

Zeus

1.01

Generated by Doxygen 1.9.6



<b>1 Hierarchical Index</b>	<b>1</b>
1.1 Class Hierarchy	1
<b>2 Class Index</b>	<b>3</b>
2.1 Class List	3
<b>3 File Index</b>	<b>5</b>
3.1 File List	5
<b>4 Class Documentation</b>	<b>7</b>
4.1 Backend Class Reference	7
4.1.1 Detailed Description	8
4.1.2 Member Function Documentation	8
4.1.2.1 addSetToSets()	8
4.1.2.2 addWavelength()	9
4.1.2.3 averageIndividualKeyValuePairs()	9
4.1.2.4 findHighestKeyValuePair()	9
4.1.2.5 findKeys()	10
4.1.2.6 findRequestedValues()	10
4.1.2.7 findRequestedValuesCalibration()	11
4.1.2.8 getAveragedSpectra()	11
4.1.2.9 getRequestedSpectraCalibrationMode()	11
4.1.2.10 getRequestedSpectraStandardMode()	12
4.1.2.11 getRSquared()	12
4.1.2.12 initializeMemoryFiles()	13
4.1.2.13 initializeSets()	13
4.1.2.14 loadFiles()	13
4.1.2.15 numberOfValidSets()	14
4.1.2.16 processFileIntoDictionary()	14
4.1.2.17 removeWavelength()	15
4.1.2.18 returnDivisionFromTwoFirst()	15
4.1.2.19 saveToFile()	15
4.1.2.20 saveToFileCalibration()	16
4.1.2.21 sumDictionaries()	16
4.2 LIBSProcessing::Window Class Reference	16
4.2.1 Detailed Description	18
4.2.2 Constructor & Destructor Documentation	18
4.2.2.1 Window()	19
4.2.3 Member Function Documentation	19
4.2.3.1 addSetButton_Click()	19
4.2.3.2 calibrationToolStripMenuItem_Click()	19
4.2.3.3 fileSelect_Click()	20
4.2.3.4 fileSelect_setB_Click()	20

---

4.2.3.5 handleSelection()	20
4.2.3.6 howManySubmit_Click()	21
4.2.3.7 InitializeComponent()	21
4.2.3.8 preview_Click()	21
4.2.3.9 removeWave_Click()	22
4.2.3.10 saveFolderSelect_Click()	22
4.2.3.11 saveToFile_Click()	22
4.2.3.12 setCalibrationGroup()	23
4.2.3.13 setSetAddedGroup()	23
4.2.3.14 standardToolStripMenuItem_Click()	23
4.2.3.15 waveSubmit_Click()	24
4.2.3.16 Window_Load()	24
<b>5 File Documentation</b>	<b>25</b>
5.1 C:/MEng/LIBSProcessing/LIBSProcessing/Backend.h File Reference	25
5.1.1 Detailed Description	25
5.2 Backend.h	26
5.3 C:/MEng/LIBSProcessing/LIBSProcessing/Window.cpp File Reference	36
5.3.1 Detailed Description	36
5.4 C:/MEng/LIBSProcessing/LIBSProcessing/Window.h File Reference	36
5.4.1 Detailed Description	37
5.5 Window.h	37
<b>Index</b>	<b>49</b>

# Chapter 1

## Hierarchical Index

### 1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Backend . . . . .	7
System::Windows::Forms::Form	
LIBSProcessing::Window . . . . .	16



## Chapter 2

# Class Index

### 2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

#### [Backend](#)

Class handling the backend functions of the entire programme. Public variables are expanded on within the code . . . . . 7

#### [LIBSProcessing::Window](#)

Class handling the display of the user interface (later referred to as GUI or UI). Primarily has functions responsible for recording button presses . . . . . 16





## Chapter 3

# File Index

### 3.1 File List

Here is a list of all documented files with brief descriptions:

C:/MEng/LIBSProcessing/LIBSProcessing/ <a href="#">Backend.h</a>	
File handling all the backend functions, invisible to the user . . . . .	25
C:/MEng/LIBSProcessing/LIBSProcessing/ <a href="#">Window.cpp</a>	
File running the programme . . . . .	36
C:/MEng/LIBSProcessing/LIBSProcessing/ <a href="#">Window.h</a>	
File handling the UI of the programme . . . . .	36



## Chapter 4

# Class Documentation

### 4.1 Backend Class Reference

Class handling the backend functions of the entire programme. Public variables are expanded on within the code.

```
#include <Backend.h>
```

#### Public Member Functions

- int [saveToFile](#) (String^ name, bool selectedOnly, bool lowestPoint)
- int [saveToFileCalibration](#) (String^ name, bool lowestPoint)
- int [loadFiles](#) (array< String^>^ fileNames, float cutoff, int whichMode)
- int [initializeSets](#) (int length)
- int [addSetToSets](#) (float concentration, int index, float cutoff)
- int [getRequestedSpectraStandardMode](#) (int option, float range, bool doLowerRange, float lowerRange)
- int [getRequestedSpectraCalibrationMode](#) (int option, float range, bool doLowerRange, float lowerRange)
- int [getAveragedSpectra](#) ()
- int [addWavelength](#) (float wavelength)
- int [removeWavelength](#) (float wavelength)
- float [getRSquared](#) ()

#### Public Attributes

- String ^ **directory**
- String ^ **nameOfFile**
- List< String^> ^ **filesToExtract**
- List< Dictionary< float, float >^> ^ **listOfDictionaries**
- Dictionary< float, float > ^ **result**
- List< float > ^ **indexedKeys**
- Dictionary< float, float > ^ **presentToUserResult**
- List< float > ^ **selectedWavelengths**
- List< float > ^ **userSelectionsToKeys**
- List< int > ^ **userSelectionsIndexes**
- List< Dictionary< float, float >^> ^ **listOfResultsForFiles**
- List< float > ^ **listOfAveragedIndividualResults**
- List< String^> ^ **filesToExtract\_B**

- List< Dictionary< float, float >^> ^ **listOfDictionaries\_B**
- List< String^> ^ **metadata**
- List< List< Dictionary< float, float >^>^> ^ **listOfSets**
- List< float > ^ **listOfConcentrations**
- List< float > ^ **listOfAverages**
- List< List< Dictionary< float, float >^>^> ^ **listOfProcessedSets**
- List< List< float >^> ^ **listOfAveragedIndividualResults\_sets**
- float **global\_r2**

## Private Member Functions

- int [processFileIntoDictionary](#) (Dictionary< float, float >^ dict, String^ filename, float cutoff)
- int [initializeMemoryFiles](#) (float cutoff, int whichMode)
- int [sumDictionaries](#) ()
- int [findKeys](#) ()
- int [findRequestedValues](#) (int option, float range, bool doLowerRange, float lowerRange)
- int [findRequestedValuesCalibration](#) (int option, float range, bool doLowerRange, float lowerRange)
- int [numberOfValidSets](#) ()
- System::Tuple< float, float > ^ [findHighestKeyValuePair](#) (int i, float key, int option, float rangeEachWay, bool whichDir, Dictionary< float, float >^ inputDict)
- List< float > ^ [averageIndividualKeyValuePairs](#) (List< Dictionary< float, float >^>^ LOD)
- float [returnDivisionFromTwoFirst](#) (List< float >^ givenList)

### 4.1.1 Detailed Description

Class handling the backend functions of the entire programme. Public variables are expanded on within the code.

The class has the following main functions:

- opening files after UI has provided this class a list of filenames.
- converting said files into a data format processable by the programme.
- storing the selected peaks, as well as any database present in memory.
- saving information to a file.

### 4.1.2 Member Function Documentation

#### 4.1.2.1 addSetToSets()

```
int Backend::addSetToSets (
    float concentration,
    int index,
    float cutoff ) [inline]
```

Function adding loaded files to a set at a given index.

#### Parameters

<i>concentration</i>	User-supplied for a given set.
<i>index</i>	of the current set selected. Provided by which set user currently has selected in the GUI.
<i>cutoff</i>	Values of light intensities below which the intensity will be set to 0 when initializing data structures. Default -199.

**Returns**

Returns 1 on success and 0 on failure.

**4.1.2.2 addWavelength()**

```
int Backend::addWavelength (
    float wavelength ) [inline]
```

Function adding an user-supplied wavelength to an internal data structure. Checks if wavelength is correct as well.

**Parameters**

<i>wavelength</i>	Wavelength to be added.
-------------------	-------------------------

**Returns**

Returns 1 if wavelength is valid; 0 otherwise.

**4.1.2.3 averageIndividualKeyValuePairs()**

```
List< float > ^ Backend::averageIndividualKeyValuePairs (
    List< Dictionary< float, float > ^ ^ LOD ) [inline], [private]
```

Private function that averages the values of keys in the same user-supplied keys. However, the keys don't necessarily have to be the same - this enables calculations for slightly different keys (for example, the highest keys) that had the same user-supplied key value.

**Parameters**

<i>LOD</i>	List of Dictionaries - passed by reference to find a result in a set of dictionaries.
------------	---

**Returns**

Returns a smart pointer to a list that holds the results for a given list of dictionaries (files).

**4.1.2.4 findHighestKeyValuePair()**

```
System::Tuple< float, float > ^ Backend::findHighestKeyValuePair (
    int i,
    float key,
    int option,
```

```
float rangeEachWay,
bool whichDir,
Dictionary< float, float >^ inputDict ) [inline], [private]
```

Private function for finding the highest key in a given set.

#### Parameters

<i>i</i>	The index of the key in the indexedKeys array.
<i>key</i>	The value of the key for the dictionary.
<i>option</i>	Option for how to process the data. 1 - find the highest peak within range, 2- sum all datapoints within range. If whichDir == false, this parameter is discarded.
<i>rangeEachWay</i>	The range to check, each way away from the key.
<i>whichDir</i>	boolean determining whether the function finds the highest or lowest value in range.
<i>inputDict</i>	Dictionary (i. e. file) on which the search is performed; passed by reference.

#### Returns

Returns a tuple of values <highestKey, highestValue> or <lowestKey, lowestValue>, depending on whichDir.

#### 4.1.2.5 findKeys()

```
int Backend::findKeys ( ) [inline], [private]
```

Private function that finds the appropriate keys for the values that user supplies. It does so by looking for the closest key to the value supplied. Heavily amortized O(n) time complexity for this search. Fills the self-contained dictionary.

#### Returns

Always returns 1.

#### 4.1.2.6 findRequestedValues()

```
int Backend::findRequestedValues (
    int option,
    float range,
    bool doLowerRange,
    float lowerRange ) [inline], [private]
```

Private function finding requested values of intensities for previously provided wavelengths in range for standard LIBS mode.

#### Parameters

<i>option</i>	Option for how to process the data. 1 - find the highest peak within range, 2- sum all datapoints within range.
<i>range</i>	Range in which to find the highest peak value.
<i>doLowerRange</i>	Does the user want to find lowest value in range?
<i>lowerRange</i>	If yes, what is that range?

**Returns**

Always returns 1.

**4.1.2.7 findRequestedValuesCalibration()**

```
int Backend::findRequestedValuesCalibration (
    int option,
    float range,
    bool doLowerRange,
    float lowerRange ) [inline], [private]
```

Private function finding requested values of intensities for previously provided wavelengths in range for calibration LIBS mode.

**Parameters**

<i>option</i>	Option for how to process the data. 1 - find the highest peak within range, 2- sum all datapoints within range.
<i>range</i>	Range in which to find the highest peak value.
<i>doLowerRange</i>	Does the user want to find lowest value in range?
<i>lowerRange</i>	If yes, what is that range?

**Returns**

Always returns 1.

**4.1.2.8 getAveragedSpectra()**

```
int Backend::getAveragedSpectra ( ) [inline]
```

Function averaging all spectra out in a set. Serves as an intermediate step between UI and a private function.

**Returns**

Returns 1 on success, 0 otherwise.

**4.1.2.9 getRequestedSpectraCalibrationMode()**

```
int Backend::getRequestedSpectraCalibrationMode (
    int option,
    float range,
    bool doLowerRange,
    float lowerRange ) [inline]
```

Function called by the UI frontend, processing loaded data in accordance to the selected wavelengths in calibration mode. Serves as an intermediate step between other functions.

**Parameters**

<i>option</i>	Option for how to process the data. 1 - find the highest peak within range, 2- sum all datapoints within range.
<i>range</i>	Float specifying how many datapoints are looked at. User provided values for wavelengths assumed to be in them middle of this range.
<i>doLowerRange</i>	- boolean specifying whether the user has requested to find lowest values in some range as well.
<i>lowerRange</i>	If the user requests that lowest point is found - analogous to parameter 'range'.

**Returns**

Returns 1 on success, 0 on failure.

**4.1.2.10 getRequestedSpectraStandardMode()**

```
int Backend::getRequestedSpectraStandardMode (
    int option,
    float range,
    bool doLowerRange,
    float lowerRange ) [inline]
```

Function called by the UI frontend, processing loaded data in accordance to the selected wavelengths in single mode. Serves as an intermediate step between other functions.

**Parameters**

<i>option</i>	Option for how to process the data. 1 - find the highest peak within range, 2- sum all datapoints within range.
<i>range</i>	Float specifying how many datapoints are looked at. User provided values for wavelengths assumed to be in them middle of this range.
<i>doLowerRange</i>	boolean specifying whether the user has requested to find lowest values in some range as well.
<i>lowerRange</i>	If the user requests that lowest point is found - analogous to parameter 'range'.

**Returns**

Always returns 1 - programme will fail before reaching the call of this function.

**4.1.2.11 getRSquared()**

```
float Backend::getRSquared ( ) [inline]
```

Function calculating  $R^2$  value; called after files in calibration mode have been initialized.

**Returns**

Returns the value of  $R^2$ . Also initializes the global value of  $R^2$  in the programme.



#### 4.1.2.12 initializeMemoryFiles()

```
int Backend::initializeMemoryFiles (
    float cutoff,
    int whichMode ) [inline], [private]
```

Private function to extract data from all selected files (as per internally initialized list of files) and load them into RAM. Goes one by one for each filename provided.

##### Parameters

<i>cutoff</i>	Value of cutoff, below which the intensity of a given datapoint will be brought down to 0. Default -199.
<i>whichMode</i>	Integer describing which mode the programme is operating in. 1 - standard LIBS mode, 2 - calibration LIBS mode.

##### Returns

Returns 1 on success and 0 on failure.

#### 4.1.2.13 initializeSets()

```
int Backend::initializeSets (
    int length ) [inline]
```

Function initializing data structures in memory. These are initialized in order to be index-addressable in other functions.

##### Parameters

<i>length</i>	Number of sets of files that should be initialized.
---------------	---

##### Returns

Always returns 1 - function cannot fail execution.

