Final Year Project Report

Data-Driven Control for DC Motors

Oraz Ospanov, Asset Malik, Andrey Yershov

A thesis submitted in part fulfilment of the degree of

BSc in Robotics and Mechatronics

Supervisor: Do Duc Ton

Department of Robotics and Mechatronics
Nazarbayev University

March 21, 2021

Table of Contents

Abstract

This paper describes the work done for the Final Year Project on topic of Data Driven Control for DC motors. Multiple experimental approaches were considered for the project, for instance MATLAB/SIMULINK simulation of the DC motor with different configurations, a setup with an Arduino and DC motor driver and a dSpace DC motor setup. Data on voltage and position relation was collected for all of the mentioned approaches and was analyzed in MATLAB for the Data Driven controller design. The results are to be added later. Ultimate goal of this project is to create a plug-and-play reinforcement learning setup that generates controller for any given DC motor.

Acknowledgments

We would like to thank our project Supervisor Ton Duc Do for support during the project course and providing the lab space and equipment. We thank Mr. Kanat for help with the dSpace setup and comfortable atmosphere in the lab.

Chapter 1: **Introduction**

1.1 Model Based Control

The modern control theory, also known as Model Based Control has been used both for linear and non-linear systems extensively since 1960. The Model Based Control is, as could be guessed from it's name, relies solely on the model of the plant for further controller design. However, there is usually no chance of creating an exact plant model for the system, as well as no efficient way of creation high accuracy plant model using the classical model based control [1]. Moreover, the possible plant model errors have to be considered when designing the controller from the plant model, what requires additional resources and time to provide robustness of the controller. All of the used control design methods heavily rely on the assumptions of an accurate plant model design. This means that in case if unmodeled dynamics occurs, the controller is believed to be unable to provide further handling of the dynamics. As could be seen in Fig. 1.1, the Model Based Controller design starts from the assumptions about the model plant and is directed at regulation of the plant [1].

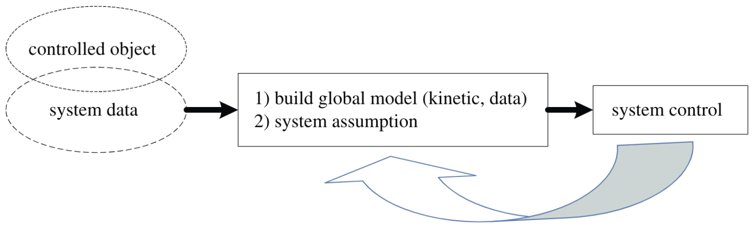


Figure 1.1: Common structure of the Model Based Control [1]

1.2 Data Driven Control

Opposed to the Model Based Control, Data Driven control is becoming more popular. [1] suggest the following definition for Data-Driven control: "Data-driven control includes all control theories and methods in which the controller is designed by directly using on-line or off-line I/O data of the controlled system or knowledge from the data processing but not any explicit information from mathematical model of the controlled process, and whose stability, convergence, and robustness can be guaranteed by rigorous mathematical analysis under certain reasonable assumptions". Data-Driven control for various purposes is becoming the superior method in systems where the model is not available or is changing over the course of life cycle, i.e. adaptability is necessary. This implies that motors and other motile systems are the ones which need Data-Driven control the most. Data-Driven controller design method allows to bypass the tedious system identification step. However, as every machine learning problem, it requires big amounts of data. This becomes less of a problem year by year, as every industry, be it information technology or metallurgy, is becoming more and more digitized and thus produces vast amounts of data. This data has a vast potential of improving the current processes if processed correctly. The data gathered from the system behaviour is used to identify the controller. In this paper, we investigate the application of data-driven control to the control of DC motor drives and experiment with data-driven controller estimation using reinforcement learning. As it could be seen on the figure 1.2, [2] outlined the Data-Driven Control workflow in a cycle figure.

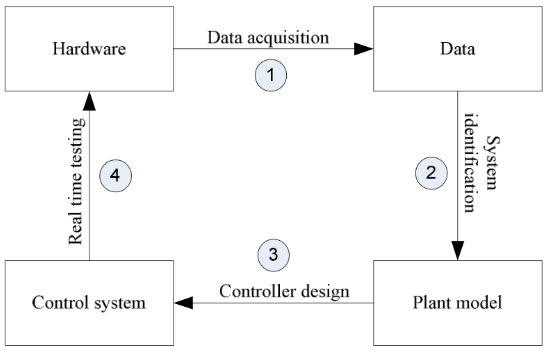


Figure 1.2: Data-Driven Control setup workflow [2]

1.3 Motivation for Data-Driven control for DC motors

DC motors are an inseparable part of modern motile systems in engineering and robotic system control. They are efficient and are able to be controlled in various ways. However, there may be different circumstances when the DC motor system is exposed to abrupt load changes, electromagnetic disturbances and highly varying inputs [2]. These circumstances are unlikely to be considered during the factory manufacturing step of the motors, and the Data-Driven controller design could help with understanding the motor behaviour at those circumstances and dealing with them. Further, the Data-Driven controller design for DC motors could be used as a calibration process for DC motors in robotic systems, which has served for a prolonged period of time and experience wear-out issues. For safety considerations, better performance and controllability a robust controller must be designed and maintained for a given DC motor. As was discussed earlier, the recent deep learning achievements provide us with an ability to do so without having the knowledge about the architecture of the DC motor. The controller is designed relying on the data from input voltage and output position.

In this paper, multiple approaches to data-driven control are going to be tested with DC motors. First, the DC motor is going to be simulated using MATLAB/SIMULINK platform in three scenarios: Open Loop DC motor with Pulse Width Modulation (PWM) input, Closed Loop DC motor with known parameters and Closed Loop DC motor with unknown parameters. The difficulty of the controller design thus will increase in 3 steps as well. Next, the plan is to setup the physical model using a simple DC motor with unknown parameters with an encoder (to control its position) and an Arduino Uno connected to the PC as a micro controller for the DC motor driver. With the MATLAB model the data is to be collected with a number of signals generated by Signal Builder tool (step, sine, triangular, nonlinear etc.). The collected data is then to be processed to build a controller for that DC motor. To demonstrate that this approach is applicable to any DC motor system, even the larger and more complicated ones, the physical model is to be tested on the dSpace DC motor.After the data collection process is finished, the data is to be used to design the controller. The initial aim of this project is to compare the available approaches, namely the ones provided by MATLAB: System Identification Toolbox and NARX Neural Network. After this is complete, a plug-and-play reinforcement learning setup is to be designed to generate controller for any given DC motor.

Chapter 2: **Related Work**

2.1 Algorithms for Data-Driven control

The article written by Naung et al. [2] is closely related to our work by the research aims and methods. Authors present Data-Driven Control used for system parameters identification, an approach that we are planning to use. The main goal of the paper is to perform system identification utilizing the data driven system control of the DC motor, with ultimate generation of the transfer function suitable for robust control of the plant. For this purpose, authors use MATLAB to create the Simulink model of the DC motor (Figure 2.1) and calculate the proportional-integral-differential (PID) controller according to the obtained models.

