```
%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% %%
%% -------------------------- MORFEO HC v. 1.0 --------------------------- %%
%% ---- Monte Carlo Simulation Ramirez Fons for Half Cell ---------------- %%
%% ---------Diffusion Experiment Evaluation and Optimization ------------- %%
%% ----------------------------------------------------------------------- %%
%% PartI: Determination of minimum evolved region for Half-Cell experiments%%
%% ----------------------- Creative Commons cc by ------------------------ %%
%% --------------------- Jordi Fons & Oriol Ramirez ---------------------- %%
%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% %%


%% DESCTRIPTION AND GUIDE FOR USERS

% This script simulates non evolved half-cell diffusion experiments and fit
% the simulated data to an error function to obtain diffusion profiles in
% order to evaluate the minimum evolved region necessary to ensure the
% contribution of migration in the evolved region observed. The key
% parameters considered are the variability in the activity concentration of
% both plugs, the non-infinitesimal sliced of the plug (due to the particle
% size of the sample analysed) and its variability.

% It is recommended just to modify the input data to adjust the simulation
% to the specific experimental conditions. Any modification of the simulation
% section may lead to a loss of functionality of the script.

clear
close

%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% %%
%-------------------------------------------------------------------------%
repeticionsacom=50; % Number of non-evolved diffusion tubes simulated
nacom=8000; % Number of point used to simulate one diffusion tube
ndacom=80; % Length of the diffusion tube in mm
ACBacom=95; % Activity concentration of the low activity plug
ACAacom=4800; % Activity concentration of the high activity plug
sAacom=0.1; % Relative Standard Deviation (RSD) of low and high activity
%concentration
cmmacom=0.0000000001; % evolved region in mm, 0.0000000001 for non-evolved
sacom=2; % mean distance between slices in mm
rsdsacom=0.3; % RSD in the distance between slices
numacom=10; % Number of slices in the plateaus (plug ends) used to calculate
%the mean value for each plateau
%-------------------------------------------------------------------------%

%% %%%%%%%%%%%%%%%%%%%%%%%%%%%% SIMULATION %%%%%%%%%%%%%%%%%%%%%%%%%%%% %
% -- Following section should not be modified unless users---------------%
%----want to adapt the script for simulating diffusion experiments-------%
%-----------------of set-ups other than half-cell------------------------%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

repacom=1;
while repacom<repeticionsacom+1;
```

```
clearvars  -except -regexp acom$; %Delete all the variables except the ones
% needed to save the results.

% Variable definition
n=nacom;
nd=ndacom;
ACB=ACBacom;
sA=sAacom;
ACA=ACAacom;
cmm=cmmacom;
c=(n/nd)*cmm;
pmm=n/nd;
s=sacom;
rsds=rsdsacom;
sds=s*rsds;
mpps=pmm*s;
sdpps=pmm*rsds;
num=numacom;

%% SIMULATION OF THE DIFFUSION TUBE
%% Generate a sigmoid curve from -n/2 to n/2 with the input evolved region(c)
    distp=[(-1*n/2):1:(n/2)]';
    actn=erf((distp)/(c));
    AM=(ACA+ACB)/2; % Mean activity in the tube
    ci=((ACA-AM)*actn)+AM;
%% Include the input dispersion in the generated points to simulate the
%% dispersion between slices
    si=ci*sA*sqrt((n)/((n/80)/sA)); %SD as a function of the activity
    aleat=randn(1,n+1); %Generates n+1 random numbers following a normal
    %distribution with a mean value of 0 and standard deviation of 1
    i=1;
    while i<length(si)+1
        cdi(i)=ci(i)+si(i).*aleat(i);
        i=i+1;
    end

%% Slices Generator (generates randomly slices with width "s" (in mm)
%  and a RSD of "rsds"

    slicepunts=0;
    i=1;
    while sum(slicepunts)<n;
        slicepunts(i)=round(mpps+sdpps.*randn(1,1));
        i=i+1;
    end
    uslice=sum(slicepunts)-n;
