Optimal DC Motor Control with a Switch-Mode Power Supply – Simulation

Numerical Optimal Control in Science and Engineering
Project Report (WS 2025-2026)
Author: Antti Paasi
Supervisor: Andrea Ghezzi

***Abstract*— Optimal control schemes can potentially improve energy efficiency and extend the lifespan of power electronic converters. In this project, an approach for optimizing switch-mode power supply usage is tested. A DC motor model is controlled with a Buck converter, while preventing aggressive switching frequency changes. Three optimal control schemes are simulated: Linear Quadratic Regulator, Open Loop Optimal Control and Model Predictive Control. All scripts are implemented in MATLAB. CasADi and Control System Toolbox are used.**

Contents

I. Introduction

In this project, a DC motor controlled with a Buck converter is modelled, and three different optimal control methods are simulated in MATLAB. The shaft speed of the DC motor is forced to follow a reference signal while preventing aggressive control changes. The motivation for implementing optimal control is the hypothesis that restricting switching speed changes could extend converter lifespan and improve energy efficiency.

After deriving a discrete-time model, LQR control is simulated with a constant reference speed. Then, more complex references are used, and open loop optimal control is simulated. Lastly, MPC with a Kalman filter is simulated as an enhanced and applicable version of the second scheme. All MATLAB scripts are available in the GitHub repository [1].

II. DC Motor and Buck Converter System Model

A continuous-time state-space model for a DC-motor is derived in [2]. The model is given by equations

$$\dot{x} = \begin{bmatrix} -\frac{b}{J} & -\frac{K}{J} \\ -\frac{K}{L} & -\frac{R}{L} \end{bmatrix} x + \begin{bmatrix} 0 \\ 1/\mathrm{L} \end{bmatrix} u, \quad (1.1)$$

$$y = [1 \quad 0]x, \quad (1.2)$$

where $x = \begin{bmatrix} \dot{\theta} \\ i \end{bmatrix}$ is the state vector containing the motor shaft angular speed $\dot{\theta}$ and armature current i. The control signal $u = V_I \cdot D$, where V_I is the converter input voltage and D is the duty cycle, which is explained subsequently. Other parameters are shaft friction coefficient b, shaft moment of inertia J, electromotive force constant K, motor inductance L, and motor resistance R.

We use a simplified model for the Buck converter, because we want to use duty cycle as the control signal. According to [3, p. 6], the continuous steady-state model for Buck converter output voltage is

$$V_O = (V_I - V_{DS}) \text{ x } D - V_D \text{ x } (1 - \mathrm{D}) \, I_L \text{ x } R_L, \quad (2)$$

where V_I and D are again the converter input voltage and duty cycle, V_{DS}, V_D and I_L x R_L are voltage drops in the circuit. However, we simplify further by assuming an ideal circuit, leading to

$$V_o = V_I D. \quad (3)$$

Parameters values are not chosen from an actual motor datasheet. We copy values for J and b from [2]. Parameters R and L are calculated by using the second order damping model, because we want a fast-enough step response with no overshoot. In principle, we could design the motor circuit to have certain resistance and inductance values. The DC motor transfer function from [2] is derived to the second order damping form with variables R and L. The damping coefficient is set to ζ=1 (critically damped). Values for R and L, as well as the resulting natural frequency ω_n are then solved. A step-by-step solution is available in the GitHub repository [1]. The goal of this project is to practice optimal control, not precise motor modeling, so this approach is acceptable. Parameter values used in the project are represented in table 1.

TABLE I. SYSTEM PARAMETER VALUES

Parameter	Description	Unit	Value
J	Shaft moment of Inertia	kg m^2	0.01
b	Shaft Friction Coefficient	N ms	0.1
K	Electromotice Force Constant	Vs	0.7
R	Motor Resistance	Ω	5.1
L	Motor Inductance	H	0.09
V_I	Converter Input Voltage	V	24
T_s	Sampling Time	s	0.01

As shown in table 1, sampling time $T_s = 0.01$ is used. We use the forward-Euler discretization method presented in [4, p. 50], because our sample time is sufficiently short. Finally, our discrete-time system model is

$$x(k+1) = \begin{bmatrix} 0.9000 & 0.7000 \\ -0.0778 & 0.4333 \end{bmatrix} \begin{bmatrix} \dot{\theta}(k) \\ i(k) \end{bmatrix} + \begin{bmatrix} 0 \\ 0.1111 \end{bmatrix} \cdot 24 \cdot D(k) \quad (4.1)$$

$$y(k) = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} \dot{\theta}(k) \\ i(k) \end{bmatrix}. \quad (4.2)$$

The step response of the discretized system is displayed in figure 1.

