PololuMotorCharacterization

March 6, 2018

1 Motor Characterization: Pololu

2018 March 3 Jean-Christophe Perrin

This jupyter notebook adapts motor characterization code from the ME112 crawler project to characterize our new motor for the final project. Jan Sokol carried out the empirical testing of the motor on March 3.

Specifically, we are looking at a Tamiya 72005 6-Speed Gearbox Kit.

```
In [1]: import matplotlib.pyplot as plt
        import pandas as pd
        import numpy as np
```

1.1 Useful Constants

These are some simple conversions and constants that will be useful later in our calculations.

```
In [2]: m_to_in = 39.37                       # Meters to inches
        N_to_lbf = 0.2248                     # Newtons to pounds force
        Nm_to_inlbf = N_to_lbf*m_to_in        # Nm to inchpounds torque
        inprsec_ftprmin = 60.0/12             # inches/second to feet/minute
        rpm_to_radps = 2*np.pi/60             # rotations/minute to radians/second
        deg_to_rad = np.pi/180                # degrees to radians
        watt_to_hp = 745.7                    # Watts to horsepower
```

1.2 Importing Lab Data

Jan Sokol collected data from our motor concerning different current-voltage pairs at stall (when the motor axel is helld fixed) and at no load (freely spinning). From these measurements we can characterize the expected output of the motor. They reside in an excel document that should not be altered

```
In [3]: xls = pd.ExcelFile('motorPololu_Data.xlsx')
        nl = pd.read_excel(xls, "noLoad")
        stall = pd.read_excel(xls, "stall")
```

1.3 Motor Constant Calculations

1.3.1 Finding R

At $\omega = 0$ we know that $V = iR$. Therefore, R of the motor is equal to V/i_stall

```
In [4]: stall['R'] = stall['V'] / stall['I']
        R = np.mean(stall['R']) # [Ohm]

        print R
```

```
0.802399868086
```

1.3.2 Finding K (Motor Constant)

At $\tau_l = 0$ we know that the motor equation becomes:

$$V - i_{nl}R - k\omega_{nl} = 0 \tag{1}$$

Rearranging, we find

$$k = \frac{V - i_{nl}R}{\omega_{nl}} \tag{2}$$

```
In [5]: nl['w'] = nl['RPM'] / 60 * 2 * np.pi       # [rad/s] converting to rad/s
        nl['k'] = (nl['V'] - nl['I'] * R)/nl['w']
```

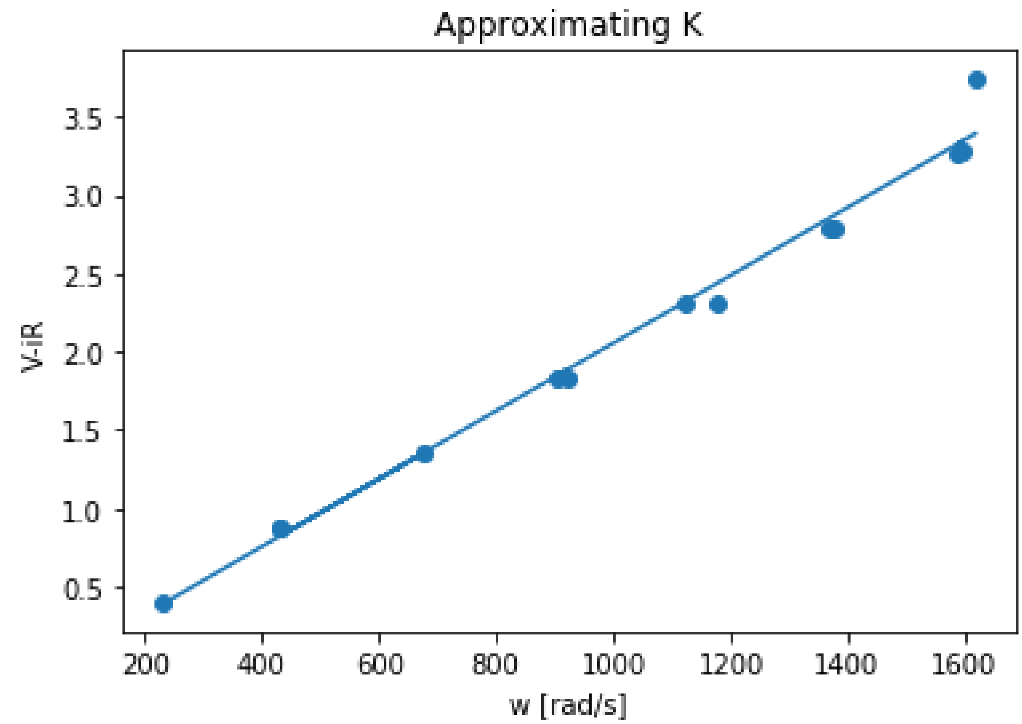
Now that we have value for an estimated k at each collected data point, let's find the linear regression of the data and use the slope (or average k) as an estimator of the actual K value.

```
In [6]: K, intercept = np.polyfit(nl['w'], nl['w']*nl['k'], 1)

        kChecking = plt.figure()
        ax1 = kChecking.add_subplot(111)
        ax1.scatter(nl['w'], nl['V'] - nl['I'] * R)
        ax1.plot(nl['w'], K*nl['w']+intercept)
        ax1.set(xlabel='w [rad/s]', ylabel='V-iR', title='Approximating K')
        kChecking.savefig('motorConstant.png')

        print K
```

```
0.0021645623179572596
```



1.4 Finding τ_l

At no load, the motor is spinning freely, $T_L = 0$, and the only torque is the friction torque, T_f.

$$ki_{nl} = T_f \quad (3)$$

In [7]:
```
nl['Tf'] = nl['k']*nl['I']
Tf = np.mean(nl['Tf'])  # [Nm] Frictional Torque of the motor
print Tf
```

```
0.00045494957087
```

2 Determining Power, Efficiency and Torque

The power output by the motor will vary over operating range of the motor (in amps). We plan on operating our motor at 3 V DC and somewhere between the stall and no-load current.