    % Modify the width of the last slices to ensure that all simulated tubes
    % have a 80mm-length
    if uslice > mpps/2;
        slicepunts(i-2)=nacom-sum(slicepunts(1:i-3));
        numslices=i-2;
    else
        slicepunts(i-1)=nacom-sum(slicepunts(1:i-2));
        numslices=i-1;
    end
    slicep=slicepunts(1:numslices);
%% Define the vector with the distance between slices in mm
    i=1;
    j=0;
    distac(1)=slicep(1);
```

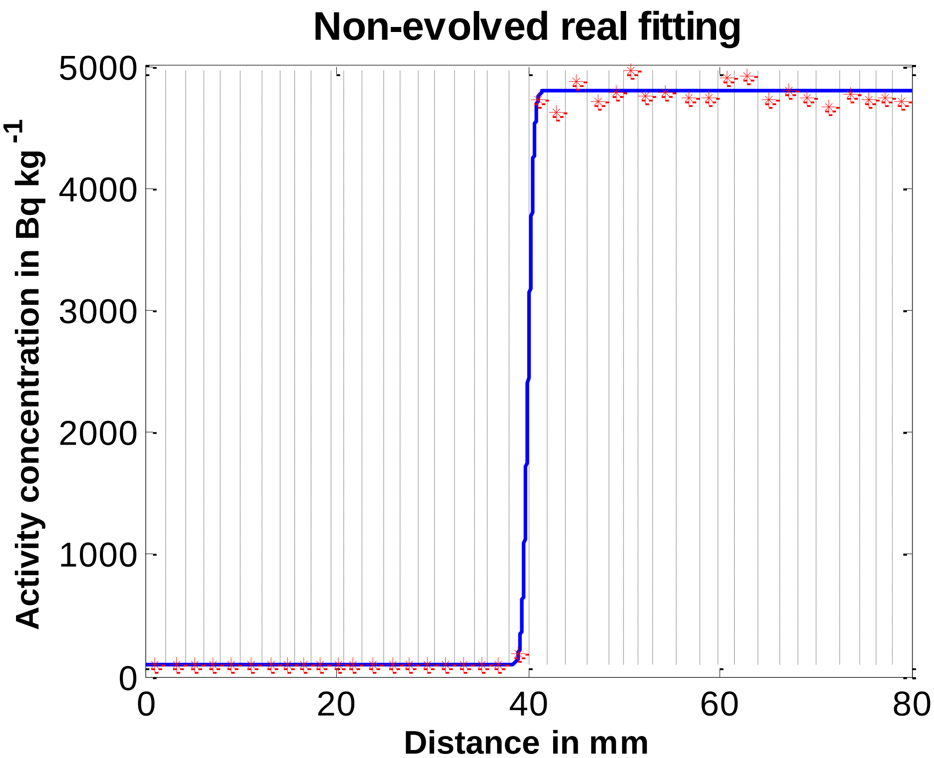
```
    while i<numslices+1;
        dist(i)=(2*j+slicep(i))/2;
        distac(i+1)=distac(i)+slicep(i);
        j=j+slicep(i);
        i=i+1;
    end
    dist=dist';
    distac=distac-distac(1);
    distac=distac';
%% Calculate the activity concnetration for each slice
    act=zeros((numslices),1);
    k=1; %Each slice
    l=1; %Each point-activity inside the vector ci
    while k<numslices+1 %For each slice
        o=1; % number of point inside the slice
        while o<(slicep(k))+1; % For all the points in the slice
        act(k)=act(k)+cdi(l);
        o=o+1;
        l=l+1;
        end
        k=k+1;
    end
%% Calculate the concentration activity for each slice
    k=1; %Each slice
    while k<numslices+1;
        act(k)=act(k)/slicep(k);
        k=k+1;
    end

%% DIFFUSION TUBE SIMULATED
%% Output data:

            % % % % % % % % % % % % % % % % % % % % % % % %
distac;     % distac = End of each slice in points        %
dist;       % dist = Centre of each slice in points       %
act;        % act = Activity concentration of each slice  %
            % % % % % % % % % % % % % % % % % % % % % % % %


%% NORMALIZATION OF THE SIMULATED TUBE
%  Activity
    mbaix=mean(act(1:num));
    malt=mean(act(end-num:end));
    mitja=(mbaix+malt)/2;
    actn=(act-mitja)/(malt-(mitja));
    % Warning!! in some Matlab versions it can not work due to "act" is a
    % vector and "mitja" and "malt" are scalars. It can be modified to:
    % actn(:,1)=(act(:,1)-mitja)/(malt-(mitja))
 %  Position
    distc=(dist-(n/2))*nd/n; %Recalculate the position considering 0 the
    % contact surface between teo plugs

%% DIFFUSION TUBE NORMALIZED
%% Output data:

            % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
actn;       %          act = Normalized activity for each slice          %
distc;      %    distc= Centre of each slices normalized to 0 in mm      %
            % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
```

```
%% FITTING OF THE SIMULATED DIFFUSION PROFILE

    opcions = fitoptions('Method','NonlinearLeastSquares',...
                   'Lower',0,...
                   'Upper',Inf,...
                   'StartPoint',1,...
                   'Robust','Bisquare');
        f = fittype('erf(x/c)','coefficients','c',...
                   'independent','x','options',opcions);
    funcio = fit(distc,actn,f);
    c=funcio.c;
    residuals=actn-funcio(distc); % Residuals calculation

%% FITTED FUNCTION
%% Output data:

             % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
funcio;      %          funcio= function activity vs position          %
c;           %                  c= fitted value of "c"                 %
residuals;   %      residuals= residuals vector for each slice         %
             % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %

%% SAVE THE RESULTS

Cacom(repacom)=c;
repacom=repacom+1;
end

%% %%%%%%%%%%%%%%%%%%%% END OF THE SIMULATION %%%%%%%%%%%%%%%%%%%% %%

%-----------------------------------------------------------------------%

%% %%%%%%%%%%%%%%%%%%%%%%%%%% RESULTS %%%%%%%%%%%%%%%%%%%%%%%%%% %%

Cacom;
Mean=mean(Cacom)% Mean of "C" determined by fitting of  all the simulated
% tubes with non evolved profiles
Sort=sort(Cacom)% 95%
Percentil95=Sort(round(repeticionsacom*0.95))


%% %%%%%%%%%%%%%%%% REPRESENTATION OF THE RESULTS %%%%%%%%%%%%%%%% %%

% These representations are just performed for the last simulated
% tube in order to make more understandable the simulation and emphasize
% the value of the results.

