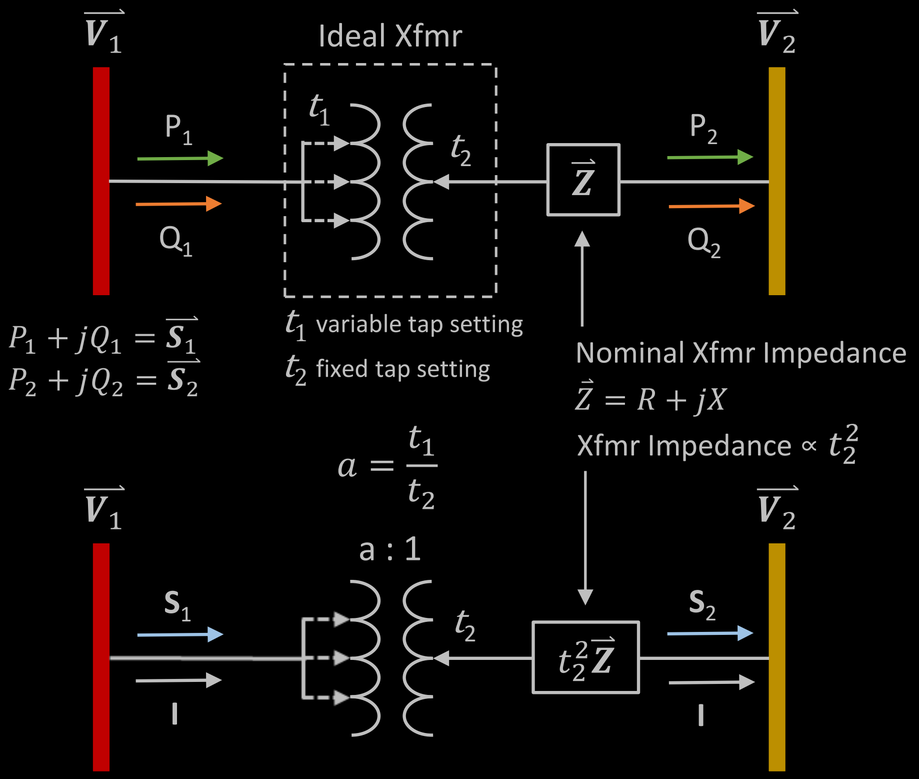
Transformer Flow Considering Tap Changes

all values per-unit

$$t_{1\,pu} = \frac{tap_{1\,setting}}{V_{1\,nominal}}$$

$$t_{2\,pu} = \frac{tap_{2\,setting}}{V_{2\,nominal}}$$



$$P_1 + jQ_1 = \vec{S}_1$$
$$P_2 + jQ_2 = \vec{S}_2$$

t_1 variable tap setting

t_2 fixed tap setting

Nominal Xfmr Impedance

$\vec{Z} = R + jX$

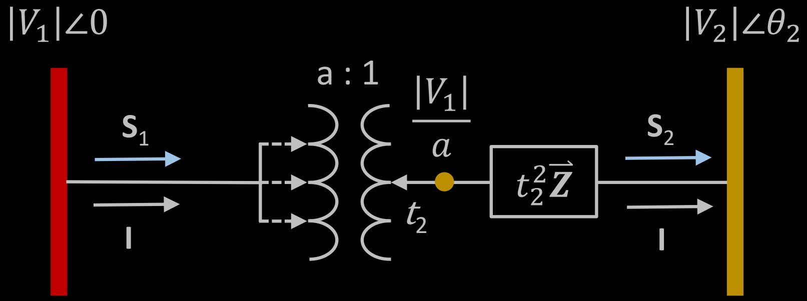
Xfmr Impedance ∝ t_2^2

$$a = \frac{t_1}{t_2}$$

Choose V_1 as reference and analyze side 2

Note:
PSSE does these calculations automatically.
The variable tap changer must be specified on side 1 (from bus).

Transformer Flow Considering Tap Changes (cont.)



$$\vec{\boldsymbol{I}} = \frac{\frac{|V_1|}{a} - |V_2|\angle\theta_2}{t_2^2\vec{\boldsymbol{Z}}}$$

$$\vec{S_2} = \vec{\boldsymbol{I}}^*|V_2|\angle\theta_2$$

$$\vec{\boldsymbol{I}} = \frac{\vec{\boldsymbol{S}}_2^*}{|V_2|\angle-\theta_2}$$

$$\frac{\frac{|V_1|}{a} - |V_2|\angle\theta_2}{t_2^2\vec{\boldsymbol{Z}}} = \frac{\vec{\boldsymbol{S}}_2^*}{|V_2|\angle-\theta_2}$$

$$|V_2|\angle(-\theta_2)\left[\frac{|V_1|}{a} - |V_2|\angle\theta_2\right] = \vec{\boldsymbol{S}}_2^* t_2^2\vec{\boldsymbol{Z}}$$

$$\frac{|V_1||V_2|\angle(-\theta_2)}{a} - |V_2|^2 = t_2^2(P_2 - jQ_2)(R + jX)$$

$$|V_1||V_2|\angle(-\theta_2) - a|V_2|^2 = at_2^2(P_2 - jQ_2)(R + jX)$$

$$|V_1||V_2|[\cos(-\theta_2) + j\sin(-\theta_2)] - a|V_2|^2 = at_2^2(P_2 - jQ_2)(R + jX)$$

$$|V_1||V_2|[\cos(\theta_2) - j\sin(\theta_2)] - a|V_2|^2 = at_2^2(P_2 - jQ_2)(R + jX)$$

$$|V_1||V_2|[\cos(\theta_2) - j\sin(\theta_2)] - a|V_2|^2 = at_2^2[(P_2R + Q_2X) + j(P_2X - Q_2R)]$$

$$|V_1||V_2|\cos(\theta_2) - a|V_2|^2 - j|V_1||V_2|\sin(\theta_2) = at_2^2(P_2R + Q_2X) + jat_2^2(P_2X - Q_2R)$$

