Variation of the refractive index with Temperature and Salinity variations

```
%
% READ ME:
% this little program aims at defining the variations of refractive index
% with the Temperature and Salinity variations.
%


clear all
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% uncomment for user commands variables%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% prompt='lambda? (in microm)';
% lambda=input(prompt) %0.589; %definition of lambda, the wavelength at which the
module is tested
%
% prompt='Boundary Layer thickness? (mm)';
% BL=input(prompt)
%
% prompt='Boundary Layer Delta T? (°K)';
% DeltaT=input(prompt)
%
% prompt='Bulk T? (°K, =<353°K)';
% BulkT=input(prompt)
%
% [rho,cumulabs,Lmax]=bendingbeamIn(lambda, BL, DeltaT,BulkT);
% plot(cumulabs(:,1), cumulabs(:,2))

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% uncomment for defined curves plotting%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% lambda=0.589;
% BL=1;
% DeltaT1=10;
% DeltaT2=5;
% BulkT=353; % =<353°K
%
% [rho1,cumulabs1,Lmax1]=bendingbeamIn(lambda, BL, DeltaT1,BulkT);
% [rho2,cumulabs2,Lmax2]=bendingbeamIn(lambda, BL, DeltaT2,BulkT);
% plot(cumulabs1(:,1), cumulabs1(:,2),'g',cumulabs2(:,1), cumulabs2(:,2),'b')


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% uncomment for several curves ploted  %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

lambda=[0.524,0.628,0.490,0.639];
BulkT=[293.15,313.15,343.15];
BulkS=[10];
DeltaTMax=5;
DeltaSMax=5;
```

```
nTTab=zeros(DeltaTMax+1,length(lambda),length(BulkT),length(BulkS)); %refraction
indexes
nSTab=zeros(DeltaSMax+1,length(lambda),length(BulkT),length(BulkS));
rhoTTab=zeros(DeltaTMax+1,length(lambda),length(BulkT),length(BulkS)); %refraction
indexes
rhoSTab=zeros(DeltaSMax+1,length(lambda),length(BulkT),length(BulkS));
CTTTab=zeros(DeltaTMax+1,length(lambda),length(BulkT),length(BulkS)); %refraction
indexes
CTSTab=zeros(DeltaSMax+1,length(lambda),length(BulkT),length(BulkS));

for i=1:length(lambda)
    for j=1:length(BulkT)
        for k=1:length(BulkS)
                [nT,nS,rhoT,rhoS,CTT,CTS]=DeltaNDeltaTS(lambda(i), BulkT(j),
BulkS(k),DeltaTMax,DeltaSMax);
                nTTab(:,i,j,k)=nT;
                nSTab(:,i,j,k)=nS;
                rhoTTab(:,i,j,k)=rhoT;
                rhoSTab(:,i,j,k)=rhoS;
                CTTTab(:,i,j,k)=CTT;
                CTSTab(:,i,j,k)=CTS;
        end
    end
end


j=1;
k=1;
for i=1:2%length(lambda)
    figure(1);
    plot([[-DeltaTMax:1:0],[0:1:DeltaSMax]], [(nTTab(:,i,j,k)-
nTTab(DeltaTMax+1,i,j,k))',(nSTab(:,i,j,k)-nSTab(1,i,j,k))'])
    title('Variations of n with T and S gradients')
    xlabel('T gradient (K) and S gradient (g/kg)')
    ylabel('Delta n')
    hold on
end


j=1;
k=1;
    figure(2);
    plot([[-DeltaTMax:1:0],[0:1:DeltaSMax]], [(nTTab(:,1,j,k)-
nTTab(DeltaTMax+1,1,j,k))'-(nTTab(:,2,j,k)-
nTTab(DeltaTMax+1,2,j,k))',(nSTab(:,1,j,k)-nSTab(1,1,j,k))'-(nSTab(:,2,j,k)-
nSTab(1,2,j,k))'])
    title('Variations of n with T and S gradients')
    xlabel('T gradient (K) and S gradient (g/kg)')
    ylabel('Delta n')
    hold on

    j=1;
k=1;
for i=2:length(lambda)%1:length(lambda)
```

```
    figure(3);
    plot([[-DeltaTMax:1:0],[0:1:DeltaSMax]], [(nTTab(:,i,j,k)-
nTTab(DeltaTMax+1,i,j,k))',(nSTab(:,i,j,k)-nSTab(1,i,j,k))'])
    title('Variations of n with T and S gradients')
    xlabel('T gradient (K) and S gradient (g/kg)')
    ylabel('Delta n')
    hold on
end


j=1;
k=1;
    figure(4);
    plot([[-DeltaTMax:1:0],[0:1:DeltaSMax]], [(nTTab(:,3,j,k)-
nTTab(DeltaTMax+1,3,j,k))'-(nTTab(:,4,j,k)-
nTTab(DeltaTMax+1,4,j,k))',(nSTab(:,3,j,k)-nSTab(1,3,j,k))'-(nSTab(:,4,j,k)-
nSTab(1,4,j,k))'])
    title('Variations of n with T and S gradients')
    xlabel('T gradient (K) and S gradient (g/kg)')
    ylabel('Delta n')
    hold on


function [nTVar, nSVar, rhoVarT, rhoVarS, CTT, CTS]=DeltaNDeltaTS(lambdaVar,
BulkTVar, BulkSVar, DeltaT, DeltaS)


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                       %
% Calcul of rho as a function of temperature, salinity and pressure %
%                                                                       %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%charging the GSW library to solve the TEOS-10 (from http://www.teos-10.org)
addpath C:\Users\marie-alix.dalle\Documents\GSW
addpath C:\Users\marie-alix.dalle\Documents\GSW\html
addpath C:\Users\marie-alix.dalle\Documents\GSW\library
addpath C:\Users\marie-alix.dalle\Documents\GSW\pdf
addpath C:\Users\marie-alix.dalle\Documents\GSW\thermodynamics_from_t


tK=BulkTVar+[-DeltaT:1:0]'; %temperature values, in °K
t=tK-273.15; %temperature values, in °C
SA=BulkSVar+[0:1:DeltaS]'; %absolute salinity (g.kg-1)
pT=10*ones(DeltaT+1,1); %pressure (dbar)
pS=10*ones(DeltaS+1,1); %pressure (dbar)

CTT=gsw_CT_from_t(BulkSVar*ones(DeltaT+1,1),t,pT); %Converting in situ T into
conservative T with constant salinity and T gradient
CTS=gsw_CT_from_t(SA,BulkTVar*ones(DeltaS+1,1)-273.15,pS); %Converting in situ T into
conservative T with constant temperature and S gradient

% CTSunit=gsw_CT_from_t(BulkSVar,BulkTVar-273.15,10); %Converting in situ T into
conservative T with constant temperature and S gradient
```

```
rhoVarT=gsw_rho(BulkSVar*ones(DeltaT+1,1),CTT,pT); %density with constant salinity
and T gradient
rhoVarS=gsw_rho(SA,CTS,pS); %density with constant temperature and S gradient


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                        %
% Calcul of n as a function of temperature, density and wavelength  %
%                                                                        %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% setting the equation coefficients (from Schiebener, Refractive [...], 1990)

a0=0.243905091;
a1=0.009535181;
a2=-0.003643581;
a3=0.000265666;
a4=0.001591893;
a5=0.002457338;
a6=0.897478251;
a7=-0.016306618;
lambdaUV=0.229202;
lambdaIR=5.432937;

lambda0=0.589;      % microm
rho0=1000;   % kg.m^-3
T0=273.15;   % K


alphaT=a0+a1*rhoVarT/rho0+a2*tK/T0+a3*lambdaVar/lambda0+a4*(lambdaVar/lambda0)^2+a5/(
(lambdaVar/lambda0)^2-lambdaUV^2)+a6/((lambdaVar/lambda0)^2-
lambdaIR^2)+a7*(rhoVarT/rho0).^2;

nTVar=((2*alphaT.*rhoVarT/rho0+1)./(1-alphaT.*rhoVarT/rho0)).^(1/2);

alphaS=a0+a1*rhoVarS/rho0+a2*BulkTVar*ones(DeltaS+1,1)/T0+a3*lambdaVar/lambda0+a4*(la
mbdaVar/lambda0)^2+a5/((lambdaVar/lambda0)^2-lambdaUV^2)+a6/((lambdaVar/lambda0)^2-
lambdaIR^2)+a7*(rhoVarS/rho0).^2;

nSVar=((2*alphaS.*rhoVarS/rho0+1)./(1-alphaS.*rhoVarS/rho0)).^(1/2);


end
```

Steady State profile

```
% READ ME
% This programm has for aim to determine the steady-state profile of the
% flow in the hot channel.

clear all

W=0.01; % width of the hot channel (in m)
H=0.005; % height of the hot channel (in m)
Q=5.75*10^(-6); % m3/s-1
mu=1*10^(-3); % Pa.s-1 dynamic viscosity of water
h=H/10; % step size
NW=int64(W/h); % number of points in the y direction
NH=int64(H/h); % number of points in the z direction



%N=5; % number of points per direction
%hW=W/N;
%hH=H/N;

B=diag(-4*ones(1,NH))+diag(1*ones(1,NH-1),1)+diag(1*ones(1,NH-1),-1);
C=diag(ones(1,NH));


testB=cell(1,NW);
for i=1:NW
    testB{i}=B;
end
AB=blkdiag(testB{1,:});


testC=cell(1,NW-1);
for i=1:NW-1
    testC{i}=C;
end
AC=blkdiag(testC{1,:});


ACup=[zeros((NW-1)*NH,NH),AC;zeros(NH,NW*NH)];
ACdown=[zeros(NH,NW*NH);AC,zeros((NW-1)*NH,NH)];

A=AB+ACup+ACdown;
A=1/h^2*A;

dP_x=mu*Q/(sum(sum(inv(A)*ones(NH*NW,1)))*h^2);
Uh=1/mu*dP_x*inv(A)*ones(NH*NW,1);

U=Uh(1:NH).';
Y=[h:h:W].';
Z=[h:h:H];
```

```
for i=1:NW-1
    U=[U;Uh(i*NH+1:i*NH+NH).'];
    Z=[Z;(h:h:H)];
end

for i=1:NH-1
    Y=[Y,(h:h:W).'];
end

%plot3(Y,Z,U)
surf(Y,Z,U)
axis equal
```

Image interpretation Code

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Image interpretation
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% The overall goal of this code is to reconstruct the temperature gradient
% from the intensity gradient


clear all
close all

% General parameters to change
%-------------------------------------------------------------------------

fl1=500; %(mm) focal length of the first lens
L=10; %(mm) width of the hot channel
TBulk=0; %(celsius) bulk temperature
treshold=0.4; %treshold to distinguish light from dark
nair=1.000272; %air index
nwater=1.33; %water index
pixsize=1.95*10^(-3);%(mm)
offset=30; %value to avoid boundary effects when fitting the intensity profiles

% Reading the image without T gradient and transforming into singles
%-------------------------------------------------------------------------
%withoutTGrad=imread('DSCF1780.jpg');
%withoutTGrad=imread('DSCF1793.jpg');
%withoutTGrad=imread('DSCF1801.jpg');
%withoutTGrad=imread('DSCF2627.jpg');
%withoutTGrad=imread('averageBackgroundG.jpg');
grayImageWTG=im2double(imread('averageBackgroundB.jpg'));

% [rows, columns, numberofColorChannels] = size(withoutTGrad);
% if numberofColorChannels > 1
%   grayImageWTG = rgb2gray(withoutTGrad);
%   grayImageWTG=im2single(grayImageWTG);
% else
%   grayImageWTG  = withoutTGrad;
%   grayImageWTG=im2single(grayImageWTG);
% end

% Reading the image with T gradient and transforming into singles
%-------------------------------------------------------------------------
%TGrad=imread('DSCF1781.jpg');
%TGrad=imread('DSCF1792.jpg');
%TGrad=imread('DSCF1800.jpg');
%TGrad=imread('DSCF2626.jpg');
%TGrad=imread('DSCF2718.jpg');
%TGrad=imread('average1840G.jpg');
grayImageTG=im2double(imread('average2023B.jpg'));
```

```
nameOfFile="20211223_1958_corr5test_err_LowPass";

%[rows, columns, numberofColorChannels] = size(TGrad);
[rows, columns, numberofColorChannels] = size(grayImageTG);

% if numberofColorChannels > 1
%   grayImageTG = rgb2gray(TGrad);
%   grayImageTG = im2single(grayImageTG);
% else
%   grayImageTG = TGrad;
%   grayImageTG = im2single(grayImageTG);
% end

plot((1:1:4000)*pixsize,grayImageWTG(:,2000))
%plot(1:1:6000,grayImageWTG(3000,:))


% Selecting only the bright part of the image
%-------------------------------------------------------------------------

jmintab=zeros(1,rows);
jmaxtab=zeros(1,rows);

j=1;
for i=1:rows
j=1;
    while (j<columns && grayImageWTG(i,j)<treshold)
        j=j+1;
    end
    j=j;
    jmintab(1,i)=j-1;
end

for i=1:rows
j=1;
    while (j<columns && grayImageWTG(i,columns-j)<treshold)
        j=j+1;
    end
    jmaxtab(1,i)=columns-j+1;
end

jmin=min(jmintab);
jmax=max(jmaxtab);


imintab=zeros(1,jmax-jmin+1);
imaxtab=zeros(1,jmax-jmin+1);
i=1;
for j=1:columns
i=1;
    while (i<rows && grayImageWTG(i,j)<treshold)
        i=i+1;
    end
    imintab(1,j)=i-1;
end
```

```
for j=1:columns
i=1;
    while (i<rows && grayImageWTG(rows-i,j)<treshold)
        i=i+1;
    end
    imaxtab(1,j)=rows-i+1;
end

imin=min(imintab);
imax=max(imaxtab);

%imin=max(imin,1)+500; %to be sure to have only the middle of the channel
%imax=imax-1000; %to be sure to have only the middle of the channel


% for the other picture
jmintab2=zeros(1,rows);
jmaxtab2=zeros(1,rows);

j=1;
for i=1:rows
j=1;
    while (j<columns && grayImageTG(i,j)<treshold)
        j=j+1;
    end
    j=j;
    jmintab2(1,i)=j-1;
end

for i=1:rows
j=1;
    while (j<columns && grayImageTG(i,columns-j)<treshold)
        j=j+1;
    end
    jmaxtab2(1,i)=columns-j+1;
end

jmin2=min(jmintab2);
jmax2=max(jmaxtab2);


imintab2=zeros(1,jmax2-jmin2+1);
imaxtab2=zeros(1,jmax2-jmin2+1);
i=1;
for j=1:columns
i=1;
    while (i<rows && grayImageTG(i,j)<treshold)
        i=i+1;
    end
    imintab2(1,j)=i-1;
end

for j=1:columns
i=1;
```

```
    while (i<rows && grayImageTG(rows-i,j)<treshold)
        i=i+1;
    end
    imaxtab2(1,j)=rows-i+1;
end

imin2=min(imintab2);
imax2=max(imaxtab2);

%imin2=max(imin2,1)+500; %to be sure to have only the middle of the channel
%imax2=imax2-1000; %to be sure to have only the middle of the channel


imin=max(1,min(imin, imin2));
imax=max(imax, imax2);
jmin=max(1,min(jmin, jmin2));
jmax=max(jmax, jmax2);

imin=max(imin,1)+200;%+500; %to be sure to have only the middle of the channel
imax=imax;%-1000; %to be sure to have only the middle of the channel


%imin=1200;
%imax=1900;
%jmin=1985;
%jmax=3601;

%imin=140;
%imax=2388;
%jmin=1795;
%jmax=3344;

grayImageWTGS=grayImageWTG(imin:1:imax,jmin:1:jmax);
grayImageTGS=grayImageTG(imin:1:imax,jmin:1:jmax);
difference=grayImageTGS-grayImageWTGS;

[rowsS, columnsS, numberofColorChannels] = size(grayImageTGS);

rowsS=rowsS-500;
% Calculating the translation between pictures and determining the border
%-------------------------------------------------------------------------

grayImageWTGSBW=grayImageWTGS;
grayImageTGSBW=grayImageTGS;

for i=1:rowsS
 for j=1:columnsS
    if grayImageWTGS(i,j)<treshold*0.9
        grayImageWTGSBW(i,j)=0;
    elseif grayImageWTGS(i,j)>treshold*1
        grayImageWTGSBW(i,j)=treshold*1;
    end

    if grayImageTGS(i,j)<treshold*0.9
        grayImageTGSBW(i,j)=0;
```

```
    elseif grayImageTGS(i,j)>treshold*1
        grayImageTGSBW(i,j)=treshold*1;
    end
 end
end

%grayImageTGSBW=medfilt2(grayImageTGSBW,[5,5]);
%grayImageWTGSBW=medfilt2(grayImageWTGSBW,[5,5]);
grayImageTGSBW=imbinarize(grayImageTGS);
grayImageWTGSBW=imbinarize(grayImageWTGS);

% loop=1;
% for loop=1:100
% for i=3:rowsS-2
%  for j=3:columnsS-2
%      %if (grayImageTGSBW(i-1,j-1)+grayImageTGSBW(i-1,j)+grayImageTGSBW(i-
1,j+1)+grayImageTGSBW(i,j-1)+grayImageTGSBW(i,j+1)+grayImageTGSBW(i+1,j-
1)+grayImageTGSBW(i+1,j)+grayImageTGSBW(i+1,j+1)>6*treshold)
%      if (grayImageTGSBW(i-2,j-2)+grayImageTGSBW(i-2,j-1)+grayImageTGSBW(i-
2,j)+grayImageTGSBW(i-2,j+1)+grayImageTGSBW(i-2,j+2)+grayImageTGSBW(i-1,j-
2)+grayImageTGSBW(i-1,j-1)+grayImageTGSBW(i-1,j)+grayImageTGSBW(i-
1,j+1)+grayImageTGSBW(i-1,j+2)+grayImageTGSBW(i,j-2)+grayImageTGSBW(i,j-
1)+grayImageTGSBW(i,j+1)+grayImageTGSBW(i,j+2)+grayImageTGSBW(i+1,j-
2)+grayImageTGSBW(i+1,j-
1)+grayImageTGSBW(i+1,j)+grayImageTGSBW(i+1,j+1)+grayImageTGSBW(i+1,j+2)+grayImageTGS
BW(i+2,j-2)+grayImageTGSBW(i+2,j-
1)+grayImageTGSBW(i+2,j)+grayImageTGSBW(i+2,j+1)+grayImageTGSBW(i+2,j+2)>16*treshold)
%         grayImageTGSBW(i,j)=treshold*1.1;
%      end
%
%      %if (grayImageWTGSBW(i-1,j-1)+grayImageWTGSBW(i-1,j)+grayImageWTGSBW(i-
1,j+1)+grayImageWTGSBW(i,j-1)+grayImageWTGSBW(i,j+1)+grayImageWTGSBW(i+1,j-
1)+grayImageWTGSBW(i+1,j)+grayImageWTGSBW(i+1,j+1)>6*treshold)
%      if (grayImageWTGSBW(i-2,j-2)+grayImageWTGSBW(i-2,j-1)+grayImageWTGSBW(i-
2,j)+grayImageWTGSBW(i-2,j+1)+grayImageWTGSBW(i-2,j+2)+grayImageWTGSBW(i-1,j-
2)+grayImageWTGSBW(i-1,j-1)+grayImageWTGSBW(i-1,j)+grayImageWTGSBW(i-
1,j+1)+grayImageWTGSBW(i-1,j+2)+grayImageWTGSBW(i,j-2)+grayImageWTGSBW(i,j-
1)+grayImageWTGSBW(i,j+1)+grayImageWTGSBW(i,j+2)+grayImageWTGSBW(i+1,j-
2)+grayImageWTGSBW(i+1,j-
1)+grayImageWTGSBW(i+1,j)+grayImageWTGSBW(i+1,j+1)+grayImageWTGSBW(i+1,j+2)+grayImage
WTGSBW(i+2,j-2)+grayImageWTGSBW(i+2,j-
1)+grayImageWTGSBW(i+2,j)+grayImageWTGSBW(i+2,j+1)+grayImageWTGSBW(i+2,j+2)>16*tresho
ld)
%         grayImageWTGSBW(i,j)=treshold*1.1;
%      end
%
%      %if (grayImageTGSBW(i-1,j-1)+grayImageTGSBW(i-1,j)+grayImageTGSBW(i-
1,j+1)+grayImageTGSBW(i,j-1)+grayImageTGSBW(i,j+1)+grayImageTGSBW(i+1,j-
1)+grayImageTGSBW(i+1,j)+grayImageTGSBW(i+1,j+1)<4*treshold)
%      if (grayImageTGSBW(i-2,j-2)+grayImageTGSBW(i-2,j-1)+grayImageTGSBW(i-
2,j)+grayImageTGSBW(i-2,j+1)+grayImageTGSBW(i-2,j+2)+grayImageTGSBW(i-1,j-
2)+grayImageTGSBW(i-1,j-1)+grayImageTGSBW(i-1,j)+grayImageTGSBW(i-
1,j+1)+grayImageTGSBW(i-1,j+2)+grayImageTGSBW(i,j-2)+grayImageTGSBW(i,j-
1)+grayImageTGSBW(i,j+1)+grayImageTGSBW(i,j+2)+grayImageTGSBW(i+1,j-
2)+grayImageTGSBW(i+1,j-
```

```
1)+grayImageTGSBW(i+1,j)+grayImageTGSBW(i+1,j+1)+grayImageTGSBW(i+1,j+2)+grayImageTGS
BW(i+2,j-2)+grayImageTGSBW(i+2,j-
1)+grayImageTGSBW(i+2,j)+grayImageTGSBW(i+2,j+1)+grayImageTGSBW(i+2,j+2)<12*treshold)
%         grayImageTGSBW(i,j)=0;
%      end
%
%      %if (grayImageWTGSBW(i-1,j-1)+grayImageWTGSBW(i-1,j)+grayImageWTGSBW(i-
1,j+1)+grayImageWTGSBW(i,j-1)+grayImageWTGSBW(i,j+1)+grayImageWTGSBW(i+1,j-
1)+grayImageWTGSBW(i+1,j)+grayImageWTGSBW(i+1,j+1)<4*treshold)
%      if (grayImageWTGSBW(i-2,j-2)+grayImageWTGSBW(i-2,j-1)+grayImageWTGSBW(i-
2,j)+grayImageWTGSBW(i-2,j+1)+grayImageWTGSBW(i-2,j+2)+grayImageWTGSBW(i-1,j-
2)+grayImageWTGSBW(i-1,j-1)+grayImageWTGSBW(i-1,j)+grayImageWTGSBW(i-
1,j+1)+grayImageWTGSBW(i-1,j+2)+grayImageWTGSBW(i,j-2)+grayImageWTGSBW(i,j-
1)+grayImageWTGSBW(i,j+1)+grayImageWTGSBW(i,j+2)+grayImageWTGSBW(i+1,j-
2)+grayImageWTGSBW(i+1,j-
1)+grayImageWTGSBW(i+1,j)+grayImageWTGSBW(i+1,j+1)+grayImageWTGSBW(i+1,j+2)+grayImage
WTGSBW(i+2,j-2)+grayImageWTGSBW(i+2,j-
1)+grayImageWTGSBW(i+2,j)+grayImageWTGSBW(i+2,j+1)+grayImageWTGSBW(i+2,j+2)<12*tresho
ld)
%         grayImageWTGSBW(i,j)=0;
%      end
%  end
% end
% loop=loop+1;
% end


tmax=800; %(pixel) maximum horizontal translation
vmax=800; %(pixel) maximum vertical translation
distancet=zeros(1,2*tmax+1);
distancev=zeros(1,2*vmax+1);