%% Plot 1
%  Slices, with its activity and fitted function
figure(1)
axes('FontSize',14)
plot(distp/100+40, actfitnorm,'b-','LineWidth',2)
hold on
plot (distc+40,act,'r*')
title('Non-evolved real fitting','Fontsize',16,'FontWeight','bold')
xlabel('Distance in mm','FontWeight','bold','Fontsize',14)
ylabel('Activity concentration in Bq kg^-^1',...
    'FontWeight','bold','Fontsize',14)
i=1;
 while i<numslices+1
```

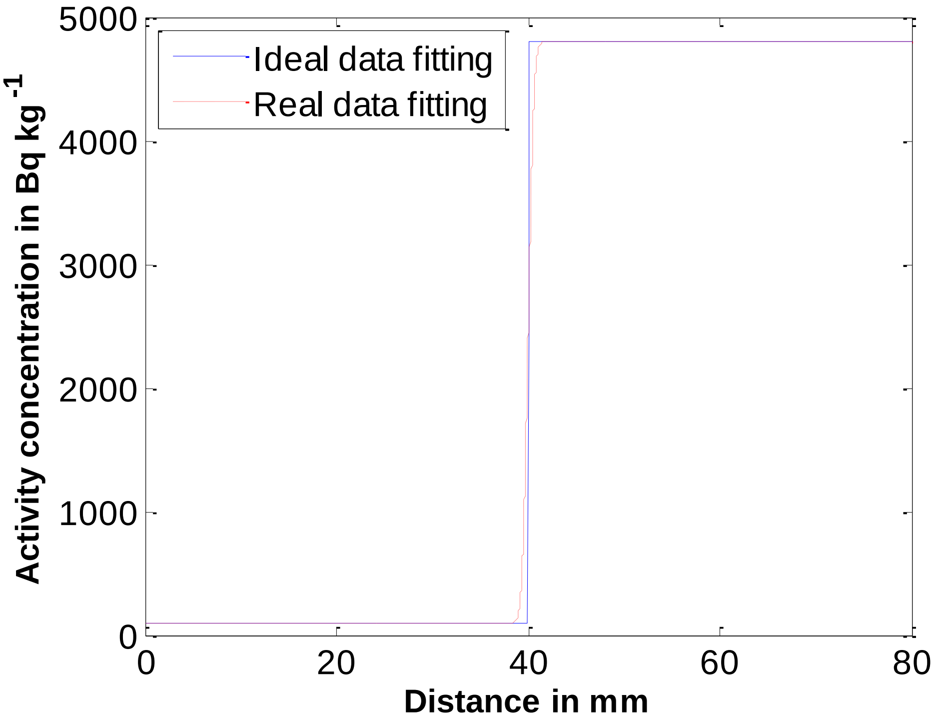
```
     plot ([distac(i)/100;distac(i)/100],[min(act);max(act)],'k:')
     i=i+1;
 end

%% Plot 2
%  Fittings for real and ideal data
figure(2)
axes('FontSize',14)
plot (distp/100+40,((ACA-AM)*erf((distp)/(cmm*100)))+AM,'b-','LineWidth',1)
hold on
plot(distp/100+40, actfitnorm,'r:','LineWidth',1)
xlabel('Distance in mm','FontWeight','bold','Fontsize',14)
ylabel('Activity concentration in Bq kg^-^1',...
    'FontWeight','bold','Fontsize',14)
legend('Ideal data fitting','Real data fitting','location','Northwest')

%  Zoom in the region of interest
figure (21)
axes('FontSize',14)
plot (distp/100+40,((ACA-AM)*erf((distp)/(cmm*100)))+AM,'b-','LineWidth',1)
hold on
plot(distp/100+40, actfitnorm,'r:','LineWidth',1)
xlabel('Distance in mm','FontWeight','bold','Fontsize',14)
ylabel('Activity concentration in Bq kg^-^1',...
    'FontWeight','bold','Fontsize',14)
axis([37.5 42.5 0 5000])
legend('Ideal data fitting','Real data fitting','location','Northwest')

%% Plot 3
%  Slices, with its activity and fitted function together with the
%  residuals for each slice

actfitada=erf(distp/(c*100));
actfitnorm=((ACA-AM)*actfitada)+AM;
figure(3)
subplot(2,1,1)
plot(distp/100+40, actfitnorm,'b-','LineWidth',2)
title('Fitting','Fontsize',12,'FontWeight','bold')
hold on
plot (distc+40,act,'r*')
legend('Fitting','Act. Slices','location','SouthEast')
i=1;
xlabel('Distance in mm','FontWeight','bold','Fontsize',10)
ylabel('Activity concentration in Bq kg^-^1',...
