ev_adoption_final

December 7, 2023

1 Imports

[ ]:
```
# imports
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import odeint, solve_ivp, solve_bvp
from scipy.optimize import minimize
import pandas as pd
import scipy.linalg as la

# setup the print and display options to make displaying easier
np.set_printoptions(precision=5, suppress=True)
pd.set_option("display.precision", 15)
np.set_printoptions(formatter={'float': lambda x: "{0:0.7f}".format(x)})
pd.set_option('display.float_format', lambda x: "{0:0.7f}".format(x))
pd.set_option('display.max_columns', None)
```

2 Base Model

2.0.1 Create the base model and plot real data to show trends

[ ]:
```
# build model
def model(t, cars, ch, he, ec, hc, eh, ce, c, h, e):
    return np.array([(hc-ch)*cars[0]*cars[1] + (ec-ce)*cars[0]*cars[2] + c,
                     (ch-hc)*cars[0]*cars[1] + (eh-he)*cars[1]*cars[2] + h,
                     (ce-ec)*cars[0]*cars[2] + (he-eh)*cars[1]*cars[2] + e])

# initialize subplots
plt.figure(figsize=(10, 3))

# read in actual data
rel_sales_df = pd.read_csv('rel_sales_df.csv')

#### CONSERVATIVE MODEL ####
# initial conditions
C = 0.9792
H = 0.02
```

```python
E = 0.0008

# time points
t0 = 0
tf = 50

# constants
ch = 0.15 # rate from combustion to hybrid
he = 0.1 # rate from hybrid to electric
ec = 0.01 # rate from electric to combustion
hc = 0.1 # rate from hybrid to combustion
eh = 0.01 # rate from electric to hybrid
ce = 0.1 # rate from combustion to electric
c = 0.0 # rate of new combustion
h = 0.0 # rate of new hybrid
e = 0.0 # rate of new electric

# solve ODE
sol = solve_ivp(model, [t0, tf], [C, H, E], args=(ch, he, ec, hc, eh, ce, c, h,␣
↪e), t_eval=np.linspace(t0, tf, 1000))


# plot actual data
plt.subplot(121)
dot_size = 10
plt.scatter(rel_sales_df['Year'], rel_sales_df['Hybrid_ratio'], color="r",␣
↪s=dot_size)
plt.scatter(rel_sales_df['Year'], rel_sales_df['Combustion_ratio'], color="k",␣
↪s=dot_size)
plt.scatter(rel_sales_df['Year'], rel_sales_df['Electric_ratio'], color="b",␣
↪s=dot_size)
plt.plot(sol.t + 2010, sol.y[0], label="Combustion", c="k")
plt.plot(sol.t + 2010, sol.y[1], label="Hybrid", c="r")
plt.plot(sol.t + 2010, sol.y[2], label="Electric", c="b")
plt.title("Conservative Model")
plt.xlabel("Year")
plt.ylabel("Proportion of Car Purchases")
plt.legend()


#### PROGRESSIVE MODEL ####
# constants for progressive model
ch = 0.19 # rate from combustion to hybrid
he = 0.2 # rate from hybrid to electric
ec = 0.01 # rate from electric to combustion
hc = 0.1 # rate from hybrid to combustion
eh = 0.18 # rate from electric to hybrid
```

```
ce = 0.1 # rate from combustion to electric
c = 0.0 # rate of new combustion
h = 0.0 # rate of new hybrid
e = 0.0007 # rate of new electric

# solve ode
sol2 = solve_ivp(model, [t0, tf], [C, H, E], args=(ch, he, ec, hc, eh, ce, c,␣
 ↪h, e), t_eval=np.linspace(t0, tf, 1000))




# plot
plt.subplot(122)
plt.scatter(rel_sales_df['Year'], rel_sales_df['Hybrid_ratio'], color="r",␣
 ↪s=dot_size)
plt.scatter(rel_sales_df['Year'], rel_sales_df['Combustion_ratio'], color="k",␣
 ↪s=dot_size)
plt.scatter(rel_sales_df['Year'], rel_sales_df['Electric_ratio'], color="b",␣
 ↪s=dot_size)
plt.plot(sol2.t + 2010, sol2.y[0], label="Combustion", c="k")
plt.plot(sol2.t + 2010, sol2.y[1], label="Hybrid", c="r")
plt.plot(sol2.t + 2010, sol2.y[2], label="Electric", c="b")
plt.title("Progressive Model")
plt.xlabel("Year")
plt.ylabel("Proportion of Car Purchases")
plt.legend()

# save figure
plt.savefig('original_model.png', dpi=200, bbox_inches='tight')
plt.show()
```

