Lecture 4

Vector Control of Three-Phase AC Machines

Objectives:

- Build a control algorithm to regulate the dq-axes machine currents by applying dq-axes voltages with PI controllers
- Examine the feedback control dynamics for tuning the current controllers
- Add feedforward terms to improve the control dynamics
- Add a third PI controller to regulate the speed of the machine

Keywords:

field-oriented control (FOC)

vector control

Proportional-Integral (PI) controller

feedback loop

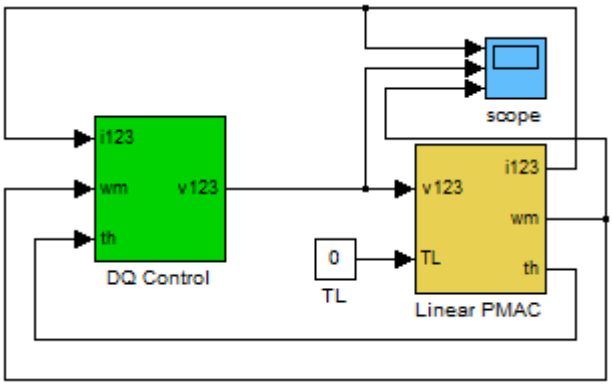
feedforward loop

Controlling the Current

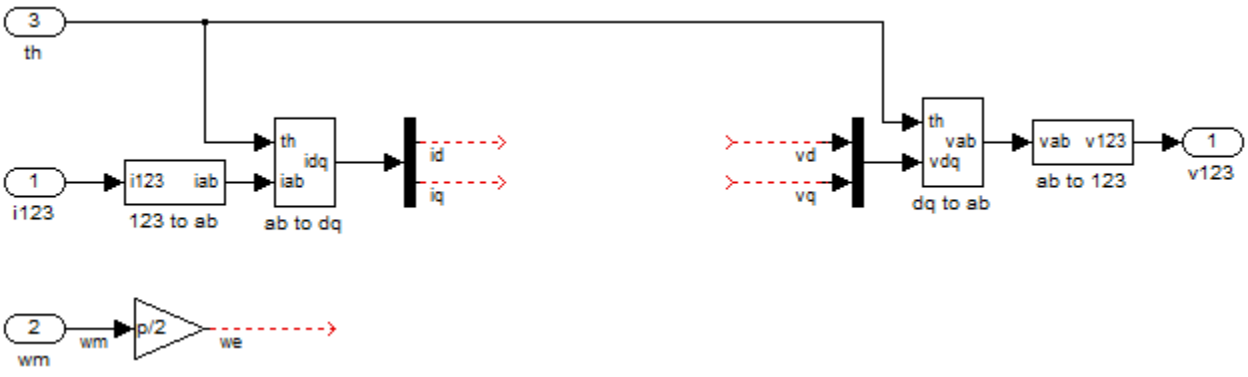
To control the machine, we apply voltages to regulate the currents. Through controlling the current, we can also control the torque and thus the speed and angle of the rotor. This allows us to implement torque, speed, or position control.

Traditionally, the regulation of the machine currents is done in the *dq*-frame. This is because all signals in the rotating frame are constant while operating at steady-state. This is commonly called *field-oriented control*, or vector control – our control algorithm is aligned (oriented) to the rotor flux field!

In practice, we must measure the three-phase currents (with current sensors mounted within the motor controller). We must also measure the rotor angle to properly apply the coordinate transformations. If speed-control of the machine is required, we must also feed a speed sensor signal to the controller. For now, we'll set the load torque to zero.