    'FontWeight','bold','Fontsize',10)

while i<numslices+1
    plot ([distac(i)/100;distac(i)/100],[ACA;ACB],'k:')
    i=i+1;
end
subplot(2,1,2)
plot(distc+40,residuals,'r')
title('Residuals','Fontsize',12,'FontWeight','bold')
hold on
plot(distc+40,residuals,'ko','MarkerSize',3)
xlabel('Distance in mm','FontWeight','bold','Fontsize',10)
```

%% IMAGES

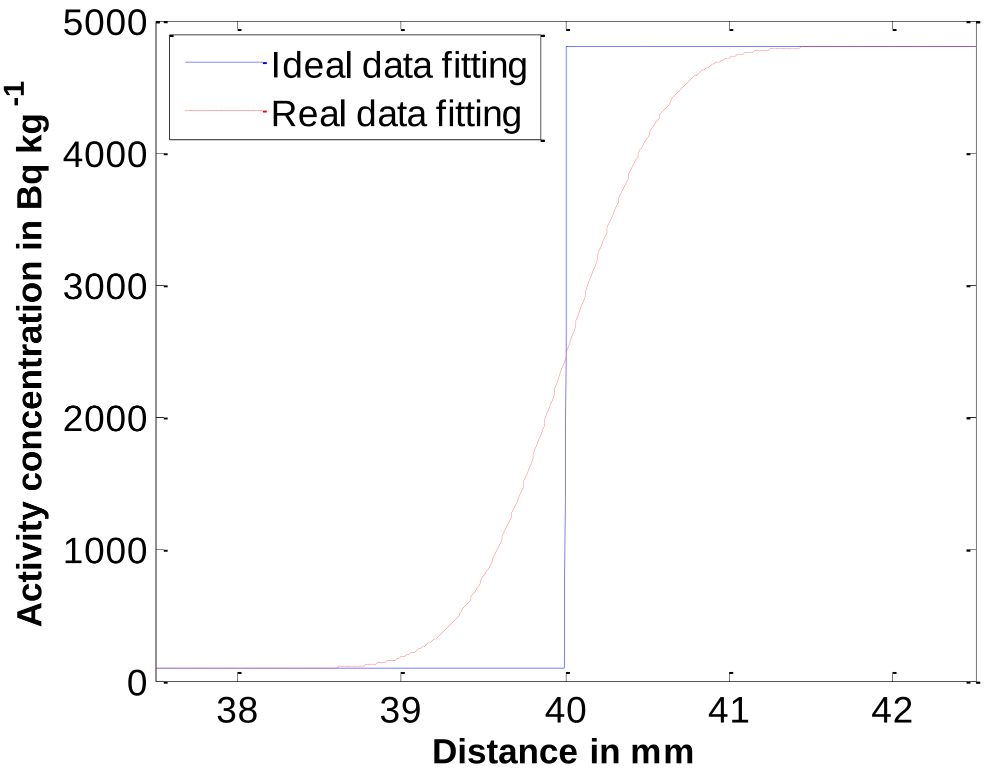
%% Plot 1



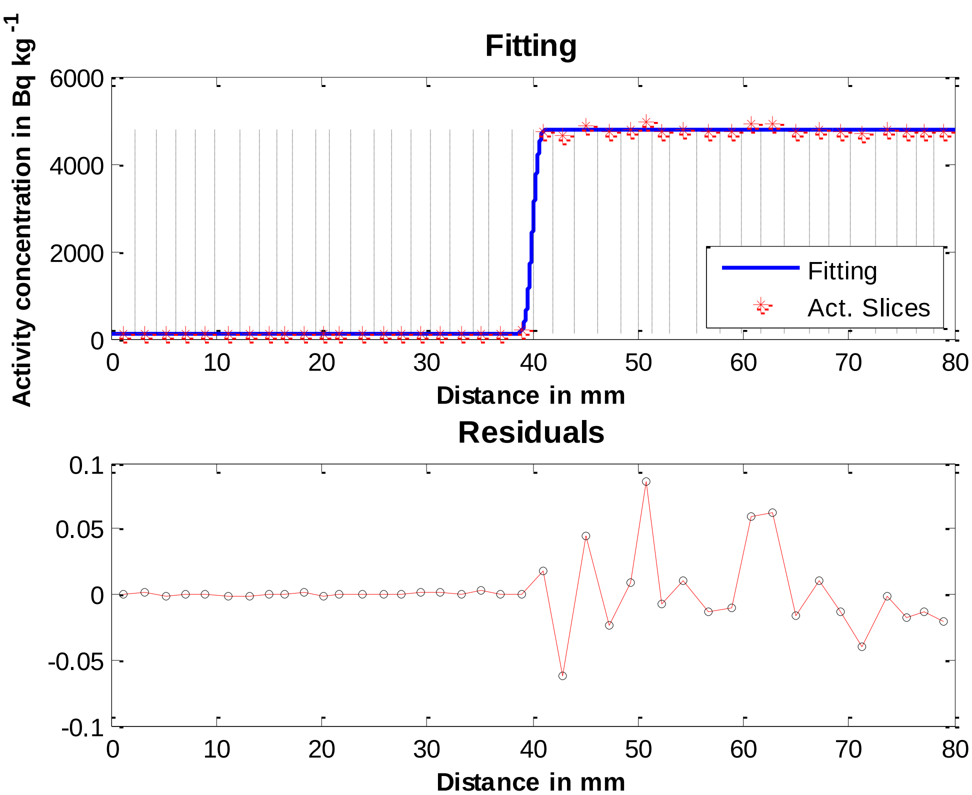
%% Plot 2



```
%% Plot 2 (Zoom)
```




```
%% Plot 3
```



```
%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% %%
%% -------------------------- MORFEO HC v. 1.0 ------------------------- %%
%% ---- Monte Carlo Simulation Ramirez Fons for Half Cell -------------- %%
%% ---------Diffusion Experiment Evaluation and Optimization ----------- %%
%% --------------------------------------------------------------------- %%
%% -- PART II: Determination of correction factor and uncertainty --- %%
%% --------------------- in Half-Cell experiments --------------------- %%
%% ---------------------- Creative Commons cc by --------------------- %%
%% ------------------- Jordi Fons & Oriol Ramirez ------------------- %%
%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% %%


%% DESCTRIPTION AND GUIDE FOR USERS %%

% This script simulates evolved half-cell diffusion experiments and fit
% the simulated data to an error function to obtain diffusion profiles in
% order to evaluate the bias between fitted real profile and the real
% evolution of the diffusion tube. The parameters considered are the
% variability in the activity concentration of both plugs, the
% non-infinitesimal sliced of the plug (due to the particle size of the
% sample analyzed) and its variability.
