EE347 HW2 Nick Porter

Text Questions 3-1 through 3-5

3-1:
Is the turns ratio of a transformer the same as the ratio of voltages across the transformer? Why or why not?

Yes. The induced electromotive force in each side is proportional to the magnetic flux shared between them. That proportion is determined by the number of turns each side has.

3-2:
Why does the magnetization current impose an upper limit on the voltage applied to a transformer core?

By Faraday's Law, voltage in a transformer is defined by the flux across the magnetic core. As the core approaches magnetic saturation, the magnetization current required to increase the flux becomes very large.

3-3:
What components compose the excitation current of a transformer? How are they modeled in the transformer's equivalent circuit?

The excitation current is the sum of the magnetization current and the core loss current through hysteresis and eddy current losses. These are the currents required to generate flux through the core, and occur even when there is no load. The magnetization current can be modeled as a reactance across the input since it lags the input voltage by 90 degrees. The core loss current can be modeled as a resistance across the input.

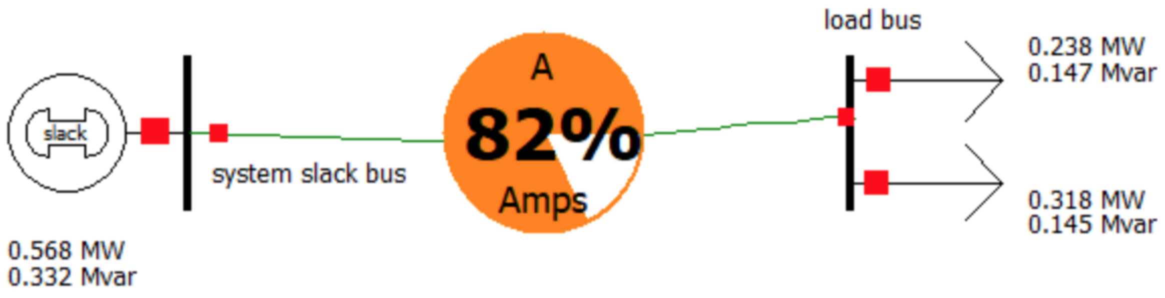
3-4:
What is the leakage flux in a transformer? Why is it modeled in a transformer equivalent circuit as an inductor?

Leakage flux is the part of the flux through a coil that does not also go through the coil on the other side of the core. The leakage flux of a coil is proportional to the current through that coil, so it can be modeled as an inductor with inductance equal to the proportion.

3-5:
List and describe the types of losses that occur in a transformer.

Flux leakage via lines of flux which only flow through one of the coils allow some flux to permeate through the air. Excitation current losses come from hysteresis, magnetization, and eddy currents. There is loss from the resistance of the copper conductors. There is also loss from the phase shift between the primary and secondary coils.

Problem 1:



$P_{gen} = 568kW$

$P_L = 238kW + 318kW = 556kW$

$\eta = \frac{P_L}{P_{gen}} \sim 0.98$

Problem 2:

$PF_{total} = 0.81$ lagging $\qquad THD = 0.68$

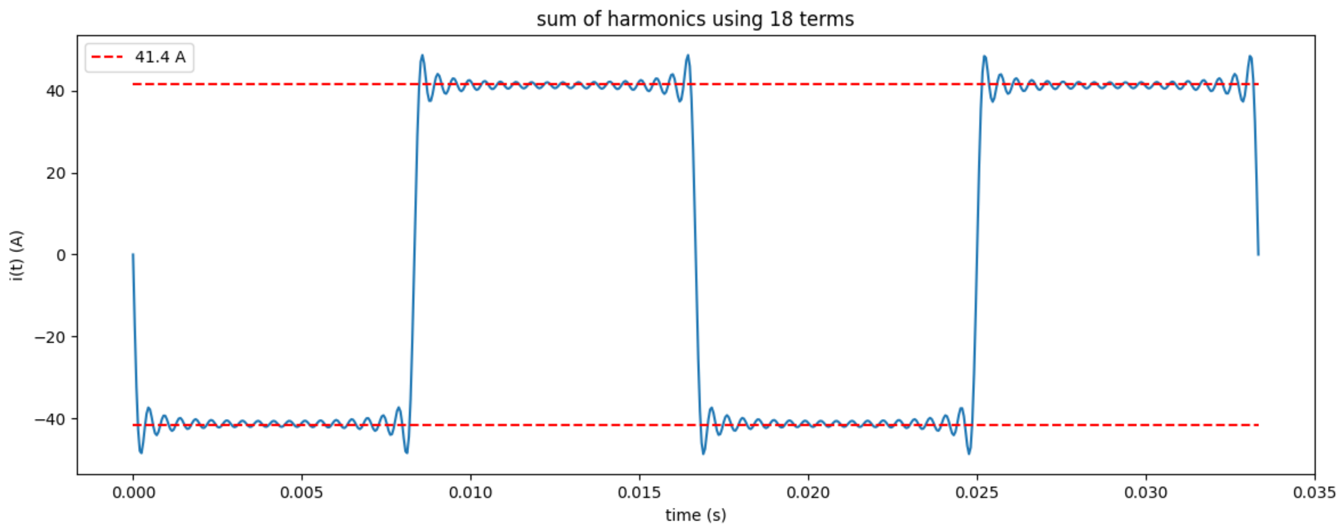
$PF_{total} = PF_{d} \cdot PF_{THD}$

$PF_{THD} = \sqrt{\frac{1}{1+THD^2}} = 0.827$

$PF_{d} = \frac{PF_{total}}{PF_{THD}} = 0.979 \approx 0.98$

Problem 3:
See Appendix A for the full python code solution to this problem.

The waveform this equation represents is a square wave:



The absolute peak value is 48 A, but the peak oscillates around 41 A
The frequency is 60Hz

THD of 17 terms = 46.8 %
THD of 18 terms = 46.9 %
Percent convergence for 18 terms is 0.19%

Problem 4:

$$THD_I = \frac{\sqrt{\sum_{n=2}^{8} I_{n,RMS}^{2}}}{I_{1,RMS}} \cdot 100 = \frac{37.7 A_{RMS}}{62 A_{RMS}} \cdot 100 = 60.8\%$$

Hot lines current (current in each line): $I_{RMS} = \sqrt{\sum_{n=1}^{8} I_{n,RMS}^{2}} = 73\ A$

RMS current in neutral line:
Since the two lines are 180° out of phase, only the even harmonics will add in the neutral line.

$$I_{N,RMS} = 2 \cdot \sum_{n=1}^{4} I_{2n,RMS} = 85\ A_{RMS}$$

Using the ambient temperature correction factor formula:

$I' = I\sqrt{\frac{T_c - T_a'}{T_c - T_a}}$ Where T_c=75°C, T_a'=45°C, T_a=30°C

The correction factor comes out to be 0.816
Using table 310.15(B)(16), in the Aluminum THW conductor column for 75°C components, The corrected amperage 90A yields a 73.5 amp rating and the corrected 120A yields a 98 A rating which means the hot lines need an AWG 2 size and the neutral line needs an AWG 1/0 size.

FE Problem 1:

$I_A = \frac{V_A}{Z_A} = \frac{277V}{35\Omega}$ $\quad$ $I_B = \frac{V_B}{Z_B} = \frac{277V\angle 120°}{30+j15\Omega}$ $\quad$ $I_C = \frac{V_C}{Z_C} = \frac{277V\angle -120°}{21-j19\Omega}$

