# Model-based and model-free learning strategies for wet clutch control

Abhishek Dutta*[a], Yu Zhong[a], Bruno Depraetere[b], Kevin Van Vaerenbergh[d], Clara Ionescu[a], Bart Wyns[a], Gregory Pinte[b], Ann Nowe[d], Jan Swevers[c], Robin De Keyser[a]

[a]*Electrical Energy, Systems and Automation, Ghent University, Sint-Pietersnieuwst. 41 Block B2, 9000 Gent, Belgium*
*Corresponding author* email: Dutta.Abhishek@UGent.be*
[b]*Flanders' Mechatronics Technology Center, Celestijnenlaan 300D, Leuven 3001, Belgium*
[c]*Department of Mechanical Engineering, Celestijnenlaan 300D, Leuven 3001, Belgium*
[d]*AI Lab, Vrije Universiteit Brussel, Pleinlaan 2, B-1050 Brussels, Belgium*

**Abstract**

This paper presents an overview of model-based (Nonlinear Model Predictive Control, Iterative Learning Control and Iterative Optimization) and model-free (Genetic-based Machine Learning and Reinforcement Learning) learning strategies for the control of wet-clutches. The benefits and drawbacks of the different methodologies are discussed, and illustrated by an experimental validation on a test bench containing wet-clutches. In general, all strategies yield a good engagement quality once they converge. The model-based strategies seems most suited for an online application, because they are inherently more robust and require a shorter convergence time. The model-free strategies meanwhile seem most suited to offline calibration procedures for complex systems where heuristic tuning rules no longer suffice.

*Keywords:* nonlinear predictive control, iterative learning control, genetic algorithm, reinforcement learning, hydraulic clutch transmission

## 1. Introduction

Wet clutches are commonly used in automatic transmissions for off-highway vehicles and agricultural machines to transfer torque from the engine to the load. By disengaging one clutch and engaging another, different transmission ratios can be realized. When a clutch engagement is requested, an operator expects a fast response without vibrations. The torque transfer should thus begin as soon as possible without introducing torque discontinuities and peaks. These machines are operated through several years and under varying environmental conditions such that clutches undergo significant amount of wear and tear, thereby making the clutch control a challenging industrial problem [1]. Contrary to wet-clutches, modeling and control of dry-clutches has received considerable attention in research, often considering a stick-slip hybrid model for analysis. A slip control using linear quadratic regulator with force on clutch piston as input is developed in [2]. While [3] concluded that an online MPC scheme for clutch control is not practically implementable due to the high computation costs, an explicit Model Predictive Control is derived in [4], using a linear cost function for slip control, amongst others. The representative work on wet-clutch includes optimal control of automotive transmission clutch filling [5], PID control for a wet plate clutch actuated by a pressure reducing valve [6], predictive control of a two stage actuation system using piezoelectric actuators for controllable industrial clutches [7], predictive control of an electro-hydraulic actuated wet-clutch for automatic transmission [8] and fast and smooth clutch engagement control for dual-clutch transmissions [9].

The two main challenges for wet clutch control are (i) the intrinsic complex, non-linear behavior [10], and (ii) the variation of these dynamics over time due to changes in load, oil temperature and wear [11]. When similar or repetitive operations have to be carried out, e.g. the successive engagements of a clutch, learning can be introduced to address these issues. By gradually improving the performance with respect to the previous trial, the complex system behavior can be learned at the cost of a convergence period, and it also becomes possible to automatically adapt to variations in the system's behaviour or operating conditions.

In this paper, the potential of several model-based (Nonlinear Model Predictive Control (NMPC), Iterative Learning Control (ILC) and Iterative Optimization(IO)) and model-free (Genetic-based Machine Learning (GA) and Reinforcement Learning(RL)) learning strategies are analyzed for the control of a wet clutch engagement. The model-based approaches rely on a model of the clutch dynamics to update the control signals at each engagement, while in contrast, the model-free ones omit this model and directly explore the input space of possible clutch control signals using a guided trial-and-error procedure, attempting to maximize the reward/fitness.

The remaining content of this paper is laid out as follows. Section 2 briefly describes the wet-clutch dynamics and objectives. Sections 3 and 4 introduce the model-based and model-free learning techniques respectively, and illustrate their application to wet clutch control. Section 5 details the experimental results followed by a comparison of their benefits and drawbacks in section 6. Section 7 finally concludes the paper.

## 2. The Wet-clutch

A wet clutch is a device which is used to transmit torque from one shaft to another by means of friction force. As illustrated in Fig. 1, it contains two sets of friction plates, one that can slide in grooves on the inside of the drum, and another that can slide in grooves on the outgoing shaft. Torque can be transferred between the shafts by pressing both sets together with a hydraulic piston, which can be realized by sending an appropriate control signal to the servovalve in the line to the clutch. Initially, during the filling phase, the clutch chamber fills up with oil and the pressure builds up, until it is high enough to compress the return spring and accelerate the piston. When the piston advances far enough and presses the plates together, the filling phase ends and the slip phase begins. During the slip phase, torque is transferred, so that the difference in the angular speeds between the shafts starts to change. This difference in angular speeds is called the slip speed, and will be shortened to slip in the remainder. This slip decreases until both shafts have the same rotation speed. A dynamic model of a hydraulic multi-plate clutch actuator controlled by an electro valve with internal pressure feedback [12] or a model based on power oriented graphs [13] have been reported in literature. However, it has been argued that, it is frequently unfeasible to transfer these models to other applications because of major modifications that would be needed [8]. Building on this argument, the work in this paper either uses simple system identified models or model free control approaches.

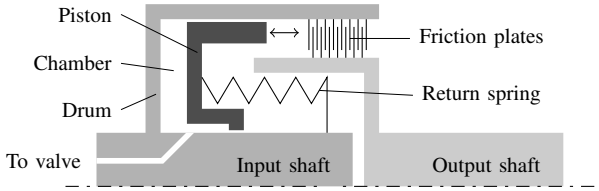


Figure 1: Schematic overview of a wet-clutch and its main components.

So far, the goals for the control were not strictly defined. In general, we want both a fast and smooth engagement. As a measure for this smoothness, we use the highest absolute value of the second derivative of the slip (the jerk), since it is strongly related to the experienced operator comfort [14]. For a given engagement duration, we then want to find the control yielding the lowest absolute value of jerk. This can be realized by a short filling phase (without torque transfer) followed by a smooth transition into the slip phase (buildup of torque), after which the load has to be synchronized further, still in a smooth manner (significant torque transfer).

To validate the developments an experimental setup is used, where an electromotor (30 $kW$) drives a flywheel (2.5 $kgm^2$) via a torque converter and two mechanical transmissions, as shown in Fig. 2. The controllers are applied to the first range clutch of the left transmission while the right transmission is used only to vary the load observed by the first transmission and to apply an adjustable braking torque. The controlled transmission is equipped with sensors measuring the speeds of the different

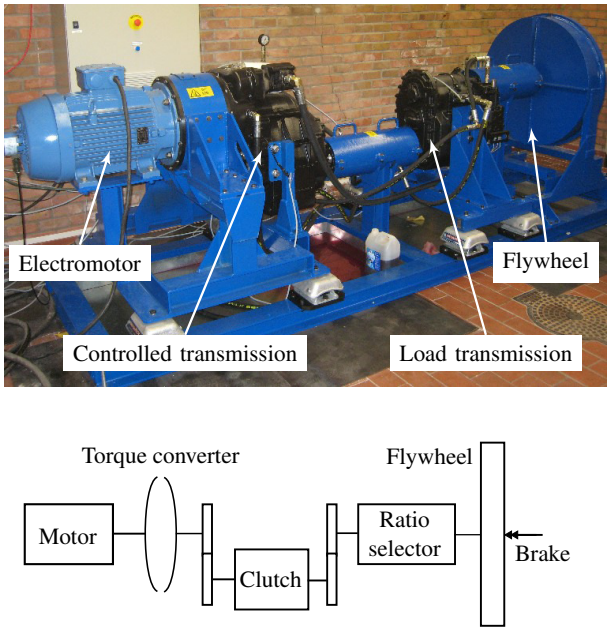


Figure 2: Experimental setup with wet clutches.

shafts and the pressure of the oil in the line to the clutches. An additional torque sensor is installed to illustrate the performance, but it is not used for the control itself. All experiments are performed with a fixed engine speed, while the output starts at standstill and is then accelerated by engaging the clutch for first gear in the controlled transmission. The initial conditions are zero current and atmospheric pressure. A dSPACE 1103 board is used to control the setup. The entire wet-clutch dynamics is subjected to the following physical constraints:

$$\begin{aligned} 0 \leq Current \ \ (Amps) \leq 0.8 \\ 0 \leq Pressure \ \ (Bars) \leq 14 \\ 0 \leq Slip \ \ (normalized) \leq 1 \end{aligned} \tag{1}$$

Clearly, the outlined goals are qualitative and therefore to realize them as parametric trajectories, an element of learning is necessary for the control of wet-clutches. This motivates us to integrate learning in model-based controllers or to develop completely model-free learning strategies.

