Facultad
de
Ciencias

APPENDIX 1 - MATLAB PROGRAMS

(Apéndice 1 - Programas de MATLAB)

Trabajo de Fin de Grado
para acceder al

GRADO EN FÍSICA

Autor: Mario Alonso González

Director: Ángel Mañanes Pérez

Co-Director: Juan Remondo Tejerina

Julio - 2015

Contents

1 Turning .txt into spectrum

This program plots a spectrum out of a .txt file. The only arguments needed are the name of the file of the spectrum to plot (first argument, "filename.txt") and, if wanted, the name of the background spectrum to plot (second argument, "backgroundname") superposed to the first one. The output arguments are two vectors: one with the energies associated to every channel and another one with all the counts in every channel (energy). The part of the program concerning plotting the spectrum without background is commented so that the manual peak fitting and area measurement programs (which repeatedly call this function to use its output arguments energy and counts of a spectrum) work faster. It should be uncommented if a plot of the spectrum is wanted.

1.1 Code

```
%%%This program plots spectra out of .txt files. The only arguments needed
%%%are the name of the file with the spectrum to plot (first argument, "filename") and,
%%%if wanted, the name of the background spectrum to plot
%%%(second argument, "backgroundname")
%%%superposed to the first one. The output arguments are two vectors: one with
%%%the energies associated to every chanel and another one with all the counts
%%%in every channel (energy). The part of plotting the spectrum without
%%%background is commented so that the manual peak fitting (which
%%%reiterively calls this function) works faster, but it should be
%%%uncommented if a plot of the spectrum is wanted.


function [E,C]=txt2spectrum(filename,backgroundname)
close all

rango=[0 2600]; %energy interval to plot

%%Energy calibration: E=a1+a2*Ch+a3*Ch^2+a4*Ch^3+a5*Ch^4+a6*Ch^5
a1=1.73563825951;
a2=0.31590428655;
a3=3.26923463777e-7;
a4=-1.04310863211e-10;
a5=1.13383090719e-14;
a6=-3.78713313512e-19;

%%Import file: "filename"
D=importdata(filename);
M=D.data();

for j=1:1024
    for i=1:8
        C((j-1)*8+i)=M(j,i);
    end
end
C(1)=0;
C(2)=0; %Both channels are set to 0 counts for the .txt associates then a huge number of c
```

```
Ch=1:8192;
for i=1:8192
    E(i)=a1+a2*i+a3*i^2+a4*i^3+a5*i^4+a6*i^5;
end

%The plotting can be either with a superposed background or without it.

if backgroundname==0 %without background
%     %Figure 1: linear spectrum
%     figure(1)
%     plot(E,C)
%     legend(filename,'Location','northeast')
%     xlabel('Energy / keV')
%     ylabel('Counts')
%     xlim(rango) %energy interval to plot
%
%     %Figure 2: logrithmic spectrum
%     figure(2)
%     semilogy(E,C)
%     legend(filename,'Location','northeast')
%     xlabel('Energy / keV')
%     ylabel('Counts')
%     xlim(rango) %energy range to plot

else %with background
    %Import background file
    DB=importdata(backgroundname);
    B=DB.data();

    for j=1:1024
        for i=1:8
            Cf((j-1)*8+i)=B(j,i);
        end
    end
    Cf(1)=0;
    Cf(2)=0;

    %Figure 1: linear spectrum
    figure(1)
    plot(E,C,'b-')
    hold on
    plot(E,Cf,'r-')
    legend(filename,backgroundname,'Location','northeast')
    xlabel('Energy / keV')
    ylabel('Counts')
    xlim(rango) %energy interval to plot

    %Figure 2: logarithmic spectrum
    figure(2)
    semilogy(E,C,'b-')
    hold on
    semilogy(E,Cf,'r-')
    legend(filename,backgroundname,'Location','northeast')
    xlabel('Energy / keV')
    ylabel('Counts')
```

```
    xlim(rango) %energy interval to plot
end
```

2 Manual peak fitting

The aim of this function is mainly to set the channel interval of a certain peak manually ("manually" meaning not with the detector software fit), although it also calculates the activity (cps) of the peak for that certain interval (as defined in section 3.3 of the main work). For that purpose, when calling the function, four input arguments are needed: the name of the spectrum file ("filename.txt"), the counting time ("time"), the starting channel of the peak ("Ps") and the end channel of the peak ("Pe"). The output argument is the activity of that peak for the specified counting time and channel interval. Also, a final figure is created, where the channels of the peak are plotted in green, the channels used to calculate the background are plotted in red and a number of surrounding channels (blue) are plotted as well in order to see the surroundings of the analyzed peak, while a yellow line determining the calculated background of the peak is plotted too. An example is showed in Figure 2.1 below. With this graphical representation, one can observe if the channel interval selected is appropriate and, if not, change it easily.