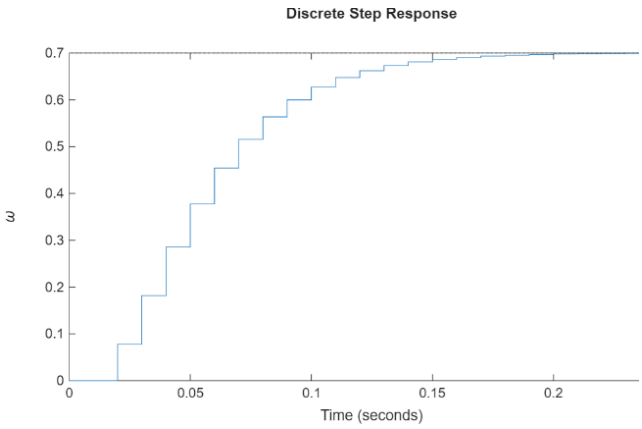

Fig. 1. Discrete-time Step Response of the System

The discrete-time system has a double pole at 0.6667 in the Z-domain, which lies inside the complex unit circle. Therefore, the system is stable.

III. LINEAR QUADRATIC REGULATOR

Theory regarding the LQR implementation was learned from [5, p.126] and the feedback loop idea from [6]. A basic principle called the Bryson's rule is used to tune the state weight and control weight matrices. This rule is stated in the LQR Design documentation by Mathworks [7]. Bryson's rule says that the matrices should be tuned according to

$$Q_{i,i} = \frac{1}{(maximum\ state\ error)^2}, \quad i \in \{1,2,\dots,n\}. \quad (5)$$

Values $Q_{i,i}$ are set to the diagonal, other values are zero. In this project, applying the rule leads to the matrices

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

$$R = [100].$$

The matrix Q is the state weight matrix and R is the control weight matrix, here a scalar. Because our goal is to minimize converter switching changes, we define using controls as being more costly than state errors. In the simulation script, the optimal feedback control matrix K is calculated with MATLAB function `dlqr`.

The idea is to linearize the system around a constant set point. The simulation is run for a 0.3s timeframe. Three different situations with different initial values and goal shaft speeds, marked with ω_g, are simulated. Uniformly distributed noise is added to the state vector in each timestep to mimic real world behavior. The noise is defined with MATLAB command

```
randi([-1000,1000])*0.00005*ones(2,1),
```

which leads to deviations from calculated system state in the range of 10e-3. This is a fairly small error, which is handled well by the feedback loop. A higher level of noise would demand a filter. Both using higher noise levels and filtering noisy measurements are tested in the next chapters.

Due to the control signal being limited between values 0 and 1, the motor saturation speed is around 17 1/s. Unfeasible controls are clipped in the simulation. Results are presented in figures 2 and 3.

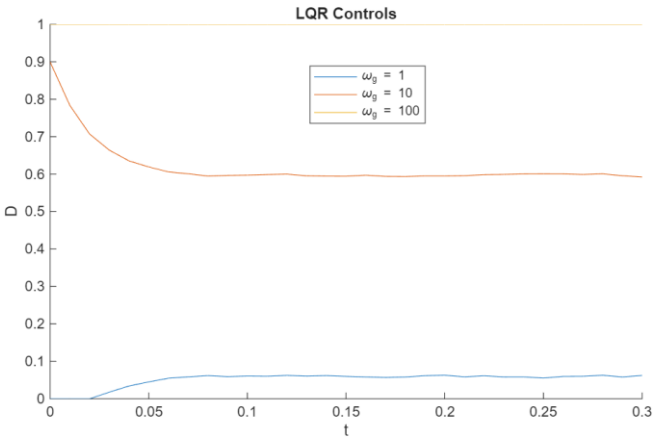

Fig. 2. LQR controls for different initial values and set points. Time on x-axis, duty cycle on y-axis.

