Tuning of model predictive engine controllers over transient drive cycles [⋆]

**Alejandro I. Maass [*] Chris Manzie [*] Iman Shames [*]
Robert Chin [*,**] Dragan Nešić [*] Nalika Ulapane [*]
Hayato Nakada [***]**

[*] *Department of Electrical and Electronic Engineering, The University of Melbourne, Victoria 3010, Australia (e-mail: {alejandro.maass,manziec,iman.shames,dnesic,nalika.ulapane}@unimelb.edu.au).*
[**] *School of Computer Science, The University of Birmingham, United Kingdom (e-mail: chinr@student.unimelb.edu.au)*
[***] *Advanced Unit Management System Development Division, Toyota Motor Corporation, Japan (e-mail: hayato_nakada@mail.toyota.co.jp).*

Abstract: A framework for tuning the parameters of model predictive controllers (MPCs) based on gradient-free optimisation (GFO) is proposed. Efficient calibration of MPCs is often a difficult task given the large number of tuning parameters and their non-intuitive correlation with the output response. We propose an efficient and systematic framework for the tuning of MPC parameters that can be implemented iteratively within the closed-loop setting. The performance of the proposed GFO-based algorithm is evaluated through its application to air-path control for diesel engines over simulations and experiments. We illustrate that the tuned parameters provide satisfactory tracking of reference trajectories over engine drive cycles with only a few iterations. Thereby, we extend existing MPC tuning approaches that calibrate parameters using step responses on the fuel rate and engine speed onto tuning over a full drive cycle response.

Keywords: Gradient-free optimisation, model-based control, controller calibration, diesel engines, automotive control

1. INTRODUCTION

Model predictive control (MPC) is of significant and growing interest, and it has been widely utilised in industry since the 1970s, see e.g. (Mayne et al., 2000; Morari and Lee, 1999). These controllers are particularly attractive because they can naturally and easily deal with multivariable linear and non-linear systems, whilst handling input/output constraints by directly incorporating these in the optimisation. In order to successfully implement MPCs and obtain a satisfactory performance, several tuning parameters need to be chosen properly. These parameters strongly affect performance and are related to the process dynamics in a very complex way, making controller tuning a non-trivial task. Although a number of papers can be found in the literature which discuss MPC calibration (see e.g. (Garriga and Soroush, 2010) for a complete survey) these are often based on ad-hoc heuristic guidelines. Analytical studies of MPC parameter tuning are proposed in (Bagheri and Khaki-Sedigh, 2014; Shah and Engell, 2011; Shridhar and Cooper, 1998). However, these results hold for specific scenarios and MPC formulations, thus trial-and-error approaches are unavoidable adopted in practice.

More general tuning frameworks based on meta-heuristic algorithms can be found in the literature, such as multi-scenario (Santos et al., 2019), zone-control strategies (Yamashita et al., 2016), particle swarm optimisation (Júnior et al., 2014; Suzuki et al., 2007), genetic algorithms (Van der Lee et al., 2008), gradient-descent algorithms (Bunin et al., 2012), and machine learning methods (Ira et al., 2018). The works (Ira et al., 2018; Santos et al., 2019; Suzuki et al., 2007; Van der Lee et al., 2008) focus on set-point tuning by using characteristics of a step response on the corresponding operating condition, e.g. overshoot, settling and rise times, steady-state errors, etc. This hinders their application to control loops that require proper MPC tuning for trajectory tracking, including chemical batch processes, robotics, and engine control. The authors in (Yamashita et al., 2016) deal with trajectory tuning of MPC but these trajectories are generated by first-order transfer function models and hence do not apply to broader problems.

Frameworks that can deal with tuning over general trajectories are provided by (Bunin et al., 2012; Júnior et al., 2014). Particularly, (Júnior et al., 2014) develops an optimal tuning strategy based on particle swarm optimisation (PSO) that is capable of explicitly dealing with model uncertainty in their formulation. (Bunin et al., 2012) presents a gradient-based algorithm in which the MPC tuning parameters may be brought to a locally optimal set. A practical disadvantage of PSO-based algorithms like (Júnior et al., 2014), is that the number of particles depends on the specific problem and it usually oscillates around 20 to 50 particles, meaning the algorithm needs to run 20 to 50 experiments per iteration, which is not practical for many

[⋆] This work was supported by Toyota Motor Corporation, Japan.

real applications. Gradient-based methods such as (Bunin et al., 2012) require less experiments per iteration, but results are presented only for diagonal tuning matrices.

In this paper, and as our main contribution, we propose an efficient and systematic MPC tuning framework based on gradient-free optimisation (GFO), which covers broader scenarios than the aforementioned works in the literature. Particularly, it can deal with MPC tuning over trajectories in a general setting as opposed to (Ira et al., 2018; Santos et al., 2019; Suzuki et al., 2007; Van der Lee et al., 2008; Yamashita et al., 2016), and it provides MPC weighting matrices that satisfy the required constraints of being symmetric and positive (semi) definite, and thus are more general and provide more degrees of freedom than the usual diagonal choices in (Bunin et al., 2012; Júnior et al., 2014). Moreover, our proposed GFO-based algorithm requires only two experiments per iteration, making it more efficient in practice than PSO-based algorithms (Júnior et al., 2014). In a nutshell, our proposed algorithm utilises the random search method for non-smooth and stochastic optimisation provided in (Nesterov and Spokoiny, 2017). Gradient-free methods may be preferred in practice since they only require cost function values as opposed to gradient (or any higher derivative) information. These methods are particularly helpful when an analytical expression for the cost function is not available, or when it is impractical to obtain. In the MPC tuning context, the cost function is often chosen to be a relevant closed-loop performance index, e.g. tracking error. This cost function value might only be computed after the entire experiment response is available, and since experiments are relatively expensive to perform, having only two experiment runs per iteration in the proposed GFO-based algorithm is useful in practice.

