Ethan Russell

Capstone Project Proposal

For my CS capstone, I want to implement a field oriented controller of a permanent magnet motor. A field oriented controller uses a sensor to detect rotor position (I'll be using an encoder first, and then see if I can use the measured voltage on each phase instead) and measures the current on three phases to make sure that a motor is spinning at optimal efficiency and with the commanded torque. There are four parts to a working field oriented controller:

-Rotor position – to find an accurate velocity at any given time, the quantized position input from an optical rotary encoder (or back-EMF if I decide work on a sensorless implementation) is integrated using an alpha-beta filter which uses a mathematical model of the system to interpolate a higher-resolution position of the rotor.

-Transform the spinning flux vector (the current going through the three phases in the stator) into a reference frame relative to the rotor. This is done first with a Clarke transformation, to turn the three phase vector into a two-phase (sin/cos) reference frame. Then, we execute the Park transformation on the resulting X/Y axis of the flux vector using the rotor position to give two relatively time-constant values: quadrature axis current and direct axis current – each proportional to flux in each axis relative to the rotor. Develop a lookup table for these values to make processing faster.

-PI Control loop using the stator current vector and expected rotor position and generate error signals for the future direct and quadrature voltages so that we have the ideal current going to the quadrature axis with respect to the rotor, and none going to the direct axis, so that we always have current going into torque instead of wasted heat, and

the flux being driven by the stator is always 90 degrees from the rotor position. This doesn't need to be a PID control loop because there will be no phase delay in this system.

-Transform the voltages generated by the control loop back into three phase voltages using the reverse park transformation. With the stationary three-phase reference frame, output PWM on the three MOSFET bridges.

Once the field oriented controller works, I want to make a servo controller. An outer PID control loop gives torque commands to the field oriented controller to drive the motor to certain speeds or positions. This will require tuning with a certain motor/inertial disk to make sure that it stops at the requested position as efficiently as possible and without oscillating. There are also several other features of the controller I want to add to make the system better than commercially available brushless servo drivers. It would have cutoffs into sensorless (back-EMF sensed) control at medium rpms, and then onto trapezoidal commutation at much higher (10k+) rpms. This would allow smooth and fine low RPM control for servos control, but would also allow the motors to reach the peak of their torque/efficiency curves.

To develop a display for this system, I will rig up a simple robot that uses a sensor to control the motor's motion. If I can get three motors working, I could make a floating camera gimbal that would be stabilized. If I can only drive a single motor, I will make a power steering sort of system that would amplify a user's torque input onto a heavy load.

All of the software will be written on a family of processors I have never used before: STM32F4 family ARM chips by ST microelectronics. I chose this processor because of its control over many PWM outputs and timers and the fact that it has much more processing power than the 8-bit microcontrollers I'm used to.

The power electronics will be on a three phase MOSFET bridge board I have already designed and built. The motor will be a simple hobby brushless motor fitted with an optical servo and an inertial weight for tuning the servo loop.

I'm envisioning several challenges:

-I have to get a working STM32 toolchain working, and I have to understand its registers/timers

-I need to figure out accurate positions of the magnets on the rotor with respect to the stator and make a position table

-I need to figure out and understand the math involved with alpha-beta filters

-I need to figure out and understand the math of Clarke/Park transformations

-I need to figure out how to tune control loops – both the inner and outer loops

-I need to get the hardware (mechanical and electrical) systems fully operational

Timeline (out of 14 weeks not counting spring break):

Week 1: Get a working STM32 setup with PWM outputs to all three MOSFET phases, digital inputs into an encoder with interrupts, three phase current analog input, and three phase analog voltage input

Week 2: Figure out a lookup table for the measured encoder position (angle of quadrature/direct axis relative to the real world sensors) and finish a working mechanical testbed.

Week 3: Develop an alpha/beta filter to figure out accurate rotor position at high speeds. Get trapezoidal commutation to spin the motor.

Week 4: Develop park transformation and lookup table and a reverse park transformation/lookup table

Week 5: Develop PI control loop and figure out tuning systems

Week 6: Get PI control loop to accurately make voltage outputs and transform into the three phases. FOC should work at low-medium speeds. Graph real world rotor/stator position/motor current measurements

Week 7: Work on adapting FOC implementation to sensorless/back-EMF input

Week 8: Build outer PID control loop

Week 9: Begin building presentation graphics from graphs. Tune outer control loop.

Week 10: Finish up servo tuning. Get position control to be as fast as possible with physical potentiometers as tuning parameters and build graphic displaying the differences

Week 10: Build mechanical display – gimbal or power steering example.

Week 11: Add functionality for RPM cutoffs

Week 12: Touch up software/hardware

Week 13: Presentation preparation

Week 14: Presentation preparation