Problem 1

Derive the dual of the equivalent problem

$$\text{minimize} \quad \sum_{i=1}^{m} h(\|y_i\|_2) - c^T x$$

$$\text{subject to} \quad A_i x + b_i - y_i = 0, \quad i = 1 \cdots m$$

with variables $x \in \mathbb{R}^n$, $y_i \in \mathbb{R}^3$. and $h(u) = \begin{cases} (u-1)^2/2, & u \geq 1 \\ 0 & \text{otherwise} \end{cases}$

$$L(x, y_i, \lambda_i) = \left[\sum_{i=1}^{m} (h\|y_i\|_2) - c^T x \right] + \sum_{i=1}^{m} \lambda^T (A_i x + b_i - y_i)$$

$$g(\lambda_i) = \inf_{y_i} \left(\sum_{i=1}^{m} [h(\|y_i\|_2) - \lambda_i y_i) \right) + \inf_{x} \left(\sum_{i=1}^{m} -c^T x + \lambda_i (A_i x) \right)$$

$$+ \sum_{i=1}^{m} \lambda_i^T b_i$$

Change to sup

$$\sup_{y_i} \left(\sum_{i=1}^{m} \lambda_i^T y_i - h(\|y_i\|_2) \right)$$

Check $\|y_i\|_2 \geq 1$

Using Cauchy-Schwarz and dropping the sum

$$\Rightarrow \lambda_i^T y_i - \frac{1}{2}(\|y_i\| - 1)^2 \leq \underbrace{\|\lambda_i\|_2}_{a} \underbrace{\|y_i\|_2}_{x} - \frac{1}{2}(\|y_i\| - 1)^2$$

$$\Rightarrow \nabla_x \, ax - \frac{1}{2}(x-1)^2 = 0$$

$$\Rightarrow \quad a - x - 1 = 0$$

$$x = a + 1$$

hence, $\|y_i\|_2 = \|\lambda_i\|_2 + 1$. ← plug back in

We then have,

$$\|\lambda_i\|_2(\|\lambda_i\|_2+1) - \frac{1}{2}(\|\lambda_i\|+1-1)^2$$

$$= \|\lambda_i\|_2(\|\lambda_i\|_2+1) - \frac{1}{2}\|\lambda_i\|_2^2$$

$$= \frac{1}{2}\|\lambda_i\|_2^2 + \|\lambda_i\|_2 \quad \leftarrow \text{objective}$$

Now check $\|y_i\|_2 \leq 1$

$$\lambda_i^T y_i - h(\|y_i\|_2)$$

$$\lambda_i^T y_i \leq \|\lambda_i\|_2 \underbrace{\|y_i\|_2}_{\leq 1}$$

$$\leq \|\lambda_i\|_2$$

then $\max\{\|\lambda_i\|+1, \quad\} = \underline{\|\lambda_i\|+1}$.

Now we look at

$$\inf_x \left(\sum_{i=1}^m \lambda_i^T(A_i x) - c^T x\right)$$

$$= \inf_x \sum_{i=1}^m (\lambda_i^T A_i)x - c^T x$$

$$= \inf_x \left[\sum_{i=1}^m A_i^T \lambda_i - c\right)x$$

$$= \begin{cases} 0 & \text{if } \sum_{i=1}^m A_i^T \lambda_i = c \\ -\infty & \text{otherwise} \end{cases}$$

Therefore the dual problem can be written as

$$\max \quad \frac{1}{2}\|\lambda_i\|_2^2 + \|\lambda_i\|_2$$

$$\text{subject to} \quad \sum_{i=1}^{m} A_i^T \lambda_i = C$$

$$\lambda_i \geq 0$$

Problem 2

Fix $S \in S_+^n$ and $\lambda > 0$. Consider the following optimization problem:

$$\text{maximize} \quad \log\det X - \mathrm{Tr}(SX) - \lambda \textstyle\sum_{ij} t_{ij}$$
$$\text{subject to} \quad |X_{ij}| \leq t_{ij}$$

where domain of logdet is S_{++}^n. Show that if the optimal solution is a block diagnol matrix, then $|S_{ij}| \leq \lambda$ whenever the (i,i) entry and (j,j) entry are in two different blocks.

