Danny Abraham

CMPS 351

Assignment 8

In [1]:
```
import numpy as np
from numpy import linalg as la
from scipy.optimize import linprog as lp
import matplotlib.pyplot as plt
import cvxpy as cvx
import sympy as sp
import quadprog as qp
```

Problem in R

In [2]:
```
def f(x):
    return x**2+1
```

In [3]:
```
def g(x):
    return (6*x/(2+2*x))**2 + 1 + x*(6*x/(2+2*x)-2)*(6*x/(2+2*x)-4)
```

$$\min\ x^2 + 1$$

$$\textit{subject to} (x-2)(x-4) \leq 0$$

Lagrangian

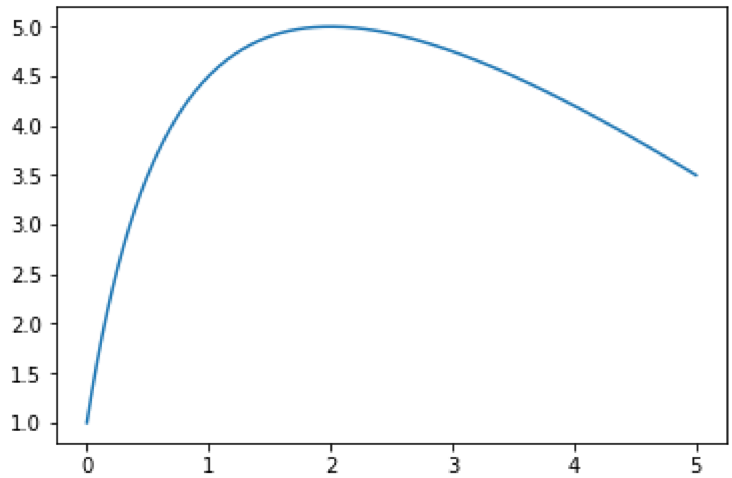
$$L(x, \lambda) = x^2 + 1 + \lambda(x-2)(x-4)$$

$$\nabla L_x = 0$$

$$x = \frac{6\lambda}{2+2\lambda}$$

$$g(\lambda) = (\frac{6\lambda}{2+2\lambda})^2 + 1 + \lambda(\frac{6\lambda}{2+2\lambda} - 2)(\frac{6\lambda}{2+2\lambda} - 4)$$

In [4]:
```
x = np.linspace(0,5,10000)
plt.plot(x,g(x))
plt.show()
```



In [5]:
```
x = cvx.Variable()
constraints = [
                -x<=-2,
                 x<= 4
              ]
obj = cvx.Minimize(f(x))
prob = cvx.Problem(obj, constraints)
prob.solve()
```

Out[5]: 5.000000000000002

In [6]:
```
x = sp.Symbol('x')

g_symb = [(6*x/(2+2*x))**2 + 1 + x*(6*x/(2+2*x)-2)*(6*x/(2+2*x)-4)]
g = sp.lambdify([x], g_symb, 'numpy')

derivative = sp.derive_by_array(g_symb, x)
g_div = sp.lambdify([x], derivative, 'numpy')
```

In [7]:
```
sp.solvers.solve(derivative)
```

Out[7]: [{x: -4}, {x: 2}]

In [8]:
```
g(2)
```

Out[8]: [5.0]

$$\lambda^* = 2$$

$$g^* = 5$$

$$f^* = 5$$

Dual in R^n

$$min\ p^t x$$

$$subject\ to \qquad 0 \leq x \leq 1$$

$$1^t x = b$$

Lagragian

$$L(x, \lambda, \nu) = p^t x - \lambda^t (x - 1) - \lambda_2^t (x) - \nu(1^t x - b)$$

$$L(x, \lambda, \nu) = (p^t - \lambda^t - \lambda_2^t - \nu 1^t)x + b\nu + \lambda^t 1$$