3 California Regulation

3.0.1 Showing real data trends in the california car sale market

[ ]:
```
# real data
eh = [0,.075,.12,.18,.254, .36, 1]
c = [1, 1-.075, 1-.12, 1-.18,1-.254, 1-.36, 0]
t = [2010,2018,2021,2022,2023,2026,2035]

# plot
plt.plot(t,eh, label = "Electric/Hybrid")
plt.scatter(t,eh)
plt.plot(t,c, label = "Combustion")
plt.scatter(t,c)
plt.xlabel("Years")
```

```
plt.ylabel("Proportion of Car Purchases")
plt.legend()
plt.show()
```

3.0.2 Create model and fit it to California data

[ ]:
```
# build model
def model(t, cars, ec, ce, c, e):
    return np.array([(ec)*cars[0]*cars[1] + c*cars[0],
                     (ce)*cars[0]*cars[1] + e*cars[1]])

# constants
ec = -0.4 # rate from electric to combustion
ce = .3 # rate from combustion to electric
c = -0.023# rate of new combustion
e = 0.1 # rate of new electric

# initial conditions
C = 0.9792
E = 0.0008

# time points
t0 = 0
tf = 25

# solve ODE
sol = solve_ivp(model, [t0, tf], [C,E], args=(ec, ce, c,e), t_eval=np.
 ↪linspace(t0, tf, 1000))

# plot
plt.plot(sol.t, sol.y[0], label="Combustion")
plt.plot(sol.t, sol.y[1], label="Electric/Hybrid")
plt.xlabel("Years Since 2010")
plt.ylabel("Proportion of Car Purchases")
plt.legend()
plt.show()

# print values at t=10
print(len(sol.y[0]))
year = 2020
time = int((year - 2010) * 1000 / tf)
print(time)
print("Combustion: ", sol.y[0][time])
print("Electric: ", sol.y[1][time])
```

4 Disaster Effect

4.0.1 Showing historical car sales and marking recessions

[ ]:
```
# read car sales data
car_sales = pd.read_csv('TOTALSA.csv')
car_sales['DATE'] = pd.to_datetime(car_sales['DATE']) # convert to datetime␣
↪format
# read recession dates
recession = pd.read_csv('recession_dates.csv')[['Peaks', 'Troughs']]
# convert to datetime format
recession['Peaks'] = pd.to_datetime(recession['Peaks'])
recession['Troughs'] = pd.to_datetime(recession['Troughs'])

# create figure and plot car sales
fig, ax = plt.subplots(1,1,figsize=(12,3))
car_sales.plot(x='DATE', y='TOTALSA', ax=ax, c='#1f77b4')

# gray out the recession areas
for i in range(recession.shape[0]):
    ax.axvspan(recession.loc[i, 'Peaks'], recession.loc[i, 'Troughs'], alpha=0.
↪3, color='gray')

# label plot
ax.set_title('Total Vehicle Sales')
ax.set_ylabel('Millions of Units')
ax.legend(['Sales', 'Recessions'])
ax.set_xlabel('Year')
plt.tight_layout()
plt.savefig('Total Vehicle Sales')
plt.show()
```

5 Charging Ports

5.0.1 Setup the Data

[ ]:
```
# manually create a dataframe with the data from the US Department of Energy
# regarding number of charging ports and station locations
charging_df = pd.DataFrame({
    'Year': [2007, 2008, 2009, 2010, 2011, 2012, 2013, 2014, 2015, 2016, 2017,␣
↪2018, 2019, 2020, 2021, 2022, 2023],
    'Charging Ports': [417, 564, 771, 1256, 5248, 10726, 16619, 22470, 26532,␣
↪33165, 45789, 56842, 73838, 96190, 114451, 136513, 163753],
    'Station Locations': [139, 196, 259, 407, 2109, 5444, 6938, 9207, 10710,␣
↪13150, 16170, 19893, 23282, 28602, 46407, 53764, 60649]
})
```

```
# save the charging data to a csv in the data folder
charging_df.to_csv("data/charging.csv", index=False)
```