Lastly, to illustrate the performance of the algorithm, we study the MPC tuning problem in the context of air-path control for diesel engines. By doing so, we extend the existing MPC tuning methods for air-path control (see e.g. (Ira et al., 2018; Sankar et al., 2019)) that rely on characteristics of a step-response like overshoot, rise time, etc., which are only indirect performance indicators when the engine is required to track trajectories over drive cycles. We iteratively tune the MPC parameters within the closed-loop setting with the goal of improving the overall tracking performance over drive cycles. Although theoretical guarantees for convergence of GFO-based algorithms require a large number of iterations, and these algorithms converge to a local optima thus relying in the choice of initial condition, we illustrate that even with a few iterations and a heuristic choice of initial condition, the corresponding GFO-tuned MPC parameters provide a significant improvement of tracking performance in both simulations (offline tuning) and experiments (online tuning with real-time data).

Notation: Denote by $\mathbb{R}^{m\times n}$ the set of real matrices of dimension $m \times n$. Let $\mathbb{R}_{>0} \doteq (0,\infty)$, $\mathbb{N} \doteq \{1,2,\dots\}$, and $\mathbb{N}_0 \doteq \{0,1,2,\dots\}$. Given a matrix M, $M^\top$ denotes its transpose, and $M > 0$ (resp. $M \geq 0$) denotes that M is positive (resp. semi-) definite. We use $\operatorname{diag}\{M_1,\dots,M_n\}$ to denote the standard block diagonal matrix. The identity matrix of dimension $m \times n$ is denoted by $I_{m\times n}$. For a vector, $\|\cdot\|_2$ denotes its Euclidean norm, and the same notation is used to denote the induced 2-norm of a matrix. $\mathbf{E}\{\cdot\}$ denotes the expectation operator.

2. SETUP AND PROBLEM DEFINITION

We consider discrete-time systems of the form

$$x_{k+1} = A^\sigma x_k + B^\sigma u_k, \tag{1a}$$

$$y_k = C^\sigma x_k + D^\sigma u_k, \tag{1b}$$

where $x_k \in \mathbb{R}^{n_x}$, $u_k \in \mathbb{R}^{n_u}$, and $y_k \in \mathbb{R}^{n_y}$ are the state, input, and output at time instant $k \in \mathbb{N}_0$, respectively, $n_x, n_u, n_y \in \mathbb{N}$, and $(A^\sigma, B^\sigma, C^\sigma, D^\sigma)$ for $\sigma \in \{1,2,\dots,M\}$ are matrices of appropriate dimensions. These models may come from linearisation of a non-linear model at various linearisation points, or from system identification of a real plant at different operating points.

For a given $\sigma \in \{1,2,\dots,M\}$ and corresponding model (1), we formulate a model predictive controller. Define the cost function as

$$J(x_k,\mathbf{u}) \triangleq V_f(x_{k+H}, w_f^\sigma) + \sum_{i=0}^{H-1} \ell(x_{k+i}, u_{k+i}, w^\sigma),$$

where $H \in \mathbb{N}$ is the prediction horizon, V_f and ℓ denote the terminal and stage costs respectively, $\mathbf{u} \triangleq \{u_k, u_{k+1}, \dots, u_{k+H-1}\}$ is the sequence of control values applied over the horizon H, and w_f^σ, w^σ are vectors containing all the tuning weights of controller σ. The corresponding MPC problem of minimising $J(x_k,\mathbf{u})$ subject to the control, state, and terminal constraints is solved, yielding the optimising control sequence $\mathbf{u}^* = \{u_k^*, u_{k+1}^*, \dots, u_{k+H-1}^*\}$. The first element of the sequence $\mathbf{u}^*$ is applied to the system and the whole process is repeated as k is incremented.

For transient operations between operating points, we use a switching LTI-MPC architecture so that the controllers are switched based on the current system operating condition. That is, as in (Sankar et al., 2019), the LTI-MPC strategy selects the MPC controller at the nearest operating point to the current operating condition.

In the remainder of this paper, we assume that the terminal and stage costs are quadratic, i.e. $V_f \triangleq x_{k+H}^\top P^\sigma x_{k+H}$ and $\ell \triangleq x_{k+i}^\top Q^\sigma x_{k+i} + u_{k+i}^\top R^\sigma u_{k+i}$, where $P^\sigma \in \mathbb{R}^{n_x\times n_x}, Q^\sigma \in \mathbb{R}^{n_x\times n_x}$, and $R^\sigma \in \mathbb{R}^{n_u\times n_u}$ are matrices containing the tuning weights.

