```python
import numpy as np
from matplotlib import pyplot as plt
import time
from matplotlib import gridspec
from numpy import arange
from scipy.optimize import curve_fit
start_time = time.time()

def besteDekke(t0):

    plotteGrenser = [{"mm": [], "m": [], "tap": [],  "P0": [], "friktap": [], "svinntap": [], "armering": [],
                      "Ap": [], "kryp": [], "relak": [], "låsetap": [], "ovre": [], "nedre": [], "grense_overforing": [],
                      "REI": [], "REI_": [], "t0_": [], "m_": [], "vibrasjon": [], "co2": [], "mmlang":[], "mlang":[],
                      "vaiere": [], "armering": []},

                     {"mm": [], "m": [], "tap": [],  "P0": [], "friktap": [], "svinntap": [], "armering": [],
                      "Ap": [], "kryp": [], "relak": [], "låsetap": [], "ovre": [], "nedre": [], "grense_overforing": [],
                      "REI": [], "REI_": [], "t0_": [], "m_": [], "vibrasjon": [], "co2": [], "mmlang":[], "mlang":[],
                      "vaiere": [], "armering": []},

                     {"mm": [], "m": [], "tap": [],  "P0": [], "friktap": [], "svinntap": [], "armering": [],
                      "Ap": [], "kryp": [], "relak": [], "låsetap": [], "ovre": [], "nedre": [], "grense_overforing": [],
                      "REI": [], "REI_": [], "t0_": [], "m_": [], "vibrasjon": [], "co2": [], "mmlang":[], "mlang":[],
                      "vaiere": [], "armering": []},

                     {"mm": [], "m": [], "tap": [],  "P0": [], "friktap": [], "svinntap": [], "armering": [],
                      "Ap": [], "kryp": [], "relak": [], "låsetap": [], "ovre": [], "nedre": [], "grense_overforing": [],
                      "REI": [], "REI_": [], "t0_": [], "m_": [], "vibrasjon": [], "co2": [], "mmlang":[], "mlang":[],
                      "vaiere": [], "armering": []},

                     {"mm": [], "m": [], "tap": [],  "P0": [], "friktap": [], "svinntap": [], "armering": [],
                      "Ap": [], "kryp": [], "relak": [], "låsetap": [], "ovre": [], "nedre": [], "grense_overforing": [],
                      "REI": [], "REI_": [], "t0_": [], "m_": [], "vibrasjon": [], "co2": [], "mmlang":[], "mlang":[],
                      "vaiere": [], "armering": []},

                     {"mm": [], "m": [], "tap": [],  "P0": [], "friktap": [], "svinntap": [], "armering": [],
                      "Ap": [], "kryp": [], "relak": [], "låsetap": [], "ovre": [], "nedre": [], "grense_overforing": [],
                      "REI": [], "REI_": [], "t0_": [], "m_": [], "vibrasjon": [], "co2": [], "mmlang":[], "mlang":[],
                      "vaiere": [], "armering": []},

                     {"mm": [], "m": [], "tap": [],  "P0": [], "friktap": [], "svinntap": [], "armering": [],
                      "Ap": [], "kryp": [], "relak": [], "låsetap": [], "ovre": [], "nedre": [], "grense_overforing": [],
                      "REI": [], "REI_": [], "t0_": [], "m_": [], "vibrasjon": [], "co2": [], "mmlang":[], "mlang":[],
                      "vaiere": [], "armering": []},

                     {"mm": [], "m": [], "tap": [],  "P0": [], "friktap": [], "svinntap": [], "armering": [],
                      "Ap": [], "kryp": [], "relak": [], "låsetap": [], "ovre": [], "nedre": [], "grense_overforing": [],
                      "REI": [], "REI_": [], "t0_": [], "m_": [], "vibrasjon": [], "co2": [], "mmlang":[], "mlang":[],
                      "vaiere": [], "armering": []},

                     {"mm": [], "m": [], "tap": [],  "P0": [], "friktap": [], "svinntap": [], "armering": [],
                      "Ap": [], "kryp": [], "relak": [], "låsetap": [], "ovre": [], "nedre": [], "grense_overforing": [],
                      "REI": [], "REI_": [], "t0_": [], "m_": [], "vibrasjon": [], "co2": [], "mmlang":[], "mlang":[],
                      "vaiere": [], "armering": []},

                     {"mm": [], "m": [], "tap": [],  "P0": [], "friktap": [], "svinntap": [], "armering": [],
                      "Ap": [], "kryp": [], "relak": [], "låsetap": [], "ovre": [], "nedre": [], "grense_overforing": [],
                      "REI": [], "REI_": [], "t0_": [], "m_": [], "vibrasjon": [], "co2": [], "mmlang":[], "mlang":[],
                      "vaiere": [], "armering": []}]


    total_tap_alle_dekker = []
    meter = []
    xx = []
    yy = []
    yy_frik = []
    yy_svinn = []
    yy_kryp = []
    yy_relak = []
    yy_låsetap = []
    yy_vibrasjon = []

    As_min_krav =[]
    As_min_L = []
    Ap_bruker = []
    Ap_P0 = []

    nedboyninger = []
    lengde = []
    grense = []
    tap_forspenning = []
    t0_verdier = []
    sigma_ovre_graf = []
    sigma_nedre_graf = []
    overforing_max = []

    Dekker = [{"hoyde": 200, "zc": 98.3974, "g1": 2.6,
             "ac": 119959.4297, "iy": 0.62*10**9, "kabler": 7, "iy_påstop": 2680439606},

            {"hoyde": 220, "zc": 102.9530, "g1": 3.1,
             "ac": 146379.2960, "iy": 0.88*10**9, "kabler": 7, "iy_påstop": 3644589882},

            {"hoyde": 265, "zc": 132.5216, "g1": 3.7,
             "ac": 172397.9131, "iy": 1.53*10**9, "kabler": 6, "iy_påstop": 4976569252},

            {"hoyde": 285, "zc": 135.5603, "g1": 4.2,
             "ac": 198735.2650, "iy": 1.98*10**9, "kabler": 6, "iy_påstop":6287956702 },

            {"hoyde": 320, "zc": 157.2759, "g1": 4.2,
             "ac": 188740.8729, "iy": 2.45*10**9, "kabler": 6, "iy_påstop":7132684312 },

            {"hoyde": 340, "zc": 159.5386, "g1": 4.7,
             "ac": 212923.0742, "iy": 3.05*10**9, "kabler": 6, "iy_påstop": 8753722096},

            {"hoyde": 400, "zc": 206.9283, "g1": 5,
             "ac": 215624.6757, "iy": 4.4*10**9, "kabler": 5, "iy_påstop": 12166405021},

            {"hoyde": 420, "zc": 202.79, "g1": 5.5,
             "ac": 274364, "iy": 5.92*10**9, "kabler": 5, "iy_påstop": 14046948900},

            {"hoyde": 500, "zc": 236.7968, "g1": 6.8,
             "ac": 305202.6371, "iy": 9.03*10**9, "kabler": 5, "iy_påstop": 20272618449},

            {"hoyde": 520, "zc": 244.5994, "g1": 7.3,
             "ac": 311094.0677, "iy": 10.29*10**9, "kabler": 6, "iy_påstop": 22461619872}]

    def finnerP(t0,L,dekke):
        P = []
        P_prover = []

        for i in range(1, 10000, 50):
            P_forsok = i * 1000

            f_ck = 45
            f_cm = 53
            f_pk = 1700
            f_p01k = 1550
```

```
            g1_1 = 1.2 * Dekker[dekke]["g1"]
            g1_2 = 1.2 * 0.13 * 27
            g2 = 3
            E_cm = 36000
            E_p = 195000

            # Tverrsnittsvariabler
            h = Dekker[dekke]["hoyde"]
            zc = Dekker[dekke]["zc"]
            Ac = Dekker[dekke]["ac"]
            i = Dekker[dekke]["iy"]
            antall_kabler = Dekker[dekke]["kabler"]

            # Mulige armeringstyper
            armering = [5.2, 6.5, 6.8, 7.0, 7.5, 9, 11, 12.5, 13, 15.2, 16]
            verdige3 = []
            verdige7 = []
            verdige19 = []

            # Min. armering
            As_min = P_forsok/(0.95*f_p01k)

            dmin_3 = (np.sqrt((As_min/antall_kabler)*(1/3)*(4/np.pi)))
            dmin_7 = (np.sqrt((As_min/antall_kabler)*(1/7)*(4/np.pi)))
            dmin_19 = (np.sqrt((As_min/antall_kabler)*(1/19)*(4/np.pi)))

            # Bestemmer armeringsdiameter
            for x in range(len(armering)):
                storrelse = armering[x]

                if storrelse >= dmin_3:
                    verdige3.append(storrelse)


                if storrelse >= dmin_7:
                    verdige7.append(storrelse)


                if storrelse >= dmin_19:
                    verdige19.append(storrelse)


            #3
            if verdige3 != []:
                As_del_3 = 3*(np.pi/4)*(verdige3[0]*verdige3[0])
