Numerical Optimization - Assignment 6

David Alvear(187594) - Nouf Farhoud(189731)

In [1]:
```
import numpy as np
import matplotlib
import cvxpy as cp
import matplotlib.pyplot as plt
```

1. Image Reconstruction Revisited

1. Image reconstruction revisited. Redo the reconstruction problem using a *total variation* roughness measure, which is an L_1 norm of the form:

$$\sum_{i=2}^{M}\sum_{j=2}^{N}|U_{ij} - U_{i-1,j}| + |U_{ij} - U_{i,j-1}|$$

This objective is piecewise linear and therefore has no useful pointwise Hessian information and does not even have a unique gradient at some points.

- Solve the problem using `cvxpy` and comment on the quality of the solution.
- Write down a reformulation of this L_1 problem as a linear optimization problem. (`cvxpy` does this automatically under the hood).

Problem Formulation

$$\text{minimize} \sum_{i=1}^{m} \sum_{j=1}^{N} |u_{ij} - u_{i,j-1}| + |u_{ij} - u_{i-1,j}|$$
$$\text{subject to} \quad u_{ij} - u_{ij}^0 = 0$$

$$\text{Define} \quad K = m \times j + i, \text{ where } x_k \text{ corresponds to } u_{ij}.$$
$$\text{minimize} \sum_{k=1}^{mN} |x_k - x_{k-1}| + |x_k - x_{k-m}|$$
$$\text{subject to} \quad x_k - x_k^0 = 0$$

$$\text{Now}: -u \leq x_k - x_{k-1} \leq u \quad \text{and} \quad -v \leq x_k - x_{k-m} \leq v$$

$$\text{minimize} \sum_{k=1}^{mN} u_k + v_k$$
$$\text{subject to}$$
$$\begin{aligned} & x_k - x_{k-1} \leq u_k \\ & -(x_k - x_{k-1}) \leq u_k \\ & x_k - x_{k-m} \leq v_k \\ & -(x_k - x_{k-m}) \leq v_k \\ & x_k - x_k^0 = 0 \\ & x_k \geq 0 \end{aligned}$$

Then:

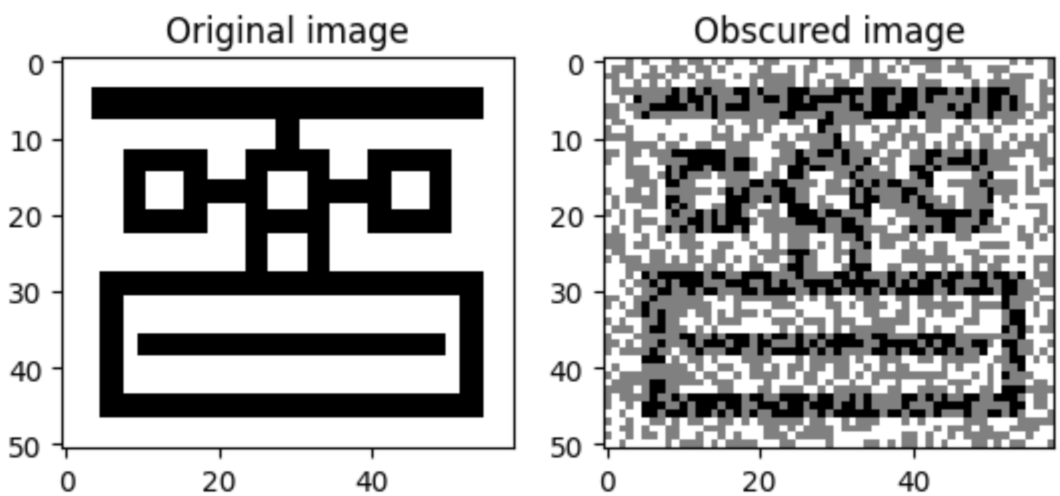
$$\text{minimize} \quad \begin{bmatrix} 0^T & 1^T & 1^T \end{bmatrix} \begin{bmatrix} x \\ u \\ v \end{bmatrix}$$

Subject to

$$\begin{bmatrix} B & -I & 0 \\ -B & -I & 0 \\ C & 0 & -I \\ -C & 0 & -I \end{bmatrix} \begin{bmatrix} x \\ u \\ v \end{bmatrix} \leq \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$
$$\begin{bmatrix} A & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ u \\ v \end{bmatrix} = \begin{bmatrix} b \end{bmatrix}$$
$$\begin{bmatrix} x \\ u \\ v \end{bmatrix} \geq \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

