```
************************************************************************
*     This program fits optical data to a causal dielectric function
*     The chi-squared function is weighted with sigma**2 factors for
*     each data point.
*     From fit.ini it searches:
***** type of experiment (Rs (1), Rpiso (2), Rpani (3), Trans (4)
*****    grazing reflection on film+substrate (5)
*****    RsAsymOsci(6) SigAsymOsci(7))
***** geometrical and optical parameters of the experiment
***** required precision per data point of fitresult
***** logical (.T. or .F.) T/F if data table contains (or not)
*****               a third column with experimental errorbars
************************************************************************
      program fitdat
      parameter(nnpar=77,nndat=21234,nten=10)
      integer ndat,npar,nfit,noscix,nosciz,lista(nnpar),nfr,iter,nrange
     *  ,datan,nosc(3),noscz(3)
      real teta,tpi,sn2,dfl,dsb,pre,pim,x(nndat),y(nndat),s(nndat)
     *  ,param(nnpar),chilim,chisq,xmn(nten),xmx(nten),frmn,frmx
     *  ,xold,yold,datax(nndat),datay(nndat)
      complex psbstr
      logical errflg,logpar(nnpar)
      character*40 fldat,flpar,flfit,fleps
      character*1 a,b,epsmod
      common/INtet/sn2
      common/INnosc/noscix,nosciz,nosc,noscz
      common/INtran/dfl,dsb,psbstr
      common/extra/noscis,teta
      common/epsmo/epsmod
      external rfs,rfpani,rfpiso,trans,rfpans,sigma,rfps,psidel,ep1ep2
*-----------------------------------------------------------------------
      write(*,*) '                                                      '
      write(*,*) '                                                      '
      write(*,*) '*********   HI   ***********'
      write(*,*)
      write(*,*) '**  I am your Optics Pal  **'
      write(*,*)
      write(*,*) 'Version 1.1 by Dirk van der Marel, May 1996.'
      write(*,*) ' Update 1.7 by Diana Dulic'
      write(*,*) ' Update 2.x by Markus Gr\"uninger, Nov.1998.'
      write(*,*) ' Update 2.2 by Dirk van der Marel, Dec.1998.'
      write(*,*) ' Update 2.3 by Dirk van der Marel, Feb.1999.'
      write(*,*) ' Update 2.4 by Dirk van der Marel, Apr.1999.'
      write(*,*)
      write(*,*)
1     write(*,*) '********  MAIN MENU  *******'
      write(*,*)
      write(*,*) 'Choose one of the following options:                '
      write(*,*) '1: Read optpal.data.in, fit.ini and parameter files'
      write(*,*) '   and fit the data'
      write(*,*) '2: Get extended help'
      write(*,*) '3: exit'
      read(*,'(a1)') b
      if (b.eq.'2') then
        call help
        goto 1
      endif
      if (b.eq.'1') then
```

```
      write(*,*) 'writing optpal.fit.old'
      open(10,file='optpal.fit.out')
      open(20,file='optpal.fit.old')
      do 10 i=1,nndat
        read(10,*,end=11) xold, yold
        write(20,*) xold, yold
10    continue
11    close(20)
      close(10)
      write(*,*) 'reading fit.ini'
      open(10,file='fit.ini')
      read(10,'(a1)') a
      read(10,'(a1)') epsmod
      if ((a.eq.'1').or.(a.eq.'2').or.(a.eq.'3').or.(a.eq.'5')
     *     .or.(a.eq.'7').or.(a.eq.'8')) then
        read(10,*) teta
        tpi=8.*atan(1.)
        sn2=sin(teta*tpi/360.)**2
      endif
      if ((a.eq.'4').or.(a.eq.'5')) read(10,*) dfl,dsb,pre,pim
      psbstr=cmplx(pre,pim)
      read(10,*) chilim
      read(10,*) errflg
      read(10,*) nrange
      do 20 i=1,nrange
       read(10,*) xmn(i),xmx(i)
20    continue
      read(10,*) frmn,frmx,nfr
      read(10,*) iter
      close(10)
      write(*,*) 'reading parameters'
      call inpar(a,logpar,param,lista,nnpar,npar,nfit,noscis)
      write(*,*) 'reading optpal.data.in'
      call indat(errflg,x,y,s,nndat,ndat,xmn,xmx,nrange,datax,
     *datay,datan)
      write(*,*) 'starting the fit with function ',a
      if (a.eq.'1') call dofit(a,x,y,s,frmn,frmx,nfr,rfs
     *              ,nndat,ndat,param,lista,nnpar,npar,nfit
     *              ,chilim,chisq,iter,datax,datay,datan)
      if (a.eq.'2') call dofit(a,x,y,s,frmn,frmx,nfr,rfpiso
     *              ,nndat,ndat,param,lista,nnpar,npar,nfit
     *              ,chilim,chisq,iter,datax,datay,datan)
      if (a.eq.'3') call dofit(a,x,y,s,frmn,frmx,nfr,rfpani
     *              ,nndat,ndat,param,lista,nnpar,npar,nfit
     *              ,chilim,chisq,iter,datax,datay,datan)
      if (a.eq.'4') call dofit(a,x,y,s,frmn,frmx,nfr,trans
     *              ,nndat,ndat,param,lista,nnpar,npar,nfit
     *              ,chilim,chisq,iter,datax,datay,datan)
      if (a.eq.'5') call dofit(a,x,y,s,frmn,frmx,nfr,rfpans
     *              ,nndat,ndat,param,lista,nnpar,npar,nfit
     *              ,chilim,chisq,iter,datax,datay,datan)
      if (a.eq.'6') call dofit(a,x,y,s,frmn,frmx,nfr,sigma
     *              ,nndat,ndat,param,lista,nnpar,npar,nfit
     *              ,chilim,chisq,iter,datax,datay,datan)
      if (a.eq.'7') call dofit(a,x,y,s,frmn,frmx,nfr,rfps
     *              ,nndat,ndat,param,lista,nnpar,npar,nfit
     *              ,chilim,chisq,iter,datax,datay,datan)
      if (a.eq.'8') call dofit(a,x,y,s,frmn,frmx,nfr,psidel
```

```
     *               ,nndat,ndat,param,lista,nnpar,npar,nfit
     *               ,chilim,chisq,iter,datax,datay,datan)
        if (a.eq.'9') call dofit(a,x,y,s,frmn,frmx,nfr,ep1ep2
     *               ,nndat,ndat,param,lista,nnpar,npar,nfit
     *               ,chilim,chisq,iter,datax,datay,datan)
        write(*,*) 'writing output (optpal.fit.out etc.)'
        call outpar(a,logpar,param,nnpar,npar,noscis)
        goto 1
       endif
       if (b.eq.'3') goto 100
       write(*,*)
       write(*,*) '    ************  INVALID INPUT   ***********'
       write(*,*)
       goto 1
100    end
*********************************************************************


*********************************************************************
***** Read the experimental data from file 'optpal.data.in'
*********************************************************************
       subroutine indat(errflg,x,data,err,nndat,ndat,xmn,xmx,nrange,
      *datax,datay,datan)
       real x(nndat),data(nndat),err(nndat),xmn(nrange),xmx(nrange)
       real xx,yy,ee,datax(nndat),datay(nndat)
       integer i,j,ndat,nrange,datan
       logical errflg,flg
*--------------------------------------------------------------------
       open(25,file='optpal.data.in')
       ndat=0
       datan=0
       do 10 i=1,nndat
        if (errflg) then
         read(25,*,end=11) xx,yy,ee
         datax(i)=xx
         datay(i)=yy
         datan=datan+1
         flg=.false.
         do 5 j=1,nrange
          if ((xx.le.xmx(j)).and.(xx.ge.xmn(j))) flg=.true.
5        continue
         if (flg) then
          ndat=ndat+1
          x(ndat)=xx
          data(ndat)=yy
          err(ndat)=ee
         endif
        else
         read(25,*,end=11) xx,yy
         datax(i)=xx
         datay(i)=yy
         datan=datan+1
         flg=.false.
         do 7 j=1,nrange
          if ((xx.le.xmx(j)).and.(xx.ge.xmn(j))) flg=.true.
7        continue
         if (flg) then
          ndat=ndat+1
```

```
          x(ndat)=xx
          data(ndat)=yy
          err(ndat)=1.
        endif
       endif
10    continue
11    close(25)
      return
      end
*********************************************************************

*********************************************************************
***** Read Infile(s) 'optpal.param.x.in' and 'optpal.param.z.in'
***** with 'special' startup parameters and fitflags.
***** If (datyp.eq.3) (p-polarized reflectivity on an anisotropic
***** medium) a second set of parameters needs to be read.
***** If (datyp.eq.5) (p-polarized reflectivity on an anisotropic
***** medium on an isotropic substrate) two more sets of parameters need
***** to be read.
***** On input it reads from file optpal.param.x.in/optpal.param.z.in:
*****   epsinf, flag(epsinf)
*****   No. of 'special' Oscillator terms parallel to surface (x):
*****   wt, wp, gamma, flag(wt), flag(wp), flag(gamma)
*****   wmn, wmx, No. of oscillators. Only S is varied, log-grid is used
*
*     In case of asymmetric Drude-Lorentz model (slots '6' and '7') it reads
*       wt,wp,gamma,theta, flag(wt), flag(wp), flag(gamma), flag(theta)
*********************************************************************
      subroutine inpar(datyp,logpar,param,lista,nnpar,npar,nfit,noscis)
*      parameter(mmpar=77)
c*****N.B.: MAKE SURE THAT: mmpar <= nnpar
      integer lista(nnpar)
      real param(nnpar)
      logical logpar(nnpar)
      integer i,npar,nfit,noscix,nosciz,noscis,nosc(3),noscz(3)
      character*1 datyp,epsmod
      common/INnosc/noscix,nosciz,nosc,noscz
      common/epsmo/epsmod
*---------------------------------------------------------------------
      npar=0
      open(15,file='optpal.param.x.in')
      if (epsmod.eq.'1') then
        call rdpar(logpar,param,nnpar,npar,noscix)
      endif
      if (epsmod.eq.'2') then
        call rdparasym(logpar,param,nnpar,npar,noscix)
      endif
      if (epsmod.eq.'3') then
        call rdparlayer(logpar,param,nnpar,npar,nosc)
        noscix=nosc(1)+nosc(2)+nosc(3)
      endif
      if (epsmod.eq.'4') then
        call rdpar4(logpar,param,nnpar,npar,nosc)
        noscix=nosc(1)+nosc(2)
      endif
      if (epsmod.eq.'5') then
        call rdpar5(logpar,param,nnpar,npar,noscix)
      endif
```

```
      close(15)
******if data type is p-pol, aniso, read eps_z input parameters:
      if ((datyp.eq.'3').or.(datyp.eq.'5')) then
        open(15,file='optpal.param.z.in')
        if (epsmod.eq.'1') then
          call rdpar(logpar,param,nnpar,npar,nosciz)
        endif
        if (epsmod.eq.'2') then
          call rdparasym(logpar,param,nnpar,npar,nosciz)
        endif
        if (epsmod.eq.'3') then
          call rdparlayer(logpar,param,nnpar,npar,noscz)
          nosciz=noscz(1)+noscz(2)+noscz(3)
        endif
        if (epsmod.eq.'4') then
          call rdpar4(logpar,param,nnpar,npar,noscz)
          nosciz=noscz(1)+noscz(2)
        endif
        if (epsmod.eq.'5') then
          call rdpar5(logpar,param,nnpar,npar,nosciz)
        endif
        close(15)
      endif
      if (datyp.eq.'5') then
        open(15,file='optpal.param.s.in')
        call rdpar(logpar,param,nnpar,npar,noscis)
        close(15)
      endif
***** define the list of variational parameters:
      nfit=0
      do 35 i=1,npar
      if (logpar(i)) then
       nfit=nfit+1
       lista(nfit)=i
      endif
35    continue
      return
      end
*************************************************************
***** subprogram 1(5) of inpar (for Drude-Lorentz model)
*************************************************************
      subroutine rdpar(logpar,param,nnpar,npar,noscil)
      logical logpar(nnpar)
      real param(nnpar)
      integer i,npar,noscil,nwt
      real epsinf,wp,g,wt
      logical fe,fp,ft,fg
*-------------------------------------------------------------------------
      read(15,*) epsinf,fe
      param(npar+1)=epsinf
      logpar(npar+1)=fe
      npar=npar+1
      read(15,*) noscil
      do 25 i=1,noscil
       read(15,*) wt,wp,g,ft,fp,fg
       param(npar+1)=wt
       logpar(npar+1)=ft
       param(npar+2)=wp
```

```
       logpar(npar+2)=fp
       param(npar+3)=sqrt(g)
       logpar(npar+3)=fg
       npar=npar+3
25     continue
       return
       end
*************************************************************
***** subprogram 2(5) of inpar (for asymmetric Drude-Lorentz model)
*************************************************************
       subroutine rdparasym(logpar,param,nnpar,npar,noscil)
       logical logpar(nnpar)
       real param(nnpar)
       integer i,npar,noscil,nwt
       real epsinf,wp,g,wt,theta
       logical fe,fp,ft,fg,ftheta
*-------------------------------------------------------------------
       read(15,*) epsinf,fe
       param(npar+1)=epsinf
       logpar(npar+1)=fe
       npar=npar+1
       read(15,*) noscil
       do 25 i=1,noscil
        read(15,*) wt,wp,g,theta,ft,fp,fg,ftheta
        param(npar+1)=wt
        logpar(npar+1)=ft
        param(npar+2)=wp
        logpar(npar+2)=fp
        param(npar+3)=sqrt(g)
        logpar(npar+3)=fg
        param(npar+4)=theta
        logpar(npar+4)=ftheta