## 3. Model-Based Learning Control

This section discusses three model-based learning techniques for wet clutch control. A two-level learning control scheme based on NMPC is presented first, followed by a similar two-level control scheme using ILC instead of MPC. Afterwards, the IO technique is presented as an alternative.

### *3.1. Two-level NMPC (2l-NMPC)*

For the wet clutch with its nonlinear transitions between two phases, it is difficult to develop a single performant control algorithm. We therefore propose to use separate controllers for

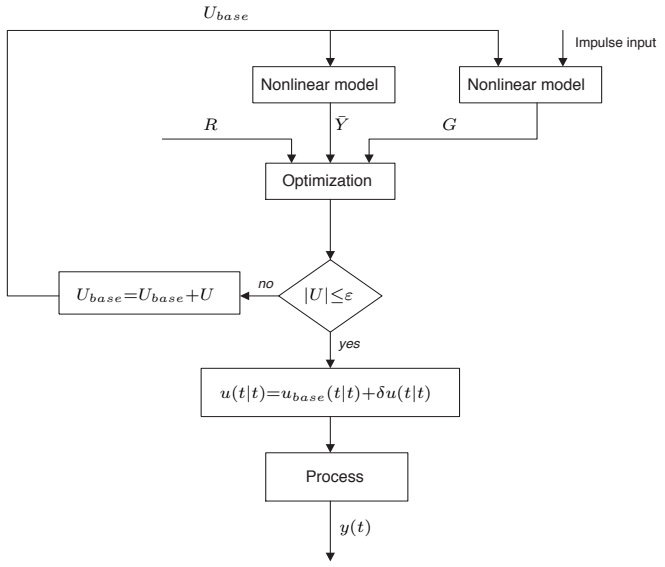


Figure 3: NEPSAC algorithm flowchart, where $R, \bar{Y}, G, U_{base}$ are the reference base output, step response matrix, base input and $u, \delta u, y$ are the input, increment and output respectively.

each phase. This simplifies the control design, but also the identification, since a model for each phase separately is sufficient instead of a global model. To further reduce the complexity, we only consider tracking controllers. For the clutch, we then have a first controller aiming to track a pressure reference in the filling phase, which is deactivated once the slip phase begins, at which point a second controller is activated to track a slip reference.

MPC is a form of control in which the current control action is obtained by solving on-line, during each sampling period, a finite horizon open-loop optimal control problem taking into account the various constraints on the system [15]. Over the last three decades MPC has occupied the center stage in the control research community and had a tremendous impact on the advances in process industry as well [16].

The (Nonlinear) Extended Prediction Self-Adaptive Control i.e. (N)EPSAC [17], (N)MPC principle is depicted in Fig. 3. The process is modeled as in [18]:

$$y(t) = x(t) + n(t) \tag{2}$$

with $y(t), x(t), n(t)$ as process output, model output, process/model disturbance respectively. The fundamental step is based on the output prediction using the process model given by:

$$y(t+k|t) = x(t+k|t) + n(t+k|t) \tag{3}$$

where $y(t+k|t)$ is the prediction of process output after $k$ samples made at time instant $t$, over the prediction horizon from $N_1$ to $N_2$, based on prior measurements and postulated values of inputs. Prediction of model output $x(t+k|t)$ and of colored noise process $n(t+k|t)$ can be obtained by the recursion of process model and filtering techniques, respectively. The future response can be expressed as:

$$y(t+k|t) = y_{base}(t+k|t) + y_{optimize}(t+k|t) \tag{4}$$

where the two contributing terms have the following origins:

- $y_{base}(t+k|t)$ is the cumulative effect of past control inputs, the apriori defined future control actions $u_{base}(t+k|t)$ and the predicted disturbances. To predict these disturbances, $n(t) = C(q^{-1})/D(q^{-1}).e(t)$ is used, with $e(t)$ white noise, and the filter $C/D$ is often chosen as an integrator to ensure zero steady state error and $(q^{-1})$ is the backward shift operator.

- $y_{optimize}(t+k|t)$ is the effect of the additions $\delta u(t+k|t)$ that are optimized and added to $u_{base}(t+k|t)$, according to $\delta u(t+k|t) = u(t+k|t) - u_{base}(t+k|t)$. The effect of these additions is the discrete time convolution of $\Delta U = \{\delta u(t|t), \ldots \delta u(t+N_u-1|t)\}$ with the impulse response coefficients of the system ($G$ matrix), where $N_u$ is the chosen control horizon.

The control $\Delta U$ is the solution to the following constrained optimization problem:

$$min_{\Delta U}\{V = \Sigma_{k=N_1}^{N_2}[r(t+k|t) - y(t+k|t)]^2 + \lambda\Sigma_{k=0}^{N_u-1}[\delta u(t+k|t)]^2\}$$
$$subject\ \ to \quad M.\Delta U \leq N \tag{5}$$

where the first term in $V$ aims to achieve a good tracking of the reference $r(t+k|t)$, while the second term aims to reduce the control effort, and the weighting factor $\lambda$ selecting their relative importance. The various input and output constraints can all be expressed in terms of $\Delta U$, resulting in the matrices $M, N$ and is solved online by active-sets based primal-dual optimization [19].

When a nonlinear system $f[.]$ is used for $x(t)$, the superposition of (4) is valid only if the term $y_{optimize}(t+k|t)$ is small enough compared to $y_{base}(t+k|t)$. This is true when $\delta u(t+k|t)$ is small, which is the case if $u_{base}(t+k|t)$ is close to the optimal $u^*(t+k|t)$. To address this condition, the idea is to recursively compute $\delta u(t+k|t)$, within the same sampling instant, until $\delta u(t+k|t)$ converges to 0. Inside the recursion $u_{base}(t+k|t)$ is updated each time to $u_{base}(t+k|t) + \delta u(t+k|t)$. This is illustrated in Fig. 3, where $R, \bar{Y}, U_{base}$ are now in the vector form of the signals $r, y_{base}, u_{base}$ introduced before.

A third-order linear input-output model for the filling phase (from current to pressure) sampled at 1ms and a fourth-order polynomial nonlinear state-space (PNLSS) model containing terms in powers of states and input for the slip phase (from current to slip) sampled at 10ms have been identified [20]. An MPC controller with $N_1 = 2, N_u = 1, N_2 = 10, \lambda = 0, \frac{C}{D} = 1/(1-q^{-1})$ is used to obtain a mean-level control in the fill phase. The short control-horizon ensures that the optimization is tractable within the allowed 1ms sampling time. The NMPC controller is designed with parameters $N_1 = 1, N_u = 4, N_2 = 5, \lambda = O(10^2), C/D = 1/(1-q^{-1})$ for slip control. The chosen combination of control horizon and control penalty gives the controller enough degrees of freedom for tracking and simultaneously ensures smooth control action. The slower sampling time of 10ms allows sufficient time for the NEPSAC iterations (less than 5) to converge. The prediction horizons for both these controllers are chosen to ensure feasibility and stability as they are subjected to polytopic input and output constraints (1).

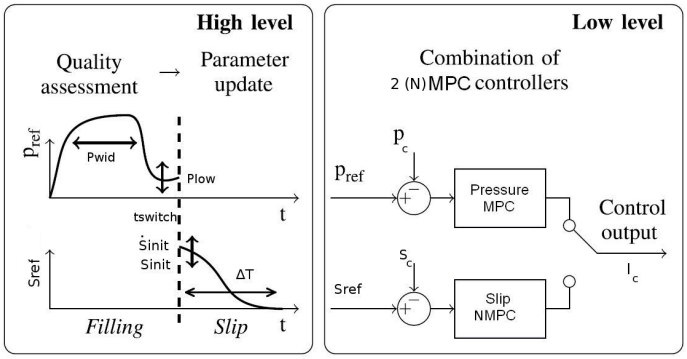


Figure 4: A schematic illustration of the proposed two-level control scheme, where $p_c, p_{ref}, p_{wid}, p_{low}$ are the clutch pressure, reference pressure, high pressure width, low pressure value respectively and $s_c, s_{ref}, s_{init}, \dot{s}_{init}$ are the measured, reference, initial, derivative of initial values of slip respectively with $t_{switch}, \Delta T, I_c$ denoting switching time,slip interval, input current respectively

It is clear that an iterative procedure is required to solve the optimization problem with inequality constraints, because we did not know which constraints would become active constraints. The maximum number of constraints that can be active equals the number of decision variables. If there are many constraints, the computational load is quite large. Since, we work with shorter control, prediction horizons and impose convex polytopic constraints on inputs and outputs separately, even in the worst case the computation times for the linear and nonlinear MPC are well within the sampling intervals. Note that computational cost can be further reduced by checking for and removing redundant constraints. Further details on the control design can be found in [21].

The remaining difficulty is the generation of good references profiles, since the specifications for a good engagement do not allow to easily formulate optimal references. To address this issue, we use two-level control scheme as illustrated in Fig. 4. On the low level, a classical tracking-based NMPC controller is used, while at the high level, ILC-type learning algorithms are added that learn the parameters of parameterized references, aiming to translate the original non-tracking problem into a tracking problem. The goal is to learn the parameters of these references based on the observed system behaviour, such that they eventually lead to the desired engagements. To achieve this, these high-level laws further also have to ensure a smooth transition between the two controllers, and to compensate to changes in the operating conditions.