In [2]:
```
import numpy as np
import matplotlib
import cvxpy as cp
```

```
import matplotlib.pyplot as plt

# Read a sample (``original'') image
U0 = plt.imread('bwicon.png')   # values are between 0.0 (black) and 1.0 (wh
m, n = U0.shape

# Create 50% mask of known pixels and use it to obscure the original
np.random.seed(211)       # seed the random number generator (for repeatabil
unknown = np.random.rand(m,n) < 0.5
U1 = U0*(1-unknown) + 0.5*unknown

# Display images
plt.figure(1)
plt.subplot(1, 2, 1)
plt.imshow(U0, cmap='gray')
plt.title('Original image')
plt.subplot(1, 2, 2)
plt.imshow(U1, cmap='gray')
plt.title('Obscured image')
plt.show()
```



In [3]:

```
m , n = U0.shape
# Build the matrices
B = np.zeros((m*n, m*n))
C = np.zeros((m*n, m*n))

# Create the adjacency matrices
for i in range (1, m*n):
  B[i, i-1] = -1
  B[i, i] = 1
  C[i, i-m] = -1
  C[i, i] = 1

# Constraints
b = np.zeros(m*n)
_A = np.zeros([m*n, 1])

for j in range(0,n):
    for i in range(0,m):
        k = m * j + i
```

```
        if unknown[i][j]==True:
            b[k]=0
        elif unknown[i][j]==False:
            b[k]=U0[i][j]
            _A[k]=1
A=np.diag(np.reshape(_A,[m*n,],order='F'))

# Define the matrices of the new formulation
I = np.identity(B.shape[0])
zeros_matrix = np.zeros_like(I)
A_L1 = np.block([A, np.zeros_like(A), np.zeros_like(A)])
BC = np.block([[B, -I, zeros_matrix],
              [-B, -I, zeros_matrix],
              [C, zeros_matrix, -I],
              [-C, zeros_matrix, -I]])
zeros_BC = np.zeros(BC.shape[0])

# Objective matrix
zeros = np.zeros(n*m).T
ones = np.ones(n*m).T
P = np.block([zeros, ones, ones])
```

In [4]:

```
# Define variable
x = cp.Variable(3*m*n)
# Define objective
objective = cp.Minimize(P @ x)
# Constraints
constraints = [A_L1 @ x == b,
               BC @ x <= zeros_BC]

# Create problem
problem = cp.Problem(objective, constraints)

# Solve the problem
problem.solve()
```

Out[4]:

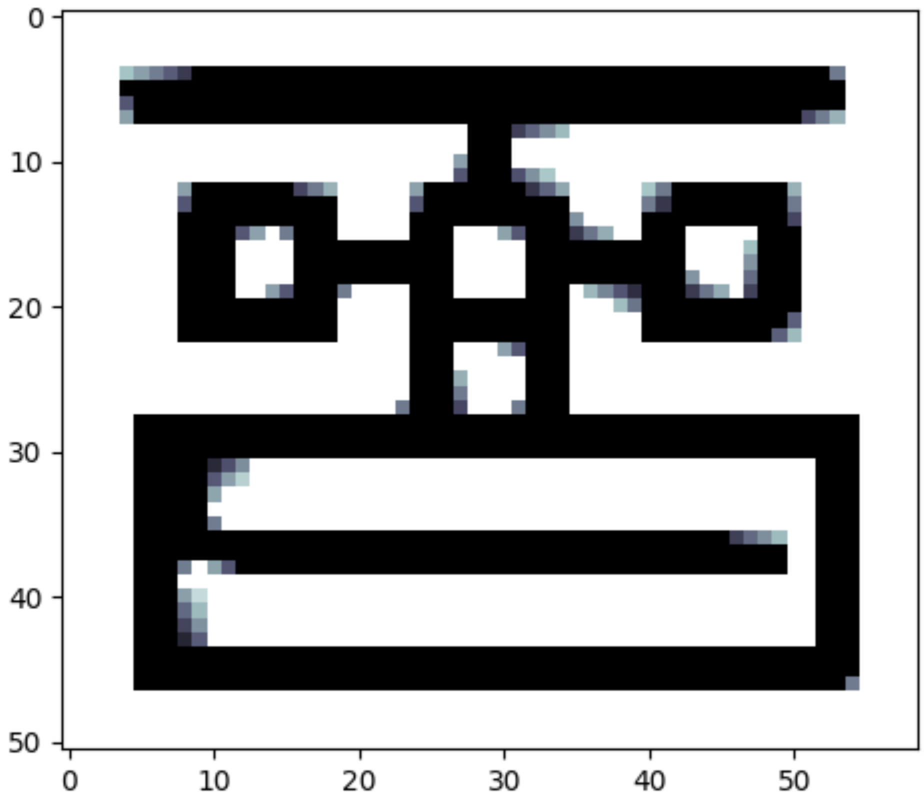
```
649.9999999938655
```

In [5]:

```
image = np.array(x.value)[:m*n]
image = np.reshape(image, (m, n), order='F')
plt.imshow(image, cmap='bone')
```