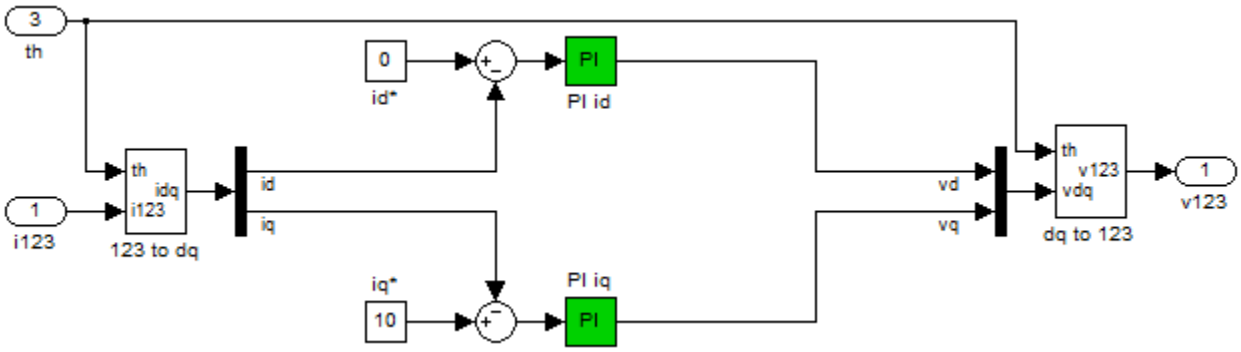
We will again use the Clarke and Park transforms to convert the currents and voltages to and from the rotating frame. Initially, the inside of our control system might look like the block diagram below – next we'll add blocks to regulate the currents to fixed reference values.



Regulating the Two Currents

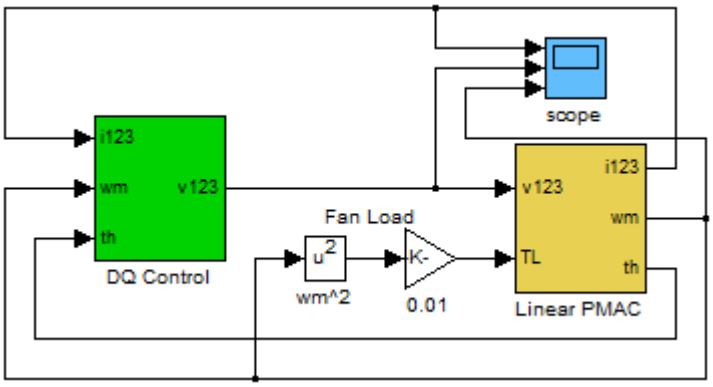
Since the dq-currents will be constant at steady-state, we can use Proportional-Integral (PI) controllers to regulate the currents in both frames to their respective reference values. For now, let's pick constant values as reference values.

- Since the d-axis current does not produce any mechanical output torque, usually we will regulate this current to zero.
- The reference for the q-axis current determines our torque, which is related to the rotational acceleration. Thus, to spin faster in the positive direction, we can set a positive i_q reference. To slow down or spin faster in the negative direction, we can set a negative i_q reference.



The control diagram above will produce a constant torque on the shaft, and thus the motor speed will increase linearly (if the friction losses are small). A better example for this control setup might be to model a fan as the load on the machine. We can model the fan's load as a function of speed squared.

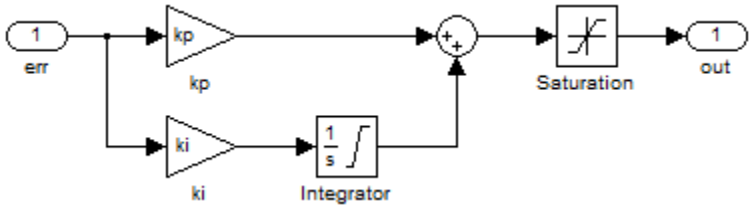
$$T_{L,fan} = 0.01\omega_m^{\,2}$$



Building the PI Controllers

To construct a PI controller, we can use the basic mathematical definition and model it in Simulink with an integrator block and gains.