Assumption 1. We assume that R^σ is a symmetric positive definite matrix, and that P^σ and Q^σ are symmetric and positive semidefinite. ■

Under this setup, our goal is to propose an efficient and automated tuning framework for the MPC weighting matrices $\{P^\sigma, Q^\sigma, R^\sigma\}$ in order to achieve a satisfactory performance metric for a given prediction horizon $H \in \mathbb{N}$.

Remark 2. Given Assumption 1, the number of tuning parameters for each MPC is $n \triangleq 2n_X + n_U$, where $n_X \triangleq n_x(n_x+1)/2$ and $n_U \triangleq n_u(n_u+1)/2$. That is, n_X parameters for each P^σ and Q^σ, and n_U parameters for each R^σ. Any automated tuning method must therefore be able to efficiently deal with a large number of tuning variables and satisfy the constraints imposed by Assumption 1.

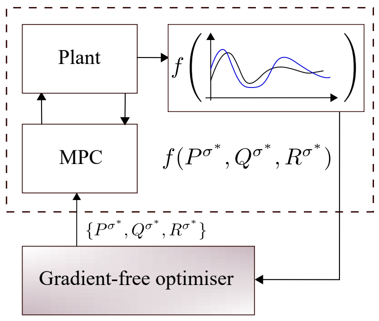
Plant
f
MPC
$f(P^{\sigma^*}, Q^{\sigma^*}, R^{\sigma^*})$
$\{P^{\sigma^*}, Q^{\sigma^*}, R^{\sigma^*}\}$
Gradient-free optimiser

Fig. 1. Overall gradient-free tuning scheme in-the-loop.

3. MPC TUNING FRAMEWORK

3.1 Description

In this section, we describe the proposed algorithm for the tuning of $\{P^\sigma, Q^\sigma, R^\sigma\}$ within the closed-loop composed of the plant and MPC controllers. Suppose there exists an initial heuristic calibration $\{P_0^\sigma, Q_0^\sigma, R_0^\sigma\}$ for every $\{P^\sigma, Q^\sigma, R^\sigma\}$, $\sigma \in \{1, \ldots, M\}$. The choice of $\{P_0^\sigma, Q_0^\sigma, R_0^\sigma\}$ may be based on model dynamics, experience, or using ad-hoc guidelines as per (Garriga and Soroush, 2010).

The first stage of the proposed framework is to identify a controller σ^* that shows poor closed-loop performance after the initial calibration, e.g. poor tracking error. The second stage is then to automatically optimise the parameters $\{P^{\sigma^*}, Q^{\sigma^*}, R^{\sigma^*}\}$ in the sense of minimising a performance function $f(P^{\sigma^*}, Q^{\sigma^*}, R^{\sigma^*})$ which depends on the closed-loop response. This function measures closed-loop performance of controller σ^* for a given triplet $\{P^{\sigma^*}, Q^{\sigma^*}, R^{\sigma^*}\}$. The goal is to improve the closed-loop performance by fine-tuning the underperforming controller σ^*. That is, we attempt to solve the problem

$$\mathbf{f}^\star \triangleq \min_{P^{\sigma^*}, Q^{\sigma^*} \in \mathbb{S}, R^{\sigma^*} \in \mathbb{S}_+} f(P^{\sigma^*}, Q^{\sigma^*}, R^{\sigma^*}),$$

where $\mathbb{S} \subseteq \mathbb{R}^{n_x \times n_x}$ and $\mathbb{S}_+ \subseteq \mathbb{R}^{n_u \times n_u}$ denote the set of symmetric positive semi-definite matrices and the set of symmetric positive definite matrices, respectively.

The overall tuning scheme is illustrated in Fig. 1. The GFO block iteratively updates the MPC parameters $\{P^{\sigma^*}, Q^{\sigma^*}, R^{\sigma^*}\}$ to minimise $f(P^{\sigma^*}, Q^{\sigma^*}, R^{\sigma^*})$, which is calculated from closed-loop response experiments (or simulations) carried out within the dashed box.

3.2 GFO tuning algorithm

The algorithm we propose is based on the gradient-free method in (Nesterov and Spokoiny, 2017), which uses the so-called *random gradient-free oracles* for the random search. In our context, the oracle is defined as

$$g^\star(X, Y, Z) \triangleq \frac{1}{\mu}\Big(f(X + \mu M^X, Y + \mu M^Y, Z + \mu M^Z) - f(X, Y, Z)\Big) M^\star, \qquad (2)$$

where $\star \in \{P, Q, R\}$, $M^\star$ are random symmetric matrices of appropriate dimensions, and $\mu \in \mathbb{R}_{>0}$ is the smoothing parameter.

The underlying algorithm of the scheme in Fig. 1 is represented in pseudocode in Algorithm 1. The operators $\pi_{\mathbb{S}}$

