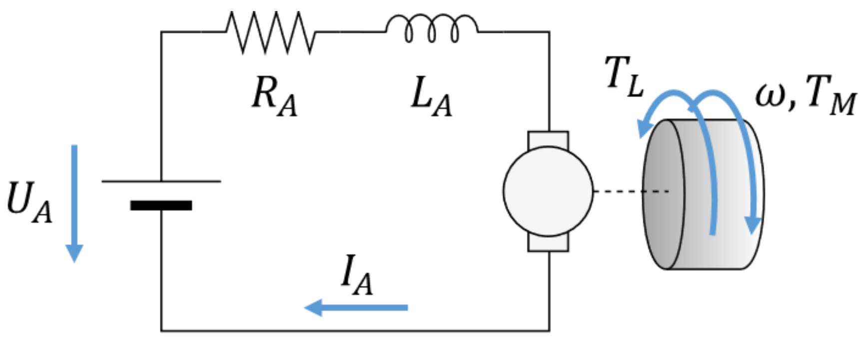
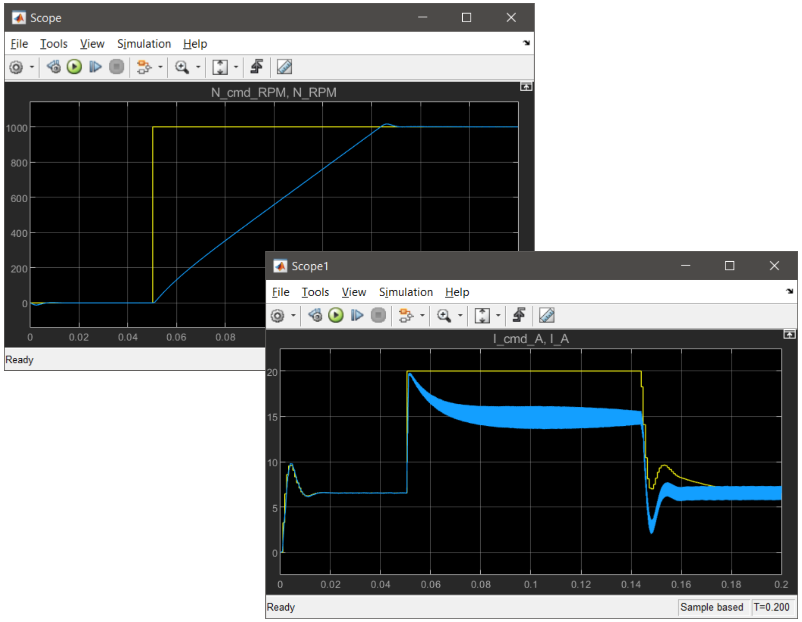
Simulation of a DC motor with RPM control loop and PWM voltage regulation

