Analysis of a Hyundai Sonata IPM motor

Aaron Yeiser

June 28, 2020

Abstract

Interior permanent magnet motors are commonly used as traction motors for automobiles because they are power–dense and very efficient. IPMs have permanent magnets buried in a ferromagnetic rotor, which allows for very effective field weakening. Field weakening allows the motor to maintain maximum output power over a wide range of motor speeds, which is desirable in traction motors because it means no transmission is required for the motor to perform optimally. The Hyundai Sonata alternator motor is an interior permanent magnet motor designed for charging the hybrid car battery through regenerative braking. In order to design a motor controller for this motor, I wanted to accurately simulate the motor dynamics. The motor model that I developed did not take into account saturation because of the large currents required to start saturating the magnetic materials int his motor.

1 Motor modeling

Interior permanent magnets function like a switched reluctance motor with permanent magnets embedded in the rotor. There are two mechanisms for torque production: permanent magnet torque and reluctance torque. The permanent magnets in the motor have high coercivity, so the field produced by the magnets is roughly the same regardless of the current through the motor. However, since $\mu_{magnet} \ll \mu_{steel}$, the holes left by the magnets in the rotor act similarly to air gaps, causing the reluctance to vary with rotor position.

Labelling the phases u, v, w, we can model the currents through the motor as $\mathbf{I}_{uvw}$, and the flux linkage as $\mathbf{\Lambda}_{uvw}$. Assuming that all magnetic materials have constant μ (over the range of currents we are using), we can model

$$\mathbf{\Lambda}_{uvw} = \mathbf{L}_{uvw}(\theta)\mathbf{I}_{uvw} + \mathbf{\Psi}_{uvw}(\theta).$$

Note that $\mathbf{L}_{uvw} = \begin{pmatrix} L_{uu} & L_{uv} & L_{uw} \\ L_{vu} & L_{vv} & L_{vw} \\ L_{wu} & L_{wv} & L_{ww} \end{pmatrix}$, and $\mathbf{\Lambda}_{uvw}$, $\mathbf{I}_{uvw}$, and $\mathbf{\Psi}_{uvw}$ are vectors.

Using three phases to model the motor behavior has some serious drawbacks. Since the net current through the motor is zero, this three phase model has an extra degree of freedom since

$I_u + I_v + I_w = 0$. The solution to this problem is the Clarke transform, also known as the $\alpha\beta\gamma$ transform. The Clarke transform maps three phase currents or voltages to two orthogonal components with a 90°, rather than three components with a 120° phase difference.

The Clarke transform matrix and its inverse can be written as

$$\mathbf{T} = \frac{2}{3}\begin{pmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} \end{pmatrix}, \qquad \mathbf{T}^{-1} = \begin{pmatrix} 1 & 0 \\ -\frac{1}{2} & \frac{\sqrt{3}}{2} \\ -\frac{1}{2} & -\frac{\sqrt{3}}{2} \end{pmatrix}.$$

$\mathbf{I}_{\alpha\beta} = \mathbf{T}\mathbf{I}_{uvw}$, and similarly for $\mathbf{\Lambda}$ and $\mathbf{\Psi}$. For inductance, we can write $\mathbf{L}_{\alpha\beta} = \mathbf{T}\mathbf{L}_{uvw}\mathbf{T}^{-1}$ and similarly, $\mathbf{L}_{uvw} = \mathbf{T}^{-1}\mathbf{L}_{\alpha\beta}\mathbf{T}$.

Furthermore, we can apply the Park transform (also known as the DQ0 transform). When neglecting the phase–independent current (which is zero in a three phase motor), the Park transform $\mathbf{K}_P(\theta) = \mathbf{R}(-\theta)$, where $\mathbf{R} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$ is the standard rotation matrix in two dimensions.

This has the effect of giving $\mathbf{I}_{dq} = \mathbf{K}_P\mathbf{I}_{\alpha\beta}$ and $\mathbf{L}_{dq} = \mathbf{K}_P\mathbf{L}_{\alpha\beta}\mathbf{K}_P^T$.

