Summary of Motor Control

Jingren XIE

Introduction

This document is to give a tutorial of how to design a motor control system and how to implement and test the control system in an appropriate fashion. Permanent Magnetic Synchronous Motor (PMSM) is widely used in the industry for it has high power density and it is relatively easy to control. In the sensorless vector control, there are three blocks using the rotor position information: calculate i_d and i_q using park transformation; calculate v_α and v_β using the inverse park transformation; and rotor speed calculation. There are four chapters in the following to elaborate Motor Control.

Chapter 1

Development of the correct machine model through mathematical model is of importance. The mathematical model will determine control and drive and way of computation. We will represent the mathematical model as follows.

Mathematical Model for Generic PMSM

Voltage Equations in dq reference frame

$$\begin{cases} u_d = R_s i_d + p\psi_d - \omega\psi_q \\ u_q = R_s i_q + p\psi_q + \omega\psi_d \end{cases}$$

Flux Equation

$$\begin{cases} \psi_d = L_d i_d + \psi_f \\ \psi_q = L_q i_q \end{cases}$$

Substituting flux equation into voltage equation gives

$$\begin{cases} u_d = R_s i_d + L_d p i_d - \omega L_q i_d \\ u_q = R_s i_q + L_q p i_q + \omega L_d i_q + \omega\psi_f \end{cases}$$

Torque Equation

$$T_e = n_p[\psi_f i_q + (L_d - L_q) i_d i_q]$$

Motion Equation

$$T_e - T_L = J\frac{d\omega}{dt} + B\omega$$

In sensorless vector control, the most important thing is to estimate the position and speed accurately. There are now three major categories to estimate the rotor position and speed.

Indirect position sensing methods

We can estimate the position from the sensed position-related quantities, e.g. back EMF components. The instantaneous magnitude of the back EMF, which is a function of rotor position. An open-loop starting procedure is needed.

Model-based methods

We can use the fundamental-frequency model of PMSM, measured stator current to estimate the rotor position information. This is widely used and is effective for medium and high speed applications. This can be two different categories: open-loop calculation and closed loop observers. The open-loop position/speed estimation methods are very straightforward and easy to implement. These methods behave like real-time dynamic models of the PMSMs. Based on the dynamic model, rotor flux can be calculated , from which the rotor position and speed information can be extracted.

The control inputs and the output tracking error are used as the inputs to the observer. the closed loop observer can be viewed as an adaptive filter, which has a good disturbance rejection property and good robust.

We write the equation in a matrix format.

$$\begin{bmatrix} u_d \\ u_q \end{bmatrix} = \begin{bmatrix} R_s & -\omega L_q \\ \omega L_d & R_s \end{bmatrix} \begin{bmatrix} i_d \\ i_q \end{bmatrix} + \begin{bmatrix} L_d & 0 \\ 0 & L_q \end{bmatrix} p \begin{bmatrix} i_d \\ i_q \end{bmatrix} + \omega \psi_f \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

Using the inverse Park Transformation, the PMSM model can be modeled in the αβ stationary reference frame

$$\begin{bmatrix} u_\alpha \\ u_\beta \end{bmatrix} = \begin{bmatrix} R_s & 0 \\ 0 & R_s \end{bmatrix} \begin{bmatrix} i_\alpha \\ i_\beta \end{bmatrix} + \begin{bmatrix} L + \Delta L\cos(2\theta) & \Delta L\sin(\theta) \\ \Delta L\sin(\theta) & L - \Delta L\cos(2\theta) \end{bmatrix} p \begin{bmatrix} i_\alpha \\ i_\beta \end{bmatrix} + \omega \psi_f \begin{bmatrix} -\sin(\theta) \\ \cos(\theta) \end{bmatrix}$$

For a nonsalient pole PMSM, the model can be simplified as follows

$$\begin{bmatrix} u_\alpha \\ u_\beta \end{bmatrix} = \begin{bmatrix} R_s & 0 \\ 0 & R_s \end{bmatrix} \begin{bmatrix} i_\alpha \\ i_\beta \end{bmatrix} + \begin{bmatrix} L & 0 \\ 0 & L \end{bmatrix} p \begin{bmatrix} i_\alpha \\ i_\beta \end{bmatrix} + \omega \psi_f \begin{bmatrix} -\sin(\theta) \\ \cos(\theta) \end{bmatrix}$$

For a salient pole PMSM, we rewrite the equation in dq reference frame.

$$\begin{bmatrix} u_d \\ u_q \end{bmatrix} = \begin{bmatrix} R_s & -\omega L_q \\ \omega L_q & R_s \end{bmatrix} \begin{bmatrix} i_d \\ i_q \end{bmatrix} + \begin{bmatrix} L_d & 0 \\ 0 & L_d \end{bmatrix} p \begin{bmatrix} i_d \\ i_q \end{bmatrix} + E \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$
$$E = \omega \left[(L_d - L_q) i_d + \psi_f \right] - (L_d - L_q)(p i_q)$$

We transform it into the αβ stationary reference frame

$$\begin{bmatrix} u_\alpha \\ u_\beta \end{bmatrix} = \begin{bmatrix} R_s & -\omega(L_d - L_q) \\ \omega(L_d - L_q) & R_s \end{bmatrix} \begin{bmatrix} i_\alpha \\ i_\beta \end{bmatrix} + \begin{bmatrix} L_d & 0 \\ 0 & L_d \end{bmatrix} p \begin{bmatrix} i_\alpha \\ i_\beta \end{bmatrix} + E \begin{bmatrix} -\sin(\theta) \\ \cos(\theta) \end{bmatrix}$$

Open Loop Calculation

Back EMF-based Methods

The EMF components is calculated as

$$\begin{cases} E_\alpha = -u_\alpha + R_s i_\alpha + \omega(L_d - L_q) i_\beta + L_d p i_\alpha \\ E_\beta = +u_\beta - R_s i_\beta + \omega(L_d - L_q) i_\alpha - L_d p i_\beta \end{cases}$$

The rotor position can be calculated as

$$\theta = \tan^{-1}(E_\alpha / E_\beta)$$

The performance of this method is subjected to the accuracy of the sensed current and machine parameters.

Flux Linkage-based Methods

At the steady state, where $di_\alpha/dt = 0$ and $di_\beta/dt = 0$, the stator and rotor flux vectors rotate synchronously. The position angle of the stator flux can be calculated, the rotor position can be determined.

$$\begin{cases} \psi_{s\alpha} = \int (u_\alpha - R_s i_\alpha) dt \\ \psi_{s\beta} = \int (u_\beta - R_s i_\beta) dt \end{cases}$$

$$\begin{cases} \psi_{r\alpha} = \psi_{s\alpha} - L i_\alpha \\ \psi_{r\beta} = \psi_{s\beta} - L i_\beta \end{cases}$$

The position can be calculated as

$$\theta = \tan^{-1}(\psi_{r\alpha} / \psi_{r\beta})$$

The accuracy highly depends on the quality and accuracy of the voltage and

current measurements. A high pass filter is recommended to eliminate DC component.

All of these methods are limited by the numerical resolution , which depends on the sampling frequency and control loop frequency of the control system. The accuracy of the methods strongly depends on the accuracy of the machine parameters current and voltage measurements.

Closed Loop Observers

Linear State Observers

$$\frac{d}{dt}\begin{bmatrix} i_\alpha \\ i_\beta \end{bmatrix} = \begin{bmatrix} R_s / L & 0 \\ 0 & R_s / L \end{bmatrix}\begin{bmatrix} i_\alpha \\ i_\beta \end{bmatrix} + \frac{1}{L}\left(\begin{bmatrix} u_\alpha \\ u_\beta \end{bmatrix} - \begin{bmatrix} e_\alpha \\ e_\beta \end{bmatrix}\right)$$

A disturbance observer can be designed as

$$\frac{d}{dt}\begin{bmatrix} \hat{i}_\alpha \\ \hat{i}_\beta \end{bmatrix} = \begin{bmatrix} R_s / L & 0 \\ 0 & R_s / L \end{bmatrix}\begin{bmatrix} i_\alpha \\ i_\beta \end{bmatrix} + \frac{1}{L}\left(\begin{bmatrix} u_\alpha \\ u_\beta \end{bmatrix} - \begin{bmatrix} \hat{e}_\alpha \\ \hat{e}_\beta \end{bmatrix}\right)$$

$$\frac{d}{dt}\begin{bmatrix} \hat{e}_\alpha \\ \hat{e}_\beta \end{bmatrix} = G\frac{d}{dt}\begin{bmatrix} \hat{i}_\alpha - i_\alpha \\ \hat{i}_\beta - i_\beta \end{bmatrix}$$

The position can be calculated as

$$\hat{\theta} = \tan^{-1}(-\hat{e}_\alpha / \hat{e}_\beta)$$

The rotor position and speed information can be extracted from these estimated states using an appropriate observer or algorithm.

Besides the arctangent algorithm, the phase-locked loop and the angle tracking observer are effective methods.

