

```

1 import numpy as np
2 from matplotlib import pyplot as plt
3 import time
4 from matplotlib import gridspec
5 from numpy import arange
6 from scipy.optimize import curve_fit
7 start_time = time.time()
8
9 def besteDekke(t0):
10
11     ploteGrenser = [{ "mm": [], "m": [], "tap": [], "P0": [], "friktap": [], "svinntap": [], "armering": [],
12                       "Ap": [], "kryp": [], "relak": [], "låsetap": [], "ovre": [], "nedre": [], "grense_overforing": [],
13                       "REI": [], "REI_": [], "t0_": [], "m_": [], "vibrasjon": [], "co2": [], "mmlang": [], "mlang": [],
14                       "valere": [], "armering": [] },
15
16                       { "mm": [], "m": [], "tap": [], "P0": [], "friktap": [], "svinntap": [], "armering": [],
17                       "Ap": [], "kryp": [], "relak": [], "låsetap": [], "ovre": [], "nedre": [], "grense_overforing": [],
18                       "REI": [], "REI_": [], "t0_": [], "m_": [], "vibrasjon": [], "co2": [], "mmlang": [], "mlang": [],
19                       "valere": [], "armering": [] },
20
21                       { "mm": [], "m": [], "tap": [], "P0": [], "friktap": [], "svinntap": [], "armering": [],
22                       "Ap": [], "kryp": [], "relak": [], "låsetap": [], "ovre": [], "nedre": [], "grense_overforing": [],
23                       "REI": [], "REI_": [], "t0_": [], "m_": [], "vibrasjon": [], "co2": [], "mmlang": [], "mlang": [],
24                       "valere": [], "armering": [] },
25
26                       { "mm": [], "m": [], "tap": [], "P0": [], "friktap": [], "svinntap": [], "armering": [],
27                       "Ap": [], "kryp": [], "relak": [], "låsetap": [], "ovre": [], "nedre": [], "grense_overforing": [],
28                       "REI": [], "REI_": [], "t0_": [], "m_": [], "vibrasjon": [], "co2": [], "mmlang": [], "mlang": [],
29                       "valere": [], "armering": [] },
30
31                       { "mm": [], "m": [], "tap": [], "P0": [], "friktap": [], "svinntap": [], "armering": [],
32                       "Ap": [], "kryp": [], "relak": [], "låsetap": [], "ovre": [], "nedre": [], "grense_overforing": [],
33                       "REI": [], "REI_": [], "t0_": [], "m_": [], "vibrasjon": [], "co2": [], "mmlang": [], "mlang": [],
34                       "valere": [], "armering": [] },
35
36                       { "mm": [], "m": [], "tap": [], "P0": [], "friktap": [], "svinntap": [], "armering": [],
37                       "Ap": [], "kryp": [], "relak": [], "låsetap": [], "ovre": [], "nedre": [], "grense_overforing": [],
38                       "REI": [], "REI_": [], "t0_": [], "m_": [], "vibrasjon": [], "co2": [], "mmlang": [], "mlang": [],
39                       "valere": [], "armering": [] },
40
41                       { "mm": [], "m": [], "tap": [], "P0": [], "friktap": [], "svinntap": [], "armering": [],
42                       "Ap": [], "kryp": [], "relak": [], "låsetap": [], "ovre": [], "nedre": [], "grense_overforing": [],
43                       "REI": [], "REI_": [], "t0_": [], "m_": [], "vibrasjon": [], "co2": [], "mmlang": [], "mlang": [],
44                       "valere": [], "armering": [] },
45
46                       { "mm": [], "m": [], "tap": [], "P0": [], "friktap": [], "svinntap": [], "armering": [],
47                       "Ap": [], "kryp": [], "relak": [], "låsetap": [], "ovre": [], "nedre": [], "grense_overforing": [],
48                       "REI": [], "REI_": [], "t0_": [], "m_": [], "vibrasjon": [], "co2": [], "mmlang": [], "mlang": [],
49                       "valere": [], "armering": [] },
50
51                       { "mm": [], "m": [], "tap": [], "P0": [], "friktap": [], "svinntap": [], "armering": [],
52                       "Ap": [], "kryp": [], "relak": [], "låsetap": [], "ovre": [], "nedre": [], "grense_overforing": [],
53                       "REI": [], "REI_": [], "t0_": [], "m_": [], "vibrasjon": [], "co2": [], "mmlang": [], "mlang": [],
54                       "valere": [], "armering": [] },
55
56                       { "mm": [], "m": [], "tap": [], "P0": [], "friktap": [], "svinntap": [], "armering": [],
57                       "Ap": [], "kryp": [], "relak": [], "låsetap": [], "ovre": [], "nedre": [], "grense_overforing": [],
58                       "REI": [], "REI_": [], "t0_": [], "m_": [], "vibrasjon": [], "co2": [], "mmlang": [], "mlang": [],
59                       "valere": [], "armering": [] } ]
60
61
62     total_tap_alle_dekker = []
63     meter = []
64     xx = []
65     yy = []
66     yy_frik = []
67     yy_svinntap = []
68     yy_kryp = []
69     yy_relak = []
70     yy_låsetap = []
71     yy_vibrasjon = []
72
73     As_min_krav = []
74     As_min_L = []
75     Ap Bruker = []
76     Ap_P0 = []
77
78     nedboyninger = []
79     lengde = []
80     grense = []
81     tap_forspenning = []
82     t0_verdier = []
83     sigma_ovre_graf = []
84     sigma_nedre_graf = []
85     overforing_max = []
86
87     Dekker = [{"hoyde": 200, "zc": 98.3974, "g1": 2.6,
88               "ac": 119959.4297, "iy": 0.62*10**9, "kabler": 7, "iy_påstop": 2680439606},
89
90               {"hoyde": 220, "zc": 102.9530, "g1": 3.1,
91               "ac": 146379.2960, "iy": 0.88*10**9, "kabler": 7, "iy_påstop": 3644589882},
92
93               {"hoyde": 265, "zc": 132.5216, "g1": 3.7,
94               "ac": 172397.9131, "iy": 1.53*10**9, "kabler": 6, "iy_påstop": 4976569252},
95
96               {"hoyde": 285, "zc": 135.5603, "g1": 4.2,
97               "ac": 198735.2650, "iy": 1.98*10**9, "kabler": 6, "iy_påstop": 6287956702 },
98
99               {"hoyde": 320, "zc": 157.2759, "g1": 4.2,
100              "ac": 188740.8729, "iy": 2.45*10**9, "kabler": 6, "iy_påstop": 7132684312 },
101
102              {"hoyde": 340, "zc": 159.5386, "g1": 4.7,
103              "ac": 212923.0742, "iy": 3.05*10**9, "kabler": 6, "iy_påstop": 8753722096},
104
105              {"hoyde": 400, "zc": 206.9283, "g1": 5,
106              "ac": 215624.6757, "iy": 4.4*10**9, "kabler": 5, "iy_påstop": 12166405021},
107
108              {"hoyde": 420, "zc": 202.79, "g1": 5.5,
109              "ac": 274364, "iy": 5.92*10**9, "kabler": 5, "iy_påstop": 14046948900},
110
111              {"hoyde": 500, "zc": 236.7968, "g1": 6.8,
112              "ac": 305202.6371, "iy": 9.03*10**9, "kabler": 5, "iy_påstop": 20272618449},
113
114              {"hoyde": 520, "zc": 244.5994, "g1": 7.3,
115              "ac": 311094.0677, "iy": 10.29*10**9, "kabler": 6, "iy_påstop": 22461619872}]
116
117     def finnerP(t0,L,dekke):
118         P = []
119         P_prover = []
120
121         for i in range(1, 10000, 50):
122             P_forsok = i * 1000
123
124             f_ck = 45
125             f_cm = 53
126             f_pk = 1700
127             f_p0lk = 1550

```

```

128 g1_1 = 1.2 * Dekker[dekke]["g1"]
129 g1_2 = 1.2 * 0.13 * 27
130 g2 = 3
131 E_cm = 36000
132 E_p = 195000
133
134 # Tverrsnittsvariabler
135 h = Dekker[dekke]["hoyde"]
136 zc = Dekker[dekke]["zc"]
137 Ac = Dekker[dekke]["ac"]
138 i = Dekker[dekke]["iy"]
139 antall_kabler = Dekker[dekke]["kabler"]
140
141 # Mulige armeringstyper
142 armering = [5.2, 6.5, 6.8, 7.0, 7.5, 9, 11, 12.5, 13, 15.2, 16]
143 verdige3 = []
144 verdige7 = []
145 verdige19 = []
146
147 # Min. armering
148 As_min = P_forsok / (0.95*f_p01k)
149
150 dmin_3 = (np.sqrt((As_min/antall_kabler)*(1/3)*(4/np.pi)))
151 dmin_7 = (np.sqrt((As_min/antall_kabler)*(1/7)*(4/np.pi)))
152 dmin_19 = (np.sqrt((As_min/antall_kabler)*(1/19)*(4/np.pi)))
153
154 # Bestemmer armeringsdiameter
155 for x in range(len(armering)):
156     storrelse = armering[x]
157
158     if storrelse >= dmin_3:
159         verdige3.append(storrelse)
160
161     if storrelse >= dmin_7:
162         verdige7.append(storrelse)
163
164     if storrelse >= dmin_19:
165         verdige19.append(storrelse)
166
167 #3
168 if verdige3 != []:
169     As_del_3 = 3*(np.pi/4)*(verdige3[0]**verdige3[0])
170     A_p_3 = antall_kabler * As_del_3
171
172 else:
173     As_del_3 = 3*(np.pi/4)*(100**100)
174     A_p_3 = antall_kabler * As_del_3
175
176 #7
177 As_del_7 = 7*(np.pi/4)*(verdige7[0]**verdige7[0])
178 A_p_7 = antall_kabler * As_del_7
179
180 #19
181 As_del_19 = 19*(np.pi/4)*(verdige19[0]**verdige19[0])
182 A_p_19 = antall_kabler * As_del_19
183
184 As_del = min(As_del_3, As_del_7, As_del_19)
185 A_p = antall_kabler * As_del
186
187 if As_del == As_del_3:
188     vaere = 3
189     #cmin_b = 1.5 * verdige3[0] * 1.866
190     cmin_b = max(35, 1.5 * verdige3[0] * 1.866)
191
192     e_p = zc - cmin_b - ((1.5*verdige3[0])/2)
193
194 if As_del == As_del_7:
195     vaere = 7
196     #cmin_b = 1.5 * verdige7[0] * 3
197     cmin_b = max(35, 1.5 * verdige7[0] * 3)
198
199     e_p = zc - cmin_b - ((3*verdige7[0])/2)
200
201 if As_del == As_del_19:
202     vaere = 19
203     #cmin_b = 1.5 * verdige19[0] * 5
204     cmin_b = max(35, 1.5 * verdige19[0] * 5)
205
206     e_p = zc - cmin_b - ((5*verdige19[0])/2)
207
208 #Tap pga l s
209 if P_forsok == 0:
210     P_jekk = 1000
211 else:
212     P_jekk = P_forsok
213
214 delta_L_1000 = 5
215 delta_epsilon_1000 = delta_L_1000 / (L*1000)
216 epsilon_P_jekk = ((P_jekk) / (E_p * A_p))
217 delta_P_1000 = (delta_epsilon_1000/epsilon_P_jekk) * P_jekk
218 tap_1000_prosent = (delta_P_1000/P_jekk) * 100
219
220 #Tap friksjon
221 my_frik = 0.19
222 k_frik = 0.005
223 theta_A = (2*e_p)/((L*1000)/2)
224 theta_B = (2*e_p)/((L*1000)/2)
225 theta_midt = 0
226 theta_L = theta_A + theta_B
227
228 delta_P_friksjon = (P_forsok)*(1 - np.exp(-my_frik*(theta_L+k_frik*((L*1000)*10**(-3)))))
229 friksjonstap_prosent = (delta_P_friksjon/P_forsok)*100
230
231 P_forsok_2 = P_forsok - delta_P_friksjon - delta_P_1000
232 tap_friksjon_prosent = (delta_P_friksjon/P_forsok)*100
233
234 # Transformert tverrsnitt for beregninger av laster i tverrsnitt ved overføring (SLS)
235 B_cc_t0 = np.e**((0.2*(1-(28/t0)**0.5))
236 f_cm_t0 = B_cc_t0 * f_cm
237 E_cm_t0 = (f_cm_t0 / f_cm)**0.3 * E_cm
238 n = E_p / E_cm_t0
239 A_t = Ac + (n-1)*A_p
240 yt = (n-1)*A_p*(e_p/A_t)
241 It = i + Ac*yt**2 + (n-1)*A_p*(e_p-yt)**2
242
243 # Laster p  tverrsnitt og maks spenning ved overforing
244 sigma_max = 0.7 * (f_cm_t0 - 8)
245 M_egen = (g1_1 * L**2) / 8 * 10**6
246 N = -P_forsok_2
247 M = N * (e_p-yt)
248 M_e = N * (-yt)
249
250 # Lastfordelinger i tverrsnitt ved overforing
251
252 # Nedre ende

```

```

256     sigma_ende_overforing = (N/A_t) + ((M_e-N*yt)/It)*(zc-yt)
257
258     # ovre midt
259     sigma_ovre_overforing = (N/A_t) + (((M+M_egen)-N*yt)/It)*(-(h-zc)-yt)
260
261     # Nedre midt
262     sigma_nedre_overforing = (N/A_t) + (((M+M_egen)-N*yt)/It)*(zc-yt)
263
264     if np.abs(sigma_nedre_overforing) < sigma_max:
265         P.append(P_forsok_2)
266         P_prover.append(P_forsok)
267     elif P == []:
268         return [0], [0]
269     else:
270         return P, P_prover
271     break
272
273
274 def Nedboyning(L,dekke, p_test):
275
276     #P0 = finnerP(t0,L,dekke)[1][p_test]
277     P0 = max(1, finnerP(t0,L,dekke)[1][p_test])
278
279     f_ck = 45
280     f_cm = 53
281     f_pk = 1700
282     f_p0lk = 1550
283     g1_1 = 1.2 * Dekker[dekke]["g1"]
284     g1_2 = 1.2 * 0.13 * 27
285     g1_2_mindre_p0stop = 1.2 * 0.05 * 27
286     g2 = 3
287     E_cm = 36000
288     E_p = 195000
289
290     # Tverrsnittsvariabler
291     h = Dekker[dekke]["hoyde"]
292     zc = Dekker[dekke]["zc"]
293     Ac = Dekker[dekke]["ac"]
294     i = Dekker[dekke]["iy"]
295     antall_kabler = Dekker[dekke]["kabler"]
296
297     # Mulige armeringstyper
298     armering = [5.2, 6.5, 6.8, 7.0, 7.5, 9, 11, 12.5, 13, 15.2, 16]
299     verdige3 = []
300     verdige7 = []
301     verdige19 = []
302
303     # Min. armering
304     As_min = P0/(0.95*f_p0lk)
305
306     dmin_3 = (np.sqrt((As_min/antall_kabler)*(1/3)*(4/np.pi)))
307     dmin_7 = (np.sqrt((As_min/antall_kabler)*(1/7)*(4/np.pi)))
308     dmin_19 = (np.sqrt((As_min/antall_kabler)*(1/19)*(4/np.pi)))
309
310     # Bestemmer armeringsdiameter
311     for x in range(len(armering)):
312         storrelse = armering[x]
313
314         if storrelse >= dmin_3:
315             verdige3.append(storrelse)
316
317
318         if storrelse >= dmin_7:
319             verdige7.append(storrelse)
320
321
322         if storrelse >= dmin_19:
323             verdige19.append(storrelse)
324
325
326     #3
327     if verdige3 != []:
328         As_del_3 = 3*(np.pi/4)*(verdige3[0]*verdige3[0])
329         A_p_3 = antall_kabler * As_del_3
330
331     else:
332         As_del_3 = 3*(np.pi/4)*(100*100)
333         A_p_3 = antall_kabler * As_del_3
334
335     #7
336     As_del_7 = 7*(np.pi/4)*(verdige7[0]*verdige7[0])
337     A_p_7 = antall_kabler * As_del_7
338
339     #19
340     As_del_19 = 19*(np.pi/4)*(verdige19[0]*verdige19[0])
341     A_p_19 = antall_kabler * As_del_19
342
343     As_del = min(As_del_3, As_del_7, As_del_19)
344     A_p = antall_kabler * As_del
345
346     if As_del == As_del_3:
347         vaiere = 3
348         #cmin_b = 1.5 * verdige3[0] * 1.5
349         cmin_b = max(35, 1.5 * verdige3[0] * 1.5)
350
351         e_p = zc - cmin_b - ((1.5*verdige3[0])/2)
352         diameter = verdige3[0]
353
354         REI = brannkrav(dekke, e_p, antall_kabler, L, verdige3[0], cmin_b, vaiere)[0]
355         REI_justert = brannkrav(dekke, e_p, antall_kabler, L, verdige3[0], cmin_b, vaiere)[1]
356         moment = brannkrav(dekke, e_p, antall_kabler, L, verdige3[0], cmin_b, vaiere)[2]
357
358     if As_del == As_del_7:
359         vaiere = 7
360         #cmin_b = 1.5 * verdige7[0] * 3
361         cmin_b = max(35, 1.5 * verdige7[0] * 3)
362
363         e_p = zc - cmin_b - ((3*verdige7[0])/2)
364         diameter = verdige7[0]
365
366         REI = brannkrav(dekke, e_p, antall_kabler, L, verdige7[0], cmin_b, vaiere)[0]
367         REI_justert = brannkrav(dekke, e_p, antall_kabler, L, verdige7[0], cmin_b, vaiere)[1]
368         moment = brannkrav(dekke, e_p, antall_kabler, L, verdige7[0], cmin_b, vaiere)[2]
369
370     if As_del == As_del_19:
371         vaiere = 19
372         #cmin_b = 1.5 * verdige19[0] * 5
373         cmin_b = max(35, 1.5 * verdige19[0] * 5)
374
375         e_p = zc - cmin_b - ((5*verdige19[0])/2)
376         diameter = verdige19[0]
377
378         REI = brannkrav(dekke, e_p, antall_kabler, L, verdige19[0], cmin_b, vaiere)[0]
379         REI_justert = brannkrav(dekke, e_p, antall_kabler, L, verdige19[0], cmin_b, vaiere)[1]
380         moment = brannkrav(dekke, e_p, antall_kabler, L, verdige19[0], cmin_b, vaiere)[2]
381
382     resonans = vibrasjon(L, dekke, E_cm, g1_1, g1_2, g1_2_mindre_p0stop)
383
384

```

```

384
385 #Tap pga lås
386 if P0 == 0:
387     P_jekk = 1
388 else:
389     P_jekk = P0
390 delta_L_10s = 5
391 delta_epsilon_10s = delta_L_10s / (L*1000)
392 epsilon_P_jekk = (P_jekk / (E_p * A_p))
393 delta_P_10s = (delta_epsilon_10s/epsilon_P_jekk) * P_jekk
394 tap_10s_prosent = (delta_P_10s/P_jekk) * 100
395
396 #Tap friksjon
397 my_frik = 0.19
398 k_frik = 0.005
399 theta_A = (2*e_p)/((L*1000)/2)
400 theta_B = (2*e_p)/((L*1000)/2)
401 theta_midt = 0
402 theta_L = theta_A + theta_B
403
404 delta_P_friksjon = P0*(1 - np.exp(-my_frik*(theta_L+k_frik*((L*1000)*10**(-3)))))
405 friksjonstap_prosent = (delta_P_friksjon/P0)*100
406
407 P = P0 - delta_P_friksjon - delta_P_10s
408 tap_friksjon_prosent = (delta_P_friksjon/P0)*100
409
410 # Transformert tverrsnitt for beregninger av laster i tverrsnitt ved overføring (SLS)
411 B_cc_t0 = np.e**(0.2*(1-(28/t0)**0.5))
412 f_cm_t0 = B_cc_t0 * f_cm
413 E_cm_t0 = (f_cm_t0 / f_cm)**0.3 * E_cm
414 n = E_p / E_cm_t0
415 A_t = A_c + (n-1)*A_p
416 yt = (n-1)*A_p*(e_p/A_t)
417 It = I + Ac*yt**2 + (n-1)*A_p*(e_p-yt)**2
418
419 # Laster på tverrsnitt og maks spenning ved overføring
420 sigma_max = 0.7 * (f_cm_t0 - 8)