        npar=npar+4
25     continue
       return
       end
*************************************************************
***** subprogram 3(5) of inpar (for two-layer Drude-Lorentz model)
*************************************************************
       subroutine rdparlayer(logpar,param,nnpar,npar,nos)
       logical logpar(nnpar)
       real param(nnpar)
       integer i,npar,nos(3),nwt,layer
       real epsinf,wp,g,wt,d1,d2
       logical fe,fp,ft,fg,fd1,fd2
*-------------------------------------------------------------------
       read(15,*) epsinf,fe
       param(npar+1)=epsinf
       logpar(npar+1)=fe
       npar=npar+1
       do 27 layer=1,3
       read(15,*) nos(layer)
       do 25 i=1,nos(layer)
        read(15,*) wt,wp,g,ft,fp,fg
        param(npar+1)=wt
        logpar(npar+1)=ft
        param(npar+2)=wp
        logpar(npar+2)=fp
```

```
       param(npar+3)=sqrt(g)
       logpar(npar+3)=fg
       npar=npar+3
25     continue
27     continue
       read(15,*) d1,d2,fd1,fd2
       param(npar+1)=d1
       param(npar+2)=d2
       logpar(npar+1)=fd1
       logpar(npar+2)=fd2
       npar=npar+2
30     continue
       return
       end
*************************************************************
***** subprogram 4(5) of inpar (for factorized 4-parameter-model)
*************************************************************
       subroutine rdpar4(logpar,param,nnpar,npar,nos)
       logical logpar(nnpar)
       real param(nnpar)
       integer i,npar,nos(3),nwt
       real epsinf,wt,wl,wp,g,as,b
       logical fe,ft,fl,fp,fg,fas,fb
*-------------------------------------------------------------------------
       read(15,*) epsinf,fe
       param(npar+1)=epsinf
       logpar(npar+1)=fe
       npar=npar+1
       read(15,*) nos(1)
       do 20 i=1,nos(1)
        read(15,*) wt,wp,g,b,ft,fp,fg,fb
        param(npar+1)=wt
        logpar(npar+1)=ft
        param(npar+2)=wp
        logpar(npar+2)=fp
        param(npar+3)=sqrt(g)
        logpar(npar+3)=fg
        param(npar+4)=b
        logpar(npar+4)=fb
        npar=npar+4
20     continue
       read(15,*) epsinf,fe
       param(npar+1)=epsinf
       logpar(npar+1)=fe
       npar=npar+1
       read(15,*) nos(2)
       do 25 i=1,nos(2)
        read(15,*) wt,wl,g,as,ft,fl,fg,fas
        param(npar+1)=wt
        logpar(npar+1)=ft
        param(npar+2)=wl
        logpar(npar+2)=fl
        param(npar+3)=sqrt(g)
        logpar(npar+3)=fg
        param(npar+4)=as
        logpar(npar+4)=fas
        npar=npar+4
25     continue
```

```
      return
      end
*****************************************************************************
************************************************************
************************************************************
***** subprogram 5(5) of inpar (for non fermi liquid model)
************************************************************
      subroutine rdpar5(logpar,param,nnpar,npar,noscil)
      logical logpar(nnpar)
      real param(nnpar)
      integer i,npar,noscil,nwt
      real epsinf,wp,g,glo,ghi,wt,alfa
      logical fe,fp,fg,ft,fgl,fgh,fa
*-----------------------------------------------------------------------
      read(15,*) epsinf,fe
      param(npar+1)= epsinf
      logpar(npar+1)= fe
      read(15,*) alfa,wp,glo,ghi,fa,fp,fgl,fgh
      param(npar+2)=alfa
      logpar(npar+2)=fa
      param(npar+3)=wp
      logpar(npar+3)=fp
      param(npar+4)=sqrt(glo)
      logpar(npar+4)=fgl
      param(npar+5)=sqrt(ghi)
      logpar(npar+5)=fgh
      npar=npar+5
c       write(*,*) param(1),param(2),param(3),param(4),param(5)
      read(15,*) noscil
      do 25 i=1,noscil
       read(15,*) wt,wp,g,ft,fp,fg
       param(npar+1)=wt
       logpar(npar+1)=ft
       param(npar+2)=wp
       logpar(npar+2)=fp
       param(npar+3)=sqrt(g)
       logpar(npar+3)=fg
       npar=npar+3
c      write(*,*) param(6),param(7),param(8)
c       write(*,*)
c       write(*,*)
25    continue
        write(*,*) param(1),param(2),param(3),param(4),param(5)
     * ,param(6),param(7),param(8)
        write(*,*) logpar(1),logpar(2),logpar(3),logpar(4),logpar(5)
     * ,logpar(6),logpar(7),logpar(8)
      return
      end


*****************************************************************************
************************************************************
***** Fit the data
*********************************************************************
      subroutine dofit(datyp,x,y,s,frmn,frmx,nfr,myfunc,nndat
     * ,ndat,a,lista,nnpar,npar,nfit,chilm,chisq,iter,datax,datay
     * ,datan)
      parameter(mmpar=77)
```

```
      real x(nndat),y(nndat),xx,yy,s(nndat),a(nnpar)
      integer lista(nnpar),noscix,nosciz,nosc(3),noscz(3)
      integer i,ndat,npar,nfit,flag,nfr,iter,maxout,datan
      real ochisq,chisq,alamda,alamol,chilim,chilm,frmn,frmx
     *  ,dydp(mmpar),covar(mmpar,mmpar)
     *  ,alpha(mmpar,mmpar),beta(mmpar),datax(nndat),datay(nndat)
      character*1 datyp
      complex e2x,de2x(mmpar),e2z,de2z(mmpar),e3x,de3x(mmpar)
      common/INnosc/noscix,nosciz,nosc,noscz
      common/result/e2x,de2x
      common/resulz/e2z,de2z
      common/resuls/e3x,de3x
      external myfunc
*-----------------------------------------------------------------------
      chilim=ndat*(chilm**2)
      alamda=-1.
      flag=0
      alamol=alamda
      do 45 i=1,iter
        call mrqmin(x,y,s,ndat,a,dydp,npar,lista,nfit,
     *      covar,alpha,beta,npar,chisq,ochisq,alamda,myfunc)
        write(*,*) i,'  chi^2=',chisq ,'  ','prec/point=',
     *   sqrt(chisq/ndat)
       if (alamda.gt.alamol) then
        flag=flag+1
       else
        flag=0
       endif
       alamol=alamda
       if (chisq.lt.chilim) then
        write(*,*) 'Iteration interrupted. Reason: '
        write(*,*) chisq,' = chisq dropped below limit = ',chilim
        goto 46
       endif
       if (flag.gt.6+npar) then
        write(*,*) 'Iteration interrupted. Reason: '
        write(*,*) alamda,' = alamda increased six times '
        goto 46
       endif
45     continue
       write(*,*) 'Iteration interrupted. Reason: '
       write(*,*) 'number of iterations equals ', i
46     alamda=0
       call mrqmin(x,y,s,ndat,a,dydp,npar,lista,nfit,
     *    covar,alpha,beta,npar,chisq,ochisq,alamda,myfunc)
       open(22,file='optpal.fit.out')
       open(23,file='optpal.eps.x.out')
       open(13,file='sig.out')
       if ((datyp.eq.'3').or.(datyp.eq.'5')) then
         open(24,file='optpal.eps.z.out')
       endif
       if (datyp.eq.'5') open(26,file='optpal.eps.s.out')
       if (nfr.eq.-1) open(27,file='optpal.fitdiff.out')
       if ((nfr.eq.0).or.(nfr.eq.-1)) then
          maxout=datan-1
       else
          maxout=nfr
       endif
```

```
      do 60 i=0,maxout
       if ((nfr.eq.0).or.(nfr.eq.-1)) then
         xx=datax(i+1)
       else
         xx=frmn+i*(frmx-frmn)/nfr
       endif
       call myfunc(xx,a,yy,dydp,nnpar)
       if (nfr.eq.-1) write(27,*) xx,datay(i+1)-yy
       if (abs(xx).gt.0) write(22,*) xx,yy
       if (xx.gt.0) write(23,*) xx,real(e2x),aimag(e2x)
       if (xx.gt.0) write(13,*) xx, aimag(e2x)*xx*0.5*0.0333795
       if (((datyp.eq.'3').or.(datyp.eq.'5')).and.(xx.gt.0)) then
         write(24,*) xx,real(e2z),aimag(e2z)
       endif
       if ((datyp.eq.'5').and.(xx.gt.0))
     * write(26,*) xx,real(e3x),aimag(e3x)
60    continue
      close(13)
      close(22)
      close(23)
      if ((datyp.eq.'3').or.(datyp.eq.'5')) close(24)
      if (datyp.eq.'5') close(26)
      if (nfr.eq.-1) close(27)
      return
      end
*********************************************************************

*********************************************************************
***** write outfile(s) with final fit-parameters.
*****  If (datyp.eq.3) (p-polarized reflectivity on an anisotropic
*****  medium) a second set of parameters is produced.
***** If (datyp.eq.5) (p-polarized reflectivity on an anisotropic
***** medium on an isotropic substrate) two more sets of parameters
***** are produced.
***** On output it writes to file optpal.param.x.out/optpal.param.z.out
*****   epsinf, flag(epsinf)
*****   # of 'special' Oscillator terms parallel to surface (x):
*****    wt, wp, gamma, flag(wt), flag(wp), flag(gamma)
*****   wmn, wmx, # of oscillators. Only S is varied, log-grid is used
*
*     In case of asymmetric Drude-Lorentz model (slots '6' and '7') it writes
*       wt,wp,gamma,theta, flag(wt), flag(wp), flag(gamma), flag(theta)
*********************************************************************
      subroutine outpar(datyp,logpar,param,nnpar,npar,noscis)
      parameter(mmpar=77)
c*****N.B.: MAKE SURE THAT: mmpar = nnpar
      real param(nnpar)
      integer i,npar,noscix,nosciz,noscis,nosc(3),noscz(3)
      logical logpar(nnpar)
      character*1 datyp,epsmod
      common/INnosc/noscix,nosciz,nosc,noscz
      common/epsmo/epsmod
*********************************************************
      npar=0
      open(15,file='optpal.param.x.out')
      if (epsmod.eq.'1') then
        call wrpar(logpar,param,nnpar,npar,noscix)
      endif
```

```
      if (epsmod.eq.'2') then
        call wrparasym(logpar,param,nnpar,npar,noscix)
      endif
      if (epsmod.eq.'3') then
        call wrparlayer(logpar,param,nnpar,npar,nosc)
      endif
      if (epsmod.eq.'4') then
        call wrpar4(logpar,param,nnpar,npar,nosc)
      endif
      if (epsmod.eq.'5') then
        call wrpar5(logpar,param,nnpar,npar,noscix)
      endif
      close(15)
******if data type is p-pol, aniso, write eps_z output parameters:
      if ((datyp.eq.'3').or.(datyp.eq.'5')) then
        open(15,file='optpal.param.z.out')
        if (epsmod.eq.'1') then
          call wrpar(logpar,param,nnpar,npar,nosciz)
        endif
        if (epsmod.eq.'2') then
          call wrparasym(logpar,param,nnpar,npar,nosciz)
        endif
        if (epsmod.eq.'3') then
          call wrparlayer(logpar,param,nnpar,npar,noscz)
        endif
        if (epsmod.eq.'4') then
          call wrpar4(logpar,param,nnpar,npar,noscz)
        endif
        if (epsmod.eq.'5') then
          call wrpar5(logpar,param,nnpar,npar,nosciz)
        endif
        close(15)
      endif
      if (datyp.eq.'5') then
        open(15,file='optpal.param.s.out')
        call wrpar(logpar,param,nnpar,npar,noscis)
        close(15)
      endif
      return
      end
*********************************************************
*******subprogram 1(5) for Drude-Lorentz model
*********************************************************
      subroutine wrpar(logpar,param,mmpar,npar,noscil)
      real param(mmpar),s
      integer i,npar,noscil
      logical logpar(mmpar)
      write(15,*) param(npar+1)
      write(15,*) logpar(npar+1)
      npar=npar+1
      write(15,*) noscil
      do 25 i=1,noscil
       s=(param(npar+2)/param(npar+1))**2
       write(15,26) param(npar+1),param(npar+2),param(npar+3)**2,
     *  logpar(npar+1),logpar(npar+2),logpar(npar+3),s
      npar=npar+3
25    continue
26    format(f10.2,'  ',f10.2,'  ',f10.2,'  ',L2,L2,L2,' ',f7.2)
```

```
      return
      end
*********************************************************
*******subprogram 2(5) for asymmetric Drude-Lorentz model
*********************************************************
      subroutine wrparasym(logpar,param,mmpar,npar,noscil)
      real param(mmpar),s,pi
      integer i,npar,noscil
      logical logpar(mmpar)
      pi=4.*atan(1.)
      write(15,*) param(npar+1)
      write(15,*) logpar(npar+1)
      npar=npar+1
      write(15,*) noscil
      do 25 i=1,noscil
       s=((param(npar+2)/param(npar+1))**2)*cos(param(npar+4)*pi)
       write(15,26) param(npar+1),param(npar+2),param(npar+3)**2,
     * param(npar+4),logpar(npar+1),logpar(npar+2),
     * logpar(npar+3),logpar(npar+4),s
       npar=npar+4
25    continue
26    format(f10.2,'  ',f10.2,'  ',f10.2,'  ',f6.3,' ',L2,L2,L2,L2,
     * ' ',f7.2)
      return
      end
*********************************************************
*******subprogram 3(5) for two-layer Drude-Lorentz model
*********************************************************
      subroutine wrparlayer(logpar,param,mmpar,npar,nos)
      real param(mmpar),s,pi
      integer i,npar,nos(3),layer
      logical logpar(mmpar)
      pi=4.*atan(1.)
