Optimization of quality assured dataflow from biosensors: time series analysis of plankton respiration by oxygen optodes

Manfred Lindmark

2015
Optimization of quality assured dataflow from biosensors

Time series analysis of plankton respiration by oxygen optodes

Manfred Lindmark
Abstract

Data analysis can be a time consuming part of an experimental method, especially when the method is used frequently and large amounts of data are produced each time. In this study, an application software was developed to improve work flow and data management for respiration rate measurements using an optical oxygen sensor. The application was used to analyze data files from the oxygen sensor without the need to manually enter and analyze the data in a spreadsheet application. The software was written in the Python programming language and utilized available scientific computing packages as well as a graphical user interface framework to provide user friendly access to all functions. Any number of files with experimental data were imported into the program and a linear regression analysis was done for each file and viewed to verify the quality of the data. Tables and summarizing graphs were used to display the key information and statistical results. The final results were exported for use in other applications. Data processing that used to take an hour to complete was done with the new application in five to ten minutes and the risk of introducing human errors in the data was simultaneously reduced. User tests indicated that learning the basics of the program was easy. This study shows the usefulness of a bioinformatics approach and the tools provided by Python and its related software to solve problems that arise with managing large volumes of numerical data.
## Contents

Introduction                                                                                     ......................................  3  
1.1 Overall objective                                                                                        ..................................  3  
1.2 The biological quantity measured                                                                  ..................................  3  
1.3 Current solution to data flow                                                                      ..................................  4  
1.4 Problem description                                                                                    ..................................  4  
1.5 Aim of the project                                                                                   ..................................  4  
1.6 The new bioinformatics tool                                                                           ..................................  5  
2 Experimental                                                                                    .....................................................  5  
  2.1 Theory                                                                                             .....................................................  5  
    2.1.1 Data analysis                                                                                    .....................................................  5  
    2.1.2 Regression analysis                                                                             .....................................................  5  
  2.2 Solution strategies                                                                                .....................................................  6  
    2.2.1 Importing data                                                                                  .....................................................  6  
    2.2.2 Graphical user interface                                                                        .....................................................  6  
    2.2.3 Graphs                                                                                          .....................................................  7  
    2.2.4 Editing data ranges for regression analysis                                                   .....................................................  7  
    2.2.5 Saving analysis                                                                                .....................................................  7  
    2.2.6 Exporting summary                                                                              .....................................................  8  
    2.2.7 Program access                                                                                  .....................................................  8  
  2.3 Implementation                                                                                    .....................................................  8  
    2.3.1 File import                                                                                    .....................................................  8  
    2.3.2 Regression analysis                                                                             .....................................................  8  
    2.3.3 User interface                                                                                  .....................................................  9  
    2.3.4 Distribution                                                                                    ..................................................... 12  
3 Results                                                                                             ..................................................... 12  
  3.1.1 Verification of statistical results                                                              ..................................................... 12  
  3.1.2 User tests                                                                                      ..................................................... 13  
  3.1.3 Performance                                                                                    ..................................................... 14  
4 Discussion                                                                                           ..................................................... 14  
5 Conclusions                                                                                          ..................................................... 15  
6 Proposed future work                                                                                ..................................................... 15  
7 References                                                                                           ..................................................... 17  

2
Introduction

1.1 Overall objective
Sensor applications are becoming increasingly utilized in biological research and environmental analysis. This increase puts requirements on data organization, as well as analysis and synthesis of results in a quality assured manner to reduce working time and project costs. In other words, development of bioinformatics tools to rapidly transform large data flows to utilizable knowledge with assured data quality is needed. In this study, the data flow from oxygen sensors used to measure biological respiration in natural aquatic environments was optimized by creating a new software for bio-information.