%horizontal translation
%only jmax/2 to avoid the bigger differences of the wall side
for t=-tmax:tmax
    %grayImageWTGSBWT = imtranslate(grayImageWTGSBW,[t, 0]);
    grayImageTGSBWT = imtranslate(grayImageTGSBW,[t, 0]);
    diff=grayImageWTGSBW(:,1:floor(columnsS/2))-
grayImageTGSBWT(:,1:floor(columnsS/2));
    distancet(1,tmax+t+1)=mean(mean(abs(diff)));
end

dmint=min(distancet(1,:));
ttest=1;
while distancet(1,ttest)~=dmint
    ttest=ttest+1;
end
tmin=ttest-1-1-tmax;

%vertical translation
%only jmax/2 to avoid the bigger differences of the wall side
for v=-vmax:vmax
    %grayImageWTGSBWT = imtranslate(grayImageWTGSBW,[t, 0]);
    grayImageTGSBWT = imtranslate(grayImageTGSBW,[tmin, v]);
```

```
    diff=grayImageWTGSBW(:,1:floor(columnsS/2))-
grayImageTGSBWT(:,1:floor(columnsS/2));
    distancev(1,vmax+v+1)=mean(mean(abs(diff)));
end
dminv=min(distancev(1,:));
vtest=1;
while distancev(1,vtest)~=dminv
    vtest=vtest+1;
end
vmin=vtest-1-1-vmax;

f1=figure;
plot((1:2*tmax+1)*pixsize,distancet(1,:))
f2=figure;
plot((1:2*vmax+1)*pixsize,distancev(1,:))

grayImageTGS = imtranslate(grayImageTG,[tmin, vmin]);
%grayImageTGSBW = imtranslate(grayImageTGBW,[tmin, vmin]);


grayImageWTGS=grayImageWTG(imin:1:imax,jmin:1:jmax);
grayImageTGS=grayImageTGS(imin:1:imax,jmin:1:jmax);


grayImageTGSBW=imbinarize(grayImageTGS);
grayImageWTGSBW=imbinarize(grayImageWTGS);


border=zeros(1,rowsS-vmin);

for i=1:rowsS-abs(vmin)
    j=1;
    while (grayImageWTGS(i,j)<treshold && j<size(grayImageWTGS,2))
        j=j+1;
    end
    border(i)=j;
end


wall=zeros(1,rowsS-vmin);

for i=1:rowsS-abs(vmin)
    j=size(grayImageWTGS,2);
    while (grayImageWTGS(i,j)<treshold && j>1)
        j=j-1;
    end
    wall(i)=j;
end

%----------------------------------------
% median filter applied to the gray image
%----------------------------------------

%%grayImageWTGSM=medfilt2(grayImageWTGS,[5,5]);
%%grayImageTGSM=medfilt2(grayImageTGS,[5,5]);
```

```
%grayImageWTGSM=medfilt2(grayImageWTGS,[10,10]);
%grayImageTGSM=medfilt2(grayImageTGS,[10,10]);

%----------------------------------------
% butterworth filter applied to the gray image
%----------------------------------------

sample_rate=1;

%you can play with these two below
lowpass_freq=0.008;
lowpass_order=2;
Fnorm=lowpass_freq/(sample_rate/2);

[num,den]=butter(lowpass_order,Fnorm);
grayImageWTGSM=filtfilt(num, den, grayImageWTGS);
grayImageTGSM=filtfilt(num, den, grayImageTGS);


% Calculating the exit angle and the corresponding dn and dT
%-------------------------------------------------------------------------

% dx=zeros(rowsS,columnsS);
% alpha=zeros(rowsS,columnsS);
% dn=zeros(rowsS,columnsS);
% dT=zeros(rowsS,columnsS);
%
% for i=1:rowsS
%     for j=1:columnsS
%         dx(i,j)=-45/4*(log10(grayImageTGS(i,j))-log10((grayImageWTGS(i,j))));
%         alpha(i,j)=atan(dx(i,j)/fl1);
%         dn(i,j)=alpha(i,j)/L;
%         dT(i,j)=dn(i,j)/0.0002;
%     end
% end
%
% dTmean=zeros(1,columnsS);
%
% for i=1:columnsS
%       dTmean(i)=mean(dT(:,i));
% end


% Display the images
%-------------------------------------------------------------------------

f3=figure;
imshow(grayImageWTGS)
title('Image without Temperature Gradient')
xlabel('x (mm)')
ylabel('y')


f4=figure;
imshow(grayImageTGS)
```

```
title('Image with Temperature Gradient')
xlabel('x (mm)')
ylabel('y')


f5=figure;
imshow(grayImageWTGSBW)
title('Black and White image without Temperature Gradient')
xlabel('x (mm)')
ylabel('y')


f6=figure;
imshow(grayImageTGSBW)
title('Black and White image with Temperature Gradient')
xlabel('x (mm)')
ylabel('y')


f7=figure;
imshow(grayImageTGSBW-grayImageWTGSBW)
title('Difference between Black and White images')
xlabel('x (mm)')
ylabel('y')


f3=figure;
imshow(grayImageWTGSM)
title('Image without Temperature Gradient with Median Filter')
xlabel('x (mm)')
ylabel('y')


f4=figure;
imshow(grayImageTGSM)
title('Image with Temperature Gradient with Median filter')
xlabel('x (mm)')
ylabel('y')

% Fit a polynomial curve to the dT
%-------------------------------------------------------------------------

%columnsS=1700;
columnmin=max(1,tmin);
columnsS=min(columnsS, columnsS+tmin)-200;

rowmin=max(1,vmin);
rowsS=min(rowsS,rowsS+vmin);

borderini=border;

%Sliding-mean filter on border to smooth the curve
% l=20;
% for i=1:3
%     for i=1+l:length(border)-l
```

```
%         border(i)=mean(border(i-l:i+l));
%     end
% end
%
% for i=1:3
%     for i=1+l:length(border)-l
%         bordermax(i)=max(border(i-l:i+l));
%     end
% end
%
% for i=1:3
%     for i=1+l:length(border)-l
%         bordermin(i)=min(border(i-l:i+l));
%     end
% end



%Butterworth Low-pass filter on border to smooth the curve
sample_rate=1;

%you can play with these two below
lowpass_freq=0.008;
lowpass_order=2;
Fnorm=lowpass_freq/(sample_rate/2);

[num,den]=butter(lowpass_order,Fnorm);
border=filtfilt(num,den,border);

bordermax=border+1.96*std(border-borderini)/sqrt(length(border));
bordermin=border-1.96*std(border-borderini)/sqrt(length(border));




fitwall=fit([1:size(wall,2)]',wall(1,1:size(wall,2))','poly2');
wallini=wall;
wall=fitwall;

pixresolution=mean(bordermax-bordermin);

%Finding the local minimums
ipoints=islocalmin(border,2,'MinSeparation',400);
ii=1;
for i=1:length(ipoints)
    if ipoints(1,i)==1
        xminpoints(1,ii)=i;
        ii=ii+1;
    end
end

jpoints=islocalmax(border,2,'MinSeparation',400);
jj=1;
for j=1:length(jpoints)
    if jpoints(1,j)==1
```

```
        xmaxpoints(1,jj)=j;
        jj=jj+1;
    end
end


minpoints=border(1,ipoints);
maxpoints=border(1,jpoints);

f=fit(xminpoints(1,:)',minpoints(1,:)','poly2');

figure
plot(f,[1:length(border(1,:))]',border(1,:)')
title('Raw border and its sliding mean')
xlabel('Pixels rows')
ylabel('Pixels columns')

figure
plot((1:length(borderini))*pixsize,borderini,(1:length(border))*pixsize,border)
title('Raw border and low-pass filtered border')
xlabel('x (mm)')
ylabel('y (px)')

figure
plot(borderini,(1:length(borderini))*pixsize,border,(1:length(border))*pixsize)
hold on
plot(border,(1:length(border))*pixsize,'y','LineWidth',2.0)
title('Raw border and low-pass filtered border')
xlabel('First light pixel (px)')
ylabel('Module length (mm)')
saveas(gcf, nameOfFile + "_MP",'jpeg')
%
% figure
% plot((1:length(bordermax))*pixsize,
borderini(1,1:length(bordermax)),(1:length(bordermax))*pixsize,
border(1,1:length(bordermax)), (1:length(bordermax))*pixsize, bordermax,
(1:length(bordermax))*pixsize, bordermin)
% title('Raw border, low-pass filtered border and imprecision range')
% xlabel('x (mm)')
% ylabel('y (px)')
% %saveas(gcf, nameOfFile + "_MP",'jpeg')

figure
plot(borderini(1,1:length(bordermax)),(1:length(bordermax))*pixsize)
hold on
plot(border(1,1:length(bordermax)),(1:length(bordermax))*pixsize, 'y',
bordermax,(1:length(bordermax))*pixsize, bordermin,
(1:length(bordermax))*pixsize,'LineWidth',2.0)
title('Raw border, low-pass filtered border and imprecision range')
xlabel('First light pixel (px)')
ylabel('Module length (mm)')
saveas(gcf, nameOfFile + "_MPvert",'jpeg')

figure
```

```
plot(([1:length(border(1,:))]')*pixsize,border(1,:)',(xminpoints(1,:)')*pixsize,minpo
ints(1,:)')
title('Fit of the border')
xlabel('y')
ylabel('x (mm)')

% for i=1:length(border(1,:))
%     bordertest(1,i)=round(abs(maxpoints(1,1)-
minpoints(1,1))/2*sin(2*3.14/(2*abs(xmaxpoints(1,1)-xminpoints(1,1)))*(i)-
4*3.14/8)+f(i)'+20);
% end

%to use border as the origin of the membrane instead of bordertest
bordertest=border;
for i=1:length(border(1,:))
     bordertest(1,i)=round(border(1,i));
end

border=bordertest;

figure
plot((1:length(border(1,:)))*pixsize,border(1,:),(1:length(border(1,:)))*pixsize,bord
ertest(1,:))
title('Fit of the border')
xlabel('y')
ylabel('x (mm)')



Tmean=zeros(rowsS-rowmin+1,columnsS-columnmin+1);
Tmean(:,1)=0;%TBulk;
Tmeanmin=zeros(rowsS-rowmin+1,columnsS-columnmin+1);
Tmeanmin(:,1)=0;%TBulk;
Tmeanmax=zeros(rowsS-rowmin+1,columnsS-columnmin+1);
Tmeanmax(:,1)=0;%TBulk;
dTreconstruct=zeros(rowsS-rowmin+1,columnsS-columnmin+1);



for i=1:rowsS-rowmin%+1
    jstart=max(2,bordertest(i));
    l=1;

% to remove the constant noise
%     for j=jstart+1:columnsS
%         if difference(i,j)~=0
%             grayImageTGSMWN(i,l,1)=j;
%             grayImageTGSMWN(i,l,2)=grayImageTGSM(i,j);
%
%             grayImageWTGSMWN(i,l,1)=j;
%             grayImageWTGSMWN(i,l,2)=grayImageWTGSM(i,j);
%
%             l=l+1;
%         end
%     end
```

```
    %x=max(bordertest(i),1):1:columnsS;
    %x=grayImageWTGSMWN(i,:,1)';%x';


    %f=fit(grayImageWTGSMWN(i,1:columnsS-border(i)-(columnsS-
l),1)',grayImageWTGSMWN(i,1:columnsS-border(i)-(columnsS-l),2)','poly2'); % only fit
the part starting from the membrane border
    %g=fit(grayImageTGSMWN(i,1:columnsS-border(i)-(columnsS-
l),1)',grayImageTGSMWN(i,1:columnsS-border(i)-(columnsS-l),2)','poly2'); % only fit
the part starting from the membrane border
    %f=fit([1:columnsS-border(i)]',grayImageWTGSM(i,border(i):columnsS-1)','poly2');
% only fit the part starting from the membrane border
    %g=fit([1:columnsS-border(i)]',grayImageTGSM(i,border(i):columnsS-1)','poly2'); %
only fit the part starting from the membrane border


    %f=fit([border(i):wall(i)]',grayImageWTGSM(i,border(i):wall(i))','poly2'); % only
fit the part starting from the membrane border
    %g=fit([border(i):wall(i)]',grayImageTGSM(i,border(i):wall(i))','cubicinterp'); %
only fit the part starting from the membrane border
    %offset=50;

%     for o=1:15
%         [f1,Sf]=polyfit((max(1,border(i)-
offset):wall(i)+offset)*pixsize,grayImageWTGSM(i,max(1,border(i)-
offset):wall(i)+offset),o);
%         [f,deltaf(i,1:wall(i)-offset-
(border(i)+offset)+1)]=polyval(f1,(border(i)+offset:wall(i)-offset)*pixsize,Sf);
%         Rf2(o,1)=o;
%         Rf2(o,2)=1-(Sf.normr/norm(grayImageWTGSM(i,max(1,border(i)-
offset):wall(i)+offset)-mean(grayImageWTGSM(i,max(1,border(i)-
offset):wall(i)+offset))))^2;
%
%         [g1,Sg]=polyfit((max(1,border(i)-
offset):wall(i)+offset)*pixsize,grayImageTGSM(i,max(1,border(i)-
offset):wall(i)+offset),o);
%         [g,deltag(i,1:wall(i)-offset-
(border(i)+offset)+1)]=polyval(g1,(border(i)+offset:wall(i)-offset)*pixsize,Sg);
%         Rg2(o,1)=o;
%         Rg2(o,2)=1-(Sg.normr/norm(grayImageTGSM(i,max(1,border(i)-
offset):wall(i)+offset)-mean(grayImageTGSM(i,max(1,border(i)-
offset):wall(i)+offset))))^2;
%
%     end
%
%     o=find(Rf2(:,2)==max(Rf2(:,2)))
%     Rf2=Rf2(o,2)
%
    %[f1,Sf]=polyfit((max(1,border(i)-
offset):wall(i)+offset)*pixsize,grayImageWTGSM(i,max(1,border(i)-
offset):wall(i)+offset),o);
    %[f,deltaf(i,1:wall(i)-offset-
(border(i)+offset)+1)]=polyval(f1,(border(i)+offset:wall(i)-offset)*pixsize,Sf);
```

```
    %[g1,Sg]=polyfit((max(1,border(i)-
offset):wall(i)+offset)*pixsize,grayImageTGSM(i,max(1,border(i)-
offset):wall(i)+offset),o);
    %[g,deltag(i,1:wall(i)-offset-
(border(i)+offset)+1)]=polyval(g1,(border(i)+offset:wall(i)-offset)*pixsize,Sg);

    sample_rate=1;

    %you can play with these two below
    lowpass_freq=0.008;
    lowpass_order=2;
    Fnorm=lowpass_freq/(sample_rate/2);

    [num,den]=butter(lowpass_order,Fnorm);
    f=filtfilt(num,den,grayImageWTGSM(i,max(1,border(i)):wall(i)));
    g=filtfilt(num,den,grayImageTGSM(i,max(1,border(i)):wall(i)));

    deltaf(i,:)=std(f-grayImageWTGSM(i,max(1,border(i)):wall(i)));
    deltag(i,:)=std(g-grayImageTGSM(i,max(1,border(i)):wall(i)));

    ft(i,border(i):wall(i))=f;
    gt(i,border(i):wall(i))=g;
    %h=g(x)-f(x);

    %figure
    %plot(f,grayImageWTGSMWN(i,1:columnsS-border(i)-
6,1)',grayImageWTGSMWN(i,1:columnsS-border(i)-6,2)')
    %plot(f,(border(i):columnsS-1)',grayImageWTGSM(i,border(i):columnsS-1)')
    %plot(g,(border(i):columnsS-1)',grayImageTGSM(i,border(i):columnsS-1)')
    %plot(border(i)+offset:wall(i)-offset,f,border(i)+offset:wall(i)-
offset,grayImageWTGSM(i,border(i)+offset:wall(i)-offset)')
    %plot(border(i)+offset:wall(i)-offset,g,border(i)+offset:wall(i)-
offset,grayImageTGSM(i,border(i)+offset:wall(i)-offset)')

    p=[473.619427566442,-1836.93963497222,2889.31097873429,-
2362.01825914816,1070.95691689800,-267.287228487073,38.1499068962742,-
3.76762110940723]; % polynomial fit of the camera response curve, giving
log(Ireal)=p(pixvalue) for ixvalue in [0,1]


    jstart=max(2,bordertest(i))+offset;
    for j=1:wall(i)-offset-border(i)-offset%jstart+1:wall(i)-offset%columnsS-
border(i)%-(columnsS-l)
        if f(j)>0 && g(j)>0
        %dx(i,j)=-45/4*(log10(g(j))-log10((f(j))));
        %dx(i,j)=-45/4*(log10(g(j))-log10((f(j)))); %in mm if camera
        %response is linear
        dx(i,j)=45/4*(polyval(p,f(j))-polyval(p,g(j))); %in mm to take into account
the non linear camera
        dxmin(i,j)=45/4*(polyval(p,f(j)-deltaf(i))-polyval(p,g(j)+deltag(i))); %in mm
to take into account the non linear camera
        dxmax(i,j)=45/4*(polyval(p,f(j)+deltaf(i))-polyval(p,g(j)+deltaf(i))); %in mm
to take into account the non linear camera
        %alphaprim(i,j)=atan(dx(i,j)/fl1); % this is the wrong alpha, the
        %akpha without taking into account the impact of the window
```

```
        myfun=@(x,deltax,e,dist)s(x,deltax,e,dist);
        deltax=dx(i,j); %inmm
        e=8; %in mm
        dist=fl1; %in mm
        alpha0=atan(dx(i,j)/fl1); %in rad
        fun=@(x)myfun(x,deltax,e,dist);
        alphaprim(i,j)=fzero(fun,double(alpha0));

%         myfunmin=@(x,deltaxmin,e,dist)s(x,deltaxmin,e,dist);
%         deltaxmin=dxmin(i,j); %inmm
%         e=8; %in mm
%         dist=fl1; %in mm
%         alpha0min=atan(dxmin(i,j)/fl1); %in rad
%         funmin=@(x)myfunmin(x,deltax,e,dist);
%         alphaprimmin(i,j)=fzero(funmin,double(alpha0min));
%
%
%         myfunmax=@(x,deltaxmax,e,dist)s(x,deltaxmax,e,dist);
%         deltax=dxmax(i,j); %inmm
%         e=8; %in mm
%         dist=fl1; %in mm
%         alpha0max=atan(dxmax(i,j)/fl1); %in rad
%         funmax=@(x)myfunmax(x,deltax,e,dist);
%         alphaprimmax(i,j)=fzero(funmax,double(alpha0max));



        alpha(i,j)=asin(nair/nwater*sin(alphaprim(i,j)));
        dn(i,j)=alpha(i,j)/L; %refraction index gradient per mm
        dT(i,j)=dn(i,j)/0.0002; %temperature gradient per mm

%         alphamin(i,j)=asin(nair/nwater*sin(alphaprimmin(i,j)));
%         dnmin(i,j)=alphamin(i,j)/L; %refraction index gradient per mm
%         dTmin(i,j)=dnmin(i,j)/0.0002; %temperature gradient per mm
%
%         alphamax(i,j)=asin(nair/nwater*sin(alphaprimmax(i,j)));
%         dnmax(i,j)=alphamax(i,j)/L; %refraction index gradient per mm
%         dTmax(i,j)=dnmax(i,j)/0.0002; %temperature gradient per mm
%

        k=polyder(p); %derivative of the polynomial p of the camera response function
        deltapf=std(polyval(p,f)-f);
        deltapg=std(polyval(p,g)-g);
        %dTdf(i,j)=45/(4*fl1*0.0002*10)*1/(1+(45/(4*fl1)*(polyval(p,f(j))-
polyval(p,g(j))))^2)*1/1.33*cos(atan(45/(4*fl1)*(polyval(p,f(j))-
polyval(p,g(j)))))/(sqrt(1-(1/1.33*sin(atan(45/(4*fl1)*(polyval(p,f(j))-
polyval(p,g(j))))))^2))*((f(j)-f(j-1))/(pixsize)*polyval(k,f(j))*deltaf+(g(j)-g(j-
1))/(pixsize)*polyval(k,f(j))*deltag);
        dTdf(i,j)=45/(4*fl1*0.0002*10)*1/(1+(45/(4*fl1)*(polyval(p,f(j))-
polyval(p,g(j))))^2)*1/1.33*cos(atan(45/(4*fl1)*(polyval(p,f(j))-
polyval(p,g(j)))))/(sqrt(1-(1/1.33*sin(atan(45/(4*fl1)*(polyval(p,f(j))-
polyval(p,g(j))))))^2))*((f(j)-f(max(1,j-
1)))/(pixsize)*polyval(k,f(j))*(1.96*deltaf(i,:)/sqrt(wall(i)-
border(i))+0.99979*10^(-4)*f(j))-(g(j)-g(max(1,j-
```

```
1)))/(pixsize)*polyval(k,f(j))*(1.96*deltag(i,:)/sqrt(wall(i)-
border(i))+0.99979*10^(-4)*g(j))); %error for dT : delta is the standard deviation,
and +/- 1.96delta/sqrt(n) is the 95%confidence interval


        end
    end


    for j=2:min(size(dT,2),wall(i)-offset-border(i)-offset)%jstart+1:wall(i)-
offset%columnsS-border(i)%-(columnsS-l)
      %Tmean(i,j-jstart+1)=-dT(i,j-1)*10^(-3)+(Tmean(i,j-jstart+1-1));%-TBulk)+TBulk;
      Tmean(i,j)=-dT(i,j-1)*pixsize+(Tmean(i,j-1));%-TBulk)+TBulk;
    end

%     for j=2:min(size(dT,2),wall(i)-offset-border(i)-offset)%jstart+1:wall(i)-
offset%columnsS-border(i)%-(columnsS-l)
%       %Tmean(i,j-jstart+1)=-dT(i,j-1)*10^(-3)+(Tmean(i,j-jstart+1-1));%-
TBulk)+TBulk;
%       Tmeanmin(i,j)=-dTmin(i,j-1)*pixsize+(Tmeanmin(i,j-1));%-TBulk)+TBulk;
%     end
%
%     for j=2:min(size(dT,2),wall(i)-offset-border(i)-offset)%jstart+1:wall(i)-
offset%columnsS-border(i)%-(columnsS-l)
%       %Tmean(i,j-jstart+1)=-dT(i,j-1)*10^(-3)+(Tmean(i,j-jstart+1-1));%-
TBulk)+TBulk;
%       Tmeanmax(i,j)=-dTmax(i,j-1)*pixsize+(Tmeanmax(i,j-1));%-TBulk)+TBulk;
%     end
%
    %%Tmean(i,:)=Tmean(i,:)+(TBulk-Tmean(i,columnsS-border(i)-jstart+1-
1))*ones(1,size(Tmean,2));
    %for j=1:wall(i)-offset-border(i)-offset%jstart+1:wall(i)-offset%columnsS-
border(i)%-(columnsS-l)
      %%Tmean(i,j-jstart+1)=Tmean(i,j-jstart+1)+(TBulk-Tmean(i,columnsS-border(i)-
jstart+1));
      %Tmean(i,j)=Tmean(i,j)+(TBulk-Tmean(i,wall(i)-offset-border(i)-offset));
    %end


    for j=1:min(size(dT,2),wall(i)-offset-border(i)-offset)-1%1:size(Tmean,2)-
1%jstart+1:columnsS-border(i)%-(columnsS-l)
      %dTreconstruct(i,j-jstart+1)=-(Tmean(i,j-jstart+1+1)-Tmean(i,j-jstart+1));
      dTreconstruct(i,j)=-(Tmean(i,j+1)-Tmean(i,j))/pixsize;
    end


    i

end

writematrix(Tmean,nameOfFile +"_Tmean.txt")
writematrix(deltaf,nameOfFile +"_deltaf.txt")
writematrix(deltag,nameOfFile +"_deltag.txt")
writematrix(ft,nameOfFile +"_f.txt")
writematrix(gt,nameOfFile +"_g.txt")
```