Dynamic Model of a Salient Pole PMSM

$$\begin{cases} u_\alpha = R_s i_\alpha + L_d p(i_d \cos(\theta)) - L_q p(i_q \sin(\theta)) + \psi_f p(\cos(\theta)) \\ u_\beta = R_s i_\beta + L_d p(i_d \sin(\theta)) + L_q p(i_q \cos(\theta)) + \psi_f p(\sin(\theta)) \end{cases}$$

which can be written as follows

$$\begin{bmatrix} u_\alpha \\ u_\beta \end{bmatrix} = R_s\begin{bmatrix} i_\alpha \\ i_\beta \end{bmatrix} + Lp\begin{bmatrix} i_\alpha \\ i_\beta \end{bmatrix} + \Delta Lp\left(\begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix}\begin{bmatrix} i_d \\ i_q \end{bmatrix}\right) + \omega\psi_f\begin{bmatrix} -\sin(\theta) \\ \cos(\theta) \end{bmatrix}$$

The flux part can be rewritten as

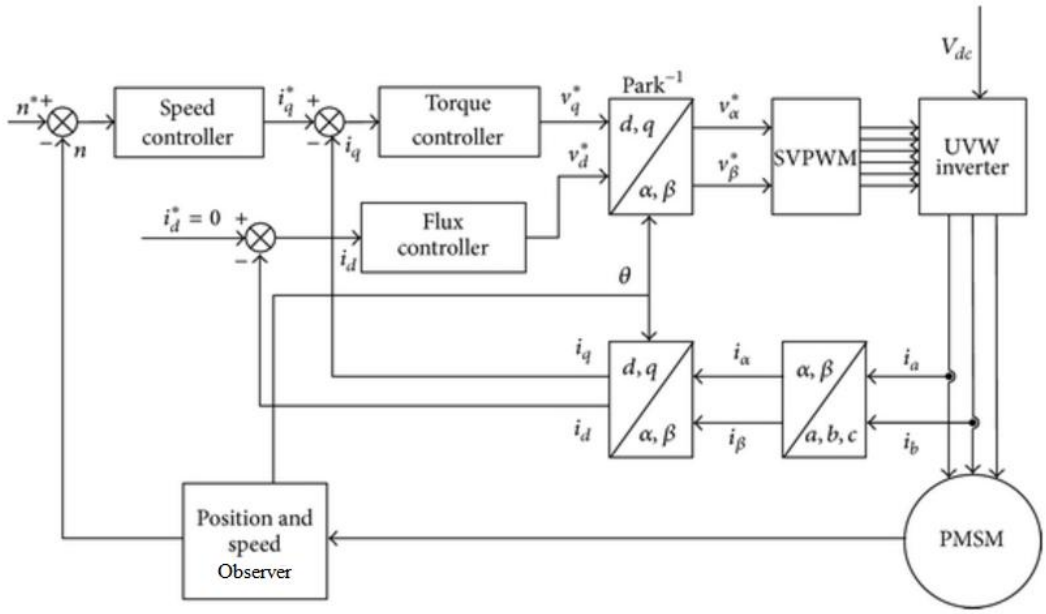
$$\Delta Lp\left(\begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix}\begin{bmatrix} i_d \\ i_q \end{bmatrix}\right) + \omega\psi_f\begin{bmatrix} -\sin(\theta) \\ \cos(\theta) \end{bmatrix} = \begin{bmatrix} -\Delta Lp(i_\alpha) + p\left[\left(\psi_f + 2\Delta L i_d\right)\cos(\theta)\right] \\ -\Delta Lp(i_\beta) + p\left[\left(\psi_f + 2\Delta L i_d\right)\sin(\theta)\right] \end{bmatrix}$$

Saliency-based methods

The rotor position information is extracted from the position-dependent machine saliency and high frequency signal is injected.
Hence, the observer design has become the core part of position estimation.

Chapter 2

Field-oriented Control System can be shown as follows.



The basic idea is to design controllers to control the motor which means that we solve the mathematical motor model by estimating the velocity and position using the feedback currents and also calculate the voltage required.
There are two parts in the mathematical model: linear part and coupled part. We rewrite the model as follows.

$$\begin{cases} u_d = \hat{u}_d + u_q^c \\ u_q = \hat{u}_q + u_d^c \end{cases}$$

$$\begin{cases} \hat{u}_d = R_s i_d + L_d p i_d \\ \hat{u}_q = R_s i_q + L_q p i_q \end{cases}$$

The above equations are linear differential equations.
We adopt id=0 control, then the torque can be approximated by

$$T_e = k_t i_q$$

$$k_t = n_p \psi_f$$

Current Loop

The linear part of the voltage equation can be transformed into s domain, which gives

$$U(s) = R_s I(s) + L_s s I(s)$$

$$G(s) = \frac{I(s)}{U(s)} = \frac{1}{L_s s + R_s} = \frac{\frac{1}{R_s}}{\frac{L_s}{R_s} s + 1} = \frac{K_s}{T_s s + 1}$$

Some time delay caused by PWM outputs, sampling, and filtering are denoted as T_{pwm}, T_{sam}, T_{filter}. PI controller can be

$$PI(s) = Kcp + \frac{Kci}{s} = Kcp \frac{\tau_c s + 1}{\tau_c s}$$

Hence, the transfer function of current loop can be

$$G_i(s) = \frac{1}{T_\Sigma s + 1} \frac{K_s}{T_s s + 1} \left(kcp \frac{\tau_c s + 1}{\tau_c s} \right)$$

$T_s >> T_\Sigma$, $let\ \tau_c = T_s$

Then we get

$$G_i(s) = \frac{K_s kcp / \tau_c}{s(T_\Sigma s + 1)} = \frac{K}{Ts^2 + s}$$

$KT < 1$, $let\ KT = 0.5$

$$\tau_c = T_s,\ kcp = \frac{T_s}{2T_\Sigma K_s}$$

Speed Loop

The closed loop transfer function of current loop is

$$H(s) = \frac{K}{Ts^2 + s + K}$$

$$T = T_\Sigma << 1$$

Hence, the transfer function can be

$$H(s) = \frac{1}{2T_\Sigma s + 1}$$

The motion equation can be

$$M(s)=\frac{\frac{1}{J}}{s+\frac{B}{J}}=\frac{b}{s+a}$$

$$PI(s)=ksp\left(\frac{\tau_s s+1}{\tau_s s}\right)$$

Hence, the transfer function can be

$$G_s(s)=ksp\left(\frac{\tau_s s+1}{\tau_s s}\right)K_t\frac{1}{T_{s\Sigma}s+1}\frac{b}{s+a}$$

$$T_{s\Sigma}=2T_\Sigma+T_{filter}+T_{sam}$$

$$\frac{1}{a}>\frac{1}{K}$$

Hence,

$$\tau_s=\frac{1}{a},\ ksp=\frac{K}{2}\frac{1}{K_t b}$$

The discrete system computation

$$Iu = 8 * Isin(wt) = Asin(\omega t) \tag{2-1}$$

$$Iv = 8 * Isin(wt-\varphi) = Asin(\omega t-\varphi) \tag{2-2}$$

其中 $A>0$ 是幅值，φ是在 120°波动的相位。

定点数的范围是$[-32767,32767]$， 那么 $A\in[0,32767]$。

采用等幅值的 Clark 变换

$$i_\alpha = i_u \tag{2-3}$$

$$i_\beta = \frac{(i_u+2i_v)}{\sqrt{3}} \tag{2-4}$$

$$i_\alpha = i_u = Asin(\omega t) \tag{2-5}$$

$$i_\beta = \frac{(i_u+2i_v)}{\sqrt{3}} = \frac{A\sqrt{m^2+n^2}}{\sqrt{3}}sin\,(\omega t-\varphi_1) \tag{2-6}$$

Clark 变换

$$i_\alpha = i_u$$

$$i_\beta = \frac{(i_u+2i_v)}{\sqrt{3}} = \frac{A}{\sqrt{3}}(sin(\omega t)+2sin(\omega t-\varphi))$$

$$= \frac{A}{\sqrt{3}}((1 + 2\,cos\,(\varphi)\,)\,sin\,(\omega t) - 2\,sin\,(\varphi)\,cos\,(\omega t))$$

$$= \frac{A\sqrt{m^2 + n^2}}{\sqrt{3}} sin\,(\omega t - \varphi_1)$$

$$\begin{bmatrix} I_\alpha \\ I_\beta \end{bmatrix} = \begin{bmatrix} A_1 sin(\omega t) \\ A_1\,sin\,(\omega t - \varphi_1) \end{bmatrix}$$

$$\varphi_1 = arctan\left(\frac{n}{m}\right)$$

$$m\ =\ 1 + 2cos(\varphi)$$

$$n\ =\ 2sin(\varphi)$$

$$0 \le \frac{\mathrm{A}\sqrt{\mathrm{m}^2 + \mathrm{n}^2}}{\sqrt{3}} \le 32767$$

将静止坐标投射到同步旋转坐标

$$i_d = i_\alpha \cos\theta + i_\beta \sin\theta \qquad (2\text{-}7)$$

$$i_q = -\,i_\alpha \sin\theta + i_\beta \cos\theta \qquad (2\text{-}8)$$

假设估计出来的位置$\theta = \omega t$ 是准确的，那么

$$i_d = A_1\sqrt{m_1^2 + n_1^2}sin(2\omega t - \varphi_2) + P \qquad (2\text{-}9)$$

$$i_q = A_1\sqrt{m_1^2 + n_1^2}sin(2\omega t + \varphi_3) + Q \qquad (2\text{-}10)$$

如果估计出来的位置$\theta = \omega t - \Delta\theta$，那么

$$i_d = A_1\sqrt{m_2^2 + n_2^2}sin(2\omega t + \varphi_4) + P_1 \qquad (2\text{-}11)$$

$$i_q = A_1\sqrt{m_2^2 + n_2^2}sin(2\omega t + \varphi_5) + Q_1 \qquad (2\text{-}12)$$