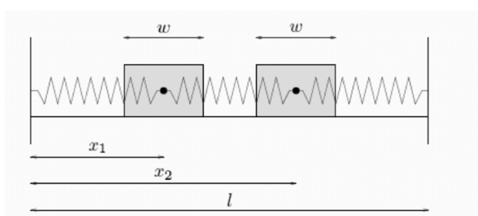
Out[5]:

```
<matplotlib.image.AxesImage at 0x7af23642d6f0>
```



2. Contact Problem in 1D

2. Contact problem in 1D. Contact problems that arise in mechanics can be formulated as optimization problems. In fact, the understanding of mechanics problems was one of Lagrange's original motivations for the study of optimization. As an example consider the following set of rigid blocks connected by deformable springs.



- Using the model described on p. 247 of reference 2, find the equilibrium position of the blocks for the following values of the spring stiffnesses $k_1 = 1; k_2 = 10; k_3 = 2$ (units are force per length). Use $l = 1; w = 0.2$.
- What is the significance of the multipliers in this problem?
- Consider an extension to this problem where the blocks are deformable, and their deformation is defined by the elastic energy $1/2c_i(\Delta w_i)^2$, where Δw_i is the change in width of the i-th block. The problem becomes one of minimizing the sum of potential energies of the springs and the blocks. Formulate and solve the problem for the following values $c_1 = 2; c_2 = 4$.

Problem formulation and optimal value computation

Problem Formulation

$$
\begin{aligned}
\text{minimize} \quad & \frac{1}{2}\left(k_1x_1^2 + k_2(x_2 - x_1)^2 + k_3(l - x_2)^2\right) \\
\text{subject to} \quad & \frac{w}{2} - x_1 \leq 0 \\
& w + x_1 - x_2 \leq 0 \\
& \frac{w}{2} - l + x_2 \leq 0
\end{aligned}
$$

We have:

$$
\begin{aligned}
f(\mathbf{x}) &= \frac{1}{2}\left(k_1x_1^2 + k_2x_2^2 + 2k_2x_1x_2 + k_3x_2^2 + k_3l^2 - 2k_3lx_2 + k_3x_2^2\right) \\
&= \frac{1}{2}\begin{bmatrix} x_1 & x_2 \end{bmatrix}\begin{bmatrix} (k_1+k_2) & -k_2 \\ -k_2 & (k_2+k_3) \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \frac{1}{2}k_3l^2 \\
&= \frac{1}{2}\begin{bmatrix} x_1 & x_2 \end{bmatrix}\begin{bmatrix} k_1+k_2 & -k_2 \\ -k_2 & k_2+k_3 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 & -2k_3l \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + k_3l^2
\end{aligned}
$$

Define: $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$

Then the problem becomes:

$$
\begin{aligned}
\text{minimize} \quad & \frac{1}{2}(\mathbf{x}^T\mathbf{P}\mathbf{x} + k_3l^2 + \mathbf{q}^T\mathbf{x}) \\
\text{Subject to} \quad & \mathbf{A}\mathbf{x} - \mathbf{b} \leq \mathbf{0} \\
& \mathbf{x} \geq \mathbf{0}
\end{aligned}
$$

$$
\text{where:} \quad \mathbf{P} = \begin{bmatrix} k_1+k_2 & -k_2 \\ -k_2 & k_2+k_3 \end{bmatrix}, \quad \mathbf{q} = \begin{bmatrix} 0 \\ -2k_3l \end{bmatrix},
$$

$$
\text{and} \quad \mathbf{A} = \begin{bmatrix} -1 & 0 \\ 1 & -1 \\ 0 & 1 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} \frac{-w}{2} \\ w \\ l - \frac{w}{2} \end{bmatrix}.
$$

In [6]:

```
k1 = 1 # N/m
k2 = 10
k3 = 2
l = 1
w = 0.2

P = np.array([
    [k1+k2, -k2],
    [-k2, k2 + k3]
])

a = np.array([0, -2*l*k3])

A = np.array([
    [-1, 0],
```

```
    [1, -1],
    [0, 1]
])

b = np.array([-w/2, -w, l - w/2])

# Define variable
x = cp.Variable(2)

# Define objective
objective = cp.Minimize(0.5*(cp.quad_form(x, P) + a @ x + k3*(l**2)))
# objective = cp.Minimize(((x.T @ P) @ x) + k3*l)

# Define constraints
constraints = [A @ x <= b]

# Define problem and solve
problem = cp.Problem(objective, constraints)

# Solve the problem
problem.solve()
```

Out[6]:

```
0.4133333333333328
```

In [7]:

```
# optimal value
print(f'The optimum solution is the vector x_* = {x.value}')

# Get the dual values or lagrange multipliers of the constraints
print(f"lagrange multipliers of the constraints: {problem.constraints[0].dua
```

```
The optimum solution is the vector x_* = [0.53333333 0.73333333]
lagrange multipliers of the constraints: [0.         1.46666667 0.        ]
```

The optimal value or equilibrium point for the blocks is when the block 1 is centered at 0.53333 m and block number 2 at 0.733333.

