ISyE6669 HW05 - Graham Billey, Spring 2020

In [167]:
```
# import sys
# !pip install --prefix {sys.prefix} cvxpy
```

In [168]:
```
import numpy as np
import matplotlib.pyplot as plt
import cvxpy as cp
```

1) Consider the following linear program. Answer the following questions.

$min\ 3x_1 + x_2$

$s.t.\ 3x_1 + 2x_2 \geq 6$

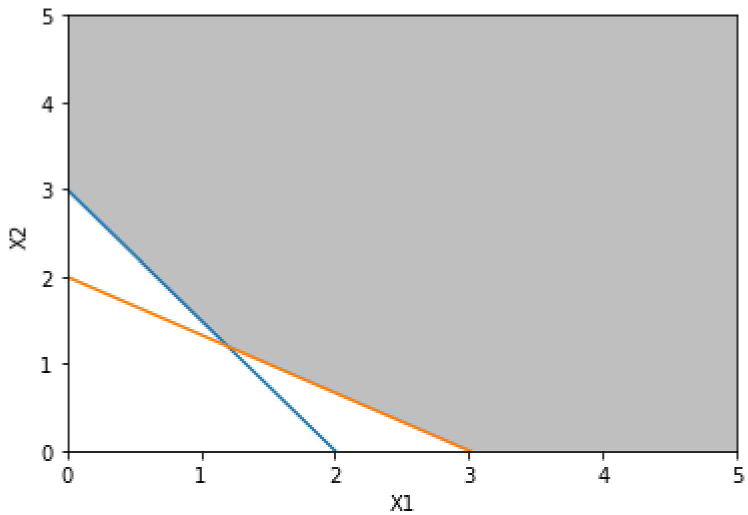
$2x_1 + 3x_2 \geq 6$

$x_1 \geq 0, x_2 \geq 0$

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

In [169]:
```
x = np.linspace(0, 10, 100)
b1 = 3 - (3/2)*x
b2 = 2 - (2/3)*x
boundary = np.maximum(b1, b2)

plt.plot(x, b1)
plt.plot(x, b2)
plt.xlim((0, 5))
plt.ylim((0, 5))
plt.xlabel('X1')
plt.ylabel('X2')
plt.fill_between(x, boundary, 5, color='grey', alpha=0.5)
plt.plot(aspect='equal')
plt.show()
```



(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.

Since the gradient points along $\begin{bmatrix} 3 \\ 1 \end{bmatrix}$, the minimum will be in the opposite direction.

That would appear to me to be where the two lines cross, at `(1.2, 1.2)` **(Note this is wrong, but I'm leaving my original guess here anyway).**

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

In [170]:
```
# Define and solve the CVXPY problem.
x = cp.Variable(2)
# print('x: ', x.shape)

c = np.array([3, 1])
# print('c: ', c.shape)

A = np.array([[3, 2],
              [2, 3],
              [1, 0],
              [0, 1]])
# print('A: ', A.shape)

b = np.array([6, 6, 0, 0])
# print('b: ', b.shape)

#Objective function
obj = cp.Minimize(c.T@x)
prob = cp.Problem(obj, [A @ x >= b])
prob.solve(verbose=True)

print(f'The optimal value is: {prob.value}')
print(f'The optimal x is: {x.value}')
```

```
-----------------------------------------------------------------
           OSQP v0.6.0  -  Operator Splitting QP Solver
              (c) Bartolomeo Stellato,  Goran Banjac
        University of Oxford  -  Stanford University 2019
-----------------------------------------------------------------
problem:  variables n = 2, constraints m = 4
          nnz(P) + nnz(A) = 6
settings: linear system solver = qdldl,
          eps_abs = 1.0e-05, eps_rel = 1.0e-05,
          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, time_limit: off

iter   objective    pri res    dua res    rho        time
   1  -1.2078e+01   1.69e+01   3.53e+00   1.00e-01   8.98e-05s
 200   3.0000e+00   5.77e-05   7.65e-05   1.00e-01   2.36e-04s
 225   3.0000e+00   8.45e-05   1.16e-05   1.00e-01   3.33e-04s
plsh   3.0000e+00   7.69e-16   1.33e-15   --------   4.31e-04s

status:               solved
solution polish:      successful
number of iterations: 225
optimal objective:    3.0000
run time:             4.31e-04s
optimal rho estimate: 1.56e-01

The optimal value is: 3.0000000000000004
The optimal x is: [6.42370214e-23 3.00000000e+00]
```

My original guess of `(1.2, 1.2)` was wrong, which is pretty obvious if you look at the objective function and compute the value at each corner point of the feasible region. The actual optimal is `(0, 3)`.

