

CZECH TECHNICAL UNIVERSITY IN PRAGUE
FACULTY OF CIVIL ENGINEERING
GEODESY, CARTOGRAPHY AND GEOINFORMATICS



BACHELOR THESIS
Smoothing of spectrometric data

Author: Petra Millarová
Advisor: prof. Ing. Aleš Čepek, CSc.
Prague, 2017



ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: Millarová Jméno: Petra Osobní číslo: 439263

Zadávací katedra: Katedra geomatiky

Studijní program: Geodézie a kartografie

Studijní obor: Geodézie, kartografie a geoinformatika

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce: Vyhlazování spectrometrických dat

Název bakalářské práce anglicky: Smoothing of spectrometric data

Pokyny pro vypracování:

Popište metodu vyhlazování spektrometrických dat Savitzky-Golay Filtering. Napište software v C++ (sadu dílčích programů), který bude demonstrovat metodu na příkladech spektrometrických měření, porovnejte ji se zpracováním nefiltrovaných dat pomocí programu qpdata a dále s vybranými dalšími metodami používanými pro filtrování dat.

Seznam doporučené literatury:

Savitzky, A.; Golay, M.J.E. (1964). "Smoothing and Differentiation of Data by Simplified Least Squares Procedures". Analytical Chemistry. 36 (8): 1627–39. doi:10.1021/ac60214a047.

Čepek, A., & Pytel, J. (2009). A note on numerical solutions of least squares adjustment in GNU project gama. In Interfacing Geostatistics and GIS (pp. 173-187). Springer Berlin Heidelberg.

Čepek, A. "qpdata - Qt based experimental program for processing spectral data".


<https://sourceforge.net/projects/qpdata/>

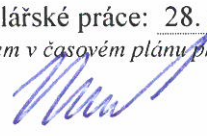
Jméno vedoucího bakalářské práce: prof. Ing. Aleš Čepek, CSc.

Datum zadání bakalářské práce: 22. 2. 2017

Termín odevzdání bakalářské práce: 28. 5. 2017

Údaj uveďte v souladu s datem v časovém plánu příslušného ak. roku


Podpis vedoucího práce


Podpis vedoucího katedry

III. PŘEVZETÍ ZADÁNÍ

Beru na vědomí, že jsem povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je nutné uvést v bakalářské práci a při citování postupovat v souladu s metodickou příručkou ČVUT „Jak psát vysokoškolské závěrečné práce“ a metodickým pokynem ČVUT „O dodržování etických principů při přípravě vysokoškolských závěrečných prací“.

23.2.2017

Datum převzetí zadání



Podpis studenta(ky)

ABSTRACT

The aim of this thesis is to describe the Savitzky-Golay filtering method for smoothing of spectrometric data and to compare its properties with another filtering method, the moving average. The computation of derivatives is also included, along with their use in data comparison. This work also demonstrates the method on a series of spectrometric measurements with our own software and compares it with processing of unfiltered data with the use of the *QSdata* program developed by prof. Ing. Aleš Čepěk, CSc.

KEY WORDS

spectrometry, Savitzky-Golay filter, data smoothing, C++, derivatives

ABSTRAKT

Cílem práce je popsat metodu vyhlazování spektrometrických dat Savitzky-Golay filtrem a porovnat její vlastnosti s dalším filtrem, s klouzavým průměrováním. Součástí je i výpočet derivací a jejich využití pro porovnávání dat. Tato práce také pomocí vlastního software demonstruje metodu na příkladech spektrometrických měření a porovnává ji se zpracováním nefiltrovaných dat pomocí programu *QSdata*, vyvinutého prof. Ing. Alešem Čepkem, CSc.

KLÍČOVÁ SLOVA

spektrometrie, Savitzky-Golay filtr, vyhlazování dat, C++, derivace

I hereby declare that I have written this thesis on Smoothing of spectrometric data by myself and only using the literature provided in the list of references at the end of this work.

Prohlašuji, že jsem práci na téma Vyhlazování spektrometrických dat vypracovala samostatně a pouze za použití literatury uvedené na konci textu.

In Prague, 29th May 2017

.....
Petra Millarová

Acknowledgements

I would like to thank everybody who has helped me in the past couple of months, especially my advisor, prof. Ing. Aleš Čeppek, CSc. and all the people around me that supported me. Last but not least I would like to thank myself for having enough self-discipline to actually finish this work.

Petra Millarová

Contents

1	Introduction	1
2	The Savitzky-Golay filter	2
2.1	Derivative computation	2
2.2	Properties of the filter	4
3	Spectrometric data	8
3.1	Equipment	8
3.2	<i>QSdata</i>	8
4	Development of <i>SGF.exe</i>	10
4.1	Analysis	10
4.2	Implementation	11
4.2.1	Graphic representation of the results	12
5	Comparison of data performance using <i>QSdata</i>	13
5.1	Etalons and samples	14
5.2	Derivatives	17
6	Conclusion	19
6.1	Further improvements	19
	References	20
	List of figures	22
	List of tables	23
	List of used abbreviations	23
	List of used programs	24

1 Introduction

Spectrometric data usually contains a lot of noise, which makes processing the data more difficult. One of the possible ways to eliminate this noise is to filter the data.

This thesis aims to describe the Savitzky-Golay filtering (also known as DISPO – digital smoothing polynomial – filtering). This filter was chosen for the purpose of our work because it preserves higher order moments and results in less distortion than for example moving average filtering [8], as well as reduces white noise [3], which is desirable in our case and it is also the reason why it's one of the more popular filters in spectrometry.

The Savitzky-Golay algorithm is also great for computing derivatives which can be used for identifying extremes and other properties of smoothed data.

The aim of this work is also to produce a program that smooths data with a filter of a chosen length and polynomial degree. This data will then be used in the *QSdata* program and the results compared to the results of unfiltered data.

Theoretically, the smoothed data should give off better results than the unsmoothed data, first order derivatives will also be compared.

The Savitzky-Golay filter, first introduced in paper [14], is based on local approximation by least square polynomials. If we have a set of samples $y[n]$ centred around $n=0$, the coefficients of the polynomial would then be:

$$\sum_{k=0}^N c_k y[n]^k = p(n), \quad (1.1)$$

N being the order of the polynomial and $N+1$ the number of coefficients [16]. The final smoothed value is equal to the 0th coefficient and is obtained by evaluating the polynomial at $n=0$:

$$y^*[0] = p(0) = c_0. \quad (1.2)$$

Let's consider M the half width of the approximation interval, $2M+1$ therefore being the width of the interval, including the point that is being smoothed. The next sample can then be obtained by shifting the center of the interval by one point ($n+1$) and repeating the process, finding a new polynomial every time and evaluating a new $y^*[n]$.

However, the paper [14] also shows that fitting a polynomial to a set of samples and then evaluating it at a single point within the interval is effectively the same as discrete convolution with a set impulse response, since the coefficients of the polynomial are linear in respect to the data. A fixed set of linear coefficients for a given polynomial order N and approximation interval $2M+1$ can then be obtained and make the smoothing more efficient.

2 The Savitzky-Golay filter

As it was stated above, the smoothed data can be computed by discrete convolution of the shifted impulse response [16]

$$y^*[n] = \sum_{m=n-M}^{n+M} h[n-m]y[m]. \quad (2.1)$$

The normal equations for this problem can be written in matrix form [13] as

$$(A^T A)c = A^T y, \quad (2.2)$$

where y is a column vector of input samples of the size $(2M+1) \times 1$, A is the design matrix of size $(2M+1) \times (N+1)$ and its elements are calculated as

$$a_{k,i} = k^i, \quad (2.3)$$

where $i = 0, 1, 2, \dots, N$ and $k \in \langle -M; M \rangle$. The vector of polynomial coefficients can now be written as follows:

$$c = (A^T A)^{-1} A^T y. \quad (2.4)$$

Then we can declare matrix H :

$$H = (A^T A)^{-1} A^T \quad (2.5)$$

and we get

$$c = Hy, \quad (2.6)$$

Which is composed of elements $h[i]$.

