Simuleringskod för System

Självständigt arbete i energisystem

Delrapport

Oskar Andersson

18 maj 2024

Listings

Huvudkod

```

import matplotlib.pyplot as plt
import IslandSystem as IS
import pandas as pd

# ------------------------ L ser all data -------------------------
file_paths = ['elkonsumtion.xlsx', 'power_output_solceller_hustak.xlsx', '
    power_output_solceller_mark.xlsx', 'energy_production_wind2.xlsx', 'average_wind_speeds.
    xlsx']

columns_to_read = {
    'elkonsumtion': ['Unnamed: 1', 'Avl st  elanv ndning per timme'],
    'power_output_solceller_hustak': ['Date and time', 'Cell temperature', 'Module
    efficiency', 'PV system output'],
    'power_output_solceller_mark': ['Date and time', 'Cell temperature', 'Module efficiency'
    , 'PV system output'],
    'energy_production_wind2': ['Date', 'Power Output (W)'],
    'average_wind_speeds': ['Month-Day-Time', 'Average Wind Speed (m/s)']
}

# Tom dictionary f  r att spara data i
dfs = {}

# L  s igenom alla filer och spara i dfs var f  r sig
for file_path in file_paths:
    filename = file_path.split('.')[0]
    df = pd.read_excel(file_path, engine='openpyxl', usecols=columns_to_read.get(filename))

    # Sparar data med filnamnet som nyckel
    dfs[filename] = df

# l gger till energiproduktion fr n Aeosol-vindkrafverk
file_path_A = 'energy_production_wind2.xlsx'
filename_A = file_path_A.split('.')[0]+'_A'
dfs[filename_A] = pd.read_excel(file_path_A, engine='openpyxl', usecols=['Date', 'Power
    Output (Wh)'], sheet_name='Aeolos_V-3000')


# -------- g  r om fr n halvtimme till timme vind -----------
x = 0
energy_production_wind_hour = [] # F  r Windstar
while x < 17520: # Antal halvtimmar p   ett  r
    column_index = dfs['energy_production_wind2'].columns.get_loc('Power Output (W)')
    datapoint1 = dfs['energy_production_wind2'].iloc[x, column_index]
    datapoint2 = dfs['energy_production_wind2'].iloc[x+1, column_index]
    energy_production_wind_hour.append(datapoint1 + datapoint2)
    x += 2

x = 0
energy_production_wind_hour_A = [] # F  r Aeosol
while x < 17520: # Antal halvtimmar p   ett  r
    column_index = dfs['energy_production_wind2_A'].columns.get_loc('Power Output (Wh)')
    datapoint1 = dfs['energy_production_wind2_A'].iloc[x, column_index]
    datapoint2 = dfs['energy_production_wind2_A'].iloc[x+1, column_index]
    energy_production_wind_hour_A.append(datapoint1 + datapoint2)
    x += 2


#-------------- J  mf  r produktion och konsumtion per timme f  r olika typer av
    anl ggningsalternativ ------

# tomma listor f  r att spara data i
sum_power_tmvW = [] # tak, mark, vind (Windstar)
sum_power_tmvA = [] # tak, mark, vind (Aeolos)
sum_power_tm = [] # tak, mark
sum_power_mvW = [] # mark, vind (Aeolos)
```

```
sum_power_mvA = []
sum_power_tvW = [] # tak, vind (W)
sum_power_tvA = []
sum_power_t = [] # tak
sum_power_m = [] # mark
sum_power_vW = [] # vind Windstar
sum_power_vA = [] # vind Aeolos


# Loopar igenom varje timme och r knar ut summan av produktion och konsumtion
for i in range(8760):
    sum_power_tmvW.append(dfs['power_output_solceller_hustak'].iloc[i, dfs['
    power_output_solceller_hustak'].columns.get_loc('PV system output')]/1000 + dfs['
    power_output_solceller_mark'].iloc[i, dfs['power_output_solceller_mark'].columns.get_loc
    ('PV system output')]/1000 + energy_production_wind_hour[i]/1000 - dfs['elkonsumtion'].
