Lesson 2:
CHATWIN, MOMENTS & CALIBRATION METHODS

2020

Giulia Comunale & Paolo Peruzzo

giulia.comunale@dicea.unipd.it

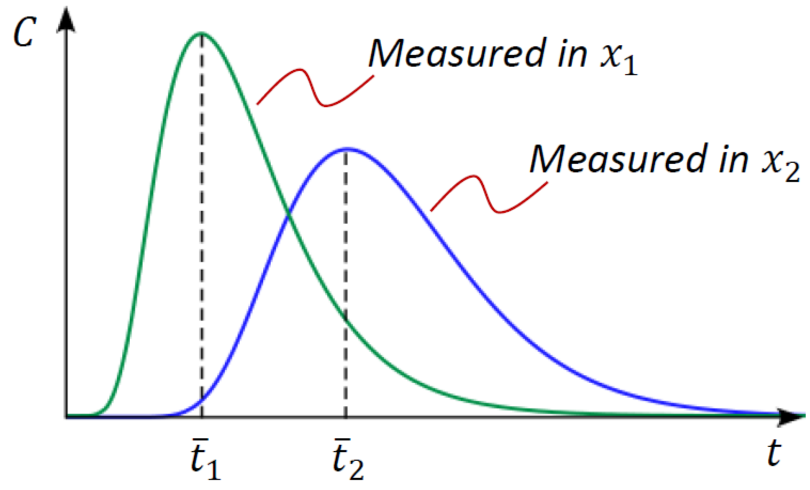
The fundamental solution of the advection dispersion equation is (Taylor):

$$C(x,t) = \frac{M}{A\sqrt{4\pi K_x t}} \exp\left(-\frac{(x-U_0 t)^2}{4K_x t}\right)$$

A = cross-sectional area

M = injected mass (mg)

K_x = coefficient of diffusion



Time-Concentration

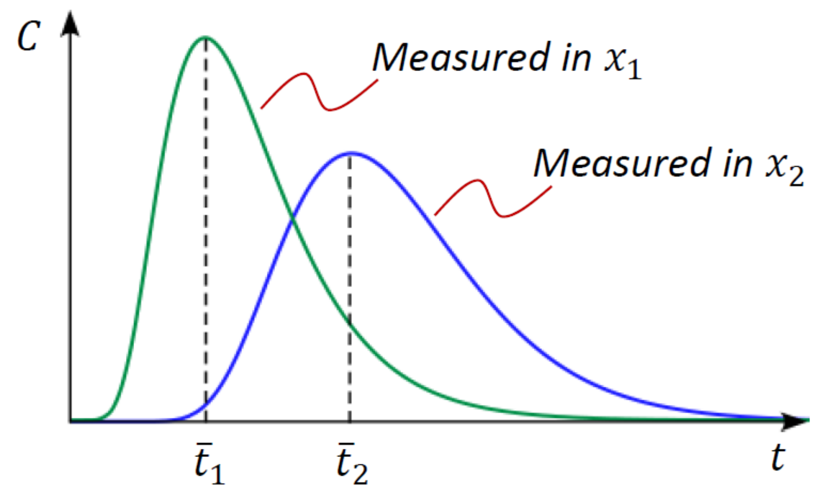
The fundamental solution of the advection dispersion equation is (Taylor):

$$C(x,t) = \frac{M}{A\sqrt{4\pi \boxed{K_x} t}} \exp\left(-\frac{(x - \boxed{U_0} t)^2}{4K_x t}\right)$$

A = cross-sectional area

M = injected mass (mg)

K_x = coefficient of diffusion

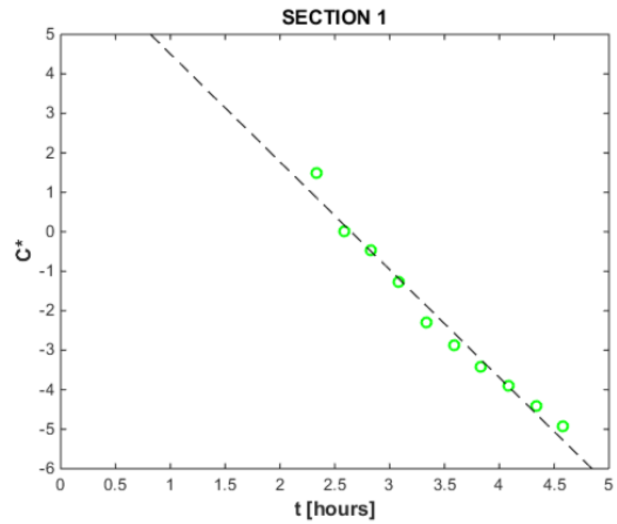


CHATWIN METHOD

Chatwin method is based on the rewriting of the fundamental solution proposed by Taylor for 1D Fickian type process.

