the soil and scooping the soil up[28].

Experiential operators often fill buckets to 80% capacity in order to reduce cycle time and increase fuel efficiency. Hence, scooping the desired amount of soil is a crucial requirement for automatic excavators.Bernold et al [29] proposed a force feedback and impedance control as effective methods for controlling the path of the bucket as it drags through soil. This work mainly examines how to control the bucket during the drag phase.

2.4.4 Energy- Aware Optimal Control of Variable Stiffness[30].

Variable stiffness actuated (VSA) robots, as well as their generalization to Variable Impedance Actuation (VIA), provide a novel design paradigm aiming primarily at improving safety in physical human-robot interaction and human-level job execution performance. The major problem in energy-aware optimum robotic system control is to propose approaches that can be applied to a wide range of robot topologies while being simple to implement. As a result, this lies in the description of the framework for energy-aware motion planning of VSA robots based on numerical optimum control, which can be designed using state-of-the-art software tools for optimal control and is in principle applicable to any VSA robot topology.

The use of position-controlled servomotors to activate the nonlinear elastic components in VSA robots is studied in [31] [32].This enables for direct analysis and control of the system's energy usage, but also necessitates changes to the previously considered models.

To develop distinct sorts of OCPs, a state-space model of the total system will be used.VSA systems, like other passive or quasi-passive techniques, are quite likely to help robots save energy.The issues raised have to do with energy usage during job execution. P(t), the robot's power consumption at a particular time instant t, is calculated by adding the power consumption of all motors, each of which is affected by both control inputs (motor currents) and sta-

tus variables.The purpose of the second type of OCP is to reduce the amount of energy used to complete the activity while maintaining an acceptable target performance.

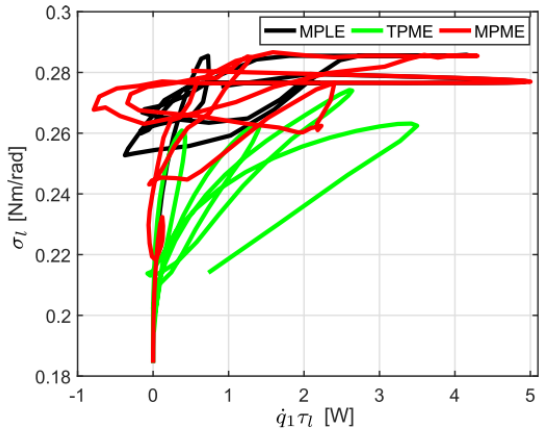

Figure 25. The variation of system stiffness as a function of Mechanical Power[30].

A direct numerical technique was used to solve the specified OCPs. The time period T was uniformly split into intervals using the ACADOToolkit [33], and the temporal evolution of the problem variables was discretized using a fourth-order Runge-Kutta integrator with a timestep. The resultant discrete-time OCP is then solved using a simultaneous method yielding a nonlinear program that is subsequently solved using the active-set QP solver qpOASES [34].

In Fig: 25, three scenarios of mechanical power are illustrated. As a function of time, the variation in joint stiffness(MPLE, TPME, MPME). TPME is for d = 2m and T = 2 s, whereas MPME is = 0.04 m. The findings show that minimizing energy consumption involves robot operation in low stiffness mode (TMPE), but achieving goal performance required high joint stiffness.

2.4.5 State Suprema Optimal COntrol in Differential Systems

This paper[35] investigates an efficient and practical algorithm based on the motion of a car-like robot using curves and straight lines to find junction points, then using optimal control theory to generate the desired optimal path vector points using only the robot's initial and final positions, and finally a non-linear control strategy based on feedback linearization to verify that the robot can follow this optimal path using positive and negative velocity

Remember that differential equations with the sup-operator on the right hand sides of the FDEs imply delayed differential equations. The state-dependent delays in the resultant dynamic models are sophisticated (implicit). In the presence of state-dependent delays, the resulting mathematical procedures are insufficiently advanced to OCPs linked with differential equations. The same result holds true for OCPs in systems that evolve with state suprema.