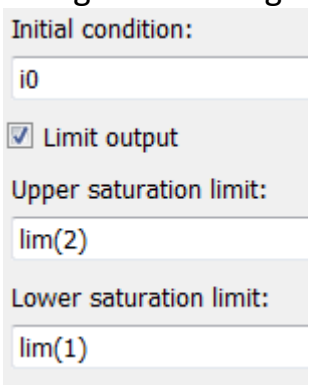
$$C(s) = k_p + \frac{k_i}{s}$$



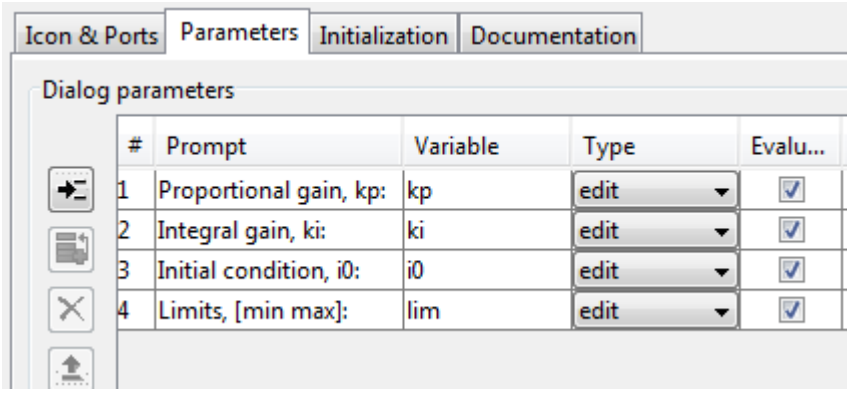
It's usually a good idea to add saturation to the integrator and the output. In practice, there are limits to what types of magnitudes a controller can generate (for example, a motor controller can only generate a finite AC voltage magnitude).

To make the tuning process easier, Simulink allows us to "mask" a subsystem. This basically means that we can have a dialog box pop up when we double-click the subsystem. We can add parameters to the mask, which correspond to variables in the masked block diagram below. The controller gains, initial condition, and saturation limits have all been parameterized here.

Integrator Settings



PI Controller Mask Parameters



After adding the mask to the subsystem, double-clicking on a PI controller block will pop up a window where we can easily enter numeric values for all of the parameters we have defined above.

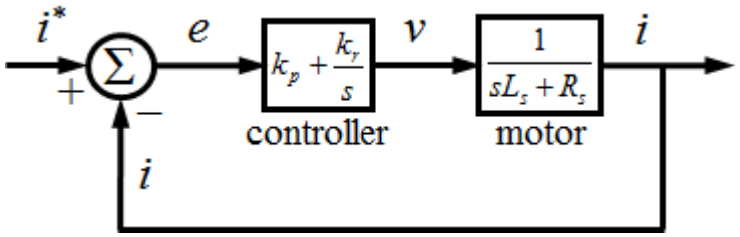
Tuning the PI Controllers (1 of 2)

To select values for the controllers, we can do a simple controls-analysis of the system in the dq-frame. We are trying to regulate the current in each axis by applying a corresponding-axis voltage. Recall the transfer function between voltage and current in both axes of the machine:

$$H(s)=\frac{i_d(s)}{v_d(s)}=\frac{i_q(s)}{v_q(s)}=\frac{\frac{1}{R_s}\left(s\frac{L_s}{R_s}+1\right)}{s^2+2s\frac{R_s}{L_s}+\left(\frac{R_s}{L_s}\right)^2+\omega^2}$$

To simplify things, let's first analyze the machine at a speed of zero: $\omega = 0$ rad/s. In this condition, the two poles will be located at frequency $-R_s/L_s$. Notice that we also have a zero in the transfer function at that same frequency! The system will reduce to a first-order system when the motor is in the stall condition.

Constructing the feedback control block diagram, we have the following:



Finding the closed-loop transfer function between our reference current *i** and the actual output current *i*, we can use the traditional closed-loop result:

$$\frac{i^*}{i}=\frac{G}{1+GH}=\frac{\frac{k_ps+k_i}{s}\frac{1}{sL_s+R_s}}{1+\frac{k_ps+k_i}{s}\frac{1}{sL_s+R_s}}=\frac{k_ps+k_i}{s(sL_s+R_s)+k_ps+k_i}$$

The simplified expression and its pole values are given below:

$$\frac{i^*}{i}=\frac{\frac{k_i}{L_s}\left(s\frac{k_p}{k_i}+1\right)}{s^2+s\left(\frac{R_s+k_p}{L_s}\right)+\frac{k_i}{L_s}} \qquad s=-\frac{R_s+k_p}{2L_s}\pm\sqrt{\left(\frac{R_s+k_p}{2L_s}\right)^2-\frac{k_i}{L_s}}$$

Tuning the PI Controllers (2 of 2)

For this exercise, let's assume R_s and L_s to be 25mΩ and 100μH, respectively.

$$R_s = 25\text{m}\Omega \qquad\qquad L_s = 100\mu\text{H}$$

The settling time of the current is related to the center-point of the pole locations. A typical equation for settling time is given below. Usually, we want the settling time of the current to be quite fast – around 5ms is a good value.

$$T_{stl} = \frac{3.9}{\zeta\omega_n} = \frac{3.9}{\left(\frac{R_s + k_p}{2L_s}\right)} \quad \text{solving for } k_p\text{...} \quad \left(\frac{3.9(2L_s)}{T_{stl}} - R_s\right) = k_p$$

This gives a k_p value around 0.131. We'll round down the result for k_p to 0.1 – usually we want to keep k_p small to help reduce the noise from the current sensors from showing up in the voltage signals.