To define the high-level update laws, we start with process knowledge to select a profile or procedure that allows to construct the reference trajectories from a few discrete parameters, say $w_i(k)$, with $i$ the variable index and $k$ the iteration number. These parameters are then related to some observable performance indices $PI_i^{est}$, for which we also define a target value $PI_i^d$. We then choose an initial set of parameters, evaluate the resulting performance to obtain $PI_i^{est}$, and calculate a parameter update based on the difference between the desired and the measured indices as follows:

$$w_i(k+1) = w_i(k) + \eta_i \cdot (PI_i^d(k) - PI_i^{est}(k)), \tag{6}$$

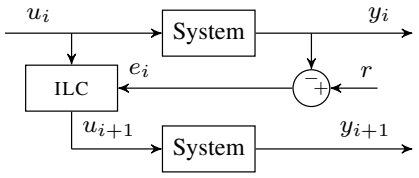


Figure 5: Schematic representation of an ILC controller; first, during trial $i$, the system to be controlled is excited using $u_i$, then after completion of this trial, $u_i$ is used along with the measured tracking error $e_i = r - y_i$ to find the excitation $u_{i+1}$ to be used during trial $i+1$.

where $\eta_i$ is a gain depending on the specific quantities involved. This process is repeated until the parameters of the references converge to the desired optimal values. For the clutch, the reference profiles are shown in (Fig. 4), where we can see that the profile for the pressure $P_{ref}$ has two two parameters, the duration of the high pressure phase $p_{wid} = t_{switch} - 50$ ms and the pressure at the end of the reference, $p_{low}$. These are both updated based on estimates on the time instants at which torque starts to be transferred, since we ideally want this to occur just after the high pressure pulse (to avoid high initial torque spikes), and we want to gradually build up torque. Thus $P_{ref}$ accelerates the piston for fast engagement and at the same time prepares for smooth engagement. For the slip reference, we learn the two initial values $s_{init}$ and $\dot{s}_{init}$, based on the observed values at the end of the filling phase. Once these are specified the remainder of the slip reference is calculated analytically by finding the trajectory of a given duration $\Delta T$ that minimizes the highest absolute value of the jerk, given the initial conditions above and terminal conditions $s_{ref}(\Delta T) = \dot{s}_{ref}(\Delta T) = 0$ [22].

### *3.2. Two-Level ILC (2l-ILC)*

In this section, a similar approach is used as in the last two-level NMPC, but the low-level controllers are now ILC instead of NMPC controllers. The reason for doing so it that ILC is itself a learning control technique, unlike NMPC, which uses experience gained during previous iterations to improve the tracking performance for repetitive systems [23, 24]. An NMPC thus requires an accurate model to be able to obtain a good tracking performance, whereas an ILC algorithm can, due to its learning strategy, realize a good tracking performance even when there is a large model uncertainty due to its learning behaviour. The downside of this is that we will not be able to update the reference profile parameters after each trial, since we cannot yet judge the parameter's quality since the ILC controller has not yet learned to track the profile closely. Instead, we wait for 5 trials now each time the parameters are updated, allowing the ILC to converge before we calculate the performance indices and update the reference parameters.

Fig. 5 shows a first order ILC control scheme, as is used in this paper. Here, $y$ is the output of the plant and $r$ is a reference trajectory. The ILC control signal for the $(i+1)^{th}$ iteration, $u_{i+1}$, is calculated based on the previous ILC control signal, $u_i$ and the previous tracking error $e_i$. We use a linear update law such that

$$u_{i+1}(k) = Q(q^{-1})\Big(u_i(k) + L(q^{-1})e_i(k)\Big), \tag{7}$$

with linear operators $Q$ and $L$ that can be chosen during the design of the ILC controller. For this update law, a convenient frequency domain criterion for monotonic convergence can be derived [23, 24]. For a plant with FRF $P(\omega)$, a monotonically decreasing tracking error is obtained with controller (7) if

$$|Q(j\omega)\Big(1 - L(j\omega)P(j\omega)\Big)| < 1, \tag{8}$$

with $Q(j\omega)$ and $L(j\omega)$ the FRF's of the operators $Q$ and $L$. It is also possible to derive an expression for the remaining error after convergence, $E_\infty(j\omega)$, which becomes

$$E_\infty(j\omega) = \frac{1 - Q(j\omega)}{1 - Q(j\omega)\Big(1 - L(j\omega)P(j\omega)\Big)} R(j\omega), \tag{9}$$

where $R(\omega)$ is the Fourier transform of the reference $r$.

Based on these expressions, [23] and [24] show that by selecting $L(j\omega) = P(j\omega)^{-1}$ and $Q(j\omega) = 1$, perfect tracking would be obtained after only one iteration. However when there is uncertainty about $P(j\omega)$, this choice of $L(j\omega)$ becomes impossible. It is then needed to select an estimate $\hat{P}(j\omega)$ of the plant and use $L(j\omega) = \alpha\hat{P}(j\omega)^{-1}$ with $0 < \alpha < 1$. This way, the robustness increases while the learning slows down, but a good performance is still achieved. This is possible for all frequencies where the angular deviation between the system and the nominal model does not exceed 90°. Once this deviation becomes larger, the value of $|Q(j\omega)|$ has to be decreased in order to satisfy (8). It then follows from (9) that perfect tracking can no longer be achieved, not even by learning more slowly. As the uncertainty typically increases with the frequency, $Q(j\omega)$ is often chosen as a low pass filter, effectively deactivating the ILC controller for high frequencies with much uncertainty, while obtaining good tracking in the less uncertain, lower frequency range.

Since an accurate plant model is not required to achieve a good tracking performance, ILC is well suited to the control of wet-clutch engagements, where the plant dynamics are nonlinear and vary significantly over time. For each of the two ILC controllers, a single, linearized model, approximating the plant dynamics in all conditions suffices, keeping the required modeling effort small. With these choices it becomes possible to design ILC controllers that achieve bandwidths of > 10 Hz, but in practise the controllers are detuned intentionally. Especially in the slip phase this is needed, as it is preferable to keep the jerk low instead of aggressively tracking the reference.

For ILC the computational cost is very low, since it only requires linear filtering operations ($Q$ and $L$), but to do so it does require storing a number of vectors of a length equal to the duration of an engagement (for example the previous tracking error). Implementation on typical industrial controllers is thus possible, but when this is not the case it is more likely due to memory problems than due to a high computational load. A more detailed description of the implementation applied to the wet clutch can be found in [25].

### *3.3. Iterative Optimization*

Another technique based on learning control, denoted iterative optimization, has been developed as an alternative to the two-level ILC approach. It is aimed at (i) reducing the amount of time needed before a good performance is obtained, at (ii) allowing an easier adaptation mechanism to deal with changes in the oil temperature and the load, and at (iii) removing the dependence on the parameterization of the reference profile. The method should however not require accurate models, as was the case for two-level NMPC scheme, but should instead operate using simplified linear models. To achieve this result, it is useful to note that the two-level ILC scheme learns in an indirect manner. At the low level learning takes place, but there the only goal is to accurately track a reference so this reference's quality can be evaluated, even though this reference's parameters are likely to change at the high level so that the low learning will have to be restarted. This is a consequence of the fact that the task itself does not deal with tracking at all, but was formulated in such a manner to be able to use classical tracking-based ILC techniques. The idea in iterative optimization is to omit this indirect approach, and to directly learn based on the specifications themselves. Solving the problem more directly will reduce the convergence period and reduce the effort needed to adjust to varying conditions, and since this effectively removes the parameterized reference, this will also remove the dependence on the chosen parameterization.

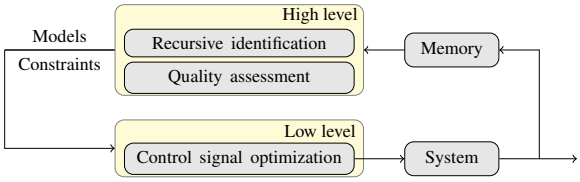


Figure 6: Two-level iterative optimization control scheme: At the high level, the models and constraints for the optimization problem are updated after each engagement, which are then used at the low level to optimize the control signal for the next engagement.

The resulting IO technique again uses a two-level control scheme, but learning is now only included at the high level. At the low level, a numerical optimal control problem is solved, formulated directly from the specifications. Since it is in general very difficult to accurately solve such a problem without a large amount of prior information, some constraints can be included whose exact value in order to reach optimality is initially unknown, and afterwards learning laws can be added at the high level to find appropriate values for these constraints, based on the results observed using the control signals calculated with the current values. Besides these laws, the high level also contains algorithms for a recursive model estimation to describe the system dynamics, combining previously estimated models with the newly measured data. A schematic overview of this control scheme is presented in Figure 6, where it can be seen that an optimal control problem is essentially solved before each trial, using models and constraints that are adapted after completion of each trial based on the observed performance.