3. Conclusion

3.1. Conclusion for LQR

LQR has become an important method in optimal control problems. The LQR controller is easy to design and only two matrices are enough to achieve steady-state control. Even for nonlinear systems, it can be solved by designing a controller with local linearization. The parameter selection, as described above, can be obtained automated by combining it with advanced algorithms. As a basic control method, LQR is essentially to design a feedback gain matrix K. In addition to the realization of the LQR controller, it is a major research direction to combine it with other methods. For example, efficient control can be implemented by combining LQR with the MPC (Model predictive control) method and Neural Network [36]. And a data-driven LQR controller, which means combining LQR with Markov methods, is also popular [37].

3.2. Conclusion and Outlook for MPC

MPC can incorporate more precise dynamics and constraints and be applied in real-time applications because the environment is dynamic and stochastic. With the successes in the field of machine learning and neural networks, the combination or integration with MPC and learning method such as Reinforcement Learning has received substantial. interest in recent years. Like the previous paper from Chen et.al [13], the learning method can be implemented in model describing. It also can be implemented into controller design since the controller also has a lot of parameters such as penalty parameters.

3.3. Conclusion for DeePC

In the section 2.3, a data-driven algorithm [15] that can be applied to unknown LTI systems has been presented and its equivalence to the classical MPC algorithm has been explicitly stated. This algorithm uses a finite set of data to learn the behavior of an unknown system. For example, it can compute optimal commands using real-time performance feedback to guide a system along a desired trajectory while respecting system constraints. In addition, a regularized version of the algorithm was introduced and successfully rivaled the stochastic nonlinear dynamics of the quad-

copter, illustrating its capabilities beyond deterministic LTI systems. The performance was superior to that of the ID system followed by the MPC.[16][17] We then introduced an extension of the data-based predictive control algorithm to handle control problems on unknown stochastic LTI systems, using additional data without increasing the dimension of the optimization problem to be solved at each iteration via offline averaging of multiple Hankel matrices to obtain a cleaner data-based model.The performance of the proposed method has been experimentally validated on a stochastic LTI system showing improvements over the standard DeePC [18]. We then mentioned an application of the regularized DeePC algorithm that shows that it is suitable for real-time control of a real-world quadcopter, thus bridging the gap between theory and practice. Through this real-world implementation, it has been shown that the DeePC algorithm is computable and can be adequately solved in real time, with resolution times well below the requirements of real time [17] [19]. We also mentioned another application of the DeePC algorithm as a model-free approach to perform large-scale optimal control based solely on the measured input/output trajectories of the unknown system to predict future behaviors. In the stated context of power systems, DeePC leverages the high control capability and flexibility of VSCHVDC stations to mitigate low frequency oscillations. [20]

DeePC, unlike MPC, is very novel and each improvement is showing some promising results, therefore it is a subfield that warrants great consideration.

3.4. Conclusion for Stochastic Optimal Control

Designing stable systems that meet task needs is a basic requirement of autonomous robotics. The contributions in Section 2.4 studied Stochastic Optimal Control with associated problems. These solutions can be applied to societal challenges in smart mobility, healthcare, manufacturing and many more.

The effectiveness of all proposed solutions and frameworks reviewed has been evaluated by conducting comparative experiments and making sound assumptions based on scanned literature. Results provide evidence of the importance of the superior performance of stochastic Optimal control strategies in autonomous systems in generating desired trajectories, efficient execution of tasks and proper utilization of system resources.

4. Contributions

Xingyu Song: Chapter 1 + Section 2.1 + Section 3.1
Zehua Zhang: Section 2.2 + Section 3.2
Hanady Gebran: Section 2.3 + Section 3.3
Yaw Obeng Okofo Dartey: Section 2.4 + Section 3.4

References

[1] Donald E Kirk. *Optimal control theory: an introduction*. Courier Corporation, 2004. 1

[2] Michael Athans and Peter L Falb. *Optimal control: an introduction to the theory and its applications*. Courier Corporation, 2013. 1

[3] Katsuhiko Ogata et al. *Modern control engineering*, volume 5. Prentice hall Upper Saddle River, NJ, 2010. 2.1.1, 2.1.1, 2.1.1, 2.1.2

[4] RM Murray. Control and dynamical systems. 2006. 2.1.1

[5] Saeed Abdolshah and Erfan Shojaei Barjuei. Linear quadratic optimal controller for cable-driven parallel robots. *Frontiers of Mechanical Engineering*, 10(4):344–351, 2015. 2.1.2, 2, 2.1.2, 3