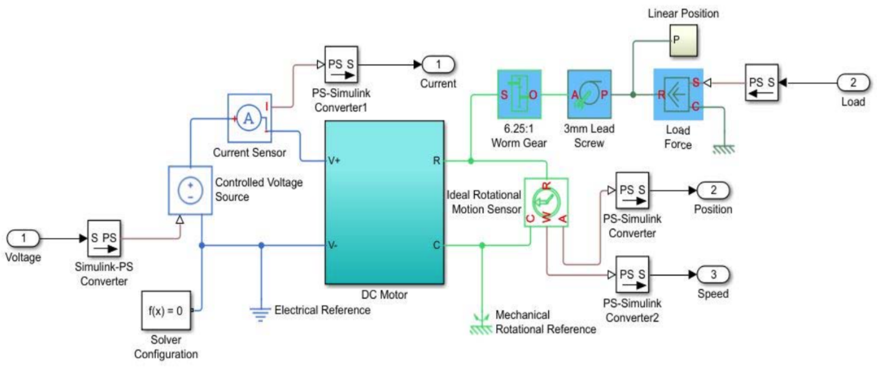


Figure 2.1: Simulink diagram of simscape physical plant DC motor [2].

Authors compare the performance of the nonlinear autoregressive exogenous inputs (NARX) artificial neural network and nonlinear black-box model structures, summarized in Figure 2.11. They conclude that the proposed control method performs with acceptable accuracy and is utilizable with more complex dynamic systems.

Coulson, Lygeros, and Dorfler introduce novel data-enabled predictive control (DeePC) algorithm [3], that utilizes real-time feedback of the system, and is able to drive the system

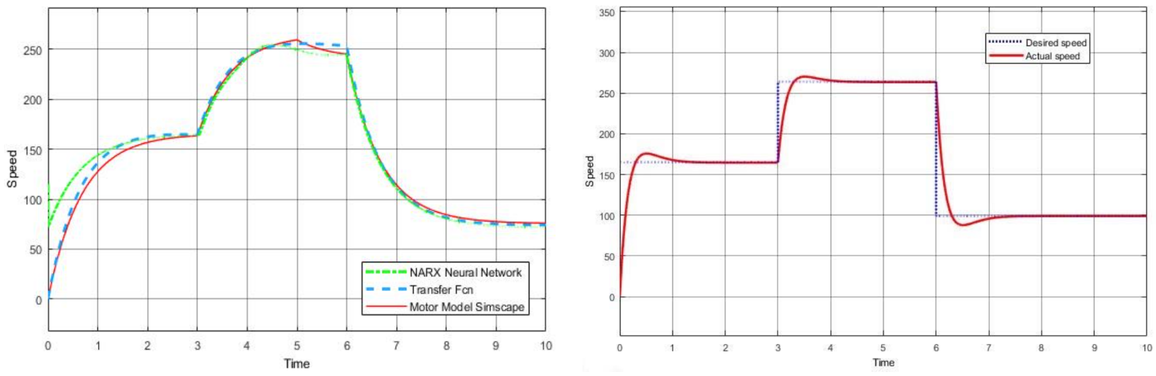


Figure 2.2: Results of the speed response of the DC motor using NARX Neural Network and PID controller [2].

along a desired trajectory with given system constraints. The DeePC algorithm was applied to unknown linear time-invariant (LTI) system and was compared to classical Model Predictive Control (MPC) algorithm. Authors state that the DeePC was created with thinking from a behavioural systems theory perspective. The algorithm description is provided in both mathematical proof form and a pseudo-code description of the algorithm steps. As an example of sophisticated nonlinear application of the DeePC, a case, a regularized version of the algorithm was simulated on stochastic nonlinear quadcopter dynamics, illustrating its capabilities beyond deterministic LTI systems - figure 2.3.

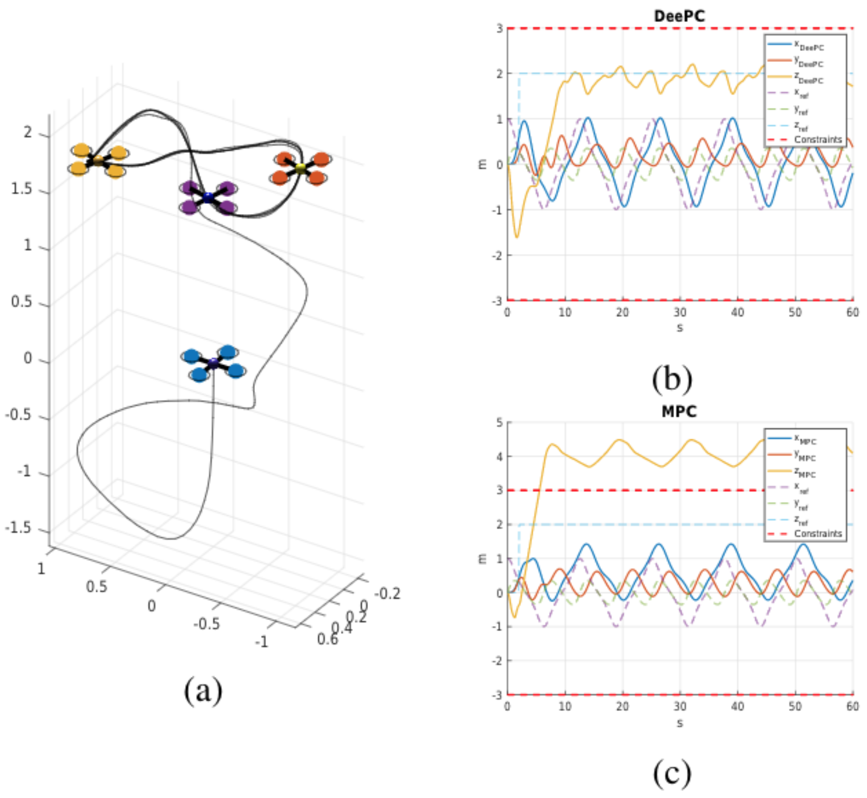


Figure 2.3: Figure (a): three dimensional plot of the trajectory of the quadcopter at different instances of time controlled with DeePC. Figure (b) and (c): trajectories of the spatial coordinates when controlled by DeePC and MPC, respectively. The horizontal red dashed lines represent constraints. [3]

A recent work by Carlet et al. [4] utilizes the aforementioned DeePC algorithm in a setup with motors. This is an important article for our project, as it explores predictive current control for permanent magnet synchronous motor drives - a topic closely related to ours. Authors utilize two of the most popular data-driven algorithms - the Subspace Predictive Control algorithm and the Data-Enabled Predictive Control algorithm. Current controller design procedure for both methods is discussed and presented in detail. Simulation results are reported to be comparably robust and with strong correlation for both of the unconstrained and constrained versions of the data driven methods, as it could be seen in figure 2.4.

Hou and Wang created a canonical survey paper in the field in 2013 [1]. Almost every other related work cites this paper, as it contains fundamental definitions for the Data driven control and its applications. The paper begins with a brief description and comparison of Model based control theory and Data driven control theory. Motivation for the data based control is stated as well. Further, they state the focus of data-driven control theory in terms of objects that could be controlled with the use of it. For the rest of the survey paper authors explore the recent advances in DDC and discuss the perspectives of future related work.

