www.itp.uni-hannover/˜zawischa/ITP/computecolour.html

How to calculate and render colours – thin films as an example

Dietrich Zawischa
ITP, Leibniz University Hannover, Germany

Determination of the colour stimulus

We consider a piece of a flat soap film (which consists essentially of water), or an oil film on water. Assume that this film is seen in front of a dark background. Further assume that we look at the mirror image of a uniformly white surface (this might be the overcast sky) reflected by the film, and, for simplicity, assume that the light is incident perpendicular to the film.

Let the origin of the coordinate system be on the upper surface of the film and let the z-axis point towards the observer. The incoming wave is assumed to be linearly polarized; its electric field is written as

$$\mathbf{E}_{\text{in}}(z,t) = \mathbf{E}_0 \cos(-kz - \omega t) \tag{1}$$

where $k = 2\pi/\lambda$, $\omega = 2\pi\nu$, with λ the wavelength and ν the frequency. The brightness (energy flux) is proportional to the square of the amplitude $\mathbf{E}_0$.

Only a small fraction of the incident light is reflected at the front surface of the film. Another small fraction is reflected at the back surface. (Multiple inner reflections are neglected.) The sum of these two reflected waves reaches the eye of the observer. In order to compute its colour, we first determine how the reflected intensity depends on the wavelength λ of monochromatic light.

The electric field of the wave reflected at the upper surface is

$$-\alpha \mathbf{E}_0 \cos(kz - \omega t) \tag{2}$$

where the positive number $\alpha < 1$ is given by the square root of the reflection coefficient. The wave reflected at the back surface then is

$$\alpha\beta \mathbf{E}_0 \cos(k(z+s) - \omega t) \tag{3}$$

where β is a number a little bit smaller than 1, accounting for the fact that the intensity of the incoming light at the back surface is reduced and that part of the light reflected at the back surface is reflected again at the front surface. The wave (2) has suffered a change of sign, being reflected at the more dense medium, which did not happen to wave (3). The quantity s is the optical path length difference of the two interfering waves; as the rays are perpendicular to the film it is just

$$s = 2dn \tag{4}$$

with d the thickness and n the index of refraction of the film.

The superposition (sum) of the terms (2) and (3) gives the total reflected wave.

$$\begin{aligned} \mathbf{E}_{\mathrm{refl}}(z,t) &= \alpha \mathbf{E}_0[-\cos(kz-\omega t) + \beta\cos(kz-\omega t)\cos(ks) \\ &\quad -\beta\sin(kz-\omega t)\sin(ks)] \end{aligned} \tag{5}$$

The time average of its square at a fixed value of z is proportional to the reflected intensity. As $\overline{\cos^2(kz-\omega t)} = \frac{1}{2}$, $\overline{\sin^2(kz-\omega t)} = \frac{1}{2}$, $\overline{\cos(kz-\omega t)\ \sin(kz-\omega t)} = 0$, the result can be written as

$$I_{\mathrm{refl}}(\lambda) \propto \frac{\alpha^2 E_0^2}{2}\left[1 + \beta^2 - 2\beta\cos(\frac{2\pi s}{\lambda})\right] \tag{6}$$

The incident light is supposed to have the spectral distribution $S(\lambda)$ of blackbody radiation of temperature $T = 6504$ K, which is similar to the standard daylight D65, approximating the overcast sky. From Planck's formula we get

$$S(\lambda) \propto [\lambda^5(\exp(\frac{hc}{\lambda k_\mathrm{B}T} - 1)]^{-1} \tag{7}$$

where h is Planck's constant, c the velocity of light, and k_B Boltzmann's constant.

Replacing E_0^2 in equation (6) by $S(\lambda)$, the the colour stimulus for a given thickness of the film (or optical path difference, cf. eq. (4)) is obtained, i.e.

$$\phi(s,\lambda) = C\frac{1+\beta^2 - 2\beta\cos(\frac{2\pi s}{\lambda})}{\lambda^5(\exp(\frac{hc}{\lambda k_\mathrm{B}T} - 1)} \tag{8}$$

All constant factors have been absorbed in one constant C. This will be chosen in the last step to optimize the brightness of the colours rendered on the screen.

