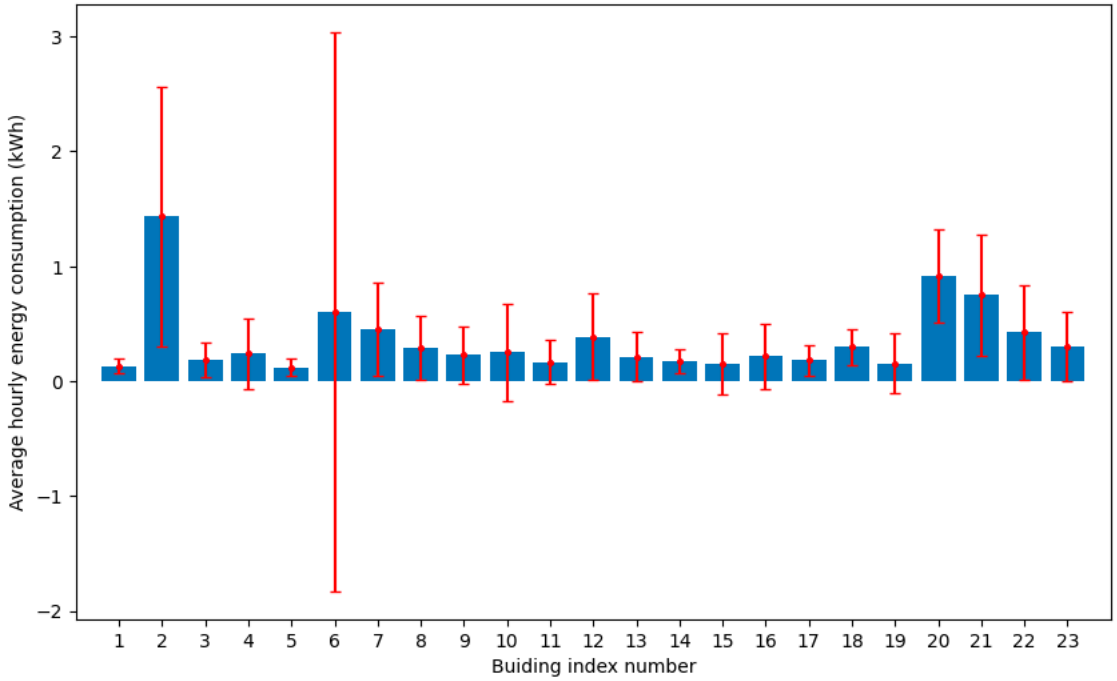
HW 4: Forecasting Residential Electricity Power Consumption