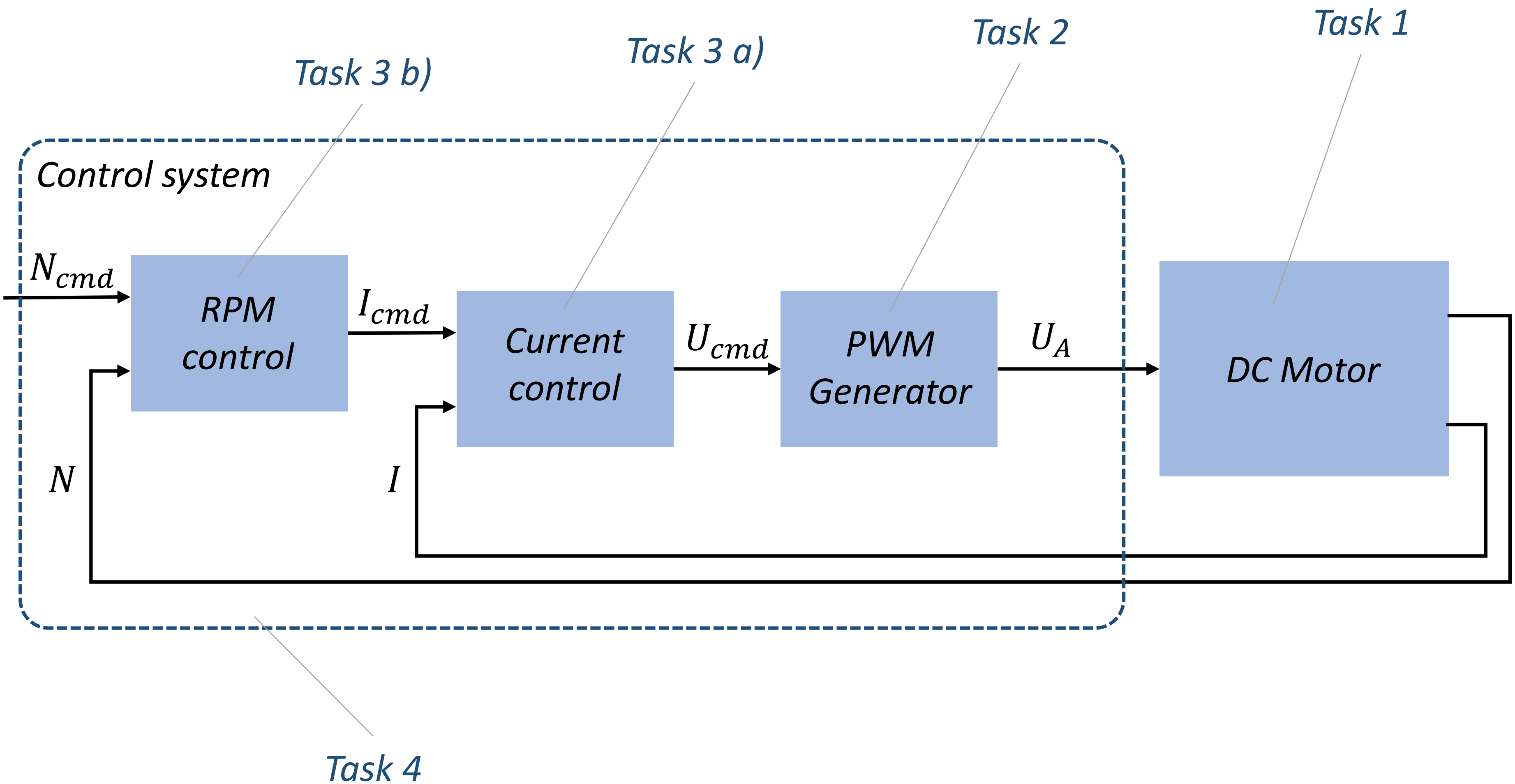




Tasks

1. Modeling and simulation of the DC motor
2. Modeling and simulation of the PWM voltage control
3. Modeling of the discrete-time current and RPM control loops
4. Separation of the controller functions using Model Reference, Library, Simulink Project and Data Dictionary

Signal flow


Task 3 b)
Task 3 a)
Task 2
Task 1
Control system
N_{cmd}
RPM control
I_{cmd}
Current control
U_{cmd}
PWM Generator
U_A
DC Motor
N
I
Task 4

Task 1

Build a model of the DC motor dynamics given below (parameters see next slide)

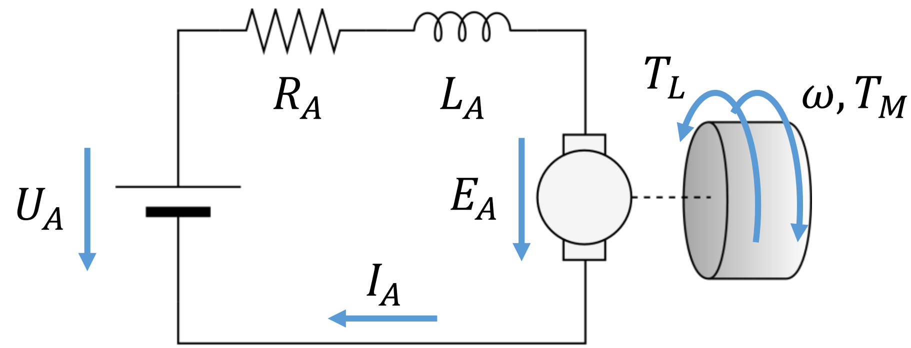
- Start with the voltage equations, neglecting the Back EMF) and validate the model by estimating the time constant from the response to a input voltage step
- Extend the model by the torque equation
- Simulate the response to a $20\ V$ input voltage step and validate the results

Voltage balance:

$$U_A = E_A + R_A I_A + L_A \dot{I}_A\ ,\ E_A = C_M \cdot \omega \cdot \Psi$$

Torque balance:

$$J\dot{\omega} = (T_M - T_L)\ ,\ T_M = C_M \cdot I_A \cdot \Psi$$



U_A ... Armature voltage, E_A ... Back EMF voltage, I_A ... Armature current ,

T_M ... Motor torque, T_L ... Load torque , ω ... Rotor angular velocity

Source: [1, p. 401 f.]

