hw5

September 23, 2018

1 Homework 5

1.1 Question 1

Consider the following linear program. Then Answer the following questions.

$$\begin{aligned} \min\ & 3x_1 + x_2 \\ \text{s.t. } & x_1 + 2x_2 \geq 2 \\ & 2x_1 + x_2 \geq 3 \\ & x_1 \geq 0 \\ & x_2 \geq 0 \end{aligned}$$

(a) Draw the feasible region of this LP in (x_1, x_2).

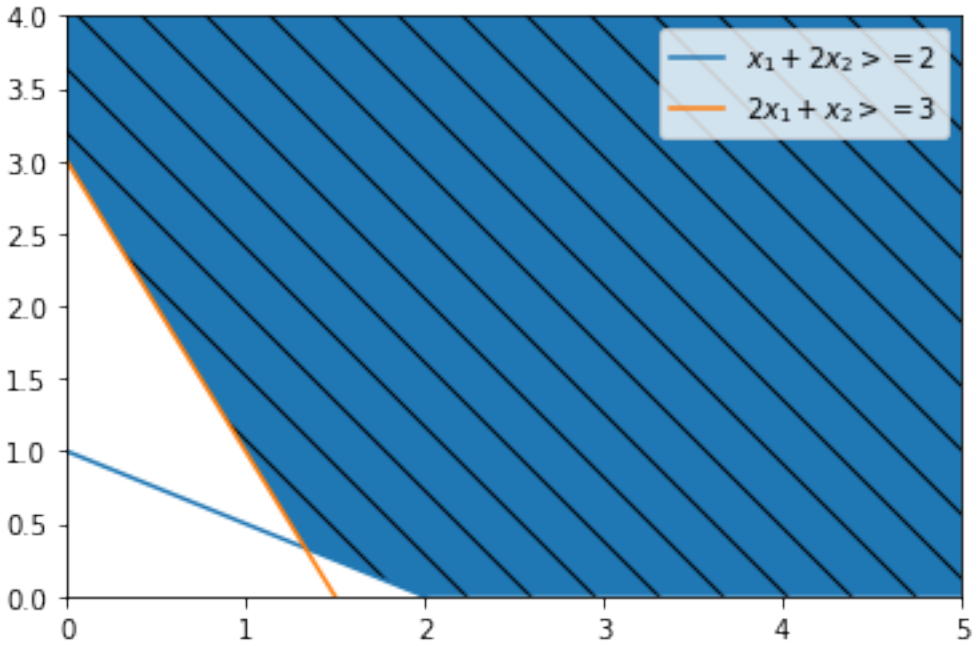
In [3]:
```
#plot the solution using matplotlib
import matplotlib.pyplot as plt
import numpy as np

# x-values for our plot
xmax = 5
ymax = 4
x = np.arange(0, xmax, 0.1)

# the constraints to plot
y1 = 1 - x / 2.
y2 = 3 - 2.*x

# plot the constraints
plt.xlim(0, xmax)
plt.ylim(0, ymax)
plt.plot(x, y1, x, y2, label='Feasible Region')
plt.legend([r'$x_1 + 2x_2 >= 2$', r'$2x_1 + x_2 >= 3$']);

# fill in the feasable region (using a polygon)
xp = [0, 0, xmax, xmax, 2, 4./3.]
yp = [3, ymax, ymax, 0, 0, 1./3.]
plt.fill(xp ,yp, hatch='\\');
```



(b) Find the optimal solution using the picture of the feasible region. Hint: The optimal solution should be one of the corner points of the feasible region. The optimal solution may be at (x_1, x_2) of $(0,3)$, $(\frac{4}{3}, \frac{1}{3})$, or $(2,0)$. This yields objective values of 3, $\frac{13}{3}$, and 6 repsectively. Therefore the optimal objective is at $(0,3)$ with value 3.

(c) Write a CVXPY code to find the optimal solution.

```
In [4]: import cvxpy as cp
        import numpy as np

        #setup variables and coeffcients
        x = cp.Variable(2, 1)
        c = np.array([3., 1.])
        A = np.array([[1.,2.],[2.,1.],[1.,0.],[0.,1.]])
        b = np.array([2., 3., 0., 0.])

        #setup objective and constraints
        objective = cp.Minimize(c*x)
        constraints = [A*x >= b]

        # solve
        prob = cp.Problem(objective, constraints)
        result = prob.solve()

        # display optimal value of variables
```

```
        print('The optimal value is ', round(result))
        print('The optimal x1, x2 is ', [round(xn) for xn in x.value])
```

```
The optimal value is  3.0
The optimal x1, x2 is  [-0.0, 3.0]
```