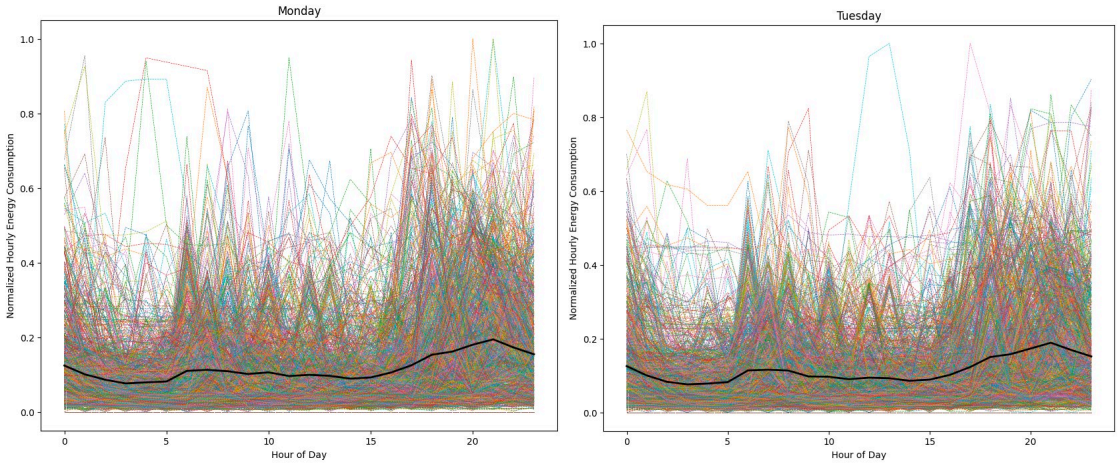
Problem 1

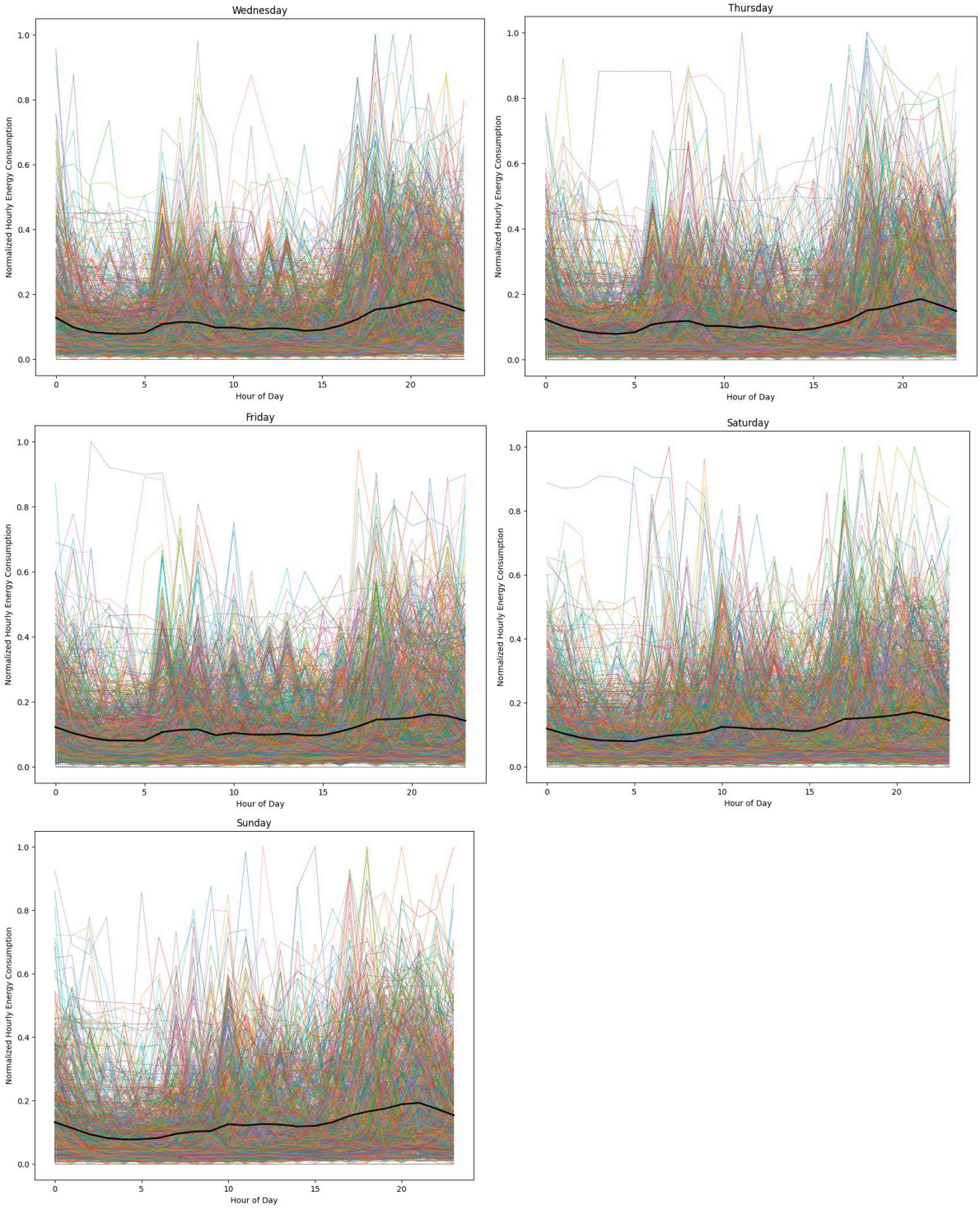
a.



b. Building 6 has negative power consumption, so I removed Building 6 from data set.

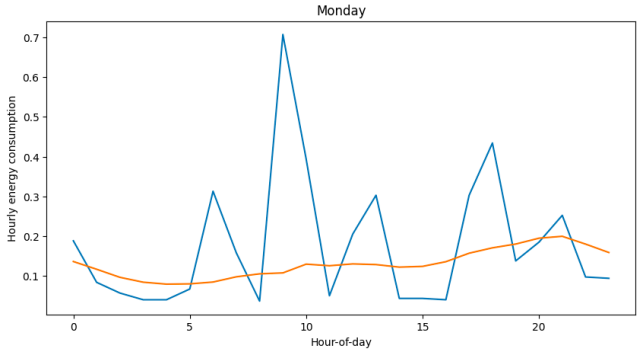
c.

