EE 364a HW6

Yen-Yu Chang

February 21, 2020

6.9 The domain of the objective function is:

$$D = \{(a,b) \in \mathbf{R}^{m+1} \times \mathbf{R}^n | q(t) > 0, \alpha \le t \le \beta\}.$$

Suppose (a_x, b_x), (a_y, b_y) in D and $0 \le \theta \le 1$.
Then we let

$$(a,b) = \theta(a_x, b_x) + (1-\theta)(a_y, b_y) = (\theta a_x + (1-\theta)a_y, \theta b_x + (1-\theta) b_y)\,.$$

We can see that

$$(a,b) \in \mathbf{R}^{m+1} \times \mathbf{R}^n.$$

And we have

$$\begin{aligned} q(t) &= 1 + b_1 t + \ldots + b_n t^n = 1 + (\theta b_{x1} + (1-\theta) b_{y1})\, t + \ldots + (\theta b_{xn} + (1-\theta) b_{yn})\, t^n \\ &= (1 + b_{x1} t + \ldots + b_{xn} t^n) + (1 + b_{y1} t + \ldots + b_{yn} t^n) > 0 \end{aligned}$$

Also, by definition, $\alpha \le t \le \beta$. Thus, $(a,b) \in D$, which implies that the domain of the objective function is convex.

Since $q(t_i) > 0$, for $i = 1, \ldots, k$, we have:

$$\max_{i=1,\ldots,k} \left| \frac{p(t_i)}{q(t_i)} - y_i \right| \le \gamma$$

if and only if

$$-\gamma q(t_i) \le p(t_i) - y_i q(t_i) \le \gamma q(t_i), \quad i = 1, \ldots, k.$$

We can see that it is a pair of linear inequalities, so the sublevel sets are convex for all γ. Thus, the original problem is quasiconvex.

7.5

(a) Given the observed sample sequence $y(1) = k_1, y(2) = k_2, \ldots, y(N) = k_n$, we have:

$$P_{k_2,k_1} \times P_{k_3,k_2} \times \ldots \times P_{k_n,k_{n-1}} = \prod_{j=1}^{n} \prod_{i=1}^{n} P_{ij}^{n_{ij}}$$

where n_{ij} is the number of times the state changed from j to i.

The ML estimation problem can be expressed as:

$$\begin{array}{ll} \text{maximize} & \sum_{j=1}^{n}\sum_{i=1}^{n} n_{ij}\log P_{ij}, \\ \text{subject to} & \mathbf{1}^T P = \mathbf{1}^T \end{array}$$

We can solve the problem by each column separately, then we set j'th column of P as $p_j = (P_{1j}, \ldots, P_{nj})$. Then the problem can be reformulated as:

$$\begin{array}{ll} \text{maximize} & \sum_{i=1}^{n} n_{ij}\log p_{ij}, \\ \text{subject to} & \mathbf{1}^T p_j = 1 \end{array}$$

The Lagrangian is:

$$L(p_j, \lambda) = -\sum_{i=1}^{n} n_{ij}\log p_{ij} + \lambda(\mathbf{1}^T p_j - 1)$$

The Lagrange dual function is:

$$g(\lambda) = \inf_{p_j}\left(-\sum_{i=1}^{n} n_{ij}\log p_{ij} + \lambda(\mathbf{1}^T p_j - 1)\right)$$

Set gradient equal to zero:

$$\nabla_{p_j} L(p_j, \lambda) = [\frac{-n_{1j}}{p_{1j}} + \lambda, \ldots, \frac{-n_{nj}}{p_{nj}} + \lambda]^T = 0$$

Thus, we have

$$\frac{n_{1j}}{p_{1j}} = \ldots = \frac{n_{nj}}{p_{nj}}.$$

Therefore, we prove that:

$$P_{ij} = \frac{n_{ij}}{\sum_{i=1}^{n} n_{ij}}$$

(b) Then ML estimation problem can be expressed as:

$$\begin{array}{ll} \text{maximize} & \sum_{j=1}^{n}\sum_{i=1}^{n} n_{ij}\log P_{ij}, \\ \text{subject to} & \mathbf{1}^T P = \mathbf{1}^T \\ & Pq = q \end{array}$$

A5.2 We have $\max_{i=1,\ldots,k} |f(t_i) - y_i| \le \gamma$ if and only if

$$\left| \frac{a_0 + a_1 t_i + a_2 t_i^2}{1 + b_1 t_i + b_2 t_i^2} - y_i \right| \le \gamma, \quad i = 1, \ldots, k.$$

Since $1 + b_1 t_i + b_2 t_i^2 > 0$ for $i = 1, \ldots, k$, the inequality above can be reformulated as

$$|a_0 + a_t t_i + a_2 t_i^2 - y_i(1 + b_1 t_i + b_2 t_i^2)| \le \gamma(1 + b_1 t_i + b_2 t_i^2), \quad i = 1, \ldots, k$$

The inequality above is a set of $2k$ linear inequalities in a and b, which implies that the objective function is quasiconvex.