#### 4.1.2.14 loadFiles()

```
int Backend::loadFiles (
    array< String^>^ fileNames,
    float cutoff,
    int whichMode ) [inline]
```

Function reading files from disc. Admittedly could be simplified together with GUI as it doesn't need two file selection buttons.

## Parameters

<i>fileNames</i>	Array of filenames. This list of files should be selected by the user in the GUI; using standard Windows libraries.
<i>cutoff</i>	Values of light intensities below which the intensity will be set to 0 when initializing data structures. Default -199.
<i>whichMode</i>	Integer describing which mode the programme is operating in. 1 - standard LIBS mode, 2 - calibration LIBS mode.

## Returns

Returns an integer indicating status. 0 - loading of files failed. 1 - load successful.

**4.1.2.15 numberOfValidSets()**

```
int Backend::numberOfValidSets ( ) [inline], [private]
```

Function for counting how many non-null elements there are in the sets. Example: user supplies 4 values of wavelengths. However, user also claims there are 3 sets to process - but user only submitted to set 1 and 3. This code works around that.

## Returns

Returns the number of valid sets.

**4.1.2.16 processFileIntoDictionary()**

```
int Backend::processFileIntoDictionary (
    Dictionary< float, float >^ dict,
    String^ filename,
    float cutoff ) [inline], [private]
```

Private function converting a raw file to a dictionary data structure.

## Parameters

<i>dict</i>	Pass by reference - dictionary into which a file should be parsed.
<i>filename</i>	Name of file to be processed into a datastructure.
<i>cutoff</i>	Value of cutoff, below which the intensity of a given datapoint will be brought down to 0. Default -199.

## Returns

Returns a status flag. 1 - file successfully processed. 0 - file read failed.

#### 4.1.2.17 removeWavelength()

```
int Backend::removeWavelength (
    float wavelength ) [inline]
```

Function removing an user-supplied wavelength from an internal data structure.

##### Parameters

<i>wavelength</i>	Wavelength to be removed.
-------------------	---------------------------

##### Returns

1 if removal successful, 0 otherwise. 0 should never be reached and this should be ensured in the [Window.h](#) file.

#### 4.1.2.18 returnDivisionFromTwoFirst()

```
float Backend::returnDivisionFromTwoFirst (
    List< float >^ givenList ) [inline], [private]
```

Function that divides first two items of a list. For programmer's convenience.

##### Parameters

<i>givenList</i>	List that the division is performed on.
------------------	---

##### Returns

Returns the result of the division.

#### 4.1.2.19 saveToFile()

```
int Backend::saveToFile (
    String^ name,
    bool selectedOnly,
    bool lowestPoint ) [inline]
```

Save a processed data structure to a file in standard LIBS mode.

##### Parameters

<i>name</i>	Name of file to save to.
<i>selectedOnly</i>	Save all wavelengths to a file; or only the ones selected by the user.
<i>lowestPoint</i>	Option describing whether the user is also interested in saving the lowest values in range to file.

**Returns**

Returns an integer describing success/fail of function. 1 - file saved; 0 - file locked by OS; -1 - file not saved because of user/programmer error.

**4.1.2.20 saveToFileCalibration()**

```
int Backend::saveToFileCalibration (
    String^ name,
    bool lowestPoint ) [inline]
```

Save a processed data structure to a file in calibration LIBS mode.

**Parameters**

<i>name</i>	Name of file to save to. Does not have to have a correct extension.
<i>lowestPoint</i>	Option describing whether the user is also interested in saving the lowest values in range to file.

**Returns**

Returns an integer describing success/fail of function. 1 - file saved; 0 - file locked by OS; -1 - file not saved because of user/programmer error.

**4.1.2.21 sumDictionaries()**

```
int Backend::sumDictionaries ( ) [inline], [private]
```

Private function summing all dictionaries into one resulting dictionaries, for when the strongest signal wants to be seen across n files. This function also initializes indexed keys, which in itself is crucial for O(1) operation - however, it would be good practice to have them be initialized in a separate function and decouple the code a bit.

**Returns**

Always returns 1, as function will be always successful if the programme execution reaches this point.

The documentation for this class was generated from the following file:

- C:/MEng/LIBSProcessing/LIBSProcessing/[Backend.h](#)

**4.2 LIBSProcessing::Window Class Reference**

Class handling the display of the user interface (later referred to as GUI or UI). Primarily has functions responsible for recording button presses.

```
#include <Window.h>
```

## Public Member Functions

- [Window](#) (void)

## Protected Member Functions

- [~Window](#) ()  
*Clean up any resources being used.*

## Protected Attributes

- [Backend](#) **b**

## Private Member Functions

- void [InitializeComponent](#) (void)  
*Automatically generated code for the use of Windows Forms Designer. Should not be modified directly by the user.*
- System::Void [elemSubmit\\_Click](#) (System::Object^ sender, System::EventArgs^ e)
- System::Void [waveSubmit\\_Click](#) (System::Object^ sender, System::EventArgs^ e)
- System::Void [saveFolderSelect\\_Click](#) (System::Object^ sender, System::EventArgs^ e)
- System::Void [removeWave\\_Click](#) (System::Object^ sender, System::EventArgs^ e)
- System::Void [preview\\_Click](#) (System::Object^ sender, System::EventArgs^ e)
- System::Void [saveToFile\\_Click](#) (System::Object^ sender, System::EventArgs^ e)
- System::Void [fileSelect\\_Click](#) (System::Object^ sender, System::EventArgs^ e)
- System::Void [fileSelect\\_setB\\_Click](#) (System::Object^ sender, System::EventArgs^ e)
- System::Void [Window\\_Load](#) (System::Object^ sender, System::EventArgs^ e)
- System::Void [standardToolStripMenuItem\\_Click](#) (System::Object^ sender, System::EventArgs^ e)
- System::Void [calibrationToolStripMenuItem\\_Click](#) (System::Object^ sender, System::EventArgs^ e)
- System::Void [howManySubmit\\_Click](#) (System::Object^ sender, System::EventArgs^ e)
- System::Void [addSetButton\\_Click](#) (System::Object^ sender, System::EventArgs^ e)
- void [handleSelection](#) (int selectionWindow)  
*Helper functions, not to clutter the main UI code - mainly with a single switch.*
- void [setCalibrationGroup](#) (bool value)
- void [setSetAddedGroup](#) (bool value)

## Private Attributes

- System::Windows::Forms::CheckBox ^ **saveSelectedBox**
- System::Windows::Forms::RadioButton ^ **highestCheckbox**
- System::Windows::Forms::RadioButton ^ **sumCheckbox**
- System::Windows::Forms::GroupBox ^ **groupBox1**
- System::Windows::Forms::TextBox ^ **noiseCutoff**
- System::Windows::Forms::Label ^ **label8**
- System::Windows::Forms::Label ^ **label9**
- System::Windows::Forms::Label ^ **cutoffLabel**
- System::Windows::Forms::MenuStrip ^ **menuStrip1**
- System::Windows::Forms::ToolStripMenuItem ^ **modeToolStripMenuItem**
- System::Windows::Forms::ToolStripMenuItem ^ **standardToolStripMenuItem**
- System::Windows::Forms::ToolStripMenuItem ^ **calibrationToolStripMenuItem**
- System::Windows::Forms::Label ^ **setALabel**

- System::Windows::Forms::Label ^ **setBlabel**
- System::Windows::Forms::Label ^ **selectFilesLabel\_setB**
- System::Windows::Forms::Button ^ **fileSelect\_setB**
- System::Windows::Forms::Label ^ **analyteLabel\_setB**
- System::Windows::Forms::TextBox ^ **analyteBox\_setB**
- System::ComponentModel::BackgroundWorker ^ **backgroundWorker1**
- System::Windows::Forms::Label ^ **setNumbersLabel**
- System::Windows::Forms::Label ^ **howManyLabel**
- System::Windows::Forms::Button ^ **howManySubmit**
- System::Windows::Forms::TextBox ^ **howManySets**
- System::Windows::Forms::ComboBox ^ **setsOfData**
- System::Windows::Forms::Label ^ **setsOfData\_label**
- System::Windows::Forms::Button ^ **addSetButton**
- System::Windows::Forms::Label ^ **Rscore**
- System::Windows::Forms::TextBox ^ **rangeLowerInput**
- System::Windows::Forms::Label ^ **label1**
- System::Windows::Forms::GroupBox ^ **groupBox2**
- System::Windows::Forms::RadioButton ^ **lowerRangeYes**
- System::Windows::Forms::RadioButton ^ **radioButton2**
- System::Windows::Forms::TextBox ^ **waveEdit**
- System::Windows::Forms::Button ^ **waveSubmit**
- System::Windows::Forms::Label ^ **label2**
- System::Windows::Forms::TextBox ^ **rangeInput**
- System::Windows::Forms::Label ^ **label3**
- System::Windows::Forms::ToolTip ^ **toolTip1**
- System::Windows::Forms::ComboBox ^ **allWavelengths**
- System::Windows::Forms::Label ^ **label4**
- System::Windows::Forms::Button ^ **removeWave**
- System::Windows::Forms::Button ^ **saveFolderSelect**
- System::Windows::Forms::FolderBrowserDialog ^ **folderBrowser**
- System::Windows::Forms::Label ^ **label5**
- System::Windows::Forms::TextBox ^ **savePath**
- System::Windows::Forms::TextBox ^ **nameOfFile**
- System::Windows::Forms::Label ^ **label6**
- System::Windows::Forms::Button ^ **saveToFile**
- System::Windows::Forms::Button ^ **fileSelect**
- System::Windows::Forms::Label ^ **label7**
- System::Windows::Forms::Button ^ **preview**
- System::Windows::Forms::Label ^ **noOfFiles**
- System::Windows::Forms::OpenFileDialog ^ **fileOpener**
- System::ComponentModel::IContainer ^ **components**

### 4.2.1 Detailed Description

Class handling the display of the user interface (later referred to as GUI or UI). Primarily has functions responsible for recording button presses.

### 4.2.2 Constructor & Destructor Documentation

### 4.2.2.1 Window()

```
LIBSProcessing::Window::Window (
    void ) [inline]
```

Initialization of the GUI.

## 4.2.3 Member Function Documentation

### 4.2.3.1 addSetButton\_Click()

```
System::Void LIBSProcessing::Window::addSetButton_Click (
    System::Object^ sender,
    System::EventArgs^ e ) [inline], [private]
```

GUI handler - set has been added, initialize it in the backend.

#### Parameters

<i>sender</i>	NA
<i>e</i>	NA

#### Returns

### 4.2.3.2 calibrationToolStripMenuItem\_Click()

```
System::Void LIBSProcessing::Window::calibrationToolStripMenuItem_Click (
    System::Object^ sender,
    System::EventArgs^ e ) [inline], [private]
```

GUI handler - select calibration mode in toolbar.

#### Parameters

<i>sender</i>	
<i>e</i>	

#### Returns

#### 4.2.3.3 fileSelect\_Click()

```
System::Void LIBSProcessing::Window::fileSelect_Click (
    System::Object^ sender,
    System::EventArgs^ e ) [inline], [private]
```

GUI handler - select files.

##### Parameters

<i>sender</i>	NA
<i>e</i>	NA

##### Returns

Void.

#### 4.2.3.4 fileSelect\_setB\_Click()

```
System::Void LIBSProcessing::Window::fileSelect_setB_Click (
    System::Object^ sender,
    System::EventArgs^ e ) [inline], [private]
```

GUI handler - select files - calibration mode.

##### Parameters

<i>sender</i>	
<i>e</i>	

##### Returns

Void.

#### 4.2.3.5 handleSelection()

```
void LIBSProcessing::Window::handleSelection (
    int selectionWindow ) [inline], [private]
```

Helper functions, not to clutter the main UI code - mainly with a single switch.

GUI helper function - verify whether files selected by the user are valid & check the cutoff value.

##### Parameters

<i>selectionWindow</i>	
------------------------	--



#### 4.2.3.6 howManySubmit\_Click()

```
System::Void LIBSProcessing::Window::howManySubmit_Click (
    System::Object^ sender,
    System::EventArgs^ e ) [inline], [private]
```

GUI handler - see how many sets the user is trying to initialize & then do so.

##### Parameters

<i>sender</i>	NA
<i>e</i>	NA

##### Returns

Void.

#### 4.2.3.7 InitializeComponent()

```
void LIBSProcessing::Window::InitializeComponent (
    void ) [inline], [private]
```

Automatically generated code for the use of Windows Forms Designer. Should not be modified directly by the user.

Required method for Designer support - do not modify the contents of this method with the code editor.  
Functions will be calling the backend internally.

#### 4.2.3.8 preview\_Click()

```
System::Void LIBSProcessing::Window::preview_Click (
    System::Object^ sender,
    System::EventArgs^ e ) [inline], [private]
```

GUI handler - process the submitted data.

##### Parameters

<i>sender</i>	NA
<i>e</i>	NA

##### Returns

Void.

#### 4.2.3.9 removeWave\_Click()

```
System::Void LIBSProcessing::Window::removeWave_Click (
    System::Object^ sender,
    System::EventArgs^ e ) [inline], [private]
```

GUI handler - remove wavelength from analysis.

##### Parameters

<i>sender</i>	N/A
<i>e</i>	N/A

##### Returns

Void.

#### 4.2.3.10 saveFolderSelect\_Click()

```
System::Void LIBSProcessing::Window::saveFolderSelect_Click (
    System::Object^ sender,
    System::EventArgs^ e ) [inline], [private]
```

GUI handler - save the folder path for user results..

##### Parameters

<i>sender</i>	N/A
<i>e</i>	N/A

##### Returns

Void.

#### 4.2.3.11 saveToFile\_Click()

```
System::Void LIBSProcessing::Window::saveToFile_Click (
    System::Object^ sender,
    System::EventArgs^ e ) [inline], [private]
```

GUI handler - save to a file.

##### Parameters

<i>sender</i>	NA
<i>e</i>	NA

**Returns**

Void.

**4.2.3.12 setCalibrationGroup()**

```
void LIBSProcessing::Window::setCalibrationGroup (
    bool value ) [inline], [private]
```

GUI helper function - set buttons to Calibration mode.

**Parameters**

<i>value</i>	true if enable calibration group, false otherwise.
--------------	--

**4.2.3.13 setSetAddedGroup()**

```
void LIBSProcessing::Window::setSetAddedGroup (
    bool value ) [inline], [private]
```

GUI helper function - set buttons to active/inactive after number of sets have been added.

**Parameters**

<i>value</i>	True if enable buttons in set, false if disable.
--------------	--

**4.2.3.14 standardToolStripMenuItem\_Click()**

```
System::Void LIBSProcessing::Window::standardToolStripMenuItem_Click (
    System::Object^ sender,
    System::EventArgs^ e ) [inline], [private]
```

GUI handler - select standard mode in toolbar.