The Park transform has the effect of translating currents and voltages to the rotor frame of view, which is objectively useful when controlling the motor. In particular I_q roughly corresponds to the torque through the motor and I_d corresponds to the amount of field weakening.

The phase voltages of the motor are

$$\mathbf{V}_{uvw} = R\mathbf{I}_{uvw} + \frac{d}{dt}\mathbf{\Lambda}_{uvw} = R\mathbf{I}_{uvw} + \mathbf{L}'_{uvw}\mathbf{I}_{uvw} + \mathbf{L}_{uvw}\mathbf{I}'_{uvw} + \mathbf{\Psi}'_{uvw},$$

or in the $\alpha\beta$ frame,

$$\mathbf{V}_{\alpha\beta} = R\mathbf{I}_{\alpha\beta} + \frac{d}{dt}\mathbf{\Lambda}_{\alpha\beta} = R\mathbf{I}_{\alpha\beta} + \mathbf{L}'_{\alpha\beta}\mathbf{I}_{\alpha\beta} + \mathbf{L}_{\alpha\beta}\mathbf{I}'_{\alpha\beta} + \mathbf{\Psi}'_{\alpha\beta}.$$

The dq frame is slightly more difficult, because the Park transform has a nonzero time derivative. Specifically, $\mathbf{K}'_P = \omega\begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}\mathbf{K}_P = \omega\mathbf{R}_{-90^\circ}\mathbf{K}_P$ and $(\mathbf{K}_P^T)' = \omega\begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}\mathbf{K}_P^T = \omega\mathbf{R}_{90^\circ}\mathbf{K}_P^T$.

Substituting and simplifying gives

$$\mathbf{V}_{dq} = R\mathbf{I}_{dq} + \omega\mathbf{R}_{90^\circ}(\mathbf{L}_{dq}\mathbf{I}_{dq} + \mathbf{\Psi}_{dq}) + \mathbf{L}_{dq}\mathbf{I}'_{dq} + \mathbf{L}'_{dq}\mathbf{I}_{dq} + \mathbf{\Psi}'_{dq}.$$

We can further simplify matters by ignoring saturation, non–sinusoidal flux linkage, and angle-dependent $\mathbf{L}_{dq}$ to give

$$\mathbf{V}_{dq} = R\mathbf{I}_{dq} + \omega\mathbf{R}_{90^\circ}(\mathbf{L}_{dq}\mathbf{I}_{dq} + \mathbf{\Psi}_{dq}) + \mathbf{L}_{dq}\mathbf{I}'_{dq}.$$

2 Measurement procedure

In order to model the motor effectively, I needed the flux through the motor as a function of rotor angle and α and β phase currents. The easiest way to do this without painfully stepping through measurements taken at different angles was to spin the motor with a drill and inject a 5 Vpp 2 kHz sinusoidal signal into the motor. Injecting this signal in the motor at two different angles and probing the outputs of the motor with the oscilloscope was sufficient to obtain both the back emf of the motor and the α and β inductances, albeit with a substantial amount of signal processing. Figure 1 shows a diagram of the experimental setup. The signal generators are set in phase to generate a sinusoid aligned with only α voltage applied and set out of phase to generate a sinusoid with only β voltage. All three output terminals of the motor are monitored with the oscilloscope.