The CIE tristimulus values and chromaticity coordinates

From the colour stimulus ϕ, using the CIE 1931 colour matching functions[1] $\bar{x}(\lambda)$, $\bar{y}(\lambda)$, and $\bar{z}(\lambda)$, the colour coordinates X, Y, and Z (depending on the optical path difference s) are obtained as follows:

$$\begin{aligned} X(s) &= \sum_i \phi(s,\lambda_i)\,\bar{x}(\lambda_i)\,\Delta\lambda \\ Y(s) &= \sum_i \phi(s,\lambda_i)\,\bar{y}(\lambda_i)\,\Delta\lambda \\ Z(s) &= \sum_i \phi(s,\lambda_i)\,\bar{z}(\lambda_i)\,\Delta\lambda \end{aligned} \tag{9}$$

[1]Tables of these functions can be downloaded from http://cvrl.ioo.ucl.ac.uk/index.htm

Note that $\bar{y}(\lambda)$ is proportional to the lightness sensitivity curve $V(\lambda)$ of the eye (of the standard observer), and therefore $Y(s)$ is a measure of the brightness (luminance).

It is interesting to obtain in addition the colour coordinates x, y and to plot the locus of reflected colours in the CIE chromaticity diagram.

$$x(s) = \frac{X(s)}{X(s) + Y(s) + Z(s)}, \qquad y(s) = \frac{Y(s)}{X(s) + Y(s) + Z(s)}. \tag{10}$$

This is shown in figure 1. We see that some of the colours seen on thin films or soap bubbles can not be reproduced exactly with the sRGB primaries. In particular, there appear brilliant bluish green hues which can not be displayed. Nevertheless, as we shall see, the picture to be obtained on the screen is quite satisfactory!

Parameters of the output device (screen)

We assume that the output device conform to the sRGB recommendations[2]. This means, in brief, that the white point is D65

$$x_W = 0.3127 \qquad y_W = 0.3290, \tag{11}$$

and the colour coordinates of the primaries **R**, **G**, and **B** are

$$\begin{array}{rclrcl} x_R &=& 0.640 & y_R &=& 0.330 \\ x_G &=& 0.300 & y_G &=& 0.600 \\ x_B &=& 0.150 & y_B &=& 0.060, \end{array} \tag{12}$$

and the effective CRT gamma is $\gamma = 2.2$.

The tansformation matrix from X, Y, Z to R, G, B, as given in the reference cited[2], can be calculated from the above data:

[2]http://www.color.org/sRGB.xalter

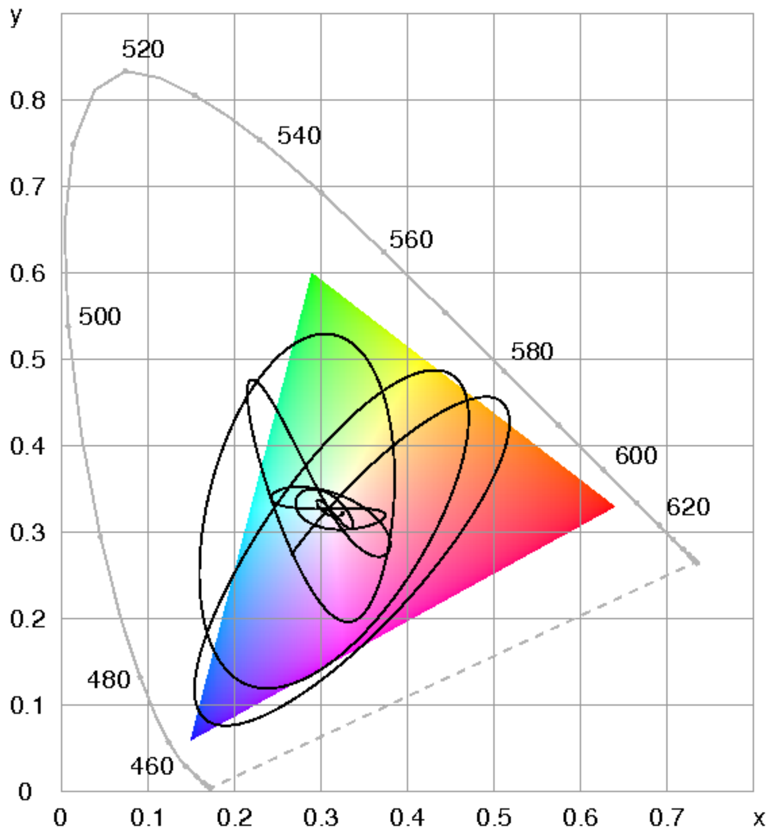

Figure 1: The CIE 1931 xy diagram with the gamut of colours which can be produced with the sRGB primaries, and the locus of the soap film colours as seen in reflection. The starting point at zero thickness is in the bluish grey.

First, from the colour coordinates the tristimulus values of the primaries are obtained

$$\begin{aligned} X_\mathrm{R} &= u_1 x_\mathrm{R} & Y_\mathrm{R} &= u_1 y_\mathrm{R} & Z_\mathrm{R} &= u_1(1 - x_\mathrm{R} - y_\mathrm{R}) \\ X_\mathrm{G} &= u_2 x_\mathrm{G} & Y_\mathrm{G} &= u_2 y_\mathrm{G} & Z_\mathrm{G} &= u_2(1 - x_\mathrm{G} - y_\mathrm{G}) \\ X_\mathrm{B} &= u_3 x_\mathrm{B} & Y_\mathrm{B} &= u_3 y_\mathrm{B} & Z_\mathrm{B} &= u_3(1 - x_\mathrm{B} - y_\mathrm{B})\,, \end{aligned} \tag{13}$$

as well as those of the white point:

$$X_\mathrm{W} = x_\mathrm{W}/y_\mathrm{W} \qquad Y_\mathrm{W} = 1.0 \qquad Z_\mathrm{W} = (1 - x_\mathrm{W} - y_\mathrm{W})/y_\mathrm{W}\,. \tag{14}$$

The requirement that

$$\mathbf{R} + \mathbf{G} + \mathbf{B} = \mathbf{W} \tag{15}$$

yields three equations for the three unknowns u_1, u_2, u_3 which determine the relative strengths of the primaries.

We next express the primary **X** through **R**, **G**, **B**:

$$\begin{aligned} X_\mathrm{R} R_\mathrm{X} + X_\mathrm{G} G_\mathrm{X} + X_\mathrm{B} B_\mathrm{X} &= 1 \\ Y_\mathrm{R} R_\mathrm{X} + Y_\mathrm{G} G_\mathrm{X} + Y_\mathrm{B} B_\mathrm{X} &= 0 \\ Z_\mathrm{R} R_\mathrm{X} + Z_\mathrm{G} G_\mathrm{X} + Z_\mathrm{B} B_\mathrm{X} &= 0 \end{aligned} \tag{16}$$

The coefficients $X_\mathrm{R}, X_\mathrm{G} \ldots$ are known from eq. (13), the quantities $R_\mathrm{X}, G_\mathrm{X}, B_\mathrm{X}$ are the unknowns to be determined from this system of equations. Similarly for **Y** and **Z** we obtain

$$\begin{aligned} X_\mathrm{R} R_\mathrm{Y} + X_\mathrm{G} G_\mathrm{Y} + X_\mathrm{B} B_\mathrm{Y} &= 0 \\ Y_\mathrm{R} R_\mathrm{Y} + Y_\mathrm{G} G_\mathrm{Y} + Y_\mathrm{B} B_\mathrm{Y} &= 1 \\ Z_\mathrm{R} R_\mathrm{Y} + Z_\mathrm{G} G_\mathrm{Y} + Z_\mathrm{B} B_\mathrm{Y} &= 0 \end{aligned} \tag{17}$$

and

$$\begin{aligned} X_\mathrm{R} R_\mathrm{Z} + X_\mathrm{G} G_\mathrm{Z} + X_\mathrm{B} B_\mathrm{Z} &= 0 \\ Y_\mathrm{R} R_\mathrm{Z} + Y_\mathrm{G} G_\mathrm{Z} + Y_\mathrm{B} B_\mathrm{Z} &= 0 \\ Z_\mathrm{R} R_\mathrm{Z} + Z_\mathrm{G} G_\mathrm{Z} + Z_\mathrm{B} B_\mathrm{Z} &= 1\,. \end{aligned} \tag{18}$$