**Parameters**

<i>sender</i>	NA
<i>e</i>	NA

**Returns**

Void.

#### 4.2.3.15 waveSubmit\_Click()

```
System::Void LIBSProcessing::Window::waveSubmit_Click (
    System::Object^ sender,
    System::EventArgs^ e ) [inline], [private]
```

GUI handler - submitting custom wavelengths to the list.

##### Parameters

<i>sender</i>	N/A
<i>e</i>	N/A

##### Returns

Void.

#### 4.2.3.16 Window\_Load()

```
System::Void LIBSProcessing::Window::Window_Load (
    System::Object^ sender,
    System::EventArgs^ e ) [inline], [private]
```

GUI handler - load the window. Currently empty.

##### Parameters

<i>sender</i>	NA
<i>e</i>	NA

##### Returns

Void.

The documentation for this class was generated from the following file:

- C:/MEng/LIBSProcessing/LIBSProcessing/[Window.h](#)

## Chapter 5

# File Documentation

### 5.1 C:/MEng/LIBSProcessing/LIBSProcessing/Backend.h File Reference

File handling all the backend functions, invisible to the user.

#### Classes

- class [Backend](#)

*Class handling the backend functions of the entire programme. Public variables are expanded on within the code.*

#### Macros

- `#define DATASIZE 26607`
- `#define LINESTOSKIP 105`

#### 5.1.1 Detailed Description

File handling all the backend functions, invisible to the user.

Author

PR

Date

April 2023

## 5.2 Backend.h

[Go to the documentation of this file.](#)

```

00001 /*****
00008 #pragma once
00009
00010
00011 using namespace System;
00012 using namespace System::IO;
00013 using namespace System::Windows::Forms;
00014 using namespace System::Collections::Generic;
00015
00016
00017 //how many datapoints are there?
00018 #define DATASIZE 26607
00019 #define LINESKIP 105
00020
00021
00022 public ref class Backend {
00023
00024
00025 public:
00026     String^ directory;
00027     String^ nameOfFile;
00028     List<String>^ filesToExtract;
00029     List<Dictionary<float, float>^> listOfDictionaries; //dictionary has O(1) access time when
00030     passed the key. Holds all files.
00031     Dictionary<float, float>^ result; //summed dictionary of provided files.
00032     List<float>^ indexedKeys; //Indexing the keys in the dictionary
00033     for easy sequential access.
00034     Dictionary<float, float>^ presentToUserResult; //Final-final result that is saved to
00035     the computer.
00036     List<float>^ selectedWavelengths; //A list of floats that the user
00037     selected for processing.
00038     List<float>^ userSelectionsToKeys; //A list which holds the user
00039     selections, but as keys that can be used with dictionaries.
00040     List<int>^ userSelectionsIndexes; //As dictionaries are unsorted, we
00041     need to keep track of the indexes as well.
00042
00043     List<Dictionary<float, float>^> listOfResultsForFiles; //List holding results for selected
00044     wavelengths for each file.
00045     List<float>^ listOfAveragedIndividualResults; //List holding the individual results.
00046
00047     //Calibration mode variables.
00048     List<String>^ filesToExtract_B; //Temporarily selected files &
00049     temporarily initialized dictionary when user is selecting.
00050     List<Dictionary<float, float>^> listOfDictionaries_B;
00051
00052     //multi-set processing.
00053     List<String>^ metadata; //Metadata about a set to be displayed
00054     about a given set.
00055     List< List<Dictionary<float, float>^>^> listOfSets; //Processed files as sets inside of a
00056     list.
00057     List<float>^ listOfConcentrations; //Concentrations as supplied by the
00058     user.
00059     List<float>^ listOfAverages; //result of average division of first
00060     wavelength/second wavelength for each set of files.
00061
00062     List< List<Dictionary<float, float>^>^> listOfProcessedSets; //List holding results for
00063     selected wavelengths for each file, for each set.
00064     List<List<float>^>^> listOfAveragedIndividualResults_sets; //List holding individual results
00065     for each file for each set - average from division.
00066
00067     float global_r2;
00068     Backend() {
00069         directory = Application::StartupPath;
00070         //for now.
00071         nameOfFile = "default.csv";
00072         //initialize the (now empty) dynamically allocated array.
00073         selectedWavelengths = gcnew List<float>();
00074     }
00075     int saveToFile(String^ name, bool selectedOnly, bool lowestPoint) {
00076         //see if user put in any input; if not, do a default
00077         if (name) {
00078             nameOfFile = "\\\"+name;
00079         }
00080         else {
00081             nameOfFile = "\\\"+System::DateTime::Now.ToString("dd_MM_hhmm");
00082         }
00083         if (!name->EndsWith(".csv")) {
00084             nameOfFile = nameOfFile + ".csv";
00085         }
00086
00087         Dictionary<float, float>^ whatToSave = selectedOnly ? presentToUserResult : result;

```

```

00093     StreamWriter^ sw;
00094     try {
00095         sw = gcnew StreamWriter(directory + nameOfFile);
00096     }
00097     catch (...) {
00098         return 0;
00099     }
00100     //loop through all the keyss
00101     try {
00102         sw->Write("AVERAGE OF SUM OF FILES,\n");
00103         sw->Write("Highest key in range, result\n");
00104         for each (float key in whatToSave->Keys) {
00105             sw->Write(Convert::ToString(key));
00106             sw->Write(',');
00107             sw->Write(Convert::ToString(result[key]));
00108             sw->Write('\n');
00109         }
00110         sw->Write("AVERAGE OF INDIVIDUAL HIGHEST WITHIN RANGE\n");
00111         int i = 0, j = 0;
00112         sw->Write("Supp. wavelength,Result\n");
00113         for each (float result in listOfAveragedIndividualResults) {
00114             if (i % 2 != 0 && lowestPoint) {
00115                 i++;
00116                 continue;
00117             }
00118             sw->Write(Convert::ToString(userSelectionsToKeys[j])+"",Convert::ToString(result)+"");
00119             i++;
00120             j++;
00121         }
00122         sw->Write("\n");
00123         sw->Write("Division of averaged first intensity over the
other:,,,,"+Convert::ToString(returnDivisionFromTwoFirst(listOfAveragedIndividualResults))+"\n\n");
00124         sw->Write("Individual results\n");
00125         if (lowestPoint) {
00126             sw->Write(",,,(optional), (optional)\n");
00127         }
00128         if (lowestPoint) {
00129             sw->Write("File no., wavelength, intensity,Lowest key in range, result,peak?\n");
00130         }
00131         else {
00132             sw->Write("File no., wavelength,intensity,\n");
00133         }
00134     }
00135     i = 1;
00136     for each (Dictionary<float, float> ^ fileAsDictionary in listOfResultsForFiles) {
00137         sw->Write("File " + (Convert::ToString(i)) + ",");
00138         //this is quite bad, as no calculations should be done inside this function. However,
00139         I am too tired to do it otherwise right now.
00140         j = 0;
00141         float tempHighest, tempLowest;
00142         for each (float key in fileAsDictionary->Keys) {
00143             if (lowestPoint) {
00144                 if (j % 2 == 0) {
00145                     tempHighest = fileAsDictionary[key];
00146                 }
00147                 if (j % 2 != 0) {
00148                     tempLowest = fileAsDictionary[key];
00149                 }
00150             }
00151             sw->Write(Convert::ToString(key) + ", " +
Convert::ToString(fileAsDictionary[key])+"");
00152             j++;
00153             if (j % 2 == 0 && j > 0 && lowestPoint) {
00154                 if (tempHighest > 3 * tempLowest) {
00155                     sw->Write("yes,");
00156                 }
00157                 else {
00158                     sw->Write("no,");
00159                 }
00160             }
00161         }
00162         sw->Write("\n");
00163         i++;
00164     }
00165 }
00166 catch (...) {
00167     //data structure not initialized
00168     sw->Close();
00169     return -1;
00170 }
00171 sw->Close();
00172 return 1;
00173 }
00174 }
00182 int saveToFileCalibration(String^ name, bool lowestPoint) {

```

```

00183         if (name) {
00184             nameOfFile = "\\\" + name;
00185         }
00186     else {
00187         nameOfFile = "\\\" + System::DateTime::Now.ToString("dd_MM_hhmm");
00188     }
00189     if (!name->EndsWith(".csv")) {
00190         nameOfFile = nameOfFile + ".csv";
00191     }
00192     StreamWriter^ sw;
00193     try {
00194         sw = gcnew StreamWriter(directory + nameOfFile);
00195     }
00196     catch (...) {
00197         return 0;
00198     }
00199     //write the header of the excel file.
00200     try {
00201         int elems = numberOfValidSets(); int whichKey = 0;
00202         bool differentDivisors = false;
00203         if (userSelectionsToKeys->Count / 2 == elems) {
00204             differentDivisors = true;
00205         }
00206         //write headers
00207         sw->Write(",Average (sum first intensity then sum second intensity then divide one by the
other),");
00208         sw->Write("dividend,");
00209         sw->Write("divisor,");
00210         sw->Write("Concentration\n");
00211         for (int i = 0; i < metadata->Count; i++) {
00212             //coded for "no set input"
00213             sw->Write("S" + Convert::ToString(i + 1) + ",");
00214             if (listOfAverages[i] == -1) {
00215                 sw->Write(",",");
00216                 continue;
00217             }
00218             sw->Write(Convert::ToString(listOfAverages[i]) + ",");
00219             if (differentDivisors) {
00220                 sw->Write(Convert::ToString(userSelectionsToKeys[whichKey]) + ",");
00221                 sw->Write(Convert::ToString(userSelectionsToKeys[whichKey + 1]) + ",");
00222                 whichKey += 2;
00223             }
00224             else {
00225                 sw->Write(Convert::ToString(userSelectionsToKeys[0]) + ",");
00226                 sw->Write(Convert::ToString(userSelectionsToKeys[1]) + ",");
00227             }
00228             sw->Write(Convert::ToString(listOfConcentrations[i]) + "\n");
00229         }
00230         sw->Write("R2 score," + Convert::ToString(global_r2) + "\n");
00231         sw->Write("Individual results\n");
00232         if (lowestPoint) {
00233             sw->Write(",HIGHEST->,,LOWEST->,,\n");
00234         }
00235         sw->Write("File no., divid. wavelength, intensity, divisor wavelength, intensity,");
00236         if (lowestPoint) {
00237             sw->Write(",,1.peak?,2.peak?,");
00238         }
00239         for (int i = 0; i < metadata->Count; i++) {
00240             sw->Write("\nS" + Convert::ToString(i + 1) + "\n");
00241             if (listOfAverages[i] == -1) {
00242                 sw->Write(",",");
00243                 continue;
00244             }
00245             int j = 1;
00246             for each (Dictionary<float, float> ^ fileAsDictionary in listOfProcessedSets[i]) {
00247                 sw->Write("File " + (Convert::ToString(j)) + ",");
00248                 int k = 0;
00249                 float firstHighest, firstLowest, secondHighest, secondLowest;
00250                 for each (float key in fileAsDictionary->Keys) {
00251                     if (k % 4 == 0) {
00252                         firstHighest = fileAsDictionary[key];
00253                     }
00254                     else if (k % 4 == 1) {
00255                         secondHighest = fileAsDictionary[key];
00256                     }
00257                     else if (k % 4 == 2) {
00258                         firstLowest = fileAsDictionary[key];
00259                     }
00260                     else if (k % 4 == 3) {
00261                         secondLowest = fileAsDictionary[key];
00262                     }
00263                     sw->Write(Convert::ToString(key) + "," +
Convert::ToString(fileAsDictionary[key]) + ",");
00264                     k++;
00265                     if (k % 4 == 0 && k > 0 && lowestPoint) {
00266                         if (firstHighest > 3 * firstLowest) {
00267                             sw->Write(",yes,");

```



```

00268         }
00269         else {
00270             sw->Write(",no,");
00271         }
00272         if (secondHighest > 3 * secondLowest) {
00273             sw->Write("yes,");
00274         }
00275         else {
00276             sw->Write("no,");
00277         }
00278     }
00279     }
00280     j++;
00281     sw->Write("\n");
00282 }
00283 }
00284 }
00285 catch (...) {
00286     //Data structures not initialized yet.
00287     sw->Close();
00288     return -1;
00289 }
00290 sw->Close();
00291
00292
00293     return 1;
00294 }
00303 int loadFiles(array<String^>^ fileNames, float cutoff, int whichMode) {
00304     List<String^>^ files;
00305     if (whichMode == 1) {
00306         //reinitialize the arrays each time.
00307         filesToExtract = gcnew List<String^>();
00308         //point to the relevant array.
00309         files = filesToExtract;
00310     }
00311     else if (whichMode == 2) {
00312         filesToExtract_B = gcnew List<String^>();
00313         files = filesToExtract_B;
00314     }
00315     else {
00316         //this should never be reached.
00317         files = gcnew List<String^>();
00318         return 0;
00319     }
00320     for each (String^ filename in fileNames) {
00321         if (filename->EndsWith(".asc")) {
00322             files->Add(filename);
00323         }
00324         else {
00325             return 0;
00326         }
00327     }
00328     return initializeMemoryFiles(cutoff, whichMode);
00329 }
00330 }
00337 int initializeSets(int length) {
00338     metadata = gcnew List<String^>(length);
00339     listOfSets = gcnew List<List<Dictionary<float, float>^>^>(length);
00340     listOfConcentrations = gcnew List<float>(length);
00341     listOfAverages = gcnew List<float>(length);
00342
00343     for (int i = 0; i < length; i++) {
00344         metadata->Add(Convert::ToString(i+1) + ". THIS SET IS EMPTY.");
00345         //I'd like these structures to be index-addressable after they have been initialized.
00346         listOfSets->Add(nullptr);
00347         listOfConcentrations->Add(-1);
00348         listOfAverages->Add(-1);
00349     }
00350 }
00351
00352     return 1;
00353 }
00354 }
00363 int addSetToSets(float concentration, int index, float cutoff) {
00364     try {
00365         listOfConcentrations[index] = concentration;
00366         listOfSets[index] = gcnew List<Dictionary<float, float>^>();
00367         listOfSets[index]->AddRange(listOfDictionaries_B);
00368         metadata[index] = Convert::ToString(index + 1) + ".SET: " + listOfDictionaries_B->Count +
" FILES, CUTOFF " +
00369             cutoff + ", CONT." + concentration;
00370         return 1;
00371     }
00372     catch (...) {
00373         return 0;
00374     }
00375 }