421 M_egen = (g1_1 * L**2) / 8 * 10**6
422 N = -P
423 M = N * (e_p-yt)
424 M_e = N * (-yt)
425
426 # Lastfordelinger i tverrsnitt ved overføring
427 sigma_ende_overforing = (N/A_t) + ((M_e-N*yt)/It)*(zc-yt)
428
429 # ovre midt
430 sigma_ovre_overforing = (N/A_t)+(((M+M_egen)-N*yt)/It)*(-(h-zc)-yt)
431
432 # Nedre midt
433 sigma_nedre_overforing = (N/A_t)+(((M+M_egen)-N*yt)/It)*(zc-yt)
434
435 # Langtidsvirkninger
436
437 # Effektiv E-modul
438
439 alfa_1 = (35/f_cm)**0.7
440 alfa_2 = (35/f_cm)**0.2
441 alfa_3 = (35/f_cm)**0.5
442
443 u = 1200
444 h_0 = (2*Ac)/u
445 RH = 40
446
447 def B_H(h_0, alfa_3, RH):
448     if True:
449         B_H = 1.5 * (1 + (0.012*RH)**18) * h_0 + 250 * alfa_3
450         if B_H <= 1500 * alfa_3:
451             return B_H
452         else:
453             return 1500 * alfa_3
454
455 B_H = B_H(h_0, alfa_3, RH)
456
457 Bc_t_t0 = 1
458 B_fcm = 16.8/np.sqrt(f_cm)
459 B_3 = 1 / (1 + t0**0.2)
460 B_28 = 1 / (1 + 28**0.2)
461 B_180 = 1 / (1 + 180**0.2)
462
463 phi_RH = (1 + ((1-(RH/100)) / (0.1*h_0**(1/3)))) * alfa_1 * alfa_2
464
465 # phi(t)
466 phi_0_3 = phi_RH * B_fcm * B_3
467 phi_0_28 = phi_RH * B_fcm * B_28
468 phi_0_180 = phi_RH * B_fcm * B_180
469
470 # Kryptall
471 phi_3 = phi_0_3
472 phi_28 = phi_0_28
473 phi_180 = phi_0_180
474
475 # Effektiv E-modul
476 E_cl3 = E_cm / (1 + phi_3)
477 E_cl28 = E_cm / (1 + phi_28)
478 E_cl180 = E_cm / (1 + phi_180)
479
480 # Laster
481 M_p = - P * e_p
482 M_g1_1 = ((g1_1 * L**2) / 8) * 10**6
483 M_g1_2 = ((g1_2 * L**2) / 8) * 10**6
484 M_g2_perm = ((0.5 * g2 * L**2) / 8) * 10**6
485 M_g2_var = ((0.5 * g2 * L**2) / 8) * 10**6
486
487 # For spenninger i tverrsnittet etter lang tid
488 M_laster_langtid = M_g1_1 + M_g1_2 + M_g2_perm
489
490 # Gjennomsnittlig E-modul
491 over_br_ek = ((np.abs(M_p)) + (np.abs(M_g1_1)) + (np.abs(M_g1_2)) + (np.abs(M_g2_perm)) + (np.abs(M_g2_var)))
492 under_br_ek = ( ((np.abs(M_p))/(E_cl3)) + ((np.abs(M_g1_1))/(E_cl3)) + ((np.abs(M_g1_2))/(E_cl28)) +
493               ((np.abs(M_g2_perm))/(E_cl180)) + ((np.abs(M_g2_var))/(E_cm)) )
494 E_middel = over_br_ek / under_br_ek
495
496 # Justert transformert tverrsnitt mhp kryp
497 kryp_n = E_p / E_middel
498 kryp_A_t = A_c + (kryp_n-1)*A_p
499 kryp_yt = (kryp_n-1)*A_p*(e_p/kryp_A_t)
500 kryp_It = I + Ac*kryp_yt**2 + (kryp_n-1)*A_p*(e_p-kryp_yt)**2
501
502 # Justerte laster
503 M_kryp = N * (e_p-kryp_yt)
504 M_e_kryp = N * (-kryp_yt)
505
506 # Spenningsfordeling mhp kryp, langtid
507
508 # Nedre ende
509 sigma_ende_kryp = (N/kryp_A_t) + ((M_e_kryp-N*kryp_yt)/kryp_It)*(zc-kryp_yt)
510 #print('sigma_nedre_ende_kryp =',round(sigma_ende_kryp,3),'MPa')
511
512 # ovre midt

```

```

512 sigma_ovre_kryp = (N/kryp_A_t)+(((M_kryp+M_laster_langtid)-N*kryp_yt)/kryp_it)^(-(h-zc)-kryp_yt)
513 #print('sigma_ovre_midt_kryp =',round(sigma_ovre_kryp,3),'MPa')
514
515 # Nedre midt
516 sigma_nedre_kryp = (N/kryp_A_t)+(((M_kryp+M_laster_langtid)-N*kryp_yt)/kryp_it)*(zc-kryp_yt)
517 #print('sigma_nedre_midt_kryp =',round(sigma_nedre_kryp,3),'MPa')
518
519 # Armering
520 sigma_Pk = (N/A_t)+(((M_kryp+M_laster_langtid)-N*yt)/It)*(e_p-yt)
521 sigma_PL = (N/kryp_A_t)+(((M_kryp+M_laster_langtid)-N*kryp_yt)/kryp_it)*(e_p-kryp_yt)
522 epsilon_k = sigma_Pk / E_cm
523 epsilon_L = sigma_PL / E_middel
524 delta_sigma_P_kryp = (epsilon_L - epsilon_k) * E_p
525 delta_P_kryp = delta_sigma_P_kryp * A_p
526 tap_kryp_prosent = -(delta_P_kryp/P0)*100
527
528 # Svinn - Antar sementklasse R
529 alfa_ds1 = 6
530 alfa_ds2 = 0.11
531 f_cm0 = 10
532 RH_0 = 100
533 B_RH = 1.55 * (1-(RH/RH_0)**3)
534
535 epsilon_cd0 = 0.85 * ( (220+110*alfa_ds1)*np.e**(-alfa_ds2*(f_cm/f_cm0)) ) * 10**(-6) * B_RH
536
537 k_h = 0.717
538 B_ds_t_ts = 1
539 epsilon_cd_t = B_ds_t_ts * k_h * epsilon_cd0
540 epsilon_ca_t = 2.5 * (f_ck - 10) * 10**(-6)
541 epsilon_cs_t = - (epsilon_cd_t + epsilon_ca_t)
542
543 # Krefter grunnt svinn
544 N_s = - epsilon_cs_t * E_p * A_p
545 M_s = N_s * (e_p - kryp_yt)
546
547 # Endring av spenning i tverrsnittet grunnet svinn
548
549 # ovre midt
550 delta_sigma_ovre_midt_svin = (N_s / (kryp_A_t)) + (M_s / (kryp_it))*(-(h-zc)-kryp_yt)
551
552 # Nedre Midt
553 delta_sigma_nedre_midt_svin = (N_s / (kryp_A_t)) + (M_s / (kryp_it))*(zc-kryp_yt)
554
555 # Armering
556 epsilon_c_s_e = epsilon_cs_t + N_s/(E_middel * kryp_A_t) + M_s/(E_middel * kryp_it)*(e_p-kryp_yt)
557 delta_sigma_P_s = epsilon_c_s_e * E_p
558 delta_P_svin = delta_sigma_P_s * A_p
559 tap_svin_prosent = -(delta_P_svin / P0)*100
560
561 # Relaksasjon
562 rho_1000 = 2.5
563 sigma_pm0 = 1275
564 my = sigma_pm0 / f_pk
565 t = 500000
566 delta_sigma_pr = 0.8 * (sigma_pm0 * 0.66 * rho_1000 * np.e**((9.1*my) * (t/1000)**(0.75*(1-my))) * 10**(-5))
567
568 # Laster pga relaksasjon
569 N_r = delta_sigma_pr * A_p
570 M_r = N_r * (e_p - kryp_yt)
571
572 # Endring av spenning i tverrsnittet grunnet relaksasjon
573
574 # ovre midt
575 delta_sigma_ovre_midt_relaksasjon = (N_r / (kryp_A_t)) + (M_r / (kryp_it))*(-(h-zc)-kryp_yt)
576
577 # Nedre midt
578 delta_sigma_nedre_midt_relaksasjon = (N_r / (kryp_A_t)) + (M_r / (kryp_it))*(zc-kryp_yt)
579
580 # Armering
581 delta_sigma_armering = -delta_sigma_pr
582 delta_P_relaksasjon = delta_sigma_armering * A_p
583 if P0 == 0:
584     P0 = 1
585     tap_relak_prosent = -(delta_P_relaksasjon / P0 ) * 100
586 else:
587     tap_relak_prosent = -(delta_P_relaksasjon / P0 ) * 100
588
589 # Endelig spenning i tverrsnittet etter lang tid
590
591 # ovre midt
592 sigma_endelig_ovre = sigma_ovre_kryp + delta_sigma_ovre_midt_svin + delta_sigma_ovre_midt_relaksasjon
593
594 # Nedre midt
595 sigma_endelig_nedre = sigma_nedre_kryp + delta_sigma_nedre_midt_svin + delta_sigma_nedre_midt_relaksasjon
596
597 # Armering
598 P_endelig = P + delta_P_kryp + delta_P_svin + delta_P_relaksasjon
599
600
601 # Nedboyninger i tverrsnittet etter lang tid
602 qed = 1.2*(gl_1 + gl_2) + 1.5*(g2)
603 q_forspenning = (8 * P_endelig * e_p) / (L*10**3)**2
604
605 #Beregning av nedboyning
606 def nedboyning(p):
607     return (5*p*(L*10**3)**4)/(384*E_middel*It)
608
609 #Total nedboyning, total tap
610 tot_nedboyning = nedboyning(qed) - nedboyning(q_forspenning)
611 total_tap_P = (1 - P_endelig/P0) * 100
612
613 E_dyn = 1.175 * E_cm
614 i_p_stopt = 11.1 * 10**9
615 m_p_stopt = ((gl_1+gl_2)*1000)/9.81
616 fL = (np.pi/(2*L**2)) * (np.sqrt((E_dyn*10**6*i_p_stopt*10**(-12))/m_p_stopt))
617
618 t0_verdier.append(t0)
619 tap_forspenning.append(total_tap_P)
620 sigma_ovre_graf.append(sigma_endelig_ovre)
621 sigma_nedre_graf.append(sigma_endelig_nedre)
622
623 co2 = kg_co2(dekke, A_p)
624
625 return(tot_nedboyning, total_tap_P, P0, tap_friksjon_prosent, tap_svin_prosent, As_min, A_p, tap_kryp_prosent,
626        tap_relak_prosent, tap_Lf_prosent, sigma_ovre_overforing, sigma_nedre_overforing, f_cm_t0,REI, REI_justert,
627        resonans, co2, sigma_endelig_nedre, sigma_endelig_ovre, diameter, vaiere)
628
629 def kg_co2(dekke, A_p):
630     kubikk_sement = (Dekker[dekke]["ac"] / 10**6) * 0.135 #13.5% sement i betongen
631     tonn_sement = kubikk_sement * 1.44 #1.44 tonn sement per kubikk
632     tonn_co2_sement = tonn_sement * 0.8 #0.8 tonn co2 per tonn sement produsert
633
634     kubikk_armering = (A_p / 10**6)
635     tonn_armering = kubikk_armering * 7.85 #7.85 tonn per kubikk
636     tonn_co2_armering = tonn_armering * 5.64 #5.64 tonn co2 per tonn armering produsert
637
638     total_co2 = (tonn_co2_sement + tonn_co2_armering) * 1000 #kilo
639
640     return total_co2

```

```

640     return total_Coz
641
642 def brannkrav(dekke, e_p, antall_kabler, L, diameter, cmin_b, vaiere):
643     z1 = (Dekker[dekke]["hoyde"] - Dekker[dekke]["zc"]) / 2
644     z2 = e_p
645     z = z1 + z2
646     r = diameter / 2
647
648     g1_1 = 1.2 * Dekker[dekke]["g1"]
649     g1_2 = 1.2 * 0.13 * 27
650     g2 = 3
651     qed = 1.2 * g1_1 + 1.2 * g1_2 + 1.5 * g2
652
653     motstand = [{"rei": 30, "hekv": 60, "dybde": 25},