Now we have the complete transformation matrix to convert any set of tristimulus values X, Y, Z to R, G, B, the intensities of the sRGB primaries:

$$\begin{aligned} R &= R_\mathrm{X}X + R_\mathrm{Y}Y + R_\mathrm{Z}Z \\ G &= G_\mathrm{X}X + G_\mathrm{Y}Y + G_\mathrm{Z}Z \\ B &= B_\mathrm{X}X + B_\mathrm{Y}Y + B_\mathrm{Z}Z\,. \end{aligned} \tag{19}$$

Negative Values, Normalization, and Gamma Correction

As has been shown in figure 1, not all occurring colours can be reproduced within the gamut chosen, which means that some of the tristimulus values will turn out to be negative. These values will finally be put equal to zero. This will, however, also affect the brightness, and for highest possible accuracy, the remaining positive tristimulus values should be scaled down so that the brightness is not changed.

Assume that for some value of s we get negative values of $R(s)$. The brightness is proportional to

$$Y(s) = Y_\mathrm{R}R(s) + Y_\mathrm{G}G(s) + Y_\mathrm{B}B(s)\,. \tag{20}$$

If now the first term on the right hand side is missing, we have to multiply $G(s)$ and $B(s)$ by

$$Y(s)/(Y(s) - Y_\mathrm{R}R(s)) \tag{21}$$

to get the correct luminance. Negative values of $G(s)$ and $B(s)$ are treated analogously.

Colour stimulus functions are in most cases normalized in an arbitrary way. The primary intensities on the display, on the other hand, are restricted to values between 0 and 1, where 1 means maximum intensity. Therefore, in our example, after computing the tristimulus values R, G, B using equations (8, 9, 19) for all interesting values of s, one has to find the maximum value of $R(s), G(s)$, and $B(s)$, and divide all obtained tristimulus values by it.

Next we have to account for the nonlinearity of brightness perception. This is done according to the sRGB recommendations[2] as follows:

If $R, G, B \leq 0.00304$,

$$\begin{aligned} R' &= 12.92\ R \\ G' &= 12.92\ G \\ B' &= 12.92\ B \end{aligned} \tag{22}$$

else if $R, G, B > 0.00304$,

$$\begin{aligned} R' &= 1.055\ R^{1.0/2.4} - 0.055 \\ G' &= 1.055\ G^{1.0/2.4} - 0.055 \\ B' &= 1.055\ B^{1.0/2.4} - 0.055\ , \end{aligned} \tag{23}$$

where the nonlinear R', G', B' values are then used as input for colour setting (as in the PostScript language) or are converted to 24-bit encoding (8 bits/channel). Assuming the digital count 0 for black and 255 for white, this is

$$\begin{aligned} R_{\text{8bit}} &= 255\ R' \\ G_{\text{8bit}} &= 255\ G' \\ B_{\text{8bit}} &= 255\ B' \end{aligned} \tag{24}$$

and the resulting numbers are rounded to integer. This is the final result to be displayed, giving the colour as a function of the optical path difference s.

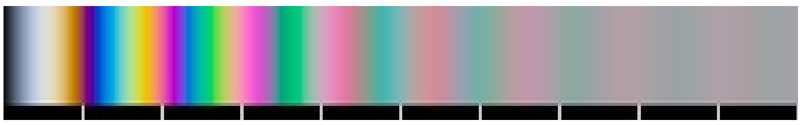

Figure 2: The result of the FORTRAN sample code of appendix A: colours of a tapered soap film in reflected light.

Figure 2 shows the result of the FORTRAN sample code shown in appendix A. The ticks mark the optical path difference between the two interfering rays and are at 500 nm, 1000 nm, 1500 nm . . .