For the integrator k_i term, we want to prevent any overshoot, so the poles must remain purely real. Analyzing the terms under the radical in our expression for the poles, we can find the value of k_i where the system will be critically damped.

$$\left(\frac{R_s + k_p}{2L_s}\right)^2 - \frac{k_i}{L_s} = 0 \quad \text{solving for } k_i\text{...} \quad \frac{(R_s + k_p)^2}{4L_s} = k_i$$

The critically-damped value of k_i in this case works out to be around 39. Let's select a value of 20 for k_i (again, to make the controller less sensitive to noise). Thus, with the values selected above, we get the following gains and response:

$k_p \approx 0.1$ $\qquad$ $poles = -1061\,\text{rad/s}, -188\,\text{rad/s}$

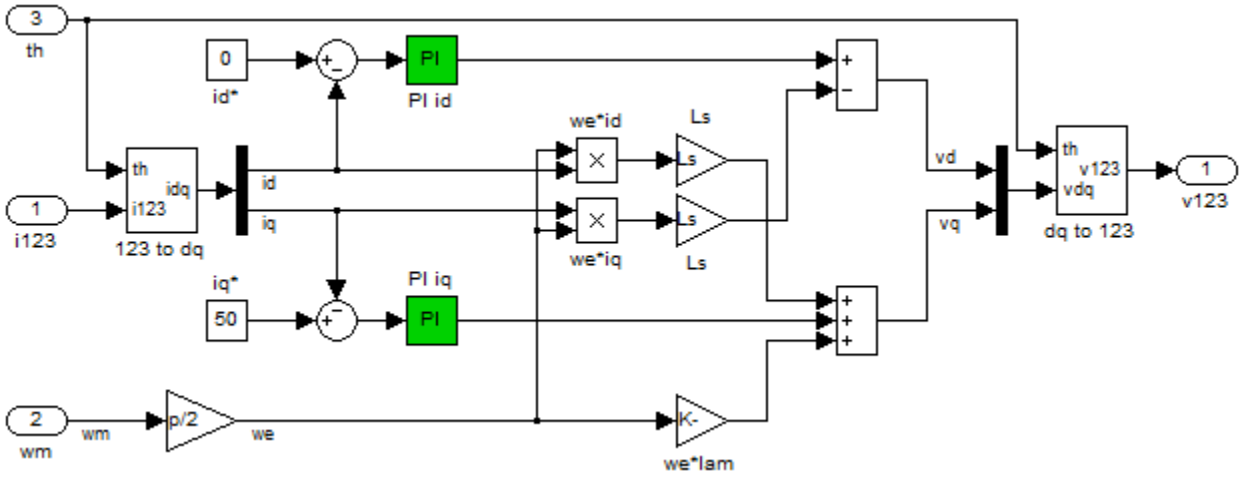
$k_i \approx 20$ $\qquad$ $zeros = -200\,\text{rad/s}$

$T_{stl} \approx 6.2\text{ms}$ $\qquad$ No overshoot

Decoupling the Axes Using FeedForward

The previous analysis was valid when the machine was mechanically at rest. What about when the machine is spinning? This becomes a time-varying system.

Rather than increasing the complexity of our feedback control system and its mathematics, we can eliminate the speed dependency of the control loop. We do this by canceling out the cross-coupling voltage terms and the Back-EMF. This is done by adding feedforward paths in our control loop, shown below.



