DATA 609 - Homework 4: Convex Functions

Eddie Xu

Instructions

Please submit a .qmd file along with a rendered pdf to the Brightspace page for this assignment. You may use whatever language you like within your qmd file, I recommend python, julia, or R.

Problem 1: (Exercise 3.2 in CVX Book)

(a) The figure below shows three levels sets for a function f. The value of the function on the level set is indicated by the number next to each curve. For example, the curve labeled 1 corresponds to the points $\mathbf{x} \in \mathbb{R}^2$ satisfying $f(\mathbf{x}) = 1$.

Determine whether it is possible for the function f to be convex, concave, quasiconvex, or quasiconcave. Give a brief justification for your answer.

Note: it may be that several options are possible, that one is possible, or that none at all are.

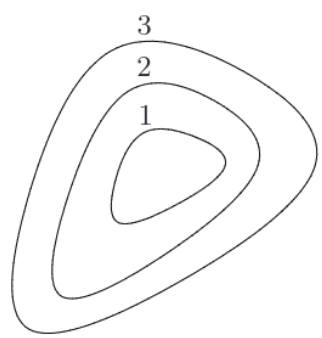


Figure 1: Level Curves of f

Problem 1(a) Solution

The first function appears to be a quasiconvex or convex because the sublevel sets appear to be convex. It is definitely not concave or quasiconcave because the superlevel sets appears not to be convex.

(b) The figure below shows level sets for a different function g. Again determine whether it is possible for the function g to be convex, concave, quasiconvex, or quasiconcave. Note: it may be that several options are possible, that one is possible, or that none at all are.

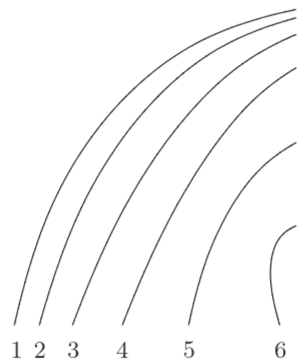


Figure 2: Level Curves of g

Problem 1(b) Solution

For this funcion, it appears not to be convex, concave, quasiconvex, or quasiconcave. The reason why the the sublevel sets appears not to be convex.

Problem 2: (CVX Book Extended Exercises 3.1)

Determine the curvature of the functions below. Your responses can be affine, convex, concave, or none (meaning not convex or concave). Provide a brief justification

(a) $f(x) = \min(2, x, \sqrt{x})$ with $\mathbf{dom} f = \{x \mid x \geq 0\}$ (i.e. $\mathbb{R}_+$)
(b) $f(x) = x^3$, with $\mathbf{dom} f = \mathbb{R}$
(c) $f(x) = x^3$, with $\mathbf{dom} f = \{x \mid x \geq 1\}$
(d) $f(x, y) = \sqrt{x \min(y, 2)}$, with $\mathbf{dom} f = \{(x, y) \mid x \geq 0\,, y \geq 0\}$ (i.e. $\mathbb{R}^2_+$)
(e) $f(x, y) = \left(\sqrt{x} + \sqrt{y}\right)^2$ with $\mathbf{dom} f = \{(x, y) \mid x \geq 0\,,\, y \geq 0\}$ (i.e. $\mathbb{R}^2_+$)

Problem 2 Solution

(a) This function is convex on $\mathbb{R}_+$ because 2 and x are affline functions and the second derviative of $\sqrt{x}$ is concave.
(b) This function is not convex or concave because the second derivative of x^3 can be negative, positive or zero.
(c) This function is convex because the second derivative of x^3 is positive when $\{x \geq 1\}$.
(d) This function is concave because the $\sqrt{x}$ is concave for when x is positive or zero and the $\sqrt{min(y, 2)}$ is concave when y is positive or zero.
(e) This function is convex because the square of a sum of $\sqrt{x} + \sqrt{y}$ is convex, therefore the square of that will be convex as well.

Problem 3: (Selected from CVX Book Extended Exercises 3.51-3.52)

For each of the following problems implement the following functions using Disciplined Convex Programming and CVX, and use CVX to verify that they are convex.