$I_N = I_A + I_B + I_C = 9.6A\angle - 8°$

(D) $9.6\angle - 8°$

Appendix A:
Code for the solution of Problem 3

```
# EE347 HW2
import numpy as np
import matplotlib.pyplot as plt
import mpmath as mp
from scipy import fft
pi = np.pi

# problem 3
# define parameters
A = -165.0 / pi      # amplitude of fundamental (A)
f0 = 60              # Hz
T = 2/f0             # seconds simulated
fs = 20000           # Hz sampling freq
ts = 1/fs            # delta t (s)
N = int(T/ts)        # number of samples
t = np.linspace(0,T,N)
num = 18             # number of terms to simulate

# for num - 1 terms:
num_2 = num - 1
n = np.arange(0,num_2)
# setup harmonics array of arrays
harmonic = np.zeros([num_2, N])
# calculate harmonic sinusoids
ih = 0           # sum of harmonic amplitudes
for index in n:
   h = 2*(index+1) - 1      # python index starts from 0
   c = 1/h                  # harmonic coefficient
   harmonic[index] = c * np.sin(2*pi*h*f0*t)
   if index > 0:
       ih_n = mp.power(A/(mp.sqrt(2)*h),2)
       ih = ih + ih_n

# determine THD and convergence
i1_rms = abs(A) / mp.sqrt(2)
ih_rms = mp.sqrt(ih)
THD_2 = ih_rms / i1_rms * 100
print('THD of n-1 terms = %.1f %%' %THD_2)

# for num terms:
num_n = num
n = np.arange(0,num_n)
# setup harmonics array of arrays
harmonic = np.zeros([num_n, N])
# calculate harmonic sinusoids
ih = 0           # sum of harmonic amplitudes
```

```
for index in n:
   h = 2*(index+1) - 1      # python index starts from 0
   c = 1/h                  # harmonic coefficient
   harmonic[index] = c * np.sin(2*pi*h*f0*t)
   if index > 0:
       ih_n = mp.power(A/(mp.sqrt(2)*h),2)
       ih = ih + ih_n

# determine THD and convergence
i1_rms = abs(A) / mp.sqrt(2)
ih_rms = mp.sqrt(ih)
THD_n = ih_rms / i1_rms * 100
print('THD of n terms = %.1f %%' %THD_n)

# determine % convergence between values for THD
conv = abs((THD_n - THD_2)/THD_n) * 100
print("Percent convergence for %d terms is %.2f%%" %(num_n,conv))

# plot harmonics separately
num_plots = 10
n = np.arange(0,num_plots)
fig, axs = plt.subplots(num_plots,1,figsize=(10, 9), sharex=True)
for row,ax,index in zip(harmonic,axs,n):
   h = 2*(index+1) - 1
   ax.plot(t,row)
   ax.set_title("harmonic %d" %h)

fig.tight_layout(pad=.1)
fig.add_subplot(111, frame_on=False)
plt.tick_params(labelcolor="none", bottom=False, left=False)
plt.xlabel("time (s)")
plt.ylabel("mag")

# plot sum of harmonics
n = np.arange(0,num)
wave = np.zeros(N)
for index in n:
   wave = wave + harmonic[index]

# apply amplitude
wave = A * wave

# find max and peak center amplitude
vsum = 0
count = 0
for value in wave:
   val = abs(value)
   if val > 36:
       vsum = vsum + val
```

```
        count = count + 1

pk_avg = vsum / count
pk_max = max(wave)
pk_avg_str = "%.1f A" %pk_avg
pk_max_str = "%.1f A" %pk_max

print("average max = %.1f" %pk_avg)
print("peak max = %.1f" %pk_max)

plt.figure(figsize=(14, 5))
plt.plot(t,wave)
#plt.plot(t,A*np.sin(2*pi*f0*t))
plt.hlines([pk_avg,-pk_avg],0,T,colors='r',linestyles='dashed',label=pk_avg_st
r)
plt.legend()
plt.title('sum of harmonics using %d terms' %num)
plt.xlabel('time (s)')
plt.ylabel('i(t) (A)')
plt.show()

wave_fft = 20*np.log10(abs(fft.fft(wave)))
f = np.linspace(0,fs,N)
print("max freq = %.1f Hz" %f[np.argmax(wave_fft)])
```