Name: John Paolo Acosta

Class: EE 104 Fall 2021

About this code:

This code uses ahkab library to create circuit designs and evaluate them by running OP analysis, AC and Tran analysis, PZ analysis, and Symbolic analysis to get the transfer functions. This code contains some code provided by Christopher Pham through SJSU Canvas.

What does this code do:

This code will generate an IRV circuit based on the programmers design and use a matrix with arrays to evaluate the current in each resistor. A matrix for the resistor and loops was assigned to rv in array format and then inverted using linalg from np. The inverse of the resistors was multiplied by the voltage sources array that was assigned to vs.

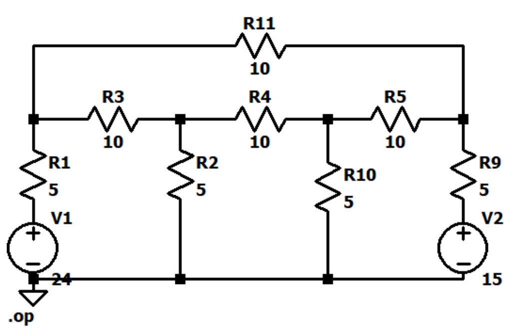
```
rv = np.array([[20, -5, 0, -10],
               [-5, 20, -5, -10],
               [0, -5, 20, -10],
               [-10, -10, -10, 40]])
rv
irv = np.linalg.inv(rv)
irv
vs = np.array([[24], [0],[-15], [0]])
vs
print(irv.dot(vs))
```

Figure 1 inverse of rv (irv) times voltage source

```
Populating the interactive namespace from
numpy and matplotlib
[[ 1.5  ]
 [ 0.45 ]
 [-0.45 ]
 [ 0.375]]
```

Figure 2 Output Current

```
        --- Operating Point ---

V(n001):        16.5            voltage
V(n005):        24              voltage
V(n003):        5.25            voltage
V(n004):        4.5             voltage
V(n002):        12.75           voltage
V(n006):        15              voltage
I(R11):         -0.375          device_current
I(R10):         0.9             device_current
I(R9):          -0.45           device_current
I(R5):          0.825           device_current
I(R4):          -0.075          device_current
I(R3):          -1.125          device_current
I(R2):          1.05            device_current
I(R1):          -1.5            device_current
I(V2):          -0.45           device_current
I(V1):          -1.5            device_current
```

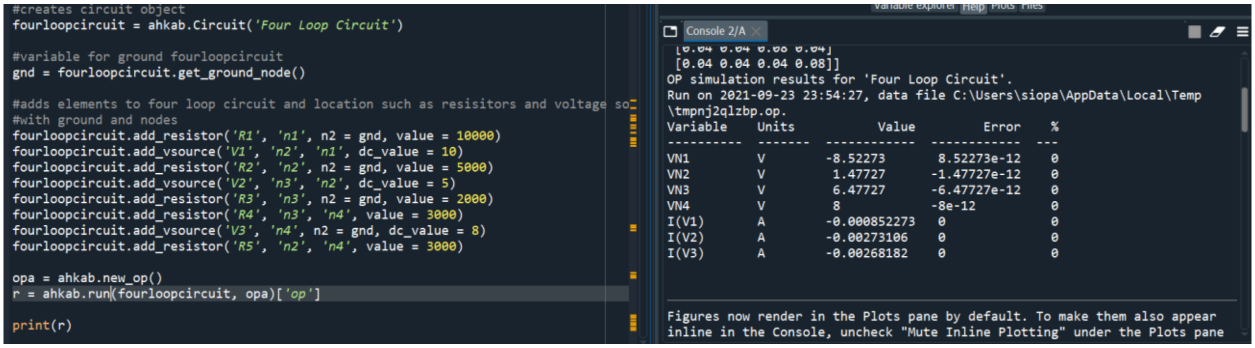


* C:\Users\siopa\Documents\SJSU\Fall 2021\EE 104\Labs\Lab3\Matrix_Ciruit.asc

```
        --- Operating Point ---

V(n001):        16.5            voltage
V(n005):        24              voltage
V(n003):        5.25            voltage
V(n004):        4.5             voltage
V(n002):        12.75           voltage
V(n006):        15              voltage
I(R11):         -0.375          device_current
I(R10):         0.9             device_current
I(R9):          -0.45           device_current
I(R5):          0.825           device_current
I(R4):          -0.075          device_current
I(R3):          -1.125          device_current
I(R2):          1.05            device_current
I(R1):          -1.5            device_current
I(V2):          -0.45           device_current
I(V1):          -1.5            device_current
```