Since the matrix A is only dependent on the filter half-length M and the order of the smoothing polynomial N , we can calculate this matrix prior to the actual smoothing.

2.1 Derivative computation

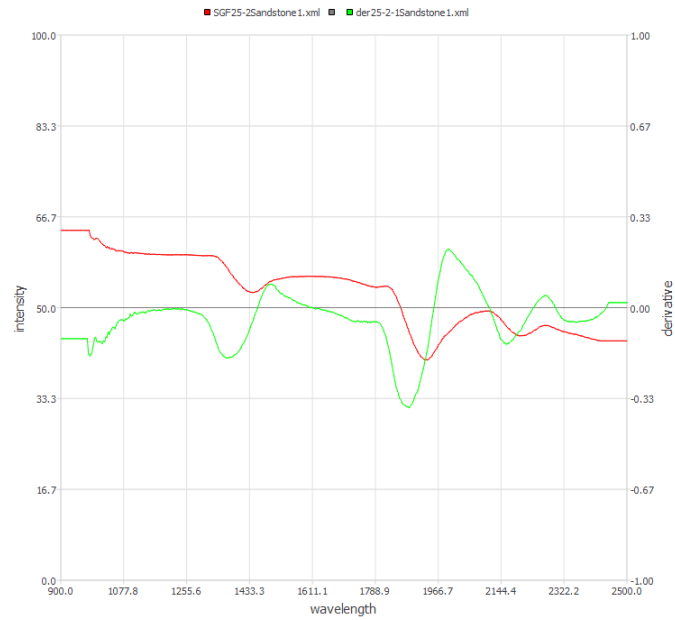
The Savitzky-Golay filter can also be used as a differentiation algorithm [11] [6].

$$y^s[n] = \sum_{m=n-M}^{n+M} h[s, n-m]x[m] \quad (2.7)$$

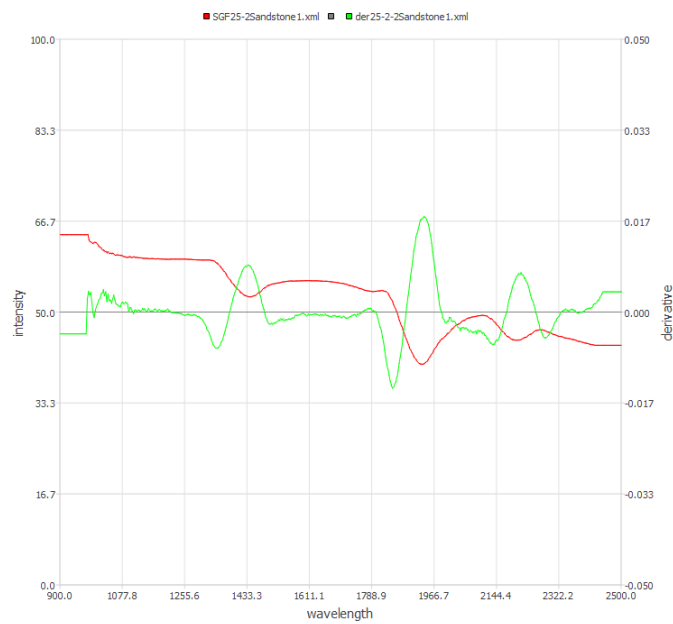
As we can see in the equation 2.7 [17], to calculate the s th derivative of a smoothed spectra (provided that the row numbering starts from zero), one only has to use the s th row of the polynomial coefficients matrix, this is another advantage of this particular method.

If we look at the first derivative of a set of spectrometric data in Fig. 2.1a, we can see that where, for example, is a maximum in the original data (in red), there is a downward zero-crossing in the first derivative and similarly an upward zero-crossing in a minimum, this property is used in peak detection [11]. For the detection of inflection points, we must look at the maximums in the first derivative and if they correspond with any zero-crossings in the second derivative, as we can see in Fig. 2.1b

In spectroscopy, the derivative methods are also used on similar spectra to find the small differences in their course and distinguish between them more easily [11]. Additionally, the fourth derivative can be for example also used for trace analysis [11], especially in the pharmaceutical industry. It is much more useful for detecting slight peaks than finding etalons that match the given sample, which is what this thesis is concerned with.



(a)



(b)

Fig. 2.1: Smoothed data and its derivatives

2.2 Properties of the filter

The Savitzky-Golay filter is a type of a digital filter - it can be applied to a discrete representation of a continuous set of data points to smooth them. It is also a finite response filter (or FIR), which means that its impulse response settles to zero in finite time. As a low-pass filter, it passes signals lower than the cutoff frequency, but unlike some of the other filters it has its properties defined in the time domain rather than having to translate them from the Fourier domain first [13].

The nominal cutoff frequency $f_c = \omega_c/\pi$ depends on the polynomial order N as well as on the window size M . The higher the polynomial order, the wider the passband of the filter (the range of frequencies that can pass through). For our purposes and the passband frequency of 3dB, the frequency can be approximately expressed as

$$f_c = \frac{N + 1}{3.2M - 4.6}, \quad (2.8)$$

assuming that $M \geq 25$ and $N < M$ [15]. This allows us to satisfyingly measure the cutoff frequency in relation to variables M and N . In Fig. 2.2 you can see that the larger the filter half-length and the smaller the polynomial, the narrower the passband, i.e. the filter is less effective with this configuration, resulting in values that are not accurate enough.

We can demonstrate this property on a close-up of a series of data. As you can see in

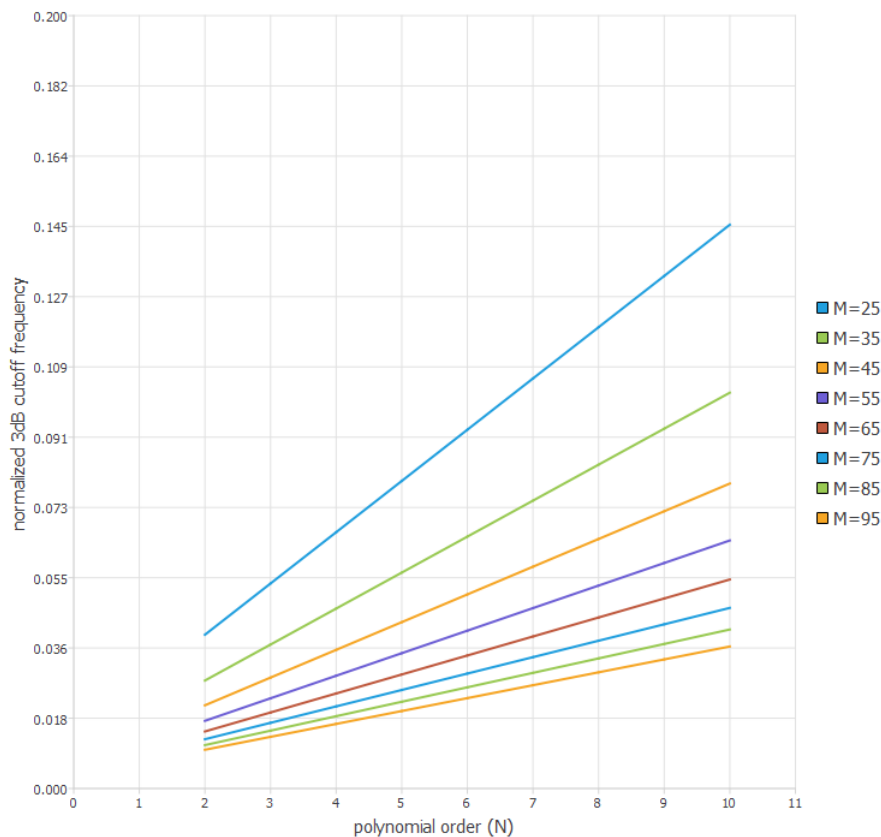


Fig. 2.2: Relation between polynomial order N , filter half-length M and the normalized cutoff frequency

Fig. 2.3, the higher the polynomial order the less smoothed the curve is, given that the filter length is fixed. Similarly, the shorter the filter length is, the more distorted the data will be, as seen in Fig. 2.4. However, if the chosen filter length is too long, the filter clearly oversimplifies the data.