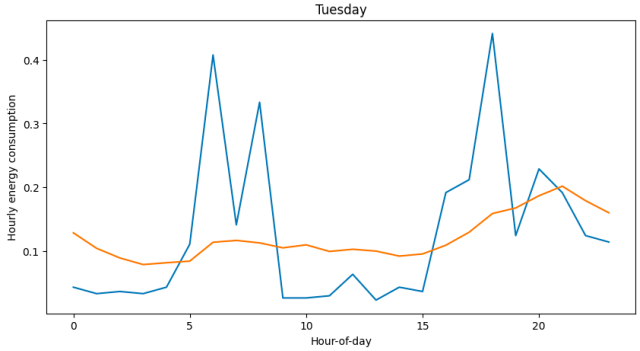


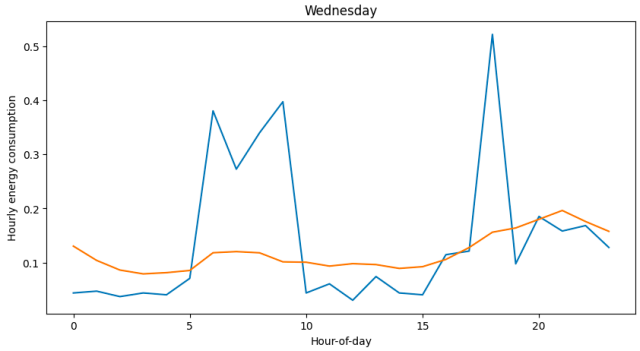


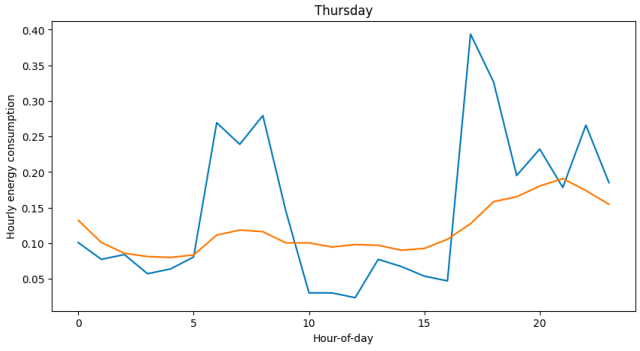
Consumption of weekdays are quite similar; however, Monday has a higher consumption in the evening.

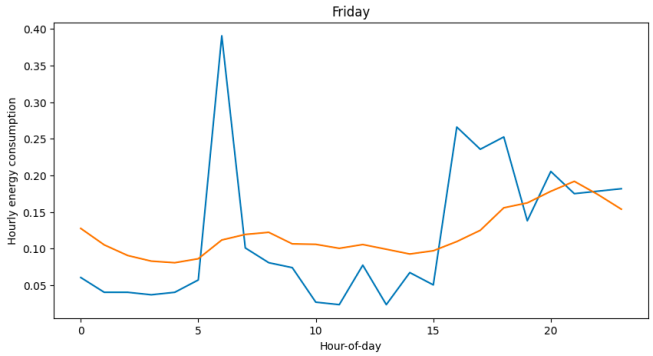
Problem 2

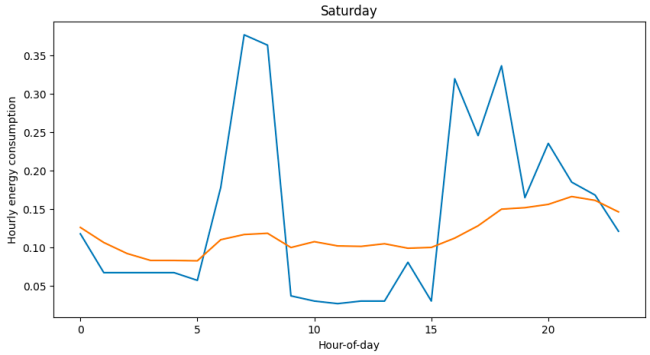

Monday
Hourly energy consumption
Hour-of-day

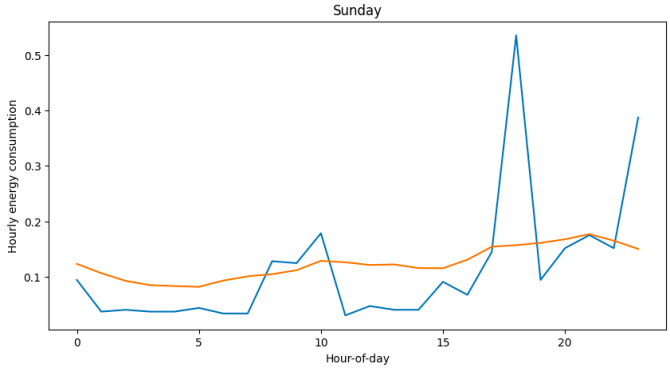

Tuesday
Hourly energy consumption
Hour-of-day


Wednesday
Hourly energy consumption
Hour-of-day


Thursday
Hourly energy consumption
Hour-of-day







b.
Monday 0.108930
Tuesday 0.079239
Wednesday 0.084258
Thursday 0.067724
Friday 0.062153
Saturday 0.079013
Sunday 0.072854
MAE for the entire week: 0.07956024123513142