Task 1

Build a model of the DC motor dynamics given below

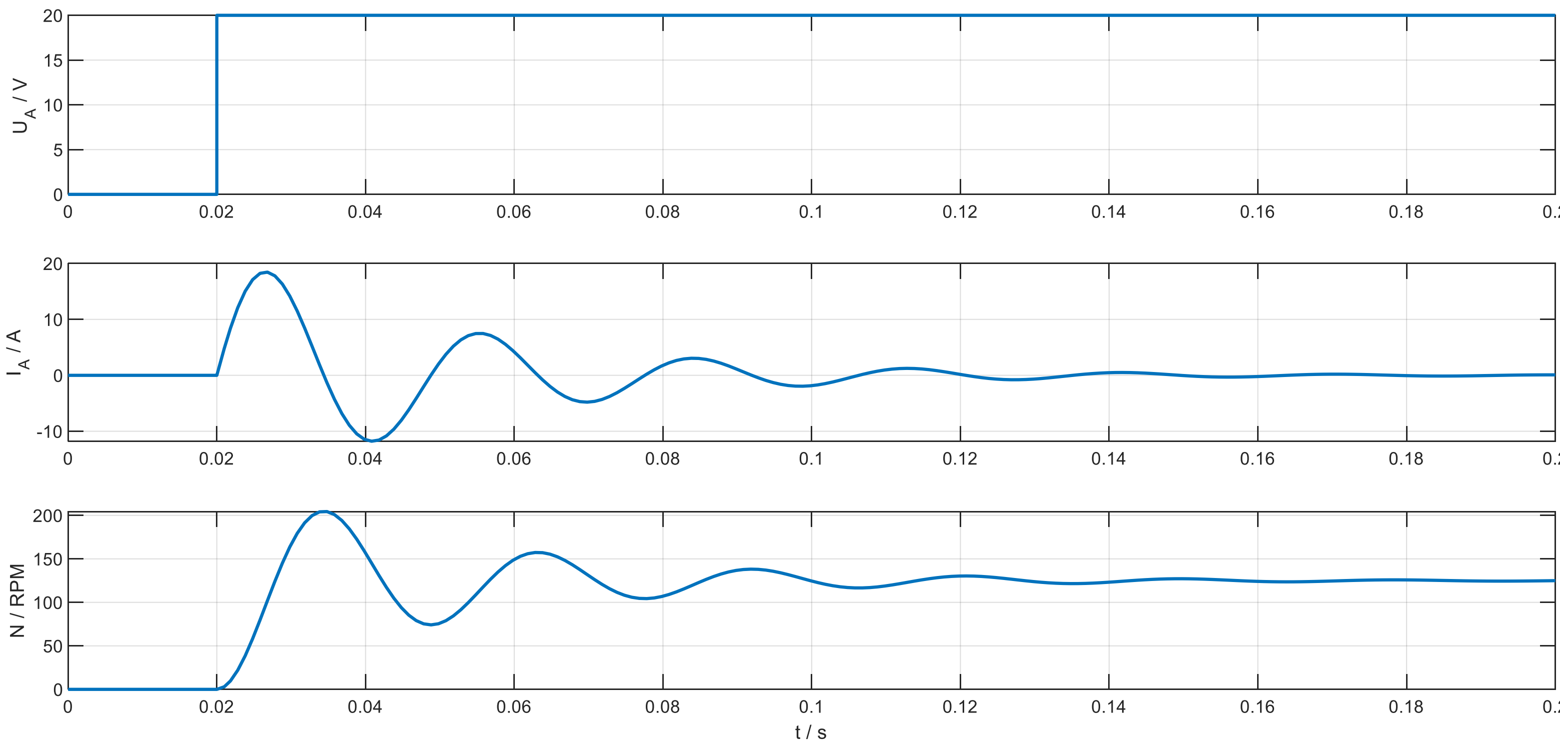
- Store the parameters in the model workspace of the DC motor model

Parameter		Value
R_A	Armature Resistance	$250\ m\Omega$
L_A	Armature Inductance	$4\ mH$
Ψ	Nominal Flux	$0.04\ Vs$
C_M	Machine constant	38.2
J	Moment of inertia	$0.012\ kg\ m^2$

Source: [1, p. 401 f.]

