Multiple Linear Regression for Robot Calibration

In this lab, we will illustrate the use of multiple linear regression for calibrating robot control. In addition to reviewing the concepts in the multiple linear regression demo, you will see how to use multiple linear regression for time series data -- an important concept in dynamical systems such as robotics.

The robot data for the lab is taken generously from the TU Dortmund's Multiple Link Robot Arms Project. As part of the project, they have created an excellent public dataset: MERIt -- A Multi-Elastic-Link Robot Identification Dataset that can be used for understanding robot dynamics. The data is from a three link robot:

We will focus on predicting the current draw into one of the joints as a function of the robot motion. Such models are essential in predicting the overall robot power consumption. Several other models could also be used.

Load and Visualize the Data

First, import the modules we will need.

```
In [6]:  import pandas as pd
         import numpy as np
         import matplotlib
         import matplotlib.pyplot as plt
         # %matplotlib inline
```

The full MERIt dataset can be obtained from the MERIt site. But, this dataset is large. Included in this repository are two of the ten experiments. Each experiments corresonds to 80 seconds of recorded motion. We will use the following files:

exp1.csv for training exp2.csv for test Below, I have supplied the column headers in the names array. Use the pd.read_csv command to load the data. Use the index_col option to specify that column 0 (the one with time) is the index column. You can review simple linear regression demo for examples of using the pd.read_csv command.

```
In [7]:  names =[
             't',                              # Time (secs)
             'q1', 'q2', 'q3',                 # Joint angle   (rads)
             'dq1', 'dq2', 'dq3',              # Joint velocity (rads/sec)
             'I1', 'I2', 'I3',                 # Motor current (A)
             'eps21', 'eps22', 'eps31', 'eps32',  # Strain gauge measurements ($\mu$m
          /m )
             'ddq1', 'ddq2', 'ddq3'            # Joint accelerations (rad/sec^2)
         ]
         # TODO
         # df = pd.read_csv(...)
         df = pd.read_csv('exp1.csv',header=None,sep=',',names=names,index_col=0)
```

Print the first six lines of the pandas dataframe and manually check that they match the first rows of the csv file.

In [8]: `print(df.head(6))`

```
                q1      q2      q3         dq1             dq2            dq3  \
t
0.00 -0.000007  2.4958 -1.1345 -7.882100e-21 -4.940656e-321  3.913100e-29
0.01 -0.000007  2.4958 -1.1345 -2.258200e-21 -4.940656e-321  2.626200e-31
0.02 -0.000007  2.4958 -1.1345 -6.469800e-22 -4.940656e-321  1.762500e-33
0.03 -0.000007  2.4958 -1.1345 -1.853600e-22 -4.940656e-321  1.182800e-35
0.04 -0.000007  2.4958 -1.1345 -5.310600e-23 -4.940656e-321 -5.270900e-03
0.05 -0.000007  2.4958 -1.1345 -1.521500e-23 -4.940656e-321  3.252600e-04

           I1       I2       I3    eps21   eps22   eps31    eps32  \
t
0.00 -0.081623 -0.40812 -0.30609 -269.25 -113.20  3.5918  1.57860
0.01 -0.037411 -0.37241 -0.26698 -270.91 -116.05  1.4585 -1.73980
0.02 -0.066319 -0.40302 -0.31459 -269.25 -112.97  3.5918  0.86753
0.03 -0.068020 -0.43703 -0.28398 -269.97 -114.39  1.6956 -0.08059
0.04 -0.052715 -0.40472 -0.30779 -269.97 -114.15  3.1177  0.86753
0.05 -0.088425 -0.42342 -0.29589 -269.25 -114.15  2.4066 -0.08059

            ddq1          ddq2          ddq3
t
0.00 -9.904900e-19 -6.210306e-319  4.917400e-27
0.01  4.248100e-19 -1.766878e-319 -1.381100e-27
0.02  3.233800e-19 -4.990557e-320 -4.117300e-28
0.03  1.500500e-19 -1.394253e-320 -1.173100e-28
0.04  5.932400e-20 -3.581976e-321 -3.770800e-01
0.05  2.164600e-20 -1.141292e-321  2.930300e-01
```
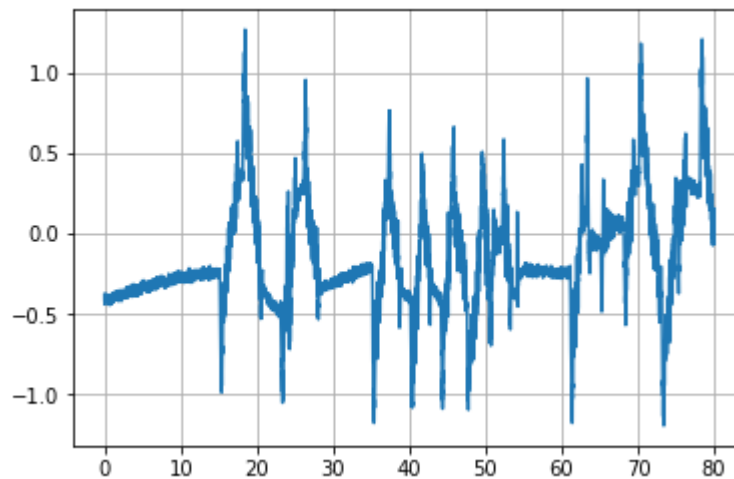
From the dataframe df, extract the time indices into a vector t and extract I2, the current into the second joint. Place the current in a vector y and plot y vs. t. Label the axes with the units.

```
In [9]:  # TODO
         # y = ...
         # t = ...
         # plt.plot(...)
         y = np.array(df['I2'])
         t = np.array(df.index)

         plt.plot(t,y)
         plt.grid()
         plt.show()
```