(a) $f(x, y) = \frac{1}{xy}$, with $\mathbf{dom} f = \mathbb{R}^2_{++}$. Hint: Use the Atoms listed below part (b) as well as addition, subtraction, and scalar multiplication.

There are multiple ways to solve this problem.

Problem 3(a) Solution

```
import cvxpy as cp

# define both variables which need to be positive
x = cp.Variable(pos=True)
y = cp.Variable(pos=True)

# Define the function f(x, y) = 1 / (x * y)
z = cp.multiply(x, y)
f = cp.inv_pos(z)

# miniminze and set up the problem
objective = cp.Minimize(f)
problem = cp.Problem(objective)

# check if the problem is convex
print(f'Is the problem convex? {problem.is_dcp()}')
```

```python
# solve the problem following DCP rules
problem.solve(gp=True)

# print results
print(f'The Optimal x is {x.value} and the optimal y is {y.value}.')
```

```
Is the problem convex? False
The Optimal x is None and the optimal y is None.
```

(b) $f(x,y) = \sqrt{1 + \frac{x^2}{y}}$, with $\mathbf{dom} f = \mathbb{R} \times \mathbb{R}_{++}$ (this means x is any real number and y is strictly greater than 0.

Problem 3(b) Solution

```python
# load packages
import cvxpy as cp

# define variables
x = cp.Variable()
y = cp.Variable(pos=True)

# define the function f(x, y) = sqrt(1 + x^2 / y)
z = cp.sqrt(cp.quad_over_lin(x, y))
f = cp.sqrt(1) + z

# define the constraint and minimize
constraints = [y >= 0]
objective = cp.Minimize(f)

# define the problem
problem = cp.Problem(objective, constraints)

# Check if the problem is convex
print(f'Is the problem convex? {problem.is_dcp()}')

# solve the problem
problem.solve(qcp=True)

# print results
print(f'The Optimal x is {x.value} and the optimal y is {y.value}.')
```

```
Is the problem convex? False
The Optimal x is 0.0 and the optimal y is 0.8613653287833809.
```

Hint: The following atoms may be helpful, there are multiple ways to solve this problem.

- `inv_pos(u)`, which is $1/u$, with domain $\mathbb{R}_{++}$
- `square(u)`, which is `u^2` , with domain $\mathbb{R}$
- `sqrt(u)`, which is $\sqrt{u}$, with domain $\mathbb{R}_{+}$
- `geo_mean(u,v)`, which is $\sqrt{uv}$, with domain $\mathbb{R}^2_{+}$
- `quad_over_lin(u,v)`, which is u^2/v, with domain $\mathbb{R} \times \mathbb{R}_{++}$
- `norm2(u,v)`, which is $\sqrt{u^2+v^2}$, with domain $\mathbb{R}^2$.

Problem 4: Periodic Poisson Regression to predict Car Crashes

For this problem, we will be working with the dataset manhattan_crashes.csv, which contains a time series of the number of car crashes occurring in Manhattan every hour for a period of time.

Here we will develop a Poisson regression model to predict rate of crashes during each time of day.

(a) Consider the following statistical model for the number of crashes N_i during hour i of a given day.

$$N_i \sim \exp(-\lambda_i)\frac{\lambda_i^{N_i}}{N_i!}$$

where i ranges from 0 to 23 and corresponds to the hour of the day. Suppose that we have a dataset of counts for crashes where C_{ni} is the number of crashes that occur in the ith hour of the nth day. Then the `log` likelihood function for the parameters λ is:

$$-\log(p(C|\lambda)) = \sum_{n=1}^{N}\sum_{i=0}^{23}(\lambda_i - C_{ni}\log(\lambda_i) + \log(C_{ni}!))$$

However, we can drop terms terms that don't depend on λ because they will have no impact on the maximum likelihood solution. This lets us form a simpler objective function:

$$L(C|\lambda) = \sum_{n=1}^{N}\sum_{i=0}^{23}(\lambda_i - C_{ni}\log(\lambda_i))$$

or if we use matrix-vector notation:

$$L(C|\lambda) = N\mathbf{1}^T\lambda - \mathbf{1}^T C\log(\lambda)$$

where $\log(\lambda)$ is interpreted as a vector whose ith entry is $\log(\lambda_i)$ and the vector $\mathbf{1}^T$ has all entries equal to 1 and has the right dimension in each case to make the resulting expressions scalars (24 and N, for our dataset N will be 43).