[ ]:
```
# load the charging data
charging_df = pd.read_csv("data/charging.csv")

# load the sales data
sales_df = pd.read_csv("data/vehicle_sales_2000_2023.csv")

# combine the charging stations data to the vehicle sales data based
# on the year column
sales_df = sales_df.merge(charging_df, on='Year', how='left')

# since charging ports did not exist in the early years of sales, fill these␣
↪with zeros
sales_df['Charging Ports'] = sales_df['Charging Ports'].fillna(0).astype(float)␣
↪/ 1000
sales_df['Station Locations'] = sales_df['Station Locations'].fillna(0).
↪astype(float) / 1000

# compute the station locations ratio
sales_df['Station Locations Ratio'] = sales_df['Station Locations'] /␣
↪sales_df['Total']

# only consider years where there were electric vehicle sales and charging␣
↪stations
rel_sales_df = sales_df.loc[sales_df['Year'] >= 2011, :].copy()
```

5.0.2 Find a Best Fit Exponential for Charging Stations

[ ]:
```
# extract the years as a numpy array
ts = charging_df['Year'].values.astype(int) - 2000

# create a constant column for the OLS
const_col = np.ones(len(ts))

# setup the A and the the y and run OLS on this
A = np.vstack([const_col, ts]).T
sol = la.lstsq(A, np.log(charging_df['Charging Ports']/1000))[0]

# extract the coefficients for the best fit exponential
# exp(a*x + b)
b, a = sol

# display the charging location data and the best-fit exponential
plt.scatter(charging_df['Year'], charging_df['Station Locations'] / 1000,␣
↪label="Charging Stations")
```

```python
plt.plot(ts + 2000, np.exp(a * ts + b), color='orange', label="Exponential")
plt.xlabel("Year")
plt.ylabel("Number of Charging Stations (thousands)")
plt.title("Number of US Charging Stations over Time")
plt.legend()
plt.show()
```