```
% for i=1:rowsS-rowmin%+1
%     jstart=max(2,bordertest(i));
%     for j=1:wall(i)-offset-border(i)-offset%jstart+1:wall(i)-offset%columnsS-
border(i)%-(columnsS-l)
%
%         k=polyder(p); %derivative of the polynomial p of the camera response
function
%         dTdf(i,j)=45/(4*fl1*0.0002*10)*1/(polyval(p,f(i,j))-
polyval(p,g(i,j)))^2*1/1.33*cos(atan(45/(4*fl1)*(polyval(p,f(i,j))-
polyval(p,g(i,j)))))/(sqrt(1-(1/1.33*sin(atan(45/(4*fl1)*(polyval(p,f(i,j))-
polyval(p,g(i,j))))))^2))*(polyval(k,f(i,j))-polyval(k,g(i,j)));
%
%     end
% end


%--------------------------
%test du fit de temperature
%--------------------------
    i=2000;

    %offset=10;
    %[f1,Sf]=polyfit((border(i)-
offset:wall(i)+offset)*pixsize,grayImageWTGSM(i,border(i)-offset:wall(i)+offset),o);
    %[f, deltaff]=polyval(f1,(border(i)+offset:wall(i)-offset)*pixsize,Sf);
    %[g1,Sg]=polyfit((border(i)-
offset:wall(i)+offset)*pixsize,grayImageTGSM(i,border(i)-offset:wall(i)+offset),o);
    %[g,deltagg]=polyval(g1,(border(i)+offset:wall(i)-offset)*pixsize,Sg);

    %f=lowpass(grayImageWTGSM(i,max(1,border(i)-offset):wall(i)+offset),20,1);
    %g=lowpass(grayImageTGSM(i,max(1,border(i)-offset):wall(i)+offset),500, 1000);

    sample_rate=1;

    %you can play with these two below
    lowpass_freq=0.008;
    lowpass_order=2;
    Fnorm=lowpass_freq/(sample_rate/2);

    [num,den]=butter(lowpass_order,Fnorm);
    f=filtfilt(num,den,grayImageWTGSM(i,max(1,border(i)):wall(i)));
    g=filtfilt(num,den,grayImageTGSM(i,max(1,border(i)):wall(i)));

    deltaf(i,:)=std(f-grayImageWTGSM(i,max(1,border(i)):wall(i)));
    deltag(i,:)=std(g-grayImageTGSM(i,max(1,border(i)):wall(i)));

    %ft(i,border(i)+offset:wall(i)-offset)=f;
    %gt(i,border(i)+offset:wall(i)-offset)=g;


    figure

plot((max(1,border(i)):wall(i))*pixsize,g,(1:size(grayImageTGSM,2))*pixsize,grayImage
TGSM(i,1:size(grayImageTGSM,2))')
```

```
    grid on
    title('Fit intensity profile')
    xlabel('Distance from the membrane (mm)')
    ylabel('Intensity')
    %saveas(gcf, nameOfFile + "_IP",'jpeg')

    figure

plot((border(i):wall(i))*pixsize,f,(1:size(grayImageWTGSM,2))*pixsize,grayImageWTGSM(
i,1:size(grayImageWTGSM,2))')
    grid on
    title('Fit intensity profile')
    xlabel('Distance from the membrane (mm)')
    ylabel('Intensity')
    %saveas(gcf, nameOfFile + "_IPWTG",'jpeg')

    figure
    plot((border(i):wall(i))*pixsize,g)
    grid on
    title('Fit intensity profile')
    xlabel('Distance from the membrane (mm)')
    ylabel('Intensity')
    %saveas(gcf, nameOfFile + "_IPWTG",'jpeg')

    figure

plot((border(i):wall(i))*pixsize,f,(1:size(grayImageWTGSM,2))*pixsize,grayImageWTGSM(
i,1:size(grayImageWTGSM,2))',(border(i):wall(i))*pixsize,g,(1:size(grayImageTGSM,2))*
pixsize,grayImageTGSM(i,1:size(grayImageTGSM,2))')
    grid on
    title('Fit intensity profile')
    xlabel('Distance from the membrane (mm)')
    ylabel('Intensity')
    %saveas(gcf, nameOfFile + "_IPWTG",'jpeg')

maxborder=max(border(1,:));
ups=xmaxpoints;%[253,740,1252,1740,2247];%,1936,2448];
downs=xminpoints;%[20,495,1010,1505,2016,2507,2994,35521];%,1602,2016];
writematrix(downs,nameOfFile +"_downs.txt")
delta=6;
colors=jet((2*delta+1)*(length(ups)-1));
dTbord=dT;

%---------------------------
% Uncomment to get pictures
%---------------------------

% figure
% for i=1:rowsS-rowmin+1
%     plot((1:columnsS-columnmin+1-maxborder)*pixsize,Tmean(i,1:columnsS-columnmin+1-
maxborder))
%     title('Mean Temperature')
%     xlabel('Distance from the membrane (mm)')
%     ylabel('T (C)')
%     hold on
```

```
% end
%
%
%
% figure
% for i=1:length(dx)%rowsS-rowmin%+1
%     dTbord(i,:)=dT(i,:);%[dT(i,bordertest(i):size(dT,2)),zeros(1,bordertest(1,i))];
%     %plot(1:max(columnsS-columnmin+1-maxborder,100),dTbord(i,1:max(columnsS-
columnmin+1-maxborder,100)))
%     plot((1:wall(i)-offset-border(i)-offset)*pixsize,dTbord(i,1:wall(i)-offset-
border(i)-offset))
%     title('Temperature gradient')
%     xlabel('x(mm)')
%     ylabel('dT')
%     hold on
% end
%
% writematrix(dTbord,nameOfFile + "_Tgrad.txt")
%
% %ups=[404,940,1382];%,1936,2448];
% %downs=[134,639,1131];%,1602,2111];
% %ups=[313,857,1359];%,1936,2448];
% %downs=[65,590,1124];%,1602,2111];
% %ups=[420,958,1457];%,1936,2448];
% %downs=[166,695,1233];%,1602,2111];
%
%
% % Plot at all the bumps
% %-----------------------
%
%
%
% figure
% for i=2:length(ups)-1
%
%     for j=-delta:delta
%
%         %l=2;
%         %while Tmean(ups(i)+j,l)>0 && l<size(Tmean,2)
%         %    l=l+1;
%         %end
%
%         %plot(1:max(columnsS-columnmin+1-
maxborder,100),Tmean(ups(i)+j,1:max(columnsS-columnmin+1-maxborder,100)))
%         %plot(2:l-1,dTbord(ups(i)+j,2:l-1),'Color',colors(delta +1+(2*delta+1)*(i-
1) + j,:))
%         plot((2:wall(i)-offset-border(i)-offset)*pixsize,dTbord(ups(i)+j,2:wall(i)-
offset-border(i)-offset),'Color',colors(delta +1+(2*delta+1)*(i-1) + j,:))
%         title('Temperature gradient')
%         xlabel('Distance from the membrane (mm)')
%         ylabel('dT/dx')
%         hold on
%     end
% end
% saveas(gcf, nameOfFile + "_TGB",'jpeg')
```

```
%
%
% colors=jet((2*delta+1)*(length(ups)-1));
% figure
% for i=2:length(ups)-1
%
%     for j=-delta:delta
%
%         %l=2;
%         %while Tmean(ups(i)+j,l)>0 && l<size(Tmean,2)
%         %    l=l+1;
%         %end
%
%         %plot(1:max(columnsS-columnmin+1-
maxborder,100),Tmean(ups(i)+j,1:max(columnsS-columnmin+1-maxborder,100)))
%         %plot(2:l-1,Tmean(ups(i)+j,2:l-1),'Color',colors(delta +1+(2*delta+1)*(i-1)
+ j,:))
%         plot((2:wall(i)-offset-border(i)-offset)*pixsize,Tmean(ups(i)+j,2:wall(i)-
offset-border(i)-offset),'Color',colors(delta +1+(2*delta+1)*(i-1) + j,:))
%         axis([0 4 -15 15])
%         title('Temperature profiles at the bumps of the membrane')
%         xlabel('Distance from the membrane (mm)')
%         ylabel('T (C)')
%         hold on
%     end
% end
% saveas(gcf, nameOfFile + "_B",'jpeg')
%
% colors=jet(size(Tmean,1)-1);
% figure
% for i=1:size(Tmean,1)-1
%
%         %l=2;
%         %while Tmean(i,l)>0 && l<size(Tmean,2)
%         %    l=l+1;
%         %end
%
%         %plot(1:max(columnsS-columnmin+1-
maxborder,100),Tmean(ups(i)+j,1:max(columnsS-columnmin+1-maxborder,100)))
%         %plot(2:l-1,Tmean(i,2:l-1),'Color',colors(i,:))
%         plot((2:wall(i)-offset-border(i)-offset)*pixsize,Tmean(i,2:wall(i)-offset-
border(i)-offset),'Color',colors(i,:))
%         axis([0 4 -15 15])
%         title('Temperature profiles')
%         xlabel('Distance from the membrane (mm)')
%         ylabel('T (C)')
%         hold on
%     end
%
%
%
% % Plot at all the hollows
% %-------------------------
%
% colors=jet((2*delta+1)*(length(downs)-1));
```

```
% figure
% for i=2:length(downs)-1
%
%     for j=-delta:delta
%
%         %l=2;
%         %while Tmean(downs(i)+j,l)>0 && l<size(Tmean,2)
%         %    l=l+1;
%         %end
%
%         %plot(1:max(columnsS-columnmin+1-
maxborder,100),Tmean(ups(i)+j,1:max(columnsS-columnmin+1-maxborder,100)))
%         %plot(2:l-1,Tmean(downs(i)+j,2:l-1),'Color',colors(delta +1+(2*delta+1)*(i-
1) + j,:))
%         plot((2:wall(i)-offset-border(i)-
offset)*pixsize,Tmean(downs(i)+j,2:wall(i)-offset-border(i)-
offset),'Color',colors(delta +1+(2*delta+1)*(i-1) + j,:))
%         axis([0 4 -15 15])
%         title('Temperature profiles at the hollows of the membrane')
%         xlabel('Distance from the membrane (mm)')
%         ylabel('T (C)')
%         hold on
%     end
% end
% saveas(gcf, nameOfFile + "_H",'jpeg')
%
% % Plot at each hollow
% %--------------------
%
% %figure
 colors=jet(2*delta+1);
%
% for i=3:length(downs)
%     figure
%     for j=-delta:delta
%
%         %l=2;
%         %while Tmean(downs(i)+j,l)>0 && l<size(Tmean,2)
%         %    l=l+1;
%         %end
%
%         %plot(1:max(columnsS-columnmin+1-
maxborder,100),Tmean(downs(i)+j,1:max(columnsS-columnmin+1-maxborder,100)))
%         %plot(2:l-1,Tmean(downs(i)+j,2:l-1),'Color',colors(delta +1 + j,:))
%         plot((2:wall(i)-offset-border(i)-
offset)*pixsize,Tmean(downs(i)+j,2:wall(i)-offset-border(i)-
offset),'Color',colors(delta +1 + j,:))
%         axis([0 4 -15 15])
%         title("Temperature profiles hollow " + i)
%         xlabel('Distance from the membrane (mm)')
%         ylabel('T (C)')
%         %'fontsize',16
%         hold on
%     end
%     saveas(gcf, nameOfFile + "_H" +i,'jpeg')
```

```
% end
%
%
for i=1:length(downs)
    figure
    for j=-delta:delta

        %l=2;
        %while Tmean(downs(i)+j,l)>0 && l<size(Tmean,2)
        %    l=l+1;
        %end

        %plot(1:max(columnsS-columnmin+1-
maxborder,100),Tmean(downs(i)+j,1:max(columnsS-columnmin+1-maxborder,100)))
        %plot(2:l-1,Tmean(downs(i)+j,2:l-1),'Color',colors(delta +1 + j,:))
        plot((2:wall(i)-offset-border(i)-offset)*pixsize,dTbord(downs(i)+j,2:wall(i)-
offset-border(i)-offset),'Color',colors(delta +1 + j,:))
        grid on
        axis([0 4 -20 30])
        title("Temperature gradient profiles hollow " + i)
        xlabel('Distance from the first observable ray from the membrane (mm)')
        ylabel('dT/dx (C/mm)')
        %'fontsize',16
        hold on
    end
    saveas(gcf, nameOfFile + "_TG_H" +i,'jpeg')
end



%----------------------------
% end uncomment to get plots
%----------------------------


%-------------------------------------
% TG and Boundary layer determination
%-------------------------------------
deltaTmean=zeros(length(downs),size(dTbord,2));
deltadTbordmean=zeros(length(downs),size(dTbord,2));
deltafmean=zeros(length(downs),size(dTbord,2));
error=zeros(length(downs),size(dTbord,2));

for i=1:length(downs)
    l=1;
    k=1;
    kmin=1;
    kmax=1;

    for j=1:size(dTbord,2)
        dTbordmeanH(i,j)=mean(dTbord(downs(i)-delta:downs(i)+delta,j));
        TmeanMean(i,j)=mean(Tmean(downs(i)-delta:downs(i)+delta,j));
        %deltaTmean(i,j)=deltaTmean(i,j)+sum((Tmeanmax(downs(i)-
delta:downs(i)+delta,j)-Tmeanmin(downs(i)-
```

```
delta:downs(i)+delta,j)).*(Tmeanmax(downs(i)-delta:downs(i)+delta,j)-
Tmeanmin(downs(i)-delta:downs(i)+delta,j)));
        %deltadTbordmean(i,j)=deltadTbordmean(i,j)+sum((dTmax(downs(i)-
delta:downs(i)+delta,j)-dTmin(downs(i)-delta:downs(i)+delta,j)).*(dTmax(downs(i)-
delta:downs(i)+delta,j)-dTmin(downs(i)-delta:downs(i)+delta,j)));
        %deltamean(i,j)=sum((deltaf(downs(i)-delta:downs(i)+delta,j)-deltag(downs(i)-
delta:downs(i)+delta,j)).*((deltaf(downs(i)-delta:downs(i)+delta,j))-deltag(downs(i)-
delta:downs(i)+delta,j)));
        deltamean(i,j)=sum((deltaf(downs(i)-
delta:downs(i)+delta)).*((deltaf(downs(i)-delta:downs(i)+delta))));
        dTdfmean(i,j)=mean(dTdf(downs(i)-delta:downs(i)+delta,j));
    end



    %deltaTmean(i,:)=sqrt(deltaTmean(i,:));
    deltamean(i,:)=sqrt(deltamean(i,:)/(2*delta));
    %deltadTbordmean(i,:)=sqrt(deltadTbordmean(i,:));

    %TmeanMeanmin=TmeanMean-deltaTmean(i,:);
    %TmeanMeanmax=TmeanMean+deltaTmean(i,:);

    %dTbordmeanmin=dTbordmeanH-deltadTbordmean(i,:);
    %dTbordmeanmax=dTbordmeanH+deltadTbordmean(i,:);

    %maxTmin=max(TmeanMeanmin(i,:));
    %maxTmax=max(TmeanMeanmax(i,:));


    %while TmeanMeanmin(i,kmin)<99*maxTmin/100 && kmin<size(dTbordmeanH,2)
    %    kmin=kmin+1;
    %end

    % while TmeanMeanmax(i,kmax)<99*maxTmax/100 && kmax<size(dTbordmeanH,2)
    %    kmax=kmax+1;
    %end

    error(i,:)=dTdfmean(i,:); % mean error for dT
    errorT(i,:)=dTdfmean(i,:)*pixsize; % mean error for T

    TmeanMeanmax(i,1)=0;
    TmeanMeanmin(i,1)=0;

    for j=2:min(size(dT,2),wall(i)-offset-border(i)-offset)%jstart+1:wall(i)-
offset%columnsS-border(i)%-(columnsS-l)
      %Tmean(i,j-jstart+1)=-dT(i,j-1)*10^(-3)+(Tmean(i,j-jstart+1-1));%-TBulk)+TBulk;
      TmeanMeanmax(i,j)=-(dTbordmeanH(i,j-1)+error(i,j-1))*pixsize+(TmeanMeanmax(i,j-
1));%-TBulk)+TBulk;
    end
    for j=2:min(size(dT,2),wall(i)-offset-border(i)-offset)%jstart+1:wall(i)-
offset%columnsS-border(i)%-(columnsS-l)
      %Tmean(i,j-jstart+1)=-dT(i,j-1)*10^(-3)+(Tmean(i,j-jstart+1-1));%-TBulk)+TBulk;
      TmeanMeanmin(i,j)=-(dTbordmeanH(i,j-1)-error(i,j-1))*pixsize+(TmeanMeanmin(i,j-
1));%-TBulk)+TBulk;
    end
```

```
    maxT=max(TmeanMean(i,:));
    while TmeanMean(i,k)<99*maxT/100 && k<size(dTbordmeanH,2)
        k=k+1;
    end
    maxTmin=max(TmeanMeanmin(i,:));
    while TmeanMeanmin(i,kmin)<99*maxTmin/100 && kmin<size(dTbordmeanH,2)
        kmin=kmin+1;
    end
    maxTmax=max(TmeanMeanmax(i,:));
    while TmeanMeanmax(i,kmax)<99*maxTmax/100 && kmax<size(dTbordmeanH,2)
        kmax=kmax+1;
    end

    BoundLayH(i,1)=(k-1)*pixsize;
    BoundLayH(i,2)=(kmin-1)*pixsize;
    BoundLayH(i,3)=(kmax-1)*pixsize;

    [TGradMemH(i,1),z(i)]=min(dTbordmeanH(i,1:250));
    TGradMemH(i,2)=dTbordmeanH(i,z(i))+error(i,z(i));
    TGradMemH(i,3)=dTbordmeanH(i,z(i))-error(i,z(i));
    %BoundLayHmin(i)=(kmin-1)*pixsize;
    %TGradMemHmin(i)=min(dTbordmeanmin(i,1:150));%min(dTbordmeanH(i,1:150)-
deltadTbordmean(i,1:150));

    %BoundLayHmax(i)=(kmax-1)*pixsize;
    %TGradMemHmax(i)=min(dTbordmeanmax(i,1:150));


    testTP1=polyfit(1:size(TmeanMean,2),TmeanMean(i,1:size(TmeanMean,2)),7);
    testTP=polyval(testTP1,1:size(TmeanMean,2));


    figure

plot(1:size(TmeanMean,2),testTP,1:size(TmeanMean,2),TmeanMean(i,1:size(TmeanMean,2))'
,1:size(TmeanMeanmax,2),TmeanMeanmax(i,1:size(TmeanMeanmax,2))','m--
',1:size(TmeanMeanmin,2),TmeanMeanmin(i,1:size(TmeanMeanmin,2))','m--');
    grid on
    title('Fit temperature profile ')
    xlabel('Distance from the first observable ray from the membrane (mm)')
    ylabel('Temperature (C)')
    saveas(gcf, nameOfFile + "_fitTP" + i,'jpeg')

writematrix([(1:min([size(TmeanMean,2),size(TmeanMeanmin,2),size(TmeanMeanmax,2)]))*p
ixsize;testTP(1:min([size(TmeanMean,2),size(TmeanMeanmin,2),size(TmeanMeanmax,2)]));T
meanMeanmin(i,1:min([size(TmeanMean,2),size(TmeanMeanmin,2),size(TmeanMeanmax,2)]));T
meanMeanmax(i,1:min([size(TmeanMean,2),size(TmeanMeanmin,2),size(TmeanMeanmax,2)]))],
nameOfFile + "_FitTP" + i + ".txt");


end
```

```
csvwrite(nameOfFile + "_BLH.csv",BoundLayH)
csvwrite(nameOfFile + "_TGMH.csv",TGradMemH)
%writematrix(BoundLayHmin,nameOfFile + "_BLHmin.txt")
%writematrix(TGradMemHmin,nameOfFile + "_TGMHmin.txt")
%writematrix(BoundLayHmax,nameOfFile + "_BLHmax.txt")
%writematrix(TGradMemHmax,nameOfFile + "_TGMHmax.txt")

% to get the polynome for the fit of the temperature profile, and guess the
% intensity with the Ray tracing code
i=5;
%testTP1=polyfit(1:size(TmeanMean,2),TmeanMean(i,1:size(TmeanMean,2)),7);
%testTP=polyval(testTP1,1:size(TmeanMean,2));


sample_rate=1;

    %you can play with these two below
    lowpass_freq=0.008;
    lowpass_order=2;
    Fnorm=lowpass_freq/(sample_rate/2);

    [num,den]=butter(lowpass_order,Fnorm);
    testTP=filtfilt(num,den,TmeanMean(i,1:size(TmeanMean,2)));

    deltaf(i,:)=std(testTP-TmeanMean(i,1:size(TmeanMean,2)));



figure
plot(1:size(TmeanMean,2),testTP,1:size(TmeanMean,2),TmeanMean(i,1:size(TmeanMean,2))'
,1:size(TmeanMean,2),TmeanMean(i,1:size(TmeanMean,2))'+error(i,1:size(TmeanMean,2))'/
sqrt(12),'m--',1:size(TmeanMean,2),TmeanMean(i,1:size(TmeanMean,2))'-
error(i,1:size(TmeanMean,2))'/sqrt(12),'m--');
grid on
title('Fit temperature profile')
xlabel('Distance from the first observable ray from the membrane (mm)')
ylabel('Temperature (mm)')
saveas(gcf, nameOfFile + "_fitTP",'jpeg')

%-----------------------------------------------------------------
%test du fit de temperature à l'endroit où le polynome est pris
%-----------------------------------------------------------------
    i=downs(i);

    %offset=50;
    %f1=polyfit(border(i)-offset:wall(i)+offset,grayImageWTGSM(i,border(i)-
offset:wall(i)+offset),o);
    %f=polyval(f1,(border(i)+offset:wall(i)-offset));
    %g1=polyfit(border(i)-offset:wall(i)+offset,grayImageTGSM(i,border(i)-
offset:wall(i)+offset),o);
    %g=polyval(g1,(border(i)+offset:wall(i)-offset));

    %f=lowpass(grayImageWTGSM(i,max(1,border(i)-offset):wall(i)+offset));
    %g=lowpass(grayImageTGSM(i,max(1,border(i)-offset):wall(i)+offset));
```

```
    sample_rate=1;

    %you can play with these two below
    lowpass_freq=0.008;
    lowpass_order=2;
    Fnorm=lowpass_freq/(sample_rate/2);

    [num,den]=butter(lowpass_order,Fnorm);
    f=filtfilt(num,den,grayImageWTGSM(i,max(1,border(i)):wall(i)));
    g=filtfilt(num,den,grayImageTGSM(i,max(1,border(i)):wall(i)));