Equating Real and Imaginary terms:

$$|V_1||V_2|\cos(\theta_2) - a|V_2|^2 = at_2^2(P_2R + Q_2X) \qquad -|V_1||V_2|\sin(\theta_2) = at_2^2(P_2X - Q_2R)$$

Transformer Flow Considering Tap Changes (cont.)

$$|V_1||V_2|\cos(\theta_2) - a|V_2|^2 = a\underbrace{t_2^2(P_2R + Q_2X)}_{B}$$

$$-|V_1||V_2|\sin(\theta_2) = at_2^2(P_2X - Q_2R)$$
$$|V_1||V_2|\sin(\theta_2) = a\underbrace{t_2^2(Q_2R - P_2X)}_{C}$$

since these are known… make a substitution

$$|V_1||V_2|\cos(\theta_2) - a|V_2|^2 = aB$$
$$\cos(\theta_2) = \frac{aB + a|V_2|^2}{|V_1||V_2|}$$
$$\cos^2(\theta_2) = \frac{a^2B^2 + 2a^2B|V_2|^2 + a^2|V_2|^4}{|V_1|^2|V_2|^2}$$

$$|V_1||V_2|\sin(\theta_2) = aC$$
$$\sin(\theta_2) = \frac{aC}{|V_1||V_2|}$$
$$\sin^2(\theta_2) = \frac{a^2C^2}{|V_1|^2|V_2|^2}$$

$$\cos^2(\theta_2) + \sin^2(\theta_2) = 1$$

$$\frac{a^2B^2 + 2a^2B|V_2|^2 + a^2|V_2|^4}{|V_1|^2|V_2|^2} + \frac{a^2C^2}{|V_1|^2|V_2|^2} = 1$$
$$a^2B^2 + 2a^2B|V_2|^2 + a^2|V_2|^4 + a^2C^2 = |V_1|^2|V_2|^2$$
$$a^2B^2 + 2a^2B|V_2|^2 + a^2|V_2|^4 + a^2C^2 - |V_1|^2|V_2|^2 = 0$$
$$a^2B^2 + 2a^2B|V_2|^2 + a^2|V_2|^4 + a^2C^2 - |V_1|^2|V_2|^2 = 0$$
$$a^2|V_2|^4 + 2a^2B|V_2|^2 - |V_1|^2|V_2|^2 + a^2B^2 + a^2C^2 = 0$$
$$\underbrace{a^2|V_2|^4 + (2a^2B - |V_1|^2)|V_2|^2 + a^2(B^2 + C^2) = 0}$$

we ended up with a mess …

- 4th order equation
- numerical solution required

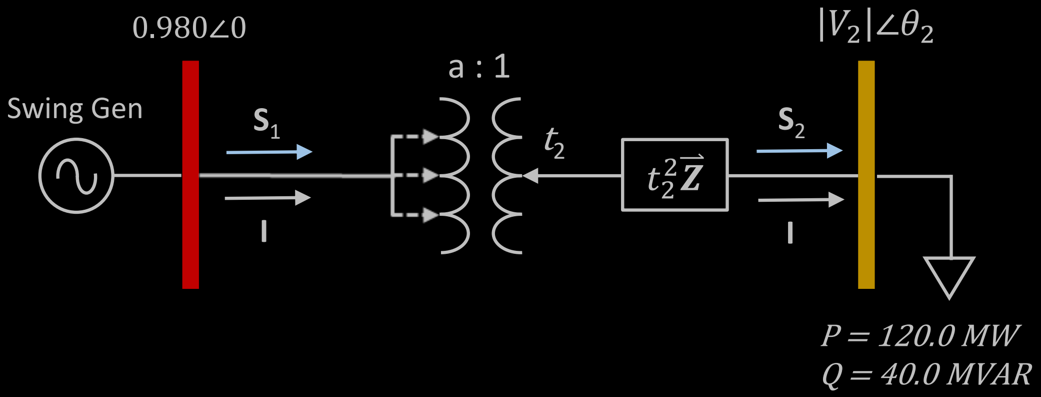
After V$_2$ is found …

$$\theta_2 = \sin^{-1}\left(\frac{aC}{|V_1||V_2|}\right)$$
$$\boldsymbol{I} = \frac{\frac{|V_1|}{a} - V_2\angle\theta_2}{t_2^2(R + jX)}$$
$$\boldsymbol{S}_1 = \frac{|V_1|}{a}\boldsymbol{I}^*$$

Transformer Flow Considering Tap Changes (cont.)

Numerical Solution

let's put some numbers in and try to find a solution …



Power flow simulations must have a swing generator …

- maintains Bus1 voltage at 0.980_{pu} (for this example) and 0°
- provides real and reactive power needed to solve the system (load + losses)

Nominal voltage of Bus1 is 138kV

Nominal voltage of Bus2 is 69kV

t_1 is 0.9875 (136.275 kV) (step 2)

t_2 is 1.025 (70.725 kV)

$\therefore \quad a = 0.9634$

Transformer impedance is 8.0% @200MVA with X/R = 30.0

$\therefore \quad Z = 0.00133 + j0.03998$ @ 100MVA

Load is 1.20 + j0.40 @100MVA

$\therefore \quad S_2 = 1.20 + j0.40$

Find: V_2, I, S_1

Why is a numerical solution required?…

- this type of problem is know as load flow or power flow
- notice that the load is not a "passive" impedance
- if the load were simply $R + jX$… simple solution
- since the load power is constant …
- the load power will determine the current
- the current will determine the Bus2 voltage
- If Bus2 voltage increases …
- current must decrease to provide the same power
- If current is changed … Bus2 voltage will change
- so you have to iterate through Bus2 voltage values to find the operating point that satisfies the Constant load power

