

```
1
2 # *** EEE-3003. ELECTROMECHANICAL ENERGY CONVERSION GROUP PROJECT ***
3 # AHMAD ZAMEER NAZARI (220702706) & İSMET MERT ŞEN (210702011)
4
5
6 # *** REFERENCES ***
7 # Electric Machinery Fundamentals, Stephen J. Chapman, ed.5
8 # https://docs.pysimplegui.com/en/latest/
9 # https://matplotlib.org/stable/users/index
10 # https://stackoverflow.com/
11
12
13
14 # *** IMPORTING LIBRARIES ***
15
16 import PySimpleGUI as sg
17 import numpy as np
18 from math import floor, log10
19 import matplotlib.pyplot as plt
20 from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
21
22
23
24 # ***SETTING UP THE UI***
25
26 sg.theme('Default')
27 font = ('Aptos', 13)
28
29 app_title = 'SINGLE PHASE TRANSFORMER CALCULATOR'
30 course = 'EEE-3003: Electromechanical Energy Conversion'
31 author1 = 'Ahmad Zameer NAZARI'
32 author2 = 'İsmet Mert ŞEN'
33 credits = 'PySimpleGUI, matplotlib, numpy draw.io, latex2image by joeraut, icon from  
thenounproject \n  
cover image is TR21 Single Phase Transformer from DF Electric'
34
35
36
37 # INPUT TAB
38
39 tab_input_instr = 'Provide all given values to obtain transformer characteristics'
40 tab_input_note = '*Note: only step-down voltages permitted'
41
42 col_input_data = sg.Column([
43     [sg.Push(), sg.Text('Power Rating: '), sg.Input(key = '-INP_S-', size=7),  
sg.Text('VA', s=5)],
44     [sg.Push(), sg.Text('Primary Voltage: '), sg.Input(key = '-INP_V_P-', size=7),  
sg.Text('V', s=5)],
45     [sg.Push(), sg.Text('Secondary Voltage: '), sg.Input(key = '-INP_V_S-', size=7),  
sg.Text('V', s=5)]
46 ])
```

```
47
48 col_input_oc = sg.Column([
49     [sg.Push(), sg.Image('images/V_OC.png'), sg.Input(key = '-INP_V_OC-', size=5),
50      sg.Text('V', s=5)],
51     [sg.Push(), sg.Image('images/I_OC.png'), sg.Input(key = '-INP_I_OC-', size=5),
52      sg.Text('A', s=5)],
53     [sg.Push(), sg.Image('images/P_OC.png'), sg.Input(key = '-INP_P_OC-', size=5),
54      sg.Text('W', s=5)]
55 ])
56
57 col_input_sc = sg.Column([
58     [sg.Push(), sg.Image('images/V_SC.png'), sg.Input(key = '-INP_V_SC-', size=5),
59      sg.Text('V', s=5)],
60     [sg.Push(), sg.Image('images/I_SC.png'), sg.Input(key = '-INP_I_SC-', size=5),
61      sg.Text('A', s=5)],
62     [sg.Push(), sg.Image('images/P_SC.png'), sg.Input(key = '-INP_P_SC-', size=5),
63      sg.Text('W', s=5)]
64 ])
65
66 col_ratio = sg.Column([
67     [
68         sg.Text('Transformer Ratio,      '),
69         sg.Image('images/N.png'),
70         sg.Input(key = '-OUT_RATIO-', disabled=True, s=5),
71         sg.Text('', s=5)
72     ]
73 ])
74
75 col_input = sg.Column([
76     [sg.Sizer(5,5)],
77     [col_input_data],
78     [sg.Sizer(5,5)],
79     [sg.Text('Open Circuit Test', expand_x=True, justification='center'),
80      col_input_oc],
81     [sg.Sizer(5,5)],
82     [sg.Text('Short Circuit Test', expand_x=True, justification='center'),
83      col_input_sc],
84     [sg.Sizer(5,5)],
85     [col_ratio],
86     [sg.Sizer(5,5)]
87 ], element_justification='right')
88
89 frame_input = sg.Column([
90     [
91         sg.Frame('',
92             layout = [
93                 [sg.Sizer(15,15), col_input, sg.Sizer(10,10)]
94             ]
95         )
96     ]
97 ])
98 ])
```

```
89
90
91 # IMPEDANCE AND EQUIV CIRCUIT TAB
92