Paper [13] suggests that, as a rough guideline, when we have a fourth degree polynomial filter, the best results are obtained with the full length between 1 and 2 times the full width at half of maximum of the desired features in the data. The higher order filters tend to tackle narrow features better, but at then the broader features are less smoothed. Lower order polynomials are usually used so that the low frequency characteristics of the spectrum are preserved while the higher frequency noise is lost in the approximation error [8].

The Savitzky-Golay filter with the polynomial order of $N=0$ is identical to the moving average filter (MA) [15]:

$$y^*[n] = \sum_{m=n-M}^{n+M} \frac{y[m]}{2M+1}. \quad (2.9)$$

Any symmetrical smoothing window preserves the zeroth and first moment (the area and retention time/location) of the peaks [5]. However, for example the moving average filter tends to filter out higher frequencies of the data [18]. As it was already mentioned in section 1, one of the main properties of the Savitzky-Golay filter is its ability to preserve moments of a higher order [8]. More specifically, filters of order N preserve all the moments up to $N+1$. As a result of this, the Savitzky-Golay filter preserves properties such as the height of the peak [10] or its width [18]. This property is demonstrated in Fig. 2.5, where is is clear that the SGF is better at preserving the shape of the peak than a moving average filter of the same length.

Coefficients of the impulse response that have odd indexes are all equal to zero. [16] This means, among other things, that smoothing with a polynomial of degree N and $N+1$ produces the same results assuming N is an even integer. As you can see in Fig. 2.6, the data smoothed with $M = 20$ and $N = 2$ equals to data smoothed with $N = 3$ and the same filter half-length.

Since we can see that $h[n] = h[-n]$, the impulse response is symmetric and therefore purely real [15].

This work deals with equally spaced data, but if the data were irregularly sampled, one would, instead of computing the convolution coefficients only once, have to compute them and preform least square fitting on each point separately. This would significantly slow down the computing time for larger values of filter width M and polynomial order N .

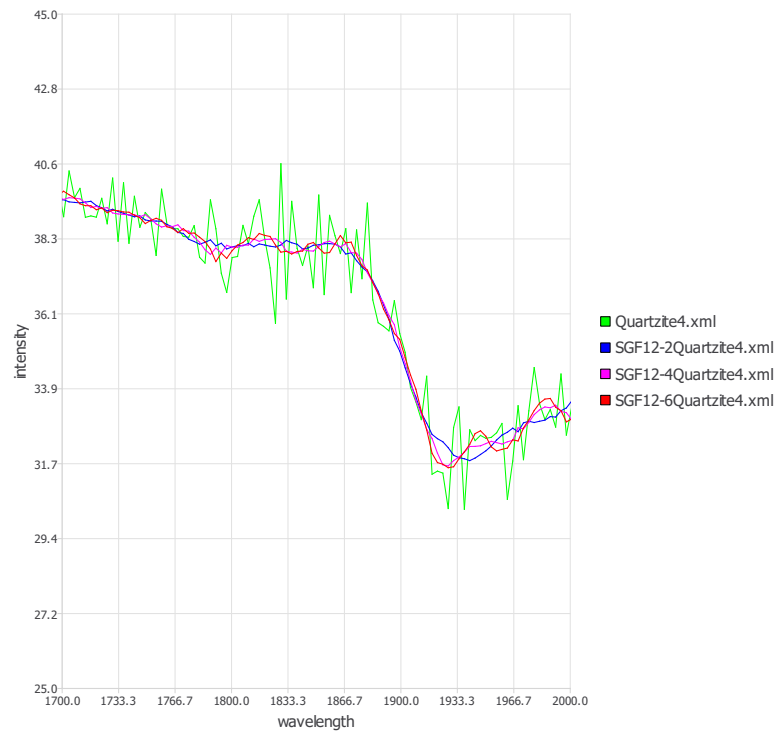


Fig. 2.3: The results for smoothing with different polynomial orders with a fixed filter length.

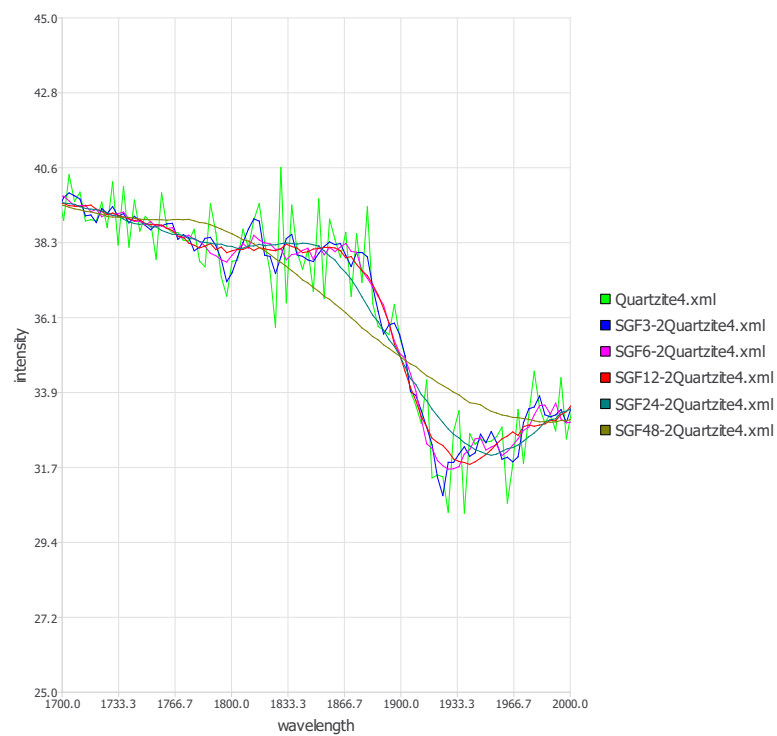


Fig. 2.4: The results for smoothing with different filter lengths with a fixed polynomial order.