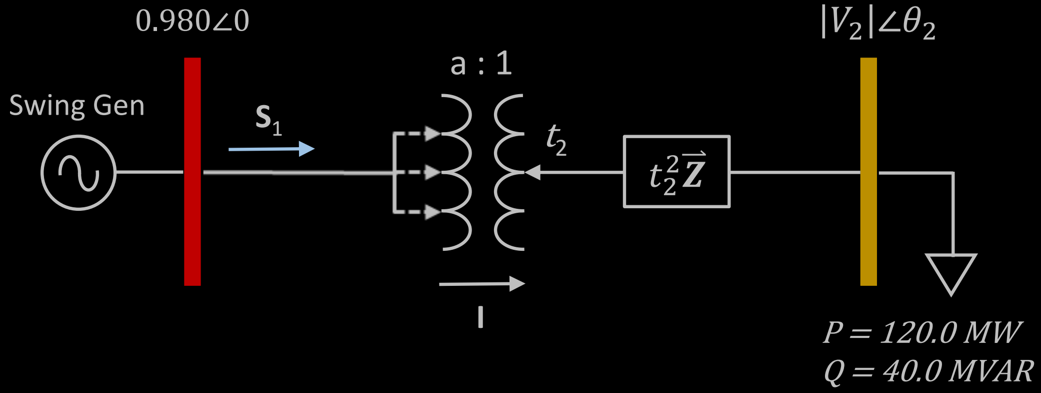
we will solve this one with a short python script that uses the Newton – Raphson method

$$a^2|V_2|^4 + (2a^2B - |V_1|^2)|V_2|^2 + a^2(B^2 + C^2) = 0$$

Transformer Flow Considering Tap Changes (cont.)

Numerical Solution

$$B = P_2 R + Q_2 X$$
$$C = Q_2 R - P_2 X$$



$$a^2|V_2|^4 + (2a^2B - |V_1|^2)|V_2|^2 + a^2(B^2 + C^2) = 0$$

Newton – Raphson requires the derivative:

$$4a^2|V_2|^3 + 2(2a^2B - |V_1|^2)|V_2|$$

each iteration adjusts the value of V_2 until a solution is found

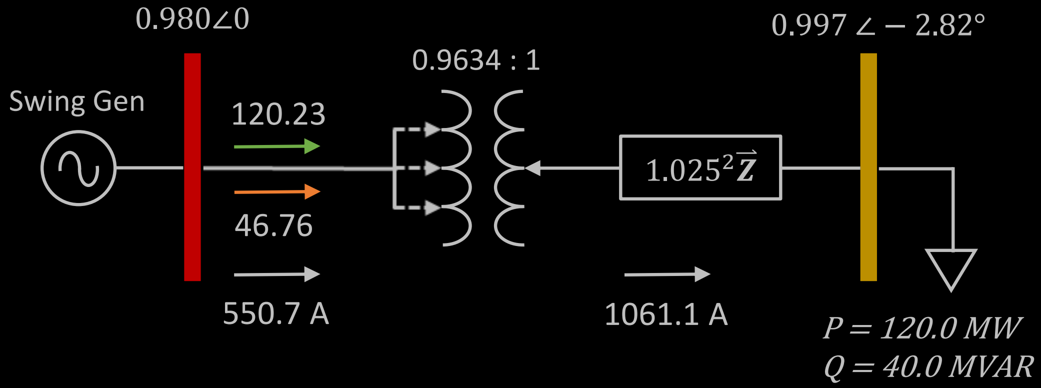
After V_2 is found …

$$\theta_2 = \sin^{-1}\left(\frac{aC}{|V_1||V_2|}\right)$$

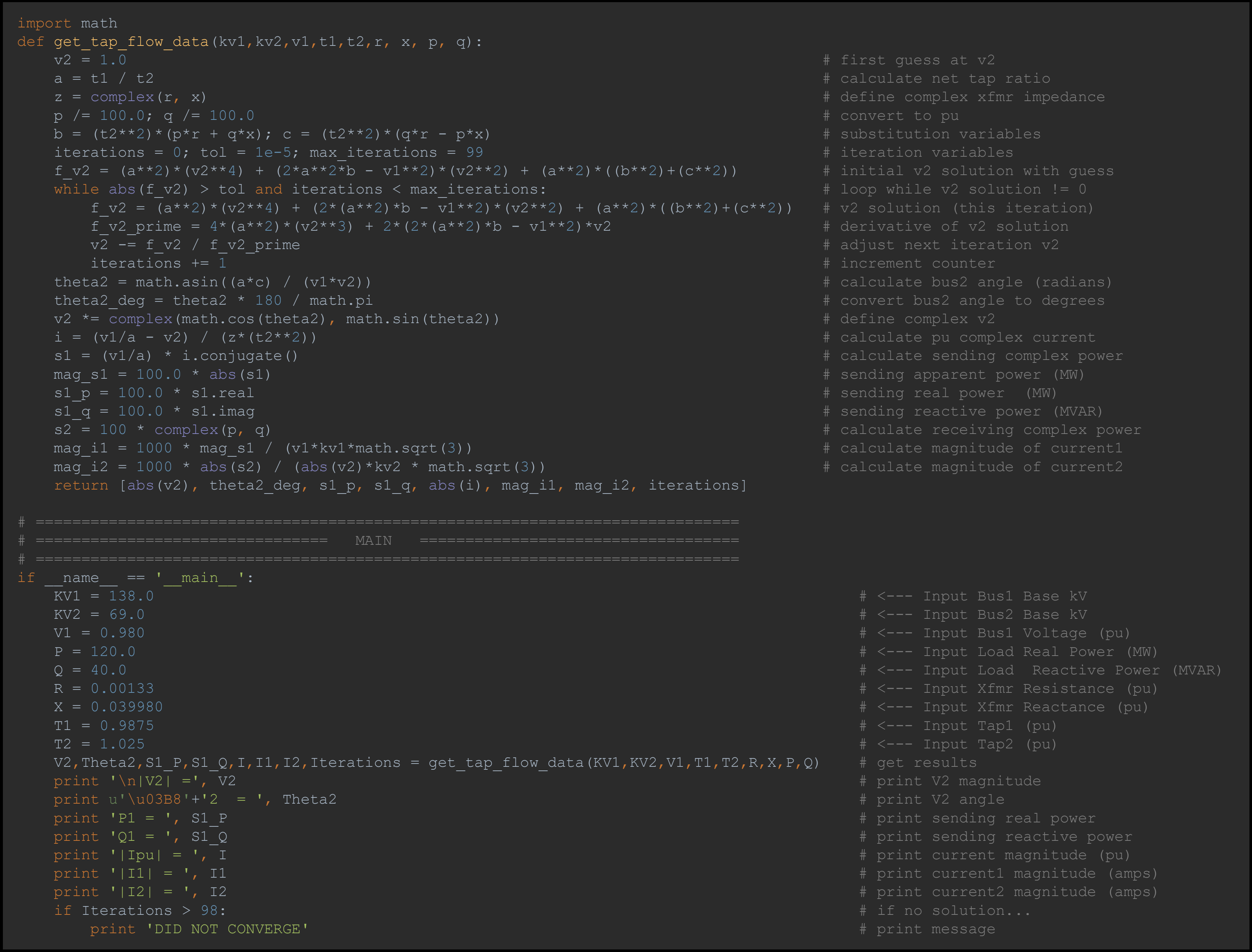
$$\boldsymbol{I} = \frac{\frac{|V_1|}{a} - V_2\angle\theta_2}{t_2^2(R + jX)}$$