      write(15,*) param(npar+1)
      write(15,*) logpar(npar+1)
      npar=npar+1
      do 27 layer=1,3
      write(15,*) nos(layer)
      do 25 i=1,nos(layer)
       s=((param(npar+2)/param(npar+1))**2)
       write(15,26) param(npar+1),param(npar+2),param(npar+3)**2,
     * logpar(npar+1),logpar(npar+2),logpar(npar+3),s
       npar=npar+3
25    continue
26    format(f10.2,'  ',f10.2,'  ',f10.2,' ',L2,L2,L2,' ',f7.2)
27    continue
      write(15,*) param(npar+1), param(npar+2),
     * logpar(npar+1),logpar(npar+2)
      npar=npar+2
      return
      end
*********************************************************
*******subprogram 4(5) for factorized 4-parameter model
*********************************************************
      subroutine wrpar4(logpar,param,mmpar,npar,nos)
      real param(mmpar),s,pi
      integer i,npar,nos(3)
      logical logpar(mmpar)
```

```
      write(15,*) param(npar+1)
      write(15,*) logpar(npar+1)
      npar=npar+1
      write(15,*) nos(1)
      do 20 i=1,nos(1)
       write(15,26) param(npar+1),param(npar+2),param(npar+3)**2,
     * param(npar+4),logpar(npar+1),logpar(npar+2),
     * logpar(npar+3),logpar(npar+4)
       npar=npar+4
20    continue
      write(15,*) param(npar+1)
      write(15,*) logpar(npar+1)
      npar=npar+1
      write(15,*) nos(2)
      do 25 i=1,nos(2)
       write(15,26) param(npar+1),param(npar+2),param(npar+3)**2,
     * param(npar+4),logpar(npar+1),logpar(npar+2),
     * logpar(npar+3),logpar(npar+4)
       npar=npar+4
25    continue
26    format(f10.2,'  ',f10.2,'  ',f10.2,'  ',f10.2,' ',L2,L2,L2,L2)
      return
      end
*********************************************************
*******subprogram 5(5) for Non-Fermi-liquid+Lorentz model
*********************************************************
      subroutine wrpar5(logpar,param,mmpar,npar,noscil)
      real param(mmpar),s,alfa,gamma,wp,glo,ghi
      integer i,npar,noscil
      logical logpar(mmpar),fa,fw,fg,fgl,fgh
      write(15,*) param(npar+1), logpar(npar+1)
      alfa=param(npar+2)
        fa=logpar(npar+2)
        wp=param(npar+3)
        fw=logpar(npar+3)
        glo=param(npar+4)**2
        fgl=logpar(npar+4)
        ghi=param(npar+5)**2
        fgh=logpar(npar+5)
      npar=npar+5
        write(15,*) alfa,wp,glo,ghi,fa,fw,fgl,fgh
      write(15,*) noscil
      do 25 i=1,noscil
       s=(param(npar+2)/param(npar+1))**2
       write(15,26) param(npar+1),param(npar+2),param(npar+3)**2,
     *  logpar(npar+1),logpar(npar+2),logpar(npar+3),s
      npar=npar+3
25    continue
26    format(f10.2,'  ',f10.2,'  ',f10.2,'  ',L2,L2,L2,' ',f7.2)
      return
      end
*************************************************************************
*************************************************************

*Slot '1'
*****************************************************************
* Calculation of s-polarized reflectivity and d(rs)/d(parameter)
*----------------------------------------------------------------
```

```
* Label       || Meaning                                    || input/output
*-------------------------------------------------------------------------
* x           || Frequency                                  || in
* sn2         || sin(teta)**2                               || in
* param(nfix) || parameters for epsilon                     || in
* noscil      || no of oscillators for epsilon              || in
* nnpar       || physical dimension of arrays               || in
*             || nnpar >= 1+3*noscil required               ||
* e2x         || epsilon                                    || out
* de2x        || derivate of e2x with resp to param(i)      || out
* rs          || reflectivity                               || out
* drsdpa(nfix)|| derivate of rs with resp to param(i)       || out
*-------------------------------------------------------------------------
      subroutine rfs(x,param,rs,drsdpa,nnpar)
      parameter(mmpar=77)
c*****N.B.: MAKE SURE THAT: mmpar = nnpar
      real den,cs2,sn2
      real param(nnpar),drsdpa(nnpar)
      integer i,noscix,ndum,nosc(3),noscz(3),limit
      complex eps,est,drsde,n
      complex e2x,de2x(mmpar),sqreps
      character*1 epsmod
      common/INtet/sn2
      common/INnosc/noscix,ndum,nosc,noscz
      common/result/e2x,de2x
      common/epsmo/epsmod
*-------------------------------------------------------------------------
      if (epsmod.eq.'1') then
         call epsilo(x,param,noscix,mmpar,e2x,de2x)
         limit=1+noscix*3
      endif
      if (epsmod.eq.'2') then
         call epsiloasym(x,param,noscix,mmpar,e2x,de2x)
         limit=1+noscix*4
      endif
      if (epsmod.eq.'3') then
         call epslayer(x,param,nosc,mmpar,e2x,de2x)
         limit=1+noscix*3+2
      endif
      if (epsmod.eq.'4') then
c         call epsilo4(x,param,nosc,mmpar,e2x,de2x,sqreps)
         call epsilo4(x,param,nosc,mmpar,e2x,de2x)
         limit=2+noscix*4
      endif
      if (epsmod.eq.'5') then
         call epsnfl(x,param,noscix,mmpar,e2x,de2x)
         limit=1+noscix*3+4
      endif
      cs2=1.-sn2
      eps=(e2x-sn2)/cs2
c      if (epsmod.eq.'4') then
c       n=sqreps
c      else
c       n=csqrt(eps)
c      endif
      n=csqrt(eps)
c      n=cmplx(abs(real(n)),abs(imag(n)))
      den=(cabs(1+n))**4
```

```
      est=conjg(eps)
      drsde=(est-1.)/(n*den)
      rs=(cabs((1.-n)/(1.+n)))**2
      do 10 i=1,limit
       drsdpa(i)=real(2*drsde*de2x(i)/cs2)
10    continue
      return
      end
*****************************************************************************

*Slot '2'
*****************************************************************************
* Calculation of p-polarized reflectivity and d(rp)/d(parameter)
* for an isotropic material (epsx = epsz)
*---------------------------------------------------------------------
* Label       || Meaning                                  || input/output
*---------------------------------------------------------------------
* x           || Frequency                                || in
* sn2         || sin(teta)**2                             || in
* param(nfix) || parameters for epsilon                   || in
* noscil      || no of oscillators for epsilon            || in
* nnpar       || physical dimension of arrays             || in
*             || nnpar >= 1+3*noscil required             ||
* e2x         || epsilon                                  ||
* de2x        || derivate of e2x with resp to param(i)    || out
* rp          || reflectivity                             || out
* drpdpa(nfix)|| derivate of rp with resp to param(i)     || out
*---------------------------------------------------------------------
      subroutine rfpiso(x,param,rp,drpdpa,nnpar)
      parameter(mmpar=77)
c*****N.B.: MAKE SURE THAT: mmpar = nnpar
      real den,cs2,sn2
      real param(nnpar),drpdpa(nnpar)
      integer i,noscix,ndum,nosc(3),noscz(3),limit
      complex eps,est,drpde,n
      complex e2x,de2x(mmpar)
      character*1 epsmod
      common/INtet/sn2
      common/INnosc/noscix,ndum,nosc,noscz
      common/result/e2x,de2x
      common/epsmo/epsmod
*---------------------------------------------------------------------
      if (epsmod.eq.'1') then
         call epsilo(x,param,noscix,mmpar,e2x,de2x)
         limit=1+noscix*3
      endif
      if (epsmod.eq.'2') then
         call epsiloasym(x,param,noscix,mmpar,e2x,de2x)
         limit=1+noscix*4
      endif
      if (epsmod.eq.'3') then
         call epslayer(x,param,nosc,mmpar,e2x,de2x)
         limit=1+noscix*3+2
      endif
      if (epsmod.eq.'4') then
         call epsilo4(x,param,nosc,mmpar,e2x,de2x)
         limit=2+noscix*4
      endif
```

```
      if (epsmod.eq.'5') then
         call epsnfl(x,param,noscix,mmpar,e2x,de2x)
         limit=1+noscix*3+4
      endif
      cs2=1.-sn2
      eps=(e2x*e2x*cs2)/(e2x-sn2)
      n=csqrt(eps)
      den=(cabs(1+n))**4
      est=conjg(eps)
      drpde=(est-1.)/(n*den)
      rp=(cabs((1.-n)/(1.+n)))**2
      do 10 i=1,limit
       drpdpa(i)=real(2*drpde*de2x(i)*(eps/e2x)*(2.-eps/(e2x*cs2)))
10    continue
      return
      end
*****************************************************************************

*Slot '3'
*****************************************************************************
* Calculation of p-polarized reflectivity and d(rp)/d(parameter)
* for an anisotropic material (epsx not equal epsz)
*---------------------------------------------------------------------------
* Label       || Meaning                                  || input/output
*---------------------------------------------------------------------------
* x           || Frequency                                || in
* sn2         || sin(teta)**2                             || in
* param(nfix) || parameters for epsilonx and epsilonz     || in
* noscix      || no of oscillators for epsilonx           || in
* nosciz      || no of oscillators for epsilonz           || in
* nnpar       || physical dimension of arrays             || in
*             || nnpar >= 1+3*noscix required             ||
*             || nnpar >= 1+3*nosciz required             ||
* e2x         || epsilon_x                                ||
* de2x        || derivate of e2x with resp to paramx(i)   || out
* e2z         || epsilon_z                                || out
* de2z        || derivate of e2z with resp to paramz(i)   || out
* rp          || reflectivity                             || out
* drpdpa(nfix)|| derivate of rp with resp to param(i)     || out
*---------------------------------------------------------------------------
      subroutine rfpani(x,param,rp,drpdpa,nnpar)
      parameter(mmpar=77)
c*****N.B.: MAKE SURE THAT: mmpar = nnpar
      real den,cs2,sn2
      dimension param(nnpar),drpdpa(nnpar)
      real paramx(mmpar),paramz(mmpar)
      integer i,nparax,noscix,nparaz,nosciz,nosc(3),noscz(3)
      complex eps,est,drpde,n
      complex e2x,de2x(mmpar),e2z,de2z(mmpar)
      character*1 epsmod
      common/INtet/sn2
      common/INnosc/noscix,nosciz,nosc,noscz
      common/result/e2x,de2x
      common/resulz/e2z,de2z
      common/epsmo/epsmod
*---------------------------------------------------------------------------
      if (epsmod.eq.'1') nparax=1+3*noscix
      if (epsmod.eq.'2') nparax=1+4*noscix
```

```
      if (epsmod.eq.'3') nparax=1+3*noscix+2
      if (epsmod.eq.'4') nparax=2+4*noscix
      if (epsmod.eq.'5') nparax=1+3*noscix+4
      if (epsmod.eq.'1') nparaz=1+3*nosciz
      if (epsmod.eq.'2') nparaz=1+4*nosciz
      if (epsmod.eq.'3') nparaz=1+3*nosciz+2
      if (epsmod.eq.'4') nparaz=2+4*nosciz
      if (epsmod.eq.'5') nparaz=1+3*nosciz+4
      do 5 i=1,nparax
       paramx(i)=param(i)
5     continue
      if (epsmod.eq.'1') call epsilo(x,paramx,noscix,nnpar,e2x,de2x)
      if (epsmod.eq.'2') call epsiloasym(x,paramx,noscix,nnpar,e2x,de2x)
      if (epsmod.eq.'3') call epslayer(x,paramx,nosc,nnpar,e2x,de2x)
      if (epsmod.eq.'4') call epsilo4(x,paramx,nosc,nnpar,e2x,de2x)
      if (epsmod.eq.'5') call epsnfl(x,paramx,noscix,nnpar,e2x,de2x)
      do 6 i=1,nparaz
       paramz(i)=param(i+nparax)
6     continue
      if (epsmod.eq.'1') call epsilo(x,paramz,nosciz,nnpar,e2z,de2z)
      if (epsmod.eq.'2') call epsiloasym(x,paramz,nosciz,nnpar,e2z,de2z)
      if (epsmod.eq.'3') call epslayer(x,paramz,noscz,nnpar,e2z,de2z)
      if (epsmod.eq.'4') call epsilo4(x,paramz,noscz,nnpar,e2z,de2z)
      if (epsmod.eq.'5') call epsnfl(x,paramz,nosciz,nnpar,e2z,de2z)
        cs2=1.-sn2
      eps=cs2*e2x/(1.-sn2/e2z)
      n=csqrt(eps)
      den=(cabs(1+n))**4
      est=real(eps)-(0.,1.)*aimag(eps)
      drpde=(est-1.)/(n*den)
      rp=(cabs((1.-n)/(1.+n)))**2
      do 10 i=1,nparax
       drpdpa(i)=real(2*drpde*de2x(i)*eps/e2x)
10    continue
      do 20 i=1,nparaz
       drpdpa(i+nparax)=
     *       real(2*drpde*de2z(i)*(eps/e2z)*(1.-eps/(e2x*cs2)))
20    continue
      return
      end
*********************************************************************

*Slot '4'
*********************************************************************
* Calculation of transmission and d(tr)/d(parameter)
*-------------------------------------------------------------------
* Label       || Meaning                                  || input/output
*-------------------------------------------------------------------
* x           || Frequency   (1/cm)                       || in
* dfl         || film thickness  (cm)                     || in
* dsb         || substrate thickness (cm)                 || in
* pr          || real part of substrate optical cnst      || in
* pi          || imag part of substrate optical cnst      || in
* param(nfix) || parameters for epsilon                   || in
* noscil      || no of oscillators for epsilon            || in
* nnpar       || physical dimension of arrays             || in
*             || nfix >= 1+3*noscil          required     ||
*             || nfix >= 1+3*noscil          required     ||
```

```
* e2x         || epsilon                                  || out
* de2x        || derivate of e2x with resp to param(i)    || out
* rp          || reflectivity                             || out
* drpdpa(nfix)|| derivate of rp with resp to param(i)     || out
*-----------------------------------------------------------------------
      subroutine trans(x,param,tr,dtrdpa,nnpar)
      parameter(mmpar=77)
      dimension param(nnpar),dtrdpa(nnpar)
      real dfl,dsb,tpi,tr
      complex psi,p,p2,snpsi,cspsi,phi,n,snphi,csphi
     * ,ci,ep,em,y,ys,dtrdy,dydphi,dphidn,dnde,dydn
      integer i,noscix,ndum,nosc(3),noscz(3),limit
      complex e2x,de2x(mmpar),psbstr
      character*1 epsmod
      common/INnosc/noscix,ndum,nosc,noscz
      common/INtran/dfl,dsb,psbstr
      common/result/e2x,de2x
      common/epsmo/epsmod
*-----------------------------------------------------------------------
      p=psbstr
      tpi=8.*atan(1.)
      ci=(0.,1.)
      if (epsmod.eq.'1') then
         call epsilo(x,param,noscix,nnpar,e2x,de2x)
         limit=1+noscix*3
      endif
      if (epsmod.eq.'2') then
         call epsiloasym(x,param,noscix,nnpar,e2x,de2x)
         limit=1+noscix*4
      endif
      if (epsmod.eq.'3') then
         call epslayer(x,param,nosc,nnpar,e2x,de2x)
         limit=1+noscix*3+2
      endif
      if (epsmod.eq.'4') then
         call epsilo4(x,param,nosc,nnpar,e2x,de2x)
         limit=2+noscix*4
      endif
      if (epsmod.eq.'5') then
         call epsnfl(x,param,noscix,nnpar,e2x,de2x)
         limit=1+noscix*3+4
      endif
      n=csqrt(e2x)
      phi=dfl*x*tpi*n
      ep=cexp(ci*phi)
      em=1/ep
      csphi=(ep+em)/(2.,0.)
      snphi=(ep-em)/(0.,2.)
      p2=p**2
      psi=dsb*x*tpi*p
      ep=cexp(ci*psi)
      em=1/ep
      cspsi=(ep+em)/(2.,0.)
      snpsi=(ep-em)/(0.,2.)
      y=csphi*cspsi-snphi*snpsi*(n/p+p/n)/2
     * -ci*snphi*cspsi*(n+1./n)/2-ci*csphi*snpsi*(p+1./p)/2
      ys=conjg(y)
      tr=real(1/(y*ys))
```

```
      dtrdy=-tr/y
      dydphi=-snphi*cspsi-csphi*snpsi*(n/p+p/n)/2
     * -ci*csphi*cspsi*(n+1./n)/2+ci*snphi*snpsi*(p+1./p)/2
      dphidn=dfl*x*tpi
      dydn=-snphi*snpsi*(1./p-p/e2x)/2.