假设估计出来的位置$\theta = \omega t$ 是准确的，那么

Park 变换

$$\begin{bmatrix} I_d \\ I_q \end{bmatrix} = \begin{bmatrix} cos\,(\omega t) & sin\,(\omega t) \\ -\,sin\,(\omega t) & cos\,(\omega t) \end{bmatrix} \begin{bmatrix} I_\alpha \\ I_\beta \end{bmatrix}$$

$$= \begin{bmatrix} cos\,(\omega t) & sin\,(\omega t) \\ -\,sin\,(\omega t) & cos\,(\omega t) \end{bmatrix} \begin{bmatrix} A_1 sin(\omega t) \\ A_1\,sin\,(\omega t - \varphi_1) \end{bmatrix}$$

$$= \begin{bmatrix} A_1 sin(\omega t)\cos\,(\omega t) + A_1\,sin\,(\omega t - \varphi_1)\,sin\,(\omega t) \\ -\,A_1\,sin\,(\omega t)\,sin(\omega t) +\ A_1\,sin\,(\omega t - \varphi_1)\,cos\,(\omega t) \end{bmatrix}$$

$$= \begin{bmatrix} \frac{A_1}{2}(1-sin\varphi_1)sin(2\omega t) - \frac{A_1}{2}cos\varphi_1 cos\,(2\omega t) + \frac{A_1}{2}cos\varphi_1 \\ \frac{A_1}{2}cos\varphi_1 sin\,(2\omega t) + \frac{A_1}{2}(1-sin\varphi_1)cos\,(2\omega t) - \frac{A_1}{2}(1+sin\varphi_1) \end{bmatrix}$$

$$= A_1 \begin{bmatrix} \sqrt{m_1^2+n_1^2}sin(2\omega t-\varphi_2)+P \\ \sqrt{m_1^2+n_1^2}sin(2\omega t+\varphi_3)+Q \end{bmatrix}$$

$$= \begin{bmatrix} \frac{1}{2}(1-sin\varphi_1) & -\frac{1}{2}cos\varphi_1 \\ \frac{1}{2}cos\varphi_1 & \frac{1}{2}(1-sin\varphi_1) \end{bmatrix} \begin{bmatrix} A_1 sin(2\omega t) \\ A_1 cos\,(2\omega t) \end{bmatrix} + A_1 \begin{bmatrix} \frac{1}{2}cos\varphi_1 \\ -\frac{1}{2}(1+sin\varphi_1) \end{bmatrix}$$

$$= \begin{bmatrix} m_1 & -n_1 \\ n_1 & m_1 \end{bmatrix} \begin{bmatrix} A_1 sin(2\omega t) \\ A_1 cos\,(2\omega t) \end{bmatrix} + A_1 \begin{bmatrix} P \\ Q \end{bmatrix}$$

$$\leq A_1 \begin{bmatrix} \sqrt{m_1^2+n_1^2}+|P| \\ \sqrt{m_1^2+n_1^2}+|Q| \end{bmatrix}$$

$$0 \leq A_1\left(\sqrt{m_1^2+n_1^2}+|P|\right) \leq 32767$$

$$0 \leq A_1\left(\sqrt{m_1^2+n_1^2}+|Q|\right) \leq 32767$$

如果估计出来的位置$\theta = \omega t - \Delta\theta$，那么

$$\begin{bmatrix} I_d \\ I_q \end{bmatrix}_{\Delta\theta} = \begin{bmatrix} cos\,(\omega t-\Delta\theta) & sin\,(\omega t-\Delta\theta) \\ -sin\,(\omega t-\Delta\theta) & cos\,(\omega t-\Delta\theta) \end{bmatrix} \begin{bmatrix} A_1 sin(\omega t) \\ A_1\,sin\,(\omega t-\varphi_1) \end{bmatrix}$$

$$= A_1 \begin{bmatrix} sin(\omega t)\,cos\,(\omega t-\Delta\theta)+sin\,(\omega t-\varphi_1)\,sin\,(\omega t-\Delta\theta) \\ -sin\,(\omega t)\,sin(\omega t-\Delta\theta)+sin\,(\omega t-\varphi_1)\,cos\,(\omega t-\Delta\theta) \end{bmatrix}$$

$$= A_1 \begin{bmatrix} \frac{1}{2}(1-sin\varphi_1-\Delta\theta cos\varphi_1)sin(2\omega t) - \frac{1}{2}(cos\varphi_1+\Delta\theta-\Delta\theta sin\varphi_1)\,cos\,(2\omega t) + \frac{1}{2}(cos\varphi_1+\Delta\theta+\Delta\theta sin\varphi_1) \\ \frac{1}{2}(cos\varphi_1+\Delta\theta-\Delta\theta sin\varphi_1)\,sin\,(2\omega t) + \frac{1}{2}(1-sin\varphi_1-\Delta\theta cos\varphi_1)\,cos\,(2\omega t) - \frac{1}{2}(1+sin\varphi_1-\Delta\theta cos\varphi_1) \end{bmatrix}$$

$$= A_1 \begin{bmatrix} \sqrt{m_2^2+n_2^2}sin(2\omega t-\varphi_4)+P_1 \\ \sqrt{m_2^2+n_2^2}sin(2\omega t+\varphi_5)+Q_1 \end{bmatrix}$$

$$= \begin{bmatrix} \frac{1}{2}(1-sin\varphi_1-\Delta\theta cos\varphi_1) & -\frac{1}{2}(cos\varphi_1+\Delta\theta-\Delta\theta sin\varphi_1) \\ \frac{1}{2}(cos\varphi_1+\Delta\theta-\Delta\theta sin\varphi_1) & \frac{1}{2}(1-sin\varphi_1-\Delta\theta cos\varphi_1) \end{bmatrix} \begin{bmatrix} A_1 sin(2\omega t) \\ A_1 cos\,(2\omega t) \end{bmatrix}$$

$$+ A_1 \begin{bmatrix} \frac{1}{2}(cos\varphi_1+\Delta\theta+\Delta\theta sin\varphi_1) \\ -\frac{1}{2}(1+sin\varphi_1-\Delta\theta cos\varphi_1) \end{bmatrix}$$

$$= \begin{bmatrix} m_2 & -n_2 \\ n_2 & m_2 \end{bmatrix} \begin{bmatrix} A_1 sin(2\omega t) \\ A_1 \cos(2\omega t) \end{bmatrix} + A_1 \begin{bmatrix} P_1 \\ Q_1 \end{bmatrix}$$

$$\leq A_1 \begin{bmatrix} \sqrt{m_2^2 + n_2^2} + |P_1| \\ \sqrt{m_2^2 + n_2^2} + |Q_1| \end{bmatrix}$$

$$0 \leq A_1 \left(\sqrt{m_2^2 + n_2^2} + |P_1| \right) \leq 32767$$

$$0 \leq A_1 \left(\sqrt{m_2^2 + n_2^2} + |Q_1| \right) \leq 32767$$

与位置估计无偏差时比较，得到误差表达式

$$\begin{bmatrix} \Delta I_d \\ \Delta I_q \end{bmatrix} = \begin{bmatrix} I_d \\ I_q \end{bmatrix}_{\Delta\theta} - \begin{bmatrix} I_d \\ I_q \end{bmatrix}$$

$$= A_1 \begin{bmatrix} -\frac{1}{2}\Delta\theta cos\varphi_1 sin(2\omega t) - \frac{1}{2}(\Delta\theta - \Delta\theta sin\varphi_1)\cos(2\omega t) + \frac{1}{2}(\Delta\theta + \Delta\theta sin\varphi_1) \\ \frac{1}{2}(\Delta\theta - \Delta\theta sin\varphi_1) sin(2\omega t) - \frac{1}{2}\Delta\theta cos\varphi_1 \cos(2\omega t) + \frac{1}{2}\Delta\theta cos\varphi_1 \end{bmatrix}$$

$$= A_1 \begin{bmatrix} -\frac{1}{2}\Delta\theta cos\varphi_1 & -\frac{1}{2}(\Delta\theta - \Delta\theta sin\varphi_1) \\ \frac{1}{2}(\Delta\theta - \Delta\theta sin\varphi_1) & -\frac{1}{2}\Delta\theta cos\varphi_1 \end{bmatrix} \begin{bmatrix} sin(2\omega t) \\ cos(2\omega t) \end{bmatrix} + A_1 \begin{bmatrix} \frac{1}{2}(\Delta\theta + \Delta\theta sin\varphi_1) \\ \frac{1}{2}\Delta\theta cos\varphi_1 \end{bmatrix}$$

磁链估计连续域的表达式为

$$d\psi = U - RI \tag{2-13}$$

采用积分器和高通滤波器进行磁链的求解

$$d\psi \frac{1}{s} \frac{\tau s}{1+\tau s} = y \tag{2-14}$$

整理上式得到

$$d\psi \frac{\tau}{1+\tau s} = y \tag{2-15}$$

结合前向差分公式，将公式离散化

$$s = \frac{z-1}{T} \tag{2-16}$$

代入式（2-15），得到

$$y(k+1) = \frac{U(k) - RI(k)}{f} - \frac{\left(\frac{1}{\tau} - \frac{1}{T}\right)}{f} y(k) \tag{2-17}$$

$$y(k) = y(k+1) \tag{2-18}$$

磁链估计

$$d\psi(k)\frac{\tau}{1+\tau\frac{z-1}{T}}=y(k)$$

$$d\psi(k)\frac{\tau}{1+\tau\frac{z-1}{T}}=y(k)$$

$$(U(k)-RI(k))\tau=y\left(1+\tau\frac{z-1}{T}\right)$$

$$=\left(y(k)+\frac{\tau}{T}y(k+1)-\frac{\tau}{T}y(k)\right)$$

$$\frac{\tau}{T}y(k+1)=(U(k)-RI(k))\tau+\left(1-\frac{\tau}{T}\right)y(k)$$

$$y(k+1)=\frac{U(k)-RI(k)}{f}-\frac{\left(\frac{1}{\tau}-\frac{1}{T}\right)}{f}y(k)$$

$$y(k)=y(k+1)$$

将式（2-17）和式（2-18）应用到α和β轴上磁链的估计，考虑到式（2-17）和式（2-18）中的迭代项 $y(k)$，防止累计误差偏大，应进行相应的误差补偿， 得到如下表达式。

$$\psi_\alpha^{temp}(k+1)=\psi_\alpha^{temp}(k)+\frac{u_\alpha(k)*Vmax-\frac{R_s i_\alpha(k)}{64}-K_{gain}\psi_\alpha(k)}{f_{pwm}} \quad (2\text{-}19)$$

$$\psi_\alpha(k+1)=\psi_\alpha^{temp}(k+1)-\frac{i_\alpha(k)(L_D+L_Q)}{2^{13}} \quad (2\text{-}20)$$

$$\psi_\beta^{temp}(k+1)=\psi_\beta^{temp}(k)+\frac{u_\beta(k)*Vmax-\frac{R_s i_\beta(k)}{64}-K_{gain}\psi_\beta(k)}{f_{pwm}} \quad (2\text{-}21)$$

$$\psi_\beta(k+1)=\psi_\beta^{temp}(k+1)-\frac{i_\beta(k)(L_D+L_Q)}{2^{13}} \quad (2\text{-}22)$$