1.2 The biological quantity measured
The concentration of dissolved oxygen is an important component of aquatic ecosystems and is a powerful indicator of the state of an ecosystem. In the Baltic Sea, oxygen depletion, a well-known phenomenon, is thought to be the result of human-caused pollution and eutrophication, but may also be influenced by climate drivers. Hypoxia can severely decrease biodiversity in affected waters by causing a loss of desirable fish species. There is however still much about the ecology of the Baltic Sea that is not well understood, and research is ongoing in this area to better understand the causes and effects of oxygen depletion.

The optical oxygen sensor is commonly used in applications such as environmental monitoring or biological research for which continuous water oxygen concentration monitoring is needed. Optical oxygen sensors have grown in popularity due to their low cost, long term stability, and low power requirements. These properties make them suitable for use in buoy monitoring systems. One such sensor is the oxygen optode produced by Aanderaa Data Instruments. This sensor is a digital optode (optical sensor) based on a dynamic luminescence quenching technique for which the fluorescence of a sensing foil depends on the oxygen concentration. The sensor has a built-in calibrator and outputs the exact water oxygen concentration, air saturation, and temperature as well as additional data. These data are sent to a computer or other data logger via a serial connection. With software from Aanderaa, the measurements can be continuously logged to text files on the connected computer. The optode has a sample rate of up to one sample per second and can sample over very long periods of time. Thus, the amount of data that an optode produces can be considerable.

An oxygen optodes is also suitable for laboratory scale measurements, as the sensor does not affect the oxygen

Figure 1. Oxygen optode in water sample bottle on the left side, and complete experimental setup to the right.
concentration or other parameters of the water sample, is very accurate, and measures extremely small variations. By measuring a change in oxygen concentration over time in a water sample in a vial with a gas-tight seal, the respiration rate of microorganisms and zooplankton in a sample can be obtained. An analytical setup based on this method using oxygen optodes was previously developed at the Umeå Marine Sciences Centre (Wikner et al. 2013). The sensor is connected to a one-liter glass bottle containing the water sample and the bottle is placed in a water bath with precise temperature control (Figure 1). The water in the sample is circulated with a magnetic stirrer and the sensor takes one measurement every minute for about 24 hours. This setup enables measuring plankton respiration from water samples at different temperatures and dissolved organic carbon content.

1.3 Current solution to data flow
The existing procedure for analyzing the sensor data begins with importing the sensor data text files into separate spreadsheets in Microsoft Excel. Data import requires adjustment of the import settings and manual copying and pasting of data into the spreadsheets. Then, some of the columns must be converted to the appropriate formats, such as date and time. A few graphs of the data must also be created manually. By plotting a time series of the oxygen and temperature readings, it is possible to see where the values have stabilized and thus, what data to include in the statistical analysis. These readings are inserted manually in the Excel cell pathway for several statistical quantities, with risk of manual errors. The linear regression tool that is part of Excel’s statistical functions is used to find the slope of the oxygen concentration as a function of time, which is the rate of oxygen concentration change per unit of time. The regression tool also gives a table of regression statistics and analysis of variance (ANOVA) results. This requires manual definition of ranges for x- and y-values as well as position of the statistical results. These results are then summarized in another spreadsheet by set links.

1.4 Problem description
Because this method produces a large amount of data, especially when performing many measurements simultaneously, the analysis and interpretation of the data are time consuming processes. The number of actions that must be performed manually for each data file takes time and introduces a risk of human errors. To ensure the quality of the data, a reliable method of producing overviews and statistical summaries from the data is required. Thus, a dedicated software for handling and analyzing the data obtained from the sensor is needed to improve the work flow, which will reduce time spent on data analysis and increase the quality of the data.

1.5 Aim of the project
The aim of this project was to develop software that imports the data logged from the optodes, to perform an analysis of the primary data and then to display and export the relevant statistical information. The software should also assure data quality in a traceable manner. To do this, the application was written in the Python programming language and features a graphical user interface to control all of its functions.
1.6 The new bioinformatics tool

The new application would have to perform all of the above functions, to automate as much as possible, and to simplify all the steps that require human input. To do this, the program needs to import and convert the data to a correct format, to display tables with different data, to draw graphs, to calculate the regression analysis and to have functions for manually choosing what data to include in the analysis. Python is a suitable language for scientific computing due to the many free tools, modules and libraries that are available (Oliphant 2007).