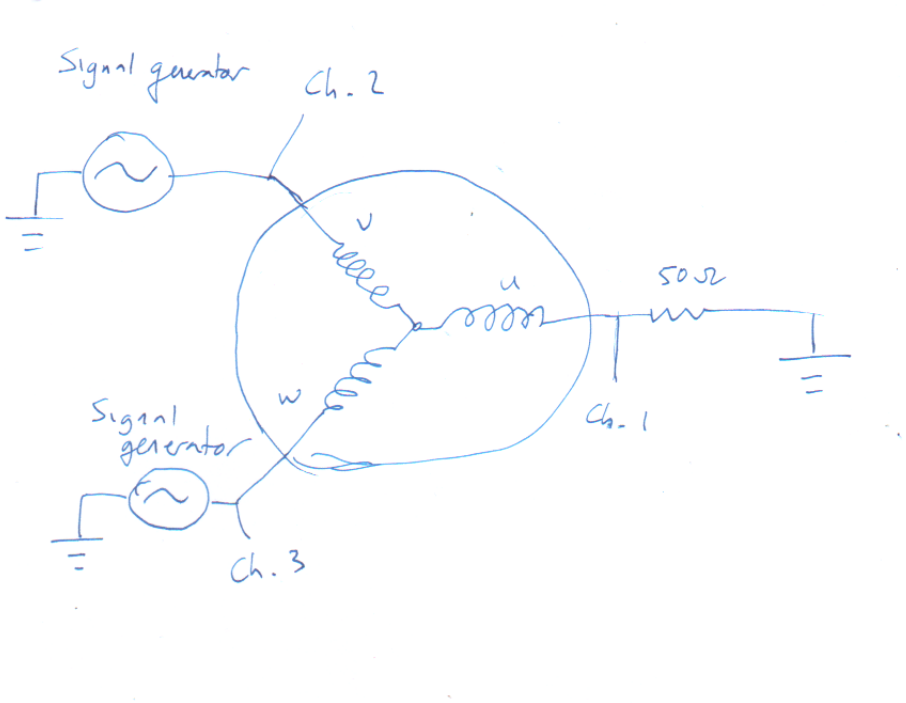

Figure 1: The output impedance of the signal generator is 50 Ω.

3 Analysis

The voltages observed at the motor terminals were fed into a Clarke transform to get α and β voltages. Low pass filtering gives the back-emf and bandpass filtering around 2 kHz isolates the injected high frequency signal used to measure inductance. The electrical frequency of the motor can be obtained by finding zero crossings in the filtered back-emf and extracting one period. The permanent magnet–induced flux linkage can be obtained by integrating the back-emf over one period.

Finding the inductance matrix as a function of rotor angle is more complicated. Here, the bandpass filtered α and β voltages are multiplied by $\cos\left(\frac{2000t}{2\pi}\right)$ and $\sin\left(\frac{2000t}{2\pi}\right)$ and then

low pass filtered, acting as a quadrature demodulator. This has the useful feature of being able to describe the α and β voltages as $V_c(t)\cos\left(\frac{2000t}{2\pi}\right) + V_s(t)\sin\left(\frac{2000t}{2\pi}\right)$. Integrating these orthogonal components to obtain flux linkage is very simple and numerically stable with this representation. In addition, we can extract α and β currents from the phase voltages and knowledge that all terminals of the motor have a $50\,\Omega$ impedance. Running the signal generators in phase, $I_\alpha = I_u = -V_u/50\,\Omega$ and $I_\beta = -V_\beta/50\,\Omega$. Running the signal generators out of phase, $I_\alpha = -V_\alpha/50\,\Omega$. I_β is a bit more tricky to calculate for running the signal generators out of phase. When I_α is zero, $L_{\alpha\beta}$ is zero because L_α is undriven, and I_β is 90° out of phase with V_β because L_β acts like a pure inductor. Since we know the amplitude of $V_{\beta_{in}}$, we can calculate the phase difference between the input voltage and the measured voltage, allowing us to calculate $I_\beta = (V_{\beta_{in}} - V_\beta)/50\,\Omega$.

