### Matlab integration tool

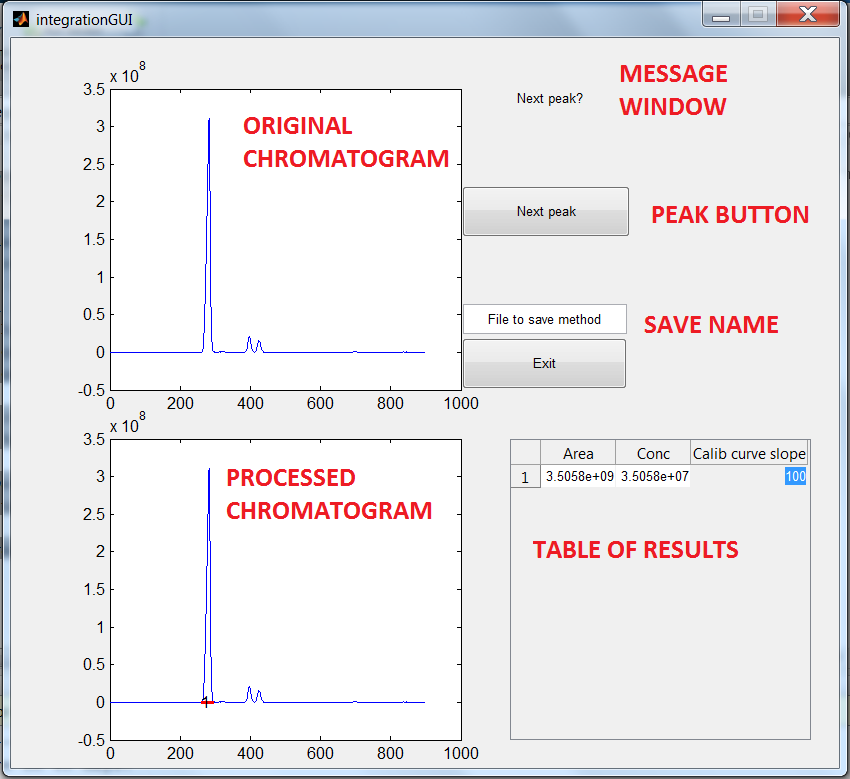
The Matlab integration tool was written to help processing the excel files saved out by the code. A graphical user interfase (GUI) was crafted for easier use. Two files were submitted, ‘integrationGUI.m’ manually processes excel measurement files generated by the HPLC software, and saves out an excel report. The ‘listintegration.m’ file uses an excel report and analyses every measurement excel file in the folder, saving the results in a report file per measurement file. Both the raw Matlab code and an executable was supplied for both m files. Executables can be run with Matlab Runtime, a freeware supplied by MathWorks, so it can be used on any computer.

#### integrationGUI.m

Once the the program is started, it prompts the user to choose the measurement excel file. When this is done the upper graph shows the original chromatogram, while the lower graph thecurrent state of the processed one. The results are summarized in a table in the lower right corner. As a peak is recognized the calibration slope (area of 1 g/l concentration peak) can be input anytime and the concentration calculated instantly. The next peak button can be pressed for new peak recognition, the desired report name set in the stringbox and the exit button pressed to terminate the program. The software gives hints for the user in the upper right corner.

After start the next peak button has to be pressed and the message box shows that the left side of the peak has to be shown. After the user clicks on the left side (only the X position of the curser counts), the message box asks for the right side. When this is marked by clicking, the area is shown instantly, the peak baseline underlined with a red line and the index of the peak visualized. The only important thing when marking the boundaries is that the hgihgest point should be the top of the desired peak. The Algorithm searches for the maximum value, finds the first valley before and after it (it is a sesnsitive method for bad datasets, but for the current setup it was tested and approved), fits a linear to it and calculates the area of the peak above it.

Once the next peak button is pressed the cycle starts again. The desired report name should be inserted in the stringbox, and by pressing the exit button the report file is generated and the software terminates. Caution, if no name is set, no file is saved!



The report excel file contains the peak area, concentration, calibration slope, time of peak start and end for every peak. Besides that it also saves the last state of the complete GUI, showing the original and processed graph and the results table. A report file is also the method file for integration.



##### Matlab raw code

function varargout = integrationGUI(varargin)

% INTEGRATIONGUI MATLAB code for integrationGUI.fig

% INTEGRATIONGUI, by itself, creates a new INTEGRATIONGUI or raises the existing

% singleton\*.