Dual Objective

$$\begin{array}{rc} & G(\lambda, \nu) = b\nu + \lambda^t 1 \\ subject\ to & p - \lambda - \lambda_2 - \nu = 0 \\ & \lambda \geq 0 \end{array}$$

$$\begin{array}{rc} & G(\lambda, \nu) = b\nu + \lambda^t 1 \\ subject\ to & \lambda + \nu \leq p \\ & \lambda \geq 0 \end{array}$$

In [9]:
```
res = lp([1], A_eq=[[1]], b_eq =[1],bounds = [(0,1)], options={"disp": True})
```

```
The solution was determined in presolve as there are no non-trivial constraint
s.
         Current function value: 1.000000
         Iterations: 0
```

In [10]:
```
res2 = lp([-1,-1], A_ub=[[1,1]], b_ub = [1], options={"disp": True})
```

```
Optimization terminated successfully.
         Current function value: -1.000000
         Iterations: 1
```

We test out this equation in linprog with an example in which we have: $b = 1$ $p = 1$ Since we are mnimizing $-G(\lambda, \nu)$ we can say that $max\ G(\lambda, \nu) = 1$ thus, we have strong duality in this case since f - g = 0

Dual of a Quadratic Problem

Primal Problem

$$minimize\ x^t x$$

$$subject\ to \qquad Ax = b$$

$$\lambda \geq 0$$

Lagrangian

$$L(x, \lambda) = x^t x - \lambda^t (Ax - b)$$

$$\frac{\partial \lambda}{\partial x} = 2x - A^t \lambda$$

$$\therefore x = \frac{A^t \lambda}{2}$$

Dual Problem

$$g(x, \lambda) = \frac{(A^t \lambda)^t (A^t \lambda)}{4} - \lambda^t (\frac{AA^t \lambda}{2} - b)$$

$$g(x, \lambda) = -\frac{\lambda^t AA^t \lambda}{4} + \lambda^t b$$

$$\lambda > 0$$

Piece-wise Linear Minimization

$$\min t$$

$$subject\ to\ \begin{bmatrix} A & -1 \end{bmatrix} \begin{bmatrix} x \\ t \end{bmatrix} \leq b$$

Lagrangian

$$L(x, \lambda, \nu) = t + \lambda^t (Ax - 1t - b)$$

$$L(x, \lambda, \nu) = (1 - \lambda^t 1)t + \lambda^t Ax - \lambda^t b$$

$$\nabla L_x = 0$$

$$\frac{d}{dx} \lambda^t Ax = A^t \lambda = 0$$

$$\frac{d}{dt}(t - \lambda^t 1t) = 1 - 1^t \lambda = 0$$

$$1^t \lambda = 1$$

$$\lambda \geq 0$$

Dual Objective

$$\max -b^t\lambda$$

$$\begin{aligned} subject\ to\ A^t\lambda &= 0 \\ 1^t\lambda &= 1 \\ \lambda &\geq 0 \end{aligned}$$

KKT Conditions

$$L(x_1, x_2, x_3, \lambda) = -b^t\lambda - x_1^t(A^t\lambda) - x_2^t(1^t\lambda - 1) - x_3^t(\lambda)$$

$$\nabla L_\lambda = -b - Ax_1 - 1x_2 - 1x_3 = 0$$

$$\begin{aligned} A^t\lambda &= 0 \\ 1^t\lambda &= 1 \\ \lambda &\geq 0 \end{aligned}$$