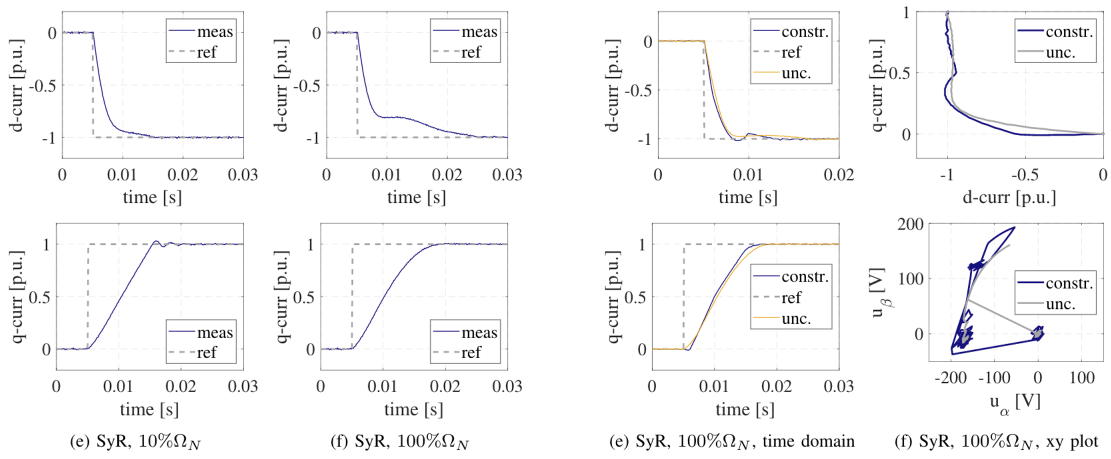


Figure 2.4: Step response of SPC data-driven current controllers [4]

They conclude with categorization of different available Data-driven techniques according to required usage. However, the survey's drawback is that most of the methods are currently outdated, as this is a fast evolving field. The general introductory information provided by the authors is nevertheless valuable.

Next article by Hanke, Wallscheid and Bocker [5] introduces Permanent Magnet Synchronous Motor ripple avoidance using DDC. It also compares three types of linear controllers and the input/output data.

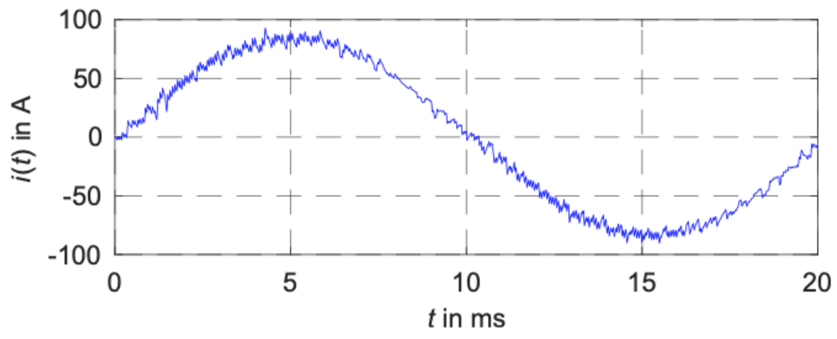


Figure 2.5: CCS-MPC at 60A [5]

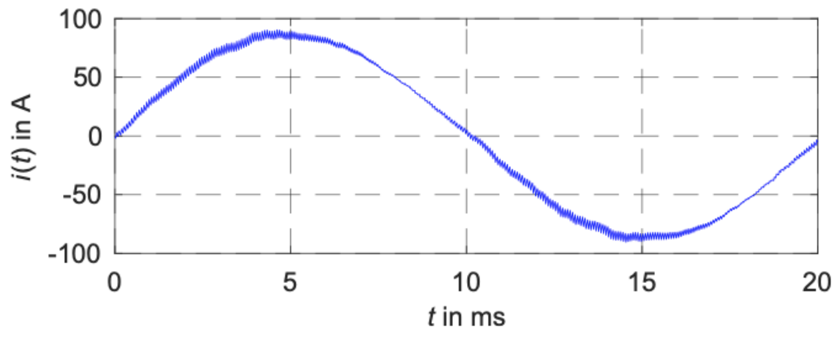


Figure 2.6: FOC at 60A [5]

We can clearly see that total harmonic distortion of the FOC system is smaller and that

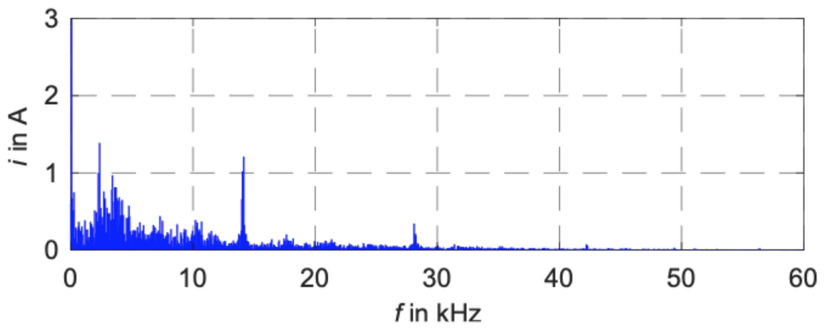


Figure 2.7: CCS-MPC at 60A [5]

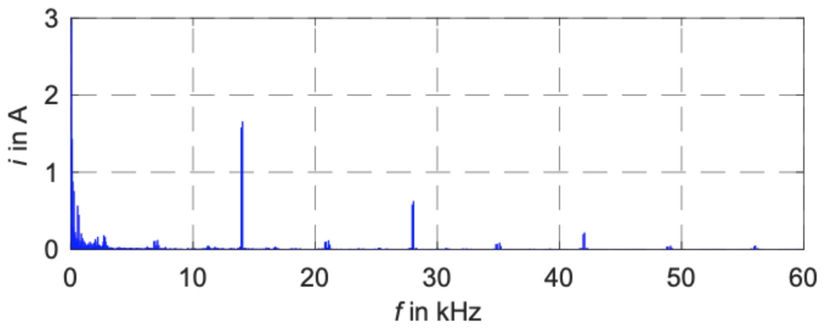


Figure 2.8: FOC at 60A [5]

CCS-MPC only better at those special harmonics of the three phase system.

Article [6] discusses the important attributes for systems with data-driven control: Stabilization, Optimality, and Robustness. Authors argue that it is possible to create a linear approximation of a non linear closed loop system using only data-dependant linear matrix equations.

Article [7] explores the ability of the data-driven system to adapt and thus improve its performance in case of changes of parameters. The author argues that it is possible to have a learning model that could be started without correct initial state and even learn from data without diverging back from better state, which is backed by simulation. Each discreet step of the controller is used as an iteration for the learning algorithm. Model Predictive Control (MPC) was chosen as a back-bone for the closed loop control and re-applied as MPC for Batch processes (iterations) and then replaced by LQR as LMCPC.

2.2 Applications for Data-Driven control

Article [8] analyzes the behavior of data-driven control in the extreme case of output saturation. This paper also uses a linear system approximation approach with dynamic linearization, but also applies assumption that the system has an output saturation. This does increase the tracking performance but decreases the resting time of the closed loop system.

Article [9] is an official MATLAB tutorial to DDC, which we follow with our designed controller simulation in the beginning of our project.

