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

Antti Paasi

University of Freiburg

Course “Numerical Optimal Control in Science and Engineering”

edited on 7.1.2026

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

Introduction 1

DC Motor and Buck Converter System Model 1

Linear Quadratic Regulator 3

Open Loop Optimal Control 4

Model Predictive Control with Kalman Filter 5

Summary 7

Sources 8

Use of AI

ChatGPT was used to

- Review the project idea
- Review ready scripts
- Review of the Abstract and Introduction chapters.

The author sees no sense in having AI complete things for you in a university project. Therefore, it was only used for brainstorming and reviewing ideas before asking the supervisor.

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. 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].

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

$$\dot{\mathrm{x}} = \begin{bmatrix} \frac{-b}{\mathrm{J}} & \frac{-K}{J} \\ \frac{-K}{L} & \frac{-R}{L} \end{bmatrix} \begin{bmatrix} \dot{\theta} \\ i \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{L} \end{bmatrix} u$$

$$y = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} \dot{\theta} \\ i \end{bmatrix}, \text{ where}$$

$x = \begin{bmatrix} \dot{\theta} \\ i \end{bmatrix}$ is the state vector containing the motor shaft angular speed $\dot{\theta}$ and armature current i. The control signal $u = V_I D$, where V_I is the converter input voltage and D is the duty cycle, 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

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

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

$$V_o = V_I D.$$

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 $\zeta = 1$ (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 1: System Parameter Values

Parameter	Description	Unit	Value
J	Shaft moment of inertia	$kg\ m^2$	0.01
b	Shaft friction coefficient	Nms	0.1
K	Electromotive 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 Ts = 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

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

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

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

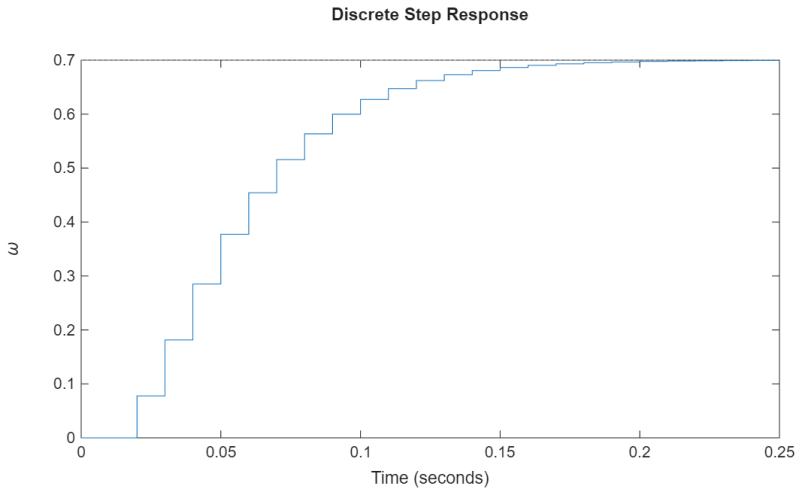

Figure 1: Discrete-time Step Response of the System

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.

Linear Quadratic Regulator

Theory regarding the LQR implementation was learned from [5, p.126] and the feedback loop idea from [6]. Weight matrices are tuned based on Bryson's rule [7]. 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 with simulation time 0.3s. We simulate three different situations with different initial values and goal shaft speeds, marked with ω_g. Some noise is added to mimic real world behavior. Due to the control signal being limited between values 0 and 1, the motor saturation speed is around 17 $\frac{1}{s}$. Unfeasible controls are clipped in the simulation. Results are presented in figures 2 and 3.