Thus, the original problem can be expressed as a linear programming feasibility problem as:

$$\begin{array}{ll} \text{find} & a, b, \\ \text{subject to} & a_0 + a_1 t_i + a_2 t_i^2 - y_i(1 + b_1 t_i + b_2 t_i^2) \le \gamma(1 + b_1 t_i + b_2 t_i^2), \quad i = 1, \ldots, k \\ & a_0 + a_1 t_i + a_2 t_i^2 - y_i(1 + b_1 t_i + b_2 t_i^2) \ge -\gamma(1 + b_1 t_i + b_2 t_i^2), \quad i = 1, \ldots, k \end{array}$$

The following Python code solves the problem

```
import cvxpy as cvx
import numpy as np
import matplotlib.pyplot as plt

k = 201
t = -3 + 6 * np.arange(k) / (k - 1)
y = np.exp(t)

Tpowers = np.vstack((np.ones(k), t, t ** 2)).T

a = cvx.Variable((3, 1))
b = cvx.Variable((2, 1))
gamma = cvx.Parameter(nonneg = True)

left = cvx.abs(Tpowers * a - (y.reshape((-1, 1)) * Tpowers)\
* cvx.vstack((np.ones((1, 1)), b)))
right = gamma * Tpowers * cvx.vstack((np.ones((1, 1)), b))

problem = cvx.Problem(cvx.Minimize(0), [left <= right])

lower_bound = 0
upper_bound = np.exp(3)
```

```
tolerance = 1e-3

while upper_bound - lower_bound >= tolerance:
  gamma.value = (upper_bound + lower_bound) / 2
  problem.solve(solver = cvx.ECOS)
  if problem.status == 'optimal':
    upper_bound = gamma.value
    a_opt = a.value
    b_opt = b.value
    obj_opt = gamma.value
  else:
    lower_bound = gamma.value

y_fit = (Tpowers @ a_opt / (Tpowers @ np.vstack((np.ones((1, 1)), b_opt))))

print("a is {}".format(a_opt))
print("b is {}".format(b_opt))
print("optimal objective value is {}".format(obj_opt))

y = y.reshape((1, -1))
y_fit = y_fit.reshape((1, -1))
t = t.reshape((1, -1))

plt.figure()
plt.plot(t[0], y[0], 'b')
plt.plot(t[0], y_fit[0], 'r+')
plt.xlabel('t')
plt.ylabel('y')
plt.savefig('optimal_rational_function.png')

plt.figure()
plt.plot(t[0], y_fit[0] - y[0])
plt.xlabel('t')
plt.ylabel('error')
plt.savefig('error.png')
```

The optimal values are

$$a_0 = 1.00971, \quad a_1 = 0.61197, \quad a_2 = 0.11353, \quad b_1 = -0.41454, \quad b_2 = 0.048487$$

and the optimal objective value is 0.02329. We plot the fit and the error in the following two figures.

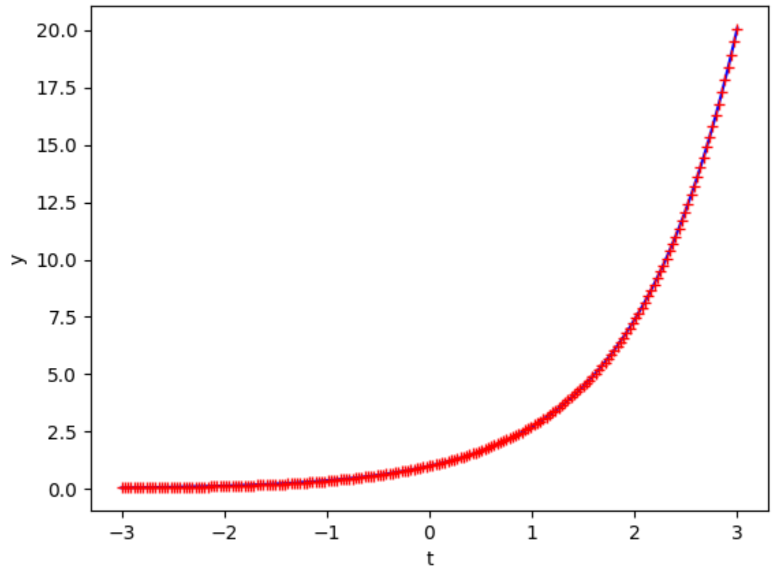

Figure 1: Fit with rational function. The line represents the data and the crosses the fitted points