First change optimization problem

$$\min \; -\log\det + \mathrm{Tr}(SX) + \lambda \sum_{ij} t_{ij}$$

$$\text{s.t.} \; |X_{i,j}| \leq t_{ij} \rightarrow -t_{ij} \leq X_{ij} \leq t_{ij}$$

$$X_{ij} - t_{ij} \leq 0$$
$$-X_{ij} - t_{ij} \leq 0$$

$$L(\mu, \gamma) = -\log\det X + \mathrm{Tr}(SX) + \lambda \sum_{ij} t_{ij} + \sum_{ij} \mu_{ij}(X_{ij} - t_{ij}) + \sum_{ij} \gamma_{ij}(-X_{ij} - t_{ij})$$

$$\nabla_X L = -X^{-1} + S + M - \Gamma = 0$$

$$\nabla_\Sigma L = \lambda 11^T - M - \Gamma = 0$$

given: $X^{-1} = \begin{bmatrix} X_1^{-1} & 0 \\ 0 & X_2^{-1} \end{bmatrix}$ off-diagonals $X^{-1}_{ij} = 0$

$$\Rightarrow \cancel{-X^{-1}_{ij}}^{\,0} + s_{ij} + \mu_{ij} - \gamma_{ij} = 0$$

$$\Rightarrow \underline{s_{ij} = \gamma_{ij} - \mu_{ij}}$$

we want $|s_{ij}| \leq \lambda$

$$\lambda - \mu_{ij} - \gamma_{ij} = 0$$

$$\Rightarrow \underline{\lambda = \mu_{ij} + \gamma_{ij}}$$

apply absolute value

$$|s_{ij}| = |\gamma_{ij} - \mu_{ij}|$$

Two cases

$\gamma_{ij} - \mu_{ij}$, $\gamma \geq \mu$

$-(\gamma_{ij} - \mu_{ij})$, $\gamma \leq \mu$

first case where $\gamma \geq \mu$

$$|s_{ij}| = \gamma_{ij} - \mu_{ij}$$

$$|s_{ij}| + \mu_{ij} = \gamma_{ij}$$ ← plug into other equation

$$\Rightarrow \lambda = \underbrace{2\mu_{ij}}_{\geq 0} + |s_{ij}|$$

$$\lambda \geq |s_{ij}|$$

Second case

$$|s_{ij}| = -(\gamma_{ij} - \mu_{ij}) \quad \text{where} \quad \gamma \leq \mu$$

$$|s_{ij}| = \gamma_{ij} + \mu_{ij}$$

$$\mu_{ij} = |s_{ij}| + \gamma_{ij}$$

$$\Rightarrow \lambda = \underbrace{2\gamma_{ij}}_{\geq 0} + |s_{ij}|$$

$$\lambda \geq |s_{ij}| \quad . \quad \text{QED.}$$

Problem 3

(a) Express the problem as an SDP

We have $A = \begin{bmatrix} 50 & 6 & ? & 4 & ? \\ 6 & ? & ? & ? & -15 \\ ? & ? & 46 & ? & 17 \\ 4 & ? & ? & ? & -13 \\ ? & -15 & 17 & -13 & 70 \end{bmatrix}$

From page 169 in the example we can formulate this as a matrix norm minization.

Let $A(x) = A_0 + x_1 A_1 + \dots + x_n A_n$ where $A_i \in \mathbb{R}^{p \times q}$.

We will have A_0 and A_i where $i = 1, \dots, 7$, defined as

$$A_0 = \begin{bmatrix} 50 & 6 & 0 & 4 & 0 \\ 6 & 0 & 0 & 0 & -15 \\ 0 & 0 & 46 & 0 & 17 \\ 4 & 0 & 0 & 0 & -13 \\ 0 & -15 & 17 & -13 & 70 \end{bmatrix}$$

and $A_1 = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$, $A_2 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$

$\dots$ $A_7 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$.

filling in the diagnol elements.

Therefore by example in the book our SDP formulation is

$$\begin{aligned} &\text{minimize} && t \\ &\text{subject to} && \begin{bmatrix} tI & A(x) \\ A(x)^T & tI \end{bmatrix} \succeq 0 \end{aligned}$$

with variables x and t.