When applying this method to the clutch a numerical optimization problem has to be formulated and solved, and we will again try to separate the two phases, although a single large optimization is solved. First, in the filling, the goal is to advance

into to the slip phase as soon as possible, without causing unwanted torque spikes that could cause operator discomfort. Afterwards, once the slip phase begins, the goal becomes to further engage the clutch while keeping the jerk as low as possible. To optimize the control signals accordingly, a piece-wise linear model structure is selected, with one model to predict the pressure and piston position in the filling phase and one to predict the pressure and the slip in the slip phase, while recursive estimation techniques are added to learn these online, so that these models are tuned to the observed behaviour. Transition constraints are also added to ensure a smooth transition occurs between both phases, but since the optimal conditions in which to go from the filling to the slip are unknown, values for these are chosen and afterwards their optimal values are found using learning laws. Using the notation that $\dot{z}(k)$ denote the discrete time finite difference $(z(k+1) - z(k))/T_s$ with $T_s$ the sampling time, the problem to be solved at the low level is then

$$\min_{\substack{u(:),\\ x(:),\, p(:),\, s(:),\, \tilde{z}(:),\\ j_{\max},\, K_1,\, K_2}} \quad K_1 \;+\; \gamma * j_{\max}, \tag{10a}$$

s.t.

$$\text{filling phase:} \quad k = 1 : K_1$$

$$\mathbf{x}(k+1) = \mathbf{A_1}\mathbf{x}(k) + \mathbf{B_1}u(k), \tag{10b}$$

$$\begin{pmatrix} p(k) \\ \tilde{z}(k) \end{pmatrix} = \mathbf{C_1}\mathbf{x}(k) + \mathbf{D_1}u(k), \tag{10c}$$

$$u_{\min} \le u(k) \le u_{\max}, \tag{10d}$$

$$p_{\min} \le p(k) \le p_{\max}, \tag{10e}$$

$$\text{slip phase:} \quad k = K_1 + 1 : K_1 + K_2$$

$$\mathbf{x}(k+1) = \mathbf{A_2}\mathbf{x}(k) + \mathbf{B_2}u(k), \tag{10f}$$

$$\begin{pmatrix} p(k) \\ s(k) \end{pmatrix} = \mathbf{C_2}\mathbf{x}(k) + \mathbf{D_2}u(k), \tag{10g}$$

$$0 \le s(k) \le s_{\text{trans}}, \tag{10h}$$

$$-j_{\max} \le \ddot{s}(k) \le j_{\max}, \tag{10i}$$

$$\text{transition and terminal constraints:}$$

$$\mathbf{x}(K_1+1) = \mathbf{x}_{\text{trans}}, \tag{10j}$$

$$p(K_1) = p_1, \tag{10k}$$

$$\tilde{z}(K_1) = z_{\text{final}}, \qquad \dot{\tilde{z}}(K_1) \le \epsilon, \tag{10l}$$

$$s(K_1+K_2) = 0, \quad \dot{s}(K_1+K+2) = 0 \tag{10m}$$

In this problem, the piecewise structure can clearly be seen, as the problem is split into two parts with $K_1$ and $K_2$ samples for each phase respectively (with $K_1 + K_2 = T/T_s$), and a set of constraints which need to be respected during the transition. In order for the solutions of this problem to yield good engagements, the high-level learning laws recursively identify the matrices $\mathbf{A_i}$, $\mathbf{B_i}$, $\mathbf{C_i}$ and $\mathbf{D_i}$. Since the piston position is not measured, its model $\tilde{z}$ can however not be estimated so easily, so here we use a simple first principles model and rescale it using a rule similar to (6). Similar rules are included for $\mathbf{x}_{\text{trans}}$, $p_1$ and $z_{\text{final}}$.

In terms of computational load, problem (10) reduces to a convex optimization problem (either a linear or quadratic program depending on the regularization), assuming we know the duration of the filling phase. Since in reality we don't know this duration and need to find the optimal one, this problem is not solved as a single convex problem, but we instead solve it by solving a series of convex sub-problems, each with a different but fixed filling phase duration, after which we then use the one with the shortest still feasible duration. Since this only requires solving a series of convex problems, the total solutions are found in about 1s on a normal laptop CPU. However, no attempts have been made to further reduce the calculation time since the problems are solved in between engagements (and not at every timestep), so that the calculation can be spread out over time. A more detailed description of the implementation can be found in [26].

The control techniques presented so far are based on certain characterization of the system dynamics. In very complex systems, however, another approach could be to focus entirely on improving the performance without the intermediate step of modeling the system. Two representative techniques which fall in this category are discussed next.

## 4. Model-free Learning Control

Next to the model-based algorithms described so far in section 3, the potential of model-free algorithms has also been investigated. To date, most complex mechatronic systems are controlled using either a model-based technique, or using controllers tuned during an experimental calibration. Even though these latter are often tuned without the use of a model, this tuning is usually done in an ad-hoc manner derived from system knowledge or insight, and a systematic model-free machine learning (ML) strategy is rarely applied. These strategies would however make it possible to also learn controllers for more complex situations, where insight would not be sufficient to yield the desired behavior. This can improve the current controllers by being able to use more complex control laws, or allowing to optimize a cost criterion and taking into account constraints. This can further also make it possible to develop controllers for more complex applications, for which now no good controllers can be tuned automatically.

Nowadays, wet clutches in industrial transmissions are filled using a feed forward controller of the current (with a set of tunable parameters) to the electro-hydraulic valve. These are now tuned using some heuristic rules, but now we will use model-free learning control methods instead, while still looking for optimal parameterized control signals.

### *4.1. Genetic Algorithm*

Genetic Algorithm (GA) is a stochastic search algorithm that mimics the mechanism of natural selection and natural genetics, and belong to the larger class of evolutionary algorithms (EA). They are routinely applied to generate useful solutions for optimization and search problems, often for complex non-convex problem where gradient-based methods fails to find the

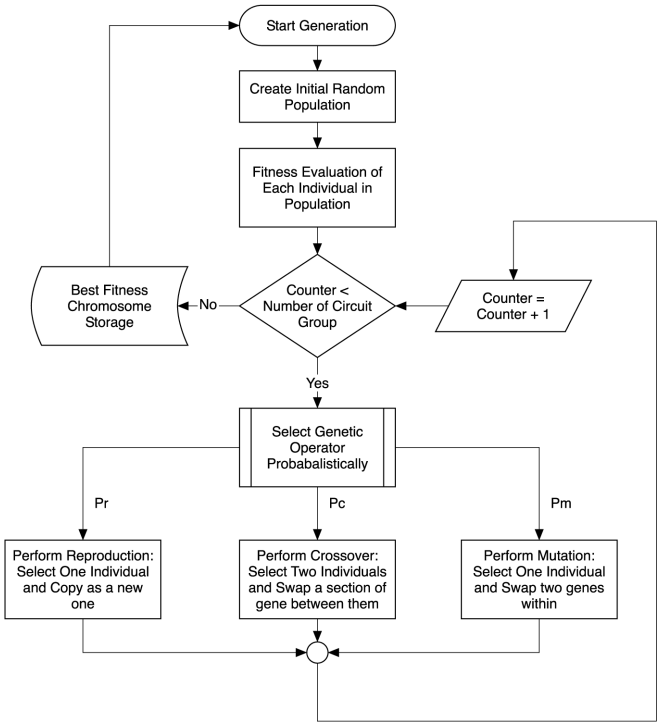


Figure 7: General structure of a genetic algorithm

correct solution. One of the main strengths of GA is that multi-objective optimization problems [27, 28] can be studied.

Unlike conventional optimization techniques, GA starts with an initial set of random solutions (satisfying the boundary and/or system constraints though), called the population. Each individual in the population is called a chromosome, which represents a possible solution to the implementation. Usually, a chromosome is a string of symbols, but not necessarily is a binary bit string. The idea of a GA is that the chromosomes evolve through successive iterations called generations, and converge towards the solution. To achieve this, the chromosomes are evaluated throughout their evolution by a function to obtain a fitness value. Once a complete generation is evaluated, the next generation, with new chromosomes called offspring, are formed by i) copying from the parents using a reproduction operator; ii) merging two chromosomes from current generation using a crossover operator; iii) modifying a chromosome using a mutation operator [29]. The selection of which parents' chromosomes will be used is based on the fitness values, with fitter chromosomes having a higher probability of being selected. Fig. 7 illustrates how a generation is used to define the next one in simple genetic algorithm [30].