                A_p_3 = antall_kabler * As_del_3

            else:
                As_del_3 = 3*(np.pi/4)*(100*100)
                A_p_3 = antall_kabler * As_del_3

            #7
            As_del_7 = 7*(np.pi/4)*(verdige7[0]*verdige7[0])
            A_p_7 = antall_kabler * As_del_7

            #19
            As_del_19 = 19*(np.pi/4)*(verdige19[0]*verdige19[0])
            A_p_19 = antall_kabler * As_del_19

            As_del = min(As_del_3, As_del_7, As_del_19)
            A_p = antall_kabler * As_del

            if As_del == As_del_3:
                vaiere = 3
                #cmin_b = 1.5 * verdige3[0] * 1.866
                cmin_b = max(35, 1.5 * verdige3[0] * 1.866)

                e_p = zc - cmin_b - ((1.5*verdige3[0])/2)

            if As_del == As_del_7:
                vaiere = 7
                #cmin_b = 1.5 * verdige7[0] * 3
                cmin_b = max(35, 1.5 * verdige7[0] * 3)

                e_p = zc - cmin_b - ((3*verdige7[0])/2)

            if As_del == As_del_19:
                vaiere = 19
                #cmin_b = 1.5 * verdige19[0] * 5
                cmin_b = max(35, 1.5 * verdige19[0] * 5)

                e_p = zc - cmin_b - ((5*verdige19[0])/2)

            #Tap pga lås
            if P_forsok == 0:
                P_jekk = 1000
            else:
                P_jekk = P_forsok

            delta_L_lås = 5
            delta_epsilon_lås = delta_L_lås / (L*1000)
            epsilon_P_jekk = ((P_jekk) / (E_p * A_p))
            delta_P_lås = (delta_epsilon_lås/epsilon_P_jekk) * P_jekk
            tap_lås_prosent = (delta_P_lås/P_jekk) * 100

            #Tap friksjon
            my_frik = 0.19
            k_frik = 0.005
            theta_A = (2*e_p)/((L*1000)/2)
            theta_B = (2*e_p)/((L*1000)/2)
            theta_midt = 0
            theta_L = theta_A + theta_B

            delta_P_friksjon = (P_forsok)*(1 - np.exp(-my_frik*(theta_L+k_frik*((L*1000)*10**(-3)))))
            friksjonstap_prosent = (delta_P_friksjon/P_forsok)*100

            P_forsok_2 = P_forsok - delta_P_friksjon - delta_P_lås
            tap_friksjon_prosent = (delta_P_friksjon/P_forsok)*100

            # Transformert tverrsnitt for beregninger av laster i tverrsnitt ved overforing (SLS)
            B_cc_t0 = np.e**(0.2*(1-(28/t0)**0.5))
            f_cm_t0 = B_cc_t0 * f_cm
            E_cm_t0 = (f_cm_t0 / f_cm)**0.3 * E_cm
            n = E_p / E_cm_t0
            A_t = Ac + (n-1)*A_p
            yt = (n-1)*A_p*(e_p/A_t)
            It = i + Ac*yt**2 + (n-1)*A_p*(e_p-yt)**2

            # Laster på tverrsnitt og maks spenning ved overforing
            sigma_max = 0.7 * (f_cm_t0 - 8)
            M_egen = (g1_1 * L**2) / 8 * 10**6
            N = -P_forsok_2
            M = N * (e_p-yt)
            M_e = N * (-yt)

            # Lastfordelinger i tverrsnitt ved overforing

            # Nedre ende
```

```
            sigma_ende_overforing = (N/A_t) + ((M_e-N*yt)/It)*(zc-yt)

            # ovre midt
            sigma_ovre_overforing = (N/A_t)+(((M+M_egen)-N*yt)/It)*(-(h-zc)-yt)

            # Nedre midt
            sigma_nedre_overforing = (N/A_t)+(((M+M_egen)-N*yt)/It)*(zc-yt)

            if np.abs(sigma_nedre_overforing) < sigma_max:
                P.append(P_forsok_2)
                P_prover.append(P_forsok)
            elif P == []:
                return [0], [0]
            else:
                return P, P_prover
                break


    def Nedboyning(L,dekke, p_test):

        #P0 = finnerP(t0,L,dekke)[1][p_test]
        P0 = max(1, finnerP(t0,L,dekke)[1][p_test])

        f_ck = 45
        f_cm = 53
        f_pk = 1700
        f_p01k = 1550
        g1_1 = 1.2 * Dekker[dekke]["g1"]
        g1_2 = 1.2 * 0.13 * 27
        g1_2_mindre_pastop = 1.2 * 0.05 * 27
        g2 = 3
        E_cm = 36000
        E_p = 195000

        # Tverrsnittsvariabler
        h = Dekker[dekke]["hoyde"]
        zc = Dekker[dekke]["zc"]
        Ac = Dekker[dekke]["ac"]
        i = Dekker[dekke]["iy"]
        antall_kabler = Dekker[dekke]["kabler"]

        # Mulige armeringstyper
        armering = [5.2, 6.5, 6.8, 7.0, 7.5, 9, 11, 12.5, 13, 15.2, 16]
        verdige3 = []
        verdige7 = []
        verdige19 = []

        # Min. armering
        As_min = P0/(0.95*f_p01k)

        dmin_3 = (np.sqrt((As_min/antall_kabler)*(1/3)*(4/np.pi)))
        dmin_7 = (np.sqrt((As_min/antall_kabler)*(1/7)*(4/np.pi)))
        dmin_19 = (np.sqrt((As_min/antall_kabler)*(1/19)*(4/np.pi)))

        # Bestemmer armeringsdiameter
        for x in range(len(armering)):
            storrelse = armering[x]

            if storrelse >= dmin_3:
                verdige3.append(storrelse)


            if storrelse >= dmin_7:
                verdige7.append(storrelse)


            if storrelse >= dmin_19:
                verdige19.append(storrelse)


        #3
        if verdige3 != []:
            As_del_3 = 3*(np.pi/4)*(verdige3[0]*verdige3[0])
            A_p_3 = antall_kabler * As_del_3

        else:
            As_del_3 = 3*(np.pi/4)*(100*100)
            A_p_3 = antall_kabler * As_del_3

        #7
        As_del_7 = 7*(np.pi/4)*(verdige7[0]*verdige7[0])
        A_p_7 = antall_kabler * As_del_7

        #19
        As_del_19 = 19*(np.pi/4)*(verdige19[0]*verdige19[0])
        A_p_19 = antall_kabler * As_del_19

        As_del = min(As_del_3, As_del_7, As_del_19)
        A_p = antall_kabler * As_del

        if As_del == As_del_3:
            vaiere = 3
            #cmin_b = 1.5 * verdige3[0] * 1.5
            cmin_b = max(35, 1.5 * verdige3[0] * 1.5)

            e_p = zc - cmin_b - ((1.5*verdige3[0])/2)
            diameter = verdige3[0]

            REI = brannkrav(dekke, e_p, antall_kabler, L, verdige3[0], cmin_b, vaiere)[0]
            REI_justert = brannkrav(dekke, e_p, antall_kabler, L, verdige3[0], cmin_b, vaiere)[1]
            moment = brannkrav(dekke, e_p, antall_kabler, L, verdige3[0], cmin_b, vaiere)[2]

        if As_del == As_del_7:
            vaiere = 7
            #cmin_b = 1.5 * verdige7[0] * 3
            cmin_b = max(35, 1.5 * verdige7[0] * 3)

            e_p = zc - cmin_b - ((3*verdige7[0])/2)
            diameter = verdige7[0]

            REI = brannkrav(dekke, e_p, antall_kabler, L, verdige7[0], cmin_b, vaiere)[0]
            REI_justert = brannkrav(dekke, e_p, antall_kabler, L, verdige7[0], cmin_b, vaiere)[1]
            moment = brannkrav(dekke, e_p, antall_kabler, L, verdige7[0], cmin_b, vaiere)[2]

        if As_del == As_del_19:
            vaiere = 19
            #cmin_b = 1.5 * verdige19[0] * 5
            cmin_b = max(35, 1.5 * verdige19[0] * 5)

            e_p = zc - cmin_b - ((5*verdige19[0])/2)
            diameter = verdige19[0]

            REI = brannkrav(dekke, e_p, antall_kabler, L, verdige19[0], cmin_b, vaiere)[0]
            REI_justert = brannkrav(dekke, e_p, antall_kabler, L, verdige19[0], cmin_b, vaiere)[1]
            moment = brannkrav(dekke, e_p, antall_kabler, L, verdige19[0], cmin_b, vaiere)[2]


        resonans = vibrasjon(L, dekke, E_cm, g1_1, g1_2, g1_2_mindre_pastop)
```

```
        #Tap pga lås
        if P0 == 0:
            P_jekk = 1
        else:
            P_jekk = P0
        delta_L_lås = 5
        delta_epsilon_lås = delta_L_lås / (L*1000)
        epsilon_P_jekk = (P_jekk / (E_p * A_p))
        delta_P_lås = (delta_epsilon_lås/epsilon_P_jekk) * P_jekk
        tap_lås_prosent = (delta_P_lås/P_jekk) * 100

        #Tap friksjon
        my_frik = 0.19
        k_frik = 0.005
        theta_A = (2*e_p)/((L*1000)/2)
        theta_B = (2*e_p)/((L*1000)/2)
        theta_midt = 0
        theta_L = theta_A + theta_B

        delta_P_friksjon = P0*(1 - np.exp(-my_frik*(theta_L+k_frik*((L*1000)*10**(-3)))))