Monday has largest MAE
Friday has smallest MAE

Problem 3

(a). $Y = \hat{P}_{arx} - \hat{P}_{avg}$

$$\Phi = \begin{bmatrix} P(L) & P(L-1) & \cdots & P(1) \\ P(L+1) & P(L) & \cdots & P(2) \\ \vdots & & & \\ P(n-1) & P(n-2) & \cdots & P(n-L) \end{bmatrix}$$

, n is the number of data points

$$\theta = [\alpha_1, \alpha_2, \cdots \alpha_L]^T$$

(b). $\min_{\theta} \| \Phi\theta - (Y_{train} - \hat{P}_{avg}) \|_2^2$

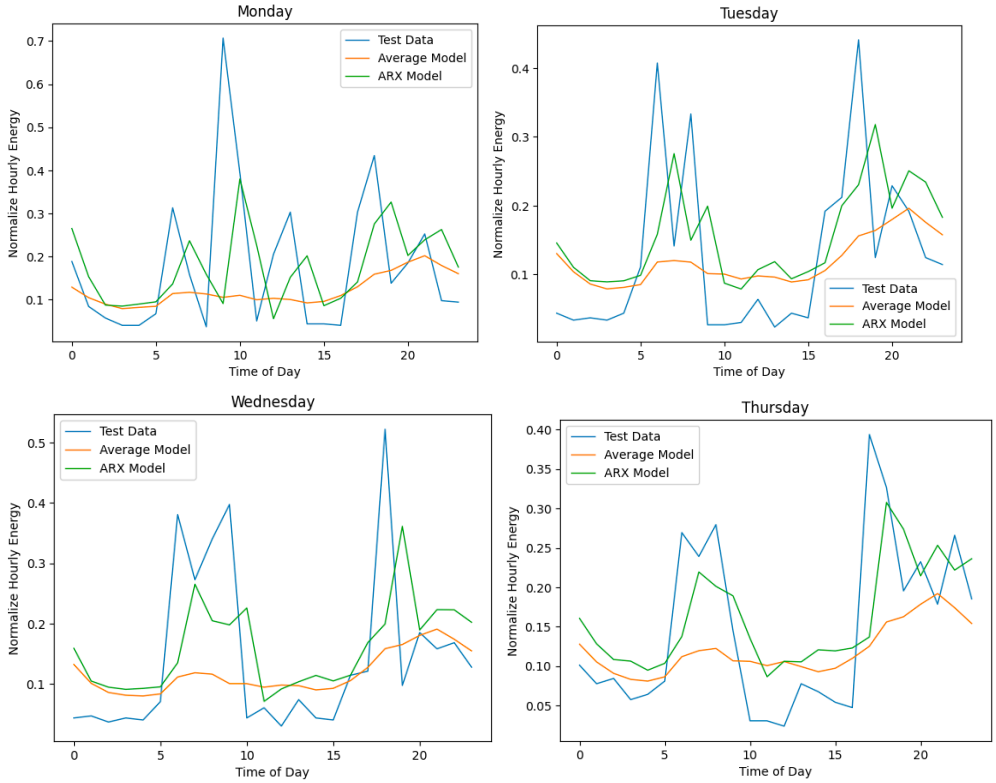
Hessian of θ : $H = 2\Phi^T\Phi \succ 0$

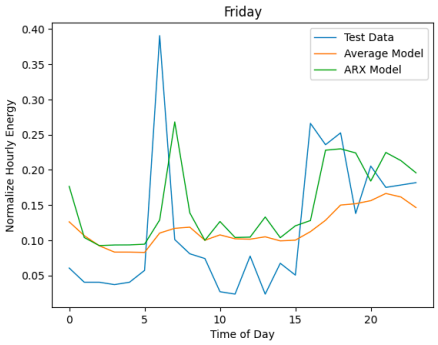
$\Rightarrow$ It's a convex program

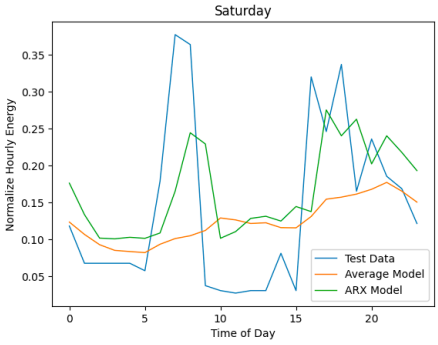
c.
alpha_1: 0.4004167150282461*
alpha_2: -0.04093498583659052*
alpha_3: -0.07277227527726028*