For the application to the clutch, each chromosome contains values of the parameters of the parameterized control signal that is applied to engage the clutch. It contains five variables for tuning as shown in Fig. 8. First, a step signal with maximum height and width $d_1$ is sent to the valve to generate a high pressure level in the clutch. With this pressure, the piston will overcome the force from the return spring, and start to get closer to the clutch disks. After this pulse, the signal will give some lower current

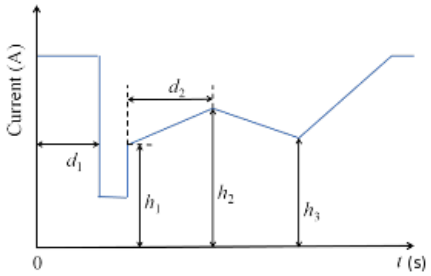


Figure 8: Parameterized signal with five tunable parameters $d_1, h_1, d_2, h_2, h_3$, optimized by both GA and RL to obtain fast and smooth engagement

with fixed height and width to decelerate the piston and try to position it close to the clutch disks. Once the piston is close to the clutch disks and with very low velocity, a force is needed to push the piston forward further, so that the clutch disks are compressed together. Since, the change between the fill phase and slip phase would happen within this period, by providing more freedom to the signal, better engagement performance can be achieved. As a result, two slopes are used to cover this critical period, defined by combination of $h_1, d_2, h_2, h_3$ . Then a ramp current signal with fixed slope and the end height is sent to the valve so that the pressure inside the clutch will increase again gradually. In order to secure the full closing of the clutch, the current signal will be kept constant at the end.

In each generation, each of the chromosomes is evaluated, which is done by applying the corresponding control signal experimentally to the clutch, and afterwards calculating a scalar reward to express the engagement quality[1]. For the reward, we want a function that is monotonically decreasing with the maximum jerk. We could therefore choose it as $r(jerk) = e^{k_1(1-jerk/k_2)}$, with $k_2 = 5000$, corresponding to what can be considered a typical value for the jerk during an engagement. Regardless of the value of $k_1$, this will give a reward $r = 1$ for a jerk of $jerk = k_2 = 5000$, and rewards higher and lower than 1 for lower and higher jerk values. The constant $k_1$ controls the steepness of the reward, and we choose it as $k_1 = 5$. Next, to take into account engagement time, we discount the reward with a discount factor $\gamma$, so the overall reward is given by

$$r = \gamma^{ent}.e^{(5-jerk/1000)} \tag{11}$$

where the *ent* is the engagement time. Since $\gamma$ is chosen as $\gamma = 0.8$, longer engagements will yield lower rewards than shorter engagements, so that to find the highest reward it is needed to do an engagement that is both smooth and fast, similar to our control objectives. A more detailed description of the implementation applied to the wet clutch can be found in [31].

### *4.2. Reinforcement Learning*

RL problems [32] are a class of machine learning problems, where an agent must learn to interact with an unknown environment, using a trial and error approach. At a given timestep $t$, the

[1]For GA's, we can also treat is as a multi-objective problem, in which the reward is not a scalar function. For comparison to the RL technique discussed further on, we do use GA with a scalar reward here though. See [31] for more details and an example of using multi-objective GA for wet clutch control.

agent may execute one of a set of actions, possibly causing the environment to change its state and generate a (scalar) reward. Both state and action spaces can be multidimensional, continuous or discrete. An agent is represented by a policy, mapping states to actions. The aim of a RL algorithm is to optimize the policy, maximizing the reward accumulated by the agent.

In this work, we apply an existing variant of the basic Policy Gradient method [33], called Policy Gradients with Parameter-based Exploration (PGPE) [34]. In this approach, the parameters of a controller are adapted based on the return collected during the whole epoch, regardless of the trajectory in the state space. The advantage of using a direct policy search method is that it easily allows to use a policy that has been optimized on a simulated plant as a good starting point for learning to control the real plant. In the remainder of this section we briefly describe PGPE, referring the reader to [34, 35] for further details.

In Policy Gradients (PG) methods, the policy is represented as a parametric probability distribution over the action space, conditioned by the current state of the environment. Epochs are subdivided into discrete time steps: at every step, an action is randomly drawn from the distribution, conditioned by the current state, and executed on the environment, which updates its state accordingly. After an epoch has been completed, the parameters of the policy are updated, following a Monte Carlo estimate of the expected cumulative (discounted) reward.

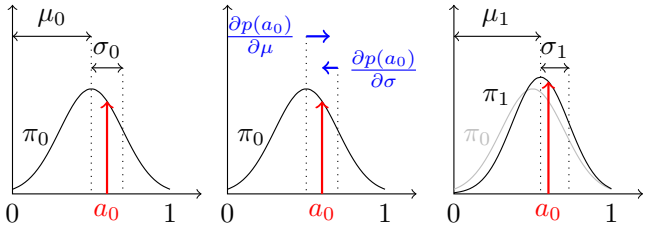


Figure 9: A simple example illustrating the effect of one step of PGPE, with no state information and single stage epochs ($T = 1$). A single policy parameter $\mathcal{A} = [0, 1]$ is sampled from a Gaussian prior $\pi$, with $\boldsymbol{\theta} = (\mu, \sigma)$. *Left:* the first epoch is executed, drawing a parameter value $a_0 \sim \pi_0(a)$, and observing a return $R_0$. *Center:* as $R_0 > b$, following the gradient (12) increases $\pi(a_0)$. *Right:* updated prior $\pi_1$, ready for the next epoch.

A major disadvantage of PG methods is that drawing a random action at every timestep may result in noisy control signals, as well as noisy gradient estimates. Moreover, the policy is required to be differentiable w.r.t. its parameters. To overcome these issues, PG with Parameter-based Exploration (PGPE) was introduced [34, 36]. In this method, the random sampling and policy evaluation steps are, in a sense, 'inverted': the policy is a parametric function, not necessarily differentiable, therefore it can be an arbitrary parametric controller; the parameter value to be used is sampled at the beginning of each epoch from a Gaussian distribution, whose parameters are in turn updated at the end of the epoch, again following a Monte Carlo estimate of the gradient of the expected return. In other words, rather than searching the parametric policy space directly, PGPE performs a search in a 'meta-parameter' space, whose points correspond to probability distributions over the (parametric) policy space.

To simplify notation, we consider a parametric policy $f_a$ with a scalar parameter $a$. Be $\boldsymbol{\alpha} = (\mu, \sigma)$ the meta-parameter defining the Gaussian distribution $p_{\boldsymbol{\alpha}}(a)$ over parameter values. The index we intend to maximize is the expected value of the return $R$ given $a$, $J = E\{R|a\}$. The gradient of this expected return $J$ with respect to the metaparameter $\boldsymbol{\alpha}$ is then estimated as follows (see [34] for details):

$$\nabla_{\boldsymbol{\alpha}} J \approx \frac{1}{N} \sum_{n=1}^{N} \nabla_{\boldsymbol{\alpha}} \log p_{\boldsymbol{\alpha}}(a^n)(R^n - b), \tag{12}$$

where $\boldsymbol{\theta}^n$ is the parameter used at the $n$-th of the $N$ epochs considered (typically $N = 1$), and $b$ is a *baseline* return, which, in the simplest case, is the average return observed so far. Based on this estimated gradient, the policy is then updated, as illustrated in Fig. 9.

For application of PGPE to control of the wet clutch, choice of the policy and reward function are critical. We discard the state information entirely, and adopt an open loop approach, defining the five DOF control signal parameterized in $d_1, h_1, d_2, h_2, h_3$, as stated before in Fig. 8 as the policy to be applied to the plant. Thus the RL problem is reduced to a simpler optimization problem, in which only the parameters of the control signal need to be optimized. To do so, we use a scalar reward function $r$ as in (11) that favours both the objectives of fast and smooth engagement at once, which is the same as used by GA described in the previous section. Note that, the important thing is just that this reward function monotonically decreases in the objective which we intend to minimize. A more detailed description of the implementation applied to the wet clutch can be found in [37].

## 5. Experimental results

To validate the three model-based and two model-free control techniques developed in 3, 4 respectively, they have been applied to the experimental setup described in section 2. The objective is to keep the engagement time during slip, $\Delta T$ within 1s and fill as fast as possible, such that the bound on jerk is minimized.

### 5.1. Model-based controllers

In a first test, the goal is to investigate the performance at a fixed set of nominal conditions, with the oil maintained at $40°C$ and the observed inertia fixed at $8.4 kgm^2$. All parameter values to be learned are intentionally initialized poorly to illustrate the convergence process. First, let's compare the two-level NMPC and ILC approaches, for which the evolution of the reference parameters is shown in Fig. 10, and the resulting engagements are shown in Fig. 11. For both, the initial performance is poor, with a high torque peak due to an initial overfilling, resulting in an uncomfortable engagement for the operator. As a result, during the first parameter update, which is after 1 engagement for the NMPC approach and after 5 engagements for the ILC

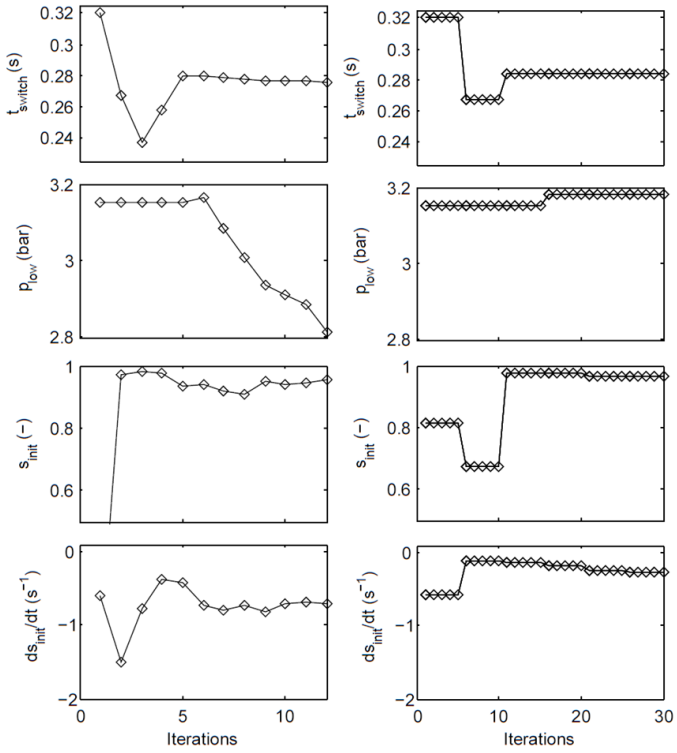


Figure 10: Two-level NMPC (left) and ILC (right): Evolution of the reference signal parameters, where $t_{switch}, P_{low}, S_{init}, dS_{init}/dt$ are the switching time, low pressure, initial slip and its derivative respectively.

