HW5 Q3

EECS 16A: Designing Information Devices and Systems I, Fall 2016

Circuit solver

In this question we will write a program that solves circuits methodically.

In [89]:
```
import numpy as np
from numpy import linalg
from __future__ import print_function
```

(i) Write the incidence matrix F for the graph.

In [90]:
```
F = np.matrix([[1, -1, 0, 0],
               [0, 1, -1, 0],
               [1, 0, -1, 0],
               [0, 1, 0, -1],
               [0, 0, 1, -1]])
print('\nF:\n',F)
```

```
F:
 [[ 1 -1  0  0]
 [ 0  1 -1  0]
 [ 1  0 -1  0]
 [ 0  1  0 -1]
 [ 0  0  1 -1]]
```

(ii) Specify the resistance matrix R.

In [91]:
```
R1, R2, R3, R4, R5 = 100000, 200, 100, 100000, 100
R = np.matrix([[R1,0,0,0,0],
               [0,R2,0,0,0],
               [0,0,R3,0,0],
               [0,0,0,R4,0],
               [0,0,0,0,R5]])

# For convenience, we will define the conductance matrix G as the inverse
G = np.linalg.inv(R)

print('\nR:\n',R)
print('\nG:\n',G)
```

```
R:
 [[100000      0      0      0      0]
 [     0    200      0      0      0]
 [     0      0    100      0      0]
 [     0      0      0 100000      0]
 [     0      0      0      0    100]]

G:
 [[  1.00000000e-05   0.00000000e+00   0.00000000e+00   0.00000000e+00
    0.00000000e+00]
 [  0.00000000e+00   5.00000000e-03   0.00000000e+00   0.00000000e+00
    0.00000000e+00]
 [  0.00000000e+00   0.00000000e+00   1.00000000e-02   0.00000000e+00
    0.00000000e+00]
 [  0.00000000e+00   0.00000000e+00   0.00000000e+00   1.00000000e-05
    0.00000000e+00]
 [  0.00000000e+00   0.00000000e+00   0.00000000e+00   0.00000000e+00
    1.00000000e-02]]
```

(iii) Write down the vector f so that KCL is satisfied as: $F^T i + f = 0$

In [92]:
```
I = 3
f = np.array([[-I],
              [0],
              [0],
              [I]])
print('\nf:\n', f)
```

```
f:
 [[-3]
 [ 0]
 [ 0]
 [ 3]]
```

(iv) Setting a potential in v to 0 corresponds to deleting a column of F. Write down our new "grounde

In [93]:
```
F_grounded = np.delete(F, np.s_[0:1] ,1)
f_1 = np.delete(f, 0, 0)
print('\nF_grounded:\n', F_grounded)
```

```
F_grounded:
 [[-1  0  0]
 [ 1 -1  0]
 [ 0 -1  0]
 [ 1  0 -1]
 [ 0  1 -1]]
```

(v) Implement your algebraic solution to compute v in terms of F_grounded, G, and f.

Hint: if you have an equation Av=b where A is a square matrix and b is a vector, use np.linalg.solve t

In [94]:
```
A = np.dot(F_grounded.T, np.dot(G,F_grounded))
v = np.linalg.solve(A,-f_1)
print('\nv:\n', v)
```

```
v:
 [[-299.7002997]
 [-299.7002997]
 [-599.4005994]]
```

(vi) Compute $\vec{i}$ with your solution of $\vec{v}$.

In [96]:
```
i = np.linalg.solve(F.T, -f)

print('\ni:\n', i)
```

```
---------------------------------------------------------------------------
LinAlgError                               Traceback (most recent call last
<ipython-input-96-8d19665f8ca0> in <module>()
----> 1 i = np.linalg.solve(F.T, -f)
      2
      3 print('\ni:\n', i)

/Users/Dawvid/anaconda3/lib/python3.4/site-packages/numpy/linalg/linalg.py
    353     a, _ = _makearray(a)
    354     _assertRankAtLeast2(a)
--> 355     _assertNdSquareness(a)
    356     b, wrap = _makearray(b)
    357     t, result_t = _commonType(a, b)

/Users/Dawvid/anaconda3/lib/python3.4/site-packages/numpy/linalg/linalg.py
(*arrays)
    210     for a in arrays:
    211         if max(a.shape[-2:]) != min(a.shape[-2:]):
--> 212             raise LinAlgError('Last 2 dimensions of the array must
    213
    214 def _assertFinite(*arrays):

LinAlgError: Last 2 dimensions of the array must be square
```

In [ ]:
```

```