```

```

00376
00377
00389     int getRequestedSpectraStandardMode(int option, float range, bool doLowerRange, float lowerRange)
00390     {
00391         //first, keys CLOSEST to the value that the user input must be found.
00392         findKeys();
00393         findRequestedValues(option, range, doLowerRange, lowerRange);
00394         return 1;
00395     }
00406     int getRequestedSpectraCalibrationMode(int option, float range, bool doLowerRange, float
00407     lowerRange) {
00408         //if we have less than 1 item, discard. We will check for whether each set has it's own
00409         wavelenghts later.
00410         if (selectedWavelengths->Count < 1) {
00411             return 0;
00412         }
00413         //first, keys CLOSEST to the value that the user input must be found.
00414         findKeys();
00415         findRequestedValuesCalibration(option, range, doLowerRange, lowerRange);
00416         return 1;
00417     }
00421     int getAveragedSpectra() {
00422         return sumDictionaries();
00423     }
00430     int addWavelength(float wavelength) {
00431         if (wavelength < 200.93 || wavelength > 1031.86) {
00432             return 0;
00433         }
00434         selectedWavelengths->Add(wavelength);
00435         return 1;
00436     }
00443     int removeWavelength(float wavelength) {
00444         if (!selectedWavelengths->Contains(wavelength)) {
00445             return 0;
00446         }
00447         selectedWavelengths->Remove(wavelength);
00448         return 1;
00449     }
00450
00456     float getRSquared() {
00457         //calculate average intensity & average concentration; all concentrations.
00458         float runningSumInt = 0, runningSumCon = 0;
00459         int goodEntries = 0;
00460
00461         for (int i = 0; i < listOfConcentrations->Count; i++) {
00462             if (listOfConcentrations[i] == -1) {
00463                 continue;
00464             }
00465             goodEntries++;
00466             runningSumInt += listOfAverages[i];
00467             runningSumCon += listOfConcentrations[i];
00468         }
00469         float averageConcentration = runningSumCon / goodEntries;
00470         float averageIntensity = runningSumInt / goodEntries;
00471         //now, we have everything we need. Plug it into the R score formula.
00472         //1. Dividend - sum(Ii-Iavg)(Ci-Cavg)
00473         float div = 0;
00474         for (int i = 0; i < listOfConcentrations->Count; i++) {
00475             if (listOfConcentrations[i] == -1) {
00476                 continue;
00477             }
00478             div += (listOfConcentrations[i] - averageConcentration)*(listOfAverages[i] -
00479             averageConcentration);
00480         }
00481         //2. Divisor - sqrt(sum(Ii-Iavg)^2)*sqrt(sum(Ci-Cavg)^2)
00482         float divisor;
00483         float sumI = 0, sumC = 0;
00484         for (int i = 0; i < listOfConcentrations->Count; i++) {
00485             if (listOfConcentrations[i] == -1) {
00486                 continue;
00487             }
00488             sumC += (listOfConcentrations[i] - averageConcentration)*(listOfConcentrations[i] -
00489             averageConcentration);
00490             sumI += (listOfAverages[i] - averageIntensity) * (listOfAverages[i] - averageIntensity);
00491         }
00492         sumC = Convert::ToSingle(Math::Sqrt(sumC));
00493         sumI = Convert::ToSingle(Math::Sqrt(sumI));
00494         divisor = sumC * sumI;
00495         //3. Finally, calculate the R score.
00496         float R = div / divisor;
00497         global_r2 = R * R;
00498         return R * R;
00499     }
00500
00501

```

```

00502
00503 private:
00512 int processFileIntoDictionary(Dictionary<float, float>^ dict, String^ filename, float cutoff) {
00513     //file might be opened by another process
00514     StreamReader^ sr;
00515     try {
00516         sr = gcnew StreamReader(filename);
00517     }
00518     catch (...) {
00519         //0 - file used by another process
00520         return 0;
00521     }
00522
00523
00524     String^ line;
00525     array<String^>^ thisKeyValArray;
00526     int i = 0;
00527
00528     //read file line by line.
00529     while (line = sr->ReadLine()) {
00530         if (i < LINESKIP) {
00531             i++;
00532             continue;
00533         }
00534         //split line into two strings
00535         thisKeyValArray = line->Split(',', 2);
00536         //and add to the dictionary.
00537         if (Convert::ToSingle(thisKeyValArray[1]) < cutoff) {
00538             dict->Add(Convert::ToSingle(thisKeyValArray[0]), 0);
00539         }
00540         else {
00541             dict->Add(Convert::ToSingle(thisKeyValArray[0]),
00542                 Convert::ToSingle(thisKeyValArray[1]));
00543         }
00544     }
00545     return 1;
00546 }
00555 int initializeMemoryFiles(float cutoff, int whichMode) {
00556     List<String^>^ files;
00557     int lengthOfListOfMaps;
00558     List<Dictionary<float, float>^>^ currentListOfDictionaries;
00559     if (whichMode == 1) {
00560         lengthOfListOfMaps = filesToExtract->Count;
00561         listOfDictionaries = gcnew List<Dictionary<float, float>^>(lengthOfListOfMaps);
00562         //point to the relevant array.
00563         currentListOfDictionaries = listOfDictionaries;
00564         files = filesToExtract;
00565     }
00566
00567     else if (whichMode == 2) {
00568         lengthOfListOfMaps = filesToExtract_B->Count;
00569         listOfDictionaries_B = gcnew List<Dictionary<float, float>^>(lengthOfListOfMaps);
00570         currentListOfDictionaries = listOfDictionaries_B;
00571         files = filesToExtract_B;
00572     }
00573
00574     else {
00575         //this should never be reached.
00576         currentListOfDictionaries = gcnew List<Dictionary<float, float>^>();
00577         return 0;
00578     }
00579
00580     //create and populate a dictionary of data for each of the files.
00581     for (int i = 0; i < lengthOfListOfMaps; i++) {
00582         currentListOfDictionaries->Add(gcnew Dictionary<float, float>(DATASIZE - LINESKIP));
00583         processFileIntoDictionary(currentListOfDictionaries[i], files[i], cutoff);
00584     }
00585     return 1;
00586 }
00587
00595 int sumDictionaries() {
00596     //Even if there was a dictionary before, garbage collect it and create a new one.
00597     result = gcnew Dictionary<float, float>(DATASIZE - LINESKIP);
00598     indexedKeys = gcnew List<float>(DATASIZE - LINESKIP);
00599     if (listOfDictionaries == nullptr) {
00600         //in case user is operating in mode B.
00601         listOfDictionaries = listOfDictionaries_B;
00602     }
00603
00604     for each (float key in listOfDictionaries[0]->Keys) {
00605         float tempValue = 0;
00606         Dictionary<float, float>^ temp;
00607         for (int j = 0; j < listOfDictionaries->Count; j++) {
00608             //it's upset when I'm doing double dereferencing. It's a good thing temp is a pointer
00609             //of overhead; still the reassignments are taking up processor cycles
00610             temp = listOfDictionaries[j];

```

```

00611         tempValue += temp[key];
00612     }
00613     //average of n spectra.
00614     result->Add(key, tempValue / listOfDictionaries->Count);
00615     indexedKeys->Add(key);
00616 }
00617 return 1;
00618
00619
00620 }
00621
00622 int findKeys() {
00623     //if (userSelectionsToKeys == nullptr) {
00624     //every time we process again; we reset the list - easier to handle
00625     userSelectionsToKeys = gcnew List<float>();
00626     userSelectionsIndexes = gcnew List<int>();
00627     //}
00628
00629     for each (float wavelength in selectedWavelengths) {
00630         //guesstimate the index based the distances of the datapoints.
00631         //A line of best fit that maps index to wavelength is  $y = 201.96 \cdot e^{(6 \cdot 10^{-5} \cdot x)}$ , as per
Excel.
00632         //I have modified it slightly after manually inspecting the mappings, and the formula for
loosely recovering
00633         //the index is presented below.
00634         int assumedIndex = Convert::ToInt32(Math::Floor(Math::Pow(10, 5) / 6.21 *
Math::Log(wavelength / 200.9381)));
00635         //now, see and compare. This is heavily amortized computationally, and will not go over
more than 200 iterations for each point.
00636         //1.Rarest scenario. We have estimated exactly the correct key
00637         float tempDiff, newDiff;
00638         if (indexedKeys[assumedIndex] == wavelength) {
00639             userSelectionsToKeys->Add(wavelength);
00640             userSelectionsIndexes->Add(assumedIndex);
00641             continue;
00642         }
00643         //2. We have estimated the key to be too large. Try a lower key one by one.
00644         else if (indexedKeys[assumedIndex] >= wavelength) {
00645             while (indexedKeys[assumedIndex] >= wavelength) {
00646                 tempDiff = Math::Abs(indexedKeys[assumedIndex] - wavelength);
00647                 assumedIndex -- 1;
00648                 newDiff = Math::Abs(indexedKeys[assumedIndex] - wavelength);
00649             }
00650             //now, we are pretty much spot on. Just check whether to select assumedIndex or
assumedIndex+1.
00651             if (tempDiff > newDiff) {
00652                 userSelectionsToKeys->Add(indexedKeys[assumedIndex]);
00653                 userSelectionsIndexes->Add(assumedIndex);
00654             }
00655             else {
00656                 userSelectionsToKeys->Add(indexedKeys[assumedIndex + 1]);
00657                 userSelectionsIndexes->Add(assumedIndex+1);
00658             }
00659             continue;
00660         }
00661         //3. We have estimated the key to be too small. Try a higher key.
00662         else {
00663             while (indexedKeys[assumedIndex] <= wavelength) {
00664                 tempDiff = Math::Abs(indexedKeys[assumedIndex] - wavelength);
00665                 assumedIndex += 1;
00666                 newDiff = Math::Abs(indexedKeys[assumedIndex] - wavelength);
00667             }
00668             if (tempDiff > newDiff) {
00669                 userSelectionsToKeys->Add(indexedKeys[assumedIndex]);
00670                 userSelectionsIndexes->Add(assumedIndex);
00671             }
00672             else {
00673                 userSelectionsToKeys->Add(indexedKeys[assumedIndex - 1]);
00674                 userSelectionsIndexes->Add(assumedIndex-1);
00675             }
00676             continue;
00677         }
00678     }
00679 }
00680 //success.
00681 return 1;
00682
00683 }
00684
00685 int findRequestedValues(int option, float range, bool doLowerRange, float lowerRange) {
00686     //reset the "presentToUser" dictionary.
00687     presentToUserResult = gcnew Dictionary<float, float>();
00688     float rangeEachWay = range / 2;
00689     //i must be kept track of - it indicates which key we're currently looking at
00690     int i = 0;
00691     for each (float key in userSelectionsToKeys) {
00692         //first, find it for the averaged dictionaries
00693         Tuple<float, float> retVal = findHighestKeyValuePair(i, key, option, rangeEachWay, true,
result);

```

```

00708         //now we have found the highest value. Add to the proper result dictionary.
00709         presentToUserResult->Add(retVal->Item1, retVal->Item2);
00710         i++;
00711     }
00712
00713     //now, find the results for individual files
00714     int j = 0; //keeping track of which file we're in right now
00715     listOfResultsForFiles = gcnew List<Dictionary<float, float>>^(listOfDictionaries->Count);
00716     for each (Dictionary<float, float> ^ fileAsDictionary in listOfDictionaries) {
00717         //TODO - have a size predetermined for this dictionary
00718         listOfResultsForFiles->Add(gcnew Dictionary<float, float>());
00719         i = 0;
00720         for each (float key in userSelectionsToKeys) {
00721             Tuple<float, float>^ retVal = findHighestKeyValuePair(i, key, option, rangeEachWay,
00722                 true, fileAsDictionary);
00723             Tuple<float, float>^ retValLower;
00724             if (doLowerRange) {
00725                 retValLower = findHighestKeyValuePair(i, key, option, lowerRange/2, false,
00726                     fileAsDictionary);
00727             }
00728             listOfResultsForFiles[j]->Add(retVal->Item1, retVal->Item2);
00729             if (doLowerRange) {
00730                 listOfResultsForFiles[j]->Add(retValLower->Item1, retValLower->Item2);
00731             }
00732             i++;
00733         }
00734         j++;
00735     }
00736     //finally, average out the individual elements
00737     listOfAveragedIndividualResults = averageIndividualKeyValuePairs(listOfResultsForFiles);
00738     return 1;
00739 }
00740 int findRequestedValuesCalibration(int option, float range, bool doLowerRange, float lowerRange) {
00741     float rangeEachWay = range / 2;
00742     //first, we must measure what kind of values have been selected in the "Currently selected
00743     wavelenghts" box...
00744     bool differentDivisors = false;
00745     int elems = numberOfValidSets();
00746     //check if we have as many selected pairs of wavelenghts as sets
00747     if (userSelectionsToKeys->Count / 2 == elems) {
00748         differentDivisors = true;
00749     }
00750
00751     int whichKey = 0, curr = 0;
00752     float divisor, dividend;
00753     bool singleMode = false;
00754
00755     //case of no division - code still runs, but it divides by 1 (i. e. returns intensity)
00756     if (userSelectionsToKeys->Count == 1) {
00757         dividend = userSelectionsToKeys[0];
00758         divisor = 1;
00759         //makeshift solution - it's quite bad practice since this key does not exist;
00760         //but it works out for saving a file
00761         userSelectionsToKeys->Add(1);
00762         userSelectionsIndexes->Add(-1);
00763         singleMode = true;
00764     }
00765
00766     else if (!differentDivisors) {
00767         dividend = userSelectionsToKeys[0];
00768         divisor = userSelectionsToKeys[1];
00769     }
00770
00771     //...We have now measured what kind of values have been selected in the "Currently selected
00772     wavelenghts" box.
00773     //outer loop - run through all the sets of files.
00774     listOfProcessedSets = gcnew List<List<Dictionary<float, float>>>^();
00775     int i, j = 0;
00776     for each (List<Dictionary<float, float>> ^ set in listOfSets) {
00777         j = 0;
00778         listOfProcessedSets->Add(gcnew List<Dictionary<float, float>>^());
00779         //first, handle the first dictionary.
00780         if (set == nullptr) {
00781             i++;
00782             continue;
00783         }
00784         if (differentDivisors) {
00785             dividend = userSelectionsToKeys[whichKey];
00786             divisor = userSelectionsToKeys[whichKey + 1];
00787         }
00788         for each (Dictionary<float, float> ^ fileAsDictionary in set) {
00789             listOfProcessedSets[i]->Add(gcnew Dictionary<float, float> ());
00790
00791             //We now definitely have the correct key for dividend and divisor - now, find the
00792             highest key/value pair within range
00793             Tuple<float, float>^ retValDividend = findHighestKeyValuePair(whichKey, dividend,
00794                 option, rangeEachWay, true, fileAsDictionary);

