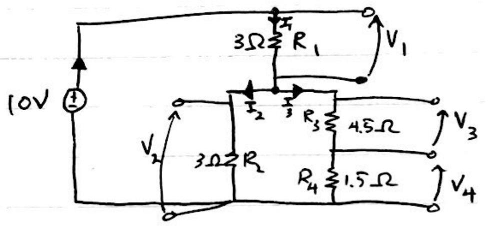
EE 16A HOMEWORK 5 DAVID J. LEE-3031796951

1) Matthew Soh - 3032109159
William Song - 3031799759
I worked alone for about 3 hours - less this time around b/c of time constraints - and discussed with group on Sunday.

2) a)



Ohm's Law:

$V_1 = 3I_1$

$V_2 = 3I_2$

$V_3 = 4.5I_3$

$V_4 = 1.5I_3$

applying KVL multiple times we get:

$$V_2 = V_3 + V_4$$
$$V_1 + V_2 = 10$$
$$V_1 + V_3 + V_4 = 10$$

applying KCL on the junction we get:

$$I_1 = I_2 + I_3$$

Combining the three sets of equations, we get:

$$\begin{aligned} 3I_2 - 4.5I_3 - 1.5I_3 &= 0 \\ 3I_1 + 3I_2 &= 10 \\ 3I_1 + 4.5I_3 + 1.5I_3 &= 10 \\ I_1 - I_2 - I_3 &= 0 \end{aligned}$$

Solving for the currents, we get:

$$\left[\begin{array}{ccc|c} 0 & 3 & -6 & 0 \\ 3 & 3 & 0 & 10 \\ 3 & 0 & 6 & 10 \\ 1 & -1 & -1 & 0 \end{array}\right] \sim \left[\begin{array}{ccc|c} 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & \frac{4}{3} \\ 0 & 0 & 1 & \frac{2}{3} \\ 0 & 0 & 0 & 0 \end{array}\right] \therefore \boxed{\begin{aligned} I_1 &= 2\text{ A} \\ I_2 &= \tfrac{4}{3}\text{ A} \\ I_3 &= \tfrac{2}{3}\text{ A} \end{aligned}}$$

Scanned by CamScanner

We can now solve for the voltages across the resistors by plugging in the I_1, I_2, I_3 into the Ohm's Law equations:

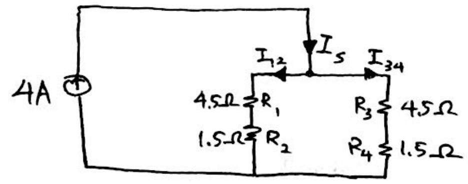
$$V_1 = 3I_1 = 3(2) = \boxed{6\ V}$$
$$V_2 = 3I_2 = 3(\tfrac{4}{3}) = \boxed{4V}$$
$$V_3 = 4.5I_3 = 4.5(\tfrac{2}{3}) = \boxed{3V}$$
$$V_4 = 1.5I_3 = 1.5(\tfrac{2}{3}) = \boxed{1\ V}$$

Thus, the voltages across and currents flowing through each resistor is given by:

R_1 : $6V$, $2\ A$	R_3 : $3V$, $\frac{2}{3}A$
R_2 : $4V$, $\frac{4}{3}\ A$	R_4 : $1V$, $\frac{2}{3}A$

b) Disregarding R_5, we can see that (R_1+R_2) parallels and is symmetrical to (R_3+R_4). Thus we can safely analyze the circuit as if R_5 wasn't there.

Also, the symmetry tells us that $I_S = 4A$ splits at the first junction evenly and $\boxed{I_{12} = I_{34} = 2A}$:



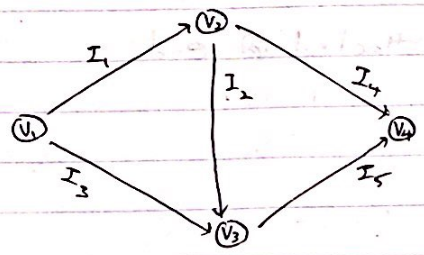
Voltages across each resistor now can be calculated easily applying Ohm's Law,

$$V_1 = I_{12}R_1 = 2(4.5) = \boxed{9\ V = V_1}$$
$$V_2 = I_{12}R_2 = 2(1.5) = \boxed{3\ V = V_2}$$
$$V_3 = I_{34}R_3 = 2(4.5) = \boxed{9V = V_3}$$
$$V_4 = I_{34}R_4 = 2(1.5) = \boxed{3V = V_4}$$

