

```
1
2 # *** REFERENCES ***
3
4 # regarding the topic:
5 # Electric Machinery Fundamentals, Stephen J. Chapman, ed.5
6
7 # regarding the gui library and code:
8 # https://docs.pysimplegui.com/en/latest/
9 # https://youtu.be/kQ8DGP9p2LY?si=1K0mzYeDvAGHe7ox
10 # https://matplotlib.org/stable/users/index
11 # https://stackoverflow.com/
12
13
14
15 import PySimpleGUI as sg
16 import numpy as np
17 from math import floor, log10
18 import matplotlib.pyplot as plt
19 from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
20
21
22
23 # ***BUILDING THE UI***
24
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 = 'Ismet Mert ŞEN'
33
34
35 # INPUT TAB
36
37 tab_input_instr = 'Provide all given values to obtain transformer characteristics'
38 tab_input_note = '*Note: only step-down voltages permitted'
39
40 col_input_data = sg.Column([
41     [sg.Push(), sg.Text('Power Rating: '), sg.Input(15000, key = '-INP_S-', size=7),
42      sg.Text('VA', s=5)],
43     [sg.Push(), sg.Text('Primary Voltage: '), sg.Input(2300, key = '-INP_V_P-',
44      size=7), sg.Text('V', s=5)],
45     [sg.Push(), sg.Text('Secondary Voltage: '), sg.Input(230, key = '-INP_V_S-',
46      size=7), sg.Text('V', s=5)]
47 ])
48
49 col_input_oc = sg.Column([
```

```
47     [sg.Push(), sg.Image('images/V_OC.png'), sg.Input(230, key = '-INP_V_OC-',
48 size=5), sg.Text('V', s=5)],
49     [sg.Push(), sg.Image('images/I_OC.png'), sg.Input(2.1, key = '-INP_I_OC-',
50 size=5), sg.Text('A', s=5)],
51     [sg.Push(), sg.Image('images/P_OC.png'), sg.Input(50, key = '-INP_P_OC-', size=5),
52 sg.Text('W', s=5)]
53 ])
54
55 col_input_sc = sg.Column([
56     [sg.Push(), sg.Image('images/V_SC.png'), sg.Input(47, key = '-INP_V_SC-', size=5),
57 sg.Text('V', s=5)],
58     [sg.Push(), sg.Image('images/I_SC.png'), sg.Input(6, key = '-INP_I_SC-', size=5),
59 sg.Text('A', s=5)],
60     [sg.Push(), sg.Image('images/P_SC.png'), sg.Input(160, key = '-INP_P_SC-',
61 size=5), sg.Text('W', s=5)]
62 ])
63
64 col_ratio = sg.Column([
65     [
66         sg.Text('Transformer Ratio, '),
67         sg.Image('images/N.png'),
68         sg.Input(key = '-OUT_RATIO-', disabled=True, s=5),
69         sg.Text('', s=5)
70     ]
71 ])
72
73 col_input = sg.Column([
74     [sg.Sizer(5,5)],
75     [col_input_data],
76     [sg.Sizer(5,5)],
77     [sg.Text('Open Circuit Test', expand_x=True, justification='center'),
78 col_input_oc],
79     [sg.Sizer(5,5)],
80     [sg.Text('Short Circuit Test', expand_x=True, justification='center'),
81 col_input_sc],
82     [sg.Sizer(5,5)],
83     [col_ratio],
84     [sg.Sizer(5,5)]
85 ], element_justification='right')
86
87 frame_input = sg.Column([
88     [
89         sg.Frame('',
90             layout = [
91                 [sg.Sizer(15,15), col_input, sg.Sizer(10,10)]
92             ]
93         )
94     ]
95 ])
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
```

```
89 # IMPEDANCE AND EQUIV CIRCUIT TAB
90
91 col_imp_refer_p_ser = sg.Column([
92     [
93         sg.Image('images/Z_eq.png', s=(50,20)),
94         sg.Multiline(key = '-OUT_Z_EQ_P-', s=(15,2), disabled=True,
95 background_color='#F0F0F0', no_scrollbar=True)
96     ],
97     [sg.Image('images/R_eq.png', s=(50,20)), sg.Input(key = '-OUT_R_EQ_P-',
98 disabled=True, s=15)],
99     [sg.Image('images/X_l_eq.png', s=(50,20)), sg.Input(key = '-OUT_X_L_EQ_P-',
100 disabled=True, s=15)]
101 ])
102
103 col_imp_refer_p_phi = sg.Column([
104     [
105         sg.Image('images/Y_phi.png', s=(50,20)),
106         sg.Multiline(key = '-OUT_Y_PHI_P-', s=(15,2), disabled=True,
107 background_color='#F0F0F0', no_scrollbar=True)
108     ],
109     [sg.Image('images/G_phi.png', s=(50,20)), sg.Input(key = '-OUT_G_PHI_P-',
110 disabled=True, s=15)],
111     [sg.Image('images/B_phi.png', s=(50,20)), sg.Input(key = '-OUT_B_PHI_P-',
112 disabled=True, s=15)],
113     [
114         sg.Image('images/Z_phi.png', s=(50,20)),
115         sg.Multiline(key = '-OUT_Z_PHI_P-', s=(15,2), disabled=True,
116 background_color='#F0F0F0', no_scrollbar=True)
117     ],
118     [sg.Image('images/R_c.png', s=(50,20)), sg.Input(key = '-OUT_R_PHI_P-',
119 disabled=True, s=15)],
120     [sg.Image('images/X_m.png', s=(50,20)), sg.Input(key = '-OUT_X_PHI_P-',
121 disabled=True, s=15)]
122 ])
123
124 col_imp_refer_s_ser = sg.Column([
125     [
126         sg.Image('images/Z_eq.png', s=(50,20)),
127         sg.Multiline(key = '-OUT_Z_EQ_S-', s=(15,2), disabled=True,
128 background_color='#F0F0F0', no_scrollbar=True)
129     ],
130     [sg.Image('images/R_eq.png', s=(50,20)), sg.Input(key = '-OUT_R_EQ_S-',
131 disabled=True, s=15)],
132     [sg.Image('images/X_l_eq.png', s=(50,20)), sg.Input(key = '-OUT_X_L_EQ_S-',
133 disabled=True, s=15)]
134 ])
135
136 col_imp_refer_s_phi = sg.Column([
137     [
138         sg.Image('images/Y_phi.png', s=(50,20)),
```