$$\lambda_i x_i = 0$$

$$\begin{aligned} x_1 &\geq 0 \\ x_2 &\geq 0 \\ x_3 &\geq 0 \end{aligned}$$

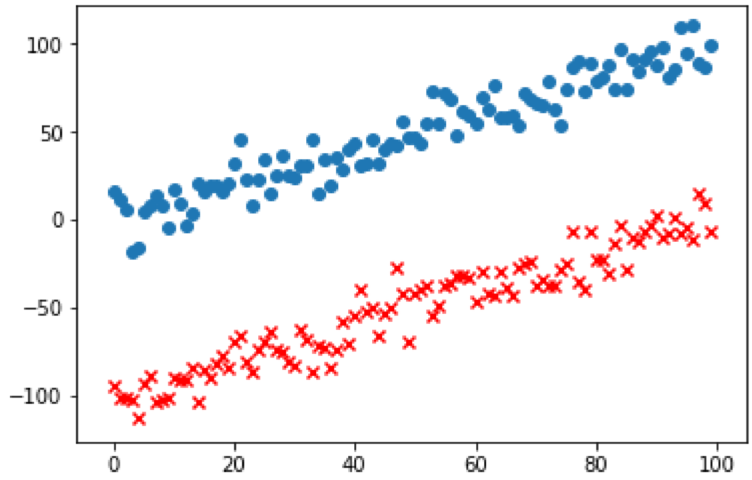
Classification

Linearly Separable

In [11]:
```
size = 100
mult = 10
r = np.random.normal(size=size)
x = np.array(range(size))
r *= mult
r += x
r2 = np.random.normal(size=size)
x = np.array(range(size))
r2 *= mult
r2 -= x[::-1]
r = np.array(r)
r2 = np.array(r2)

y = np.array([r2,x])
x = np.array([r,x])
```

In [12]:
```
plt.scatter(range(size), r)
plt.scatter(range(size), r2, color='red', marker='x')
```

Out[12]: <matplotlib.collections.PathCollection at 0x1652e372898>



In [13]:
```
a = cvx.Variable(2)
b = cvx.Variable()
```

In [14]:
```
obj = cvx.Minimize(cvx.norm(a,2)/2)
```

In [15]:
```
x_constraints = [a.T * y.T[i] - b >=  1  for i in range(size)]
y_constraints = [a.T * x.T[i] - b <= -1 for i in range(size)]

constraints = x_constraints + y_constraints
```

In [16]:
```
prob = cvx.Problem(obj, constraints)
prob.solve()
```

Out[16]: 0.026271849996648815

In [17]:
```
a.value
```

Out[17]: array([-0.03698816, 0.03731911])

In [18]:
```
b.value
```

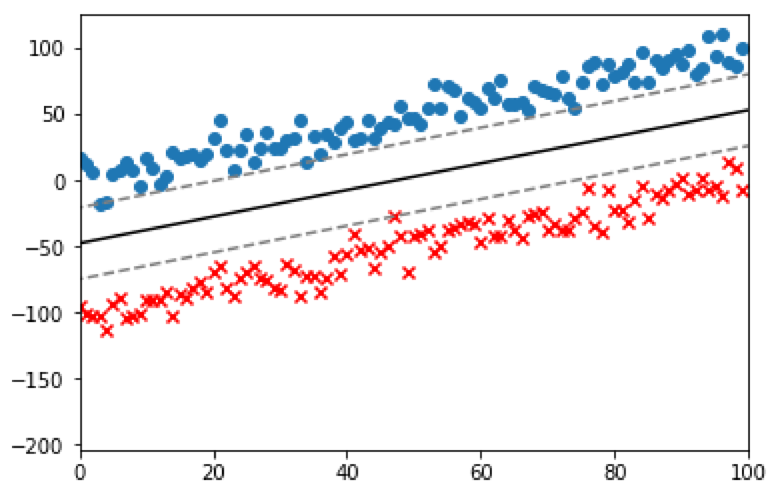
Out[18]: array(1.76417536)