And simply, $\boxed{V_5 = 0}$

Scanned by CamScanner

3) a)



$$F = \begin{bmatrix} 1 & -1 & 0 & 0 \\ 0 & 1 & -1 & 0 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & -1 \end{bmatrix}$$

b) let $\vec{v}$ represent the voltage nodes.

$$F\vec{v} = \begin{bmatrix} 1 & -1 & 0 & 0 \\ 0 & 1 & -1 & 0 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & -1 \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \\ V_3 \\ V_4 \end{bmatrix} = \begin{bmatrix} V_1 - V_2 \\ V_2 - V_3 \\ V_1 - V_3 \\ V_2 - V_4 \\ V_3 - V_4 \end{bmatrix}$$ is a vector of ΔV, the voltage differences.

since $R = \begin{bmatrix} R_1 & & & & \\ & R_2 & & & \\ & & R_3 & & \\ & & & R_4 & \\ & & & & R_5 \end{bmatrix}$ is the diagonal matrix of branch matrices and $\vec{i} = \begin{bmatrix} I_1 \\ I_2 \\ I_3 \\ I_4 \\ I_5 \end{bmatrix}$ the current vector,

We can rewrite the Ohm's Law ($V = IR$) as:

$$\boxed{F\vec{v} = R\vec{i}}$$

c) First, taking $F\vec{v} = R\vec{i}$ and solving for $\vec{i}$, $\vec{i} = R^{-1}F\vec{v}$.

also, $f = \begin{bmatrix} -I \\ 0 \\ 0 \\ I \end{bmatrix}$

We know that R, the diagonal matrix, is invertible, and $R^{-1} = G$ where G is the conductance matrix. So we can write $\vec{i} = GF\vec{v}$.

Now, using $F^T\vec{i} + f = 0$, we can write $\vec{v}$ in terms of known quantities by plugging in $\vec{i}$:

$$\boxed{F^T G F \vec{v} = -f}$$

which is in the form $Ax = b$, so $\vec{v}$ can be solved.

d) Solving for $\vec{i}$ from the rewritten Ohm's Law, $\vec{i} = GF\vec{v}$. Then using the above $\vec{v}$,

$$\vec{i} = -GF(F^TGF)^{-1}f = -GFF^{-1}G^{-1}F^{T^{-1}}f$$

$$\boxed{\vec{i} = -F^{T^{-1}}f}$$

Scanned by CamScanner

e) See IPython Notebook attached @ end.

Scanned by CamScanner

4) a) $P = IV \rightarrow I = \frac{P}{V} = \frac{0.4\,W}{3.8\,V} \approx 0.1053$ Amps

Assume the Galaxy S3's average power draw rate is 0.1053 Amps. Then a full charge of 2200 mAh would last:

$$\frac{2200\,mAh}{105.2632\,mA} \approx \boxed{20.9 \text{ hours}}$$

b) Energy for full charge:

$$0.4\,\frac{J}{s} \times 20.9\,hr \times \frac{3600\,s}{1\,hr} = \boxed{30{,}096\,J},$$

which is $\frac{30096\,J}{3.8\,V} = \boxed{7{,}920\ C}$ Char

c) For a month of 31 days:

$$\$0.16/kWh \times \frac{1\,kW}{10^3\,W} \times 3.8\,V \times 2.2\,Ah \times 31 = \boxed{\$0.040128}$$

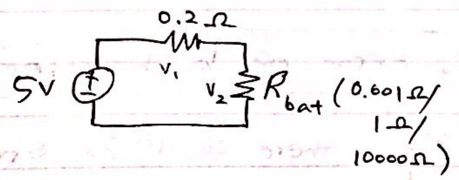
d) In 16 hours, you get $16\,h \times 40\,mA = 640\,mAh$, which is not enough.