Manipulated solution gives the Fictitious Concentration as:

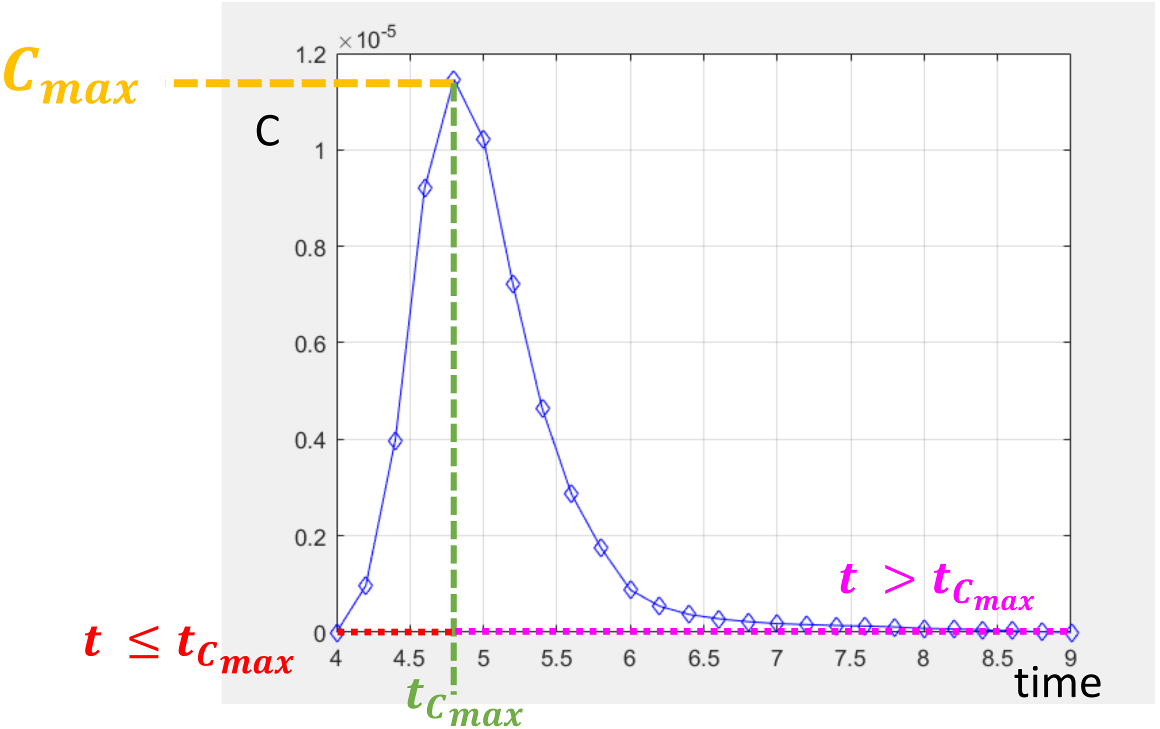
$$C^* = -\underbrace{\boxed{\frac{U}{2\sqrt{K}}}}_{K}t + \underbrace{\boxed{\frac{x}{2\sqrt{K}}}}_{U}$$



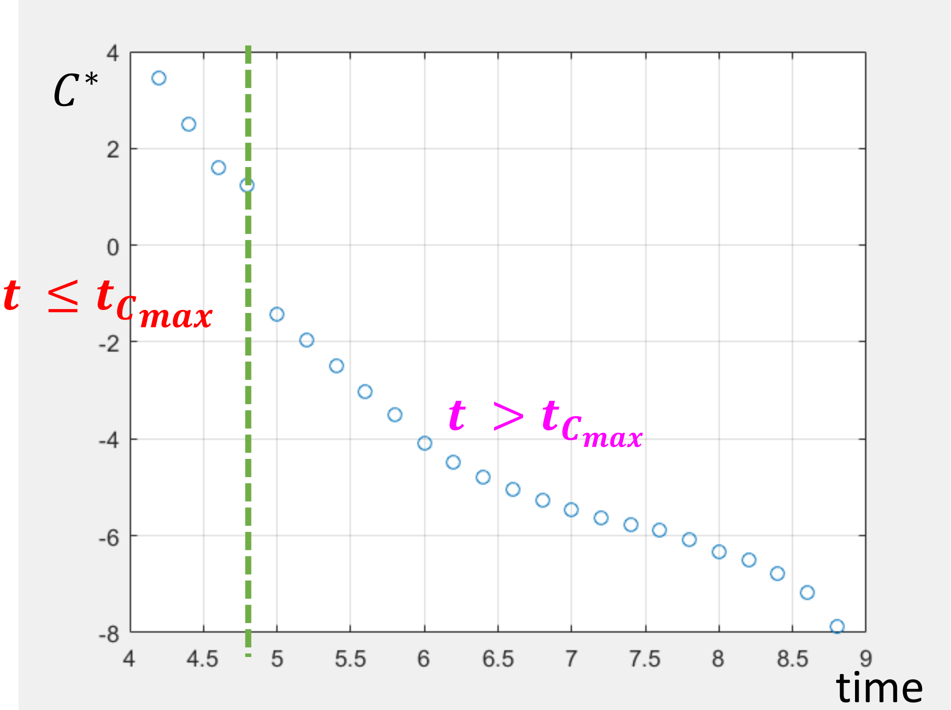
$$C^* = \begin{cases} -\sqrt{\boldsymbol{t} \cdot ln\dfrac{C_{max}\sqrt{t_{C_{max}}}}{C\sqrt{t}}} & if \quad \boldsymbol{t} > \boldsymbol{t_{C_{max}}} \\ \sqrt{\boldsymbol{t} \cdot ln\dfrac{C_{max}\sqrt{t_{C_{max}}}}{C\sqrt{t}}} & if \quad \boldsymbol{t} \leq \boldsymbol{t_{C_{max}}} \end{cases}$$