    iloc[i, dfs['elkonsumtion'].columns.get_loc('Unnamed: 1')])

    sum_power_tmvA.append(dfs['power_output_solceller_hustak'].iloc[i, dfs['
    power_output_solceller_hustak'].columns.get_loc('PV system output')]/1000 + dfs['
    power_output_solceller_mark'].iloc[i, dfs['power_output_solceller_mark'].columns.get_loc
    ('PV system output')]/1000 + energy_production_wind_hour_A[i]/1000 - dfs['elkonsumtion'
    ].iloc[i, dfs['elkonsumtion'].columns.get_loc('Unnamed: 1')])

    sum_power_tm.append(dfs['power_output_solceller_hustak'].iloc[i, dfs['
    power_output_solceller_hustak'].columns.get_loc('PV system output')]/1000 + dfs['
    power_output_solceller_mark'].iloc[i, dfs['power_output_solceller_mark'].columns.get_loc
    ('PV system output')]/1000 - dfs['elkonsumtion'].iloc[i, dfs['elkonsumtion'].columns.
    get_loc('Unnamed: 1')])

    sum_power_mvW.append(dfs['power_output_solceller_mark'].iloc[i, dfs['
    power_output_solceller_mark'].columns.get_loc('PV system output')]/1000 +
    energy_production_wind_hour[i]/1000 - dfs['elkonsumtion'].iloc[i, dfs['elkonsumtion'].
    columns.get_loc('Unnamed: 1')])

    sum_power_mvA.append(dfs['power_output_solceller_mark'].iloc[i, dfs['
    power_output_solceller_mark'].columns.get_loc('PV system output')]/1000 +
    energy_production_wind_hour_A[i]/1000 - dfs['elkonsumtion'].iloc[i, dfs['elkonsumtion'].
    columns.get_loc('Unnamed: 1')])

    sum_power_tvW.append(dfs['power_output_solceller_hustak'].iloc[i, dfs['
    power_output_solceller_hustak'].columns.get_loc('PV system output')]/1000 +
    energy_production_wind_hour[i]/1000 - dfs['elkonsumtion'].iloc[i, dfs['elkonsumtion'].
    columns.get_loc('Unnamed: 1')])

    sum_power_tvA.append(dfs['power_output_solceller_hustak'].iloc[i, dfs['
    power_output_solceller_hustak'].columns.get_loc('PV system output')]/1000 +
    energy_production_wind_hour_A[i]/1000 - dfs['elkonsumtion'].iloc[i, dfs['elkonsumtion'].
    columns.get_loc('Unnamed: 1')])

    sum_power_t.append(dfs['power_output_solceller_hustak'].iloc[i, dfs['
    power_output_solceller_hustak'].columns.get_loc('PV system output')]/1000 - dfs['
    elkonsumtion'].iloc[i, dfs['elkonsumtion'].columns.get_loc('Unnamed: 1')])

    sum_power_m.append(dfs['power_output_solceller_mark'].iloc[i, dfs['
    power_output_solceller_mark'].columns.get_loc('PV system output')]/1000 - dfs['
    elkonsumtion'].iloc[i, dfs['elkonsumtion'].columns.get_loc('Unnamed: 1')])

    sum_power_vW.append(energy_production_wind_hour[i] / 1000 - dfs['elkonsumtion'].iloc[i,
    dfs['elkonsumtion'].columns.get_loc('Unnamed: 1')])

    sum_power_vA.append(energy_production_wind_hour_A[i] / 1000 - dfs['elkonsumtion'].iloc[i
    , dfs['elkonsumtion'].columns.get_loc('Unnamed: 1')])

sum_power_list = [sum_power_tmvW, sum_power_tmvA, sum_power_tm, sum_power_mvW, sum_power_mvA
    , sum_power_tvW, sum_power_tvA, sum_power_t, sum_power_m, sum_power_vW, sum_power_vA]
sum_power_list_names = ['sum_power_tmvW', 'sum_power_tmvA', 'sum_power_tm', 'sum_power_mvW',
     'sum_power_mvA', 'sum_power_tvW', 'sum_power_tvA', 'sum_power_t', 'sum_power_m', '
    sum_power_vW', 'sum_power_vA']
```

```
96
97
98  # ------------------------
99
100 results = {} # Tom dictionary
101 batteri_quant = [0, 1, 2, 3, 4, 50] # Potentiellt antal batterier som vill simuleras  ver
102 H_lagring_quant = [0, 1] # Potentiellt antal v tgasanl ggningar som vill simuleras  ver
103
104
105 # ----------- G r  ver  alla anl ggningsalternativ  f  r att ber kna var energin g  r