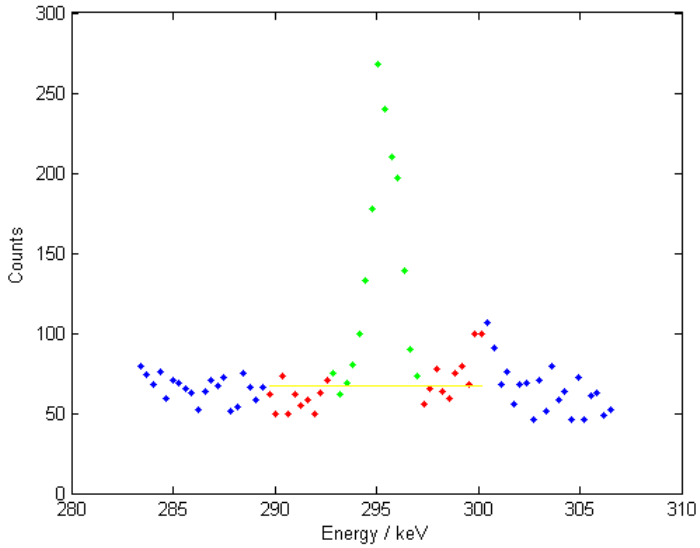

Figure 2.1: Example of the graphical representation of the manual peak fitting program. The channels of the peak are plotted in green, the channels used to calculate the background are plotted in red and a number of surrounding channels are plotted in blue, while a yellow line determining the background of the peak is plotted as well.

2.1 Code

```
function A=peakfitconstmanual(filename,time,Ps,Pe)
    close all

    [energia,cuentas]=txt2spectrum(filename,0);

    %Peak adjustment
```

```
P210s=Ps; %starting channel
P210e=Pe; %end channel

n=10; %number of background channels taken to the right and left of the peak
nch=20; %number of channels to plot in the final plot to the right and to the left of
P210bs=P210s-n;
P210be=P210e+n;
w=P210e-P210s; %peak width (channels)
Pchs=P210bs-nch;
Pche=P210be+nch;

j=0;
k=0;
Ech=zeros(1,2*nch);
Cch=zeros(1,2*nch);
Eb=zeros(1,2*n);
Cb=zeros(1,2*n);
Ep=zeros(1,w);
Cp=zeros(1,w);
for i=1:8192
    if i>=Pchs && i<P210bs
        k=k+1;
        Ech(k)=energia(i);
        Cch(k)=cuentas(i);
    elseif i>P210be && i<=Pche
        k=k+1;
        Ech(k)=energia(i);
        Cch(k)=cuentas(i);
    elseif i>=P210s && i<=P210e %peak counts
        Ep(i-P210s+1)=energia(i);
        Cp(i-P210s+1)=cuentas(i);
    elseif i>=P210bs && i<P210s %left background
        j=j+1;
        Eb(j)=energia(i);
        Cb(j)=cuentas(i);
    elseif i>P210e && i<=P210be %right background
        j=j+1;
        Eb(j)=energia(i);
        Cb(j)=cuentas(i);
    elseif i>Pche
        break;
    end
end

background=sum(Cb)/(2*n);
gross=sum(Cp);
net=gross-background*w;

eA=sqrt(sum(Cp)+(w/(2*n))^2*sum(Cb))/time;
A=net/time;

Eaj=Eb;
for i=1:length(Ep)
    Eaj(2*n+i)=Ep(i);
end
Caj=zeros(1,length(Eaj));
```

```
    for i=1:length(Eaj)
        Caj(i)=background;
    end

    figure(1)
    plot(Ep,Cp,'g.')
    hold on
    plot(Eb,Cb,'r.')
    hold on
    plot(Ech,Cch,'b.')
    hold on
    plot(Eaj,Caj,'y-')
    xlabel('Energy')
    ylabel('Counts')
end
```

3 Manual peak area measurement

The aim of this function is to calculate the activity of different peaks of interest (whose channel intervals are defined within the function) by measuring its peak area manually (as described in section 3.3.3 of the main work). Also, an error and the MDA is associated to each calculated activity. The three of them (activity, error and MDA) are output arguments. The input arguments are the name of the spectrum file ("filename" in .txt format) and the counting time ("time"). It can be seen that this program is an adaptation of the previous peak fitting program. The difference is that in this case the peak fitting is not the goal (the exact values of the channel intervals of the interesting peaks are supposed to be fit with the previous program) and so there is no final figure, but the purpose is to measure the peak area of different peaks and so return the activity, error and MDA for further calculations with another program. The instance showed below corresponds to the program used to calculate the activities of the peaks of ^{210}Pb and the other ^{238}U decay chain peaks used in the calculations: ^{214}Pb, ^{214}Bi and ^{234}Th. An analogous program with little adaptions was used to calculate the activity of the ^{137}Cs peak too.