    %deltaf(i,:)=std(f-grayImageWTGSM(i,max(1,border(i)):wall(i)));
    %deltag(i,:)=std(g-grayImageTGSM(i,max(1,border(i)):wall(i)));


    figure

plot((max(1,border(i)):wall(i))*pixsize,g,(1:size(grayImageTGSM,2))*pixsize,grayImage
TGSM(i,1:size(grayImageTGSM,2))')
    grid on
    title('Fit intensity profile')
    xlabel('Distance from the first observable ray from the membrane (mm)')
    ylabel('Normalized intensity')
    saveas(gcf, nameOfFile + "_IP",'jpeg')

%     figure
%
plot((max(1,border(i)):wall(i))*pixsize,f,1:size(grayImageWTGSM,2)*pixsize,grayImageW
TGSM(i,1:size(grayImageWTGSM,2))')
%     grid on
%     title('Fit intensity profile')
%     xlabel('Distance from the first observable ray from the membrane (mm)')
%     ylabel('Normalized intensity')
%     saveas(gcf, nameOfFile + "_IPWTG",'jpeg')
%
    %testf=f(border(i)+offset:1462);
    %testg=g(border(i)+offset:1462);
    %testf=testf';
    %testg=testg';




%Plot Before each hollow
%-----------------------
colors=jet(delta+1);

for i=1:length(downs)
    figure
    for j=-delta:0

        %l=2;
        %while Tmean(downs(i)+j,l)>0 && l<size(Tmean,2)
        %    l=l+1;
        %end
```

```
        %plot(1:max(columnsS-columnmin+1-
maxborder,100),Tmean(downs(i)+j,1:max(columnsS-columnmin+1-maxborder,100)))
        %plot(2:l-1,Tmean(downs(i)+j,2:l-1),'Color',colors(delta+1+j,:))
        plot((2:wall(i)-offset-border(i)-offset)*pixsize,Tmean(downs(i)+j,2:wall(i)-
offset-border(i)-offset),'Color',colors(delta+1+j,:))
        grid on
        axis([0 4 -15 15])
        title("Temperature profiles before hollow " + i)
        xlabel('Distance from the first observable ray from the membrane (mm)')
        ylabel('T (C)')
        %'fontsize',16
        hold on
    end
    saveas(gcf, nameOfFile + "_BH" +i,'jpeg')
end



%Plot After each hollow
%----------------------
colors=jet(delta+1);

for i=1:length(downs)
    figure
    for j=0:delta

        %l=2;
        %while Tmean(downs(i)+j,l)>0 && l<size(Tmean,2)
        %    l=l+1;
        %end

        %plot(1:max(columnsS-columnmin+1-
maxborder,100),Tmean(downs(i)+j,1:max(columnsS-columnmin+1-maxborder,100)))
        %plot(2:l-1,Tmean(downs(i)+j,2:l-1),'Color',colors(delta+1-j,:))
        plot((2:wall(i)-offset-border(i)-offset)*pixsize,Tmean(downs(i)+j,2:wall(i)-
offset-border(i)-offset),'Color',colors(delta+1-j,:))
        grid on
        axis([0 4 -15 15])
        title("Temperature profiles after hollow " + i)
        xlabel('Distance from the first observable ray from the membrane (mm)')
        ylabel('Temperature (C)')
        %'fontsize',16
        hold on
    end
    saveas(gcf, nameOfFile + "_AH" +i,'jpeg')
end


% Averaging of the T profile in the hollows of the membrane
%-----------------------------------------------------------

xblock=downs(1)-delta:downs(1)+delta;
for i=2:length(downs)
    xblock=[xblock,downs(i)-delta:downs(i)+delta];
```

```
end


Tmeanmean=mean(Tmean(xblock,1:max(columnsS-columnmin+1-maxborder,100)));
figure
plot((1:length(Tmeanmean))*pixsize,Tmeanmean(1,:))
grid on
title('Mean Temperature profile at the hollows of the membrane')
xlabel('Distance from the first observable ray from the membrane (mm)')
ylabel('Temperature (C)')


Tmeanecart=zeros(length(xblock),max(columnsS-columnmin+1-maxborder,100));
Tmeanecartsum=zeros(1,max(columnsS-columnmin+1-maxborder,100));
Tmeanecartsumroot=zeros(1,max(columnsS-columnmin+1-maxborder,100));

for i=2:length(xblock)
    for j=1:1:columnsS-columnmin+1-maxborder
        Tmeanecart(i,j)=(Tmean(xblock(i),j)-Tmeanmean(1,j))^2;
        d(i,j)=Tmean(xblock(length(xblock)-i+1),j)-Tmean(xblock(i),j);
    end
end
for j=1:1:columnsS-columnmin+1-maxborder
    for i=1:length(xblock)
        Tmeanecartsum(1,j)=Tmeanecartsum(1,j)+Tmeanecart(i,j);
    end
end
```

Ray Tracing Code

```
%
% READ ME:
% this little program is based on the rayTracingSetUpMirrorsTestModuleLength
% and it aims at creating the ray paths through an optical device,
% geometrically defined by its radiuses, and taking into account the
% reflection on the membrane.


% The object encountered in this file are:


%-------------------------------------------------------------------------
% LENS : table geometrically describing all the Lenses or Mirrors on the
% optical path
% 1 - x             : position of the Lens on the optical axis
% 2 - R1            : radius of the left face (<0 if center on the left of x)
% 3 - R2            : radius of the right face (<0 if center on the left of x)
% 4 - e             : thickness of the lens
% 5 - Diam          : Diameter of the lens
% 6 - index         : refractive index of the lens material (-1 for mirrors)
% 7 - intensity     : parameter changing ray intensity (gradient filters)
% 8 - thetaLens     : inclination of the Lens compared to the horizontal
% 9 - zpositionLens : position of the Lens perpendicular to the horizontal
%-------------------------------------------------------------------------
%-------------------------------------------------------------------------
% RAY(=CUMULABS)  : collects the beam path across the Set-Up
% 1  - z          : position vertical in the final image,
% 2  - x          : position horizontal in the final image,
% 3  - theta      : angle between the beam and the optical axis
% 4  - iin        : angle between the incident beam and the surface normal
% 5  - iout       : angle between the exiting beam and the surface normal
% 6  - index      : refractive index of the medium at this position
% 7  - Boolean    : 1 if the beam can continue/0 if it is blocked by a wall
% 8  - beta       : angle of the optical axis with the horizontal of a reference
% 9  - path       : optical path traveled by the beam
% 10 - intensity  : intensity of the light beam
%-------------------------------------------------------------------------


clear all
close all
format long
digits(100)


nair=1.000272;

BL=1; %4; %[0.25,0.5,1,1.5,2,5]; % height of the BL /!\(mm)/!\
HChannel=0.025; %height of the hot water channel (m)
```

```
DeltaT=0;%5;%[5,10,15,20]; %[1,2,3,4,5,10,15]; %[5,10,15,20];
BulkT=313;%[313,333,343,353]; % (K) =<353°K
DBeams=0.0001; %distance between 2 light beams (in m)
LModule=0.01;%[0.005, 0.008, 0.01, 0.015, 0.018, 0.02]; %thickness of the module (m)

%----------------------------------
%To use an external T-gradient file
%----------------------------------
%/!\ also change zentry in the initialisation loop of the ray

% chemin="G:\MATLAB\TGradient2221_6.xlsx";
% T=xlsread(chemin); %Temperature gradient (K)
% T(:,2)=T(:,2)+BulkT;
% T(:,1)=T(:,1)/1000;


%----------------------------------------
%To use a function determined T-gradient
%----------------------------------------
tau=BL/(1000); %denominator in the exponent of the exponential exp(-t/tau): in
general, 5tau=BL height.
 for m=1:length(BL)
     for l=1:length(DeltaT)
        %1 T-gradient on the membrane side based on exponential
        %T(:,:,m,l)=[0:DBeams:HChannel;(BulkT-DeltaT(l))+DeltaT(l)*(1-exp(-(HChannel-
[0:DBeams:HChannel])/(BL(m)/(1000*10))))].';

        %1 T-gradient on the membrane side based on 4-degree polynom
        %T(:,:,m,l)=[0:DBeams:HChannel;(BulkT-DeltaT(l))+DeltaT(l)*(1-(1-2*((1-
[0:DBeams:HChannel]/HChannel))+2*((1-[0:DBeams:HChannel]/HChannel)).^3-((1-
[0:DBeams:HChannel]/HChannel)).^4))].';

        %1 T-gradient on the membrane side constant
        T(:,:,m,l)=[0:DBeams:HChannel;-
(DeltaT(l)/HChannel)*([0:DBeams:HChannel])+BulkT].';


        %1 T-gradient on the membrane side based on 4-degree polynom bis
        %(reversing)
        %T(:,:,m,l)=[0:DBeams:HChannel;BulkT+DeltaT(l)*(1-
2*(([0:DBeams:HChannel]/HChannel))+2*(([0:DBeams:HChannel]/HChannel)).^3-
(([0:DBeams:HChannel]/HChannel)).^4)].';


        %2 T-gradient at each side based on 4-degree polynom
        %T(:,:,m,l)=[0:DBeams:HChannel;(BulkT-DeltaT(l))+DeltaT(l)*(1-(1-2*((1-
[0:DBeams:HChannel]/HChannel))+2*((1-[0:DBeams:HChannel]/HChannel)).^3-((1-
[0:DBeams:HChannel]/HChannel)).^4)).*(1-(1-
2*(([0:DBeams:HChannel]/HChannel))+2*(([0:DBeams:HChannel]/HChannel)).^3-
(([0:DBeams:HChannel]/HChannel)).^4))/((1-(1-2*((1/2))+2*((1/2)).^3-((1/2)).^4)).*(1-
(1-2*((1/2))+2*((1/2)).^3-((1/2)).^4)))].';


        %interpolated Tprofile from 20211221_2201 at down 4
```

```
        %[-1.3821393e-21,1.0303302e-17,-2.6859135e-14,3.1607925e-11,-1.7004082e-
08,2.7158701e-06,0.00093925628,0.044641633]
        %T(:,:,m,l)=[0:DBeams:HChannel;(BulkT-
DeltaT(l))+DeltaT(l)*(0.044641633+0.00093925628*(([0:DBeams:HChannel])/(1.95*10^(-
6)))+2.7158701*10^(-06)*(([0:DBeams:HChannel])/(1.95*10^(-6))).^2-1.7004082*10^(-
08)*(([0:DBeams:HChannel])/(1.95*10^(-6))).^3+3.1607925*10^(-
11)*(([0:DBeams:HChannel])/(1.95*10^(-6))).^4-2.6859135*10^(-
14)*(([0:DBeams:HChannel])/(1.95*10^(-6))).^5+1.0303302*10^(-
17)*(([0:DBeams:HChannel])/(1.95*10^(-6))).^6-1.3821393*10^(-
21)*(([0:DBeams:HChannel])/(1.95*10^(-6))).^7)].';



        %2 T-gradient at each side based on 4-degree polynom bis
        %(reversing)
        %T(:,:,m,l)=[0:DBeams:HChannel;BulkT+DeltaT(l)*(1-
2*(([0:DBeams:HChannel]/HChannel))+2*(([0:DBeams:HChannel]/HChannel)).^3-
(([0:DBeams:HChannel]/HChannel)).^4).*(1-2*((1-[0:DBeams:HChannel]/HChannel))+2*((1-
[0:DBeams:HChannel]/HChannel)).^3-((1-[0:DBeams:HChannel]/HChannel)).^4)/((1-
2*((1/2))+2*((1/2)).^3-((1/2)).^4)*(1-2*((1/2))+2*((1/2)).^3-((1/2)).^4))].';


        %2 T-gradients at each side (membrane and wall) based on
        %exponential
        %T(:,:,m,l)=[0:DBeams:HChannel;(BulkT-DeltaT(l))+DeltaT(l)/((1-exp(-
(HChannel-HChannel/2)/(tau(m))))*(1-exp(-(HChannel/2)/(tau(m)))))*(1-exp(-(HChannel-
[0:DBeams:HChannel])/(tau(m)))).*(1-exp(-([0:DBeams:HChannel])/(tau(m))))].';
        %T(:,:,m,l)=[0:DBeams:HChannel;(BulkT-DeltaT(l))+DeltaT(l)/((1-exp(-
(HChannel-HChannel/2)/(tau(m))))*(1-exp(-(HChannel/2)/(2*tau(m)))))*(1-exp(-
(HChannel-[0:DBeams:HChannel])/(tau(m)))).*(1-exp(-
([0:DBeams:HChannel])/(2*tau(m))))].';

        %2 T-gradients at each side (membrane and wall) with steeper
        %T-gradient at wall
        %T(:,:,m,l)=[0:DBeams:HChannel;(BulkT-DeltaT(l))+DeltaT(l)/(-
1000*(BL(m)/1000)+log(0+0.0000000000001)-log(BL(m)/1000))*(-1000*(BL(m)/1000-
[0:DBeams:HChannel])+log([0:DBeams:HChannel]+0.0000000000001)-log(BL(m)/1000))].';

        %2 T-gradients at each side (membrane and wall) with steeper
        %T-gradient at wall
        %T(:,:,m,l)=[0:DBeams:HChannel;(BulkT-
DeltaT(l))+DeltaT(l)/(log(0+0.0000000000001)-
log(BL(m)/1000))*(log([0:DBeams:HChannel]+0.0000000000001)-log(BL(m)/1000))].';

        %2 T-gradients at each side (membrane and wall) with steeper
        %T-gradient at wall
        %T(:,:,m,l)=[0:DBeams:HChannel;(BulkT-DeltaT(l))+DeltaT(l)/DeltaT(l)*(-
2*DeltaT(l)/((HChannel)^2)*[0:DBeams:HChannel].*(HChannel-
[0:DBeams:HChannel]/2)+DeltaT(l))].';

        %No gradients
        %T(:,:,m,l)=[0:DBeams:HChannel;BulkT*ones(size([0:DBeams:HChannel]))].';

    end
 end
```

```
lambda=0.532; %[0.532, 0.635]; %[0.400,0.500,0.600,0.700,0.800]; %wavelength (microm)
minResol=5*10^(-4); %[2.9*10^(-6),2*10^(-4),5*10^(-4)]; %minimum spatial resolution
of the measuring method, hence closest beam to the membrane observable (m)
PC=0.008; %thickness of the PC sheet (m)
DScreen=0.1; %distance between the module and the screen (m)
stepz=HChannel/1387;
zEntry=T(:,1);%[0:stepz:HChannel]; %[0*BL*10^(-3);0.1*BL*10^(-3);0.5*BL*10^(-
3);0.9*BL*10^(-3)]; %minResol; % height above the membrane at which the rayo is
considered to enter (m)
Ltot=length(T)+(HChannel-BL*10^(-3)-0.0005)/(0.0005)+1;
I=100; %arbitrary value of light intensity inciding on the module
BeamSize=HChannel/length(zEntry); %size of a light beam in m
BeamIntensity=I/BeamSize; %intensity of a light beam in I.m-1



rhoTab=zeros(length(Ltot),1,length(lambda),length(BL),length(DeltaT),length(BulkT));
%densities
%cumulabsTab=zeros(Ltot,5,length(lambda),length(BL),length(DeltaT),length(BulkT));
%cumulative abscisses of the bending beam
LmaxTabs=zeros(length(lambda),length(BL),length(DeltaT),length(BulkT)); %maximum
thickness of the module before the beam hits the membrane
Lmaxmin=100; %minimum maximum thickness of the module before the beam hits the
membrane amongst all the conditions
nTab=zeros(length(Ltot),2,length(lambda),length(BL),length(DeltaT),length(BulkT));
%refraction indexes
%moduleabsTab=zeros(Ltot+1,5,length(lambda),length(BL),length(DeltaT),length(BulkT));
zScreenTab=zeros(length(lambda),length(BL),length(DeltaT),length(BulkT)); %z position
of the beam when it hits the screen
step=BeamSize/4;
Screen=[[0:step:2*HChannel].',zeros(length([0:step:2*HChannel].'),1)]; %store all the
values of the position of the beams reaching the screen by density
Module=zeros(5,2,length(LModule));

xpositionModule=0.1;
for i=1:length(LModule)
    Module(:,:,i)=[0,0;0,HChannel;LModule(i),HChannel;LModule(i),0;0,0];
    Module(:,:,i)=Module(:,:,i)+[(xpositionModule-
LModule(i)/2)*ones(5,1),zeros(5,1)];
end

fl=0.2; %0.35487;
dmodulemirror=0.5;
dknifelens=1; %m distance between the knife and the lens (normally it should be the
focal length of the lens)
xpositionLens=xpositionModule+dmodulemirror; %position of the lens on the optical
axis in m
zpositionLens=+HChannel/2; %-Diam/4;
thetaLens=0.26/2; % (rad) actual angle in the experiment pi()/5;%pi()/72; %angle
between the optical axis of the Lens (normal to the surface) and the optical path
%R1=0.20394; %radius of the left side of the lens in m, positive if the center of the
circle is on the right of the surface considered
```

```
%R2=-0.15641; %radius of the right side of the lens in m
%R4=-0.5696; %thickness of the lens in m
%e1=0.01194; %thickness of the lens in m
%e2=0.00886; %thickness of the lens in m
%Diam=0.0635; %diameter of the lens
%index1=1.5168; %refractive index of the Lens material
%index2=1.64769;
xpositionScreen=2*dknifelens;

%-----------------------------------------------------------------------
%Lenses: [R1, R2, e, Diam, refractive index, intensity factor]
%-----------------------------------------------------------------------
CM5081000P01=[-2.00004,10000,0,0.0508,-1,1]; %achromatic mirror from Thorlabs
FlatMirror=[1000,1000,0,0.0508,-1,1]; %flat mirror from Thorlabs
CM508750P01=[-1.500,10000,0,0.0508,-1,1]; %achromatic mirror from Thorlabs
PetitMiroir=[-0.2,10000,0,0.05,-1]; %little achromatic mirror
AC254500A1=[0.3373,-0.1868,0.004,0.025,1.51680,1]; %convergent lens fl=500 from
Thorlabs (first lens) material:nbk7
AC254500A2=[-0.1868,-0.5574,0.002,0.025,1.64769,1]; %convergent lens fl=500 from
Thorlabs (second lens) material:sf2
CM750500P01=[-1.0000,10000,0,0.0750,-1,1]; %achromatic mirror from Thorlabs
ACT5081000A1=[0.7579,-0.3647,0.006,0.0508,1.51958,1]; %convergent lens fl=1000 from
Thorlabs (first lens) material:nbk7
ACT5081000A2=[-0.3647,-0.9542,0.006,0.0508,1.6540,1]; %convergent lens fl=1000 from
Thorlabs (second lens) material:sf2
Filter=[1000, 1000, 0, 0.05, nair,4/0.045]; %gradient filter
%-----------------------------------------------------------------------

%ZPF=zpositionLens+(((PetitMiroir(1))/2)/cos(thetaLens))*sin(2*thetaLens); %z
coordinate of the focal point of the PetitMiroir when it receives beams angled at
thetaLens
%XPF=xpositionLens+(((PetitMiroir(1))/2)/cos(thetaLens))*cos(2*thetaLens); %x
coordinate of the focal point of the PetitMiroir when it receives beams angled at
thetaLens
ZPF2=zpositionLens+(((CM750500P01(1))/2)/cos(thetaLens)-0.1)*sin(2*thetaLens);
%zpositionLens+(((CM750500P01(1))/2))*sin(2*thetaLens);%z coordinate of the focal
point of the CM750500P01 when it receives beams angled at thetaLens
XPF2=xpositionLens+(((CM750500P01(1))/2)/cos(thetaLens)-
0.1)*cos(2*thetaLens);%xpositionLens+(((CM750500P01(1))/2))*cos(2*thetaLens);%x
coordinate of the focal point of the CM750500P01 when it receives beams angled at
thetaLens
ZPF3=zpositionLens+(((CM750500P01(1))/2)/cos(thetaLens))*sin(2*thetaLens);
%zpositionLens+(((CM750500P01(1))/2))*sin(2*thetaLens);%z coordinate of the focal
point of the CM750500P01 when it receives beams angled at thetaLens
XPF3=xpositionLens+(((CM750500P01(1))/2)/cos(thetaLens))*cos(2*thetaLens);%xpositionL
ens+(((CM750500P01(1))/2))*cos(2*thetaLens);%x coordinate of the focal point of the
CM750500P01 when it receives beams angled at thetaLens
ZPF1=zpositionLens+(((CM750500P01(1))/2)+0.095)*sin(2*thetaLens);%z coordinate of the
focal point of the CM750500P01 when it receives beams angled at thetaLens
XPF1=xpositionLens+(((CM750500P01(1))/2)+0.095)*cos(2*thetaLens);%x coordinate of the
focal point of the CM750500P01 when it receives beams angled at thetaLens
XPF=XPF3;
ZPF=ZPF3;
%x01=xpositionLens+(PetitMiroir(1)-PetitMiroir(3)/2)*cos(thetaLens); %center of the
PetitMiroir
```

```
%z0=zpositionLens+(PetitMiroir(1)-PetitMiroir(3)/2)*sin(thetaLens); %ordinate of the
centers of the PetitMiroir
%deltab=acos(sum([XPF-(x01+sqrt(PetitMiroir(1)^2-zpositionLens^2)),ZPF].*[XPF-
xpositionLens,ZPF-zpositionLens])/(sqrt((XPF-(x01+sqrt(PetitMiroir(1)^2-
zpositionLens^2)))^2+ZPF^2)*sqrt((XPF-xpositionLens)^2+(ZPF-zpositionLens)^2)))/2;
x01=xpositionLens+(CM750500P01(1)-CM750500P01(3)/2)*cos(thetaLens); %center of the
CM750500P01
z0=zpositionLens+(CM750500P01(1)-CM750500P01(3)/2)*sin(thetaLens); %ordinate of the
centers of the CM750500P01
deltab=acos(sum([XPF-(x01+sqrt(CM750500P01(1)^2-zpositionLens^2)),ZPF].*[XPF-
xpositionLens,ZPF-zpositionLens])/(sqrt((XPF-(x01+sqrt(CM750500P01(1)^2-
zpositionLens^2)))^2+ZPF^2)*sqrt((XPF-xpositionLens)^2+(ZPF-zpositionLens)^2)))/2;
delta=0;%-thetaLens/4;

%-------------------------------------------------------
%Optical Set-up: [xLens1, Lens1, thetaLens1, zLens1;...]
%-------------------------------------------------------


%Lens=[xpositionLens,PetitMiroir,thetaLens,zpositionLens;XPF-
(0.5+0.004/2)*cos(2*thetaLens-delta),AC254500A1,thetaLens-delta,ZPF-
(0.5+0.004/2)*sin(2*thetaLens-delta);XPF-(0.5+0.004+0.002/2)*cos(2*thetaLens-
delta),AC254500A2,0,ZPF-(0.5+0.004+0.002/2)*sin(2*thetaLens-
delta)];%xpositionLens+e1/2+e2/2,R2,R4,e2,Diam,index2];

%--------------------
% Real optical set-up
%--------------------
%Lens=[xpositionLens,CM750500P01,thetaLens-delta,zpositionLens;XPF,Filter,thetaLens-
delta,ZPF;XPF-(dknifelens+ACT5081000A1(3)/2)*cos(2*thetaLens-
delta),ACT5081000A1,0,ZPF-(dknifelens+ACT5081000A1(3)/2)*sin(2*thetaLens-delta);XPF-
(dknifelens+ACT5081000A1(3)+ACT5081000A2(3)/2)*cos(2*thetaLens-
delta),ACT5081000A2,0,ZPF-
(dknifelens+ACT5081000A1(3)+ACT5081000A2(3)/2)*sin(2*thetaLens-
delta)];%xpositionLens+e1/2+e2/2,R2,R4,e2,Diam,index2];

