EE 364a HW8

Yen-Yu Chang
(discuss with Cheng-Min Chiang)

March 6, 2020

A8.7 Let $f(x) = (1/2)x^T x + \log \sum_{i=1}^m \exp(a_i^T x + b_i)$, then we have the first derivative and Hessian as

$$\nabla f(x) = x + \frac{A^T z}{\sum_{i=1}^m \exp(a_i^T x + b_i)}$$

$$\nabla^2 f(x) = H = I + A^T(\mathbf{diag}(z) - zz^T)A.$$

where

$$z_i = \frac{\exp(a_i^T x + b_i)}{\sum_{i=1}^m \exp(a_i^T x + b_i)}$$

Thus, the Newton equation is

$$\left(I + A^T(\mathbf{diag}(z) - zz^T)A\right)\Delta x_{nt} = -\left(x + \frac{A^T z}{\sum_{i=1}^m \exp(a_i^T x + b_i)}\right)$$

And we know that

$$\mathbf{diag}(z) - zz^T = (\mathbf{diag}(z) - zz^T)\mathbf{diag}(z)^{-1}(\mathbf{diag}(z) - zz^T)^T$$

Let set $L = \mathbf{diag}(z) - zz^T$, then we have

$$\left(I + A^T(\mathbf{diag}(z) - zz^T)A\right)\Delta x = \left(I + A^T L\mathbf{diag}(z)L^T A\right)\delta x = -\nabla f$$

which is equivalent to

$$\begin{bmatrix} I & A^T L \\ L^T A & -\mathbf{diag}(z) \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta u \end{bmatrix} = \begin{bmatrix} -\nabla f \\ 0 \end{bmatrix}$$

Now apply block elimination to solve

$$(\mathbf{diag}(z) + L^T A A^T L)\delta u = L^T A(-\nabla f)$$

The cost is about $m^2 n + (1/3)m^3$ flops.

A9.5

(a) Let $f(x) = c^T x - \sum_{i=1}^n \log x_i$, so the Newton step δx_{nt} is defined by the KKT system as

$$\begin{bmatrix} H & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} \Delta x_{nt} \\ w \end{bmatrix} = \begin{bmatrix} -g \\ 0 \end{bmatrix}$$

where $H = \mathbf{diag}(1/x_1^2, \ldots, 1/x_n^2)$ and $g = \nabla f(x) = c - (1/x_1, \ldots, 1/x_n)$. The KKT system can be solved efficiently by block elimination

$$AH^{-1}A^T w = -AH^{-1}g$$

and setting $\Delta x_{nt} = -H^{-1}(A^T w + g)$. The KKT optimality condition is

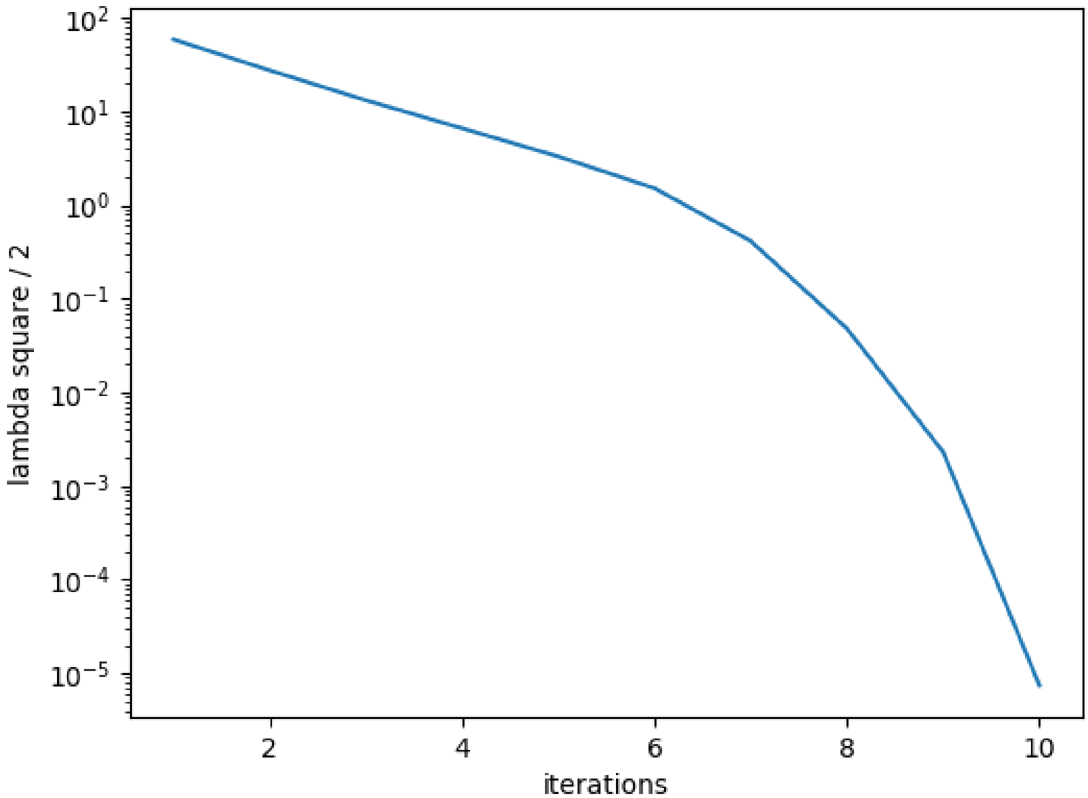
$$A^T \nu^* + c - (1/x_1^*, \ldots, 1/x_n^*) = 0.$$