K *and* ***U*** *are calculated from the slope and the intercept of the dashed line*

$$C^* = \begin{cases} -\sqrt{t \cdot ln \dfrac{C_{max}\sqrt{t_{C_{max}}}}{C\sqrt{t}}} & if \quad t > t_{C_{max}} \\ \sqrt{t \cdot ln \dfrac{C_{max}\sqrt{t_{C_{max}}}}{C\sqrt{t}}} & if \quad t \le t_{C_{max}} \end{cases}$$



$$C^*(i) = \begin{cases} -\sqrt{t(i) \cdot \ln \dfrac{C_{max}\sqrt{t_{C_{max}}}}{C(i)\sqrt{t(i)}}} & if \quad t(i) > t_{C_{max}} \\ \sqrt{t(i) \cdot \ln \dfrac{C_{max}\sqrt{t_{C_{max}}}}{C(i)\sqrt{t(i)}}} & if \quad t(i) \leq t_{C_{max}} \end{cases}$$

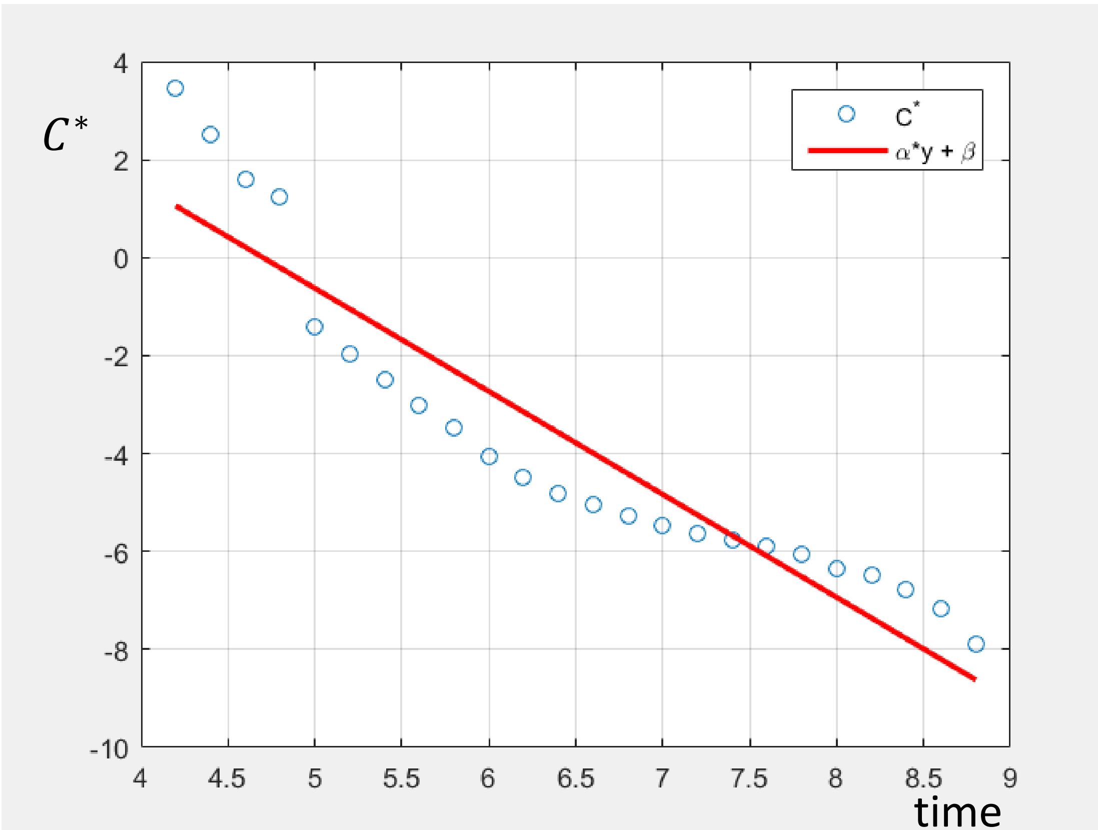


To implement this function, we need to define:

- C_{max} (*function max*);
- $t_{C_{max}}$ (*function find/max*);
- ***for* cycle**;
- ***if* function**;

$$C^* = -\boxed{\frac{U}{2\sqrt{K}}}t + \boxed{\frac{x}{2\sqrt{K}}} = \boldsymbol{\alpha y} + \boldsymbol{\beta}$$

K and U are calculated from the slope and the intercept of the dashed line



To obtain α and β (and thus the red line) use the function ***polyfit***

p = polyfit(*time*,*concentration*,1)

result: [α,β]

$$\begin{cases} \alpha = \dfrac{U}{2\sqrt{K}} \\ \beta = \dfrac{x}{2\sqrt{K}} \end{cases} \dashrightarrow \text{result: [U,K]}$$

- Create a new folder **chatwin_method;**
- Open matlab;
- Create a new M-file: ***chatwin.m;***
- Load data for each section (function load);
- Extract the time and the concentration for each section;
- Plot for each section the time vs concentration (figure);.
- Define $\boldsymbol{C_{max}}$ **for each section** (function *max*);
- Define $\boldsymbol{t_{C_{max}}}$ (function *find*);
- Define **C*** (*for* cycle and *if* function);
- Plot **C*** vs time for each section;
- Use the function polyfit to evaluate α and β;
- Plot the line $y = \alpha \cdot x + \beta$ together with the plot of C*
- Define U and K for each section;

MOMENTS METHOD

For the **Moments method** the solution was obtained by Fisher and it is based on the variance of temporal distribution of concentration.

$$\boldsymbol{K = \frac{1}{2} U_0 \frac{\sigma_t^2(x_{i+1}) - \sigma_t^2(x_i)}{\overline{t_{c(i+1)}} - \overline{t_{c(i)}}}}$$