Another way to explain this is to examine machine stator voltage equations. If we can predict some of the terms in the equation mathematically, why not add these as part of our calculated output voltage? This will reduce some of the burden on the response of the PI controllers and will reduce the complexity of the dynamics.

$$v_d = \underbrace{R_s i_d + L_s \tfrac{d}{dt}(i_d)}_{Feedback} \underbrace{- \omega L_s i_q + 0}_{Feedforward} \qquad v_q = \underbrace{R_s i_q + L_s \tfrac{d}{dt}(i_q)}_{Feedback} \underbrace{+ \omega L_s i_d + \omega \Lambda_{pm}}_{Feedforward}$$

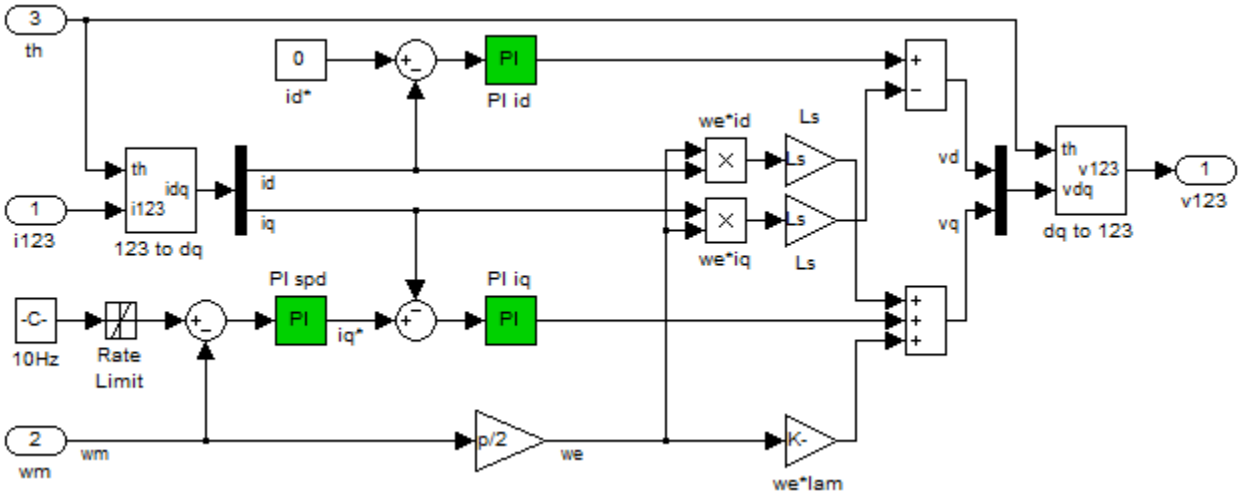
We usually let the PI controllers handle only the parts of the signal which are in their respective axis – the feedforward terms are only used to remove the cross-coupled terms and the Back-EMF term (which has no relation to either current).

You may wonder, “Why we don’t use feedforward for all of the terms?” Under NO circumstance should we compute the derivatives of any input signal! Any noise from the current sensors will introduce excessive noise into the control loops.

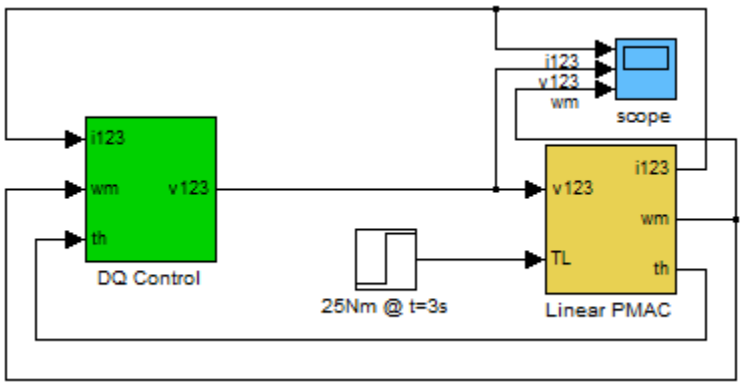
Adding a Speed Controller

So far, we've been setting the torque reference to a constant value and letting the machine accelerate freely up to some operating speed. What if we want to regulate the speed to a fixed value?

We can add a third PI controller which compares the measured machine speed to a speed reference value. The output of this controller will be a torque command (which is the same as the value of i_q, times a constant). Rather than setting a fixed speed value, let's use a rate-limited constant signal to ramp up from zero speed.



To really see this in action, let's change the load on the machine to be a step-load. It will spin freely up to the target speed, and then at a point in time, we'll load down the shaft and see how the machine and controller respond.



Speed Controlled Simulation Results

Below are the speed controller simulation results. Notice how the step-change in load causes the speed to momentarily slow down. At that time, the q-axis current increases to increase the torque and keep the speed constant (net zero torque).