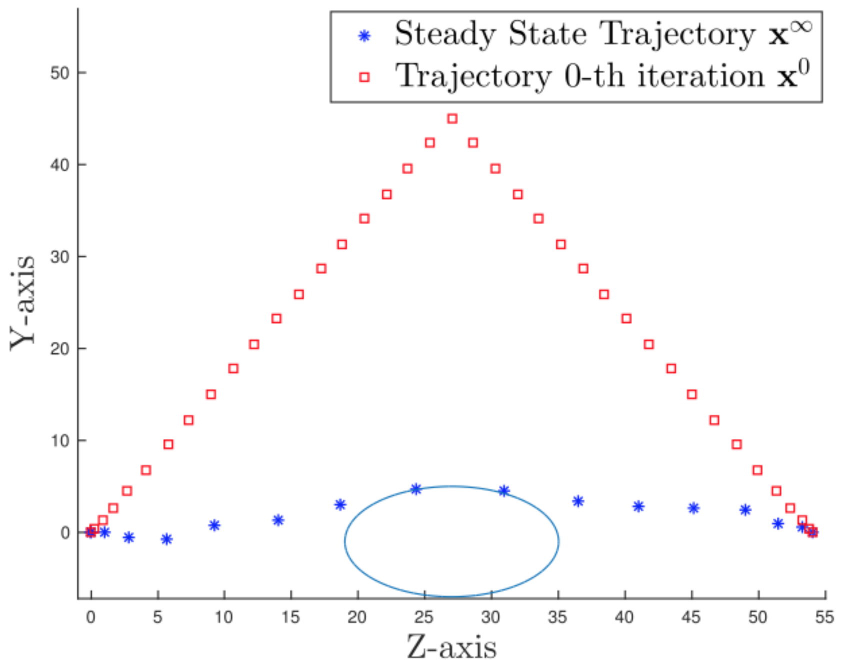


Figure 2.9: SS positions over ZY plane [7]

A recent article [10] overviews a problem of state-feedback controllers for discrete-time linear time-invariant systems, based directly on measured data. This paper presents the robust methods for data-driven control in which the stable closed-loop performance is guaranteed even in presence of noise in data. The design of state-feedback controllers is based on the parametrization technique analyzed in [6], particularly the application of robust control techniques to the parametrization, which results in stable performance of the closed loop control with noisy inputs.

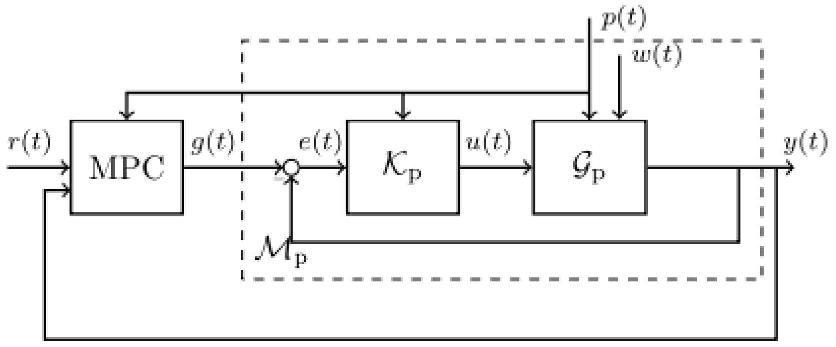


Figure 2.10: Proposed hierarchical control architecture [11]

In the paper [11] a data-driven approach is used to design a hierarchical controller which includes the inner and outer controllers to enhance the overall performance of the inner controller. The evaluation of the effectiveness of this type of method is done by means of simulation and practical experiments. Basically this paper proposes the implementation of the outer Model Predictive controller used for improving the performance of an inner closed loop controller. In such a way it is not only improves the overall performance of the system, but also takes care of constraints imposed on inputs and outputs. Figure 2.10 shows the resultant hierarchical controller, in which it can be seen that the outer Model Predictive controller tracks the input and output variables and operates on the reference signal applied to the inner loop in order to satisfy the constraints imposed on inputs and outputs of an overall

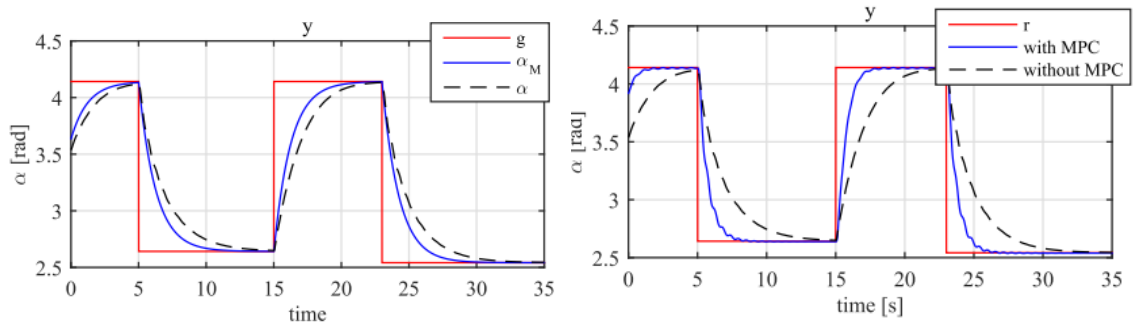


Fig. 4. Example 1: inner loop behavior. Red solid line: reference signal $g(\tau)$. Blue solid line: desired step response of the shaft angle $y_d(\tau)$. Black dashed line: actual controlled output $y(\tau)$.

ig. 6. Example 1: closed-loop behavior. Red solid line: reference signal $r(\tau)$. lue solid line: controlled output $y(\tau)$. Black dashed line: inner-loop output chieved without outer MPC.

Figure 2.11: (a) Inner loop behaviour (b) and Closed loop behaviour with added MPC [11]

system as well as improve on the system performance. To test the results the simulation of a servo positioning system in dc motor was used. The inner loop controller design resulted in the following performance shown in figure 2.10. However because of the lower degrees of freedom of the controller the controller could not obtain the faster dynamics and thus resulted in slow dynamics with the rise time of about 4 s. Adding the model predictive controller resulted in much better performance as shown in figure 2.10. The rise time resulted in 1.3 s, which is about 3 times as much faster as the previous result.

A PhD thesis [12] explores a Data-driven model reference control in the frequency-domain.

An article [13] deals with Data-Driven (DD) control design in a Model Reference (MR) framework.

Article [14] shows a discrete-time control-law from frequency-data of a continuous-time plant so that their hybrid interconnection matches a given continuous-time reference model up to the Nyquist frequency.

Article [15] Data Informativity: A New Perspective on Data-Driven Analysis and Control explores the possible issues with data collected for data-driven control. Authors conclude that data-driven methods are capable of controller design in presense of much less data, when compared to system identification methods.

Overall, it could be concluded that Data-Driven control is becoming the primary focus of the control theory research, as more data and processing powers are available to researchers. This could be noticed by comparing the number of research papers published on the topic for the last years: the amount of publications has tripled since 2015 and the rate of new work on the topic continues rising. The review of related work has acquaint us with such import algorithms as NARX and DeePC, as well as MATLAB's System Identification toolbox and the Signal Builder.

Chapter 3: **Experiment design and MATLAB simulation**

There are two main approaches considered: simulation and hardware implementation. In simulation part the DC motor model obtained from MATLAB Simscape toolbox was used as a plant model, whereas the hardware implementation was based on the simple DC motor of unknown model powered by driver and Arduino. The simulation part is considered to be the experiment design itself, as all of the required steps were first completed in the simulation. This chapter presents the taken steps for the experiment simulation in MATLAB.

3.1 Data generation and collection

The simulink model of a DC motor obtained from the Simscape toolbox is shown in figure 3.1, which is then used to log the input and output data in the setup described in figure 3.2. The default Motor values were used for the first experiment, and later were modified to match those of a real motor to compare the modeled and hardware performance. The signal generator block was used to generate the input signals. Such a setup where the input and output are collected as applied voltage and motor speed is motivated by the desire to check the ability of data-driven control technique to adapt to various models.