654                 {"rei": 60, "hekv": 80, "dybde": 35},
655                 {"rei": 90, "hekv": 100, "dybde": 45},
656                 {"rei": 120, "hekv": 120, "dybde": 55},
657                 {"rei": 180, "hekv": 150, "dybde": 70},
658                 {"rei": 240, "hekv": 175, "dybde": 80}]
659
660     motstand_justert = [{"rei": 30, "hekv": 60, "dybde": 0},
661                        {"rei": 60, "hekv": 80, "dybde": 0},
662                        {"rei": 90, "hekv": 100, "dybde": 0},
663                        {"rei": 120, "hekv": 120, "dybde": 0},
664                        {"rei": 180, "hekv": 150, "dybde": 0},
665                        {"rei": 240, "hekv": 175, "dybde": 0}]
666
667     Pp = (vaiere*np.pi*r**2) * (1550 / 1.15)
668     Sd = antall_kabler * Pp
669     Md = Sd * (z/1000)
670     Med = qed * (L**2 / 8)
671     theta0 = 350
672     h_ekv = Dekker[dekke]["ac"] / 1200
673     moment = ""
674
675     a_dybde = cmin_b
676
677     if Med <= Md:
678         moment = "ok"
679     else:
680         moment = "ikke ok"
681
682     rei = [0]
683     rei_ = [0]
684
685     for minutt in range(0,6):
686         if h_ekv >= motstand[minutt]["hekv"] and a_dybde >= motstand[minutt]["dybde"]:
687             rei.append(motstand[minutt]["rei"])
688
689     ### JUSTERT ARMERINGSDYBDE
690     qed_fi = g1_1 + g1_2 + (0.3 * g2)
691     Med_fi = qed_fi * (L**2 / 8)
692     fd_fi = 1550
693     Md_fi = (antall_kabler * fd_fi * (vaiere*np.pi*r**2) * (z/1000)) / 1000
694
695     my_fi = Med_fi / Md_fi
696
697     def myGraf(my_fi):
698         return -140.625 * my_fi**2 - 300 * my_fi + 590.625
699
700     if my_fi >= 0.2 and my_fi <= 1:
701         thetakrit = myGraf(my_fi)
702         justering = (theta0 - thetakrit) / 10
703
704
705     for minutt in range(0,6):
706         motstand_justert[minutt]["dybde"] = motstand[minutt]["dybde"] + justering
707
708         if h_ekv >= motstand_justert[minutt]["hekv"] and a_dybde >= motstand_justert[minutt]["dybde"]:
709             rei_.append(motstand[minutt]["rei"])
710
711     return rei[-1], rei_[-1], moment
712
713 def vibrasjon(L, dekke, Ecm, g1_1, g1_2, g1_2_mindre_p[stop]):
714     g = 9.81
715     vekt = (1000 * (g1_1 + g1_2)) / g
716     Edyn = 1.175 * Ecm * 10 **6
717     fl_hd = (np.pi / (2*L**2)) * np.sqrt((Edyn*Dekker[dekke]["iy_pastop"] * 10**(-12))/(vekt))
718     return fl_hd
719
720
721
722
723 for x in range(50,250):
724     print(str(round((x/250)*100)) + "%")
725     for k in range(0,10):
726
727         dekke = k
728
729         for p_test in range(len(finnerP(t0, x/10, dekke)[1])):
730
731             if Nedboying(x/10, dekke, p_test)[0] <= (x*100)/250 and Nedboying(x/10, dekke, p_test)[13] >= 0:
732
733
734
735                 if Nedboying(x/10, dekke, p_test)[0] >= 0.5:
736                     if Nedboying(x/10, dekke, p_test)[17] <= 2.7 and Nedboying(x/10, dekke, p_test)[18] >= -45:
737                         boying = Nedboying(x/10, dekke, p_test)[0]
738                         nedboyingner.append(boying)
739
740                         plotteGrenser[k]["nm"].append(boying)
741                         plotteGrenser[k]["m"].append(x/10)
742
743                         #KODE FOR TAP MOT P
744                         plotteGrenser[k]["P0"].append(Nedboying(x/10, dekke)[2]/1000)
745
746                         #KODE FOR TAP MOT L
747                         plotteGrenser[k]["Tap"].append(Nedboying(x/10, dekke, p_test)[1])
748                         yy.append(Nedboying(x/10, dekke, p_test)[1])
749                         xx.append(x/10)
750
751                         #KODE FOR FRIKSJON MOT L
752                         plotteGrenser[k]["frikatap"].append(Nedboying(x/10, dekke)[3])
753                         yy_frik.append(Nedboying(x/10, dekke)[3])
754
755                         #KODE FOR SVINN MOT L
756                         plotteGrenser[k]["svinntap"].append(Nedboying(x/10, dekke)[4])
757                         yy_svinn.append(Nedboying(x/10, dekke)[4])
758
759                         #KODE FOR MIN ARMERING MOT L
760                         plotteGrenser[k]["armering"].append(Nedboying(x/10, dekke)[5])
761                         plotteGrenser[k]["Ap"].append(Nedboying(x/10, dekke)[6])
762
763                         #KODE FOR KRYP MOT L
764                         plotteGrenser[k]["kryp"].append(Nedboying(x/10, dekke)[7])
765                         yy_kryp.append(Nedboying(x/10, dekke)[7])
766
767                         #KODE FOR RELAKSASJON MOT L
768                         plotteGrenser[k]["relaks"].append(Nedboying(x/10, dekke)[8])

```

```

768     regnbue.append(Nedboyinging(x/10, dekke)[9])
769     yy_relak.append(Nedboyinging(x/10, dekke)[8])
770
771     #KODE FOR LÅSETAP MOT L
772     plotteGrenser[k]["låsetap"].append(Nedboyinging(x/10, dekke)[9])
773     yy_låsetap.append(Nedboyinging(x/10, dekke)[9])
774
775     #KODE FOR SPENNING MOT L ØVRE
776     plotteGrenser[k]["ovre"].append(Nedboyinging(x/10, dekke)[10])
777
778     #KODE FR SPENNING MOT L NEDRE
779     plotteGrenser[k]["nedre"].append(Nedboyinging(x/10, dekke)[11])
780     plotteGrenser[k]["grense_øverforing"].append(Nedboyinging(x/10, dekke)[12])
781
782     lengde.append(x/10)
783     overforing_max.append(-(0.7 * (Nedboyinging(x/10, dekke)[12] - 8)))
784
785     plotteGrenser[k]["REI"].append(Nedboyinging(x/10, dekke, p_test)[13])
786     if Nedboyinging(x/10, dekke, p_test)[14] >= Nedboyinging(x/10, dekke, p_test)[13]: #!= 0:
787         plotteGrenser[k]["REI"].append(Nedboyinging(x/10, dekke, p_test)[14])
788         plotteGrenser[k]["m_"].append(x/10)
789
790     plotteGrenser[k]["vibrasjon"].append(Nedboyinging(x/10, dekke)[15])
791     yy_vibrasjon.append(Nedboyinging(x/10, dekke)[15])
792
793     plotteGrenser[k]["co2"].append(Nedboyinging(x/10, dekke, p_test)[16])
794     plotteGrenser[k]["vaiere"].append(Nedboyinging(x/10, dekke, p_test)[20])
795     plotteGrenser[k]["armering"].append(Nedboyinging(x/10, dekke, p_test)[19])
796     break
797
798 """
799 plt.figure(dpi=1200)
800
801 #PLOTTE KULT REGNBUE TRE
802 gs = gridspec.GridSpec(1, 2, width_ratios=[2, 1])
803 #fig = plt.figure(figsize=(80, 60))
804 plt.subplot(gs[0])
805
806 farger = ["red", "green", "blue", "black", "magenta", "orange", "limegreen", "cyan", "saddlebrown", "pink"]
807 #plt.figure(dpi=1200)
808
809 for dekke in range(0,10):
810     if plotteGrenser[dekke]["m"] != []:
811         plt.plot(plotteGrenser[dekke]["m"], plotteGrenser[dekke]["co2"], color = farger[dekke]#, label = "HD" + str(Dekker[dekke]["hoyde"]) + " - " + str(round(plotteGrenser[dekke]["co2"][-1] + 0.05, plotteGrenser[dekke]["co2"][-1]-0.2, "HD" + str(Dekker[dekke]["hoyde"])), fontsize = 6)
812         #plt.text(plotteGrenser[dekke]["m"][-1] + 0.05, plotteGrenser[dekke]["co2"][-1]-0.2, "HD" + str(Dekker[dekke]["hoyde"])), fontsize = 6)
813         #plt.text(plotteGrenser[dekke]["t0"][-1] + 0.5, plotteGrenser[dekke]["co2"][-1]-0.2, str(plotteGrenser[dekke]["vaiere"][-1]) + "x" + str(plotteGrenser[dekke]["m"][-1] + 0.5),
814         #plt.text(plotteGrenser[dekke]["t0"][0] -0.5, plotteGrenser[dekke]["co2"][0]-0.2, str(plotteGrenser[dekke]["vaiere"][0]) + "x" + str(plotteGrenser[dekke]["armering"][-1] + 0.5),
815
816 plt.grid()
817 plt.xlabel("Lengde på hulldekke [meter]")
818 plt.ylabel("CO2-utslipp ved produksjon [kg-CO2-ekv]")
819 plt.title("CO2-utslipp avhengig av L ved t0 = " + str(t0))
820 plt.xticks(np.arange(5, 27, 5))
821 plt.yticks(np.arange(0, 350, 50))
822 plt.ylim(0,350)
823 plt.xlim(5,27)
824
825 #plt.figure(dpi=1200)
826
827 plt.subplot(gs[1])
828 #Dekker = ["HD200", "HD220", "HD265", "HD285", "HD320", "HD340", "HD400", "HD420", "HD500", "HD520"]
829 farger = ["red", "green", "blue", "black", "magenta", "orange", "limegreen", "cyan", "saddlebrown", "pink"]
830 for dekke in range(0,10):
831     strek = [1,1]
832     hoyde = [(20.1-dekke*2), (21.3-dekke*2)]