```

```

00799         Tuple<float, float>^ retValDividendLower;
00800         Tuple<float, float>^ retValDivisorLower;
00801         if (doLowerRange) {
00802             retValDividendLower = findHighestKeyValuePair(whichKey, dividend, option,
lowerRange/2, false, fileAsDictionary);
00803         }
00804         Tuple<float, float>^ retValDivisor;
00805         if (!singleMode) {
00806             retValDivisor = findHighestKeyValuePair(whichKey+1, divisor, option, rangeEachWay,
true, fileAsDictionary);
00807             if (doLowerRange) {
00808                 retValDivisorLower = findHighestKeyValuePair(whichKey+1, divisor, option,
lowerRange / 2, false, fileAsDictionary);
00809             }
00810         }
00811         //res = dict[dividend] / (singleMode ? 1 : dict[divisor]);
00812         //runningSum += res;
00813         //pointer - will retain information
00814         List<Dictionary<float, float>^> temp = listOfProcessedSets[i];
00815         temp[j]->Add(retValDividend->Item1, retValDividend->Item2);
00816         if (!singleMode) {
00817             temp[j]->Add(retValDivisor->Item1, retValDivisor->Item2);
00818         }
00819         else{
00820             //adding a value of 1 with a key of 1 to divide by 1 for result. Key of 1 works as
a flag
00821             temp[j]->Add(1, 1);
00822         }
00823         if (doLowerRange) {
00824             //third and fourth column will be reserved for the smallest value within the given
range.
00825             temp[j]->Add(retValDividendLower->Item1, retValDividendLower->Item2);
00826             if (!singleMode) {
00827                 temp[j]->Add(retValDivisorLower->Item1, retValDivisorLower->Item2);
00828             }
00829         }
00830         j++;
00831     }
00832     if (differentDivisors) {
00833         whichKey += 2;
00834     }
00835     i++;
00836 }
00837 }
00838 //we have now found the wavelength of biggest intensity in range. We may proceed to
calculating the averages for each set.
00839 List<List<float>^>() listOfAveragedIndividualResults_sets = gcnew List<List<float>^>();
00840 i = 0;
00841 for each (List<Dictionary<float, float>^> ^ set in listOfProcessedSets) {
00842     listOfAveragedIndividualResults_sets->Add(averageIndividualKeyValuePairs(set));
00843     //there will actually be only two pieces of information in each list anyway, making this
perfect
00844     listOfAverages[i] = returnDivisionFromTwoFirst(listOfAveragedIndividualResults_sets[i]);
00845     i++;
00846 }
00847 return 1;
00848 }
00849
00850 int numberOfValidSets() {
00851     int elems = 0;
00852     for each (List<Dictionary<float, float>^> ^ set in listOfSets) {
00853         if (set != nullptr) { elems++; }
00854     }
00855     return elems;
00856 }
00857
00858 System::Tuple<float, float>^ findHighestKeyValuePair(int i, float key, int option, float
rangeEachWay, bool whichDir, Dictionary<float, float>^ inputDict) {
00859     int index = userSelectionsIndexes[i];
00860     float currKey = indexedKeys[index];
00861     float tempResult;
00862     if (!whichDir) {
00863         tempResult = 99999; tempKey = -1;
00864     }
00865     else if (option == 1) { tempResult = -9999; tempKey = -1; }
00866     else if (option == 2) { tempResult = 0; tempKey = key; }
00867     while (Math::Abs(key - currKey) < rangeEachWay) {
00868         //if we're finding the lowest value, we ignore what option was selected (by design)
00869         if (!whichDir) {
00870             tempResult = inputDict[currKey] < tempResult ? inputDict[currKey] : tempResult;
00871             tempKey = inputDict[currKey] == tempResult ? currKey : tempKey;
00872         }
00873         else if (option == 1) {
00874             tempResult = inputDict[currKey] > tempResult ? inputDict[currKey] : tempResult;
00875             tempKey = inputDict[currKey] == tempResult ? currKey : tempKey;
00876         }
00877     }
00878     else {

```

```

00899         tempResult += inputDict[currKey];
00900     }
00901     if (index == 0) { break; }
00902     index--;
00903
00904     currKey = indexedKeys[index];
00905 }
00906 //escaped the while loop. Now do it again, but the other way. Admittedly there is redundancy
in this code.
00907 index = userSelectionsIndexes[i];
00908 currKey = indexedKeys[index + 1];
00909 while (Math::Abs(key - currKey) < rangeEachWay) {
00910     if (option == 1 && whichDir) {
00911         tempResult = inputDict[currKey] > tempResult ? inputDict[currKey] : tempResult;
00912         tempKey = inputDict[currKey] == tempResult ? currKey : tempKey;
00913     }
00914
00915     else if (!whichDir) {
00916         tempResult = inputDict[currKey] < tempResult ? inputDict[currKey] : tempResult;
00917         tempKey = inputDict[currKey] == tempResult ? currKey : tempKey;
00918     }
00919
00920     else {
00921         tempResult += inputDict[currKey];
00922     }
00923     if (index == DATASIZE - LINESTOSKIP) { break; }
00924     index++;
00925     currKey = indexedKeys[index];
00926 }
00927 //finally, found the highest key & the value corresponding. Return
00928 System::Tuple<float, float>^ retVal = gcnew Tuple<float, float>(tempKey, tempResult);
00929 return retVal;
00930 }
00931
00932 List<float>^ averageIndividualKeyValuePairs(List<Dictionary<float, float>>^ LOD) {
00933     int i;
00934     int howManyWavelengths = LOD[0]->Count;
00935     List<float>^ pointerToResult = gcnew List<float>(howManyWavelengths);
00936     for (int i = 0; i < howManyWavelengths; i++) {
00937         pointerToResult->Add(0);
00938     }
00939     for each(Dictionary<float, float>^ dict in LOD) {
00940         i = 0;
00941         for each (float key in dict->Keys) {
00942             pointerToResult[i] += dict[key];
00943             i++;
00944         }
00945     }
00946     //now we have our list; divide each item by times called
00947     for (int i = 0; i < howManyWavelengths; i++) {
00948         pointerToResult[i] = pointerToResult[i] / LOD->Count;
00949     }
00950
00951     //success
00952     return pointerToResult;
00953 }
00954
00955 //function to return a division of two first values from list
00956 float returnDivisionFromTwoFirst(List<float>^ givenList) {
00957     if (givenList->Count < 2) {
00958         return 0;
00959     }
00960     if (givenList[0] > givenList[1]) {
00961         return givenList[0] / givenList[1];
00962     }
00963     else {
00964         return givenList[1] / givenList[0];
00965     }
00966 }
00967
00968
00969
00970
00971
00972
00973
00974
00975
00976
00977
00978
00979
00980
00981
00982
00983
00984
00985
00986
00987
00988
00989
00990
00991
00992
00993
00994
00995
00996
00997

```

```
00998
00999
01000
01001
01002
01003
01004
01005
01006
01007
01008
01009 };
```

### 5.3 C:/MEng/LIBSProcessing/LIBSProcessing/Window.cpp File Reference

File running the programme.

```
#include "Window.h"
```

#### Functions

- void `main` ()

#### 5.3.1 Detailed Description

File running the programme.

Author

PR

Date

April 2023

### 5.4 C:/MEng/LIBSProcessing/LIBSProcessing/Window.h File Reference

File handling the UI of the programme.

```
#include "Backend.h"
```

#### Classes

- class [LIBSProcessing::Window](#)

*Class handling the display of the user interface (later referred to as GUI or UI). Primarily has functions responsible for recording button presses.*



### 5.4.1 Detailed Description

File handling the UI of the programme.

Author

PR

Date

April 2023

## 5.5 Window.h

[Go to the documentation of this file.](#)

```
00001 /*****
00008 #pragma once
00009 #include "Backend.h"
00010
00011 namespace LIBSProcessing {
00012
00013     using namespace System;
00014     using namespace System::ComponentModel;
00015     using namespace System::Collections;
00016     using namespace System::Windows::Forms;
00017     using namespace System::Data;
00018     using namespace System::Drawing;
00019
00024     public ref class Window : public System::Windows::Forms::Form
00025     {
00026     public:
00030         Window(void)
00031         {
00032             InitializeComponent();
00033             nameOfFile->Text = System::DateTime::Now.ToString("dd_MM_hhmm")+".csv";
00034             //Initializing the backend here.
00035
00036
00037         }
00038     private: System::Windows::Forms::CheckBox^ saveSelectedBox;
00039     private: System::Windows::Forms::RadioButton^ highestCheckbox;
00040     private: System::Windows::Forms::RadioButton^ sumCheckbox;
00041     private: System::Windows::Forms::GroupBox^ groupBox1;
00042     private: System::Windows::Forms::TextBox^ noiseCutoff;
00043     private: System::Windows::Forms::Label^ label8;
00044     private: System::Windows::Forms::Label^ label9;
00045     private: System::Windows::Forms::Label^ cutoffLabel;
00046     private: System::Windows::Forms::MenuStrip^ menuStrip1;
00047     private: System::Windows::Forms::ToolStripMenuItem^ modeToolStripMenuItem;
00048     private: System::Windows::Forms::ToolStripMenuItem^ standardToolStripMenuItem;
00049     private: System::Windows::Forms::ToolStripMenuItem^ calibrationToolStripMenuItem;
00050     private: System::Windows::Forms::Label^ setALabel;
00051     private: System::Windows::Forms::Label^ setBLabel;
00052
00053
00054
00055     private: System::Windows::Forms::Label^ selectFilesLabel_setB;
00056
00057     private: System::Windows::Forms::Button^ fileSelect_setB;
00058
00059
00060     private: System::Windows::Forms::Label^ analyteLabel_setB;
00061
00062
00063     private: System::Windows::Forms::TextBox^ analyteBox_setB;
00064     private: System::ComponentModel::BackgroundWorker^ backgroundWorker1;
00065     private: System::Windows::Forms::Label^ setNumbersLabel;
00066     private: System::Windows::Forms::Label^ howManyLabel;
00067     private: System::Windows::Forms::Button^ howManySubmit;
00068
00069
00070     private: System::Windows::Forms::TextBox^ howManySets;
00071     private: System::Windows::Forms::ComboBox^ setsOfData;
00072     private: System::Windows::Forms::Label^ setsOfData_label;
```

```

00073     private: System::Windows::Forms::Button^ addSetButton;
00074     private: System::Windows::Forms::Label^ Rscore;
00075     private: System::Windows::Forms::TextBox^ rangeLowerInput;
00076     private: System::Windows::Forms::Label^ label1;
00077     private: System::Windows::Forms::GroupBox^ groupBox2;
00078     private: System::Windows::Forms::RadioButton^ lowerRangeYes;
00079
00080     private: System::Windows::Forms::RadioButton^ radioButton2;
00081
00082
00083
00084
00085
00086
00087
00088     public:
00089
00090     protected:
00091         Backend b;
00092         ~Window()
00093         {
00094             if (components)
00095             {
00096                 delete components;
00097             }
00098         }
00099
00100     protected:
00101
00102
00103     private: System::Windows::Forms::TextBox^ waveEdit;
00104
00105     private: System::Windows::Forms::Button^ waveSubmit;
00106
00107     private: System::Windows::Forms::Label^ label2;
00108
00109
00110
00111
00112     private: System::Windows::Forms::TextBox^ rangeInput;
00113     private: System::Windows::Forms::Label^ label3;
00114
00115     private: System::Windows::Forms::ToolTip^ toolTip1;
00116     private: System::Windows::Forms::ComboBox^ allWavelengths;
00117     private: System::Windows::Forms::Label^ label4;
00118     private: System::Windows::Forms::Button^ removeWave;
00119     private: System::Windows::Forms::Button^ saveFolderSelect;
00120     private: System::Windows::Forms::FolderBrowserDialog^ folderBrowser;
00121
00122     private: System::Windows::Forms::Label^ label5;
00123     private: System::Windows::Forms::TextBox^ savePath;
00124     private: System::Windows::Forms::TextBox^ nameOfFile;
00125
00126     private: System::Windows::Forms::Label^ label6;
00127     private: System::Windows::Forms::Button^ saveToFile;
00128     private: System::Windows::Forms::Button^ fileSelect;
00129     private: System::Windows::Forms::Label^ label7;
00130     private: System::Windows::Forms::Button^ preview;
00131     private: System::Windows::Forms::Label^ noOfFile;
00132     private: System::Windows::Forms::OpenFileDialog^ fileOpener;
00133
00134
00135
00136
00137     private: System::ComponentModel::IContainer^ components;
00138
00139     private:
00140
00141
00142
00143
00144
00145 #pragma region Windows Form Designer generated code
00146     void InitializeComponent(void)
00147     {
00148         this->components = (gcnew System::ComponentModel::Container());
00149         System::ComponentModel::ResourceManager^ resources = (gcnew
00150             System::ComponentModel::ResourceManager(Window::typeid));
00151         this->waveEdit = (gcnew System::Windows::Forms::TextBox());
00152         this->waveSubmit = (gcnew System::Windows::Forms::Button());
00153         this->label2 = (gcnew System::Windows::Forms::Label());
00154         this->rangeInput = (gcnew System::Windows::Forms::TextBox());
00155         this->label3 = (gcnew System::Windows::Forms::Label());
00156         this->toolTip1 = (gcnew System::Windows::Forms::ToolTip(this->components));
00157         this->noiseCutoff = (gcnew System::Windows::Forms::TextBox());
00158         this->analyzeBox_setB = (gcnew System::Windows::Forms::TextBox());
00159         this->setNumbersLabel = (gcnew System::Windows::Forms::Label());
00160         this->allWavelengths = (gcnew System::Windows::Forms::ComboBox());
00161         this->label4 = (gcnew System::Windows::Forms::Label());
00162         this->removeWave = (gcnew System::Windows::Forms::Button());
00163         this->saveFolderSelect = (gcnew System::Windows::Forms::Button());
00164         this->folderBrowser = (gcnew System::Windows::Forms::FolderBrowserDialog());
00165         this->label5 = (gcnew System::Windows::Forms::Label());