Figure 3 LTSpice of Matrix matches the python calculation

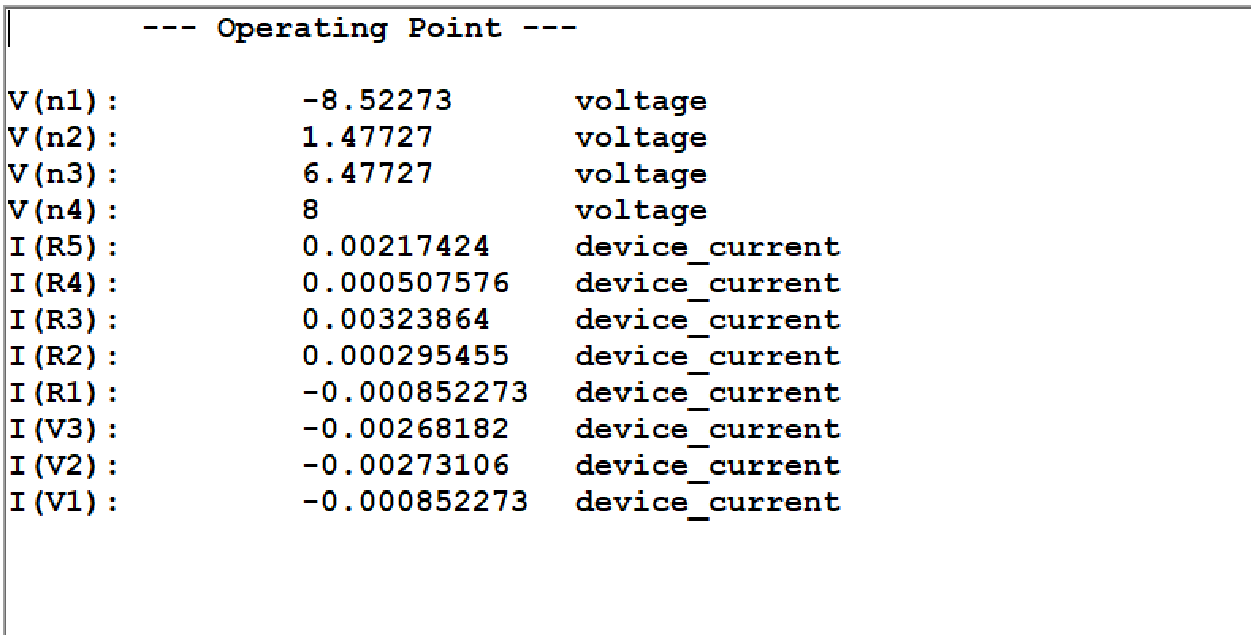
This code will generate an IRV circuit the programmer had designed and run an OP analysis on the circuit using ahkab.new_op() provided by the ahkab library that was imported. Each part of the circuit was added such as resistor and voltage source and its location of nodes were assigned.