In the application, the graphical user interface (GUI) was handled by the PySide (Qt Project 2015) library, which enables the use of the Qt framework (The Qt Company Ltd. 2015) in Python applications. The graphs were made with matplotlib (Hunter 2007), which is a 2D-plotting library for Python that can be used together with Qt. The statistical computing is done with help of the SciPy (Jones E 2001) and NumPy (Stéfan van der Walt 2011) libraries.

2 Experimental

2.1 Theory

2.1.1 Data analysis

The key information obtained from the sensor data was the rate at which the dissolved oxygen concentration changes over time, as this is a direct indicator of how much aerobic respiration takes place in the sample. Ideally, the rate of oxygen concentration change should be constant throughout the test, but this is not always the case, as there can be many unknown factors influencing the sample. The biggest influencing factor is the temperature, which needs to be kept constant. To ensure that the temperature has remained constant throughout the experiment, the oxygen optode also records the temperature. One part of the data analysis was to view a time series graph of the oxygen and temperature data points to verify that they behave normally. Problems with air leaks or temperature control can usually be identified then. During the first hour of the measurements, some time is required before the sample reaches its target temperature. These data points were excluded from the analysis.

2.1.2 Regression analysis

To determine the oxygen consumption rate, a line was fitted through the oxygen concentration data using a simple linear regression method. Simple linear regression is a linear regression model with a single explanatory variable (independent variable/predictor), which fits a straight line to the response variable (dependent variable) by the least squares method (Box et al. 1978). The slope of the line, which is its first derivative, is the same as the rate of change. The ordinary least squares (OLS) method minimizes the sum of squared residuals to find the optimal fit, which can then be expressed as a simple formula. Two assumptions are made in this model, unbiasedness and a normality assumption for the error terms. Unbiasedness requires that the estimators are interpreted as random variables with an error term whose average is zero, and the error terms are assumed to be normally distributed. Confidence intervals are used to show how precise the estimates are, and the standard
method for calculating them for linear regression coefficients relies on the normality assumption, which is justified if errors in the regression are normally distributed, or the number of observations is sufficiently large.

While the oxygen concentration data typically followed a straight line, there were cases where the data had a visible curved shape, indicating a bad fit of a line or that only a very limited section of the data was used to estimate the slope. Previously, the simple linear regression is the only model used, but in this project, a second regression method was implemented as an option to linear regression. A second degree function provides a better fit to the curved data and thus, a polynomial regression method was adopted. Polynomial regression is a special case of multiple linear regression where the relationship between the independent variable \( x \) and the dependent variable \( y \) is modelled as a polynomial, which for this project is of the second degree. As with simple linear regression, the least squares method is used to minimize the residuals and to find the best fit. A nonlinear regression method was also tested. However, the polynomial regression had an advantage as compared to using a natural logarithm based function, in that it did not require an iterative process to estimate slopes and statistics.

2.2 Solution strategies

2.2.1 Importing data
The first requirement of the new software was to import the optode sensor data from the log files. Importing text is very straightforward in Python. The log files consisted of tab-separated values, with headers followed by data rows. The logs were formatted with one row of metadata headers, followed by a row with the corresponding values, and then after a blank line the optode sensor serial number. On the fifth line were headers of the actual data, and then every subsequent row contained the data values measured for that time point. By assuming that the log files always adhered to this format, the import function could store the metadata from the second row and the sensor readings from the fifth line onwards. Such a simple function was very reliable, but if there were any changes in how the sensor data was saved in the log files, the import did not work. To avoid this, a better solution might be to let the user adjust the parameters of the import to set which rows contain data and which columns hold the relevant measurements, as this would avoid problems of assuming the column names remain constant. This solution was not used in the application because implementing it would be very time consuming.