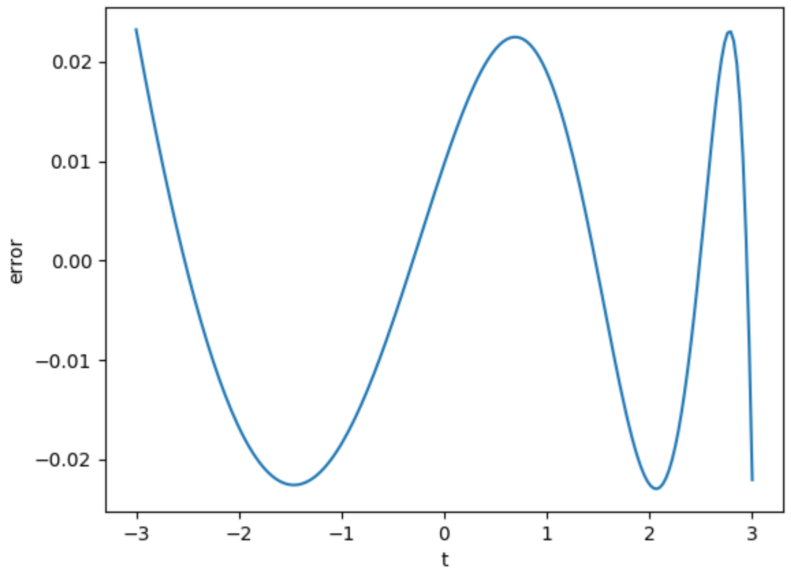

Figure 2: Fitting error

A6.4

(a) The likelihood of the outcome y given a is:

$$p(y|a) = \prod_{i=1,\ldots,n} \Phi\left(\frac{1}{\sigma} y^{(i)}(a_{j^{(i)}} - a_{k^{(i)}})\right)$$

where Φ is the CDF of the standard normal distribution. Then we have the log-likelihood function as:

$$l(a) = \log p(y|a) = \sum_{i=1}^{n} \log \Phi\left(\frac{1}{\sigma}(Aa)_i\right)$$

We can see that it is a concave function, and the maximum likelihood estimate $\hat{a}$ is any solution of

$$\begin{array}{ll} \text{maximize} & l(a) \\ \text{subject to} & 0 \preceq a \preceq 1 \end{array}$$

We can see that the objective function is maximizing a concave function with $2n$ linear inequalities, so it is a convex optimization problem.

(b) The following MATLAB code solves the problem:

```
team_data
A1 = sparse(1:m, train(:, 1), train(:, 3), m, n);
A2 = sparse(1:m, train(:, 2), -train(:, 3), m, n);
A = A1 + A2;

cvx_begin
variable a_hat(n)
minimize(-sum(log_normcdf(A * a_hat/sigma)))
subject to
a_hat >= 0
a_hat <= 1
cvx_end

a_hat
```

Then we get $\hat{a} = (1.0000, 0.0000, 0.6829, 0.3696, 0.7946, 0.5779, 0.3795, 0.0895, 0.6736, 0.5779.)$

(c) The following MATLAB code can help us predict the outcomes in the test set.

```
team_data
A1 = sparse(1:m, train(:, 1), train(:, 3), m, n);
A2 = sparse(1:m, train(:, 2), -train(:, 3), m, n);
```

```
A = A1 + A2;

cvx_begin
variable a_hat(n)
minimize(-sum(log_normcdf(A * a_hat/sigma)))
subject to
a_hat >= 0
a_hat <= 1
cvx_end

A1_test = sparse(1:m_test, test(:,1), 1, m_test, n);
A2_test = sparse(1:m_test, test(:,2), -1, m_test, n);
A_test = A1_test + A2_test;

result = sign(A_test * a_hat)
Pml = 1 - length(find(res - test(:, 3))) / m_test;
Ply = 1 - length(find(train(:, 3) - test(:, 3))) / m_test;
```

Then we have that the maximum likelihood estimate gives a correct prediction of 86.67% of the games in test set, and 75.6% of the games in test set have the same outcome as the games in train set.