93 col_imp_refer_p_ser = sg.Column([
94     [
95         sg.Image('images/Z_eq.png', s=(50,20)),
96         sg.Multiline(key = '-OUT_Z_EQ_P-', s=(15,2), disabled=True,
background_color='#F0F0F0', no_scrollbar=True)
97     ],
98     [sg.Image('images/R_eq.png', s=(50,20)), sg.Input(key = '-OUT_R_EQ_P-',
disabled=True, s=15)],
99     [sg.Image('images/X_l_eq.png', s=(50,20)), sg.Input(key = '-OUT_X_L_EQ_P-',
disabled=True, s=15)]
100 ])
101
102 col_imp_refer_p_phi = sg.Column([
103     [
104         sg.Image('images/Y_phi.png', s=(50,20)),
105         sg.Multiline(key = '-OUT_Y_PHI_P-', s=(15,2), disabled=True,
background_color='#F0F0F0', no_scrollbar=True)
106     ],
107     [sg.Image('images/G_phi.png', s=(50,20)), sg.Input(key = '-OUT_G_PHI_P-',
disabled=True, s=15)],
108     [sg.Image('images/B_phi.png', s=(50,20)), sg.Input(key = '-OUT_B_PHI_P-',
disabled=True, s=15)],
109     [
110         sg.Image('images/Z_phi.png', s=(50,20)),
111         sg.Multiline(key = '-OUT_Z_PHI_P-', s=(15,2), disabled=True,
background_color='#F0F0F0', no_scrollbar=True)
112     ],
113     [sg.Image('images/R_c.png', s=(50,20)), sg.Input(key = '-OUT_R_PHI_P-',
disabled=True, s=15)],
114     [sg.Image('images/X_m.png', s=(50,20)), sg.Input(key = '-OUT_X_PHI_P-',
disabled=True, s=15)]
115 ])
116
117 col_imp_refer_s_ser = sg.Column([
118     [
119         sg.Image('images/Z_eq.png', s=(50,20)),
120         sg.Multiline(key = '-OUT_Z_EQ_S-', s=(15,2), disabled=True,
background_color='#F0F0F0', no_scrollbar=True)
121     ],
122     [sg.Image('images/R_eq.png', s=(50,20)), sg.Input(key = '-OUT_R_EQ_S-',
disabled=True, s=15)],
123     [sg.Image('images/X_l_eq.png', s=(50,20)), sg.Input(key = '-OUT_X_L_EQ_S-',
disabled=True, s=15)]
124 ])
125
126 col_imp_refer_s_phi = sg.Column([
```

```
127     [
128         sg.Image('images/Y_phi.png', s=(50,20)),
129         sg.Multiline(key = '-OUT_Y_PHI_S-', s=(15,2), disabled=True,
background_color='#F0F0F0', no_scrollbar=True)
130     ],
131     [sg.Image('images/G_phi.png', s=(50,20)), sg.Input(key = '-OUT_G_PHI_S-',
disabled=True, s=15)],
132     [sg.Image('images/B_phi.png', s=(50,20)), sg.Input(key = '-OUT_B_PHI_S-',
disabled=True, s=15)],
133     [
134         sg.Image('images/Z_phi.png', s=(50,20)),
135         sg.Multiline(key = '-OUT_Z_PHI_S-', s=(15,2), disabled=True,
background_color='#F0F0F0', no_scrollbar=True)
136     ],
137     [sg.Image('images/R_c.png', s=(50,20)), sg.Input(key = '-OUT_R_PHI_S-',
disabled=True, s=15)],
138     [sg.Image('images/X_m.png', s=(50,20)), sg.Input(key = '-OUT_X_PHI_S-',
disabled=True, s=15)]
139 ])
140
141 frame_imp_refer_p = sg.Column([
142     [
143         sg.Frame('Referred to Primary',
144             layout = [
145                 [sg.Sizer(25,25)],
146                 [col_imp_refer_p_ser, col_imp_refer_p_phi, sg.Sizer(10,10),
sg.Image('images/Refer_p.png')],
147                 [sg.Sizer(25,25)]
148             ])
149     ]
150 ])
151
152 frame_imp_refer_s = sg.Column([
153     [sg.Frame('Referred to Secondary',
154         layout = [
155             [sg.Sizer(25,25)],
156             [sg.Image('images/Refer_s.png'), sg.Sizer(10,10),
col_imp_refer_s_ser, col_imp_refer_s_phi],
157             [sg.Sizer(25,25)]
158         ])]
159 ])
