Field Oriented Control of a Three Phase Motor

Rod Bayliss III

12 December, 2018

Contents

Chapter 1

Overview

As a member of the FSAE Electric Racecar team I've found a fascination for electric motors and the power electronics that drives them. Motor drive is a really cool intersection of mechanical design, magnetics, thermals, controls, and power electronics. One of the most prevalent control algorithms for driving these three phase motors is called Field Oriented Control which aims to control the current vectors going into the motor to eventually achieve maximum torque per amp delivered from the battery to the motor.

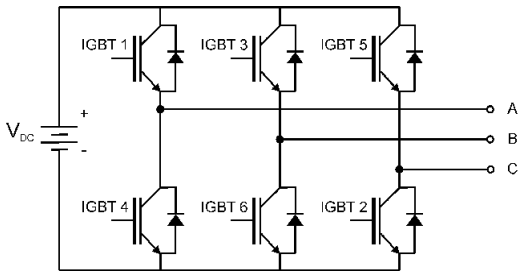


Figure 1.1: Schematic of Three phase inverter

Chapter 2

Field Oriented Control

2.1 3-phase transforms

One of the fundamentals of Field Oriented control is to transform complicated, time-varying three phase sinusoids into simplified control variables that conveniently also have a physical implication. From a controls perspective these

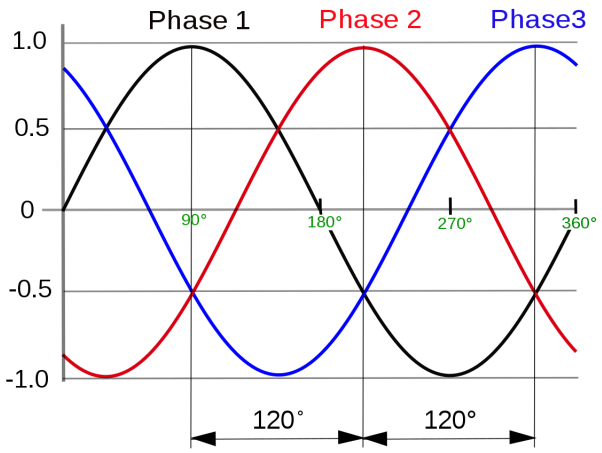


Figure 2.1: Three phase graph

time-varying sinusoids are really hard to deal with and set clear controls variables for. However, through the clarke and park transforms, the sinusoids turn into constant vectors which have important physical connections. By summing each of these vectors, the result is a vector of constant length that spins at the frequency of the original sinusoids. This is called the clarke trnsformation and puts us in the $\alpha - \beta$ frame. This frame has a single rotating vector with a stationary reference frame. With the park transform ($dq0$), the reference frame

now spins with the current vector such that the vector appears stationary, giving us DC terms to control, Id and Iq.

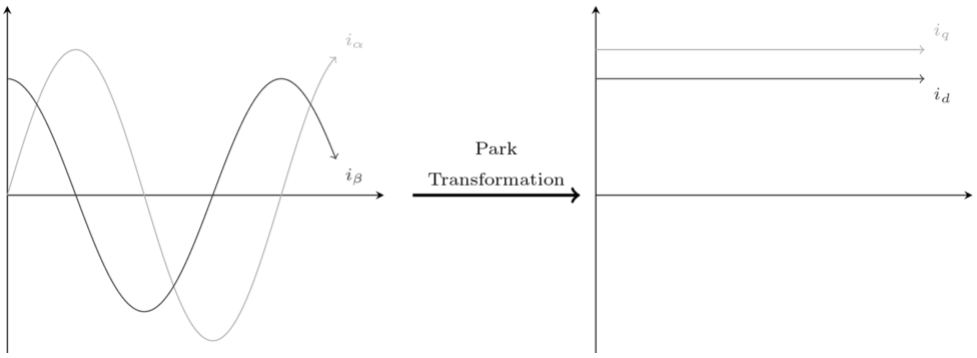


Figure 2.2: Illustration of transitioning from $\alpha - \beta$ frame to $d - q$ frame.