为了分析简单，将下标略去，得到如下表达式

$$\psi^{temp}(k+1)=\psi^{temp}(k)+\frac{u(k)*V_{max}-\frac{R_s i(k)}{64}-K_{gain}\psi(k)}{f_{pwm}} \quad (2\text{-}23)$$

$$\psi(k+1)=\psi^{temp}(k+1)-\frac{i(k)L}{8192} \quad (2\text{-}24)$$

$$\psi^{temp}(k)=\frac{1}{k_g}\left(V_{max}u(k)+\left(-\frac{R}{64}+\frac{k_g L}{8192}\right)i(k)\right) \quad (2\text{-}25)$$

$$\psi(k)=\frac{1}{k_g}\left(V_{max}u(k)-\frac{R}{64}i(k)\right) \quad (2\text{-}26)$$

在稳态时可以近似求解出式（2-23）和式（2-24）中$\psi(k)$和$\psi^{temp}(k)$的表达式。

磁链估计

$$\psi^{temp}(k+1)=\psi^{temp}(k)+\frac{u(k)*V_{max}-\frac{R_s i(k)}{64}-K_{gain}\psi(k)}{f_{pwm}}$$

$$\psi(k+1)=\psi^{temp}(k+1)-\frac{iL}{8192}$$

$$\begin{bmatrix}1 & 0\\ -1 & 1\end{bmatrix}\begin{bmatrix}\psi^{temp}(k+1)\\ \psi(k+1)\end{bmatrix}=\begin{bmatrix}1 & -\frac{k_g}{f}\\ 0 & 0\end{bmatrix}\begin{bmatrix}\psi^{temp}(k)\\ \psi(k)\end{bmatrix}+\begin{bmatrix}\frac{V_{max}}{f} & -\frac{R}{64f}\\ 0 & -\frac{L}{8192}\end{bmatrix}\begin{bmatrix}u(k)\\ i(k)\end{bmatrix}$$

假设稳态时，$\begin{bmatrix}\psi^{temp}(k+1)\\ \psi(k+1)\end{bmatrix}=\begin{bmatrix}\psi^{temp}(k)\\ \psi(k)\end{bmatrix}=\psi$，那么

$$\left(\begin{bmatrix}1 & 0\\ -1 & 1\end{bmatrix}-\begin{bmatrix}1 & -\frac{k_g}{f}\\ 0 & 0\end{bmatrix}\right)\psi=\begin{bmatrix}\frac{V_{max}}{f} & -\frac{R}{64f}\\ 0 & -\frac{L}{8192}\end{bmatrix}\begin{bmatrix}u(k)\\ i(k)\end{bmatrix}$$

$$\psi=\frac{1}{k_g}\begin{bmatrix}V_{max} & -\frac{R}{64}+\frac{k_g L}{8192}\\ V_{max} & -\frac{R}{64}\end{bmatrix}\begin{bmatrix}u(k)\\ i(k)\end{bmatrix}$$

$$\begin{bmatrix}\psi^{temp}(k)\\ \psi(k)\end{bmatrix}=\begin{bmatrix}\frac{1}{k_g}\left(V_{max}u(k)+\left(-\frac{R}{64}+\frac{k_g L}{8192}\right)i(k)\right)\\ \frac{1}{k_g}\left(V_{max}u(k)-\frac{R}{64}i(k)\right)\end{bmatrix}$$

结合 Clark 变换得到的α和β轴上的电流

$$\begin{bmatrix}I_\alpha\\ I_\beta\end{bmatrix}=\begin{bmatrix}A_1 sin(\omega k)\\ A_1\, sin\,(\omega k-\varphi_1)\end{bmatrix}$$

因为这里的$\begin{bmatrix}I_\alpha\\ I_\beta\end{bmatrix}$是正弦信号，假设电压信号也是正弦信号，那么可以得到

$$\begin{bmatrix}U_\alpha\\ U_\beta\end{bmatrix}=\begin{bmatrix}B_1 sin(\omega k-\varphi_6)\\ B_1\, sin\,(\omega k-\varphi_7)\end{bmatrix}$$

$$\psi_\alpha^{temp}(k)=\frac{1}{k_g}\left(V_{max}u(k)+\left(-\frac{R}{64}+\frac{k_g L}{8192}\right)i(k)\right)$$

$$=\frac{V_{max}}{k_g}u(k)+\left(-\frac{R}{64k_g}+\frac{L}{8192}\right)i(k)$$

$$=\frac{V_{max}}{k_g}B_1 sin(\omega k-\varphi_6)+\left(-\frac{R}{64k_g}+\frac{L}{8192}\right)A_1 sin(\omega k)$$

$$=M_1 sin(\omega k-\varphi_6)+N_1 sin(\omega k)$$

$$=(M_1 cos(\varphi_6)+N_1)sin(\omega k)-M_1 sin(\varphi_6)cos(\omega k)$$

$$=P_1\, sin\,(\omega k)-Q_1 cos(\omega k)$$

$$=\sqrt{P_1^2+Q_1^2}\, sin\,(\omega k-\varphi_7)$$

$$\varphi_7 = arctan\left(\frac{Q_1}{P_1}\right)$$

$$\psi_\beta^{temp}(k) = \frac{V_{max}}{k_g}u(k) + \left(-\frac{R}{64k_g} + \frac{L}{8192}\right)i(k)$$
$$= \frac{V_{max}}{k_g}B_1\,sin\,(\omega k - \varphi_7) + \left(-\frac{R}{64k_g} + \frac{L}{8192}\right)A_1\,sin\,(\omega k - \varphi_1)$$
$$= M_2\,sin\,(\omega k - \varphi_7) + N_2\,sin\,(\omega k - \varphi_1)$$
$$= (M_2 cos(\varphi_7) + N_2 cos(\varphi_1))sin(\omega k) - (M_2 sin(\varphi_7) + N_2 sin(\varphi_1))cos(\omega k)$$
$$= P_2\,sin\,(\omega k) - Q_2 cos(\omega k)$$
$$= \sqrt{P_2^2 + Q_2^2}\,sin\,(\omega k - \varphi_8)$$
$$\varphi_8 = arctan\left(\frac{Q_2}{P_2}\right)$$

$$\psi_\alpha(k) = \frac{V_{max}}{k_g}B_1 sin(\omega k - \varphi_6) - \frac{R}{64k_g}A_1 sin(\omega k)$$
$$= M_3 sin(\omega k - \varphi_6) - N_3 sin(\omega k)$$
$$= (M_3 cos(\varphi_6) - N_3)sin(\omega k) - M_3 sin(\varphi_6)cos(\omega k)$$
$$= P_3\,sin\,(\omega k) - Q_3 cos(\omega k)$$
$$= \sqrt{P_3^2 + Q_3^2}\,sin\,(\omega k - \varphi_9)$$
$$\varphi_9 = arctan\left(\frac{Q_3}{P_3}\right)$$

$$\psi_\beta(k) = \frac{V_{max}}{k_g}u(k) - \frac{R}{64k_g}i(k)$$
$$= \frac{V_{max}}{k_g}B_1\,sin\,(\omega k - \varphi_7) - \frac{R}{64k_g}A_1\,sin\,(\omega k - \varphi_1)$$
$$= M_4\,sin\,(\omega k - \varphi_7) - N_4\,sin\,(\omega k - \varphi_1)$$
$$= (M_4 cos(\varphi_7) - N_4 cos(\varphi_1))sin(\omega k) - (M_4 sin(\varphi_7) - N_4 sin(\varphi_1))cos(\omega k)$$
$$= P_4\,sin\,(\omega k) - Q_4 cos(\omega k)$$
$$= \sqrt{P_4^2 + Q_4^2}\,sin\,(\omega k - \varphi_{10})$$
$$\varphi_{10} = arctan\left(\frac{Q_4}{P_4}\right)$$