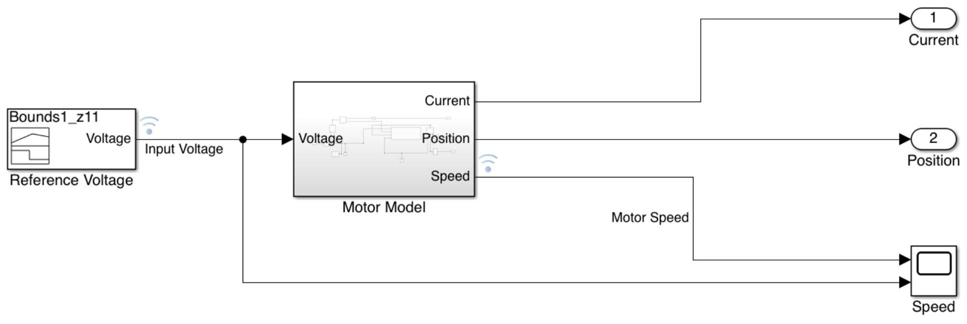


Figure 3.1: MATLAB model for workspace data acquisition Simulink model

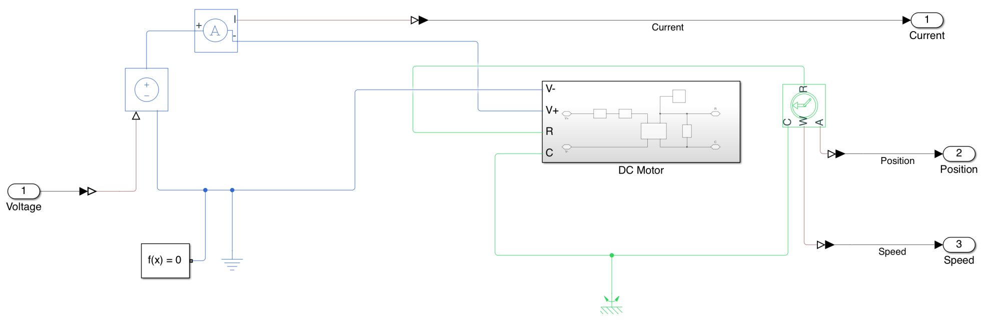


Figure 3.2: MATLAB Simulink Motor Model

The signal generator used in this setup contained twelve signals of various shapes in order to capture the model dynamics as much as possible. Example of the process could be seen in Fig. 3.3.The voltage input to the DC motor is on the bottom graph and the motor position on top.

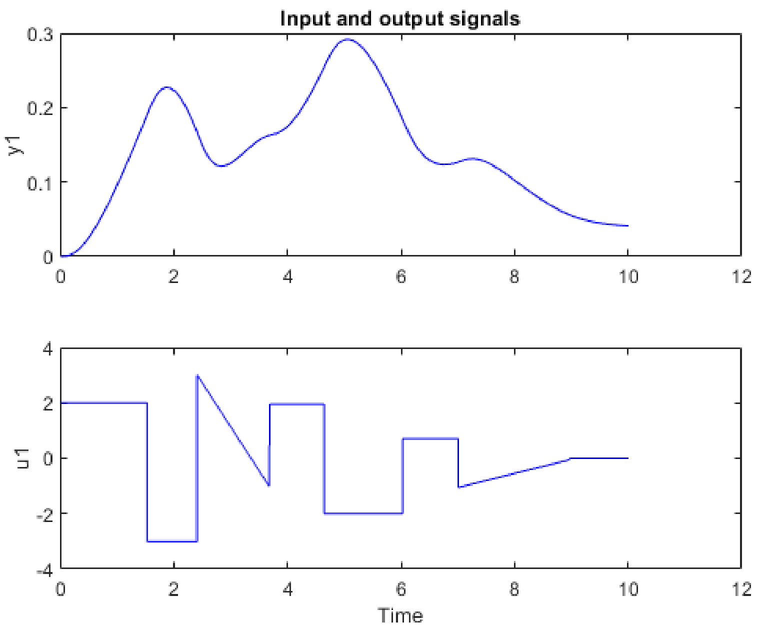


Figure 3.3: One of data collection experiments: voltage input to the DC motor is on the bottom graph and the motor position on top.

All of 12 experiments were then concatenated to a single input-output signal file for further processing.

3.2 Model identification

The model identification step could either be very detailed, long and thus computationally expensive, or it could be done based solely on the required dynamics of the model we are working on. While the first approach has its advantages and has a possible application scenarios (for example in space exploration), we are working on a simpler and faster solution. Thus, the model identification was implemented using the system identification app in MATLAB. The continuous-time transfer function was created based on the acquired data with linear behaviour experiments. Also, NLARX was used for the nonlinear system identification based on the nonlinear experiments (i.e. sine function and other variants). In further experiments, the values for the real, known DC motors were substituted in the simulation model and the resultant transfer function was compared with that of real one.

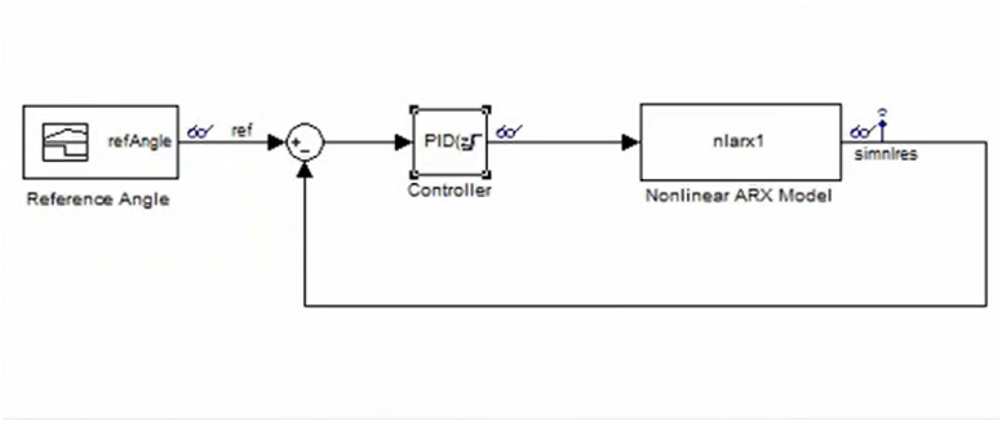


Figure 3.4: MATLAB model for the PID controller design based on the identified model

3.3 PID Controller design

The controller is designed using PID tuner toolbox in a simple setup shown in Fig. 3.4. The Simulink model obtained in previous step is linearized in this step and the controller gains are adjusted accordingly. This step provides an ability to adjust the needed controller characteristics, for example prioritize overshoot removal or decrease the response time. The balance is important in those settings as it provides optimal behaviour accuracy. The designed controller is then tested.

3.4 Testing

Testing is done by supplying a voltage signal to the above mentioned model (Fig. 3.4 with designed controller and tracking the output (DC motor position).

Further tests include the modification characteristics of DC motor model to match those of a real DC motor and compare the performance of the experiment design. The used DC motor model could be seen in Fig. 3.5.