Now that we can translate our currents into something easier to control the FOC process looks like sample my current sense ADCs, transform from abc to $dq0$ and then do some control math.

2.2 Physical Interpretation

One of the reasons these transforms make so much sense and are so useful is because of the way they relate to the motor itself. I'll limit this discussion to surface permanent magnet motors as they're simpler than IPMs when discussing maximum torque per amp and field weakening. As this figure shows, the direct

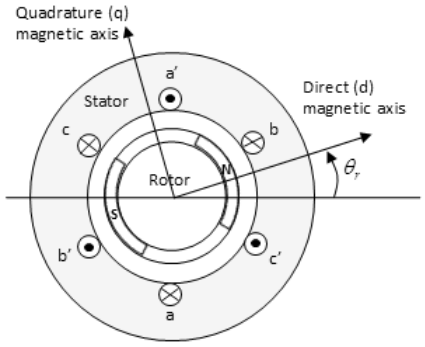


Figure 2.3: Schematic of a surface PM motor with d and q axis illustrated

axis (d for short) aligns with the rotor permanent magnet while the quadrature axis is halfway between the poles of the magnet (for a 2 pole motor that angle is 90°, for a 4 pole it's 45° etc.). When current flows exclusively on the d

axis, all the magnetic force generated also appears on that axis and as a result, no torque is produced. In contrast, if I put all of the current on the q-axis I get max torque per amp as none of the current I put into the motor ends up on the d-axis aligned with the poles of the motor. Thus my objective with designing an effective inverter is placing all of my current on the q-axis and none on the d-axis to maximuize torque per amp (a common desire for a racecar application).

2.2.1 Field Weakening

However, current on the d-axis isn't completely useless. Motors in electric vehicles often find themselves connected to high gear ratios to reduce the amount of stall torque and current that appear frequently for cars stopping and going in the city. Once the car reaches highway speeds though, the battery may run out of volts since the back-emf voltage is proportional to the speed of the motor. One solution to this is to put current on the negative d-axis, to oppose the magnetic fields generated by the permanent magnetics. Field weakening acts as a sort of transformer where, by putting energy into the system, you increase the speed the motor can spin but equivalently reduce the torque it can produce. This reduces the motor's efficiency as you're putting in energy and not getting an increase and power which is why IPMs are attractive as current on their d-axis does produce useful torque.

Chapter 3

Controls

Now that we know what to control and the next question is how to kick the system and develop a complete system. At a first order, by PWM'ing the half bridges on each of the phases we can generate a sine wave with the average of each period (which the motor gives us for free since it's an inductive load).