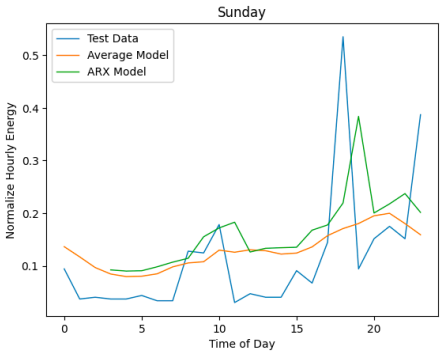
d.
Mon 0. 1178974872283360 8,
Tue 0.09217062317238867,
Wed 0.09136400133009082,
Thu 0.06325217669776463,
Fri 0.07031050008183033,
Sat 0.08290603324082714,
Sun 0.09091157932196983

MAE for week : 0.08690159325933981




Friday
Test Data
Average Model
ARX Model
Normalize Hourly Energy
Time of Day


Saturday
Test Data
Average Model
ARX Model
Normalize Hourly Energy
Time of Day


Sunday
Test Data
Average Model
ARX Model
Normalize Hourly Energy
Time of Day

Problem 4

(a). w.

(b). $\frac{\partial J}{\partial w} = \sum_{i=1}^{m} \frac{\partial J}{\partial \delta^{(i)}} \cdot \frac{\partial \delta^{(i)}}{\partial f} \cdot \frac{\partial f}{\partial z} \cdot \frac{\partial z}{\partial w}$

$\delta^{(i)} = y^{(i)} - f(w^T x^{(i)})$

$f = \tanh$

$z = w^T x$.

$\frac{\partial J}{\partial w} = \sum \delta^{(i)} (1 - \tanh^2(z)) \cdot x$

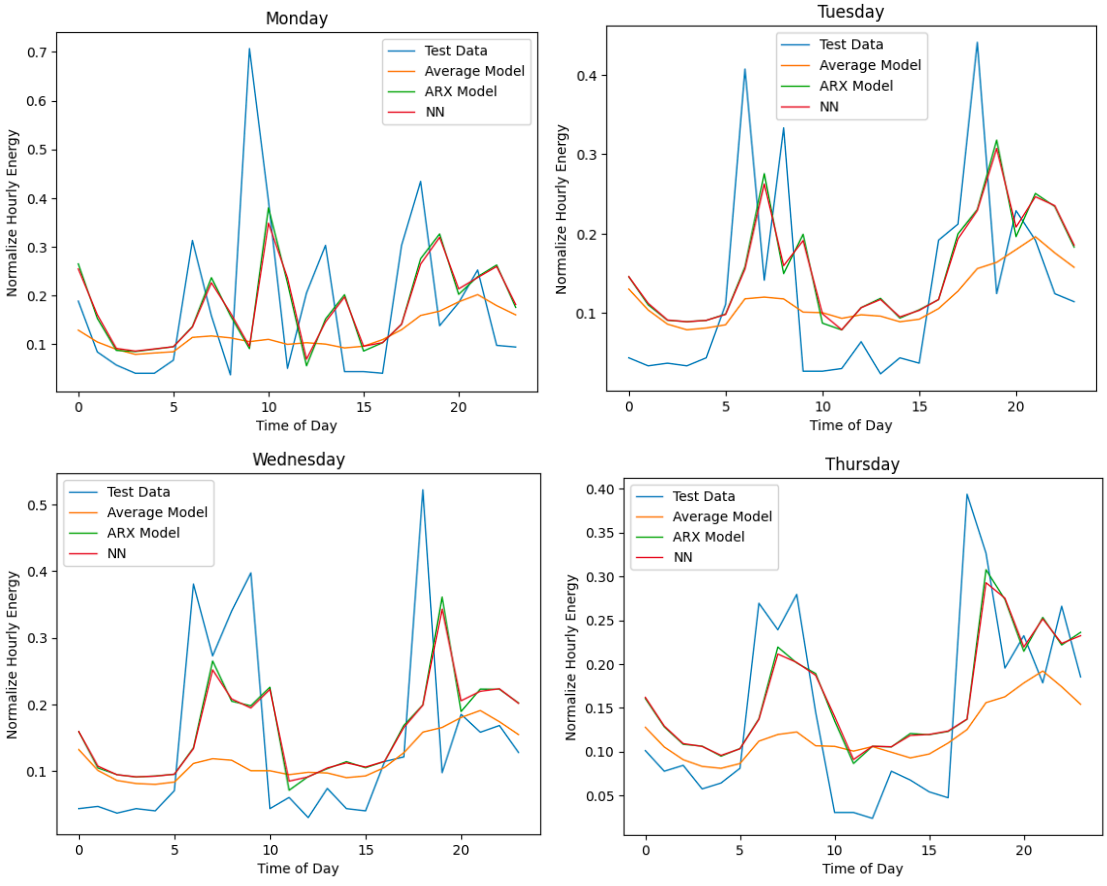
$\Rightarrow w^{k+1} = w^k + \gamma \sum_{i=1}^{m} [(y^{(i)} - \tanh((w^k)^T \cdot x^{(i)}) (1 - \tanh^2((w^k)^T x^{(i)}) \cdot x^{(i)})$