When the Newton method converges, which means that $\Delta x_{nt} \approx 0$ and w is the dual optimal point ν^*.

The following Python code solves the problem

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

def lp_acent(A, b, c, x_0):
x_0 = x_0.reshape(len(x_0), 1)
b = b.reshape(len(b), 1)
c = c.reshape(len(c), 1)
alpha = 0.01
beta = 0.5
epsilon = 1e-3
max_iters = 100

lambda_hist = np.array([])
A = np.matrix(A)

if min(x_0) <= 0 or np.linalg.norm(np.dot(A, x_0) - b) > epsilon:
print("Error: x_0 is not feasible.")
return np.array([]), np.array([]), lambda_hist

m = b.size
n = x_0.size
x = x_0
```

```
for iterNum in range(max_iters):
H = np.diag(1 / np.power(x.reshape(n), 2))
g = c - 1 / x

X = np.diag(x.reshape(n) ** 2)
w = np.linalg.lstsq(A * X * A.T, -A * X * g)[0]
dx = -X * (A.T * w + g)

lambdasqr = -np.dot(g.reshape(n), dx.reshape(n).T)
lambda_hist = np.append(lambda_hist, lambdasqr / 2)

if lambdasqr / 2 <= epsilon:
return x, w, lambda_hist

t = 1
while min(x + t * dx) <= 0:
t *= beta
while t * np.dot(c.reshape(n), dx.reshape(n).T)\
- np.sum(np.log(x.reshape(n) + t * dx.reshape(n)))\
+ np.sum(np.log(x.reshape(n)))\
- alpha * t * np.dot(g.T, dx) > 0:
t *= beta
x += t * dx
print("Error: max_iters reached")
return np.array([]), np.array([]), lambda_hist

if __name__ == '__main__':

A = np.random.rand(100, 500)
assert np.linalg.matrix_rank(A) == 100
x_0 = np.abs(np.random.rand(500))
b = A @ x_0
c = np.random.rand(500)

x_star, nu_star, lambda_hist = lp_acent(A, b, c, x_0)