- $\boldsymbol{U_0}$ is the mean velocity of the flow between two sections
- $\boldsymbol{\sigma_t^2(x_i)}$ is the time variance of the distribution
- $\boldsymbol{t_{c(i)}}$ is the time of passage of the centroid of the cloud
- $\boldsymbol{x_i}$ is the position at which the distribution of concentration is measured

$$K = \frac{1}{2}\boxed{U_0^2}\frac{\sigma_t^2(x_{i+1}) - \boxed{\sigma_t^2(x_i)}}{\overline{t_{c(i+1)}} - \boxed{t_{c(i)}}}$$

Mass (M_0, moment of 0 order) $M = \int_{-\infty}^{\infty} C(x_i, t)dt$

function trapz:
M=trapz(time,concentration);

trapz

Trapezoidal numerical integration

collaps

Syntax

```
Q = trapz(Y)
Q = trapz(X,Y)
Q = trapz(___,dim)
```

Description

`Q = trapz(Y)` computes the approximate integral of `Y` via the trapezoidal method with unit spacing. The size of `Y` determines the dimension to integrate along:

- If `Y` is a vector, then `trapz(Y)` is the approximate integral of `Y`.
- If `Y` is a matrix, then `trapz(Y)` integrates over each column and returns a row vector of integration values.
- If `Y` is a multidimensional array, then `trapz(Y)` integrates over the first dimension whose size does not equal 1. The size of this dimension becomes 1, and the sizes of other dimensions remain unchanged.

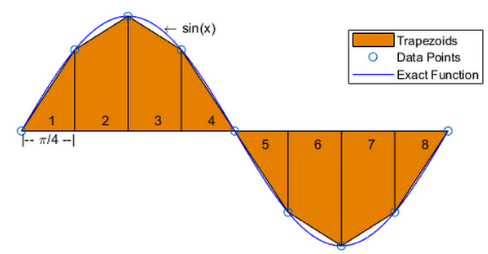
`Q = trapz(X,Y)` integrates `Y` with respect to the coordinates or scalar spacing specified by `X`.

- If `X` is a vector of coordinates, then `length(X)` must be equal to the size of the first dimension of `Y` whose size does not equal 1.
- If `X` is a scalar spacing, then `trapz(X,Y)` is equivalent to `X*trapz(Y)`.

`Q = trapz(___,dim)` integrates along the dimension `dim` using any of the previous syntaxes. You must specify `Y`, and optionally can specify `X`. If you specify `X`, then it can be a scalar or a vector with length equal to `size(Y,dim)`. For example, if `Y` is a matrix, then `trapz(X,Y,2)` integrates each row of `Y`.

Trapezoidal Method

`trapz` performs numerical integration via the trapezoidal method. This method approximates the integration over an interval by breaking the area down into trapezoids with more easily computable areas. For example, here is a trapezoidal integration of the sine function using eight evenly-spaced trapezoids:



For an integration with `N+1` evenly spaced points, the approximation is

$$\int_a^b f(x)dx \approx \frac{b-a}{2N}\sum_{n=1}^{N}(f(x_n)+f(x_{n+1}))$$
$$= \frac{b-a}{2N}[f(x_1)+2f(x_2)+...+2f(x_N)+f(x_{N+1})] ,$$

where the spacing between each point is equal to the scalar value $\frac{b-a}{N}$. By default MATLAB® uses a spacing of 1.

If the spacing between the `N+1` points is not constant, then the formula generalizes to

$$\int_a^b f(x)dx \approx \frac{1}{2}\sum_{n=1}^{N}(x_{n+1}-x_n)[f(x_n)+f(x_{n+1})] ,$$

where $a = x_1 < x_2 < \ ... \ < x_N < x_{N+1} = b$, and $(x_{n+1} - x_n)$ is the spacing between each consecutive pair of points.

Moments Method

$$K = \frac{1}{2}\boxed{U_0^2}\frac{\sigma_t^2(x_{i+1}) - \boxed{\sigma_t^2(x_i)}}{\overline{t_{c(i+1)}} - \boxed{\overline{t_{c(i)}}}}$$

Mass (M_0, moment of 0 order) $M = \int_{-\infty}^{\infty} C(x_i, t)dt$

function trapz:
M=trapz(time,concentration);

Time of passage of the centroid of the cloud across a section (M_1): $\overline{\boldsymbol{t_{c(i)}}} = \frac{\int_{-\infty}^{\infty} tC(x_i,t)dt}{\int_{-\infty}^{\infty} C(x_i,t)dt}$

Time variance of the distribution (M_2): $\boldsymbol{\sigma_t^2(x_i)} = \frac{\int_{-\infty}^{\infty} (t - t_i)^2 C(x_i,t)dt}{\int_{-\infty}^{\infty} C(x_i,t)dt}$

Velocity: $\boldsymbol{U_0} = \frac{x_{i+1} - x_i}{\overline{t_{c(i+1)}} - \overline{t_{c(i)}}}$