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

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

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

```
265 ])
266
267
268 col_crd = [
269     [sg.Text(course, font=('JetBrains Mono', 15, 'bold'))],
270     [sg.Sizer(30,30)],
271     [sg.Text(author1, font=('JetBrains Mono', 12))],
272     [sg.Sizer(5,5)],
273     [sg.Text(author2, font=('JetBrains Mono', 12))],
274     [sg.Sizer(30,30)]
275 ]
276
277
278 # LAYING DOWN ALL TABS TOGETHER IN THE WINDOW
279
280 tab_input = [
281     [sg.Sizer(25,25)],
282     [sg.Text(app_title, font=('Aptos', 15, 'bold'), s=3,
283 justification='center', expand_x=True)],
284     [sg.Sizer(15,15)],
285     [sg.Text(tab_input_instr, justification='center', expand_x=True)],
286     [sg.Text(tab_input_note, font=('Aptos', 9),
287 colors='red', justification='center', expand_x=True)],
288     [sg.Sizer(5,5)],
289     [sg.Push(), frame_input, sg.Image('images/Transformer.png') , sg.Push()],
290     [sg.Sizer(25,25)],
291     [sg.Push(), sg.Button('Calculate', key='-CALCULATE-', enable_events=True,
292 expand_x=True), sg.Push()],
293     [sg.Sizer(5,5)],
294     [sg.Push(), sg.Text(key='-WARNING-', font=('Aptos', 10), colors='red',
295 justification='center', expand_x=True), sg.Push()]
296 ]
297
298 tab_imp = [
299     [sg.Sizer(25,25)],
300     [sg.Push(), frame_imp_refer_p, sg.Push()],
301     [sg.Sizer(25,25)],
302     [sg.Push(), frame_imp_refer_s, sg.Push()],
303     [sg.Sizer(25,25)]
304 ]
305
306 tab_vr = [[sg.Push(), frame_vr_calc, sg.Push()], [sg.Push(), col_vr_canvas,
307 sg.Push()]]
308
309 tab_eff = [[sg.Push(), frame_eff_calc, sg.Push()], [sg.Sizer(28,28)], [sg.Push(),
310 col_eff_canvas, sg.Push()]]
311
312 tab_crd = [[sg.VPush()], [sg.Push(), sg.Column(col_crd, element_justification='c'),
313 sg.Push()], [sg.VPush()]]
314
315
316 layout = [[sg.TabGroup(
317     [[
318         sg.Tab('Input', tab_input),
319         sg.Tab('Impedances', tab_imp),
```

```
308         sg.Tab('Voltage Regulation', tab_vr),
309         sg.Tab('Efficiency', tab_eff),
310         sg.Tab('*', tab_crd)
311     ]]
312 )]]
313
314 window = sg.Window(app_title.title(), layout, font=font, finalize=True,
315                    element_justification='c')
316
317
318
319 # ***CALCLATIONS***
320
321 # more accurate significant figure function for smaller values
322 def sig_figs(x: float, precision: int):
323     x = float(x)
324     precision = int(precision)
325
326     return round(x, -int(floor(log10(abs(x)))) + (precision - 1))
327
328
329 # transformer ratio
330
331 def calc_ratio(v_p, v_s):
332     v_p = int(v_p)
333     v_s = int(v_s)
334
335     t_ratio = int(v_p/v_s)
336
337     return t_ratio
338
339
340 # series equivalent impedance function
341
342 def calc_z_eq(v_sc, i_sc, p_sc):
343     v_sc = float(v_sc)
344     i_sc = float(i_sc)
345     p_sc = float(p_sc)
346
347
348     pf = p_sc / (v_sc * i_sc)
349
350     # try:
351     #     isinstance(pf, complex)
352     # except:
353     #     print('Entered short circuit parameters return complex power factor!')
354     #     return
355
```