% In this script, evolved diffusion profiles from 1 mm to 40 mm by
% progressing 1 mm at time, were simulated (200 profiles for each grade
% of evolution). This profiles, with a known evolution (ideal profiles),
% were treated in the same way that experimental profiles obtaining a
% fitted evolution (real profiles). In this way, the correction factor to
% correct fitted data into the evolution of the diffusion tube and its
% uncertainty were obtained.

% It is recommended just to modify the input data to adjust the
% simulation to the specific experimental conditions. Any modification of
% the simulation section may lead to a loss of functionality of the
% script.

% Input data in this script is divided in several sections for
% operational reasons.


clear
close
tic

%% %%%%%%%%%%%%%%%%%%%%%%%%%%% INPUT DATA %%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
%-----------------------------------------------------------------------
%
cmmxacom=1;  % Minimum value of c simulated in mm
cmaxacom=40; % Maximum value of c simulated in mm
repeticionsxacom=200; % Number of diffusion tubes simulated for each c
%-----------------------------------------------------------------------%

calculsxacom=0;% Accountant
while cmmxacom<=cmaxacom % For each c
clearvars  -except -regexp xacom$; % Delete variables from previous
%simulations of different c
```

```
repacom=1;
while repacom<repeticionsxacom+1; % For each tube

clearvars  -except -regexp acom$; %% Delete variables from previous
%simulations of different tube

%% %%%%%%%%%%%%%%%%%%%%%%%%%%% INPUT DATA %%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
%-------------------------------------------------------------------------
%
n=8000; % Number of point used to simulate one diffusion tube
nd=80; % Length of the diffusion tube in mm
ACB=95; % Activity of the low activity plug
sA=0.05; % Relative Standard Deviation of ***
ACA=4800; % Activity of the high activity plug
s=2; % mean distance between slices in mm
rsds=0.3; % RSD in the distance between slices
num=10; % Number of slices in the plateaus used to calculate the mean
% value for each plateau

%-------------------------------------------------------------------------
%

% Variable definition
cmmi=0.1;

cmm=cmmi+(cmmxacom-1)/10; %value of c in mm
c=(n/nd)*cmm; %value of c in points
pmm=n/nd; %points per mm
sds=s*rsds;
mpps=pmm*s;
sdpps=pmm*rsds;



%% SIMULATION OF THE DIFFUSION TUBE
%% Generate a sigmoid curve from -n/2 to n/2 with the input evolved
region(c)
    distp=[(-1*n/2):1:(n/2)]';
    actn=erf((distp)/(c));
    AM=(ACA+ACB)/2; % Mean activity in the tube
    ci=((ACA-AM)*actn)+AM;
%% Include the input dispersion in the generated points to simulate the
%% dispersion between slices
    si=ci*sA*sqrt((n)/((n/80)/sA)); %SD as a function of the activity
    aleat=randn(1,n+1); %Generates n+1 random numbers following a normal
    %distribution with a mean value of 0 and standard deviation of 1
    i=1;
    while i<length(si)+1
        cdi(i)=ci(i)+si(i).*aleat(i);
        i=i+1;
    end