根据式（2-25）和式（2-26），可以求得相应的边界

速度位置估计

$$\omega = \Delta\theta\left(k_p + \frac{k_i}{s}\right) \tag{2-27}$$

$$\theta = \theta + \frac{\omega}{f_{pwm}} \tag{2-28}$$

采用磁链速度和位置进行估计

锁相环估计速度和位置

$$\psi_\alpha = \psi cos\theta_r$$

$$\psi_\beta = \psi sin\theta_r$$

$$\Delta\theta = \psi_\beta cos\theta - \psi_\alpha sin\theta$$

$$= \psi(sin\theta_r cos\theta - cos\theta_r sin\theta)$$

$$= \psi\, sin\,(\theta_r - \theta) \approx K(\theta_r - \theta)$$

采用 PI 求得速度

$$\omega = \Delta\theta\left(k_p + \frac{k_i}{s}\right)$$

前向差分公式 $s = \frac{(z-1)}{T}$

$$\omega(k) = \Delta\theta(k)\left(k_p + \frac{k_i}{\frac{(z-1)}{T}}\right)$$

$$= \Delta\theta(k)\left(k_p + \frac{k_i T}{z-1}\right)$$

$$= \omega_1(k) + \omega_2(k)$$

$$\omega_1(k) = \Delta\theta(k)k_p$$

$$\omega_2(k) = \Delta\theta(k)\frac{k_i T}{z-1}$$

$$\omega_2(k) = \omega_2(k-1) + \Delta\theta(k)k_i T$$

$$\theta(k+1) = \theta(k) + \omega(k)T$$

$$= \theta(k) + \frac{\omega(k)}{f_{pwm}}$$

观察到式（2-25）和式（2-26）估计得到的磁链，幅值和相位都存在偏差，那么可以假设磁链的表达式符合以下表达式。

$$\psi_\alpha = \psi_1\, cos\,(\theta_r + \phi_\alpha) \tag{2-29}$$

$$\psi_\beta = \psi_2\, sin\,(\theta_r + \phi_\beta) \tag{2-30}$$

$$\tilde{\theta}=\begin{cases}\frac{\psi_1\phi_\alpha+\psi_2\phi_\beta}{2}+\frac{\psi_2-\psi_1}{2}sin(2\theta)+\left(\frac{\psi_2\phi_\beta}{2}-\frac{\psi_1\phi_\alpha}{2}\right)cos(2\theta),\ \psi_1\le\psi_2\\ \frac{\psi_1-\psi_2}{2}(1+cos(2\theta))+\frac{\psi_2\phi_\beta}{2}(1+cos(2\theta))+\frac{\psi_1\phi_\alpha}{2}(1-cos(2\theta)),\ \psi_1>\psi_2\end{cases} \tag{2-31}$$

公式推导内在假设

$$cos\phi\approx 1, sin\phi\approx\phi \tag{2-32}$$

锁相环偏差分析

$$\Delta\theta_1=\psi_\beta cos\theta-\psi_\alpha sin\theta=\psi_2\,sin\left(\theta_r+\phi_\beta\right)cos\theta-\psi_1\,cos\left(\theta_r+\phi_\alpha\right)sin\theta$$

$$=\psi_2\left(sin\,\theta_r\,cos\phi_\beta+cos\,\theta_r\,sin\phi_\beta\right)cos\theta-\psi_1(cos\,\theta_r\,cos\phi_\alpha-sin\,\theta_r\,sin\phi_\alpha)sin\theta$$

$$=\psi_2\left(sin\,\theta_r\,cos\theta+\phi_\beta\,cos\,\theta_r\,cos\theta\right)-\psi_1(cos\,\theta_r\,sin\theta-\phi_\alpha\,sin\,\theta_r\,sin\theta)$$

稳态下，假设$\theta_r=\theta$。

当 $\psi_1\le\psi_2$

$$\Delta\theta_1=\psi_1(sin\,\theta_r\,cos\theta-cos\,\theta_r\,sin\theta)+(\psi_2-\psi_1)\,sin\,\theta_r\,cos\theta+\psi_2\phi_\beta\,cos\,\theta_r\,cos\theta$$
$$+\psi_1\phi_\alpha\,sin\,\theta_r\,sin\theta$$

$$\tilde{\theta}=\Delta\theta_1-\Delta\theta=(\psi_2-\psi_1)\,sin\,\theta_r\,cos\theta+\psi_2\phi_\beta\,cos\,\theta_r\,cos\theta+\psi_1\phi_\alpha\,sin\,\theta_r\,sin\theta$$

$$=\frac{\psi_2-\psi_1}{2}sin(2\theta)+\frac{\psi_2\phi_\beta}{2}(1+cos(2\theta))+\frac{\psi_1\phi_\alpha}{2}(1-cos(2\theta))$$

$$=\frac{\psi_1\phi_\alpha+\psi_2\phi_\beta}{2}+\frac{\psi_2-\psi_1}{2}sin(2\theta)+\left(\frac{\psi_2\phi_\beta}{2}-\frac{\psi_1\phi_\alpha}{2}\right)cos(2\theta)$$

当 $\psi_1>\psi_2$

$$\Delta\theta_1=\psi_2(sin\,\theta_r\,cos\theta-cos\,\theta_r\,sin\theta)+(\psi_1-\psi_2)\,cos\,\theta_r\,sin\theta+\psi_2\phi_\beta\,cos\,\theta_r\,cos\theta$$
$$+\psi_1\phi_\alpha\,sin\,\theta_r\,sin\theta$$

$$\tilde{\theta}=\Delta\theta_1-\Delta\theta=(\psi_1-\psi_2)\,cos\,\theta_r\,sin\theta+\psi_2\phi_\beta\,cos\,\theta_r\,cos\theta+\psi_1\phi_\alpha\,sin\,\theta_r\,sin\theta$$

$$=\frac{\psi_1-\psi_2}{2}(1+cos(2\theta))+\frac{\psi_2\phi_\beta}{2}(1+cos(2\theta))+\frac{\psi_1\phi_\alpha}{2}(1-cos(2\theta))$$

$$=\frac{\psi_1\phi_\alpha+\psi_2\phi_\beta+\psi_2-\psi_1}{2}+\left(\frac{\psi_1-\psi_2}{2}+\frac{\psi_2\phi_\beta}{2}-\frac{\psi_1\phi_\alpha}{2}\right)cos(2\theta)$$

信号本身有误差，采样也存在误差，那么滤波器就显得很有必要了。接下来设计一个低通滤波器。

采用一阶滤波器进行设计，其表达式为

$$G(s)=\frac{1}{\tau s+1} \tag{2-33}$$

采用前向差分 $s=\frac{z-1}{T}$，得到

$$y(k+1)=\frac{\omega_c}{f}u(k)+\left(1-\frac{\omega_c}{f}\right)y(k) \tag{2-34}$$

低通滤波器

$$G(s)=\frac{1}{\tau s+1}$$

前向差分 $s=\frac{z-1}{T}$

$$\frac{y(k)}{u(k)}=\frac{1}{\tau s+1}=\frac{1}{\tau\frac{z-1}{T}+1}$$

$$u(k)=y(k)\left(\tau\frac{z-1}{T}+1\right)$$

$$\frac{\tau}{T}y(k+1)-\left(\frac{\tau}{T}-1\right)y(k)=u(k)$$

$$y(k+1)=\frac{\omega_c}{f}u(k)+\left(1-\frac{\omega_c}{f}\right)y(k)$$

$$=\alpha u(k)+(1-\alpha)y(k)$$

归一化项$\alpha=\frac{\omega_c}{f}\in[0,1]$。

速度环

$$\tilde{\omega}(k)=\omega^*(k)-\omega(k)$$

$$i_q(k)=\tilde{\omega}(k)\left(k_p+\frac{k_iT}{z-1}\right)$$

$$=i_{q1}(k)+i_{q2}(k)$$

$$i_{q1}(k)=\tilde{\omega}(k)k_p$$

$$i_{q2}(k)=\tilde{\omega}(k)\frac{k_iT}{z-1}$$

$$i_{q2}(k)=i_{q2}(k-1)+\tilde{\omega}(k)k_iT$$

电流环

$$\tilde{i}_q(k)=i_q^*(k)-i_q(k)$$

$$U_q(k)=\tilde{i}_q(k)\left(k_p+\frac{k_iT}{z-1}\right)$$

$$=U_{q1}(k)+U_{q2}(k)$$

$$U_{q1}(k)=\tilde{i}_q(k)k_p$$

$$U_{q2}(k)=\tilde{i}_q(k)\frac{k_iT}{z-1}$$

$$U_{q2}(k)=U_{q2}(k-1)+\tilde{i}_q(k)k_iT$$

$$\tilde{i}_d(k) = i_d^*(k) - i_d(k)$$

$$U_d(k) = \tilde{i}_d(k)\left(k_p + \frac{k_i T}{z-1}\right)$$

$$= U_{d1}(k) + U_{d2}(k)$$

$$U_{d1}(k) = \tilde{i}_d(k) k_p$$

$$U_{d2}(k) = \tilde{i}_d(k)\frac{k_i T}{z-1}$$

$$U_{d2}(k) = U_{d2}(k-1) + \tilde{i}_d(k) k_i T$$

将得到的 U_d 和 U_q 变换到变换到α和β轴上的电压。

$$\begin{bmatrix} U_\alpha \\ U_\beta \end{bmatrix} = \begin{bmatrix} cos\,(\omega t) & -sin\,(\omega t) \\ sin\,(\omega t) & cos\,(\omega t) \end{bmatrix}\begin{bmatrix} U_d \\ U_q \end{bmatrix}$$