$$\boldsymbol{S}_1 = \frac{|V_1|}{a}\boldsymbol{I}^*$$

Solution …



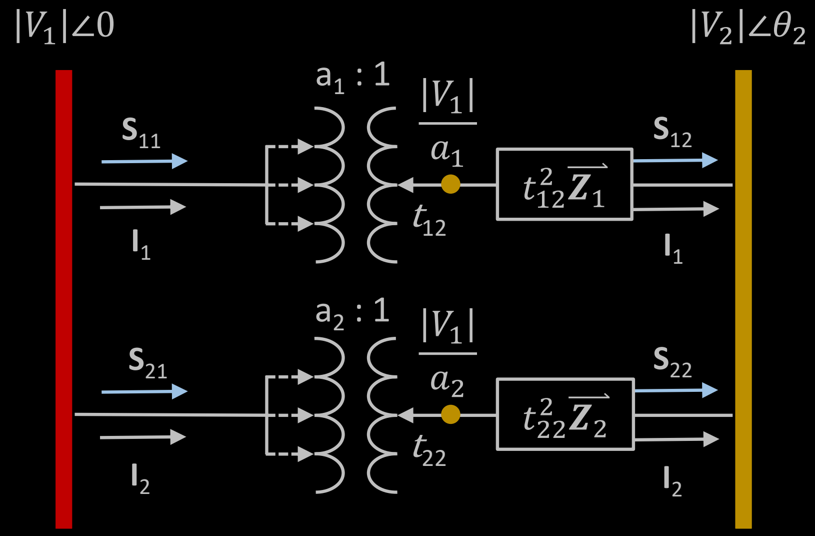
Newton – Raphson solution



```
import math
def get_tap_flow_data(kv1,kv2,v1,t1,t2,r, x, p, q):
    v2 = 1.0                                                                            # first guess at v2
    a = t1 / t2                                                                         # calculate net tap ratio
    z = complex(r, x)                                                                   # define complex xfmr impedance
    p /= 100.0; q /= 100.0                                                              # convert to pu
    b = (t2**2)*(p*r + q*x); c = (t2**2)*(q*r - p*x)                                    # substitution variables
    iterations = 0; tol = 1e-5; max_iterations = 99                                     # iteration variables
    f_v2 = (a**2)*(v2**4) + (2*a**2*b - v1**2)*(v2**2) + (a**2)*((b**2)+(c**2))         # initial v2 solution with guess
    while abs(f_v2) > tol and iterations < max_iterations:                              # loop while v2 solution != 0
        f_v2 = (a**2)*(v2**4) + (2*(a**2)*b - v1**2)*(v2**2) + (a**2)*((b**2)+(c**2))   # v2 solution (this iteration)
        f_v2_prime = 4*(a**2)*(v2**3) + 2*(2*(a**2)*b - v1**2)*v2                       # derivative of v2 solution
        v2 -= f_v2 / f_v2_prime                                                         # adjust next iteration v2
        iterations += 1                                                                 # increment counter
    theta2 = math.asin((a*c) / (v1*v2))                                                 # calculate bus2 angle (radians)
    theta2_deg = theta2 * 180 / math.pi                                                 # convert bus2 angle to degrees
    v2 *= complex(math.cos(theta2), math.sin(theta2))                                   # define complex v2
    i = (v1/a - v2) / (z*(t2**2))                                                       # calculate pu complex current
    s1 = (v1/a) * i.conjugate()                                                         # calculate sending complex power
    mag_s1 = 100.0 * abs(s1)                                                            # sending apparent power (MW)
    s1_p = 100.0 * s1.real                                                              # sending real power  (MW)
    s1_q = 100.0 * s1.imag                                                              # sending reactive power (MVAR)
    s2 = 100 * complex(p, q)                                                            # calculate receiving complex power
    mag_i1 = 1000 * mag_s1 / (v1*kv1*math.sqrt(3))                                      # calculate magnitude of current1
    mag_i2 = 1000 * abs(s2) / (abs(v2)*kv2 * math.sqrt(3))                              # calculate magnitude of current2
    return [abs(v2), theta2_deg, s1_p, s1_q, abs(i), mag_i1, mag_i2, iterations]

# ==============================================================================
# ================================   MAIN   ====================================
# ==============================================================================
if __name__ == '__main__':
    KV1 = 138.0                                                                             # <--- Input Bus1 Base kV
    KV2 = 69.0                                                                              # <--- Input Bus2 Base kV
    V1 = 0.980                                                                              # <--- Input Bus1 Voltage (pu)
    P = 120.0                                                                               # <--- Input Load Real Power (MW)
    Q = 40.0                                                                                # <--- Input Load  Reactive Power (MVAR)
    R = 0.00133                                                                             # <--- Input Xfmr Resistance (pu)
    X = 0.039980                                                                            # <--- Input Xfmr Reactance (pu)
    T1 = 0.9875                                                                             # <--- Input Tap1 (pu)
    T2 = 1.025                                                                              # <--- Input Tap2 (pu)
    V2,Theta2,S1_P,S1_Q,I,I1,I2,Iterations = get_tap_flow_data(KV1,KV2,V1,T1,T2,R,X,P,Q)    # get results
    print '\n|V2| =', V2                                                                    # print V2 magnitude
    print u'\u03B8'+'2  = ', Theta2                                                         # print V2 angle
    print 'P1 = ', S1_P                                                                     # print sending real power
    print 'Q1 = ', S1_Q                                                                     # print sending reactive power
    print '|Ipu| = ', I                                                                     # print current magnitude (pu)
    print '|I1| = ', I1                                                                     # print current1 magnitude (amps)
    print '|I2| = ', I2                                                                     # print current2 magnitude (amps)
    if Iterations > 98:                                                                     # if no solution...
        print 'DID NOT CONVERGE'                                                            # print message
```

