HW 5: Optimal PHEV Energy Management via Dynamic Programming

Franklin Zhao
SID: 3033030808

13 Apr. 2018

Problem 1

(a)

$$\min_{P_{fuel}(K)} J = \sum_{k=0}^{N-1} \alpha \Delta t P_{fuel}(K) \tag{1}$$

(b)

The constrains are shown as follows:

$$P_{fuel}(k) = \frac{P_{dem}(k) - P_{batt}(k)}{\eta(P_{dem}(k) - P_{batt}(k))} \tag{2}$$

The above equation just derived $P_{fuel}(k)$ and eliminated the variable $P_{eng}(k)$ as done in the lecture given the enginer power definition.

$$SOC(k+1) = SOC(k) - \frac{\Delta t}{Q_{cap} V_{oc}} P_{batt}(k) \qquad SOC(0) = SOC_0 \tag{3}$$

The above constraint describes the battery dynamics (in steps) with the initial condition.

$$SOC^{min} \leq SOC(k) \leq SOC^{max} \tag{4}$$

The above constraint shows the boundaries of SOC in every steps.

$$-P_{batt}^{max} \leq P_{batt}(t) \leq P_{batt}^{max} \tag{5}$$

The above constraint shows the boundaries of power generation in every steps.

$$0 \leq P_{dem}(k) - P_{batt}(k) \leq P_{eng}^{max} \tag{6}$$

The above constraint shows the boundaries of the power that a vehicle can generate.

(c)

Control variable: battery power $P_{batt}(k)$
State variable: state of charge $SOC(k)$

Problem 2

(a)

The value function is defined as the minimum of the fuel consumption in current time add the cost of all the fuel consumption in future steps.

(b)

Principle of optimality equation:

$$V_k(SOC(k)) = \min\{\alpha \Delta t P_{fuel}(k) + V_{k+1}(SOC(k+1))\}, \qquad k = 0, 1, ..., N-1 \tag{7}$$

with the boundary condition:

$$V_k(SOC(N)) = 0 \tag{8}$$

Problem 3

(a)

First limit is just Equation (5):

$$-P_{batt}^{max} \leq P_{batt}(t) \leq P_{batt}^{max} \tag{9}$$

For the second limit, From Equation (3), we know:

$$P_{batt}(k) = \frac{Q_{cap}V_{oc}}{\Delta t}(SOC(k) - SOC(K+1)) \tag{10}$$

Plugging this into Equation (5), we get:

$$\frac{Q_{cap}V_{oc}}{\Delta t}(SOC(k) - SOC^{max}) \leq P_{batt}(k) \leq \frac{Q_{cap}V_{oc}}{\Delta t}(SOC(k) - SOC^{min}) \tag{11}$$

For the third limit, transforming Equation (6), the following will be derived

$$P_{dem}(k) - P_{eng}^{max} \leq P_{batt}(k) \leq P_{dem}(k) \tag{12}$$

(b)

Now let's collapse the contraints in (a) into one inequality:

$$\begin{aligned} & \max\{-P_{batt}^{max}, \tfrac{Q_{cap}V_{oc}}{\Delta t}(SOC(k) - SOC^{max}), P_{dem}(k) - P_{eng}^{max}\} \\ \leq \;& P_{batt}(k) \\ \leq \;& \min\{P_{batt}^{max}, \tfrac{Q_{cap}V_{oc}}{\Delta t}(SOC(k) - SOC^{min}), P_{dem}(k)\} \end{aligned} \tag{13}$$

Problem 4

(a)

The plots are shown as follow:

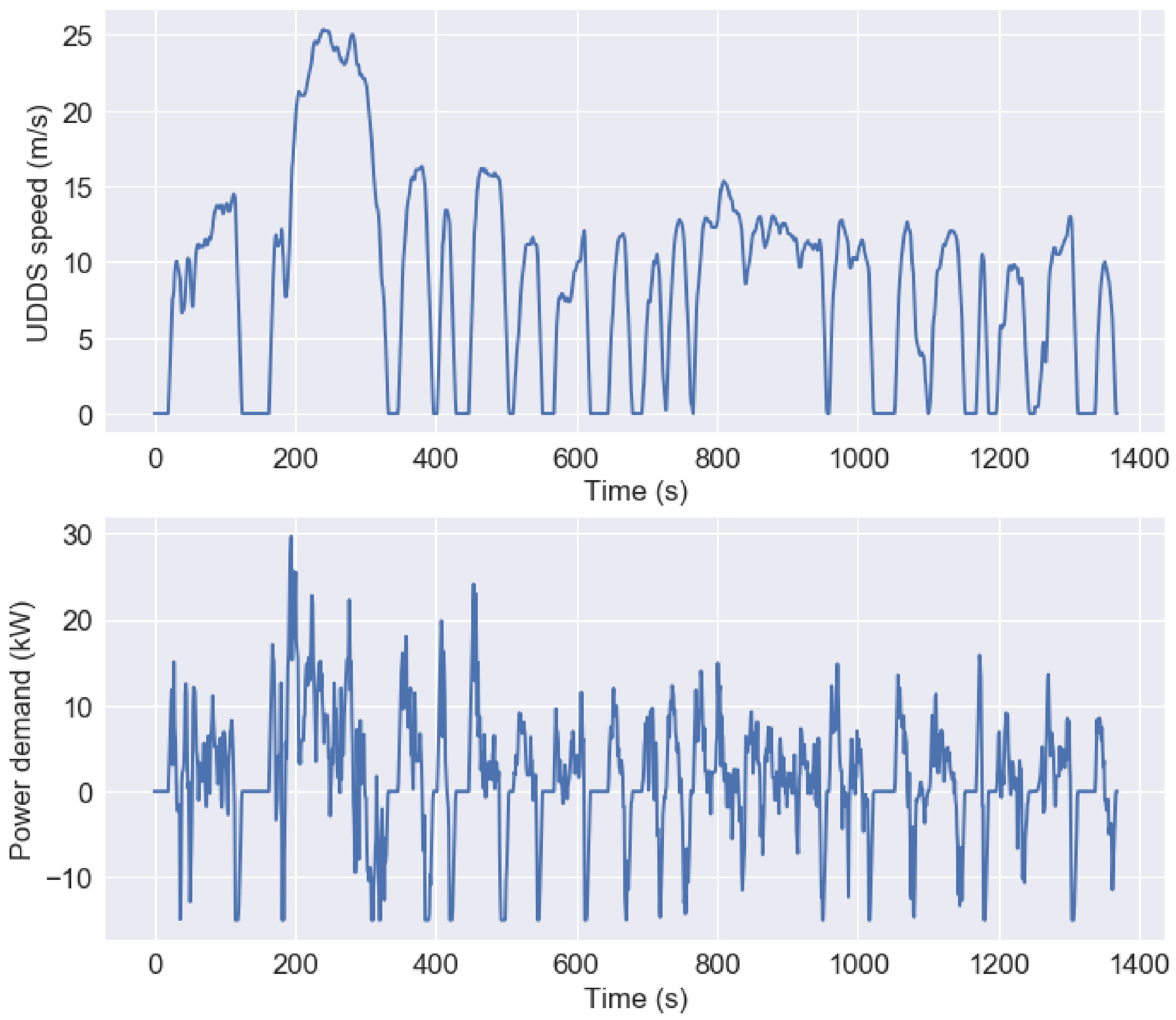

Fig. 1: UDDS spped vs. time & power demand vs. time

(b)

The plot is shown as follow:

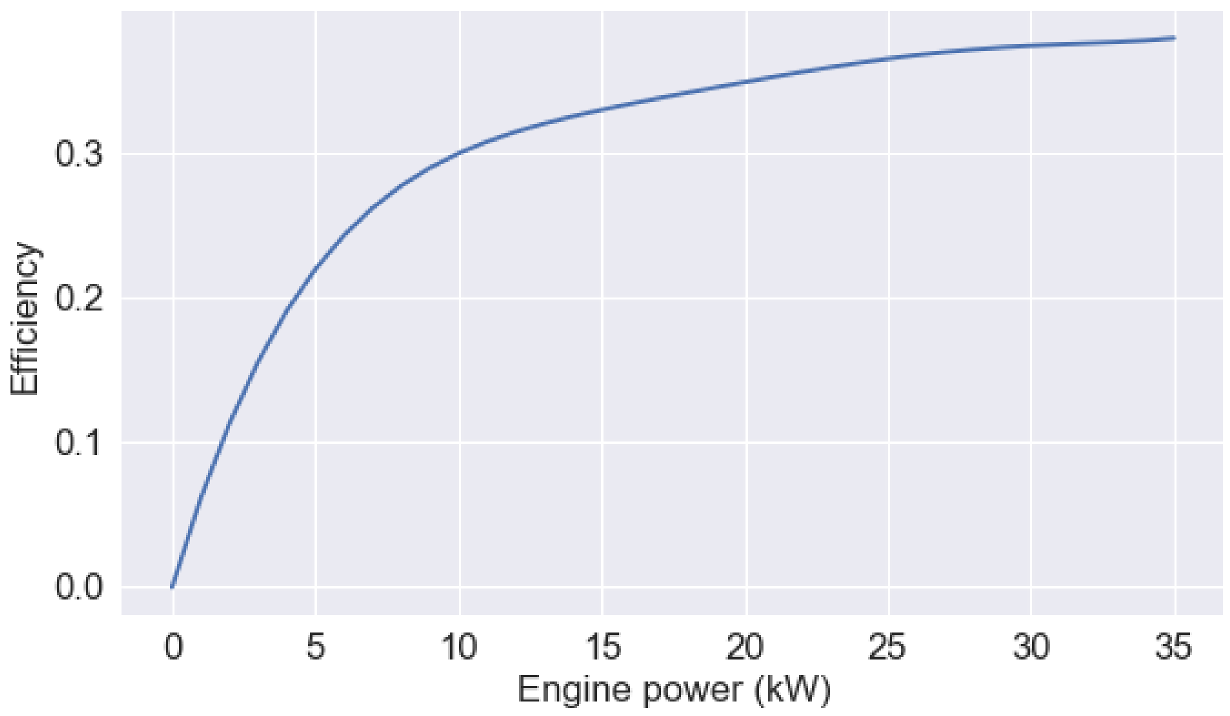

Fig. 2: Engine efficiency curve

Problem 5

(a) (b) (c)

```python
## SOLVE DYNAMIC PROGRAM
start = timeit.timeit()

# Boundary Condition of Value Function (Principle of Optimality)
V[:,N] = 0

# Iterate backward in time
for k in range(N-1, -1, -1):

# Iterate over SOC
for idx in range(0,ns):

```

```
# Find dominant bounds for P_batt
lb = max(-P_batt_max, Qcap * V_oc / Delta_t * (SOC_grid[idx]-SOC_max), P_dem[k
    ] - P_eng_max)
ub = min(P_batt_max, Qcap * V_oc / Delta_t * (SOC_grid[idx]-SOC_min), P_dem[k
    ])

# Grid Battery Power between dominant bounds
P_batt_grid = np.linspace(lb,ub,200)

# Compute engine power (vectorized for all P_batt_grid)
P_eng = -P_batt_grid + P_dem[k]

# Cost-per-time-step, a.k.a. fuel consumed at each stage (vectorized for all
    P_batt_grid)
g_k = alph * Delta_t / eta_eng(P_eng) * P_eng

# compute next SOC using dynamics
SOC_nxt = SOC_grid[idx] - Delta_t / (Qcap * V_oc) * P_batt_grid

# Compute value function at nxt time step (need to interpolate)
V_nxt = interp(SOC_nxt, SOC_grid, V[:,k+1])

# Value Function (Principle of Optimality)
V[idx,k] = min(g_k + V_nxt)
ind = np.argmin(g_k + V_nxt)

# Save Optimal Control
u_star[idx,k] = P_batt_grid[ind]

# DP Timer
end = timeit.timeit()
print(str(end - start) + " seconds")
```

Problem 6

(a)

```
## Simulate Results

# Preallocate
SOC_sim = np.zeros((N,))
P_batt_sim = np.zeros((N,))
P_eng_sim = np.zeros((N,))
J_sim = np.zeros((N,))

# Initialize
SOC_0 = 0.75# put initial SOC here!
SOC_sim[0] = SOC_0

# Simulate PHEV Dynamics
for k in range(0,(N-1)):

# Use optimal battery power, for given SOC
P_batt_sim[k] = interp(SOC_sim[k], SOC_grid, u_star[:, k])

# Compute engine power
P_eng_sim[k] = P_dem[k] - P_batt_sim[k]

# Fuel Consumption
J_sim[k] = alph * Delta_t / eta_eng(P_eng_sim[k]) * P_eng_sim[k]

# Time-step SOC dynamics
SOC_sim[k+1] = -Delta_t / (Qcap * V_oc) * P_batt_sim[k] + SOC_sim[k]
```

(b)

The minimum fuel consumption is just 0.