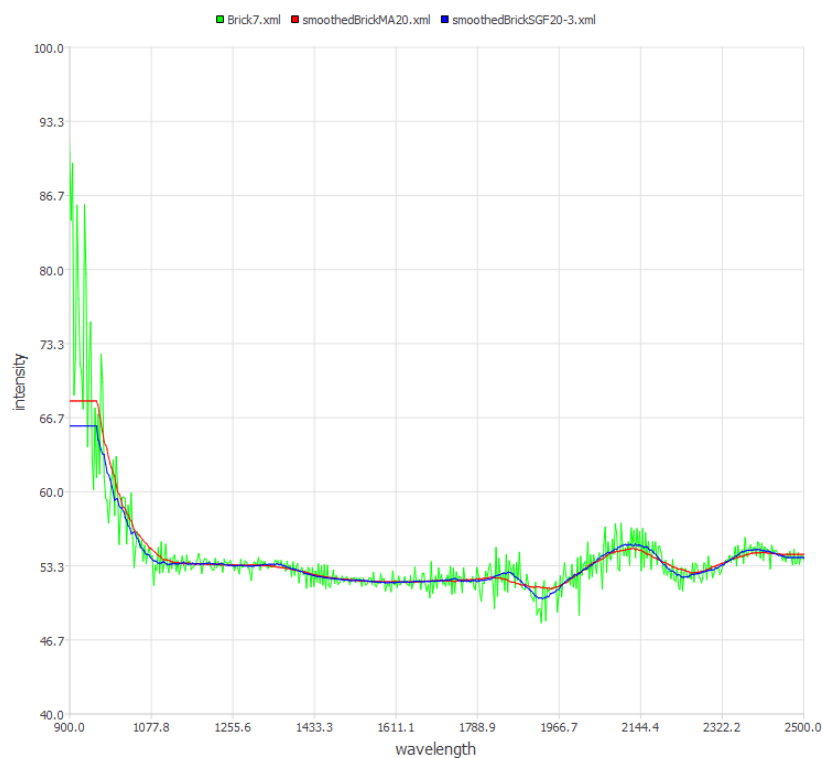


Fig. 2.5: Comparison of MA and SGF filters of the same length

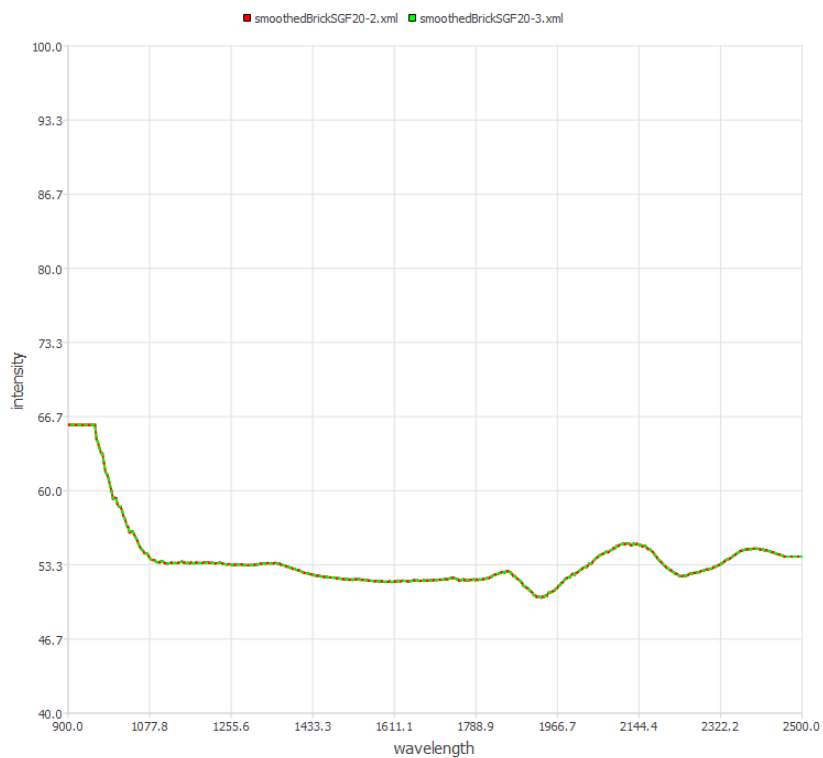


Fig. 2.6: Polynomial order N and N+1

3 Spectrometric data

This section describes obtaining and pre-processing the data used for smoothing.

3.1 Equipment

The spectrometer used was *NIRQuest512-2.5*, made by Ocean Optics. It is made for Near-Infrared measurements and the wavelength range is approximately 900-2500 nm. It is capable of measuring Absorbance as well as Reflectance and Transmittance. Its optical resolution is 6.3 nm FWHM (full width at half maximum) and it has 512 spectral bands.

The raw measurement was downloaded as a text file with a header and data in the XY format separated by a semicolon, where X is the wavelength and Y is the intensity/reflectance [1].

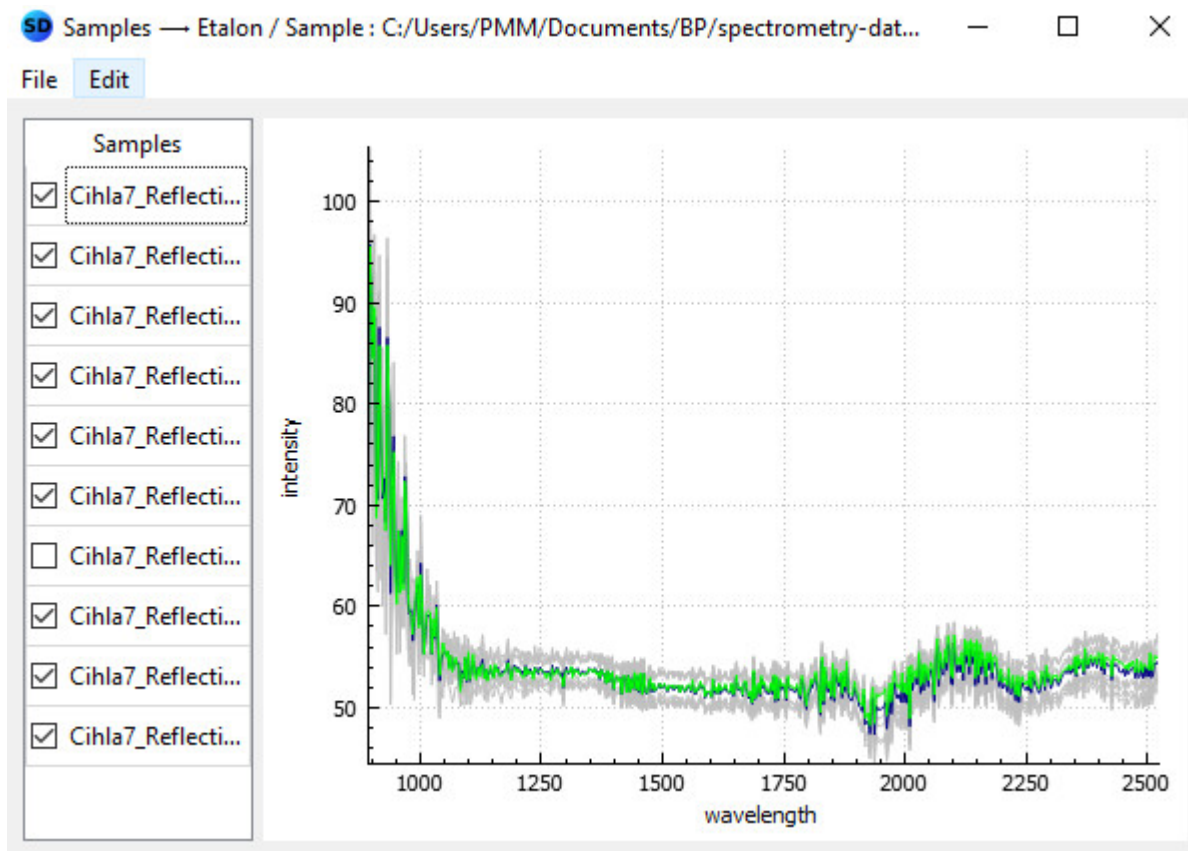


Fig. 3.1: *NIRQuest512-2.5*

3.2 *QSdata*

This data was subsequently processed by the *QSdata* [1] program, more specifically with the *Samples to etalon* function, which was developed as part of the New modern non-invasive methods of cultural heritage objects exploration grant [12]. More information about this program is given in Sec. 5.

The etalon is created either as an average or a median of the selected raw data, as seen in Fig. 3.2. The result can be then saved in **.xml* format with the structure shown in Tab. 3.1. Here each `<point>` element in the `<spectraldata>` root element has two child elements, `<x>` and `<y>`. This file is then ready for smoothing.

Fig. 3.2: Creating an etalon in *QSDATA*

Tab. 3.1: Structure of data

```

<?xml version="1.0" encoding="UTF-8"?>
<endmember>
  <header>
    <name>Brick7</name>
  </header>
  <spectraldata>
    <point>
      <x>893.116</x>
      <y>70.637</y>
    </point>
    ...
    <point>
      <x>2524.21</x>
      <y>54.891</y>
    </point>
  </spectraldata>
</endmember>

```

4 Development of *SGF.exe*

The purpose of this program is to produce a file of smoothed data.

Functional requirements

- to smooth a file of data by SGF
- to smooth a file of data with MA
- to calculate the data's S th derivative

Non-functional requirements

- to implement the program in C++
- to run the program from command line

4.1 Analysis

The following flowchart in Fig. 10 shows the algorithm of the program.