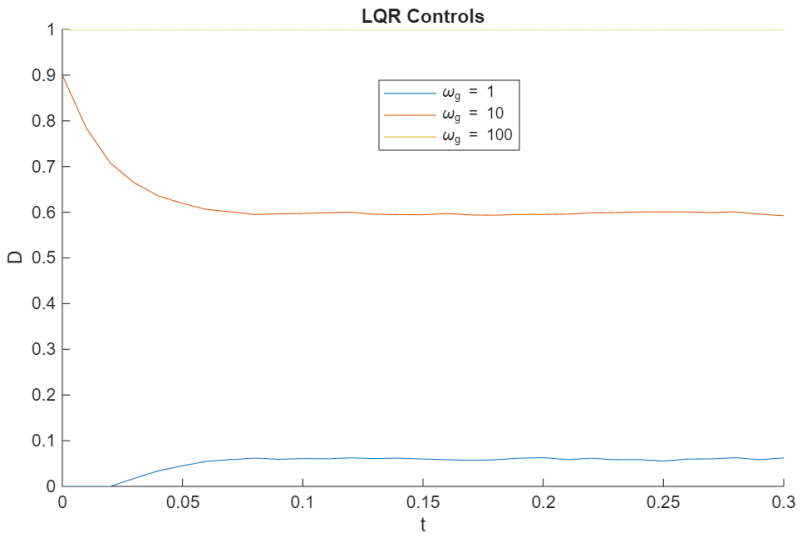

Figure 2: LQR controls for different initial values and set points. Time on x-axis, duty cycle on y-axis.

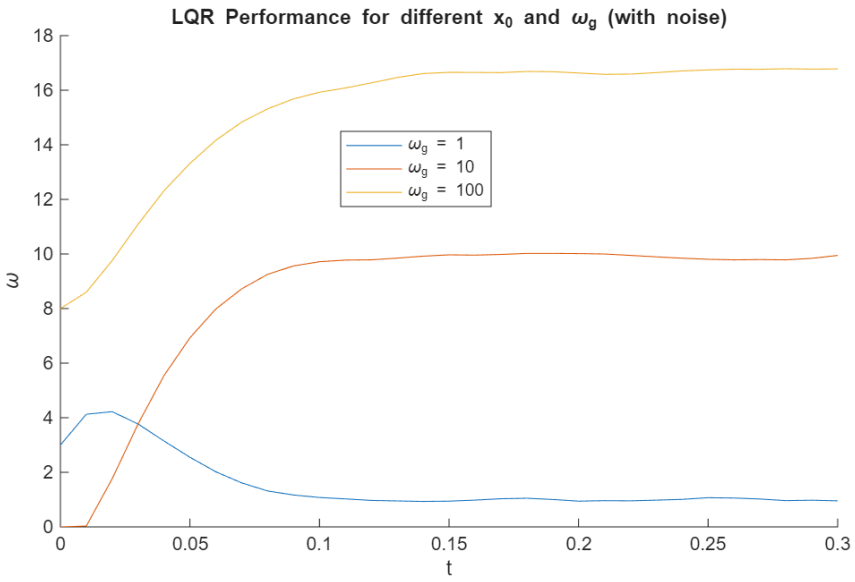

Figure 3: LQR performance for different set points and initial values.

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.

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

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

s.t.

$x(0) = x_0,$

$x(k+1) = A_d * x(k) + B_d V_I D(k),$

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

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

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

Parameters and variables are explained below.

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

x_0 is the initial state $\begin{bmatrix} 0 \\ 0 \end{bmatrix}$

$D(k)$ 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. Again, we tune the matrices q and r with Bryson's rule [7]. We add more emphasis on 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 physical maximum change would be 1. This parameter choice greatly prevents aggressive control.

Performance is simulated with added noise to mimic real world behavior. Noise is defined as a uniform distribution between -1 and 1, as MATLAB command