```

```

00169         this->savePath = (gcnew System::Windows::Forms::TextBox());
00170         this->nameOfFile = (gcnew System::Windows::Forms::TextBox());
00171         this->label6 = (gcnew System::Windows::Forms::Label());
00172         this->saveToFile = (gcnew System::Windows::Forms::Button());
00173         this->fileSelect = (gcnew System::Windows::Forms::Button());
00174         this->label7 = (gcnew System::Windows::Forms::Label());
00175         this->preview = (gcnew System::Windows::Forms::Button());
00176         this->noOfFiles = (gcnew System::Windows::Forms::Label());
00177         this->fileOpener = (gcnew System::Windows::Forms::OpenFileDialog());
00178         this->saveSelectedBox = (gcnew System::Windows::Forms::CheckBox());
00179         this->highestCheckbox = (gcnew System::Windows::Forms::RadioButton());
00180         this->sumCheckbox = (gcnew System::Windows::Forms::RadioButton());
00181         this->groupBox1 = (gcnew System::Windows::Forms::GroupBox());
00182         this->label8 = (gcnew System::Windows::Forms::Label());
00183         this->label9 = (gcnew System::Windows::Forms::Label());
00184         this->cutoffLabel = (gcnew System::Windows::Forms::Label());
00185         this->menuStrip1 = (gcnew System::Windows::Forms::MenuStrip());
00186         this->modeToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());
00187         this->standardToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());
00188         this->calibrationToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());
00189         this->setALabel = (gcnew System::Windows::Forms::Label());
00190         this->setBLabel = (gcnew System::Windows::Forms::Label());
00191         this->selectFilesLabel_setB = (gcnew System::Windows::Forms::Label());
00192         this->fileSelect_setB = (gcnew System::Windows::Forms::Button());
00193         this->analyteLabel_setB = (gcnew System::Windows::Forms::Label());
00194         this->backgroundWorker1 = (gcnew System::ComponentModel::BackgroundWorker());
00195         this->howManyLabel = (gcnew System::Windows::Forms::Label());
00196         this->howManySubmit = (gcnew System::Windows::Forms::Button());
00197         this->howManySets = (gcnew System::Windows::Forms::TextBox());
00198         this->setOfData = (gcnew System::Windows::Forms::ComboBox());
00199         this->setOfData_label = (gcnew System::Windows::Forms::Label());
00200         this->addSetButton = (gcnew System::Windows::Forms::Button());
00201         this->Rscore = (gcnew System::Windows::Forms::Label());
00202         this->rangeLowerInput = (gcnew System::Windows::Forms::TextBox());
00203         this->label1 = (gcnew System::Windows::Forms::Label());
00204         this->groupBox2 = (gcnew System::Windows::Forms::GroupBox());
00205         this->lowerRangeYes = (gcnew System::Windows::Forms::RadioButton());
00206         this->radioButton2 = (gcnew System::Windows::Forms::RadioButton());
00207         this->groupBox1->SuspendLayout();
00208         this->menuStrip1->SuspendLayout();
00209         this->groupBox2->SuspendLayout();
00210         this->SuspendLayout();
00211         //
00212         // waveEdit
00213         //
00214         this->waveEdit->Location = System::Drawing::Point(12, 46);
00215         this->waveEdit->Name = L"waveEdit";
00216         this->waveEdit->Size = System::Drawing::Size(100, 20);
00217         this->waveEdit->TabIndex = 2;
00218         //
00219         // waveSubmit
00220         //
00221         this->waveSubmit->Location = System::Drawing::Point(133, 43);
00222         this->waveSubmit->Name = L"waveSubmit";
00223         this->waveSubmit->Size = System::Drawing::Size(75, 23);
00224         this->waveSubmit->TabIndex = 3;
00225         this->waveSubmit->Text = L"Submit";
00226         this->waveSubmit->UseVisualStyleBackColor = true;
00227         this->waveSubmit->Click += gcnew System::EventHandler(this, &Window::waveSubmit_Click);
00228         //
00229         // label2
00230         //
00231         this->label2->AutoSize = true;
00232         this->label2->Location = System::Drawing::Point(9, 27);
00233         this->label2->Name = L"label2";
00234         this->label2->Size = System::Drawing::Size(89, 13);
00235         this->label2->TabIndex = 4;
00236         this->label2->Text = L"Add wavelengths";
00237         //
00238         // rangeInput
00239         //
00240         this->rangeInput->Location = System::Drawing::Point(12, 88);
00241         this->rangeInput->Name = L"rangeInput";
00242         this->rangeInput->Size = System::Drawing::Size(100, 20);
00243         this->rangeInput->TabIndex = 6;
00244         this->rangeInput->Text = L"0.07";
00245         //
00246         // label3
00247         //
00248         this->label3->AutoSize = true;
00249         this->label3->Location = System::Drawing::Point(9, 72);
00250         this->label3->Name = L"label3";
00251         this->label3->Size = System::Drawing::Size(118, 13);
00252         this->label3->TabIndex = 7;
00253         this->label3->Text = L"Range (default 0.07nm)";
00254         //
00255         // noiseCutoff

```

```

00256         //
00257         this->noiseCutoff->Location = System::Drawing::Point(244, 43);
00258         this->noiseCutoff->Name = L"noiseCutoff";
00259         this->noiseCutoff->Size = System::Drawing::Size(100, 20);
00260         this->noiseCutoff->TabIndex = 29;
00261         this->toolTip1->SetToolTip(this->noiseCutoff, L"Values below this threshold will be
dropped to 0.");
00262         //
00263         // analyteBox_setB
00264         //
00265         this->analyteBox_setB->Enabled = false;
00266         this->analyteBox_setB->Location = System::Drawing::Point(244, 258);
00267         this->analyteBox_setB->Name = L"analyteBox_setB";
00268         this->analyteBox_setB->Size = System::Drawing::Size(100, 20);
00269         this->analyteBox_setB->TabIndex = 42;
00270         this->toolTip1->SetToolTip(this->analyteBox_setB, L"Values below this threshold will be
dropped to 0.");
00271         //
00272         // setNumbersLabel
00273         //
00274         this->setNumbersLabel->AutoSize = true;
00275         this->setNumbersLabel->Enabled = false;
00276         this->setNumbersLabel->Location = System::Drawing::Point(9, 201);
00277         this->setNumbersLabel->Name = L"setNumbersLabel";
00278         this->setNumbersLabel->Size = System::Drawing::Size(161, 26);
00279         this->setNumbersLabel->TabIndex = 44;
00280         this->setNumbersLabel->Text = L"Set 1: 1st/2nd; Set 2: 3rd/4th... \r\n Hover over for more
information.";
00281         this->toolTip1->SetToolTip(this->setNumbersLabel,
resources->GetString(L"setNumbersLabel.ToolTip"));
00282         //
00283         // allWavelengths
00284         //
00285         this->allWavelengths->FormattingEnabled = true;
00286         this->allWavelengths->Location = System::Drawing::Point(12, 175);
00287         this->allWavelengths->MaxDropDownItems = 100;
00288         this->allWavelengths->Name = L"allWavelengths";
00289         this->allWavelengths->Size = System::Drawing::Size(100, 21);
00290         this->allWavelengths->TabIndex = 9;
00291         //
00292         // label4
00293         //
00294         this->label4->AutoSize = true;
00295         this->label4->Location = System::Drawing::Point(9, 159);
00296         this->label4->Name = L"label4";
00297         this->label4->Size = System::Drawing::Size(154, 13);
00298         this->label4->TabIndex = 10;
00299         this->label4->Text = L"Currently selected wavelengths";
00300         //
00301         // removeWave
00302         //
00303         this->removeWave->Location = System::Drawing::Point(133, 175);
00304         this->removeWave->Name = L"removeWave";
00305         this->removeWave->Size = System::Drawing::Size(75, 23);
00306         this->removeWave->TabIndex = 11;
00307         this->removeWave->Text = L"Remove";
00308         this->removeWave->UseVisualStyleBackColor = true;
00309         this->removeWave->Click += gcnew System::EventHandler(this, &Window::removeWave_Click);
00310         //
00311         // saveFolderSelect
00312         //
00313         this->saveFolderSelect->Location = System::Drawing::Point(170, 450);
00314         this->saveFolderSelect->Name = L"saveFolderSelect";
00315         this->saveFolderSelect->Size = System::Drawing::Size(75, 23);
00316         this->saveFolderSelect->TabIndex = 12;
00317         this->saveFolderSelect->Text = L"Browse...";
00318         this->saveFolderSelect->UseVisualStyleBackColor = true;
00319         this->saveFolderSelect->Click += gcnew System::EventHandler(this,
&Window::saveFolderSelect_Click);
00320         //
00321         // label5
00322         //
00323         this->label5->AutoSize = true;
00324         this->label5->Location = System::Drawing::Point(17, 435);
00325         this->label5->Name = L"label5";
00326         this->label5->Size = System::Drawing::Size(72, 13);
00327         this->label5->TabIndex = 13;
00328         this->label5->Text = L"Save location";
00329         //
00330         // savePath
00331         //
00332         this->savePath->Location = System::Drawing::Point(20, 452);
00333         this->savePath->Name = L"savePath";
00334         this->savePath->ReadOnly = true;
00335         this->savePath->Size = System::Drawing::Size(143, 20);
00336         this->savePath->TabIndex = 14;
00337         //

```

```

00338         // nameOfFile
00339         //
00340         this->nameOfFile->Location = System::Drawing::Point(277, 453);
00341         this->nameOfFile->Name = L"nameOfFile";
00342         this->nameOfFile->Size = System::Drawing::Size(143, 20);
00343         this->nameOfFile->TabIndex = 15;
00344         this->nameOfFile->Text = L"TEMP.csv";
00345         //
00346         // label6
00347         //
00348         this->label6->AutoSize = true;
00349         this->label6->Location = System::Drawing::Point(273, 435);
00350         this->label6->Name = L"label6";
00351         this->label6->Size = System::Drawing::Size(104, 13);
00352         this->label6->TabIndex = 16;
00353         this->label6->Text = L"Name of file to save:";
00354         //
00355         // saveToFile
00356         //
00357         this->saveToFile->Location = System::Drawing::Point(426, 452);
00358         this->saveToFile->Name = L"saveToFile";
00359         this->saveToFile->Size = System::Drawing::Size(75, 23);
00360         this->saveToFile->TabIndex = 17;
00361         this->saveToFile->Text = L"Save";
00362         this->saveToFile->UseVisualStyleBackColor = true;
00363         this->saveToFile->Click += gcnew System::EventHandler(this, &Window::saveToFile_Click);
00364         //
00365         // fileSelect
00366         //
00367         this->fileSelect->Location = System::Drawing::Point(425, 93);
00368         this->fileSelect->Name = L"fileSelect";
00369         this->fileSelect->Size = System::Drawing::Size(75, 23);
00370         this->fileSelect->TabIndex = 18;
00371         this->fileSelect->Text = L"Browse...";
00372         this->fileSelect->UseVisualStyleBackColor = true;
00373         this->fileSelect->Click += gcnew System::EventHandler(this, &Window::fileSelect_Click);
00374         //
00375         // label7
00376         //
00377         this->label7->AutoSize = true;
00378         this->label7->Location = System::Drawing::Point(241, 95);
00379         this->label7->Name = L"label7";
00380         this->label7->Size = System::Drawing::Size(113, 13);
00381         this->label7->TabIndex = 19;
00382         this->label7->Text = L"Select files to process:";
00383         //
00384         // preview
00385         //
00386         this->preview->Location = System::Drawing::Point(204, 389);
00387         this->preview->Name = L"preview";
00388         this->preview->Size = System::Drawing::Size(108, 21);
00389         this->preview->TabIndex = 20;
00390         this->preview->Text = L"Process results";
00391         this->preview->UseVisualStyleBackColor = true;
00392         this->preview->Click += gcnew System::EventHandler(this, &Window::preview_Click);
00393         //
00394         // noOfFiles
00395         //
00396         this->noOfFiles->AutoSize = true;
00397         this->noOfFiles->Location = System::Drawing::Point(422, 119);
00398         this->noOfFiles->Name = L"noOfFiles";
00399         this->noOfFiles->Size = System::Drawing::Size(80, 13);
00400         this->noOfFiles->TabIndex = 21;
00401         this->noOfFiles->Text = L"files selected: 0";
00402         //
00403         // fileOpener
00404         //
00405         this->fileOpener->FileName = L"Select your files...";
00406         this->fileOpener->Filter = L"ASC files (*.asc)|*.asc";
00407         this->fileOpener->Multiselect = true;
00408         //
00409         // saveSelectedBox
00410         //
00411         this->saveSelectedBox->AutoSize = true;
00412         this->saveSelectedBox->Checked = true;
00413         this->saveSelectedBox->CheckState = System::Windows::Forms::CheckState::Checked;
00414         this->saveSelectedBox->Location = System::Drawing::Point(12, 237);
00415         this->saveSelectedBox->Name = L"saveSelectedBox";
00416         this->saveSelectedBox->Size = System::Drawing::Size(179, 17);
00417         this->saveSelectedBox->TabIndex = 23;
00418         this->saveSelectedBox->Text = L"Save selected wavelengths only";
00419         this->saveSelectedBox->UseVisualStyleBackColor = true;
00420         //
00421         // highestCheckbox
00422         //
00423         this->highestCheckbox->AutoSize = true;
00424         this->highestCheckbox->Checked = true;