[6] Paolo Gallina and Giulio Rosati. Manipulability of a planar wire driven haptic device. *Mechanism and Machine Theory*, 37(2):215–228, 2002. 2.1.2

[7] Saber Kazeminasab, Roozbeh Jafari, and M Katherine Banks. An lqr-assisted control algorithm for an underactuated in-pipe robot in water distribution systems. In *Proceedings of the 36th Annual ACM Symposium on Applied Computing*, pages 811–814, 2021. 2.1.2, 4, 2.1.2, 2.1.2, 5, 6

[8] A Cengel Yunus. *Fluid Mechanics: Fundamentals And Applications (Si Units)*. Tata McGraw Hill Education Private Limited, 2010. 2.1.2

[9] Suppachai Howimanporn, Sasithorn Chookaew, and Warin Sootkaneung. Implementation of pso based gain-scheduled pid and lqr for dc motor control using plc and scada. In *2018 International Conference on Control and Robots (ICCR)*, pages 52–56. IEEE, 2018. 2.1.2, 7, 2.1.2, 2.1.2, 2.1.2, 8

[10] James Kennedy and Russell Eberhart. Particle swarm optimization. In *Proceedings of ICNN'95-international conference on neural networks*, volume 4, pages 1942–1948. IEEE, 1995. 2.1.2

[11] Bhaskar Varma, Nitin Swamy, and Sujoy Mukherjee. Trajectory tracking of autonomous vehicles using different control techniques (pid vs lqr vs mpc). In *2020 International Conference on Smart Technologies in Computing, Electrical and Electronics (ICSTCEE)*, pages 84–89. IEEE, 2020. 9, 2.2.2, 2.2.2, 2.2.3, 13, 14

[12] Yaser Alothman and Dongbing Gu. Using constrained nmpc to control a cable-suspended payload with two quadrotors. In *2018 24th International Conference on Automation and Computing (ICAC)*, pages 1–6. IEEE, 2018. 2.2.2, 2.2.2, 2.2.2

[13] Yize Chen, Yuanyuan Shi, and Baosen Zhang. Optimal control via neural networks: A convex approach. *arXiv preprint arXiv:1805.11835*, 2018. 2.2.2, 2.2.4, 15, 3.2

[14] Jianyu Chen, Wei Zhan, and Masayoshi Tomizuka. Autonomous driving motion planning with constrained iterative lqr. *IEEE Transactions on Intelligent Vehicles*, 4(2):244–254, 2019. 2.2.2, 2.2.2, 2.2.2, 2.2.2

[15] Zhong-Sheng Hou and Zhuo Wang. From model-based control to data-driven control: Survey, classification and perspective. *Information Sciences*, 235:3–35, 2013. 2.3.1, 3.3

[16] Henk J van Waarde, Claudio De Persis, M Kanat Camlibel, and Pietro Tesi. Willems fundamental lemma for state-space systems and its extension to multiple datasets. *IEEE Control Systems Letters*, 4(3):602–607, 2020. 2.3.2, 2.3.4, 3.3

[17] Jeremy Coulson, John Lygeros, and Florian Dörfler. Data-enabled predictive control: In the shallows of the deepc. In *2019 18th European Control Conference (ECC)*, pages 307–312. IEEE, 2019. 2.3.2, 2.3.3, 2.3.3, 2.3.4, 2.3.5, 2.3.6, 2.3.7, 2.3.8, 2.3.9, 2.3.10, 2.3.11, 2.3.11, 17, 18, 3.3

[18] Daniele Alpago, Florian Dörfler, and John Lygeros. An extended kalman filter for data-enabled predictive control. *IEEE Control Systems Letters*, 4(4):994–999, 2020. 2.3.2, 2.3.10, 3.3

[19] Ezzat Elokda, Jeremy Coulson, Paul N Beuchat, John Lygeros, and Florian Dörfler. Data-enabled predictive control for quadcopters. *International Journal of Robust and Nonlinear Control*, 31(18):8916–8936, 2021. 2.3.2, 2.3.3, 2.3.7, 2.3.8, 2.3.11, 2.3.11, 16, 3.3