[48]:
```
from cvxpy import norm, Variable, Minimize, Problem
import numpy as np

A_0 = np.array([
    [50, 6, 0, 4, 0],
    [6, 0, 0, 0, -15],
    [0, 0, 46, 0, 17],
    [4, 0, 0, 0, -13],
    [0, -15, 17, -13, 70]
])

A_1 = np.zeros((5, 5))
A_1[0, 2] = 1
A_1[2, 0] = 1

A_2 = np.zeros((5, 5))
A_2[1, 1] = 1

A_3 = np.zeros((5, 5))
A_3[0, 4] = 1
A_3[4, 0] = 1

A_4 = np.zeros((5, 5))
A_4[1, 2] = 1
A_4[2, 1] = 1

A_5 = np.zeros((5, 5))
A_5[1, 3] = 1
A_5[3, 1] = 1

A_6 = np.zeros((5, 5))
A_6[2, 3] = 1
A_6[3, 2] = 1

A_7 = np.zeros((5, 5))
A_7[3, 3] = 1

for i, matrix in enumerate([A_0, A_1, A_2, A_3, A_4, A_5, A_6, A_7]):
    print(f"A_{i}:")
    print(matrix)
    print()

x = Variable((7,), name='x')

A_x = A_0 + x[0] * A_1 + x[1] * A_2 + x[2] * A_3 + \
    x[3] * A_4 + x[4] * A_5 + x[5] * A_6 + x[6] * A_7
```

```
problem = Problem(Minimize(norm(A_x)), [A_x >> 0])
problem.solve()

print("optimal value: %s" % problem.value, "\n")
for variable in problem.variables():
    print(" %s: %s" % (variable.name(), variable.value))
```

```
A_0:
[[ 50   6   0   4   0]
 [  6   0   0   0 -15]
 [  0   0  46   0  17]
 [  4   0   0   0 -13]
 [  0 -15  17 -13  70]]

A_1:
[[0. 0. 1. 0. 0.]
 [0. 0. 0. 0. 0.]
 [1. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0.]]

A_2:
[[0. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0.]
 [0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0.]]

A_3:
[[0. 0. 0. 0. 1.]
 [0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0.]
 [1. 0. 0. 0. 0.]]

A_4:
[[0. 0. 0. 0. 0.]
 [0. 0. 1. 0. 0.]
 [0. 1. 0. 0. 0.]
 [0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0.]]

A_5:
[[0. 0. 0. 0. 0.]
 [0. 0. 0. 1. 0.]
 [0. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0.]
 [0. 0. 0. 0. 0.]]
```

```
A_6:
[[0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0.]
 [0. 0. 0. 1. 0.]
 [0. 0. 1. 0. 0.]
 [0. 0. 0. 0. 0.]]

A_7:
[[0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0.]
 [0. 0. 0. 1. 0.]
 [0. 0. 0. 0. 0.]]

optimal value: 79.75713054541654

 x: [ 8.35294180e+00  2.06672003e+01 -4.25682751e-07  2.33502959e+01
  1.79463264e+01  2.04519381e+01  1.57945562e+01]
```

Problem 4

Consider the quadratically constrained quadratic optimization problem

$$\begin{aligned} &\text{minimize} && x^T A_0 x + 2b_0^T x + c_0 \\ &\text{subject to} && x^T A_k x + 2b_k^T x + c_k \leq 1, \quad k = 1, \ldots, m \\ & && x \succeq 0 \end{aligned}$$

the matrices $A_k, k = 0, \ldots, m$ are symmetric and $n \times n$, with nonpositive off-diagonal elements. The vectors $b_k \in \mathbb{R}^n$ are also nonpositive. We do not assume that A_k are positive semidefinite, so this is not convex as stated.

(a) show that $f(u, v) = \sqrt{uv}$ is concave with $u, v \in \mathbb{R}_+$.

take (x_0, y_0) and (x_1, y_1) so we need to show

$$f(t(x_0, y_0) + (1-t)(x_1, y_1)) \geq t f((x_0, y_0)) + (1-t) f((x_1, y_1))$$

by squaring

$$(tx_0 + (1-t)x_1)(ty_0 + (1-t)y_1) \geq \left(t\sqrt{x_0 y_0} + (1-t)\sqrt{x_1 y_1}\right)^2$$

which holds by Cauchy-Schwarz.

(b) Express the problem as a convex optimization problem.

hint: use $y_i = x_i^2$.

$$x^T A_k x + 2b^T x + c_0 = \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij} x_i x_j + 2\sum_{i=1}^{n} b_i x_i + c_0$$

$$= \sum_{i=1}^{n} a_{ii} x_i^2 + \sum_{i=1}^{n}\sum_{j\neq 1}^{n} a_{ij} x_i x_j$$

here we sub $\sqrt{y_i} = x_i$ and use the fact that the off-diagnol elements of A are < 0 and $b < 0$.

$$\Rightarrow \underbrace{\sum_{i=1}^{n} a_{ii} y_i}_{\text{affine}} \underbrace{- \sum_{i=1}^{n} |a_{ij}| \sqrt{y_i y_j}}_{\text{concave}} \underbrace{- 2\sum_{i=1}^{n} |b_i| \sqrt{y_i}}_{\text{concave}} + c_0$$

affine

Convex

Convex

→ the constraints and objective are convex so this is a convex optimization problem. We can write it as

$$\underset{y}{\text{minimize}} \quad \sqrt{y}^T A_0 \sqrt{y} + 2b_0^T \sqrt{y} + c_0$$

$$\text{s.t.} \quad \sqrt{y}^T A_k \sqrt{y} + 2b_k^T \sqrt{y} + c_k \leq 1, \quad k = 1, \ldots, m.$$

$$\sqrt{y} \succeq 0.$$

(c) Write the convex problem in the previous part as an SOCP and SDP.