%

% H = INTEGRATIONGUI returns the handle to a new INTEGRATIONGUI or the handle to

% the existing singleton\*.

%

% INTEGRATIONGUI('CALLBACK',hObject,eventData,handles,...) calls the local

% function named CALLBACK in INTEGRATIONGUI.M with the given input arguments.

%

% INTEGRATIONGUI('Property','Value',...) creates a new INTEGRATIONGUI or raises the

% existing singleton\*. Starting from the left, property value pairs are

% applied to the GUI before integrationGUI\_OpeningFcn gets called. An

% unrecognized property name or invalid value makes property application

% stop. All inputs are passed to integrationGUI\_OpeningFcn via varargin.

%

% \*See GUI Options on GUIDE's Tools menu. Choose "GUI allows only one

% instance to run (singleton)".

%

% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help integrationGUI

% Last Modified by GUIDE v2.5 10-Nov-2016 01:35:28

% Begin initialization code - DO NOT EDIT

gui\_Singleton = 1;

gui\_State = struct('gui\_Name', mfilename, ...

'gui\_Singleton', gui\_Singleton, ...

'gui\_OpeningFcn', @integrationGUI\_OpeningFcn, ...

'gui\_OutputFcn', @integrationGUI\_OutputFcn, ...

'gui\_LayoutFcn', [] , ...

'gui\_Callback', []);

if nargin && ischar(varargin{1})

gui\_State.gui\_Callback = str2func(varargin{1});

end

if nargout

[varargout{1:nargout}] = gui\_mainfcn(gui\_State, varargin{:});

else

gui\_mainfcn(gui\_State, varargin{:});

end

% End initialization code - DO NOT EDIT

% --- Executes just before integrationGUI is made visible.

function integrationGUI\_OpeningFcn(hObject, eventdata, handles, varargin)

% This function has no output args, see OutputFcn.

% hObject handle to figure

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

% varargin command line arguments to integrationGUI (see VARARGIN)

[FileName,PathName] = uigetfile('\*.xls\*'); %asks for file

num = xlsread(strcat(PathName,FileName),'A:B'); %read it in

%num = xlsread('C:\Users\Valentin\Desktop\New folder\SRS3.xlsx','A:B'); %

%for debugging

handles.PathName=PathName;

handles.num=num;

handles.peak=1;

axes(handles.axes1)

plot(num(:,2),num(:,1)); %draws initial plot

%

axes(handles.axes2)

cla;

plot(num(:,2),num(:,1)); %draw second plot

handles.Data=[0 0 0]; %creates initial datafile

handles.peakData=[0 0]; %initialize

set(handles.text1,'String','Press Next peak button'); %adds name

% Choose default command line output for integrationGUI

handles.output = hObject;

% Update handles structure

guidata(hObject, handles);

% UIWAIT makes integrationGUI wait for user response (see UIRESUME)

% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.

function varargout = integrationGUI\_OutputFcn(hObject, eventdata, handles)

% varargout cell array for returning output args (see VARARGOUT);

% hObject handle to figure

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure

varargout{1} = handles.output;

% --- Executes on button press in Exit.

function Exit\_Callback(hObject, eventdata, handles)

% hObject handle to Exit (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

% save if name is given

methodname=handles.methodname; %gets name

PathName=handles.PathName;

peakData=handles.peakData;

Data=handles.Data;

if strcmp(get(methodname,'String'),'File to save method')

else

%saves out results

xlswrite(strcat(PathName,get(methodname,'String'),'.xlsx'),[{'Peak Area'} {'Concetration'} {'Calibration'} {'Peak start'} {'Peak end'}],'A1:E1');

no=size(Data,1);

xlswrite(strcat(PathName,get(methodname,'String'),'.xlsx'),[Data peakData],strcat('A2:E',num2str(1+no)));

axes(handles.axes2)

print -dmeta; %.................Copying to clipboard

FILE = strcat(PathName,get(methodname,'String'),'.xlsx');

Range ='G3';

%.............excel COM object............................................................................

Excel = actxserver ('Excel.Application');

Excel.Visible = 1;

if ~exist(FILE,'file')

ExcelWorkbook=Excel.Workbooks.Add;

ExcelWorkbook.SaveAs(FILE);

ExcelWorkbook.Close(false);

end

invoke(Excel.Workbooks,'Open',FILE); %Open the file

ActiveSheet = Excel.ActiveSheet;

ActiveSheetRange = get(ActiveSheet,'Range',Range);

ActiveSheetRange.Select;

ActiveSheetRange.PasteSpecial; %.................Pasting the figure to the selected location

end

close all force

% --- Executes during object creation, after setting all properties.

function methodname\_CreateFcn(hObject, eventdata, handles)

% hObject handle to methodname (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.