3.1 Code

```
%%% Manual peak area measurement of excess 210Pb and the supported peaks
%%%
%%% Mario Alonso Gonzalez

function [A,eA,Amda]=peakfitconst(filename,time)
    close all

    [energia,cuentas]=txt2spectrum(filename,0);

    n=10; %background channels in the calculation (n to the right and n to the left)
    np=5; %peak number
    A=zeros(1,np);
    eA=zeros(1,np);
    N=length(energia);

    for pico=1:np
        clearvars -except energia cuentas n A N pico time eA
        if pico==1
            %Fitting 46 keV peak of 210 Pb
            P210s=136; %starting channel
            P210e=149; %end channel
        elseif pico==2
            %Fitting 295 keV peak of 214 Pb
            P210s=920; %starting channel
            P210e=937; %end channel
        elseif pico==3
            %Fitting 352 keV peak of 214 Pb
            P210s=1098; %starting channel
            P210e=1117; %end channel
        elseif pico==4
```

```
            %Fitting 607 keV peak of 214 Bi
            P210s=1914; %starting channel
            P210e=1930; %end channel
        elseif pico==5
            %Fitting 63 keV  peakof 234 Th
            P210s=189; %starting channel
            P210e=202; %end channel
        end

        P210bs=P210s-n;
        P210be=P210e+n;
        w=P210e-P210s; %peak width (channels)

        j=n;
        k=0;
        for i=1:N
            if i>=P210s && i<=P210e %peak counts
                Ep(i-P210s+1)=energia(i);
                Cp(i-P210s+1)=cuentas(i);
            elseif i>=P210bs && i<P210s %left background
                k=k+1;
                Eb(k)=energia(i);
                Cb(k)=cuentas(i);
            elseif i>P210e && i<=P210be %right background
                j=j+1;
                Eb(j)=energia(i);
                Cb(j)=cuentas(i);
            elseif i>P210be
                break;
            end
        end

        background=sum(Cb)/(2*n);
        gross=sum(Cp);
        net=gross-background*w;

        A(pico)=net/time;
        eA(pico)=2*sqrt(sum(Cp)+(w/(2*n))^2*sum(Cb))/time;
        Amda(pico)=1.645*sqrt(background*(1+w/(2*n)))/time;
    end
end
```

4 CRS and CIC calculations

The aim of this program ($SV1_manual$) is to obtain the activity profiles with depth of ^{210}Pb, as well as the supported ^{210}Pb, the excess ^{210}Pb, the accumulated activity, the dating by both CIC and CRS models and the sedimentation rate by the CRS model. This is done by a prior peak are measurement with the help of the previous program applied to a number of spectra. This program therefore displays all the figures showed the Results and Discussions section of the main work (except the ones obtained by the Sediment Mixing model, whose program is introduced in the next section). Once again, for the ^{137}Cs calculation, a new program with little adaption was made. This program was applied to do the SV1 core calculations and is showed as an example, but an analogous program was used for the TM core.