CALIBRATION METHOD

Matlab script: Calibration Method

MatLab script: *longitudinalDISP_routingSIMPL.m*

longitudinalDISP_routingSIMPL_SabineRiver_CD.m

This file can be published to a formatted document. For more information, see the publishing video or help.

```matlab
%Clear all variables, globals, functions and MEX links from memory.
clear all
%closes all the open figure windows
close all
%Clear the command window and homes the cursor.
clc

%+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
% Routing methods to determine the Longitudinal Dispersion Coefficient
%+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

% Load and plot observed data.
% Carica i valori misurati della concentrazione nel tempo nelle sezioni 1 e
% 2
% Sezione 1=C
dataC=load('case17_section1.txt');
t_dataC=dataC(:,1)*3600;
C_dataC=dataC(:,2)/1000000;

% Sezione 2=D
dataD=load('case17_section2.txt');
t_dataD=dataD(:,1)*3600;
C_dataD=dataD(:,2)/1000000;

figure(1); hold on
%Plot Observed data
plot(t_dataC/3600,C_dataC,'--b','linewidth',1.5,...
        'Marker','o','MarkerSize',6,...
        'MarkerEdgeColor',[0 0 1],...
        'MarkerFaceColor',[1 1 1]);
plot(t_dataD/3600,C_dataD,'--k','linewidth',1.5,...
        'Marker','d','MarkerSize',6,...
        'MarkerEdgeColor',[1 0 1],...
        'MarkerFaceColor',[1 1 1]);
title('C(x,t)','FontSize',12,'FontWeight','bold')
set(gca,'PlotBoxAspectRatio',[2 1 1]);
axis([min(t_dataC)/3600 max(t_dataD)/3600 0  1.3*max(C_dataC)])
xlabel('time [hours]','FontSize',12,'FontWeight','bold');
ylabel('c(x,t) [mg/l]','FontSize',12,'FontWeight','bold');
set(gca,'FontSize',12,'FontWeight','bold');
legend('section1','section2')
box on
grid on
```

This script plots data of two sections and finds the best fit solution adopting **Frozen Cloud** and **Hayami methods**.

Matlab script: Calibration Method

MatLab script: *longitudinalDISP_routingSIMPL.m*

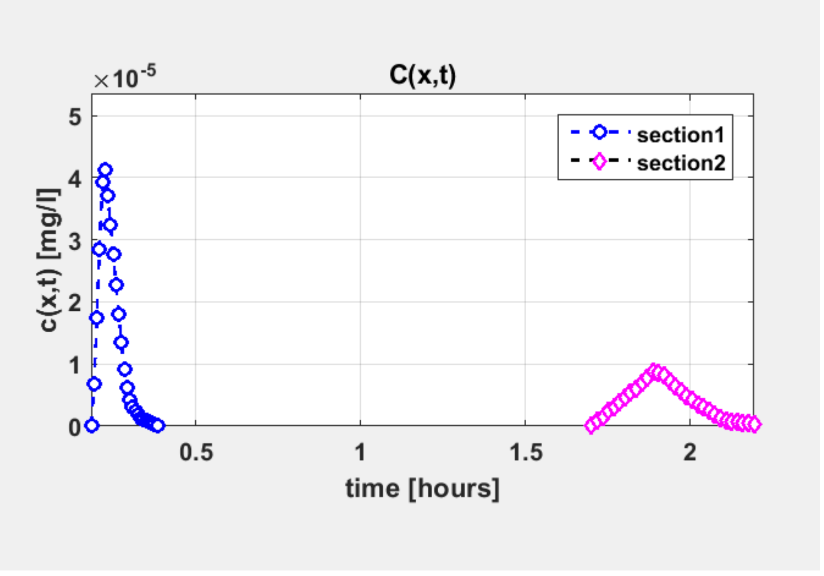
longitudinalDISP_routingSIMPL_SabineRiver_CD.m

This file can be published to a formatted document. For more information, see the publishing video or help.

```
%Clear all variables, globals, functions and MEX links from memory.
clear all
%closes all the open figure windows
close all
%Clear the command window and homes the cursor.
clc

%+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
% Routing methods to determine the Longitudinal Dispersion Coefficient
%+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

% Load and plot observed data.
% Carica i valori misurati della concentrazione nel tempo nelle sezioni 1 e
% 2
% Sezione 1=C
dataC=load('case17_section1.txt');
t_dataC=dataC(:,1)*3600;
C_dataC=dataC(:,2)/1000000;

% Sezione 2=D
dataD=load('case17_section2.txt');
t_dataD=dataD(:,1)*3600;
C_dataD=dataD(:,2)/1000000;

figure(1); hold on
%Plot Observed data
plot(t_dataC/3600,C_dataC,'--b','linewidth',1.5,...
        'Marker','o','MarkerSize',6,...
        'MarkerEdgeColor',[0 0 1],...
        'MarkerFaceColor',[1 1 1]);
plot(t_dataD/3600,C_dataD,'--k','linewidth',1.5,...
        'Marker','d','MarkerSize',6,...
        'MarkerEdgeColor',[1 0 1],...
        'MarkerFaceColor',[1 1 1]);
title('C(x,t)','FontSize',12,'FontWeight','bold')
set(gca,'PlotBoxAspectRatio',[2 1 1]);
axis([min(t_dataC)/3600 max(t_dataD)/3600 0  1.3*max(C_dataC)])
xlabel('time [hours]','FontSize',12,'FontWeight','bold');
ylabel('c(x,t) [mg/l]','FontSize',12,'FontWeight','bold');
set(gca,'FontSize',12,'FontWeight','bold');
legend('section1','section2')
box on
grid on
```