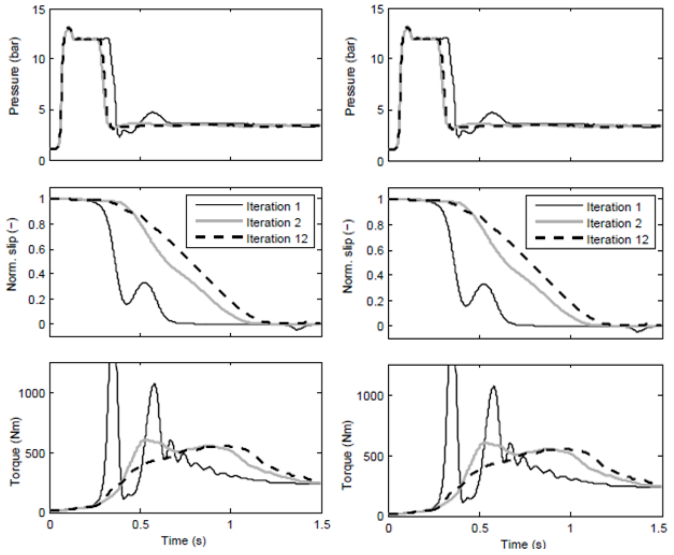


Figure 11: Two-level NMPC (left) and ILC (right): Improving engagement quality during convergence period at nominal conditions.

approach (to allow the low-level tracking time to converge), the high-level controller reacts by reducing $t_{switch}$ as shown in Fig. 10. Over the course of the following iterations, this and the other parameters are further adapted, and eventually smooth engagements are obtained, after 10 and 30 iterations respectively.

The results for the IO technique are shown in Fig. 12, where similarly to the two-level NMPC and ILC approaches, we can see that the performance improves as more iterations pass by, and eventually smooth engagements are found, synchronizing the clutch in a similar timeframe. This improvement is partially due to the learning of a few constraint parameters, but also due to the improving prediction accuracy of the models used in the low-level optimization, as shown in Fig. 13. The convergence period is 10 trials, significantly shorter than that for the two-level ILC scheme and similar to that of the two-level MPC scheme. This reduction with respect to the two-level ILC scheme results form removing the indirect approach and instead directly optimizing based on the real specifications.

In a second test, the robustness is investigated by changing the operating conditions under which the engagements are performed. First, the load is kept at the same value but the oil temperature is increased to $70°C$. Next, the oil is $40°C$ again, but the observed inertia is increased to $28.6 kgm^2$. Fig. 14 depicts the obtained results of the two-level NMPC and ILC controllers, after convergence, which again takes around 10 and 30 iterations respectively. For the increased temperature, the main

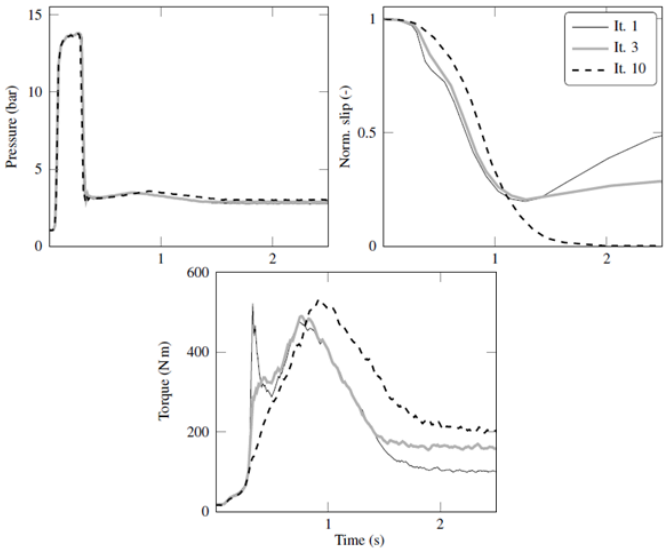


Figure 12: Iterative optimization: Improving engagement quality during convergence period at nominal conditions.

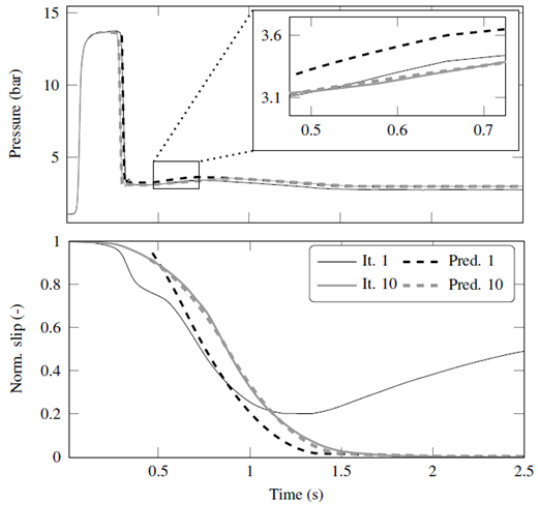


Figure 13: Iterative optimization: Improving prediction accuracy during convergence period.

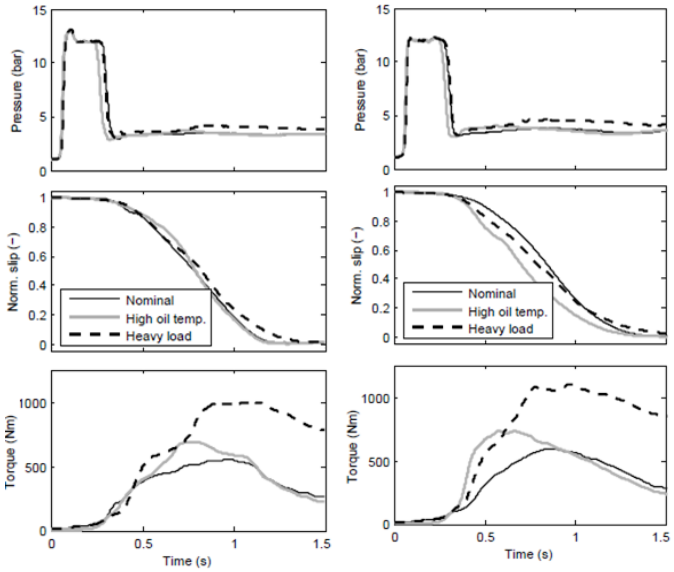


Figure 14: Two-level NMPC (left) and ILC (right): Demonstration of robustness to various operating conditions.

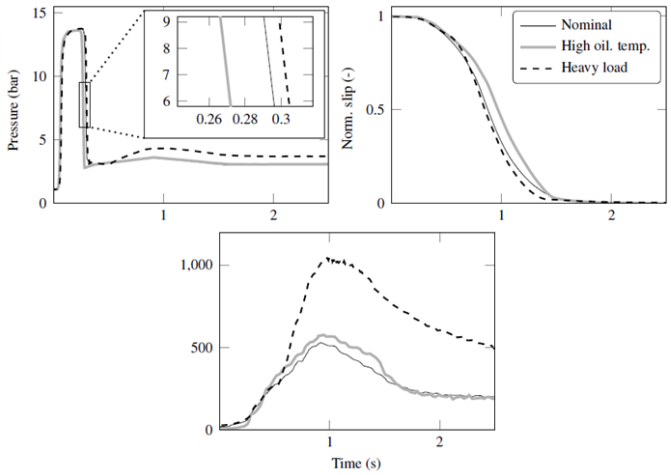


Figure 15: Iterative optimization: Demonstration of robustness to various operating conditions.

difference is that the filling is completed sooner, which results from the decreased oil viscosity, and which has been compensated for mainly by reducing the value of $p_{wid}$ in the pressure reference. More differences can be observed when the observed load is increased, as then higher torques and pressures are required, but despite this the slip signals remain very similar to the nominal case.

For the IO technique, the same tests have been performed, and the results after convergence are shown in Fig. 15. As before, a good performance is still achieved by having the high-level laws adapt to the observed changes. The fact that the performance is similar to those of the two-level NMPC and ILC approaches illustrates that the parameterization used for those two is well-chosen, as they perform similar adaptations using only a few parameters. The reconvergence period is again around 10 trials, similar to that of the two-level NMPC approach, and shorter than for the two-level ILC approach.