Task 1

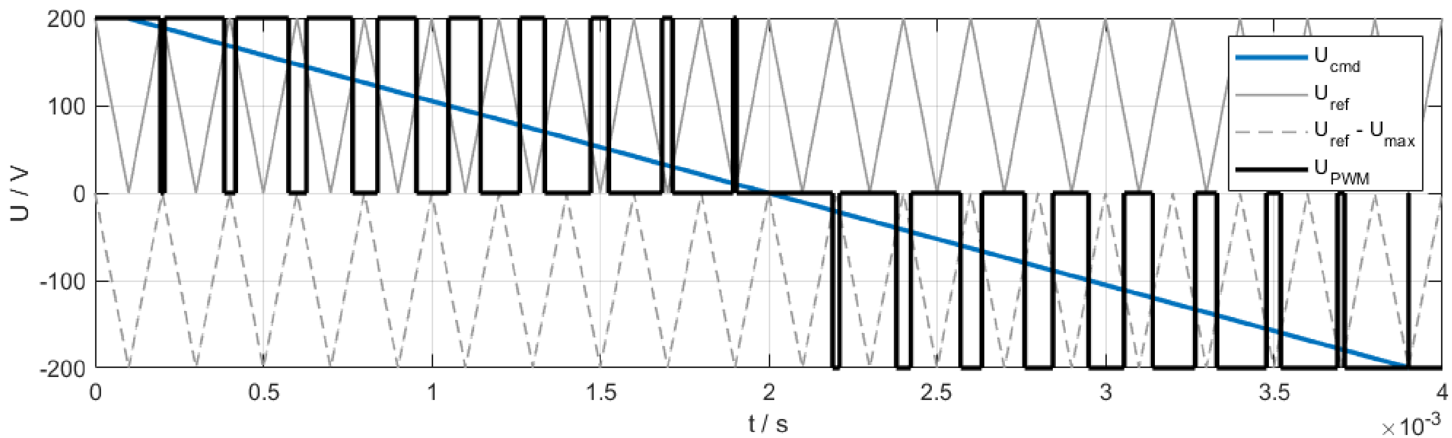
You should get the following simulation results ($20V$ U_A step at $t = 20ms$)



Task 2

Model a PWM generator for the armature voltage U_A

- The PWM runs at a frequency of $5\ kHz$ and the pulse height is $200\ V$
- Generate the output pulses by comparing the input signal to a triangular reference signal: When the input signal value is positive and greater than the reference, a positive pulse is generated, otherwise the output is zero (and vice versa for negative input signals and negative pulses):

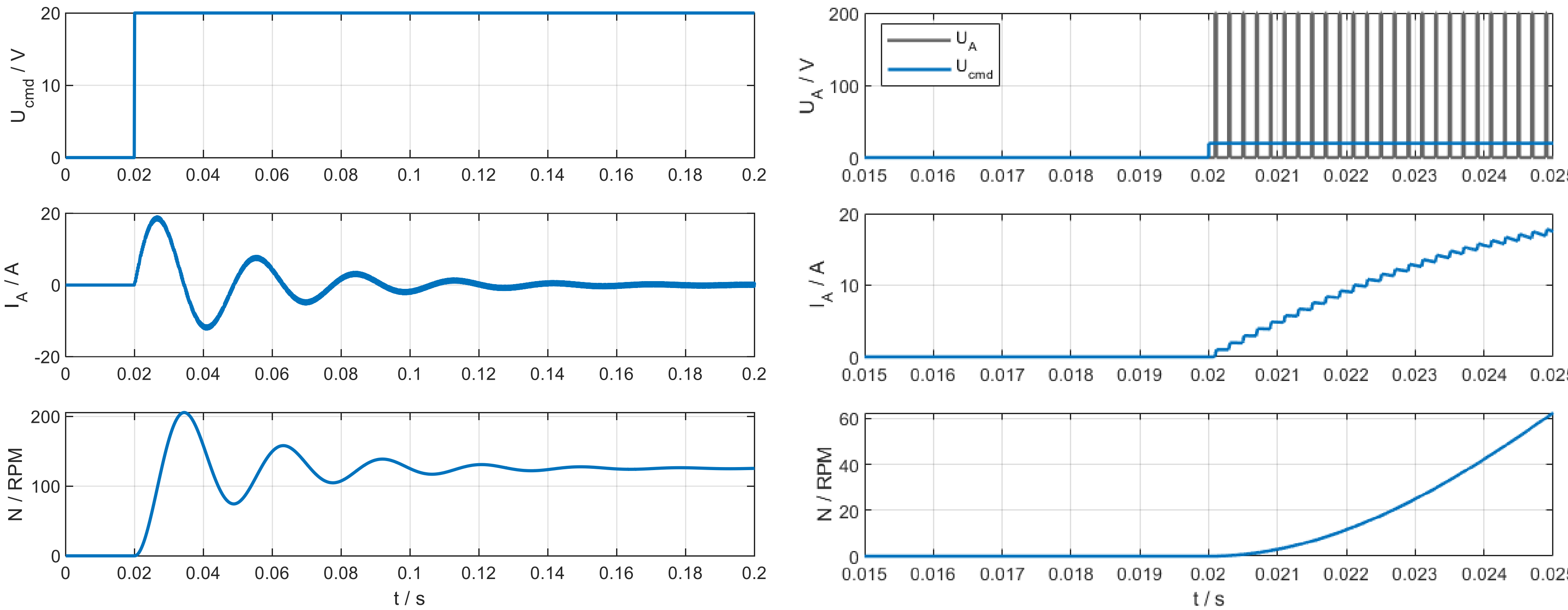


- The PWM generator should be modelled as a purely discrete time system
- *Hint*: Adapt the sample time settings to resolve the high frequency PWM signal

Source: [1, p. 402 f.]

