HW 5: Optimal PHEV Energy Management via Dynamic Programming

Due: Friday April 22 at 5:00pm PT

This assignment will provide hands-on practice for dynamic programming (DP) with application to energy management in plug-in hybrid electric vehicles (PHEVs). Unlike the example discussed in class, you will (i) use DP instead of LP, (ii) apply the algorithm to a plug-in HEV, and (iii) include more realistic nonlinearities in the vehicle model.

Reading
The following readings are intended to provided context and background for this homework assignment.

- "Hybrid and Electrified Vehicles: The Role of Dynamics and Control" by Rizzoni and Peng.
- "Smart, Connected & Electric: The Future of the Automobile" by Phillips, McGee, Kristinsson, & Yu.

Problem Data

- You will optimize for the Urban Dynamic Driving Schedule (UDDS) power demand, provided in bCourses.
- Minimize total fuel consumption $J = \sum_{k=0}^{N-1} \alpha \Delta t \ P_{fuel}(k)$.
- Engine power is given by $P_{eng}(k) = \eta(P_{eng}(k)) \cdot P_{fuel}(k)$, where $\eta(\cdot)$ is the engine efficiency as a function of power demand. This function is computed by `eta_eng` (a separate .m file in Matlab and integrated directly in the iPython Notebook).
- The battery dynamics are given by: $SOC(k+1) = SOC(k) - \frac{\Delta t}{Q_{cap} V_{oc}} P_{batt}(k)$
- The battery SOC, battery power, and engine power are limited according to:

$$\begin{array}{rcccl} SOC^{\min} & \leq & SOC(k) & \leq & SOC^{\max} \\ -P_{batt}^{\max} & \leq & P_{batt}(k) & \leq & P_{batt}^{\max} \\ 0 & \leq & P_{eng}(k) & \leq & P_{eng}^{\max} \end{array}$$

Formulate

Problem 1: Formulate an optimization problem. Provide the following:

(a) Write down the objective function.

(b) Write down ALL the constraints, after eliminating the variable $P_{eng}(k)$ as done in lecture. Label the physical meaning of each constraint.

(c) What is the control variable? What is the state variable?

Problem 2:

(a) Define - in words - an appropriate value function, as done in lecture and the CH 5 notes.

(b) Write down the principle of optimality equation and boundary condition.

Problem 3: This problem assists you to write the code. You should have formulated three pairs of upper/lower limits, i.e. inequalities.

(a) Re-arrange each pair as upper/lower limits on $P_{batt}(k)$.

(b) Clearly, one of the lower limits and one of the upper limits will dominate the others, at each time step. Collapse the three pairs into the form: $\max\{\cdot,\cdot,\cdot\} \leq P_{batt}(k) \leq \min\{\cdot,\cdot,\cdot\}$. This produces simple bounds on $P_{batt}(k)$, for each time step. This will greatly assist our coding of the principle of optimality.

Data

Problem 4: Download the files { `HW5_Skeleton.m`, `UDDS_Pdem.csv`. `eta_eng.m` } for Matlab or {`HW5_Skeleton.ipynb`, `UDDS_Pdem.csv` } for Python from bCourses. The data file `UDDS_Pdem.csv` contains vectors `t`, `v_dc`, `P_dem`, which respectively give the time [sec], drive cycle speed [m/s], and corresponding power demand [kW].

(a) In one figure, create two subplots. The top subplot plots the UDDS speed vs. time. The bottom subplot includes power demand vs. time. Use appropriate line styles, axis labels, font sizes, etc. Include the figure in your report.

(b) In a second figure, plot the engine efficiency curve, P_{eng} versus $\eta(P_{eng})$.

Code

Problem 5: For all the following subproblems, you may include the entire code in your report.

(a) Encode the value function's boundary condition in the skeleton code.

(b) For each time-step and each state in the SOC grid, find the lower/upper bounds for $P_{batt}(k)$. Use these bounds to create a grid of feasible P_{batt} values.

(c) Implement the recursive principle of optimality equation in your code. This includes (i) the cost-per-time-step calculation, (ii) computing $SOC(k+1)$, and (iii) interpolation of $V_{k+1}(SOC(k+1))$. Interpolation is necessary because V_k is defined on the SOC grid, and $SOC(k+1)$ will never fall exactly on a grid-point. Therefore we interpolate between grid points to find $V_{k+1}(SOC(k+1))$. The key idea is that the sub-optimization is done by griding P_{batt} and selecting the value which minimizes the min operator's argument.

Problem 6: Simulate optimal energy management results. Provide the code and plots in your report.

(a) Suppose the initial SOC is 0.75. Write a `for` loop that simulates the discrete-time battery dynamics, using the optimal battery power $P^*_{batt}(k)$. This includes (i) interpolating $P^*_{batt}(k)$ over the SOC grid, (ii) computing the accumulated fuel consumption, and (iii) the SOC dynamics.

(b) What is the minimum fuel consumption?

(c) Create a figure with four subplots, using appropriate line-styles, axis labels, and legends:

- **[Subplot 1]** UDDS speed versus time.
- **[Subplot 2]** SOC versus time
- **[Subplot 3]** Accumulated fuel consumption versus time [kg].
- **[Subplot 4]** Battery and engine power versus time [kW].

(d) Are all the inequality constraints satisfied?

Additional Analysis

Problem 7: Note that DP computes the optimal energy management strategy for all possible initial SOC values. Use this fact to complete Table 1.

Table 1: PHEV Energy Management Results

Initial SOC	**Final SOC**	**Total Fuel Cons. [kg]**
0.9		
0.75		
0.6		
0.45		
0.3		

Problem 8: Comment on the SOC and engine power trajectories. What do you observe?

Deliverables

Submit the following on bCourses. Be sure that the function files are named exactly as specified (including spelling and case), and make sure to zip all files.

`LASTNAME_FIRSTNAME_HW5.PDF`
`LASTNAME_FIRSTNAME_HW5.zip` which contains your respective Matlab or Python files.

Remark

In Spring 2015, two students modified this homework slightly for application to solar generation and storage, and launched energy analytics start-up *eLum*.