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

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

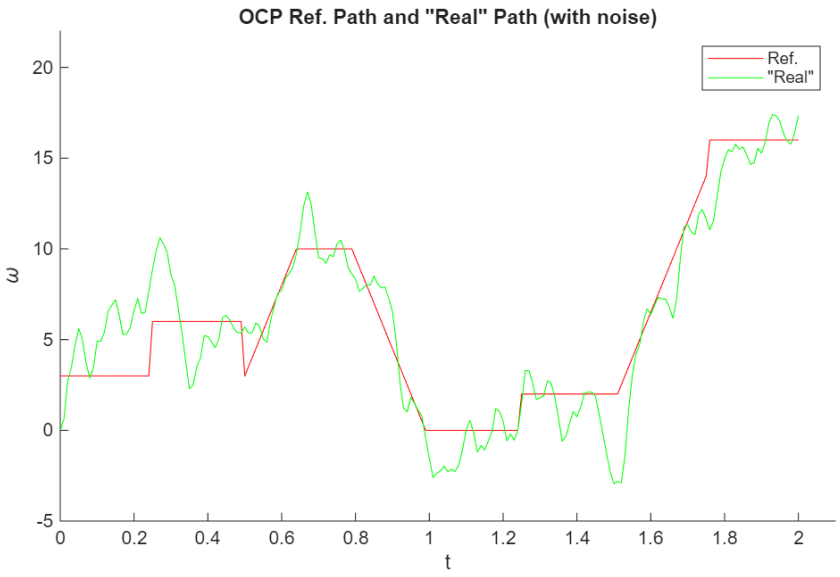

Figure 4: Open Loop Control Performance Against Reference Signal

It is important to note that the "real" path in figure X is not actually physically feasible, due to rapid changes in speed and the excess of motor maximum speed. The point is, without feedback and filtering, the actual system state is very uncertain. This problem is solved in the next chapter.

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

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

s.t.

$x(0) = x_0$

$D(0) = d_0$

$x(k+1) = A_d * x(k) + B_d V_I \, D(k),$

$0 \le D(k) \le 1,$

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

The difference is, we now solve 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) 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. We add noise to both the measurement and the process output. Unintuitively to the author, MPC performs very badly when noise levels grow. The reason is, if the controller thinks the states are all over the place, it will start applying controls very aggressively, and because of `max_dd` , the resulting performance is bad. Of course, we wouldn't want aggressive controls in this project anyway.

This problem is solved with a Kalman filter. The application principle was learned from [10]. The filter tries to estimate how much the measured state is "lying" because of noise. It outputs an estimate of the actual state. This state is then used in feedback. The exact uniform distribution of the noise is known in simulation, which obviously isn't the case in the real world, but a little twist is added by the fact that Kalman filter assumes a normal distribution. Simulation results are shown in figure 5.

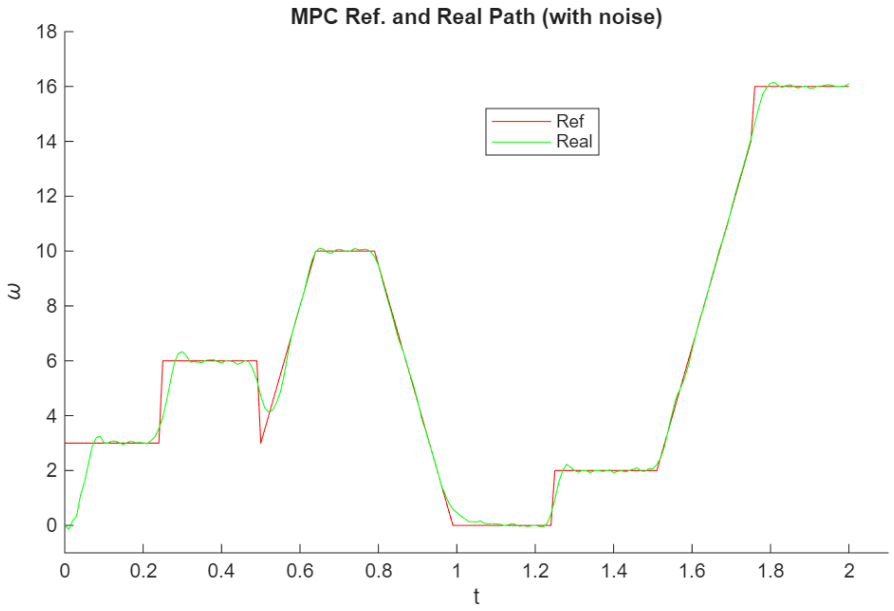

Figure 5: MPC Performance Against Reference Path

Notice how much better the reference is followed compared to the open loop simulation. Also, the simulation of a 2s timeframe takes only about 0.2s to complete, so in principle the solver is fast enough for real world applications. If the Kalman filter was not applied or it was tuned badly, the shaft speed would jump around the reference. Because of this phenomenon, a well-tuned filter is often a necessity in real world control applications.

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, 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.

Sources

[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, 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/