%trial with only the first lens
Lens=[xpositionLens,ACT5081000A1,0,HChannel/2];%xpositionLens+ACT5081000A1(3)/2+ACT50
81000A2(3)/2,ACT5081000A2,0,HChannel/2];%xpositionLens+e1/2+e2/2,R2,R4,e2,Diam,index2
];


%Lens=[xpositionLens,CM750500P01,thetaLens-delta,zpositionLens;XPF-
(dknifelens+ACT5081000A1(3)/2)*cos(2*thetaLens-delta),ACT5081000A1,thetaLens-
delta,ZPF-(dknifelens+ACT5081000A1(3)/2)*sin(2*thetaLens-delta);XPF-
(dknifelens+ACT5081000A1(3)+ACT5081000A2(3)/2)*cos(2*thetaLens-
delta),ACT5081000A2,0,ZPF-
(dknifelens+ACT5081000A1(3)+ACT5081000A2(3)/2)*sin(2*thetaLens-
delta)];%xpositionLens+e1/2+e2/2,R2,R4,e2,Diam,index2];

%xecran=XPF-(1.05+0.004+0.002+xpositionScreen)*cos(2*thetaLens-delta);
%zecran=ZPF-(0.05+0.004+0.002+xpositionScreen)*sin(2*thetaLens-delta);
xecran=XPF-
(dknifelens+(ACT5081000A1(3)+ACT5081000A2(3))+xpositionScreen)*cos(2*thetaLens-
delta);
```

```
zecran=ZPF-
(dknifelens+(ACT5081000A1(3)+ACT5081000A2(3))+xpositionScreen)*sin(2*thetaLens-
delta);


%Lens=[xpositionLens,CM5081000P01,thetaLens,zpositionLens;xpositionLens+2*((CM5081000
P01(1)/2)/cos(thetaLens)+0.5)*cos(2*thetaLens),AC254500A1,thetaLens,zpositionLens+2*(
(CM5081000P01(1)/2)/cos(thetaLens)+0.5)*sin(2*thetaLens);xpositionLens+2*((CM5081000P
01(1)/2)/cos(thetaLens)+0.5+0.004)*cos(2*thetaLens),AC254500A2,0,zpositionLens+2*((CM
5081000P01(1)/2)/cos(thetaLens)+0.5+0.004)*sin(2*thetaLens)];%xpositionLens+e1/2+e2/2
,R2,R4,e2,Diam,index2];
%Lens=[xpositionLens,CM5081000P01,thetaLens,zpositionLens;xpositionLens+2*(CM5081000P
01(1)/2)/cos(thetaLens)*cos(2*thetaLens),FlatMirror,thetaLens,zpositionLens+2*(CM5081
000P01(1)/2)/cos(thetaLens)*sin(2*thetaLens)];%xpositionLens+e1/2+e2/2,R2,R4,e2,Diam,
index2];
%Lens=[xpositionLens,CM5081000G01,thetaLens,zpositionLens;xpositionLens-
1*cos(2*thetaLens),FlatMirror,thetaLens,zpositionLens-
1*sin(2*thetaLens)];%xpositionLens+e1/2+e2/2,R2,R4,e2,Diam,index2];
%Lens=[xpositionLens,CM508750P01,thetaLens,zpositionLens;xpositionLens-
2.01*cos(2*thetaLens),CM5081000G01,thetaLens,zpositionLens-
2.01*sin(2*thetaLens)];%xpositionLens+e1/2+e2/2,R2,R4,e2,Diam,index2];
%Lens=[xpositionLens,CM5081000G01,thetaLens,zpositionLens];

%Lens=[0.15,AC254500A1,0,0]%;0.15+0.003,AC254500A2,0,0];

%pas=Diam/50;
%zEntryLens=[0:pas:Diam];  % different height entries for the beam
%zEntry=zEntryLens;
%rayTab=zeros(1+2*size(Lens,1)+1,5,length(zEntryLens)); %z,x,theta
%rayTab=zeros(45,8,size(lambda,1),size(BL,1),size(DeltaT,1),size(BulkT,1),size(zEntry
,1));
%FL=zeros(length(zEntryLens),size(Lens,1));
%cross=zeros(length(zEntryLens),size(Lens,1));
sideUp=zeros(2,2,size(Lens,1));
sideDown=zeros(2,2,size(Lens,1));

%zEntry=zeros(10+length(T(:,1))+length([HChannel+0.0005:0.0005:Diam/4].')+1,length(BL
));
%zEntry=zeros(400+length(T(:,1))+1,length(BL));
zEntry=zeros(length(T(:,1)),length(BL));
ray=zeros(1,10,length(zEntry),length(BL));
for j=1:length(BL)
    %pas=(HChannel-BL(j)*10^(-3)-0.0005-(-Diam/4))/10;
    %zEntry(:,j)=[[-Diam/4:pas:HChannel-BL(j)*10^(-3)-
0.0005].';BL(j)*T(:,1)+HChannel-BL(j)*10^(-3);[HChannel+0.0005:0.0005:Diam/4].'];
    pas=0.0001;%(HChannel-BL(j)*10^(-3)-0.0005)/400;

    % to use the external T-gradient file
    %zEntry(:,j)=[BL(j)*T(:,1)+HChannel-BL(j)*10^(-3)];%[[0:pas:HChannel-BL(j)*10^(-
3)-0.0005].';BL(j)*T(:,1)+HChannel-BL(j)*10^(-3)];

    %to use the function T-gradient
    zEntry(:,j)=T(:,1);%,m,1);

    for m=1:length(zEntry(:,j))
```

```
        ray(1,1,m,j)=zEntry(m,j);
        ray(1,7,m,j)=1;
        ray(1,10,m,j)=1;
    end
end
%zEntry=[[0:0.002:HChannel-BL*10^(-3)-0.002].';BL*T(:,1)+HChannel-BL*10^(-3)];


%-----------------------------------------------------------------------
% TO PLOT A BEAM OF LIGHT GOING THROUGH THE GRIN MEDIUM AND THE LENSES
%-----------------------------------------------------------------------


for i=1:length(lambda)
    for j=1:length(BL)
        for k=1:length(DeltaT)
            for l=1:length(BulkT)
                for m=1:length(zEntry)

[rho,iin,nList,Tmodif,rayTab(:,:,i,j,k,l,m),LstuckTab(i,j,k,l,m)]=bendingbeamInGRIN(l
ambda(i), BL(j), DeltaT(k), BulkT(l),ray(:,:,m,j),PC, LModule,
HChannel,xpositionModule, pi()/2,T(:,:,j,k));
%                     %for n=1:length(minResol)
%                     %    if ray(1,1,m)<HChannel-minResol(1,n) &&
ray(1,1,m+1)>=HChannel-minResol(1,n)
%                     %        LstuckResol(i,j,k,l,n)=Lstuck(i,j,k,l,m);
%                     %    end
%                     %end
%                     %Lstucktest=squeeze(LstuckTab);
%                         %for n=1:length(minResol)
%                            % if ray(1,1,m)<HChannel-minResol(1,n) &&
ray(1,1,m+1)>=HChannel-minResol(1,n)
%                                 Lstuckcurve(i,j,k,l,m,1)=zEntry(m);
%                                 Lstuckcurve(i,j,k,l,m,2)=LstuckTab(i,j,k,l,m);
%                                 Lstuckcurve=squeeze(Lstuckcurve);
%                                 %LstuckResol(i,j,k,l,n,o)=LstuckTab(i,j,k,l,m,o);
%                             %end
%                         %end
                end
            end
        end
    end
end
%Lstuckcurve=[zEntry,squeeze(LstuckTab)];
%LstuckResol=squeeze(LstuckResol);

for m=1:length(DeltaT)
    for i=1:length(zEntry)
        %ray=[zEntryLens(i),0,0,0,0];
        rayIn=rayTab(:,:,1,1,m,1,i);

[rayTabOut(:,:,m,i),FL(i,:),cross(i,:)]=bendingbeamInLenses(Lens,rayIn,xpositionScree
n);
        i
    end
```

```
end

for i=1:length(lambda)
    for j=1:length(BL)
        for k=1:length(DeltaT)
            for l=1:length(BulkT)
                %for o=1:size(LModule,2)
                    m=floor(size(zEntry,1)/2);
                    while LstuckTab(i,j,k,l,m)==0 %Lstuckcurve(m,k+1)==0
%Lstuckcurve(i,j,k,l,m,2)==0
                        m=m+1;
                    end
                    %LResol(i,j,k,l,2)=Lstuckcurve(i,j,k,l,m-1,2);
                    %LResol(i,j,k,l,1)=HChannel-Lstuckcurve(i,j,k,l,m-1,1);
                    LResol(i,j,k,l)=HChannel-zEntry(m-1);
                %end
            end
        end
    end
end





%-----------------------------------------------------------
% TO PLOT A BEAM OF LIGHT GOING ONLY THROUGH THE GRIN MEDIUM
%-----------------------------------------------------------
%
%
for i=1:length(lambda)
    for j=1:length(BL)
        for k=1:length(DeltaT)
            for l=1:length(BulkT)
                for m=1:size(zEntry,1)
                    for o=1:size(LModule,2)

[rho,iin,nList,Tmodif,rayTab(:,:,i,j,k,l,m),LstuckTab(i,j,k,l,m,o)]=bendingbeamInGRIN
(lambda(i), BL(j), DeltaT(k), BulkT(l),ray(:,:,m,j),PC, LModule(o),
HChannel,xpositionModule, pi()/2,T(:,:,j,k));
                        Lstucktest=squeeze(LstuckTab);
                        %for n=1:length(minResol)
                           % if ray(1,1,m)<HChannel-minResol(1,n) &&
ray(1,1,m+1)>=HChannel-minResol(1,n)
                                Lstuckcurve(i,j,k,l,m,o,1)=zEntry(m);
                                Lstuckcurve(i,j,k,l,m,o,2)=LstuckTab(i,j,k,l,m,o);
                                %LstuckResol(i,j,k,l,n,o)=LstuckTab(i,j,k,l,m,o);
                             %end
                        %end
                    end
                end
            end
        end
    end
```

```
end
%
% for i=1:length(lambda)
%     for j=1:length(BL)
%         for k=1:length(DeltaT)
%             for l=1:length(BulkT)
%                 for o=1:size(LModule,2)
%                     m=floor(size(zEntry,1)/2);
%                     while Lstuckcurve(i,j,k,l,m,o,2)==0
%                         m=m+1;
%                     end
%                     LResol(i,j,k,l,o,2)=Lstuckcurve(i,j,k,l,m,o,2);
%                     LResol(i,j,k,l,o,1)=HChannel-Lstuckcurve(i,j,k,l,m,o,1);
%                 end
%             end
%         end
%     end
% end
%
%
% %LstuckResol=squeeze(LstuckResol);
```

```
%-------------------------------------------------------------
% TO PLOT A BEAM OF LIGHT GOING THROUGH ONLY A SET OF LENSES
%-------------------------------------------------------------
```

```
% for m=1:length(BL)
%     for i=1:length(zEntry)
%         rayIn=[zEntry(i),0,0,0,0,0,1,0,0];
%
[rayTabOut(:,:,m,i),FL(i,:),cross(i,:)]=bendingbeamInLenses(Lens,rayIn,xpositionScree
n);
%         i
%     end
% end
```

```
% for i=1:length(zEntry)
%
DeltaOPL(i,:)=[rayTabOut(1,1,2,i),rayTabOut(size(rayTabOut,1),1,2,i),rayTabOut(size(r
ayTabOut,1),9,2,i),rayTabOut(size(rayTabOut,1),9,2,i)];
% end
```

```
%-------------------------------------------------------------
```

```
% TO DRAW THE OPTICS
%-----------------------------------------------------------


%FL(size(FL,1))
thetaLens1=0;
nbmirror=1;
stepTheta=abs(2*asin(min(Lens(:,6))/(2*abs(max(Lens(:,2)))))/1000);

for i=1:size(Lens,1)

  x1=Lens(i,1);
  R1=Lens(i,2);
  R2=Lens(i,3);
  e=Lens(i,4);
  Diam=Lens(i,5);
  index1=Lens(i,6);
  thetaLens1=thetaLens1+Lens(i,8);
  z1=Lens(i,9);
  %x01=x1+R1*cos(Lens(i,7))-e/2; %center of first side of the lens
  %x02=x1+R2*cos(Lens(i,7))+e/2; %center of second side of the lens
  %z0=Lens(i,8)+Diam/2+R1*sin(Lens(i,7)); %ordinate of the centers of the lenses
  %x01=x1+(R1-e/2)*cos(vpa(thetaLens1)); %center of first side of the lens
  %x02=x1+(R2+e/2)*cos(vpa(thetaLens1)); %center of second side of the lens
  %z01=z1+(R1-e/2)*sin(vpa(thetaLens1)); %ordinate of the center of first side of the
lens
  %z02=z1+(R2+e/2)*sin(vpa(thetaLens1)); %ordinate of the center of second side of
the lens
  %x01=x1+nbmirror*(R1-R1/abs(R1)*e/2)*cos(thetaLens1); %center of first side of the
lens
  %x02=x1+nbmirror*(R2-R2/abs(R2)*e/2)*cos(thetaLens1); %center of second side of the
lens
  %z01=z1+nbmirror*(R1-R1/abs(R1)*e/2)*sin(thetaLens1); %ordinate of the center of
first side of the lens
  %z02=z1+nbmirror*(R2-R2/abs(R2)*e/2)*sin(thetaLens1); %ordinate of the center of
second side of the lens
  x01=x1+nbmirror*(R1-e/2)*cos(thetaLens1); %center of first side of the lens
  x02=x1+nbmirror*(R2+e/2)*cos(thetaLens1); %center of second side of the lens
  z01=z1+nbmirror*(R1-e/2)*sin(thetaLens1); %ordinate of the center of first side of
the lens
  z02=z1+nbmirror*(R2+e/2)*sin(thetaLens1); %ordinate of the center of second side of
the lens


% drawing of the left side of the lens
  if R1>=100
      %XCercle1(:,i)=ones(1001,1)*x1;
      %ZCercle1(:,i)=[Lens(i,8)-Diam/2,Lens(i,8)+Diam/2];
      for j=1:1001
        %XCercle1(j,i)=Diam/1000*cos(vpa(pi()/2)-vpa(Lens(i,8)))*j+x1-
Diam/2*cos(vpa(pi()/2)-vpa(Lens(i,8)))-Diam/1000;
        %ZCercle1(j,i)=-Diam/1000*sin(vpa(pi()/2)-
vpa(Lens(i,8)))*j+Lens(i,9)+Diam/2*sin(vpa(pi()/2)-vpa(Lens(i,8)))+Diam/1000;
        XCercle1(j,i)=Diam/1000*cos(vpa(pi()/2)-vpa(thetaLens1))*j+x1-
Diam/2*cos(vpa(pi()/2)-vpa(thetaLens1))-Diam/1000;
```

```
        ZCercle1(j,i)=-Diam/1000*sin(vpa(pi()/2)-
vpa(thetaLens1))*j+Lens(i,9)+Diam/2*sin(vpa(pi()/2)-vpa(thetaLens1))+Diam/1000;
      end
  else
      stepTheta=(abs(2*asin(Diam/(2*abs(R1))))/1001);
      %Theta1 = ((pi()/2*(R1/abs(R1)+1)-
asin(vpa(Diam/(2*abs(R1))))+Lens(i,7)):stepTheta:(pi()/2*(R1/abs(R1)+1)+asin(vpa(Diam
/(2*abs(R1))))+Lens(i,7))).';
      Theta1 = (-
asin(vpa(Diam/(2*abs(R1))))+thetaLens1:stepTheta:asin(vpa(Diam/(2*abs(R1))))+thetaLen
s1).';
      XCercle1(:,i) = x01 - nbmirror*R1 * cos(vpa(Theta1));
      ZCercle1(:,i) = z01 - nbmirror*R1 * sin(vpa(Theta1));
  end

  if e>0 %index1>=0
% drawing of the right side of the lens
      if R2>=100
          %XCercle2(:,i)=[x1,x1];
          %ZCercle2(:,i)=[Lens(i,8)-Diam/2,Lens(i,8)+Diam/2];
          for j=1:1001
            %XCercle1(j,i)=Diam/1000*cos(vpa(pi()/2)-vpa(Lens(i,8)))*j+x1-
Diam/2*cos(vpa(pi()/2)-vpa(Lens(i,8)))-Diam/1000;
            %ZCercle1(j,i)=-Diam/1000*sin(vpa(pi()/2)-
vpa(Lens(i,8)))*j+Lens(i,9)+Diam/2*sin(vpa(pi()/2)-vpa(Lens(i,8)))+Diam/1000;
            XCercle1(j,i)=Diam/1000*cos(vpa(pi()/2)-vpa(thetaLens1))*j+x1-
Diam/2*cos(vpa(pi()/2)-vpa(thetaLens1))-Diam/1000;
            ZCercle1(j,i)=-Diam/1000*sin(vpa(pi()/2)-
vpa(thetaLens1))*j+Lens(i,9)+Diam/2*sin(vpa(pi()/2)-vpa(thetaLens1))+Diam/1000;
          end
      else
          stepTheta2=(abs(2*asin(Diam/(2*abs(R2)))))/1002;
          %Theta2= (pi()/2*(R2/abs(R2)+1)-
asin(Diam/(2*abs(R2)))+Lens(i,7):stepTheta2:pi()/2*(R2/abs(R2)+1)+asin(Diam/(2*abs(R2
)))+Lens(i,7)).';
          Theta2= (-
asin(Diam/(2*abs(R2)))+thetaLens1:stepTheta2:asin(Diam/(2*abs(R2)))+thetaLens1).';
          XCercle2(:,i) = x02 - nbmirror * R2 * cos(Theta2);
          ZCercle2(:,i) = z02 - nbmirror * R2 * sin(Theta2);

      end

%   sideUp(:,:,i)=[XCercle1(1,i),ZCercle1(1,i);XCercle2(1,i),ZCercle2(1,i)];
%
%sideDown(:,:,i)=[XCercle1(size(XCercle1,1),i),ZCercle1(size(XCercle1,1),i);XCercle2(
size(XCercle1,1),i),ZCercle2(size(XCercle1,1),i)];
%
sideDown(:,:,i)=[XCercle1(size(XCercle1,1),i),ZCercle1(size(ZCercle1,1),i);XCercle2(s
ize(XCercle2,1),i),ZCercle2(size(ZCercle2,1),i)];
%
  end

   nbmirror=nbmirror*index1/abs(index1);

end
```

```
%------------------------
%Plot the beam trajectory
%------------------------



figure(1);
colors=jet(length(zEntry));
for i=1:length(zEntry)

    %plot everything
    plot(rayTabOut(:,2,1,i), rayTabOut(:,1,1,i),'Color',colors(i,:))

    %plot path in GRIN media
    %plot(rayTab(:,2,1,1,1,1,i), rayTab(:,1,1,1,1,1,i),'Color',colors(i,:))

    %axis([-0.3 1.7 -0.08 0.05])
    %axis([0.08 0.15 0.009 0.0101])
    %axis([0.04 0.125 0.00 0.012])
    %axis([0.09 0.11 0.008 0.011])
    %axis([-0.04 0.125 -0.04 0.012])
    %axis([-0.6 0 -0.8 -0.4])
    hold on
end
title('Ray path through the optical set-up')
xlabel('z (m)')
ylabel('y (m)')
hold on




%
% for i=1:length(minResol)-1
%     figure(22);
%     plot(DeltaT,squeeze(LResol(1,1,:,1,i)))
%     title('Lstuck for different Resolution as a function of temperature')
%     xlabel('DeltaT (K)')
%     ylabel('Lstuck (m)')
%     hold on
% end
% hold on
%
% for i=1:length(lambda)
%     figure(23);
%     plot(DeltaT,squeeze(LResol(i,1,:,1,2)))
%     title('Lstuck for different wavelengths as a function of temperature')
%     xlabel('DeltaT (K)')
%     ylabel('Lstuck (m)')
%     hold on
% end
```

```
% hold on
%
% for i=1:length(DeltaT)
%     figure(24);
%     plot(BL,squeeze(LResol(1,:,i,1,2)))
%     title('Lstuck for different Boumdary Layer thicknesses as a function of
temperature')
%     xlabel('BL (m)')
%     ylabel('Lstuck (m)')
%     hold on
% end
% hold on
%
% for i=1:length(DeltaT)
%     figure(26);
%     plot(BulkT,squeeze(LResol(1,1,i,:,2)))
%     title('Lstuck for different Delta T as a function of Bulk temperature')
%     xlabel('BulkT (K)')
%     ylabel('Lstuck (m)')
%     hold on
% end
% hold on

for i=1:size(Lens,1)
  plot(XCercle1(2:size(XCercle1,1)-1,i), ZCercle1(2:size(ZCercle1,1)-1,i),'k') %left
face of the lens
  hold on
  if Lens(i,6)>=0
      plot(XCercle2(:,i), ZCercle2(:,i),'k') %right face of the lens
      hold on
      plot(sideUp(:,2,i),sideUp(:,1,i),'k')
      hold on
      plot(sideDown(:,2,i),sideDown(:,1,i),'k')
      hold on
  end
end



plot(Module(:,1), Module(:,2),'k') %
scatter(xecran,zecran,100)
scatter(xpositionLens+ACT5081000A2(3)/2+1,zpositionLens,100, '*')
%scatter(XPF1,ZPF1,100, '*')
%scatter(XPF2,ZPF2,100,'d')
%scatter(XPF3,ZPF3,100,'o')
scatter(xpositionLens,zpositionLens,100)
axis equal
%axis([min(xpositionModule,Lens(2,1))-0.05 Lens(1,1)+0.05 Lens(2,8)-0.1 0.05])
%axis([0 0.2 0 0.002])
%axis([1.995 2.01 0 0.005])

%axis([0.09 0.12 0.008 0.0106])
%axis([0.09 0.12 0.000 0.0045])%zoom on module
%axis([0.09 0.12 0.0035 0.0045])%zoom on top of module
%axis([0.09 0.12 0.0015 0.0025])%zoom on top of module
```

```
%axis([-1.8 0.12 -1.2 0.0025])%all
%axis([-2 1 -2 1])% global zoom
%axis([-1.3 -1.1 -1.2 -0.9])
%axis([-1.3 0.12 -1.2 0.0106])
%axis([0.5 0.54 -0.06 -0.0]) %zoom on focal point
%axis([0.49 0.5 -0.037 -0.0355]) %zoom on focal point
%axis([0.498 0.554 -0.064 -0.052]) %zoom on focal point
%axis([0.505 0.515 -0.0315 -0.030]) %zoom on focal point without T gradient
%axis([0.02 0.05 -0.22 -0.18]) %zoom on second lens
%axis([0.08 0.1 -0.35 -0.3]) %zoom on second lens
%axis([0.375 0.385 -0.144 -0.141]) %zoom on screen