833
834     plt.plot(strek, hoyde, color=farger[dekke])
835     plt.text(4, (20.8-dekke*2), ("HD" + str(Dekker[dekke]["hoyde"]) + " - Ap: " + str(Dekker[dekke]["kabler"])+ "x" + str(plotteGrenser[dekke]["vaiere"][-1]) + "x" + str(plotteGrenser[dekke]["m"][-1] + 0.5),
836     plt.text(4, (20.2-dekke*2), (str(round(plotteGrenser[dekke]["co2"][-1], 2)) + " kg CO2-ekv"), fontsize = 6)
837     plt.text(-0.4, (20.5-dekke*2), str(plotteGrenser[dekke]["m"][-1]) + "m", fontsize =7, horizontalalignment='right')
838
839 plt.axis("off")
840 plt.title("Lengste dekkelengde:")
841 plt.xlim(0,50)
842 plt.ylim(0,21.5)
843 plt.tight_layout()
844
845 plt.show()
846 """
847
848
849 farger = ["red", "green", "blue", "black", "magenta", "orange", "limegreen", "cyan", "saddlebrown", "pink"]
850
851 regnbue = [{"farger": [], "co2": [], "m": []}]
852
853 matrise = [{"t0": t0, "dekke": 200, "lengder": [], "co2": [], "farge": "red"},
854 {"t0": t0, "dekke": 220, "lengder": [], "co2": [], "farge": "green"},
855 {"t0": t0, "dekke": 265, "lengder": [], "co2": [], "farge": "blue"},
856 {"t0": t0, "dekke": 285, "lengder": [], "co2": [], "farge": "black"},
857 {"t0": t0, "dekke": 320, "lengder": [], "co2": [], "farge": "magenta"},
858 {"t0": t0, "dekke": 340, "lengder": [], "co2": [], "farge": "orange"},
859 {"t0": t0, "dekke": 400, "lengder": [], "co2": [], "farge": "limegreen"},
860 {"t0": t0, "dekke": 420, "lengder": [], "co2": [], "farge": "cyan"},
861 {"t0": t0, "dekke": 500, "lengder": [], "co2": [], "farge": "saddlebrown"},
862 {"t0": t0, "dekke": 520, "lengder": [], "co2": [], "farge": "pink"}]
863
864 matrise520 = {"t0": t0, "dekke": 520, "lengder": [], "co2": [], "farge": "red"}
865
866 laveste = []
867 laveste_m = []
868 for num in range(len(plotteGrenser[9]["m"])):
869     current = 500
870     current_m = 0
871     current_color = int(0)
872     for dekke in range(0,10):
873
874         if dekke == 9:
875             try:
876                 matrise520["lengder"].append(plotteGrenser[dekke]["m"][num])
877                 matrise520["co2"].append(plotteGrenser[dekke]["co2"][num])
878             except IndexError:
879                 yeet = 0
880
881
882         try:
883             if plotteGrenser[dekke]["co2"][num] < current:
884                 current = plotteGrenser[dekke]["co2"][num]
885                 current_m = plotteGrenser[dekke]["m"][num]
886                 current_color = int(dekke)
887             except IndexError:
888                 current = current
889                 current_m = current_m
890                 current_color = current_color
891         laveste.append(current)
892         laveste_m.append(current_m)
893         #print(num, current_color)
894         regnbue[0]["farger"].append(current_color)
895         regnbue[0]["co2"].append(current)
896         regnbue[0]["m"].append(current_m)

```

```

897     matrise[current_color]["lengder"].append(current_m)
898     matrise[current_color]["co2"].append(current)
899
900 forste = 0
901 siste = 0
902
903 for num in range(len(regnbue[0]["m"])):
904     farge = regnbue[0]["farger"][num]
905     sistefarge = regnbue[0]["farger"][num-1]
906     co2 = regnbue[0]["co2"][num]
907     co2_forrige = regnbue[0]["co2"][num-1]
908     try:
909         co2_neste = regnbue[0]["co2"][num+1]
910     except IndexError:
911         co2_neste = co2
912     m = regnbue[0]["m"][num]
913
914     if co2 > co2_forrige:
915         forste = num
916         #print("no")
917         #print(forste, siste)
918     if co2_neste > co2 or num == len(regnbue[0]["m"]) - 1:
919         siste = num
920         #print("yo")
921         #print(forste,siste)
922
923     xx = [regnbue[0]["m"][forste], regnbue[0]["m"][siste]]
924     yy = [regnbue[0]["co2"][forste], regnbue[0]["co2"][siste]]
925
926
927     #plt.plot(xx,yy, color = farger[farge])
928
929     #plt.plot(regnbue[0]["m"][num], regnbue[0]["co2"][num], "_", markersize = 10, color = farger[farge])
930
931
932     if regnbue[0]["co2"][num] != regnbue[0]["co2"][num-1] and num != 0:
933         x = [regnbue[0]["m"][num-1], regnbue[0]["m"][num]]
934         y = [regnbue[0]["co2"][num-1], regnbue[0]["co2"][num]]
935
936         #plt.plot(x,y, color = farger[sistefarge])
937
938
939     #print(regnbue[0]["farger"])
940     #laveste_m.sort()
941     #laveste.sort()
942     #plt.plot(laveste_m, laveste, color = "black", linestyle="--")
943     #print(laveste_m, laveste)
944
945     #print(regnbue[0]["m"])
946
947     """
948     plt.xlabel("t0 [dager]")
949     plt.ylabel("Nedboyning [mm]")
950     plt.title("Nedboyning avhengig av t0 ved L = " + str(L))
951     """
952     """
953     #plt.axhline(y=(L*4), color='red', linestyle='--', label = "Grense")
954     plt.xlim(5, 26)
955     #plt.ylim(30,250)
956     #plt.ylim(50,170)
957     #plt.legend(loc = "upper left", fontsize = 6)
958     #plt.ylim(-20,100)
959
960     plt.subplot(gs[1])
961
962     for dekke in range(0,10):
963         if plotteGrenser[dekke]["m"] != []:
964
965             strek= [1,1]
966             hoyde =[(20.1-dekke*2), (21.3-dekke*2)]
967
968             plt.plot(strek, hoyde, color=farger[dekke])
969             #plt.text(4, (20.8-dekke*2), ("HD" +str(Dekker[dekke]["hoyde"])), fontsize =6)
970             #plt.text(-0.4, (20.5-dekke*2), str(plotteGrenser[dekke]["m"][-1]) + "m", fontsize =7, horizontalalignment='right')
971             plt.text(4, (20.8-dekke*2), ("HD" +str(Dekker[dekke]["hoyde"]) + " - Ap: " + str(Dekker[dekke]["kabler"])+ "x" + str(plotteGrenser[dekke]["vaiere"][-1]) + "x" +
972             plt.text(4,(20.2-dekke*2), (str(round(plotteGrenser[dekke]["co2"][-1], 2)) + " kg CO2-ekv"), fontsize = 6)
973     plt.axis("off")
974     plt.xlim(0,50)
975     plt.ylim(0,21.5)
976     plt.title("Lengste dekkelengde:")
977
978     plt.tight_layout()
979
980     plt.show()
981     """
982     """
983
984     farger = ["red", "green", "blue", "black", "magenta", "orange", "limegreen", "cyan", "saddlebrown", "pink"]
985
986     #PLOTTE SPENNING MOT L
987     for dekke in range(0,10):
988         plt.plot(plotteGrenser[dekke]["m"], plotteGrenser[dekke]["mm"], color = farger[dekke], label= ("HD" + str(Dekker[dekke]["hoyde"]) + " - " + str(plotteGrenser[dekke]
989         #plt.text(plotteGrenser[dekke]["m"][-1]-0.4, (plotteGrenser[dekke]["m"][-1]*4)+4, "HD" + str(Dekker[dekke]["hoyde"]), fontsize = 5)
990         #plt.text(plotteGrenser[dekke]["m"][-1], -2, str(plotteGrenser[dekke]["m"][-1]), color = farger[dekke], fontsize=5)
991         plt.plot(plotteGrenser[dekke]["m"][-1], plotteGrenser[dekke]["mm"][-1], "x", color=farger[dekke])
992         plt.vlines(plotteGrenser[dekke]["m"][-1], plotteGrenser[dekke]["mm"][-1], (plotteGrenser[dekke]["m"][-1]*4) +2, color =farger[dekke])
993         #plt.plot(plotteGrenser[dekke]["mlang"], plotteGrenser[dekke]["mmiang"], color = farger[dekke])
994         #print(plotteGrenser[9]["m"], plotteGrenser[9]["mm"])
995     def mm(m):
996         return m*(1000/250)
997     m = np.linspace(0, 25, 100)
998
999
1000     plt.plot(m, mm(m), color='r', linestyle='--', label = "Grense")
1001     plt.plot(plotteGrenser[dekke]["m"][-1], plotteGrenser[dekke]["mm"][-1], "x", color=farger[dekke])
1002     #plt.text(plotteGrenser[dekke]["m"][-1] -0.7, plotteGrenser[dekke]["ovre"][-1] - 1, "HD" + str(Dekker[dekke]["hoyde"]))
1003     #plt.ylabel("CO2-utslipp ved produksjon [kg-CO2-ekv]")
1004     plt.title("CO2-utslipp avhengig av t0 ved t0 = " + str(t0))
1005     #plt.xlabel("Lengde på hulldekke [meter]")
1006     plt.ylabel("Nedboyning [mm]")
1007     plt.xlabel("Lengde på hulldekke [m]")
1008     plt.title("Nedboyning avhengig av dekkelengde ved t0 = " + str(t0))
1009     #plt.hlines(3, 0, 33, color = "black", label = "Grense")
1010     x = [0, 2, 7, 10, 12, 15, 18, 20, 22, 24]
1011     plt.xticks(np.arange(min(x), max(x)+1, 2))
1012
1013     plt.grid()