Apart from the shorter reconvergence period, the two-level and IO techniques share the additional advantage that if learning were to be restarted for a different operating point, it is easier to hotstart with a good guess of either the reference parameters or the constraint parameters and models, since it is easy to store and interpolate these values. For the two-level ILC approach the reference parameters could also be stored, but to fully allow a hotstarting and reuse all knowledge learned at a previous set of operating conditions, it would also be needed to transform the learned control signal to the current operating conditions, which is not a straightforward task.

### 5.2. *Model-free controllers*

Before we look into the results, it should be noted that each set of control signal parameters were first tested under conditions with a reduced load, to ensure it could be safely applied to the clutch under normal operating conditions. These additional tests are not included in any of the results, nor are they counted in the number of trials before convergence, but they do slow down the overall learning process. Ideally, other methods to

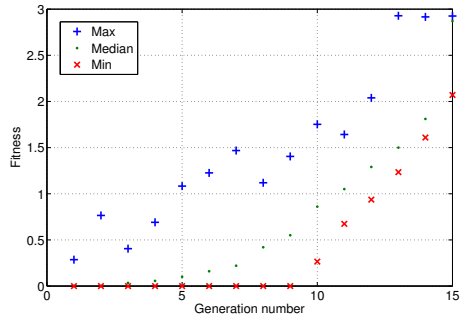


Figure 16: GA: Minimum, median, and maximum fitness values during the GA evolution process.

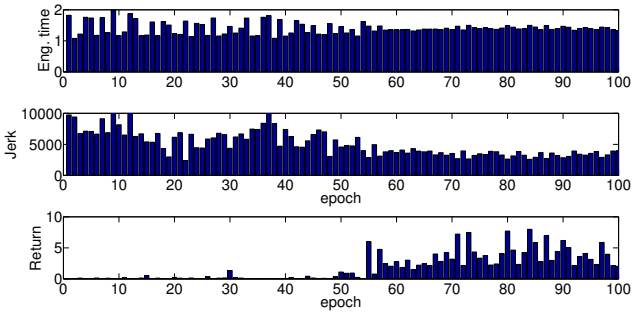


Figure 17: PGPE: Evolution of engagement time (above), jerk (center), and reward (below) during learning process.

ensure safety would need to be derived, not requiring (as many) additional experiments.

To illustrate the convergence process of the model-free methods, the nominal conditions defined earlier for model-based controllers are reused. Under these conditions, the optimization processes for both are shown in Fig. 16 and Fig. 17 respectively. The GA maximized the fitness within 13 generations; each generation containing 50 individuals, while for reinforcement learning the reward is maximized after 85 test runs. The results obtained with each are presented in Fig. 18, where it can be seen that engagements similar to those of the model-based techniques are obtained.

The robustness with respect to an increase in the oil temperature has also been checked for these methods. In this case, we reuse knowledge from the previous experiment under nominal conditions to narrow down the range of the parameter's value to reduce the amount of learning needed, and the results after convergence are also included in Fig. 18. Similar observations can be made as before with the model-based controllers, with a good performance still being achieved by both controllers.

## 6. Comparison of model-based and model-free learning control

This section compares the results of all the methods on the clutch, and presents a general discussion of their benefits and drawbacks.

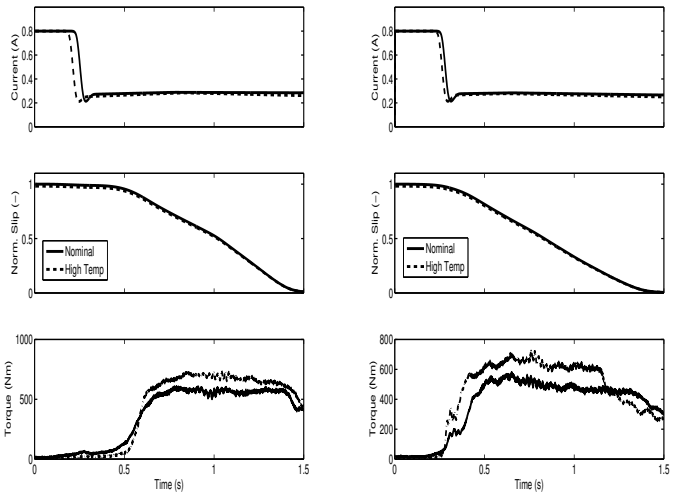


Figure 18: GA (left) and RL (right): Illustration of engagements achieved under nominal conditions and with an increased temperature.

### 6.1. Comparison of engagement results

A comparison of all the presented methods for clutch control in terms of engagement time and jerk is presented in Table 1. Among the model-based methods, the results are fairly similar, and none of them are clearly worse than any of the others in both categories. Among the model-free methods, RL performs slightly better than GA though in both categories, which probably indicates that the GA has not fully converged yet, as it should normally find a similar result as RL. Comparing the model-based and model-free techniques, we see that the model-based ones find engagements that are both faster and yield lower jerk values. This can be explained by the parameterization that is used for the model-free methods, which restricts the possible adaptations made by the controller, and which can lead to a reduced performance.

### 6.2. Discussion on model-based techniques

Comparing the model-based techniques, we immediately see that the two-level NMPC and IO technique require a similar convergence period, which is shorter than for the two-level ILC technique. This is a result of the fact that the two-level ILC technique learns at both low and high levels, requiring additional iterations in which the low level converges, before good high-level updates can be made. As a result, the reconvergence period when the operating conditions change is also better for the two-level NMPC and the IO technique. Apart from the shorter reconvergence period, they have the additional advantage that if learning were to be restarted for a different operating point, it is easier to hotstart with a good guess of either the reference parameters or the constraint parameters and models, since it is easy to store and interpolate these values. This could also be done for the reference parameters of the two-level ILC approach, but to fully allow a hotstarting and reuse all knowledge learned at a previous set of operating conditions, it would also

| Index | 2l-NMPC | 2l-ILC | IO | GA | RL |
|---|---|---|---|---|---|
| Abs(Max(Jerk)) | 3.6683 | 2.8945 | 3.4133 | 3.9256 | 3.618 |
| Eng.Time(s) | 1.199 | 1.39 | 1.277 | 1.434 | 1.317 |

Table 1: An empirical comparison between the model-based and non-model based control techniques based on jerk and engagement times

| Method/Property | 2l-NMPC | 2l-ILC | IO | GA | RL |
|---|---|---|---|---|---|
| Modeling requirement | ⌢ | ⌣ | ⌣ | ⌣⌣ | ⌣⌣ |
| Learning rate | ⌣⌣ | ⌣ | ⌣⌣ | ⌢⌢ | ⌢ |
| Stability | ⌣ | ⌣ | −− | −− | −− |
| Learning transient/Safety | ⌣ | ⌣ | ⌣ | ⌢⌢ | ⌢⌢ |
| Multi-objective | ⌣ | ⌣ | ⌣ | ⌣⌣ | ⌣ |

Table 2: Characteristic features of the model-based and model-free techniques

be needed to transform the learned control signal to the current operating conditions, which is not a straightforward task.

When comparing the required modeling effort, the two-level ILC technique outperforms the other methods however, as a highly accurate tracking control can be achieved despite having a large model uncertainty. In contrast, for NMPC it is needed to have an accurate non-linear model, which requires a time-consuming identification. For the IO technique this is not needed again, but here the performance is limited by the chosen model structure, of which the parameters are afterwards learnt online.

Even though the ILC approach improves the tracking behaviour, so that the references can be tracked more accurately than with the NMPC approach, it turns out that this may not be beneficial for the current application, where aggressive control can lead to unwanted vibrations and high jerk values.

### *6.3. Discussion on model-free techniques*

Comparing the model-free techniques, we see that with the same reward/fitness function, both methods manage to yield similar engagements, but RL converges faster than GA.

The type of reward/fitness that was used is the same for both, and is a trade-off between the jerk and the engagement duration. These can be combined into a single scalar reward, which is the way in which RL typically operates. For GA it is however also possible not to use a fixed trade-off, but to really treat it as multiple objectives without extra cost, and find a complete pareto-front.

### *6.4. Comparison of model-based and model-free techniques*

Model-based methods have a few advantages over model-free methods. First of all, they allow more freedom in the determination of the shape of the control signals. They will therefore generally be able to find the optimal solution, whereas for the model-free methods this is only possible if the chosen parameterization allows this. The convergence period is also significantly shorter for model-based methods, and hotstarting is often possible, which makes them more applicable for online adaptive purposes. Another advantage is that they posses an inherent robustness to parameter uncertainties due to the ability to use feedback. They finally also make it easier to predict the behaviour and thus ensure safety, even during the convergence period.