plt.figure()
plt.plot(np.arange(len(lambda_hist)) + 1, lambda_hist)
plt.xlabel('iterations')
plt.ylabel('lambda square / 2')
plt.yscale('log')
plt.savefig('A9.5a.png')
```



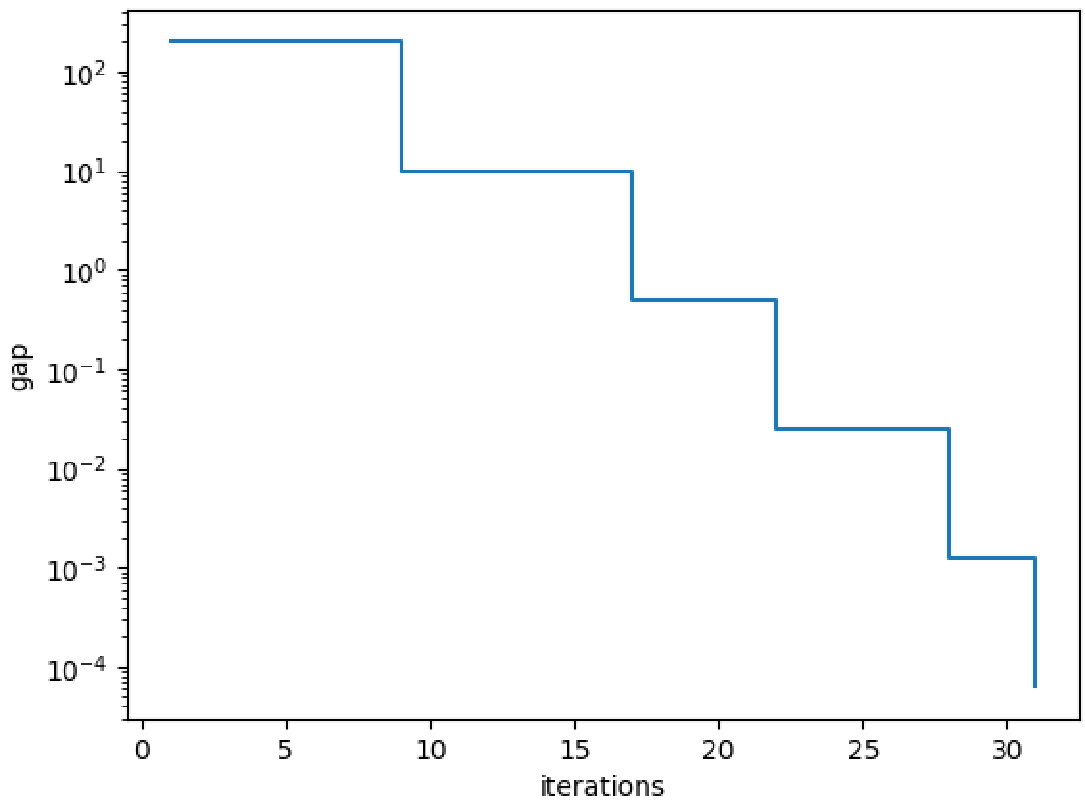
The random data is generated as given problem statement, with $A \in \mathbf{R}^{100\times 500}$. We can observe the quadratic convergence optimality.

(b) The following Python code solves the problem

```
import numpy as np
import cvxpy as cp
import matplotlib.pyplot as plt
from A9_5a import *

def lp_barrier(A, b, c, x_0):
  T_0 = 1
  mu = 20
  epsilon = 1e-3
  n = len(x_0)
  t = T_0
  x = x_0
  num_newton_steps = list()
```

```
  duality_gaps = list()
  gap = float(n) / t
  while True:
    x_star, nu_star, lambda_hist = lp_acent(A, b, t * c, x)
    if len(x_star) == 0:
      return np.array([]), gap, num_newton_steps, duality_gaps
    x = x_star
    gap = float(n) / t
    num_newton_steps.append(len(lambda_hist))
    duality_gaps.append(gap)
    if gap < epsilon:
      return x_star, gap, num_newton_steps, duality_gaps
    t *= mu

if __name__ == '__main__':
  m = 10
  n = 200
  np.random.seed(2)
  A = np.vstack((np.random.randn(m - 1, n), np.ones((1, n))))
  A = np.matrix(A)
  x_0 = np.random.rand(n, 1) + 0.1
  b = A * x_0
  c = np.random.randn(n, 1)

  x_star, nu_star, lambda_hist = lp_acent(A, b, c, x_0)
#plt.semilogy(range(1, len(lambda_hist) + 1), lambda_hist)
#plt.show()

  x_star, gap, num_newton_steps, duality_gaps\
    = lp_barrier(A, b, c, x_0)

  plt.figure()
  plt.step(np.cumsum(num_newton_steps), duality_gaps, where='post')
  plt.yscale('log')
  plt.xlabel('iterations')
  plt.ylabel('gap')
  plt.savefig('A9.5b.png')

  x = cp.Variable((n, 1))
  obj = cp.Minimize(c.T @ x)
  prob = cp.Problem(obj, [A * x == b, x >= 0])
  prob.solve()
```