%% Slices Generator (generates randomly slices with width "s" (in mm)
%  and a RSD of "rsds"
```

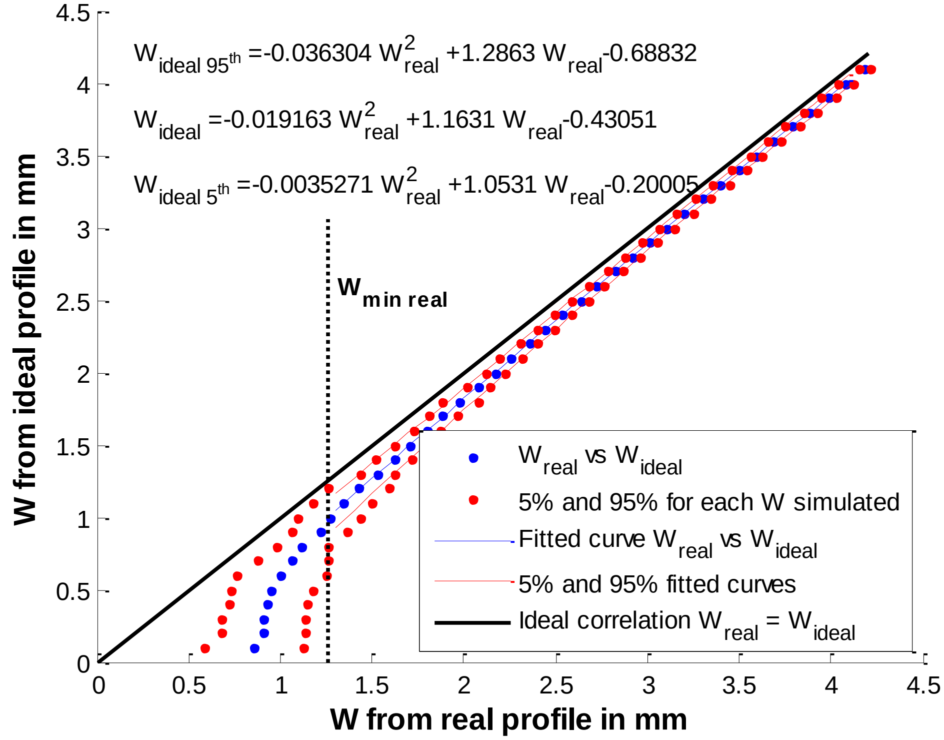
```
    slicepunts=0;
    i=1;
    while sum(slicepunts)<n;
        slicepunts(i)=round(mpps+sdpps.*randn(1,1));
        i=i+1;
    end
    uslice=sum(slicepunts)-n;
    % Modify the width of the last slices to ensure that all simulated tubes
    % have a 80mm-length
    if uslice > mpps/2;
        slicepunts(i-2)=n-sum(slicepunts(1:i-3));
        numslices=i-2;
    else
        slicepunts(i-1)=n-sum(slicepunts(1:i-2));
        numslices=i-1;
    end
    slicep=slicepunts(1:numslices);
%% Define the vector with the distance between slices in mm
    i=1;
    j=0;
    distac(1)=slicep(1);
    while i<numslices+1;
        dist(i)=(2*j+slicep(i))/2;
        distac(i+1)=distac(i)+slicep(i);
        j=j+slicep(i);
        i=i+1;
    end
    dist=dist';
    distac=distac-distac(1);
    distac=distac';
%% Calculate the activity for each slice
    act=zeros((numslices),1);
    k=1; %Each slice
    l=1; %Each point-activity inside the vector ci
    while k<numslices+1 %For each slice
        o=1; % number of point inside the slice
        while o<(slicep(k))+1; % For all the points in the slice
        act(k)=act(k)+cdi(l);
        o=o+1;
        l=l+1;
        end
        k=k+1;
    end
%% Calculate the concentration activity for each slice
    k=1; %Each slice
    while k<numslices+1;
        act(k)=act(k)/slicep(k);
        k=k+1;
    end

%% DIFFUSION TUBE SIMULATED
%% Output data:

            % % % % % % % % % % % % % % % % % % % % % % % %
distac;     % distac = End of each slice in points        %
```

```
dist;         % dist = Centre of each slice in points       %
act;          % act = Activity concentration of each slice  %
              % % % % % % % % % % % % % % % % % % % % % % % %


%% NORMALIZATION OF THE SIMULATED TUBE
%  Activity
    mbaix=mean(act(1:num));
    malt=mean(act(end-num:end));
    mitja=(mbaix+malt)/2;
    actn=(act-mitja)/(malt-(mitja));
    % Warning!! in some Matlab versions it can not work due to "act" is a
    % vector and "mitja" and "malt" are scalars. It can be modified to:
    % actn(:,1)=(act(:,1)-mitja)/(malt-(mitja))
 %  Position
    distc=(dist-(n/2))*nd/n; %Recalculate the position considering 0 the
    % contact surface between two plugs

%% DIFFUSION TUBE NORMALIZED
%% Output data:

              % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
actn;         %          act = Normalized activity for each slice         %
distc;        %    distc= Centre of each slices normalized to 0 in mm     %
              % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %

%% FITTING OF THE SIMULATED DIFFUSION PROFILE
% The fitting options can be changed to fit different purposes

    opcions = fitoptions('Method','NonlinearLeastSquares',...
                  'Lower',0,...
                  'Upper',Inf,...
                  'StartPoint',1,...
                  'Robust','Bisquare');
        f = fittype('erf(x/c)','coefficients','c',...