Problem 02

Consider a transportation problem with 4 suppliers and 3 customers. The amounts of supplyand demand are shown in Figure 1.

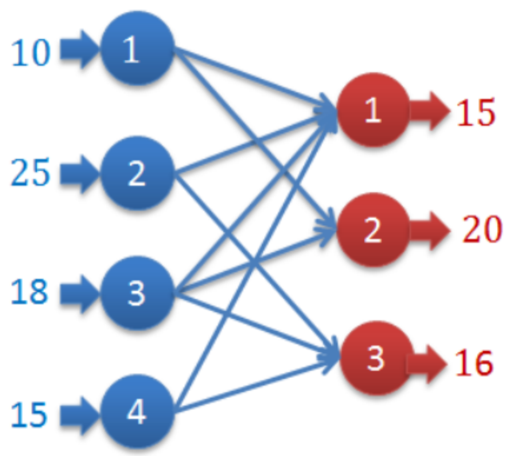


Figure 1: Transportation between 4 suppliers and 3 customers.

The unit transportation cost c_{ij} between supplier i and consumer j are given as
$c_{11} = 5,\ c_{12} = 4,\ c_{21} = 3,\ c_{23} = 2,\ c_{31} = 5,\ c_{32} = 4,\ c_{33} = 3,\ c_{41} = 2,\ c_{43} = 5$.

(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.

$min\ \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij}x_{ij}$ ----- Minimize the transportation cost between all suppliers and consumers

$s.t.\quad \sum_{i=1}^{m} x_{ij} \geq d_j$ ----- The total flow into consumer j is $\geq$ the demand of consumer j

$\sum_{j=1}^{n} x_{ij} \leq s_i$ ----- The total flow out of supplier i is $\leq$ the supply of supplier i

$x_{ij} \geq 0, \quad i = 1, \ldots, m, \quad j = 1, \ldots, n$ ----- Flow is non-negative

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

In [171]:
```
x = cp.Variable(shape=(4, 3))

#Constants
cost = np.array([[5, 4, 999],
                 [3, 999, 2],
                 [5, 4, 3],
                 [2, 999, 5]])
s = np.array([10, 25, 18, 15])
d = np.array([15, 20, 16])

# #Constraints
constraints = [cp.sum(x, axis=1) <= s, cp.sum(x, axis=0) >= d, x[0,2] == 0, x[
1,1] == 0, x[3,1] == 0, x >= 0]

#Objective function
obj = cp.Minimize(cp.sum(cost.T * x))

prob = cp.Problem(obj, constraints)
prob.solve(verbose=True)

print(f'The optimal value is: {prob.value}')
print(f'The optimal x is: \n{np.around(x.value,1)}')
print(f'The total amount of product produced is {np.around(np.sum(x.value),
1)}')
```

```
-----------------------------------------------------------------
           OSQP v0.6.0  -  Operator Splitting QP Solver
              (c) Bartolomeo Stellato,  Goran Banjac
        University of Oxford  -  Stanford University 2019
-----------------------------------------------------------------
problem:  variables n = 12, constraints m = 22
          nnz(P) + nnz(A) = 39
settings: linear system solver = qdldl,
          eps_abs = 1.0e-05, eps_rel = 1.0e-05,
          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, time_limit: off

iter   objective    pri res    dua res    rho        time
   1  -2.0424e+04   2.44e+01   2.44e+03   1.00e-01   8.65e-05s
 200   3.3366e+04   2.10e-03   6.40e-02   1.16e-02   3.81e-04s
 300   3.3368e+04   1.36e-04   9.60e-03   1.16e-02   5.86e-04s

status:               solved
solution polish:      unsuccessful
number of iterations: 300
optimal objective:    33368.0414
run time:             7.19e-04s
optimal rho estimate: 8.75e-03

The optimal value is: 33368.041437671745
The optimal x is:
[[ 0.   2.   0. ]
 [12.3 -0.  12.8]
 [ 0.  18.   0. ]
 [ 2.7 -0.   3.2]]
The total amount of product produced is 51.0
```

To solve this, I had to implement a very high cost where there is no actual connection between suppliers and consumers. I also forced the solution to equal `0` there when defining the constraints.

(c) Modify your code so that every demand is satisfied exactly, i.e. cannot be exceeded. You do not need to write down the LP model. What is the optimal solution? Is it the same as the first model?

In [172]:
```
x = cp.Variable(shape=(4, 3))

#Constants
cost = np.array([[5, 4, 999],
                 [3, 999, 2],
                 [5, 4, 3],
                 [2, 999, 5]])
s = np.array([10, 25, 18, 15])
d = np.array([15, 20, 16])

# #Constraints
constraints = [cp.sum(x, axis=1) <= s, cp.sum(x, axis=0) == d, x[0,2] == 0, x[
1,1] == 0, x[3,1] == 0, x >= 0]

#Objective function
obj = cp.Minimize(cp.sum(cost.T * x))

prob = cp.Problem(obj, constraints)
prob.solve(verbose=True)

print(f'The optimal value is: {prob.value}')
print(f'The optimal x is: \n{np.around(x.value,1)}')
print(f'The total amount of product produced is {np.around(np.sum(x.value),
1)}')
```

```
-----------------------------------------------------------------
           OSQP v0.6.0  -  Operator Splitting QP Solver
              (c) Bartolomeo Stellato,  Goran Banjac
        University of Oxford  -  Stanford University 2019
-----------------------------------------------------------------
problem:  variables n = 12, constraints m = 22
          nnz(P) + nnz(A) = 39
settings: linear system solver = qdldl,
          eps_abs = 1.0e-05, eps_rel = 1.0e-05,
          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, time_limit: off

iter   objective    pri res    dua res    rho        time
   1  -8.4720e+03   2.00e+01   2.02e+06   1.00e-01   1.10e-04s
 200   3.3367e+04   1.90e-03   9.06e-02   5.17e-02   4.51e-04s
 300   3.3368e+04   2.10e-04   5.84e-04   5.17e-02   6.63e-04s

status:               solved
solution polish:      unsuccessful
number of iterations: 300
optimal objective:    33367.8626
run time:             7.67e-04s
optimal rho estimate: 1.97e-01

The optimal value is: 33367.8626085308
The optimal x is:
[[-0.  2. -0.]
 [14.  0. 11.]
 [-0. 18. -0.]
 [ 1.  0.  5.]]
The total amount of product produced is 51.0
```

The solution did change for some reason, even though in the first scenario the demand was also satisfied exactly. The objective value at the optimum changed by only `1/33338` , which is basically no change.

Problem 03

Consider the following electric power network shown in Figure 2. This network is taken from a real-world electric power system. Electricity generators are located at nodes `1`, `3`, and `5` and producing p_1, p_2, p_3 amounts of electricity, respectively. Electricity loads are located at nodes `2`, `4`, and `6` and are consuming d_1, d_2, d_3 amounts of electricity, respectively.

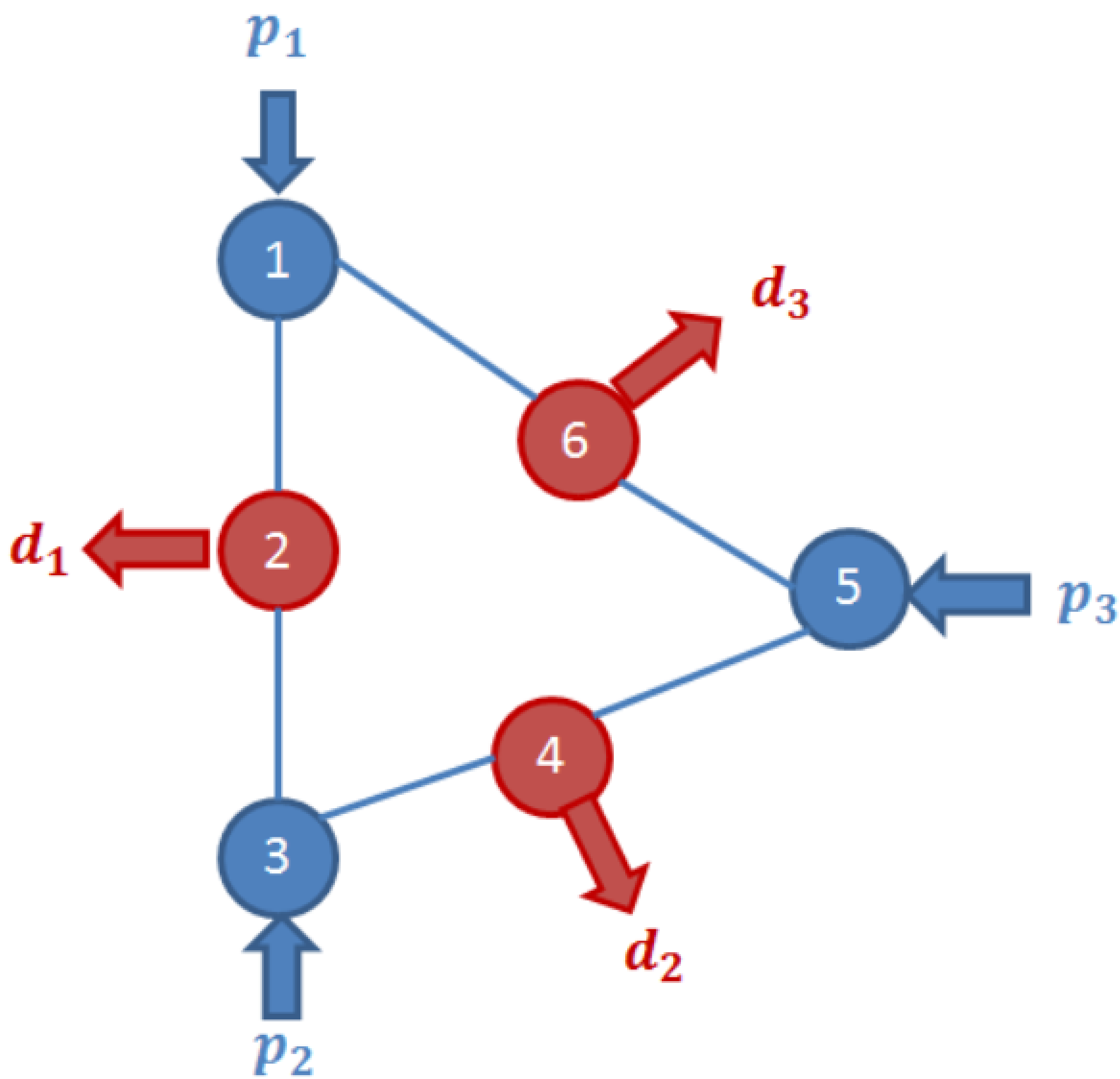


Figure 2: Electric generation problem.

The demand is fixed and given as $d_1 = 125, d_2 = 90, d_3 = 100$.

Each generator i's production must be within an upper and a lower bound as $p_i^{min} \leq pi \leq p_i^{max}$.

The bounds are given as $p_1^{min} = 20, p_1^{max} = 270, p_2^{min} = 20, p_2^{max} = 250, p_3^{min} = 10, p_3^{max} = 300$.

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

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 $c_1 = 3, c_2 = 5, c_3 = 2$.

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

$min \ \sum_{i=1}^{|G|} c_i p_i$ ----- Minimize the generation cost across all generators

$$s.t. \quad \sum_{j=O(i)} f_{ij} - \sum_{j=I(i)} f_{ij} = p_i \quad \forall i \in G$$

Flow conservation: The total flow out of each generator minus the total flow into each generator equals the generator output for all generators.

$$\sum_{j=O(i)} f_{ij} - \sum_{j=I(i)} f_{ij} = -d_i \quad \forall i \in D$$

Flow conservation: The total flow out of each load minus the total flow into each load equals the power consumed at each load.

$$\sum_{j=O(i)} f_{ij} - \sum_{j=I(i)} f_{ij} = 0 \quad \forall i \notin (G \cup D)$$

Flow conservation: The total flow out of each load minus the total flow into each load equals 0 for all nodes that are not generators or loads.

$$f_{ij} = B_{ij}(\theta_i - \theta_j) \quad \forall (i,j) \in E$$

Branch flow and nodal potential: The flow from node i to node j is proportional to the nodal potential between nodes i and j.

$$-F_{ij} \leq f_{ij} \leq F_{ij} \quad \forall (i,j) \in E$$

Flow limit constraint: The absolute value of flow between two nodes must be at or below a given flow limit.

$$p_i^{min} \leq p_i \leq p_i^{max} \quad \forall i \in G$$

Generator production constraint: The power produced by each generator must be within a lower and upper bound.