```
print("Optimal value from barrier method: {}"\
  .format(np.dot(c.reshape(n), x_star.reshape(n))))
print("Optimal value from CVXPY: {}".format(prob.value))
print("Dualty gap from barrier method: {}".format(gap))
```



The optimal value from barrier method is -220.9224510, and the optimal value from CVXPY is -200.9225005. The duality from barrier method is 6.25×10^{-5}.

(c) The following Python code solves the problem

```
import numpy as np
import cvxpy as cp
import matplotlib.pyplot as plt
from A9_5a import *
from A9_5b import *

def lp_solve(A, b, c):
```

```
  m, n = A.shape
  b = b.reshape(m, 1)
  nsteps = np.zeros(2)
  x0 = np.linalg.lstsq(A, b)[0]
  t0 = 2 + max(0, -min(x0))

  A1 = np.hstack((A, -np.dot(A, np.ones(n)).reshape(m, 1)))
  b1 = b - np.dot(A, np.ones(n)).reshape(m, 1)
  z0 = x0.reshape(n, 1) + t0 * np.ones((n, 1)) - np.ones((n, 1))
  c1 = np.vstack((z0, t0)).reshape(n + 1, 1)
  x_0 = np.vstack((z0, t0)).reshape(n + 1, 1)
  z_star, gap, num_newton_steps, duality_gaps =\
    lp_barrier(A1, b1, c1, x_0)
  nsteps[0] = sum(num_newton_steps)
  if len(z_star) == 0:
    print("Phase I: problem is infeasible.")
    return np.array([]), np.inf, np.inf, "Infeasible", nsteps

  print("Phase I: feasible point found.")
  x_0 = z_star[:n] - z_star[n][0] * np.ones((n, 1)) + np.ones((n, 1))

  x_star, gap, num_newton_steps, duality_gaps =\
    lp_barrier(A, b, c, x_0)
  nsteps[1] = sum(num_newton_steps)
  if len(x_star) == 0:
    return np.array([]), np.inf, np.inf, "Infeasible", nsteps

  p_star = np.dot(c.reshape(len(c)), x_star.reshape(len(c)))
  return x_star, p_star, gap, "Optimal", nsteps

if __name__ == '__main__':
  m = 100
  n = 500

  #Infeasible problem
  A = np.vstack((np.random.randn(m - 1, n), np.ones((1, n))))
  b = np.random.randn(m, 1)
  c = np.random.randn(n, 1)
  x_star, p_star, gap, status, nsteps = lp_solve(A, b, c)

  # Compare to CVXPY
  x = cp.Variable((n, 1))
```

```
obj = cp.Minimize(c.T @ x)
prob = cp.Problem(obj, [A * x == b, x >= 0])
prob.solve()

print("Status from lp solver: {}".format(status))
print("Status from CVXPY: {}".format(prob.status))

# Feasible problem
A = np.vstack((np.random.randn(m - 1, n), np.ones((1, n))))
v = np.random.rand(n) + 0.1
b = np.dot(A, v)
c = np.random.randn(n)
x_star, p_star, gap, status, nsteps = lp_solve(A, b, c)