1.2 Question 2

Consider a transportation problem with 4 suppliers and 3 customers. The supply for each supplier s_i are given as $s_1 = 10, s_2 = 25, s_3 = 18, s_4 = 15$. The demand for each consumer d_i are given as $d_1 = 15, d_2 = 20, d_3 = 16$. The unit transportation cost c_{ij} between supplier i and consumer j are given as: $c_{11} = 2, c_{12} = 3, c_{21} = 4, c_{23} = 5, c_{31} = 2, c_{32} = 3, c_{33} = 4, c_{41} = 5, c_{43} = 3$.

(a) Formulate a linear program to find the minimum total transportation cost to satisfy all the demand (the demand can be exceeded). Write down the LP with the given data. Let E be the set of edges between the supplier and consumer. Thus $E = \{(1,1),(1,2),(2,1),(2,3),(3,1),(3,2),(3,3),(4,1),(4,3)\}$

Let x_{ij} be the demand for consumer j satisfied by supplier i. This yields the following formulation.

$$\begin{aligned}
\min \quad & \sum_{E} x_{ij} c_{ij} && \forall (i,j) \in E \\
\text{s.t.} \quad & \sum_{j=1}^{3} x_{ij} \leq s_i && \forall i \in \{1,2,3,4\} \\
& \sum_{i=1}^{4} x_{ij} \geq d_j && \forall j \in \{1,2,3\} \\
& x_{ij} \geq 0 && \forall (i,j) \in E
\end{aligned}$$

(b) Write a CVXPY code to find the optimal solution of the above LP.

In [5]:
```
# 9 variables for the connected nodes
i = 4
j = 3
n = 9

x = cp.Variable(n, 1)

# cost variables as given
c = np.array([2., 3., 4., 5., 2., 3., 4., 5., 3.])
bs = np.array([10., 25., 18., 15.])
bd = np.array([15., 20., 16.])

S = np.zeros((i,n))
S[0,0] = 1.;S[0,1] = 1;
S[1,2] = 1.;S[1,3] = 1;
S[2,4] = 1.;S[2,5] = 1;S[2,6] = 1.;
S[3,7] = 1.;S[3,8] = 1;
```

```
        D = np.zeros((j,n))
        D[0,0] = 1.;D[0,2] = 1.;D[0,4] = 1.;D[0,7] = 1.;
        D[1,1] = 1.;D[1,5] = 1.;
        D[2,3] = 1.;D[2,6] = 1.;D[2,8] = 1.;

        #setup objective and constraints
        objective = cp.Minimize(c*x)
        constraints = [S*x <= bs, D*x >= bd, x >= 0]

        # solve
        prob = cp.Problem(objective, constraints)
        result = prob.solve()

        # display optimal value of variables
        print('The optimal value is ', round(result))
        print('The optimal x is ', [round(xij) for xij in x.value])
```

```
The optimal value is  154.0
The optimal x is  [3.0, 7.0, 7.0, 1.0, 5.0, 13.0, 0.0, -0.0, 15.0]
```

(c) Modify your code so that every demand is satisfied exactly, i.e. cannot be exceeded.

```
In [6]: # modify constraint to ==
        constraints = [S*x <= bs, D*x == bd, x >= 0]

        # solve
        prob = cp.Problem(objective, constraints)
        result = prob.solve()

        # display optimal value of variables
        print('The optimal value is ', round(result))
        print('The optimal x is ', [round(xij) for xij in x.value])
```

```
The optimal value is  154.0
The optimal x is  [1.0, 9.0, 7.0, 1.0, 7.0, 11.0, 0.0, 0.0, 15.0]
```

The previous result where the demand constraint was changed to an equality yields a different solution but with the same objective value.

1.3 Question 3

Consider the following electric power network taken from a real-world electric power system. Electricity generators are located at nodes 1, 3, and 5. Electricity loads are located at nodes 2, 4, and 6.

Denote the amount of generation of generator i as p_i and the demand of consumer j as d_j.

The demand is given as $d_1 = 90$, $d_2 = 100$, $d_3 = 125$. The range of generation lower and upper bounds are given as $p_1^{min} = 10$, $p_1^{max} = 250$, $p_2^{min} = 10$, $p_2^{max} = 300$, $p_3^{min} = 10$, $p_3^{max} = 270$.