Algorithm 1 GFO tuning scheme in Fig. 1

```
1: Choose $\{P_0^{\sigma^*}, Q_0^{\sigma^*}, R_0^{\sigma^*}\}$.
2: for $j = \{0, \ldots, J\}$, $J \in \mathbb{N}$ do
3:     Perform a closed-loop response with parameters $\{P_j^{\sigma^*}, Q_j^{\sigma^*}, R_j^{\sigma^*}\}$ for the $\sigma^*$-th controller.
4:     $f_1 \leftarrow f(P_j^{\sigma^*}, Q_j^{\sigma^*}, R_j^{\sigma^*})$.
5:     Generate random symmetric matrices $M_j^P, M_j^Q, M_j^R$.
6:     Perform a closed-loop response with parameters $\{P_j^{\sigma^*} + \mu M_j^P, Q_j^{\sigma^*} + \mu M_j^Q, R_j^{\sigma^*} + \mu M_j^R\}$ for the $\sigma^*$-th controller.
7:     $f_2 \leftarrow f(P_j^{\sigma^*} + \mu M_j^P, Q_j^{\sigma^*} + \mu M_j^Q, R_j^{\sigma^*} + \mu M_j^R)$.
8:     Compute the random oracles
           $g^P(P_j^{\sigma^*}, Q_j^{\sigma^*}, R_j^{\sigma^*}) \leftarrow \frac{1}{\mu}(f_2 - f_1) M_j^P$
           $g^Q(P_j^{\sigma^*}, Q_j^{\sigma^*}, R_j^{\sigma^*}) \leftarrow \frac{1}{\mu}(f_2 - f_1) M_j^Q$
           $g^R(P_j^{\sigma^*}, Q_j^{\sigma^*}, R_j^{\sigma^*}) \leftarrow \frac{1}{\mu}(f_2 - f_1) M_j^R$
9:     Compute the next update for tuning parameters
           $P_{j+1}^{\sigma^*} \leftarrow \pi_{\mathbb{S}}(P_j^{\sigma^*} - h_j g^P(P_j^{\sigma^*}, Q_j^{\sigma^*}, R_j^{\sigma^*}))$
           $Q_{j+1}^{\sigma^*} \leftarrow \pi_{\mathbb{S}}(Q_j^{\sigma^*} - h_j g^Q(P_j^{\sigma^*}, Q_j^{\sigma^*}, R_j^{\sigma^*}))$
           $R_{j+1}^{\sigma^*} \leftarrow \pi_{\mathbb{S}_+}(R_j^{\sigma^*} - h_j g^R(P_j^{\sigma^*}, Q_j^{\sigma^*}, R_j^{\sigma^*}))$
10: end for
11: Return $\{\hat{P}^{\sigma^*}, \hat{Q}^{\sigma^*}, \hat{R}^{\sigma^*}\}$.
```

and $\pi_{\mathbb{S}_+}$ correspond to the Euclidean projection onto $\mathbb{S}$ and $\mathbb{S}_+$, respectively, and the parameter h_j denotes the step size. We compute the projections $\pi_{\mathbb{S}}$ and $\pi_{\mathbb{S}_+}$ by following well known results in (Higham, 1988). Let $K = V \Lambda V^\top$ be the eigenvalue decomposition of a matrix K. Then, $\pi_{\mathbb{S}}(K) \triangleq V \max\{0, \Lambda\} V^\top$ and $\pi_{\mathbb{S}_+}(K) \triangleq V \max\{d, \Lambda\} V^\top$ for some $d \in \mathbb{R}_{>0}$. To generate the random symmetric matrices M_j^P, M_j^Q, and M_j^R at each iteration $j \in \mathbb{N}_0$, we construct matrices of the form $M_j^\star = U_j^\star L_j^\star (U_j^\star)^\top$, $\star \in \{P, Q, R\}$, where $L_j^\star$ is a diagonal matrix with diagonal elements uniformly chosen from an interval of interest, and $U_j^\star$ is a random unitary matrix constructed using the technique described in (Ozols, 2009). Lastly, the output of the algorithm $\{\hat{P}^{\sigma^*}, \hat{Q}^{\sigma^*}, \hat{R}^{\sigma^*}\}$ is defined as

$$\{\hat{P}^{\sigma^*}, \hat{Q}^{\sigma^*}, \hat{R}^{\sigma^*}\} \triangleq \operatorname*{argmin}_{(X,Y,Z) \in \{(P_0^{\sigma^*}, Q_0^{\sigma^*}, R_0^{\sigma^*}), \ldots, (P_J^{\sigma^*}, Q_J^{\sigma^*}, R_J^{\sigma^*})\}} f(X, Y, Z).$$

The results in (Nesterov and Spokoiny, 2017) guarantee convergence of the sequence $\{P_j^{\sigma^*}, Q_j^{\sigma^*}, R_j^{\sigma^*}\}$ to some local minimiser $\{\mathbf{P}^*, \mathbf{Q}^*, \mathbf{R}^*\}$ given a proper choice of the algorithm parameters. Particularly, by (Nesterov and Spokoiny, 2017, Theorem 6), if we pick $\mu \leq \frac{\epsilon}{2L_0(f)\sqrt{n}}$ and $h_j = \frac{G}{(n+4)\sqrt{j+1}L_0(f)}$, the accuracy $\mathbf{E}\{f(\hat{P}^{\sigma^*}, \hat{Q}^{\sigma^*}, \hat{R}^{\sigma^*})\} - \mathbf{f}^* \leq \epsilon$ is guaranteed by Algorithm 1 in

$$J = \frac{4(n+4)^2}{\epsilon^2} L_0(f)^2 G^2, \qquad (3)$$

iterations, where n is the number of tuning parameters (see Remark 2), $L_0(f)$ is the Lipschitz constant for the cost function f which can be computed numerically as