# Compare to CVXPY
x = cp.Variable((n, 1))
obj = cp.Minimize(c.T @ x)
prob = cp.Problem(obj, [A * x == b.reshape((-1, 1)), x >= 0])
prob.solve()
print("Optimal value from barrier method: {}"\
  .format(np.dot(c.reshape(n), x_star.reshape(n))))
print("Optimal value from CVXPY: {}".format(prob.value))
print("Duality gap from barrier method: {}".format(gap))
```

The random data is generated as given problem statement, with $A \in \mathbf{R}^{100\times 500}$. Then we have the optimal value from barrier method -377.493947, and the optimal value from CVXPY $= 377.5099092$. The duality gap from barrier method is 0.00015625.

A9.12

(a) Let

$$\phi(\alpha) = \psi(\hat{t} + \alpha u, \hat{x} + \alpha v) = -\log\left(\hat{t} + \alpha u - f(\hat{x} + \alpha v)\right) - \sum_{i=1}^{n} \log(\hat{x}_i + \alpha v_i)$$

Then we have

$$\phi'(\alpha) = \frac{f'(\hat{x} + \alpha v)v - u}{\hat{t} + \alpha u - f(\hat{x} + \alpha v)} - \sum_{i=1}^{m} \frac{v_i}{\hat{x}_i + \alpha v_i}$$

$$\phi''(\alpha) = \frac{f''(\hat{x} + \alpha v)v^2}{\hat{t} + \alpha u - f(\hat{x} + \alpha v)} + \left(\frac{f'(\hat{x} + \alpha v)v - u}{\hat{t} + \alpha u - f(\hat{x} + \alpha v)}\right)^2 - \sum_{i=1}^{n} \frac{-v_i^2}{(\hat{x}_i + \alpha v_i)^2}$$

$$\phi'''(\alpha) = \frac{f'''(\hat{x}+\alpha v)v^3}{\hat{t}+\alpha u - f(\hat{x}+\alpha v)} - \frac{f''(\hat{x}+\alpha v)v^2\left(u - f'(\hat{x}+\alpha v)\right)}{\left(\hat{t}+\alpha u - f(\hat{x}+\alpha v)\right)^2}$$
$$+ 2\frac{f''(\hat{x}+\alpha v)v^2\left(f'(\hat{x}+\alpha v)v - u)\right)}{\left(\hat{t}+\alpha u - f(\hat{x}+\alpha v)\right)^2} + 2\left(\frac{f'(\hat{x}+\alpha v)v - u}{\hat{t}+\alpha u - f(\hat{x}+\alpha v)}\right)^3$$
$$- 2\sum_{i=1}^{n}\frac{v_i^3}{(\hat{x}_i+\alpha v_i)^3}$$

By definition,

$$\phi''(0) = v^T\nabla^2\psi(y)v = \left(\frac{f'(\hat{x})v-u}{f(\hat{x})-\hat{t}}\right)^2 - \frac{f''(\hat{x})v^2}{f(\hat{t})} + \sum_{i=1}^{n}\frac{v_i^2}{\hat{x_i^2}}$$

$$\begin{aligned}\phi'''(0) &= \nabla^3\psi(y)[v,v,v] \\ &= \frac{-f'''(\hat{x})v^3}{f(\hat{x})-\hat{t}} - 2\left(\frac{f'(\hat{x})v-u}{f(\hat{x})-\hat{t}}\right)^3 + 3\left(\frac{f''(\hat{x})v^2(f'(\hat{x})v-u)}{\left(f(\hat{x})-\hat{t}\right)^2}\right) - 2\sum_{i=1}^{n}\frac{v_i^3}{\hat{x_i^3}}\end{aligned}$$

We have

$$\begin{aligned}|\phi'''(0)| &\le \frac{|f'''(\hat{x})v^3|}{-\left(f(\hat{x})-\hat{t}\right)} + 2\left(\frac{|f'(\hat{x}v-u)|}{-\left(f(\hat{x})-\hat{t}\right)}\right)^3 + 3\frac{f''(\hat{x})v^2|f'(\hat{x})v-u|}{\left(f(\hat{x})-\hat{t}\right)^2} + 2\sum_{i=1}^{n}\frac{v_i^3}{\hat{x_i^3}} \\ &\le \frac{3f''(\hat{x})v^2\sqrt{\sum_{i=1}^{n}\frac{v_i^2}{\hat{x_i}^2}}}{-\left(f(\hat{x})-\hat{t}\right)} + 2\left(\frac{|f'(\hat{x}v-u)|}{-\left(f(\hat{x})-\hat{t}\right)}\right)^3 + 3\frac{f''(\hat{x})v^2|f'(\hat{x})v-u|}{\left(f(\hat{x})-\hat{t}\right)^2} + 2\sum_{i=1}^{n}\frac{v_i^3}{\hat{x_i^3}}\end{aligned}$$