Task 2

You should get the following simulation results ($20V\ U_{cmd}$ step at $t = 20ms$)



Task 3 a)

Build a current control loop and integrate it into the simulation model

- The structure for the current controller is a PI controller:

$$U_{cmd} = K_{P,I} \cdot (I_{cmd} - I_{meas}) + K_{I,I} \cdot \int (I_{cmd} - I_{meas})dt$$

- The sample time for the controller is $T_s = 800\mu s$
- Use the following controller parameters (i.e. design according to magnitude optimum method [2]):

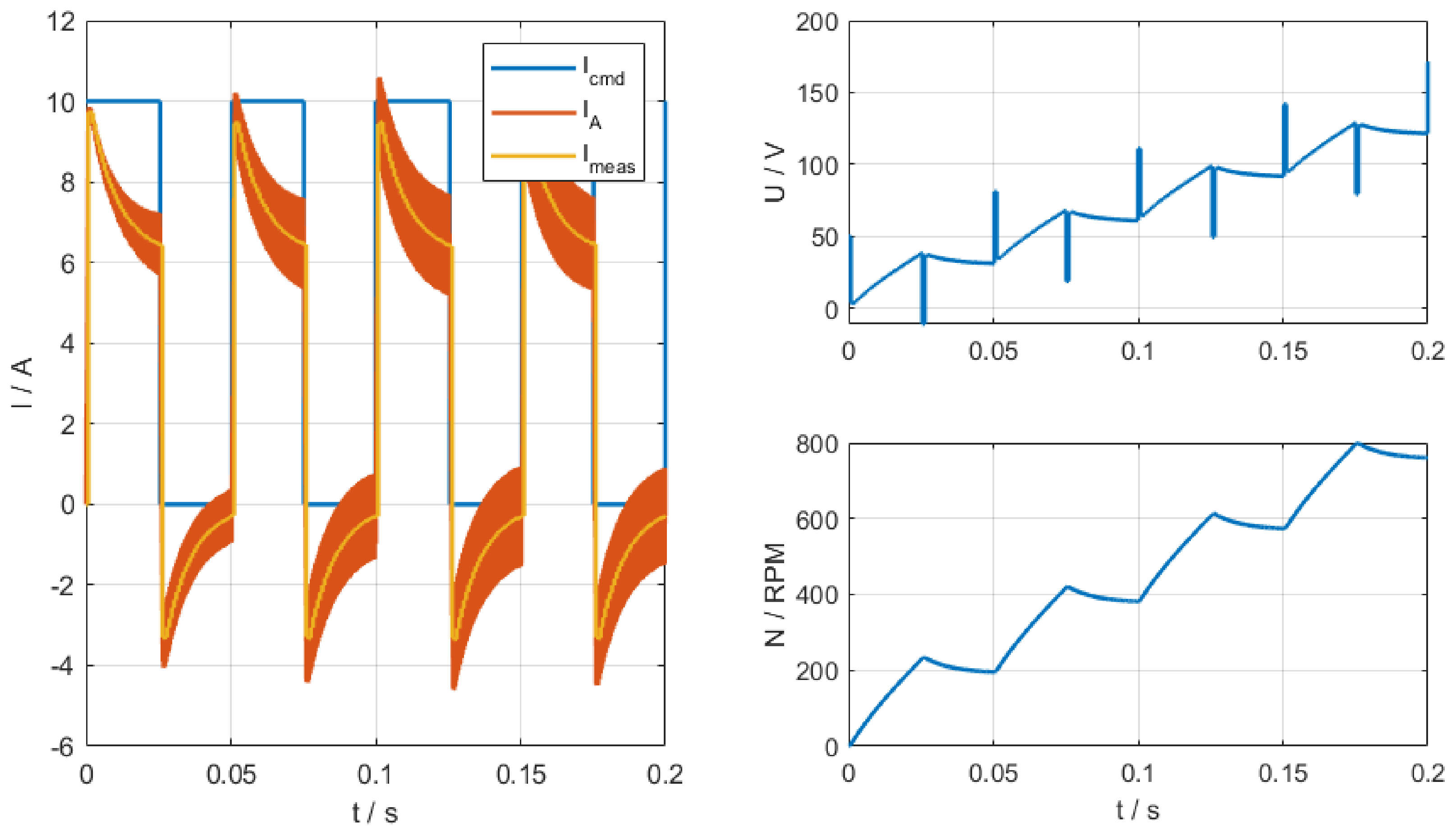
$$T_I = L_A/R_A \ , \ K_{P,I} = L_A/T_s \ , \ K_{I,I} = K_{P,I}/T_I = R_A/T_s$$