Show that $L(C|\lambda)$ is a convex function of the coefficients λ on the domain $\lambda \in \mathbb{R}^{24}_{++}$.

(b) The log-likelihood function L can be minimized to find the maximum likelihood estimate for the λ coefficients. Formulate this constrained optimization problem in `CVX` and solve it for the crashes dataset, that is solve:

$$\min_{\lambda} L(C|\lambda)\lambda \in \mathbb{R}^{24}_{+}$$

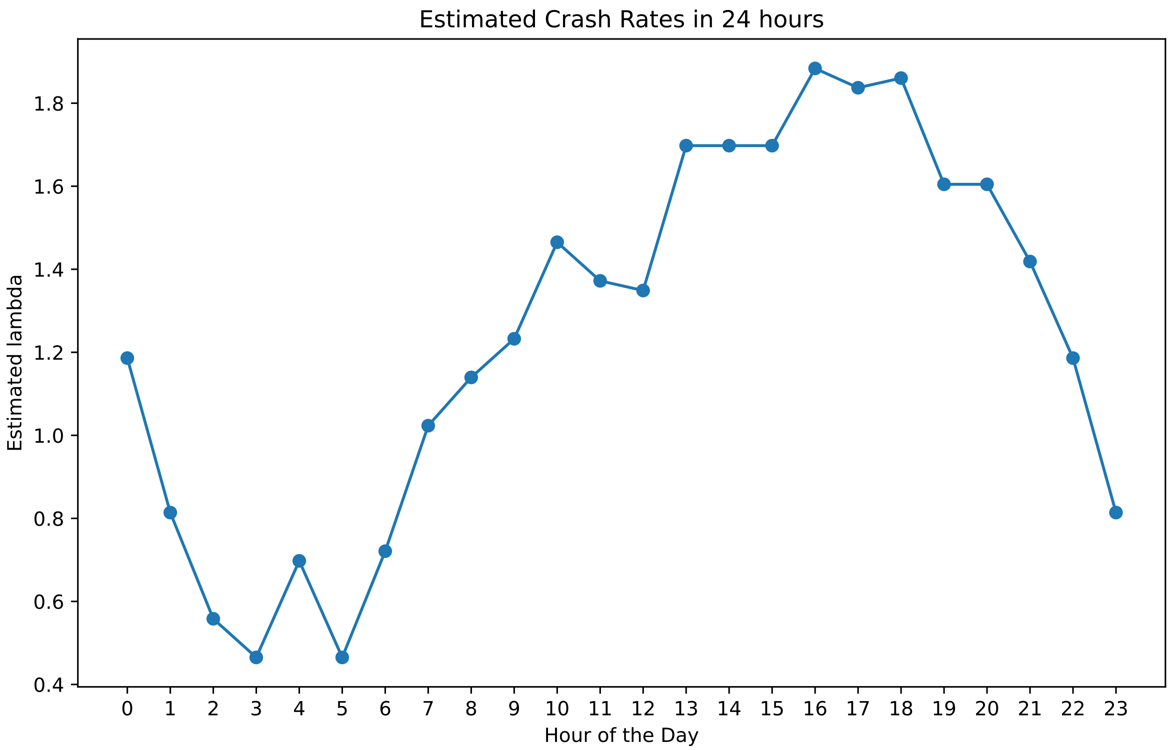
Make a plot of the λ coefficients

Problem 4(a) and (b) Solution

```
# load packages
import cvxpy as cp
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# load the dataset and extract hours and numbers of crashes
crash_url = 'https://media.githubusercontent.com/media/georgehagstrom/DATA609Spring2025/refs/
crash_data = pd.read_csv(crash_url)

# convert the data into matrix group by the date and hour
crash_matrix = crash_data.pivot_table(index='date', columns='hour', values='Num_Crashes', agg

# convert to a numpy matrix for further operations if needed
C = crash_matrix.to_numpy()

# get the shape
N, M = crash_matrix.shape

# define optimization variable (rates for each hour)
lambda_ = cp.Variable(M, pos=True)
```

```python
# define the log-likelihood function
L = N * cp.sum(lambda_) - cp.sum(C * cp.log(lambda_))

# create the optimization problem
problem = cp.Problem(cp.Minimize(L))

# solve the problem
problem.solve()

# Plot the lambda values (estimated crash rates)
plt.figure(figsize=(10, 6))
plt.plot(range(M), lambda_.value, marker='o')
plt.title('Estimated Crash Rates in 24 hours')
plt.xlabel('Hour of the Day')
plt.ylabel('Estimated lambda')