1014     plt.xlim(4,24)
1015
1016     plt.ylim(0,95)
1017     plt.legend(loc="lower right", fontsize=6)
1018     plt.show()
1019     """
1020     """
1021     farger = ["red", "green", "blue", "black", "magenta", "orange", "limegreen", "cyan", "saddlebrown", "pink"]
1022
1023     #PLOTTE SPENNING MOT L
1024     for dekke in range(0,10):

```



```

1025     plt.plot(plotteGrenser[dekke]["m"], plotteGrenser[dekke]["vibrasjon"], color = farger[dekke])
1026
1027
1028 #plt.plot(lengde, overforing_max, color="red", label="Grense")
1029
1030
1031     #plt.text(plotteGrenser[dekke]["m"][-1] -0.7, plotteGrenser[dekke]["ovre"][-1] - 1, "HD" + str(Dekker[dekke]["hoyde"]))
1032     plt.xlabel("Lengde på hulldekke [meter]")
1033     plt.ylabel("#f1 [Hz]")
1034     plt.title("Frekvens avhengig av dekkelengde ved t0 = " + str(t0) + " 50mm påstop")
1035     plt.hlines(3, 0, 33, color = "red", label = "Grense")
1036     plt.grid()
1037     plt.xlim(0,25)
1038     plt.ylim(0,15)
1039     plt.legend(loc="upper right")
1040     plt.show()
1041
1042     """
1043     """
1044
1045
1046     plt.figure(dpi=1200)
1047     farger = ["red", "green", "blue", "black", "red", "green", "blue", "black", "red", "green"]
1048     farger = ["red", "green", "blue", "black", "magenta", "orange", "limegreen", "cyan", "saddlebrown", "pink"]
1049
1050     for dekke in range(0,10):
1051         if dekke >= 0 and dekke<5:
1052             ny_liste = []
1053             ny_liste2 = []
1054             for num in range(len(plotteGrenser[dekke]["REI"])):
1055                 ny_liste.append(plotteGrenser[dekke]["REI"][num] - (5-dekke))
1056             for num in range(len(plotteGrenser[dekke]["REI_"])):
1057                 ny_liste2.append(plotteGrenser[dekke]["REI_"][num] - (5-dekke))
1058
1059             if plotteGrenser[dekke]["REI"] != []:
1060                 plt.plot(plotteGrenser[dekke]["m_"], ny_liste2, color = farger[dekke], ls="--")
1061                 plt.plot(plotteGrenser[dekke]["m"], ny_liste, color = farger[dekke], linestyle = "-", label= str(Dekker[dekke]["hoyde"]) + " - REI" + str(plotteGrenser[dekke]
1062
1063         if dekke >= 5 and dekke<10:
1064             ny_liste = []
1065             ny_liste2 = []
1066             for num in range(len(plotteGrenser[dekke]["REI"])):
1067                 ny_liste.append(plotteGrenser[dekke]["REI"][num] + (-4+dekke))
1068             for num in range(len(plotteGrenser[dekke]["REI_"])):
1069                 ny_liste2.append(plotteGrenser[dekke]["REI_"][num] + (-4+dekke))
1070
1071             if plotteGrenser[dekke]["REI"] != []:
1072                 plt.plot(plotteGrenser[dekke]["m_"], ny_liste2, color = farger[dekke], ls="--")
1073                 plt.plot(plotteGrenser[dekke]["m"], ny_liste, color = farger[dekke], linestyle = "-", label= str(Dekker[dekke]["hoyde"]) + " - REI" + str(plotteGrenser[dekke]
1074
1075     plt.xlabel("Lengde på hulldekke [meter]")
1076     plt.ylabel("REI [minutter]")
1077     plt.title("REI avhengig av lengde ved t0 = " + str(t0))
1078     #plt.hlines(-19, 0, 33, color = "red", label = "Grense")
1079     plt.grid()
1080     plt.xlim(5,25)
1081     plt.ylim(0,150)
1082     plt.yticks(np.arange(0, 150, 30))
1083     plt.xticks(np.arange(5, 25, 5))
1084     plt.legend(loc="lower right", fontsize = 7)
1085     plt.show()
1086
1087     """
1088     """
1089
1090     farger = ["red", "green", "blue", "black", "magenta", "orange", "limegreen", "cyan", "saddlebrown", "pink"]
1091
1092     #PLOTTE SPENNING MOT L
1093     for dekke in range(0,10):
1094         plt.plot(plotteGrenser[dekke]["m"], plotteGrenser[dekke]["nedre"], color = "black")
1095         plt.plot(plotteGrenser[dekke]["m"], plotteGrenser[dekke]["ovre"], color = "black", linestyle = "--")
1096
1097     #plt.plot(lengde, overforing_max, color="red", label="Grense")
1098
1099
1100     #plt.text(plotteGrenser[dekke]["m"][-1] -0.7, plotteGrenser[dekke]["ovre"][-1] - 1, "HD" + str(Dekker[dekke]["hoyde"]))
1101     plt.xlabel("Lengde på hulldekke [meter]")
1102     plt.ylabel("Spennning [MPa]")
1103     plt.title("Spennning i tverrsnitt ved overforing ved t0 = " + str(t0))
1104     plt.hlines(-19, 0, 33, color = "red", label = "Grense")
1105     plt.grid()
1106     plt.xlim(5,20)
1107     plt.ylim(-20,0)
1108     plt.legend(loc="upper right")
1109     plt.show()
1110     """
1111
1112     """
1113     #PLOTTE LÅSETAP MOT L
1114     plt.plot(plotteGrenser[0]["m"], plotteGrenser[0]["låsetap"], color = "black")
1115     plt.plot(plotteGrenser[1]["m"], plotteGrenser[1]["låsetap"], color = "black")
1116     plt.plot(plotteGrenser[2]["m"], plotteGrenser[2]["låsetap"], color = "black")
1117     plt.plot(plotteGrenser[3]["m"], plotteGrenser[3]["låsetap"], color = "black")
1118     plt.plot(plotteGrenser[4]["m"], plotteGrenser[4]["låsetap"], color = "black")
1119     plt.plot(plotteGrenser[5]["m"], plotteGrenser[5]["låsetap"], color = "black")
1120     plt.plot(plotteGrenser[6]["m"], plotteGrenser[6]["låsetap"], color = "black")
1121     plt.plot(plotteGrenser[7]["m"], plotteGrenser[7]["låsetap"], color = "black")
1122     plt.plot(plotteGrenser[8]["m"], plotteGrenser[8]["låsetap"], color = "black")
1123     plt.plot(plotteGrenser[9]["m"], plotteGrenser[9]["låsetap"], color = "black")
1124     plt.xlabel("Lengde på hulldekke [meter]")
1125     plt.ylabel("Tap pga Låsetap [%]")
1126     plt.title('Låsetap avhengig av lengde på hulldekke ved t0 = ' + str(t0))
1127     plt.grid()
1128
1129     #Regresjonslinje
1130     def model_f(x,a,b,c):
1131         return a*(x-b)**2+c
1132     popt, pcov = curve_fit(model_f, xx, yy_låsetap)
1133
1134     a_opt, b_opt, c_opt = popt
1135     x_model = np.linspace(min(xx), max(xx), 100)
1136     y_model = model_f(x_model, a_opt, b_opt, c_opt)
1137
1138     #plt.scatter(xx, yy)
1139     plt.plot(x_model, y_model, color='r', linestyle = "--", label = "Trendlinje")
1140     plt.legend(loc='upper right')
1141     plt.ylim(0, 15)
1142     plt.show()
1143     """
1144     """
1145     #PLOTTE RELAK MOT L
1146     plt.plot(plotteGrenser[0]["m"], plotteGrenser[0]["relak"], color = "black")
1147     plt.plot(plotteGrenser[1]["m"], plotteGrenser[1]["relak"], color = "black")
1148     plt.plot(plotteGrenser[2]["m"], plotteGrenser[2]["relak"], color = "black")
1149     plt.plot(plotteGrenser[3]["m"], plotteGrenser[3]["relak"], color = "black")
1150     plt.plot(plotteGrenser[4]["m"], plotteGrenser[4]["relak"], color = "black")
1151     plt.plot(plotteGrenser[5]["m"], plotteGrenser[5]["relak"], color = "black")
1152     plt.plot(plotteGrenser[6]["m"], plotteGrenser[6]["relak"], color = "black")

```

```

1153 plt.plot(plotteGrenser[7]["m"], plotteGrenser[7]["relak"], color = "black")
1154 plt.plot(plotteGrenser[8]["m"], plotteGrenser[8]["relak"], color = "black")
1155 plt.plot(plotteGrenser[9]["m"], plotteGrenser[9]["relak"], color = "black")
1156 plt.xlabel("Lengde på hulldekke [meter]")
1157 plt.ylabel("Tap pga relaksasjon[%]")
1158 plt.title('Tap pga relaksasjon avhengig av lengde på hulldekke ved t0 = ' + str(t0))
1159 plt.grid()
1160
1161 #Regresjonslinje
1162 def model_f(x,a,b,c):
1163     return a*(x-b)**2+c
1164 popt, pcov = curve_fit(model_f, xx, yy_relak)
1165
1166 a_opt, b_opt, c_opt = popt
1167 x_model = np.linspace(min(xx), max(xx), 100)
1168 y_model = model_f(x_model, a_opt, b_opt, c_opt)
1169
1170 plt.scatter(xx, yy)
1171 plt.plot(x_model, y_model, color='r', linestyle = "--", label = "Trendlinje")
1172 plt.legend(loc='upper left')
1173 plt.ylim(0,10)
1174 plt.xlim(4,21)
1175
1176 plt.show()
1177 """
1178 """
1179 #PLOTTE KRYP MOT L
1180 plt.plot(plotteGrenser[0]["m"], plotteGrenser[0]["kryp"], color = "black")
1181 plt.plot(plotteGrenser[1]["m"], plotteGrenser[1]["kryp"], color = "black")
1182 plt.plot(plotteGrenser[2]["m"], plotteGrenser[2]["kryp"], color = "black")
1183 plt.plot(plotteGrenser[3]["m"], plotteGrenser[3]["kryp"], color = "black")