        friksjonstap_prosent = (delta_P_friksjon/P0)*100

        P = P0 - delta_P_friksjon - delta_P_lås
        tap_friksjon_prosent = (delta_P_friksjon/P0)*100

        # Transformert tverrsnitt for beregninger av laster i tverrsnitt ved overforing (SLS)
        B_cc_t0 = np.e**(0.2*(1-(28/t0)**0.5))
        f_cm_t0 = B_cc_t0 * f_cm
        E_cm_t0 = (f_cm_t0 / f_cm)**0.3 * E_cm
        n = E_p / E_cm_t0
        A_t = Ac + (n-1)*A_p
        yt = (n-1)*A_p*(e_p/A_t)
        It = i + Ac*yt**2 + (n-1)*A_p*(e_p-yt)**2

        # Laster på tverrsnitt og maks spenning ved overforing
        sigma_max = 0.7 * (f_cm_t0 - 8)
        M_egen = (g1_1 * L**2) / 8 * 10**6
        N = -P
        M = N * (e_p-yt)
        M_e = N * (-yt)

        # Lastfordelinger i tverrsnitt ved overforing
        sigma_ende_overforing = (N/A_t) + ((M_e-N*yt)/It)*(zc-yt)

        # ovre midt
        sigma_ovre_overforing = (N/A_t)+(((M+M_egen)-N*yt)/It)*(-(h-zc)-yt)

        # Nedre midt
        sigma_nedre_overforing = (N/A_t)+(((M+M_egen)-N*yt)/It)*(zc-yt)

        # Langtidsvirkninger

        # Effektiv E-modul

        alfa_1 = (35/f_cm)**0.7
        alfa_2 = (35/f_cm)**0.2
        alfa_3 = (35/f_cm)**0.5

        u = 1200
        h_0 = (2*Ac)/u
        RH = 40

        def B_H(h_0, alfa_3, RH):
            if True:
                B_H = 1.5 * (1 + (0.012*RH)**18) * h_0 + 250 * alfa_3
                if B_H <= 1500 * alfa_3:
                    return B_H
                else:
                    return 1500 * alfa_3

        BH = B_H(h_0, alfa_3, RH)

        Bc_t_t0 = 1
        B_fcm = 16.8/np.sqrt(f_cm)
        B_3 = 1 / (1 + t0**0.2)
        B_28 = 1 / (1 + 28**0.2)
        B_180 = 1 / (1 + 180**0.2)

        phi_RH = (1 + ((1-(RH/100)) / (0.1*h_0**(1/3))) * alfa_1) * alfa_2

        # phi(t)
        phi_0_3 = phi_RH * B_fcm * B_3
        phi_0_28 = phi_RH * B_fcm * B_28
        phi_0_180 = phi_RH * B_fcm * B_180

        # Kryptall
        phi_3 = phi_0_3
        phi_28 = phi_0_28
        phi_180 = phi_0_180

        # Effektiv E-modul
        E_cL3 = E_cm / (1 + phi_3)
        E_cL28 = E_cm / (1 + phi_28)
        E_cL180 = E_cm / (1 + phi_180)

        # Laster
        M_p = - P * e_p
        M_g1_1 = ((g1_1 * L**2) / 8) * 10**6
        M_g1_2 = ((g1_2 * L**2) / 8) * 10**6
        M_g2_perm = ((0.5 * g2 * L**2) / 8) * 10**6
        M_g2_var = ((0.5 * g2 * L**2) / 8) * 10**6

        # For spenninger i tverrsnittet etter lang tid
        M_laster_langtid = M_g1_1 + M_g1_2 + M_g2_perm

        # Gjennomsnittlig E-modul
        over_brøk = ((np.abs(M_p)) + (np.abs(M_g1_1)) + (np.abs(M_g1_2)) + (np.abs(M_g2_perm)) + (np.abs(M_g2_var)))
        under_brøk = ( ((np.abs(M_p))/(E_cL3)) + ((np.abs(M_g1_1))/(E_cL3)) + ((np.abs(M_g1_2))/(E_cL28)) +
                      ((np.abs(M_g2_perm))/(E_cL180)) + ((np.abs(M_g2_var))/(E_cm)) )
        E_middel = over_brøk / under_brøk

        # Justert transformert tverrsnitt mhp kryp
        kryp_n = E_p / E_middel
        kryp_A_t = Ac + (kryp_n-1)*A_p
        kryp_yt = (kryp_n-1)*A_p*(e_p/kryp_A_t)
        kryp_It = i + Ac*kryp_yt**2 + (kryp_n-1)*A_p*(e_p-kryp_yt)**2
        # Justerte laster
        M_kryp = N * (e_p-kryp_yt)
        M_e_kryp = N * (-kryp_yt)

        # Spenningsfordeling mhp kryp, langtid

        # Nedre ende
        sigma_ende_kryp = (N/kryp_A_t) + ((M_e_kryp-N*kryp_yt)/kryp_It)*(zc-kryp_yt)
        #print('sigma_nedre_ende_kryp =',round(sigma_ende_kryp,3),'MPa')

        # ovre midt
        sigma_ovre_kryp = (N/kryp_A_t)+(((M_kryp+M_laster_langtid)-N*kryp_yt)/kryp_It)*(-(h-zc)-kryp_yt)
```

```
        sigma_ovre_kryp = (N/kryp_A_t)+(((M_kryp+M_laster_langtid)-N*kryp_yt)/kryp_It)*(-(h-zc)-kryp_yt)
        #print('sigma_ovre_midt_kryp =',round(sigma_ovre_kryp,3),'MPa')

        # Nedre midt
        sigma_nedre_kryp = (N/kryp_A_t)+(((M_kryp+M_laster_langtid)-N*kryp_yt)/kryp_It)*(zc-kryp_yt)
        #print('sigma_nedre_midt_kryp =',round(sigma_nedre_kryp,3),'MPa')

        # Armering
        sigma_Pk = (N/A_t)+(((M+M_laster_langtid)-N*yt)/It)*(e_p-yt)
        sigma_PL = (N/kryp_A_t)+(((M_kryp+M_laster_langtid)-N*kryp_yt)/kryp_It)*(e_p-kryp_yt)
        epsilon_k = sigma_Pk / E_cm
        epsilon_L = sigma_PL / E_middel
        delta_sigma_P_kryp = (epsilon_L - epsilon_k) * E_p
        delta_P_kryp = delta_sigma_P_kryp * A_p
        tap_kryp_prosent = -(delta_P_kryp/P0)*100

        # Svinn - Antar sementklasse R
        alfa_ds1 = 6
        alfa_ds2 = 0.11
        f_cm0 = 10
        RH_0 = 100
        B_RH = 1.55 * (1-(RH/RH_0)**3)

        epsilon_cd0 = 0.85 * ( (220+110*alfa_ds1)*np.e**(-alfa_ds2*(f_cm/f_cm0)) ) * 10**(-6) * B_RH

        k_h = 0.717
        B_ds_t_ts = 1
        epsilon_cd_t = B_ds_t_ts * k_h * epsilon_cd0
        epsilon_ca_t = 2.5 * (f_ck - 10) * 10**(-6)
        epsilon_cs_t = - (epsilon_cd_t + epsilon_ca_t)

        # Krefter grunnt svinn
        N_s = - epsilon_cs_t * E_p * A_p
        M_s = N_s * (e_p - kryp_yt)

        # Endring av spenning i tverrsnittet grunnet svinn

        # ovre midt
        delta_sigma_ovre_midt_svinn = (N_s / (kryp_A_t)) + (M_s / (kryp_It))*(-(h-zc)-kryp_yt)

        # Nedre Midt
        delta_sigma_nedre_midt_svinn = (N_s / (kryp_A_t)) + (M_s / (kryp_It))*(zc-kryp_yt)

        # Armering
        epsilon_c_s_e = epsilon_cs_t + N_s/(E_middel * kryp_A_t) + M_s/(E_middel * kryp_It)*(e_p-kryp_yt)
        delta_sigma_P_s = epsilon_c_s_e * E_p
        delta_P_svinn = delta_sigma_P_s * A_p
        tap_svinn_prosent = -(delta_P_svinn / P0)*100

        # Relaksasjon
        rho_1000 = 2.5
        sigma_pm0 = 1275
        my = sigma_pm0 / f_pk
        t = 500000
        delta_sigma_pr = 0.8 * (sigma_pm0 * 0.66 * rho_1000 * np.e**(9.1*my) * (t/1000)**(0.75*(1-my)) * 10**(-5))

        # Laster pga relaksasjon
        N_r = delta_sigma_pr * A_p
        M_r = N_r * (e_p - kryp_yt)

        # Endring av spenning i tverrsnittet grunnet relaksasjon

        # ovre midt
        delta_sigma_ovre_midt_relaksasjon = (N_r / (kryp_A_t)) + (M_r / (kryp_It))*(-(h-zc)-kryp_yt)

        # Nedre midt
        delta_sigma_nedre_midt_relaksasjon = (N_r / (kryp_A_t)) + (M_r / (kryp_It))*(zc-kryp_yt)

        # Armering
        delta_sigma_armering = -delta_sigma_pr
        delta_P_relaksasjon = delta_sigma_armering * A_p
        if P0 == 0:
            P0 = 1
            tap_relak_prosent = -(delta_P_relaksasjon / P0 )*100
        else:
            tap_relak_prosent = -(delta_P_relaksasjon / P0 )*100

        # Endelig spenning i tverrsnittet etter lang tid

        # ovre midt
        sigma_endelig_ovre = sigma_ovre_kryp + delta_sigma_ovre_midt_svinn + delta_sigma_ovre_midt_relaksasjon

        # Nedre midt
        sigma_endelig_nedre = sigma_nedre_kryp + delta_sigma_nedre_midt_svinn + delta_sigma_nedre_midt_relaksasjon

        # Armering
        P_endelig = P + delta_P_kryp + delta_P_svinn + delta_P_relaksasjon


        # Nedboyninger i tverrsnittet etter lang tid
        qed = 1.2*(gl_1 + gl_2) + 1.5*(g2)
        q_forspenning = (8 * P_endelig * e_p) / (L*10**3)**2

        #Beregning av nedboyning
        def nedboyning(p):
            return (5*p*(L*10**3)**4)/(384*E_middel*It)

        #Total nedboyning, total tap
        tot_nedboyning = nedboyning(qed) - nedboyning(q_forspenning)
        total_tap_P = (1 - P_endelig/P0) * 100

        E_dyn = 1.175 * E_cm
        i_pastopt = 11.1 * 10**9
        m_pastopt = ((gl_1+gl_2)*1000)/9.81
        fL = (np.pi/(2*L**2)) * (np.sqrt((E_dyn*10**6*i_pastopt*10**(-12))/m_pastopt))

        t0_verdier.append(t0)
        tap_forspenning.append(total_tap_P)
        sigma_ovre_graf.append(sigma_endelig_ovre)
        sigma_nedre_graf.append(sigma_endelig_nedre)

        co2 = kg_co2(dekke, A_p)