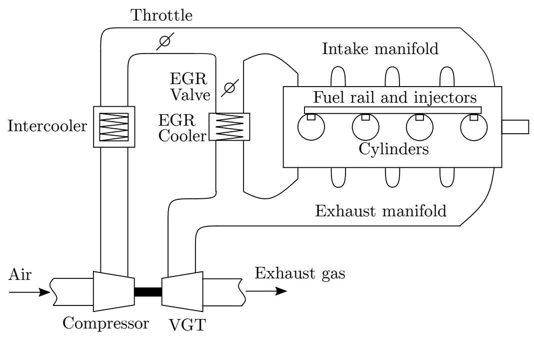

Fig. 2. A diesel engine air-path schematic.

in e.g. (Bunin and François, 2016), $G \triangleq G_P + G_Q + G_R$ where G_P, G_Q and G_R are such that $\|P_0^{\sigma^*} - \mathbf{P}^*\|_2 \le G_P$, $\|Q_0^{\sigma^*} - \mathbf{Q}^*\|_2 \le G_Q$, and $\|R_0^{\sigma^*} - \mathbf{R}^*\|_2 \le G_R$ (these can also be estimated numerically).

Remark 3. Note that, at each iteration step j of Algorithm 1, the oracles are computed once the entire closed-loop response is available. Particularly, two experiments (or simulations) are required to compute these oracles, one with parameters $\{P_j^{\sigma^*} + \mu M_j^P, Q_j^{\sigma^*} + \mu M_j^Q, R_j^{\sigma^*} + \mu M_j^R\}$ and one with $\{P_j^{\sigma^*}, Q_j^{\sigma^*}, R_j^{\sigma^*}\}$. Once the two closed-loop responses are finished, then Algorithm 1 computes the oracles and the next update $\{P_{j+1}^{\sigma^*}, Q_{j+1}^{\sigma^*}, R_{j+1}^{\sigma^*}\}$. Two new closed-loop experiments are then performed with these new controller parameters and the process continues iteratively in this fashion.

4. GFO TUNING FOR A DIESEL ENGINE AIR-PATH

4.1 Diesel engine air-path model

A schematic representation of the diesel air-path is shown in Fig. 2. By following (Sankar et al., 2019), the engine operating range is divided into twelve regions, as shown in Fig. 3, with a linear model representing the engine dynamics in each region. The model state is chosen to consist in the intake manifold pressure p_{im} (also known as boost pressure), the exhaust manifold pressure p_{em}, the compressor flow W_{comp}, and the EGR rate y_{EGR}. The input consists of the throttle position u_{thr}, EGR valve position u_{EGR}, and VGT valve position u_{VGT}. Lastly, the measured output consists in $(p_{\text{im}}, y_{\text{EGR}})$.

For a given operating point parametrised by the engine speed ω_e and fuel rate $\dot{m}_f$, $(\omega_e^\sigma, \dot{m}_f^\sigma)$, $\sigma \in \{1, \dots, 12\}$, the engine control unit applies certain steady-state actuator values. We denote the steady-state values of the input, state, and output by $\bar{u}^\sigma \in \mathbb{R}^3$, $\bar{x}^\sigma \in \mathbb{R}^4$, and $\bar{y}^\sigma \in \mathbb{R}^2$, respectively, at each operating condition $(\omega_e^\sigma, \dot{m}_f^\sigma)$. By following the system identification procedure detailed in (Shekhar et al., 2017), we get a model for the diesel engine air-path in the form of (1), that is

$$\tilde{x}_{k+1} = A^\sigma \tilde{x}_k + B^\sigma \tilde{u}_k, \tag{4a}$$

$$\tilde{y}_k = C^\sigma \tilde{x}_k + D^\sigma \tilde{u}_k, \tag{4b}$$

where $\sigma \in \{1, \dots, 12\}$, $\tilde{x}_k = x_k - \bar{x}^\sigma$ is the perturbed state around the corresponding steady-state $\bar{x}^\sigma$, $\tilde{u}_k = u_k - \bar{u}^\sigma$ is the perturbed input around the corresponding steady-state input $\bar{u}^\sigma$, and $\tilde{y}_k = y_k - \bar{y}^\sigma$ is the perturbed output around the corresponding steady-state output $\bar{y}^\sigma$.

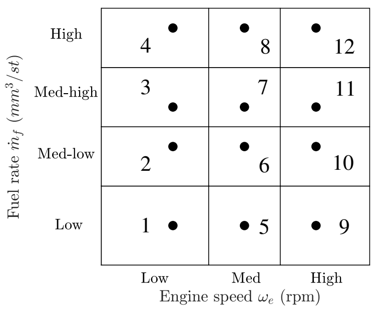
High
Med-high
Med-low
Low
Fuel rate $\dot{m}_f$ (mm^3/st)
4
8
12
3
7
11
2
6
10
1
5
9
Low
Med
High
Engine speed ω_e (rpm)

Fig. 3. Engine operational space divisions and the corresponding linearisation points.

The LTI-MPC strategy described in Section 2 is used to control the engine air-path. In the following sections, we illustrate the performance of our tuning framework when tuning the MPC parameters. Note that $\tilde{x}_k \in \mathbb{R}^4, \tilde{u} \in \mathbb{R}^3$, and $\tilde{y} \in \mathbb{R}^2$, then the number of parameters to be tuned per controller are $n = 26$, which follows from Remark 2.