理论情况下，$\begin{bmatrix} U_d \\ U_q \end{bmatrix}$应该是直流分量，但是在实际计算中会有谐波叠加在$\begin{bmatrix} U_d \\ U_q \end{bmatrix}$中。假设$\begin{bmatrix} U_d \\ U_q \end{bmatrix}$符合以下表达式

$$\begin{bmatrix} U_d \\ U_q \end{bmatrix} = \begin{bmatrix} M + A_d\,sin\,(\omega t + \varphi_d) \\ N + A_q\,sin\,(\omega t + \varphi_q) \end{bmatrix}$$

$$= \begin{bmatrix} M \\ N \end{bmatrix} + \begin{bmatrix} A_d cos\varphi_d & A_d sin\varphi_d \\ A_q cos\varphi_q & A_q cos\varphi_q \end{bmatrix}\begin{bmatrix} sin\,(\omega t) \\ cos\,(\omega t) \end{bmatrix}$$

$$\begin{bmatrix} U_\alpha \\ U_\beta \end{bmatrix} = \begin{bmatrix} cos\,(\omega t) & -sin\,(\omega t) \\ sin\,(\omega t) & cos\,(\omega t) \end{bmatrix}\left(\begin{bmatrix} X \\ Y \end{bmatrix} + \begin{bmatrix} A_d cos\varphi_d & A_d sin\varphi_d \\ A_q cos\varphi_q & A_q sin\varphi_q \end{bmatrix}\begin{bmatrix} sin\,(\omega t) \\ cos\,(\omega t) \end{bmatrix}\right)$$

$$= \begin{bmatrix} X\,cos\,(\omega t) - Y\,sin\,(\omega t) \\ X\,sin\,(\omega t) + Y\,cos\,(\omega t) \end{bmatrix}$$

$$+ \begin{bmatrix} \frac{A_d cos\varphi_d - A_q sin\varphi_q}{2} sin(2\omega t) + \frac{A_d sin\varphi_d + A_q cos\varphi_q}{2} cos(2\omega t) + \frac{A_d sin\varphi_d - A_q cos\varphi_q}{2} \\ \frac{A_d sin\varphi_d + A_q cos\varphi_q}{2} sin(2\omega t) + \frac{A_q sin\varphi_q - A_d cos\varphi_d}{2} cos(2\omega t) + \frac{A_d cos\varphi_d + A_q sin\varphi_q}{2} \end{bmatrix}$$

$$\begin{bmatrix} U_\alpha \\ U_\beta \end{bmatrix} = \begin{bmatrix} Z_1\,sin\left(\omega t + \varphi_{dq1}\right) + A_{dq1} sin\left(2\omega t + \varphi_{dq2}\right) + C_1 \\ Z_2\,sin\left(\omega t + \varphi_{dq3}\right) + A_{dq2} sin\left(2\omega t + \varphi_{dq4}\right) + C_2 \end{bmatrix}$$

$$\leq \begin{bmatrix} Z_1 + A_{dq1} + C_1 \\ Z_2 + A_{dq2} + C_2 \end{bmatrix}$$

将由反 Park 变换得到的α和β轴上的电压通过反 Clark 变换转换到 U、V、W 三相电压。

$$\begin{bmatrix} U_\alpha \\ U_\beta \end{bmatrix} = \begin{bmatrix} Z_1 \sin\left(\omega t + \varphi_{dq1}\right) + A_{dq1} sin\left(2\omega t + \varphi_{dq2}\right) + C_1 \\ Z_2 \sin\left(\omega t + \varphi_{dq3}\right) + A_{dq2} sin\left(2\omega t + \varphi_{dq4}\right) + C_2 \end{bmatrix}$$

$$\begin{bmatrix} U_u \\ U_v \\ U_w \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ \frac{\sqrt{3}}{2} & -\frac{1}{2} \\ -\frac{\sqrt{3}}{2} & -\frac{1}{2} \end{bmatrix} \begin{bmatrix} U_\alpha \\ U_\beta \end{bmatrix}$$

$$= \begin{bmatrix} U_\alpha \\ \frac{\sqrt{3}U_\alpha - U_\beta}{2} \\ -\frac{\sqrt{3}U_\alpha + U_\beta}{2} \end{bmatrix}$$

$$\begin{bmatrix} U_u \\ U_v \\ U_w \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ \frac{\sqrt{3}}{2} & -\frac{1}{2} \\ -\frac{\sqrt{3}}{2} & -\frac{1}{2} \end{bmatrix} \begin{bmatrix} Z_1 \sin\left(\omega t + \varphi_{dq1}\right) + A_{dq1} sin\left(2\omega t + \varphi_{dq2}\right) + C_1 \\ Z_2 \sin\left(\omega t + \varphi_{dq3}\right) + A_{dq2} sin\left(2\omega t + \varphi_{dq4}\right) + C_2 \end{bmatrix}$$

矢量控制计算

反 Clark 变换计算得到的三相电压

计算三相电压输出的时间

$$T_a = \frac{U_u}{Vdc} T_{pwm}$$

$$T_b = \frac{U_v}{Vdc} T_{pwm}$$

$$T_c = \frac{U_w}{Vdc} T_{pwm}$$

加入死区补偿

$$T_a = T_a + T_{ud}$$

$$T_b = T_b + T_{vd}$$

$$T_c = T_c + T_{wd}$$

扇区判断

$$T_a \geq T_b \geq T_c, Sector\ 1$$

$$T_a \geq T_c \geq T_b, Sector\ 2$$

$$T_b \geq T_c \geq T_a, Sector\ 3$$

$$T_b \geq T_a \geq T_c, Sector\ 4$$

$$T_c \geq T_a \geq T_b, Sector\ 5$$

$$T_c \geq T_b \geq T_a, Sector\ 6$$

计算中间时间

$$Sector\ 1, T_{mid} = T_{\frac{pwm}{2}} - \frac{T_a + T_c}{2}$$

$$Sector\ 2, T_{mid} = T_{\frac{pwm}{2}} - \frac{T_a + T_b}{2}$$

$$Sector\ 3, T_{mid} = T_{\frac{pwm}{2}} - \frac{T_a + T_b}{2}$$

$$Sector\ 4, T_{mid} = T_{\frac{pwm}{2}} - \frac{T_b + T_c}{2}$$

$$Sector\ 5, T_{mid} = T_{\frac{pwm}{2}} - \frac{T_b + T_c}{2}$$

$$Sector\ 6, T_{mid} = T_{\frac{pwm}{2}} - \frac{T_a + T_c}{2}$$

注入三次谐波

$$T_a = T_a + T_{mid}$$

$$T_b = T_b + T_{mid}$$

$$T_c = T_c + T_{mid}$$

计算 T1 和 T2

$$Sector\ 1, T_{mid} = T_{\frac{pwm}{2}} - \frac{T_a + T_c}{2}$$

$$Sector\ 2, T_{mid} = T_{\frac{pwm}{2}} - \frac{T_a + T_b}{2}$$

$$Sector\ 3, T_{mid} = T_{\frac{pwm}{2}} - \frac{T_a + T_b}{2}$$

$$Sector\ 4, T_{mid} = T_{\frac{pwm}{2}} - \frac{T_b + T_c}{2}$$

$$Sector\ 5, T_{mid} = T_{\frac{pwm}{2}} - \frac{T_b + T_c}{2}$$

$$Sector\ 6, T_{mid} = T_{\frac{pwm}{2}} - \frac{T_a + T_c}{2}$$

Chapter 3

Implementation

The architecture of the control system is designed according to the ***V-model*** and ***Layered Software Design***. The Simple V-model and the layered structure are shown in the figure 1and figure 2 respectively.

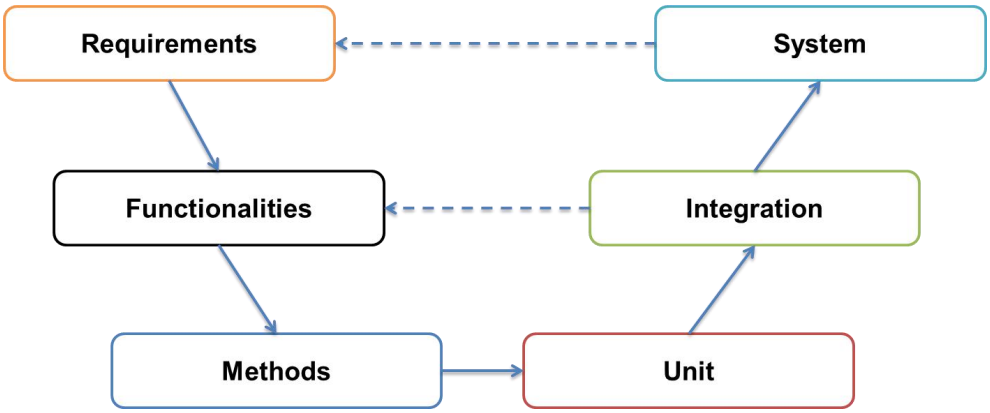


Figure 1 V-Model for Software Design

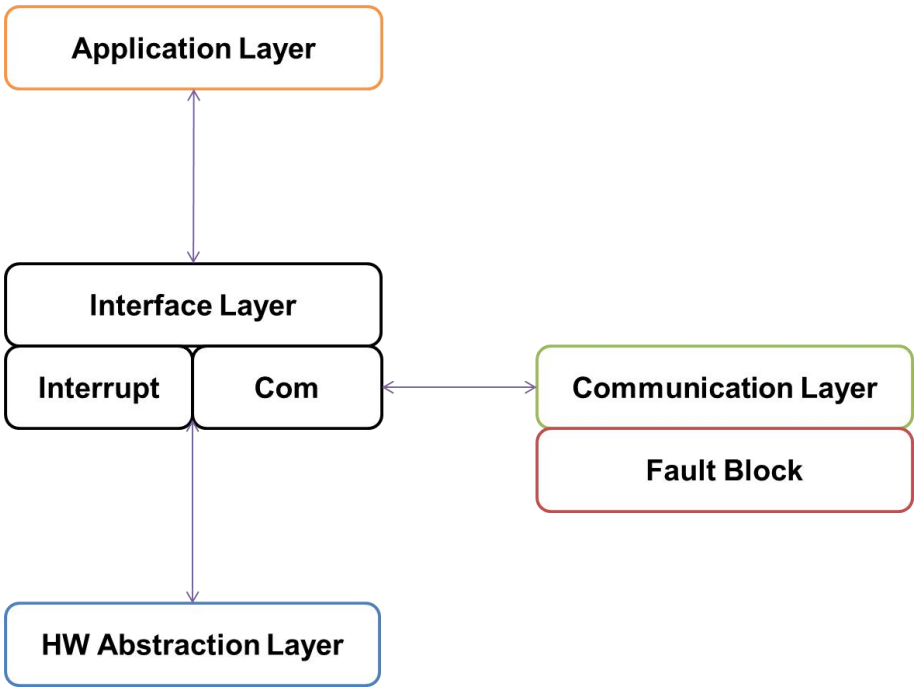


Figure 2 Layered Framework of the Control System

Layered Design

According to the pipeline, the layered software involves three parts: Hardware Abstract Layer (***HAL***), Interface Layer (***IL***), and Application Layer (***AL***).