Loading of the measured concentration in different sections

Plot of observed data



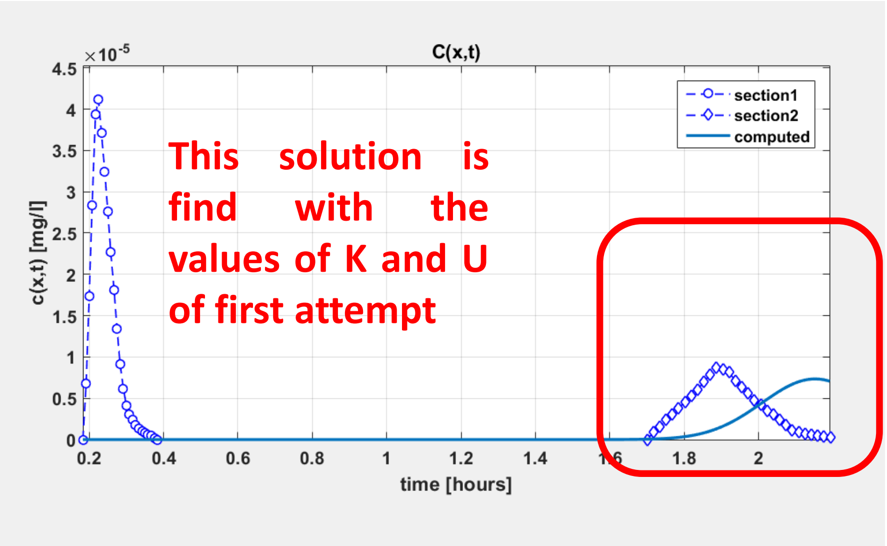
MatLab script: *longitudinalDISP_routingSIMPL.m*

longitudinalDISP_routingSIMPL_SabineRiver_CD.m

This file can be published to a formatted document. For more information, see the publishing video or help.

```
grid on
% pause
% close(1)

%++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
% Definition and initialization of parameters and variables
% Sabine River LA

M = 0.1;                    % mass of dye injected, kg
Kx =9.0;                    % longitudinal dispersion coefficient, m^2/s
U = 0.7;                    % mean current velocity, m/s;
i_f= 17.7e-003;             % channel slope
Do = 0.85;                  % mean depth, m
Bo = 13.7;                  % Average width, m

% Longitudinal Coordinates of sections 1 and 2
xfix=[1130 5950]; % xcoordinate of sites along the Sabine river, m



%**************************************************************************
%**************************************************************************
%Calcola la soluzione al variare di Kx e U, cercando i valori ottimi di
%Kx e U
%--------------------------------------------------------------------------
%Definisce la function che calcola la soluzione
fun=@simpl_CD;
%--------------------------------------------------------------------------
input_par=[Kx;U];
%--------------------------------------------------------------------------
figure(2)
hold on
% [res,fval,exitflag,output] = fminsearch(fun,input_par);
% [res,fval] = fminsearch(fun,input_par);
[res] = fminsearch(fun,input_par);
%-------------------------------------------------------------------------
% Valori ottimi calcolati di Kx e U
Kx =res(1);             % longitudinal dispersion coefficient, m^2/s
U = res(2);             % mean current velocity, m/s;
%**************************************************************************
%**************************************************************************

```

- Definition of parameters and initial values;
- All the parameters are listed in the paper of Nordin and Sabol (1974) .
- For every case study there is a different sets of parameters



Matlab script: Calibration Method

MatLab script: *longitudinalDISP_routingSIMPL.m*

longitudinalDISP_routingSIMPL_SabineRiver_CD.m

This file can be published to a formatted document. For more information, see the publishing video or help.

```
grid on
% pause
% close(1)

%+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
% Definition and initialization of parameters and variables
% Sabine River LA

M = 0.1;                % mass of dye injected, kg
Kx =9.0;                % longitudinal dispersion coefficient, m^2/s
U = 0.7;                % mean current velocity, m/s;
i_f= 17.7e-003;         % channel slope
Do = 0.85;              % mean depth, m
Bo = 13.7;              % Average width, m

% Longitudinal Coordinates of sections 1 and 2
xfix=[1130 5950]; % xcoordinate of sites along the Sabine river, m



%***************************************************************************
%***************************************************************************
%Calcola la soluzione al variare di Kx e U, cercando i valori ottimi di
%Kx e U
%------------------------------------------------------
%Definisce la function che calcola la soluzione
fun=@simpl_CD;
%------------------------------------------------------
input_par=[Kx;U];
%------------------------------------------------------
figure(2)
hold on
% [res,fval,exitflag,output] = fminsearch(fun,input_par);
% [res,fval] = fminsearch(fun,input_par);
[res] = fminsearch(fun,input_par);
%---------------------------------------------------------------------------
% Valori ottimi calcolati di Kx e U
Kx =res(1);             % longitudinal dispersion coefficient, m^2/s
U = res(2);             % mean current velocity, m/s;
%***************************************************************************
%***************************************************************************

```

- Function **fun = simple_CD.m** that evaluates the solution with the calibrated method

MatLab script: *longitudinalDISP_routingSIMPL.m*

longitudinalDISP_routingSIMPL_SabineRiver_CD.m

This file can be published to a formatted document. For more information, see the publishing video or help.

```
grid on
% pause
% close(1)

%++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
% Definition and initialization of parameters and variables
% Sabine River LA

M = 0.1;                % mass of dye injected, kg
Kx =9.0;                % longitudinal dispersion coefficient, m^2/s
U = 0.7;                % mean current velocity, m/s;
i_f= 17.7e-003;         % channel slope
Do = 0.85;              % mean depth, m
Bo = 13.7;              % Average width, m

% Longitudinal Coordinates of sections 1 and 2
xfix=[1130 5950]; % xcoordinate of sites along the Sabine river, m



%****************************************************************************
%****************************************************************************
%Calcola la soluzione al variare di Kx e U, cercando i valori ottimi di
%Kx e U
%-------------------------------------------------------
%Definisce la function che calcola la soluzione
fun=@simpl_CD;
%-------------------------------------------------------
input_par=[Kx;U];
%-------------------------------------------------------
figure(2)
hold on
% [res,fval,exitflag,output] = fminsearch(fun,input_pa
% [res,fval] = fminsearch(fun,input_par);
[res] = fminsearch(fun,input_par);
%-------------------------------------------------------
% Valori ottimi calcolati di Kx e U
Kx =res(1);             % longitudinal dispersion coeff
U = res(2);             % mean current velocity, m/s;
%****************************************************************************
%**********
```