%-----------------------------------
% Plot of the Intensity on the screen
%-----------------------------------
figure(2);
for m=1:length(DeltaT)
    for i=3:length(zEntry)
        %ray=[zEntryLens(i),0,0,0,0];

rayxecran(i,m)=sqrt((rayTabOut(size(rayTabOut,1),1,m,i))^2+(rayTabOut(size(rayTabOut,
1),2,m,i))^2)-sqrt(zecran^2+xecran^2);%(rayTabOut(size(rayTabOut,1),1,m,i)-
zecran)/abs(rayTabOut(size(rayTabOut,1),1,m,i)-
zecran)*sqrt((rayTabOut(size(rayTabOut,1),1,m,i)-
zecran)^2+(rayTabOut(size(rayTabOut,1),2,m,i)-xecran)^2);
    end
    %plot(rayxecran(:,m),m*ones(size(rayxecran)),'MarkerSize',20)
    %scatter(rayxecran(:,m),m*ones(size(rayxecran,1),1),50,[0 .5 .5],'filled')
    c = linspace(1,10,length(rayxecran(:,m)));
    %scatter(rayxecran(:,m),m*ones(size(rayxecran,1),1),50,c,'filled')
    scatter(rayxecran(:,m),m*ones(size(rayxecran,1),1),50,colors(:,:),'filled')
    hold on
end
title('Intensity on the Screen')
xlabel('Horizontal axis on the Screen')
ylabel('Intensity')
%axis([-0.1 0.04 0 m+1])


figure(3);
for m=1:length(DeltaT)
    for i=3:length(zEntry)
        %ray=[zEntryLens(i),0,0,0,0];

%rayxecran(i,m)=sqrt((rayTabOut(size(rayTabOut,1),1,m,i))^2+(rayTabOut(size(rayTabOut
,1),2,m,i))^2)-sqrt(zecran^2+xecran^2);%(rayTabOut(size(rayTabOut,1),1,m,i)-
zecran)/abs(rayTabOut(size(rayTabOut,1),1,m,i)-
zecran)*sqrt((rayTabOut(size(rayTabOut,1),1,m,i)-
zecran)^2+(rayTabOut(size(rayTabOut,1),2,m,i)-xecran)^2);
        rayxecran(i,m)=(rayTabOut(size(rayTabOut,1),2,m,i)-
xecran)/abs(rayTabOut(size(rayTabOut,1),2,m,i)-
xecran)*sqrt((rayTabOut(size(rayTabOut,1),1,m,i)-
zecran)^2+(rayTabOut(size(rayTabOut,1),2,m,i)-
xecran)^2);%(rayTabOut(size(rayTabOut,1),1,m,i)-
zecran)/abs(rayTabOut(size(rayTabOut,1),1,m,i)-
```

```
zecran)*sqrt((rayTabOut(size(rayTabOut,1),1,m,i)-
zecran)^2+(rayTabOut(size(rayTabOut,1),2,m,i)-xecran)^2);
        rayintensityecran(i,m)=rayTabOut(size(rayTabOut,1),10,m,i)*1000; %*1000 is
just to see something
    end
    %plot(rayxecran(:,m),m*ones(size(rayxecran)),'MarkerSize',20)
    %scatter(rayxecran(:,m),m*ones(size(rayxecran,1),1),50,[0 .5 .5],'filled')
    c = linspace(1,10,length(rayxecran(:,m)));
    %scatter(rayxecran(:,m),m*ones(size(rayxecran,1),1),50,c,'filled')
    %scatter(rayxecran(:,m),m*ones(size(rayxecran,1),1),50,colors(:,:),'filled')
    scatter(rayxecran(:,m),rayintensityecran(:,m),50,colors(:,:),'filled')
    hold on
end
title('Intensity on the Screen')
xlabel('Horizontal axis on the Screen')
ylabel('Intensity')
%axis([-0.0005 0.00004 0 m+1])

writematrix(rayxecran,"rayxecran.txt")
writematrix(rayintensityecran,"rayintensityecran.txt")


%---------------------------------
% Plot of the Temperature gradient
%---------------------------------

figure(4);
plot(T(:,1),T(:,2));
title('Temperature profile in the Hot Channel')
xlabel('Height in the channel (m)')
ylabel('T (K)')



%---------------------------------
% Plot of the Minimum Resolution
%---------------------------------

figure(5);
for i=1:length(lambda)
    for j=1:length(BL)
        %for k=1:length(DeltaT)
            for l=1:length(BulkT)

                    %for o=1:size(LModule,2)
                        plot(DeltaT(1,:)/(BL),squeeze(LResol(i,j,:,l)))
                        title('dmin as a function of Temperature gradient')
                        xlabel('grad T (C/mm)')
                        ylabel('dmin (m)')
                        grid on
                        hold on
                    %end

            end
        %end
```

```
    end
end




function [rhoVar, iin, nVar ,TmodifVar, ray, Lstuck]=bendingbeamInGRIN(lambdaVar,
BLVar, DeltaTVar, BulkTVar, rayVar, PCVar, LModuleVar, HChannelVar, xpositionModule,
alphanormal, TVar)

Lstuck=0;
nair=1.000272;
alphanormal=pi()/2; %angle between the optical axis and the normal to the GRIM medium

%-----------------------------------------------------------------------
% this function calculates the beam path (ray=cumulabsVar) accross a GRIN
% medium of length LModuleVar, height HChannelVar and thickness of the
% polycarbonate sheet PCVar ; with T profile contained in chemin, adapted
% by the boudary layer thickness BLVar, the Temperature Gradient DeltaTVar,
% the bulk temperature of the hot channel BulkTVar ; for different
% wavelengths lambdaVar and entry height of the beam zEntryVar.
% it can also be used to display the Schlieren intensity pattern on a
% screen at a distance DScreenVar.
%-----------------------------------------------------------------------

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                     %
% Calcul of rho as a function of temperature, salinity and pressure %
%                                                                     %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%charging the GSW library to solve the TEOS-10 (from http://www.teos-10.org)
%addpath C:\Users\marie-alix.dalle\Documents\Documents\GSW
%addpath C:\Users\marie-alix.dalle\Documents\Documents\GSW\html
%addpath C:\Users\marie-alix.dalle\Documents\Documents\GSW\library
%addpath C:\Users\marie-alix.dalle\Documents\Documents\GSW\pdf
%addpath C:\Users\marie-alix.dalle\Documents\Documents\GSW\thermodynamics_from_t


addpath G:\GSW
addpath G:\GSW\html
addpath G:\GSW\library
addpath G:\GSW\pdf
addpath G:\GSW\thermodynamics_from_t

%-----------------------------------
% To use an external T-gradient file
%-----------------------------------
%chemin="C:\Users\marie-
alix.dalle\Documents\Documents\MATLAB\TGradientTemplate_1.xlsx"; %TGradientTemplate
is a table with first column=z in mm distance between bulk and membrane, and second
column=DeltaT between bulk and membrane(T(length(T),2)
%T=xlsread(chemin); %Temperature gradient (K)
```

```
%tK=DeltaTVar*T(:,2)+BulkTVar-DeltaTVar; %in-situ temperature values, in °K
%%t=tK-273.15*ones(length(T),1); %only in-situ temperature values, in °C
%z=BLVar*T(:,1); %height steps of the T gradient, in m

%TmodifVar=[z+HChannelVar-BLVar*10^(-3),tK];%[[0:0.0005:HChannelVar-BLVar*10^(-3)-
0.0005].',BulkTVar*ones(length([0:0.0005:HChannelVar-BLVar*10^(-3)-
0.0005].'),1);z+HChannelVar-BLVar*10^(-3),tK];

%-----------------------------------------
% To use the T-gradient in argument (TVar)
%-----------------------------------------
TmodifVar=TVar;



tKmodif=TmodifVar(:,2); %only in-situ temperature values, in °C
tmodif=tKmodif-273.15*ones(length(TmodifVar),1); %only in-situ temperature values, in
°C
pasz=TmodifVar(3,1)-TmodifVar(2,1);
%zmodif=[TmodifVar(1,1)-pasz;TmodifVar(:,1);TmodifVar(size(TmodifVar,1),1)+pasz];
%height steps of the T gradient, in m
zmodif=TmodifVar(:,1); %height steps of the T gradient, in m
%pasz=zmodif(3)-zmodif(2);

SA=zeros(length(TmodifVar),1); %absolute salinity (g.kg-1)
p=10*ones(length(TmodifVar),1); %pressure (dbar)

CT=gsw_CT_from_t(SA,tmodif,p); %Converting in situ T into conservative T

rhoVar=gsw_rho(SA,CT,p); %density


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                             %
% Calcul of n as a function of temperature, density and wavelength  %
%                                                                             %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% setting the equation coefficients (from Schiebener, Refractive [...], 1990)

a0=0.243905091;
a1=0.009535181;
a2=-0.003643581;
a3=0.000265666;
a4=0.001591893;
a5=0.002457338;
a6=0.897478251;
a7=-0.016306618;
lambdaUV=0.229202;
lambdaIR=5.432937;

lambda0=0.589;      % microm
rho0=1000;   % kg.m^-3
T0=273.15;   % K
```

```
alpha=a0+a1*rhoVar/rho0+a2*tKmodif/T0+a3*lambdaVar/lambda0+a4*(lambdaVar/lambda0)^2+a
5/((lambdaVar/lambda0)^2-lambdaUV^2)+a6/((lambdaVar/lambda0)^2-
lambdaIR^2)+a7*(rhoVar/rho0).^2;

nVar=((2*alpha.*rhoVar/rho0+1)./(1-alpha.*rhoVar/rho0)).^(1/2);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                        %
% Calcul of the beam path                                                %
%                                                                        %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% rayVar is the beam path before reaching the module. The only relevant
% data for the calculations here lays in its last line.

xmoduleIn=xpositionModule-LModuleVar/2;
xmoduleOut=xpositionModule+LModuleVar/2;

ray=[rayVar;zeros(length(TmodifVar),10)];
iray=size(rayVar,1);
ray(1,1)
iin=1; %index of the TGRADIENT table at which the light ray enters, ie value of i in
TGRADIENT table such that theta(i,1)>zEntryVar && theta(i-1,1)<zEntryVar
ray(iray,7)

if ray(iray,7)==1 && ray(iray,2)<xmoduleIn % if the beam is not blocked, first make
it reach the module located at xpositionModule
        ray(iray,6)=nair;
        ray(iray+1,1)=ray(iray,1)+(xmoduleIn-ray(iray,2))*tan(vpa(ray(iray,3)));
        ray(iray+1,2)=xmoduleIn;
        ray(iray+1,3)=ray(iray,3);
        ray(iray+1,4)=ray(iray,4);
        ray(iray+1,5)=ray(iray,5);
        ray(iray+1,6)=ray(iray,6);
        ray(iray+1,7)=ray(iray,7);
        ray(iray+1,8)=ray(iray,8);
        ray(iray+1,10)=ray(iray,10);

        if iray>1
            ray(iray+1,9)=ray(iray,9)+ray(iray,6)*(sqrt((vpa(ray(iray,2))-
vpa(ray(iray-1,2)))^2+(vpa(ray(iray,1))-vpa(ray(iray-1,1)))^2));
        else

ray(iray+1,9)=ray(iray,9)+ray(iray,6)*sqrt((vpa(ray(iray,2)))^2+(vpa(ray(iray,1)))^2)
;
        end

        iray=iray+1;

        if or(ray(iray,1)>zmodif(size(zmodif,1)),ray(iray,1)<zmodif(1)) %if the beam
is above or below the module
            if tan(ray(iray,4))==0
                ray(iray+1,1)=ray(iray,1);
```

```
            else
                ray(iray+1,1)=(xmoduleOut+PCVar-ray(iray,2))/tan(vpa(ray(iray,4)));
            end
            ray(iray+1,2)=xmoduleOut+PCVar;   % the beam goes straight until the end
of the module
            ray(iray+1,6)=ray(iray,6);
            ray(iray+1,3)=ray(iray,3); % angle between the beam and the horizontal
            ray(iray+1,4)=ray(iray,4); % incident angle between the beam and the GRIN
media surface normal
            ray(iray+1,5)=ray(iray,5); % exiting angle between the beam and the GRIN
media surface normal
            ray(iray+1,7)=1; % the beam can continue its path
            ray(iray+1,8)=ray(iray,8);
            ray(iray+1,10)=ray(iray,10);

            if iray>1
                ray(iray+1,9)=ray(iray,9)+ray(iray,6)*sqrt((ray(iray,2)-ray(iray-
1,2))^2+(ray(iray,1)-ray(iray-1,1))^2);
            else

ray(iray+1,9)=ray(iray,9)+ray(iray,6)*sqrt((ray(iray,2))^2+(ray(iray,1))^2);
            end
            %ray
            iray=iray+1;

            abovebelow=1

        elseif or(ray(iray,1)==zmodif(size(zmodif,1)),ray(iray,1)==zmodif(1)) %if the
beam is exactly on the verges of the module
            ray(iray,7)=0; %the beam is stuck by the membrane
            ray(iray,10)=0; %the beam is stuck by the membrane
            Lstuck=ray(iray,2);

        else %if the beam crosses the module
            for i=2:length(TmodifVar) % to initialize the values of z and n at the
location where the light ray enters the t gradient according to the value of
zEntryVar
                if zmodif(i)>ray(iray,1) && zmodif(i-1)<=ray(iray,1)
                    iin=i-1;
                    ray(iray+1,:)=ray(iray,:);
                    ray(iray+1,4)=(alphanormal+ray(iray,4)); %set up of the incident
angle between the beam and the GRIN media surface normal
                    ray(iray+1,5)=(alphanormal+ray(iray,5)); %set up of the incident
angle between the beam and the GRIN media surface normal
                    iray=iray+1;
                end
            end


            iin

            if or(iin>=length(zmodif), iin<=1)
                ray(iray+1,7)=0;
```

```
            elseif iin<length(zmodif) && iin>1 && nVar(iin)==nVar(iin+1) &&
nVar(iin)==nVar(iin-1) && ray(iray,3)==0 %if the beam is in the module in the Bulk
phase and horizontal
                ray(iray+1,1)=ray(iray,1);
                ray(iray+1,2)=xmoduleOut+PCVar;
                ray(iray+1,6)=ray(iray,6);
                ray(iray+1,3)=ray(iray,3); % angle between the beam and the
horizontal
                ray(iray+1,4)=ray(iray,4); % incident angle between the beam and the
GRIN media surface normal
                ray(iray+1,5)=ray(iray,5); % exiting angle between the beam and the
GRIN media surface normal
                ray(iray+1,7)=1; % the beam can continue its path
                ray(iray+1,8)=ray(iray,8);
                ray(iray+1,10)=ray(iray,10);

                if iray>1
                    ray(iray+1,9)=ray(iray,9)+ray(iray,6)*sqrt((ray(iray,2)-ray(iray-
1,2))^2+(ray(iray,1)-ray(iray-1,1))^2);
                else

ray(iray+1,9)=ray(iray,9)+ray(iray,6)*sqrt((ray(iray,2))^2+(ray(iray,1))^2);
                end
                ray;
                iray=iray+1;

                bulkhorizontal=2

            else % if the beam crosses the temperature gradient
                while ray(iray,2)<xmoduleOut && round(ray(iray,1),15)>=0 &&
round(ray(iray,1),15)<=HChannelVar && ray(iray,7)>0 %while the beam is inside the
GRIN medium and not blocked
                    iray
                    j=1; %to find at which "position" is the beam, to know at what
refractive index it correspond
                    while ray(iray,1)>zmodif(j) && j<length(zmodif) && j>=1
                        j=j+1;
                    end

                    j=j-1;
                    %ray(iray+1,3)=vpa(ray(iray,5))-alphanormal; % angle between the
beam and the horizontal
                    %ray(iray+1,4)=(alphanormal+ray(iray,3)); %set up of the incident
angle between the beam and the GRIN media surface normal
                    ray(iray+1,4)=ray(iray,5); % incident angle between the beam and
the GRIN media surface normal
                    ray(iray+1,8)=ray(iray,8);
                    ray(iray+1,10)=ray(iray,10);
                    ray(iray+1,7)=1;

%                     if ray(iray,1)>HChannelVar %iin+j+1==size(zmodif,1)+1
%                         ray(iray+1,6)=1.00072;
%                         ray(iray+1,5)=vpa(pi()/2)-vpa(ray(iray,3)); % exiting angle
between the beam and the GRIN media surface normal
%                         ray(iray+1,3)=vpa(ray(iray,5))-alphanormal;
```

```
%                          ray(iray+1,1)=ray(iray,1)+pasz;
%                          %ray(iray+1,2)=ray(iray,2)+pasz*tan(ray(iray+1,5));

                        if or(j<1, j>length(nVar))
                            ray(iray+1,7)=0;
                            ray(iray+1,10)=0;
                            Lstuck=ray(iray,2);
                        elseif
or(round(ray(iray,1),15)==0,round(ray(iray,1),15)==HChannelVar)%if the beam is on the
membrane or on the wall it is reflected
                            ray(iray+1,:)=ray(iray,:);

                            %^^^^^^^^^^^
                            %if we want the beam to be stuck at the
                            %membrane
                            ray(iray+1,7)=0;
                            ray(iray+1,10)=0;
                            Lstuck=ray(iray,2)

                            %^^^^^^^^^^^
                            % if we want reflection on the membrane
                            %ray(iray+1,4)=-ray(iray,5);
                            %ray(iray+1,5)=-ray(iray,5);
                            %ray(iray+1,3)=pi()+ray(iray,5)-alphanormal;
                            %
                            %if round(ray(iray,1),15)==0
                            %    ray(iray+1,1)=ray(iray,1)+pasz;
                            %else
                            %    ray(iray+1,1)=ray(iray,1)-pasz;
                            %end




                        elseif vpa(ray(iray+1,4))==vpa(pi()/2) %if the beam arrives
parallel to the surface, it goes into the medium of the smallest refractive index

                            if j<length(nVar) && nVar(j+1)>nVar(j)
                                ray(iray+1,6)=nVar(j+1);
                                if abs(nVar(j)/nVar(j+1)*sin(vpa(ray(iray+1,4))))<=1
                                    ray(iray+1,5)=-
asin(nVar(j)/vpa(ray(iray+1,6))*sin(vpa(ray(iray+1,4)))); % exiting angle between the
beam and the GRIN media surface normal
                                    ray(iray+1,1)=ray(iray,1)+pasz;
                                    ray(iray+1,3)=vpa(ray(iray,5))+alphanormal;
                                else
                                    ray(iray+1,5)=-ray(iray+1,4); % exiting angle
between the beam and the GRIN media surface normal
                                    ray(iray+1,1)=ray(iray,1)-pasz;
                                    ray(iray+1,3)=pi+vpa(ray(iray,5))-alphanormal;
                                end
                                %ray(iray+1,2)=ray(iray,2)+pasz*tan(ray(iray+1,5));
                            elseif j>1 && nVar(j-1)>nVar(j)
                                ray(iray+1,6)=nVar(j-1);
```

```
                                if abs(nVar(j)/nVar(j-1)*sin(vpa(ray(iray+1,4))))<=1

ray(iray+1,5)=asin(nVar(j)/vpa(ray(iray+1,6))*sin(vpa(ray(iray+1,4)))); % exiting
angle between the beam and the GRIN media surface normal
                                    ray(iray+1,3)=vpa(ray(iray,5))+alphanormal;
                                    ray(iray+1,1)=ray(iray,1)-pasz;
                                else
                                    ray(iray+1,5)=-ray(iray+1,4); % exiting angle
between the beam and the GRIN media surface normal
                                    ray(iray+1,1)=ray(iray,1)+pasz;
                                    ray(iray+1,3)=pi+vpa(ray(iray,5))-alphanormal;
                                end
                                %ray(iray+1,2)=ray(iray,2)+pasz*tan(ray(iray+1,5));
                            else %if all the surrounding medium are of higher
refractive index, the beam is blocked (or does it continue straight?)
                                ray(iray+1,7)=0;
                                Lstuck=ray(iray,2)
                            end
                        elseif j>1 && ray(iray+1,4)>0 % if the beam arrives from the
top of the surface
                            ray(iray+1,6)=nVar(j-1);
                            if abs(nVar(j)/nVar(j-1)*sin(vpa(ray(iray+1,4))))<=1

ray(iray+1,5)=asin(nVar(j)/vpa(ray(iray+1,6))*sin(vpa(ray(iray+1,4)))); % exiting
angle between the beam and the GRIN media surface normal
                                ray(iray+1,3)=vpa(ray(iray,5))+alphanormal;
                                ray(iray+1,1)=ray(iray,1)-pasz;
                            else
                                ray(iray+1,5)=-ray(iray+1,4); % exiting angle between
the beam and the GRIN media surface normal
                                ray(iray+1,1)=ray(iray,1)+pasz;
                                ray(iray+1,3)=pi+vpa(ray(iray,5))-alphanormal;
                            end
                            %ray(iray+1,2)=ray(iray,2)+pasz*tan(ray(iray+1,5));
                        elseif j<length(nVar) && ray(iray+1,4)<0  % if the beam
arrives from the bottom of the surface
                            iray+1
                            j+1
                            ray(iray+1,6)=nVar(j+1);
                            if abs(nVar(j)/nVar(j+1)*sin(vpa(ray(iray+1,4))))<=1

ray(iray+1,5)=asin(nVar(j)/vpa(ray(iray+1,6))*sin(vpa(ray(iray+1,4)))); % exiting
angle between the beam and the GRIN media surface normal
                                ray(iray+1,3)=vpa(ray(iray,5))+alphanormal;
                                ray(iray+1,1)=ray(iray,1)+pasz;
                            else
                                ray(iray+1,5)=-ray(iray+1,4); % exiting angle between
the beam and the GRIN media surface normal
                                ray(iray+1,1)=ray(iray,1)-pasz;
                                ray(iray+1,3)=pi+vpa(ray(iray,5))-alphanormal;
                            end
                            %ray(iray+1,2)=ray(iray,2)+pasz*tan(ray(iray+1,5));
                        else
                            ray(iray+1,7)=0;
                            ray(iray+1,10)=0;
```

```
                            Lstuck=ray(iray,2);
                        end


                    if ray(iray+1,7)>0
                        if ray(iray+1,1)>HChannelVar
                            ray(iray+1,1)=HChannelVar;
                            ray(iray+1,2)=ray(iray,2)+(HChannelVar-
ray(iray,1))*abs(tan(vpa(ray(iray+1,5))));
                        else
                            if
round(ray(iray,2)+pasz*abs(tan(ray(iray+1,5))),15)<=xmoduleOut

ray(iray+1,2)=ray(iray,2)+pasz*abs(tan(vpa(ray(iray+1,5))));
                            else
                                ray(iray+1,2)=xmoduleOut;
                                ray(iray+1,1)=ray(iray,1)-(xmoduleOut-
ray(iray,2))/tan(vpa(ray(iray+1,5)));
                            end
                        end
                    end


                    if iray>1
                        ray(iray+1,9)=ray(iray,9)+ray(iray,6)*sqrt((vpa(ray(iray,2))-
vpa(ray(iray-1,2)))^2+(vpa(ray(iray,1))-vpa(ray(iray-1,1)))^2);
                    else

ray(iray+1,9)=ray(iray,9)+ray(iray,6)*sqrt((vpa(ray(iray,2)))^2+(vpa(ray(iray,1)))^2)
;
                    end
                    crossed=3 %ray

                    iray=iray+1;
                end

%                 if
or(round(ray(iray,1),15)==round(HChannelVar,15),round(ray(iray,1),15)==0) &&
ray(iray,2)<xmoduleOut && ray(iray,7)>0 %if the beam touches the membrane or the
bottom wall of the hot channel
%                            %initialization of the reflection
%                            ray(iray+1,:)=ray(iray,:);
%                            ray(iray+1,4)=-ray(iray,5);
%                            ray(iray+1,5)=-ray(iray,5);
%                            ray(iray+1,3)=pi()+ray(iray,5)-alphanormal;
%                            %ray(iray+1,5)=alphanormal-vpa(ray(iray,3)); % exiting
angle between the beam and the GRIN media surface normal
%                            %ray(iray+1,6)=n(length(n));
%                            %ray(iray+1,6)=ray(iray-1,6);
%         %                    ray(iray+1,6)=n(iin+j-1);
%         %                    ray(iray+1,3)=-ray(iray,3); % angle between the beam
and the horizontal
%         %                    ray(iray+1,4)=pi()/2+ray(iray+1,3); % incident angle
between the beam and the GRIN media surface normal
```