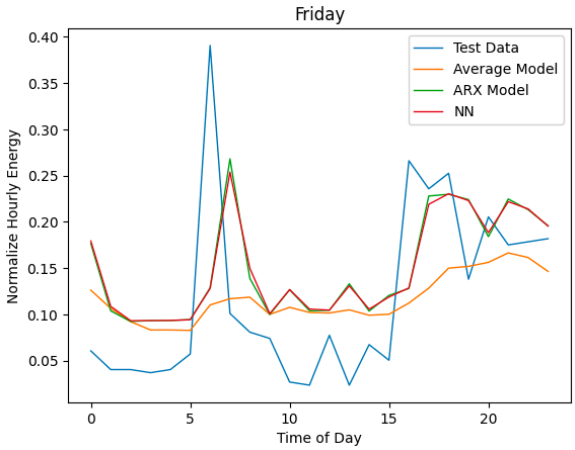
c.
w1 0.360244
w2 -0.002709
w3 -0.072028

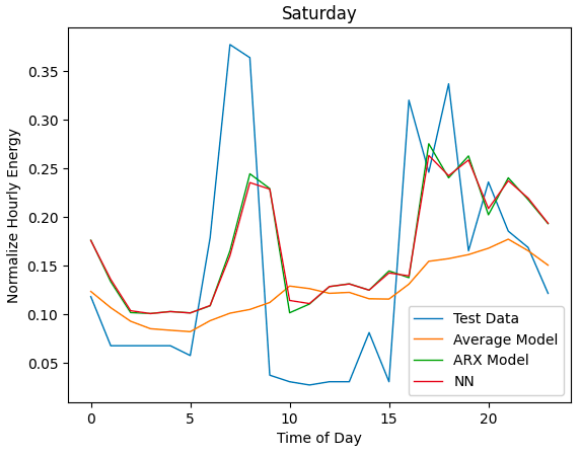
d.

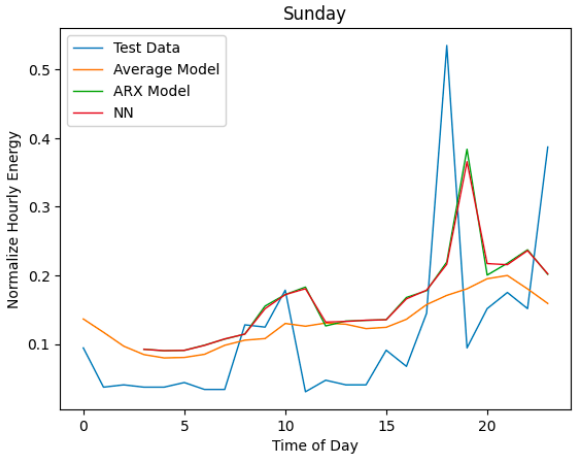
Mon 0.12014379522883845,
Tue 0.09092650856353467,
Wed 0.09220096510663013,
Thu 0.06416041353153716,
Fri 0.07060019636938865,
Sat 0.08291285717534137,
Sun 0.09070733210571848

MAE for week : 0.08731834931894897




Friday
Test Data
Average Model
ARX Model
NN
Normalize Hourly Energy
Time of Day


Saturday
Test Data
Average Model
ARX Model
NN
Normalize Hourly Energy
Time of Day


Sunday
Test Data
Average Model
ARX Model
NN
Normalize Hourly Energy
Time of Day

HW4

April 18, 2023

[ ]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
# # Default plot configurations
# %matplotlib inline
# sns.set()
# plt.rcParams['figure.figsize'] = (16,8)
# plt.rcParams['figure.dpi'] = 100
# plt.rcParams['xtick.labelsize'] = 15
# plt.rcParams['ytick.labelsize'] = 15
# plt.rcParams['legend.fontsize'] = 15
# plt.rcParams['axes.labelsize'] = 15
```

1 Problem 1: Exploratory Data Analysis

[ ]:
```
#1a
data_train = pd.read_csv('HW4_Train_Data.csv')
building_id = np.arange(1,24)
building_avg = pd.Series(data_train.iloc[:, 3:].mean().tolist(),␣
 ↪index=building_id)
building_std = pd.Series(data_train.iloc[:, 3:].std().tolist(),␣
 ↪index=building_id)
fig, ax = plt.subplots(figsize=(10, 6))
plt.bar(building_id, building_avg)
plt.errorbar(building_id, building_avg, yerr=building_std, fmt='.', ms=5,␣
 ↪c='r', capsize=3)
plt.xticks(building_id)
plt.xlim([0, 24])
plt.xlabel('Buiding index number')
plt.ylabel('Average hourly energy consumption (kWh)')
plt.show()
```

[ ]:
```
# 1b
bldg_info = data_train.iloc[:, 3:].T
bldg_neg = bldg_info[(bldg_info.iloc[:, :] < 0).any(axis = 1)]
bldg_neg
```

[ ]:
```
datetime = pd.to_datetime(data_train['Start Time (GMT-0800,PST)'])