A7.23

(a) The objective function of the sum of the disk areas is $\sum_{i=1}^{n} \pi r_i^2$, and the objective function of the sum of the disk perimeters is $\sum_{i=1}^{n} 2\pi r_i$. Both of the objective functions are convex.

Now we consider about the constraints. The first constraint is that the first k disks are fixed, which is a set of linear constraints. The second constraint is the intersection constraint. The intersection constraint can be expresses as $||c_i - c_j||_2 \leq r_i + r_j, \quad (i,j) \in I$. We can see that the intersection constraint is convex since the left hand side is a norm function (convex) and the right hand side is an affine function.

Hence. the first problem can be expressed as:

$$\begin{array}{ll} \text{minimize} & \sum_{i=1}^{n} \pi r_i^2 \\ \text{subject to} & c_i = c_i^{fix}, \quad r_i = r_i^{fix}, \quad i = 1, \ldots, k, \\ & r_i \geq 0, \quad i = 1, \ldots, n, \\ & ||c_i - c_j||_2 \leq r_i + r_j, \quad (i,j) \in I \end{array}$$

And the second problem can be formulated as:

$$\begin{array}{ll} \text{minimize} & \sum_{i=1}^{n} 2\pi r_i \\ \text{subject to} & c_i = c_i^{fix}, \quad r_i = r_i^{fix}, \quad i = 1, \ldots, k, \\ & r_i \geq 0, \quad i = 1, \ldots, n, \\ & ||c_i - c_j||_2 \leq r_i + r_j, \quad (i,j) \in I \end{array}$$

(b) The following Python code solves the problems above:

```
import numpy as np
import matplotlib.pyplot as plt
from disks_data import *
import cvxpy as cvx

c = cvx.Variable((n, 2))
r = cvx.Variable(n)

min_area_obj = cvx.Minimize(cvx.sum_squares(r))
min_perim_obj = cvx.Minimize(cvx.sum(r))

constraints = [r >= 0, c[:k, :] == Cgiven[:k, :], r[:k] == Rgiven[:k]]

for i in range(len(Gindexes)):
constraints += [cvx.norm(c[Gindexes[i, 0], :] - c[Gindexes[i, 1], :])\
<= (r[Gindexes[i, 0]] + r[Gindexes[i, 1]])]

min_total_area = cvx.Problem(min_area_obj, constraints)
min_total_perim = cvx.Problem(min_perim_obj, constraints)

opt_area = min_total_area.solve(solver = cvx.CVXOPT)
print("optimal area: {}".format(np.pi * opt_area))
plot_disks(c.value, r.value, Gindexes, 'area.png')

opt_perim = min_total_perim.solve(solver = cvx.CVXOPT)
print("optimal perimeter: {}".format(2 * np.pi * opt_perim))
plot_disks(c.value, r.value, Gindexes, 'perim.png')
```

The optimal total area is 210.7667, and the optimal total perimeter is 139.3459. The result of optimal total area and optimal total perimeter are below:

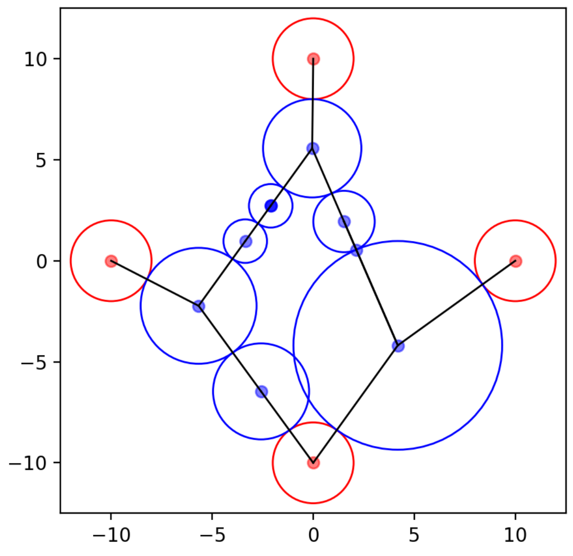

Figure 3: The resulting plot for optimal area disks

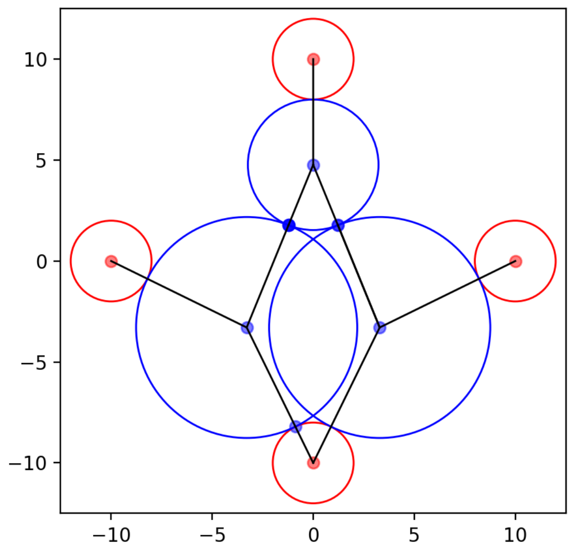

Figure 4: The resulting plot for optimal perimeter disks

A14.7

(a) (i) if: Let $Rx \succ 0$. Since $Rx \in \mathbf{R}^m$ and $Rx \succ 0$, which implies that $(Rx)_i > 0$ for

$i = 1, \ldots, m$. Suppose there exists a vector $p \in \mathbf{R}^m$ such that

$$R^T p = 0, \quad p \succeq 0, \quad p \neq 0$$

Since $Rx \succ 0, p \succeq 0$, and $p \neq 0$, these conditions imply that $p^T Rx > 0$. However, $R^T p = 0$ implies that $(R^T p)^T x = 0 \iff p^T Rx = 0$, which makes a contradiction to $p^T Rx > 0$. Therefore, there exists no vector $p \in \mathbf{R}^m$ such that $R^T p = 0$, $p \succeq 0$, and $p \neq 0$.

(ii) only if: It is equivalent to prove that "there exists a vector $p \in \mathbf{R}^m$ if $Rx \preceq 0$." first of all, the $p \in \mathbf{R}^m$ such that $R^T p = 0$, $p \succeq 0$, and $p \neq 0$. Suppose $Rx \succ 0$. Since $Rx \succ 0$, $p \succeq 0$, and $p \neq 0$, these mean that $p^T Rx > 0$. However, $R^T p = 0$ implies that $p^T Rx = 0$, which makes a contradiction to previous $p^T Rx > 0$. Thus, $Rx \preceq 0$.

In conclusion, we prove that $Rx \succ 0$ if and only if there exists no vector $p \in \mathbf{R}^m$ that satisfies

$$R^T p = 0, \quad p \succeq 0, \quad p \neq 0.$$

(b) By (a), we know that there is no sure-win betting scheme if there is a vector $p \in \mathbf{R}^m$ satisfies that $R^T p = 0$, $p \succeq 0$, and $p \neq 0$. Since p is a probability vector on the outcomes, we have $\mathbf{1}^T p = 1$. If we solve $R^T p = 0$ with $\mathbf{1}^T p = 1$, we have the following equations

$$\begin{cases} \lambda_1 p_1 - p_2 - p_3 - \ldots - p_m & = 0 \\ -p_1 + \lambda_2 p_2 - p_3 - \ldots - p_m & = 0 \\ \vdots & \\ -p_1 - p_2 - p_3 - \ldots + \lambda_m p_m & = 0 \\ p_1 + p_2 + p_3 + \ldots + p_m & = 1 \end{cases} \tag{1}$$

By solving the system of equations, we get

$$p_i = \frac{1}{1+\lambda_i}, \quad i = 1, \ldots, m.$$

Therefore,

$$\sum_{i=1}^{m} p_i = \sum_{i=1}^{m} \frac{1}{1+\lambda_i} = 1.$$

If the equality is not satisfied and

$$x_i = \frac{1/(1+\lambda_i)}{1 - \sum_{i=1}^{m} 1/(1+\lambda_i)}$$

Then we have

$$\begin{aligned}
(Rx)_i &= \frac{\frac{-1}{1+\lambda_1} + \frac{-1}{1+\lambda_2} + \ldots + \frac{-1}{1+\lambda_{i-1}} + \frac{\lambda_i}{1+\lambda_i} + \frac{-1}{1+\lambda_{i+1}} + \ldots + \frac{-1}{1+\lambda_m}}{1 - \sum_{i=1}^m 1/(1+\lambda_i)} \\
&= \frac{\frac{\lambda_i}{1+\lambda_i} - \sum_{i=1}^m 1/(1+\lambda_i) + \frac{1}{1+\lambda_i}}{1 - \sum_{i=1}^m 1/(1+\lambda_i)} \\
&= \frac{1 - \sum_{i=1}^m 1/(1+\lambda_i)}{1 - \sum_{i=1}^m 1/(1+\lambda_i)} \\
&= 1
\end{aligned}$$

Therefore, $Rx \succ 0$, which means that the betting strategy always results in a profit.