                  'independent','x','options',opcions);
    funcio = fit(distc,actn,f);
    c=funcio.c;
    residuals=actn-funcio(distc); % Residuals calculation

%% FITTED FUNCTION
%% Output data:

               % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
funcio;        %          funcio= function activity vs position          %
c;             %                  c= fitted value of "c"                 %
residuals;     % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %

%% SAVE THE RESULTS

calculsxacom=calculsxacom+1; % Accountant of total fittings performed
Cacom(repacom)=c;
repacom=repacom+1;
end
Cxacom(cmmxacom)=mean(Cacom); % mean of "real" c for each "ideal" c
```

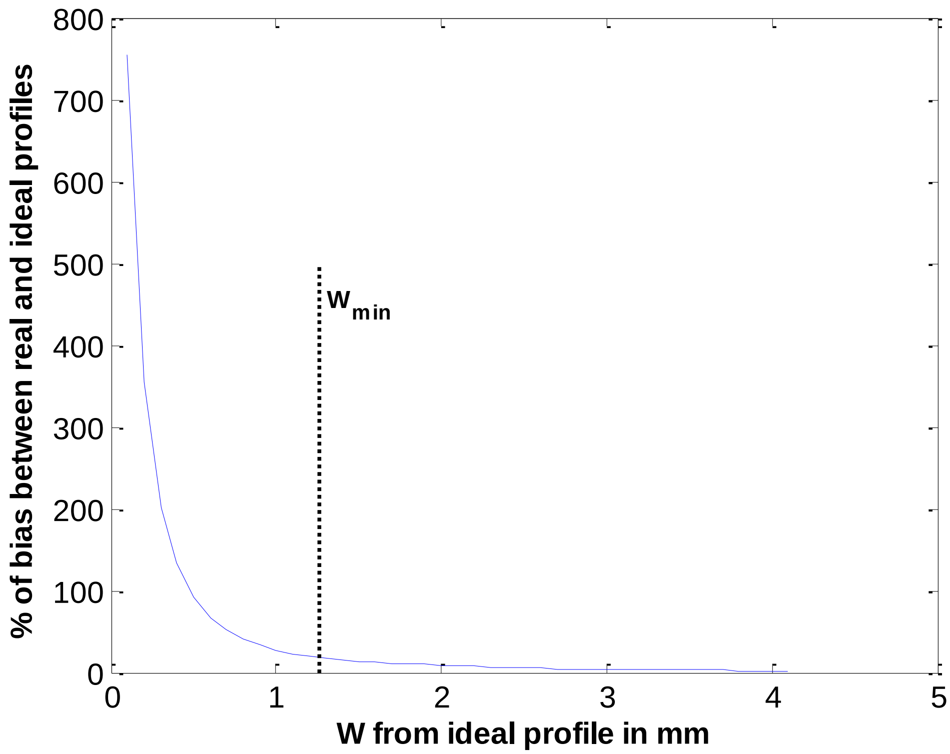
```
Desvxacom(cmmxacom)=std(Cacom); % SD in "real" c for each "ideal" c
cmmxacom=cmmxacom+1
end
toc
%% %%%%%%%%%%%%%%%%%%%%%% END OF THE SIMULATION %%%%%%%%%%%%%%%%%%%%%% %%

%------------------------------------------------------------------------%

%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%% RESULTS %%%%%%%%%%%%%%%%%%%%%%%%%%%%% %%

%% Check results
if calculsxacom==cmaxacom*repeticionsxacom
disp(' OK ')
else
    disp('Warning!! Error in repetitions or on results accumulation')
end
%% Simulated data treatment
Cxacom; % mean of "real" c for each "ideal" c
Desvxacom; % SD in "real" c for each "ideal" c
Desvestacom=(Desvxacom./Cxacom); % RSD in "real" c for each "ideal" c
csimulades=cmmi:0.1:cmmi+(cmaxacom-1)/10; % c simulated
errorc=(Cxacom-csimulades)./csimulades*100; % Bias in C between "real" %
and "ideal"
toc

%% %%%%%%%%%%%%%%%%%% REPRESENTATION OF THE RESULTS %%%%%%%%%%%%%%%%%% %%

%% PLOT 1

MER=1.26 % Minimum Evolved Region (it can be calculated using MORFEO_HC
part I
% The data is fitted from MER to maximum C simulated
% ----------
% Calculations
Cplus=Cxacom+Desvxacom;
Cminus=Cxacom-Desvxacom;
mer=round(MER*10)

opcions = fitoptions('Method','NonlinearLeastSquares',...
                   'Lower',0,...
                   'Upper',Inf,...
                   'StartPoint',1,...
                   'Robust','Bisquare');

funcio = fit(Cxacom(mer:end)',csimulades(mer:end)','poly2');
funciop = fit(Cplus(mer:end)',csimulades(mer:end)','poly2');
funciom = fit(Cminus(mer:end)',csimulades(mer:end)','poly2');


figure(1)
axes('FontSize',10)
hold on
plot(Cxacom,[csimulades],'b.')% Cideal vs real
plot(Cplus,[csimulades],'r.')
plot(csimulades(mer:end),funcio(csimulades(mer:end)),'b-','Linewidth',1)
```

```
%fitted Cideal vs real
plot(csimulades(mer:end),funciop(csimulades(mer:end)),'r-','Linewidth',1)
%fitted 95 % Cideal vs real
plot([0,cmmi+cmaxacom/10],[0,cmmi+cmaxacom/10],'k','Linewidth',1.5)
%Ideal correlation (ideal = real)
plot([MER MER],[0 MER+1.8],'k:','Linewidth',1.5)
plot(Cminus,[csimulades],'r.')
plot(csimulades(mer:end),funciom(csimulades(mer:end)),'r-','Linewidth',1)
%fitted 5 % Cideal vs real
ylabel('W from ideal profile in mm','FontWeight','bold','Fontsize',12)
xlabel('W from real profile in mm','FontWeight','bold','Fontsize',12)
legend('W_{real} vs W_{ideal}','5% and 95% for each W simulated',...