     *  -ci*snphi*cspsi*(1.-1./e2x)/2.
      dnde=0.5/n
      do 10 i=1,limit
       dtrdpa(i)=2*real(dtrdy*(dydphi*dphidn+dydn)*dnde*de2x(i))
10    continue
      return
      end
*********************************************************************


*Slot '5'
*********************************************************************************
*
* Calculation of p-polarized reflectivity and d(rp)/d(parameter) for an
* anisotropic film on an isotropic substrate (epsx not equal epsz not equal
* epss)
*********************************************************************************
*
      subroutine rfpans(x,param,rp,drpdpa,nnpar)
      parameter(mmpar=77)
c*****N.B: MAKE SURE THAT: mmpar = nnpar
      real tpi,kd,teta,sn1,cs1,sn1q,cs1q
      dimension param(nnpar), drpdpa(nnpar)
      real paramx(mmpar),paramz(mmpar),params(mmpar)
      real dfl,dsb
      integer i,nparax,noscix,nparaz,nosciz,nparas,noscis
      integer nosc(3),noscz(3)
      complex ic,psbstr,drpde2x,drpde2z,drpde3x
      complex n2x,n2z,n3x,n3z,e2x,e2z,e3x,e3z,cs2,cs3,tg2,tg3
      complex de2x(mmpar),de2z(mmpar),de3x(mmpar)
      complex efac,cntfrc,r12,r21,t12,t21,r23,t23,rl,t13,r13
      complex k1z,k2z,k3z,z1,z2,z3
      complex erc,drlde2x,dr12de2x,dz2de2x,dk2zde2x,dn2xde2x
      complex dt12de2x,dcs2de2x,dt21de2x,dr23de2x
      complex defacde2x,dexpde2x,dcntfrcde2x
      complex drlde2z,dr12de2z,dz2de2z,dk2zde2z
      complex dt12de2z,dcs2de2z,dt21de2z,dr23de2z
      complex defacde2z,dexpde2z,dcntfrcde2z
      complex dr23de3x,dz3de3x,dk3zde3x,dn3xde3x
      complex dn2zde2z,drlde3x
      character*1 epsmod
      common/INnosc/noscix,nosciz,nosc,noscz
      common/INtran/dfl,dsb,psbstr
      common/result/e2x,de2x
      common/resulz/e2z,de2z
      common/resuls/e3x,de3x
      common/extra/noscis,teta
      common/epsmo/epsmod
*---------------------------------------------------------------------
      if (epsmod.eq.'1') nparax=1+3*noscix
      if (epsmod.eq.'2') nparax=1+4*noscix
      if (epsmod.eq.'3') nparax=1+3*noscix+2
      if (epsmod.eq.'4') nparax=2+4*noscix
```

```
      if (epsmod.eq.'5') nparax=1+3*noscix+4
      if (epsmod.eq.'1') nparaz=1+3*nosciz
      if (epsmod.eq.'2') nparaz=1+4*nosciz
      if (epsmod.eq.'3') nparaz=1+3*nosciz+2
      if (epsmod.eq.'4') nparaz=2+4*nosciz
      if (epsmod.eq.'5') nparaz=1+3*nosciz+4
      nparas=1+3*noscis
      do 5 i=1,nparax
       paramx(i)=param(i)
5     continue
      if (epsmod.eq.'1') call epsilo(x,paramx,noscix,nnpar,e2x,de2x)
      if (epsmod.eq.'2') call epsiloasym(x,paramx,noscix,nnpar,e2x,de2x)
      if (epsmod.eq.'3') call epslayer(x,paramx,nosc,nnpar,e2x,de2x)
      if (epsmod.eq.'4') call epsilo4(x,paramx,nosc,nnpar,e2x,de2x)
      if (epsmod.eq.'5') call epsnfl(x,paramx,noscix,nnpar,e2x,de2x)
      do 6 i=1,nparaz
       paramz(i)=param(i+nparax)
6     continue
      if (epsmod.eq.'1') call epsilo(x,paramz,nosciz,nnpar,e2z,de2z)
      if (epsmod.eq.'2') call epsiloasym(x,paramz,nosciz,nnpar,e2z,de2z)
      if (epsmod.eq.'3') call epslayer(x,paramz,noscz,nnpar,e2z,de2z)
      if (epsmod.eq.'4') call epsilo4(x,paramz,noscz,nnpar,e2z,de2z)
      if (epsmod.eq.'5') call epsnfl(x,paramz,nosciz,nnpar,e2z,de2z)
      do 7 i=1,nparas
       params(i)=param(i+nparax+nparaz)
7     continue
      call epsilo(x,params,noscis,nnpar,e3x,de3x)

      tpi=8.*atan(1.)
      ic=(0.,1.)
c     sn1q=sn2 u glavni program

      sn1=sin(teta*tpi/360)
      sn1q=sn1**2
      cs1=cos(teta*tpi/360)
      cs1q=cs1**2
c
c      da li treba da se zameni
c
       e3z=e3x
       n2x=csqrt(e2x)
       n2z=csqrt(e2z)
       n3x=csqrt(e3x)
       n3z=csqrt(e3z)
c       print *,'NNN ',n2x,n2z,n3x,n3z,dfl
       kd=x*tpi*dfl
*
*      p-polarization
       k1z=cs1
       k2z=n2x*csqrt(1-sn1q/e2z)
       if (aimag(k2z).lt.0) k2z=conjg(k2z)
       k3z=n3x*csqrt(1-sn1q/e3z)
       if (aimag(k3z).lt.0) k3z=conjg(k3z)
       z1=k1z
       z2=k2z/e2x
       z3=k3z/e3x
c       print *,'ZZZKKK ',z1,z2,z3,k1z,k2z,k3z
       tg2=sn1*n2x/(n2z*k2z)
```

```
        tg3=sn1*n3x/(n3z*k3z)
        cs2=1/csqrt(1+tg2*tg2)
        cs3=1/csqrt(1+tg3*tg3)
        r12=(z1-z2)/(z1+z2)
        t12=(1-r12)*cs1/cs2
        r21=-r12
        t21=(1-r21)*cs2/cs1
        r23=(z2-z3)/(z2+z3)
        t23=(1-r23)*cs2/cs3
        r13=(z1-z3)/(z1+z3)
        t13=(1-r13)*cs1/cs3
c        print *,'R1T1 ',r12,r21,t12,t21,r23
        efac=cexp(ic*kd*k2z)
c        print *,'IC ',ic,kd,k2z
        cntfrc=1/(1-r23*efac*r21*efac)
        rl=r12+t12*efac*r23*efac*cntfrc*t21
        erc=conjg(rl)
        rp=erc*rl
c        print *,'EFAC ',efac,cntfrc,rl,erc,rp
*  izvod po e2x
*  prvi clan
        dn2xde2x=1/(2*n2x)
        dk2zde2x=dn2xde2x*k2z/n2x
        dz2de2x=(dk2zde2x*e2x-k2z)/(e2x*e2x)
        dr12de2x=-2*z1*dz2de2x/((z1+z2)*(z1+z2))
*  drugi clan
        dtg2de2x=sn1*(k2z*dn2xde2x-n2x*dk2zde2x)/(n2z*k2z*k2z)
        dcs2de2x=-cs2*cs2*cs2*tg2*dtg2de2x
        dt12de2x=cs1/cs2*(-cs2*dr12de2x-(1-r12)*dcs2de2x)/cs2
*  treci clan
        dt21de2x=(cs2*dr12de2x+(1-r21)*dcs2de2x)/cs1
*  cetvrti clan
        dr23de2x=2*z3*dz2de2x/((z2+z3)*(z2+z3))
*  peti clan
        dexpde2x=ic*kd*dk2zde2x
        defacde2x=efac*dexpde2x
*  sesti clan
        dcntfrcde2x=cntfrc*cntfrc*efac
      *           *(dr23de2x*r21*efac-dr12de2x*r23*efac+
      *             2*r23*r21*defacde2x)
*  suma
        drlde2x=dr12de2x
      *        +dt12de2x*t21*efac*efac*cntfrc*r23
      *        +dt21de2x*t12*efac*efac*cntfrc*r23
      *        +dr23de2x*t12*t21*efac*efac*cntfrc
      *        +2*defacde2x*r23*t12*t21*efac*cntfrc
      *        +dcntfrcde2x*t12*t21*r23*efac*efac

        drpde2x=erc*drlde2x

*  izvod po e2z (ez)
*  prvi clan
        dk2zde2z=n2x*n2x*sn1q/(2*k2z*e2z*e2z)
        dz2de2z=dk2zde2z/e2x
        dr12de2z=-2*z1*dz2de2z/((z1+z2)*(z1+z2))
*  drugi clan
        dn2zde2z=1/(2*n2z)
        dtg2de2z=-sn1*n2x*(n2z*dk2zde2z+k2z*dn2zde2z)/(k2z*k2z*n2z*n2z)
```

```
       dcs2de2z=-cs2*cs2*cs2*tg2*dtg2de2z
       dt12de2z=cs1/cs2*(-cs2*dr12de2z-(1-r12)*dcs2de2z)/cs2
*  treci clan
       dt21de2z=(cs2*dr12de2z+(1-r21)*dcs2de2z)/cs1
*  cetvrti clan
       dr23de2z=2*z3*dz2de2z/((z2+z3)*(z2+z3))
*  peti clan
       dexpde2z=ic*kd*dk2zde2z
       defacde2z=efac*dexpde2z
*  sesti clan
       dcntfrcde2z=cntfrc*cntfrc*efac
     *           *(dr23de2z*r21*efac-dr12de2z*r23*efac+2*r23*r21
     *           *defacde2z)



*  suma
        drlde2z=dr12de2z
     *          +dt12de2z*t21*efac*efac*cntfrc*r23
     *        +dt21de2z*t12*efac*efac*cntfrc*r23
     *        +dr23de2z*t12*t21*efac*efac*cntfrc
     *        +2*defacde2z*r23*t12*t21*efac*cntfrc
     *        +dcntfrcde2z*t12*t21*r23*efac*efac
       drpde2z=erc*drlde2z
c       print *,'DI 2 ',dr12de2z,dt12de2z,t21,efac,cntfrc,r23
c       print *,'DI 2 ',dt21de2z,t12,dr23de2z,defacde2,dcntfrcde2z
c       print *,'DI 2 ',erc,drlde2z,drpde2z
*  izvod po e3x (es)
       dn3xde3x=1/(2*n3x)
       dk3zde3x=dn3xde3x*k3z/n3x+n3x*n3x*sn1q/(2*k3z*e3x*e3x)
       dz3de3x=(e3x*dk3zde3x-k3z)/(e3x*e3x)
       dr23de3x=-z2*2*dz3de3x/((z2+z3)*(z2+z3))
*  suma
       drlde3x=dr23de3x*t21*t12*efac*efac*cntfrc


        drpde3x=erc*drlde3x
c        print *,'DI 3 ',erc,drlde3x,drpde3x
       do 10  i=1,nparax
        drpdpa(i)=real(2*drpde2x*de2x(i))
   10  continue
       do 11 i=1,naraz
        drpdpa(i+nparax)=real(2*drpde2z*de2z(i))
   11  continue
       do 12 i=1,naras
        drpdpa(i+nparax+naraz)=real(2*drpde3x*de3x(i))
   12  continue

      return
      end


*
* SLOT '6'
*****************************************************************
* Calculation of sigma_1 and d(sigma)/d(parameter)
*---------------------------------------------------------------
* Label       || Meaning                                || input/output
```

```
*-----------------------------------------------------------------------
* x           || Frequency                                  || in
* sn2         || sin(teta)**2                               || in
* param(nfix) || parameters for epsilon                     || in
* noscil      || no of oscillators for epsilon              || in
* nnpar       || physical dimension of arrays               || in
*             || nnpar >= 1+3*noscil required               ||
* e           || epsilon                                    || out
* de          || derivate of e with resp to param(i)        || out
* sig         || optical conductivity                       || out
* dsidpa(nfix)|| derivate of sig with resp to param(i)      || out
*-----------------------------------------------------------------------
      subroutine sigma(x,param,sig,dsidpa,nnpar)
      parameter(mmpar=77)
c*****N.B.: MAKE SURE THAT: mmpar = nnpar
      real den,pi
      real param(nnpar),dsidpa(nnpar)
      integer i,noscix,ndum,nosc(3),noscz(3),limit
      complex eps,est,dside,n
      complex ci,e2x,de2x(mmpar)
      character*1 epsmod
      common/INtet/sn2
      common/INnosc/noscix,ndum,nosc,noscz
      common/result/e2x,de2x
      common/epsmo/epsmod
*-----------------------------------------------------------------------
      if (epsmod.eq.'1') then
        call epsilo(x,param,noscix,mmpar,e2x,de2x)
        limit=1+noscix*3
      endif
      if (epsmod.eq.'2') then
         call epsiloasym(x,param,noscix,mmpar,e2x,de2x)
         limit=1+noscix*4
      endif
      if (epsmod.eq.'3') then
         call epslayer(x,param,nosc,mmpar,e2x,de2x)
         limit=1+noscix*3+2
      endif
      if (epsmod.eq.'4') then
         call epsilo4(x,param,nosc,mmpar,e2x,de2x)
         limit=2+noscix*4
      endif
      if (epsmod.eq.'5') then
        call epsnfl(x,param,noscix,mmpar,e2x,de2x)
        limit=1+noscix*3+4
      endif
      sig=aimag(e2x)*x*0.5*0.0333795
      do 10 i=1,limit
      dsidpa(i)=x*0.5*0.0333795*aimag(de2x(i))
10    continue
      return
      end
**************************************************************************

*
* SLOT '7'
**************************************************************************
* Calculation of ratio of p- and s-polarized reflectivity and
```

```
* d(rp/rs)/d(parameter) for an isotropic material
*-----------------------------------------------------------------------
* Label       || Meaning                                  || input/output
*-----------------------------------------------------------------------
* x           || Frequency                                || in
* sn2         || sin(teta)**2                             || in
* param(nfix) || parameters for epsilon                   || in
* noscil      || no of oscillators for epsilon            || in
* nnpar       || physical dimension of arrays             || in
*             || nnpar >= 1+3*noscil required             ||
* e2x         || epsilon                                  ||
* de2x        || derivate of e2x with resp to param(i)    || out
* rps         || p-reflectivity / s-reflectivity          || out
* drpsdpa(nfix)|| derivate of rp/rs with resp to param(i)  || out
*-----------------------------------------------------------------------
      subroutine rfps(x,param,rps,drpsdpa,nnpar)
      parameter(mmpar=77)
c*****N.B.: MAKE SURE THAT: mmpar = nnpar
      real den,cs2,sn2
      real param(nnpar),drpsdpa(nnpar),rps
      integer i,noscix,ndum,nosc(3),noscz(3),limit
      complex eps,est,drpde,n,drsde,rs,rp
      complex e2x,de2x(mmpar),drpdpa(mmpar),drsdpa(mmpar)
      character*1 epsmod
      common/INtet/sn2
      common/INnosc/noscix,ndum,nosc,noscz
      common/result/e2x,de2x
      common/epsmo/epsmod
*-----------------------------------------------------------------------
      if (epsmod.eq.'1') then
         call epsilo(x,param,noscix,mmpar,e2x,de2x)
         limit=1+noscix*3
      endif
      if (epsmod.eq.'2') then
         call epsiloasym(x,param,noscix,mmpar,e2x,de2x)
         limit=1+noscix*4
      endif
      if (epsmod.eq.'3') then
         call epslayer(x,param,nosc,mmpar,e2x,de2x)
         limit=1+noscix*3+2
      endif
      if (epsmod.eq.'4') then
         call epsilo4(x,param,nosc,mmpar,e2x,de2x)
         limit=2+noscix*4
      endif
      if (epsmod.eq.'5') then