        -----
106 years = 5  # Antal  r  simuleras. Bra f  r att se utveckling  ver  tid
107 power_step = 0
108 for power in sum_power_list:
109     sum_power = IS.data_extender(power, years) # F  rl nger  l ngden av data med x  r
110     name = sum_power_list_names[power_step]
111     power_step += 1
112     for x in batteri_quant:
113         for y in H_lagring_quant:
114             batteri, H_lagring, buy_electricity, sell_electricity = IS.system(sum_power, x,
        y) # Skickar datan till simuleringsmodell och f  r tillbaka data
115             result_key = f"result_{name}_{x}_{y}"
116             results[result_key] = (batteri, H_lagring, buy_electricity, sell_electricity) #
        Sparar datan i results
117
118
119 summary = {} # Tom dictionary f  r att spara relevanta parametrar f  r presentation av
        resultat
120 for key, value in results.items(): # Itererar  ver  all data som har f tts av simulering
121     total_buy_electricity_sum = 0 # Summa av s  ld energi - Fungerar som m  tt  p
          verskott  av konsumtion eller underskott av lagringskapacitet
122     total_sell_electricity_sum = 0 # Summa av k  pt energi - Fungerar som m  tt  p
          verskott  av produktion eller underskott av lagringskapacitet
123
124     # Extraherar parametrar fr n simulering
125     batteri_values = value[0] if isinstance(value[0], list) else [value[0]]
126     H_lagring_values = value[1] if isinstance(value[1], list) else [value[1]]
127     buy_electricity_values = value[2] if isinstance(value[2], list) else [value[2]]
128     sell_electricity_values = value[3] if isinstance(value[3], list) else [value[3]]
129
130     mean_batteri = sum(batteri_values) / len(batteri_values) # Medelv rde av m ngd energi
        i batteri
131     mean_H_lagring = sum(H_lagring_values) / len(H_lagring_values) # Medelv rde av m ngd
        energi i v tgastank
132     total_buy_electricity_sum += sum(buy_electricity_values) # Summa av k  pt energi
133     total_sell_electricity_sum += sum(sell_electricity_values) # Summa av s  ld energi
134
135     value_list = [mean_batteri, mean_H_lagring, total_buy_electricity_sum,
        total_sell_electricity_sum]
136     summary_key = f"summary_{key}"  # Dynamically generate result key
137     summary[summary_key] = value_list
138
139 #---------------------- Matriser f  r lagring av presenterbar data
        ----------------------------
140
141 tmvW_matrix_batteri = [['tmvW',0,1,2,3,4,5], [0,0,0,0,0,0,0], [1,0,0,0,0,0,0]]
142 tmvW_matrix_H_lagring = [['tmvW',0,1,2,3,4,5], [0,0,0,0,0,0,0], [1,0,0,0,0,0,0]]
143 tmvW_matrix_buy = [['tmvW',0,1,2,3,4,5], [0,0,0,0,0,0,0], [1,0,0,0,0,0,0]]
144 tmvW_matrix_sell = [['tmvW',0,1,2,3,4,5], [0,0,0,0,0,0,0], [1,0,0,0,0,0,0]]
145
146 tmvA_matrix_batteri = [['tmvA',0,1,2,3,4,5], [0,0,0,0,0,0,0], [1,0,0,0,0,0,0]]
147 tmvA_matrix_H_lagring = [['tmvA',0,1,2,3,4,5], [0,0,0,0,0,0,0], [1,0,0,0,0,0,0]]
148 tmvA_matrix_buy = [['tmvA',0,1,2,3,4,5], [0,0,0,0,0,0,0], [1,0,0,0,0,0,0]]
149 tmvA_matrix_sell = [['tmvA',0,1,2,3,4,5], [0,0,0,0,0,0,0], [1,0,0,0,0,0,0]]
150
151 tm_matrix_batteri = [['tm',0,1,2,3,4,5], [0,0,0,0,0,0,0], [1,0,0,0,0,0,0]]
152 tm_matrix_H_lagring = [['tm',0,1,2,3,4,5], [0,0,0,0,0,0,0], [1,0,0,0,0,0,0]]
153 tm_matrix_buy = [['tm',0,1,2,3,4,5], [0,0,0,0,0,0,0], [1,0,0,0,0,0,0]]
```

```
tm_matrix_sell = [['tm',0,1,2,3,4,5], [0,0,0,0,0,0,0], [1,0,0,0,0,0,0]]

mvW_matrix_batteri = [['mvW',0,1,2,3,4,5], [0,0,0,0,0,0,0], [1,0,0,0,0,0,0]]