Use all the samples from the experiment 1 dataset to create the training data:

ytrain: A vector of all the samples from the I2 column Xtrain: A matrix of the data with the columns:
['q2','dq2','eps21', 'eps22', 'eps31', 'eps32','ddq2']

```
In [10]:  # TODO
          # ytrain = ...
          # Xtrain = ...
          ytrain = np.array(df['I2'])
          Xtrain = np.array(df[['q2','dq2','eps21','eps22','eps31','eps32','ddq2']])
```

Fit a Linear Model

Use the sklearn.linear_model module to create a LinearRegression class regr.

```
In [11]:  from sklearn import linear_model

          # Create linear regression object
          # TODO
          # regr = ...
          regr = linear_model.LinearRegression()
```

Train the model on the training data.

```
In [12]: regr.fit(Xtrain,ytrain)

Out[12]: LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)
```

Using the trained model, compute, ytrain_pred, the predicted current. Plot ytrain_pred vs. time t. On the same plot, plot the actual current ytrain vs. time t. Create a legend for the plot.

```
In [13]: ytrain_pred = regr.predict(Xtrain)
         plt.plot(t,ytrain)
         plt.plot(t,ytrain_pred)
         plt.legend(['actual','predicted'])
         plt.show()
```
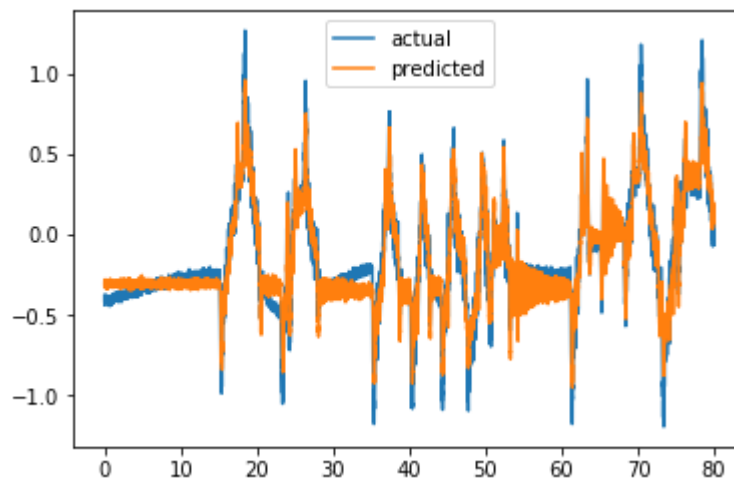

Measure the normalized RSS given by `RSS / (n s^2_y).

```
In [14]: # TODO
         # RSS_train = ...
         RSS_train = np.mean((ytrain-ytrain_pred)**2)/np.mean((ytrain-np.mean(ytrain))*
         *2)
         print(RSS_train)

         0.09583263861233196
```

Measure the Fit on an Indepdent Dataset

Up to now, we have only tested the model on the same data on which it was trained. In general, we need to test model on independent data not used in the training. For this purpose, load the data in exp2.csv. Compute the regression predicted values on this data and plot the predicted and actual values over time.

```
In [15]: df = pd.read_csv('exp2.csv',header=None,sep=',',names=names,index_col=0)


         ytest = np.array(df['I2'])
         Xtest = np.array(df[['q2','dq2','eps21','eps22','eps31','eps32','ddq2']])
         ttest = np.array(df.index)
         ytest_pred = regr.predict(Xtest)


         plt.plot(t,ytest)
         plt.plot(t,ytest_pred)
         plt.legend(['actual','predicted'])
         plt.show()
```
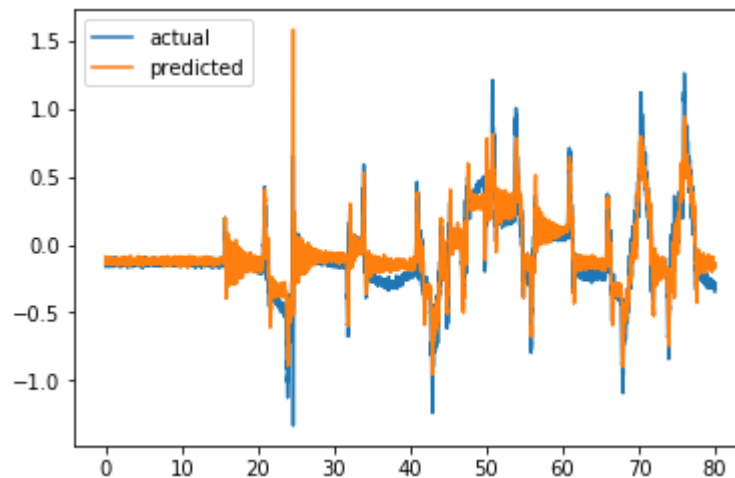

Measure the normalized RSS on the test data.

```
In [16]: RSS_test = np.mean((ytest-ytest_pred)**2)/np.mean((ytest-np.mean(ytest))**2)
         print(RSS_test)
```

```
0.12678048804762396
```