What is the significance of the multipliers of this problem?

The lagrange multipliers in this example represent the contact forces of the blocks with the walls and between them. If $\lambda = 0$ implies that the constraint is not active. The result of the problem evidence that constraints 1 and 3 are not active, and that means that there is no force applied to walls constraint. However, if $\lambda > 0$ implies that the lagrange multiplier is active and there exist a force in the spring between the blocks to maintain them apart.

Elastic energy in the blocks

Problem Formulation

Elastic energy: $\quad \frac{1}{2}c_i(\Delta w_i)^2$

$\Delta w_i \rightarrow$ change of width i block

Define:

$$\Delta w_1 = x_1 - \frac{w}{2} + x_2 - x_1 - w$$

$$\Delta w_2 = l - x_2 - \frac{w}{2} + x_2 - x_1 - w$$

Now define the elastic energy of the blocks:

$$f_1(x) = \frac{1}{2}c_1(x_1 - u + x_2 - x_1 - w)^2 + \frac{1}{2}c_2(l - x_2 - \frac{w}{2} + x_2 - x_1 - w)^2$$

$$f_1(x) = \frac{1}{2}c_1\left(\frac{x_2 - 3w}{2}\right)^2 + \frac{1}{2}c_2\left(\frac{(l - 3w/2) - x_2}{2}\right)^2$$

$$f_1(x) = \frac{1}{2}c_1(x_2 - 3w)^2 + \frac{9w^2}{4} + \frac{1}{2}c_2\left(\left(\frac{l-3w}{2}\right) - x_2\right)^2 - 2x_1\left(\frac{l-3w}{2}\right)c_2 + x_1^2$$

$$f_1(x) = \frac{1}{2}c_2x_1^2 - c_2x_1\left(\frac{l-3w}{2}\right) + \frac{1}{2}c_1x_2^2 - \frac{3}{2}c_1wx_2 + \frac{9w^2c_1}{8} + \left(\frac{l-3w}{2}\right)^2\frac{c_2}{2}$$

$$f_1(x) = \begin{bmatrix} x_1 & x_2 \end{bmatrix}\begin{bmatrix} \frac{1}{2}c_2 & 0 \\ 0 & \frac{1}{2}c_1 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

$$+ \begin{bmatrix} -\frac{c_2(l-3w)}{2} & -\frac{3}{2}c_1w \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \frac{9w^2c_1}{4} + \left(\frac{l-3w}{2}\right)^2\frac{c_2}{2}.$$

In [8]:

```
c1 = 2
c2 = 4
E = np.array([
    [0.5*c2, 0],
    [0, 0.5*c1]
])
q = np.array([-c2*(l - 1.5*w), -1.5*c1*w])
e_const = (9/8)*(w**2)*c1 + 0.5*(l - 1.5*w)

# Define objective with the potential and elastic energy of the blocks
objective = cp.Minimize(0.5*(cp.quad_form(x, P) + a @ x + k3*(l**2)) \
                        + cp.quad_form(x, E) + q @ x + e_const)

# Define constraints
constraints = [A @ x <= b]

# Define problem and solve
problem = cp.Problem(objective, constraints)

# Solve the problem
problem.solve()
```

Out[8]: 0.024444444444444102

In [9]:
```
# optimal value
print(f'The optimum solution is the vector x_* = {x.value}')

# Get the dual values or lagrange multipliers of the constraints
print(f"lagrange multipliers of the constraints: {problem.constraints[0].dua
```

```
The optimum solution is the vector x_* = [0.51111111 0.71111111]
lagrange multipliers of the constraints: [0.         2.24444444 0.        ]
```

We can see that the addition of the elastic energy of the blocks reduced the objective valu of the optimization problem. From 0.4134 to 0.0245. The position of the blocks remain in a similar value but the force between the two blocks becomes larger with the consideration of the elastic energy.