mvW_matrix_H_lagring = [['mvW',0,1,2,3,4,5], [0,0,0,0,0,0,0], [1,0,0,0,0,0,0]]
mvW_matrix_buy = [['mvW',0,1,2,3,4,5], [0,0,0,0,0,0,0], [1,0,0,0,0,0,0]]
mvW_matrix_sell = [['mvW',0,1,2,3,4,5], [0,0,0,0,0,0,0], [1,0,0,0,0,0,0]]

mvA_matrix_batteri = [['mvA',0,1,2,3,4,5], [0,0,0,0,0,0,0], [1,0,0,0,0,0,0]]
mvA_matrix_H_lagring = [['mvA',0,1,2,3,4,5], [0,0,0,0,0,0,0], [1,0,0,0,0,0,0]]
mvA_matrix_buy = [['mvA',0,1,2,3,4,5], [0,0,0,0,0,0,0], [1,0,0,0,0,0,0]]
mvA_matrix_sell = [['mvA',0,1,2,3,4,5], [0,0,0,0,0,0,0], [1,0,0,0,0,0,0]]

tvW_matrix_batteri = [['tvW',0,1,2,3,4,5], [0,0,0,0,0,0,0], [1,0,0,0,0,0,0]]
tvW_matrix_H_lagring = [['tvW',0,1,2,3,4,5], [0,0,0,0,0,0,0], [1,0,0,0,0,0,0]]
tvW_matrix_buy = [['tvW',0,1,2,3,4,5], [0,0,0,0,0,0,0], [1,0,0,0,0,0,0]]
tvW_matrix_sell = [['tvW',0,1,2,3,4,5], [0,0,0,0,0,0,0], [1,0,0,0,0,0,0]]

tvA_matrix_batteri = [['tvA',0,1,2,3,4,5], [0,0,0,0,0,0,0], [1,0,0,0,0,0,0]]
tvA_matrix_H_lagring = [['tvA',0,1,2,3,4,5], [0,0,0,0,0,0,0], [1,0,0,0,0,0,0]]
tvA_matrix_buy = [['tvA',0,1,2,3,4,5], [0,0,0,0,0,0,0], [1,0,0,0,0,0,0]]
tvA_matrix_sell = [['tvA',0,1,2,3,4,5], [0,0,0,0,0,0,0], [1,0,0,0,0,0,0]]

t_matrix_batteri = [['t',0,1,2,3,4,5], [0,0,0,0,0,0,0], [1,0,0,0,0,0,0]]
t_matrix_H_lagring = [['t',0,1,2,3,4,5], [0,0,0,0,0,0,0], [1,0,0,0,0,0,0]]
t_matrix_buy = [['t',0,1,2,3,4,5], [0,0,0,0,0,0,0], [1,0,0,0,0,0,0]]
t_matrix_sell = [['t',0,1,2,3,4,5], [0,0,0,0,0,0,0], [1,0,0,0,0,0,0]]

m_matrix_batteri = [['m',0,1,2,3,4,5], [0,0,0,0,0,0,0], [1,0,0,0,0,0,0]]
m_matrix_H_lagring = [['m',0,1,2,3,4,5], [0,0,0,0,0,0,0], [1,0,0,0,0,0,0]]
m_matrix_buy = [['m',0,1,2,3,4,5], [0,0,0,0,0,0,0], [1,0,0,0,0,0,0]]
m_matrix_sell = [['m',0,1,2,3,4,5], [0,0,0,0,0,0,0], [1,0,0,0,0,0,0]]

vW_matrix_batteri = [['vW',0,1,2,3,4,5], [0,0,0,0,0,0,0], [1,0,0,0,0,0,0]]
vW_matrix_H_lagring = [['vW',0,1,2,3,4,5], [0,0,0,0,0,0,0], [1,0,0,0,0,0,0]]
vW_matrix_buy = [['vW',0,1,2,3,4,5], [0,0,0,0,0,0,0], [1,0,0,0,0,0,0]]
vW_matrix_sell = [['vW',0,1,2,3,4,5], [0,0,0,0,0,0,0], [1,0,0,0,0,0,0]]

vA_matrix_batteri = [['vA',0,1,2,3,4,5], [0,0,0,0,0,0,0], [1,0,0,0,0,0,0]]
vA_matrix_H_lagring = [['vA',0,1,2,3,4,5], [0,0,0,0,0,0,0], [1,0,0,0,0,0,0]]
vA_matrix_buy = [['vA',0,1,2,3,4,5], [0,0,0,0,0,0,0], [1,0,0,0,0,0,0]]
vA_matrix_sell = [['vA',0,1,2,3,4,5], [0,0,0,0,0,0,0], [1,0,0,0,0,0,0]]

# Extraherar och placerar data  p   r tt plats i  r tt matris
for key, value in summary.items():
    input_string = key
    split_string = input_string.split('_')

    if split_string[4] == 'tmvW':
        tmvW_matrix_batteri[int(split_string[6]) + 1][int(split_string[5]) + 1] = value[0]
        tmvW_matrix_H_lagring[int(split_string[6]) + 1][int(split_string[5]) + 1] = value[1]
        tmvW_matrix_buy[int(split_string[6]) + 1][int(split_string[5]) + 1] = value[2]
        tmvW_matrix_sell[int(split_string[6]) + 1][int(split_string[5]) + 1] = value[3]
    elif split_string[4] == 'tmvA':
        tmvA_matrix_batteri[int(split_string[6]) + 1][int(split_string[5]) + 1] = value[0]
        tmvA_matrix_H_lagring[int(split_string[6]) + 1][int(split_string[5]) + 1] = value[1]
        tmvA_matrix_buy[int(split_string[6]) + 1][int(split_string[5]) + 1] = value[2]
        tmvA_matrix_sell[int(split_string[6]) + 1][int(split_string[5]) + 1] = value[3]
    elif split_string[4] == 'tm':
        tm_matrix_batteri[int(split_string[6]) + 1][int(split_string[5]) + 1] = value[0]
        tm_matrix_H_lagring[int(split_string[6]) + 1][int(split_string[5]) + 1] = value[1]