(c)

The plots are shown as follow:

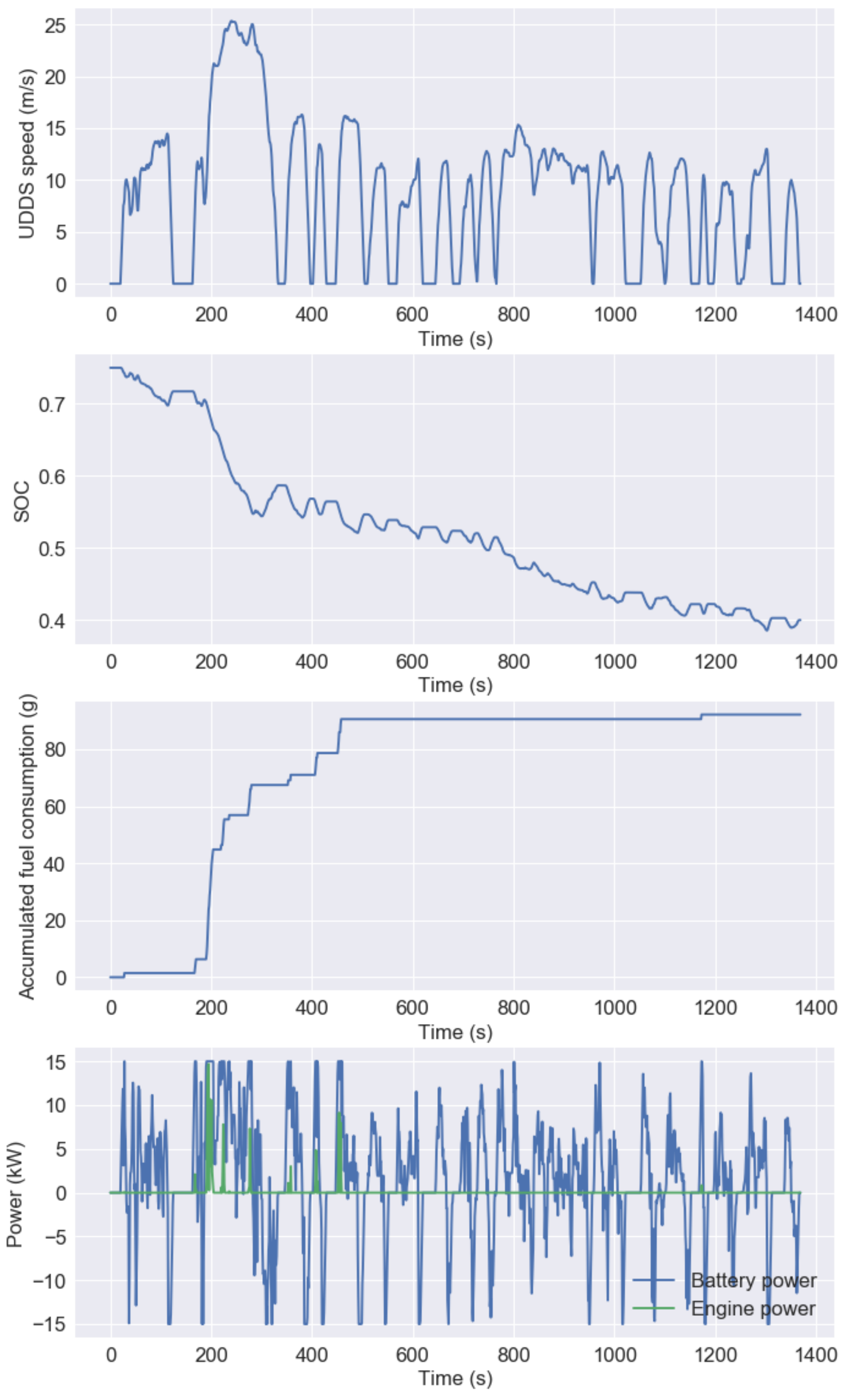

Fig. 3: Data and result visualization

(d)

Yes, all the inequality constraints are satisfied.

Problem 7

We can change the initial SOC "*SOC_0*" in the code, and compute the simulated final SOC and total fuel consumption using the following script:

```
SOC_sim[-1], np.cumsum(J_sim)[-1]/1000
```

The results are shown in Table 1.

Table 1: PHEV energy management results

Initial SOC	**Final SOC**	**Total Fuel Cons. (kg)**
0.9	0.55	0.09
0.75	0.40	0.09
0.6	0.27	0.13
0.45	0.27	0.41
0.3	0.27	0.65

Problem 8

From the result obtained in the previous problem, we notice that as the initial SOC decreases, the SOC trajectory decreases and finally remain unchanged (converged); while the total fuel consumption keeps increasing. I think the reason is that PHEV has the priority to use the electrical energy (battery) and use only a little fuel energy in the beginning. As battery SOC goes down to a certain level (0.27 shown in the result), fuel begins to be consumed as the main energy source to keep the battery in good conditions and avoid the low power issue.