3. Second order optimality conditions

3. Second order optimality conditions. Consider the problem of finding $x \in \mathbb{R}^2$:

$$\begin{array}{ll} \text{minimize} & \|x\|^2 \\ \text{subject to} & 2 - x_1 x_2 + 3x_1 \leq 0 \end{array}$$

- Verify that the point $x^* = [2, 4]^T, \lambda^* = 4$ satisfies the first order KKT conditions.
- Does this point satisfy the second order optimality conditions? Comment.

The KKT conditions for inequality constraints consist of the following three conditions:

- Stationarity condition: ∇f(x*) + ∑ λ∇g(x) = 0
- Primal feasibility condition: g(x*) ≤ 0
- Dual feasibility condition: λ* ≥ 0
- Complementary slackness condition: λ * g(x) = 0

Stationary Condition:

$$\nabla f(x^*) + \lambda \nabla g(x^*) = \nabla(||x||^2) + 4\nabla(2 - x_2x_1 + 3x_1)$$
$$= \begin{bmatrix} \frac{\partial x_1^2}{\partial x_1} + \frac{\partial x_2^2}{\partial x_1} + 4\frac{\partial(-x_2x_1+3x_1)}{\partial x_1} \\ \frac{\partial x_1^2}{\partial x_2} + \frac{\partial x_2^2}{\partial x_2} + 4\frac{\partial(-x_2x_1+3x_1)}{\partial x_2} \end{bmatrix}$$
$$= \begin{bmatrix} 2x_1 \\ 2x_2 \end{bmatrix} + y \begin{bmatrix} 3 - x_2 \\ -x_1 \end{bmatrix}$$
$$= \begin{bmatrix} 2(2) \\ 2(4) \end{bmatrix} + 4 \begin{bmatrix} 3 - 4 \\ -2 \end{bmatrix}$$
$$= \begin{bmatrix} 4 \\ 8 \end{bmatrix} + 4 \begin{bmatrix} -1 \\ -2 \end{bmatrix}$$
$$= \begin{bmatrix} 4 \\ 8 \end{bmatrix} + \begin{bmatrix} -4 \\ -8 \end{bmatrix}$$
$$= \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \text{satisfies the stationary condition.}$$

Primal Feasibility Condition:

$$g(x^*) \leq 0$$
$$g(x^*) = 2 - x_2x_1 + 3x_1$$
$$= 2 - 8 + 6$$
$$= 0 \quad \text{satisfies the primal feasibility condition.}$$

Dual Feasibility Condition:

$$y^* \geq 0$$
$$y^* = 4 \quad \text{satisfies the dual feasibility condition.}$$

Complementary Slackness Condition:

$$\lambda^* g(x^*) = 0$$
$$4 \times 0 = 0 \quad \text{satisfies the complementary slackness condition.}$$

To check the second-order optimality conditions, we need to examine the Hessian matrix and the second-order derivatives of the objective function and the constraint function.

Hessian matrix of the objective function:

$$H(f(x)) = \begin{bmatrix} \frac{\partial^2 ||x||^2}{\partial x_1^2} & \frac{\partial^2 ||x||^2}{\partial x_1 x_2} \\ \frac{\partial^2 ||x||^2}{\partial x_1 x_2} & \frac{\partial^2 ||x||^2}{\partial x_2^2} \end{bmatrix} = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$$

$\alpha = 2 - x_2 x_1 + 3x_1$

Hessian matrix of the constraint function:

$$H(g(x)) = \begin{bmatrix} \frac{\partial^2 \alpha}{\partial x_1^2} & \frac{\partial^2 \alpha}{\partial x_1 x_2} \\ \frac{\partial^2 \alpha}{\partial x_2 x_1} & \frac{\partial^2 \alpha}{\partial x_2^2} \end{bmatrix} = \begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix}$$

Hessian matrix of the Lagrangian function:

$HL(x, y) = H(f(x)) + \lambda H(g(x))$

$$= \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} + \lambda \begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix}$$

$$= \begin{bmatrix} 2 & -\lambda \\ -\lambda & 2 \end{bmatrix}$$

Determinant condition:

$\det(HL(x^*, y^*) - \lambda I) = 0$

$$\det\left(\begin{bmatrix} 2 & -4 \\ -4 & 2 \end{bmatrix} - \lambda \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\right) = 0$$

$$\det\left(\begin{bmatrix} 2-\lambda & -4 \\ -4 & 2-\lambda \end{bmatrix}\right) = 0$$

$(2-\lambda)^2 - 16 = 0$

$4 - 4\lambda + \lambda^2 - 16 = 0$

$\lambda^2 - 4\lambda - 12 = 0$

$\lambda_1 = 6, \quad \lambda_2 = -2$

λ_2 is negative, therefore, the second-order condition is not satisfied.