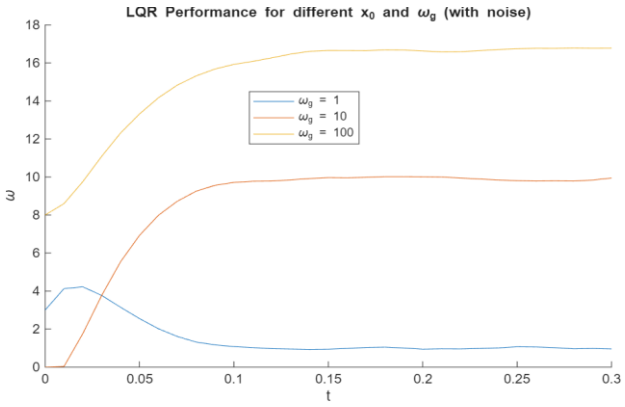

Fig. 3. LQR performance for different set points and initial values.

Here, the LQR scheme results in a relatively fast regulation with soft controls. This kind of regulation can be useful for applications with set points that don't change rapidly. It is worth noting that a well-tuned PI controller would probably perform equally well here.

IV. Open Loop Optimal Control

Now we want to dive deeper into optimization and solve an open loop optimal control problem with CasADi and its `qpsol` solver. The solver finds an optimal way of controlling the system for one timeframe so that the motor shaft speed closely follows a more complicated reference signal. The objective function to be solved is

$$\min \sum_{k=0}^{N-1} [\omega_e(k)^T \cdot q \cdot \omega_e(k) + D(k)^T \cdot r \cdot D(k)] + \omega_e(N)^T \cdot q \cdot \omega_e(N) \quad (6.1)$$

s.t.

$$x(0) = x_0, \quad (6.3)$$

$$d(0) = d_0, \quad (6.4)$$

$$x(k+1) = A_d \cdot x(k) + B_d V_l \, D(k), \quad (6.5)$$

$$0 \leq D(k) \leq 1, \quad (6.6)$$

$$-dd_{max} \leq D(k) - D(k-1) \leq dd_{max} \quad (6.7)$$

$$\omega(N) - ref(N) = 0. \quad (6.8)$$

Parameters and variables are explained below.

$\omega_e(k) = ref(k) - \omega(k)$ is shaft speed error compared to reference speed signal,

$x_0 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ is the initial state ,
$d_0 = 0$ is the initial control,
$D(k)$ is the duty cycle,
$dd_{max} = 0.07$ is the maximum control change in one timestep (in code, marked as `max_dd`),
$q = [1000]$ is the state weighting matrix (here, a scalar) and
$r = [100]$ is the control weighting matrix (here, a scalar).

Sequential approach is used for implementation, so only $D(k), k = 1,2,3, \dots, N-1$ are decision variables. The matrices are tuned using the same principle as in chapter III, equation (5). Emphasis is again more on the control cost. In addition, we add constraints for maximum change of control per timestep. We mark this as `max_dd = 0.07`. Because the control signal (converter duty cycle) is between 0 and 1, the physically feasible maximum change would be 1. This parameter choice greatly prevents aggressive control.

Performance is simulated with added noise to mimic real world behavior. After calculating optimal controls with the `qpsol` solver from CasADi, uniformly distributed noise between -1 and 1 is added to each state. The noise in each timestep is defined with MATLAB command

```
randi([-1000,1000])*1e-3*ones(2,1),
```

which leads to both state variables having error in the range of ± [0,1] in each timestep. As demonstrated in figure 4, open loop control does not perform well when deviations from the system model occur.