Parallel Transformer Flow with Taps



Choose V_1 as reference and analyze side 2

Use Thevenin Equivalent…

$$\overrightarrow{\boldsymbol{V}}_{TH} = \frac{|V_1|}{a_1} - \left[\frac{|V_1|}{a_1} - \frac{|V_1|}{a_2}\right]\left[\frac{t_{12}^2\overrightarrow{\boldsymbol{Z}}_1}{t_{12}^2\overrightarrow{\boldsymbol{Z}}_1 + t_{22}^2\overrightarrow{\boldsymbol{Z}}_2}\right] = |V_{TH}|\angle\beta$$

$$\overrightarrow{\boldsymbol{Z}}_{TH} = \underbrace{t_{12}^2\overrightarrow{\boldsymbol{Z}}_1 \parallel t_{22}^2\overrightarrow{\boldsymbol{Z}}_2 = \frac{t_{12}^2\overrightarrow{\boldsymbol{Z}}_1 t_{22}^2\overrightarrow{\boldsymbol{Z}}_2}{t_{12}^2\overrightarrow{\boldsymbol{Z}}_1 + t_{22}^2\overrightarrow{\boldsymbol{Z}}_2}} = R_{TH} + jX_{TH}$$

since these are known…
solve on the side for $\mathbf{V}_{TH}$ and $\mathbf{Z}_{TH}$

Parallel Transformer Flow with Taps (cont.)

$$\vec{\boldsymbol{I}} = \frac{\overrightarrow{\boldsymbol{V}_{TH}} - |V_2|\angle\theta_2}{\overrightarrow{\boldsymbol{Z}_{TH}}}$$

$$\overrightarrow{\boldsymbol{S}_2} = \vec{\boldsymbol{I}}^*|V_2|\angle\theta_2$$

$$\vec{\boldsymbol{I}} = \frac{\vec{\boldsymbol{S}}_2^*}{|V_2|\angle-\theta_2}$$

where:

$$\overrightarrow{\boldsymbol{S}_2} = P + jQ \ \ (Load)$$

or:

$$\overrightarrow{\boldsymbol{S}_2} = \overrightarrow{\boldsymbol{S}_{12}} + \overrightarrow{\boldsymbol{S}_{22}}$$

$$\frac{|V_{TH}|\angle\beta \ - |V_2|\angle\theta_2}{R_{TH} + jX_{TH}} = \frac{P - jQ}{|V_2|\angle-\theta_2}$$

$$|V_{TH}||V_2|\angle(\beta - \theta_2) - |V_2|^2 = (P - jQ)(R_{TH} + jX_{TH})$$

$$|V_{TH}||V_2|\angle(\beta - \theta_2) - |V_2|^2 = (\underbrace{PR_{TH} + QX_{TH}}_{E}) + j(\underbrace{PX_{TH} - QR_{TH})}_{F}$$

since these are known… make a substitution

$$|V_{TH}||V_2|\angle(\beta - \theta_2) - |V_2|^2 = E + jF$$

$$|V_{TH}||V_2|\cos(\beta - \theta_2) + j\,|V_{TH}||V_2|\sin(\beta - \theta_2) - |V_2|^2 = E + jF$$

Equating Real and Imaginary terms:

$$|V_{TH}||V_2|\cos(\beta - \theta_2) - |V_2|^2 = E$$

$$\cos(\beta - \theta_2) = \frac{E + |V_2|^2}{|V_{TH}||V_2|}$$

$$\cos^2(\beta - \theta_2) = \frac{|V_2|^4 + 2E|V_2|^2 + E^2}{|V_{TH}|^2|V_2|^2}$$

$$|V_{TH}||V_2|\sin(\beta - \theta_2) = F$$

$$\sin(\beta - \theta_2) = \frac{F}{|V_{TH}||V_2|}$$

$$\sin^2(\beta - \theta_2) = \frac{F^2}{|V_{TH}|^2|V_2|^2}$$

$$\cos^2(\beta - \theta_2) + \sin^2(\beta - \theta_2) = 1$$

Parallel Transformer Flow with Taps (cont.)