3. Dual problem in R

3. Dual problem in R**.** Consider the problem:

$$\begin{array}{ll} \text{minimize} & x^2 + 1 \\ \text{subject to} & (x-2)(x-4) \leq 0 \end{array}$$

- Derive an expression for the dual function $g(\lambda)$. Plot $g(\lambda)$.
- Solve the dual problem to find λ^* and g^*. Verify that this problem has strong duality.

Primal Problem:

minimize $x^2 + 1$ -- Convex parabola open upwards

s.t. $(x-2)(x-4) \leq 0$

so its minimum occurs at the feasible region

Constraint is satisfied when $2 \leq x \leq 4$

2 is the minimum:

$f(2) = 2^2 + 1$

$= 5$

$f(x^*) = 5$

Dual Problem:

Lagrangian $L(x, \lambda) = x^2 + 1 + \lambda(x^2 - 6x + 8)$

To find the dual function $g(\lambda)$

$$\frac{\partial L(x,\lambda)}{\partial x} = 2x + \lambda(2x-6) = 0$$

$2x + 2\lambda(x-3) = 0$

$2x = -2\lambda x + 6\lambda$

$2x + 2\lambda x = 6\lambda$

$2x(1+\lambda) = 6\lambda$

$$2x = \frac{6\lambda}{1+\lambda} \rightarrow x = \frac{3\lambda}{1+\lambda}$$

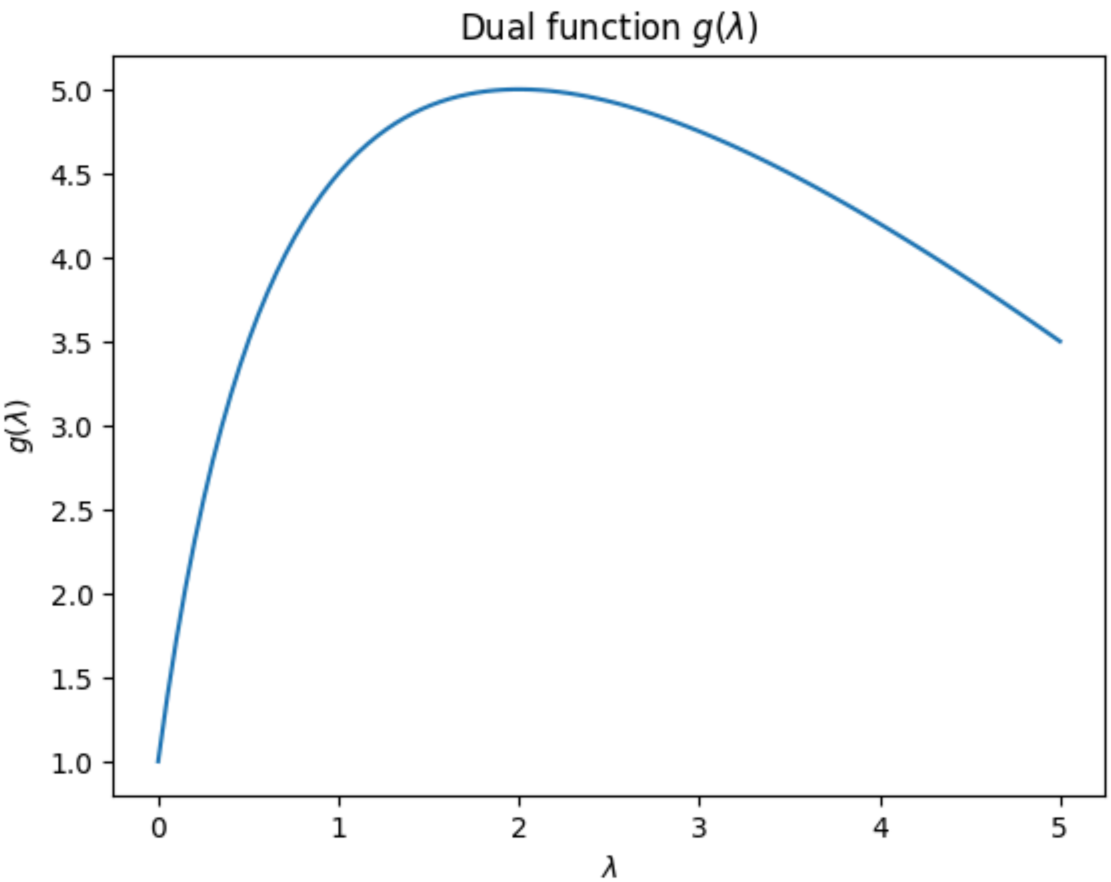
$$g(\lambda) = L\left(\frac{3\lambda}{1+\lambda}, \lambda\right) = \left(\frac{3\lambda}{1+\lambda}\right)^2 + 1 + \lambda\left(\left(\frac{3\lambda}{1+\lambda}\right) - 2\right)\left(\left(\frac{3\lambda}{1+\lambda}\right) - 4\right)$$