We can write

$$\begin{pmatrix} \lambda_{\alpha 1} & \lambda_{\alpha 2} & \cdots \\ \lambda_{\beta 1} & \lambda_{\beta 2} & \cdots \end{pmatrix} = \begin{pmatrix} L_\alpha & L_{\alpha\beta} \\ L_{\alpha\beta} & L_\beta \end{pmatrix} \begin{pmatrix} I_{\alpha 1} & I_{\alpha 2} & \cdots \\ I_{\beta 1} & I_{\beta 2} & \cdots \end{pmatrix},$$

which gives

$$\begin{pmatrix} L_\alpha & L_{\alpha\beta} \\ L_{\alpha\beta} & L_\beta \end{pmatrix} = \begin{pmatrix} \lambda_{\alpha 1} & \lambda_{\alpha 2} & \cdots \\ \lambda_{\beta 1} & \lambda_{\beta 2} & \cdots \end{pmatrix} \begin{pmatrix} I_{\alpha 1} & I_{\alpha 2} & \cdots \\ I_{\beta 1} & I_{\beta 2} & \cdots \end{pmatrix}^+,$$

where A^+ is the Moore–Penrose pseudoinverse of A.

Figures 2, 3, and 4 display the measured motor characteristics. The Q and D inductances differ by a factor of 3 or so, which allows the motor to generate reluctance torque.

The method for measuring the motor was far from perfect, but it provided a good reference point for the properties of the motor. A possible improvement to the measurement method would be to accurately track the reference input signal. This could be achieved by using an oscilloscope function generator, rather than the independent unit I used. Another strategy could be to increase the resistance in series with the motor phases and increase the excitation frequency. This strategy has the disadvantage of inducing more eddy currents in the laminations of the motor, possibly resulting in flawed measurements.

The most reliable strategy for measuring motor characteristics is to use a motor controller to spin the motor at different speeds with different Q and D currents. This method has the advantage of being able to take good measurements with the motor in saturation, because the motor controller is capable of pushing enough current through the motor that the stator teeth and rotor start to saturate. Measuring motor characteristics in saturation is extremely useful because the motor will saturate when it is outputting large amounts of power. These measurements can be used to optimize the performance of the motor when it actually matters—when the motor is under load.

A Analysis code

This code was put together to analyze scope traces and attempt to find the motor flux linkage and Q and D inductances with respect to electrical phase angle.