We will use the hint to put this into SOCP form. Let t_{ij} be a slack variable such that $\sqrt{y_i y_j} \geq t_{ij}^{(a)}$ and by squaring $y_i y_j \geq (t_{ij}^{(a)})^2$

Since $y_i \geq 0$ and given the hint [trick for hyperbolic cone constraints from ex. 4.16 in the book] we can write

$$(t_{ij}^{(a)})^2 \leq y_i y_j \iff \left\| \begin{bmatrix} 2t_{ij}^{a} \\ y_i \quad y_j \end{bmatrix} \right\|_2 \leq y_i + y_j$$

and similarly we define $t_i^{(b)}$ for $i = 1, \ldots n$ such that $\sqrt{y_i} \geq t_i^{(b)}$ and by squaring $y_i \geq (t_i^{(b)})^2$ so we can write the constraint as

$$(t_i^{(b)})^2 \leq y_i \iff \left\| \begin{bmatrix} 2t_i^{(b)} \\ y_i \quad 1 \end{bmatrix} \right\|_2 \leq y_i + 1$$

We can now write the optimization problem in epigraph form given the previous constraints and this is an SOCP:

minimize t

subject to $\sum_i \sum_j A_{0ij} t_{ij}^{(a)} + \sum_i b_{0i} t_i^{(b)} + c_0 \le t$

$$\sum_i \sum_j A_{kij} t_{ij}^{(a)} + \sum_i b_{ki} t_i^{(b)} + c_k \le 1, \quad k = 1, \dots, m$$

$$\left\| \begin{bmatrix} 2t_i^{(b)} \\ y_i - 1 \end{bmatrix} \right\|_2 \le y_i + 1, \quad i = 1, \dots, n$$

$$\left\| \begin{bmatrix} 2t_{ij}^{(a)} \\ y_i - y_j \end{bmatrix} \right\|_2 \le y_i + y_j, \quad i, j = 1, \dots, n.$$

Problem 5

Two investments with Random returns R_1 and R_2.
Probability of loss $p^{loss} = Prob(R_1 + R_2) \leq 0$.

R_1 and R_2 have Gaussian marginal distributions with known means μ_1 and μ_2 and standard deviations σ_1 and σ_2.

R_1 and R_2 are correlated with correlation coefficient

$$E(R_1 - \mu_1)(R_2 - \mu_2) = \rho \sigma_1 \sigma_2 .$$

Want to find the worst case p^{loss} over any joint distribution of R_1 and R_2 consistent with the given marginals and correlation coefficient.

To approximately solve it, we discretize the values that R_1 and R_2 can take on, to $n = 100$ values $r_1, \ldots, r_n$, uniformly spaced $r_1 = -30$ to $r_n = 70$. We use the discretized marginals $p^{(1)}$ and $p^{(2)}$ for R_1 and R_2 given by

$$p_i^{(k)} = prob(R_k = r_i) = \frac{\exp(-(r_i - \mu_k)^2/(2\sigma_k^2))}{\sum_{j=1}^{n} \exp(-(r_j - \mu_k)^2/(2\sigma_k^2))}$$

for $k = 1, 2$, $i = 1, \dots, n$.

Formulate the convex optimization problem and solve it.

Define a joint probability matrix $P \in \mathbb{R}_+^{n \times n}$ where

$$P_{i,j} = \text{prob}(R_1 = r_i, R_2 = r_j).$$

Correlation constraint

$$(r - \mu_1 1)^T P (r - \mu_2 1) = \rho \sigma_1 \sigma_2$$

We we given marginals $p^{(1)}$ and $p^{(2)}$ so our matrix of joint probabilitres will sum to them such that

$$P1 = p^{(1)} \quad \text{and} \quad P^T 1 = p^{(2)}.$$

We know that $p^{\text{loss}} = \text{prob}(R_1 + R_2 \leq 0)$ so we look at every entry of the joint probability matrix such that

$$\text{prob}(R_1 + R_2 \leq 0) = \sum_{r_i + r_j \leq 0} P_{ij}.$$

Now that we have the constraints we formulate this as a convex optimization program. We want to find the worst-case (maximum possible value) ρ^{loss} over any joint distribution. So we write this specifically as a maximization problem.