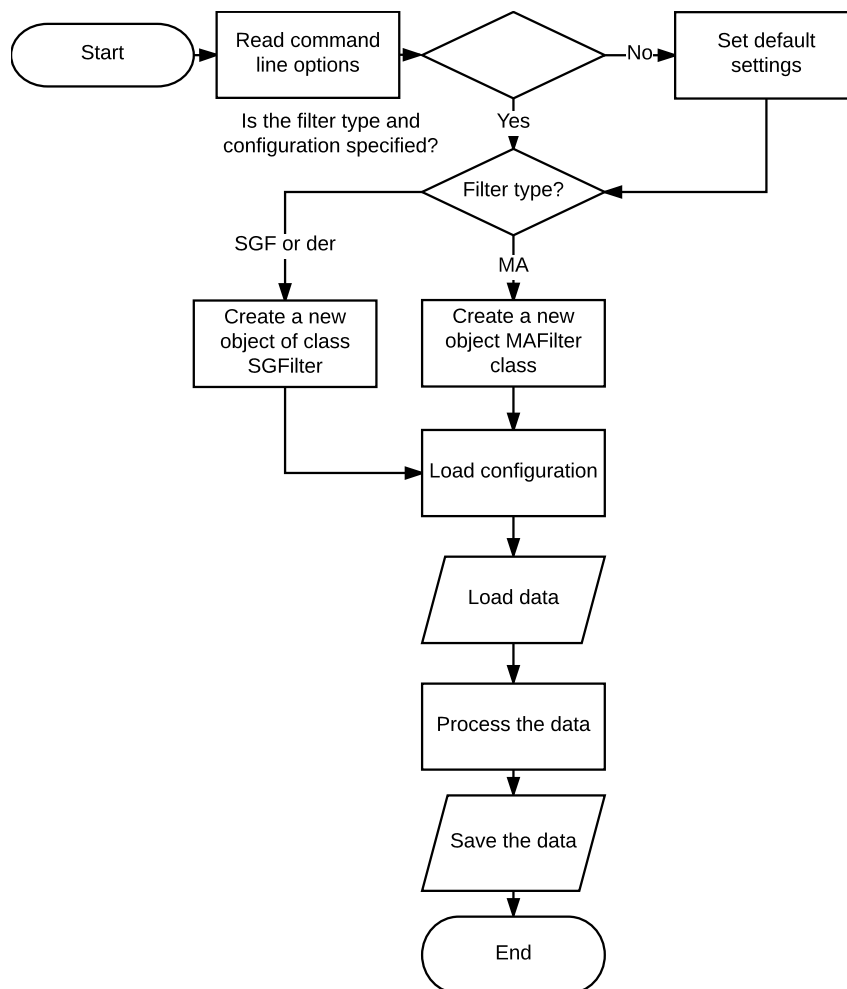


Fig. 4.1: Flowchart for the *SGF.exe*.

Input

The program loads one file of spectrometric data at a time and then proceeds to process it. Earlier versions of the program iterated over a folder of data, but that showed to be impractical for smoothing more data from different folders and with different parameters. Now if the user wants to run the program with multiple sets of data and different parameters, it is best to create a *.bat* file that runs the program several times over the desired data and configuration.

Processing the data

After the data is loaded, the program proceeds to filter the data with the chosen filter (or find the derivative of a given order). The moving average option was added for comparison.

In case of the Savitzky-Golay filter (and the derivatives), the program first creates the matrix H and then proceeds to use it in the filtering. Because of the nature of the filter, it is necessary to start going over the data at the M -th position, so that the full length of the filter can be used. For the same reason, the filter stops at at the M -th position from the end. The data at the ends is then substituted with the first and last filtered point.

There are other possibilities of treating endpoint data. One of the other options, as suggested by [11] and [6] would be to use gradually smaller values of M (filter half-width). This of course effects the simplicity of the code and the computation time but is useful if the data contains crucial information at the ends of the spectra.

Output

The filtered data is stored in the *points* structure, which is described in more detail in the next section. It is then saved into **.xml* format with the same structure as the input data. (For input data structure see Tab. 3.1)

4.2 Implementation

Parts of the program were deliberately not optimized to keep the code easy to read and understand and are ready for improvement in later work. This decision was inspired by Jackson's two rules for optimization [7].

The program is designed to load a selected file in **.xml* format. The data the user is willing to smooth needs to have the format described in 3.1. This structure means that the data is easy to read, parse and then load into the data structure in the program.

The data is loaded with the `load` function which uses the `boost property tree` [2] library to parse it into the *points* structure. This structure has three data members of type double, `x`, `yin`, which are the values from the **.xml* file and `yout`, which is used to store smoothed y-values and has the default value of 0.

Smoothing

After parsing the file, the program checks if the operation type and configuration is specified, if not, the default settings are used. If the filter is specified, the program creates a new child class of the respective type. The `config` function is then used to parse the parameters of the selected filter and proceeds to check if it's possible to actually smooth the data. The conditions for

the Savitzky-Golay filter are as listed in Tab. 4.1

If the **SGF** filter or the derivatives' option is selected, the matrix **H** is created with the use of **boost numeric** [2] library and is then passed on to the function **filter** or **derivative** which do the filtering itself and store the smoothed values into the above mentioned **points** structure.

If the **MA** filter is selected, the data is also smoothed accordingly and stored into the **points** structure.

Tab. 4.1: Table of **SGFilter** conditions.

condition:

$M <= 0$ OR $N <= 0$	both arguments must be larger than 0
$N < M$	the polynomial order cannot be larger than the filter half-length
$M >= 2 * \text{data size} + 1$	the filter half-length must fit into the data size at least twice

Saving

Saving the data also happens through the **boost property tree** [2] library; the data is first loaded into a tree and then saved in the **.xml* format to the desired location.

At the end of each cycle, the memory is cleared.

Command-line options

The program runs from the terminal and the options are read in the following sequence:

```
[input file path] [output folder] [filter type] [filter configuration]
```

In case of the **SGF** filter configuration, the individual parameters must be separated by a dash (-). This is to make the process of naming the output file easier. Below are listed all of the program's options with which it can be run.

Filter type **SGF** - Savitzky-Golay filter
 der - Savitzky-Golay derivatives
 MA - moving average filter

Filter parameters for **SGF** filter: parameters **M**, **N** which are the filter half-length and polynomial order. The parameters must be in this order and separated by a dash only for the program to correctly parse them.

for the **der** option: parameters **M**, **N** are entered in the same way as in the previous case but in addition to that, the **S** parameter (derivative order) is also required.

for **MA** filter: only the window size **M** is needed.

4.2.1 Graphic representation of the results

The graphs used in this thesis were generated with own program *sgraph* using the **Qt** library [4], also written in C++.

5 Comparison of data performance using *QSdata*

The program's function *Unmixing samples* calculates the coefficients of a linear combination of the etalons to find which correspond with the provided sample the most by performing non-negative least squares computation as described in [9]. The goal is for the function to find what etalons is the sample composed of and what is the percentage composition (if any) of them. The final numbers don't add up to 100%, this condition wasn't implemented in the program for the fear that it may impact the results in a negative way. In addition to that, not all samples are composed solely of supplied etalons, so the 100% condition would also be illogical.

Output of the *QSdata* [1] program is shown in Fig. 5.1, where the *Unmixing samples* option was used on unsmoothed etalons and data.