In [19]:
```
a = a.value
b = b.value
```

In [20]:
```
plt.scatter(range(size), r)
plt.scatter(range(size), r2, color='red', marker='x')

d1_min = np.min([x.T[:,0],y.T[:,0]])
d1_max = np.max([x.T[:,0],y.T[:,0]])

d2_atD1min = (-a[1]*d1_min + b ) / a[0]
d2_atD1max = (-a[1]*d1_max + b ) / a[0]

print(d1_min)
print(d1_max)
print(d2_atD1min)
print(d2_atD1max)
plt.plot([d1_min,d1_max],[d2_atD1min,d2_atD1max],color='black')

sup_up_atD1min = (-a[1]*d1_min + b + 1 ) / a[0]
sup_up_atD1max = (-a[1]*d1_max + b + 1 ) / a[0]
sup_dn_atD1min = (-a[1]*d1_min + b - 1 ) / a[0]
sup_dn_atD1max = (-a[1]*d1_max + b - 1 ) / a[0]

plt.plot([d1_min,d1_max],[sup_up_atD1min,sup_up_atD1max],'--',color='gray')
plt.plot([d1_min,d1_max],[sup_dn_atD1min,sup_dn_atD1max],'--',color='gray')

plt.xlim([0, 100])
```

```
-112.99351956097266
110.30040347389146
-161.7001953037467
63.59163612946652
```

Out[20]: (0, 100)

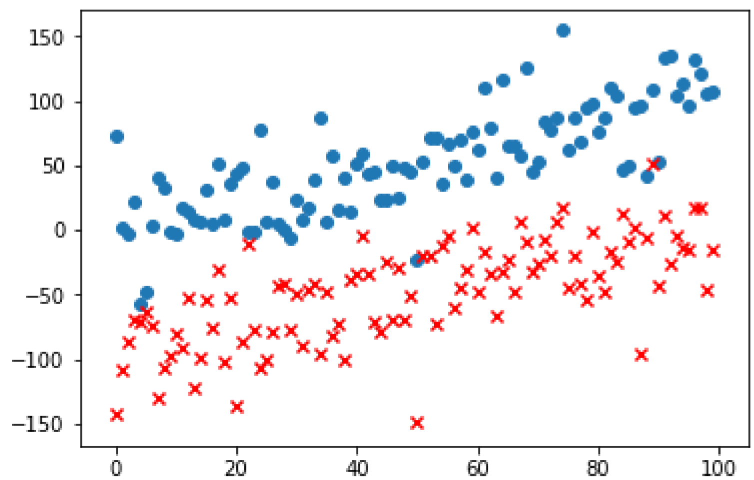


Nonlinearly Separable

In [21]:
```
size = 100
mult = 30
r = np.random.normal(size=size)
x = np.array(range(size))
r *= mult
r += x
r2 = np.random.normal(size=size)
x = np.array(range(size))
r2 *= mult
r2 -= x[::-1]
r = np.array(r)
r2 = np.array(r2)

y = np.array([r2,x])
x = np.array([r,x])
```

In [22]:
```
plt.scatter(range(size), r)
plt.scatter(range(size), r2, color='red', marker='x')
```

Out[22]: <matplotlib.collections.PathCollection at 0x16529371320>



In [23]:
```
a = cvx.Variable(2)
b = cvx.Variable()
```

In [24]:
```
obj = cvx.Minimize(cvx.norm(a,2)/2)
```

In [25]:
```
x_constraints = [a.T * y.T[i] - b >=  1  for i in range(size)]
y_constraints = [a.T * x.T[i] - b <= -1 for i in range(size)]

constraints = x_constraints + y_constraints
```