         call epsnfl(x,param,noscix,mmpar,e2x,de2x)
         limit=1+noscix*3+4
      endif
      cs2=1.-sn2
******p-polarized
      eps=(e2x*e2x*cs2)/(e2x-sn2)
      n=csqrt(eps)
      den=(cabs(1+n))**4
      est=conjg(eps)
      drpde=(est-1.)/(n*den)
      rp=(cabs((1.-n)/(1.+n)))**2
      do 10 i=1,limit
```

```
       drpdpa(i)=real(2*drpde*de2x(i)*(eps/e2x)*(2.-eps/(e2x*cs2)))
10    continue
******s-polarized
      eps=(e2x-sn2)/cs2
      n=csqrt(eps)
      den=(cabs(1+n))**4
      est=conjg(eps)
      drsde=(est-1.)/(n*den)
      rs=(cabs((1.-n)/(1.+n)))**2
      do 20 i=1,limit
       drsdpa(i)=real(2*drsde*de2x(i)/cs2)
20    continue
******p-polarized/s-polarized
      rps=rp/rs
      do 30 i=1,limit
       drpsdpa(i) = (drpdpa(i) - rps*drsdpa(i)) / rs
30    continue
      return
      end


**********************************************************************
*
* SLOT '8'
* Added 10 february 1999 by D. van der Marel
**********************************************************************
* Calculation of psi or delta and d(psi)/d(parameter) or
* d(delta)/d(parameter) for an isotropic material
* frequency gt 0 returns psi and  d(psi)/d(parameter)
* frequency le 0 returns delta and  d(delta)/d(parameter)
*---------------------------------------------------------------------
* Label       || Meaning                                || input/output
*---------------------------------------------------------------------
* x           || Frequency                              || in
* sn2         || sin(teta)**2                           || in
* param(nfix) || parameters for epsilon                 || in
* noscil      || no of oscillators for epsilon          || in
* nnpar       || physical dimension of arrays           || in
*             || nnpar >= 1+3*noscil required           ||
* e2x         || epsilon                                ||
* de2x        || derivate of e2x with resp to param(i)  || out
* psidelt     || psi or delta                           || out
* dpsidel(nfix)|| derivate of psi/delta with resp to param(i)  || out
*---------------------------------------------------------------------
      subroutine psidel(x,param,psidlt,dpsidel,nnpar)
      parameter(mmpar=77)
c*****N.B.: MAKE SURE THAT: mmpar = nnpar
      real cs2,sn2,cs1,psifac,psidlt,rho
      real param(nnpar),dpsidel(nnpar)
      integer i,noscix,ndum,nosc(3),noscz(3),limit
      complex drpde,drsde,rs,rp,rps,n1,n2,dlnrps
      complex e2x,de2x(mmpar)
      character*1 epsmod
      common/INtet/sn2
      common/INnosc/noscix,ndum,nosc,noscz
      common/result/e2x,de2x
      common/epsmo/epsmod
*---------------------------------------------------------------------
```

```
      if (epsmod.eq.'1') then
         call epsilo(abs(x),param,noscix,mmpar,e2x,de2x)
         limit=1+noscix*3
      endif
      if (epsmod.eq.'2') then
         call epsiloasym(abs(x),param,noscix,mmpar,e2x,de2x)
         limit=1+noscix*4
      endif
      if (epsmod.eq.'3') then
         call epslayer(abs(x),param,nosc,mmpar,e2x,de2x)
         limit=1+noscix*3+2
      endif
      if (epsmod.eq.'4') then
         call epsilo4(abs(x),param,nosc,mmpar,e2x,de2x)
         limit=2+noscix*4
      endif
      if (epsmod.eq.'5') then
         call epsnfl(abs(x),param,noscix,mmpar,e2x,de2x)
         limit=1+noscix*3+4
      endif
      cs2=1.-sn2
      cs1=sqrt(cs2)
******p-polarized
      n1=e2x*cs1
      n2=csqrt(e2x-sn2)
      rp=(n1-n2)/(n1+n2)
      drpde=(2*cs1*n2-n1/n2)/((n1+n2)**2)
******s-polarized
      n1=cs1
*     n2=csqrt(e2x-sn2)
      rs=(n1-n2)/(n1+n2)
      drsde=(-n1/n2)/((n1+n2)**2)
******psi and delta
      rps=rp/rs
      rho=real(cabs(rps))
      dlnrps=drpde/rp-drsde/rs
      psifac=rho/(1+rho**2)
      if (x.gt.0) then
*       psi and dpsi/dparam
        psidlt=atan(rho)
       else
*       delta and ddelta/dparam
        psidlt=aimag(clog(rps))
      endif
      do 20 i=1,limit
        if (x.gt.0) then
         dpsidel(i)=psifac*real(dlnrps*de2x(i))
        else
         dpsidel(i)=aimag(dlnrps*de2x(i))
        endif
20    continue
      return
      end
**********************************************************************



**********************************************************************
```

```
*
* SLOT '9'
* Added 4 january 2001 by D. van der Marel
*********************************************************************
* Calculation of eps1 or eps2 and d(eps1)/d(parameter) or
* d(eps2)/d(parameter) for an isotropic material
* frequency gt 0 returns eps1 and  d(eps1)/d(parameter)
* frequency le 0 returns eps2 and  d(eps2)/d(parameter)
*--------------------------------------------------------------------
* Label       || Meaning                                  || input/output
*--------------------------------------------------------------------
* x           || Frequency                                || in
* param(nfix) || parameters for epsilon                   || in
* noscil      || no of oscillators for epsilon            || in
* nnpar       || physical dimension of arrays             || in
*             || nnpar >= 1+3*noscil required             ||
* e2x         || epsilon                                  ||
* de2x        || derivate of e2x with resp to param(i)    || out
* e1e2        || eps1 or eps2                             || out
* de1e2(nfix) || derivate of eps1/eps2 with resp to param(i)  || out
*--------------------------------------------------------------------
      subroutine ep1ep2(x,param,e1e2,de1e2,nnpar)
      parameter(mmpar=77)
c*****N.B.: MAKE SURE THAT: mmpar = nnpar
      real e1e2
      real param(nnpar),de1e2(nnpar)
      integer i,noscix,ndum,nosc(3),noscz(3),limit
      complex drpde,drsde,rs,rp,rps,n1,n2,dlnrps
      complex e2x,de2x(mmpar)
      character*1 epsmod
      common/INnosc/noscix,ndum,nosc,noscz
      common/result/e2x,de2x
      common/epsmo/epsmod
*---------------------------------------------------------------------
      if (epsmod.eq.'1') then
         call epsilo(abs(x),param,noscix,mmpar,e2x,de2x)
         limit=1+noscix*3
      endif
      if (epsmod.eq.'2') then
         call epsiloasym(abs(x),param,noscix,mmpar,e2x,de2x)
         limit=1+noscix*4
      endif
      if (epsmod.eq.'3') then
         call epslayer(abs(x),param,nosc,mmpar,e2x,de2x)
         limit=1+noscix*3+2
      endif
      if (epsmod.eq.'4') then
         call epsilo4(abs(x),param,nosc,mmpar,e2x,de2x)
         limit=2+noscix*4
      endif
      if (epsmod.eq.'5') then
         call epsnfl(abs(x),param,noscix,mmpar,e2x,de2x)
         limit=1+noscix*3+4
      endif

******eps1 and eps2
      if (x.gt.0) then
*       e1 and de1/dparam
```

```
        e1e2=real(e2x)
       else
*       e2 and de2/dparam
        e1e2=aimag(e2x)
      endif

      do 20 i=1,limit
        if (x.gt.0) then
         de1e2(i)=real(de2x(i))
        else
         de1e2(i)=aimag(de2x(i))
        endif

20    continue
      return
      end
*****************************************************************************



*****************************************************************************
* Calculation of epsilon and d(epsilon)/d(parameter)
*---------------------------------------------------------------------------
* Label       || Meaning                                  || input/output
*---------------------------------------------------------------------------
* x           || Frequency                                || in
* param(nfix) || parameters for epsilon                   || in
* noscil      || no of oscillators for epsilon            || in
* npar        || physical dimension of arrays             || in
*             || npar >= 1+3*noscil required              ||
* eps         || the complex dielectric function          || out
* de          || derivate of e with resp to param(i)      || out
*---------------------------------------------------------------------------
      subroutine epsilo(x,param,noscil,nnpar,eps,de)
      parameter(mmpar=77)
c*****N.B.: MAKE SURE THAT: mmpar = nnpar
      real x,x2,wp,wp2,g,g2,wt,wt2
      real param(nnpar)
      complex ci,fac,fac2,eps,de(mmpar)
      integer noscil,ipar,i
*---------------------------------------------------------------------------
      ci=cmplx(0.,1.)
      x2=x*x
      ipar=0
      eps=param(ipar+1)
      de(ipar+1)=1.
      ipar=ipar+1
      do 20 i=1,noscil
       wt=param(ipar+1)
       wt2=wt**2
       wp=param(ipar+2)
       wp2=wp**2
       g=param(ipar+3)
       g2=g**2
       fac=(1.,0.)/cmplx(wt2-x2,-g2*x)
       fac2=fac**2
       eps=eps+wp2*fac
       de(ipar+1)=-2*wt*wp2*fac2
```

```
       de(ipar+2)= 2*wp*fac
       de(ipar+3)= 2*ci*g*x*wp2*fac2
c        write(*,*) x,wt,wp,g2,de(ipar+1),de(ipar+2),de(ipar+3)
       ipar=ipar+3
20    continue
      return
      end
*****************************************************************

*****************************************************************
* Calculation of epsilon and d(epsilon)/d(parameter)
* with asymmetric Drude-Lorentz model
*---------------------------------------------------------------
* Label       || Meaning                              || input/output
*---------------------------------------------------------------
* x           || Frequency                            || in
* param(nfix) || parameters for epsilon               || in
* noscil      || no of oscillators for epsilon        || in
* npar        || physical dimension of arrays         || in
*             || npar >= 1+3*noscil required          ||
* eps         || the complex dielectric function      || out
* de          || derivate of e with resp to param(i)  || out
*---------------------------------------------------------------
      subroutine epsiloasym(x,param,noscil,nnpar,eps,de)
      parameter(mmpar=77)
c*****N.B.: MAKE SURE THAT: mmpar = nnpar
      real x,x2,wp,wp2,g,g2,wt,wt2,theta,pi
      real param(nnpar)
      complex ci,fac,fac2,eps,de(mmpar),asym
      integer noscil,ipar,i
*---------------------------------------------------------------
      pi=4.*atan(1.)
      ci=cmplx(0.,1.)
      x2=x*x
      ipar=0
      eps=param(ipar+1)
      de(ipar+1)=1.
      ipar=ipar+1
      do 20 i=1,noscil
       wt=param(ipar+1)
       wt2=wt**2
       wp=param(ipar+2)
       wp2=wp**2
       g=param(ipar+3)
       g2=g**2
       theta=param(ipar+4)
       asym=cmplx(cos(theta*pi),sin(theta*pi))
       fac=(1.,0.)/cmplx(wt2-x2,-g2*x)
       fac2=fac**2
       eps=eps+wp2*fac*asym
       de(ipar+1)=-2*wt*wp2*fac2*asym
       de(ipar+2)= 2*wp*fac*asym
       de(ipar+3)= 2*ci*g*x*wp2*fac2*asym
       de(ipar+4)= ci*wp2*fac*asym*pi
       ipar=ipar+4
20    continue
      return
      end
```

```
**********************************************************************

**********************************************************************
* Calculation of epsilon and d(epsilon)/d(parameter)
* with Drude-Lorentz model for bi-layered structure
*---------------------------------------------------------------------
* Label       || Meaning                                || input/output
*---------------------------------------------------------------------
* x           || Frequency                              || in
* param(nfix) || parameters for epsilon                 || in
* noscil      || no of oscillators for epsilon          || in
* npar        || physical dimension of arrays           || in
*             || npar >= 1+3*noscil required            ||
* eps         || the complex dielectric function        || out
* de          || derivate of e with resp to param(i)    || out
*---------------------------------------------------------------------
      subroutine epslayer(x,param,nos,nnpar,eps,de)
      parameter(mmpar=77)
c*****N.B.: MAKE SURE THAT: mmpar = nnpar
      real x,x2,wp,wp2,g,g2,wt,wt2,pi
      real d1,d2
      real param(nnpar)
      complex ci,fac,fac2,eps,de(mmpar),epsstuff,eps1,eps2
      complex dedest,dede1,dede2,dest(mmpar),de1(mmpar),de2(mmpar)
      integer nos(3),ipar,i
*---------------------------------------------------------------------
      pi=4.*atan(1.)