        tm_matrix_buy[int(split_string[6]) + 1][int(split_string[5]) + 1] = value[2]
        tm_matrix_sell[int(split_string[6]) + 1][int(split_string[5]) + 1] = value[3]
    elif split_string[4] == 'mvW':
        mvW_matrix_batteri[int(split_string[6]) + 1][int(split_string[5]) + 1] = value[0]
        mvW_matrix_H_lagring[int(split_string[6]) + 1][int(split_string[5]) + 1] = value[1]
        mvW_matrix_buy[int(split_string[6]) + 1][int(split_string[5]) + 1] = value[2]
        mvW_matrix_sell[int(split_string[6]) + 1][int(split_string[5]) + 1] = value[3]
    elif split_string[4] == 'mvA':
```

```
        mvA_matrix_batteri[int(split_string[6]) + 1][int(split_string[5]) + 1] = value[0]
        mvA_matrix_H_lagring[int(split_string[6]) + 1][int(split_string[5]) + 1] = value[1]
        mvA_matrix_buy[int(split_string[6]) + 1][int(split_string[5]) + 1] = value[2]
        mvA_matrix_sell[int(split_string[6]) + 1][int(split_string[5]) + 1] = value[3]
    elif split_string[4] == 'tvW':
        tvW_matrix_batteri[int(split_string[6]) + 1][int(split_string[5]) + 1] = value[0]
        tvW_matrix_H_lagring[int(split_string[6]) + 1][int(split_string[5]) + 1] = value[1]
        tvW_matrix_buy[int(split_string[6]) + 1][int(split_string[5]) + 1] = value[2]
        tvW_matrix_sell[int(split_string[6]) + 1][int(split_string[5]) + 1] = value[3]
    elif split_string[4] == 'tvA':
        tvA_matrix_batteri[int(split_string[6]) + 1][int(split_string[5]) + 1] = value[0]
        tvA_matrix_H_lagring[int(split_string[6]) + 1][int(split_string[5]) + 1] = value[1]
        tvA_matrix_buy[int(split_string[6]) + 1][int(split_string[5]) + 1] = value[2]
        tvA_matrix_sell[int(split_string[6]) + 1][int(split_string[5]) + 1] = value[3]
    elif split_string[4] == 't':
        t_matrix_batteri[int(split_string[6]) + 1][int(split_string[5]) + 1] = value[0]
        t_matrix_H_lagring[int(split_string[6]) + 1][int(split_string[5]) + 1] = value[1]
        t_matrix_buy[int(split_string[6]) + 1][int(split_string[5]) + 1] = value[2]
        t_matrix_sell[int(split_string[6]) + 1][int(split_string[5]) + 1] = value[3]
    elif split_string[4] == 'm':
        m_matrix_batteri[int(split_string[6]) + 1][int(split_string[5]) + 1] = value[0]
        m_matrix_H_lagring[int(split_string[6]) + 1][int(split_string[5]) + 1] = value[1]
        m_matrix_buy[int(split_string[6]) + 1][int(split_string[5]) + 1] = value[2]
        m_matrix_sell[int(split_string[6]) + 1][int(split_string[5]) + 1] = value[3]
    elif split_string[4] == 'vW':
        vW_matrix_batteri[int(split_string[6]) + 1][int(split_string[5]) + 1] = value[0]
        vW_matrix_H_lagring[int(split_string[6]) + 1][int(split_string[5]) + 1] = value[1]
        vW_matrix_buy[int(split_string[6]) + 1][int(split_string[5]) + 1] = value[2]
        vW_matrix_sell[int(split_string[6]) + 1][int(split_string[5]) + 1] = value[3]
    elif split_string[4] == 'vA':
        vA_matrix_batteri[int(split_string[6]) + 1][int(split_string[5]) + 1] = value[0]
        vA_matrix_H_lagring[int(split_string[6]) + 1][int(split_string[5]) + 1] = value[1]
        vA_matrix_buy[int(split_string[6]) + 1][int(split_string[5]) + 1] = value[2]
        vA_matrix_sell[int(split_string[6]) + 1][int(split_string[5]) + 1] = value[3]

# S tter ihop alla matriser
matrix_excel_batteri = [tmvW_matrix_batteri, tmvA_matrix_batteri, tm_matrix_batteri,
    mvW_matrix_batteri, mvA_matrix_batteri, tvW_matrix_batteri, tvA_matrix_batteri,
    t_matrix_batteri, m_matrix_batteri, vW_matrix_batteri, vA_matrix_batteri]