In [26]:
```
prob = cvx.Problem(obj, constraints)
prob.solve()
```

Out[26]: inf

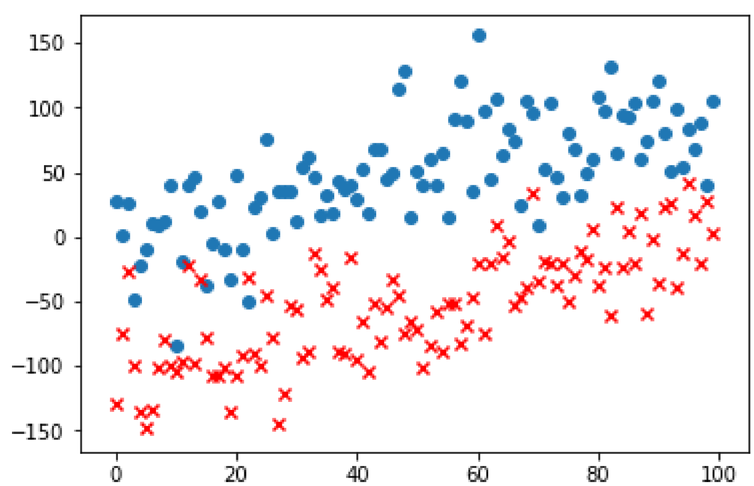
The problem is infeasible

Pareto Optimization

In [27]:
```
size = 100
mult = 30
r = np.random.normal(size=size)
x = np.array(range(size))
r *= mult
r += x
r2 = np.random.normal(size=size)
x = np.array(range(size))
r2 *= mult
r2 -= x[::-1]
r = np.array(r)
r2 = np.array(r2)

y = np.array([r2,x])
x = np.array([r,x])
```

In [28]:
```
plt.scatter(range(size), r)
plt.scatter(range(size), r2, color='red', marker='x')
```

Out[28]: <matplotlib.collections.PathCollection at 0x1652e4f76a0>



In [29]:
```
a = cvx.Variable(2)
b = cvx.Variable()
u = cvx.Variable(200)
```

In [30]:
```
alpha = 0.5
obj = cvx.Minimize(cvx.norm(a,2)/2 + alpha*np.ones(200)@u)
```

In [31]:
```
x_constraints = [a.T * y.T[i] - b + u[i]>=  1 for i in range(size)]
y_constraints = [a.T * x.T[i] - b - u[i + 100]<= -1 for i in range(size)]
u_constraints = [u[i] >= 0 for i in range(200)]

constraints = x_constraints + y_constraints + u_constraints
```