- Start by creating a continuous time transfer function of the controller using the `tf(…)` function
- Use the `c2d(…)` function to discretize the continuous time transfer function (use „Zero order hold“ as the discretization method)
- Implement the resulting controller in simulink by using appropiate blocks from the *discrete* library

Source: [1, p. 416 f.]

Task 3 a)

You should get the following simulation results ($10\ A\ I_{cmd}$ pulses with $T_{pulse} =$



Task 3 b)

Build a RPM control loop and integrate it into the simulation model

- The structure for the RPM controller is a PI controller
- Use the following controller parameters (i.e. design according to symmetric optimum method [2]) :

$$T_{RPM} = 4T_s \ , K_{P,RPM} = \pi \cdot J/(60 \cdot C_M \Psi \cdot \mathrm{T_s}) \ , K_{I,RPM} = K_{P,RPM}/T_{RPM}$$

- Create the controller directly using appropiate blocks from the *discrete* library (without discretizing with `c2d(…)`)
- Limit the output current command to $\pm 20 \ A$ and implement a simple *anti-windup* loop:

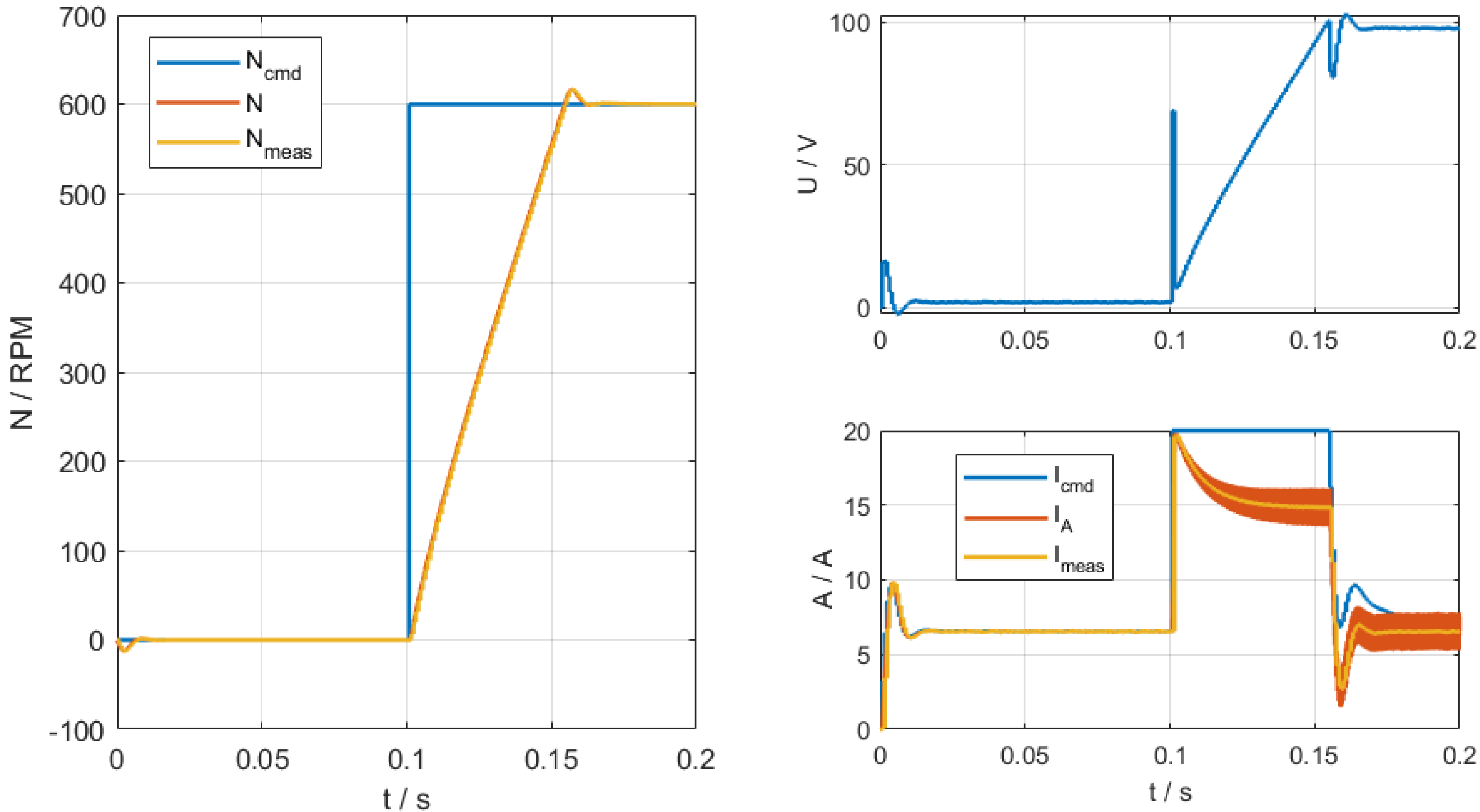
$$I_{cmd} = K_{P,RPM} \cdot (N_{cmd} - N_{meas}) + \frac{1}{T_{RPM}} \cdot \int K_{P,RPM}(N_{cmd} - N_{meas}) \ - \Delta I \ dt$$