1184 plt.plot(plotteGrenser[4]["m"], plotteGrenser[4]["kryp"], color = "black")
1185 plt.plot(plotteGrenser[5]["m"], plotteGrenser[5]["kryp"], color = "black")
1186 plt.plot(plotteGrenser[6]["m"], plotteGrenser[6]["kryp"], color = "black")
1187 plt.plot(plotteGrenser[7]["m"], plotteGrenser[7]["kryp"], color = "black")
1188 plt.plot(plotteGrenser[8]["m"], plotteGrenser[8]["kryp"], color = "black")
1189 plt.plot(plotteGrenser[9]["m"], plotteGrenser[9]["kryp"], color = "black")
1190 plt.xlabel("Lengde på hulldekke [meter]")
1191 plt.ylabel("Tap pga kryp[%]")
1192 plt.title('Tap pga kryp avhengig av lengde på hulldekke ved t0 = ' + str(t0) + " og Ap lik Asmin")
1193 plt.grid()
1194
1195 #Regresjonslinje
1196 def model_f(x,a,b,c):
1197     return a*(x-b)**2+c
1198 popt, pcov = curve_fit(model_f, xx, yy_kryp, maxfev=100000)
1199
1200 a_opt, b_opt, c_opt = popt
1201 x_model = np.linspace(min(xx), max(xx), 100)
1202 y_model = model_f(x_model, a_opt, b_opt, c_opt)
1203
1204 plt.scatter(xx, yy)
1205 plt.plot(x_model, y_model, color='r', linestyle = "--", label = "Trendlinje")
1206 plt.legend(loc='upper left')
1207 plt.ylim(0,10)
1208 plt.xlim(4,21)
1209
1210 plt.show()
1211 """
1212 """
1213 #PLOTTE ARMERING MOT L
1214 plt.plot(plotteGrenser[0]["m"], plotteGrenser[0]["armering"], color = "r")
1215 plt.plot(plotteGrenser[1]["m"], plotteGrenser[1]["armering"], color = "r")
1216 plt.plot(plotteGrenser[2]["m"], plotteGrenser[2]["armering"], color = "r")
1217 plt.plot(plotteGrenser[3]["m"], plotteGrenser[3]["armering"], color = "r")
1218 plt.plot(plotteGrenser[4]["m"], plotteGrenser[4]["armering"], color = "r")
1219 plt.plot(plotteGrenser[5]["m"], plotteGrenser[5]["armering"], color = "r")
1220 plt.plot(plotteGrenser[6]["m"], plotteGrenser[6]["armering"], color = "r")
1221 plt.plot(plotteGrenser[7]["m"], plotteGrenser[7]["armering"], color = "r")
1222 plt.plot(plotteGrenser[8]["m"], plotteGrenser[8]["armering"], color = "r")
1223 plt.plot(plotteGrenser[9]["m"], plotteGrenser[9]["armering"], color = "r")
1224
1225
1226 plt.plot(plotteGrenser[0]["m"], plotteGrenser[0]["Ap"], color = "black")
1227 plt.plot(plotteGrenser[1]["m"], plotteGrenser[1]["Ap"], color = "black")
1228 plt.plot(plotteGrenser[2]["m"], plotteGrenser[2]["Ap"], color = "black")
1229 plt.plot(plotteGrenser[3]["m"], plotteGrenser[3]["Ap"], color = "black")
1230 plt.plot(plotteGrenser[4]["m"], plotteGrenser[4]["Ap"], color = "black")
1231 plt.plot(plotteGrenser[5]["m"], plotteGrenser[5]["Ap"], color = "black")
1232 plt.plot(plotteGrenser[6]["m"], plotteGrenser[6]["Ap"], color = "black")
1233 plt.plot(plotteGrenser[7]["m"], plotteGrenser[7]["Ap"], color = "black")
1234 plt.plot(plotteGrenser[8]["m"], plotteGrenser[8]["Ap"], color = "black")
1235 plt.plot(plotteGrenser[9]["m"], plotteGrenser[9]["Ap"], color = "black")
1236
1237
1238 plt.xlabel("Lengde på hulldekke [meter]")
1239 plt.ylabel("As_min [mm2]")
1240 plt.title('Minimum armeringsareal avhengig av lengde på hulldekke ved t0 = ' + str(t0))
1241 plt.grid()
1242 """
1243 """
1244
1245 """
1246 #PLOTTE SVINNTAP MOT L
1247 plt.plot(plotteGrenser[0]["m"], plotteGrenser[0]["svinntap"], color="black")
1248 plt.plot(plotteGrenser[1]["m"], plotteGrenser[1]["svinntap"], color="black")
1249 plt.plot(plotteGrenser[2]["m"], plotteGrenser[2]["svinntap"], color="black")
1250 plt.plot(plotteGrenser[3]["m"], plotteGrenser[3]["svinntap"], color="black")
1251 plt.plot(plotteGrenser[4]["m"], plotteGrenser[4]["svinntap"], color="black")
1252 plt.plot(plotteGrenser[5]["m"], plotteGrenser[5]["svinntap"], color="black")
1253 plt.plot(plotteGrenser[6]["m"], plotteGrenser[6]["svinntap"], color="black")
1254 plt.plot(plotteGrenser[7]["m"], plotteGrenser[7]["svinntap"], color="black")
1255 plt.plot(plotteGrenser[8]["m"], plotteGrenser[8]["svinntap"], color="black")
1256 plt.plot(plotteGrenser[9]["m"], plotteGrenser[9]["svinntap"], color="black")
1257 plt.xlabel("Lengde på hulldekke [meter]")
1258 plt.ylabel("Svinntap[%]")
1259 plt.title('Svinntap avhengig av lengde på hulldekke ved t0 = ' + str(t0) + " og Ap lik Asmin")
1260 plt.grid()
1261 #Regresjonslinje
1262 def model_f(x,a,b,c):
1263     return a*(x-b)**2+c
1264 popt, pcov = curve_fit(model_f, xx, yy_svinntap, maxfev=100000)
1265
1266 a_opt, b_opt, c_opt = popt
1267 x_model = np.linspace(min(xx), max(xx), 100)
1268 y_model = model_f(x_model, a_opt, b_opt, c_opt)
1269
1270 plt.scatter(xx, yy)
1271 plt.plot(x_model, y_model, color='r', linestyle = "--", label = "Trendlinje")
1272 plt.legend(loc='upper left')
1273 plt.ylim(0,10)
1274 plt.xlim(4,21)
1275
1276 plt.show()
1277 """
1278 """
1279
1280 #PLOTTE FRIKSJONSTAP MOT L

```

```

1281 plt.plot(plotteGrenser[0]["m"], plotteGrenser[0]["friktap"], color="black")
1282 plt.plot(plotteGrenser[1]["m"], plotteGrenser[1]["friktap"], color="black")
1283 plt.plot(plotteGrenser[2]["m"], plotteGrenser[2]["friktap"], color="black")
1284 plt.plot(plotteGrenser[3]["m"], plotteGrenser[3]["friktap"], color="black")
1285 plt.plot(plotteGrenser[4]["m"], plotteGrenser[4]["friktap"], color="black")
1286 plt.plot(plotteGrenser[5]["m"], plotteGrenser[5]["friktap"], color="black")
1287 plt.plot(plotteGrenser[6]["m"], plotteGrenser[6]["friktap"], color="black")
1288 plt.plot(plotteGrenser[7]["m"], plotteGrenser[7]["friktap"], color="black")
1289 plt.plot(plotteGrenser[8]["m"], plotteGrenser[8]["friktap"], color="black")
1290 plt.plot(plotteGrenser[9]["m"], plotteGrenser[9]["friktap"], color="black")
1291 plt.xlabel("Lengde på hulldekke [meter]")
1292 plt.ylabel("Friksjonstap[%]")
1293 plt.title('Friksjonstap avhengig av lengde på hulldekke ved t0 = ' + str(t0))
1294 plt.grid()
1295 #Regresjonslinje
1296 def model_f(x,a,b,c):
1297     return a*(x-b)**2+c
1298 popt, pcov = curve_fit(model_f, xx, yy_frik, maxfev=100000)
1299
1300 a_opt, b_opt, c_opt = popt
1301 x_model = np.linspace(min(xx), max(xx), 100)
1302 y_model = model_f(x_model, a_opt, b_opt, c_opt)
1303
1304 #plt.scatter(xx, yy)
1305 plt.plot(x_model, y_model, color='r', linestyle = "--", label = "Trendlinje")
1306 plt.legend(loc='upper left')
1307 plt.ylim(0,10)
1308
1309
1310 plt.show()
1311 """
1312 """
1313 #PLOTTE P0 MOT LENGDE for en t0
1314
1315 #plt.plot(meter, total_tap_alle_dekker)
1316
1317
1318 plt.plot(plotteGrenser[0]["P0"], plotteGrenser[0]["tap"], color="black")
1319 plt.plot(plotteGrenser[1]["P0"], plotteGrenser[1]["tap"], color="black")
1320 plt.plot(plotteGrenser[2]["P0"], plotteGrenser[2]["tap"], color="black")
1321 plt.plot(plotteGrenser[3]["P0"], plotteGrenser[3]["tap"], color="black")
1322 plt.plot(plotteGrenser[4]["P0"], plotteGrenser[4]["tap"], color="black")
1323 plt.plot(plotteGrenser[5]["P0"], plotteGrenser[5]["tap"], color="black")
1324 plt.plot(plotteGrenser[6]["P0"], plotteGrenser[6]["tap"], color="black")
1325 plt.plot(plotteGrenser[7]["P0"], plotteGrenser[7]["tap"], color="black")
1326 plt.plot(plotteGrenser[8]["P0"], plotteGrenser[8]["tap"], color="black")
1327 plt.plot(plotteGrenser[9]["P0"], plotteGrenser[9]["tap"], color="black")
1328 plt.xlabel("For spenningskraft P0 [kN]")
1329 plt.ylabel("Totalt tap i forspenning[%]")
1330 leg = plt.legend(loc='upper left')
1331 plt.title('Totalt tap i forspenning avhengig av P0 ved t0 = ' + str(t0) + " og Ap = As_min")
1332
1333 plt.grid()
1334
1335 """
1336
1337 #PLOTTE TAP MOT LENGDE for en t0
1338
1339 #plt.plot(meter, total_tap_alle_dekker)
1340 """
1341
1342 for tall in range(0,10):
1343     if plotteGrenser[tall]["tap"] != []:
1344         plt.plot(plotteGrenser[tall]["m"], plotteGrenser[tall]["tap"], color = "black")
1345         #plt.text(plotteGrenser[tall]["m"][-1] + 0.3, plotteGrenser[tall]["tap"][-1] - 0.035, "HD" + str(Dekker[tall]["hoyde"]))
1346
1347
1348 plt.xlabel("Lengde på hulldekke [meter]")
1349 plt.ylabel("Totalt tap i forspenning[%]")
1350
1351 plt.title('Totalt tap i forspenning avhengig av dekkelengde ved t0 = ' + str(t0))