160
161
162 # VOLTAGE REGULATION TAB
163
164 col_vr_calc = sg.Column([
165     [
166         sg.Text('Rated Secondary Current,'),
167         sg.Image('images/I_2,rated.png', s=(50,20)),
168         sg.Input(key = '-OUT_I_S_RATED-', disabled=True, s=10)
```

```
169     ],
170     [
171         sg.Text('Full Load Secondary Voltage, '),
172         sg.Image('images/V_2,fl.png', s=(50,20)),
173         sg.Input(key = '-OUT_V_S_FL-', disabled=True, s=10)
174     ],
175     [
176         sg.Text('No Load Secondary Voltage, '),
177         sg.Image('images/V_2,nl.png', s=(50,20)),
178         sg.Input(key = '-OUT_V_S_NL-', disabled=True, s=10)
179     ],
180     [
181         sg.Text('Voltage Regulation, '),
182         sg.Image('images/VR.png', s=(50,20)),
183         sg.Input(key = '-OUT_VR-', disabled=True, s=10)
184     ],
185     [
186         sg.Text('Voltage Regulation at 0.8PF lagging (Inductive Load, '),
187             sg.Image('images/VR_0.8,lag.png', s=(75,20)),
188         sg.Input(key = '-OUT_VR_LAG-', disabled=True, s=10)
189     ],
190     [
191         sg.Text('Voltage Regulation at 0.8PF leading (Capacitive Load, '),
192             sg.Image('images/VR_0.8,lead.png', s=(75,20)),
193         sg.Input(key = '-OUT_VR_LEAD-', disabled=True, s=10)
194     ]
195 ], element_justification='right')
196
197 frame_vr_calc = sg.Column([
198     [sg.Frame('',
199         layout=[
200             [sg.Sizer(10,10)],
201             [sg.Sizer(20,1), col_vr_calc, sg.Sizer(20,1)],
202             [sg.Sizer(10,10)]
203         ])]
204 ])
205
206 col_vr_canvas = sg.Column([
207     [
208         sg.Frame('Plot',
209             layout = [
210                 [sg.Sizer(15,15)],
211                 [sg.Sizer(20,20), sg.Canvas(key = '-VR_CANVAS-', sg.Sizer(20,20)],
212                 [sg.Sizer(20,20)]
213             ]
214         )
215     ]
216 ])
```

```
217
218
219 # EFFICIENCY TAB
220
221 col_eff_calc = sg.Column([
222     [
223         sg.Text('Input Power, '),
224         sg.Image('images/P_in.png', s=(40,20)),
225         sg.Input(key = '-OUT_P_IN-', disabled=True, s=10)
226     ],
227     [
228         sg.Text('Output Power, '),
229         sg.Image('images/P_out.png', s=(40,20)),
230         sg.Input(key = '-OUT_P_OUT-', disabled=True, s=10)
231     ],
232     [
233         sg.Text('Copper Loss, '),
234         sg.Image('images/P_Cu.png', s=(40,20)),
235         sg.Input(key = '-OUT_P_CU-', disabled=True, s=10)
236     ],
237     [
238         sg.Text('Core Loss, '),
239         sg.Image('images/P_core.png', s=(40,20)),
240         sg.Input(key = '-OUT_P_C-', disabled=True, s=10)
241     ],
242     [
243         sg.Text('Efficiency, '),
244         sg.Image('images/eta.png', s=(40,20)),
245         sg.Input(key = '-OUT_EFF-', disabled=True, s=10)
246     ]
247 ], element_justification='right')
248
249 frame_eff_calc = sg.Column([
250     [sg.Frame('',
251         layout=[
252             [sg.Sizer(10,10)],
253             [sg.Sizer(70,1), col_eff_calc, sg.Sizer(70,1)],
254             [sg.Sizer(10,10)]
255         ])]
256 ])
257
258 col_eff_canvas = sg.Column([
259     [
260         sg.Frame('Plot',
261             layout = [
262                 [sg.Sizer(15,15)],
263                 [sg.Sizer(20,150), sg.Canvas(key = '-