The picture looks very realistic and compares well with a digital photograph, as has been shown; but it is interesting to see the locus of the colours in the CIE-chromaticity-diagram, as shown in figure 1: the curve does not fit into the gamut of colours provided by the sRGB primaries. This, however, does not impair the agreement with the photograph, as the photograph's colours are also restricted to the sRGB gamut. In fact, the soap film exhibits more vivid blue-green colours than can be displayed.

The FORTRAN code shown in the following appendix can also be viewed or downloaded from
http://www.itp.uni-hannover.de/˜zawischa/ITP/soapfilm.for

Appendix A: FORTRAN sample code

```
c    Colours of a thin film in reflected light.
c    Illumination: blackbody radiation,
c    colour temperature Tabs
c
      implicit real*8 (a-h,o-z)
      dimension iskalf(1000),icv(3,1000),Yuarray(3)
      real*4 colcomp(3,1000)
      common XuR,YuR,ZuR,XuG,YuG,ZuG,XuB,YuB,ZuB,
     1        RuX,GuX,BuX,RuY,GuY,BuY,RuZ,GuZ,BuZ,
     2        pi
c
      pi=3.141592653589d0
      Tabs=6504
      nhor=1000

c     sRGB primaries
      xr=0.64
      yr=0.33
      zr=1.d0-(xr+yr)
      xg=0.30
      yg=0.60
      zg=1.d0-(xg+yg)
      xb=0.15
      yb=0.06
      zb=1.d0-(xb+yb)

c     white D65
      xw=0.3127
      yw=0.3290
      zw=1.d0-(xw+yw)
```

```
!   Calibrate primaries to white
!   (As FORTRAN does not discriminate upper and lower case,
!   instead of X rather Xu is written, u like uppercase)
       XuW=xw/yw
       YuW=1.0d0
       ZuW=zw/yw
       call SSLE(xr,xg,xb,yr,yg,yb,zr,zg,zb,XuW,YuW,ZuW,
     *     u1,u2,u3)
       XuR=xr*u1
       YuR=yr*u1
       ZuR=zr*u1
       XuG=xg*u2
       YuG=yg*u2
       ZuG=zg*u2
       XuB=xb*u3
       YuB=yb*u3
       ZuB=zb*u3

!  the brightnesses of the primaries stored in array Yuarray
      Yuarray(1)=YuR
      Yuarray(2)=YuG
      Yuarray(3)=YuB

!  Determination of RGB-coordinates of the primaries XYZ.

c      superpose X from R, G, B
       call SSLE(XuR,XuG,XuB,YuR,YuG,YuB,ZuR,ZuG,ZuB,
     *     1.d0,0.d0,0.d0,RuX,GuX,BuX)
c      superpose Y from R, G, B
       call SSLE(XuR,XuG,XuB,YuR,YuG,YuB,ZuR,ZuG,ZuB,
     *     0.d0,1.d0,0.d0,RuY,GuY,BuY)
```

```
c      superpose Z from R, G, B
       call SSLE(XuR,XuG,XuB,YuR,YuG,YuB,ZuR,ZuG,ZuB,
     *     0.d0,0.d0,1.d0,RuZ,GuZ,BuZ)

c Transformation matrix is now ready.

c To go from XYZ to RGB one computes
c     RuColor = RuX*XuColor + RuY*YuColor + RuZ*ZuColor
c and analogously for G and B

c-------------------------------------------------------


!  Here starts the main loop

      do ic=1,1000
         ds=0.005d0*ic
!          Lengths are measured in mu-m (micrometers)
!          ds is the optical path difference of the
!          two interfering rays

!  compute XYZ
         call lambInt(XF,YF,ZF,ds,Tabs)

!  convert XYZ to RGB
         RuF=RuX*XF+RuY*YF+RuZ*ZF
         GuF=GuX*XF+GuY*YF+GuZ*ZF
         BuF=BuX*XF+BuY*YF+BuZ*ZF
           colcomp(1,ic)=RuF