matrix_excel_H_lagring = [tmvW_matrix_H_lagring, tmvA_matrix_H_lagring, tm_matrix_H_lagring,
     mvW_matrix_H_lagring, mvA_matrix_H_lagring, tvW_matrix_H_lagring, tvA_matrix_H_lagring,
     t_matrix_H_lagring, m_matrix_H_lagring, vW_matrix_H_lagring, vA_matrix_H_lagring]
matrix_excel_buy = [tmvW_matrix_buy, tmvA_matrix_buy, tm_matrix_buy, mvW_matrix_buy,
    mvA_matrix_buy, tvW_matrix_buy, tvA_matrix_buy, t_matrix_buy, m_matrix_buy,
    vW_matrix_buy, vA_matrix_buy]
matrix_excel_sell = [tmvW_matrix_sell, tmvA_matrix_sell, tm_matrix_sell, mvW_matrix_sell,
    mvA_matrix_sell, tvW_matrix_sell, tvA_matrix_sell, t_matrix_sell, m_matrix_sell,
    vW_matrix_sell, vA_matrix_sell]

# F rbereder matriser f r skrivning i excel
excel_batteri = pd.DataFrame(matrix_excel_batteri)
excel_H_lagring = pd.DataFrame(matrix_excel_H_lagring)
excel_buy = pd.DataFrame(matrix_excel_buy)
excel_sell = pd.DataFrame(matrix_excel_sell)

# Skriver matriser till Excel
with pd.ExcelWriter('test123.xlsx') as writer: # D per filen som skrivs i Excel
    excel_batteri.to_excel(writer, sheet_name='batteri', index=False, header=False)
    excel_H_lagring.to_excel(writer, sheet_name='H_lagring', index=False, header=False)
    excel_buy.to_excel(writer, sheet_name='buy_electricity', index=False, header=False)
    excel_sell.to_excel(writer, sheet_name='sell_electricity', index=False, header=False)

#-----------------------------------------------------------------------------------------



# V ljer vilka anl ggningsalternav som vill plottas
# 'resultat_sum_power_{typ av anl ggning}_{antal batterier}_{antal v tgasanl ggningar}'
```

```
intressanta = ['result_sum_power_tmvA_50_0']

for plotta in intressanta:

    batteri = results[plotta][0]
    new_list = [x + 30*0.2 for x in batteri]
    H_lagring = results[plotta][1]
    buy_electricity = results[plotta][2]
    sell_electricity = results[plotta][3]

    plt.subplot(4, 1, 1)
    plt.plot(range(len(sum_power) + 1), new_list, color="blue")
    plt.ylim(0, 30)
    plt.title('Laddning Batteri')
    plt.xlabel('Tid (h)')
    plt.ylabel('Energi (kWh)')

    plt.subplot(4, 1, 2)
    plt.plot(range(len(sum_power) + 1), H_lagring, color="green")
    plt.title('Laddning V tgas')
    plt.xlabel('Tid (h)')
    plt.ylabel('Energi (kWh)')

    plt.subplot(4, 1, 3)
    plt.plot(range(len(sum_power) + 1), buy_electricity, color="purple")
    plt.title('K pt energi')
    plt.xlabel('Tid (h)')
    plt.ylabel('Energi (kWh)')

    plt.subplot(4, 1, 4)
    plt.plot(range(len(sum_power) + 1), sell_electricity, color="orange")

    plt.title('S ld energi')
    plt.xlabel('Tid (h)')
    plt.ylabel('Energi (kWh)')

    plt.tight_layout()
    plt.show()
```

Kod för allokering av energi

```
import SpecsMalsta as Specs
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import openpyxl
from datetime import datetime

def data_extender(data_list, years):
    data_extended = []
    data = data_list
    data_extended.extend(data)
    for i in range(years-1):
        data_extended.extend(data)
    return data_extended

def system(sum_power,x, y):
    batteri_spec, H_lagring_spec = Specs.specifikationer(x, y)

    batteri = np.zeros(len(sum_power) + 1, dtype=float).tolist()
    batteri[0] = batteri_spec[0] * 1  # fulladdat batteri, kan multipliceras till fler
    batterier

    H_lagring = np.zeros(len(sum_power) + 1, dtype=float).tolist()
    # H_lagring[0] = H_lagring_spec[0]*1 # full tank v tgas, kan multipliceras f r fler