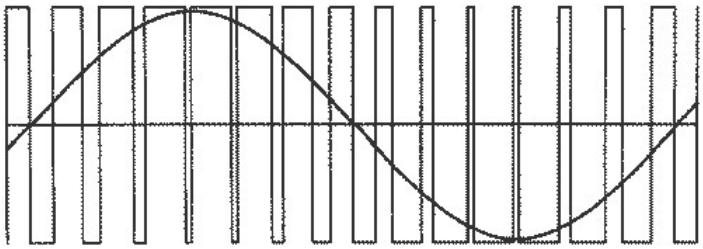


Figure 3.1: Illustration of using PWM to trace a sine wave

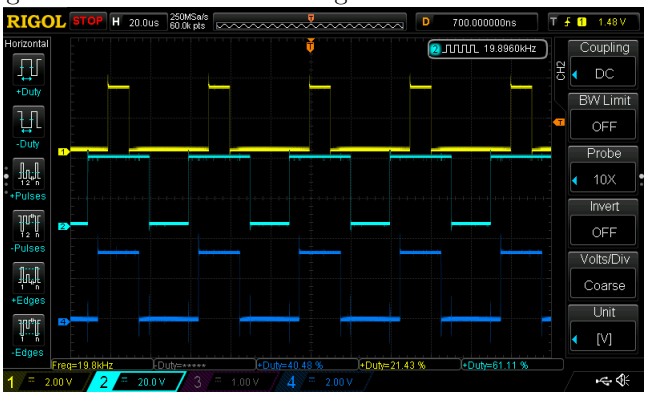


Figure 3.2: Example of 3-phases of center-aligned PWS

So our controls has to read in the currents going into the motor (our control

variable for FOC), and genearate a duty cycle command for the voltages on each of the phases. Luckily there are inverse variants of the clarke and park transforms so this task isn't too hard. A control period looks like sampling the currents on the motor phases, transforming them to the $d-q$ axis, feeding them into a PI controller, taking the inverse to get $a-b-c$ voltage commands and then translating that into a duty cycle command for the half bridges (whether through space vector modulation or sinusoidal PWM, I chose the latter). The overall system block diagram looks like this.

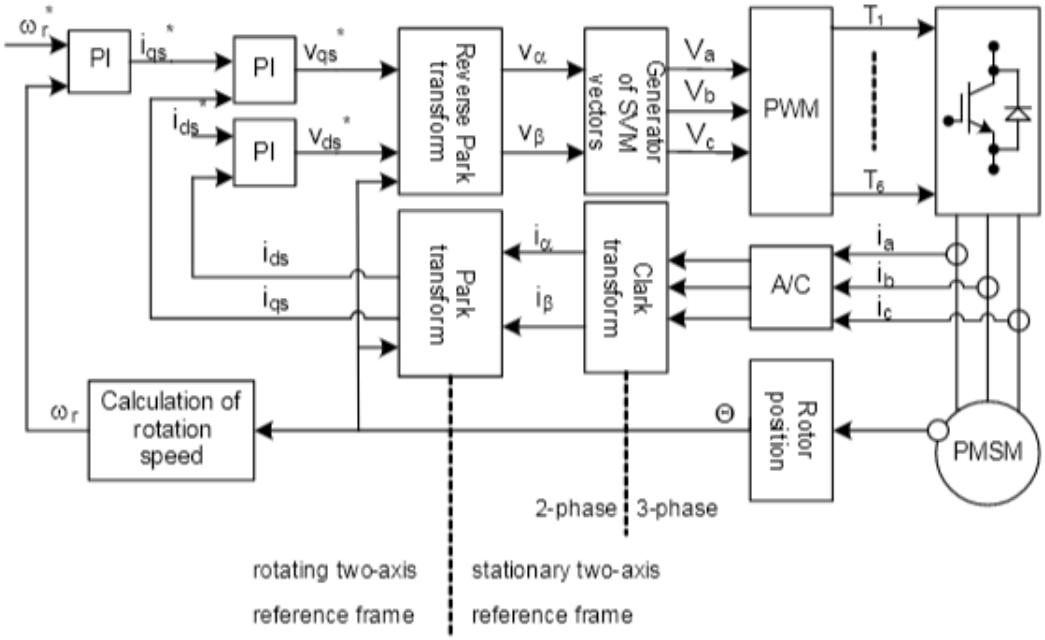


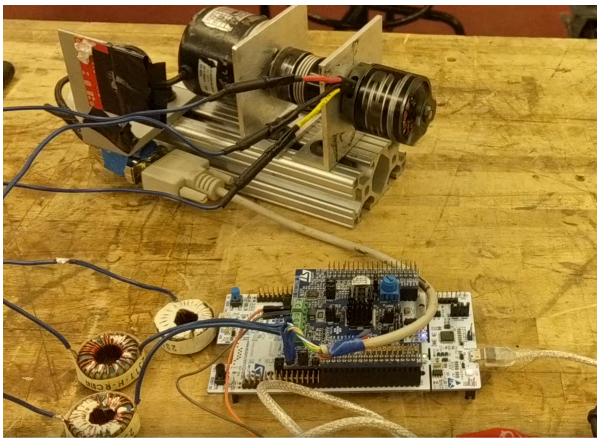
Figure 3.3: Big picture control block to implement Field Oriented control

Although this example control block diagram shows speed control I to only control torque (and thus only iq).