4.1 Code

```
clear all
close all

Calib_vol_SV1; %density measurements
close all
clc

%%Efficiency-energy calibration parameters of the Ge detector
p1=0.1031;
p2=-1.946;
p3=11.07;
p4=-22.61;

x=[1 2.75 4.25 5.75 7.25 8.75 10.25 11.75 13.25 14.75 16.25 17.75 19.25 20.75 22.25 23.75]
N=length(x); %Number of sediment layers
m=0.05; %Masas (kg) of sample

E=[46.52 295.22 351.99 609.32 63.29]; %Energy of the studied peaks
p=[0.0425 0.185 0.358 0.448 0.0484]; %Probability of emission of each peak
b=[0.0015 0.0030  0.0056 0.0036 0.0059]; %Background cps
np=length(E); %number of peaks

%Peak are measurements using the .txt spectrums of all the samples
cps=zeros(N,np);
error=zeros(N,np);
[cps(1,:),error(1,:),mda(1,:)]=peakfitconst('SV1_31.txt',129600);
[cps(2,:),error(2,:),mda(2,:)]=peakfitconst('SV1_30.txt',86400);
[cps(3,:),error(3,:),mda(3,:)]=peakfitconst('SV1_29_1.txt',86400);
[cps(4,:),error(4,:),mda(4,:)]=peakfitconst('SV1_28.txt',86400);
[cps(5,:),error(5,:),mda(5,:)]=peakfitconst('SV1_27.txt',86400);
[cps(6,:),error(6,:),mda(6,:)]=peakfitconst('SV1_26_1.txt',86400);
[cps(7,:),error(7,:),mda(7,:)]=peakfitconst('SV1_25.txt',86400);
[cps(8,:),error(8,:),mda(8,:)]=peakfitconst('SV1_24.txt',86400);
[cps(9,:),error(9,:),mda(9,:)]=peakfitconst('SV1_23.txt',86400);
[cps(10,:),error(10,:),mda(10,:)]=peakfitconst('SV1_22.txt',86400);
[cps(11,:),error(11,:),mda(11,:)]=peakfitconst('SV1_21.txt',86400);
```

```
[cps(12,:),error(12,:),mda(12,:)]=peakfitconst('SV1_20.txt',86400);
[cps(13,:),error(13,:),mda(13,:)]=peakfitconst('SV1_19_1.txt',86400);
[cps(14,:),error(14,:),mda(14,:)]=peakfitconst('SV1_18.txt',86400);
[cps(15,:),error(15,:),mda(15,:)]=peakfitconst('SV1_17.txt',86400);
[cps(16,:),error(16,:),mda(16,:)]=peakfitconst('SV1_16.txt',86400);

cps=transpose(cps);
error=transpose(error);
mda=transpose(mda);

%Loop that goes through all the peaks
n=length(E); %number of analyzed peaks
for i=1:n
    %Efficiency calculation for each peak energy
    logE(i)=log(E(i));
    eff(i)=exp(p1*logE(i)^3+p2*logE(i)^2+p3*logE(i)+p4);
    %efficiency error
    erEff=0; %set to zero because it is neglectable

    for j=1:N
        %%Net counts = cps - background
        net(i,j)=cps(i,j)-b(i);
        %%Specific activity (Bq/kg) calculation
        A(i,j)=net(i,j)/(eff(i)*p(i)*m);

        %Error of activity
        er(i,j)=sqrt((error(i,j)/cps(i,j))^2+(erEff)^2);
        e(i,j)=er(i,j)*A(i,j);

        %%mda
        Amda(i,j)=mda(i,j)/(eff(i)*p(i)*m);
    end

end

lambda=0.03114; %decay constant of 210Pb

%Activity of supported 210Pb
Asop=zeros(1,N);
%Average of 214Pb y 214Bi activities
for j=1:N
   for i=2:4
      Asop(j)=Asop(j)+A(i,j);
   end
   Asop(j)=Asop(j)/3;
end

%Error supported 210Pb
eSop=zeros(1,N);
tam=size(A);
nn=tam(1); %peak number
for j=1:N
    for i=2:nn
        eSop(j)=eSop(j)+(abs(A(i,j)-Asop(j)))^2;
    end
    eSop(j)=sqrt(eSop(j)/(nn));
```

```
end

%Calculate excess 210Pb activity
Aexc=A(1,:)-Asop;

%Error dof excess 210Pb