           colcomp(2,ic)=GuF
           colcomp(3,ic)=BuF
      enddo   ! ic
```

```
!    Scale to make the maximum of R, G or B equal to 1,
!    Gamma-correction and conversion to 8-bit-format
       themaximum=0.d0
       do j=1,nhor
         do k=1,3
          if (colcomp(k,j) .gt. themaximum)
     *       themaximum=colcomp(k,j)
         enddo
       enddo
       ammaG=1.0d0/2.4d0
       do j=1,nhor
       Yu=(colcomp(1,j)*YuR+colcomp(2,j)*YuG+colcomp(3,j)*YuB)
     *    /themaximum      !  brightness
       do i=1,3
        colcomp(i,j)=colcomp(i,j)/themaximum
        if (colcomp(i,j).lt.0.0) then
           Corrf=Yu/(Yu-colcomp(i,j)*Yuarray(i))
           do k=1,3                            ! readjust brightness
             colcomp(k,j)=colcomp(k,j)*Corrf
           enddo
           colcomp(i,j)=0.0
        endif
        if (colcomp(i,j).le.0.00304) then
          colcomp(i,j)=colcomp(i,j)*12.92
          else
          colcomp(i,j)=1.055*colcomp(i,j)**ammaG-0.055
        endif
        icv(i,j)=int(255.0*colcomp(i,j))
       enddo
       enddo
```

```
!   Produce a ppm-image
!   http://netpbm.sourceforge.net/doc/ppm.html
       ndepth=255
       nvert=80
       open(unit=7,file='thinfilm.ppm',status='unknown')
       write(7,*) 'P3            '
       write(7,*) '# thinfilm.ppm  '
       write(7,*) nhor,nvert
       write(7,*) ndepth
       do j=1,nvert-10
        do i=1,nhor
         write(7,*) icv(1,i),icv(2,i),icv(3,i)
        enddo
       enddo
!   make scale-ticks in 500-nm-intervals
      do i=1,nhor
       iskalf(i)=0
      enddo
      do i=1,10
       iskalf(100*i-1)=255
       iskalf(100*i)=255
       if (i.lt.10) iskalf(100*i+1)=255
      enddo
      do j=nvert-9,nvert
       do i=1,nhor
        write(7,*) iskalf(i),iskalf(i),iskalf(i)
       enddo
      enddo
      stop
      end
c---------------------------------------------------------
```

```
      function Refl(s,alambd)
!    Reflection at thin film, not normalized
      implicit real*8 (a-h,o-z)
      common XuR,YuR,ZuR,XuG,YuG,ZuG,XuB,YuB,ZuB,
     1        RuX,GuX,BuX,RuY,GuY,BuY,RuZ,GuZ,BuZ,
     2        pi
      beta=0.95d0
      Refl=1.d0+beta*beta-2.d0*beta*cos(s*2.d0*pi/alambd)
      end
c--------------------------------------------------------
        subroutine lambInt(Xw,Yw,Zw,ds,Tabs)
        implicit real*8 (a-h,o-z)
        real*4 xbar(95),ybar(95),zbar(95)
        common XuR,YuR,ZuR,XuG,YuG,ZuG,XuB,YuB,ZuB,
     1        RuX,GuX,BuX,RuY,GuY,BuY,RuZ,GuZ,BuZ,
     2        pi

!   Colour matching functions, 5 nm intervals,
!   starting at 360 nm
!   Source:
!   http://cvrl.ioo.ucl.ac.uk/database/data/cmfs/ciexyz31.txt
!   (new ordering)
        data (xbar(i),i=1,95)/
     *   0.129900E-03,  0.232100E-03,  0.414900E-03,
     *   0.741600E-03,  0.136800E-02,  0.223600E-02,
     *   0.424300E-02,  0.765000E-02,  0.143100E-01,
     *   0.231900E-01,  0.435100E-01,  0.776300E-01,
     *   0.134380E+00,  0.214770E+00,  0.283900E+00,