      ci=cmplx(0.,1.)
      x2=x*x
      ipar=0
      epsstuff=param(ipar+1)
      eps1=cmplx(0.,0.)
      eps2=cmplx(0.,0.)
      de(ipar+1)=cmplx(1.,0.)
      ipar=ipar+1
      do 10 i=1,nos(1)
       wt=param(ipar+1)
       wt2=wt**2
       wp=param(ipar+2)
       wp2=wp**2
       g=param(ipar+3)
       g2=g**2
       fac=(1.,0.)/cmplx(wt2-x2,-g2*x)
       fac2=fac**2
       eps1=eps1+wp2*fac
       de1(ipar+1)=-2*wt*wp2*fac2
       de1(ipar+2)= 2*wp*fac
       de1(ipar+3)= 2*ci*g*x*wp2*fac2
       de2(ipar+1)= cmplx(0.,0.)
       de2(ipar+2)= cmplx(0.,0.)
       de2(ipar+3)= cmplx(0.,0.)
       dest(ipar+1)= cmplx(0.,0.)
       dest(ipar+2)= cmplx(0.,0.)
       dest(ipar+3)= cmplx(0.,0.)
       ipar=ipar+3
10    continue
      do 11 i=1,nos(2)
       wt=param(ipar+1)
```

```
       wt2=wt**2
       wp=param(ipar+2)
       wp2=wp**2
       g=param(ipar+3)
       g2=g**2
       fac=(1.,0.)/cmplx(wt2-x2,-g2*x)
       fac2=fac**2
       eps2=eps2+wp2*fac
       de2(ipar+1)=-2*wt*wp2*fac2
       de2(ipar+2)= 2*wp*fac
       de2(ipar+3)= 2*ci*g*x*wp2*fac2
       de1(ipar+1)= cmplx(0.,0.)
       de1(ipar+2)= cmplx(0.,0.)
       de1(ipar+3)= cmplx(0.,0.)
       dest(ipar+1)= cmplx(0.,0.)
       dest(ipar+2)= cmplx(0.,0.)
       dest(ipar+3)= cmplx(0.,0.)
       ipar=ipar+3
11     continue
       do 12 i=1,nos(3)
       wt=param(ipar+1)
       wt2=wt**2
       wp=param(ipar+2)
       wp2=wp**2
       g=param(ipar+3)
       g2=g**2
       fac=(1.,0.)/cmplx(wt2-x2,-g2*x)
       fac2=fac**2
       epsstuff=epsstuff+wp2*fac
       dest(ipar+1)=-2*wt*wp2*fac2
       dest(ipar+2)= 2*wp*fac
       dest(ipar+3)= 2*ci*g*x*wp2*fac2
       de1(ipar+1)= cmplx(0.,0.)
       de1(ipar+2)= cmplx(0.,0.)
       de1(ipar+3)= cmplx(0.,0.)
       de2(ipar+1)= cmplx(0.,0.)
       de2(ipar+2)= cmplx(0.,0.)
       de2(ipar+3)= cmplx(0.,0.)
       ipar=ipar+3
12     continue
       d1=param(ipar+1)
       d2=param(ipar+2)
c      de(ipar+1)=cmplx(0.,0.)
c      de(ipar+2)=cmplx(0.,0.)
       ipar=ipar+2
       eps=(epsstuff+eps1)*(epsstuff+eps2)/(epsstuff+d1*eps2+d2*eps1)
       fac=epsstuff+d1*eps2+d2*eps1
       fac2=fac**2
       dedest=(2*epsstuff+eps1+eps2)*fac-(epsstuff+eps1)*(epsstuff+eps2)
       dedest=dedest/fac2
       dede1=d1*((epsstuff+eps2)**2)/fac2
       dede2=d2*((epsstuff+eps1)**2)/fac2
       do 20 i=2,ipar-2
       de(i)=dedest*dest(i)+dede1*de1(i)+dede2*de2(i)
20     continue
       de(ipar-1)=-eps2*(epsstuff+eps1)*(epsstuff+eps2)/fac2
       de(ipar)=-eps1*(epsstuff+eps1)*(epsstuff+eps2)/fac2
       return
```

```
      end
************************************************************************

************************************************************************
* Calculation of epsilon and d(epsilon)/d(parameter)
* with factorized 4-parameter model + additional Drude-Lorentz terms
*-----------------------------------------------------------------------
* Label       || Meaning                                  || input/output
*-----------------------------------------------------------------------
* x           || Frequency                                || in
* param(nfix) || parameters for epsilon                   || in
* noscil      || no of oscillators for epsilon            || in
* npar        || physical dimension of arrays             || in
*             || npar >= 1+3*noscil required              ||
* eps         || the complex dielectric function          || out
* de          || derivate of e with resp to param(i)      || out
*-----------------------------------------------------------------------
      subroutine epsilo4(x,param,nos,nnpar,eps,de)
c      subroutine epsilo4(x,param,nos,nnpar,eps,de,sqreps)
      parameter(mmpar=77)
c*****N.B.: MAKE SURE THAT: mmpar = nnpar
      real x,x2,wt,wl,wp,wt2,wl2,wp2,g,g2,as,b
      real epsinfdl,epsinffac
      real param(nnpar),epsabs,epsph,abs,ph
      complex ci,facl,fact,fact2,eps,de(mmpar),sqreps
      complex epsfac,epsdl
      integer nos(3),ipar,i
*-----------------------------------------------------------------------
      ci=cmplx(0.,1.)
      x2=x*x
      ipar=0
      epsinfdl=param(ipar+1)
      epsdl=epsinfdl
      de(ipar+1)=1.
      ipar=ipar+1
      do 10 i=1,nos(1)
       wt=param(ipar+1)
       wt2=wt**2
       wp=param(ipar+2)
       wp2=wp**2
       g=param(ipar+3)
       g2=g**2
       b=param(ipar+4)
       fact=(1.,0.)/cmplx(wt2-x2,-g2*x)
       fact2=fact**2
       epsdl=epsdl+(wp2-ci*b*x)*fact
       de(ipar+1)=-2*wt*(wp2-ci*b*x)*fact2
       de(ipar+2)= 2*wp*fact
       de(ipar+3)= 2*ci*g*x*(wp2-ci*b*x)*fact2
       de(ipar+4)=-ci*x*fact
       ipar=ipar+4
10    continue
      epsinffac=param(ipar+1)
      if (nos(2).eq.0) then
        epsfac=0
      else
        epsfac=epsinffac
      endif
```

```
      ipar=ipar+1
      do 20 i=1,nos(2)
       wt=param(ipar+1)
       wt2=wt**2
       wl=param(ipar+2)
       wl2=wl**2
       g=param(ipar+3)
       g2=g**2
       as=param(ipar+4)
       fact=(1.,0.)/cmplx(wt2-x2,-g2*x)
c       fact=(wl2-x2-ci*as*g2*x)/cmplx(wt2-x2,-g2*x)
c       call absph(fact,abs,ph)
c       epsabs=epsabs*abs
c       epsph=epsph+ph
c       call absph(wl2-x2-ci*as*g2*x,abs,ph)
c       epsabs=epsabs*abs
c       epsph=epsph+ph
       epsfac=epsfac*(wl2-x2-ci*as*g2*x)*fact
       ipar=ipar+4
20    continue
c      eps=cmplx(real(eps),abs(imag(eps)))
c      call absph(eps,abs,ph)
c     eps=cmplx(abs*cos(ph),abs*sin(ph))
c      eps=cmplx(epsabs*cos(epsph),epsabs*sin(epsph))
c      epsabs=sqrt(epsabs)
c      sqreps=cmplx(epsabs*cos(epsph/2),epsabs*sin(epsph/2))
c      sqreps=cmplx(sqrt(abs)*cos(ph/2),sqrt(abs)*sin(ph/2))
      eps=epsfac+epsdl
      ipar=1+4*nos(1)
      if (epsinffac.eq.0) then
        de(ipar+1)=0.00000000001
      else
        de(ipar+1)=epsfac/epsinffac
      endif
      ipar=ipar+1
      do 30 i=1,nos(2)
       wt=param(ipar+1)
       wt2=wt**2
       wl=param(ipar+2)
       wl2=wl**2
       g=param(ipar+3)
       g2=g**2
       as=param(ipar+4)
       fact=(1.,0.)/cmplx(wt2-x2,-g2*x)
       facl=(1.,0.)/cmplx(wl2-x2,-as*g2*x)
       de(ipar+1)= -1*epsfac*2*wt*fact
       de(ipar+2)= epsfac*2*wl*facl
       de(ipar+3)= -2*epsfac*ci*g*x*(a*(wt2-x2)-(wl2-x2))*facl*fact
       de(ipar+4)= -1*epsfac*ci*g2*x*facl
       ipar=ipar+4
30    continue
      return
      end
******subroutine of epsilo4, calculating absolute value and phase of c
      subroutine absph(c,abs,ph)
      real pi,abs,ph
      complex c
      pi=4.*atan(1.)
```

```
      abs=sqrt((real(c))**2+(imag(c))**2)
      ph=atan(imag(c)/real(c))
      if (real(c).lt.0.and.imag(c).ge.0) ph=ph+pi
      if (real(c).lt.0.and.imag(c).lt.0) ph=ph-pi
      return
      end
*****************************************************************


*****************************************************************
* Calculation of epsilon and d(epsilon)/d(parameter)
* with Non-Fermi-liquid model + Lorentz oscillators
*---------------------------------------------------------------
* Label       || Meaning                              || input/output
*---------------------------------------------------------------
* x           || Frequency                            || in
* param(nfix) || parameters for epsilon               || in
* alfa        || NFL-exponent                         || in
* wp          || NFL-Plasma Frequency                 || in
* gamlo       || NFL-Scattering rate   , low          || in
* gamhi       || NFL-Scattering rate   , high         || in
* noscil      || no of oscillators for epsilon        || in
* npar        || physical dimension of arrays         || in
*             || npar >= 1+3*noscil required          ||
* eps         || the complex dielectric function      || out
* de          || derivate of e with resp to param(i)  || out
*---------------------------------------------------------------
      subroutine epsnfl(x,param,noscil,nnpar,eps,de)
      parameter(mmpar=77)
c*****N.B.: MAKE SURE THAT: mmpar = nnpar
      real x,x2,wp,wp2,g,g2,wt,wt2,alfa,gamlo,gamhi,gamlo2,gamhi2
      real param(nnpar)
      complex ci,fac,fac2,eps,de(mmpar),dl,dh
      integer noscil,ipar,i
*---------------------------------------------------------------
      ci=cmplx(0.,1.)
      x2=x*x
      ipar=0
      eps=param(ipar+1)
      de(ipar+1)=1.
        alfa=param(ipar+2)
        wp=param(ipar+3)
        wp2=wp**2
        gamlo=param(ipar+4)
        gamlo2=gamlo**2
        gamhi=param(ipar+5)
        gamhi2=gamhi**2
        dl=cmplx(x,gamlo2)**(1.-2.*alfa)
        dh=cmplx(x,gamhi2)**(2.*alfa)
        fac=(1.,0.)/(x*dl*dh)
      eps=eps-wp2*fac
        de(ipar+2)=-wp2*fac*2.*clog(cmplx(x,gamlo2)/cmplx(x,gamhi2))
        de(ipar+3)=-2*wp*fac
        de(ipar+4)=2*ci*gamlo*wp2*fac*(1-2*alfa)/cmplx(x,gamlo2)
        de(ipar+5)=2*ci*gamhi*wp2*fac*(2*alfa)/cmplx(x,gamhi2)
c       write(*,*) alfa,wp,gamlo2,gamhi2
c       write(*,*) x,eps,de(ipar+1),de(ipar+2)
c       write(*,*) de(ipar+3),de(ipar+4),de(ipar+5)
```

```
      ipar=ipar+5
      do 20 i=1,noscil
       wt=param(ipar+1)
       wt2=wt**2
       wp=param(ipar+2)
       wp2=wp**2
       g=param(ipar+3)
       g2=g**2
       fac=(1.,0.)/cmplx(wt2-x2,-g2*x)
       fac2=fac**2
       eps=eps+wp2*fac
       de(ipar+1)=-2*wt*wp2*fac2
       de(ipar+2)= 2*wp*fac
       de(ipar+3)= 2*ci*g*x*wp2*fac2
c        write(*,*) x,wt,wp,g2,ipar+1,de(ipar+1),ipar+3,de(ipar+3)
       ipar=ipar+3
20    continue
      return
      end
*********************************************************************



*********************************************************************
      SUBROUTINE MRQMIN(X,Y,SIG,NDATA,A,DYDA,MA,LISTA,MFIT,
     *    COVAR,ALPHA,beta,NCA,CHISQ,ochisq,ALAMDA,myfunc)
      PARAMETER (MMAX=77)
      REAL X(NDATA),Y(NDATA),SIG(NDATA),A(MA),DYDA(MA),
     *  COVAR(NCA,NCA),ALPHA(NCA,NCA),ATRY(MMAX),BETA(ma),DA(MMAX)
      INTEGER LISTA(MA)
      integer kk,j,ihit,k
      external myfunc
      IF(ALAMDA.LT.0.)THEN
        KK=MFIT+1
        DO 12 J=1,MA
          IHIT=0
          DO 11 K=1,MFIT
            IF(LISTA(K).EQ.J)IHIT=IHIT+1
11        CONTINUE
          IF (IHIT.EQ.0) THEN
            LISTA(KK)=J
            KK=KK+1
          ELSE IF (IHIT.GT.1) THEN
            write(*,*) 'Improper permutation in LISTA'
          ENDIF
12      CONTINUE
        IF (KK.NE.(MA+1)) write(*,*) 'Improper permutation in LISTA'
        ALAMDA=0.001
        CALL MRQCOF(X,Y,SIG,NDATA,A,MA,LISTA,MFIT,ALPHA,BETA,NCA,
     *        CHISQ,dyda,myfunc)
        OCHISQ=CHISQ
        DO 13 J=1,MA
          ATRY(J)=A(J)
13      CONTINUE
      ENDIF
      DO 15 J=1,MFIT
        DO 14 K=1,MFIT
```

```
          COVAR(J,K)=ALPHA(J,K)
14      CONTINUE
        COVAR(J,J)=ALPHA(J,J)*(1.+ALAMDA)
        DA(J)=BETA(J)
15    CONTINUE
      CALL GAUSSJ(COVAR,MFIT,NCA,DA,1,1)
      IF(ALAMDA.EQ.0.)THEN
        CALL COVSRT(COVAR,NCA,MA,LISTA,MFIT)
        RETURN
      ENDIF
      do 36 j=1,ma
       atry(j)=a(j)
36    continue
      DO 16 J=1,MFIT
        ATRY(LISTA(J))=A(LISTA(J))+DA(J)
16    CONTINUE
      CALL MRQCOF(X,Y,SIG,NDATA,ATRY,MA,LISTA,MFIT,COVAR,DA,NCA,
     *  CHISQ,dyda,myfunc)
      IF(CHISQ.LT.OCHISQ)THEN
        ALAMDA=0.1*ALAMDA
        OCHISQ=CHISQ
        DO 18 J=1,MFIT
          DO 17 K=1,MFIT
            ALPHA(J,K)=COVAR(J,K)
17        CONTINUE
          BETA(J)=DA(J)
          A(LISTA(J))=ATRY(LISTA(J))
18      CONTINUE
      ELSE
        ALAMDA=10.*ALAMDA
        CHISQ=OCHISQ
      ENDIF
      RETURN
      END
*****************************************************************


*****************************************************************
      SUBROUTINE MRQCOF(X,Y,SIG,NDATA,A,MA,LISTA,MFIT
     * ,ALPHA,BETA,nalp,CHISQ,dyda,myfunc)
      REAL X(NDATA),Y(NDATA),SIG(NDATA),ALPHA(NALP,NALP),BETA(MA),
     *    A(MA),DYDA(MA)
      INTEGER LISTA(MFIT)
      integer j,k,i
      real ymod,sig2i,dy,wt
      external myfunc
      DO 12 J=1,MFIT
        DO 11 K=1,J
          ALPHA(J,K)=0.