In [ ]:
```
#The variable
x = cp.Variable(1)

#Objective
Objective = cp.Minimize(x**2 + 1)

#Defining the constaraints
Constraints = [x**2 - 6*x + 8 <=0]

#The problem
problem = cp.Problem(Objective, Constraints)

#solve Problem
problem.solve()
print(f"Primal problem Optimum value: {problem.value}")
print(f"Primal optimum x value: {x.value}")
```

```
Primal problem Optimum value: 4.999999987444469
Primal optimum x value: [2.]
```

In [ ]:
```
#plotting the dual function
_lam= np.linspace(0,5,100)
```

```
def g(x):
  return ((3*x)/(1+x))**2  + 1 +x *(((3*x)/(1+x))**2- 6*((3*x)/(1+x)) +8)

plt.plot(_lam,g(_lam));
plt.xlabel('$\lambda$')
plt.ylabel('$g(\lambda)$')
plt.title('Dual function $g(\lambda)$');
```



In [ ]:
```
from scipy.optimize import fmin
#solving with scipy
min = fmin(lambda x: -g(x),1)
print(f"Minimum of the function: {min[0]}")
```

```
Optimization terminated successfully.
         Current function value: -5.000000
         Iterations: 17
         Function evaluations: 34
Minimum of the function: 2.000000000000002
```

In [ ]:
```
from scipy.optimize import minimize
f = lambda x: x**2 +1
const = ({'type': 'ineq','fun': lambda x: -(x-2)*(x-4)})
res = minimize(f, x0=np.array([2]), constraints=const)
print (res)
```

```
message: Optimization terminated successfully
success: True
 status: 0
    fun: 5.0
      x: [ 2.000e+00]
    nit: 1
    jac: [ 4.000e+00]
   nfev: 2
   njev: 1
```

In [ ]:
```python
g = lambda x: ((3*x)/(1+x))**2 +1+x *(((3*x)/(1+x))**2- 6*((3*x)/(1+x)) +8)
res = minimize(lambda x: -g(x), x0=np.array([2]))

print(res)
```

```
 message: Optimization terminated successfully.
 success: True
  status: 0
     fun: -5.0
       x: [ 2.000e+00]
     nit: 0
     jac: [-1.192e-07]
hess_inv: [[1]]
    nfev: 2
    njev: 1
```

4. Dual of a quadratic problem

4. Dual of a quadratic problem. Consider the following least squares solution of the set of underdetermined linear equations.

$$\begin{array}{ll} \text{minimize} & x^T x \\ \text{subject to} & Ax = b \end{array}$$

where $A \in R^{p \times n}$. Derive the dual problem.

Primal:

$$L(x,\lambda) = x^T x + \lambda^T(Ax - b)$$
$$\nabla_x L(x,\lambda) = \nabla_x(x^T x) + \nabla_x(\lambda^T(Ax-b))$$
$$= 2x + \lambda^T A$$
$$0 = x + \lambda^T A$$
$$x = -\frac{1}{2}A^T\lambda \quad \text{substitute back in the Lagrangian}$$

$$L(x,\lambda) = \left(-\frac{1}{2}A^T\lambda\right)^\top \left(-\frac{1}{2}A^T\lambda\right) + \lambda^T\left(A\left(-\frac{1}{2}A^T\lambda\right) - b\right)$$
$$= -\frac{1}{4}\lambda^T A\left(-\lambda A^T\right) + \lambda^T\left(-\frac{1}{2}\lambda A^T A - b\right)$$
$$= -\frac{1}{4}\lambda^T A A^T \lambda - \lambda^T b$$

Dual function: $g(\lambda)$ which we get by minimizing $L(x,\lambda)$ w.r.t λ

$$g(\lambda) = -\frac{1}{4}\lambda^T A A^T\lambda - \lambda^T b$$

Dual Problem:

$$\text{maximize } -\frac{1}{4}\lambda^T A A^T\lambda - \lambda^T b$$
$$\text{subject to } \lambda \geq 0 \quad \text{ensure feasibility}$$

5. Dual in R^n

5. Dual in R^n. Consider the following optimization problem where $x \in R^n$, $\mathbf{1}$ is the one vector in R^n, and b is a scalar.

$$\begin{array}{ll} \text{minimize} & p^T x \\ \text{subject to} & 0 \leq x \leq 1 \\ & \mathbf{1}^T x = b \end{array}$$