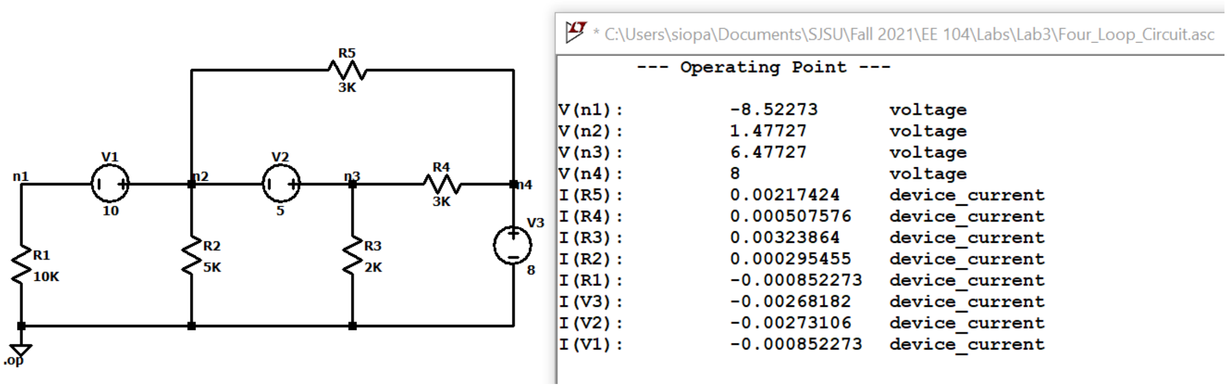
```
#creates circuit object
fourloopcircuit = ahkab.Circuit('Four Loop Circuit')

#variable for ground fourloopcircuit
gnd = fourloopcircuit.get_ground_node()

#adds elements to four loop circuit and location such as resisitors and voltage so
#with ground and nodes
fourloopcircuit.add_resistor('R1', 'n1', n2 = gnd, value = 10000)
fourloopcircuit.add_vsource('V1', 'n2', 'n1', dc_value = 10)
fourloopcircuit.add_resistor('R2', 'n2', n2 = gnd, value = 5000)
fourloopcircuit.add_vsource('V2', 'n3', 'n2', dc_value = 5)
fourloopcircuit.add_resistor('R3', 'n3', n2 = gnd, value = 2000)
fourloopcircuit.add_resistor('R4', 'n3', 'n4', value = 3000)
fourloopcircuit.add_vsource('V3', 'n4', n2 = gnd, dc_value = 8)
fourloopcircuit.add_resistor('R5', 'n2', 'n4', value = 3000)

opa = ahkab.new_op()
r = ahkab.run(fourloopcircuit, opa)['op']

print(r)
```

```
Console 2/A
 [0.04 0.04 0.04 0.08]]
OP simulation results for 'Four Loop Circuit'.
Run on 2021-09-23 23:54:27, data file C:\Users\siopa\AppData\Local\Temp
\tmpnj2qlzbp.op.
Variable    Units          Value         Error    %
----------  -------  ------------  ------------  ---
VN1         V        -8.52273       8.52273e-12   0
VN2         V         1.47727      -1.47727e-12   0
VN3         V         6.47727      -6.47727e-12   0
VN4         V         8            -8e-12         0
I(V1)       A        -0.000852273   0             0
I(V2)       A        -0.00273106    0             0
I(V3)       A        -0.00268182    0             0

Figures now render in the Plots pane by default. To make them also appear
inline in the Console, uncheck "Mute Inline Plotting" under the Plots pane
```

Figure 4 IRV circuit design/code input and output



```
       --- Operating Point ---

V(n1):          -8.52273       voltage
V(n2):          1.47727        voltage
V(n3):          6.47727        voltage
V(n4):          8              voltage
I(R5):          0.00217424     device_current
I(R4):          0.000507576    device_current
I(R3):          0.00323864     device_current
I(R2):          0.000295455    device_current
I(R1):          -0.000852273   device_current
I(V3):          -0.00268182    device_current
I(V2):          -0.00273106    device_current
I(V1):          -0.000852273   device_current
```




```
       --- Operating Point ---

V(n1):          -8.52273       voltage
V(n2):          1.47727        voltage
V(n3):          6.47727        voltage
V(n4):          8              voltage
I(R5):          0.00217424     device_current
I(R4):          0.000507576    device_current
I(R3):          0.00323864     device_current
I(R2):          0.000295455    device_current
I(R1):          -0.000852273   device_current
I(V3):          -0.00268182    device_current
I(V2):          -0.00273106    device_current
I(V1):          -0.000852273   device_current
```

Figure 5 We can confirm this IRV circuit works with the LTSpice OP Analysis

This code will generate a RLC circuit with four loops that the programmer designed and will go through an OP, AC, and Tran analysis. The AC and Tran analysis will be plotted as well using the imported library of pylab.

```python
#creates a RLC circuit object
rlc_circuit = circuit.Circuit(title="RLC")

#variable for ground RLC circuit
gnd2 = rlc_circuit.get_ground_node()

#add elements of RLC circuit
rlc_circuit.add_resistor('R1', 'n1', 'n2', value = 500)
rlc_circuit.add_inductor('L1', 'n2', 'n3', value = 13e-3)
rlc_circuit.add_capacitor('C1', 'n3', n2 = gnd2, value = 120e-9)
rlc_circuit.add_inductor('L2', 'n3', 'n4', value = 70e-3)
rlc_circuit.add_capacitor('C2', 'n4', n2 = gnd2, value = 150e-9)
rlc_circuit.add_inductor('L3', 'n4', 'n5', value = 90e-3)
rlc_circuit.add_capacitor('C3', 'n5', n2 = gnd2, value = 170e-9)
rlc_circuit.add_resistor('R2', 'n5', n2 = gnd2, value = 1500)

voltage_step = time_functions.pulse(v1 = 0, v2 = 1, td = 500e-9,
tr = 1e-12, pw  =1, tf = 1e-12, per = 2)

rlc_circuit.add_vsource('V1', 'n1', n2 = gnd2, dc_value = 5,
ac_value = 1, function = voltage_step)

op_analysis = ahkab.new_op()
ac_analysis = ahkab.new_ac(start = 1e3, stop = 1e5, points =100)
tran_analysis = ahkab.new_tran(tstart = 0, tstop = 1.2e-3, tstep = 1e-6,
x0 = None)

q = ahkab.run(rlc_circuit, an_list = [op_analysis, ac_analysis, tran_analysis])
```