11      CONTINUE
        BETA(J)=0.
12    CONTINUE
      CHISQ=0.
      DO 15 I=1,NDATA
        CALL myfunc(X(I),A,YMOD,DYDA,MA)
        SIG2I=1./(SIG(I)*SIG(I))
        DY=Y(I)-YMOD
        DO 14 J=1,MFIT
```

```
          WT=DYDA(LISTA(J))*SIG2I
          DO 13 K=1,J
            ALPHA(J,K)=ALPHA(J,K)+WT*DYDA(LISTA(K))
13        CONTINUE
          BETA(J)=BETA(J)+DY*WT
14      CONTINUE
        CHISQ=CHISQ+DY*DY*SIG2I
15    CONTINUE
      DO 17 J=2,MFIT
        DO 16 K=1,J-1
          ALPHA(K,J)=ALPHA(J,K)
16      CONTINUE
17    CONTINUE
      RETURN
      END
*****************************************************************


*****************************************************************
      SUBROUTINE COVSRT(COVAR,NCVM,MA,LISTA,MFIT)
      real COVAR(NCVM,NCVM)
      integer LISTA(MFIT)
      integer j,i
      real swap
      DO 12 J=1,MA-1
        DO 11 I=J+1,MA
          COVAR(I,J)=0.
11      CONTINUE
12    CONTINUE
      DO 14 I=1,MFIT-1
        DO 13 J=I+1,MFIT
          IF(LISTA(J).GT.LISTA(I)) THEN
            COVAR(LISTA(J),LISTA(I))=COVAR(I,J)
          ELSE
            COVAR(LISTA(I),LISTA(J))=COVAR(I,J)
          ENDIF
13      CONTINUE
14    CONTINUE
      SWAP=COVAR(1,1)
      DO 15 J=1,MA
        COVAR(1,J)=COVAR(J,J)
        COVAR(J,J)=0.
15    CONTINUE
      COVAR(LISTA(1),LISTA(1))=SWAP
      DO 16 J=2,MFIT
        COVAR(LISTA(J),LISTA(J))=COVAR(1,J)
16    CONTINUE
      DO 18 J=2,MA
        DO 17 I=1,J-1
          COVAR(I,J)=COVAR(J,I)
17      CONTINUE
18    CONTINUE
      RETURN
      END
*****************************************************************


*****************************************************************
```

```
      SUBROUTINE GAUSSJ(A,N,NP,B,M,MP)
      PARAMETER (NMAX=77)
      real A(NP,NP),B(NP,MP)
      integer IPIV(NMAX),INDXR(NMAX),INDXC(NMAX)
      integer j,i,k,irow,icol,l,ll
      real big,dum,pivinv
      DO 11 J=1,N
        IPIV(J)=0
11    CONTINUE
      DO 22 I=1,N
        BIG=0.
        DO 13 J=1,N
          IF(IPIV(J).NE.1)THEN
            DO 12 K=1,N
              IF (IPIV(K).EQ.0) THEN
                IF (ABS(A(J,K)).GE.BIG)THEN
                  BIG=ABS(A(J,K))
                  IROW=J
                  ICOL=K
                ENDIF
              ELSE IF (IPIV(K).GT.1) THEN
                write(*,*) 'Singular matrix'
              ENDIF
12          CONTINUE
          ENDIF
13      CONTINUE
        IPIV(ICOL)=IPIV(ICOL)+1
        IF (IROW.NE.ICOL) THEN
          DO 14 L=1,N
            DUM=A(IROW,L)
            A(IROW,L)=A(ICOL,L)
            A(ICOL,L)=DUM
14        CONTINUE
          DO 15 L=1,M
            DUM=B(IROW,L)
            B(IROW,L)=B(ICOL,L)
            B(ICOL,L)=DUM
15        CONTINUE
        ENDIF
        INDXR(I)=IROW
        INDXC(I)=ICOL
        IF (A(ICOL,ICOL).EQ.0.) write(*,*) 'Singular matrix.'
        PIVINV=1./A(ICOL,ICOL)
        A(ICOL,ICOL)=1.
        DO 16 L=1,N
          A(ICOL,L)=A(ICOL,L)*PIVINV
16      CONTINUE
        DO 17 L=1,M
          B(ICOL,L)=B(ICOL,L)*PIVINV
17      CONTINUE
        DO 21 LL=1,N
          IF(LL.NE.ICOL)THEN
            DUM=A(LL,ICOL)
            A(LL,ICOL)=0.
            DO 18 L=1,N
              A(LL,L)=A(LL,L)-A(ICOL,L)*DUM
18          CONTINUE
            DO 19 L=1,M
```

```
              B(LL,L)=B(LL,L)-B(ICOL,L)*DUM
19          CONTINUE
          ENDIF
21      CONTINUE
22    CONTINUE
      DO 24 L=N,1,-1
        IF(INDXR(L).NE.INDXC(L))THEN
          DO 23 K=1,N
            DUM=A(K,INDXR(L))
            A(K,INDXR(L))=A(K,INDXC(L))
            A(K,INDXC(L))=DUM
23        CONTINUE
        ENDIF
24    CONTINUE
      RETURN
      END
*****************************************************************

*****************************************************************
      subroutine help
      character*1 a
      call dontpa
      write(*,*) '                                                  '
      write(*,*) '                                                  '
      write(*,*) 'HI'
      write(*,*)
      write(*,*) 'I am your Optics Pal '
      write(*,*)
      write(*,*) 'Version 1.1 by Dirk van der Marel, May 1996.'
      write(*,*) ' Update 2.x by Markus Gr\"uninger, Nov.1998.'
      write(*,*)
      write(*,*) 'I am the froodiest hooper in cyber-space, and I help'
      write(*,*) 'you to get the best out of your infrared or optical '
      write(*,*) 'data. My aim is, to help you with the extraction of '
      write(*,*) 'the complex dielectric function (DF) from your'
      write(*,*) 'experimental data.'
      write(*,*) 'I offer a number of different models for the DF '
      write(*,*) 'and I can fit different kinds of data.'
      write(*,*) 'You can feed me with IR/VIS/UV data like: '
      write(*,*) '(1) s-polarized reflection at any angle of incidence'
      write(*,*) '(2) p-polarized reflection at any angle of incidence
     *for an isotropic material'
      write(*,*) '(3) p-polarized reflection at any angle of incidence
     *for an anisotropic material'
      write(*,*) '(4) transmission through a film supported on a plan-
     *parallel substrate'
      write(*,*) '(5) p-polarized reflection at any angle of incidence
     *for an anisotropic material'
      write(*,*) '    on an isotropic substrate'
      write(*,*) '(6) optical conductivity data (Sigma_1) '
      write(*,*) '(7) p-/s-polarized reflection at any angle of
     *incidence'
      write(*,*) '(8) psi (positive frequencies) and delta (negative
     *frequencies) at any angle of incidence'
      write(*,*) 'and I will retreive an epsilon for you,which has the'
      write(*,*) 'usual properties of a causal response function(CRF)'
      write(*,*) '(in most cases at least ;-) ).'
      write(*,*) '                                                  '
```

```
 write(*,*) '(9) Re(eps) (positive frequencies) and Im(eps) (negative
*frequencies)'
 write(*,*) 'and I will retreive an epsilon for you,which has the'
 write(*,*) 'usual properties of a causal response function(CRF)'
 write(*,*) '(in most cases at least ;-) ).'
 write(*,*) '                                                '
 write(*,*) ' (more)   or "q" to quit                        '
 read(*,'(a1)') a
 if (a.eq.'q') return
 write(*,*) 'The different models for the dielectric function '
 write(*,*) 'epsilon are:'
 write(*,*) '(1) Drude-Lorentz (or Helmholtz-Kettler) model'
 write(*,*) '(2) asymmetric Drude-Lorentz model'
 write(*,*) '(3) Drude-Lorentz model for a two-layer system'
 write(*,*) '(4) the Cow-Pig-Sheep or Tomato-Potato model'
 write(*,*) '(5) Non Fermi Liquid + Lorentz Oscillators'
 write(*,*) ''
 write(*,*) 'ad (1):'
 write(*,*) 'eps=eps_inf + SUM_j wpj^2/(w0j^2-w^2-i*gammaj*w)'
 write(*,*) 'ad (2):'
 write(*,*) 'replace wpj^2 by wpj^2 * exp(i*thetaj*pi)'
 write(*,*) 'ad (3):'
 write(*,*) 'Your system consists of an alternating stack of two'
 write(*,*) 'different layers like 1/2/1/2/1/2 with thicknesses'
 write(*,*) 'd1 and d2. In each layer we have a local dielectric '
 write(*,*) 'function eps_j, and we allow for a global eps_hom, '
 write(*,*) 'i.e. the contribution of eps_hom is the same in the '
 write(*,*) 'two sublayers (homogeneous).'
 write(*,*) 'With x_j=d_j/(d1+d2) we obtain 1/eps (!):'
 write(*,*) '1/eps = x1/(eps_hom+eps_1) + x2/(eps_hom+eps_2)'
 write(*,*) 'ad (4):'
 write(*,*) 'Combination of an (extended) Drude-Lorentz model and'
 write(*,*) 'a factorized four parameter model.'
 write(*,*) 'eps = eps_DL + eps_fac with'
 write(*,*) 'eps_DL: replace wpj^2 by wpj^2-i*(gammapj-gammaj)*w '
 write(*,*) 'eps_fac=eps_fac_infty * PRODUCT Z_k/P_k with '
 write(*,*) ' Z_k=wlk^2-w^2-i*gammalk*w and'
 write(*,*) ' P_k=wtk^2-w^2-i*gammatk*w'
 write(*,*) 'ad (5):'
 write(*,*) 'eps=epsinf-wp^2/(w(w+i gl)^(1-2a)(w+i gh)^(2a))) + '
 write(*,*) '    + SUM_j wpj^2/(w0j^2-w^2-i*gammaj*w)'
 write(*,*) ''
 write(*,*) ' (more)   or "q" to quit                         '
 read(*,'(a1)') a
 if (a.eq.'q') return
 write(*,*) 'A few words about the different models:'
 write(*,*) 'The Drude-Lorentz model is fine and nice and gives a'
 write(*,*) 'causal repsonse function. However, it only describes '
 write(*,*) 'more or less uncoupled harmonic oscillators and it '
 write(*,*) 'does not take into account local feld corrections in'
 write(*,*) 'a rigorous way.'
 write(*,*)
 write(*,*) 'The "asymmetric" model is NOT okay in the sense that'
 write(*,*) 'it is not symmetric in the complex frequency plane, '
 write(*,*) 'hence it does not give a causal response function!'
 write(*,*) 'Strictly speaking, it does not make too much sense,'
 write(*,*) 'but it is common use and convenient, as it allows a  '
 write(*,*) 'direct comparison of wt, wp and gamma.'