In [32]:
```
prob = cvx.Problem(obj, constraints)
prob.solve()
```

Out[32]:
```
13.94687306276909
```

In [33]:
```
a.value
```

Out[33]:
```
array([-0.06033148,  0.05566791])
```

In [34]:
```
b.value
```

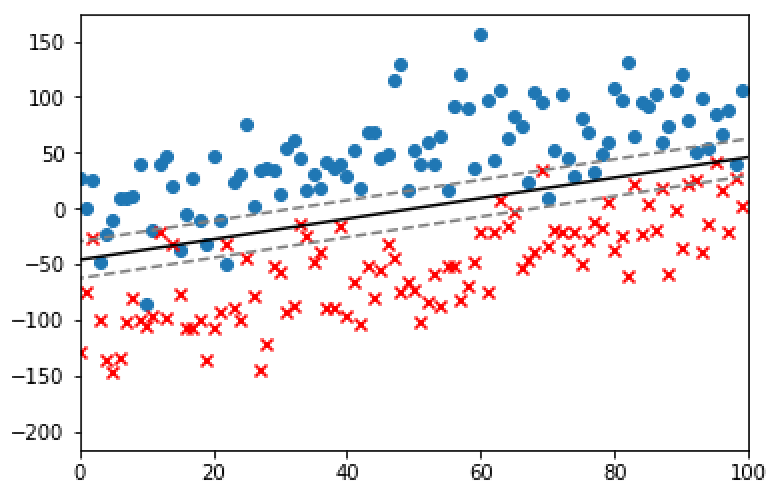
Out[34]:
```
array(2.78669228)
```

In [35]:
```
a = a.value
b = b.value
```

In [36]:
```
plt.scatter(range(size), r)
plt.scatter(range(size), r2, color='red', marker='x')

d1_min = np.min([x.T[:,0],y.T[:,0]])
d1_max = np.max([x.T[:,0],y.T[:,0]])

d2_atD1min = (-a[1]*d1_min + b ) / a[0]
d2_atD1max = (-a[1]*d1_max + b ) / a[0]

print(d1_min)
print(d1_max)
print(d2_atD1min)
print(d2_atD1max)
plt.plot([d1_min,d1_max],[d2_atD1min,d2_atD1max],color='black')

sup_up_atD1min = (-a[1]*d1_min + b + 1 ) / a[0]
sup_up_atD1max = (-a[1]*d1_max + b + 1 ) / a[0]
sup_dn_atD1min = (-a[1]*d1_min + b - 1 ) / a[0]
sup_dn_atD1max = (-a[1]*d1_max + b - 1 ) / a[0]

plt.plot([d1_min,d1_max],[sup_up_atD1min,sup_up_atD1max],'--',color='gray')
plt.plot([d1_min,d1_max],[sup_dn_atD1min,sup_dn_atD1max],'--',color='gray')

plt.xlim([0, 100])
```

```
-147.28785583657998
155.99337290299803
-182.0923286881247
97.74553620405995
```

Out[36]: (0, 100)



In [37]:
```
l = np.linspace(0, 2, 99)
```

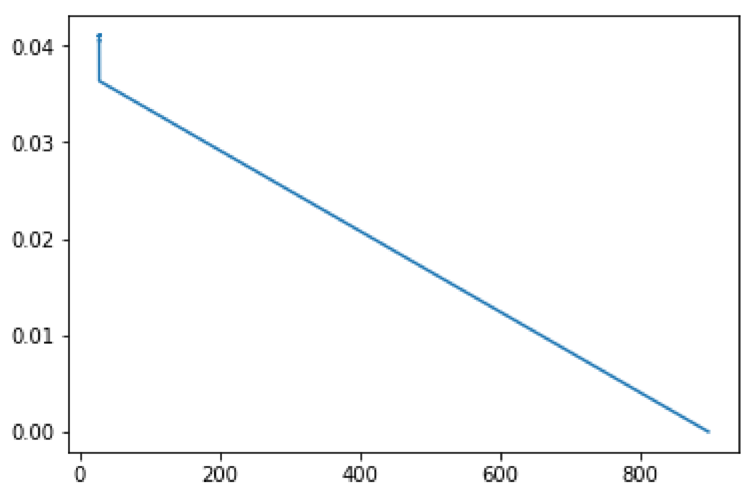
In [38]:
```
a = cvx.Variable(2)
b = cvx.Variable()
u = cvx.Variable(200)

x_constraints = [a.T * y.T[i] - b + u[i]>=  1 for i in range(size)]
y_constraints = [a.T * x.T[i] - b - u[i + 100]<= -1 for i in range(size)]
u_constraints = [u[i] >= 0 for i in range(200)]

constraints = x_constraints + y_constraints + u_constraints
```

In [39]:
```
term1 = []
term2 = []

for i, alpha in enumerate(l):
    obj = cvx.Minimize(cvx.norm(a,2)/2 + alpha*np.ones(200)@u)
    prob = cvx.Problem(obj, constraints)
    prob.solve()
    term1.append(0.5*la.norm(a.value))
    term2.append(np.ones(200)@u.value)
    print(i, end='\r')
```

```
98
```

In [40]:
```
term1 = np.array(term1)
term2 = np.array(term2)
plt.plot(term2, term1)
```

Out[40]:
```
[<matplotlib.lines.Line2D at 0x1652d0c3160>]
```



Dual

$$\min \ \frac{1}{2}a^t a + \alpha 1^t u$$

$$subject\ to\ D(Xa - b1) \geq 1 - u$$

$$L(a, b, u, \lambda, \sigma) = \frac{1}{2}a^t a + \alpha 1^t u - \lambda^t (D(XA - b1) - 1 + u) - \sigma^t u$$

$$L(a, b, u, \lambda, \sigma) = \frac{1}{2}a^t a + \alpha 1^t u - \lambda^t DXa + \lambda^t Db1 + \lambda^t D1 + \lambda^t Du - \sigma^t u$$

$$L(a, b, u, \lambda, \sigma) = \frac{1}{2}a^t a - \lambda^t DXa + \lambda^t Db1 + (\alpha 1^t - \lambda^t - \sigma^t)u + \lambda^t 1$$

$$\nabla L_a = 0$$
$$a - X^t D\lambda = 0$$
$$a = X^t D\lambda$$

$$\therefore$$

$$\nabla L_b = 0$$
$$1^t D\lambda = 0$$
$$\nabla L_u = 0$$
$$\alpha 1^t - \lambda^t - \sigma^t = 0$$
$$\lambda^t + \sigma^t = \alpha 1^t$$