     *   0.328500E+00,  0.348280E+00,  0.348060E+00,
     *   0.336200E+00,  0.318700E+00,  0.290800E+00,
     *   0.251100E+00,  0.195360E+00,  0.142100E+00,
     *   0.956400E-01,  0.579500E-01,  0.320100E-01,
     *    0.147000E-01,  0.490000E-02,  0.240000E-02,
     *    0.930000E-02,  0.291000E-01,  0.632700E-01,
     *    0.109600E+00,  0.165500E+00,  0.225750E+00,
     *    0.290400E+00,  0.359700E+00,  0.433450E+00,
     *    0.512050E+00,  0.594500E+00,  0.678400E+00,
     *    0.762100E+00,  0.842500E+00,  0.916300E+00,
     *    0.978600E+00,  0.102630E+01,  0.105670E+01,
     *    0.106220E+01,  0.104560E+01,  0.100260E+01,
     *    0.938400E+00,  0.854450E+00,  0.751400E+00,
     *    0.642400E+00,  0.541900E+00,  0.447900E+00,
     *    0.360800E+00,  0.283500E+00,  0.218700E+00,
     *    0.164900E+00,  0.121200E+00,  0.874000E-01,
     *    0.636000E-01,  0.467700E-01,  0.329000E-01,
     *    0.227000E-01,  0.158400E-01,  0.113592E-01,
     *    0.811092E-02,  0.579035E-02,  0.410946E-02,
     *    0.289933E-02,  0.204919E-02,  0.143997E-02,
     *    0.999949E-03,  0.690079E-03,  0.476021E-03,
     *    0.332301E-03,  0.234826E-03,  0.166150E-03,
     *    0.117413E-03,  0.830753E-04,  0.587065E-04,
     *    0.415099E-04,  0.293533E-04,  0.206738E-04,
     *    0.145598E-04,  0.102540E-04,  0.722146E-05,
     *    0.508587E-05,  0.358165E-05,  0.252252E-05,
     *    0.177651E-05,  0.125114E-05/
        data (ybar(i),i=1,95)/
     *    0.391700E-05,  0.696500E-05,  0.123900E-04,
     *    0.220200E-04,  0.390000E-04,  0.640000E-04,
     *    0.120000E-03,  0.217000E-03,  0.396000E-03,
     *    0.640000E-03,  0.121000E-02,  0.218000E-02,
     *    0.400000E-02,  0.730000E-02,  0.116000E-01,
     *    0.168400E-01,  0.230000E-01,  0.298000E-01,
     *    0.380000E-01,  0.480000E-01,  0.600000E-01,
     *    0.739000E-01,  0.909800E-01,  0.112600E+00,
*   0.139020E+00,  0.169300E+00,  0.208020E+00,
*   0.258600E+00,  0.323000E+00,  0.407300E+00,
*   0.503000E+00,  0.608200E+00,  0.710000E+00,
*   0.793200E+00,  0.862000E+00,  0.914850E+00,
*   0.954000E+00,  0.980300E+00,  0.994950E+00,
*   0.100000E+01,  0.995000E+00,  0.978600E+00,
*   0.952000E+00,  0.915400E+00,  0.870000E+00,
*   0.816300E+00,  0.757000E+00,  0.694900E+00,
*   0.631000E+00,  0.566800E+00,  0.503000E+00,
*   0.441200E+00,  0.381000E+00,  0.321000E+00,
*   0.265000E+00,  0.217000E+00,  0.175000E+00,
*   0.138200E+00,  0.107000E+00,  0.816000E-01,
*   0.610000E-01,  0.445800E-01,  0.320000E-01,
*   0.232000E-01,  0.170000E-01,  0.119200E-01,
*   0.821000E-02,  0.572300E-02,  0.410200E-02,
*   0.292900E-02,  0.209100E-02,  0.148400E-02,
*   0.104700E-02,  0.740000E-03,  0.520000E-03,
*   0.361100E-03,  0.249200E-03,  0.171900E-03,
*   0.120000E-03,  0.848000E-04,  0.600000E-04,
*   0.424000E-04,  0.300000E-04,  0.212000E-04,
*   0.149900E-04,  0.106000E-04,  0.746570E-05,
*   0.525780E-05,  0.370290E-05,  0.260780E-05,
*   0.183660E-05,  0.129340E-05,  0.910930E-06,