```
%         %
ray(iray+1,5)=asin(ray(iray,6)/ray(iray+1,6)*sin(ray(iray+1,4))); % exiting angle
between the beam and the GRIN media surface normal
%         %                     ray(iray+1,1)=zmodif(iin+j-1-1);
%         %                     ray(iray+1,8)=ray(iray,8);
%         %                     ray(iray+1,7)=1;
%         % %
%         %                     if ray(iray,2)+(zmodif(iin+j-1)-zmodif(iin+j-1-
1))*tan(ray(iray+1,4))<=xmoduleOut
%         %                         ray(iray+1,2)=ray(iray,2)+(zmodif(iin+j-1)-
zmodif(iin+j-1-1))*tan(ray(iray+1,4));
%         %                     else
%         %                         ray(iray+1,2)=xmoduleOut;
%         %                         ray(iray+1,1)=ray(iray,1)-(xmoduleOut-
ray(iray,2))/tan(ray(iray+1,4));
%         %                     end
%         %
%                             iray=iray+1;
%
%
%                             %propagation in the GRIN media again
%                             while ray(iray,1)<=HChannelVar && ray(iray,1)>=0 &&
round(ray(iray,2),15)<round(xmoduleOut,15) && ray(iray,7)>0
%
%                                 k=1;%to find at which "position" is the beam, to
know what refractive index it is
%                                 while ray(iray,1)>zmodif(k)
%                                     k=k+1;
%                                 end
%
%
%                                 %ray(iray+1,3)=alphanormal-vpa(ray(iray,5)); %
angle between the beam and the horizontal
%                                 ray(iray+1,4)=ray(iray,5); % incident angle between
the beam and the GRIN media surface normal
%                                 ray(iray+1,8)=ray(iray,8);
%                                 ray(iray+1,7)=1;
%
%
% %                                 if ray(iray+1,4)==pi()/2
% %                                     ray(iray+1,5)=pi()/2;
% %                                 else
% %                                     if
ray(iray,6)/ray(iray+1,6)*sin(ray(iray+1,4))<1
% %
ray(iray+1,5)=asin(ray(iray,6)/ray(iray+1,6)*sin(ray(iray+1,4))); % exiting angle
between the beam and the GRIN media surface normal
% %                                     else
% %                                         ray(iray+1,5)=pi()/2;
% %
ray(iray+1,6)=ray(iray,6)*sin(ray(iray+1,4));
% %                                         ray(iray+1,1)=(pasz)/(n(k)-
n(k+1))*(ray(iray+1,6)-n(k))+zmodif(k);
% %                                         %faut il recalculer x aussi? -> non car
le
```

```
% %                                          %rayon va tout droit
% %                                       end
% %                                    end
% %                                    ray(iray+1,1)=vpa(zmodif(k));
% %                                    ray(iray+1,8)=vpa(ray(iray,8));
% %                                    ray(iray+1,7)=1;
%
%
%                                  if ray(iray,1)>HChannelVar
%iin+j+1==size(zmodif,1)+1
%                                      ray(iray+1,6)=1.00072;
%                                      ray(iray+1,5)=alphanormal-vpa(ray(iray,3)); %
exiting angle between the beam and the GRIN media surface normal
%                                      ray(iray+1,1)=ray(iray,1)+pasz;
%
%ray(iray+1,2)=ray(iray,2)+pasz*tan(ray(iray+1,5));
%                                  else
%                                      if vpa(ray(iray+1,4))==vpa(pi()/2) %if the beam
arrives parallel to the surface, it goes into the medium of the smallest refractive
index
%                                          if nVar(k+1)>nVar(k)
%                                              ray(iray+1,6)=nVar(k+1);
%                                              ray(iray+1,5)=-
asin(nVar(k)/vpa(ray(iray+1,6))*sin(vpa(ray(iray+1,4)))); % exiting angle between the
beam and the GRIN media surface normal
%
ray(iray+1,3)=vpa(ray(iray,5))+alphanormal;
%                                              ray(iray+1,1)=ray(iray,1)+pasz;
%
%ray(iray+1,2)=ray(iray,2)+pasz*tan(ray(iray+1,5));
%                                          elseif nVar(k-1)>nVar(k)
%                                              ray(iray+1,6)=nVar(k-1);
%
ray(iray+1,5)=asin(nVar(k)/vpa(ray(iray+1,6))*sin(vpa(ray(iray+1,4)))); % exiting
angle between the beam and the GRIN media surface normal
%
ray(iray+1,3)=vpa(ray(iray,5))+alphanormal;
%                                              ray(iray+1,1)=ray(iray,1)-pasz;
%
%ray(iray+1,2)=ray(iray,2)+pasz*tan(ray(iray+1,5));
%                                          else %if all the surrounding medium are of
higher refractive index, the beam is blocked (or does it continue straight?)
%                                              ray(iray+1,7)=0;
%                                          end
%                                      elseif ray(iray+1,4)>0 % if the beam arrives
from the top of the surface
%                                          ray(iray+1,6)=nVar(k-1);
%
ray(iray+1,5)=asin(nVar(k)/vpa(ray(iray+1,6))*sin(vpa(ray(iray+1,4)))); % exiting
angle between the beam and the GRIN media surface normal
%                                          ray(iray+1,3)=vpa(ray(iray,5))+alphanormal;
%                                          ray(iray+1,1)=ray(iray,1)-pasz;
%
%ray(iray+1,2)=ray(iray,2)+pasz*tan(ray(iray+1,5));
```

```
%                                       elseif ray(iray+1,4)<0 % if the beam arrives
from the bottom of the surface
%                                           ray(iray+1,6)=nVar(k+1);
%
ray(iray+1,5)=asin(nVar(k)/vpa(ray(iray+1,6))*sin(vpa(ray(iray+1,4)))); % exiting
angle between the beam and the GRIN media surface normal
%
ray(iray+1,3)=+vpa(ray(iray,5))+alphanormal;
%                                           ray(iray+1,1)=ray(iray,1)+pasz;
%
%ray(iray+1,2)=ray(iray,2)+pasz*tan(ray(iray+1,5));
%                                       end
%                                   end
%
%
%                                   if
ray(iray,2)+pasz*abs(tan(ray(iray+1,5)))<=xmoduleOut
%
ray(iray+1,2)=ray(iray,2)+pasz*abs(tan(vpa(ray(iray+1,5))));
%                                   else
%                                       ray(iray+1,2)=xmoduleOut;
%                                       ray(iray+1,1)=ray(iray,1)-(xmoduleOut-
ray(iray,2))/tan(vpa(ray(iray+1,5)));
%                                   end
%
%
%                                   if iray>1
%
ray(iray+1,9)=ray(iray,9)+ray(iray,6)*sqrt((vpa(ray(iray,2))-vpa(ray(iray-
1,2)))^2+(vpa(ray(iray,1))-vpa(ray(iray-1,1)))^2);
%                                   else
%
ray(iray+1,9)=ray(iray,9)+ray(iray,6)*sqrt((vpa(ray(iray,2)))^2+(vpa(ray(iray,1)))^2)
;
%                                   end
% %                                   crossed=3 %ray
% %
% %                                   iray=iray+1;
%
%
%
%                               %ray(iray+1,2)=ray(iray,2)+(zmodif(iin+j-1-1-k)-
zmodif(iin+j-1-1-k-1))*tan(ray(iray+1,5));
%
%
% %                                   if
round(ray(iray,2)+(pasz)*tan(ray(iray+1,5)),15)<=round(xmoduleOut,15)
% %
ray(iray+1,2)=vpa(ray(iray,2))+(pasz)*vpa(tan(vpa(ray(iray+1,5))));
% %                                   else
% %                                       ray(iray+1,2)=vpa(xmoduleOut);
% %                                       ray(iray+1,1)=vpa(ray(iray,1))-
(vpa(xmoduleOut)-vpa(ray(iray,2)))/vpa(tan(vpa(ray(iray+1,5))));
% %                                   end
% %
```

```
% %
% %
% %                                   if iray>1
% %
ray(iray+1,9)=ray(iray,9)+ray(iray,6)*sqrt((ray(iray,2)-ray(iray-
1,2))^2+(ray(iray,1)-ray(iray-1,1))^2);
% %                                   else
% %
ray(iray+1,9)=ray(iray,9)+ray(iray,6)*sqrt((ray(iray,2))^2+(ray(iray,1))^2);
% %                                   end
%
%                                 reflected=7 %ray
%
%                                 iray=iray+1;
%                             end
%
%                 end



                if or(ray(iray,7)==0,round(ray(iray,2),15)<round(xmoduleOut,15) &&
round(ray(iray,1),15)==round(HChannelVar,15)) %beam stuck at the membrane
                    zScreenVar=-100;
                    ray(iray,7)=0; %the beam is stuck by the membrane
                    ray(iray,10)=0; %the beam is stuck by the membrane
                else %beams exiting the membrane

                    %to get out of the module into the PC sheet
                    ray(iray+1,2)=ray(iray,2)+PCVar; %value of x after crossing the
PC sheet
                    ray(iray+1,4)=-ray(iray,5)/abs(ray(iray,5))*(pi()/2+ray(iray,5));
% angle of the incident beam with the normal of the PC sheet
                    ray(iray+1,5)=asin(ray(iray,6)/1.591*sin(ray(iray+1,4))); % angle
of the exiting beam with the normal of the PC sheet (=angle between the beam and the
horizontal)
                    %ray(iray+1,3)=asin(ray(iray,6)/1.591*sin(-
ray(iray,5)/abs(ray(iray,5))*(pi()/2+ray(iray,5)))); % angle between the beam and the
horizontal
                    ray(iray+1,3)=ray(iray+1,5);
                    ray(iray+1,1)=ray(iray,1)+PCVar*tan(ray(iray+1,5)); %value of z
after crossing the PC sheet
                    ray(iray+1,7)=1; %the beam can continue its path
                    ray(iray+1,8)=ray(iray,8);
                    ray(iray+1,10)=ray(iray,10);

                    if iray>1
                        ray(iray+1,9)=ray(iray,9)+ray(iray,6)*sqrt((ray(iray,2)-
ray(iray-1,2))^2+(ray(iray,1)-ray(iray-1,1))^2);
                    else

ray(iray+1,9)=ray(iray,9)+ray(iray,6)*sqrt((ray(iray,2))^2+(ray(iray,1))^2);
                    end
                    iray=iray+1;
                    blocked=4 %ray
```

```
                end
            end

             %end

        end



end

while iray< 2*(size(TmodifVar,1)+size(rayVar)-1+1+1)
    ray(iray+1,:)=ray(iray,:);
    ray(iray+1,6)=nair;
    iray=iray+1;
end
%ray
end

function [ray,FL,cross]=bendingbeamInLenses(Lens,rayVar,xpositionScreen)

nair=1.000272;
ray=rayVar;
nbefore=nair;

  function [xsol,zsol]=C(x0,z0,R,ray,nbmirror) % this function calculates the
intersection between the surfaces of the lenses and the ray
        a=tan(ray(3));
        b=ray(1)-a*ray(2);
        A=(1+a^2);
        B=2*(a*b-a*z0-x0);
        C=x0^2+b^2+z0^2-2*b*z0-R^2;

        if R*nbmirror<0%R<0
            if B^2-4*A*C>0
                xsol=(-B+sqrt(B^2-4*A*C))/(2*A);
            else
                xsol=ray(2);
            end
        elseif R*nbmirror>0%R>0
            if B^2-4*A*C>0
                xsol=(-B-sqrt(B^2-4*A*C))/(2*A);
            else
                xsol=ray(2);
            end
        end
        zsol=a*xsol+b;
  end

if rayVar(size(rayVar,1),7)==0 %if the beam is blocked before reaching the lens
    for i=1:size(Lens,1)
        ray=[ray;ray(size(ray,1),:)];
        ray=[ray;ray(size(ray,1),:)];
```

```
        FL(i)=0;
        cross(i)=0;

        ray=[ray;ray(size(ray,1),:)];

    end
    ray=[ray;ray(size(ray,1),:)];

    blocked=1

else %if the beam can pursue its way

thetaLensVar=0;
nbmirror=1;
    for i=1:size(Lens,1)
    %-----------------------------------------------------------------------
    % This function calculates the beam path (ray) of the beam rayVar accross
    % a lens located at at the position x1, with radius of left surface R1/
    % right surface R2, thickness in the middle e, Diameter Diam, index of the
    % material index.
    %-----------------------------------------------------------------------

    % ray is a table collecting the successives coordinates of the beam, with:
    % - first column the z position,
    % - second column the x position,
    % - third column the angle theta between the beam and the horizontal
    % - fourth column the angle iin between the incident beam and the suface
    % normal
    % - fifth column the angle iout between the exiting beam and the surface
    % normal
    % - sixth column the refractive index
    % - seventh column the boolean telling whether the beam is blocked
    % - eighth column beta
    % - ninth column path the optical path
    % - tenth column intensity of the beam

    % positionning of the centers of the lens
        x1=Lens(i,1);
        R1=Lens(i,2);
        R2=Lens(i,3);
        e=Lens(i,4);
        Diam=Lens(i,5);
        index1=Lens(i,6);
        aintensity=Lens(i,7)
        thetaLensVar=thetaLensVar+Lens(i,8);
        zpositionLensVar=Lens(i,9);
        beta=ray(size(ray,1),8);
        intensity=ray(size(ray,1),10);
        xextr=x1+Diam/2*sin(thetaLensVar); % x of filter extremity
        zextr=zpositionLensVar-Diam/2*cos(thetaLensVar); % z of filter extremity

%         if i<size(Lens,1)
%             nafter=Lens(i+1,6);
%         else
%             nafter=nair;
```

```
%         end
%         if i>1
%             nbefore=Lens(i-1,6);
%         else
%             nbefore=nair;
%         end
        nbefore=ray(size(ray,1),6);
        nafter=nair;


        if index1==-1 %if the Lens is a mirror

            mirror=1

            %x01=x1+R1*cos(thetaLens)-e/2; %center of the mirror
            %z0=zpositionLens+Diam/2+R1*sin(thetaLens); %ordinate of the centers of
the mirror
            x01=x1+(R1-e/2)*cos(thetaLensVar); %center of the mirror
            z0=zpositionLensVar+(R1-e/2)*sin(thetaLensVar); %ordinate of the centers
of the mirror

            [xsol1,zsol1]=C(x01,z0,R1,ray(size(ray,1),:),nbmirror);
            xray1=xsol1;
            zray1=zsol1;
            alphan1=min(1,R1/abs(R1)*(zray1-z0)/abs(zray1-
z0))*(acos(sum([1,0].*[R1/abs(R1)*(xray1-x01),R1/abs(R1)*(zray1-z0)])/(abs(R1))));%-
pi()+atan((zray1-z0)/(xray1-x01));%+pi()/2*(1-R1/abs(R1)); %slope of the normal to
the surface

            if round(ray(size(ray,1),2),10)==round(xray1,10) &&
round(ray(size(ray,1),1),10)==round(zray1,10) %if the last location of the beam is
already exactly on the lens (eg for double lenses)
                ray=[ray;ray(size(ray,1),:)];

                pointequal=2

            elseif ray(size(ray,1),3)==alphan1 %if the beam arrives parallel to the
normal to the surface
                iin1=0;
                iout1=0;
                theta1=0;

ray=[ray;zray1,xray1,theta1,iin1,iout1,index1,1,beta,ray(size(ray,1),9)+sqrt((zray1-
ray(size(ray,1),2))^2+(xray1-ray(size(ray,1),1))^2),ray(size(ray,1),10)];

                parallel=3

            else %for every other direction of the beam
                iin1=-min(1,(pi+ray(size(ray,1),3)-
alphan1)/abs(pi+ray(size(ray,1),3)-alphan1))*acos(sum([ray(size(ray,1),2)-
xray1,ray(size(ray,1),1)-zray1].*[R1/abs(R1)*(xray1-x01),R1/abs(R1)*(zray1-
z0)])/(abs(R1)*sqrt((xray1-ray(size(ray,1),2))^2+(zray1-ray(size(ray,1),1))^2)));
%angle of the incident beam to the surface normal
                iout1=-iin1; %angle of the exiting beam to the surface normal
```

```
                %iout1=asin(nbefore/index1*sin(iin1))+pi()/2*(1-index1/abs(index1));
%angle of the exiting beam to the surface normal
                %theta1=ray(size(ray,1),3)+pi()-2*iin1; %angle of the beam to the
horizontal
                %theta1=ray(size(ray,1),3)+pi()+alphan1-iin1; %angle of the beam to
the horizontal
                %theta1=min(1,(zray1-z0)/abs(zray1-
z0))*(acos(sum([1,0].*[R1/abs(R1)*(xray1-x01),R1/abs(R1)*(zray1-
z0)])/(abs(R1)))+pi())-iin1+iout1; %angle of the beam to the horizontal
                theta1=(min(1,R1/abs(R1)*(zray1-z0)/abs(zray1-
z0))*(acos(sum([1,0].*[R1/abs(R1)*(xray1-x01),R1/abs(R1)*(zray1-
z0)])/(abs(R1))))+iout1); %angle of the beam to the horizontal


ray=[ray;zray1,xray1,theta1,iin1,iout1,index1,1,beta,ray(size(ray,1),9)+ray(size(ray,
1),6)*sqrt((zray1-ray(size(ray,1),2))^2+(xray1-
ray(size(ray,1),1))^2),ray(size(ray,1),10)];

                other=4

            end
                ray=[ray;ray(size(ray,1),:)];
                cross(i)=(z0-ray(size(ray,1),1))/tan(pi()-
ray(size(ray,1),3))+ray(size(ray,1),2);
                FL(i)=(z0-ray(size(ray,1),1))/tan(pi()-
ray(size(ray,1),3))+ray(size(ray,1),2)-x1;

                nbmirror=nbmirror*index1;

        elseif index1==nair %if the optical equipment is the gradient filter

            filter=1

            nbefore=nair;

            x01=x1+nbmirror*(R1-e/2)*cos(thetaLensVar); %center of first side of the
filter
            %x02=x1+nbmirror*(R2+e/2)*cos(thetaLensVar); %center of second side of
the filter
            z01=zpositionLensVar+nbmirror*(R1-e/2)*sin(thetaLensVar); %ordinate of
the center of first side of the lens
            %z02=zpositionLensVar+nbmirror*(R2+e/2)*sin(thetaLensVar); %ordinate of
the center of second side of the lens

            [xsol1,zsol1]=C(x01,z01,R1,ray(size(ray,1),:),nbmirror);
            xray1=xsol1;
            zray1=zsol1;
            if zray1==z01
                alphan1=0;
            else
                alphan1=R1/abs(R1)*min(1,(zray1-z01)/abs(zray1-
z01))*acos(sum([1,0].*[R1/abs(R1)*(xray1-x01),R1/abs(R1)*(zray1-
z01)])/(abs(R1)));%atan((zray1-z01)/(xray1-x01)); %slope of the normal to the surface
            end
```

```
            if round(ray(size(ray,1),2),10)==round(xray1,10) &&
round(ray(size(ray,1),1),10)==round(zray1,10) %if the last location of the beam is
already exactly on the lens (eg for double lenses)
%                 %iin1=pi()+ray(size(ray,1),3)-alphan1;
%                 iin1=ray(size(ray,1),5);
%                 iout1=iin1; %angle of the exiting beam to the surface normal
%                 theta1=-pi+alphan1+iout1;
                  sqrt((xray1-x1)^2+(zray1-zpositionLensVar)^2)
                  intensity=intensity*10^(-aintensity*(sqrt((xray1-xextr)^2+(zray1-
zextr)^2)+0.02))
%
%
ray=[ray;zray1,xray1,theta1,iin1,iout1,index1,1,beta,ray(size(ray,1),9)+ray(size(ray,
1),6)*sqrt((zray1-ray(size(ray,1),2))^2+(xray1-ray(size(ray,1),1))^2), intensity];
%
                  ray=[ray;ray(size(ray,1),:)];
                  ray(size(ray,1),10)=intensity;


            elseif ray(size(ray,1),3)==alphan1 %if the beam arrives parallel to the
normal to the surface
                %iin1=0;
                %iout1=0;
                %theta1=0;%alphan1;
                sqrt((xray1-x1)^2+(zray1-zpositionLensVar)^2)
                intensity=intensity*10^(-aintensity*(sqrt((xray1-xextr)^2+(zray1-
zextr)^2)+0.02))


%ray=[ray;zray1,xray1,theta1,iin1,iout1,index1,1,beta,ray(size(ray,1),9)+ray(size(ray
,1),6)*sqrt((zray1-ray(size(ray,1),2))^2+(xray1-ray(size(ray,1),1))^2),intensity];

                ray=[ray;ray(size(ray,1),:)];
                ray(size(ray,1),10)=intensity;


            else %for every other direction of the beam
                %iin1=min(1,(pi()+ray(size(ray,1),3)-alphan1)/abs(ray(size(ray,1),3)-
alphan1))*acos(sum([ray(size(ray,1),2)-xray1,ray(size(ray,1),1)-
zray1].*[R1/abs(R1)*(xray1-x01),R1/abs(R1)*(zray1-z01)])/(abs(R1)*sqrt((xray1-
ray(size(ray,1),2))^2+(zray1-ray(size(ray,1),1))^2))); %angle of the incident beam to
the surface normal
                %iout1=asin(nbefore/index1*sin(iin1)); %angle of the exiting beam to
the surface normal
                %theta1=-pi+alphan1+iout1;
                sqrt((xray1-x1)^2+(zray1-zpositionLensVar)^2)
                intensity=intensity*10^(-aintensity*(sqrt((xray1-xextr)^2+(zray1-
zextr)^2)+0.02))


%ray=[ray;zray1,xray1,theta1,iin1,iout1,index1,1,beta,ray(size(ray,1),9)+ray(size(ray
,1),6)*sqrt((zray1-ray(size(ray,1),2))^2+(xray1-ray(size(ray,1),1))^2),intensity];

                ray=[ray;ray(size(ray,1),:)];
```

```
                ray(size(ray,1),10)=intensity;

            end

            ray=[ray;ray(size(ray,1),:)];
            cross(i)=(zpositionLensVar-
ray(size(ray,1),1))/tan(ray(size(ray,1),3))+ray(size(ray,1),2);
            FL(i)=(zpositionLensVar-
ray(size(ray,1),1))/tan(ray(size(ray,1),3))+ray(size(ray,1),2)-x1;




        else %if the optical equipment is a Lens

            lens=1

            %x01=x1+R1*cos(thetaLens)-e/2; %center of first side of the lens
            %x02=x1+R2*cos(thetaLens)+e/2; %center of second side of the lens
            %z0=zpositionLens+Diam/2+R1*sin(thetaLens); %ordinate of the centers of
the lenses
            %x01=x1+nbmirror*(R1-R1/abs(R1)*e/2)*cos(thetaLens); %center of first
side of the lens
            %x02=x1+nbmirror*(R2-R2/abs(R2)*e/2)*cos(thetaLens); %center of second
side of the lens
            %z01=zpositionLens+nbmirror*(R1-R1/abs(R1)*e/2)*sin(thetaLens); %ordinate
of the center of first side of the lens
            %z02=zpositionLens+nbmirror*(R2-R2/abs(R2)*e/2)*sin(thetaLens); %ordinate
of the center of second side of the lens
            x01=x1+nbmirror*(R1-e/2)*cos(thetaLensVar); %center of first side of the
lens
            x02=x1+nbmirror*(R2+e/2)*cos(thetaLensVar); %center of second side of the
lens
            z01=zpositionLensVar+nbmirror*(R1-e/2)*sin(thetaLensVar); %ordinate of
the center of first side of the lens
            z02=zpositionLensVar+nbmirror*(R2+e/2)*sin(thetaLensVar); %ordinate of
the center of second side of the lens