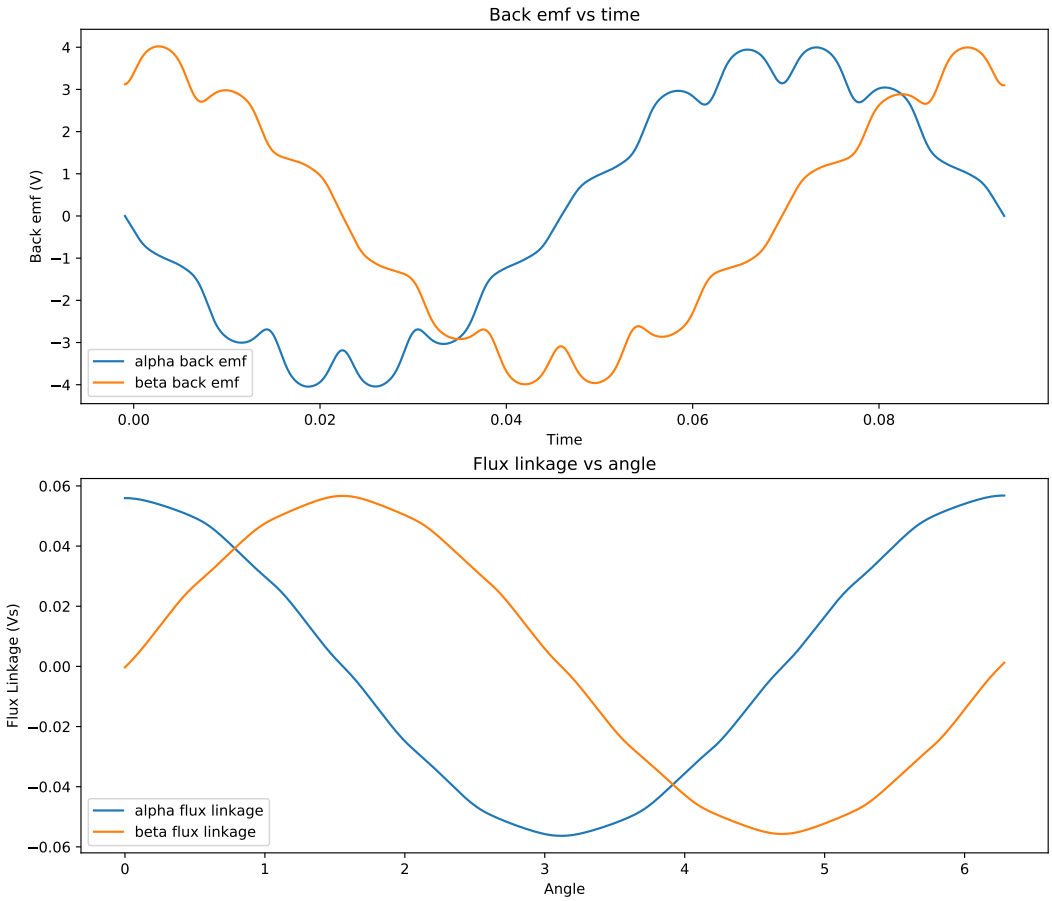

Figure 2: Measured flux and back emf. The back emf is slightly bumpy, probably due to the shape of the distributed windings.

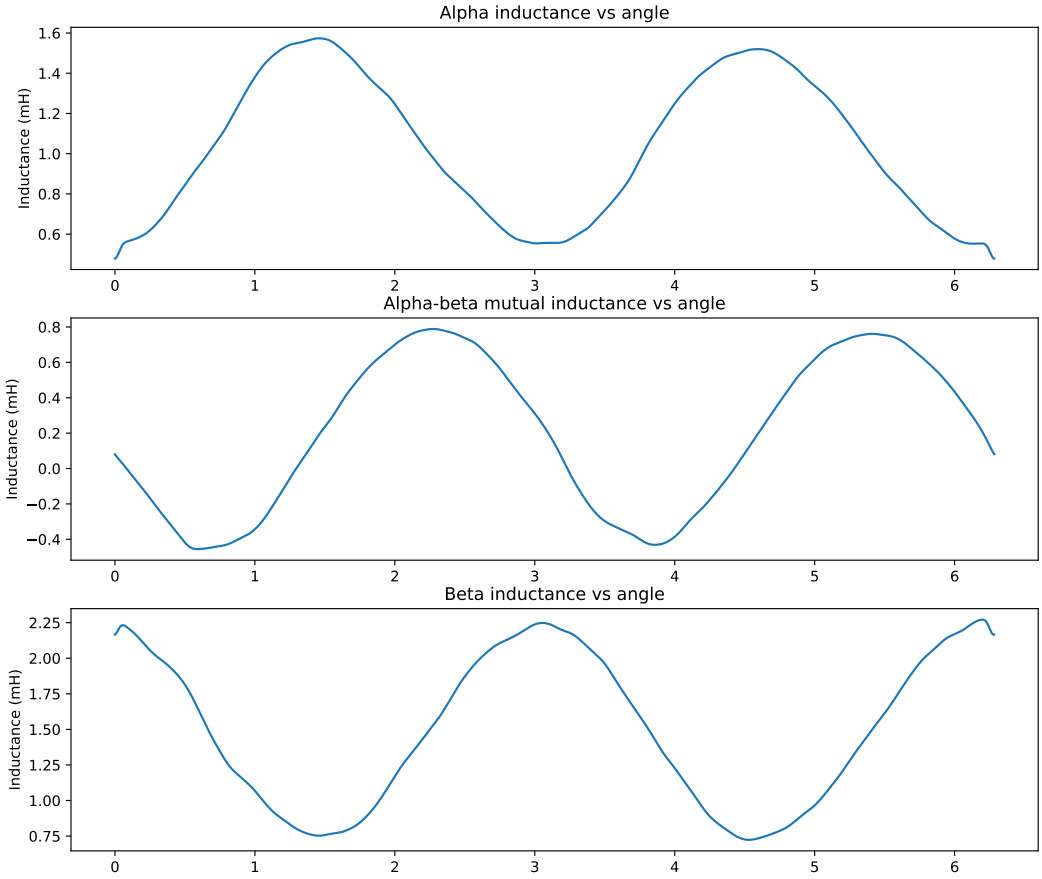

Figure 3: Measured alpha and beta inductance. Alpha and beta inductance vary somewhat sinusoidally with phase angle.