We will show that

$$\frac{3f''(\hat{x})v^2\sqrt{\sum_{i=1}^{n}\frac{v_i^2}{\hat{x_i}^2}}}{-\left(f(\hat{x})-\hat{t}\right)} + 2\left(\frac{|f'(\hat{x}v-u)|}{-\left(f(\hat{x})-\hat{t}\right)}\right)^3 + 3\frac{f''(\hat{x})v^2|f'(\hat{x})v-u|}{\left(f(\hat{x})-\hat{t}\right)^2} + 2\sum_{i=1}^{n}\frac{v_i^3}{\hat{x_i^3}}$$
$$\le \left((\frac{f'(\hat{x})v-u}{f(\hat{x})-\hat{t}})^2 - \frac{f''(\hat{x})v^2}{f(\hat{x})-\hat{t}} + \sum_{i=1}^{n}\frac{v_i^2}{\hat{x_i}^2}\right)^{\frac{3}{2}}$$

To simplify the formulas, we define

$$a = \frac{\left(-\frac{f'(\hat{x})v^2}{f(\hat{x})-\hat{t}}\right)^{\frac{1}{2}}}{\left((\frac{f'(\hat{x})v-u}{f(\hat{x})-\hat{t}})^2 - \frac{f''(\hat{x})v^2}{f(\hat{x})-\hat{t}} + \sum_{i=1}^{n}\frac{v_i^2}{\hat{x_i}^2}\right)^{\frac{1}{2}}}$$

$$b = \frac{-\frac{|f'(\hat{x})v-u|}{f(\hat{x})-\hat{t}}}{\left((\frac{f'(\hat{x})v-u}{f(\hat{x})-\hat{t}})^2 - \frac{f''(\hat{x})v^2}{f(\hat{x})-\hat{t}} + \sum_{i=1}^n \frac{v_i^2}{\hat{x_i}^2}\right)^{\frac{1}{2}}}$$

$$c = \frac{\sqrt{\sum_{i=1}^n \frac{v_i^2}{\hat{x_i}^2}}}{\left((\frac{f'(\hat{x})v-u}{f(\hat{x})-\hat{t}})^2 - \frac{f''(\hat{x})v^2}{f(\hat{x})-\hat{t}} + \sum_{i=1}^n \frac{v_i^2}{\hat{x_i}^2}\right)^{\frac{1}{2}}}$$

The inequality that we want to prove reduces to the inequality

$$\frac{3}{2}ca^2 + b^3 + \frac{3}{2}a^2b + c^3 \leq 1.$$

Using the hint in the problem, we have $a^2 + b^2 + c^2 = 1$ so we have the inequality $\frac{3}{2}a^2c + b^3 + c^3 + \frac{3}{2}a^2b \leq 1$. Therefore, we prove the inequality

$$\begin{aligned} &|\phi'''(0)| \leq 2\left(\phi''(0)\right)^{3/2} \\ \iff &\left(\nabla^3\psi(y)[v,v,v]\right)^2 \leq 2\left(v^T\nabla^2\psi(y)v\right)^3. \end{aligned}$$

(b) Let $g(t) = f(\hat{y}+tu, \hat{z}+tv) = (\hat{y}+tu)\log\frac{\hat{y}+tu}{\hat{z}+tv}$. Then we have

$$g''(0) = \frac{(u\hat{z}-v\hat{y})^2}{\hat{y}\hat{z}^2}$$

$$g'''(0) = \frac{-(u\hat{z}-v\hat{y})^2(u\hat{z}+2v\hat{y})}{\hat{y}^2\hat{z}^3}$$

$$\sqrt{\sum_{i=1}^n \frac{v_i^2}{x_i^2}} = \sqrt{\frac{u^2}{\hat{y}^2} + \frac{v^2}{\hat{z}^2}} = \sqrt{\frac{u^2\hat{z}^2 + v^2\hat{y}^2}{\hat{y}^2\hat{z}^2}}$$

The left side of the inequality (42) becomes

$$|\nabla^3 f(x)[v,v,v]| = |g'''(0)| = \frac{(u\hat{z}-v\hat{y})^2(u\hat{z}+2v\hat{y})}{\hat{y}^2\hat{z}^3}$$

The right side of the inequality (42) becomes

$$3v^T\nabla^2 f(x)v\sqrt{\sum_{i=1}^n \frac{v_i^2}{x_i^2}} = 3g''(0)\sqrt{\sum_{i=1}^n \frac{v_i^2}{x_i^2}} = \frac{3(u\hat{z}-v\hat{y})^2\sqrt{u^2\hat{z}^2+v^2\hat{y}^2}}{\hat{y}^2\hat{z}^3}$$