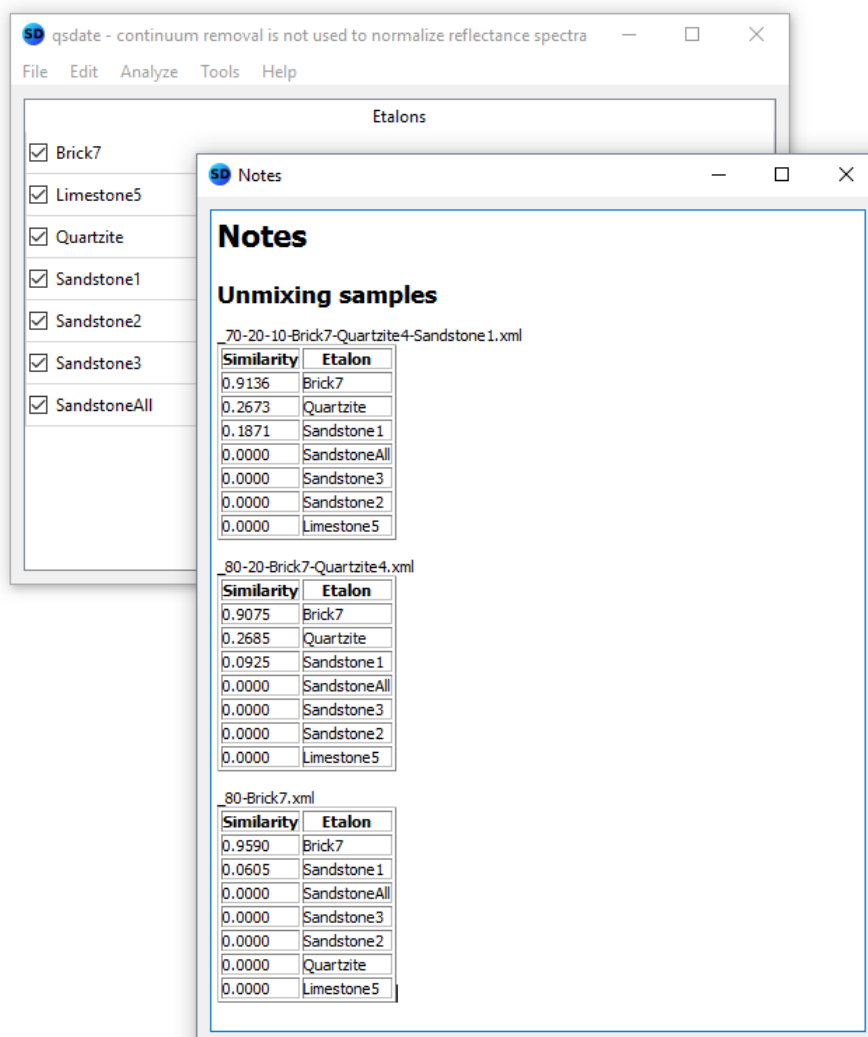


Fig. 5.1: Example of the *Unmixing samples* output.

5.1 Etalons and samples

The etalons listed in Tab. 5.1 were created by averaging the supplied measured data using the *QSdata* program [1]. The sandstone etalons were each taken from one set of measurements and the *SandstoneAll.xml* is an average of all the sandstone measurements together.

The samples used in this thesis were kindly provided by prof. Ing. Aleš Čepek, CSc. and are listed in Tab. 5.2. These raw samples were created as weighted averages of the measured data, so 70% of material A and 30% of material B would be calculated as $0.7A + 0.3B$.

These etalons and samples were used smoothed and unsmoothed in various combinations to

Tab. 5.1: The list of used etalons.

Brick7.xml
Limestone5.xml
Quartzite4.xml
Sandstone1.xml
Sandstone2.xml
Sandstone3.xml
SandstoneAll.xml

Tab. 5.2: Overview of the samples.

File name	description
<i>80-Brick7.txt</i>	80% Brick7
<i>80-20-Brick7-Quartzite4.txt</i>	80% Brick7 and 20% Quartzite4
<i>70-20-10-Brick7-Quartzite4-Sandstone1.txt</i>	70% Brick7, 20% Quartzite4, 10% Sandstone1

produce results presented in the following part. The *Etalon min. X* and *Etalon max. X* options have been set to 900 and 2500, which is the wavelength range (in nm) of the used spectrometer stated by the manufacturer. The smoothing parameters were chosen as $M=6$ and $N=2$. Results which are smaller than 0.1 can be omitted due to the expected calculation error, as stated in [1].

In Tab. 5.3, 5.4, 5.5, you can see that when we only smooth the sample, the percentage proportion of the main etalon is relatively close to the truth, but the other etalons' representation is not accurate. Also another etalon which is not represented in the sample at all shows up in the results. In the case of all three samples it is the *Sandstone3* etalon. As you can see in Fig. 5.2, this is most likely due to the similar shape of this etalon to the samples. The sample smoothing only enhances this similarity and thus the etalon shows up in the results.

When we smooth the etalon and not the sample, we get results which are similar to the ones with no smoothing at all. It is interesting to note that while the percentage proportion of the main etalon remains roughly the same, the proportion of the second mostly represented etalon (in both our cases *Quartzite4*) shows up more when we only smooth the etalon and in Tab. 5.5, the least represented etalon *Sandstone1* shows up more.

On the contrary, when we smooth both the sample and the etalon, *Quartzite4* has a lower percentage proportion while *Sandstone1* is represented more.

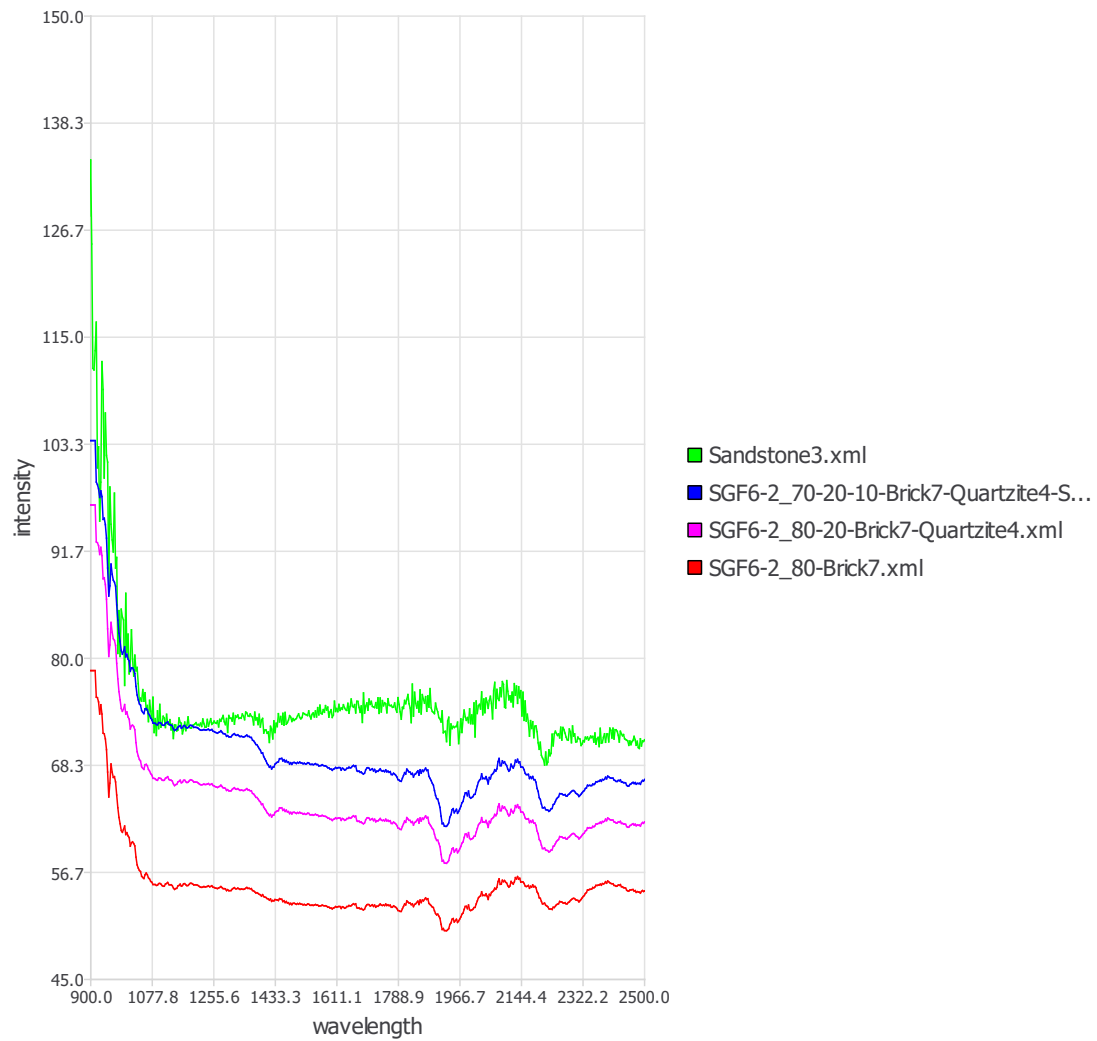


Fig. 5.2: Comparison of the *Sandstone3* etalon with the samples.