% See ISPC and COMPUTER.

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))

set(hObject,'BackgroundColor','white');

end

% --- Executes during object creation, after setting all properties.

function axes1\_CreateFcn(hObject, eventdata, handles)

% hObject handle to axes1 (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles empty - handles not created until after all CreateFcns called

% Hint: place code in OpeningFcn to populate axes1

% --- Executes during object creation, after setting all properties.

function text1\_CreateFcn(hObject, eventdata, handles)

% hObject handle to text1 (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles empty - handles not created until after all CreateFcns called

% --- Executes during object creation, after setting all properties.

function uitable\_CreateFcn(hObject, eventdata, handles)

% hObject handle to uitable (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles empty - handles not created until after all CreateFcns called

function methodname\_Callback(hObject, eventdata, handles)

% hObject handle to Nextpeak (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% --- Executes on button press in Nextpeak.

function Nextpeak\_Callback(hObject, eventdata, handles)

% hObject handle to Nextpeak (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

%start peak find

peak=handles.peak; %access values between callbacks

num=handles.num;

peakData=handles.peakData;

for i=1:2

%hold on

datacursormode on %adds cursor

if i==1

set(handles.text1,'String',strcat('Choose ',num2str(peak), ' peak left side'))

else

set(handles.text1,'String',strcat('Choose ',num2str(peak), ' peak right side'))

end

peakboundaries(peak,i,:) = ginput(1); %waits for boundaries

end

set(handles.text1,'String',strcat('Next peak?'));

%find the peak between two boundaries

[valleys,locsv]=findpeaks(-num(:,1)); %all the valleys in dtaset

indexstart=find(num(:,2)>peakboundaries(peak,1,1),1,'first'); %gets position of first datapoint

indexend=find(num(:,2)>peakboundaries(peak,2,1),1,'first'); % position of mast datapoint

[peaks,locs]=findpeaks(num(indexstart:indexend), 'SORTSTR','descend'); % the peak inbetween boundearies in descending order

leftside=locsv(find(locsv<(locs(1)+indexstart),1,'last')); %finds first valley before peak

rightside=locsv(find(locsv>(locs(1)+indexstart),1,'first')); %finds last valley after peak

%integrate

base=polyfit([num(leftside,2) num(rightside,2)],[num(leftside,1) num(rightside,1)],1); %fit linear between rigth and left side valleys

%first is slope, second intersection

%numerical integration

%integrates area and subtracts baseline

peakarea=trapz(num(leftside:rightside,2),num(leftside:rightside,1))-trapz([num(leftside,2) num(rightside,2) ],[num(leftside,1) num(rightside,1)]);

peakData(peak,:)=[num(leftside,2) num(rightside,2)];

handles.peakData=peakData;

%write values, updata Data

Data=handles.Data;

Data(peak,:)=[peakarea 0 0];

set(handles.uitable,'Data',Data);

handles.Data=Data;

line=num(leftside:rightside,2)\*base(1)+base(2);

%plot peakfind

axes(handles.axes2)

hold on

hline=plot(num(leftside:rightside,2),line,'Color',[1 0 0],'LineWidth',2);

text(num(leftside,2),min(num(leftside:rightside,1)),num2str(peak));

handles.hline=hline;

handles.peak=peak+1;

guidata(hObject, handles);

% --- Executes when entered data in editable cell(s) in uitable.

function uitable\_CellEditCallback(hObject, eventdata, handles)

% hObject handle to uitable (see GCBO)

% eventdata structure with the following fields (see UITABLE)

% Indices: row and column indices of the cell(s) edited

% PreviousData: previous data for the cell(s) edited

% EditData: string(s) entered by the user

% NewData: EditData or its converted form set on the Data property. Empty if Data was not changed

% Error: error string when failed to convert EditData to appropriate value for Data

% handles structure with handles and user data (see GUIDATA)

Indices=eventdata.Indices; %gets place of change

NewData=eventdata.NewData; %gets value of change

Data=handles.Data;