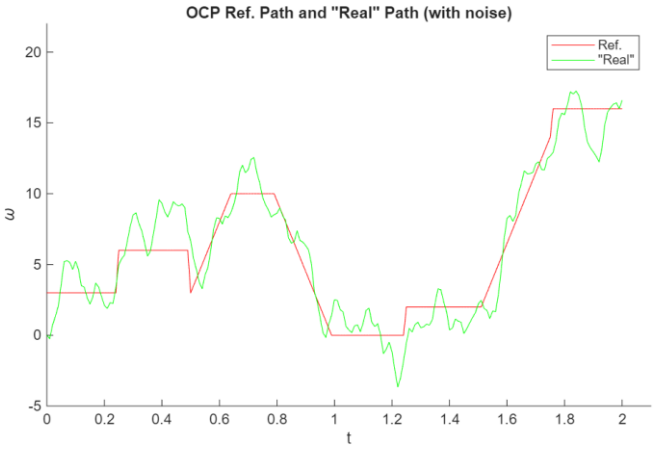

Fig. 4. Open Loop Control Performance Against Reference Signal

It is important to note that the "real" path in figure 4 is not actually physically feasible, due to rapid changes in speed and the excess of motor maximum speed. However, this simulation demonstrates that without feedback, the precise control of the system is difficult, because slight behavioral deviations from the system model cause big errors. This problem is solved in the next chapter by adding feedback and using a filter to smoothen the control actions.

V. Model Predictive Control with Kalman Filter

Theory regarding MPC was learned from [6, p.261-266] and [7]. A more practical example was studied with documentation and scripts from [8]. In MPC simulation, the objective function and constraints are now

$$\min \sum_{k=0}^{N} [\, \omega_e(k)^T \cdot q \cdot \omega_e(k) + D(k)^T \cdot r \cdot D(k) \,] \quad (7.1)$$

s.t. (7.2)

$$x(0) = x_0 \quad (7.3)$$

$$D(0) = d_0 \quad (7.4)$$

$$x(k+1) = A_d \cdot x(k) + V_I\, D(k), \tag{7.5}$$

$$0 \le D(k) \le 1 \tag{7.6}$$

$$-dd_{max} \le D(k) - D(k-1) \le dd_{max}\,, \tag{7.7}$$

Terminal constraint is not used, because in this case, it causes the solver to fail. The difference to the open loop problem is, we now solve many optimization problems in a loop. In this simulation, `control_window = 5` is used, which means that we only calculate controls for 5 time points (0.05s forward) in each iteration. Only the first control is implemented after each solution. In each iteration, the solver takes the latest state and the latest control as initial parameters. The latter is important, because of the `max_dd` restriction.

Again, noise is used to disturb the system. The noise is of the same distribution as before and it is again added to the calculated system state in each timestep. This time noise is added to both the calculated next state and the current state (adding noise both to the process and the measurement). If the noise level is low, MPC and open loop controller perform similarly. However, when noise level grows, both start to deviate from the desired path. With open loop, the explanation is simple: no feedback leads to no knowledge of the occurring deviation, and thus controls are not changed. For MPC, the controller thinks that the noisy measurements are the actual system state. This leads to the controller aggressively trying to get back on the desired path, which does not work too well either. In this project, the goal is to use as soft controls as possible and to follow the reference path. As a solution, a Kalman filter is utilized to filter process and measurement noise, so that the controller gets better estimates of the actual state of the system.

Kalman filter assumes normally distributed noise, while here the noise is uniformly distributed. The resulting state estimate is still useful. It also reflects a realistic situation where the exact noise distribution is not known. Simulation results without the filter are shown in figure 5, and simulation results with the filter are shown subsequently in figure 6. By a trial and error approach, the following Kalman filter parameters where chosen.

$$Q_k = \begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \end{bmatrix},$$

$$P_k = \begin{bmatrix} 10 & 0 \\ 0 & 10 \end{bmatrix} \text{ and}$$

$$R_k = [10],$$

Q_k being the state covariance matrix, P_k is the process noise covariance matrix and R_k is the measurement noise covariance matrix, here a scalar.