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

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

```
451
452     vr_range = ((np.absolute(v_p_a_range) - v_s) / v_s) * 100
453     vr_lag_range = ((np.absolute(v_p_a_lag_range) - v_s) / v_s) * 100
454     vr_lead_range = ((np.absolute(v_p_a_lead_range) - v_s) / v_s) * 100
455
456
457     return i_s, v_s, v_s_nl, vr, vr_lag, vr_lead, \
458           i_s_range, vr_range, i_s_lag_range, vr_lag_range, i_s_lead_range,
459 vr_lead_range
460
461 # transformer efficiency function
462
463 def calc_eff(sRated, v_s, i_s, v_s_nl, r_eq, r_phi):
464
465     sRated = float(sRated)
466     v_s = float(v_s)
467     i_s = float(i_s)
468     v_s_nl = float(v_s_nl)
469     r_eq = float(r_eq/100)
470     r_phi = float(r_phi)
471
472
473     pow_out = round(v_s * i_s, 2)
474     loss_cu = round((i_s**2)*r_eq, 2)
475     loss_core = round((v_s_nl**2)/r_phi, 2) # voltage dependent. doesnt vary with
476 load
477     pow_in = round(pow_out + loss_cu + loss_core, 2)
478
479     eff = round((pow_out / pow_in) * 100, 2)
480
481     i_s = sRated / v_s
482     i_s_range_eff = np.linspace(0, 2*i_s,100)
483
484     pow_out_range = v_s * i_s_range_eff
485
486     loss_cu_range = (i_s_range_eff**2)*r_eq
487
488     pow_in_range = pow_out_range + loss_cu_range + loss_core
489
490     loss_cu_range_perc = loss_cu_range
491     loss_core_range_perc = [loss_core] * len(i_s_range_eff)
492     eff_range = (pow_out_range / pow_in_range ) * 100
493
494     return pow_in, pow_out, loss_cu, loss_core, eff, \
495           eff_range, i_s_range_eff, loss_cu_range_perc, loss_core_range_perc
496
497
```

```
498
499 # *** PLOTS ***
500
501 # VR plot
502
503 fig_vr = plt.figure(1,figsize = (6,4.5))
504 fig_vr.add_subplot(111).plot([],[])
505 tkcanvas_agg_vr = FigureCanvasTkAgg(fig_vr, window['-VR_CANVAS-'].TKCanvas)
506 tkcanvas_agg_vr.draw()
507 tkcanvas_agg_vr.get_tk_widget().pack()
508
509 def vr_plot(i_s_range, vr_range,
510             i_s_lag_range, vr_lag_range,
511             i_s_lead_range, vr_lead_range):
512
513     ax = fig_vr.axes
514     ax[0].cla()
515     ax[0].plot(i_s_range, vr_range)
516     ax[0].plot(i_s_lag_range, vr_lag_range, 'r-.')
517     ax[0].plot(i_s_lead_range, vr_lead_range, 'g--')
518     ax[0].set_xlabel('Load (A)')
519     ax[0].set_ylabel('VR (%)')
520     ax[0].set_title('VR variation with load amount and type')
521     ax[0].legend(['1 PF', '0.8 lag PF', '0.8 lead PF'], loc='best')
522     tkcanvas_agg_vr.draw()
523     tkcanvas_agg_vr.get_tk_widget().pack()
524
525
526 # efficiency plot
527
528 fig_eff = plt.figure(2,figsize = (6,4.5))
529 fig_eff.add_subplot(111).plot([],[])
530 tkcanvas_agg_eff = FigureCanvasTkAgg(fig_eff, window['-EFF_CANVAS-'].TKCanvas)
531 tkcanvas_agg_eff.draw()
532 tkcanvas_agg_eff.get_tk_widget().pack()
533
534 def eff_plot(i_s_range_eff, eff_range, loss_cu_range_perc, loss_core_range_perc):
535
536     axes = fig_eff.axes
537     axes[0].cla()
538     axes[0].plot(i_s_range_eff, eff_range, label='Efficiency')
539     axes[0].set_xlabel('Current Load (A)')
540     axes[0].set_ylabel('Efficiency (%)')
541     axes[0].set_ylim(80,100)
542
543     for ax in fig_eff.axes:
544         if ax is not axes[0]:
545             fig_eff.delaxes(ax)
546
```

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

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

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

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



```
734
735         # plot efficiency
736         eff_plot(i_s_range_eff, eff_range, loss_cu_range_perc,
loss_core_range_perc)
737
738
739
740         else:
741             prompt('Enter valid input!')
742
743
744         else:
745             prompt("Fill all the fields!")
746
747
748
749 window.close()
750
```