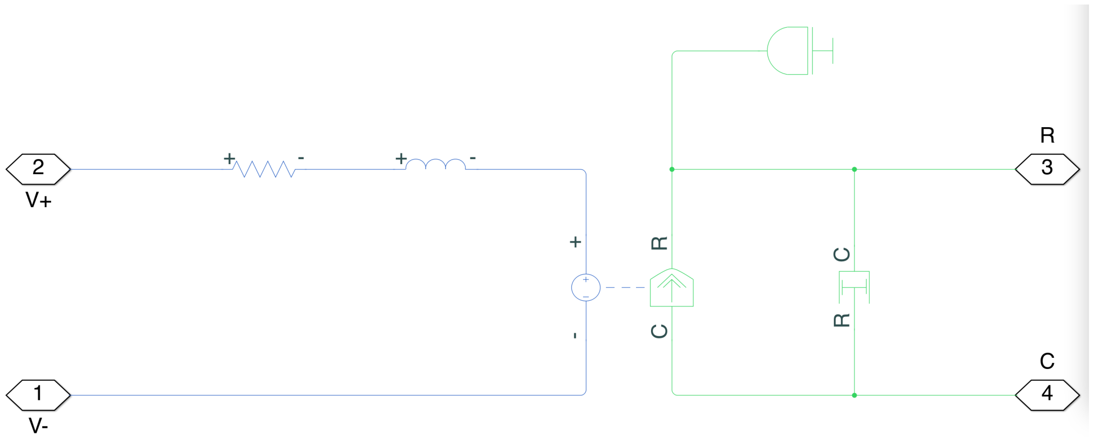


Figure 3.5: MATLAB Simulink DC Motor isolated part

Chapter 4: **Hardware Implementation**

4.1 Arduino

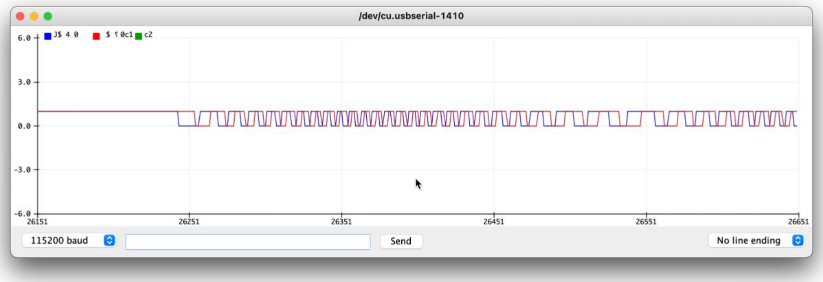
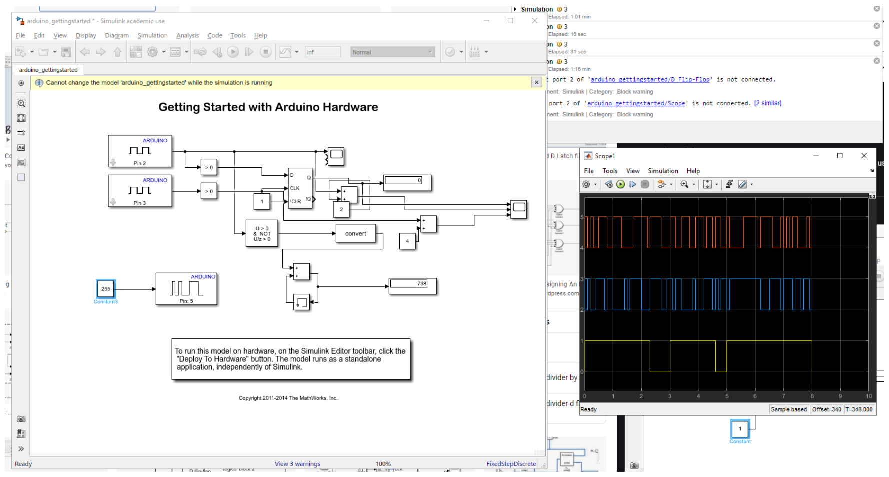


Figure 4.1: Arduino plot for encoder signal

Arduino IDE is a software environment that creates near native machine code with included Arduino.h header. Figure 4.1 shows the incoder signal plotted using Arduino.h's digital read and Serial plot libraries. The signal is produced by rotating rotor with encoder by hand. This might be important because even thought signal looks great, it is plotted on low angular velocity.


arduino_gettingstarted * - Simulink academic use
File Edit View Display Diagram Simulation Analysis Code Tools Help
arduino_gettingstarted
ARDUINO
Pin 2
Pin 3
Pin 5
Constant
convert
Scope1
File Tools View Simulation Help
Ready
View 3 warnings
100%
FixedStepDiscrete
Sample based
Offset=340
T=348.000

Cannot change the model 'arduino_gettingstarted' while the simulation is running

Getting Started with Arduino Hardware

To run this model on hardware, on the Simulink Editor toolbar, click the "Deploy To Hardware" button. The model runs as a standalone application, independently of Simulink.

Copyright 2011-2014 The MathWorks, Inc.

port 2 of 'arduino_gettingstarted/D Flip-Flop' is not connected.

port 2 of 'arduino_gettingstarted/Scope' is not connected. [2 similar]

Figure 4.2: MATLAB Simulink Arduino model

Comparing to a traditional Arduino programming, MATLAB does not provide low level hardware access into the registers and peripherals of the board. Instead it is given as abstract Simulink model blocks with predefined clock and settings.

The Arduino MATLAB hardware setup consists from three major parts: a DC motor, Arduino compatible board (Arduino UNO R3), Encoder. The motor is driven by an external motor driver board. The exact board used is a simple no brand model based on MX1508 chip that also would require external fly-back diodes to operate in more safe conditions, but we have omitted them due to low power nature of chosen motor and just prayed God that we would not fry our personal computers or lab equipment. The driven board seems to have

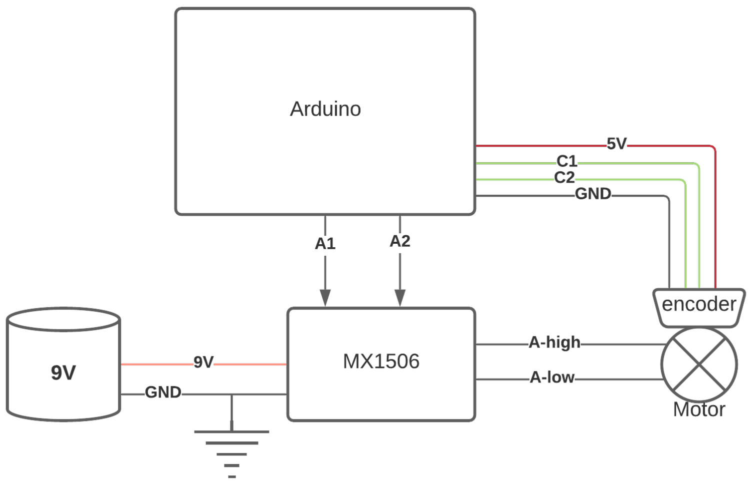


Figure 4.3: Arduino motor hardware setup diagram

no dedicated ground for the signal wires, which is a bit worrying. We did nothing about it, but could just have shared power ground. The MX1508 is a simple and cheap dual channel (A and B) motor driver that can drive up to 2 amperes of current on a voltage range of 2 to 9.6V nominal. we have hooked it up to common 9V alkaline battery, again due to low power nature of the motor, it was believed that this battery will suffice, use more power full power supply otherwise. The motor that we used came with built-in encoder that has two power and two signal terminals that provide variable frequency square wave signal with a phase shift of quarter cycle. This type of encoders are easy to read and can provide both direction and angular velocity of the motor.