A17.13

(a) The problem can be expressed as:

$$\begin{array}{ll}
\text{minimize} & \frac{1}{4}(R^{buy})^T \max\{p^{grid}, 0\} - \frac{1}{4}(R^{sell})^T \max\{-p^{grid}, 0\} \\
\text{subject to} & -C\mathbf{1} \leq p^{batt} \leq D\mathbf{1} \\
& p^{ld} - p^{grid} - p^{batt} - p^{PV} = 0 \\
& 0 \preceq q \preceq Q\mathbf{1} \\
& q_{i+1} = q_i - \frac{1}{4}p_i^{batt}, \quad i = 1, \ldots, 95 \\
& q_1 = q_{96} - \frac{1}{4}p_{96}^{batt}
\end{array}$$

with variables p^{grid}, p^{batt}, and q.

The objective function and the constraints are convex, but the objective function is not in DCP since there is a minus sign in the second term. We can break up grid power into two components as $p^{grid} = p^{buy} - p^{sell}$, where $p^{buy} \succeq 0$ and $p^{sell} \succeq 0$. Therefore, we can reformulate the problem as

$$\begin{array}{ll}
\text{minimize} & \frac{1}{4}(R^{buy})^T p^{buy} - \frac{1}{4}(R^{sell})^T p^{sell} \\
\text{subject to} & -C\mathbf{1} \leq p^{batt} \leq D\mathbf{1} \\
& p^{ld} - p^{grid} - p^{batt} - p^{PV} = 0 \\
& 0 \preceq q \preceq Q\mathbf{1} \\
& q_{i+1} = q_i - \frac{1}{4}p_i^{batt}, \quad i = 1, \ldots, 95 \\
& q_1 = q_{96} - \frac{1}{4}p_{96}^{batt} \\
& p^{grid} = p^{buy} - p^{sell} \\
& p^{buy} \succeq 0 \\
& p^{sell} \succeq 0
\end{array}$$

with variables $p^{grid}, p^{buy}, p^{sell}, p^{batt}$, and q.

By solving the convex optimization problem above, we get the optimal cost as \$33.161. And the figures of $p^{grid}, p^{load}, p^{pv}, p^{batt}, q$ versus i are follow:

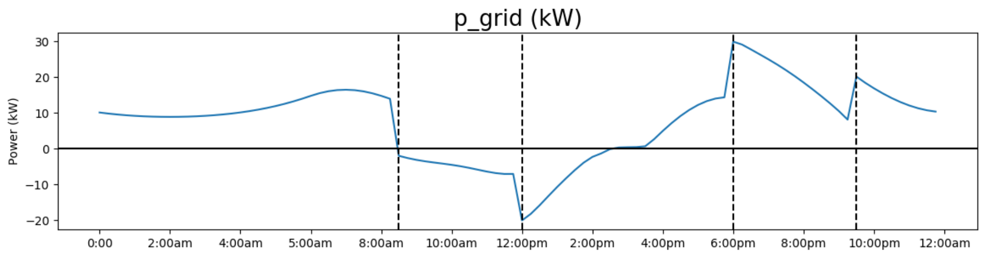

Figure 5: P^{grid}

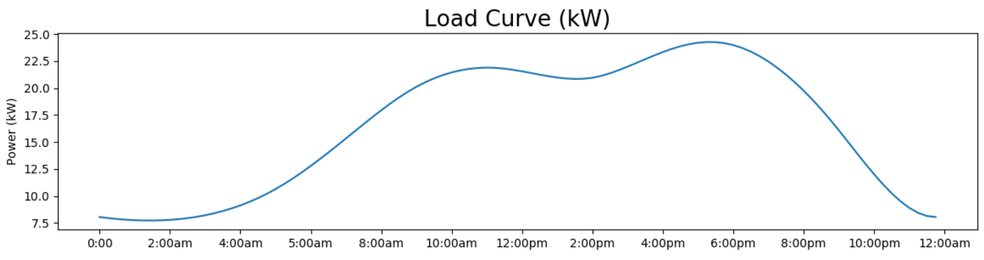

Figure 6: P^{ld}

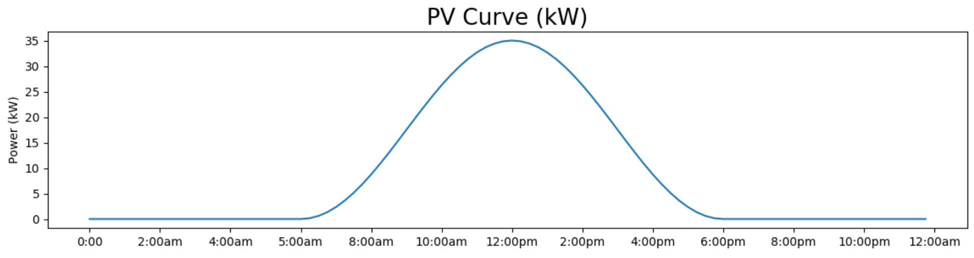

Figure 7: P^{pv}

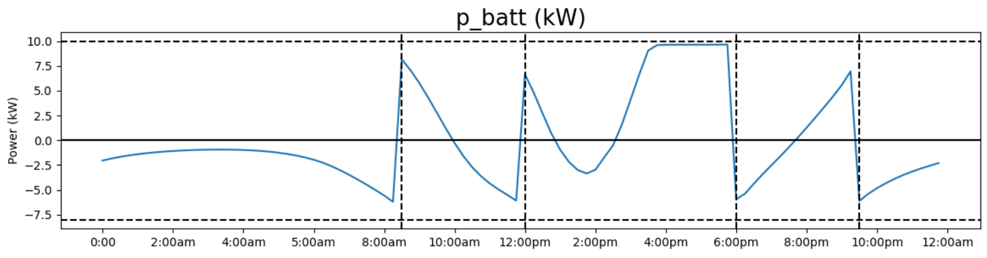

Figure 8: P^{batt}

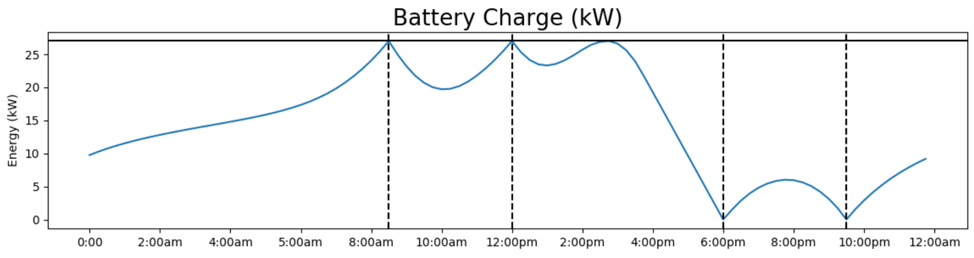

Figure 9: q

(b) The figure of LMP is follow:

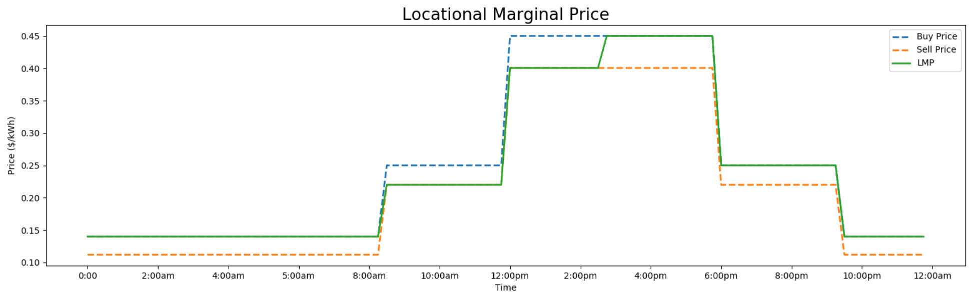

Figure 10: LMP

The LMP is equal to the buy price of the current period when we're buying power and the sell price of the current period if we're selling power.

(c) we get load cost $= \$107.982$, battery cost $= \$-8.370$, PV cost $= \$-66.451$, and the effective grid cost is \$33.161. And effective cost minus the sum of the others is 0.