Figure 6 RLC circuit design and OP, AC, and Tran Analysis

```
#plots tran simulation
fig = plt.figure()
plt.title(rlc_circuit.title + " - TRAN Simulation")
plt.plot(q['tran']['T'], q['tran']['VN1'], label = "Input Voltage")
#plt.hold(True)
plt.plot(q['tran']['T'], q['tran']['VN4'], label = "Output Voltage")
plt.legend()
#plt.hold(False)
plt.grid(True)
plt.ylim([0, 1.2])
plt.ylabel('Step Response')
plt.xlabel('Time [s]')
fig.savefig('tran_plot.png')

#plots ac simulation
fig = plt.figure()
plt.subplot(211)
plt.semilogx(q['ac']['f'], np.abs(q['ac']['Vn4']), 'o-')
plt.ylabel('abs(V(n4)) [V]')
plt.title(rlc_circuit.title + " - AC Simulation")
plt.subplot(212)
plt.grid(True)
plt.semilogx(q['ac']['f'], np.angle(q['ac']['Vn4']), 'o-')
plt.ylabel('arg(V(n4)) [rad]')
plt.xlabel('Frequency')
fig.savefig('ac_plot.png')
plt.show()
```

Figure 7 Plots the Tran and AC simulation

This code will generate a RLC circuit with one loop and solve for the poles and zeroes. The Resonance frequency will be calculated from both the analytical and PZ analysis.

```python
#creates object RLC poles and zeros
rlc_pz = ahkab.Circuit('RLC pole and zero')
gnd2 = rlc_pz.get_ground_node()
#add elements to the rlc_pz
rlc_pz = ahkab.Circuit('RLC bandpass')
rlc_pz.add_inductor('L1', 'in', 'n1', 2e-6)
rlc_pz.add_capacitor('C1', 'n1', 'out', 3.4e-12)
rlc_pz.add_resistor('R1', 'out', gnd2, 20)
rlc_pz.add_vsource('V1', 'in', gnd2, dc_value=1, ac_value=1)

#print the netlist of rlc_pz
print(rlc_pz)

#results are saved in the pz_solution in object r
pza = ahkab.new_pz('V1', ('out', gnd2), x0=None, shift=1e3)
r = ahkab.run(rlc_pz, pza)['pz']
r.keys()

#prints the poles and zeros
print('Singularities:')
for x, _ in r:
    print ("* %s = %+g %+gj Hz" % (x, np.real(r[x]), np.imag(r[x])))
```

Figure 8 Creates RLC with poles and zeroes, then calculates the poles and zeroes

```
 * RLC bandpass
L1 in n1 2e-06
C1 n1 out 3.4e-12
R1 out 0 20
V1 in 0 type=vdc value=1 vac=1
Singularities:
* p0 = -795775 -6.10279e+07j Hz
* p1 = -795775 +6.10279e+07j Hz
* z0 = -7.23753e-14 +0j Hz
Resonance frequency from analytic calculations: 6.10331e+07 Hz
Resonance frequency from PZ analysis: 6.10228e+07 Hz
```

Figure 9 Output for RLC with poles and zeroes

The transfer function can be calculated using the following code:

symba = ahkab.new_symbolic(source='V1')

rs, tfs = ahkab.run(rlc_pz, symba)['symbolic']

```
#gets transfer function
print(rs)
print (tfs)
tfs['VOUT/V1']
Hs = tfs['VOUT/V1']['gain']
s, C1, R1, L1 = rs.as_symbols('s C1 R1 L1')
HS = sympy.lambdify(w, Hs.subs({s:I*w, C1:3.4e-12, R1:20., L1:2e-6}))

np.allclose(dB20(abs(HS(rac.get_x()))), dB20(abs(Hl(rac.get_x()))), atol=1)
```

Figure 10 Gets the transfer function

INSTALL FOLLOWING PYTHON MODULES

import pylab as plt

import numpy as np

import ahkab

from matplotlib.pyplot import *

get_ipython().run_line_magic('pylab', 'inline')

figsize = (15, 10)

from ahkab import circuit, printing, time_functions

import sympy

sympy.init_printing()

Instructions

- Make sure the above modules are installed
- Create an irv circuit using ahkab if user wants to change code
- Create RLC circuit using ahkab if user wants to change code
- Run program