4.2 Numerical simulations

We now perform offline tuning of MPC paremeters using GFO. That is, we implement the scheme in Fig. 1, where the plant block is composed of an engine model. We will test the GFO tuning framework when the engine model is being controlled over a drive-cycle. Drive cycles are used to assess the performance of vehicles by means of a schedule over the vehicle's speed. We restrict our attention to the urban drive cycle (UDC), but this framework can be applied to any drive cycle of choice.

We pick an initial calibration such that $\{P^1, Q^1, R^1\} = \dots = \{P^{12}, Q^{12}, R^{12}\} = \{P_0, Q_0, R_0\}$, with $P_0 = Q_0 = \text{diag}\{0.01, 0, 0.2, 0.01\}$ and $R_0 = 10^{-5} \cdot I_{3\times3}$. We use the tracking error as the measure of performance in the remainder of this paper, that is, we define

$$f(P^{\sigma^*}, Q^{\sigma^*}, R^{\sigma^*}) \triangleq \frac{1}{\sqrt{N}} \sqrt{\sum_{k=0}^{N-1} \left\| y_k(P^{\sigma^*}, Q^{\sigma^*}, R^{\sigma^*}) - y_k^{\text{ref}} \right\|_2^2},$$

where $N \in \mathbb{N}$ is the experiment (or simulation) length, y_k^{ref} is the vector containing the boost pressure and EGR rate references, respectively, and $y_k(P^{\sigma^*}, Q^{\sigma^*}, R^{\sigma^*})$ is the process measured output when the σ^*-th controller is using tuning parameters $P^{\sigma^*}, Q^{\sigma^*}, R^{\sigma^*}$, and the rest of controllers are using $\{P_0, Q_0, R_0\}$.

The response of the engine model over the UDC using the initial calibration $\{P_0, Q_0, R_0\}$ can be seen in Fig. 4. We have encircled parts of the drive cycle with unsatisfactory performance. We identify that this is associated with controller $\sigma^* = 6$, and we have also used light grey to shade regions of the drive cycle in which controller $\sigma^* = 6$ is acting. We then fine tune this controller by running Algorithm 1 with parameters $\mu = 10^{-9}, h_j = 10^{-6}/\sqrt{j+1}$ as per Section 3.2 ($L_0(f) = 10^6, \epsilon = 0.01, R = 30$). For $J = 15$ iterations, the resulting tuned matrices are given by

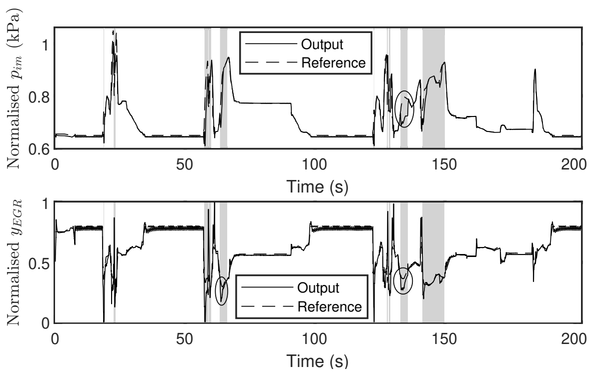

Fig. 4. Engine model response over the UDC drive cycle using the initial calibration $\{P_0, Q_0, R_0\}$.

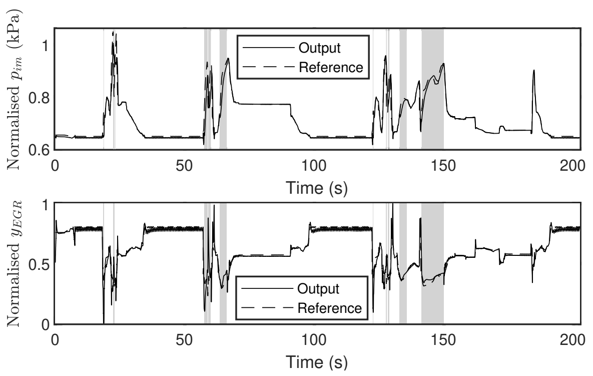

Fig. 5. Engine model response over the UDC drive cycle using the GFO-tuned $\{\hat{P}^6, \hat{Q}^6, \hat{R}^6\}$ for controller six, and $\{P_0, Q_0, R_0\}$ for the remaining controllers.

$$\hat{P}^6 = \begin{bmatrix} 68.7777 & -3.9213 & -7.9706 & 23.4118 \\ -3.9213 & 77.6175 & 22.2896 & 7.1257 \\ -7.9706 & 22.2896 & 68.7995 & 2.3130 \\ 23.4118 & 7.1257 & 2.3130 & 77.2866 \end{bmatrix},$$

$$\hat{Q}^6 = \begin{bmatrix} 86.2979 & 18.2538 & 3.3847 & -0.0318 \\ 18.2538 & 68.6798 & 13.5989 & 22.1637 \\ 3.3847 & 13.5989 & 81.6394 & -3.0665 \\ -0.0318 & 22.1637 & -3.0665 & 73.2421 \end{bmatrix},$$

$$\hat{R}^6 = \begin{bmatrix} 81.0912 & 2.7255 & 9.4127 \\ 2.7255 & 53.5783 & -6.1731 \\ 9.4127 & -6.1731 & 10.8003 \end{bmatrix}.$$