    tankar
    H_lagring[0] = 0  # tom tank i b rjan
```

```
25
26      buy_electricity = np.zeros(len(sum_power) + 1, dtype=float).tolist()
27      sell_electricity = np.zeros(len(sum_power) + 1, dtype=float).tolist()
28
29      for i in range(len(sum_power)):  # G    ver  alla punkter i sum_power
30          # ------------ Om ingen energi stoppas in eller tas ur v tgaslagring eller batteri
        s   beh ller de sin laddning
31          batteri[i + 1] = batteri[i]
32          H_lagring[i + 1] = H_lagring[i]
33
34          # ------------ Block f r  n r energin producerad  r  mindre  n  den som konsumeras
35          if sum_power[i] < 0 and batteri[i] == 0 and H_lagring == 0:  # Om produktion  r
        mindre  n  konsumtion och batteriet samt v tgaslagring  r  tomt
36              buy_electricity[i] = -sum_power[i]  # K per el, - f r att vi vill ha positivt
        v rde  p  W vi k per
37          elif sum_power[i] < 0 and batteri[i] == 0:  # Om produktion  r  mindre  n
        konsumtion och batteriet  r  slut men det finns laddning kvar i v tgaslagringen
38              H_lagring_step = H_lagring[i] + sum_power[i]  # Vill se hur mycket energi vi kan
         ta ut, kikar med ett steg fram t. + pga negativ sum_power
39              if H_lagring_step < 0:  # Om vi tar ut s  mycket energi vi vill ha kommer det
        resultera i en tom v tgastank
40                  H_lagring[i + 1] = 0  # Tom v tgastank
41                  buy_electricity[i] = -H_lagring_step  # Den el vi m ste k pa. + pga
        negativ sum_power
42              else:
43                  H_lagring[i + 1] = H_lagring[i] + sum_power[i]  # Om inte, ta ut energi som
        vanligt fr n v tgaslagret. + pga negativ sum_power
44          elif sum_power[i] < 0:  # Om produktion  r  mindre  n  konsumtion men det finns
        energi i batteriet och v tgaslagret
45              batteri_step = batteri[i] + sum_power[i]  # Se ett steg fram t om laddningen
        kommer ta slut i batteriet
46              H_lagring_step = batteri[i] + H_lagring[i] + sum_power[i]  # Se ett steg fram t
         om laddning f r b de batteri och v tgas kommer att ta slut
47              if batteri_step == 0:  # Om batteriet tar exakt slut beh vs ingen mer energi
        tas fr n annan plats
48                  batteri[i + 1] = 0  # s tt laddning till noll
49              elif batteri_step < 0:  # Om batteriet kommer att ta mer  n  slut
50                  batteri[i + 1] = 0  # s tt laddning till noll
51                  if H_lagring_step < 0:  # Om v tgaslagret  ven  kommer att ta slut
52                      H_lagring[i + 1] = 0  # S tt laddning till noll
53                      buy_electricity[i] = -H_lagring_step  # k p resterande energi. - P
        grund av H_lagring_step har negativt v rde
54                  else:
55                      H_lagring[i + 1] = H_lagring[i] + sum_power[i]  # Om v tgaslagret inte
        kommer att ta slut, eller precis ta slut, ber kna hur mycket som  r  kvar
56              else:
57                  batteri[i + 1] = batteri[i] + sum_power[i]  # Om det kommer att finnas
        laddning kvar i batteriet.+ f r att sum_power  r  negativt
58
59          if sum_power[i] > 0 and batteri[i] == batteri_spec[0] and H_lagring[i] ==
        H_lagring_spec[0]:  # Om produktion  r  st rre  n  konsumtion och b de batteri  r
        v tgaslagring  r  fulla
60              sell_electricity[i] = sum_power[i]  # S lj elektriciteten