    'Fitted curve W_{real} vs W_{ideal}','5% and 95% fitted curves',...
    'Ideal correlation W_{real} = W_{ideal}','location','Southeast')
text(0.2,3.75,strcat('W_{ideal} = ',num2str(1000*funcio.p1/1000),...
    ' W_{real}^2 + ',num2str(1000*funcio.p2/1000),...
    ' W_{real} ',num2str(1000*funcio.p3/1000)),'Fontsize',10)
text(0.2,4.20,strcat('W_{ideal{ 95^{th}}} = ',...
    num2str(1000*funciop.p1/1000),' W_{real}^2 + ',...
    num2str(1000*funciop.p2/1000),' W_{real} ',...
    num2str(1000*funciop.p3/1000)),'Fontsize',10)
text(0.2,3.30,strcat('W_{ideal{ 5^{th}}} = ',...
    num2str(1000*funciom.p1/1000),' W_{real}^2 + ',...
    num2str(1000*funciom.p2/1000),' W_{real} ',...
    num2str(1000*funciom.p3/1000)),'Fontsize',10)
text(MER+0.05,MER+1.3,strcat('W_{min real}'),'FontWeight','bold',...
    'Fontsize',10)

%% PLOT 2

% Bias between real and ideal evolved profiles
figure (21)
axes('FontSize',12)
plot (csimulades, errorc)
hold on
plot([MER MER],[0 500],'k:','Linewidth',1.5)
xlabel('W from ideal profile in mm','FontWeight','bold','Fontsize',12)
ylabel('% of bias between real and ideal
profiles','FontWeight','bold',...
    'Fontsize',12)
text(MER+0.05,450,strcat('W_{min}'),'FontWeight','bold','Fontsize',10)

% Zoom on values higher than Wmin of previous plot
figure (22)
axes('FontSize',12)
plot (csimulades, errorc)
hold on
plot([MER MER],[0 500],'k:','Linewidth',1.5)
xlabel('W from ideal profile in mm','FontWeight','bold','Fontsize',12)
ylabel('% of bias between real and ideal
profiles','FontWeight','bold',...
    'Fontsize',12)
axis([MER-0.5 cmaxacom/10 0 max(errorc(mer:end))+5])
text(MER+0.05,2,strcat('W_{min}'),'FontWeight','bold','Fontsize',10)
%% IMAGES
```

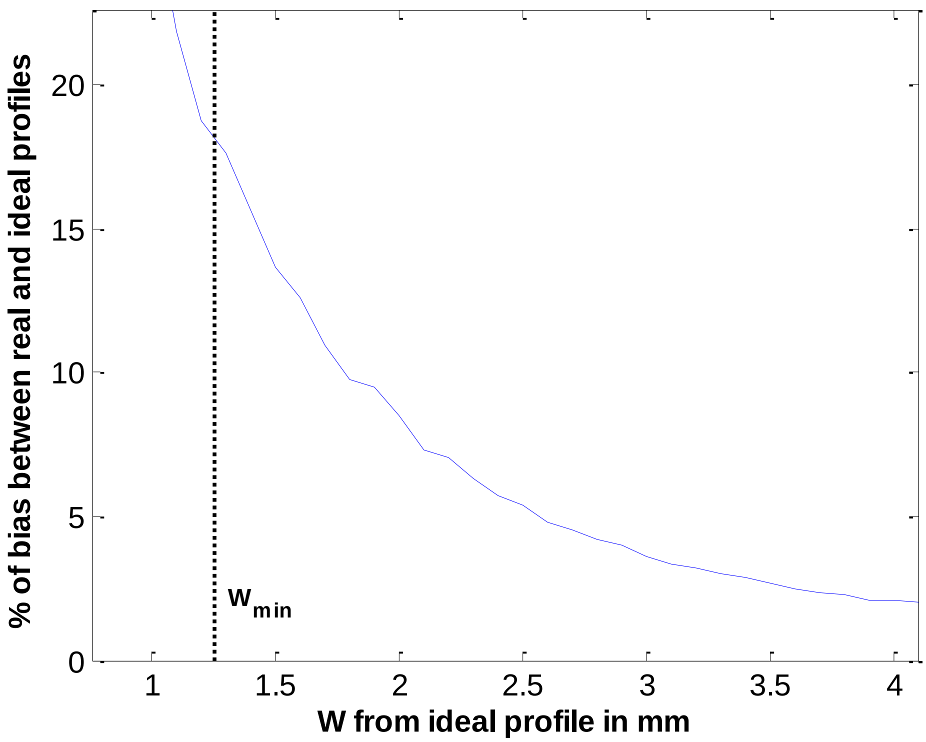
```
%% Plot 1
```




```
%% Plot 2.1
```




```
%% Plot 2.2
```


% of bias between real and ideal profiles
0
5
10
15
20
1
1.5
2
2.5
3
3.5
4
W from ideal profile in mm
W_{min}