Chapter 4

Implementation

To create my motor controller I used an STM32F413 microcontroller (the same chip we use on our car). I paired this with a motor controller shield also by STMicro whic gave me half bridges (STM L6230), current-sense resistors and amplifiers, and an encoder connection. I used a prop-drive sinusoidal SPM motor set up on a jig lent to me by some MITERS friends. Fully set-up the system looks like this:

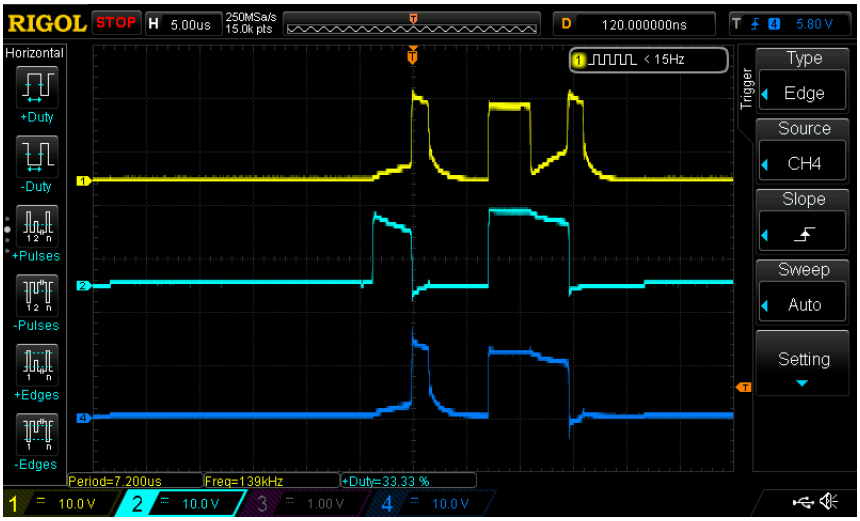


Every switching period (which ended up at around 5 kHz), I sampled my ADC (which was multiplexed over multiple channel which turned out to be a problem), transformed my currents from the abc frame to the $d-q$ frame, fed them into my PI block, inverse transformed from $d-q$ to abc then scaled and shifted that resulting control variable so it was centered around 50 (for 50% duty cycle) and ranged from about 20% to 80%. The STM uses one Timer dedicated for PWM generation and interrupting every switching period and a second to decode the encoder.

Chapter 5

Bugs and Challenges

This system was full of lots of bugs that I had to fix to get it working so here are some of the most impactful ones. First, one of the strangest modifications I had to add to my motor was the series inductance. The half-bridge IC has a pretty low curret-limit (≈ 2 A) which caused my phases to go low in the middle of the switching cycle. This led to some pretty strange behaviour but adding



the series inductance reduced the current into the motor and helped stabilized its performance.

One thing I grappled a lot with was getting the ADCs to perform well. The STM has three modes, polling, interrupt, and DMA, to interface with its ADCs and the mode I spent the most time fighting (polling) with would not work with what I intended to use it for. After finding this information in the corner of a datasheet I switched modes and was able to get the adcs to feedback on my currents up and running.

Another register for the timer also lead to some bugs where my timer interrupt would trigger in the middle of a period, updating the register that set the duty cycle in the middle of a switching period, also creating un-desirable behaviour. Lastly I initially did not enable the PLLs on the chip, operating at a much lower clock frequency than maximum (100 MHz). This caused my ISRs to take really long and not finish before the next switching period, causing more problems.

Chapter 6

Conclusion

Motor control is a very intriguing space because it's vital to the performance of EVs and is a hard problem to solve, especially when running at high RPMs. By driving the power electronics effectively, we can reduce the size and increase the efficiency of our cars and get as much out of them as possible. Through two smart transforms, we turn complicated AC waveforms into constant vectors that are much easier to understand and control in the PID world and have important physical implications for motors.