264 EFF_CANVAS-', sg.Sizer(20,150)],
265                 [sg.Sizer(20,20)]
266             ]
267         )
268     ]
269 ])
```

```
265         )
266     ]
267 ])
268
269
270 # CREDITS TAB
271
272 col_crd = [
273     [sg.Text(course, font=('JetBrains Mono', 15, 'bold'))],
274     [sg.Sizer(30,30)],
275     [sg.Text(author1, font=('JetBrains Mono', 12))],
276     [sg.Sizer(5,5)],
277     [sg.Text(author2, font=('JetBrains Mono', 12))],
278     [sg.Sizer(50,50)],
279     [sg.Text('CREDITS:', font=('Aptos', 9, 'bold'))],
280     [sg.Text(credits, font=('Aptos', 9, 'italic'), justification='center')]
281 ]
282
283
284 # LAYING DOWN ALL TABS TOGETHER IN THE WINDOW
285
286 tab_input = [
287     [sg.Sizer(25,25)],
288     [sg.Text(app_title, font=('Aptos', 15, 'bold'), s=3,
289 justification='center', expand_x=True)],
290     [sg.Sizer(15,15)],
291     [sg.Text(tab_input_instr, justification='center', expand_x=True)],
292     [sg.Text(tab_input_note, font=('Aptos', 9),
293 colors='red', justification='center', expand_x=True)],
294     [sg.Sizer(5,5)],
295     [sg.Push(), frame_input, sg.Image('images/Transformer.png') , sg.Push()],
296     [sg.Sizer(25,25)],
297     [sg.Push(), sg.Button('Calculate', key='-CALCULATE-', enable_events=True,
298 expand_x=True), sg.Push()],
299     [sg.Sizer(5,5)],
300     [sg.Push(), sg.Text(key='-WARNING-', font=('Aptos', 10), colors='red',
301 justification='center', expand_x=True), sg.Push()]
302 ]
303
304 tab_imp = [
305     [sg.Sizer(25,25)],
306     [sg.Push(), frame_imp_refer_p, sg.Push()],
307     [sg.Sizer(25,25)],
308     [sg.Push(), frame_imp_refer_s, sg.Push()],
309     [sg.Sizer(25,25)]
310 ]
311
312 tab_vr = [[sg.Push(), frame_vr_calc, sg.Push()], [sg.Push(), col_vr_canvas,
313 sg.Push()]]
314
315 tab_eff = [[sg.Push(), frame_eff_calc, sg.Push()], [sg.Sizer(28,28)], [sg.Push(),
316 col_eff_canvas, sg.Push()]]
```

```
308 tab_crd = [[sg.VPush()], [sg.Push(), sg.Column(col_crd, element_justification='c'),
309 sg.Push()], [sg.VPush()]]
310 layout = [[sg.TabGroup(
311     [[
312         sg.Tab('Input', tab_input),
313         sg.Tab('Impedances', tab_imp),
314         sg.Tab('Voltage Regulation', tab_vr),
315         sg.Tab('Efficiency', tab_eff),
316         sg.Tab('*', tab_crd)
317     ]]]
318 )]]
319
320 window = sg.Window(app_title.title(), layout, font=font, finalize=True,
321 element_justification='c')
322
323
324
325 # ***CALCLATIONS***
326
327 # more accurate significant figure function for smaller values
328 def sig_figs(x: float, precision: int):
329     x = float(x)
330     precision = int(precision)
331
332     return round(x, -int(floor(log10(abs(x)))) + (precision - 1))
333
334
335 # transformer ratio
336
337 def calc_ratio(v_p, v_s):
338     v_p = int(v_p)
339     v_s = int(v_s)
340
341     t_ratio = int(v_p/v_s)
342
343     return t_ratio
344
345
346 # series equivalent impedance function
347
348 def calc_z_eq(v_sc, i_sc, p_sc):
349     v_sc = float(v_sc)
350     i_sc = float(i_sc)
351     p_sc = float(p_sc)
352
353
354     pf = p_sc / (v_sc * i_sc)
```



```
355
356     # try:
357     #     isinstance(pf, complex)
358     # except:
359     #     print('Entered short circuit parameters return complex power factor!')