plt.xticks(range(M))
plt.show()
```

```
/Users/eddiexuexia/opt/anaconda3/lib/python3.9/site-packages/cvxpy/expressions/expression.py


This use of ``*`` has resulted in matrix multiplication.
Using ``*`` for matrix multiplication has been deprecated since CVXPY 1.1.
    Use ``*`` for matrix-scalar and vector-scalar multiplication.
    Use ``@`` for matrix-matrix and matrix-vector multiplication.
    Use ``multiply`` for elementwise multiplication.
This code path has been hit 1 times so far.
```



(c) It would be reasonable to expect that the rate of crashes in two adjacent hours should be similar, i.e. the observation that there has been a crash at 3:01PM should influence the estimate of the rate of crashes between 2:00PM and 3:00PM in addition to between 3:00PM and 4:00PM.

One way to implement this is through the following regularization term, which applies a penalty proportional to the square of the difference between λ_i and λ_{i+1}:

$$L_{pen}(C|\lambda, \rho) = L(C|\lambda) + \rho \left(\sum_{i=1}^{23} (\lambda_i - \lambda_{i-1})^2 + (\lambda_0 - \lambda_{23})^2 \right)$$

Assuming that $\rho > 0$, show that $L_{pen}(C|\lambda)$ is a convex function on $\mathbb{R}^{24}_{++}$

(d) Formulate the regularized maximum likelihood problem:

$$\min_{\lambda} L_{pen}(C|\lambda, \rho) \lambda \in \mathbb{R}^{24}_{+}$$

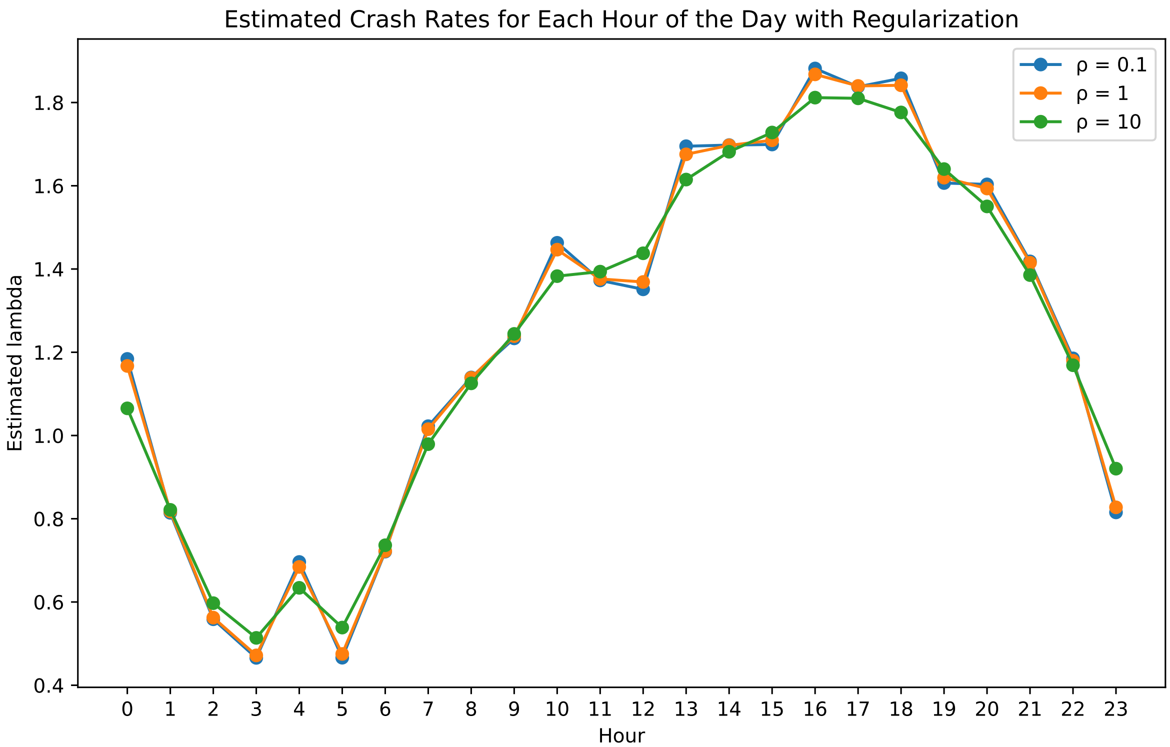
and solve it using `CVX`. Solve it for a range of positive values ρ such that for your smallest values the solution appears like your solution to (b) and for your largest values the λ show much less variation over time.