for i=1:N
    logExc(i)=log(Aexc(i));
    eExc(i)=sqrt(e(1,i)^2+eSop(i)^2);
    elogExc(i)=eExc(i)/Aexc(i);
end

%Initial activity of 210Pb
Aexc0=Aexc(1);
eExc0=eExc(1);
eT0=eExc(1);
Ac=zeros(1,N); %Accumulated activity (Bq/m^2) of excess 210Pb


eAcMom=0;
for i=1:N
    %CIC dating with excess 210Pb
    tCic(i)=1/lambda*log(Aexc0/Aexc(i));
    eTCic(i)=1/lambda*sqrt((eExc0/Aexc0)^2+(eExc(i)/Aexc(i))^2);

    %Accumulated activity of excess 210Pb
    for j=i:N
        if j==1
            dx=2;
        else
            dx=1.5;
        end
        Ac(i)=Ac(i)+Aexc(j)*d(j)*dx/100;
        eAcMom=eAcMom+(Aexc(j)*d(j)*dx/100)^2*((eD(j)/d(j))^2+(eExc(j)/Aexc(j))^2);
    end
    eAc(i)=sqrt(eAcMom); %error
    logAc(i)=log(Ac(i)); %logarithm
    eLogAc(i)=eAc(i)/Ac(i); %error logarithm
    eAcMom=0;

    %CRS dating
    tCrs(i)=1/0.031*log(Ac(1)/Ac(i));
    eTCrs(i)=1/lambda*sqrt((eAc(1)/Ac(1))^2+(eAc(i)/Ac(i))^2);

    %Sedimentation rate
    s(i)=lambda*Ac(i)/(d(i)*Aexc(i))*100;
    tA(i)=2015-tCrs(i);
    eSd(i)=sqrt((eD(i)/d(i))^2+(eExc(i)/Aexc(i))^2+(eAc(i)/Ac(i))^2)*s(i);
end


%%DRAW
figure(1)
errorbar(x,A(1,:),e(1,:),'ko')
hold on
plot(x,Amda(1,:),'k-')
```

```
title('210Pb activity and MDA')
xlabel('Profundidad / cm')
ylabel('Actividad / Bq kg^{-1}')

figure(2)
errorbar(x,A(1,:),e(1,:),'ko')
hold on
errorbar(x,A(2,:),e(2,:),'ro')
hold on
errorbar(x,A(3,:),e(3,:),'bo')
hold on
errorbar(x,A(4,:),e(4,:),'go')
hold on
errorbar(x,A(5,:),e(5,:),'mo')
title('Black: 210Pb, Red: 295 keV of 214Pb, Blue: 352 keV of 214Pb, Green: 609 keV of 214E
xlabel('Depth / cm')
ylabel('Activity / Bq kg^{-1}')

zerox=[0 50];
zeroy=zerox.*0;

figure(3)
errorbar(x,Aexc,eExc,'ro')
hold on
plot(zerox,zeroy,'k-')
title('Activity of excess 210Pb')
xlabel('Depth / cm')
ylabel('Activity of excess 210Pb / Bq kg^{-1}')
xlim([0 25])

figure(4)
errorbar(x,Asop,eSop,'go')
title('Activity of supported 210Pb')
xlabel('Depth / cm')
ylabel('Activity of supported 210Pb / Bq kg^{-1}')
ylim([0 150])

figure(5)
errorbar(x,logExc,elogExc,'ro')
title('Activity of excess 210Pb')
xlabel('Depth / cm')
ylabel('log A_{exc} / Bq kg^{-1}')
ylim([-2 8])

figure(6)
errorbar(x,tCic,eTCic,'o')
title('CIC dating')
xlabel('Depth / cm')
ylabel('Formation time / years')
ylim([0 250])

figure(7)
errorbar(x,Ac,eAc,'ko')
title('Accumulated activity of excess 210Pb')
xlabel('Depth / cm')
ylabel('Accumulated activity / Bq/m^2')
```

```
ylim([0 14000])

figure(8)
errorbar(x,logAc,eLogAc,'ko')
title('Accumulated activity of excess 210Pb')
xlabel('Depth / cm')
ylabel('log A_{ac} / Bq/m^2')
ylim([2 10])

figure(9)
errorbar(x,tCrs,eTCrs,'o')
title('CRS dating')
xlabel('Depth / cm')
ylabel('Formation time / year')
ylim([0 250])

for i=1:15
    tA2(i)=tA(i);
    s2(i)=s(i);
    eSd2(i)=eSd(i);
    eTCrs2(i)=eTCrs(i);
end

figure(10)
errorbar(tA2,s2,eSd2,'o')
hold on
herrorbar(tA2,s2,eTCrs2,'o')
xlabel('Year')
ylabel('Sedimentation rate / cm yr^{-1}')
ylim([0 1])
xlim([1880 2020])
```