360     #     return
361
362     z_eq_mag = round(v_sc/i_sc,2)
363     z_eq_ang = round((np.arccos(pf)*180/np.pi),2)
364
365     r_eq = round(p_sc/(i_sc**2),2)
366     x_l_eq = round(np.sqrt((z_eq_mag**2) - (r_eq**2)),2)
367
368     return r_eq, x_l_eq, z_eq_mag, z_eq_ang
369
370
371 # parallel (excitation branch) impedance function
372
373 def calc_z_phi(v_oc, i_oc, p_oc):
374     v_oc = float(v_oc)
375     i_oc = float(i_oc)
376     p_oc = float(p_oc)
377
378     pf = round(p_oc / (v_oc * i_oc),3)
379
380     y_phi_mag = sig_figs((i_oc/v_oc),3)
381     y_phi_ang = sig_figs((-np.arccos(pf)*180/np.pi),3)
382
383     g_phi = sig_figs(y_phi_mag * pf, 3)
384     b_phi = sig_figs(y_phi_mag * np.sin(y_phi_ang * np.pi / 180),3)
385
386     r_phi = sig_figs((1 / g_phi), 3)
387     x_phi = sig_figs(1 / abs(b_phi), 3)
388     z_phi_mag = sig_figs((1 / y_phi_mag), 3)
389     z_phi_ang = -y_phi_ang
390
391     return g_phi, b_phi, y_phi_mag, y_phi_ang, r_phi, x_phi, z_phi_mag, z_phi_ang
392
393
394 # find values referred to primary or secondary
395
396 def refer_to_s(value_at_p, t_ratio):
397
398     value_at_p = float(value_at_p)
399     t_ratio = int(t_ratio)
400
401     value_at_s = sig_figs((value_at_p/(t_ratio**2)), 3)
402
403     return value_at_s
```

```
404
405 def refer_to_p(value_at_s, t_ratio):
406
407     value_at_s = float(value_at_s)
408     t_ratio = int(t_ratio)
409
410     value_at_p = sig_figs((value_at_s*(t_ratio**2)), 3)
411
412     return value_at_p
413
414
415 # voltage regulation
416
417 def calc_vr(sRated, v_p, v_s, r_eq, x_l_eq):
418
419     sRated = float(sRated)
420     v_p = float(v_p)
421     v_s = float(v_s)
422     r_eq = float(r_eq/100)
423     x_l_eq = float(x_l_eq/100)
424
425     i_s = round((sRated / v_s), 2)
426
427     v_p_a = complex(v_s,0) + (r_eq * complex(i_s,0)) +
428     (complex(0,x_l_eq)*complex(i_s,0))
429     # v_p_a = v_s + (r_eq * i_s) + ((x_l_eq*1j)*i_s)
430     v_s_n1 = round(abs(v_p_a),2)
431
432     vr = sig_figs(((v_s_n1 - v_s) / v_s) * 100,2)
433
434     pf_lag = 0.8
435     pf_lead = 0.8
436     i_s_ang_lag = (-np.arccos(pf_lag))
437     i_s_ang_lead = (np.arccos(pf_lead))
438     i_s_lag = complex(i_s*np.cos(i_s_ang_lag), i_s*np.sin(i_s_ang_lag))
439     i_s_lead = complex(i_s*np.cos(i_s_ang_lead), i_s*np.sin(i_s_ang_lead))
440
441     v_p_a_lag = complex(v_s,0) + (r_eq * i_s_lag) + (complex(0,x_l_eq)*i_s_lag)
442     v_p_a_lead = complex(v_s,0) + (r_eq * i_s_lead) + (complex(0,x_l_eq)*i_s_lead)
443     v_s_n1_lag = round(abs(v_p_a_lag),2)
444     v_s_n1_lead = round(abs(v_p_a_lead),2)
445
446     vr_lag = sig_figs(((v_s_n1_lag - v_s) / v_s) * 100, 2)
447     vr_lead = sig_figs(((v_s_n1_lead - v_s) / v_s) * 100, 2)
448
449
450     i_s_range = np.linspace(0,i_s,100)
```

```
451     i_s_lag_range = i_s_range*np.cos(i_s_ang_lag) +  
(i_s_range*np.sin(i_s_ang_lag)*1j)  
452     i_s_lead_range = i_s_range*np.cos(i_s_ang_lead) +  
(i_s_range*np.sin(i_s_ang_lead)*1j)  
453  
454     v_p_a_range = v_s + (r_eq * i_s_range) + ((x_l_eq*1j)*i_s_range)  
455     v_p_a_lag_range = v_s + (r_eq * i_s_lag_range) + ((x_l_eq*1j)*i_s_lag_range)  
456     v_p_a_lead_range = v_s + (r_eq * i_s_lead_range) + ((x_l_eq*1j)*i_s_lead_range)  
457  
458     vr_range = ((np.absolute(v_p_a_range) - v_s) / v_s) * 100  
459     vr_lag_range = ((np.absolute(v_p_a_lag_range) - v_s) / v_s) * 100  
460     vr_lead_range = ((np.absolute(v_p_a_lead_range) - v_s) / v_s) * 100  
461  
462  
463     return i_s, v_s, v_s_nl, vr, vr_lag, vr_lead, \  
464         i_s_range, vr_range, i_s_lag_range, vr_lag_range, i_s_lead_range,  
vr_lead_range  
465  
466  
467 # transformer efficiency function  
468  
469 def calc_eff(sRated, v_s, i_s, v_s_nl, r_eq, r_phi):  
470  
471     sRated = float(sRated)  
472     v_s = float(v_s)  
473     i_s = float(i_s)  
474     v_s_nl = float(v_s_nl)  
475     r_eq = float(r_eq/100)  
476     r_phi = float(r_phi)  