```

```

00425         this->highestCheckbox->Location = System::Drawing::Point(6, 10);
00426         this->highestCheckbox->Name = L"highestCheckbox";
00427         this->highestCheckbox->Size = System::Drawing::Size(102, 17);
00428         this->highestCheckbox->TabIndex = 26;
00429         this->highestCheckbox->TabStop = true;
00430         this->highestCheckbox->Text = L"Highest in range";
00431         this->highestCheckbox->UseVisualStyleBackColor = true;
00432         //
00433         // sumCheckbox
00434         //
00435         this->sumCheckbox->AutoSize = true;
00436         this->sumCheckbox->Location = System::Drawing::Point(6, 26);
00437         this->sumCheckbox->Name = L"sumCheckbox";
00438         this->sumCheckbox->Size = System::Drawing::Size(87, 17);
00439         this->sumCheckbox->TabIndex = 27;
00440         this->sumCheckbox->Text = L"Sum in range";
00441         this->sumCheckbox->UseVisualStyleBackColor = true;
00442         //
00443         // groupBox1
00444         //
00445         this->groupBox1->Controls->Add(this->highestCheckbox);
00446         this->groupBox1->Controls->Add(this->sumCheckbox);
00447         this->groupBox1->Location = System::Drawing::Point(132, 67);
00448         this->groupBox1->Name = L"groupBox1";
00449         this->groupBox1->Size = System::Drawing::Size(103, 43);
00450         this->groupBox1->TabIndex = 28;
00451         this->groupBox1->TabStop = false;
00452         //
00453         // label8
00454         //
00455         this->label8->AutoSize = true;
00456         this->label8->Location = System::Drawing::Point(241, 24);
00457         this->label8->Name = L"label8";
00458         this->label8->Size = System::Drawing::Size(192, 13);
00459         this->label8->TabIndex = 30;
00460         this->label8->Text = L"Noise cutoff (select before loading files)";
00461         //
00462         // label9
00463         //
00464         this->label9->AutoSize = true;
00465         this->label9->Location = System::Drawing::Point(364, 46);
00466         this->label9->Name = L"label9";
00467         this->label9->Size = System::Drawing::Size(126, 13);
00468         this->label9->TabIndex = 31;
00469         this->label9->Text = L"Leave blank for no cutoff";
00470         //
00471         // cutoffLabel
00472         //
00473         this->cutoffLabel->AutoSize = true;
00474         this->cutoffLabel->Location = System::Drawing::Point(422, 132);
00475         this->cutoffLabel->Name = L"cutoffLabel";
00476         this->cutoffLabel->Size = System::Drawing::Size(64, 13);
00477         this->cutoffLabel->TabIndex = 32;
00478         this->cutoffLabel->Text = L"at no cutoff.";
00479         //
00480         // menuStrip1
00481         //
00482         this->menuStrip1->BackColor = System::Drawing::SystemColors::ButtonShadow;
00483         >(1) { this->menuStrip1->Items->AddRange(gcnew cli::array< System::Windows::Forms::ToolStripItem^
00484             this->menuStrip1->Location = System::Drawing::Point(0, 0);
00485             this->menuStrip1->Name = L"menuStrip1";
00486             this->menuStrip1->Size = System::Drawing::Size(519, 24);
00487             this->menuStrip1->TabIndex = 33;
00488             this->menuStrip1->Text = L"menuStrip1";
00489             //
00490             // modeToolStripMenuItem
00491             //
00492             this->modeToolStripMenuItem->BackColor = System::Drawing::SystemColors::Control;
00493             this->modeToolStripMenuItem->BackgroundImageLayout =
System::Windows::Forms::ImageLayout::Center;
00494             this->modeToolStripMenuItem->DropDownItems->AddRange(gcnew cli::array<
System::Windows::Forms::ToolStripItem^ >(2) {
00495                 this->standardToolStripMenuItem,
00496                 this->calibrationToolStripMenuItem
00497             });
00498             this->modeToolStripMenuItem->Name = L"modeToolStripMenuItem";
00499             this->modeToolStripMenuItem->Size = System::Drawing::Size(50, 20);
00500             this->modeToolStripMenuItem->Text = L"Mode";
00501             //
00502             // standardToolStripMenuItem
00503             //
00504             this->standardToolStripMenuItem->Checked = true;
00505             this->standardToolStripMenuItem->CheckState = System::Windows::Forms::CheckState::Checked;
00506             this->standardToolStripMenuItem->Name = L"standardToolStripMenuItem";
00507             this->standardToolStripMenuItem->Size = System::Drawing::Size(132, 22);
00508             this->standardToolStripMenuItem->Text = L"Standard";

```

```

00509         this->standardToolStripMenuItem->Click += gcnew System::EventHandler(this,
&Window::standardToolStripMenuItem_Click);
00510         //
00511         // calibrationToolStripMenuItem
00512         //
00513         this->calibrationToolStripMenuItem->Name = L"calibrationToolStripMenuItem";
00514         this->calibrationToolStripMenuItem->Size = System::Drawing::Size(132, 22);
00515         this->calibrationToolStripMenuItem->Text = L"Calibration";
00516         this->calibrationToolStripMenuItem->Click += gcnew System::EventHandler(this,
&Window::calibrationToolStripMenuItem_Click);
00517         //
00518         // setALabel
00519         //
00520         this->setALabel->AutoSize = true;
00521         this->setALabel->Location = System::Drawing::Point(241, 77);
00522         this->setALabel->Name = L"setALabel";
00523         this->setALabel->Size = System::Drawing::Size(110, 13);
00524         this->setALabel->TabIndex = 34;
00525         this->setALabel->Text = L"Single-set processing:";
00526         //
00527         // setBLabel
00528         //
00529         this->setBLabel->AutoSize = true;
00530         this->setBLabel->Enabled = false;
00531         this->setBLabel->Location = System::Drawing::Point(241, 141);
00532         this->setBLabel->Name = L"setBLabel";
00533         this->setBLabel->Size = System::Drawing::Size(103, 13);
00534         this->setBLabel->TabIndex = 39;
00535         this->setBLabel->Text = L"Multi-set processing:";
00536         //
00537         // selectFilesLabel_setB
00538         //
00539         this->selectFilesLabel_setB->AutoSize = true;
00540         this->selectFilesLabel_setB->Enabled = false;
00541         this->selectFilesLabel_setB->Location = System::Drawing::Point(241, 221);
00542         this->selectFilesLabel_setB->Name = L"selectFilesLabel_setB";
00543         this->selectFilesLabel_setB->Size = System::Drawing::Size(113, 13);
00544         this->selectFilesLabel_setB->TabIndex = 36;
00545         this->selectFilesLabel_setB->Text = L"Select files to process:";
00546         //
00547         // fileSelect_setB
00548         //
00549         this->fileSelect_setB->Enabled = false;
00550         this->fileSelect_setB->Location = System::Drawing::Point(425, 219);
00551         this->fileSelect_setB->Name = L"fileSelect_setB";
00552         this->fileSelect_setB->Size = System::Drawing::Size(75, 23);
00553         this->fileSelect_setB->TabIndex = 35;
00554         this->fileSelect_setB->Text = L"Browse...";
00555         this->fileSelect_setB->UseVisualStyleBackColor = true;
00556         this->fileSelect_setB->Click += gcnew System::EventHandler(this,
&Window::fileSelect_setB_Click);
00557         //
00558         // analyteLabel_setB
00559         //
00560         this->analyteLabel_setB->AutoSize = true;
00561         this->analyteLabel_setB->Enabled = false;
00562         this->analyteLabel_setB->Location = System::Drawing::Point(241, 244);
00563         this->analyteLabel_setB->Name = L"analyteLabel_setB";
00564         this->analyteLabel_setB->Size = System::Drawing::Size(174, 13);
00565         this->analyteLabel_setB->TabIndex = 43;
00566         this->analyteLabel_setB->Text = L"Analyte concentration (ppm) for given set:";
00567         //
00568         // howManyLabel
00569         //
00570         this->howManyLabel->AutoSize = true;
00571         this->howManyLabel->Enabled = false;
00572         this->howManyLabel->Location = System::Drawing::Point(241, 157);
00573         this->howManyLabel->Name = L"howManyLabel";
00574         this->howManyLabel->Size = System::Drawing::Size(131, 13);
00575         this->howManyLabel->TabIndex = 47;
00576         this->howManyLabel->Text = L"How may sets to process\?";
00577         //
00578         // howManySubmit
00579         //
00580         this->howManySubmit->Enabled = false;
00581         this->howManySubmit->Location = System::Drawing::Point(425, 176);
00582         this->howManySubmit->Name = L"howManySubmit";
00583         this->howManySubmit->Size = System::Drawing::Size(75, 23);
00584         this->howManySubmit->TabIndex = 46;
00585         this->howManySubmit->Text = L"Submit";
00586         this->howManySubmit->UseVisualStyleBackColor = true;
00587         this->howManySubmit->Click += gcnew System::EventHandler(this,
&Window::howManySubmit_Click);
00588         //
00589         // howManySets
00590         //
00591         this->howManySets->Enabled = false;

```



```

00592         this->howManySets->Location = System::Drawing::Point(244, 176);
00593         this->howManySets->Name = L"howManySets";
00594         this->howManySets->Size = System::Drawing::Size(100, 20);
00595         this->howManySets->TabIndex = 45;
00596         //
00597         // setsOfData
00598         //
00599         this->setsOfData->Enabled = false;
00600         this->setsOfData->FormattingEnabled = true;
00601         this->setsOfData->Location = System::Drawing::Point(244, 311);
00602         this->setsOfData->MaxDropDownItems = 20;
00603         this->setsOfData->Name = L"setsOfData";
00604         this->setsOfData->Size = System::Drawing::Size(256, 21);
00605         this->setsOfData->TabIndex = 48;
00606         //
00607         // setsOfData_label
00608         //
00609         this->setsOfData_label->AutoSize = true;
00610         this->setsOfData_label->Enabled = false;
00611         this->setsOfData_label->Location = System::Drawing::Point(241, 295);
00612         this->setsOfData_label->Name = L"setsOfData_label";
00613         this->setsOfData_label->Size = System::Drawing::Size(118, 13);
00614         this->setsOfData_label->TabIndex = 49;
00615         this->setsOfData_label->Text = L"Sets of data information";
00616         //
00617         // addSetButton
00618         //
00619         this->addSetButton->Enabled = false;
00620         this->addSetButton->Location = System::Drawing::Point(425, 258);
00621         this->addSetButton->Name = L"addSetButton";
00622         this->addSetButton->Size = System::Drawing::Size(75, 23);
00623         this->addSetButton->TabIndex = 50;
00624         this->addSetButton->Text = L"Add set";
00625         this->addSetButton->UseVisualStyleBackColor = true;
00626         this->addSetButton->Click += gnew System::EventHandler(this,
&Window::addSetButton_Click);
00627         //
00628         // Rscore
00629         //
00630         this->Rscore->AutoSize = true;
00631         this->Rscore->Location = System::Drawing::Point(201, 413);
00632         this->Rscore->Name = L"Rscore";
00633         this->Rscore->Size = System::Drawing::Size(0, 13);
00634         this->Rscore->TabIndex = 51;
00635         //
00636         // rangeLowerInput
00637         //
00638         this->rangeLowerInput->Location = System::Drawing::Point(12, 129);
00639         this->rangeLowerInput->Name = L"rangeLowerInput";
00640         this->rangeLowerInput->Size = System::Drawing::Size(100, 20);
00641         this->rangeLowerInput->TabIndex = 52;
00642         this->rangeLowerInput->Text = L"0.2";
00643         //
00644         // label1
00645         //
00646         this->label1->AutoSize = true;
00647         this->label1->Location = System::Drawing::Point(9, 113);
00648         this->label1->Name = L"label1";
00649         this->label1->Size = System::Drawing::Size(147, 13);
00650         this->label1->TabIndex = 53;
00651         this->label1->Text = L"Range for finding lowest point";
00652         //
00653         // groupBox2
00654         //
00655         this->groupBox2->Controls->Add(this->lowerRangeYes);
00656         this->groupBox2->Controls->Add(this->radioButton2);
00657         this->groupBox2->Location = System::Drawing::Point(176, 113);
00658         this->groupBox2->Name = L"groupBox2";
00659         this->groupBox2->Size = System::Drawing::Size(59, 49);
00660         this->groupBox2->TabIndex = 29;
00661         this->groupBox2->TabStop = false;
00662         //
00663         // lowerRangeYes
00664         //
00665         this->lowerRangeYes->AutoSize = true;
00666         this->lowerRangeYes->Checked = true;
00667         this->lowerRangeYes->Location = System::Drawing::Point(6, 10);
00668         this->lowerRangeYes->Name = L"lowerRangeYes";
00669         this->lowerRangeYes->Size = System::Drawing::Size(43, 17);
00670         this->lowerRangeYes->TabIndex = 26;
00671         this->lowerRangeYes->TabStop = true;
00672         this->lowerRangeYes->Text = L"Yes";
00673         this->lowerRangeYes->UseVisualStyleBackColor = true;
00674         //
00675         // radioButton2
00676         //
00677         this->radioButton2->AutoSize = true;

```



```

00678         this->radioButton2->Location = System::Drawing::Point(6, 26);
00679         this->radioButton2->Name = L"radioButton2";
00680         this->radioButton2->Size = System::Drawing::Size(46, 17);
00681         this->radioButton2->TabIndex = 27;
00682         this->radioButton2->Text = L"Skip";
00683         this->radioButton2->UseVisualStyleBackColor = true;
00684         //
00685         // Window
00686         //
00687         this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);
00688         this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;
00689         this->ClientSize = System::Drawing::Size(519, 495);
00690         this->Controls->Add(this->groupBox2);
00691         this->Controls->Add(this->label1);
00692         this->Controls->Add(this->rangeLowerInput);
00693         this->Controls->Add(this->Rscore);
00694         this->Controls->Add(this->addSetButton);
00695         this->Controls->Add(this->setsOfData_label);
00696         this->Controls->Add(this->setsOfData);
00697         this->Controls->Add(this->howManyLabel);
00698         this->Controls->Add(this->howManySubmit);
00699         this->Controls->Add(this->howManySets);
00700         this->Controls->Add(this->setNumbersLabel);
00701         this->Controls->Add(this->analyteLabel_setB);
00702         this->Controls->Add(this->analyteBox_setB);
00703         this->Controls->Add(this->setBLabel);
00704         this->Controls->Add(this->selectFilesLabel_setB);
00705         this->Controls->Add(this->fileSelect_setB);
00706         this->Controls->Add(this->setALabel);
00707         this->Controls->Add(this->cutoffLabel);
00708         this->Controls->Add(this->label9);
00709         this->Controls->Add(this->label8);
00710         this->Controls->Add(this->noiseCutoff);
00711         this->Controls->Add(this->groupBox1);
00712         this->Controls->Add(this->saveSelectedBox);
00713         this->Controls->Add(this->noOfFiles);
00714         this->Controls->Add(this->preview);
00715         this->Controls->Add(this->label7);
00716         this->Controls->Add(this->fileSelect);
00717         this->Controls->Add(this->saveToFile);
00718         this->Controls->Add(this->label6);
00719         this->Controls->Add(this->nameOfFile);
00720         this->Controls->Add(this->savePath);
00721         this->Controls->Add(this->label5);
00722         this->Controls->Add(this->saveFolderSelect);
00723         this->Controls->Add(this->removeWave);
00724         this->Controls->Add(this->label4);
00725         this->Controls->Add(this->allWavelengths);
00726         this->Controls->Add(this->label3);
00727         this->Controls->Add(this->rangeInput);
00728         this->Controls->Add(this->label2);
00729         this->Controls->Add(this->waveSubmit);
00730         this->Controls->Add(this->waveEdit);
00731         this->Controls->Add(this->menuStrip1);
00732         this->MainMenuStrip = this->menuStrip1;
00733         this->Name = L"Window";
00734         this->Text = L"Zeus";
00735         this->Load += gcnew System::EventHandler(this, &Window::Window_Load);
00736         this->groupBox1->ResumeLayout(false);
00737         this->groupBox1->PerformLayout();
00738         this->menuStrip1->ResumeLayout(false);
00739         this->menuStrip1->PerformLayout();
00740         this->groupBox2->ResumeLayout(false);
00741         this->groupBox2->PerformLayout();
00742         this->ResumeLayout(false);
00743         this->PerformLayout();
00744     }
00745 }
00746 #pragma endregion
00747
00748 //CODE HANDLING THE UI & calls to the "backend".
00749
00750 //GUI handler - submit an ELEMENT'S wavelengths to the list
00751 private: System::Void elemSubmit_Click(System::Object^ sender, System::EventArgs^ e) {
00752     //Currently empty
00753 }
00754 private: System::Void waveSubmit_Click(System::Object^ sender, System::EventArgs^ e) {
00755     float attemptConversion;
00756     //try converting - if failed, show a message to the user
00757     try {
00758         attemptConversion = Convert::ToSingle(waveEdit->Text);
00759     }
00760     catch (...) {
00761         MessageBox::Show("Error - please input a float value");
00762     }
00763 }