$$\frac{|V_2|^4 + 2E|V_2|^2 + E^2}{|V_{TH}|^2|V_2|^2} + \frac{F^2}{|V_{TH}|^2|V_2|^2} = 1$$

$$|V_2|^4 + 2E|V_2|^2 + E^2 + F^2 = |V_{TH}|^2|V_2|^2$$

$$|V_2|^4 + 2E|V_2|^2 + E^2 + F^2 - |V_{TH}|^2|V_2|^2 = 0$$

$$|V_2|^4 + 2E|V_2|^2 - |V_{TH}|^2|V_2|^2 + E^2 + F^2 = 0$$

$$\underbrace{|V_2|^4 + (2E - |V_{TH}|^2)|V_2|^2 + E^2 + F^2}_{} = 0$$

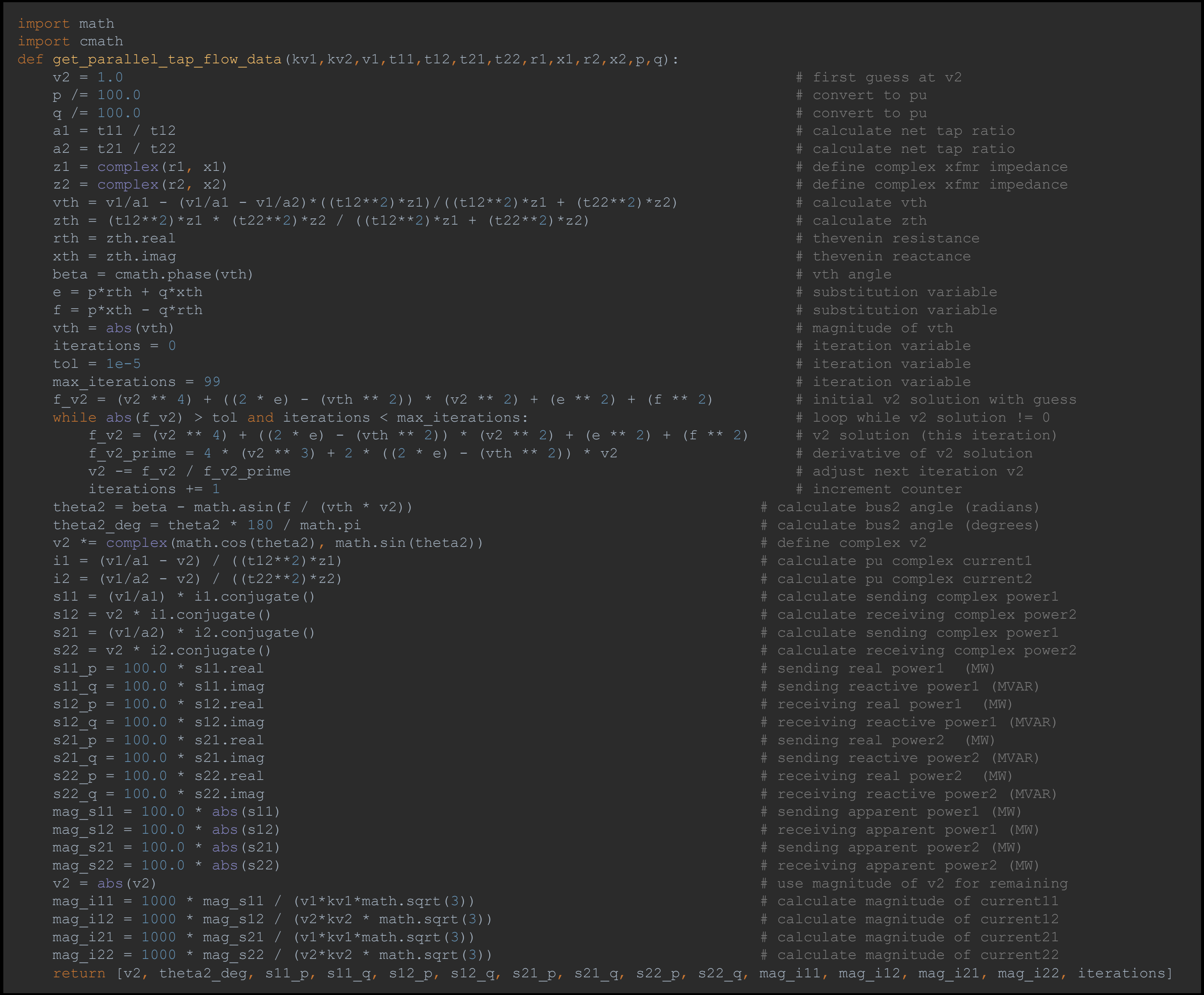
- 4th order equation
- numerical solution required
- will solve with Newton – Raphson Python Script
- Newton – Raphson requires the derivative:

$$4|V_2|^3 + 2(2E - |V_{TH}|^2)[V_2] + E^2 + F^2 = 0$$

After $|V_2|$ is found…

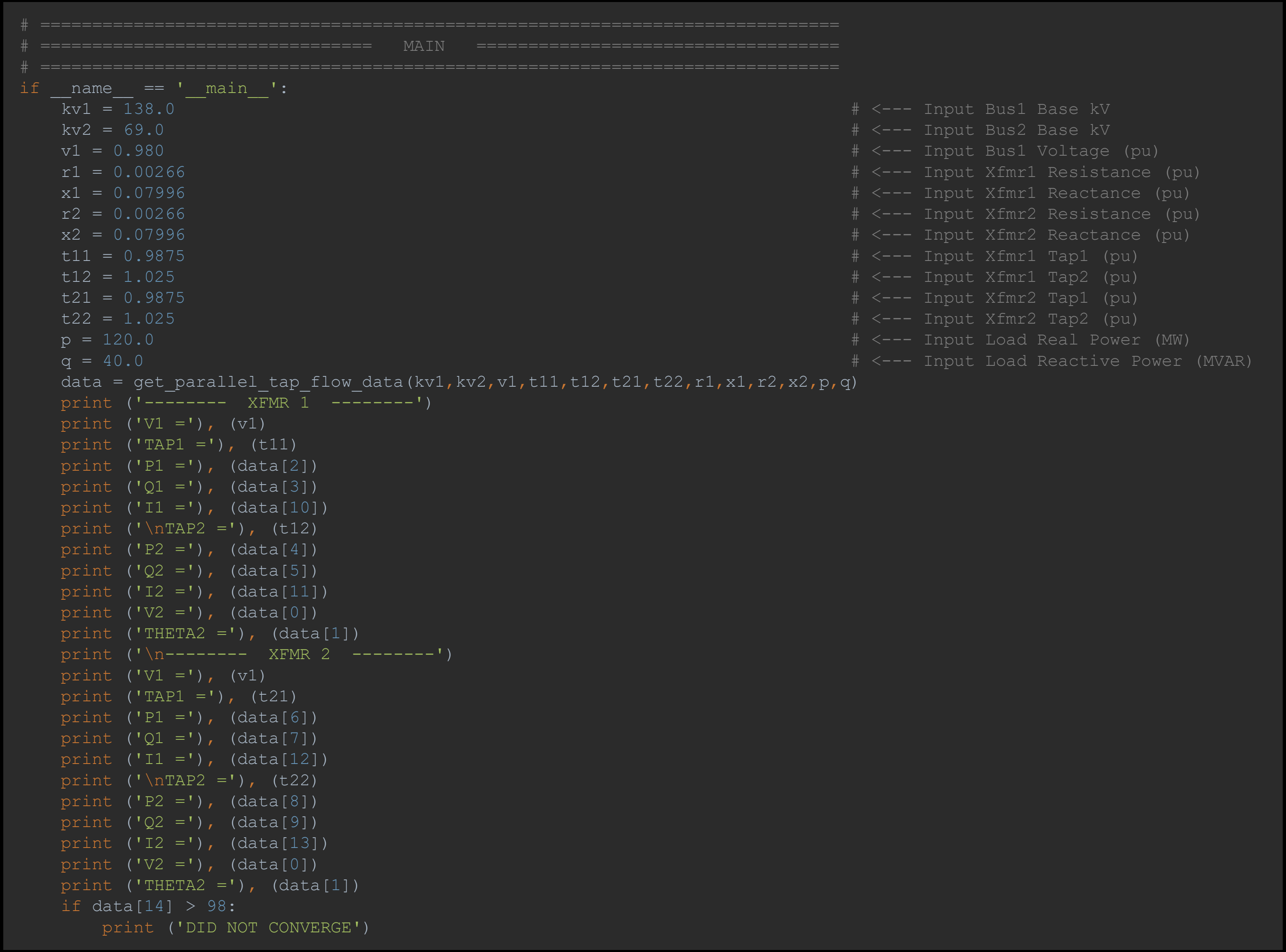
$$\boxed{\begin{aligned}
\theta_2 &= \beta - \sin^{-1}\left(\frac{F}{|V_{TH}||V_2|}\right) \\
\overrightarrow{\boldsymbol{I}_1} &= \frac{\frac{|V_1|}{a_1} - V_2\angle\theta_2}{t_{12}^2(R_1 + jX_1)} \\
\overrightarrow{\boldsymbol{I}_2} &= \frac{\frac{|V_1|}{a_2} - V_2\angle\theta_2}{t_{22}^2(R_2 + jX_2)} \\
\boldsymbol{S}_{12} &= \overrightarrow{\boldsymbol{I}_1^*}|V_2|\angle\theta_2 \\
\boldsymbol{S}_{22} &= \overrightarrow{\boldsymbol{I}_2^*}|V_2|\angle\theta_2 \\
\boldsymbol{S}_{11} &= \frac{|V_1|}{a_1}\overrightarrow{\boldsymbol{I}_1^*} \\
\boldsymbol{S}_{21} &= \frac{|V_1|}{a_2}\overrightarrow{\boldsymbol{I}_2^*}
\end{aligned}}$$