This is currently a hard-coded value from the data collected because I couldn't figure out how to slice the dataFrame correctly. Future developers of this code should figure out how to pull the iMin and iMax values directly from the collected data.

In [8]:
```
operatingVoltage = 3 # [V]
iMin = 0.26
```

```
iMax = 3.41

iRange = np.linspace(iMin, iMax)
```

In [9]:
```
torqueLoad = K*iRange - Tf
angularVelocity = (operatingVoltage - iRange*R)/K
powerOut = torqueLoad * angularVelocity
powerIn = iRange * operatingVoltage
efficiency = powerOut/powerIn
```

Here we plot a number of the different curves over the operating range. We'll combine the three curves into a single figure as they are frequently referenced together.

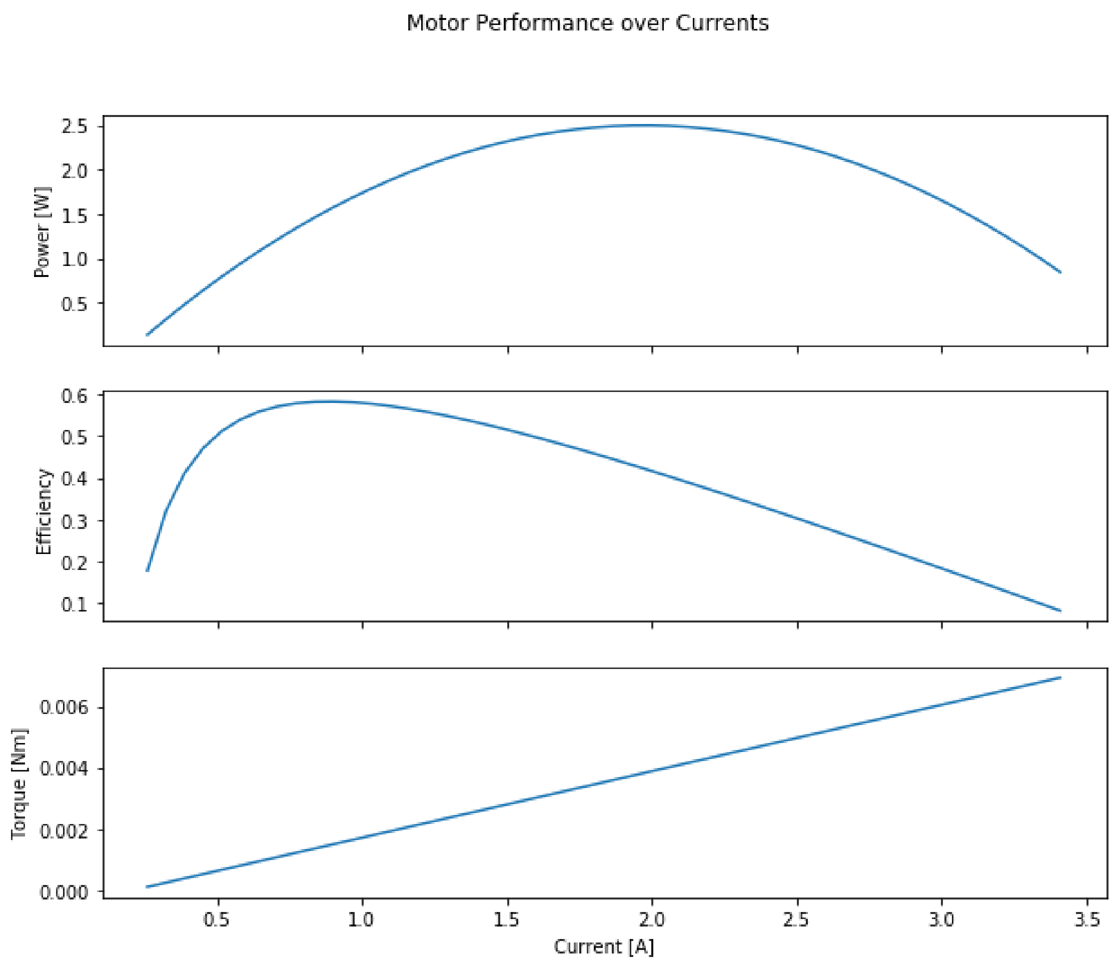
In [10]:
```
fig, (powerCurve, efficiencyCurve, torqueCurve) = plt.subplots(nrows=3, ncols=1, sharex
fig.suptitle('Motor Performance over Currents')

powerCurve.plot(iRange, powerOut)
powerCurve.set(ylabel='Power [W]')

efficiencyCurve.plot(iRange, efficiency)
efficiencyCurve.set(ylabel='Efficiency')

torqueCurve.plot(iRange, torqueLoad)
torqueCurve.set(xlabel='Current [A]', ylabel='Torque [Nm]')

fig.savefig('motorPerformance.png')
```


Motor Performance over Currents
Power [W]
2.5
2.0
1.5
1.0
0.5
Efficiency
0.6
0.5
0.4
0.3
0.2
0.1
Torque [Nm]
0.006
0.004
0.002
0.000
0.5
1.0
1.5
2.0
2.5
3.0
3.5
Current [A]