The response of the engine model over the UDC using the GFO tuned matrices is shown in Fig. 5. The improvement in tracking performance has significantly improved in the regions where controller six is acting. Particularly, the initial calibration provides a tracking error of $f(P_0, Q_0, R_0) = 3.5652$, and the final calibration in which controller six has been tuned using the GFO framework provides a tracking error of $f(\hat{P}^6, \hat{Q}^6, \hat{R}^6) = 3.0391$. This corresponds to a 14.8% performance improvement. The cost function values for the tuned controller per iteration are given in Fig. 6. We can see that convergence can be achieved within a few number of iterations in this simulation, which leads us to think that the theoretical guarantee (3) is quite conservative. In fact, the theoretical number of iterations that ensure convergence is $J = 3.24 \cdot 10^{22}$ (see (3)), which is really conservative.

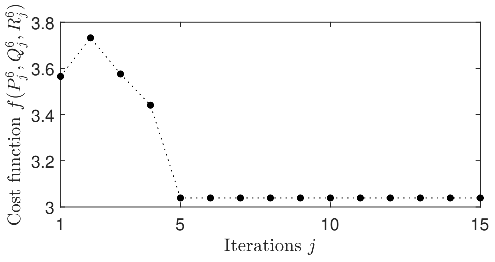

Fig. 6. Convergence of the cost function $f(P_j^6, Q_j^6, R_j^6)$.

4.3 Experimental implementation

We now perform online GFO tuning in-the-loop with a physical diesel engine. That is, we use the architecture of Fig. 1 in which the engine block is a real diesel engine at Toyota's Higashi-Fuji Technical Center, and the GFO minimiser is using real-time drive cycle data coming from the closed-loop experiment in the dashed box. We note that the experiments were performed using the switching LTI-MPC in Section 2 but with an added integrator augmentation to deal with offsets exhibited in the real engine (see (Rawlings and Mayne, 2009, Section 1.5.2) for more details). In addition, since each iteration of Algorithm 1 requires two experiments (cf. Remark 3), it is useful to reduce the experimental length. For this experiment we focus on performing only the third (last) segment of the UDC drive cycle. The engine outputs with the MPC initial calibration $Q_0 = \text{diag}\{0.01, 0.01, 0.2, 1\}, P_0 = Q_0, R_0 = I_{3\times 3}$ are given in Fig. 7. We have encircled the areas with unsatisfactory tracking performance, which are related to controller $\sigma^* = 6$, and we have also used light grey to shade regions in which this controller is acting. The algorithm parameters used in this experiment are $\mu = 2.5 \cdot 10^{-6}$ and $h_j = 10^{-5}/\sqrt{j+1}$ as per Section 3.2 ($L_0(f) = 4000, \epsilon = 0.1, R = 1.2$). The GFO-tuned matrices for $J = 8$ iterations are given by

$$\hat{P}^6 = \begin{bmatrix} 0.0502 & -0.0117 & -0.0398 & 0.0057 \\ -0.0117 & 0.0102 & 0.0146 & 0.0118 \\ -0.0398 & 0.0146 & 0.2003 & -0.0426 \\ 0.0057 & 0.0118 & -0.0426 & 1.0022 \end{bmatrix},$$

$$\hat{Q}^6 = \begin{bmatrix} 0.4699 & 0.0708 & 0.0192 & 0.0901 \\ 0.0708 & 0.2329 & -0.1597 & -0.1684 \\ 0.0192 & -0.1597 & 0.3459 & 0.0016 \\ 0.0901 & -0.1684 & 0.0016 & 1.1911 \end{bmatrix},$$

$$\hat{R}^6 = \begin{bmatrix} 1.1227 & -0.1419 & 0.0305 \\ -0.1419 & 1.0621 & 0.0534 \\ 0.0305 & 0.0534 & 1.2843 \end{bmatrix}.$$

The engine response with GFO-tuned matrices is given in Fig. 8, where we can see a clear improvement in the reference tracking for controller six. Particularly, the tuned matrices provide an improvement of performance of 7.73% with only eight iterations. As in the simulations carried in Section 4.2, this illustrates the conservativeness of the theoretical convergence guarantee (3) from (Nesterov and Spokoiny, 2017). The relatively fast convergence of the algorithm in this high dimensional space maintains a degree of hope for tuning with hardware-in-the-loop over complicated drive cycles, even though the theoretical degrees are conservative.

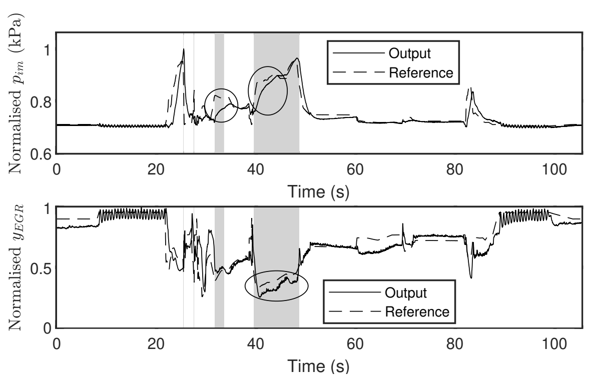

Fig. 7. Real engine response over the third UDC segment using the initial calibration $\{P_0, Q_0, R_0\}$.

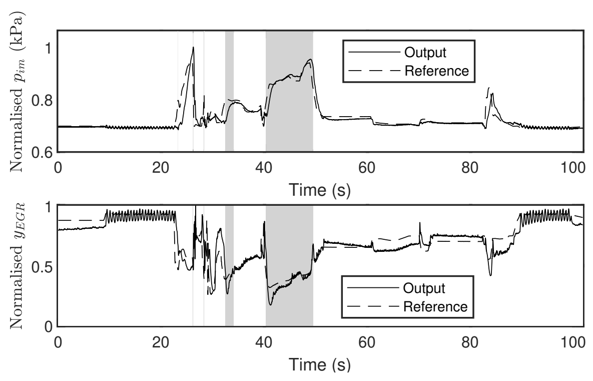

Fig. 8. Real engine response over the third UDC segment using the GFO-tuned $\{\hat{P}^6, \hat{Q}^6, \hat{R}^6\}$ for controller six, and $\{P_0, Q_0, R_0\}$ for the remaining controllers.

5. CONCLUSION

This paper described a gradient-free strategy to successfully tune MPC parameters iteratively within the control loop. We applied this method to air-path control in diesel engines both in simulations and experiments. The observed nature of the tuning algorithm in practice belies the apparent conservative nature of the theoretical results, and therefore creates potential for the development of efficient tuning tools for advanced controllers (and potentially even retuning online). Insights on how to choose the algorithm parameters is left for future work.

ACKNOWLEDGEMENTS

The authors thank the engineering staff at Toyota Higashi-Fuji Technical Centre, Susono, Japan, for their assistance in running the experiments related to this work.

REFERENCES

Bagheri, P. and Khaki-Sedigh, A. (2014). An analytical tuning approach to multivariable model predictive controllers. *Journal of Process Control*, 24(12), 41–54.

Bunin, G.A. and François, G. (2016). Lipschitz constants in experimental optimization. *arXiv preprint arXiv:1603.07847*.

Bunin, G.A., Tirado, F.F., François, G., and Bonvin, D. (2012). Run-to-run MPC tuning via gradient descent. In *Computer Aided Chemical Engineering*, volume 30, 927–931. Elsevier.

Garriga, J.L. and Soroush, M. (2010). Model predictive control tuning methods: A review. *Industrial & Engineering Chemistry Research*, 49(8), 3505–3515.

Higham, N.J. (1988). Computing a nearest symmetric positive semidefinite matrix. *Linear algebra and its applications*, 103, 103–118.

Ira, A.S., Shames, I., Manzie, C., Chin, R., Nešić, D., Nakada, H., and Sano, T. (2018). A machine learning approach for tuning model predictive controllers. In *2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, 2003–2008. IEEE.

Júnior, G.A.N., Martins, M.A., and Kalid, R. (2014). A pso-based optimal tuning strategy for constrained multivariable predictive controllers with model uncertainty. *ISA transactions*, 53(2), 560–567.

Mayne, D., Rawling, J., Rao, C., and Scokaert, P. (2000). Constrained model predictive control: Stability and optimality. *Automatica*, 36, 789–814.

Morari, M. and Lee, J.H. (1999). Model predictive control: past, present and future. *Computers & Chemical Engineering*, 23(4-5), 667–682.

Nesterov, Y. and Spokoiny, V. (2017). Random gradient-free minimization of convex functions. *Foundations of Computational Mathematics*, 17(2), 527–566.

Ozols, M. (2009). How to generate a random unitary matrix.

Rawlings, J.B. and Mayne, D.Q. (2009). *Model predictive control: Theory and design.* Nob Hill Pub.

Sankar, G.S., Shekhar, R.C., Manzie, C., Sano, T., and Nakada, H. (2019). Fast calibration of a robust model predictive controller for diesel engine airpath. *IEEE Transactions on Control Systems Technology.*

Santos, J.E.W., Trierweiler, J.O., and Farenzena, M. (2019). Robust tuning for classical mpc through the multi-scenarios approach. *Industrial & Engineering Chemistry Research*, 58(8), 3146–3158.

Shah, G. and Engell, S. (2011). Tuning MPC for desired closed-loop performance for mimo systems. In *Proceedings of the 2011 American Control Conference*, 4404–4409. IEEE.

Shekhar, R.C., Sankar, G.S., Manzie, C., and Nakada, H. (2017). Efficient calibration of real-time model-based controllers for diesel engines–Part I: Approach and drive cycle results. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, 843–848. IEEE.

Shridhar, R. and Cooper, D.J. (1998). A tuning strategy for unconstrained multivariable model predictive control. *Industrial & engineering chemistry research*, 37(10), 4003–4016.

Suzuki, R., Kawai, F., Ito, H., Nakazawa, C., Fukuyama, Y., and Aiyoshi, E. (2007). Automatic tuning of model predictive control using particle swarm optimization. In *2007 IEEE Swarm Intelligence Symposium*, 221–226. IEEE.

Van der Lee, J., Svrcek, W., and Young, B. (2008). A tuning algorithm for model predictive controllers based on genetic algorithms and fuzzy decision making. *ISA transactions*, 47(1), 53–59.

Yamashita, A.S., Alexandre, P.M., Zanin, A.C., and Odloak, D. (2016). Reference trajectory tuning of model predictive control. *Control Engineering Practice*, 50, 1–11.