$$I_{cmd,sat} = \max\left(\min\left(I_{cmd}, I_{cmd,max}\right), I_{cmd,min}\right), \Delta I = I_{cmd} \ - I_{cmd,sat}$$

Source: [1, p. 417 f.]

Task 3 b)

You should get the following simulation results ($10\ Nm$ load torque and step to $N_{cmd} = 600\ RPM$ at $t = 100ms$)



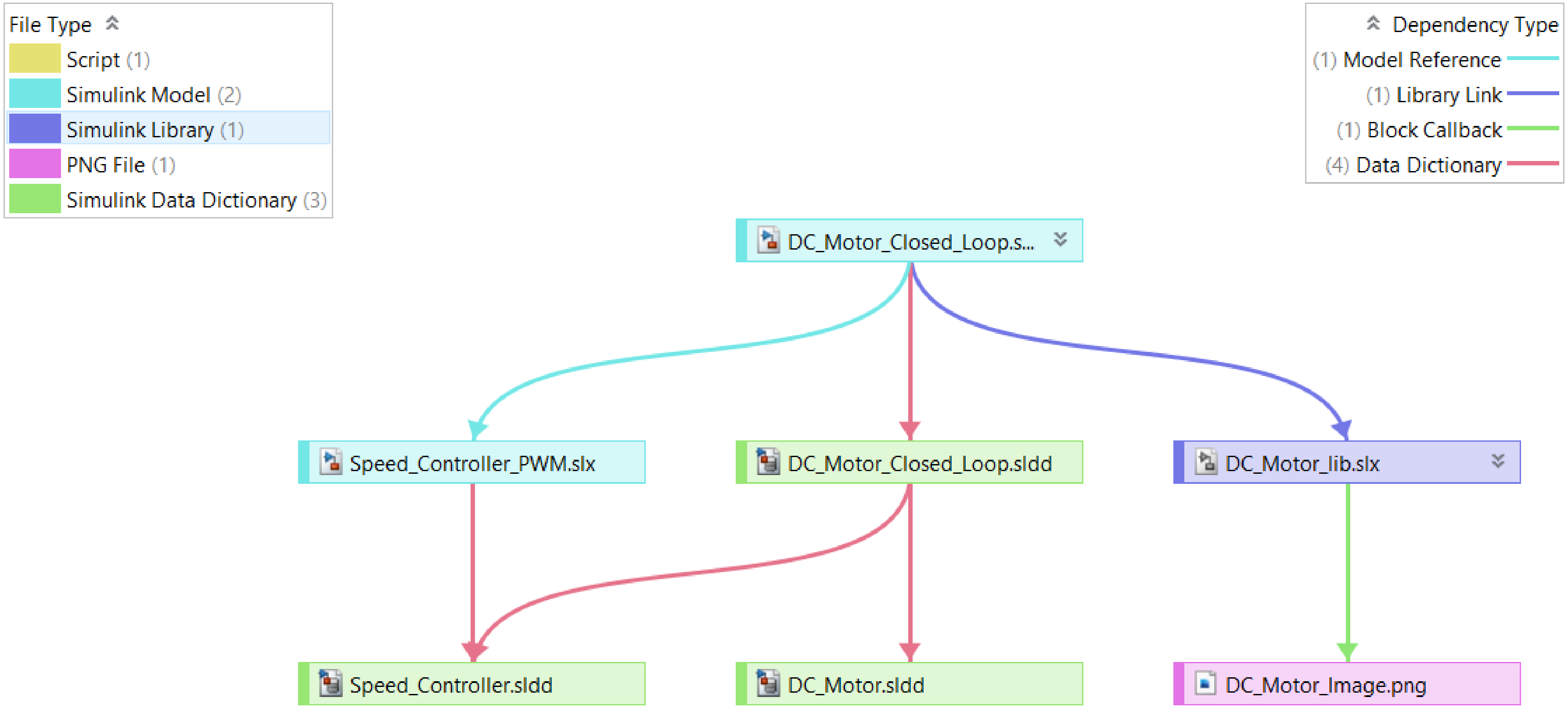
Task 4

Separate the control functions from the plant model:

- Move the control loops and PWM generator to their own Simulink model and the DC motor to a library
- Reference the controller model using a model reference and integrate the library
- Create a data dictionary for the controller model, which contains all relevant controller parameters (e.g. gains and sample times) and link it to the controller model
- Create a data dictionary containing all the DC motor parameters
- Create a data dictionary for the closed-loop model
- Create a Simulink project for the DC motor model containing the library and the parameter data dictionary
- Create a Simulink project for the controller model containing the controller model, the closed-loop model and the controller data dictionary
- Link the projects and data dictionaries such that all model references, library links and parameter dependencies are resolved

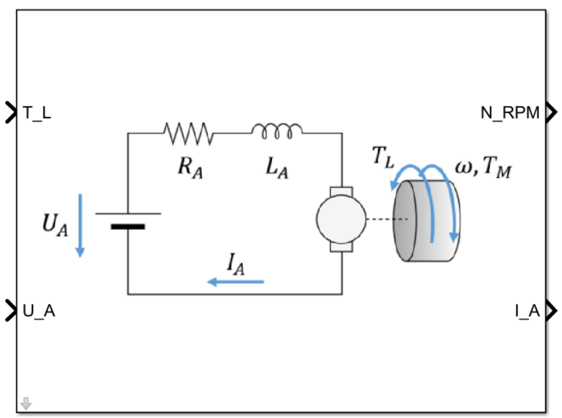
Task 4

Your dependency structure should look something like this one

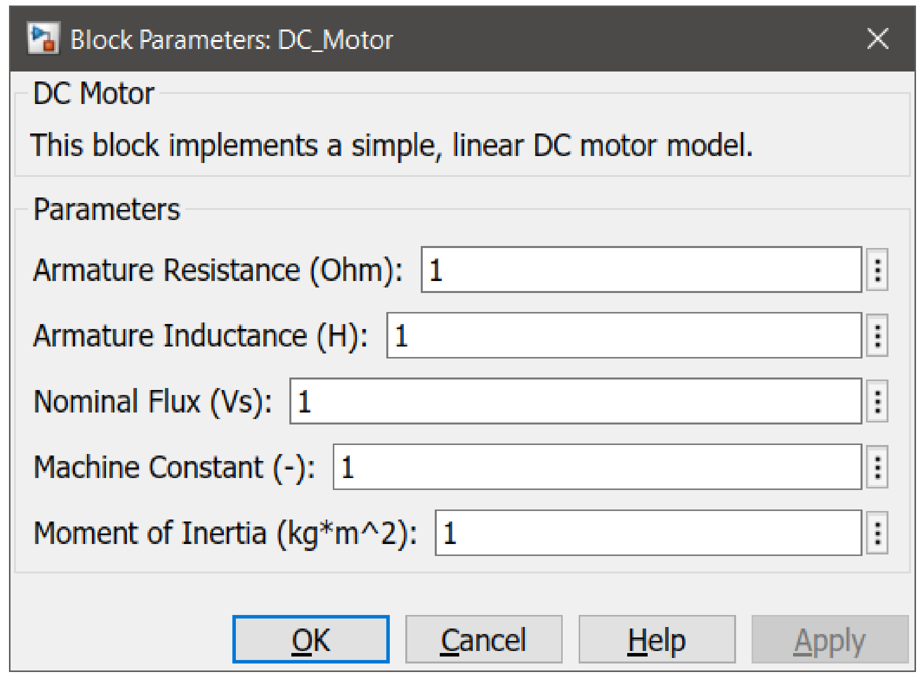


Task 4

The DC motor block in the library could look like this one



DC_Motor



References

[1] Angermann, A: et. Al.: *MATLAB - Simulink – Stateflow, Grundlagen, Toolboxen, Beispiele*, 8. Auflage, De Gruyter, Oldenburg, 2014.

[2] Umland, J, W. and Safiuddin, M.: *Magnitude and symmetric optimum criterion for the design of linear control systems: What is it and how does it compare with the others?,* IEEE Transactions on Industry Applications, 1990.