In each layer, there may be ***workflow, dataflow,*** or ***stateflow*** (where necessary). Along the flow we can make the code clean and logic, sometimes we can refactor the whole control system by analysis.

There are some configurations of the Hardware in the ***HAL***: Clock, Pin, Interrupt Disable/Enable, Power, Trigmux, Hardware Initialization, PDB, ADC, DMA, UART, FTM, PWM, and Communication.

The blocks needed to be configured are shown below.

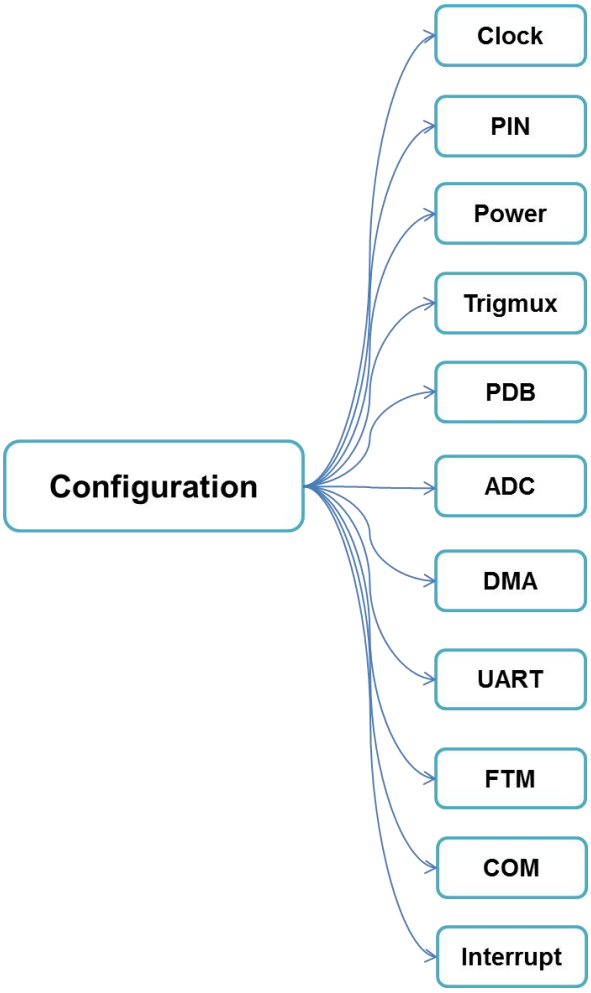


Figure 3 The Blocks needed to configured in the HAL

<HAL>

<CLOCK>

Name: McuclockConfig()

Description: This function initializes the system clock and other clocks.

<PIN>

Name: McuPinsConfig()

Description: This function initializes the PINs used.

<POWER>

Name: McuPowerConfig()

Description: This function initialize the POWER mode (HRUN, RUN, …)

<TRIGMUX>

Name: McuTrigmuxConfig()

Description: This function configures multi-triggers for FTM, PDB, etc.

<PDB>

Name: McuPdbConfig()

Description: This function configures PDB and pre-triggers for ADC.

<ADC>

Name: McuAdcConfig()

Description: This function configures ADC channels for sampling currents, voltage, Temperature, etc.

<DMA>

Name: McuDmaConfig()

Description: This function configures DMA for CAN and UART.

<UART>

Name: McuUartConfig()

Description: This function configures UART.

<FTM>

Name: McuFtmConfig()

Description: This function configures FTM/PWM and Interrupt mode.

Note: In order to integrate the code and for the convenience,

Timeflow Interrupt and Reload Interrupt shall be modified and clarified.

<PWM>

Name: McuFtmConfig()

Description: This function configures PWM Outputs and operating mode

<CAN>

Name: McuCANConfig()

Description: This function configures CAN.

<Interrupt>

Name: intenable()

Description: This function configures FTM and ADC Interrupts, respectively.

<IL>

We are going to package all the configurations together to reduce the dependencies between layers. Hence, the interface functions between layers are **<Motor_Para_Init>**, **<intenable>**, and **<Communication_Service>**.

Therefore, the logic here is very straightforward. Along this pipeline, we minimize the dependencies between layers. Only the interface functions interact with each layer.

There are some interfaces to the AL in the IL. Here the interfaces are mostly structure variables which are links to AL. Phase currents and target frequency, for example.

Name: Init_MtrParaIni()

Description: This function assigns the values to the structured data.

```
void MtrPara_Init(void)
{
    int32_t *p;
    uint16_t lu16_i;
    p = (int32_t*)&ParaRam;
    for(lu16_i = 0u; lu16_i < TABLE_NUMBER; lu16_i++)
    {
        *p = (int32_t)TABLE_MTR_PARA[lu16_i];
         p++;
    }
}
```

Name: Ms_1_job()

Description: This is a call back function with a time base counting in the PWM Interrupt. This function involves the following functions.

Name: Inner_time1ms_mission()

Description: This is for LIN and to check if LIN is connected.

Name: Motor_Run_Time()

Description: This is a time counter that enables that each function is called at right time.

Name: CALCULATEP_I()

Description: This is to calculate the parameters like current, voltage, power, speed.

Name: Faultclear()

Description: This is to clear all the faults generated by the motor protection measures. Stop for 5s without doing anything then set all the

fault-related bits to zeros.

Name: Start_after_2s_job()

Description: This is to increase or decrease to the target speed when switched to closed loop after 2s and check if SVPWM is in phase Shift mode.

Name: Ms_100_job()

Description: This is a call back function with a time base counting in the PWM Interrupt. The function involves the following functions.

Name: Freqset_with_condition()

Description: This is to set the frequency for the motor under the condition that no error is occurred.

Name: Can_send_id_change_count()

Description: This is to interchange the CAN_ID for Senddata() every 0.1s.

Name: motorParaChange()

Description: This is to change the LD and LQ online such that the motor can run smoothly at high speed.

The following functions will be communication-related. Here we focus on the LIN communication which is double_ communication() in the code.

Name: Communication_service()

Double_communicaiton()

Description: This is to receive the fault-clearing bit, the power-on bit, and

the Target speed and transmit the state of the motor and parameters required via UART.

Name: Communication_Drive()

InCom_Uart_Drive()

Description: This is to realize the first RX and alternatively RX and TX.

There are two interrupt functions. These functions belong to AL.

Name: Intenable()

Description: This is to enable the interrupt functions required and call the ADC sampling function and output the updating PWM.

Name: ftmTimerISR()

Description: This is to calculate the duty cycles for three phases.

Int_PWM_after_shakehand()

Name: TrqComp()

Description: This is to calculate the amount for torque compensation.

Name: Fault_Warning_Detect()

Description: This is to detect whether fault occurs and stop the motor if necessary.

Name: TimeBaseCount()

Description: This is time counter.

Name: Int_AD_GetPhaseCurrent()

Description: This is to get offset of sample currents.

Name: Motor_state_machine()

Description: This is a state machine for motor control. In each case, do the corresponding action.

Name: adc1isr()

Description: This is to get the values from the ADC channels. Hence, the phase currents, DC bus voltage, IPM temperature, and PCB Temperature. All of these values are scaling values and the range is 0~4096. adcRawValue_n to indicate the corresponding parameters.

<AL>

The first thing is to initialize the corresponding parameters. Here we adopt a strategy where values in the constant table are assigned to the structured data, and then assign the structured data to global variables.

Name: Init_Drive_HardwareandSoftware()

Description: This function assigns the values to the global variables for intended use.

Name: Init_MotorPara()

Description: This function assigns the values to the motor parameters (Resistance, Inductance, Poles, and Flux Linkage).

The idea here is to assign the values in the table to structure-ParaRam, then the assign the values to global variables where needed. Initialize the motor parameters independently.

Name: Motor_reset()

Description: This is to initialize or reset the structured data which are interface variables.

Name: ALLlowpass_init()

Description: This is to reset all the variables going through Lowpass Filter.

Name: Motor_U_I_Angle_clear()

Description: This is to reset current, voltage, and theta.

Name: Init_SVPWM()

Description: This is to initialize the SVPWM variables.

Name: Init_FLUXOBSERVER()

Description: This is to initialize the flux on the alpha and beta axes.

Name: Init_MtrAcr()

Description: This is to set PI for Current Loop, and Clear the flag for integral saturation.

Name: Init_MtrAsr()

Description: This is to set PI for Speed Loop and initialize lowpass filter for Omega.

Name: Init_MtrFwc()

Description: This is to set FWK_LVL and FWK_MAX and PI for the Field weakening control loop.

Name: Init_MtrPll()

Description: This is to set PI for Phase-locked Loop and some lowpass filters for motor speed and set the actual frequency.

Name: OpenandParkSet()

Description: This is to initialize the structured data for both openloop and parkloop.