5 Sediment Mixing simulation for $^{210}Pb_{xs}$

The aim of this program is to perform the numerical simulation of the Sediment Mixing model (section 2.4 of the main work). Starting from the initial condition of the sediment column being void of tracer at the initial time (1800 A.C.) and provided the values of all the parameters needed for the simulation, a temporal evolution of the $^{210}Pb_{xs}$ profile was simulated including sedimentation, diffusion and radioactive decay until the collecting time t_c (2015). The temporal evolution is graphically observed by plotting the profile at every time step (this can be commented seeking a faster running time). The $^{210}Pb_{xs}$ influx to the system is assumed to be constant except for an episodic event in 1983 corresponding to the "cold drop" phenomenon. In the final figure, the results of the numerical simulation are compared with the experimental results.

5.1 Code

```
close all
clear all
clc

SV1_manual;
close all;

%Simulation characteristics
xf=100; %maximum depth of simulation
t0=1800; %starting year of simulation
tc=2015; %final year of simulation
deltax=0.25; %delta depth
deltat=0.05; %delta time
Nx=xf/deltax; %number of points in depth
Nt=(tc-t0)/deltat; %number of points in time
xsim=0:deltax:xf-deltax;
I=eye(Nx); %identity matrix

%Parameters (to change)
rho=2.5;
lambda=0.03141; %decay constant of 210Pb
Db_0=0.3; %bioturbation-diffusion constant at the surface
P=0.001; %transfer function
w_inf=0.185; %sedimentation rate at deepest
w_inf_event=1.85; %sedimentation rate of the event
tevent=1983; %time of the event
sigmat=0.5; %duration of the event in years
xstar=3; %mixing depth, depth where Db/Db_0=0.5
sigma=3; %constant that controls dDb/dx in vecinity of x=xs
scale=0.66; %scale factor of input concentration

%Porosity fit (eq. 2 Nie and Suayah 2001 Oceanography)
beta=0.0714;
phi_0=0.6353;
phi_inf=0.4891;
```

```
phi=zeros(Nx,1);
for i=1:Nx
    phi(i)=phi_inf+(phi_0-phi_inf)*exp(-beta*(i-1)*deltax);
end
w_0=w_inf*(1-phi_inf)/(1-phi_0); %sedimentation rate at surface

%Bioturbation-diffusion constant
Db=zeros(Nx,1);
for i=1:Nx
    Db(i)=Db_0/2*erfc(((i-1)*deltax-xstar)/sigma);
    %Db(i)=Db_0;
end

%Sedimentation rate
w=zeros(Nx,1);
for i=1:Nx
    w(i)=w_inf*(1-phi_inf)/(1-phi(i));
end

%Sedimentation rate during the event
w_event=zeros(Nx,1);
for i=1:Nx
    w_event(i)=w_inf_event*(1-phi_inf)/(1-phi(i));
end

%Boundaries & initial conditions
C_in=0; %initial conditions
C_low=0; %lower-boundary
F_0=0; %upper-boundary
for i=1:N
    if j==1
        dx=2;
    else
        dx=1.5;
    end
    F_0=F_0+Aexc(i)*d(i)*dx/1000;
end
%F_0=F_0*lambda/exp(-lambda*tc)*(exp(lambda*tevent)*(1-exp(lambda*sigmat)+(P+(1-P)*w_inf_e
F_0=lambda*F_0;
C_up=F_0/((1-phi(1))*rho*w_inf);
C_up_event=F_0*(P+(1-P)*w_inf_event/w_inf)/((1-phi(1))*rho*w_inf);

%Calculate f(x) and its derivative
fx=(1-phi)*rho.*Db;
dfx=zeros(Nx,1);
for i=2:Nx-1
    dfx(i)=(fx(i)-fx(i-1))/deltax;
end
dfx(1)=dfx(2);
dfx(Nx)=dfx(Nx-1);

%Calculate g(x) and its derivative
gx=-(1-phi)*rho.*w;
dgx=zeros(Nx,1);
for i=2:Nx-1
    dgx(i)=(gx(i)-gx(i-1))/deltax;
```

```
end
dgx(1)=dgx(2);
dgx(Nx)=dgx(Nx-1);

%Calculate g(x) and its derivative during the event
gx_event=-(1-phi)*rho.*w_event;
dgx_event=zeros(Nx,1);
for i=2:Nx-1
    dgx_event(i)=(gx_event(i)-gx_event(i-1))/deltax;
end
dgx_event(1)=dgx_event(2);
dgx_event(Nx)=dgx_event(Nx-1);

%Calcular l(x)
lx=-(1-phi)*rho*lambda;


%Second derivative matrix
for i=1:Nx-1
    d211(i)=(fx(i)/(rho*(1-phi(i))))/deltax^2;
end
for i=2:Nx
    d212(i-1)=(fx(i)/(rho*(1-phi(i))))/deltax^2;
end
for i=1:Nx
    d22(i)=(-2*fx(i)/(rho*(1-phi(i))))/deltax^2;
end
M2=diag(d22)+diag(d212,-1)+diag(d211,1);

%First derivative matrix
for i=2:Nx
    d11(i-1)=(-1*(dfx(i)+gx(i))./(rho*(1-phi(i))))/deltax;
end
for i=1:Nx
    d12(i)=((dfx(i)+gx(i))./(rho*(1-phi(i))))/deltax;
end
M1=diag(d12)+diag(d11,-1);

%First derivative matrix during the event
for i=2:Nx
    d11_event(i-1)=(-1*(dfx(i)+gx_event(i))./(rho*(1-phi(i))))/deltax;
end
for i=1:Nx
    d12_event(i)=((dfx(i)+gx_event(i))./(rho*(1-phi(i))))/deltax;
end
M1_event=diag(d12_event)+diag(d11_event,-1);

%Zero order matrix
d01=ones(Nx,1).*(dgx+lx)./(rho*(1-phi));
M0=diag(d01);

%Zero order matrix during the event
d01_event=ones(Nx,1).*(dgx_event+lx)./(rho*(1-phi));
M0_event=diag(d01_event);

%Total matrix
```

```
M=M2+M1+M0;

%Total matrix during the event
M_event=M2+M1_event+M0_event;

%initialize C (activity vector)
C=ones(Nx,1)*C_in;

%Time goes byy
for tt=t0:deltat:tc
    if tt<tevent || tt>tevent+sigmat %before and after the event
        %Applying boundaries
        C(1)=C_up*scale;
        C(Nx)=C_low;

        %Drawing
        plot(xsim,C)
        title(tt)
        xlim([0 25])
        drawnow
        pause(0.02)

        %Nie-Suayah equation (Crank-Nicholson method)
        C=(I-M.*deltat/2)^(-1)*(I+M.*deltat/2)*C;
    else %during the event
        %Applying boundaries
        C(1)=C_up_event*scale;
        C(Nx)=C_low;

        %Drawing
        plot(xsim,C)
        xlim([0 25])
        title(tt)
        drawnow
        pause(0.02)

        %Nie-Suayah equation (Crank-Nicholson method)
        C=(I-M_event.*deltat/2)^(-1)*(I+M_event.*deltat/2)*C;
    end

end

%boundary conditions again
C(1)=C_up*scale;
C(Nx)=C_low;

%final plot
plot(xsim,C,'b-')
hold on
errorbar(x,Aexc,eExc,'ro')
ylabel('Activity ^{210}Pb_{xs} / Bq kg^{-1}')
xlabel('Depth / cm')
xlim([0 25])
ylim([0 180])
```