1352 #Regresjonslinje
1353 def model_f(x,a,b,c):
1354     return a*(x-b)**2+c
1355 popt, pcov = curve_fit(model_f, xx, yy, maxfev = 100000)
1356
1357 a_opt, b_opt, c_opt = popt
1358 x_model = np.linspace(min(xx), max(xx), 100)
1359 y_model = model_f(x_model, a_opt, b_opt, c_opt)
1360
1361 #plt.scatter(xx, yy)
1362 plt.plot(x_model, y_model, color='r', linestyle = "--", label = "Trendlinje")
1363
1364 plt.title("hmm")
1365 plt.ylabel("Tap")
1366 plt.xlabel("Lengde")
1367 plt.legend(loc='upper right')
1368 plt.ylim(0, 100)
1369 plt.grid()
1370 plt.show()
1371 """
1372 """
1373 #PLOTTE BESTE DEKKE
1374
1375 #plt.plot(lengde, nedboyninger, label = "Nedboyning [mm]", color="black")
1376 plt.plot(plotteGrenser[0]["m"], plotteGrenser[0]["mm"], color="black")
1377 plt.plot(plotteGrenser[1]["m"], plotteGrenser[1]["mm"], color="black")
1378 plt.plot(plotteGrenser[2]["m"], plotteGrenser[2]["mm"], color="black")
1379 plt.plot(plotteGrenser[3]["m"], plotteGrenser[3]["mm"], color="black")
1380 plt.plot(plotteGrenser[4]["m"], plotteGrenser[4]["mm"], color="black")
1381 plt.plot(plotteGrenser[5]["m"], plotteGrenser[5]["mm"], color="black")
1382 plt.plot(plotteGrenser[6]["m"], plotteGrenser[6]["mm"], color="black")
1383 plt.plot(plotteGrenser[7]["m"], plotteGrenser[7]["mm"], color="black")
1384 plt.plot(plotteGrenser[8]["m"], plotteGrenser[8]["mm"], color="black")
1385 plt.plot(plotteGrenser[9]["m"], plotteGrenser[9]["mm"], color="black")
1386
1387
1388
1389 plt.axhline(y=0, color='black', linestyle='--')
1390
1391 def mm(m):
1392     return m*(1000/250)
1393 m = np.linspace(0, 25, 100)
1394
1395
1396 #plt.plot(lengde, grense, color='r', linestyle='--')
1397 plt.plot(m, mm(m), color='r', label = "Nedboyningskrav", linestyle='--')
1398 plt.xlabel("Lengde på hulldekke [meter]")
1399 plt.ylabel("Nedboyning [mm]")
1400 leg = plt.legend(loc='upper left')
1401 plt.title('Langtidsnedboyning avhengig av dekkelengde ved t0 = ' + str(t0))
1402 plt.xlim(0, 25)
1403 plt.ylim(-20,100)
1404 plt.grid()
1405 """
1406
1407 #print(matrise)
1408 print(matrise520)
1409 return matrise, matrise520

```

```

1409
1410 def forste():
1411     t0 = 3
1412     matrise200_plot = besteDekke(t0)[1]
1413     matrisen_temp = []
1414
1415     #for t0 in range(3,29):
1416     #    print(str(100*(t0/28)) + "%")
1417     #    print("Finner data for t0: " + str(t0))
1418     matrise = besteDekke(t0)[0]
1419     matrisen_temp.append(matrise)
1420
1421     #####
1422
1423     matrisen = []
1424     #try:
1425     #    x420 = [matrise420_plot["lengder"][-1], matrise420_plot["lengder"][-1]]
1426     #    y420 = [163.17742177285788, 119.94360364182425]
1427
1428
1429     #    plt.plot(x420, y420, linestyle = "dotted", color = "black")
1430     #    plt.plot(matrise420_plot["lengder"], matrise420_plot["co2"], color = matrise420_plot["farge"])
1431     #    plt.text(matrise420_plot["lengder"][-1] + 0.2, matrise420_plot["co2"][-1]-2, "HD420", fontsize = 6)
1432
1433     # except:
1434     #    print("hd420 funker ikke her")
1435
1436     for lengde in range(50,250):
1437         lengde = lengde/10
1438         temp = {"meter": 0, "co2": 0, "dekke": 0, "t0": 0, "farge": ""}
1439         farger = ["red", "green", "blue", "black", "magenta", "orange", "limegreen", "cyan", "saddlebrown", "pink"]
1440         dekke = 0
1441         co2 = 1000
1442         m = 0
1443         t0 = 0
1444         hdfarge = 0
1445         hdfarge_ = 0
1446
1447
1448
1449
1450     for t0 in range(len(matrisen_temp)):
1451         for hd in range(0,10):
1452
1453             if matrisen_temp[t0][hd]["lengder"] != []:
1454
1455                 for verdier in range(len(matrisen_temp[t0][hd]["lengder"])):
1456                     #print(verdier, hd, t0)
1457                     #print(matrisen_temp[t0][hd]["lengder"])
1458                     #print(matrisen_temp[t0][hd]["co2"])
1459
1460                     if matrisen_temp[t0][hd]["lengder"][verdier] == lengde and matrisen_temp[t0][hd]["co2"][verdier] < co2:
1461                         co2 = matrisen_temp[t0][hd]["co2"][verdier]
1462                         m = matrisen_temp[t0][hd]["lengder"][verdier]
1463                         dekke = matrisen_temp[t0][hd]["dekke"]
1464                         t0 = matrisen_temp[t0][hd]["t0"]
1465                         hdfarge = matrisen_temp[t0][hd]["farge"]
1466                         hdfarge_ = hd
1467                         #plt.plot(matrisen_temp[t0][hd]["lengder"][verdier], matrisen_temp[t0][hd]["co2"][verdier], "_", markersize = 1,color = farger[hd] )
1468
1469                     if m != 0:
1470                         temp["meter"] = m
1471                         temp["co2"] = co2
1472                         temp["t0"] = t0
1473                         temp["dekke"] = dekke
1474                         temp["farge"] = hdfarge_
1475                         matrisen.append(temp)
1476
1477
1478
1479
1480     print(matrisen)
1481     farger = ["red", "green", "blue", "black", "magenta", "orange", "limegreen", "cyan", "saddlebrown", "pink"]
1482     forste = 0
1483     siste = 0
1484     for meter in range(len(matrisen)):
1485         farge = matrisen[meter]["farge"]
1486         #plt.plot(matrisen[meter]["meter"], matrisen[meter]["co2"], "o", markersize = 10,color = farger[farge] )
1487         if meter == len(matrisen)-1:
1488             plt.text(matrisen[meter]["meter"] + 0.2,matrisen[meter]["co2"]-2 , "t0 3", fontsize=6)
1489
1490
1491
1492
1493         sistefarge = matrisen[meter-1]["farge"]
1494         co2 = matrisen[meter]["co2"]
1495         co2_forrige = matrisen[meter-1]["co2"]
1496         try:
1497             co2_neste = matrisen[meter+1]["co2"]
1498         except IndexError:
1499             co2_neste = co2
1500         m = matrisen[meter]["meter"]
1501
1502         if co2 > co2_forrige:
1503             forste = meter
1504
1505         if co2_neste > co2 or meter == len(matrisen) - 1:
1506             siste = meter
1507             xx = [matrisen[forste]["meter"], matrisen[siste]["meter"]]
1508             yy = [matrisen[forste]["co2"], matrisen[siste]["co2"]]
1509             plt.plot(xx,yy, color = farger[farge], ls="--", dashes=(2, 1))
1510
1511             if matrisen[meter]["co2"] != matrisen[meter-1]["co2"] and meter != 0:
1512                 x = [matrisen[meter-1]["meter"], matrisen[meter]["meter"]]
1513                 y = [matrisen[meter-1]["co2"], matrisen[meter]["co2"]]
1514                 plt.plot(x,y, color = farger[sistefarge], ls="--", dashes=(2, 1))
1515
1516
1517     plt.figure(dpi=1200)
1518     gs = gridspec.GridSpec(1, 2, width_ratios=[3, 1])
1519     plt.subplot(gs[0])
1520     plt.grid()
1521     forste()
1522     plt.xticks(np.arange(5, 27, 5))
1523     plt.yticks(np.arange(0, 350, 50))
1524     plt.ylim(0,350)
1525     plt.xlim(5,27)
1526     plt.xlabel("Lengde på hulldekke [meter]")
1527     plt.ylabel("CO2-utslipp ved produksjon [kg-CO2-ekv]")
1528     #plt.vlines(21, 0, 350, color = "red", ls="--", label = "f1 < 3.0 Hz")
1529     plt.title("CO2-utslipp avhengig av L")
1530     #plt.legend(loc = "center right", fontsize = 7.5)
1531
1532     plt.subplot(gs[1])
1533     Dekker = ["HD200", "HD220", "HD265", "HD285", "HD320", "HD340", "HD400", "HD420", "HD500", "HD520"]
1534     farger = ["red", "green", "blue", "black", "magenta", "orange", "limegreen", "cyan", "saddlebrown", "pink"]
1535     for dekke in range(0,10):
1536         strek = [1,1]

```

```
1537     hoyde = [(20.1-dekke*2), (21.3-dekke*2)]
1538
1539     plt.plot(strek, hoyde, color=farger[dekke])
1540     plt.text(4, (20.8-dekke*2), str(Dekker[dekke]), fontsize = 6)
1541
1542     plt.axis("off")
1543     plt.xlim(0,50)
1544     plt.ylim(0,21.5)
1545     plt.tight_layout()
1546     plt.show()
1547
1548     #besteDekke(3)
1549     print("Koden brukte %s sekunder" % (time.time() - start_time))
```