2.2.2 Graphical user interface
An important feature of the application was to view all basic information of a sensor log file, the data values, the statistical results and appropriate plots. It should also be possible for the user to load multiple log files and switch between them. Other than the necessary functionality, it is important for a GUI to be user-friendly, which is defined as software being easy to use or understand. This means that the layout of information and interactive elements in the application window should be designed to let the user get the results as fast as possible. Some GUI elements, e.g. the ‘file’ menu, are used for common actions like opening or saving files. Including these common features will aid new users in learning how to use the application. To implement a GUI
In Python, any of a huge number of GUI frameworks (or toolkits) can be used. For this project, PySide was chosen, which is a Python binding for the Qt cross-platform application and UI framework. Qt was chosen because of its support of multiple platforms, and widespread use in software applications.

2.2.3 Graphs
Data visualization in the form of line charts provides a quick overview of the sensor data. Time series graphs have been used previously for oxygen concentration, temperature and the related air saturation. Showing these plots along with the statistics results and summaries for each sensor data file, therefore, seemed like the best solution. When viewing a list and summary of all data files imported into the program, another group of graphs were needed: a bar chart of the oxygen consumption rate for all files, a control chart of the coefficient of variation (CV) for each analysis and a control chart with the degree of explanation (R², coefficient of determination) of absolute oxygen concentration versus temperature on the y-axis, and the oxygen consumption rate on the x-axis. These graphs have been used in the method before and are useful for verifying the quality of data. For creating these graphs in the GUI, a Python library called matplotlib, which provides extensive graphing capabilities and is compatible with Qt, was used.

2.2.4 Editing data ranges for regression analysis
When sections of the data are more influenced by other factors than planktonic respiration due to initial temperature equilibration, temperature irregularities, interrupted stirring or other disturbances of the incubation, the user needs to adjust what data points are included in the regression. Previously, values were selected to be used for the analysis from the data tables in a spreadsheet, but for this new application a better method was needed that did not require as many manual steps. A table with all the data values from the log file was visible in the new application and by using mouse selection, values were chosen for exclusion. A few functions, such as marking rows for exclusion, clearing the selection, confirming the exclusion and redoing the regression analysis on the edited data range, were added.

Calculating the regression function for a least-squares linear regression can easily be implemented, but methods are also available in the NumPy and SciPy libraries that can calculate linear and polynomial regressions. The problem with the available functions is that they do not return all of the statistics that might be needed, such as the residual standard error.

2.2.5 Saving analysis
There is often a need to either review or continue a previous analysis of files in the application. Thus, the potential to save your work and to continue where you left off were important features to add to the new application. Rather than creating a custom format for saving all data to files, a much less time-consuming implementation employing the “pickle” module available in the Python standard library was used. This module serializes Python object structures and saves them to files, and then deserializes them to retrieve the objects.
2.2.6 Exporting summary
After the analysis and quality assurance is completed for a set of files, the most important results need to be saved in an easily accessible format. This was achieved by exporting the results as a data table formatted as a text file in the “comma separated values” (csv) format.

2.2.7 Program access
Python programs are distributed as source code only and rely on the user to get the correct version of the Python interpreter and to install the needed modules. This is suitable for any programmers that might need to work with the code, but for the end user of a finished application, this may become an unnecessary barrier to being able to use the program. A second option is to distribute the program together with a complete Python package; however, this also has problems due to file size and number. A third option is to package the python code into a stand-alone executable program. There are third-party tools such as PyInstaller that can do this for common operating systems.

2.3 Implementation

2.3.1 File import
The file import functions were the first parts added to the program. The sensor log files were imported by assuming the column headers and data values always start on the same lines. However, the order of the columns did not matter, as the data was then stored in a dictionary object (hash table) with the names of the columns as keys and the data for each column in a list. As long as the date, time, O$_2$ concentration, temperature and air saturation columns were present, the file import worked. For each new file that is imported, a new instance of a “data handler” class was created that stores all imported data, regression results, tables and all other sensor data file specific information.