5.0.3 Run the Charging Station ODE to Make Future Predictions

[ ]:
```python
####################################
# SOLVING THE CHARGING STATION IVP #
####################################

# define the altered charing station base model
def ces(t, y, K_ch, K_ce, K_he, alpha, beta, C_s):
    #print(y[0], y[1], y[2], y[3], C_s, y[3] / C_s)
    return (
        -K_ch * y[0] * y[1] - K_ce * y[0] * y[2],
        K_ch * y[0] * y[1] - K_he * y[1] * y[2],
        (K_ce * y[0] * y[2] + K_he * y[1] * y[2] + alpha * y[3]) * (1-y[2]),
        beta * y[3] * (1 - y[3] / C_s)
    )

# go from 2011 to 2050
t_span = (0, 50-11)
ts = np.linspace(*t_span, 500)

# create a clean version of the years as integers
ts_int = ts.astype(int)
ts_yr_inds = np.where(ts_int[1:] != ts_int[:-1])[0] + 1
ts_yr_inds = np.concatenate([[0], ts_yr_inds])

# find which years are using historical data and which years involve future data
mn_yr, mx_yr = rel_sales_df['Year'].min(), rel_sales_df['Year'].max()
ts_2011 = (ts[ts_yr_inds] + 2011).astype(int)
inner_years = (ts_2011 >= mn_yr) & (ts_2011 <= mx_yr)
inner_ts_yr_inds = ts_yr_inds[inner_years]

# setup y0 based on the actual ratios at the start of 2011
y0 = np.array([
    rel_sales_df['Combustion_ratio'].values[0],
    rel_sales_df['Hybrid_ratio'].values[0],
    rel_sales_df['Electric_ratio'].values[0],
    rel_sales_df['Station Locations Ratio'].values[0]
])

# set constants to make the trends match the data from 2011 through 2022
```

```
K_ch = 0.10
K_ce = 0.11
K_he = 0.01
alpha = 1.7
beta = 0.28
C_s = 0.15

# solve the modified ivp
solution = solve_ivp(ces, t_span, y0, t_eval=ts, args=(K_ch, K_ce, K_he, alpha,
 ↪beta, C_s))

# extract the true values
C = rel_sales_df['Combustion_ratio'].values
H = rel_sales_df['Hybrid_ratio'].values
E = rel_sales_df['Electric_ratio'].values
S = rel_sales_df['Station Locations Ratio'].values

# find the predicted values in the future
model_yrs = ts + 2011
model_C = solution.y[0]
model_H = solution.y[1]
model_E = solution.y[2]
model_S = solution.y[3]

# define a color map that maps the default colors to
# our specified uniform colors
color_map = {
    'blue': 'black', # combustion
    'orange': 'red', # hybrid
    'green': 'blue', # electric
    'red': 'green' # charging stations
}

##################################################
# PLOTTING THE SOLUTION'S FIT TO THE ACTUAL DATA #
##################################################

# since the combustion ratios are a lot higher than the other ratios, plot the
# combustion fit separate from the other fits
plt.subplot(1,2,1)
plt.scatter(rel_sales_df['Year'], rel_sales_df['Combustion_ratio'],
 ↪color=color_map['blue'], label='Combustion Actual')
plt.plot(rel_sales_df['Year'], model_C[inner_ts_yr_inds],
 ↪color=color_map['blue'], label='Combustion Model')
plt.vlines(2022, rel_sales_df['Combustion_ratio'].min(), 1,
 ↪linestyles='dashed', color='black')
plt.legend(loc='lower left')
```

```
plt.ylabel("Percentage of Total Sales")
plt.xlabel("Year")

# plot the hybrid, electric and station location fits
plt.subplot(1,2,2)
plt.scatter(rel_sales_df['Year'], rel_sales_df['Hybrid_ratio'],␣
 ↪color=color_map['orange'], label='Hybrid Actual')
plt.plot(rel_sales_df['Year'], model_H[inner_ts_yr_inds],␣
 ↪color=color_map['orange'], label='Hybrid Fit')
plt.scatter(rel_sales_df['Year'], rel_sales_df['Electric_ratio'],␣
 ↪color=color_map['green'], label='Electric Actual')
plt.plot(rel_sales_df['Year'], model_E[inner_ts_yr_inds],␣
 ↪color=color_map['green'], label='Electric Fit')
plt.scatter(rel_sales_df['Year'], rel_sales_df['Station Locations Ratio'],␣
 ↪color=color_map['red'], label='Charging Stations Actual')
plt.plot(rel_sales_df['Year'], model_S[inner_ts_yr_inds],␣
 ↪color=color_map['red'], label='Charging Stations Fit')
plt.vlines(2022, 0, rel_sales_df['Hybrid_ratio'].max(), linestyles='dashed',␣
 ↪color='black')
plt.legend()
plt.xlabel("Year")
plt.ylabel("Percentage of Total Sales")

# show the fitting plot
plt.suptitle("Fitting Model Parameters to the Data")
plt.gcf().set_size_inches(12, 4)
plt.show()

###########################
# PLOT FUTURE PREDICTIONS #
###########################

dot_size = 10

# plot the actual data from 2011 to 2022
plt.scatter(rel_sales_df['Year'], rel_sales_df['Combustion_ratio'],␣
 ↪color=color_map['blue'], s=dot_size)
plt.scatter(rel_sales_df['Year'], rel_sales_df['Hybrid_ratio'],␣
 ↪color=color_map['orange'], s=dot_size)
plt.scatter(rel_sales_df['Year'], rel_sales_df['Electric_ratio'],␣
 ↪color=color_map['green'], s=dot_size)
plt.scatter(rel_sales_df['Year'], rel_sales_df['Station Locations Ratio'],␣
 ↪color=color_map['red'], s=dot_size)

# plot the future predictions
plt.plot(model_yrs, model_C, label='Combustion Model', color=color_map['blue'])
```

```
plt.plot(model_yrs, model_H, label='Hybrid Model', color=color_map['orange'])
plt.plot(model_yrs, model_E, label='Electric Model', color=color_map['green'])
plt.plot(model_yrs, model_S, label='Charging Stations Model',␣
 ↪color=color_map['red'])

# plot a vertical line to demonstrate where the future predictions start
plt.vlines(2022, 0, 1, linestyles='dashed', color='black', label='2022')

# set other model parameters
plt.xlabel("Year")
plt.ylabel("Percentage of Total Sales")
plt.legend()
plt.title("Model Predictions from 2022 to 2050")
plt.show()

# print how accurate the model fits the data as of 2022
ind_2022 = np.argsort(np.abs(ts + 2011 - 2022))[0]
print("model_C in 2022:", model_C[ind_2022])
print("actualC in 2022:", C[-1])
print("model_H in 2022:", model_H[ind_2022])
print("actualH in 2022:", H[-1])
print("model_E in 2022:", model_E[ind_2022])
print("actualE in 2022:", E[-1])
print("model_S in 2022:", model_S[ind_2022])
print("actualS in 2022:", S[-1])
```