```

```
write(*,*) 'And you might hope that for small values of theta '
write(*,*) 'everything is more or less alright.'
write(*,*) 'NB: it is NOT the Fano model!!'
write(*,*)
write(*,*) 'The two-layer model is fine and nice again because it'
write(*,*) 'uses the Drude-Lorentz model. Just be sure that you '
write(*,*) 'apply it to a true two-layer system ....'
write(*,*) ''
write(*,*) ' (more)   or "q" to quit                                '
read(*,'(a1)') a
if (a.eq.'q') return
write(*,*) 'With the potato-tomato model it depends!!!'
write(*,*) 'On what? Well, on you! The factorized model is okay '
write(*,*) 'if you feed it with the right parameters, but nothing'
write(*,*) 'stops you from creating a very unphysical response '
write(*,*) 'function, e.g. negative epsilon_2, reflectivity > 1 '
write(*,*) 'etc. The OBVIOUS thing you have to check for is'
write(*,*) 'that you create an ALTERNATING sequence of transverse'
write(*,*) 'and longitudinal eigenfrequencies, i.e. if you have a'
write(*,*) 'strong, broad band with wt1 << wl1 and there is a    '
write(*,*) 'weak mode sitting in between, then for this weak mode'
write(*,*) 'wl2 will be < wt2 ! You have to put this in by hand!'
write(*,*) 'And for the case of finite damping this asks for some'
write(*,*) 'concentration.'
write(*,*) 'But if you are willing to spend some more thoughts on'
write(*,*) 'your fit, then this model can be very helpful.'
write(*,*) 'It allows for asymmetry and directly gives you the '
write(*,*) 'longitudinal eigenfrequencies, which can be nice if '
write(*,*) 'you fit e.g. R_p for large angles of incidence.'
write(*,*) ''
write(*,*) 'The Non-Fermi-liquid model gives a non-Drude'
write(*,*) 'causal repsonse function, which satisfies sumrules.'
write(*,*) 'Setting alfa=0 makes NFL a Drude peak, width=g-low'
write(*,*) 'Setting alfa=0.5 makes NFL a Drude peak, width=g-high'
write(*,*) 'Inbetween it can be used to fit tho Luttinger liquids'
write(*,*) 'An arbitrary number of harm. oscillators can be added'
write(*,*)
write(*,*) '    BE CAREFUL!'
write(*,*) ''
write(*,*) ' (more)   or "q" to quit                                '
read(*,'(a1)') a
if (a.eq.'q') return
write(*,*) 'R-perp has the nice property,that log(R_perp)+i teta'
write(*,*) 'has its poles in the lower complex half-plane. Hence'
write(*,*) 'Kramers-Kronig Relations(KKR) can be used to calcu-'
write(*,*) 'late the teta from log(R_perp).'
write(*,*) 'However,some of the above types of experimental data'
write(*,*) 'can not always be converted to something which has '
write(*,*) 'all poles in the lower complex half-plane, although'
write(*,*) 'it is a analytical function of epsilon, which of'
write(*,*) 'course satisfies all criteria of a CRF.'
write(*,*) 'The safest way to get around this kind of trouble is'
write(*,*) 'not to use KKR for the measured data at all, but in-'
write(*,*) 'stead fit the data to a sum over as many oscillators'
write(*,*) 'as you like, while using the expression for the mea-'
write(*,*) 'sured curve in terms of epsilon.'
write(*,*) 'It is crude, but just as good as good old KKR.'
write(*,*) 'It actually offers some advantages over KKR: '
```

```
 write(*,*) 'The program will not crash if the experiment creates'
 write(*,*) 'unphysical values, e.g. R or T > 1 or < 0. '
 write(*,*) 'You can also use it to smoothen epsilon, or skip bad'
 write(*,*) 'parts of the spectrum. '
 write(*,*) ''
 write(*,*) ' (more)   or "q" to quit                                '
 read(*,'(a1)') a
 if (a.eq.'q') return
 write(*,*) 'This is how to proceed: First prepare an init-file'
 write(*,*) 'called fit.ini with the following info: '
 write(*,*) 'NB: use blanks to separate items on the same line'
 write(*,*)
 write(*,*) 'datatype 1=Rs, 2=Rp_iso, 3=Rp_ansio, 4=Trans. etc.'
 write(*,*) 'model of epsilon 1=Drude-Lorentz, 2=asymmetric etc.'
 write(*,*) 'Angle of incidence in degree '
 write(*,*) '   (!!obsolete for Transmission and Sigma, i.e. '
 write(*,*) '       datatype=4 or 6!!)'
 write(*,*) 'd-film d-substr Re(n_substr) Im(n_substr)'
 write(*,*) '   (!!only for Transmission and Rp_aniso/Substrate,'
 write(*,*) '       i.e. for datatype=4 or 5)'
 write(*,*) 'Absolute value of wanted precision/datapoint in fit'
 write(*,*) 'Flag for errorbars in third column of data file'
 write(*,*) 'Number of frequency ranges to be fitted (nrange)'
 write(*,*) 'Lower and upper bound of 1st frequency range '
 write(*,*) 'if nrange>1: Lower and upper bound of xth freq. range'
 write(*,*) 'Lower & upper bound of output frequency range & No.
* of points for output'
 write(*,*) '     (NB!! If No. of points for output = 0, then '
 write(*,*) '      output is produced at the input x-values.'
 write(*,*) '      If it equals -1, an additional output file '
 write(*,*) '      optpal.fitdiff.out is produced, which is '
 write(*,*) '      optpal.data.in - optpal.fit.out'
 write(*,*) '      Nice if you fit sigma!!)          '
 write(*,*) 'Number of iterations before stopping'
 write(*,*)
 write(*,*) ' (more)   or "q" to quit                                '
 read(*,'(a1)') a
 if (a.eq.'q') return
 write(*,*) 'Rename your data-file:  optpal.data in '
 write(*,*) 'For any datatype you have to prepare a file '
 write(*,*) 'optpal.param.x.in,  the contents of which depend on'
 write(*,*) 'the model for the dielectric function you use:'
 write(*,*)
 write(*,*) '(1) for the Drude-Lorentz model:'
 write(*,*) 'Initial value of eps_infty, flag'
 write(*,*) 'No. of oscillators. For each of these:'
 write(*,*) 'wtrans wplasm width flag(wt) flag(wp) flag(gamma)'
 write(*,*)
 write(*,*) '(2) for the asymmetric Drude-Lorentz model:'
 write(*,*) 'Initial value of eps_infty, flag'
 write(*,*) 'No. of oscillators. For each of these:'
 write(*,*) 'wt wp width theta flag(wt) flag(wp) flag(wid) flag(t)'
 write(*,*) '       (theta in units of pi, theta < = 0.5)'
 write(*,*)
 write(*,*) '(3) for the two-layer Drude-Lorentz model:'
 write(*,*) 'Initial value of eps_infty, flag'
 write(*,*) 'No. of oscillators in layer 1. For each of these:'
 write(*,*) 'wt wp width flag(wt) flag(wp) flag(gamma)'
```

```
write(*,*) 'No. of oscillators in layer 2. For each of these:'
write(*,*) 'wt wp width flag(wt) flag(wp) flag(gamma)'
write(*,*) 'No. of homogeneous oscillators. For each of these:'
write(*,*) 'wt wp width flag(wt) flag(wp) flag(gamma)'
write(*,*) 'x1 x2 flag(x1) flag(x2) (x1+x2=1 NOT checked)'
write(*,*)
write(*,*) '(4) for the extended Drude-Lorentz + factor model:'
write(*,*) 'Initial value of eps_DruLo_infty, flag'
write(*,*) 'No. of DruLo oscillators. For each of these:'
write(*,*) 'wt wp width b flag(wt) flag(wp) flag(gamma) flag(b)'
write(*,*) '   (here b= gamma_p - gamma_0, DruLo for b=0)'
write(*,*) 'Initial value of eps_fac_infty, flag'
write(*,*) 'No. of factorized oscillators. For each of these:'
write(*,*) 'wt wl gamma_t a flag(wt) flag(wl) flag(g_t) flag(a)'
write(*,*) '   (here a= gamma_l / gamma_t, DruLo for a=1)'
write(*,*)
write(*,*) '(5) for the Non-Fermi-liquid + Lorentz model:'
write(*,*) 'Initial value of eps_infty, flag'
write(*,*) 'Initial value of alfa,wp,gam-low,gam-high, 4 flags'
write(*,*) 'No. of oscillators. For each of these:'
write(*,*) 'wtrans wplasm width flag(wt) flag(wp) flag(gamma)'
write(*,*)
write(*,*) ' (more)   or "q" to quit                                '
read(*,'(a1)') a
if (a.eq.'q') return
write(*,*)
write(*,*)
write(*,*)
write(*,*) 'For data of Rp of an anisotropic material (datatype 3'
write(*,*) 'or 5) you need a similar file for the axis '
write(*,*) 'perpendicular to the surface, i.e. optpal.param.z.in'
write(*,*) 'The model of the dielectric function is the SAME '
write(*,*) 'for optpal.param.x.in and optpal.param.z.in, so '
write(*,*) 'the structure of the two files is identical.    '
write(*,*)
write(*,*) 'For data of Rp of an anisotropic material on an '
write(*,*) 'isotropic substrate (datatype 5) you need a similar '
write(*,*) 'file for the substrate, optpal.param.s.in'
write(*,*) 'However,  for this the Drude-Lorentz model is used.'
write(*,*)
write(*,*)
write(*,*) 'The format of flags is fortran-boolean: '
write(*,*) '.f. or .t.   or   f or t'
write(*,*)
write(*,*) 'Start the program by typing : '
write(*,*) 'optpal2.1'
write(*,*)
write(*,*) 'A few examples for fit.ini and optpal.param.x.in:'
write(*,*) ''
write(*,*) ' (more)   or "q" to quit                                '
read(*,'(a1)') a
if (a.eq.'q') return
write(*,*) 'examples for fit.ini:'
write(*,*) ''
write(*,*) '1         -> fit R_s'
write(*,*) '4         -> use tomato-potato model'
write(*,*) '0         -> 0 degrees of incidence'
write(*,*) '0.002     -> wanted precission per point'
```

```
write(*,*) 'f         -> no errorbars'
write(*,*) '2         -> 2 fit ranges'
write(*,*) '10 500    -> first range: 10-500 cm-1'
write(*,*) '550 1000  -> second range: 550-1000 cm-1'
write(*,*) '1 2000 0  -> *'
write(*,*) '20        -> 20 iterations at most '
write(*,*) ''
write(*,*) '*  the zero produces output on the x-values of the '
write(*,*) 'input file, the 1 and the 2000 loose their meaning '
write(*,*) 'in this case.'
write(*,*) ''
write(*,*) '6         -> fit Sigma'
write(*,*) '1         -> use Drude-Lorentz model'
write(*,*) '0.002     -> wanted precission per point'
write(*,*) 'f         -> no errorbars'
write(*,*) '1         -> 1 fit range'
write(*,*) '10 1000   -> first range: 10-1000 cm-1'
write(*,*) '1 2000 -1 -> *'
write(*,*) '20        -> 20 iterations at most '
write(*,*) ''
write(*,*) '*  the -1 produces output on the x-values of the '
write(*,*) 'input file, the 1 and the 2000 loose their meaning'
write(*,*) 'in this case. An additional output file '
write(*,*) 'optpal.fitdiff.out is produced, which is '
write(*,*) 'optpal.data.in minus optpal.fit.out.'
write(*,*) ''
write(*,*) ' (more)   or "q" to quit                                '
read(*,'(a1)') a
if (a.eq.'q') return
write(*,*) 'examples for optpal.param.x.in:'
write(*,*) '2 oscillators for Drude-Lorentz:'
write(*,*) ''
write(*,*) '5                 -> epsilon_infinity'
write(*,*) 'f                 -> do not fit epsilon_infinity'
write(*,*) '2                 -> No. of oscillators'
write(*,*) '0     1000 300  f t t  -> Drude with wp=1000, ...'
write(*,*) '100    200  20  t t t  -> Lorentzian at wt=100'
write(*,*) ''
write(*,*) '1 oscillator for asymmeric Drude-Lorentz:'
write(*,*) ''
write(*,*) '5                 -> epsilon_infinity'
write(*,*) 'f                 -> do not fit epsilon_infinity'
write(*,*) '1                 -> No. of oscillators'
write(*,*) '100  200  20 0.2  t t t t'
write(*,*) ''
write(*,*) 'two-layer model:'
write(*,*) ''
write(*,*) '5                 -> epsilon_infinity for eps_hom'
write(*,*) 't                 -> fit also epsilon_infinity'
write(*,*) '1                 -> No. of oscillators in layer 1'
write(*,*) '100  200  20  t t t'
write(*,*) '2                 -> No. of oscillators in layer 2'
write(*,*) '50    70   5  t t t'
write(*,*) '550  570   5  t t t'
write(*,*) '1                 -> No. of homogeneus oscillators'
write(*,*) '300  350   2  t t t'
write(*,*) '0.72 0.28 f f     -> d1/D d2/D (in principle D=d1+d2)'
write(*,*) ''
```

```
      write(*,*) ' (more)   or "q" to quit                                  '
      read(*,'(a1)') a
      if (a.eq.'q') return
      write(*,*) 'potato-tomato factorized + Drude-Lorentz model'
      write(*,*) 'with 0 Drude-Lorentz and 2 factorized oscillators:'
      write(*,*) ''
      write(*,*) '0                -> epsilon_infinity Drude-Lorentz'
      write(*,*) 'f                -> do not fit epsilon_infinity_DL'
      write(*,*) '0                -> No. of Drude-Lorentz oscillators'
      write(*,*) '5                -> epsilon_infinity factorized'
      write(*,*) 't                -> fit also epsilon_infinity_fact  '
      write(*,*) '2                -> No. of factorized oscillators'
      write(*,*) '50    70   5  1  t t t t'
      write(*,*) '550  570   5  1  t t t t'
      write(*,*) ''
      write(*,*) ' (more) and the most important of all:                '
      read(*,'(a1)') a
      call dontpa
      return
      end

      subroutine dontpa
      character*1 a
       write(*,*) '                                                     '
       write(*,*) '                                                     '
       write(*,*) '                                                     '
       write(*,*) '          d     oo     nnn   &&  ttttt             '
       write(*,*) '          d    o  o    n  n   &    t               '
       write(*,*) '        ddd    o  o    n  n        t               '
       write(*,*) '       d  d    o  o    n  n        t               '
       write(*,*) '       d  d    o  o    n  n        t               '
       write(*,*) '        ddd     oo     n  n        t               '
       write(*,*) '                                                     '
       write(*,*) '                                                     '
       write(*,*) '                                                     '
       write(*,*) '     ppp     aa     nnn     .i.     ccc            '
       write(*,*) '    p   p   a  a    n  n           c   c           '
       write(*,*) '    pppp    a  a    n  n     i     c               '
       write(*,*) '    p       aaaa    n  n     i     c               '
       write(*,*) '    p       a  a    n  n     i     c   c           '
       write(*,*) '    p       a  a    n  n     i      ccc            '
       write(*,*) '                                                     '
       write(*,*) '                                                     '
       write(*,*) '                                                     '
       write(*,*) ' (more)                                              '
       read(*,'(a1)') a
      return
      end
*********************************************************************
```