```

```

00772         return;
00773     }
00774
00775     if (!b.addWavelength(attemptConversion)) {
00776         MessageBox::Show("Error - please input a value between 200.93 and 1031.86");
00777     }
00778     //for some reason, it order to update the list in the GUI it needs to be fully reset
00779     allWavelengths->DataSource = nullptr;
00780     allWavelengths->DataSource = b.selectedWavelengths;
00781     waveEdit->Text = "";
00782
00783 }
00791 private: System::Void saveFolderSelect_Click(System::Object^ sender, System::EventArgs^ e) {
00792     if (folderBrowser->ShowDialog() == System::Windows::Forms::DialogResult::OK)
00793     {
00794         String^ folderName = folderBrowser->SelectedPath;
00795         savePath->Text = folderName;
00796         b.directory = folderName;        //set the directory in the backend
00797     }
00798 }
00806 private: System::Void removeWave_Click(System::Object^ sender, System::EventArgs^ e) {
00807     float waveToRemove = Convert::ToSingle(allWavelengths->Text);
00808     b.removeWavelength(waveToRemove);
00809     allWavelengths->DataSource = nullptr;
00810     allWavelengths->DataSource = b.selectedWavelengths;
00811 }
00812
00813 //GUI handler - preview all options - actually processes the data for now as well
00821 private: System::Void preview_Click(System::Object^ sender, System::EventArgs^ e) {
00822     if (b.getAveragedSpectra()) {}
00823     else { MessageBox::Show("Error - no files loaded"); return; }
00824     int option;
00825     //option 1 - highest point in range, skip range for finding lowest point
00826     //option 2 - sum in range, skip range for finding lowest point
00827     //option 3 - highest point in range, yes for range for finding lowest point
00828     //option 4 - sum in range, yes for range for finding lowest point
00829     if (highestCheckbox->Checked) { option = 1; }
00830
00831     float range;
00832     float lowerRange;
00833     try {
00834         range = Convert::ToSingle(rangeInput->Text);
00835         lowerRange = Convert::ToSingle(rangeLowerInput->Text);
00836         if (range < 0) { range = -1 * range; };
00837     }
00838     catch (...) {
00839         MessageBox::Show("Error - range must be a float");
00840         return;
00841     }
00842     //Perform operations to retrieve division information
00843     if (calibrationToolStripMenuItem->Checked) {
00844         b.getRequestSpectraCalibrationMode(option, range, lowerRangeYes->Checked, lowerRange);
00845         Rscore->Text = "R^2 score: " + Convert::ToString(b.getRSquared());
00846     }
00847     //standard mode operation
00848     else {
00849         b.getRequestSpectraStandardMode(option, range, lowerRangeYes->Checked, lowerRange);
00850     }
00851 }
00852
00860 private: System::Void saveToFile_Click(System::Object^ sender, System::EventArgs^ e) {
00861     if (calibrationToolStripMenuItem->Checked) {
00862         int success = b.saveToFileCalibration(nameOfFile->Text, lowerRangeYes->Checked);
00863         if (success == 0) {
00864             MessageBox::Show("Error - file was unable to be saved with name " + b.nameOfFile);
00865         }
00866         else if (success == 1) {
00867             MessageBox::Show("File saved at " + b.directory + b.nameOfFile);
00868         }
00869         else {
00870             MessageBox::Show("Error - one of the data structures has not been initialized. This
most often happens if 'Process' has not been clicked. ");
00871         }
00872     }
00873     //standard mode saving.
00874     else {
00875         int success = b.saveToFile(nameOfFile->Text, saveSelectedBox->Checked,
lowerRangeYes->Checked);
00876         if (success == 0) {
00877             MessageBox::Show("Error - file was unable to be saved with name " + b.nameOfFile);
00878         }
00879         else if (success == 1) {
00880             MessageBox::Show("File saved at " + b.directory + b.nameOfFile);
00881         }
00882         else {
00883         }
00884     }

```

```

00885         MessageBox::Show("Error - one of the data structures has not been initialized. This
most often happens if 'Process' has not been clicked. ");
00886     }
00887 }
00888
00889
00890 }
00891
00892 private: System::Void fileSelect_Click(System::Object^ sender, System::EventArgs^ e) {
00893     handleSelection(1);
00894 }
00895 private: System::Void fileSelect_setB_Click(System::Object^ sender, System::EventArgs^ e) {
00896     handleSelection(2);
00897     selectFilesLabel_setB->Text = "Files selected.";
00898 }
00899
00900 private: System::Void Window_Load(System::Object^ sender, System::EventArgs^ e) {
00901 }
00902
00903 private: System::Void standardToolStripMenuItem_Click(System::Object^ sender, System::EventArgs^
e) {
00904     setCalibrationGroup(true);
00905     setSetAddedGroup(false);
00906 }
00907 private: System::Void calibrationToolStripMenuItem_Click(System::Object^ sender,
System::EventArgs^ e) {
00908     setCalibrationGroup(false);
00909 }
00910 private: System::Void howManySubmit_Click(System::Object^ sender, System::EventArgs^ e) {
00911     int attemptConversion;
00912     try {
00913         attemptConversion = Convert::ToInt32(howManySets->Text);
00914     }
00915     catch (...) {
00916         MessageBox::Show("Error - please input an integer value");
00917         return;
00918     }
00919     b.initializeSets(attemptConversion);
00920     setsOfData->DataSource = nullptr;
00921     setsOfData->DataSource = b.metadata;
00922     setSetAddedGroup(true);
00923     howManySets->Text = "";
00924 }
00925 private: System::Void addSetButton_Click(System::Object^ sender, System::EventArgs^ e) {
00926     if (b.filesToExtract_B == nullptr || b.filesToExtract_B->Count == 0) {
00927         MessageBox::Show("Error - no files selected");
00928         return;
00929     }
00930     if (b.metadata == nullptr || b.metadata->Count == 0) {
00931         MessageBox::Show("Error - no empty sets initialized");
00932         return;
00933     }
00934     float concentration;
00935     try {
00936         concentration = Convert::ToSingle(analyteBox_setB->Text);
00937     }
00938     catch (...) {
00939         MessageBox::Show("Error - please input a float value for the concentration");
00940         return;
00941     }
00942     int i = setsOfData->SelectedIndex;
00943     float cutoff;
00944     try {
00945         cutoff = Convert::ToSingle(noiseCutoff->Text);
00946     }
00947     catch (...) {
00948         cutoff = -199;
00949     }
00950     b.addSetToSets(concentration, i, cutoff);
00951     setsOfData->DataSource = nullptr;
00952     setsOfData->DataSource = b.metadata;
00953     selectFilesLabel_setB->Text = "Select files to process.";
00954     analyteBox_setB->Text = "";
00955     setsOfData->SelectedIndex = i >= b.metadata->Count-1 ? i : i + 1;
00956     return;
00957 }
00958
00959 private: void handleSelection(int selectionWindow) {
00960     float cutoff;
00961     //if loaded files successfully,
00962     if (fileOpener->ShowDialog() == System::Windows::Forms::DialogResult::OK) {

```

```

01026         //try converting noiseCutoff to double,
01027         try {
01028             if (noiseCutoff->Text == "") {
01029                 cutoff = -199;
01030             }
01031             else {
01032                 cutoff = Convert::ToSingle(noiseCutoff->Text);
01033             }
01034         }
01035         //if unsuccessful, inform user and continue operation.
01036         catch (...) {
01037             MessageBox::Show("Cutoff value is not a float, setting to no cutoff.");
01038             cutoff = -199;
01039         }
01040         //and process the files into the memory.
01041         if (b.loadFiles(fileOpener->FileNames, cutoff, selectionWindow)) {
01042             if (selectionWindow == 1) {
01043                 noOfFiles->Text = "files selected: " + fileOpener->FileNames->Length;
01044                 if (cutoff == -199) {
01045                     cutoffLabel->Text = "at no cutoff. ";
01046                 }
01047                 else {
01048                     cutoffLabel->Text = "at cutoff: " + cutoff;
01049                 }
01050             }
01051             else if (selectionWindow == 2) {
01052                 //noOfFiles_setB->Text = "files selected: " + fileOpener->FileNames->Length;
01053                 if (cutoff == -199) {
01054                     //cutoffLabel_setB->Text = "at no cutoff. ";
01055                 }
01056                 else {
01057                     //cutoffLabel_setB->Text = "at cutoff: " + cutoff;
01058                 }
01059             }
01060         }
01061     }
01062 }
01063 //very basic handling for now.
01064 else {
01065     noOfFiles->Text = "Error: one of the files is not an .asc file.";
01066 }
01067 }
01068 }
01074 private: void setCalibrationGroup(bool value) {
01075     standardToolStripMenuItem->Checked = value;
01076     calibrationToolStripMenuItem->Checked = !value;
01077     //single set processing bit
01078     setAlabel->Enabled = value;
01079     label7 -> Enabled = value;
01080     fileSelect->Enabled = value;
01081     noOfFiles->Enabled = value;
01082     cutoffLabel->Enabled = value;
01083     //dual set processing bit
01084     setBlabel->Enabled = !value;
01085     howManyLabel->Enabled = !value;
01086     howManySets->Enabled = !value;
01087     howManySubmit->Enabled = !value;
01088     selectFilesLabel_setB->Enabled = !value;
01089     analyteLabel_setB->Enabled = !value;
01090     analyteBox_setB->Enabled = !value;
01091     //fileSelect_setB->Enabled = !value;
01092     selectFilesLabel_setB->Enabled = !value;
01093     //addSetButton->Enabled = !value;
01094     //information
01095     setsOfData->Enabled = !value;
01096     setsOfData_label->Enabled = !value;
01097     //left hand side
01098     label3->Enabled = value;
01099     //rangeInput->Enabled = value;
01100     setNumbersLabel->Enabled = !value;
01101     saveSelectedBox->Enabled = value;
01102 }
01108 private: void setSetAddedGroup(bool value) {
01109     fileSelect_setB->Enabled = value;
01110     addSetButton->Enabled = value;
01111 }
01112 }
01113 };
01114 }

```

# Index

- addSetButton\_Click
  - LIBSProcessing::Window, [19](#)
- addSetToSets
  - Backend, [8](#)
- addWavelength
  - Backend, [9](#)
- averageIndividualKeyValuePairs
  - Backend, [9](#)
- Backend, [7](#)
  - addSetToSets, [8](#)
  - addWavelength, [9](#)
  - averageIndividualKeyValuePairs, [9](#)
  - findHighestKeyValuePair, [9](#)
  - findKeys, [10](#)
  - findRequestedValues, [10](#)
  - findRequestedValuesCalibration, [11](#)
  - getAveragedSpectra, [11](#)
  - getRequestedSpectraCalibrationMode, [11](#)
  - getRequestedSpectraStandardMode, [12](#)
  - getRSquared, [12](#)
  - initializeMemoryFiles, [12](#)
  - initializeSets, [13](#)
  - loadFiles, [13](#)
  - numberOfValidSets, [14](#)
  - processFileIntoDictionary, [14](#)
  - removeWavelength, [14](#)
  - returnDivisionFromTwoFirst, [15](#)
  - saveToFile, [15](#)
  - saveToFileCalibration, [16](#)
  - sumDictionaries, [16](#)
- C:/MEng/LIBSProcessing/LIBSProcessing/Backend.h,  
[25](#), [26](#)
- C:/MEng/LIBSProcessing/LIBSProcessing/Window.cpp,  
[36](#)
- C:/MEng/LIBSProcessing/LIBSProcessing/Window.h,  
[36](#), [37](#)
- calibrationToolStripMenuItem\_Click
  - LIBSProcessing::Window, [19](#)
- fileSelect\_Click
  - LIBSProcessing::Window, [19](#)
- fileSelect\_setB\_Click
  - LIBSProcessing::Window, [20](#)
- findHighestKeyValuePair
  - Backend, [9](#)
- findKeys
  - Backend, [10](#)
- findRequestedValues
  - Backend, [10](#)
- findRequestedValuesCalibration
  - Backend, [11](#)
- getAveragedSpectra
  - Backend, [11](#)
- getRequestedSpectraCalibrationMode
  - Backend, [11](#)
- getRequestedSpectraStandardMode
  - Backend, [12](#)
- getRSquared
  - Backend, [12](#)
- handleSelection
  - LIBSProcessing::Window, [20](#)
- howManySubmit\_Click
  - LIBSProcessing::Window, [21](#)
- InitializeComponent
  - LIBSProcessing::Window, [21](#)
- initializeMemoryFiles
  - Backend, [12](#)
- initializeSets
  - Backend, [13](#)
- LIBSProcessing::Window, [16](#)
  - addSetButton\_Click, [19](#)
  - calibrationToolStripMenuItem\_Click, [19](#)
  - fileSelect\_Click, [19](#)
  - fileSelect\_setB\_Click, [20](#)
  - handleSelection, [20](#)
  - howManySubmit\_Click, [21](#)
  - InitializeComponent, [21](#)
  - preview\_Click, [21](#)
  - removeWave\_Click, [21](#)
  - saveFolderSelect\_Click, [22](#)
  - saveToFile\_Click, [22](#)
  - setCalibrationGroup, [23](#)
  - setSetAddedGroup, [23](#)
  - standardToolStripMenuItem\_Click, [23](#)
  - waveSubmit\_Click, [24](#)
  - Window, [18](#)
  - Window\_Load, [24](#)
- loadFiles
  - Backend, [13](#)
- numberOfValidSets
  - Backend, [14](#)
- preview\_Click
  - LIBSProcessing::Window, [21](#)

processFileIntoDictionary  
    Backend, [14](#)

removeWave\_Click  
    LIBSProcessing::Window, [21](#)

removeWavelength  
    Backend, [14](#)

returnDivisionFromTwoFirst  
    Backend, [15](#)

saveFolderSelect\_Click  
    LIBSProcessing::Window, [22](#)

saveToFile  
    Backend, [15](#)

saveToFile\_Click  
    LIBSProcessing::Window, [22](#)

saveToFileCalibration  
    Backend, [16](#)

setCalibrationGroup  
    LIBSProcessing::Window, [23](#)

setSetAddedGroup  
    LIBSProcessing::Window, [23](#)

standardToolStripMenuItem\_Click  
    LIBSProcessing::Window, [23](#)

sumDictionaries  
    Backend, [16](#)

waveSubmit\_Click  
    LIBSProcessing::Window, [24](#)

Window  
    LIBSProcessing::Window, [18](#)

Window\_Load  
    LIBSProcessing::Window, [24](#)