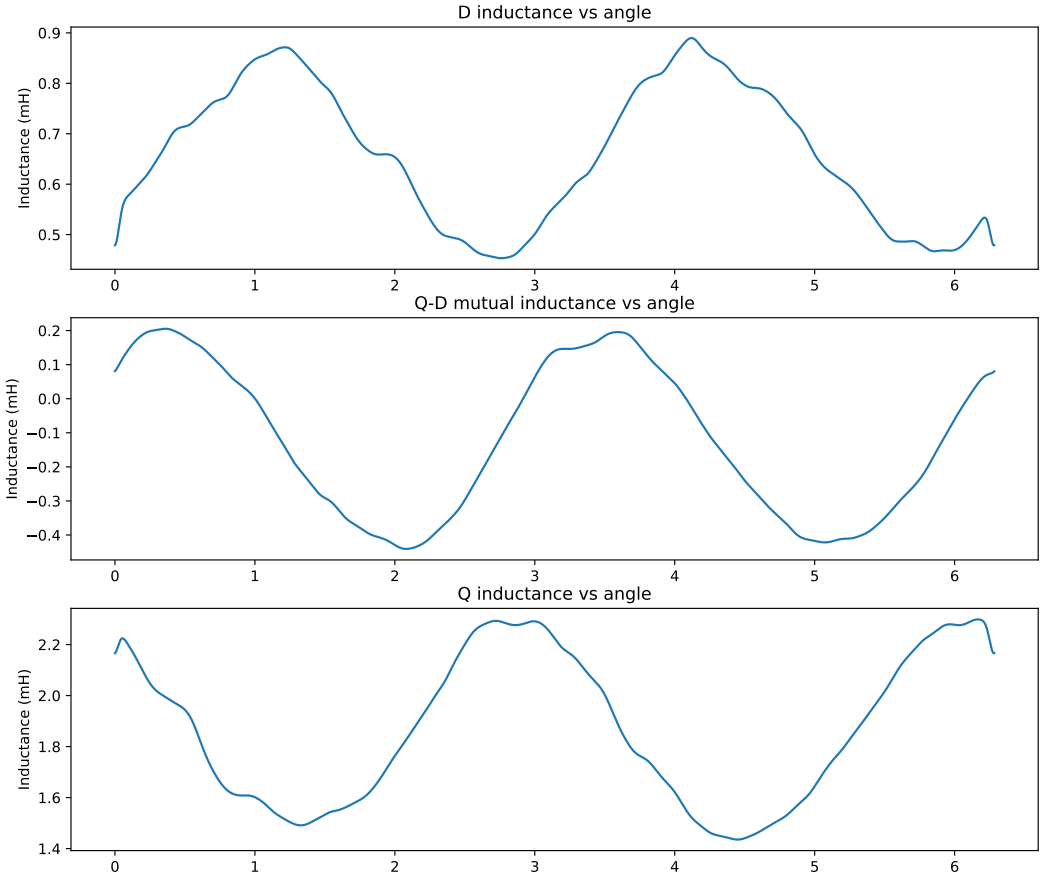

Figure 4: Measured Q and D inductance. The inductance in phase with the rotor is substantially lower than the inductance 90° out of phase. This difference in inductance is useful for generating reluctance torque.

```
import numpy as np
import scipy.signal
from numpy import genfromtxt
import matplotlib.pyplot as plt
import os

filename0 = 'scope_captures/RigolDS0.csv'
filename1 = 'scope_captures/RigolDS1.csv'
dt = 5e-7

# applies a filter to an fft
def bandpass(arr, coeff, offset=0):
    arr *= np.exp(-((np.arange(arr.shape[-1])-offset)/coeff)**2)

clarke = 1/3 * np.array([[2, -1, -1], [0, np.sqrt(3), -np.sqrt(3)]])


# extract 2 khz and low frequency components of the measured signals
# extract exactly one cycle
def get_waveforms(filename):
    x = genfromtxt(filename, delimiter=',')
    t = x[1:, 0]
    uvw = x[1:, 1:].transpose()
    dt = 5e-7

    uvw_fft = np.fft.rfft(uvw)
    uvw_emf_fft = uvw_fft.copy()
    bandpass(uvw_emf_fft, 250, 0)
    uvw_ind_fft = uvw_fft.copy()
    bandpass(uvw_ind_fft, 250, 1000)
    uvw_emf = np.fft.irfft(uvw_emf_fft)
    uvw_ind = np.fft.irfft(uvw_ind_fft)
    ab_emf = clarke @ uvw_emf
    zero_cross = np.arange(len(t)-1)[np.diff(np.sign(ab_emf[0, :])) < 0]
    t_start = zero_cross[3]
    t_end = zero_cross[4]

    r = range(t_start, t_end)
    return t[r], ab_emf[:, r], uvw_ind[:, r]

# determine real and imaginary parts of high frequency injected signals
def quadrature_demodulate(sig):
    N = sig.shape[-1]
    D_ax = np.cos(np.arange(N)*2000*dt*(2*np.pi))
    Q_ax = np.sin(np.arange(N)*2000*dt*(2*np.pi))
```

```
    D_ind = 2 * D_ax * sig
    Q_ind = 2 * Q_ax * sig

    D_ind_fft = np.fft.rfft(D_ind)
    bandpass(D_ind_fft, 100, 0)
    D_ind = np.fft.irfft(D_ind_fft)

    Q_ind_fft = np.fft.rfft(Q_ind)
    bandpass(Q_ind_fft, 100, 0)
    Q_ind = np.fft.irfft(Q_ind_fft)