Tab. 5.3: *Unmixing samples* results for sample *80-Brick7*.

	<i>all unsmoothed</i>	<i>smoothed sample</i>	<i>smoothed etalon</i>	<i>all smoothed</i>
<i>Brick7</i>	0.9590	0.6820	0.9589	0.9546
<i>Limestone5</i>	-	-	-	-
<i>Quartzite4</i>	-	-	0.0084	-
<i>Sandstone1</i>	0.0605	0.0945	0.0550	0.0649
<i>Sandstone2</i>	-	-	-	-
<i>Sandstone3</i>	-	0.1751	-	-
<i>SandstoneAll</i>	-	-	-	-

Tab. 5.4: *Unmixing samples* results for *80-20-Brick7-Quartzite4*.

	<i>all unsmoothed</i>	<i>smoothed sample</i>	<i>smoothed etalon</i>	<i>all smoothed</i>
<i>Brick7</i>	0.9075	0.5712	0.9077	0.9016
<i>Limestone5</i>	-	-	-	-
<i>Quartzite4</i>	0.2685	0.1220	0.2889	0.2588
<i>Sandstone1</i>	0.0925	0.2269	0.0779	0.1056
<i>Sandstone2</i>	-	-	-	-
<i>Sandstone3</i>	-	0.2242	-	-
<i>SandstoneAll</i>	-	-	-	-

Tab. 5.5: *Unmixing samples* results for *70-20-10-Brick7-Quartzite4-Sandstone1*.

	<i>all unsmoothed</i>	<i>smoothed sample</i>	<i>smoothed etalon</i>	<i>all smoothed</i>
<i>Brick7</i>	0.9136	0.5842	0.9151	0.9087
<i>Limestone5</i>	-	-	-	-
<i>Quartzite4</i>	0.2673	0.1222	0.2866	0.2566
<i>Sandstone1</i>	0.1871	0.3187	0.1722	0.2001
<i>Sandstone2</i>	-	-	-	-
<i>Sandstone3</i>	-	0.2206	-	-
<i>SandstoneAll</i>	-	-	-	-

5.2 Derivatives

In addition to only smoothing, a test was carried out to see if calculating derivatives while smoothing the data (with a process described in Sec. 2.1) would give off better results. For this purpose, the first derivatives were calculated for the filter configurations of $3-2$, $6-2$ and $6-4$, where the filter half-length and polynomial order are given in the format $M-N$. The results can be seen in Tab. 5.6, 5.7 and 5.8.

With the $3-2$ configuration, the derivatives performed well in terms of identifying the most represented sample, however the other etalon estimations were wrong. In case of Tab. 5.7 and Tab. 5.8, the filter again identified *Sandstone3* as a significant sample, as it did in Sec. 5.1.

The configuration $6-2$ showed to give the best results, successfully eliminating all the unrepresented etalons and determining the percentage proportion most accurately.

When using the fourth polynomial for smoothing, we can see that the main etalon was again successfully identified, but in the case of the sample *80-Brick7* in Tab. 5.6, we can see that the algorithm wrongly stated the proportion of *Quartzite4*, which is not present at all.

As we can see, with the right configuration, the use of derivatives proves to be much better than using just the smoothed data. This is most likely due to the fact that the raw data from one material usually has roughly the same shape, but is shifted either up or down, as can be seen in Fig. 5.3. The etalons made from this shifted data are therefore not accurately positioned on the y-axis. Using derivatives eliminates this shift, because the constant is lost in the process, so that is why this method gives better results.

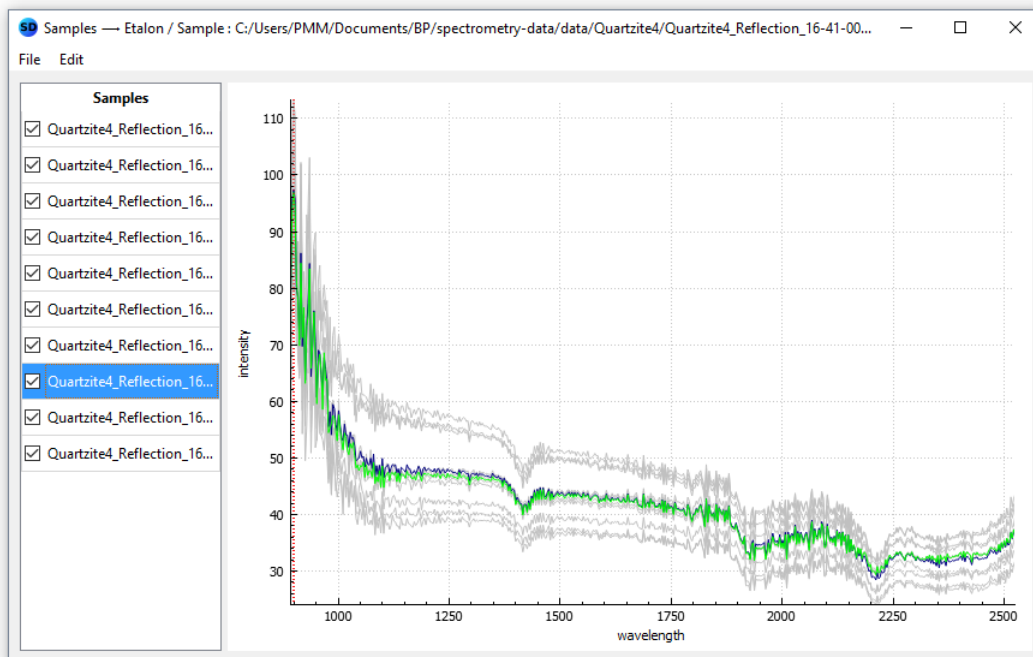


Fig. 5.3: Creating the etalons

Tab. 5.6: *Unmixing samples* results for derivatives of *80-Brick7*

<i>configuration (M-N):</i>	3-2	6-2	6-4
<i>Brick7</i>	0.7461	0.7868	0.7947
<i>Limestone5</i>	-	-	-
<i>Quartzite4</i>	0.1469	0.0629	0.1136
<i>Sandstone1</i>	0.0051	0.0036	-
<i>Sandstone2</i>	-	-	-
<i>Sandstone3</i>	-	0.0777	0.0480
<i>SandstoneAll</i>	-	-	-

Tab. 5.7: *Unmixing samples* results for derivatives of *80-20-Brick7-Quartzite4*

<i>configuration (M-N):</i>	3-2	6-2	6-4
<i>Brick7</i>	0.8341	0.8303	0.8921
<i>Limestone5</i>	-	-	-
<i>Quartzite4</i>	0.1397	0.3156	0.3128
<i>Sandstone1</i>	0.0367	0.0298	-
<i>Sandstone2</i>	-	-	-
<i>Sandstone3</i>	0.1351	-	0.0022
<i>SandstoneAll</i>	-	-	-

Tab. 5.8: *Unmixing samples* results for derivatives of *70-20-10-Brick7-Quartzite4-Sandstone1*

<i>configuration (M-N):</i>	3-2	6-2	6-4
<i>Brick7</i>	0.8275	0.8167	0.8880
<i>Limestone5</i>	-	-	-
<i>Quartzite4</i>	0.1345	0.3337	0.3135
<i>Sandstone1</i>	0.1213	0.1042	0.0919
<i>Sandstone2</i>	-	-	-
<i>Sandstone3</i>	0.1523	-	0.0134
<i>SandstoneAll</i>	-	-	-

6 Conclusion

In the first part of this work the principles and algorithm of the Savitzky-Golay filter were described, along with some of the filter's properties. The calculation of derivatives was also explained.