*   0.641530E-06,  0.451810E-06/
   data (zbar(i),i=1,95)/
*   0.606100E-03,  0.108600E-02,  0.194600E-02,
*   0.348600E-02,  0.645000E-02,  0.105500E-01,
*   0.200500E-01,  0.362100E-01,  0.678500E-01,
*   0.110200E+00,  0.207400E+00,  0.371300E+00,
*   0.645600E+00,  0.103905E+01,  0.138560E+01,
*   0.162296E+01,  0.174706E+01,  0.178260E+01,
*   0.177211E+01,  0.174410E+01,  0.166920E+01,
     *   0.152810E+01,  0.128764E+01,  0.104190E+01,
     *   0.812950E+00,  0.616200E+00,  0.465180E+00,
     *   0.353300E+00,  0.272000E+00,  0.212300E+00,
     *   0.158200E+00,  0.111700E+00,  0.782500E-01,
     *   0.572500E-01,  0.421600E-01,  0.298400E-01,
     *   0.203000E-01,  0.134000E-01,  0.875000E-02,
     *   0.575000E-02,  0.390000E-02,  0.275000E-02,
     *   0.210000E-02,  0.180000E-02,  0.165000E-02,
     *   0.140000E-02,  0.110000E-02,  0.100000E-02,
     *   0.800000E-03,  0.600000E-03,  0.340000E-03,
     *   0.240000E-03,  0.190000E-03,  0.100000E-03,
     *   0.500000E-04,  0.300000E-04,  0.200000E-04,
     *   0.100000E-04,  0.000000E+00,  0.000000E+00,
     *   0.000000E+00,  0.000000E+00,  0.000000E+00,
     *   0.000000E+00,  0.000000E+00,  0.000000E+00,
     *   0.000000E+00,  0.000000E+00,  0.000000E+00,
     *   0.000000E+00,  0.000000E+00,  0.000000E+00,
     *   0.000000E+00,  0.000000E+00,  0.000000E+00,
     *   0.000000E+00,  0.000000E+00,  0.000000E+00,
     *   0.000000E+00,  0.000000E+00,  0.000000E+00,
     *   0.000000E+00,  0.000000E+00,  0.000000E+00,
     *   0.000000E+00,  0.000000E+00,  0.000000E+00,
     *   0.000000E+00,  0.000000E+00,  0.000000E+00,
     *   0.000000E+00,  0.000000E+00,  0.000000E+00,
     *   0.000000E+00,  0.000000E+00/

      alf=14.388e3
!   alf = h.c/kB (Planck constant times velocity of light
!   divided by Boltzmann constant)
      alamo=0.560
!   the spectral distribution sl of the illumination is
!   normalized to 1 at this wavelength
```

```
        Xw=0.
        Yw=0.
        Zw=0.
        do i=1,95
        wlambda=0.355+0.005*i
        sl=(alamo/wlambda)**5*(exp(alf/(Tabs*alamo))-1.0)
     +      /(exp(alf/(Tabs*wlambda))-1.0)
        hil=Refl(ds,wlambda)*sl
        Xw=Xw+xbar(i)*hil
        Yw=Yw+ybar(i)*hil
        Zw=Zw+zbar(i)*hil
        enddo
        return
        end
c---------------------------------------------------------
      subroutine SSLE(a11,a12,a13,a21,a22,a23,a31,a32,a33,
     1     b1,b2,b3,x,y,z)
!   solve system of 3 linear equations
!   A_ik . x_k = b_i
      implicit real*8 (a-h,o-z)
      det=a11*a22*a33+a12*a23*a31+a13*a21*a32
     1  -a13*a22*a31-a12*a21*a33-a11*a23*a32
      x =(b1*a22*a33+a12*a23*b3+a13*b2*a32
     1  -a13*a22*b3-a12*b2*a33-b1*a23*a32)/det
      y =(a11*b2*a33+b1*a23*a31+a13*a21*b3
     1  -a13*b2*a31-b1*a21*a33-a11*a23*b3)/det
      z =(a11*a22*b3+a12*b2*a31+b1*a21*a32
     1  -b1*a22*a31-a12*a21*b3-a11*b2*a32)/det
      return
      end
c---------------------------------------------------------
```