    return D_ind, Q_ind

# calculate Q and D inductances
def calculate_inductances(uvw_ind0, uvw_ind1):
    Duvw0, Quvw0 = quadrature_demodulate(uvw_ind0)
    Duvw1, Quvw1 = quadrature_demodulate(uvw_ind1)
    N = 2**16
    Duvw0 = scipy.signal.resample(Duvw0, N, axis=1)
    Quvw0 = scipy.signal.resample(Quvw0, N, axis=1)
    Duvw1 = scipy.signal.resample(Duvw1, N, axis=1)
    Quvw1 = scipy.signal.resample(Quvw1, N, axis=1)

    Dab0 = clarke @ Duvw0
    Qab0 = clarke @ Quvw0

    Dab1 = clarke @ Duvw1
    Qab1 = clarke @ Quvw1

    R = 50
    w = 2000*np.pi*2
    DIa0 = -Duvw0[0, :] / R
    QIa0 = -Quvw0[0, :] / R

    DIb0 = -Dab0[1, :] / R
    QIb0 = -Qab0[1, :] / R

    DIa1 = -Dab1[0, :] / R
    QIa1 = -Qab1[0, :] / R

    b1_voltage = 5 * 2*np.sqrt(3)/3 / (2*np.sqrt(2))
    b1_init_mag = np.sqrt(Dab1[1, 0]**2 + Qab1[1, 0]**2)
    phase_offset_b1 = np.arccos(b1_init_mag / b1_voltage)
```

```
    print(b1_voltage)
    print(b1_init_mag)
    print(phase_offset_b1)

    DVin1 = b1_voltage * (Dab1[1, 0] * np.cos(phase_offset_b1) \
            - Qab1[1, 0] * np.sin(phase_offset_b1))
    QVin1 = b1_voltage * (Qab1[1, 0] * np.cos(phase_offset_b1) \
            + Dab1[1, 0] * np.sin(phase_offset_b1))

    DIb1 = (DVin1-Dab1[1, :]) / R
    QIb1 = (QVin1-Qab1[1, :]) / R

    Dfluxab0 = -Qab0 / w
    Qfluxab0 =  Dab0 / w

    Dfluxab1 = -Qab1 / w
    Qfluxab1 =  Dab0 / w

    Lab = np.zeros((N, 2, 2))
    park = np.zeros((N, 2, 2))
    theta = np.arange(N)*2*np.pi/N

    for i in range(N):
        lam = np.array([Dfluxab0[:, i], Qfluxab0[:, i], \
                Dfluxab1[:, i], Qfluxab1[:, i]]).transpose()
        cur =  np.array([[DIa0[i], QIa0[i], DIa1[i], QIa1[i]], \
                [DIb0[i], QIb0[i], DIb1[i], QIb1[i]]])
        Lab[i, :, :] =  lam @ np.linalg.pinv(cur)
        Lab[i, :, :] = 0.5 * (Lab[i, :, :] + Lab[i, :, :].transpose())