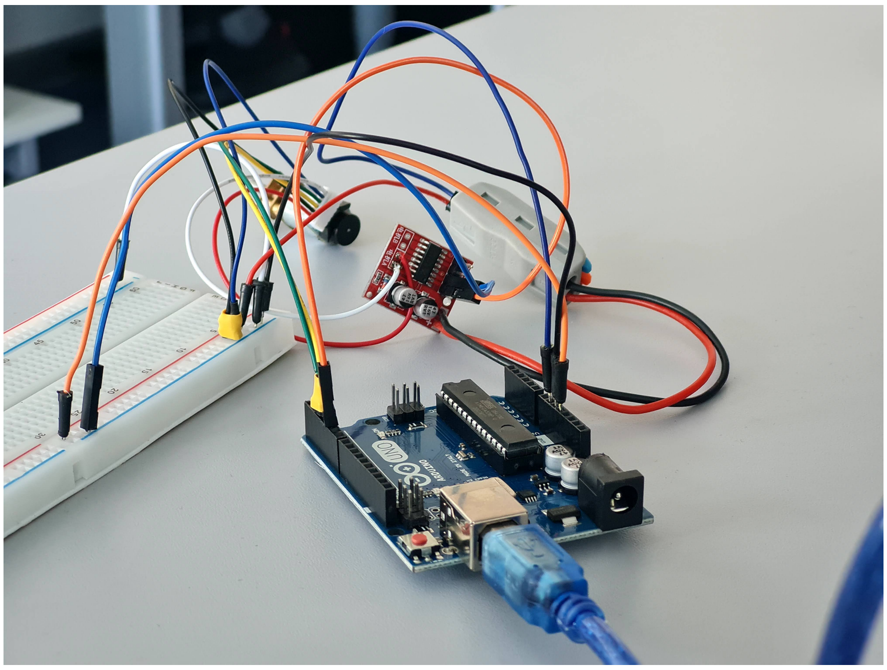


Figure 4.4: Arduino motor hardware setup

After checking encoder signal we assembled a MATLAB Simulink 4.2 model to be able to read encoder's speed and direction. This time rotor was driven by applied voltage on the motor. This way we got stable uniform encoder signal to check the encoder velocity units and direction. We were partially successful. Quick observation shows that even though the motor had constant velocity with no load or oscillations on direction the model showed chaotic alternation in motor direction. After analysing the recorded signal we come to the conclusion that model with Arduino could not keep up with the signal and the samples resulted in aliasing. This was further proven when we tried to increase the sample frequency of the Arduino simulink model. There is a possibility that there is some additional sample rate related parameters of the model that could solve problem with aliasing, but we did not find it and came to a conclusion that Arduino's MATLAB Simulink implementation is just simply not fast enough. There could be made a walk-around by coding the Arduino with its native Arduino IDE to minimize the overhead and set up custom bare minimum communication method over virtual com port UART.

4.1.1 Different motor setup

After we have failed to read the encoder's positional data from Arduino using MATLAB we have decided to try to use another type of motor 4.5. Even thought it is still essentially simple DC motor, it has a high rate gears and feedback potentiometer. Typically this type of servo motors are used as open loop controllers with no feedback what so ever. But we decidede to open it up and solder Arduino's internal analog to digital converter right on the middle terminal of the potentiometer.

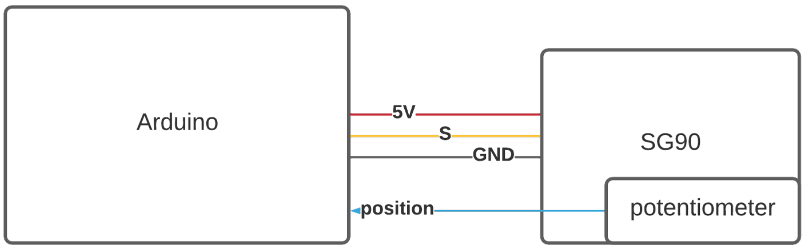


Figure 4.5: Arduino with hardware setup model (model2)

4.2 dSpace

The dSpace is a tool that comes as PCIe expansion card. The card is designed to be connected with external PCB inverter that runs the motor and ADC. It interfaces peripherals as MATLAB Simulink block that interfaces with motor simulation models. There is also a 3rd party monitoring software to record and view the build model.

The dspace system consists of four parts: the DSpace "ControlDesk" software, DSpace controller board, power electronics board containing the dc-dc converter necessary to drive the motor, and the last the DC motor with the specifications shown in figure 4.6. The overall setup is also shown in figure 4.6. Overall, the dSpace controller board sends the control signal which represent the amplitude and phase of the PWM signal to the power board and the board in turn generates the according PWM signal which is sent to motor. The motor position is obtained from the Quadrature encoder embedded in the motor. Also the motor current also can be sent as a feedback to further control processing through the ADC. As can be seen from the motor specifications the maximum voltage applied is 42V. The simulink model with which we directly interact shown on the right part of figure 4.7 transforms the

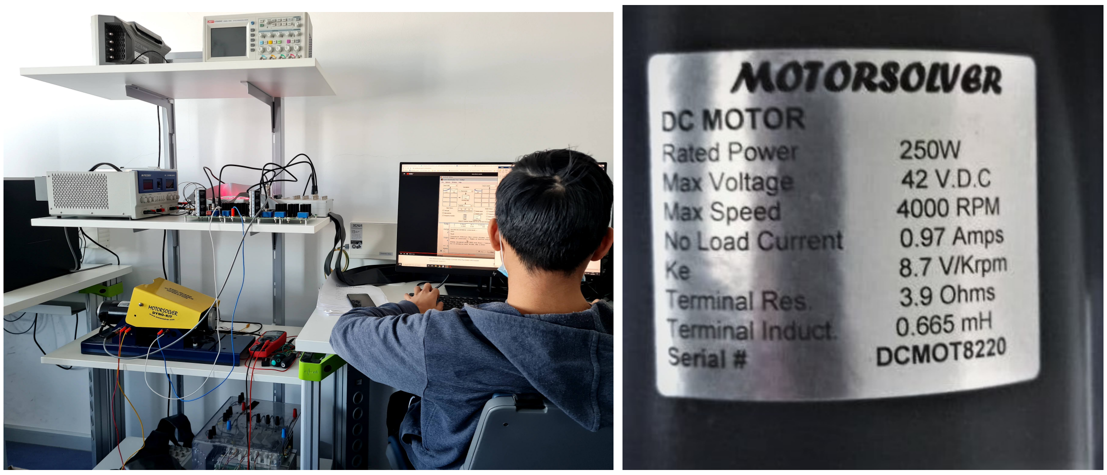


Figure 4.6: (a) dSpace Hardware setup and (b) the DC motor characteristics

input signal into the range ± 21 V. The DSpace controller board operates with sampling time of 0.0001 s.

The model design was implemented in simulink environment within which the Simulink Library provided the I/O interface to the DSpace control board. The built model from simulink is converted and transfered to the DSpace control board. Using the "ControlDesk" software which provides the real-time interface to the input and output signals of the DSpace control board and also to the variables from the simulink model, the input signals of various shapes were applied to motor and were gathered with the corresponding output signal representing the position of motor shaft. The data was used for hardware system identification as was mentioned in the experiment design chapter.

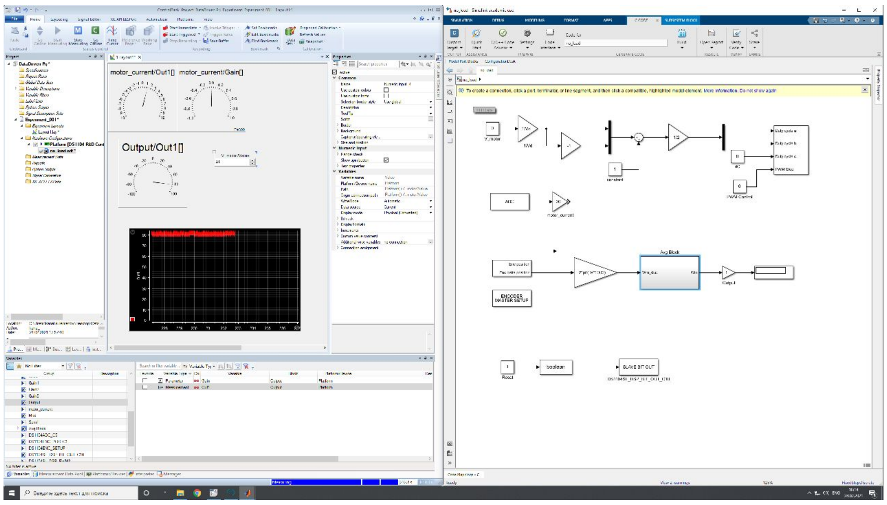


Figure 4.7: dSpace workspace for data gathering

Chapter 5: **Results**

The final obtained results are going to be presented in this chapter. It should be mentioned that currently presented results are intermediate steps results and not the final ones. The final results were not achieved yet due to difficulties with starting the custom PID controller with the dSpace hardware. Overall, it is estimated that currently the project is at 80% completion state, with the hardware setups constructed and running. The simulation models and the software parts which control the hardware are finished as well.