$$\text{maximize} \sum_{r_i + r_j \leq 0} p_{i,j}$$

$$\text{subject to} \quad p_{ij} \geq 0 \;, \; i,j = 1, \cdots, n$$

$$p\mathbf{1} = \rho^{(1)}$$

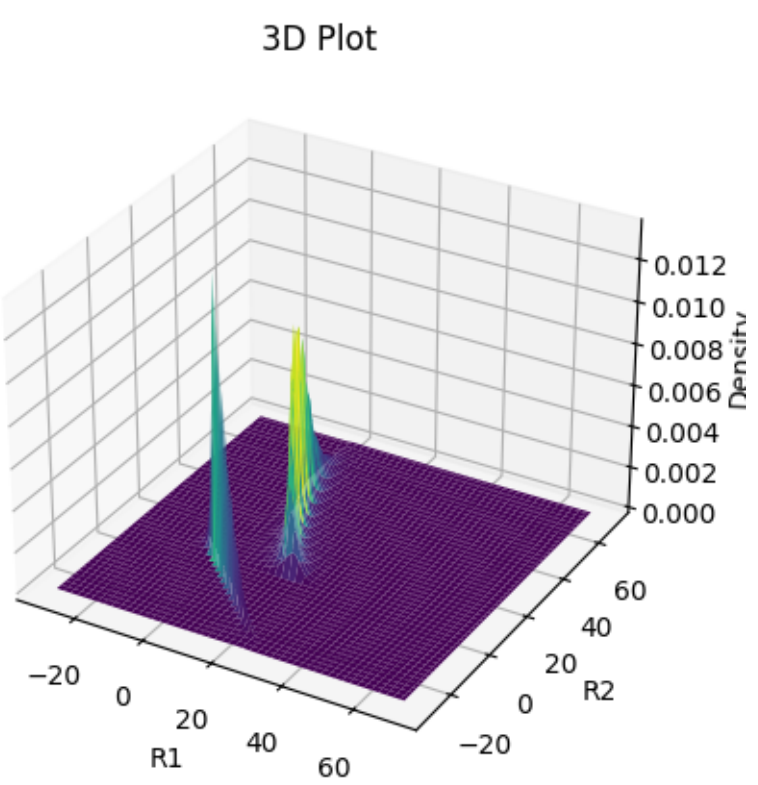
$$p^T\mathbf{1} = \rho^{(2)}$$

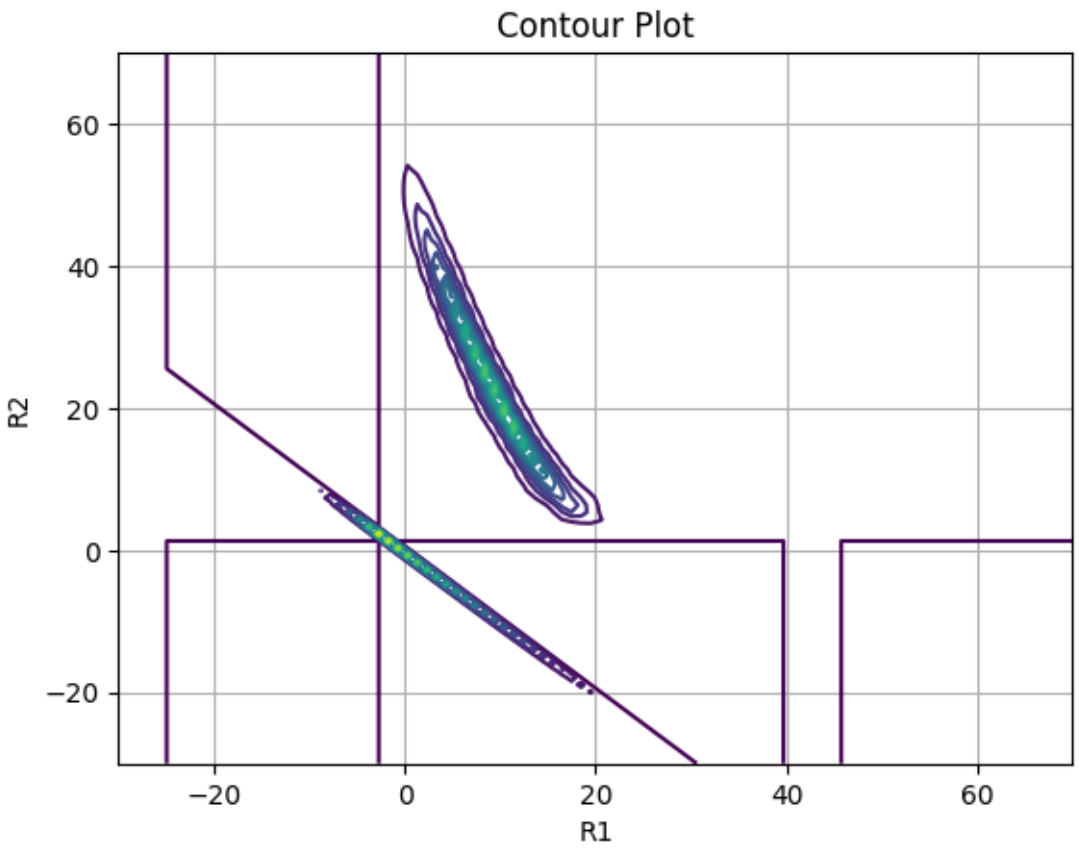
$$(r - \mu_1 \mathbf{1})^T P (r - \mu_2 \mathbf{1}) = \rho \sigma_1 \sigma_2$$