            [xsol1,zsol1]=C(x01,z01,R1,ray(size(ray,1),:),nbmirror);
            xray1=xsol1;
            zray1=zsol1;
            if zray1==z01
                alphan1=0;
            else
                alphan1=R1/abs(R1)*min(1,(zray1-z01)/abs(zray1-
z01))*acos(sum([1,0].*[R1/abs(R1)*(xray1-x01),R1/abs(R1)*(zray1-
z01)])/(abs(R1)));%atan((zray1-z01)/(xray1-x01)); %slope of the normal to the surface
            end


            if round(ray(size(ray,1),2),10)==round(xray1,10) &&
round(ray(size(ray,1),1),10)==round(zray1,10) %if the last location of the beam is
already exactly on the lens (eg for double lenses)
                %iin1=pi()+ray(size(ray,1),3)-alphan1;
                iin1=ray(size(ray,1),5);
```

```
                iout1=asin(nair/index1*sin(iin1)); %angle of the exiting beam to the
surface normal
                %theta1=R1/abs(R1)*min(1,ray(size(ray,1)-1,3)/abs(ray(size(ray,1)-
1,3)))*(acos(sum([1,0].*[R1/abs(R1)*(xray1-x01),R1/abs(R1)*(zray1-
z01)])/(abs(R1))))+(zray1-ray(size(ray,1),1))/abs(zray1 -
ray(size(ray,1),1))*pi+iout1; %angle of the beam to the horizontal
                %theta1=min(1,ray(size(ray,1)-1,3)/abs(ray(size(ray,1)-
1,3)))*(acos(sum([1,0].*[R1/abs(R1)*(xray1-x01),R1/abs(R1)*(zray1-z01)])/(abs(R1))))-
pi()+iout1; %angle of the beam to the horizontal
                %theta1=min(1,-(zray1-z01)/abs(zray1-
z01))*(acos(sum([1,0].*[R1/abs(R1)*(xray1-x01),R1/abs(R1)*(zray1-z01)])/(abs(R1))))-
pi()+iout1; %angle of the beam to the horizontal
%                 if zray1==ray(size(ray,1),1)
%                     %theta1=R1/abs(R1)*min(1,(zray1-z01)/abs(zray1-
z01))*(acos(sum([1,0].*[R1/abs(R1)*(xray1-x01),R1/abs(R1)*(zray1-
z01)])/(abs(R1))))+pi+iout1; %angle of the beam to the horizontal
%                     theta1=alphan1+pi+iout1;
%                 else
%                     %theta1=R1/abs(R1)*min(1,(zray1-z01)/abs(zray1-
z01))*(acos(sum([1,0].*[R1/abs(R1)*(xray1-x01),R1/abs(R1)*(zray1-
z01)])/(abs(R1))))+(zray1-ray(size(ray,1),1))/abs(zray1 -
ray(size(ray,1),1))*pi+iout1; %angle of the beam to the horizontal
%                     theta1=alphan1+(zray1-ray(size(ray,1),1))/abs(zray1-
ray(size(ray,1),1))*pi+iout1; %angle of the beam to the horizontal
%                 end
                %theta1=-(alphan1+iout1)/abs(alphan1+iout1)*(pi-abs(alphan1+iout1));
                theta1=-pi+alphan1+iout1;


ray=[ray;zray1,xray1,theta1,iin1,iout1,index1,1,beta,ray(size(ray,1),9)+ray(size(ray,
1),6)*sqrt((zray1-ray(size(ray,1),2))^2+(xray1-
ray(size(ray,1),1))^2),ray(size(ray,1),10)];

            elseif ray(size(ray,1),3)-(1/2-nbmirror/2)*pi()==alphan1 %if the beam
arrives parallel to the normal to the surface
                iin1=0;
                iout1=0;
                theta1=-pi+alphan1+iout1;%(1/2+nbmirror/2)*alphan1;

ray=[ray;zray1,xray1,theta1,iin1,iout1,index1,1,beta,ray(size(ray,1),9)+ray(size(ray,
1),6)*sqrt((zray1-ray(size(ray,1),2))^2+(xray1-
ray(size(ray,1),1))^2),ray(size(ray,1),10)];

            else %for every other direction of the beam
                %iin1=R1/abs(R1)*min(1,(pi()+ray(size(ray,1),3)-
alphan1)/abs(ray(size(ray,1),3)-alphan1))*acos(sum([ray(size(ray,1),2)-
xray1,ray(size(ray,1),1)-zray1].*[R1/abs(R1)*(xray1-x01),R1/abs(R1)*(zray1-
z01)])/(abs(R1)*sqrt((xray1-ray(size(ray,1),2))^2+(zray1-ray(size(ray,1),1))^2)));
%angle of the incident beam to the surface normal
                %iin1=min(1,(zray1-z01)/abs(zray1-z01)*(acos(cos(-pi()+alphan1))-
acos(cos(ray(size(ray,1),3)))))/abs(acos(cos(-pi()+alphan1))-
acos(cos(ray(size(ray,1),3)))))*acos(sum([ray(size(ray,1),2)-
xray1,ray(size(ray,1),1)-zray1].*[R1/abs(R1)*(xray1-x01),R1/abs(R1)*(zray1-
z01)])/(abs(R1)*sqrt((xray1-ray(size(ray,1),2))^2+(zray1-ray(size(ray,1),1))^2)));
%angle of the incident beam to the surface normal
```

```
                iin1=min(1,nbmirror*(R1/abs(R1)*((zray1-z01)/(xray1-x01)-
(ray(size(ray,1),1)-zray1)/(ray(size(ray,1),2)-xray1))/abs(R1/abs(R1)*((zray1-
z01)/(xray1-x01)-(ray(size(ray,1),1)-zray1)/(ray(size(ray,1),2)-
xray1)))))*acos(sum([ray(size(ray,1),2)-xray1,ray(size(ray,1),1)-
zray1].*[R1/abs(R1)*(xray1-x01),R1/abs(R1)*(zray1-z01)])/(abs(R1)*sqrt((xray1-
ray(size(ray,1),2))^2+(zray1-ray(size(ray,1),1))^2))); %angle of the incident beam to
the surface normal
                iout1=asin(nbefore/index1*sin(iin1)); %angle of the exiting beam to
the surface normal
                %theta1=R1/abs(R1)*min(1,(zray1-z01)/abs(zray1-
z01))*(acos(sum([1,0].*[R1/abs(R1)*(xray1-x01),R1/abs(R1)*(zray1-
z01)])/(abs(R1))))+(zray1-ray(size(ray,1),1))/abs(zray1 -
ray(size(ray,1),1))*pi+iout1; %angle of the beam to the horizontal
                %theta1=min(1,-(zray1-z01)/abs(zray1-
z01))*(acos(sum([1,0].*[R1/abs(R1)*(xray1-x01),R1/abs(R1)*(zray1-z01)])/(abs(R1))))-
pi()+iout1; %angle of the beam to the horizontal
%                 if zray1==ray(size(ray,1),1)
%                     %theta1=R1/abs(R1)*min(1,(zray1-z01)/abs(zray1-
z01))*(acos(sum([1,0].*[R1/abs(R1)*(xray1-x01),R1/abs(R1)*(zray1-
z01)])/(abs(R1))))+pi+iout1; %angle of the beam to the horizontal
%                     theta1=alphan1+pi+iout1; %angle of the beam to the horizontal
%                 else
%                     %theta1=R1/abs(R1)*min(1,(zray1-z01)/abs(zray1-
z01))*(acos(sum([1,0].*[R1/abs(R1)*(xray1-x01),R1/abs(R1)*(zray1-
z01)])/(abs(R1))))+(zray1-ray(size(ray,1),1))/abs(zray1 -
ray(size(ray,1),1))*pi+iout1; %angle of the beam to the horizontal
%                     theta1=alphan1+(zray1-ray(size(ray,1),1))/abs(zray1-
ray(size(ray,1),1))*pi+iout1; %angle of the beam to the horizontal
%                 end
                %theta1=-(alphan1+iout1)/abs(alphan1+iout1)*(pi-abs(alphan1+iout1));
                theta1=-pi+alphan1+iout1;




%
%                 iin1=min(1,(ray(size(ray,1),3)-alphan1)/abs(ray(size(ray,1),3)-
alphan1))*acos(sum([ray(size(ray,1),2)-xray1,ray(size(ray,1),1)-
zray1].*[R1/abs(R1)*(xray1-x01),R1/abs(R1)*(zray1-z01)])/(abs(R1)*sqrt((xray1-
ray(size(ray,1),2))^2+(zray1-ray(size(ray,1),1))^2))); %angle of the incident beam to
the surface normal
%                 iout1=asin(nbefore/index1*sin(iin1)); %angle of the exiting beam to
the surface normal
%                 %iout1=asin(nbefore/index1*sin(iin1))+pi()/2*(1-
index1/abs(index1)); %angle of the exiting beam to the surface normal
%                 %theta1=ray(size(ray,1),3)+pi()-2*iin1; %angle of the beam to the
horizontal
%                 %theta1=ray(size(ray,1),3)+pi()+alphan1-iin1; %angle of the beam to
the horizontal
%                 %theta1=min(1,(zray1-z0)/abs(zray1-
z0))*(acos(sum([1,0].*[R1/abs(R1)*(xray1-x01),R1/abs(R1)*(zray1-
z0)])/(abs(R1)))+pi())-iin1+iout1; %angle of the beam to the horizontal
%                 theta1=-((zray1-z01)/abs(zray1-z01))*(acos(sum([1,0].*[(xray1-
x01),(zray1-z01)])/(abs(R1)))+iout1); %angle of the beam to the horizontal
```

```
ray=[ray;zray1,xray1,theta1,iin1,iout1,index1,1,beta,ray(size(ray,1),9)+ray(size(ray,
1),6)*sqrt((zray1-ray(size(ray,1),2))^2+(xray1-
ray(size(ray,1),1))^2),ray(size(ray,1),10)];
            end

            [xsol2,zsol2]=C(x02,z02,R2,ray(size(ray,1),:),nbmirror);
            xray2=xsol2;
            zray2=zsol2;
            if zray2==z02
                alphan2=0;
            else
                alphan2=R2/abs(R2)*min(1,(zray2-z02)/abs(zray2-
z02))*acos(sum([1,0].*[R2/abs(R2)*(xray2-x02),R2/abs(R2)*(zray2-z02)])/(abs(R2)));%-
pi()+atan((zray2-z02)/(xray2-x02)); %slope of the normal to the surface
            end

            if round(ray(size(ray,1),2),10)==round(xray2,10) &&
round(ray(size(ray,1),1),10)==round(zray2,10)
                %iin2=-pi()+ray(size(ray,1),3)-alphan1;
                iin2=ray(size(ray,1),5);
                if abs(index1/nair*sin(iin2))<=1
                    iout2=asin(index1/nair*sin(iin2)); %angle of the exiting beam to
the surface normal
                    %theta2=R2/abs(R2)*min(1,ray(size(ray,1)-
1,3)/abs(ray(size(ray,1)-1,3)))*(acos(sum([1,0].*[R2/abs(R2)*(xray2-
x02),R2/abs(R2)*(zray2-z02)])/(abs(R2))))+(zray2-ray(size(ray,1),1))/abs(zray2 -
ray(size(ray,1),1))*pi+iout2; %angle of the beam to the horizontal
                    %theta2=min(1,ray(size(ray,1)-1,3)/abs(ray(size(ray,1)-
1,3)))*(acos(sum([1,0].*[R2/abs(R2)*(xray2-x02),R2/abs(R2)*(zray2-z02)])/(abs(R2))))-
pi()+iout2; %angle of the beam to the horizontal
                    %theta2=min(1,-(zray2-z02)/abs(zray2-
z02))*(acos(sum([1,0].*[R2/abs(R2)*(xray2-x02),R2/abs(R2)*(zray2-z02)])/(abs(R2))))-
pi()+iout2; %angle of the beam to the horizontal
%                     if zray2==ray(size(ray,1),1)
%                         %theta2=R2/abs(R2)*min(1,(zray2-z02)/abs(zray2-
z02))*acos(sum([1,0].*[R2/abs(R2)*(xray2-x02),R2/abs(R2)*(zray2-
z02)])/(abs(R2)))+pi+iout2; %alphan2+iout2;%angle of the beam to the horizontal
%                         theta2=alphan2+pi+iout2; %alphan2+iout2;%angle of the beam
to the horizontal
%                     else
%                         %theta2=R2/abs(R2)*min(1,(zray2-z02)/abs(zray2-
z02))*acos(sum([1,0].*[R2/abs(R2)*(xray2-x02),R2/abs(R2)*(zray2-
z02)])/(abs(R2)))+(zray2-ray(size(ray,1),1))/abs(zray2-ray(size(ray,1),1))*pi+iout2;
%alphan2+iout2;%angle of the beam to the horizontal
%                         theta2=alphan2+(zray2-ray(size(ray,1),1))/abs(zray2-
ray(size(ray,1),1))*pi+iout2; %alphan2+iout2;%angle of the beam to the horizontal
%                     end
                    %theta2=-(alphan2+iout2)/abs(alphan2+iout2)*(pi-
abs(alphan2+iout2));
                    theta2=-pi+alphan2+iout2;
```

```
                    notblocked=1;
                else
                    iout2=0;
                    theta2=0; %angle of the beam to the horizontal
                    notblocked=0;
                end

ray=[ray;zray2,xray2,theta2,iin2,iout2,nair,notblocked,beta,ray(size(ray,1),9)+ray(si
ze(ray,1),6)*sqrt((zray2-ray(size(ray,1),2))^2+(xray2-
ray(size(ray,1),1))^2),ray(size(ray,1),10)];


            elseif ray(size(ray,1),3)-(1/2-nbmirror/2)*pi()==alphan2
                iin2=0;
                iout2=0;
                theta2=-pi+alphan2+iout2;

ray=[ray;zray2,xray2,theta2,iin2,iout2,nair,1,beta,ray(size(ray,1),9)+ray(size(ray,1)
,6)*sqrt((zray2-ray(size(ray,1),2))^2+(xray2-
ray(size(ray,1),1))^2),ray(size(ray,1),10)];

            else
                %iin2=R2/abs(R2)*min(1,(pi()+ray(size(ray,1),3)-alphan2)/abs(pi()-
ray(size(ray,1),3)-alphan2))*acos(sum([(ray(size(ray,1),2)-
xray2),(ray(size(ray,1),1)-zray2)].*[R2/abs(R2)*(xray2-x02),R2/abs(R2)*(zray2-
z02)])/(abs(R2)*sqrt((xray2-ray(size(ray,1),2))^2+(zray2-ray(size(ray,1),1))^2)));
%angle of the incident beam to the surface normal
                %iin2=min(1,(zray2-z02)/abs(zray2-z02)*(acos(cos(-pi()+alphan2))-
acos(cos(ray(size(ray,1),3))))/abs(acos(cos(-pi()+alphan2))-
acos(cos(ray(size(ray,1),3)))))*acos(sum([(ray(size(ray,1),2)-
xray2),(ray(size(ray,1),1)-zray2)].*[R2/abs(R2)*(xray2-x02),R2/abs(R2)*(zray2-
z02)])/(abs(R2)*sqrt((xray2-ray(size(ray,1),2))^2+(zray2-ray(size(ray,1),1))^2)));
%angle of the incident beam to the surface normal
                iin2=min(1,nbmirror*(R2/abs(R2)*((zray2-z02)/(xray2-x02)-
(ray(size(ray,1),1)-zray2)/(ray(size(ray,1),2)-xray2))/abs(R2/abs(R2)*((zray2-
z02)/(xray2-x02)-(ray(size(ray,1),1)-zray2)/(ray(size(ray,1),2)-
xray2)))))*acos(sum([(ray(size(ray,1),2)-xray2),(ray(size(ray,1),1)-
zray2)].*[R2/abs(R2)*(xray2-x02),R2/abs(R2)*(zray2-z02)])/(abs(R2)*sqrt((xray2-
ray(size(ray,1),2))^2+(zray2-ray(size(ray,1),1))^2))); %angle of the incident beam to
the surface normal

                if abs(index1/nair*sin(iin2))<=1
                    iout2=asin(index1/nair*sin(iin2)); %angle of the exiting beam to
the surface normal
                    %theta2=min(1,(zray2-z02)/abs(zray2-
z02))*acos(sum([1,0].*[R2/abs(R2)*(xray2-x02),R2/abs(R2)*(zray2-z02)])/(abs(R2)))-
pi()+iout2; %alphan2+iout2;%angle of the beam to the horizontal
                    %theta2=acos(sum([1,0].*[R2/abs(R2)*(xray2-
x02),R2/abs(R2)*(zray2-z02)])/(abs(R2)))-pi()+iout2; %alphan2+iout2;%angle of the
beam to the horizontal
%                     if zray2==ray(size(ray,1),1)
```

```
%                        %theta2=R2/abs(R2)*min(1,(zray2-z02)/abs(zray2-
z02))*acos(sum([1,0].*[R2/abs(R2)*(xray2-x02),R2/abs(R2)*(zray2-
z02)])/(abs(R2)))+pi+iout2; %alphan2+iout2;%angle of the beam to the horizontal
%                        theta2=alphan2+pi+iout2; %alphan2+iout2;%angle of the beam
to the horizontal
%                    else
%                        %theta2=R2/abs(R2)*min(1,(zray2-z02)/abs(zray2-
z02))*acos(sum([1,0].*[R2/abs(R2)*(xray2-x02),R2/abs(R2)*(zray2-
z02)])/(abs(R2)))+(zray2-ray(size(ray,1),1))/abs(zray2-ray(size(ray,1),1))*pi+iout2;
%alphan2+iout2;%angle of the beam to the horizontal
%                        theta2=alphan2+(zray2-ray(size(ray,1),1))/abs(zray2-
ray(size(ray,1),1))*pi+iout2; %alphan2+iout2;%angle of the beam to the horizontal
%                    end
                    %theta2=-(alphan2+iout2)/abs(alphan2+iout2)*(pi-
abs(alphan2+iout2));
                    theta2=-pi+alphan2+iout2;




%                 %iin2=min(1,(ray(size(ray,1),3)-alphan1)/abs(ray(size(ray,1),3)-
alphan1))*acos(sum([ray(size(ray,1),2)-xray2,ray(size(ray,1),1)-
zray2].*[R1/abs(R1)*(xray2-x02),R1/abs(R1)*(zray2-z02)])/(abs(R1)*sqrt((xray2-
ray(size(ray,1),2))^2+(zray2-ray(size(ray,1),1))^2))); %angle of the incident beam to
the surface normal
%                 iout2=asin(index1/nafter*sin(iin2)); %angle of the exiting beam to
the surface normal
%                 %iout1=asin(nbefore/index1*sin(iin1))+pi()/2*(1-
index1/abs(index1)); %angle of the exiting beam to the surface normal
%                 %theta1=ray(size(ray,1),3)+pi()-2*iin1; %angle of the beam to the
horizontal
%                 %theta1=ray(size(ray,1),3)+pi()+alphan1-iin1; %angle of the beam to
the horizontal
%                 %theta1=min(1,(zray1-z0)/abs(zray1-
z0))*(acos(sum([1,0].*[R1/abs(R1)*(xray1-x01),R1/abs(R1)*(zray1-
z0)])/(abs(R1)))+pi())-iin1+iout1; %angle of the beam to the horizontal
%                 theta2=-((zray2-z02)/abs(zray2-z02))*(acos(sum([1,0].*[(xray2-
x02),(zray2-z02)])/(abs(R2)))+iout2); %angle of the beam to the horizontal



                    notblocked=1;
                else
                    iout2=0;
                    theta2=0;
                    notblocked=0;
                end

ray=[ray;zray2,xray2,theta2,iin2,iout2,nair,notblocked,beta,ray(size(ray,1),9)+ray(si
ze(ray,1),6)*sqrt((zray2-ray(size(ray,1),2))^2+(xray2-
ray(size(ray,1),1))^2),ray(size(ray,1),10)];
            end

            %ray=[ray;ray(size(ray,1),:)];
```

```
            cross(i)=(zpositionLensVar-
ray(size(ray,1),1))/tan(ray(size(ray,1),3))+ray(size(ray,1),2);
            FL(i)=(zpositionLensVar-
ray(size(ray,1),1))/tan(ray(size(ray,1),3))+ray(size(ray,1),2)-x1;

        end

%         if ray(size(ray,1),7)==0
%             ray=[ray;ray(size(ray,1),:)];
%         else
%             if ray(size(ray,1),3)==0
%
ray=[ray;ray(size(ray,1),1),ray(size(ray,1),2)+2*Diam,ray(size(ray,1),3),ray(size(ray
,1),5),ray(size(ray,1),5),nair,1,beta,ray(size(ray,1),9)+ray(size(ray,1),6)*sqrt((ray
(size(ray,1),2)+2*Diam-ray(size(ray,1),2))^2+(ray(size(ray,1),1)-
ray(size(ray,1),1))^2)];
%             else
%                 %znew=ray(size(ray,1),1)+sin(ray(size(ray,1),3))*(lengthafter);
%                 %xnew=ray(size(ray,1),2)+cos(ray(size(ray,1),3))*lengthafter;
%
ray=[ray;ray(size(ray,1),1),ray(size(ray,1),2),ray(size(ray,1),3),ray(size(ray,1),5),
ray(size(ray,1),5),nair,1,beta,ray(size(ray,1),9)];
%             end
%         end
%
%         beta=beta+index1/abs(index1)*Lens(i,7);
        ray=[ray;ray(size(ray,1),:)];

    end

    lengthafter=xpositionScreen; %FL(2)*1.01;
    if ray(size(ray,1),7)==0
        ray=[ray;ray(size(ray,1),:)];
    else
        if ray(size(ray,1),3)==0

ray=[ray;ray(size(ray,1),1),ray(size(ray,1),2)+2*Diam,ray(size(ray,1),3),0,0,nair,1,b
eta,ray(size(ray,1),9)+ray(size(ray,1),6)*sqrt((ray(size(ray,1),2)+2*Diam-
ray(size(ray,1),2))^2+(ray(size(ray,1),1)-
ray(size(ray,1),1))^2),ray(size(ray,1),10)];
        else
            znew=ray(size(ray,1),1)+sin((ray(size(ray,1),3)))*lengthafter;
            xnew=ray(size(ray,1),2)+cos((ray(size(ray,1),3)))*lengthafter;

ray=[ray;znew,xnew,ray(size(ray,1),3),0,0,nair,1,beta,ray(size(ray,1),9)+ray(size(ray
,1),6)*sqrt((xnew-ray(size(ray,1),2))^2+(znew-
ray(size(ray,1),1))^2),ray(size(ray,1),10)];
        end
    end

        beta=beta+index1/abs(index1)*Lens(i,8);


end
```

```
end
```