- Function **fun = simple_CD.m** that evaluates the solution with the calibrated method
- **Matlab function that computes iteratively** the local minimum of the function (fun) starting from (**input_par = Kx and U**)
- Calibrated parameters Kx and U

MatLab script: function *simpl_CD.m*

```
% Sezione 1 = C
dataC=load('case17_section1.txt');
t_dataC=dataC(:,1)*3600;
C_dataC=dataC(:,2)/1000000;

% Sezione 2 = D
dataD=load('case17_section2.txt');
t_dataD=dataD(:,1)*3600;
C_dataD=dataD(:,2)/1000000;

    %Calcola i tempi medi di passaggio in 1 e 2
    tmean = xfix/U + 2*Kx/U^2;
    %Intervallo di integrazione
    t_rout = [min(t_dataC):15:max(t_dataD)]';

    %Calcola l'integrale di convoluzione
    for tt=1:length(t_rout)
        t=t_rout(tt);
        %Calcola C(x2,t_rout(tt));
        for ttt=1:length(t_dataC)
            tau=t_dataC(ttt);
            %-------------------------------------------------------------------------
            %Routing a Temporal Concentration Profile with the "Frozen Cloud Method"
            Crouting(ttt)=C_dataC(ttt)*U/sqrt(4*pi*Kx*(tmean(2)-tmean(1)))...
                *exp(-U^2*((tmean(2)-tmean(1)-t+tau)^2)/(4*Kx*(tmean(2)-tmean(1))));
            %-------------------------------------------------------------------------
            %Routing a Temporal Concentration Profile with the "Hayami Solution"
            %-------------------------------------------------------------------------
            %if t>tau
            %    Crouting(ttt)=C_dataC(ttt)*(xfix(2)-xfix(1))/(t-tau)/sqrt(4*pi*Kx*(t-tau))*e:
            %else
            %    Crouting(ttt)=0;
            %end
        end
        CroutedD(tt)=trapz(t_dataC,Crouting);
    end
```

- A For cycle that evaluates a concentration profile with the frozen cloud method.
- Input: values of Kx and U.

$$C(x_2,t)=\int_{-\infty}^{+\infty}\frac{C(x_1,\tau)}{\sqrt[2]{4\pi K(t_2-t_1)}}\cdot e^{-\left[\frac{U^2(t_2-t_1-t+\tau)^2}{4K(t_2-t_1)}\right]}d\tau$$

MatLab script: function *simpl_CD.m*

```
% Seleziona la funzione obiettivo da minimizzare
%choice=1     %Minimizza la somma dei quadrati delle distanze tra la C calcolata e la
              %C misurata nella sezione 2
%choice=2     %Minimizza la distanza tra il picco di concentrazione
              %misurato in 2 ed il picco calcolato
choice =1;
% -----------------------------------------------------------------------------
if choice==1
    %Minimizza la somma dei quadrati delle distanze tra la C calcolata e la
    %C misurata nella sezione 2
    for ttt=1:length(t_dataD)
        tau=t_dataD(ttt);
        %Calcolo il punto della griglia t_rout più
        [delta_t,index]=min(abs(t_rout-tau));
        t_routCONFR(ttt)= t_rout(index);
        CroutedD_CONFR(ttt)=CroutedD(index);
    end
    %Definisce la funzione obiettivo

    f_obiettivo = sum((CroutedD_CONFR-C_dataD').^2)
end

% -----------------------------------------------------------
if choice==2
    %Individua il massimo di Concentrazione misurato
    [maxC_dataD,index_max]=max(C_dataD);
    timeC_dataD=t_dataD(index_max);

    %Individua il massimo di Concentrazione calcolat
    [maxCroutedD,index_max]=max(CroutedD);
    timeCroutedD=t_rout(index_max);

    %Definisce la funzione obiettivo:

    f_obiettivo = sqrt((maxC_dataD-maxCroutedD)^2+..
        (timeC_dataD-timeCroutedD)^2)
end
```

- The function simple_CD.m is a target function.
- TARGET: minimizes the sum of squares of the distances between observed and computed data
- CHOICE 1: the concentration distribution is used.
- CHOICE 2: the peak of concentration is used.