[20] Linbin Huang, Jeremy Coulson, John Lygeros, and Florian Dörfler. Decentralized data-enabled predictive control for power system oscillation damping. *IEEE Transactions on Control Systems Technology*, 2021. 2.3.2, 2.3.11, 2.3.11, 19, 20, 3.3

[21] Wendell H Fleming and Raymond W Rishel. *Deterministic and stochastic optimal control*, volume 1. Springer Science & Business Media, 2012. 2.4

[22] Shridhar K Shah, Herbert G Tanner, and Chetan D Pahlajani. Optimal navigation for vehicles with stochastic dynamics. *IEEE Transactions on Control Systems Technology*, 23(5):2003–2009, 2015. 2.4

[23] Harold Joseph Kushner Kushner, Harold J Kushner, Paul G Dupuis, and Paul Dupuis. *Numerical methods for stochastic control problems in continuous time*, volume 24. Springer Science & Business Media, 2001. 2.4

[24] Keng Peng Tee, Etienne Burdet, Chee-Meng Chew, and Theodore E Milner. A model of force and impedance in human arm movements. *Biological cybernetics*, 90(5):368–375, 2004. 2.4.1

[25] M Dıaz-Rodrıguez, V Mata, Á Valera, and Á Page. Dynamical movement primitives: learning attractor models for motor behaviors. *MECH MACH THEORY*, 45(9):1337–1356, 2010. 2.4.1

[26] Yuqiang Wu, Fei Zhao, Tao Tao, and Arash Ajoudani. A framework for autonomous impedance regulation of robots based on imitation learning and optimal control. *IEEE Robotics and Automation Letters*, 6(1):127–134, 2020. 21

[27] Askhat Diveev and Elizaveta Shmalko. Optimal control design for a group of mobile robots with uncertainties. In *2020 15th IEEE Conference on Industrial Electronics and Applications (ICIEA)*, pages 308–313. IEEE, 2020. 2.4.2, 22, 23

[28] Ryan J Sandzimier and H Harry Asada. A data-driven approach to prediction and optimal bucket-filling control for autonomous excavators. *IEEE Robotics and Automation Letters*, 5(2):2682–2689, 2020. 24

[29] Leonhard E Bernold. Motion and path control for robotic excavation. *Journal of Aerospace Engineering*, 6(1):1–18, 1993. 2.4.3

[30] Altay Zhakatayev, Matteo Rubagotti, and Huseyin Atakan Varol. Energy-aware optimal control of variable stiffness actuated robots. *IEEE Robotics and Automation Letters*, 4(2):330–337, 2019. 2.4.4, 25

[31] David J Braun, Florian Petit, Felix Huber, Sami Haddadin, Patrick Van Der Smagt, Alin Albu-Schäffer, and Sethu Vijayakumar. Robots driven by compliant actuators: Optimal control under actuation constraints. *IEEE Transactions on Robotics*, 29(5):1085–1101, 2013. 2.4.4

[32] Altay Zhakatayev, Matteo Rubagotti, and Huseyin Atakan Varol. Closed-loop control of variable stiffness actuated robots via nonlinear model predictive control. *IEEE Access*, 3:235–248, 2015. 2.4.4

[33] Boris Houska, Hans Joachim Ferreau, and Moritz Diehl. Acado toolkitan open-source framework for automatic control and dynamic optimization. *Optimal Control Applications and Methods*, 32(3):298–312, 2011. 2.4.4

[34] Hans Joachim Ferreau, Christian Kirches, Andreas Potschka, Hans Georg Bock, and Moritz Diehl. qpoases: A parametric active-set algorithm for quadratic programming. *Mathematical Programming Computation*, 6(4):327–363, 2014. 2.4.4

[35] Erik I Verriest and Vadim Azhmyakov. Advances in optimal control of differential systems with the state suprema. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 739–744. IEEE, 2017. 2.4.5

[36] Fangyu Wu, Guanhua Wang, Siyuan Zhuang, Kehan Wang, Alexander Keimer, Ion Stoica, and Alexandre Bayen. Composing mpc with lqr and neural networks for efficient and stable control. *arXiv preprint arXiv:2112.07238*, 2021. 3.1

[37] Gustavo R Gonçalves da Silva, Alexandre S Bazanella, Charles Lorenzini, and Luciola Campestrini. Data-driven lqr control design. *IEEE control systems letters*, 3(1):180–185, 2018. 3.1