Since

$$(3\sqrt{u^2\hat{z}^2+v^2\hat{y}^2})^2 - (u\hat{z}+2v\hat{y})^2 = 8(u\hat{z}-\frac{1}{4}v\hat{y})^2 + \frac{9}{2}v^2\hat{y}^2 \geq 0$$

We have

$$\frac{(u\hat{z}-v\hat{y})^2(u\hat{z}+2v\hat{y})}{\hat{y}^2\hat{z}^3} \leq \frac{3(u\hat{z}-v\hat{y})^2\sqrt{u^2\hat{z}^2+v^2\hat{y}^2}}{\hat{y}^2\hat{z}^3}$$

Therefore, we prove that the relative entropy satisfies

$$|\nabla^3 f(x)[v,v,v]| \leq 3v^T\nabla^2 f(x)v\sqrt{\sum_{i=1}^n \frac{v_i^2}{x_i^2}}.$$

(c) Let the t_{old} in (a) as $t_{old} = x$, and let the x_{old} in (a) as $x_{old} = (y,z)$. Then we have $f(x_{old}) = f(y,z) = y\log\frac{y}{z}$. Since $f(y,z)$ satisfies the inequality (42), $K = \{(x,y,z)|ye^{x/y} < z, y > 0\}$, and $\psi(t_{old}, f(x_{old})$ is self-concordant, we have

$$\psi(t_{old}, x_{old}) = \psi(x,y,z) = -\log y - \log z - \log(y\log\frac{z}{y} - x)$$

is self-concordant on the perspective-transformed exponential cone K.

And we calculate

$$\begin{aligned}\psi(s(x,y,z)) &= -\log sy - \log sz - \log(sy\frac{sz}{sy} - sx)\\ &= -\log s - \log y - \log s - \log z - \log s - \log(y\log\frac{z}{y} - x)\\ &\psi(x,y,z) - 3\log s\end{aligned}$$

for $s > 0$. Hence, ψ is a generalized logarithm of degree 3.

A18.18

(a) $K_{1,n}^* = \{Y \in \mathbf{S}^n | Y_{ii} \geq 0 \text{ for all i, } Y_{ij} = 0 \text{ for all } i \neq j\}$

(b) By the hint, we can represent $K_{2,n}$ as

$$K_{2,n} = \bigcap_{i\neq j}\left\{X \in \mathbf{S}^n \middle| \begin{bmatrix} X_{i,i} & X_{i,j} \\ X_{i,j} & X_{j,j}\end{bmatrix} \succeq 0\right\} = \bigcap_{i\neq j} L_{i,j}$$

By the hint, $K_{2,n}^* = \sum L_{i,j}^*$, so we can first find the characterization of $L_{i,j}^*$. If $Y \in H_{i,j}^*$, then for all $Y_{i',j'}$ such that $(i',j') \notin \{(i,i),(i,j),(j,i),(j,j)\}$, $Y_{i',j'} = 0$. Otherwise, we can construct $Y_{i',j'} = -|C|X_{i',j'}$, then the $\mathbf{tr}(Y^TX) \leq 0$ when $|C|$ is big enough. Hence, we have

$$\mathbf{tr}(Y^TX) = \mathbf{tr}\left(\begin{bmatrix} Y_{i,i} & Y_{i,j} \\ Y_{j,i} & Y_{j,j}\end{bmatrix}^T \begin{bmatrix} X_{i,i} & X_{i,j} \\ X_{j,i} & X_{j,j}\end{bmatrix}\right)$$

And the trace is greater than or equal to zero for all X if and only if

$$\begin{bmatrix} Y_{i,i} & Y_{i,j} \\ Y_{j,i} & Y_{j,j} \end{bmatrix}^T \in (\mathbf{S}_+^2)^*$$

We know that $(\mathbf{S}_+^2)^* = \mathbf{S}_+^2$ Thus, we conclude that

$$L_{i,j}^* = \left\{ Y \in \mathbf{S}^n \middle| \begin{bmatrix} Y_{i,i} & Y_{i,j} \\ Y_{i,j} & Y_{j,j} \end{bmatrix} \succeq 0 \text{ and for others, } Y_{i',j'} = 0 \right\}$$

and $K_{2,n}^* = \sum_{i \neq j} L_{i,j}^*$

(c) The following Python code solves the problem

