1 SPØRSMÅL; RUNDE 5

Jeg har prøv å kjøre simuleringer med dataen jeg har for øyeblikket. Resultatene er vist lengre nede og ser ikke helt fornuftige ut. Dette dokumentet inneholder derfor en kjapp gjennomgang av hva jeg har gjort, for å eksludere at jeg ikke har missforstått noe. All data som er brukt i denne gjennomkjøringen er fra de artiklene du sendte meg tidligere (til jorgevag@stud.ntnu.no).

1.1 DATA HENTET UT FRA ARTIKLENE

I figurene 1.1, 1.2, 1.3, er grafene funnet i artiklene vist i mot den dataen som faktisk ble hentet ut. I Figur 1.1 er det blandt annet mest overlapp og størst sjanse for feil. Kan også hende at grafene i den uthentede dataen krysser hverandre feil i de mest overlappede regionene.

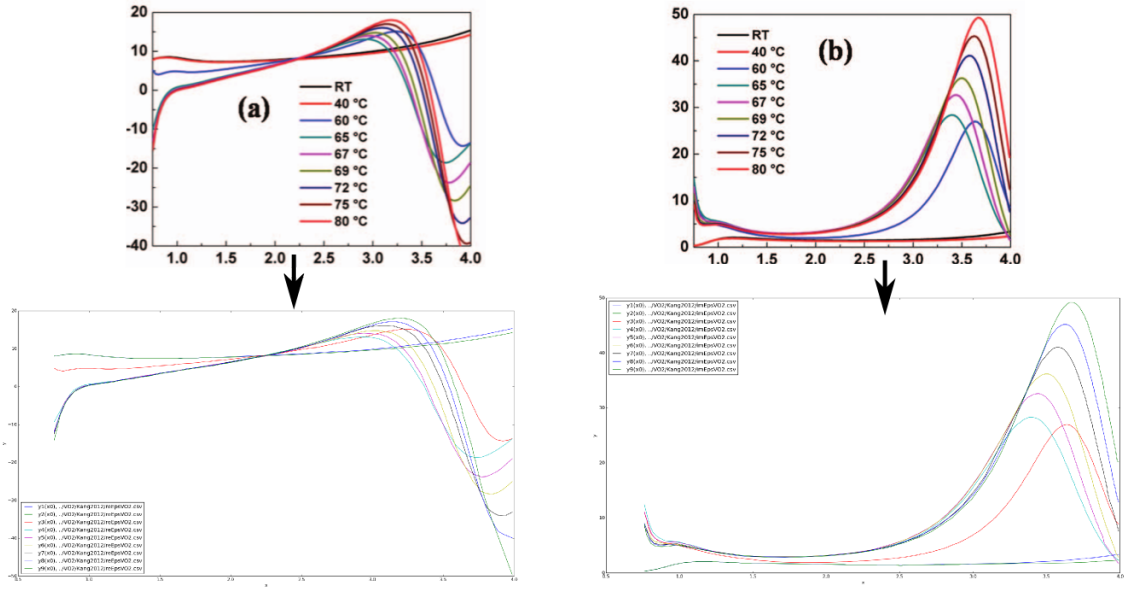

Figure 1.1: Extraction of data from Kang.

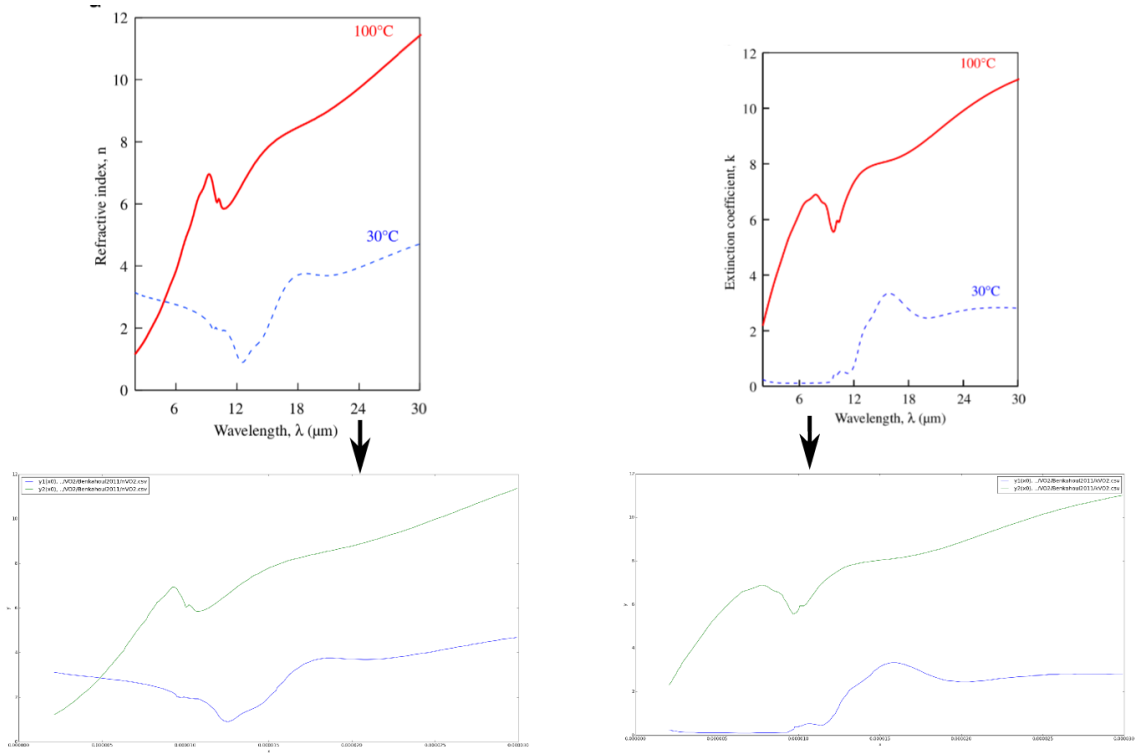

Figure 1.2: Extraction of data from Benkahoul.

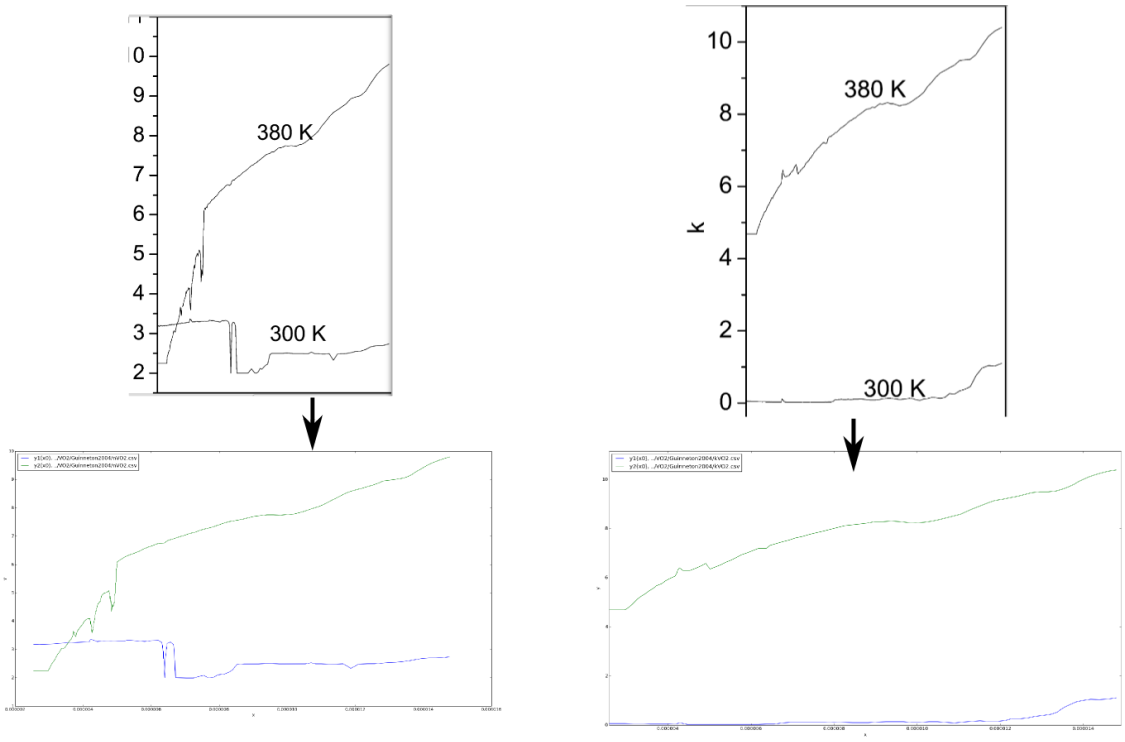

Figure 1.3: Extraction of data from Guinneton

1.2 KONVERTERING AV DATA

For å konvertere dataen til ekvidistansert n,k-data, brukte jeg samme metode som det interpoleringsprogrammet du hjap meg med tidligere. Uavhengig om hva enhetene på dataen var, valgte jeg å intrapolere til ekvidistansert data før jeg konverterte til kompleks brytningsindeks. Jeg brukte følgende kode for å konvertere den intrapolerte dataen, som her er gitt ved **re**, **im** (som kan være permittivitet ε eller brytningsindeks $\hat{n}$):

```
# If the input data is the permittivity/dielectric function,
# we have to convert to the refractive indicies n,k:
if( isPermittivity ): # data is permittivity
    import scipy.constants
    epsilon0 = scipy.constants.epsilon_0
    # convert from permittivity to refractive index n and absorbtion coeff k:
    absEpsilon = numpy.sqrt( re**2 + im**2)
    n = numpy.sqrt( (absEpsilon + re)/2.0*epsilon0 )
    k = numpy.sqrt( (absEpsilon - re)/2.0*epsilon0 )
else: # the real data should be n, and the imaginary data should be k:
    n = re
    k = im


# convert the x-values to the correct output units: unit=1->x[eV] or unit=2->x[micrometers]
if( unit == 1 ): # eV
    pass # Do nothing
elif(unit == 2): # micrometers
    pass # Do nothing
elif(unit == 3): # nm
    x_min = x_min*(10**(-3))
    x_max = x_max*(10**(-3))
    unit = 2
```

```
elif(unit == 4): # m
    x_min = x_min*(10**(6))
    x_max = x_max*(10**(6))
    unit = 2

#write to file
firstLine = str(unit)+'\t'+str(x_min)+'\t'+str(x_max)+'\t'+str(dataPoints)
numpy.savetxt(ofile, numpy.column_stack((n,k)), delimiter='\t', header=firstLine, comments='')
```

For å regne om til n og k, antar jeg at dataen er gitt som $\hat{\varepsilon} = \hat{\varepsilon}_r \varepsilon_0$ og bruker:

$$n = \sqrt{\frac{|\hat{\varepsilon}_r| + \varepsilon_r}{2}} = \sqrt{\frac{|\hat{\varepsilon}| + \varepsilon}{2\varepsilon_0}} \tag{1.1}$$

$$\kappa = \sqrt{\frac{|\hat{\varepsilon}_r| - \varepsilon_r}{2}} = \sqrt{\frac{|\hat{\varepsilon}| - \varepsilon}{2\varepsilon_0}}. \tag{1.2}$$

Før jeg leser inn dataen til GRANFILM, endrer jeg på 'Energy_Range' i '.sif'-filen ved å lese første linje fra '.nk'-filen. Om verdiene er gitt i μm (unit=2), konverterer jeg med følgende kode (før det skrives inn i '.sif'-filen):

```
# Get 'unit' and energy interval [x1,x2]:
nkfileInfo = nkfileLine1.split()
unit = int(nkfileInfo[0])
x1 = float(nkfileInfo[1])
x2 = float(nkfileInfo[2])

# Check unit:
if( unit == 1 ): # eV
    Emin = x1
    Emax = x2

if( unit == 2 ): # micrometer (wavelength)
    #then convert to eV:
    import scipy.constants as sc
    h = sc.h/sc.e # (planck's constant[J s])/(elementary charge) = planck[eV s]
    c = sc.c # speed of light

    Emin = h*c/( x2*(10**(-6)) )
    Emax = h*c/( x1*(10**(-6)) )
    #ref: http://www.pveducation.org/pvcdrom/properties-of-sunlight/energy-of-photon
```

Deretter kjøres GRANFILM med'.sif'-filen som input, som igjen kaller tilhørende '.nk'-fil.

1.3 TEST-SIMULERING

Dataen nevnt ovenfor ble konvertert til kompleks brytningsindeks og matet inn i GranFilm. Når all dataen ble simulert (Fig.1.4), så det ut til at den delte seg inn i 3 "regioner", of jeg frykter dette skyldes ulikheter fra de 3 artiklene. En nærmere titt på den flate delen til høyre i Fig.1.4, er vist i Fig.1.5, og jeg får på følelsen av at noe er veldig feil. Til venstre, Fig. 1.6, ser derimot bedre ut, men vet ikke helt om resultatet virker fornuftig eller ikke.
Sist ble det gjort en simulering for $T = 300$K og $T = 380$K med dataen fra Guinnetion. Refleksjon og Transmisjon for p-polarizert lys er gitt i Fig.1.7 og Fig.1.8.

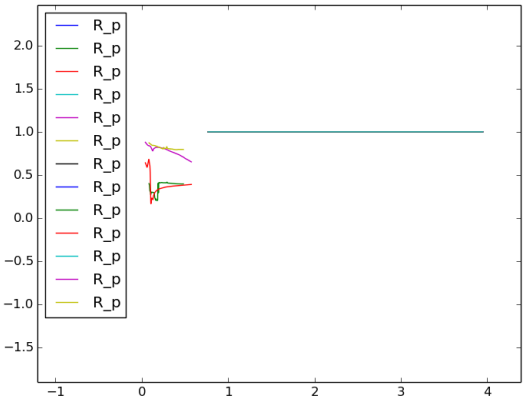

Figure 1.4: Plot of $R_p = \sqrt{Re(r_p)^2 + Im(r_p)^2}$ for all extracted temperature-data. Looks like it has formed three different regions shifted from one another.

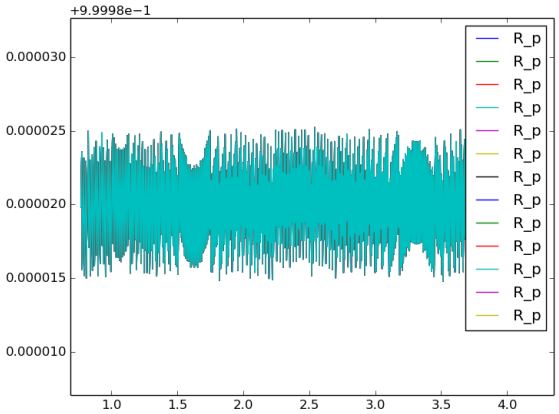

Figure 1.5: If I zoom in on the flat part (in R_p-plot on previous figure), it looks really messy.

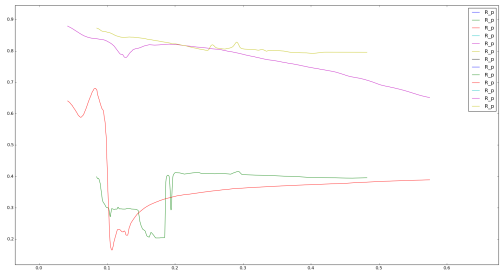

Figure 1.6: The other data in the R_p-plot.

Jeg er på dette tidspunktet veldig usikker på hvordan jeg finner ut om resultatene er fornuftig eller feil. Med tanke på hvordan Fig.1.4 og Fig.1.5 ser ut, vil jeg uansett anta at noe er fryktelig feil. **Er du enig i at dette ser feil ut og har**

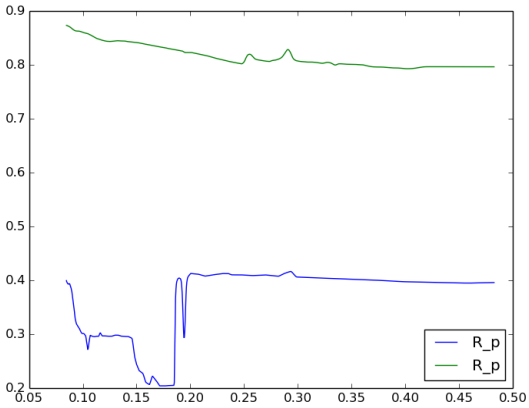

Figure 1.7: R_p from Guinneton. The blue line is for $T = 300K$; The green line is for $T = 380K$.

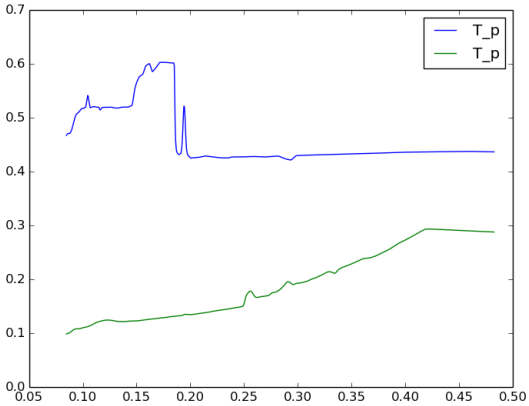

Figure 1.8: T_p from Guinneton. The blue line is for $T = 300K$; The green line is for $T = 380K$.

du noen forslag til hva som kan være feil?