Problem 4(c) and (d) Solution

```
import cvxpy as cp
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# load the dataset and extract hours and numbers of crashes
crash_url = 'https://media.githubusercontent.com/media/georgehagstrom/DATA609Spring2025/refs/
crash_data = pd.read_csv(crash_url)

# convert the data into matrix group by the date and hour
crash_matrix = crash_data.pivot_table(index='date', columns='hour', values='Num_Crashes', agg

# convert to a numpy matrix for further operations if needed
C = crash_matrix.to_numpy()

# get the shape
N, M = crash_matrix.shape

# define optimization variable (rates for each hour)
lambda_ = cp.Variable(M, pos=True)

# define the regularization matrix S (differences between adjacent hours)
S = np.zeros((M, M))
for i in range(M-1):
    S[i, i] = 1
    S[i, i+1] = -1

# define the log-likeihood function
L = N * cp.sum(lambda_) - cp.sum(C * cp.log(lambda_))

# define the regularization with the constraint to be positive
rho = cp.Parameter(nonneg=True)
regularization = rho * cp.sum_squares(S @ lambda_)

# define the optimization problem with regularization
objective = cp.Minimize(L + regularization)
problem = cp.Problem(objective)

# solve the problem for every range of P
rho_values = [0.1, 1, 10]
```

```python
lambda_values = []

for r in rho_values:
    rho.value = r
    problem.solve()
    lambda_values.append(lambda_.value)

# plot the lamda values for each
plt.figure(figsize=(10, 6))
for i, r in enumerate(rho_values):
    plt.plot(range(M), lambda_values[i], marker='o', label=f'  = {r}')

plt.title('Estimated Crash Rates for Each Hour of the Day with Regularization')
plt.xlabel('Hour')
plt.ylabel('Estimated lambda')
plt.xticks(range(M))
plt.legend()
plt.show()
```

```
/Users/eddiexuexia/opt/anaconda3/lib/python3.9/site-packages/cvxpy/expressions/expression.py


This use of ``*`` has resulted in matrix multiplication.
Using ``*`` for matrix multiplication has been deprecated since CVXPY 1.1.
    Use ``*`` for matrix-scalar and vector-scalar multiplication.
    Use ``@`` for matrix-matrix and matrix-vector multiplication.
    Use ``multiply`` for elementwise multiplication.
This code path has been hit 2 times so far.
```



Hint: There are many ways to implement this penalty term in `cvx`. A way that you might find useful is to define a matrix S where $S_{ii} = 1$, $S_{i,i-1} = -1$ for $i > 1$ and $S_{1,24} = -1$, with $S = 0$ for all other entries:

Then $S\lambda = \left[(\lambda_1 - \lambda_{24}) \quad (\lambda_2 - \lambda_1) \quad \cdots \quad (\lambda_{24} - \lambda_{23})\right]^T$, and you can use `cvx.square` and `cvx.sum` to construct the objective.