data = bldg_info.T.drop(['Bldg6 (kWh)'], axis=1)
data = data / data.max()
data['week_of_year'] =datetime.dt.isocalendar().week
data['day_of_week'] = datetime.dt.dayofweek
data['hour_of_day'] = datetime.dt.hour

# Create a 4-D array
energy_4d_array = np.zeros((22, 53, 7, 24))

for index, row in data.iloc[:,:22].iterrows():
    week = data.loc[index, 'week_of_year'] - 1
    day = data.loc[index, 'day_of_week']
    hour = data.loc[index, 'hour_of_day']
    energy_4d_array[:, week, day, hour] = row.values
```

[ ]:
```
day_names = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday',␣
 ↪'Saturday', 'Sunday']

for day in range(7):
    fig, ax = plt.subplots(figsize=(10, 8))
    for building in range(22):
        for week in range(53):
            ax.plot(range(24), energy_4d_array[building, week, day, :],␣
 ↪linestyle='--',linewidth = 0.5)

    # Plot the average hourly energy consumption in a thick black line
    avg_hourly_energy = energy_4d_array[:, :, day, :].mean(axis=(0, 1))
    ax.plot(range(24), avg_hourly_energy, 'k-', linewidth=2)

    ax.set_xlabel('Hour of Day')
    ax.set_ylabel('Normalized Hourly Energy Consumption')
    ax.set_title(f'{day_names[day]}')
    plt.show()
```

2 Problem 2: Average Model

[ ]:
```
test = pd.read_csv('HW4_Test_Data.csv')
testtime = pd.to_datetime(test['TestTime'])
```

[ ]:
```
test['week_of_year'] =testtime.dt.isocalendar().week-1
test['day_of_week'] = testtime.dt.dayofweek
test['hour_of_day'] = testtime.dt.hour
```

[ ]:
```
data
avg=[]
for day in range(7):
    avg.append(data[data['day_of_week'] == day].groupby('hour_of_day').mean().
 ↪iloc[:, :-2].mean(axis=1))
```

[ ]:
```
test
```

[ ]:
```
test
```

[ ]:
```
P_pred = np.zeros(168)
for i in range(0, 168):
    U = np.zeros((7, 24))
    U[test["day_of_week"][i], test["hour_of_day"][i]] = 1
    P_pred[i]=(sum(sum(avg * U)))
```

[ ]:
```
P_pred.shape
```

[ ]:
```
test.loc[:,"Avg"] = P_pred
```

[ ]:
```
day_names = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday',␣
 ↪'Saturday', 'Sunday']
AVG_mod = np.array(P_pred).reshape(7,24)


for day in range(7):
    fig, ax = plt.subplots(figsize=(10, 5))
    data_tmp = test[test['day_of_week'] == day]
    ax.plot(data_tmp[['hour_of_day']], data_tmp[['TestBldg']])
    ax.plot(np.arange(24), AVG_mod[day]
            , label='Average Model')
    ax.set_xlabel('Hour-of-day')
    ax.set_ylabel('Hourly energy consumption')
    ax.set_title(f'{day_names[day]}')

    plt.show()
```

[ ]:
```
## 2b
MAE_Dow = []
for day in range(7):
    MAE_Dow.append(np.mean(np.abs(test[test['day_of_week'] == day]['TestBldg'].
 ↪values
                                  -avg[day].values)))
MAE_Dow = pd.Series(MAE_Dow, day_names)
print('MAE for DoW:\n', MAE_Dow, sep='')
```

```
# For entire week
MAE_week = np.mean(abs(test['TestBldg'].values - np.array(P_pred)))
print('MAE for the entire week:', MAE_week)
```

3 Problem 3: Autoregressive with eXogeneous Inputs Model (ARX)

```
df_melt = pd.melt(data, id_vars = ["week_of_year", "day_of_week",␣
 ↪"hour_of_day"], var_name = "Bldg", value_name = "Energy")
```