        return(tot_nedboyning, total_tap_P, P0, tap_friksjon_prosent, tap_svinn_prosent, As_min, A_p, tap_kryp_prosent,
               tap_relak_prosent, tap_lås_prosent, sigma_ovre_overforing, sigma_nedre_overforing, f_cm_t0,REI, REI_justert,
               resonans, co2, sigma_endelig_nedre, sigma_endelig_ovre, diameter, vaiere)

    def kg_co2(dekke, A_p):
        kubikk_sement = (Dekker[dekke]["ac"] / 10**6) * 0.135   #13.5% sement i betongen
        tonn_sement = kubikk_sement * 1.44                        #1.44 tonn sement per kubikk
        tonn_co2_sement = tonn_sement * 0.8                      #0.8 tonn co2 per tonn sement produsert

        kubikk_armering = (A_p / 10**6)
        tonn_armering = kubikk_armering * 7.85                   #7.85 tonn per kubikk
        tonn_co2_armering = tonn_armering * 5.64                 #5.64 tonn co2 per tonn armering produsert

        total_co2 = (tonn_co2_sement + tonn_co2_armering) * 1000 #kilo

        return total_co2
```

```
        return total_co2

    def brannkrav(dekke, e_p, antall_kabler, L, diameter, cmin_b, vaiere):
        z1 = (Dekker[dekke]["hoyde"] - Dekker[dekke]["zc"]) / 2
        z2 = e_p
        z = z1 + z2
        r = diameter / 2

        g1_1 = 1.2 * Dekker[dekke]["g1"]
        g1_2 = 1.2 * 0.13 * 27
        g2 = 3
        qed = 1.2 * g1_1 + 1.2 * g1_2 + 1.5 * g2

        motstand = [{"rei": 30, "hekv": 60, "dybde": 25,},
            {"rei": 60, "hekv": 80, "dybde": 35,},
            {"rei": 90, "hekv": 100, "dybde": 45,},
            {"rei": 120, "hekv": 120, "dybde": 55,},
            {"rei": 180, "hekv": 150, "dybde": 70,},
            {"rei": 240, "hekv": 175, "dybde": 80,}]

        motstand_justert = [{"rei": 30, "hekv": 60, "dybde": 0,},
            {"rei": 60, "hekv": 80, "dybde": 0,},
            {"rei": 90, "hekv": 100, "dybde": 0,},
            {"rei": 120, "hekv": 120, "dybde": 0,},
            {"rei": 180, "hekv": 150, "dybde": 0,},
            {"rei": 240, "hekv": 175, "dybde": 0,}]

        Fp = (vaiere*np.pi*r**2) * (1550 / 1.15)
        Sd = antall_kabler * Fp
        Md = Sd * (z/1000)
        Med = qed * (L**2 / 8)
        theta0 = 350
        h_ekv = Dekker[dekke]["ac"] / 1200
        moment = ""

        a_dybde = cmin_b

        if Med <= Md:
            moment = "ok"
        else:
            moment = "ikke ok"

        rei = [0]
        rei_ = [0]

        for minutt in range (0,6):
            if h_ekv >= motstand[minutt]["hekv"] and a_dybde >= motstand[minutt]["dybde"]:
                rei.append(motstand[minutt]["rei"])

        ### JUSTERT ARMERINGSDYBDE
        qed_fi = g1_1 + g1_2 + (0.3 * g2)
        Med_fi = qed_fi * (L**2 / 8)
        fd_fi = 1550
        Md_fi = (antall_kabler * fd_fi * (vaiere*np.pi*r**2) * (z/1000)) / 1000

        my_fi = Med_fi / Md_fi

        def myGraf(my_fi):
            return -140.625 * my_fi**2 - 300 * my_fi + 590.625

        if my_fi >= 0.2 and my_fi <= 1:
            thetakrit = myGraf(my_fi)
            justering = (theta0 - thetakrit) / 10



            for minutt in range(0,6):
                motstand_justert[minutt]["dybde"] = motstand[minutt]["dybde"] + justering

                if h_ekv >= motstand_justert[minutt]["hekv"] and a_dybde >= motstand_justert[minutt]["dybde"]:
                    rei_.append(motstand[minutt]["rei"])


        return rei[-1], rei_[-1], moment

    def vibrasjon(L, dekke, Ecm, g1_1, g1_2, g1_2_mindre_påstop):
        g = 9.81
        vekt = (1000 * (g1_1 + g1_2)) / g
        Edyn = 1.175 * Ecm * 10 **6
        fl_hd = (np.pi / (2*L**2)) * np.sqrt((Edyn*Dekker[dekke]["iy_påstop"] * 10**(-12))/(vekt))
        return fl_hd


    for x in range(50,250):
        print(str(round((x/250)*100)) + "%")
        for k in range(0,10):

           dekke = k

           for p_test in range(len(finnerP(t0, x/10, dekke)[1])):

                if Nedboyning(x/10, dekke, p_test)[0] <= (x*100)/250 and Nedboyning(x/10, dekke, p_test)[13] >= 0:



                    if Nedboyning(x/10, dekke,p_test)[0] >= 0.5:
                        if Nedboyning(x/10, dekke, p_test)[17] <= 2.7 and Nedboyning(x/10, dekke, p_test)[18] >= -45:
                            boyning = Nedboyning(x/10, dekke, p_test)[0]
                            nedboyninger.append(boyning)

                            plotteGrenser[k]["mm"].append(boyning)
                            plotteGrenser[k]["m"].append(x/10)

                            #KODE FOR TAP MOT P
                            plotteGrenser[k]["P0"].append(Nedboyning(x/10, dekke)[2]/1000)

                            #KODE FOR TAP MOT L
                            plotteGrenser[k]["tap"].append(Nedboyning(x/10, dekke, p_test)[1])
                            yy.append(Nedboyning(x/10, dekke, p_test)[1])
                            xx.append(x/10)

                            #KODE FOR FRIKSJON MOT L
                            plotteGrenser[k]["friktap"].append(Nedboyning(x/10, dekke)[3])
                            yy_frik.append(Nedboyning(x/10, dekke)[3])

                            #KODE FOR SVINN MOT L
                            plotteGrenser[k]["svinntap"].append(Nedboyning(x/10, dekke)[4])
                            yy_svinn.append(Nedboyning(x/10, dekke)[4])

                            #KODE FOR MIN ARMERING MOT L
                            plotteGrenser[k]["armering"].append(Nedboyning(x/10, dekke)[5])
                            plotteGrenser[k]["Ap"].append(Nedboyning(x/10, dekke)[6])

                            #KODE FOR KRYP MOT L
                            plotteGrenser[k]["kryp"].append(Nedboyning(x/10, dekke)[7])
                            yy_kryp.append(Nedboyning(x/10, dekke)[7])

                            #KODE FOR RELAKSASJON MOT L
                            plotteGrenser[k]["relak"].append(Nedboyning(x/10, dekke)[8])
```

```
                            yy_relak.append(Nedboyning(x/10, dekke)[8])

                            #KODE FOR LÅSETAP MOT L
                            plotteGrenser[k]["låsetap"].append(Nedboyning(x/10, dekke)[9])
                            yy_låsetap.append(Nedboyning(x/10, dekke)[9])

                            #KODE FOR SPENNING MOT L oVRE
                            plotteGrenser[k]["ovre"].append(Nedboyning(x/10, dekke)[10])

                            #KODE FR SPENNING MOT L NEDRE
                            plotteGrenser[k]["nedre"].append(Nedboyning(x/10, dekke)[11])
                            plotteGrenser[k]["grense_overforing"].append(Nedboyning(x/10, dekke)[12])

                            lengde.append(x/10)
                            overforing_max.append(-(0.7 * (Nedboyning(x/10, dekke)[12] - 8)))

                            plotteGrenser[k]["REI"].append(Nedboyning(x/10, dekke, p_test)[13])
                            if Nedboyning(x/10, dekke, p_test)[14] >= Nedboyning(x/10, dekke, p_test)[13]: #!= 0:
                                plotteGrenser[k]["REI_"].append(Nedboyning(x/10, dekke, p_test)[14])
                                plotteGrenser[k]["m_"].append(x/10)

                            plotteGrenser[k]["vibrasjon"].append(Nedboyning(x/10, dekke)[15])
                            yy_vibrasjon.append(Nedboyning(x/10, dekke)[15])

                            plotteGrenser[k]["co2"].append(Nedboyning(x/10, dekke, p_test)[16])
                            plotteGrenser[k]["vaiere"].append(Nedboyning(x/10, dekke, p_test)[20])
                            plotteGrenser[k]["armering"].append(Nedboyning(x/10, dekke, p_test)[19])
                            break

    """
    plt.figure(dpi=1200)

    #PLOTTE KULT REGNBUE TRE
    gs = gridspec.GridSpec(1, 2, width_ratios=[2, 1])
    #fig = plt.figure(figsize=(80, 60))
    plt.subplot(gs[0])

    farger = ["red", "green", "blue", "black", "magenta", "orange", "limegreen", "cyan", "saddlebrown", "pink"]
    #plt.figure(dpi=1200)

    for dekke in range(0,10):
        if plotteGrenser[dekke]["m"] != []:
            plt.plot(plotteGrenser[dekke]["m"], plotteGrenser[dekke]["co2"], color = farger[dekke])#, label = "HD" + str(Dekker[dekke]["hoyde"]) + " - " + str(round(plotteG
            #plt.text(plotteGrenser[dekke]["m"][-1] + 0.05, plotteGrenser[dekke]["co2"][-1]-0.2, "HD" + str(Dekker[dekke]["hoyde"]), fontsize = 6)
            #plt.text(plotteGrenser[dekke]["t0"][-1] + 0.5, plotteGrenser[dekke]["co2"][-1]-0.2, str(plotteGrenser[dekke]["vaiere"][-1]) + "x" + str(plotteGrenser[dekke]["a
            #plt.text(plotteGrenser[dekke]["t0"][0] -0.5, plotteGrenser[dekke]["co2"][0]-0.2, str(plotteGrenser[dekke]["vaiere"][0]) + "x" + str(plotteGrenser[dekke]["armer

    plt.grid()
    plt.xlabel("Lengde på hulldekke [meter]")
    plt.ylabel("CO2-utslipp ved produksjon [kg-CO2-ekv]")
    plt.title("CO2-utslipp avhengig av L ved t0 = " + str(t0))
    plt.xticks(np.arange(5, 27, 5))
    plt.yticks(np.arange(0, 350, 50))
    plt.ylim(0,350)
    plt.xlim(5,27)

    #plt.figure(dpi=1200)

    plt.subplot(gs[1])
    #Dekker = ["HD200", "HD220", "HD265","HD285","HD320","HD340","HD400","HD420","HD500","HD520"]
    farger = ["red", "green", "blue", "black", "magenta", "orange", "limegreen", "cyan", "saddlebrown", "pink"]
    for dekke in range(0,10):
            strek= [1,1]
            hoyde =[(20.1-dekke*2), (21.3-dekke*2)]

            plt.plot(strek, hoyde, color=farger[dekke])
            plt.text(4, (20.8-dekke*2), ("HD" +str(Dekker[dekke]["hoyde"]) + " - Ap: " + str(Dekker[dekke]["kabler"])+ "x" + str(plotteGrenser[dekke]["vaiere"][-1]) + "x" +
            plt.text(4,(20.2-dekke*2), (str(round(plotteGrenser[dekke]["co2"][-1], 2)) + " kg CO2-ekv"), fontsize = 6)
            plt.text(-0.4, (20.5-dekke*2), str(plotteGrenser[dekke]["m"][-1]) + "m", fontsize =7, horizontalalignment='right')

    plt.axis("off")
    plt.title("Lengste dekkelengde:")
    plt.xlim(0,50)
    plt.ylim(0,21.5)
    plt.tight_layout()

    plt.show()
    """


    farger = ["red", "green", "blue", "black", "magenta", "orange", "limegreen", "cyan", "saddlebrown", "pink"]

    regnbue = [{"farger": [], "co2": [], "m": []}]

    matrise = [{"t0": t0, "dekke": 200, "lengder": [], "co2": [], "farge": "red"},
               {"t0": t0, "dekke": 220, "lengder": [], "co2": [], "farge": "green"},
               {"t0": t0, "dekke": 265, "lengder": [], "co2": [], "farge": "blue"},
               {"t0": t0, "dekke": 285, "lengder": [], "co2": [], "farge": "black"},
               {"t0": t0, "dekke": 320, "lengder": [], "co2": [], "farge": "magenta"},
               {"t0": t0, "dekke": 340, "lengder": [], "co2": [], "farge": "orange"},
               {"t0": t0, "dekke": 400, "lengder": [], "co2": [], "farge": "limegreen"},
               {"t0": t0, "dekke": 420, "lengder": [], "co2": [], "farge": "cyan"},
               {"t0": t0, "dekke": 500, "lengder": [], "co2": [], "farge": "saddlebrown"},
               {"t0": t0, "dekke": 520, "lengder": [], "co2": [], "farge": "pink"}]

    matrise520 = {"t0": t0, "dekke": 520, "lengder": [], "co2": [], "farge": "red"}

    laveste = []
    laveste_m = []
    for num in range(len(plotteGrenser[9]["m"])):
        current = 500
        current_m = 0
        current_color = int(0)
        for dekke in range(0,10):

            if dekke == 9:
                try:
                    matrise520["lengder"].append(plotteGrenser[dekke]["m"][num])
                    matrise520["co2"].append(plotteGrenser[dekke]["co2"][num])
                except IndexError:
                    yeet = 0


            try:
                if plotteGrenser[dekke]["co2"][num] < current:
                    current = plotteGrenser[dekke]["co2"][num]
                    current_m = plotteGrenser[dekke]["m"][num]
                    current_color = int(dekke)
            except IndexError:
                current = current
                current_m = current_m
                current_color = current_color
        laveste.append(current)
        laveste_m.append(current_m)
        #print(num, current_color)
        regnbue[0]["farger"].append(current_color)
        regnbue[0]["co2"].append(current)
        regnbue[0]["m"].append(current_m)
```

```
        matrise[current_color]["lengder"].append(current_m)
        matrise[current_color]["co2"].append(current)

    forste = 0
    siste = 0

    for num in range(len(regnbue[0]["m"])):
        farge = regnbue[0]["farger"][num]
        sistefarge = regnbue[0]["farger"][num-1]
        co2 = regnbue[0]["co2"][num]
        co2_forrige = regnbue[0]["co2"][num-1]
        try:
            co2_neste = regnbue[0]["co2"][num+1]
        except IndexError:
            co2_neste = co2
        m = regnbue[0]["m"][num]

        if co2 > co2_forrige:
            forste = num
            #print("no")
            #print(forste, siste)
        if co2_neste > co2 or num == len(regnbue[0]["m"]) - 1:
            siste = num
            #print("yo")
            #print(forste,siste)

            xx = [regnbue[0]["m"][forste], regnbue[0]["m"][siste]]
            yy = [regnbue[0]["co2"][forste], regnbue[0]["co2"][siste]]


            #plt.plot(xx,yy, color = farger[farge])

        #plt.plot(regnbue[0]["m"][num], regnbue[0]["co2"][num], "_", markersize = 10, color = farger[farge])


        if regnbue[0]["co2"][num] != regnbue[0]["co2"][num-1] and num != 0:
            x = [regnbue[0]["m"][num-1], regnbue[0]["m"][num]]
            y = [regnbue[0]["co2"][num-1], regnbue[0]["co2"][num]]

            #plt.plot(x,y, color = farger[sistefarge])


    #print(regnbue[0]["farger"])
    #laveste_m.sort()
    #laveste.sort()
    #plt.plot(laveste_m, laveste, color = "black", linestyle="--")
    #print(laveste_m, laveste)

    #print(regnbue[0]["m"])

    """
    plt.xlabel("t0 [dager]")
    plt.ylabel("Nedboyning [mm]")
    plt.title("Nedboyning avhengig av t0 ved L = " + str(L))
    """
    """
    #plt.axhline(y=(L*4), color='red', linestyle='--', label = "Grense")
    plt.xlim(5, 26)
    #plt.ylim(30,250)
    #plt.ylim(50,170)
    #plt.legend(loc = "upper left", fontsize = 6)
    #plt.ylim(-20,100)

    plt.subplot(gs[1])

    for dekke in range(0,10):
        if plotteGrenser[dekke]["m"] != []:

            strek= [1,1]
            hoyde =[(20.1-dekke*2), (21.3-dekke*2)]

            plt.plot(strek, hoyde, color=farger[dekke])
            #plt.text(4, (20.8-dekke*2), ("HD" +str(Dekker[dekke]["hoyde"])), fontsize =6)
            plt.text(-0.4, (20.5-dekke*2), str(plotteGrenser[dekke]["m"][-1]) + "m", fontsize =7, horizontalalignment='right')
            plt.text(4, (20.8-dekke*2), ("HD" +str(Dekker[dekke]["hoyde"]) + " - Ap: " + str(Dekker[dekke]["kabler"])+ "x" + str(plotteGrenser[dekke]["vaiere"][-1]) + "x" +
            plt.text(4,(20.2-dekke*2), (str(round(plotteGrenser[dekke]["co2"][-1], 2)) + " kg CO2-ekv"), fontsize = 6)
    plt.axis("off")
    plt.xlim(0,50)
    plt.ylim(0,21.5)
    plt.title("Lengste dekkelengde:")

    plt.tight_layout()

    plt.show()
    """
    """

    farger = ["red", "green", "blue", "black", "magenta", "orange", "limegreen", "cyan", "saddlebrown", "pink"]

    #PLOTTE SPENNING MOT L
    for dekke in range(0,10):
        plt.plot(plotteGrenser[dekke]["m"], plotteGrenser[dekke]["mm"], color = farger[dekke], label=("HD" + str(Dekker[dekke]["hoyde"]) + " - " + str(plotteGrenser[dekke][
        #plt.text(plotteGrenser[dekke]["m"][-1]-0.4, (plotteGrenser[dekke]["m"][-1]*4)+4, "HD" + str(Dekker[dekke]["hoyde"]), fontsize = 5)
        #plt.text(plotteGrenser[dekke]["m"][-1], -2, str(plotteGrenser[dekke]["m"][-1]), color = farger[dekke], fontsize=5)
        #plt.plot(plotteGrenser[dekke]["m"][-1], plotteGrenser[dekke]["mm"][-1], "x", color=farger[dekke])
        plt.vlines(plotteGrenser[dekke]["m"][-1], plotteGrenser[dekke]["mm"][-1], (plotteGrenser[dekke]["m"][-1]*4) +2, color =farger[dekke])
        #plt.plot(plotteGrenser[dekke]["mlang"], plotteGrenser[dekke]["mmlang"], color = farger[dekke])
    #print(plotteGrenser[9]["m"], plotteGrenser[9]["mm"])
    def  mm(m):
        return m*(1000/250)
    m = np.linspace(0, 25, 100)


    plt.plot(m, mm(m), color='r', linestyle='--', label = "Grense")
    plt.plot(plotteGrenser[dekke]["m"][-1], plotteGrenser[dekke]["mm"][-1], "x", color=farger[dekke])
        #plt.text(plotteGrenser[dekke]["m"][-1] -0.7, plotteGrenser[dekke]["ovre"][-1] - 1, "HD" + str(Dekker[dekke]["hoyde"]))
    #plt.ylabel("CO2-utslipp ved produksjon [kg-CO2-ekv]")
    #plt.title("CO2-utslipp avhengig av t0 ved t0 = " + str(t0))
    #plt.xlabel("Lengde på hulldekke [meter]")
    plt.ylabel("Nedboyning [mm]")
    plt.xlabel("Lengde på hulldekke [m]")
    plt.title("Nedboyning avhengig av dekkelengde ved t0 = " + str(t0))
    #plt.hlines(3, 0, 33, color = "black", label = "Grense")
    x = [0, 2, 7, 10, 12, 15, 18, 20, 22, 24]
    plt.xticks(np.arange(min(x), max(x)+1, 2))

    plt.grid()
    plt.xlim(4,24)

    plt.ylim(0,95)
    plt.legend(loc="lower right", fontsize=6)
    plt.show()
    """
    """
    farger = ["red", "green", "blue", "black", "magenta", "orange", "limegreen", "cyan", "saddlebrown", "pink"]

    #PLOTTE SPENNING MOT L
    for dekke in range(0,10):
```

```
        plt.plot(plotteGrenser[dekke]["m"], plotteGrenser[dekke]["vibrasjon"], color = farger[dekke])


    #plt.plot(lengde, overforing_max, color= "red", label ="Grense")


        #plt.text(plotteGrenser[dekke]["m"][-1] -0.7, plotteGrenser[dekke]["ovre"][-1] - 1, "HD" + str(Dekker[dekke]["hoyde"]))
    plt.xlabel("Lengde på hulldekke [meter]")
    plt.ylabel("fl [Hz]")
    plt.title("Frekvens avhengig av dekkelengde ved t0 = " + str(t0) + " 50mm påstop")
    plt.hlines(3, 0, 33, color = "red", label = "Grense")
    plt.grid()
    plt.xlim(0,25)
    plt.ylim(0,15)
    plt.legend(loc="upper right")
    plt.show()

    """
    """


    plt.figure(dpi=1200)
    farger = ["red", "green", "blue", "black", "red", "green", "blue", "black", "red", "green"]
    farger = ["red", "green", "blue", "black", "magenta", "orange", "limegreen", "cyan", "saddlebrown", "pink"]

    for dekke in range(0,10):
        if dekke >= 0 and dekke<5:
            ny_liste = []
            ny_liste2 = []
            for num in range(len(plotteGrenser[dekke]["REI"])):
                ny_liste.append(plotteGrenser[dekke]["REI"][num] - (5-dekke))
            for num in range(len(plotteGrenser[dekke]["REI_"])):
                ny_liste2.append(plotteGrenser[dekke]["REI_"][num] - (5-dekke))

            if plotteGrenser[dekke]["REI"] != []:
                plt.plot(plotteGrenser[dekke]["m_"], ny_liste2, color = farger[dekke], ls="--")
                plt.plot(plotteGrenser[dekke]["m"], ny_liste, color = farger[dekke], linestyle = "-", label= str(Dekker[dekke]["hoyde"]) + " - REI" + str(plotteGrenser[dekk

        if dekke >= 5 and dekke<10:
            ny_liste = []
            ny_liste2 = []
            for num in range(len(plotteGrenser[dekke]["REI"])):
                ny_liste.append(plotteGrenser[dekke]["REI"][num] + (-4+dekke))
            for num in range(len(plotteGrenser[dekke]["REI_"])):
                ny_liste2.append(plotteGrenser[dekke]["REI_"][num] + (-4+dekke))

            if plotteGrenser[dekke]["REI"] != []:
                plt.plot(plotteGrenser[dekke]["m_"], ny_liste2, color = farger[dekke], ls="--")
                plt.plot(plotteGrenser[dekke]["m"], ny_liste, color = farger[dekke], linestyle = "-", label= str(Dekker[dekke]["hoyde"]) + " - REI" + str(plotteGrenser[dekk

    plt.xlabel("Lengde på hulldekke [meter]")
    plt.ylabel("REI [minutter]")
    plt.title("REI avhengig av lengde ved t0 = " + str(t0))
    #plt.hlines(-19, 0, 33, color = "red", label = "Grense")
    plt.grid()
    plt.xlim(5,25)
    plt.ylim(0,150)
    plt.yticks(np.arange(0, 150, 30))
    plt.xticks(np.arange(5, 25, 5))
    plt.legend(loc="lower right", fontsize = 7)
    plt.show()

    """
    """

    farger = ["red", "green", "blue", "black", "magenta", "orange", "limegreen", "cyan", "saddlebrown", "pink"]

    #PLOTTE SPENNING MOT L
    for dekke in range(0,10):
        plt.plot(plotteGrenser[dekke]["m"], plotteGrenser[dekke]["nedre"], color = "black")
        plt.plot(plotteGrenser[dekke]["m"], plotteGrenser[dekke]["ovre"], color = "black", linestyle = "--")

    #plt.plot(lengde, overforing_max, color= "red", label ="Grense")


        #plt.text(plotteGrenser[dekke]["m"][-1] -0.7, plotteGrenser[dekke]["ovre"][-1] - 1, "HD" + str(Dekker[dekke]["hoyde"]))
    plt.xlabel("Lengde på hulldekke [meter]")
    plt.ylabel("Spenning [MPa]")
    plt.title("Spenning i tverrsnitt ved overforing ved t0 = " + str(t0))
    plt.hlines(-19, 0, 33, color = "red", label = "Grense")
    plt.grid()
    plt.xlim(5,20)
    plt.ylim(-20,0)
    plt.legend(loc="upper right")
    plt.show()
    """

    """
    #PLOTTE LÅSETAP MOT L
    plt.plot(plotteGrenser[0]["m"], plotteGrenser[0]["låsetap"], color = "black")
    plt.plot(plotteGrenser[1]["m"], plotteGrenser[1]["låsetap"], color = "black")
    plt.plot(plotteGrenser[2]["m"], plotteGrenser[2]["låsetap"], color = "black")
    plt.plot(plotteGrenser[3]["m"], plotteGrenser[3]["låsetap"], color = "black")
    plt.plot(plotteGrenser[4]["m"], plotteGrenser[4]["låsetap"], color = "black")
    plt.plot(plotteGrenser[5]["m"], plotteGrenser[5]["låsetap"], color = "black")
    plt.plot(plotteGrenser[6]["m"], plotteGrenser[6]["låsetap"], color = "black")
    plt.plot(plotteGrenser[7]["m"], plotteGrenser[7]["låsetap"], color = "black")
    plt.plot(plotteGrenser[8]["m"], plotteGrenser[8]["låsetap"], color = "black")
    plt.plot(plotteGrenser[9]["m"], plotteGrenser[9]["låsetap"], color = "black")
    plt.xlabel("Lengde på hulldekke [meter]")
    plt.ylabel("Tap pga Låsetap[%]")
    plt.title('Låsetap avhengig av lengde på hulldekke ved t0 = ' + str(t0))
    plt.grid()

    #Regresjonslinje
    def model_f(x,a,b,c):
        return a*(x-b)**2+c
    popt, pcov = curve_fit(model_f, xx, yy_låsetap)

    a_opt, b_opt, c_opt = popt
    x_model = np.linspace(min(xx), max(xx), 100)
    y_model = model_f(x_model, a_opt, b_opt, c_opt)

    #plt.scatter(xx, yy)
    plt.plot(x_model, y_model, color='r', linestyle = "--", label = "Trendlinje")
    plt.legend(loc='upper right')
    plt.ylim(0, 15)
    plt.show()
    """
    """
    #PLOTTE RELAK MOT L
    plt.plot(plotteGrenser[0]["m"], plotteGrenser[0]["relak"], color = "black")
    plt.plot(plotteGrenser[1]["m"], plotteGrenser[1]["relak"], color = "black")
    plt.plot(plotteGrenser[2]["m"], plotteGrenser[2]["relak"], color = "black")
    plt.plot(plotteGrenser[3]["m"], plotteGrenser[3]["relak"], color = "black")
    plt.plot(plotteGrenser[4]["m"], plotteGrenser[4]["relak"], color = "black")
    plt.plot(plotteGrenser[5]["m"], plotteGrenser[5]["relak"], color = "black")
    plt.plot(plotteGrenser[6]["m"], plotteGrenser[6]["relak"], color = "black")
```

```
    plt.plot(plotteGrenser[7]["m"], plotteGrenser[7]["relak"], color = "black")
    plt.plot(plotteGrenser[8]["m"], plotteGrenser[8]["relak"], color = "black")
    plt.plot(plotteGrenser[9]["m"], plotteGrenser[9]["relak"], color = "black")
    plt.xlabel("Lengde på hulldekke [meter]")
    plt.ylabel("Tap pga relaksasjon[%]")
    plt.title('Tap pga relaksasjon avhengig av lengde på hulldekke ved t0 = ' + str(t0))
    plt.grid()

    #Regresjonslinje
    def model_f(x,a,b,c):
        return a*(x-b)**2+c
    popt, pcov = curve_fit(model_f, xx, yy_relak)

    a_opt, b_opt, c_opt = popt
    x_model = np.linspace(min(xx), max(xx), 100)
    y_model = model_f(x_model, a_opt, b_opt, c_opt)

    #plt.scatter(xx, yy)
    plt.plot(x_model, y_model, color='r', linestyle = "--", label = "Trendlinje")
    plt.legend(loc='upper left')
    plt.ylim(0,10)
    #plt.xlim(4,21)

    plt.show()
    """
    """
    #PLOTTE KRYP MOT L
    plt.plot(plotteGrenser[0]["m"], plotteGrenser[0]["kryp"], color = "black")
    plt.plot(plotteGrenser[1]["m"], plotteGrenser[1]["kryp"], color = "black")
    plt.plot(plotteGrenser[2]["m"], plotteGrenser[2]["kryp"], color = "black")
    plt.plot(plotteGrenser[3]["m"], plotteGrenser[3]["kryp"], color = "black")
    plt.plot(plotteGrenser[4]["m"], plotteGrenser[4]["kryp"], color = "black")
    plt.plot(plotteGrenser[5]["m"], plotteGrenser[5]["kryp"], color = "black")
    plt.plot(plotteGrenser[6]["m"], plotteGrenser[6]["kryp"], color = "black")
    plt.plot(plotteGrenser[7]["m"], plotteGrenser[7]["kryp"], color = "black")
    plt.plot(plotteGrenser[8]["m"], plotteGrenser[8]["kryp"], color = "black")
    plt.plot(plotteGrenser[9]["m"], plotteGrenser[9]["kryp"], color = "black")
    plt.xlabel("Lengde på hulldekke [meter]")
    plt.ylabel("Tap pga kryp[%]")
    plt.title('Tap pga kryp avhengig av lengde på hulldekke ved t0 = ' + str(t0) + " og Ap lik Asmin")
    plt.grid()

    #Regresjonslinje
    def model_f(x,a,b,c):
        return a*(x-b)**2+c
    popt, pcov = curve_fit(model_f, xx, yy_kryp, maxfev=100000)

    a_opt, b_opt, c_opt = popt
    x_model = np.linspace(min(xx), max(xx), 100)
    y_model = model_f(x_model, a_opt, b_opt, c_opt)

    #plt.scatter(xx, yy)
    plt.plot(x_model, y_model, color='r', linestyle = "--", label = "Trendlinje")
    plt.legend(loc='upper left')
    plt.ylim(0,10)
    #plt.xlim(4,21)

    plt.show()
    """
    #PLOTTE ARMERING MOT L
    """
    plt.plot(plotteGrenser[0]["m"], plotteGrenser[0]["armering"], color = "r")
    plt.plot(plotteGrenser[1]["m"], plotteGrenser[1]["armering"], color = "r")
    plt.plot(plotteGrenser[2]["m"], plotteGrenser[2]["armering"], color = "r")
    plt.plot(plotteGrenser[3]["m"], plotteGrenser[3]["armering"], color = "r")
    plt.plot(plotteGrenser[4]["m"], plotteGrenser[4]["armering"], color = "r")
    plt.plot(plotteGrenser[5]["m"], plotteGrenser[5]["armering"], color = "r")
    plt.plot(plotteGrenser[6]["m"], plotteGrenser[6]["armering"], color = "r")
    plt.plot(plotteGrenser[7]["m"], plotteGrenser[7]["armering"], color = "r")
    plt.plot(plotteGrenser[8]["m"], plotteGrenser[8]["armering"], color = "r")
    plt.plot(plotteGrenser[9]["m"], plotteGrenser[9]["armering"], color = "r")


    plt.plot(plotteGrenser[0]["m"], plotteGrenser[0]["Ap"], color = "black")
    plt.plot(plotteGrenser[1]["m"], plotteGrenser[1]["Ap"], color = "black")
    plt.plot(plotteGrenser[2]["m"], plotteGrenser[2]["Ap"], color = "black")
    plt.plot(plotteGrenser[3]["m"], plotteGrenser[3]["Ap"], color = "black")
    plt.plot(plotteGrenser[4]["m"], plotteGrenser[4]["Ap"], color = "black")
    plt.plot(plotteGrenser[5]["m"], plotteGrenser[5]["Ap"], color = "black")
    plt.plot(plotteGrenser[6]["m"], plotteGrenser[6]["Ap"], color = "black")
    plt.plot(plotteGrenser[7]["m"], plotteGrenser[7]["Ap"], color = "black")
    plt.plot(plotteGrenser[8]["m"], plotteGrenser[8]["Ap"], color = "black")
    plt.plot(plotteGrenser[9]["m"], plotteGrenser[9]["Ap"], color = "black")


    plt.xlabel("Lengde på hulldekke [meter]")
    plt.ylabel("As_min [mm2]")
    plt.title('Minimum armeringsareal avhengig av lengde på hulldekke ved t0 = ' + str(t0))
    plt.grid()
    """


    """
    #PLOTTE SVINNTAP MOT L
    plt.plot(plotteGrenser[0]["m"], plotteGrenser[0]["svinntap"], color="black")
    plt.plot(plotteGrenser[1]["m"], plotteGrenser[1]["svinntap"], color="black")
    plt.plot(plotteGrenser[2]["m"], plotteGrenser[2]["svinntap"], color="black")
    plt.plot(plotteGrenser[3]["m"], plotteGrenser[3]["svinntap"], color="black")
    plt.plot(plotteGrenser[4]["m"], plotteGrenser[4]["svinntap"], color="black")
    plt.plot(plotteGrenser[5]["m"], plotteGrenser[5]["svinntap"], color="black")
    plt.plot(plotteGrenser[6]["m"], plotteGrenser[6]["svinntap"], color="black")
    plt.plot(plotteGrenser[7]["m"], plotteGrenser[7]["svinntap"], color="black")
    plt.plot(plotteGrenser[8]["m"], plotteGrenser[8]["svinntap"], color="black")
    plt.plot(plotteGrenser[9]["m"], plotteGrenser[9]["svinntap"], color="black")
    plt.xlabel("Lengde på hulldekke [meter]")
    plt.ylabel("Svinntap[%]")
    plt.title('Svinntap avhengig av lengde på hulldekke ved t0 = ' + str(t0) + " og Ap lik Asmin")
    plt.grid()
    #Regresjonslinje
    def model_f(x,a,b,c):
        return a*(x-b)**2+c
    popt, pcov = curve_fit(model_f, xx, yy_svinn, maxfev=100000)

    a_opt, b_opt, c_opt = popt
    x_model = np.linspace(min(xx), max(xx), 100)
    y_model = model_f(x_model, a_opt, b_opt, c_opt)

    #plt.scatter(xx, yy)
    plt.plot(x_model, y_model, color='r', linestyle = "--", label = "Trendlinje")
    plt.legend(loc='upper left')
    plt.ylim(0,10)
    #plt.xlim(4,21)

    plt.show()
    """
    """

    #PLOTTE FRIKSJONSTAP MOT L
```

```
    plt.plot(plotteGrenser[0]["m"], plotteGrenser[0]["friktap"], color="black")
    plt.plot(plotteGrenser[1]["m"], plotteGrenser[1]["friktap"], color="black")
    plt.plot(plotteGrenser[2]["m"], plotteGrenser[2]["friktap"], color="black")
    plt.plot(plotteGrenser[3]["m"], plotteGrenser[3]["friktap"], color="black")
    plt.plot(plotteGrenser[4]["m"], plotteGrenser[4]["friktap"], color="black")
    plt.plot(plotteGrenser[5]["m"], plotteGrenser[5]["friktap"], color="black")
    plt.plot(plotteGrenser[6]["m"], plotteGrenser[6]["friktap"], color="black")
    plt.plot(plotteGrenser[7]["m"], plotteGrenser[7]["friktap"], color="black")
    plt.plot(plotteGrenser[8]["m"], plotteGrenser[8]["friktap"], color="black")
    plt.plot(plotteGrenser[9]["m"], plotteGrenser[9]["friktap"], color="black")
    plt.xlabel("Lengde på hulldekke [meter]")
    plt.ylabel("Friksjonstap[%]")
    plt.title('Friksjonstap avhengig av lengde på hulldekke ved t0 = ' + str(t0))
    plt.grid()
    #Regresjonslinje
    def model_f(x,a,b,c):
        return a*(x-b)**2+c
    popt, pcov = curve_fit(model_f, xx, yy_frik, maxfev=100000)

    a_opt, b_opt, c_opt = popt
    x_model = np.linspace(min(xx), max(xx), 100)
    y_model = model_f(x_model, a_opt, b_opt, c_opt)

    #plt.scatter(xx, yy)
    plt.plot(x_model, y_model, color='r', linestyle = "--", label = "Trendlinje")
    plt.legend(loc='upper left')
    plt.ylim(0,10)


    plt.show()
    """
    """
    #PLOTTE P0 MOT LENGDE for en t0

    #plt.plot(meter, total_tap_alle_dekker)


    plt.plot(plotteGrenser[0]["P0"], plotteGrenser[0]["tap"], color="black")
    plt.plot(plotteGrenser[1]["P0"], plotteGrenser[1]["tap"], color="black")
    plt.plot(plotteGrenser[2]["P0"], plotteGrenser[2]["tap"], color="black")
    plt.plot(plotteGrenser[3]["P0"], plotteGrenser[3]["tap"], color="black")
    plt.plot(plotteGrenser[4]["P0"], plotteGrenser[4]["tap"], color="black")
    plt.plot(plotteGrenser[5]["P0"], plotteGrenser[5]["tap"], color="black")
    plt.plot(plotteGrenser[6]["P0"], plotteGrenser[6]["tap"], color="black")
    plt.plot(plotteGrenser[7]["P0"], plotteGrenser[7]["tap"], color="black")
    plt.plot(plotteGrenser[8]["P0"], plotteGrenser[8]["tap"], color="black")
    plt.plot(plotteGrenser[9]["P0"], plotteGrenser[9]["tap"], color="black")
    plt.xlabel("Forspenningskraft P0 [kN]")
    plt.ylabel("Totalt tap i forspenning[%]")
    leg = plt.legend(loc='upper left')
    plt.title('Totalt tap i forspenning avhengig av P0 ved t0 = ' + str(t0) + " og Ap = As_min")

    plt.grid()

    """

    #PLOTTE TAP MOT LENGDE for en t0

    #plt.plot(meter, total_tap_alle_dekker)
    """

    for tall in range(0,10):
        if plotteGrenser[tall]["tap"] != []:
            plt.plot(plotteGrenser[tall]["m"], plotteGrenser[tall]["tap"], color = "black")
            #plt.text(plotteGrenser[tall]["m"][-1] + 0.3, plotteGrenser[tall]["tap"][-1] - 0.035, "HD" + str(Dekker[tall]["hoyde"]))


    plt.xlabel("Lengde på hulldekke [meter]")
    plt.ylabel("Totalt tap i forspenning[%]")

    plt.title('Totalt tap i forspenning avhengig av dekkelengde ved t0 = ' + str(t0))
    #Regresjonslinje
    def model_f(x,a,b,c):
        return a*(x-b)**2+c
    popt, pcov = curve_fit(model_f, xx, yy, maxfev = 100000)

    a_opt, b_opt, c_opt = popt
    x_model = np.linspace(min(xx), max(xx), 100)
    y_model = model_f(x_model, a_opt, b_opt, c_opt)

    #plt.scatter(xx, yy)
    #plt.plot(x_model, y_model, color='r', linestyle = "--", label = "Trendlinje")

    plt.title("hmm")
    plt.ylabel("Tap")
    plt.xlabel("Lengde")
    plt.legend(loc='upper right')
    plt.ylim(0, 100)
    plt.grid()
    plt.show()
    """
    """
    #PLOTTE BESTE DEKKE

    #plt.plot(lengde, nedboyninger, label = "Nedboyning [mm]", color="black")
    plt.plot(plotteGrenser[0]["m"], plotteGrenser[0]["mm"], color="black")
    plt.plot(plotteGrenser[1]["m"], plotteGrenser[1]["mm"], color="black")
    plt.plot(plotteGrenser[2]["m"], plotteGrenser[2]["mm"], color="black")
    plt.plot(plotteGrenser[3]["m"], plotteGrenser[3]["mm"], color="black")
    plt.plot(plotteGrenser[4]["m"], plotteGrenser[4]["mm"], color="black")
    plt.plot(plotteGrenser[5]["m"], plotteGrenser[5]["mm"], color="black")
    plt.plot(plotteGrenser[6]["m"], plotteGrenser[6]["mm"], color="black")
    plt.plot(plotteGrenser[7]["m"], plotteGrenser[7]["mm"], color="black")
    plt.plot(plotteGrenser[8]["m"], plotteGrenser[8]["mm"], color="black")
    plt.plot(plotteGrenser[9]["m"], plotteGrenser[9]["mm"], color="black")



    plt.axhline(y=0, color='black', linestyle='--')

    def  mm(m):
        return m*(1000/250)
    m = np.linspace(0, 25, 100)


    #plt.plot(lengde, grense, color='r', linestyle='--')
    plt.plot(m, mm(m), color='r', label ="Nedboyningskrav", linestyle='--')
    plt.xlabel("Lengde på hulldekke [meter]")
    plt.ylabel("Nedboyning [mm]")
    leg = plt.legend(loc='upper left')
    plt.title('Langtidsnedboyning avhengig av dekkelengde ved t0 = ' + str(t0))
    plt.xlim(0, 25)
    plt.ylim(-20,100)
    plt.grid()
    """
    #print(matrise)
    print(matrise520)
    return matrise, matrise520
```

```

def forste():
    t0 = 3
    matrise200_plot = besteDekke(t0)[1]
    matrisen_temp = []

    #for t0 in range(3,29):
     #   print(str(100*(t0/28)) + "%")
      #  print("Finner data for t0: " + str(t0))
    matrise = besteDekke(t0)[0]
    matrisen_temp.append(matrise)

    ##########################

    matrisen = []
    #try:
    #    x420 = [matrise420_plot["lengder"][-1], matrise420_plot["lengder"][-1]]
    #    y420 = [163.17742177285788, 119.94360364182425]


    #    plt.plot(x420, y420, linestyle = "dotted", color = "black")
    #    plt.plot(matrise420_plot["lengder"], matrise420_plot["co2"], color = matrise420_plot["farge"])
    #    plt.text(matrise420_plot["lengder"][-1] + 0.2, matrise420_plot["co2"][-1]-2, "HD420", fontsize = 6)

   # except:
   #     print("hd420 funker ikke her")

    for lengde in range(50,250):
        lengde = lengde/10
        temp = {"meter" : 0, "co2": 0, "dekke": 0, "t0": 0, "farge": ""}
        farger = ["red", "green", "blue", "black", "magenta", "orange", "limegreen", "cyan", "saddlebrown", "pink"]
        dekke = 0
        co2 = 1000
        m = 0
        t0_ = 0
        hdfarge = 0
        hdfarge_ = 0




        for t0 in range(len(matrisen_temp)):
            for hd in range(0,10):

                if matrisen_temp[t0][hd]["lengder"] != []:

                    for verdier in range(len(matrisen_temp[t0][hd]["lengder"])):
                        #print(verdier, hd, t0)
                        #print(matrisen_temp[t0][hd]["lengder"])
                        #print(matrisen_temp[t0][hd]["co2"])

                        if matrisen_temp[t0][hd]["lengder"][verdier] == lengde and matrisen_temp[t0][hd]["co2"][verdier] < co2:
                            co2 = matrisen_temp[t0][hd]["co2"][verdier]
                            m = matrisen_temp[t0][hd]["lengder"][verdier]
                            dekke = matrisen_temp[t0][hd]["dekke"]
                            t0_ = matrisen_temp[t0][hd]["t0"]
                            hdfarge = matrisen_temp[t0][hd]["farge"]
                            hdfarge_ = hd
                            #plt.plot(matrisen_temp[t0][hd]["lengder"][verdier], matrisen_temp[t0][hd]["co2"][verdier], "_", markersize = 1,color = farger[hd] )
        if m != 0:
            temp["meter"] = m
            temp["co2"] = co2
            temp["t0"] = t0_
            temp["dekke"] = dekke
            temp["farge"] = hdfarge_
            matrisen.append(temp)




    print(matrisen)
    farger = ["red", "green", "blue", "black", "magenta", "orange", "limegreen", "cyan", "saddlebrown", "pink"]
    forste = 0
    siste = 0
    for meter in range(len(matrisen)):
        farge = matrisen[meter]["farge"]
        #plt.plot(matrisen[meter]["meter"], matrisen[meter]["co2"], "o", markersize = 10,color = farger[farge] )
        if meter == len(matrisen)-1:
            plt.text(matrisen[meter]["meter"] + 0.2,matrisen[meter]["co2"]-2 , "t0 3", fontsize=6)




        sistefarge = matrisen[meter-1]["farge"]
        co2 = matrisen[meter]["co2"]
        co2_forrige = matrisen[meter-1]["co2"]
        try:
            co2_neste = matrisen[meter+1]["co2"]
        except IndexError:
            co2_neste = co2
        m = matrisen[meter]["meter"]

        if co2 > co2_forrige:
            forste = meter

        if co2_neste > co2 or meter == len(matrisen) - 1:
            siste = meter
            xx = [matrisen[forste]["meter"], matrisen[siste]["meter"]]
            yy = [matrisen[forste]["co2"], matrisen[siste]["co2"]]
            plt.plot(xx,yy, color = farger[farge], ls="--", dashes=(2, 1))

        if matrisen[meter]["co2"] != matrisen[meter-1]["co2"] and meter != 0:
            x = [matrisen[meter-1]["meter"], matrisen[meter]["meter"]]
            y = [matrisen[meter-1]["co2"], matrisen[meter]["co2"]]
            plt.plot(x,y, color = farger[sistefarge], ls="--", dashes=(2, 1))


plt.figure(dpi=1200)
gs = gridspec.GridSpec(1, 2, width_ratios=[3, 1])
plt.subplot(gs[0])
plt.grid()
forste()
plt.xticks(np.arange(5, 27, 5))
plt.yticks(np.arange(0, 350, 50))
plt.ylim(0,350)
plt.xlim(5,27)
plt.xlabel("Lengde på hulldekke [meter]")
plt.ylabel("CO2-utslipp ved produksjon [kg-CO2-ekv]")
#plt.vlines(21, 0, 350, color = "red", ls = "--", label = "f1 < 3.0 Hz")
plt.title("CO2-utslipp avhengig av L")
#plt.legend(loc = "center right", fontsize = 7.5)

plt.subplot(gs[1])
Dekker = ["HD200", "HD220", "HD265","HD285","HD320","HD340","HD400","HD420","HD500","HD520"]
farger = ["red", "green", "blue", "black", "magenta", "orange", "limegreen", "cyan", "saddlebrown", "pink"]
for dekke in range(0,10):
        strek= [1,1]
```

```
        hoyde =[(20.1-dekke*2), (21.3-dekke*2)]

        plt.plot(strek, hoyde, color=farger[dekke])
        plt.text(4, (20.8-dekke*2), str(Dekker[dekke]), fontsize = 6)

plt.axis("off")
plt.xlim(0,50)
plt.ylim(0,21.5)
plt.tight_layout()
plt.show()

#besteDekke(3)
print("Koden brukte %s sekunder" % (time.time() - start_time))
```