However sigma is alpways > 0

$$\lambda^t \leq \alpha 1^t$$

$$\max \ -\frac{1}{2}\lambda^t DXX^t D\lambda$$
$$1^t D\lambda = 0$$
$$0 \leq \lambda \leq \alpha 1$$

In [41]:
```
z = np.ones(200)
z[100:] = -1*z[100:]
D = np.diag(z)
```

In [42]:
```
temp = np.ones([200,2])
temp[:100] = x.T
temp[100:] = y.T
```

In [43]:
```
G = -(np.dot(D,temp)@np.dot(temp.T,D))-0.000000001*np.identity(200)
```

In [44]:
```
x = cvx.Variable(200)
temp2 = np.ones(200).T@D
I = np.identity(200)

constraints_1 = [temp2*x == 0]
constraints_2 = [I*x <= 0.5]
constraints_3 = [I*x >= 0]

constraints = constraints_1 + constraints_2 + constraints_3

obj = cvx.Maximize(cvx.quad_form(x,G)+np.ones([1,200])*x)
prob = cvx.Problem(obj,constraints)
prob.is_qp()
prob.solve(verbose = True)
```

```
-----------------------------------------------------------------
           OSQP v0.5.0  -  Operator Splitting QP Solver
              (c) Bartolomeo Stellato,  Goran Banjac
        University of Oxford  -  Stanford University 2018
-----------------------------------------------------------------
problem:  variables n = 200, constraints m = 401
          nnz(P) + nnz(A) = 20700
settings: linear system solver = qdldl,
          eps_abs = 1.0e-04, eps_rel = 1.0e-04,
          eps_prim_inf = 1.0e-04, eps_dual_inf = 1.0e-04,
          rho = 1.00e-01 (adaptive),
          sigma = 1.00e-06, alpha = 1.60, max_iter = 10000
          check_termination: on (interval 25),
          scaling: on, scaled_termination: off
          warm start: on, polish: on

objective    pri res    dua res    rho        time
   1  -9.0986e-02   2.14e-03   1.92e+00   1.00e-01   6.84e-03s
 200  -1.4048e+01   1.48e-02   1.68e-01   1.21e-03   1.34e-02s
 400  -1.4051e+01   1.25e-02   3.23e-02   1.21e-03   3.60e-02s
 600  -1.3919e+01   7.81e-03   2.51e-02   1.21e-03   4.86e-02s
 800  -1.3855e+01   7.70e-03   9.69e-03   1.21e-03   6.14e-02s
1000  -1.4021e+01   1.60e-02   7.54e-03   2.28e-04   1.57e-02s
1200  -1.3824e+01   2.22e-02   9.25e-04   2.28e-04   3.39e-02s
1400  -1.3940e+01   3.81e-03   2.42e-03   2.28e-04   5.17e-02s
1600  -1.3936e+01   9.29e-03   2.57e-03   2.28e-04   6.59e-02s
1800  -1.3845e+01   1.31e-02   1.12e-03   2.28e-04   7.88e-02s
2000  -1.3947e+01   6.58e-03   1.92e-03   2.28e-04   9.14e-02s
2200  -1.3926e+01   4.87e-03   2.87e-03   2.28e-04   1.03e-01s
2400  -1.3850e+01   1.12e-02   1.08e-03   2.28e-04   1.14e-01s
2600  -1.3950e+01   7.27e-03   1.81e-03   2.28e-04   1.28e-01s
2800  -1.3919e+01   2.75e-03   2.90e-03   2.28e-04   1.39e-01s
3000  -1.3854e+01   9.76e-03   1.15e-03   2.28e-04   1.52e-01s