        park[i, :, :] = [[np.cos(theta[i]), np.sin(theta[i])], \
                [-np.sin(theta[i]), np.cos(theta[i])]]

    Ldq = park @ Lab @ np.linalg.inv(park)
    return theta, Lab, Ldq


# Plotting everything!
plt.ion()
t0, ab_emf0, uvw_ind0 = get_waveforms(filename0)
t1, ab_emf1, uvw_ind1 = get_waveforms(filename1)

ab_flux0 = np.cumsum(ab_emf0, axis=1)*dt
ab_flux0 -= np.mean(ab_flux0, axis=1).reshape((-1, 1))
angle = np.linspace(0, 2*np.pi, len(t0), endpoint=False)
```

```
ab_ind0 = clarke @ uvw_ind0
ab_ind1 = clarke @ uvw_ind1
fig, ax = plt.subplots(2, 1)
ax[0].plot(t0, ab_emf0[0, :], label='alpha back emf')
ax[0].plot(t0, ab_emf0[1, :], label='beta back emf')
ax[0].legend()
ax[0].set_xlabel("Time")
ax[0].set_ylabel("Back emf (V)")
ax[0].set_title("Back emf vs time")

ax[1].plot(angle, ab_flux0[0, :], label = 'alpha flux linkage')
ax[1].plot(angle, ab_flux0[1, :], label = 'beta flux linkage')
ax[1].legend()
ax[1].set_xlabel("Angle")
ax[1].set_ylabel("Flux Linkage (Vs)")
ax[1].set_title("Flux linkage vs angle")

theta, Lab, Ldq = calculate_inductances(uvw_ind0, uvw_ind1)

fig, ax = plt.subplots(3, 1)
ax[0].plot(theta, 1000*Lab[:, 0, 0])
ax[1].plot(theta, 1000*Lab[:, 1, 0])
ax[2].plot(theta, 1000*Lab[:, 1, 1])

ax[0].set_ylabel("Inductance (mH)")
ax[1].set_ylabel("Inductance (mH)")
ax[2].set_ylabel("Inductance (mH)")
ax[0].set_title("Alpha inductance vs angle")
ax[1].set_title("Alpha-beta mutual inductance vs angle")
ax[2].set_title("Beta inductance vs angle")


fig, ax = plt.subplots(3, 1)
ax[0].plot(theta, 1000*Ldq[:, 0, 0])
ax[1].plot(theta, 1000*Ldq[:, 1, 0])
ax[2].plot(theta, 1000*Ldq[:, 1, 1])

ax[0].set_ylabel("Inductance (mH)")
ax[1].set_ylabel("Inductance (mH)")
ax[2].set_ylabel("Inductance (mH)")

ax[0].set_title("D inductance vs angle")
ax[1].set_title("Q-D mutual inductance vs angle")
ax[2].set_title("Q inductance vs angle")
```

```
plt.show()
```