The flow limit on lines are given as $f_{12}^{max} = 50$, $f_{23}^{max} = 60$, $f_{34}^{max} = 90$, $f_{45}^{max} = 50$, $f_{56}^{max} = 120$, $f_{61}^{max} = 100$.

The line parameters are given as $B_{12} = 11.6$, $B_{23} = 5.9$, $B_{34} = 13.7$, $B_{45} = 9.8$, $B_{56} = 5.6$, $B_{61} = 10.5$.

The unit generation costs are given as $c_1 = 5$, $c_2 = 2$, $c_3 = 3$.

(a) Formulate the power system scheduling problem using the model discussed in Lecture 2.

let $\mathbf{E} = \{(1,2),(2,3),(3,4),(4,5),(5,6),(6,1)\}$

$$\min \sum_{i=1}^{3} p_i c_i$$

$$\begin{aligned}
\text{s.t. } & p_i^{min} \le p_i \le p_i^{max} && \forall i \in \{1,2,3\} \\
& -f_{ij}^{max} \le f_{ij} \le f_{ij}^{max} && \forall (i,j) \in \mathbf{E} \\
& f_{ij} = B_{ij}(\theta_i - \theta_j) && \forall (i,j) \in \mathbf{E} \\
& f_{12} - f_{61} = p_1 \\
& -f_{23} + f_{34} = p_2 \\
& -f_{45} + f_{56} = p_3 \\
& f_{12} - f_{23} = d_1 \\
& f_{34} - f_{45} = d_2 \\
& -f_{61} + f_{56} = d_3
\end{aligned}$$

(b) Implement and solve the model using CVXPY. Write down the optimal solution.

In [71]:
```
# 6 variables for 6 node potentials (thetas)
n = 6

x = cp.Variable(n, 1) #theta for each node

# variables as given
c = np.array([5., 2., 3.])
B = np.array([11.6, 5.9, 13.7, 9.8, 5.6, 10.5])
pmin = np.array([10., 10., 10.])
pmax = np.array([250., 300., 270.])
fmax = np.array([50., 60., 90., 50., 120., 100.])
d = np.array([90., 100., 125.])

#generator node flow conservation (as a function of theta and B)
P = np.array([[B[0]+B[5], -B[0], 0., 0., 0., -B[5]],
              [0., -B[1], B[1]+B[2], -B[2], 0., 0.],
              [0., 0., 0., -B[3], B[3]+B[4], -B[4]]])

#demand node flow conservation (as a function of theta and B)
D = np.array([[B[0], -B[0]-B[1], B[1], 0., 0., 0.],
              [0., 0., B[2], -B[2]-B[3], B[3], 0.],
```

```
                [B[5], 0., 0., 0., B[4], -B[4]-B[5]]])

        #flow limits (as a function of theta and B - f_ij = B_ij(theta_i-theta_j))
        F = np.array([[B[0], -B[0], 0, 0, 0, 0],
                [0, B[1], -B[1], 0, 0, 0],
                [0, 0, B[2], -B[2], 0, 0],
                [0, 0, 0, B[3], -B[3], 0],
                [0, 0, 0, 0, B[4], -B[4]],
                [-B[5], 0, 0, 0, 0, B[5]]])

        #setup objective and constraints
        objective = cp.Minimize(P*x*c)
        constraints = [P*x <= pmax, P*x >= pmin, D*x == d, F*x >= -fmax, F*x <= fmax]

        # solve
        prob = cp.Problem(objective, constraints)
        result = prob.solve()

        # display optimal value of variables
        print('The solver status is', prob.status)
        print('The optimal value is', round(result, 2))
        print('The optimal x (theta_1, ..., theta_6) is', [round(xi,2) for xi in x.value])
        #print(P)
        #print(D)
        #print(F)
        #print(prob)
        #print(x.value)
        print('The power generation values (p_1, p_2, p_3) are', [round(p,2) for p in np.dot(P,
```

```
The solver status is optimal
The optimal value is 992.04
The optimal x (theta_1, ..., theta_6) is  [-2.28, -4.87, 5.3, 1.65, 6.75, -6.9]
The power generation values (p_1, p_2, p_3) are [78.52, 110.0, 126.48]
```

(c) Find the electricity prices for demand at nodes 2, 4, 6. To do this, use the command constraints[0].dual value to find the dual variable of constraints[0]. Hint: Recall the electricity price at node i is the dual variable for the flow conservation constraint at node *i*. The demand nodes are represented by the constraint D*x == d, which is the 3rd item in the list of constraint variables.

```
In [66]: print(constraints[2].dual_value)
```

```
[-5.62968516 -2.53316407 -4.30434783]
```