477  
478  
479     pow_out = round(v_s * i_s, 2)  
480     loss_cu = round((i_s**2)*r_eq, 2)  
481     loss_core = round((v_s_nl**2)/r_phi, 2) # voltage dependent. doesnt vary with  
load  
482     pow_in = round(pow_out + loss_cu + loss_core, 2)  
483  
484     eff = round((pow_out / pow_in) * 100, 2)  
485  
486     i_s = sRated / v_s  
487     i_s_range_eff = np.linspace(0, 2*i_s,100)  
488  
489     pow_out_range = v_s * i_s_range_eff  
490  
491     loss_cu_range = (i_s_range_eff**2)*r_eq  
492  
493     pow_in_range = pow_out_range + loss_cu_range + loss_core  
494  
495     loss_cu_range_perc = loss_cu_range  
496     loss_core_range_perc = [loss_core] * len(i_s_range_eff)
```

```
497     eff_range = (pow_out_range / pow_in_range ) * 100
498
499
500     return pow_in, pow_out, loss_cu, loss_core, eff, \
501           eff_range, i_s_range_eff, loss_cu_range_perc, loss_core_range_perc
502
503
504
505 # *** PLOTS ***
506
507 # voltage regulation plot
508
509 fig_vr = plt.figure(1,figsize = (6,4.5))
510 fig_vr.add_subplot(111).plot([],[])
511 tkcanvas_agg_vr = FigureCanvasTkAgg(fig_vr, window['-VR_CANVAS-'].TKCanvas)
512 tkcanvas_agg_vr.draw()
513 tkcanvas_agg_vr.get_tk_widget().pack()
514
515 def vr_plot(i_s_range, vr_range,
516            i_s_lag_range, vr_lag_range,
517            i_s_lead_range, vr_lead_range):
518
519     ax = fig_vr.axes
520     ax[0].cla()
521     ax[0].plot(i_s_range, vr_range)
522     ax[0].plot(i_s_lag_range, vr_lag_range, 'r-.')
523     ax[0].plot(i_s_lead_range, vr_lead_range, 'g--')
524     ax[0].set_xlabel('Load (A)')
525     ax[0].set_ylabel('VR (%)')
526     ax[0].set_title('VR variation with load amount and type')
527     ax[0].legend(['1 PF', '0.8 lag PF', '0.8 lead PF'], loc='best')
528     tkcanvas_agg_vr.draw()
529     tkcanvas_agg_vr.get_tk_widget().pack()
530
531
532 # efficiency plot
533
534 fig_eff = plt.figure(2,figsize = (6,4.5))
535 fig_eff.add_subplot(111).plot([],[])
536 tkcanvas_agg_eff = FigureCanvasTkAgg(fig_eff, window['-EFF_CANVAS-'].TKCanvas)
537 tkcanvas_agg_eff.draw()
538 tkcanvas_agg_eff.get_tk_widget().pack()
539
540 def eff_plot(i_s_range_eff, eff_range, loss_cu_range_perc, loss_core_range_perc):
541
542     axes = fig_eff.axes
543     axes[0].cla()
544     axes[0].plot(i_s_range_eff, eff_range, label='Efficiency')
545     axes[0].set_xlabel('Current Load (A)')
```

```
546 axes[0].set_ylabel('Efficiency (%)')
547 axes[0].set_ylim(80,100)
548
549 for ax in fig_eff.axes:
550     if ax is not axes[0]:
551         fig_eff.delaxes(ax)
552
553 ax1 = axes[0].twinx()
554
555 ax1.plot(i_s_range_eff, loss_cu_range_perc, 'r-.', label='Copper loss')
556 ax1.plot(i_s_range_eff, loss_core_range_perc, 'g--', label='Core loss')
557 ax1.set_ylabel('Loss (W)')
558 ax1.set_title('Efficiency variation with load')
559
560 lines, labels = axes[0].get_legend_handles_labels()
561 lines2, labels2 = ax1.get_legend_handles_labels()
562 ax1.legend(lines + lines2, labels + labels2, loc='best')
563
564 tkcanvas_agg_eff.draw()
565 tkcanvas_agg_eff.get_tk_widget().pack()
566
567
568
569 # *** WHEN WINDOW ACTIVE ***
570
571 # for input validation
572
573 prompt = window['-WARNING-'].update
574 input_key_list = [key for key, value in window.key_dict.items()
575                  if isinstance(value, sg.Input)]
576 input_key_list_slice = input_key_list[:9]
577
578 while True:
579     event, values = window.read()
580     if event == sg.WIN_CLOSED:
581         break
582
583
584 # ON CALCULATE KEYPRESS
585
586 if event == '-CALCULATE-':
587
588
589     # validate input fields not empty
590
591     if all(map(str.strip, [values[key] for key in input_key_list_slice])):
592
593
594         # validate input fields numbers
```

```
595
596         if all(isinstance(values.get(key, ""), str) and any(char.isnumeric() for
char in values[key])) for key in input_key_list_slice):
597             prompt('Calculated!')