Next we looked at how the data used in this thesis was obtained. The model of the used spectrometer was mentioned, along with some notes on post processing the data. The program *QSdata* [1] was also introduced and the structure in which the data loads into the author's own program was shown.

A large part of this thesis was the program *SGF.exe*, which was used to smooth the data (and find the derivatives of a given order). Analysis of the problem was given and the requirements were specified and a flowchart was presented. Some parts of the implementation process were also described, like the input, conditions under which the smoothing can be performed and the command line usage of the program. Graphic representation of the results was also mentioned.

After smoothing, the data is loaded into the *QSdata* program where, using the *Unmixing samples* function is used. This function returns the calculated estimate of the percentage representation of all the etalons for each sample. Three samples were used with 7 etalons and one set configuration.

First, samples and etalons were used without smoothing so we could see if the other results were actually better. Then we smoothed only the samples, only the etalons and finally, we processed smoothed samples and etalons. Another option was to compare the derivatives, so we took the unsmoothed etalons and samples and using the program *SGF.exe* we computed the first derivatives and smoothed the data with three different configurations.

6.1 Further improvements

Trying to smooth the data with higher order derivatives would be useless with the current configuration of the *Unmixing samples* function, however the right configuration of the filter and derivatives of the first order gave quite good results, as we saw in Sec. 5.2.

It would also be an interesting experiment to add a feature to *QSdata* [1] that would compute the percentage representation based on the first and second order derivatives (i.e. it would detect the peaks and inflection points). Would the results be even better than what we've achieved with the first derivatives using the *Unmixing samples* option?

As an idea for further improvement, it would certainly be good to make a GUI for the program *SGF.exe* to make it more user friendly and also to implement a function for plotting graphs right in the program.

Another addition to *SGF.exe* could be the treatment of end points. Ideally the user could choose if they want to set them equal to the first and last calculated points (this is already implemented in the existing version), set them to zero or compute the linear convolution/least squares at every point separately, gradually making the filter length shorter. The last would require substantially more computing time, but the time could also be reduced by parallel computing, where the program could simultaneously smooth more files at once. Another option with parallel computation would be to compute the end points at the same time as the data in the middle.

References

- [1] Eva Matoušková Aleš Čepěk and Karel Pavelka. *QSData - zpracování dat obrazové spektrometrie*. FSv ČVUT, 2016.
- [2] Robert Klarer Beman Dawes. boost:C++ libraries. <http://www.boost.org/>, 2004–2007.
- [3] Manfred U. A. Bromba and Horst. Ziegler. Application hints for savitzky-golay digital smoothing filters. *Analytical Chemistry*, 53(11):1583–1586, 1981.
- [4] Eirik Chambe-Eng and Haavard Nord. QT library. <https://www.qt.io/>, 2017.
- [5] Attila Felinger. *Data analysis and signal processing in chromatography*, volume 21. Elsevier, 1998.
- [6] Peter A Gorry. General least-squares smoothing and differentiation by the convolution (savitzky-golay) method. *Analytical Chemistry*, 62(6):570–573, 1990.
- [7] Michael A Jackson. Principles of program design. 1975.
- [8] R. L. King, C. Ruffin, F. E. LaMastus, and D. R. Shaw. *The analysis of hyperspectral data using Savitzky-Golay filtering-theoretical basis. 1*, volume 1. 1999.
- [9] Charles L Lawson and Richard J Hanson. *Solving least squares problems*. SIAM, 1995.
- [10] Peter Monchamp, Lucio Andrade-Cetto, Jane Y Zhang, and Robert Henson. Signal processing methods for mass spectrometry. *Systems Bioinformatics: An Engineering Case-Based Approach*, Artech House Publishers, 2007.
- [11] Prof. Tom O’Haver. A pragmatic introduction to signal processing, 1995.
- [12] Pavelka, Matoušková, Faltýnová, Šedina. NAKI - DF13P01OVV002, Nové moderní metody dokumentace památkových objektů, 2013-2016.
- [13] William H Press, Brian P Flannery, Saul A Teukolsky, William T Vetterling, et al. *Numerical recipes*, volume 3. cambridge University Press, cambridge, 1989.
- [14] Abraham Savitzky and Marcel JE Golay. Smoothing and differentiation of data by simplified least squares procedures. *Analytical chemistry*, 36(8):1627–1639, 1964.
- [15] R. W. Schafer. On the frequency-domain properties of savitzky-golay filters. In *2011 Digital Signal Processing and Signal Processing Education Meeting (DSP/SPE)*, pages 54–59, Jan 2011.
- [16] R. W. Schafer. What is a savitzky-golay filter? [lecture notes]. *IEEE Signal Processing Magazine*, 28(4):111–117, July 2011.
- [17] Fuan Tsai and William Philpot. Derivative analysis of hyperspectral data. *Remote Sensing of Environment*, 66(1):41 – 51, 1998.
- [18] A. X. Zhao, X. J. Tang, Z. H. Zhang, and J. H. Liu. The parameters optimization selection of savitzky-golay filter and its application in smoothing pretreatment for ftir spectra. In *2014 9th IEEE Conference on Industrial Electronics and Applications*, pages 516–521, June 2014.

List of Figures

2.1	Smoothed data and its derivatives	3
2.2	Relation between polynomial order N, filter half-length M and the normalized cutoff frequency	4
2.3	The results for smoothing with different polynomial orders with a fixed filter length.	6
2.4	The results for smoothing with different filter lengths with a fixed polynomial order.	6
2.5	Comparison of MA and SGF filters of the same length	7
2.6	Polynomial order N and N+1	7
3.1	<i>NIRQuest512-2.5</i>	8
3.2	Creating an etalon in <i>QSdata</i>	9
4.1	Flowchart for the <i>SGF.exe</i>	10
5.1	Example of the <i>Unmixing samples</i> output.	13
5.2	Comparison of the <i>Sandstone3</i> etalon with the samples.	15
5.3	Creating the etalons	17

List of Tables

3.1	Structure of data	9
4.1	Table of <code>SGFilter</code> conditions.	12
5.1	The list of used etalons.	14
5.2	Overview of the samples.	14
5.3	<i>Unmixing samples</i> results for sample <i>80-Brick7</i>	16
5.4	<i>Unmixing samples</i> results for <i>80-20-Brick7-Quartzite4</i>	16
5.5	<i>Unmixing samples</i> results for <i>70-20-10-Brick7-Quartzite4-Sandstone1</i>	16
5.6	<i>Unmixing samples</i> results for derivatives of <i>80-Brick7</i>	18
5.7	<i>Unmixing samples</i> results for derivatives of <i>80-20-Brick7-Quartzite4</i>	18
5.8	<i>Unmixing samples</i> results for derivatives of <i>70-20-10-Brick7-Quartzite4-Sandstone1</i>	18

List of used abbreviations

GUI — graphic user interface

M — filter half-length

MA — moving average

N — polynomial order

S — order of the derivative

SGF — Savitzky-Golay filter

List of used programs

- Qt Creator 4.1.0
- QSData 0.9.6
- Texmaker 4.5

Where to access the program *SGF.exe*

The program created for the purpose of this thesis can be found on
<https://github.com/millapet/petra-millarova-bt>

This program is distributed under GNU General Public License.