```
test[["day_of_week","hour_of_day","Avg"]]
```

```
ARX_df = df_melt.merge(test[["day_of_week","hour_of_day","Avg"]], how='left',␣
 ↪on=['day_of_week', 'hour_of_day'])
```

```
ARX_df["Y"] = ARX_df["Energy"] - ARX_df["Avg"]
```

```
unique_buildings = ARX_df['Bldg'].unique()

# Initialize empty DataFrames for lag1 and lag2
lag1 = pd.DataFrame()
lag2 = pd.DataFrame()
lag3 = pd.DataFrame()
# Loop through each building and create the lag columns
for building in unique_buildings:
    building_data = ARX_df[ARX_df['Bldg'] == building]

    # Create lag1 and lag2 columns for the current building
    building_data['lag1'] = building_data['Energy'].shift(1)
    building_data['lag2'] = building_data['Energy'].shift(2)
    building_data['lag3'] = building_data['Energy'].shift(3)
    # Concatenate the building_data with the lag columns to the respective␣
 ↪DataFrames
    lag1 = pd.concat([lag1, building_data])
    lag2 = pd.concat([lag2, building_data])
    lag3 = pd.concat([lag3, building_data])
```

```
Phi = lag3[['lag1','lag2', 'lag3']].dropna()
```

```
Y = lag3.dropna()["Y"]
```

```
THETA = (np.linalg.inv(Phi.T @ Phi) @ Phi.T) @ Y
print("alpha_1*: ", THETA[0])
print("alpha_2*: ", THETA[1])
print("alpha_3*: ", THETA[2])
```

[ ]:
```
test
```

[ ]:
```
lag_df = test[["TestBldg"]]
lag_df['lag1'] = lag_df['TestBldg'].shift(1)
lag_df['lag2'] = lag_df['TestBldg'].shift(2)
lag_df['lag3'] = lag_df['TestBldg'].shift(3)
lag_df
```

[ ]:
```
test["ARX"] = (lag_df["lag1"]*THETA[0]) + (lag_df["lag2"]*THETA[1]) +␣
 ↪(lag_df["lag3"]*THETA[2]) + test["Avg"]
```

[ ]:
```
test
```

[ ]:
```
# Plot test data and models
ax = test.groupby("day_of_week").plot.line(x = "hour_of_day", y = ["TestBldg",␣
 ↪"Avg", "ARX"], label = ["Test Data", "Average Model", "ARX Model"], lw = 1)
days = ["Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday",␣
 ↪"Sunday"]
for i in range(0, len(days)):
    ax[i].set_title(days[i])
    ax[i].set_xlabel("Time of Day")
    ax[i].set_ylabel("Normalize Hourly Energy")
```

[ ]:
```
# MAE for each DoW
diff_arx = test.groupby("day_of_week").apply(lambda row: abs(row["TestBldg"] -␣
 ↪row["ARX"]))
mae_dow_arx = [np.mean(diff_arx[i]) for i in range(0, len(days))]
mae_dow_arx
```

[ ]:
```
# MAE for the entire week
mae_week_arx = np.mean(abs(test["TestBldg"] - test["ARX"]))
mae_week_arx
```

4 Problem 4: Neural Network Model

[ ]:
```
# 4c
gamma = 10**-5
omega = np.array([0, 0, 0])
iterations = 200
x_i = Phi.T
y_i = Y
for i in range(iterations):
    z_i = omega @ x_i
    dJ_dd = y_i-np.tanh(z_i)
    dd_df = -1
    df_dz = 1-(np.tanh(z_i)**2)
```

```
    dz_dw = x_i.T
    dJ_dw = (dJ_dd * dd_df * df_dz).T @ dz_dw
    omega = omega - gamma*dJ_dw
omega
```

[ ]:
```
# 4d
test.loc[:, "NN"] = np.tanh(lag_df.iloc[:, 1:].values @ omega) + test["Avg"]

# Plot test data and models
ax = test.groupby("day_of_week").plot.line(x = "hour_of_day", y = ["TestBldg",␣
 ↪"Avg", "ARX", "NN"], label = ["Test Data", "Average Model", "ARX Model",␣
 ↪"NN"], lw = 1)
days = ["Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday",␣
 ↪"Sunday"]
for i in range(0, len(days)):
    ax[i].set_title(days[i])
    ax[i].set_xlabel("Time of Day")
    ax[i].set_ylabel("Normalize Hourly Energy")
```

[ ]:
```
# MAE for each DoW
diff_nn = test.groupby("day_of_week").apply(lambda row: abs(row["TestBldg"] -␣
 ↪row["NN"]))
mae_dow_nn = [np.mean(diff_nn[i]) for i in range(0, 7)]
mae_dow_nn
```

[ ]:
```
mae_week_nn = np.mean(abs(test["TestBldg"] - test["NN"]))
mae_week_nn
```