598
599
600             # call transformer ratio function, return its value
601             t_ratio = calc_ratio(values['-INP_V_P-'], values['-INP_V_S-'])
602             out_t_ratio = t_ratio
603             window['-OUT_RATIO-'].update(out_t_ratio)
604
605
606             # call series impedance function, return outputs then display
607             # series impedance retrieved is that referred to primary
608             # since LV secondary is short circuited and values are referred to
primary
609
610             # refer_to_s function is called to find values referred to secondary
611             r_eq, x_l_eq, z_eq_mag, z_eq_ang = calc_z_eq(
612                 values['-INP_V_SC-'], values['-INP_I_SC-'], values['-INP_P_SC-']
613             )
614
615             z_eq_polar = f'{z_eq_mag} \u2220 {z_eq_ang} \u03a9'
616             z_eq_rect = f'{r_eq}{x_l_eq:+}j \u03a9'
617
618             out_z_eq = f'{z_eq_polar} \n {z_eq_rect}'
619             out_r_eq = f'{r_eq} \u03a9'
620             out_x_l_eq = f'{x_l_eq}j \u03a9'
621
622             window['-OUT_Z_EQ_P-'].update(out_z_eq)
623             window['-OUT_R_EQ_P-'].update(out_r_eq)
624             window['-OUT_X_L_EQ_P-'].update(out_x_l_eq)
625
626             z_eq_mag_refer_s = refer_to_s(z_eq_mag, t_ratio)
627             r_eq_refer_s = refer_to_s(r_eq, t_ratio)
628             x_l_eq_refer_s = refer_to_s(x_l_eq, t_ratio)
629
630             z_eq_polar_refer_s = f'{z_eq_mag_refer_s}\u2220{z_eq_ang} \u03a9'
631             z_eq_rect_refer_s = f'{r_eq_refer_s}{x_l_eq_refer_s:+}j \u03a9'
632
633             out_z_eq_refer_s = f'{z_eq_polar_refer_s}\n{z_eq_rect_refer_s}'
634             out_r_eq_refer_s = f'{r_eq_refer_s} \u03a9'
635             out_x_l_eq_refer_s = f'{x_l_eq_refer_s}j \u03a9'
636
637             window['-OUT_Z_EQ_S-'].update(out_z_eq_refer_s)
638             window['-OUT_R_EQ_S-'].update(out_r_eq_refer_s)
639             window['-OUT_X_L_EQ_S-'].update(out_x_l_eq_refer_s)
640
641             # call parallel impedance function, return and display 6 outputs
```

```
642         # parallel impedance retrieved is that referred to secondary
643         # since HV primary is open circuited and values are referred to
secondary
644         # refer_to_p function is called to calculate values referred to
primary
645         g_phi, b_phi, y_phi_mag, y_phi_ang, r_phi, x_phi, z_phi_mag,
z_phi_ang = calc_z_phi(
646             values['-INP_V_OC-'], values['-INP_I_OC-'], values['-INP_P_OC-']
647         )
648
649         y_phi_polar = f'{y_phi_mag} \u2220 {y_phi_ang} \u03a9'
650         y_phi_rect = f'{g_phi}{b_phi:+}j \u03a9'
651
652         z_phi_polar = f'{z_phi_mag} \u2220 {z_phi_ang} \u03a9'
653         z_phi_rect = f'{r_phi}{x_phi:+}j \u03a9'
654
655         out_y_phi = f'{y_phi_polar}\n{y_phi_rect}'
656         out_g_phi = f'{g_phi} \u03a9'
657         out_b_phi = f'{b_phi}j \u03a9'
658         out_z_phi = f'{z_phi_polar}\n{z_phi_rect}'
659         out_r_phi = f'{r_phi} \u03a9'
660         out_x_phi = f'{x_phi}j \u03a9'
661
662         window['-OUT_Y_PHI_S-'].update(out_y_phi)
663         window['-OUT_G_PHI_S-'].update(out_g_phi)
664         window['-OUT_B_PHI_S-'].update(out_b_phi)
665         window['-OUT_Z_PHI_S-'].update(out_z_phi)
666         window['-OUT_R_PHI_S-'].update(out_r_phi)
667         window['-OUT_X_PHI_S-'].update(out_x_phi)
668
669         y_phi_mag_refer_p = refer_to_p(y_phi_mag, t_ratio)
670         g_phi_refer_p = refer_to_p(g_phi, t_ratio)
671         b_phi_refer_p = refer_to_p(b_phi, t_ratio)