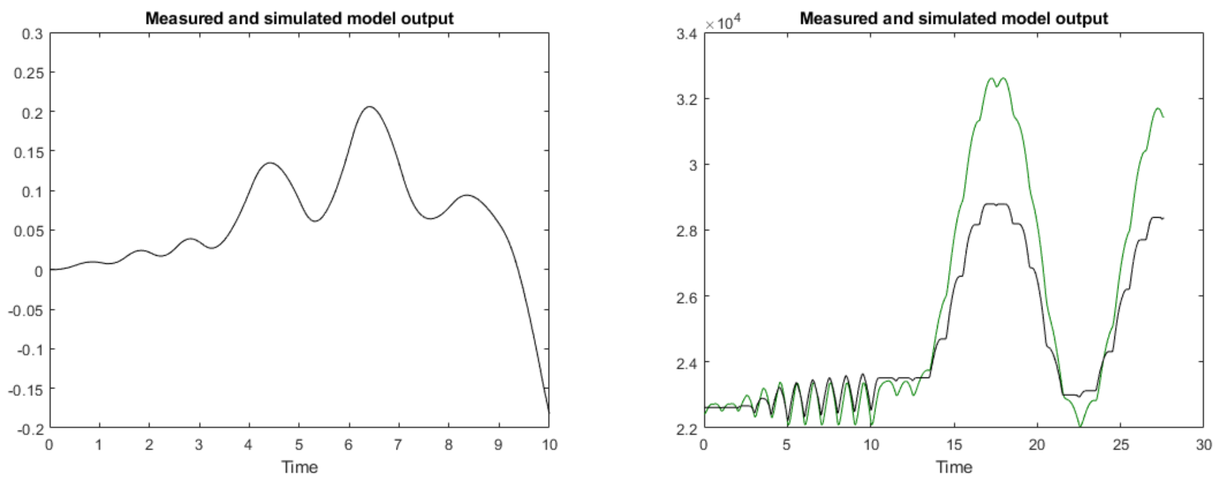


Figure 5.1: (a) Resultant transfer function match for the MATLAB simulated model and (b) Resultant transfer function match for the dSpace model

The above figures represent the results of system identification applied on gathered data from simulation and from hardware experiment. On the left you can see that the generated transfer function of the simscape motor model shows a perfect fit when applied with validation dataset. Whereas the same transfer function model generated from the hardware collected data shows a result of 73%, although the general dynamics is captured. Arduino setup result were not generated as due to reasons discussed in the Arduino setup chapter.

The final results will include the comparison of reaction of the DC motors (simulation, Arduino driven and dSpace) to the same input signal with the data-driven controllers designed for each setup respectively. Also, the dSpace motor's physical values are going to be input in simulation, accordingly the controller will be synthesized and this model shall be compared with the real hardware setup. Ultimately, in case of success with the proposed Arduino experiment, a reinforcement learning setup will be created based on Arduino, which will be the most significant result of our project.

Chapter 6: **Conclusion and Future work**

We applied the Data-Driven control method to three different setups for this project. A MATLAB simulation, an Arduino and a DC motor driver setup and a dSpace setup were used to generate the DC motor voltage-position relation data. The obtained data was then analyzed in MATLAB system identification toolbox to obtain the controller for the given setup.

There are multiple steps that have to be taken for future work. First, the obtained data should be analyzed using a different method to check if improved inference speed and robustness could be achieved. One of possible variants is the NARX Neural Network, as was suggested in the related work by [2]. The second Arduino setup has to be implemented fully as well, as we believe this is a promising method for our project. The synthesized dSpace controller has to be run with the hardware, as we were not successful with this so far. Another important step is running the simulation again with parameter of the model set to those of a real DC motor and those of the dSpace setup. The simulated model output and the real hardware setup results will be compared afterwards.

Bibliography

[1] Z.-S. Hou and Z. Wang, "From model-based control to data-driven control: Survey, classification and perspective," *Information Sciences*, vol. 235, pp. 3–35, 2013.

[2] Y. Naung, A. Schagin, H. Oo, K. Ye, and Z. Khaing, "Implementation of data driven control system of dc motor by using system identification process," in *2018 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus*, (Moscow), pp. 1801–1804,.

[3] J. Coulson, J. Lygeros, and F. Dörfler, "Data-enabled predictive control: In the shallows of the deepc," in *2019 18th European Control Conference (ECC)*, pp. 307–312, IEEE, 2019.

[4] P. G. Carlet, A. Favato, S. Bolognani, and F. Dörfler, "Data-driven predictive current control for synchronous motor drives," in *2020 IEEE Energy Conversion Congress and Exposition (ECCE)*, pp. 5148–5154, IEEE, 2020.

[5] S. Hanke, O. Wallscheid, and J. Böcker, "Continuous-control-set model predictive control with integrated modulator in permanent magnet synchronous motor applications," in *2019 IEEE International Electric Machines & Drives Conference (IEMDC)*, pp. 2210–2216, IEEE, 2019.

[6] C. De Persis and P. Tesi, "Formulas for data-driven control: Stabilization, optimality and robustness," *IEEE Transactions on Automatic Control*, vol. 1–1.

[7] U. Rosolia and F. Borrelli, "Learning model predictive control for iterative tasks. a data-driven control framework," *IEEE Transactions on Automatic Control*, vol. 63, no. 7, p. 1883–1896.

[8] X. Bu, Q. Wang, Z. Hou, and W. Qian, "Data driven control for a class of nonlinear systems with output saturation," *ISA Transactions*.

[9] Pravallika, "Data-driven control (https://www.mathworks.com /matlabcentral/fileexchange/ 42758-data-driven-control), matlab central file exchange." Retrieved February 4,.

[10] J. Berberich, A. Koch, C. W. Scherer, and F. Allgower, "Robust data-driven state-feedback design," *2020 American Control Conference (ACC)*, Jul 2020.

[11] D. Piga, S. Formentin, and A. Bemporad, "Direct data-driven control of constrained systems," *IEEE Transactions on Control Systems Technology*, vol. 26, no. 4, pp. 1422–1429, 2017.

[12] P. Kergus, *Data-driven model reference control in the frequency-domain From model reference selection to controller validation.* Theses, Institut Supérieur de l'Aéronautique et de l'Espace (ISAE-SUPAERO), Oct. 2019.

[13] L. Campestrini, D. Eckhard, A. Sanfelice Bazanella, and M. Gevers, "Data-driven model reference control design by prediction error identification," *Journal of the Franklin Institute*, vol. 354, no. 6, pp. 2628 – 2647, 2017. Special issue on recent advances on control and diagnosis via process measurements.

[14] P. Vuillemin, P. Kergus, and C. Poussot-Vassal, "Hybrid loewner data driven control," *arXiv preprint arXiv:1909.02231*, 2019.

[15] H. J. van Waarde, J. Eising, H. L. Trentelman, and M. K. Camlibel, "Data informativity: A new perspective on data-driven analysis and control," *IEEE Transactions on Automatic Control*, vol. 65, no. 11, pp. 4753–4768, 2020.