6 Sediment Mixing model for ^{137}Cs

Analogous program to the one before, numerical simulation seeking to reproduce the experimental ^{137}Cs profile. Note that all the parameters are the same than for the $^{210}Pb_{xs}$ simulation, except the diffusion parameters (biodiffusion coefficient and mixing depth) and the tracer influx to the system, which in this case is assumed to be a Lorentzian function centered in 1963 to reproduce the intense deposition of ^{137}Cs during the 1950's and 1960's, extended to a little ^{137}Cs deposition at present time due to the atmospheric ^{137}Cs reservoir. An analogous program was used to reproduce the ^{40}K profile, with all the parameters set to exactly the same values as for ^{137}Cs, except for the ^{40}K influx to the system, which was assumed to be constant (precipitated from the overlying water) and added up to a constant value present in the soils.

6.1 Code

```
close all
clear all
clc

SV1_Cs_manual;
Calib_vol_SV1;
close all;

%Simulation characteristics
xf=50; %maximum depth of simulation
t0=1950; %starting year of simulation
tc=2015; %final year of simulation
deltax=0.25; %delta depth
deltat=0.05; %delta time
Nx=xf/deltax; %number of points in depth
Nt=(tc-t0)/deltat; %number of points in time
xsim=0:deltax:xf-deltax;
I=eye(Nx); %identity matrix

%Parameters (to change)
gamma=10; %width of 137Cs peak
tCsStart=1950; %year of the beggining of 137Cs fallout
tmax=1963; %year of the 137 peak maximum
rho=2.5; %density of solid phase
lambda=0.0229; %decay constant of 210Pb
Db_0=0.5; %bioturbation-diffusion constant at the surface
P=0.001; %transfer function
w_inf=0.185; %sedimentation rate at deepest
w_inf_event=1.85; %sedimentation rate of the event
tevent=1983; %time of the event
sigmat=0.5; %duration of the event in years
xstar=15.5; %mixing depth, depth where Db/Db_0=0.5
sigma=3; %constant that controls dDb/dx in vecinity of x=xs
scale=1.9; %scale factor of input concentration

%Porosity fit (eq. 2 Nie-Suayah 2001 Oceanography)
```

```
beta=0.0714;
phi_0=0.6353;
phi_inf=0.4891;
phi=zeros(Nx,1);
for i=1:Nx
    phi(i)=phi_inf+(phi_0-phi_inf)*exp(-beta*(i-1)*deltax);
end
w_0=w_inf*(1-phi_inf)/(1-phi_0); %sedimentation rate at surface

%Bioturbation-diffusion constant
Db=zeros(Nx,1);
for i=1:Nx
    Db(i)=Db_0/2*erfc(((i-1)*deltax-xstar)/sigma);
    %Db(i)=Db_0;
end

%Sedimentation rate
w=zeros(Nx,1);
for i=1:Nx
    w(i)=w_inf*(1-phi_inf)/(1-phi(i));
end

%Sedimentation rate during the event
w_event=zeros(Nx,1);
for i=1:Nx
    w_event(i)=w_inf_event*(1-phi_inf)/(1-phi(i));
end

%Boundaries & initial conditions
C_in=0; %initial conditions
C_low=0; %lower-boundary

%Upper-boundary
F_0=0;
for i=1:N
    if j==1
        dx=2;
    else