2.3.2 Regression analysis
The regression analysis was performed instantly when files were imported to be able to show the file summaries and plots. The results could then be viewed to determine if any parameters needed to be changed. Then, if the included data range was changed, the regression would be redone. If the statistics results are satisfactory for the file, the user can mark it as “quality assured”. The type of regression, either simple linear or polynomial regression, can be chosen by the user. The calculations for the linear regression were implemented in the program to obtain the regression function, confidence intervals, R$^2$, standard error of estimate and standard error of residuals. Additionally, the “linregress” function from the SciPy library was used to find a two-sided p-value for a hypothesis test whose null hypothesis is that the slope of the regression is zero. SciPy was also needed to calculate the t-value with the complementary cumulative distribution function of the t-statistic, which is needed to compute the confidence intervals. The polynomial regression was performed very similarly, but the regression function was found by using the polyfit function in SciPy. Other than fitting a line to the data, a number of other values were calculated. These include the linear correlation between O$_2$ concentration and temperature, regression
start and end time, average O\textsubscript{2} concentration, O\textsubscript{2} variance, average O\textsubscript{2} standard deviation, initial O\textsubscript{2} and standard deviation (average of first 20 values).

2.3.3 User interface
The GUI was designed as one main window with the left side containing a tab widget (a widget is an element of interaction in a GUI, such as a table, button or menu) and the right side containing a plot area (Figure 2). A menu bar was placed at the top of the window and contained “File”, “Options” and “Help” menus. The file menu has actions for importing files or folders, saving the current project, opening a saved project, exporting a results table, closing the current project or exiting the program. The tab widget has four tabs, File list, Sensor data files, Quality assured files and Compiled results.

![Figure 2. The main window with four files loaded and the file list tab selected.](image)

In the File list tab, a summary of imported files can be seen. The Water sample, Treatment, Measurement operator, Data analysis operator and Note columns have editable text fields for organizing the experiments. Any changes to these tables will also update the metadata of the data handler object for the file. There is a column for “Quality Assured” where a checkbox indicates if a file has been analyzed and marked as ok. On the right side, the three summary plots are shown. The elements of the window are organized in layout widgets, which adapt the size of the elements to fit the window size. The plot area will always occupy half the horizontal space as long as the window is wide enough to show all tabs and buttons on the left side. The window size can be reduced to about 800 by 400 pixels, but larger views are preferable.

The plot area was implemented as a subclass of the Qt “FigureCanvas” class, which holds a matplotlib “Figure”. This was possible since a Qt user interface backend is available with matplotlib. Every time the plots need to change the contents of the Figure is cleared, and the new plots are drawn to it. Below the plots is the matplotlib toolbar, which contain a set of standard tools for manipulating the view in the plots,
or saving the plots as images. Any changes made in these plots will, however, be lost when the plots are redrawn.

Under the Sensor data files tab, there is first a drop-down menu where any of the imported files can be selected (Figure 2). Then there is another tab widget with “File Summary”, “Statistics” and “File data” tabs. These tabs all have tables with different information relating to the file. The table in File Summary contains basic information, the same as can be seen in the File list table as well as additional statistics from the regression analysis of the data. The Statistics tab has a table with many more values from the statistical analysis along with their units where applicable and is useful for verifying that the regression results are valid. Finally, the File data tab has a table that is an exact representation of the sensor log file. This table contains every column and row that is imported into the program as well as the tools for selecting and excluding data points for the statistical analysis. When the Sensor data files tab is activated, the plot area will switch the plots to the ones corresponding to the active file (Figure 3). The first plot is a dual axis graph with the O₂ concentration and temperature data on two y-axes and time in hours since incubation start on the x-axis. The second plot is for air saturation. If the Graph selection option is enabled, a horizontal span of data points can be selected from the graph by clicking and dragging the mouse cursor over them. There is also an option for displaying the data as markers (dots) instead of as a continuous line. A line for the regression function is drawn on top of the data and shows how good the fit is. The default colors of all the lines can be changed from the options menu in the application.

