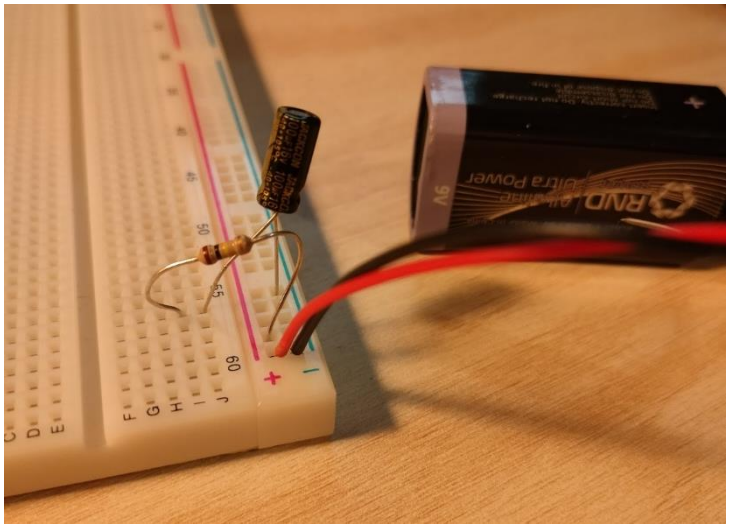
Oblig-Fil TMA4101- OPG. RC-Kretsen

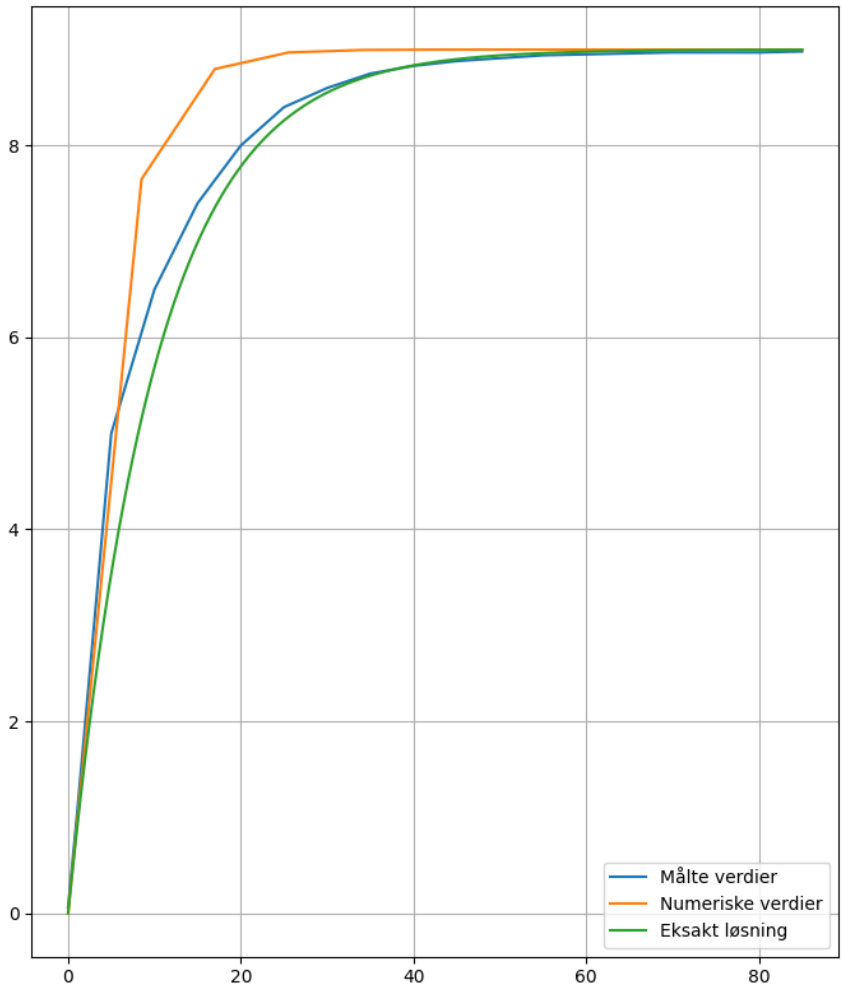
Dette er den utrolig avanserte RC-kretsen som ble brukt i forsøket. Komponentene består av et 9Volts batteri, en motstand på 100kOhm og en kondensator, med kapasitans på 100 mikroFahrad



For å løse oppgaven, målte jeg spenningen over kondensatoren i 85s, men trengte flere forsøk for å få gyldige verdier i starten av målingen. Deretter simulerte jeg forsøket med eulers metode og ved å løse differensiallikningen i python.

```python
import numpy as np
import matplotlib.pyplot as plt

målt_V = [0.06,1.5,2.7,4,5,6,7.4,8,8.4,8.6,8.7,8.75,8.83,8.88,8.91,8.94,8.95,8.96,8.97,8.97,8.97,8.98]
målt_T = [0,1,2,3,4,5,10,15,20,25,30,35,40,45,50,55,60,65,70,75,80,85]
#Målte verdier for Volt over en kondensator over tid.
#Målte flere ganger for å oppnå høyere nøyaktighet, siden den stiger raskt i starten

R = 100000                              #R = 100kOhm
C = 100 * 10**(-6)                      #C = 100 mikroFahrad

n = 85/10                               #Antall steg for plotting og numerisk løsning av difflikning

def Volt(t):
    if t == 0:
        return 0
    v_der = 9/(R*C) - Volt(t-n)/(R*C)   #Bruker rekursjon for å finne en tilnærming til den deriverte
    return Volt(t-n) + v_der * n        #Eulers metode

def løsning_V(t):
    return -9 * (np.exp(-1/(R*C)*t)-1)  #Eksakt løsning av difflikningen: v'(t) = -1/RC * v(t) + 9/RC

numerisk_Volt = []
numerisk_Tid = []
t = 0

while t <= 85:
    numerisk_Tid.append(t)              #Legger til t-verdier (Når maksimum recursion depth, ved større n)
    numerisk_Volt.append(Volt(t))       #Og V-verdier
    t+=n

løsning_Tid = []
løsning_Volt = []
t=0

while t<=85:
    løsning_Tid.append(t)               #Legger til t-verdier med mer større nøyaktighet
    løsning_Volt.append(løsning_V(t))   #Legger til eksakt-verdien til løsningen av diff-likn.
    t+=.001

plt.plot(målt_T,målt_V, label = "Målte verdier")
plt.plot(numerisk_Tid, numerisk_Volt, label = "Numeriske verdier")
plt.plot(løsning_Tid,løsning_Volt, label = "Eksakt løsning")
plt.legend()
plt.grid()
plt.show()
```

Da endte jeg opp med 3 grafer, se illustrasjonen under, som var tilnærmet like. Og som stemmer med målingene. På grunn av måten jeg løste den numeriske tilnærmingen fikk jeg liten nøyaktighet, men jeg hadde ikke brukt rekursjon på den måten i Python før og hadde lyst til å eksperimentere med det.



Det overrasker meg litt at den eksakte løsningen nesten konstant ligger under de målte verdiene. Jeg hadde i utgangspunktet trodd at kretsen ikke ville yte optimalt, og at den skulle ligge under den eksakte løsningen. En forklaring på dette kan være at målingen heller ikke var helt optimal og at jeg skrev inn den målte spenningen på et litt tidligere tidspunkt enn den egentlig skulle ligge.