[4]:
```
import numpy as np
import cvxpy as cp
import matplotlib.pyplot as plt
from scipy.stats import norm

"""
Problem 5
"""

mu1 = 8
mu2 = 20
sigma1 = 6
sigma2 = 17.5
rho = -0.25

n = 100
r_1, r_n = -30, 70

# discretize R1 and R2 by n in [-30, 70]
r = np.linspace(r_1, r_n, n)

# marginal distributions
p1 = np.exp(-(r - mu1)**2 / (2 * sigma1**2))
p1 /= np.sum(p1)

p2 = np.exp(-(r - mu2)**2 / (2 * sigma2**2))
p2 /= np.sum(p2)

# region where R1 + R2 <= 0
r1p, r2p = np.meshgrid(r, r)
loss_region = (r1p + r2p <= 0)

"""
Define optimization problem
"""
P = cp.Variable((n, n), nonneg=True)

# Maximize:
objective = cp.Maximize(cp.sum(cp.multiply(P, loss_region)))

#Subject to
constraint1 = cp.sum(P, axis=1) == p1
constraint2 = cp.sum(P, axis=0) == p2
constraint3 = (r - mu1) @ P @ (r - mu2) == rho * sigma1 * sigma2
constraints = [constraint1, constraint2, constraint3]

# solve problem
```

```
problem = cp.Problem(objective, constraints)
problem.solve(solver=cp.ECOS)

p_max = problem.value  # worst case probability of loss
print("Worst-case probability of loss: ", p_max, "\n\n")

# joint probability matrix
P = P.value
print("Joint Probability Matrix:\n", P)

# Plotting the first figure
fig1 = plt.figure()
ax1 = fig1.add_subplot(111, projection='3d')
ax1.plot_surface(r1p, r2p, P.T, cmap='viridis')
ax1.set_xlabel('R1')
ax1.set_ylabel('R2')
ax1.set_zlabel('Density')
ax1.set_title('3D Plot')

# Plotting the second figure
fig2 = plt.figure()
ax2 = fig2.add_subplot(111)
contour = ax2.contour(r1p, r2p, P.T, levels=20, cmap='viridis')
ax2.set_xlabel('R1')
ax2.set_ylabel('R2')
ax2.set_title('Contour Plot')
ax2.grid(True)

plt.show()
```

```
Worst-case probability of loss:  0.19203644950901483


Joint Probability Matrix:
 [[1.82483190e-14 1.85911393e-14 1.89484389e-14 ... 1.89344717e-14
  1.85771471e-14 1.82342987e-14]
 [4.55661709e-13 4.65022797e-13 4.74778082e-13 ... 4.74764083e-13
  4.65008772e-13 4.55647656e-13]
 [1.66723372e-12 1.70157464e-12 1.73736154e-12 ... 1.73734751e-12
  1.70156058e-12 1.66721963e-12]
 ...
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 ... 3.44981470e-14
  3.38560033e-14 3.32398294e-14]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 ... 3.39552252e-14
  3.33236631e-14 3.27176430e-14]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 ... 3.34296803e-14
  3.28083558e-14 3.22121591e-14]]
```


3D Plot
0.012
0.010
0.008
0.006
0.004
0.002
0.000
Density
−20
0
20
40
60
R1
−20
0
20
40
60
R2


Contour Plot
60
40
20
0
−20
R2
−20
0
20
40
60
R1

[ ]:

Problem 6

Total Energy consumed $E = \sum_{t=1}^{T} \phi(s_t)$

θ_t – processor effort allocation, θ_{ti} – fraction of the processor effort devoted to job i in period t.

$\mathbf{1}^T \theta_t = 1, \quad \theta_t \geq 0.$

To complete the jobs we must have

$$\sum_{t=A_i}^{D_i} \theta_{ti} s_t \geq W_i, \quad i = 1, \ldots, n$$

Availability time $A_i \in \{1, \ldots, T\}$

Deadline $D_i \in \{1, \ldots, T\}$ $\qquad D_i \geq A_i$

Job i involves a (nonnegative) total work W_i.

- We want to formulate the problem of choosing the speeds $s_1, \ldots, s_T$ and the allocations $\theta_1, \ldots, \theta_T$, in order to minimize the total energy E, as a convex optimization problem.

We want to define a matrix $P \in \mathbb{R}^{T \times n}$ where the components are $p_{ti} = \theta_{ti} s_t$ which we know must be nonnegative since we have the constraint

$$\sum_{A_i}^{D_i} \theta_{ti} s_t = \sum_{A_i}^{D_i} p_{ti} \geq W_i, \quad i = 1, \ldots, n \quad \text{for } W \geq 0.$$

Remark The change of variables $p_{ti} = \theta_{ti} s_t$ changes the above nonconvex constraint convex. This constraint was not accepted as DCP in CVXPY.

Therefore we can write the job completion constraint as

$$P^T 1 \geq W.$$

We are given the slew-rate limit $|s_{t+1} - s_t| \leq R$, for $t = 1, \ldots, T-1$.