        dx=1.5;
    end
    F_0=F_0+A(i)*d(i)*dx/1000;
end

F_1=0;
for t=tCsStart:deltat:tc
F_1=F_1+exp(-lambda*(tc-t))/(pi*gamma*(1+((t-tmax)/gamma)^2))*deltat;
end
t=tCsStart:deltat:tc;
Fmax=F_0/F_1;
F=Fmax./(pi*gamma*(1+((t-1963)/gamma).^2));

C_up=zeros(1,Nt);
for i=1:Nt
    tt=t0+deltat;
    if tt<tevent || tt>tevent+sigmat
        C_up(i)=F(i)/((1-phi(1))*rho*w_inf);
```

```
    else
        C_up(i)=F*(P+(1-P)*w_inf_event/w_inf)/((1-phi(1))*rho*w_inf);
    end
end

%Calculate f(x) and its derivative
fx=(1-phi)*rho.*Db;
dfx=zeros(Nx,1);
dfx(1)=0;
dfx(Nx)=0;
for i=2:Nx-1
    dfx(i)=(fx(i)-fx(i-1))/deltax;
end

%Calculate g(x) and its derivative
gx=-(1-phi)*rho.*w;
dgx=zeros(Nx,1);
dgx(1)=0;
dgx(Nx)=0;
for i=2:Nx-1
    dgx(i)=(gx(i)-gx(i-1))/deltax;
end

%Calculate g(x) and its derivative during the event
gx_event=-(1-phi)*rho.*w_event;
dgx_event=zeros(Nx,1);
for i=2:Nx-1
    dgx_event(i)=(gx_event(i)-gx_event(i-1))/deltax;
end
dgx_event(1)=dgx_event(2);
dgx_event(Nx)=dgx_event(Nx-1);

%Calculate l(x)
lx=-(1-phi)*rho*lambda;


%Second derivative matrix
for i=2:Nx
    d21(i-1)=fx(i)/(rho*(1-phi(i)))/deltax^2;
end
for i=1:Nx
    d22(i)=-2*fx(i)/(rho*(1-phi(i)))/deltax^2;
end
M2=diag(d22)+diag(d21,-1)+diag(d21,1);

%First derivative matrix
for i=2:Nx
    d11(i-1)=-1*(dfx(i)+gx(i))./(rho*(1-phi(i)))/deltax;
end
for i=1:Nx
    d12(i)=(dfx(i)+gx(i))./(rho*(1-phi(i)))/deltax;
end
M1=diag(d12)+diag(d11,-1);

%First derivative matrix during the event
for i=2:Nx
```

```
    d11_event(i-1)=(-1*(dfx(i)+gx_event(i))./(rho*(1-phi(i))))/deltax;
end
for i=1:Nx
    d12_event(i)=((dfx(i)+gx_event(i))./(rho*(1-phi(i))))/deltax;
end
M1_event=diag(d12_event)+diag(d11_event,-1);

%Zero order matrix
d01=ones(Nx,1).*(dgx+lx)./(rho*(1-phi));
M0=diag(d01);

%Zero order matrix during the event
d01_event=ones(Nx,1).*(dgx_event+lx)./(rho*(1-phi));
M0_event=diag(d01_event);

%Total matrix
M=M2+M1+M0;

%Total matrix during the event
M_event=M2+M1_event+M0_event;

%initialize C (activity vector)
C=ones(Nx,1)*C_in;

%Time goes byy
for i=1:Nt
    tt=t0+(i-1)*deltat;

    %Applying boundaries
    C(1)=C_up(i)*scale;
    C(Nx)=C_low;

    %Drawing
    plot(xsim,C)
    xlim([0 25])
    title(tt)
    drawnow
    pause(0.02)

    %Nie-Suayah equation (Crank-Nicholson method)
    if tt<tevent || tt>tevent+sigmat %before and after the event
        C=(I-M.*deltat/2)^(-1)*(I+M.*deltat/2)*C;
    else %during the event
        C=(I-M_event.*deltat/2)^(-1)*(I+M_event.*deltat/2)*C;
    end
end

%boundary conditions again
C(1)=C(2);
C(Nx)=C_low;

plot(xsim,C,'b-')
hold on
errorbar(x,A,e,'mo')
hold on
plot(x,Amda,'m-')
```

```
ylabel('Activity 137Cs / Bq kg^{-1}')
xlabel('Depth / cm')
xlim([0 25])
ylim([0 8])
```