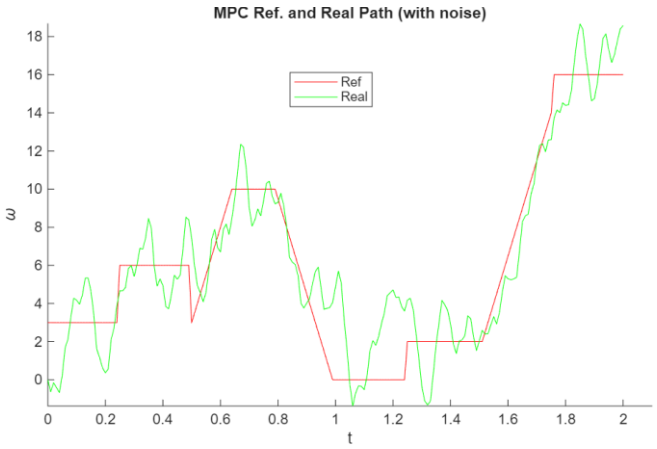

Fig. 5. MPC Performance without Kalman filter

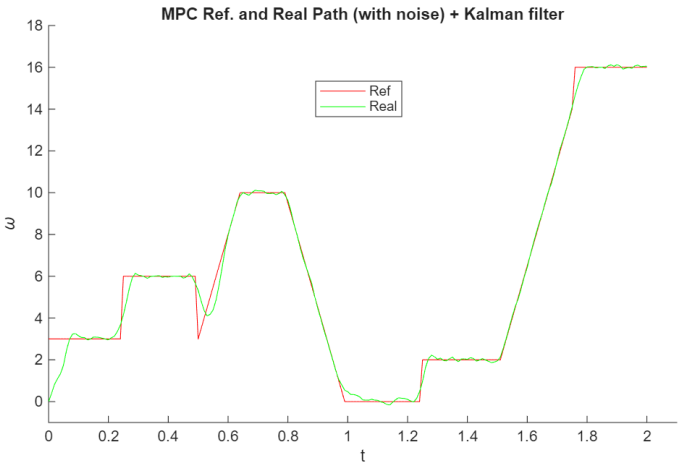

Fig. 6. MPC performance with Kalman filter

When no filter is utilized, MPC performs similarly to the open loop controller. On the contrary, when Kalman filter is used, MPC performs well. Even though the noise causes reasonably big deviations from the calculated system state, the MPC controller is still able to control the motor in a desired way. Noise is often a problem in real applications, therefore a well-tuned filter is a necessity. It is also worth adding that this type of filtering is only usable with feedback control, so we could not utilize a Kalman filter in the open loop solution.

The simulation time is also an interesting result. Simulating the 2s timeframe with MPC and Kalman filter only takes about 0.2s to complete, so in principle the solver is fast enough for real world applications. This topic is briefly discussed in the next chapter.

VI. Summary

In this project, a model for a DC Motor and converter system was derived and three different optimal control methods were simulated. The goal was to force the motor to follow a reference speed signal while preventing aggressive control changes, because of the converter lifespan lengthening and energy saving goals. In terms of performance, we saw that LQR is a suitable solution for relatively simple reference signals. MPC with Kalman filter is superior to open loop optimal control when noise is present, as is often the case in

the real world. A natural next step would be to deploy these schemes in real-time control in hardware and test if they really extend the lifespan and improve energy efficiency of a power electronic converter. The controller algorithm could be, for example, first converted to embeddable C code with the acados interface.

REFERENCES

[1] GitHub repository, Optimal-DC-Motor-Control-Simulation, https://github.com/anttipaasi/Optimal-DC-Motor-Control-Simulation

[2] DC Motor Speed: System Modeling, University of Michigan, https://ctms.engin.umich.edu/CTMS/index.php?example=MotorSpeed&section=SystemModeling

[3] Understanding Buck Power Stages in Switchmode Power Supplies, Application Report by Texas Instruments, 1999, https://www.ti.com/lit/an/slva057/slva057.pdf

[4] Model predictive control of high power converters and industrial drives, Geyer, Tobias, 2017

[5] Numerical Optimal Control (Draft), Diehl, Moritz, Gros, Sébastien, 2024. Available online at https://www.syscop.de/files/2024ws/NOC/book-NOCSE.pdf

[6] Numerical Optimal Control in Science and Engineering, exercise sheet 7, available at the course website https://www.syscop.de/teaching/ss2020/numerical-optimal-control-online

[7] Linear-Quadratic Regulator (LQR) design, Mathworks, Look at Q - state cost weight matrix definition), https://se.mathworks.com/help/control/ref/lti.lqr.html

[8] Model Predictive Control – Theory, Computation and Design, Rawlings, James B., Mayne, David Q., Diehl, Moritz M., 2024

[9] OCP Playground, GitHub repository by Florian Messerer, https://github.com/fmesserer/ocp-playground

[10] Kalman filter explained simply, https://thekalmanfilter.com/kalman-filter-explained-simply/