Its also given that the speeds must lie between given (positive) minimum and maximum values

$$s^{min} \leq s \leq s^{max}.$$

Its also required that there is at least one job available i.e., for each t, there is at least one i with $A_i \leq t$ and $D_i \geq t$.

Respecting the availability and deadline constraints requires that $\theta_{ti} = 0$ for $t < A_i$ or $t > D_i$.

We define this as $p_{ti} = 0$ for $t < A_i$ or $t > D_i$

Convex optimization problem:

$$\text{minimize} \quad E = \sum_{t=1}^{T} \phi(s_t)$$

$$\text{Subject to} \quad s^{min} \leq s \leq s^{max}$$

$$p^T 1 \geq W, \quad p \geq 0$$

$$|s_{t+1} - s_t| \leq R, \quad t = 1, \ldots, T$$

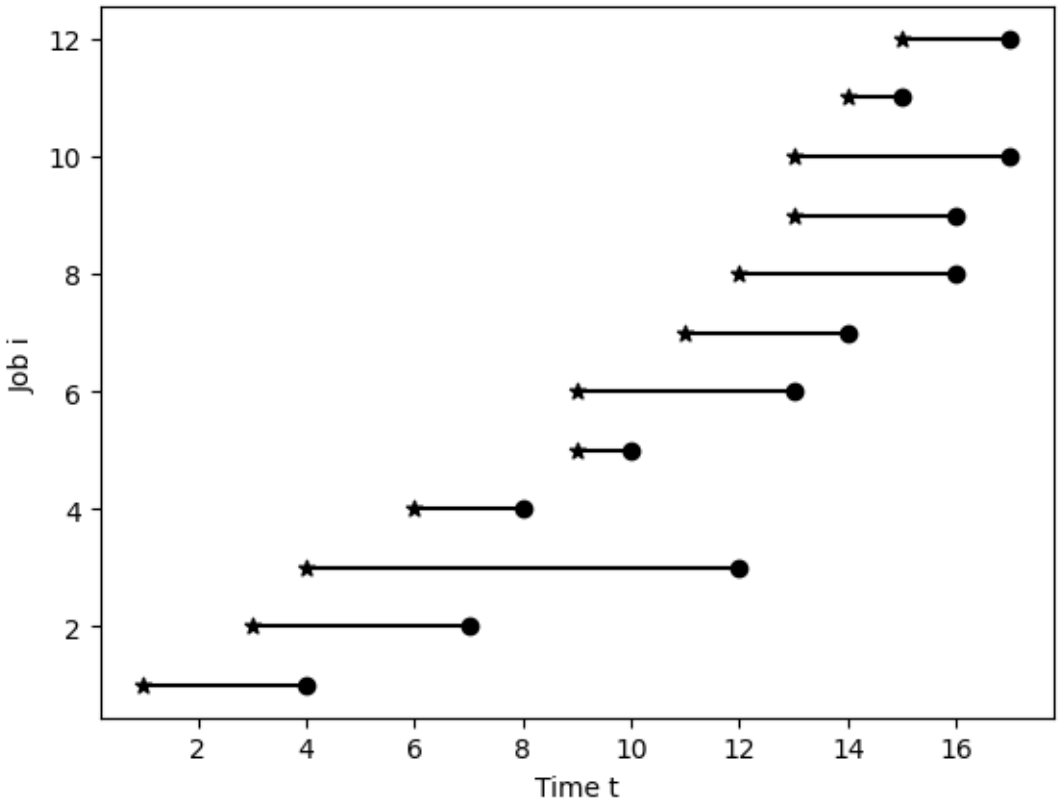
$$p_{ti} = 0, \quad A_i \leq t$$

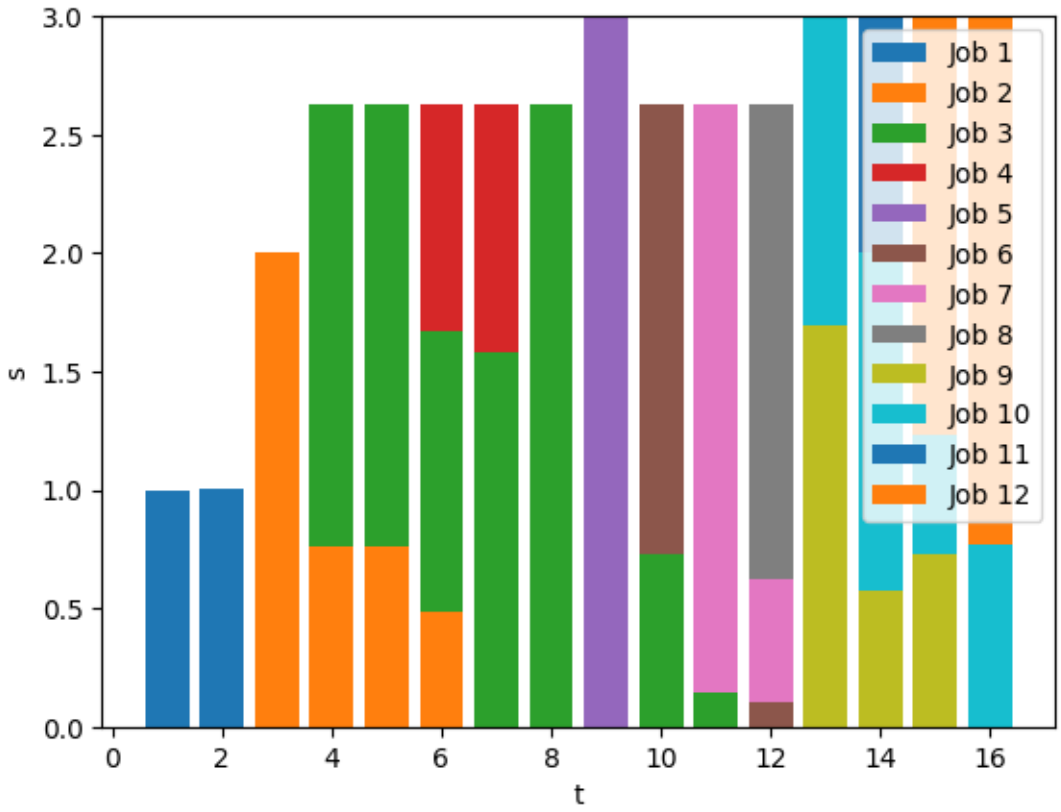
$$p_{ti} = 0, \quad b_i \geq t$$

[40]:
```
import numpy as np
import cvxpy as cp
import matplotlib.pyplot as plt
from scipy.stats import norm

"""
Problem 6

"""
def plot_job_availability(A, D, n):
    ax = plt.figure()
    plt.scatter(A, np.linspace(1,n,num=n), color='k', marker='*')
    plt.scatter(D+1, np.linspace(1,n,num=n), color='k', marker='o')
    for i in range(n):
        plt.plot(np.append(A[i],D[i]+1),np.append(i+1,i+1), 'k-')

    plt.xlabel('Time t')
    plt.ylabel('Job i')

def plot_speeds(T, allocation):
    plt.figure()
    for i in range(n):
        plt.bar(np.arange(1, T + 1), allocation[i], bottom=np.sum(allocation[:
 ↪i], axis=0), label=f'Job {i+1}')
    plt.xlabel('t')
    plt.ylabel('s')
    plt.legend()
    plt.show()

# Given parameters
n = 12   # number of jobs.
T = 16   # number of time periods.
Smin = 1 # min processor speed.
Smax = 4 # max processor speed.