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

Note: I'm defining *into a load* and *out of a generator* to be the (+) direction.

In [173]:
```
# Declare decision variables
p = cp.Variable(3)
t = cp.Variable(6)
f = cp.Variable(6)

# Declare constants
pmin = np.array([20, 20, 10])
pmax = np.array([270, 250, 300])
fmax = np.array([100, 120, 50, 90, 60, 50])
B = np.array([11.6, 5.9, 13.7, 9.8, 5.6, 10.5])
c = np.array([3, 5, 2])
d = np.array([125, 90, 100])

#Constraints
constraints3 = [
    # flow conservation at each generator
    p[0] == f[0] + f[5], # Generator 1
    p[1] == f[1] + f[2], # Generator 3
    p[2] == f[3] + f[4], # Generator 5

    # flow conservation at each load
    -d[0] == -(f[0] + f[1]), # Load 2
    -d[1] == -(f[2] + f[3]), # Load 4
    -d[2] == -(f[4] + f[5]), # Load 6

    # branch flow
    f[0] == B[0]*(t[1]-t[0]), # From Generator 1 to Load 2
    f[1] == B[1]*(t[1]-t[2]), # From Generator 3 to Load 2
    f[2] == B[2]*(t[3]-t[2]), # From Generator 3 to Load 4
    f[3] == B[3]*(t[3]-t[4]), # From Generator 5 to Load 4
    f[4] == B[4]*(t[5]-t[4]), # From Generator 5 to Load 6
    f[5] == B[5]*(t[4]-t[0]), # From Generator 1 to Load 6

    # flow limits
    f[0] <= fmax[0],
    f[1] <= fmax[1],
    f[2] <= fmax[2],
    f[3] <= fmax[3],
    f[4] <= fmax[4],
    f[5] <= fmax[5],

    # generator limits
    p[0] >= pmin[0],
    p[0] <= pmax[0],
    p[1] >= pmin[1],
    p[1] <= pmax[1],
    p[2] >= pmin[2],
    p[2] <= pmax[2],

]

#Objective function
obj3 = cp.Minimize(cp.sum(c.T * p))

prob3 = cp.Problem(obj3, constraints3)
prob3.solve(verbose=True)
```

```python
print(f'The optimal cost is: \n{prob3.value}')
print(f'The optimal solution p is: \n{np.around(p.value,1)}')
```

```
-----------------------------------------------------------------
           OSQP v0.6.0  -  Operator Splitting QP Solver
              (c) Bartolomeo Stellato,  Goran Banjac
        University of Oxford  -  Stanford University 2019
-----------------------------------------------------------------
problem:  variables n = 15, constraints m = 24
          nnz(P) + nnz(A) = 45
settings: linear system solver = qdldl,
          eps_abs = 1.0e-05, eps_rel = 1.0e-05,
          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, time_limit: off

iter   objective    pri res    dua res    rho        time
   1  -5.8062e+00   1.25e+02   6.25e+04   1.00e-01   1.05e-04s
 200   9.9500e+02   3.31e-01   2.47e-03   1.79e-02   6.07e-04s
 400   9.9357e+02   1.32e-01   1.02e-02   2.15e-02   8.21e-04s
 600   9.9240e+02   2.26e-02   1.68e-03   3.86e-03   1.01e-03s
 800   9.9251e+02   8.01e-03   2.86e-05   3.86e-03   1.19e-03s
 925   9.9256e+02   9.41e-04   3.23e-05   3.86e-03   1.35e-03s
plsh   9.9257e+02   6.05e-15   6.05e-14   --------   1.44e-03s

status:               solved
solution polish:      successful
number of iterations: 925
optimal objective:    992.5682
run time:             1.44e-03s
optimal rho estimate: 3.94e-03

The optimal cost is: 
992.5682355720803
The optimal solution p is: 
[140.   74.2 100.8]
```

(c) Find the electricity prices for demand at nodes `2`, `4`, and `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.

In [174]:
```python
print(f'Electricity price at node 2: ${round(constraints3[3].dual_value,2)}')
print(f'Electricity price at node 4: ${round(constraints3[4].dual_value,2)}')
print(f'Electricity price at node 6: ${round(constraints3[5].dual_value,2)}')
```

```
Electricity price at node 2: $7.91
Electricity price at node 4: $3.75
Electricity price at node 6: $4.63
```