61          elif sum_power[i] > 0 and batteri[i] == batteri_spec[0]:  # Om produktion  r
        st rre  n  konsumtion och endast batteriet  r  fulladdat
62              H_lagring_step = H_lagring[i] + Specs.lagring_H(sum_power[i])  # Kolla ett steg
        fram t f r att se om energin  verstiger  maxladdning f r v tgastank. Skickar
        sum_power till lagring_h f r ber kning av f rluster
63              if H_lagring_step > H_lagring_spec[0] or H_lagring_step == H_lagring_spec[0]:  #
         Om energin kommer  verfylla  tanken, toppa upp tanken och s lj resterande energi
64                  H_lagring[i + 1] = H_lagring_spec[0]  # fyller tanken
65                  sell_electricity[i] = H_lagring[i] + sum_power[i] - H_lagring_spec[0]  #
        s ljer resterande energi
66              else:
67                  H_lagring[i + 1] = H_lagring[i] + Specs.lagring_H(sum_power[i])  # Annars,
        fyll tank med energi
68          elif sum_power[i] > 0:  # Om produktion  r  st rre  n  konsumtion men batteri och
        v tgas  r  inte fulladdade
```

```
            batteri_step = batteri[i] + Specs.batteri(sum_power[i])  # Se ett steg fram t
            if batteri_step > batteri_spec[0]:  # Om batteriet blir fulladdat
                batteri[i + 1] = batteri_spec[0]  # ladda batteriet
                H_lagring_step = H_lagring[i] + Specs.lagring_H(sum_power[i])
                if H_lagring_step > H_lagring_spec[0]:
                    H_lagring[i + 1] = H_lagring_spec[0]
                    sell_electricity[i] = H_lagring[i] + sum_power[i] - H_lagring_spec[0]  #
     s ljer resterande energi
                else:
                    H_lagring[i + 1] = H_lagring[i] + batteri_step - batteri_spec[0]  #
    Resten g r in i v telagring
            elif batteri_step < batteri_spec[0] or batteri_step == batteri_spec[0]:  # Om
    batteriet inte blir fullt eller om det blir exakt full
                batteri[i + 1] = batteri[i] + Specs.batteri(sum_power[i])  # Ladda batteriet

        if sum_power[i] == 0:  # Perfekt, g r inget
            pass

    return batteri, H_lagring, buy_electricity, sell_electricity
```

Kod för specifikationer för batteri och vätgaslagring

```


def specifikationer(x, y):
    # --------------- Specifikationer --------------

    # --------------- Batteri ----------------------
    maxkapacitet_batteri = 30*0.6  # [kWh] *0.6 f r att batteriet vill helst inte laddas ur
     l gre  n  till 20 % och inte laddar mer  n  80%.
    verkningsgrad_batteri = 0.9  # [fraktion]
    startladdning_batteri = 1  # [fraktion]
    #c_tal_batteri =  # [kWh] max urladdningshastighet
    kostnad_batteri = 25000 # [kr] f r litiumbatteri

    batteri = [maxkapacitet_batteri, verkningsgrad_batteri, startladdning_batteri,
    kostnad_batteri]
    batteri_mult = [x * element for element in batteri]
    #litium_batteri =

    # --------------- V tgaslagring ----------------
    maxkapacitet_H = 8000  # [kWh]
    verkningsgrad_H = 0.4  # [fraktion]
    startladdning_H = 1  # [fraktion]
    kostnad_H = 2000000 # [kr]

    H_lagring = [maxkapacitet_H, verkningsgrad_H, startladdning_H, kostnad_H]
    H_lagring_mult = [y * element for element in H_lagring]

    return batteri_mult, H_lagring_mult

    # --------------- Solpaneler -------------------
    # kostnad_solpanel = 125000 # [kr]


def batteri(inn):
    loss = 0.9
    #stat_loss = (30*3)/100
    #ut = inn*loss-stat_loss
    ut = inn*loss
    return ut

def lagring_H(inn):
    ut = inn*0.4
    return(ut)
```