```
import numpy as np
import cvxpy as cp
from psd_cone_approx_data import *

# K = K_{1,n}
X = cp.Variable((n, n), symmetric = True)
obj = cp.Minimize(cp.trace(C @ X))
constraints = [cp.trace(A @ X) == b, cp.diag(X) >= 0]
problem = cp.Problem(obj, constraints)
problem.solve(solver = cp.SCS)
print("The optimal value of K = K_1,n is: {}".format(problem.value))

# K = K_{2,n}
X = cp.Variable((n, n), symmetric = True)
obj = cp.Minimize(cp.trace(C @ X))
constraints = [cp.trace(A @ X) == b, cp.diag(X) >= 0]
for i in range(n):
  for j in range(i + 1, n):
    constraints += [X[[i,j],:][:,[i,j]] >> 0]
problem = cp.Problem(obj, constraints)
problem.solve(solver = cp.SCS)
print("The optimal value of K = K_2,n is: {}".format(problem.value))

# K = S_+^n
X = cp.Variable((n, n), symmetric = True)
obj = cp.Minimize(cp.trace(C @ X))
constraints = [cp.trace(A @ X) == b, X >> 0]
problem = cp.Problem(obj, constraints)
problem.solve(solver = cp.SCS)
```

```
print("The optimal value of K = S_+^n is: {}".format(problem.value))

# K = S_{2,n}^*
X_list = []
for i in range(int(n * (n - 1) / 2)):
  X_list.append(cp.Variable((n, n), symmetric = True))
X = X_list[0]
for i in range(1, 10):
  X += X_list[i]
obj = cp.Minimize(cp.trace(C @ X))
constraints = [cp.trace(A @ X) == b]
for i in range(n - 1):
  for j in range(i + 1, n):
    idx = -1
    if i == 0:
      idx = i + j - 1
    elif i == 1:
      idx = i + j + 1
    else:
      idx = i + j + 2

    constraints += [X_list[idx][[i,j],:][:,[i,j]] >> 0]
    for k in range(n):
      for l in range(n):
        if (k, l) not in ((i, i), (i, j), (j, i), (j, j)) :
          constraints += [X_list[idx][k, l] == 0]
problem = cp.Problem(obj, constraints)
problem.solve(solver = cp.SCS)
print("The optimal value of K = S^*_2,n is: {}".format(problem.value))


# K = S_{1,n}^*
X = cp.Variable((n, n), symmetric = True)
obj = cp.Minimize(cp.trace(C @ X))
constraints = [cp.trace(A @ X) == b, cp.diag(X) >= 0]
for i in range(n):
  for j in range(i + 1, n):
    constraints += [X[i, j] == 0]
    constraints += [X[j, i] == 0]
problem = cp.Problem(obj, constraints)
problem.solve(solver = cp.SCS)
print("The optimal value of K = S^*_1,n is: {}".format(problem.value))
```

The optimal value of $K = K_{1,n}$ is: 3.2492139325921034e-14 The optimal value of $K = K_{2,n}$ is: 2.7427091597562647 The optimal value of $K = S_+^n$ is: 3.0353678389728045 The optimal value of $K = S_{2,n}^*$ is: 5.090749880284002 The optimal value of $K = S_{1,n}^*$ is: 9.1791363785143