Problem 1

The answers to the stated questions:

1. **Q:** Exercise 2. How many time points N_{tr} does the train set contain? How many time points N_{te} does the test set contain? At each time point, at how many electrodes D_X was the EMG collected?

   A: $N_{tr} = 5000$, $N_{te} = 5255$ and $D_X = 192$.

2. **Q:** Exercise 3. Do you notice a performance difference between train and test data set?

   A: The precision of the prediction on the test data set is lower, especially for the outliers.

3. **Q:** Exercise 4. Comment the line where we logarithmize the EMG features. Do you notice a performance difference compared to the logarithmized version?

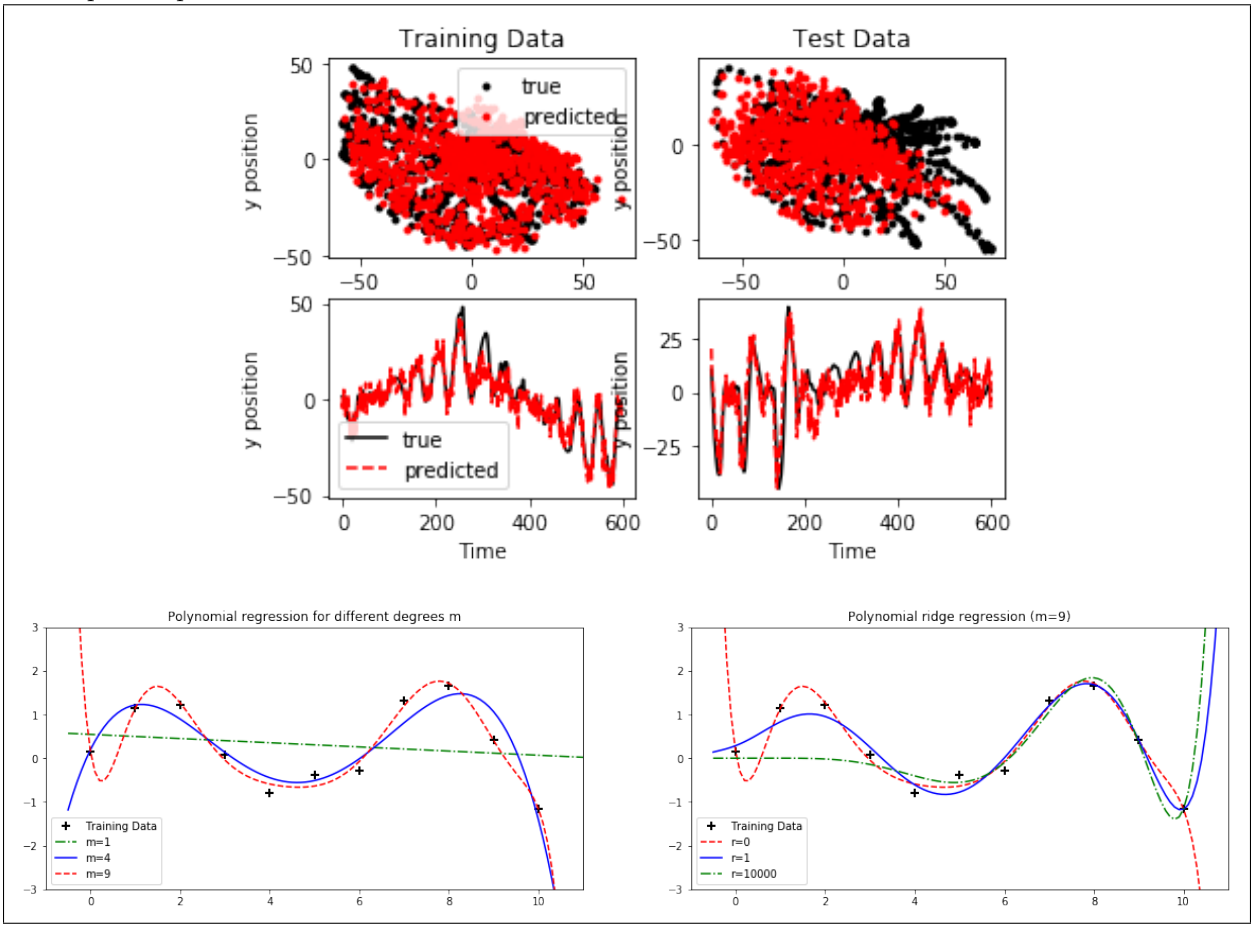
   A: The accuracy of the prediction decreases compared to the logarithmized version, on both training set and testing set.

4. **Q:** Exercise 5. If we cannot predict the labels Y perfectly by a linear regression on X, does this imply that the relationship between X and Y is non-linear?

   A: Yes! If the relationship between X and Y is linear then, by linear regression, we can perfectly predict the labels Y (but if we have some noise on Y, even if the relationship between X and Y is linear, we cannot perfectly predict the labels Y).

Problem 2

The requested plots:



Problem 3

The code for the functions *train_ols*, *apply_ols* and *test_polynomial_regression* is the flowing:

```
def train_ols(X_train, Y_train, llambda = 0):
    XXT = X_train.dot(X_train.T)
    XXTX = inv(XXT + llambda * sp.identity(len(X_train.T[0]))).dot(X_train)
    return XXTX.dot(Y_train.T)
```

```
def apply_ols(W, X_test):
    return W.T.dot(X_test)
```

```
def test_polynomial_regression():
    d_toy = sp.sin(sp.arange(11)) + sp.random.normal(0, 0.5, 11)

    #Apply and visualize polynomial ridge regression for different parameters:
    #a) m = 1, 4, 9 with Îż = 0
    pl.figure(figsize=(20,10))
    pl.subplot(2,2,1)
    pl.plot(sp.arange(11), d_toy, 'k+', label='Training Data', mew=2.0, ms=8.0)
```

```
    X, Y = train_lin_reg(2,0,d_toy)
    pl.plot(X, Y.T, 'g-.', label = 'm=1')

    X, Y = train_lin_reg(5,0,d_toy)
    pl.plot(X, Y.T, 'b-' , label = 'm=4')

    X, Y = train_lin_reg(10,0,d_toy)
    pl.plot(X, Y.T, 'r--', label = 'm=9')

    pl.axis([-1, 11, -3, 3])
    pl.legend(loc = 3)
    pl.title('Polynomial regression for different degrees m')
    ##############################################
    #b) Îż = 0, 1, 10000 with m = 9
    pl.subplot(2,2,2)
    pl.plot(sp.arange(11), d_toy, 'k+', label='Training Data', mew=2.0, ms=8.0)

    X, Y = train_lin_reg(10,0,d_toy)
    pl.plot(X, Y.T, 'r--', label = 'r=0')

    X, Y = train_lin_reg(10,1,d_toy)
    pl.plot(X, Y.T, 'b-' , label = 'r=1')

    X, Y = train_lin_reg(10,10000,d_toy)
    pl.plot(X, Y.T, 'g-.', label = 'r=10000')
    pl.title('Polynomial ridge regression (m=9)')
    pl.axis([-1, 11, -3, 3])
    pl.legend(loc = 3)
```