672         z_phi_mag_refer_p = refer_to_p(z_phi_mag, t_ratio)
673         r_phi_refer_p = refer_to_p(r_phi, t_ratio)
674         x_phi_refer_p = refer_to_p(x_phi, t_ratio)
675
676         y_phi_polar_refer_p = f'{y_phi_mag_refer_p} \u2220 {y_phi_ang}
\u03a9'
677         y_phi_rect_refer_p = f'{g_phi_refer_p}{b_phi_refer_p:+}j \u03a9'
678
679         z_phi_polar_refer_p = f'{z_phi_mag_refer_p} \u2220 {z_phi_ang}
\u03a9'
680         z_phi_rect_refer_p = f'{r_phi_refer_p}{x_phi_refer_p:+}j \u03a9'
681
682         out_y_phi_refer_p = f'{y_phi_polar_refer_p}\n{y_phi_rect_refer_p}'
683         out_g_phi_refer_p = f'{g_phi_refer_p} \u03a9'
684         out_b_phi_refer_p = f'{b_phi_refer_p}j \u03a9'
685         out_z_phi_refer_p = f'{z_phi_polar_refer_p}\n{z_phi_rect_refer_p}'
686         out_r_phi_refer_p = f'{r_phi_refer_p} \u03a9'
```

```
687         out_x_phi_refer_p = f'{x_phi_refer_p}j \u03a9'
688
689         window['-OUT_Y_PHI_P-'].update(out_y_phi_refer_p)
690         window['-OUT_G_PHI_P-'].update(out_g_phi_refer_p)
691         window['-OUT_B_PHI_P-'].update(out_b_phi_refer_p)
692         window['-OUT_Z_PHI_P-'].update(out_z_phi_refer_p)
693         window['-OUT_R_PHI_P-'].update(out_r_phi_refer_p)
694         window['-OUT_X_PHI_P-'].update(out_x_phi_refer_p)
695
696
697         # call voltage regulation function, return outputs
698         i_s, v_s_fl, v_s_nl, vr, vr_lag, vr_lead, i_s_range, vr_range, \
699             i_s_lag_range, vr_lag_range, i_s_lead_range, vr_lead_range =
700         calc_vr(
701             r_eq, x_l_eq,
702             values['-INP_S-'], values['-INP_V_P-'], values['-INP_V_S-'],
703             )
704         out_i_s = f'{i_s} A'
705         out_v_s_fl = f'{v_s_fl} V'
706         out_v_s_nl = f'{v_s_nl} V'
707         out_vr = f'{vr} %'
708         out_vr_lag = f'{vr_lag} %'
709         out_vr_lead = f'{vr_lead} %'
710
711         window['-OUT_I_S_RATED-'].update(out_i_s)
712         window['-OUT_V_S_FL-'].update(out_v_s_fl)
713         window['-OUT_V_S_NL-'].update(out_v_s_nl)
714         window['-OUT_VR-'].update(out_vr)
715         window['-OUT_VR_LAG-'].update(out_vr_lag)
716         window['-OUT_VR_LEAD-'].update(out_vr_lead)
717
718         # plot voltage regulation
719         vr_plot(i_s_range, vr_range,
720                 i_s_lag_range, vr_lag_range,
721                 i_s_lead_range, vr_lead_range
722                 )
723
724         # call efficiency function
725         pow_in, pow_out, loss_cu, loss_core, eff, eff_range, \
726             i_s_range_eff, loss_cu_range_perc, loss_core_range_perc =
727         calc_eff(
728             values['-INP_S-'], values['-INP_V_S-'], i_s, v_s_nl, r_eq, r_phi
729             )
730         out_pow_in = f'{pow_in} W'
731         out_pow_out = f'{pow_out} W'
732         out_loss_cu = f'{loss_cu} W'
733         out_loss_core = f'{loss_core} W'
734         out_eff = f'{eff} %'
```



```
733
734         window['-OUT_P_IN-'].update(out_pow_in)
735         window['-OUT_P_OUT-'].update(out_pow_out)
736         window['-OUT_P_CU-'].update(out_loss_cu)
737         window['-OUT_P_C-'].update(out_loss_core)
738         window['-OUT_EFF-'].update(out_eff)
739
740         # plot efficiency
741         eff_plot(i_s_range_eff, eff_range, loss_cu_range_perc,
loss_core_range_perc)
742
743
744
745         else:
746             prompt('Enter valid input!')
747
748
749         else:
750             prompt("Fill all the fields!")
751
752
753
754 window.close()
755
```