- Let ν denote the multiplier of the equality constraint, λ denote the multipliers of the inequalities $x \leq 1$, and λ_2 denote the multipliers of the non-negativity constraints. Write the Lagrangian of this problem.
- Show that the dual may be written as:

$$\begin{array}{ll} \text{minimize} & b\nu + \mathbf{1}^T\lambda \\ \text{subject to} & \nu\mathbf{1} + \lambda \geq -p \\ & \lambda \geq 0 \end{array}$$

1) Primal:

The Lagrangian:

$L(x, \nu, \lambda) = \rho^T x + \nu^T (Ix - b) + \lambda_1^T (x - 1) - \lambda_2^T x$

where ρ is the objective, ν is the equality constraint, and λ are the inequality const

2) Dual Problem:

Minimize Lagrangian w.r.t x

$\frac{dL}{dx} = \rho + \nu I + \lambda_1 - \lambda_2 = 0$

$g(\lambda) = \rho + \nu I + \lambda_1 - \lambda_2$

since $\lambda_2 \geq 0$

$\rho + \nu I + \lambda_1 = \lambda_2'$

$\rho + \nu I + \lambda_1 \geq 0$

$\boxed{\nu I + \lambda_1 \geq -\rho}$ Constraint

$\rho \geq -\nu I + \lambda_1$ at optimality

$\rho = -\nu I + \lambda_1$

Substitute in Lagrange

$L(x, \nu, \lambda) = (-\nu I + \lambda_1)^T x + \nu^T (Ix - b) + \lambda_1^T (x - 1) - \lambda_2^T x$

$= -\nu^T x + \nu^T x + \nu^T Ix - \nu^T b + \lambda_1^T x - \lambda_1^T x - \lambda_1^T I$

$= -\nu^T b - \lambda_1^T$

$g(\lambda) = -\nu^T b - \lambda_1^T$

maximize $-\nu^T b - I\lambda_1^T$ to minimize $\nu^T b + \lambda_1^T$

s.t. $\nu + \lambda_1 \geq -\rho$

$\lambda \geq 0$

6. Piecewise-linear minimization

6. Piecewise-linear minimization. Consider the problem:

$$\text{minimize} \quad \max_{i=1,\ldots,m}(a_i^T x - b_i)$$

- Show that it can be expressed as the smooth linear problem:

$$\begin{array}{ll} \text{minimize} & t \\ \text{subject to} & \begin{bmatrix} A & -\mathbf{1} \end{bmatrix} \begin{bmatrix} x \\ t \end{bmatrix} \leq b \end{array}$$

- Show that the dual may be written as:

$$\begin{array}{ll} \text{maximize} & -b^T \lambda \\ \text{subject to} & A^t \lambda = 0 \\ & \mathbf{1}^T \lambda = 1 \\ & \lambda \geq 0 \end{array}$$

- Write the KKT conditions of the dual problem.

- Smooth linear problem:

$$\begin{cases} \min \ t \\ \text{s.t } a_i^T x - b_i \leq t \quad \text{for } i = 1, \dots, m \end{cases}$$

rows of matrix A   b vector with components b_i

$$\min_{x,t} \ t$$

s.t $Ax - b \preceq t\mathbf{1}$ vector of all ones

$$\left.\begin{aligned} Ax - t\mathbf{1} &\preceq b \\ \begin{bmatrix} A & -\mathbf{1} \end{bmatrix} \begin{bmatrix} x \\ t \end{bmatrix} &\preceq b \end{aligned}\right\} \text{inequality constraint}$$

- Dual Problem:

$$g(\lambda) = \text{minimum } \mathcal{L}(x, t, \lambda)$$
$$= \min (t + \lambda^T(Ax - \mathbf{1}t - b))$$
$$= \min (A^T\lambda x - \mathbf{1}^T\lambda t - b^T\lambda + t)$$
$$= \min (A^T\lambda x + (-\mathbf{1}^T\lambda + 1)t - b^T\lambda)$$

$$g(\lambda) = \begin{cases} -b^T\lambda & A^T\lambda x = 0 \quad -\mathbf{1}^T\lambda + 1 = 0 \\ -\infty \end{cases}$$

$$\rightarrow \max_{\lambda} \ -b^T\lambda$$
$$\text{s.t } A^T\lambda = 0$$
$$\mathbf{1}^T\lambda = 1$$
$$\lambda \geq 0$$

- KKT Conditions:

1) Primal feasibility: $A^T\lambda = 0$
   $\mathbf{1}^T\lambda = 1$

2) Dual feasibility: $\lambda \geq 0$

3) Complementary slackness: $\lambda_i (a_i^T x - b_i - t) = 0$ for all i

4) Stationarity: $\nabla \mathcal{L} = -b + A^T v_1 + I^T v_2 + \lambda^T \omega = 0$