MatLab script: *longitudinalDISP_routingSIMPL.m*

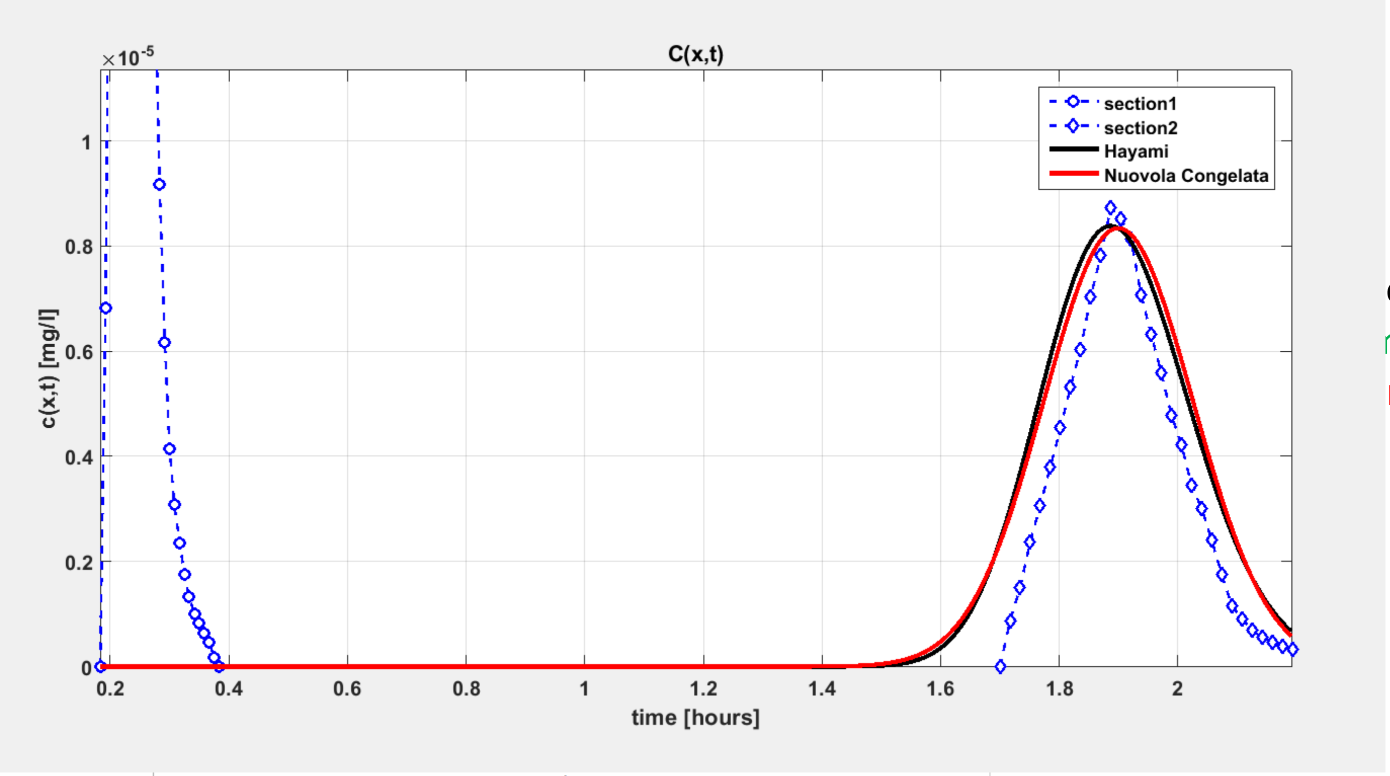
```
%Calcola i tempi medi di passaggio in 1 e 2
tmean=xfix/U + 2*Kx/U^2;
t_rout=[min(t_dataC):15:max(t_dataD)]';
for tt=1:length(t_rout)
    t=t_rout(tt);
    % Calcola C(x2,t_rout(tt));
    for ttt=1:length(t_dataC)
        tau=t_dataC(ttt);

        %Routing a Temporal Concentration Profile with the "Hayami Solution"
        %-----------------------------------------------------------------------
        if t>tau
            Crouting(ttt)=C_dataC(ttt)*(xfix(2)-xfix(1))/(t-tau)/...
                sqrt(4*pi*Kx*(t-tau))*exp(-(xfix(2)-xfix(1)-U*...
                (t-tau))^2/(4*Kx*(t-tau)));
        else
            Crouting(ttt)=0;
        end
    end
    CroutedD(tt)=trapz(t_dataC,Crouting);
end
CroutedDplot=CroutedD';
```

Last part of the script plots the calculated C distribution by Frozen Cloud and Hayami methods adopting calibrated parameters

```
%Calcola i tempi medi di passaggio in 1 e 2
tmean=xfix/U + 2*Kx/U^2;
t_rout=[min(t_dataC):15:max(t_dataD)]';
for tt=1:length(t_rout)
    t=t_rout(tt);
    %Calcola C(x2,t_rout(tt));
    for ttt=1:length(t_dataC)
        tau=t_dataC(ttt);
        %-----------------------------------------------------------------------
        %Routing a Temporal Concentration Profile with the "Frozen Cloud Method"
        Crouting(ttt)=C_dataC(ttt)*U/sqrt(4*pi*Kx*(tmean(2)-tmean(1)))*...
            exp(-U^2*((tmean(2)-tmean(1)-t+tau)^2)/...
            (4*Kx*(tmean(2)-tmean(1))));
        %-----------------------------------------------------------------------
    end
    CroutedD(tt)=trapz(t_dataC,Crouting);
end

CroutedDplot=CroutedD';
```

MatLab script: *longitudinalDISP_routingSIMPL.m*

```
%Calcola i tempi medi di passaggio in 1 e 2
tmean=xfix/U + 2*Kx/U^2;
t_rout=[min(t_dataC):15:max(t_dataD)]';
for tt=1:length(t_rout)
    t=t_rout(tt);
    % Calcola C(x2,t_rout(tt));
```




```
    CroutedD(tt)=trapz(t_dataC,Crouting);
end

CroutedDplot=CroutedD';
```

ed C
ni
rs