3200  -1.3985e+01   9.73e-03   5.04e-03   2.00e-04   1.33e-02s
3400  -1.3892e+01   1.40e-02   1.30e-02   1.40e-03   1.75e-02s
3600  -1.3916e+01   8.07e-04   2.04e-02   1.40e-03   2.92e-02s
3800  -1.3929e+01   6.08e-04   1.71e-02   1.40e-03   4.14e-02s
4000  -1.3871e+01   1.99e-03   2.92e-03   1.40e-03   5.43e-02s
4200  -1.3921e+01   1.15e-03   1.88e-02   1.40e-03   7.05e-02s
4400  -1.3932e+01   2.78e-03   1.49e-02   1.40e-03   8.32e-02s
4600  -1.3868e+01   4.92e-03   1.77e-03   1.40e-03   9.55e-02s
```

```
4800  -1.3924e+01   2.09e-03   1.48e-02   1.40e-03   1.08e-01s
5000  -1.3994e+01   4.15e-03   5.22e-03   2.73e-04   9.39e-03s
5200  -1.3874e+01   1.28e-02   7.65e-03   2.73e-04   2.12e-02s
5400  -1.3976e+01   1.26e-02   2.15e-03   2.73e-04   3.28e-02s
5600  -1.3864e+01   3.86e-03   2.35e-03   2.73e-04   4.49e-02s
5800  -1.3890e+01   6.07e-03   4.48e-03   2.73e-04   5.68e-02s
6000  -1.3970e+01   1.13e-02   2.26e-03   2.73e-04   6.91e-02s
6200  -1.3858e+01   7.21e-03   2.22e-03   2.73e-04   8.18e-02s
6400  -1.3898e+01   3.02e-03   4.30e-03   2.73e-04   9.34e-02s
6600  -1.3966e+01   1.05e-02   2.07e-03   2.73e-04   1.05e-01s
6800  -1.3680e+01   6.03e-02   6.17e-03   2.04e-04   1.34e-02s
7000  -1.3941e+01   8.69e-03   4.42e-03   2.04e-04   2.56e-02s
7200  -1.3833e+01   2.52e-02   2.85e-03   2.04e-04   3.76e-02s
7400  -1.3922e+01   1.22e-02   2.31e-03   2.04e-04   4.97e-02s
7600  -1.3960e+01   8.22e-03   1.70e-03   2.04e-04   6.22e-02s
7800  -1.3846e+01   1.54e-02   2.27e-03   2.04e-04   8.14e-02s
8000  -1.3918e+01   6.19e-03   2.49e-03   2.04e-04   9.49e-02s
8200  -1.3955e+01   7.48e-03   1.43e-03   2.04e-04   1.08e-01s
8400  -1.3851e+01   1.18e-02   1.54e-03   2.04e-04   1.21e-01s
8600  -1.3919e+01   3.92e-03   2.64e-03   2.04e-04   1.33e-01s
8800  -1.3951e+01   7.04e-03   1.41e-03   2.04e-04   1.44e-01s
9000  -1.3853e+01   1.02e-02   1.08e-03   2.04e-04   1.56e-01s
9200  -1.3921e+01   3.25e-03   2.59e-03   2.04e-04   1.68e-01s
9400  -1.3947e+01   6.51e-03   1.71e-03   2.04e-04   1.81e-01s
9600  -1.3854e+01   9.54e-03   7.41e-04   2.04e-04   1.93e-01s
9800  -1.3924e+01   3.29e-03   2.48e-03   2.04e-04   2.06e-01s
10000  -1.3995e+01   1.26e-03   2.94e-03   2.07e-04   9.48e-03s

status:               solved inaccurate
number of iterations: 10000
optimal objective:    -13.9954
run time:             9.87e-03s
optimal rho estimate: 3.07e-04
```

Out[44]:
```
13.995354502866832
```

There is a nonzero duality gap. The solution of the dual problem here acts as a lower bound to the primal problem $g^* \leq f^*$