Name: CarrierDeadCalc()

Description: This is to calculate the Carrier ticks and dead time for u, v, w phase and PWM frequency, and Time Base for other functions call.

Name: Init_DeadtimeComp()

Description: This is to initialize the Dead Time compensation for u,v,w phase.

Name: Intenable()

Description: This is to initialize interrupts and set the priorities for the corresponding interrupts.

Name: Communication_Init()

Description: This is to initialize CAN or UART.

Name: INT_SYS_EnableIRQGlobal()

Description: This is to enable the Global Interrupt.

In order to conveniently switch two PWM modes, here we are going to use a

global variable (enumeration type) to indicate which mode is selected.

For example: gu8_PWM_Freq_Sel = 1, 2, 3, (1 = 8k, 2 = 16k), gu8_HS_nPWM = 1, 2,

```
typedef enum
{
    FTM3_Ovf_IRQn       = 0x00U,     /*!< Overflow interrupt */
    FTM3_Reload_IRQn    = 0x01U      /*!< Reload interrupt */
} ftm_INT_option_t;


ftm_INT_option_t FTM3_INT_Sel;      /*!< Select the Interrupt type */
```

AL is the core part which is employed to drive and control the motor such that control the electric compressor. The whole control system is based on the Field-oriented control algorithm. The idea is statistical estimation and inference where speed and position of the motor are estimated.

In the AL, the key component will be state machine which is the core control algorithms. The control algorithm can be seen as a state machine: PRECHARGE, RUN, STOP, ERROR. There are four states.

We implement the control algorithm using C. Here we mainly focus on the methods used in FOC. As we can see, it is very common that we use Macro, Structure, Union, Enumeration in C. Keep in mind that there is a definition to data type.

```
typedef signed char  int8_t;
typedef unsigned char  uint8_t;
typedef signed short int16_t;
typedef unsigned short uint16_t;
typedef signed long  int32_t;
typedef unsigned long  uint32_t;
```

<FOC>

<ClarkTransform>

```
void ClarkTransform(uvw_ab_dq_t* p_i)
{
    p_i->s16_alpha = p_i->s16_u;
    p_i->s16_beta  = (int16_t)(((((int32_t)p_i->s16_u)*9459) +
(((int32_t)p_i->s16_v)*2*9459))/16384);
}
```

<ParkTransform>

```
void ParkTransform(uvw_ab_dq_t* p_i, theta_t* p_theta_e)
{
    p_i->s16_d  =  (int16_t)(((((int32_t)p_i->s16_alpha)  *  p_theta_e->s16_cos)  +
((p_i->s16_beta) * p_theta_e->s16_sin))/32768);
    p_i->s16_q  =  (int16_t)(((((int32_t)p_i->s16_beta)  *  p_theta_e->s16_cos)  -
((p_i->s16_alpha) * p_theta_e->s16_sin))/32768);
}
```

<FluxObserver>

```
void FluxObserver(flux_t* p_flux, uvw_ab_dq_t* p_i, uvw_ab_dq_t* p_u, motor_para_t*
p_motor_para, int16_t p_pwmHz)
{
    p_flux->s32_tempFaia                                                    =
p_flux->s32_tempFaia+(((((int32_t)p_u->s16_alpha)*630)-((p_motor_para->s16_Rs*(p_i-
>s16_alpha))/64)- (p_flux->s32_gaink * p_flux->s32_alpha_n_1))/p_pwmHz);
    p_flux->s16_alphaFlux_Q15                                               =
(int16_t)(p_flux->s32_tempFaia-((((int32_t)(p_motor_para->s16_Ld+p_motor_para->s16
_Lq))*(p_i->s16_alpha))/8192));
    p_flux->s32_alpha_n_1       = (int32_t)p_flux->s16_alphaFlux_Q15;

    p_flux->s32_tempFaib                                                    =
p_flux->s32_tempFaib+(((((int32_t)p_u->s16_beta)*630)-(((p_motor_para->s16_Rs*p_i->
s16_beta))/64)-(p_flux->s32_gaink * p_flux->s32_beta_n_1))/p_pwmHz);
    p_flux->s16_betaFlux_Q15                                                =
(int16_t)(p_flux->s32_tempFaib-((((int32_t)(p_motor_para->s16_Ld+p_motor_para->s16
_Lq))*p_i->s16_beta)/8192));
```

```
    p_flux->s32_beta_n_1          = (int32_t)p_flux->s16_betaFlux_Q15;
}
```

<PhaseLockLoop>

```
void PhaseLockedLoop(flux_t* p_flux, theta_t* p_theta_pll, pi_t* p_pi_pll, lowpass_filter_t* p_lp_speed, int16_t p_pwmHz, int16_t* p_actualWe, int16_t* p_actualWe_f)
{
    int32_t   ls32_i1;
    int32_t   ls32_i2;
    int32_t   ls32_i3;
    int16_t   ls16_pllSpeedLoopError;

    ls32_i1 = ((((int32_t)p_flux->s16_betaFlux_Q15)*p_theta_pll->s16_cos)/32767);
    ls32_i2 = ((((int32_t)p_flux->s16_alphaFlux_Q15)*p_theta_pll->s16_sin)/32767);
    ls16_pllSpeedLoopError = ssub16((int16_t)ls32_i1,(int16_t)ls32_i2);

    if(ls16_pllSpeedLoopError>5000)
    {
        ls16_pllSpeedLoopError=5000;
    }
    if(ls16_pllSpeedLoopError<-5000)
    {
        ls16_pllSpeedLoopError=-5000;
    }

    *p_actualWe = PI_Controller(ls16_pllSpeedLoopError, p_pi_pll);

    ls32_i3 = (((int32_t)600)*(*p_actualWe*2));
    ls32_i3 = ls32_i3/p_pwmHz;

    p_theta_pll->u16_theta    = (uint16_t)ls32_i3+p_theta_pll->u16_theta;
    p_theta_pll->u16_theta &= 0xffffu;
    p_theta_pll->s16_sin = (csin_calculation(p_theta_pll->u16_theta));
```

```
    p_theta_pll->s16_cos                                                    =
(csin_calculation((uint16_t)(p_theta_pll->u16_theta+0x4000u)));

    LowpassFilter(p_lp_speed, *p_actualWe, 200);
    *p_actualWe_f = p_lp_speed->s16_output;
}
```

<SpeedLoop>

```
void SpeedLoop(pi_t *p_pi_speed, int16_t p_referenceWe, int16_t p_actualWe_f, int16_t
p_actualWe, reference_current_t *p_ref_i)
{
    int16_t ls16_PI_SpeedLoopError1,ls16_PI_SpeedLoopError2;
    ls16_PI_SpeedLoopError1 = p_referenceWe - p_actualWe_f;
    p_ref_i->s16_qref_cur = PI_Controller(ls16_PI_SpeedLoopError1, p_pi_speed);
}
```

<FieldWeakeningControl>

```
void FireldWeakeningControl(pi_t *p_pi_fieldweakening, reference_current_t *p_ref_i,
svpwm_t *p_svpwm, lowpass_filter_t *p_lp_Tall)
{
    int16_t ls16_fieldWeakingError1;
    static uint32_t counter = 0;

    counter++;
    if (counter>=1000u)
    {
        LowpassFilter(p_lp_Tall, (int16_t)p_svpwm->s16_tall, 15);
        ls16_fieldWeakingError1          =          (int16_t)(gs16_FWK_LVL          -
(int32_t)p_lp_Tall->s16_output);
        p_ref_i->s16_dref_cur                    =       p_ref_i->s16_dref_cur       +
PI_Controller(ls16_fieldWeakingError1, p_pi_fieldweakening);
    }
    else
    {
```

```
            if(p_ref_i->s16_dref_cur != 0)
            {
                  p_ref_i->s16_dref_cur = 0;
            }
            p_lp_Tall->s16_output = 0;
      }
}
```

<CurrentLoop>

```
void CurrentLoop(reference_current_t* p_ref_i, uvw_ab_dq_t* p_u, uvw_ab_dq_t* p_i, pi_t* p_pi_d, pi_t* p_pi_q)
{
      int16_t ls16_open_Daxis_Error1;
      int16_t ls16_open_Qaxis_Error1;
      ls16_open_Daxis_Error1 = (int16_t)(p_ref_i->s16_dref_cur - p_i->s16_d);
      ls16_open_Qaxis_Error1 = (int16_t)(p_ref_i->s16_qref_cur - p_i->s16_q);
      p_u->s16_d = PI_Controller(ls16_open_Daxis_Error1, p_pi_d);
      p_u->s16_q = PI_Controller(ls16_open_Qaxis_Error1, p_pi_q);
}
```

<InverseParkTransform>

```
void Inverse_ParkTransform(uvw_ab_dq_t* p_u, theta_t* p_theta_e)
{
      p_u->s16_alpha = (int16_t)(((((int32_t)p_u->s16_d) * p_theta_e->s16_cos) -((p_u->s16_q) * p_theta_e->s16_sin))/32768);
      p_u->s16_beta = (int16_t)(((((int32_t)p_u->s16_d) * p_theta_e->s16_sin) +((p_u->s16_q) * p_theta_e->s16_cos))/32768);
}
```

<InverseClarkTransform>

```
void Inverse_ClarkTransform(uvw_ab_dq_t *p_u)
{
      p_u->s16_u = p_u->s16_alpha;
      p_u->s16_v                                                                    =
```

```
(int16_t)((((int32_t)p_u->s16_beta)*9459/2)/16384)-(p_u->s16_alpha/2);
    p_u->s16_w                                                              =
(int16_t)((-(((int32_t)p_u->s16_beta)*9459/2))/16384)-(p_u->s16_alpha/2);
}
```

Chapter 4

Conclusion