Data(Indices(1),2)=Data(Indices(1),1)/NewData; %calculates conc

Data(Indices(1),3)=NewData; %saves the slope

handles.Data=Data;

set(handles.uitable,'Data',Data); %write it to table

guidata(hObject, handles); %updates changes

function uitable\_DeleteFcn(varargin)

#### Listintegration

This program was written so the integration could be done automaticaly if the same type of measurements are performed (the retention times are the same, more importantly the peak heights are in between the same boundaries). All the measurement excel files should be in he same location together with at least one report excel file, additional report files are not taken into account, so they can be present. When the script is launched it asks for a report file, thereafter it lists all the nmeasurement files in the folder, and uses the retention time data and the calibration values to analyze all the files based on the report file, and generate a report file for each measurement file. As the manual GUI is not launched, the chromatograms are not included int he report excel spreadsheets. Suspicius results should be double checked with the manual integration tool.

##### Matlab raw code

[FileName,PathName] = uigetfile('\*.xls\*'); %choose methodfile

num = xlsread(strcat(PathName,FileName)); %read in method file

peakboundaries=num(:,4:5); %copy boundaries

calib=num(:,3);

cd(PathName); %change active folder

files=dir('\*.xls\*'); %list files

%% Read in only measurement files

clear datafile

counter=1;

for i=1:length(files)

buffer=xlsread(files(i).name);

if size(buffer,2)==5

else

datafile{counter}=buffer;

datafilename{counter}=files(i).name;

counter=counter+1;

end

clear buffer;

end

%% Integrate between boundaries, plot

for ii=1:length(datafile) %every file

display(datafilename(ii)) %write in command window which file is processed

figure('Name',datafilename{ii},'NumberTitle','off')

data=datafile{ii};

plot(data(:,2),data(:,1));

hold on

for jj=1:length(peakboundaries) %every peak

[valleys,locsv]=findpeaks(-data(:,1)); %all the valleys in dtaset

indexstart=find(data(:,2)>peakboundaries(jj,1),1,'first');

indexend=find(data(:,2)>peakboundaries(jj,2),1,'first');

%find peak

[peaks,locs]=findpeaks(data(indexstart:indexend), 'SORTSTR','descend'); % the peak inbetween boundearies in descending order

%locate two sides

leftside=locsv(find(locsv<(locs(1)+indexstart),1,'last'));

rightside=locsv(find(locsv>(locs(1)+indexstart),1,'first'));

%integrate

base=polyfit([data(leftside,2) data(rightside,2)],[data(leftside,1) data(rightside,1)],1);

peakarea=trapz(data(leftside:rightside,1),data(leftside:rightside,2))-trapz([data(leftside,1) data(rightside,1) ],[data(leftside,1) data(rightside,1)]);

%write values

% area conc calib left right

peakData(jj,:)=[peakarea 0 calib(jj) data(leftside,2) data(rightside,2)];

if calib jj~=0

peakData(jj,2)=peakData(jj,1)/peakData(jj,3);

end

%plot baseline

line=data(leftside:rightside,2)\*base(1)+base(2);

hline=plot(data(leftside:rightside,2),line,'Color',[1 0 0],'LineWidth',2);

hold on

text(data(leftside,2),min(data(leftside:rightside,1)),num2str(jj));

end

%Save to excel

%values

xlswrite(strcat(PathName,'Results\_',datafilename{ii}),[{'Peak Area'} {'Concetration'} {'Calibration'} {'Peak start'} {'Peak end'}],'A1:E1');

no=size(peakboundaries,1);

xlswrite(strcat(PathName,'Results\_',datafilename{ii}),peakData,strcat('A2:E',num2str(1+no)));

%plot

print -dmeta; %.................Copying to clipboard

FILE = strcat(PathName,'Results\_',datafilename{ii});

Range ='G3';

%.............excel COM object............................................................................

Excel = actxserver ('Excel.Application');

Excel.Visible = 1;

if ~exist(FILE,'file')

ExcelWorkbook=Excel.Workbooks.Add;

ExcelWorkbook.SaveAs(FILE);

ExcelWorkbook.Close(false);

end

invoke(Excel.Workbooks,'Open',FILE); %Open the file

ActiveSheet = Excel.ActiveSheet;

ActiveSheetRange = get(ActiveSheet,'Range',Range);

ActiveSheetRange.Select;

ActiveSheetRange.PasteSpecial; %.................Pasting the figure to the selected location

Excel.ActiveWorkbook.Save;

clear data

end

%kill excell

system('taskkill /F /IM EXCEL.EXE');