Despite these advantages of model-based methods, model-free methods also have some attractive properties for the control of mechatronic systems. Their main benefit is that they can operate without model, and thus require no identification or apriori system knowledge. This makes them ideal for usage as an add-on to complex existing systems, or to automate offline calibration procedures where it is not possible to rely on heuristics or insight to manually design tuning rules. It should however be stated that parameterizations are typically needed to limit the convergence period, and it is practically impossible to select the shape of the signal beforehand without system knowledge or some simple tests. While these parameterizations generally do lead to a reduced performance, a well-chosen parameterization can limit this reduction, and this choice is thus important. Since more parameters lead to a better performance but longer convergence, the difficult part is to select parameterizations with only a low number of parameters, yet which still allow a performance close to the true optimum to be achieved.

These results have been summarized in Table 2, which gives a qualitative comparison between the different techniques. The key ⌣⌣ means the best in the category, followed by ⌣ for good and then ⌢ for bad to ⌢⌢ for worst in the category. The −− key signifies that the corresponding property has not been established.

## 7. Conclusions

In this paper we presented different model-based and model-free methods for the control of wet clutches, which are typical complex mechatronic systems with fast dynamics, uncertainty, nonlinearity and unknown optimal reference trajectories. All techniques have been implemented and validated experimentally, and good results are generally achieved by all.

The model-based methods do converge in shorter time periods, and make it easier to guarantee safety during the convergence period, which makes then more suitable to online applications. The model-free methods on the other hand can be applied to complex systems whenever models are hard to come by, and are especially useful as an automated tuning method

when insight in the dynamics does not allow an experienced user to define proper tuning rules. These model-free methods can further also be used to learn complete pareto-fronts of optimal controllers, allowing a selection of which controller to be used to be made later on.

The combination and extension of all the stated methodologies for distributed control is a work in progress.

## 8. Acknowledgements

This work is carried within the framework of the LeCoPro project (grant nr. 80032) of the Institute for the Promotion of Innovation through Science and Technology in Flanders (IWT-Vlaanderen).

## References

[1] Z.Sun, K.Hebbale, Challenges and opportunities in automotive transmission control, in: American Control Conference, Oregon, USA, 2005, pp. 3284–3289.

[2] L. Glielmo, F. Vasca, Engagement control for automotive dry clutch, in: American Control Conference, Chicago, USA, 2000, pp. 1016–1017.

[3] P. J. Dolcini, Contribution to the clutch comfort, Institut National Polytechnique de Grenoble, Grenoble, France, 2007.

[4] A. Heijden, A. Serrarens, M. Camlibel, H. Nijmeijer, Hybrid optimal control of dry clutch engagement, International Journal of Control (2007) 1717–1728.

[5] M. Z. X. Song, M. Azrin, Z. Sun, Automotive transmission clutch fill optimal control: An experimental investigation, in: American Control Conference, Baltimore, USA, 2010, pp. 2748–2753.

[6] R. Morselli, R. Zanasi, E. Sereni, E. Bedogni, E. Sedoni, Modeling and control of wet clutches by pressure-control valves, in: IFAC Symposium on Advances in Automotive Control, Salerno, Italia, 2004, pp. 79–84.

[7] V. A. Neelakantan, G. N. Washington, N. K. Bucknor, Model predictive control of a two stage actuation system using piezoelectric actuators for controllable industrial and automotive brakes and clutches, Journal of Intelligent Material Systems and Structures 19 (7) (2008) 845–857.

[8] C. L. Constantin, F. C. Andreea, E. Balau, Modelling and predictive control of an electro-hydraulic actuated wet clutch for automatic transmission, in: Industrial Electronics (ISIE), 2010 IEEE International Symposium on, 2010, pp. 256 –261.

[9] K. van Berkel, T. Hofman, A. Serrarens, M. Steinbuch, Fast and smooth clutch engagement control for dual-clutch transmissions, Control Engineering Practice 22 (2014) 57–68.

[10] M. Montanari, F. Ronchi, C. Rossi, A. Tilli, A. Tonielli, Control and performance evaluation of a clutch servo system with hydraulic actuation, Control Eng. Practice, vol. 12(11) (2004) 1369–79.

[11] M. Watson, C. Byington, D. Edwards, S. Amin, Dynamic modeling and wear-based remaining useful life prediction of high power clutch systems, Tribol. Trans., vol. 48(2) (2005) 208–17.

[12] M.J.W.H.Edelaar, Modeling of a wet plate clutch in a driveline, in: WFW Report 96.071, Eindhoven, 1996.

[13] R. Morselli, R. Zanasi, Modeling of Automotive Control Systems Using Power Oriented Graphs, 32nd Annual Conference of the IEEE Industrial Electronics Society, IECON (2006) 7–10.

[14] J. Wong, Theory of ground vehicles (Third Edition), John Wiley & Sons, 2001.

[15] J. Maciejowski, Predictive Control with Constraints, Prentice Hall, 2002.

[16] J. H.Lee, Model predictive control: Review of the three decades of development, International Journal of Control, Automation, and Systems (2011) 415–424.

[17] R. De Keyser, A. Van Cauwenberghe, A self-tuning multistep predictor application, Automatica 17 (1) (1981) 167–174.

[18] R. D. Keyser, Model based predictive control for linear systems, UNESCO Encyclopaedia of Life Support Systems http://www.eolss.net (available online at: http://www.eolss.net/sample-chapters/c18/e6-43-16-01.pdf). Article contribution 6.43.16.1, Eolss Publishers Co Ltd, Oxford (2003) 35 pages.

[19] L. Wang, Model predictive control system design and implementation using matlab, Springer-Verlag, 2009.

[20] W.D.Widanage, J.Stoev, A.VanMulders, J.Schoukens, G.Pinte, Nonlinear system-identification of the filling phase of a wet-clutch system, Control Engineering Practice (2011) 1506–1516.

[21] A. Dutta, M. Loccufier, C. M. Ionescu, R. De Keyser, Penalty adaptive model predictive control (pampc) of constrained, underdamped, non-collocated mechatronic systems, in: Control Applications (CCA), 2013 IEEE International Conference on, 2013, pp. 1006–1011.

[22] A. Dutta, B. Depraetere, C. Ionescu, G. Pinte, J. Swevers, R. De Keyser, Comparison of two-level nmpc and ilc strategies for wet-clutch control, Control Engineering Practice 22 (2014) 114–124.

[23] D. Bristow, M. Tharayil, A. Alleyne, A survey of iterative learning control, Control Systems Magazine, IEEE 26 (3) (2006) 96–114.

[24] R. Longman, Iterative learning control and repetitive control for engineering practice, International Journal of Control 73 (2000) 930–954.

[25] B. Depraetere, G. Pinte, W. Symens, J. Swevers, A two-level Iterative Learning Control scheme for the engagement of wet clutches, Mechatronics 21 (3) (2011) 501–508.

[26] B. Depraetere, G. Pinte, J. Swevers, A reference free iterative learning strategy for wet clutch control, in: Proceedings of the 2011 American Control Conference, 2011, pp. 2442–2447.

[27] D. Kalyanmoy, Optimization for engineering design: Algorithms and examples, PHI Learning Pvt. Ltd., 2004.

[28] R. E. Steuer, Multiple criteria optimization: theory, computation, and application, Krieger Malabar, 1989.

[29] T. Weise, K. Geihs, Genetic programming techniques for sensor networks, in: Proceedings of, Vol. 5, 2006, pp. 21–25.

[30] R. J. Urbanowicz, J. H. Moore, Learning classifier systems: a complete introduction, review, and roadmap, Journal of Artificial Evolution and Applications (2009) 1–25.

[31] Y. Zhong, B. Wyns, R. De Keyser, G. Pinte, J. Stoev, An implementation of genetic-based learning classifier system on a wet clutch system, in: Applied Stochastic Models and Data Analysis Conference, 14th, Proceedings, 2011, pp. 1431–1439.

[32] R. Sutton, A. Barto, Reinforcement learning: An introduction, MIT Press, 1998.

[33] J. Peters, S. Schaal, Policy gradient methods for robotics, in: Proceedings of the IEEE Intl. Conf. on Intelligent Robotics Systems (IROS), 2006.

[34] F. Sehnke, C. Osendorfer, T. Rückstieß, A. Graves, J. Peters, J. Schmidhuber, Parameter-exploring policy gradients, Neural Networks 23 (4) (2010) 551–559.

[35] M. Gagliolo, K. V. Vaerenbergh, A. Rodriguez, A. Nowe, S. Goossens, G. Pinte, W. Symens, Policy search reinforcement learning for automatic wet clutch engagement, in: 15th International Conference on System Theory, Control and Computing, IEEE, 2011, pp. 250–255.

[36] F. Sehnke, C. Osendorfer, T. Ruckstiess, A. Graves, J. Peters, J. Schmidhuber, Policy gradients with Parameter-Based exploration for control, in: Proceedings of the 18th international conference on Artificial Neural Networks, Part I, Springer-Verlag, 2008, pp. 387–396.

[37] M. Gagliolo, K. Van Vaerenbergh, A. Rodriguez, A. Nowé, S. Goossens, G. Pinte, W. Symens, Policy search reinforcement learning for automatic wet clutch engagement, in: System Theory, Control, and Computing (ICSTCC), 2011 15th International Conference on, IEEE, 2011, pp. 250–255.