How many cells? $2200\,mA \times \frac{1}{40\,mA} \times 3.8\,V \times \frac{1}{0.5\,V} = \boxed{418 \text{ cells}}$

~~e)~~ $\$1.50 \times \frac{1}{0.02\,W} \times \frac{1}{16\,hr \times 365 \times 60 \times 60 \times 10} = \boxed{\$0.00000570776\,/J}$

This is a very silly idea. Who wants the hassle?

e)



$5 = V_1 + V_2$; $V_1 = 0.2I$, $V_2 = R_{bat} I$

$5 = 0.2I + R_{bat} I$

$$I = \frac{5}{0.2 + R_{bat}}$$

@ $R_{bat} = 0.001\Omega$: $I = 24.87562\,A$, $P = I^2R = 0.6188\,W$,

Time to Charge $= 8.6/0.62 \approx 13.5\,hr$

@ $R_{bat} = 1\Omega$: $I = 4.166\bar{6}\,A$, $P = I^2R = 17.36\bar{1}\,W$,

Time to Charge $= 0.48\,hr$

@ $R_{bat} = 10000\Omega$: $I = 0.00049990\,A$, $P = I^2R = 0.00249990\,W$,

Time to Charge ≈ 139 days

Scanned by CamScanner

5) a) Similar to question 2) b), we can see that if the current across the galvanometer is 0, the circuit must by symmetrical, and (R_1+R_2) be parallel to (R_3+R_x), and I splits evenly to each parallel branch.

The total resistance can be written as

$$R_{eq} = \frac{1}{\frac{1}{R_1+R_2} + \frac{1}{R_3+R_x}} = \frac{1}{\frac{R_1+R_2+R_3+R_x}{(R_1+R_2)(R_3+R_x)}}$$

$$R_{eq}(R_1+R_2+R_3+R_x) = R_1R_3 + R_1R_x + R_2R_3 + R_2R_x$$

or,

Since $V = IR_{eq} \Rightarrow R_{eq} = \frac{V}{I}$,

$$\boxed{\frac{I}{V} = \frac{1}{R_1+R_2} + \frac{1}{R_3+R_x}}$$

b) The maximum for R_x is $\boxed{250\Omega}$

c) Procedure: Subtract R_x by 100Ω, and multiply by the inverse of the temperature coefficient of PT100.:

$$\text{Temperature of } R_x\ (°C) = (R_x - 100) \times \frac{1\ °C}{0.366\ \Omega} = \frac{R_x - 100}{0.366}.$$

Maximum temp would be @ the max R_x, which is

$$\frac{250-100}{0.366} \approx \boxed{409.8361\ °C}$$

d) $\left(\frac{430.5 - 410}{410}\right) \times 100\% = \boxed{5\% \text{ error on both sides}}$

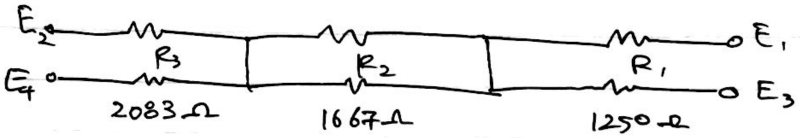
e) With the 5% max error in R_3, there is 10% error in R_1 $\left(\frac{5}{50} = 0.1\right)$. $\boxed{10\%}$

f) Not very accurate for sure!

Scanned by CamScanner

6) a) $R = 8000\,\Omega$

b)



c) Across $E_4 - E_3$:

$R_{eq} = R_1 + R_3 + \frac{R_2}{2}$

$= 4167\,\Omega$

$\therefore V_{E4-E3} = I R_{eq} = \boxed{4.167\,V}$

d) $R_{eq} = R_{plate}\left(\frac{Y_1}{H} + \frac{Y_2 - Y_1}{2H} + \frac{H - Y_2}{H}\right)$

$V = I R_{plate}\left(\frac{Y_1}{H} + \frac{Y_2 - Y_1}{2H} + \frac{H - Y_2}{H}\right)$

$\hookrightarrow Y_2 - Y_1 = 2H\left(1 - \frac{V}{I R_{plate}}\right)$

$\therefore$ It is easy to see that this gives us the information about the difference in height of the two touch positions.

Scanned by CamScanner

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 [ ]:
```
```