Newton – Raphson solution



```python
import math
import cmath
def get_parallel_tap_flow_data(kv1,kv2,v1,t11,t12,t21,t22,r1,x1,r2,x2,p,q):
    v2 = 1.0                                                                            # first guess at v2
    p /= 100.0                                                                          # convert to pu
    q /= 100.0                                                                          # convert to pu
    a1 = t11 / t12                                                                      # calculate net tap ratio
    a2 = t21 / t22                                                                      # calculate net tap ratio
    z1 = complex(r1, x1)                                                                # define complex xfmr impedance
    z2 = complex(r2, x2)                                                                # define complex xfmr impedance
    vth = v1/a1 - (v1/a1 - v1/a2)*((t12**2)*z1)/((t12**2)*z1 + (t22**2)*z2)             # calculate vth
    zth = (t12**2)*z1 * (t22**2)*z2 / ((t12**2)*z1 + (t22**2)*z2)                       # calculate zth
    rth = zth.real                                                                      # thevenin resistance
    xth = zth.imag                                                                      # thevenin reactance
    beta = cmath.phase(vth)                                                             # vth angle
    e = p*rth + q*xth                                                                   # substitution variable
    f = p*xth - q*rth                                                                   # substitution variable
    vth = abs(vth)                                                                      # magnitude of vth
    iterations = 0                                                                      # iteration variable
    tol = 1e-5                                                                          # iteration variable
    max_iterations = 99                                                                 # iteration variable
    f_v2 = (v2 ** 4) + ((2 * e) - (vth ** 2)) * (v2 ** 2) + (e ** 2) + (f ** 2)         # initial v2 solution with guess
    while abs(f_v2) > tol and iterations < max_iterations:                              # loop while v2 solution != 0
        f_v2 = (v2 ** 4) + ((2 * e) - (vth ** 2)) * (v2 ** 2) + (e ** 2) + (f ** 2)     # v2 solution (this iteration)
        f_v2_prime = 4 * (v2 ** 3) + 2 * ((2 * e) - (vth ** 2)) * v2                    # derivative of v2 solution
        v2 -= f_v2 / f_v2_prime                                                         # adjust next iteration v2
        iterations += 1                                                                 # increment counter
    theta2 = beta - math.asin(f / (vth * v2))                                       # calculate bus2 angle (radians)
    theta2_deg = theta2 * 180 / math.pi                                             # calculate bus2 angle (degrees)
    v2 *= complex(math.cos(theta2), math.sin(theta2))                               # define complex v2
    i1 = (v1/a1 - v2) / ((t12**2)*z1)                                               # calculate pu complex current1
    i2 = (v1/a2 - v2) / ((t22**2)*z2)                                               # calculate pu complex current2
    s11 = (v1/a1) * i1.conjugate()                                                  # calculate sending complex power1
    s12 = v2 * i1.conjugate()                                                       # calculate receiving complex power2
    s21 = (v1/a2) * i2.conjugate()                                                  # calculate sending complex power1
    s22 = v2 * i2.conjugate()                                                       # calculate receiving complex power2
    s11_p = 100.0 * s11.real                                                        # sending real power1  (MW)
    s11_q = 100.0 * s11.imag                                                        # sending reactive power1 (MVAR)
    s12_p = 100.0 * s12.real                                                        # receiving real power1  (MW)
    s12_q = 100.0 * s12.imag                                                        # receiving reactive power1 (MVAR)
    s21_p = 100.0 * s21.real                                                        # sending real power2  (MW)
    s21_q = 100.0 * s21.imag                                                        # sending reactive power2 (MVAR)
    s22_p = 100.0 * s22.real                                                        # receiving real power2  (MW)
    s22_q = 100.0 * s22.imag                                                        # receiving reactive power2 (MVAR)
    mag_s11 = 100.0 * abs(s11)                                                      # sending apparent power1 (MW)
    mag_s12 = 100.0 * abs(s12)                                                      # receiving apparent power1 (MW)
    mag_s21 = 100.0 * abs(s21)                                                      # sending apparent power2 (MW)
    mag_s22 = 100.0 * abs(s22)                                                      # receiving apparent power2 (MW)
    v2 = abs(v2)                                                                    # use magnitude of v2 for remaining
    mag_i11 = 1000 * mag_s11 / (v1*kv1*math.sqrt(3))                                # calculate magnitude of current11
    mag_i12 = 1000 * mag_s12 / (v2*kv2 * math.sqrt(3))                              # calculate magnitude of current12
    mag_i21 = 1000 * mag_s21 / (v1*kv1*math.sqrt(3))                                # calculate magnitude of current21
    mag_i22 = 1000 * mag_s22 / (v2*kv2 * math.sqrt(3))                              # calculate magnitude of current22
    return [v2, theta2_deg, s11_p, s11_q, s12_p, s12_q, s21_p, s21_q, s22_p, s22_q, mag_i11, mag_i12, mag_i21, mag_i22, iterations]
```

Newton – Raphson solution (cont.)



```
# =============================================================================
# ================================   MAIN   ===================================
# =============================================================================
if __name__ == '__main__':
    kv1 = 138.0                                                                 # <--- Input Bus1 Base kV
    kv2 = 69.0                                                                  # <--- Input Bus2 Base kV
    v1 = 0.980                                                                  # <--- Input Bus1 Voltage (pu)
    r1 = 0.00266                                                                # <--- Input Xfmr1 Resistance (pu)
    x1 = 0.07996                                                                # <--- Input Xfmr1 Reactance (pu)
    r2 = 0.00266                                                                # <--- Input Xfmr2 Resistance (pu)
    x2 = 0.07996                                                                # <--- Input Xfmr2 Reactance (pu)
    t11 = 0.9875                                                                # <--- Input Xfmr1 Tap1 (pu)
    t12 = 1.025                                                                 # <--- Input Xfmr1 Tap2 (pu)
    t21 = 0.9875                                                                # <--- Input Xfmr2 Tap1 (pu)
    t22 = 1.025                                                                 # <--- Input Xfmr2 Tap2 (pu)
    p = 120.0                                                                   # <--- Input Load Real Power (MW)
    q = 40.0                                                                    # <--- Input Load Reactive Power (MVAR)
    data = get_parallel_tap_flow_data(kv1,kv2,v1,t11,t12,t21,t22,r1,x1,r2,x2,p,q)
    print ('--------  XFMR 1  --------')
    print ('V1 ='), (v1)
    print ('TAP1 ='), (t11)
    print ('P1 ='), (data[2])
    print ('Q1 ='), (data[3])
    print ('I1 ='), (data[10])
    print ('\nTAP2 ='), (t12)
    print ('P2 ='), (data[4])
    print ('Q2 ='), (data[5])
    print ('I2 ='), (data[11])
    print ('V2 ='), (data[0])
    print ('THETA2 ='), (data[1])
    print ('\n--------  XFMR 2  --------')
    print ('V1 ='), (v1)
    print ('TAP1 ='), (t21)
    print ('P1 ='), (data[6])
    print ('Q1 ='), (data[7])
    print ('I1 ='), (data[12])
    print ('\nTAP2 ='), (t22)
    print ('P2 ='), (data[8])
    print ('Q2 ='), (data[9])
    print ('I2 ='), (data[13])
    print ('V2 ='), (data[0])
    print ('THETA2 ='), (data[1])
    if data[14] > 98:
        print ('DID NOT CONVERGE')
```

3ΦEE

Dedicated to Power Engineering

Questions or Comments …

contact us