R = 1    # max slew rate.


alpha = 1
beta = 1
gamma = 1

# Job arrival times and deadlines.
A = np.array([1, 3, 4, 6, 9, 9, 11, 12, 13, 13, 14, 15])
D = np.array([3, 6, 11, 7, 9, 12, 13, 15, 15, 16, 14, 16])
# Total work for each job.
W = np.array([2, 4, 10, 2, 3, 2, 3, 2, 3, 4, 1, 4])
```

```
# define problem variables
P = cp.Variable((T, n))
s = cp.sum(P, axis=1)
theta = P / cp.reshape(s, (T, 1))

# define constraints
constraints = [
    P >= 0,
    s >= Smin,
    s <= Smax,
    cp.abs(s[1:] - s[:-1]) <= R, # slew-rate constraint
    cp.sum(P, axis=0) >= W
]

# job availability constraints
for i in range(n):
    for t in range(A[i] - 1): # A[i] indexed at 1
        constraints.append(P[t, i] == 0)
    for t in range(D[i], T):
        constraints.append(P[t, i] == 0)

# define objective function
objective = cp.Minimize(cp.sum(alpha + beta * s + gamma * s**2))

problem = cp.Problem(objective, constraints)
problem.solve()

# optimal Energy
E = problem.value
print("The optimal Energy value is: ", E)

speed = s.value
print("\n The speeds that achive this are: \n", speed)
theta = theta.value.T
allocation = speed * theta


plot_job_availability(A,D,n)
plot_speeds(T, allocation)
```

```
The optimal Energy value is:  162.1244355805247

 The speeds that achive this are:
 [0.99992815 1.00020736 2.00028499 2.62493191 2.62494799 2.62494758
 2.62494704 2.62494537 3.000019   2.6249454  2.62494677 2.62494548
 3.00000926 3.00000813 3.00000816 3.0000082 ]
```


Job i
Time t
12
10
8
6
4
2
2
4
6
8
10
12
14
16


3.0
2.5
2.0
1.5
1.0
0.5
0.0
s
0
2
4
6
8
10
12
14
16
t
Job 1
Job 2
Job 3
Job 4
Job 5
Job 6
Job 7
Job 8
Job 9
Job 10
Job 11
Job 12

[ ]: