1 SPØRSMÅL RUNDE 1

Jeg prøver å forstå mer detaljert hvordan jeg skal få lest inn dataen til GranFilm programmet. Formatet til ".nk"-filene i `GranFilm/Sopra/DataBase` ser slik ut (her mgo.nk):

```
1            0.65                    10              400
     1.70969699     0.00000011
     1.71065583     0.00000011
     1.71239613     0.00000011
```

og ser ut til å ha reell og imaginær refraksjonsindeks forholdsvis på kolonne 1 og 2. I tillegg virker det som om rad èn består av de 4 verdiene:

- unit (om refraksjonsindeksen er funksjon av energi (eV) eller bølgelengde);
- x startverdi (energi/bølgelengde);
- x sluttverdi
- antall datapunkter.

Basert på dette, i tillegg til modulen fra source-koden til GranFilm gitt nedenfor, virker det som om reell og imaginær verdi er gitt for samme x-verdi (energi/bølgelengde), der x-verdiene har konstant steglengde. (For eksempel: for mgo.nk filen ville vi hatt $dx = \frac{10-0.65}{400-1}$).

Jeg har funnet et program som lar meg hente ut dataen fra grafer i artikklene. Prolemet er at dataen for reell og imaginær refraksjonsindeks er for forskjellige x-verdier, der x-verdiene ikke har konstant steglengde.

Spørsmålene er derfor:

1. Er det riktig det jeg har forstått ovenfor?
2. Trenger jeg skrive om inputfilene (filene med thermochrom-dataen) slik at reell og imaginær verdi (forholdsvis n,k) svarer til samme x-verdi, der x-verdiene ligger med avstand dx i fra hverandre?
   (Jeg tenkte å kanskje skrive en funksjon som tar inn to filer, colonner og "unit"-verdi, leser inn n og k dataen fra filene, intrapolerer for felles, nye x-verdier og skriver til en ny fil.)
3. Videre: noen grafer er gitt ved permittivitet ε, istedenfor refraksjonsindex $\mathbf{n}$. Da er det vel bare å regne om ved bruk av imaginær permittivitet? Jeg fant noen formlet og lurte på om dette er riktig: fant at

$$\boldsymbol{\varepsilon} = \frac{c^2\varepsilon_0}{\omega^2\frac{\mu}{mu_0}}\left(k^2 - \frac{\alpha^2}{4}\right) + i\left(\frac{c^2\varepsilon_0}{\omega^2\frac{\mu}{\mu_0}}k\alpha\right)$$

   (k er hær bølgetallet), hvor

$$\boldsymbol{n} = k - i\frac{\alpha}{2}, \qquad\qquad \boldsymbol{n} = \frac{c}{\omega}\boldsymbol{k}^*.$$

   Bruker at $\mu = \mu_0$ og får

$$\mathrm{Re}(n) = \frac{\mathrm{Im}(\varepsilon)}{2\varepsilon_0}$$
$$\mathrm{Im}(n) = \sqrt{\frac{\mathrm{Re}(\varepsilon)}{\varepsilon_0} - \left(\frac{\mathrm{Im}(\varepsilon)}{2\varepsilon_0}\right)^2}.$$

Til slutt vil jeg bare si at framgangen går sakte, og begynner å bli litt stresset mtp tiden. Lurte på hvor mye du hadde forestilt deg at jeg burde få til? Foreløpig, har jeg klart å lese ut VO_2-dataen fra de fire artiklene du sendte meg. Dataene trenger trolig mer arbeid før de mates inn i GranFilm som beskrevet ovenfor. Der ligger VO_2 data for 10-12 forskjellige temperaturer.
I kildekoden til `GranFilm` fant jeg modulen `dielectric_function_module.f90`. Forstår det slik at

```
Function Locate(xx,x)
```

finner indeksen i xx til elementet som ligger nærmest verdien x? Tenkte å kunne bruke funksjonen eller skrive den om, og bruke den i intrapolering om jeg må skrive om input-filene.

```
!-------------------------------------------------------------!
Subroutine Get_Dielectric_Function(energy,epsilon,material,path)
!Subroutine dielectric_constants(energy,epsilon,material,path)
!-------------------------------------------------------------!
  Use SFL_Logical_Units,        only : SFL_Get_Free_Unit
  Use Error_Module,             only : Error_Failure
  Implicit None
  Real(wp)                            :: energy(:)
  complex(wp )                        :: Epsilon(:)
  Character(len=*)                    :: material
  Character(len=*)                    :: path
  ! --- Local
  Character(len=*), parameter         :: routine = "Get_Dielectric_Function"
  !complex(wp ),Parameter              :: im=(0._wp,1._wp)
  Character(len=250)                  :: filename, str
  Logical                             :: exi
  integer                             :: lines,start,i, ifile
  Real(wp),      Allocatable          :: x(:)
  complex(wp ), Allocatable           :: y(:)
  Real(wp)                            :: tmp(2),x1,x2,dx
  integer                             :: unit, istat
  complex(wp )                        :: slope


  ! Opens the relevant data file for the given material
  ! Notice that the data-files contains the values for the
  ! index of refraction, n,k extacted from the data base of
  ! SOPRA.
  ! Therefore "epsilon = epsilon**2" at the bottom of this routine
  filename = Trim(Adjustl(path))//'/'//Trim(Adjustl(material))//'.nk'
  Inquire(file=Trim(Adjustl(filename)),exist = exi)
  If( .NOT. exi ) Then
     write(str,*) trim(adjustl(filename))
     str = "File = " // trim(adjustl(str)) // " non existing"
     Call Error_Failure(routine, trim(adjustl(str)) )
  Endif
  call SFL_Get_Free_Unit( ifile )
  Open(unit=ifile,file=filename,status='old')
  Read(unit=ifile,fmt=*) unit,x1,x2,lines
  dx = (x2 - x1)/(lines-1)

  Allocate( x(lines), y(lines) )
  Select Case (unit)
  Case(1)
     ! Unit = Electron Volt (eV)
     Do i=1,lines
        Read(unit=ifile,fmt=*) tmp(1), tmp(2)
        x(i) = x1 + (i-1)*dx
        y(i) = tmp(1)+imu*tmp(2)
     Enddo
  Case(2)
     ! Unit = WaveLength (microm)
     Do i=lines,1,-1
        Read(unit=ifile,fmt=*) tmp(1), tmp(2)
        x(i) = x1 + (lines-i)*dx
        y(i) = tmp(1)+imu*tmp(2)
     Enddo
     x(:) = 1.243_wp/x(:) ! Conversion microm-->eV
  End Select
```

```
  Close(unit=ifile)

  ! --- Do the interpolation
  Do i=1,Size(energy,1)
     start=locate(x(:),energy(i))
     If((start==0).Or.(start==lines)) Then
        write(str,"('Energy not in range : ',f5.2,' for i=',i3)") energy(i),i
        call Error_Failure( routine, trim(adjustl(str)))
     Endif
     ! Linear interpolation
     slope = (y(start+1)-y(start))/(x(start+1)-x(start))
     Epsilon(i) = y(start) + slope*(energy(i)-x(start))
     !            Write(unit=67,*) energy(i),Real(epsilon(i)),Aimag(epsilon(i))
  Enddo

  ! --- Calculates the dielectric constant (from the refraction index)
  epsilon = epsilon**2
  Deallocate(x,y,stat=istat)

contains

  Function Locate(xx,x)
    Implicit None
    Real(wp), Dimension(:), Intent(In)  :: xx
    Real(wp), Intent(In)                :: x
    Integer                             :: locate
    Integer                             :: n,jl,jm,ju
    Logical                             :: ascnd

    n=Size(xx)
    ascnd = (xx(n) >= xx(1))
    jl=0
    ju=n+1
    Do
       If(ju-jl <= 1) exit
       jm=(ju+jl)/2
       If(ascnd .eqv. (x >= xx(jm))) Then
          jl=jm
       Else
          ju=jm
       Endif
    Enddo
    If(x == xx(1)) Then
       locate=1
    Else If(x == xx(n)) Then
       locate=n-1
    Else
       locate=jl
    Endif

  End Function Locate


End Subroutine Get_Dielectric_Function
!-------------------------------------!
```