![Figure 3. Excluded data is shown in orange, selected in blue. Selections can be made from the table or the graph.](image)

Since anomalies are sometimes seen in the oxygen and temperature graphs, one of the important features of the application is the ability to change the intervals of data points that are included in the regression analysis. In the application, two ways of selecting
Data points were implemented, either by selecting rows in the “File data” table or by dragging a span in the oxygen concentration graph. Selecting cells in a table is standard function of Qt tables, but the behavior was modified to select whole rows, and not to deselect everything when clicking an unselected row. Instead, clicking a row toggles its selection state, and dragging across multiple rows will select/deselect all of them. The selection from the matplotlib graph is possible due to a “span selector” available as a matplotlib widget. To confirm the exclusion and to redo the regression analysis on the modified data range, the “Confirm exclusion...” button is used, which will add the file and new statistical results under the “Quality assured files” tab.

All files marked as “quality assured” are listed in the “Quality assured files” tab in a very similar way to the “Sensor data files” tab. The main difference, other than the included data and resulting statistical values, is that it is not possible to exclude more data points from this tab and have another quality assured version of the analysis. If needed, the data selection for the analysis can be redone by going back to the previous tab. In the Q.A. tab, a switch between simple linear regression and polynomial regression may be done by clicking a checkbox. Nothing is recalculated for this switch, so the change is instant and the plots and tables are switched to the other type of regression. When using polynomial regression, the rate of change of the oxygen concentration can be measured at any point of the line by choosing the time point from a number input field.

The currently loaded files, analysis data and all changes to them can be saved to be opened later by using the “Save as...” or “Save” actions available from the file menu. This is implemented by saving the underlying python objects storing all file data and variables to a file using the “pickle” module available in the Python standard library.

Finally, when all sensor files are imported and the quality of the data has been verified, the results of the analysis can be exported. To do this, the “Compiled results” tab is selected and then a summary table can be created. By clicking the “Create new table” button, an options window is displayed that allows the user to select which files and which results to include in the summary table (Figure 4). The table shows a row for each included file and a column for each selected value.
2.3.4 Distribution
The finished application was packaged into a stand-alone executable program for Windows using PyInstaller. By doing that, all of the necessary third party Python libraries were included and no Python interpreter needed to be installed on the system to run the application. This resulted in a file size of about 30 MB when compressed in an archive format, which makes distributing the application very easy.

Results

2.3.5 Verification of statistical results
A previous analysis done with the old method using Excel was compared to the results of the new application. The three sensor data files that had been analyzed were imported to the program, and the data ranges were set to include the same values in the program as had been used in the original analysis. The resulting statistical data was exported from the program and then put together with the matching values in the Excel analysis (Table 1). This validation was repeated for three different experiments.
Table 1. The compiled results from the analysis done in Excel (darker red color) followed by the same values exported from the program (light green color).

<table>
<thead>
<tr>
<th>CalibOxyg (μM)</th>
<th>AbsOxySlope (μM hour⁻¹)</th>
<th>AbsOxyP</th>
<th>AbsOxyR²</th>
<th>AbsOxySlope95</th>
<th>AbsOxyYcept (μM)</th>
<th>AbsOxyYcept SE</th>
<th>CorrAbsOxyTemp</th>
</tr>
</thead>
<tbody>
<tr>
<td>345.25</td>
<td>0.18</td>
<td>-0.138</td>
<td>0</td>
<td>0.94</td>
<td>0.002</td>
<td>344.93</td>
<td>0.013</td>
</tr>
<tr>
<td>364.89</td>
<td>0.10</td>
<td>-0.387</td>
<td>0</td>
<td>0.98</td>
<td>0.003</td>
<td>364.38</td>
<td>0.020</td>
</tr>
<tr>
<td>360.47</td>
<td>0.77</td>
<td>-0.153</td>
<td>0</td>
<td>0.69</td>
<td>0.005</td>
<td>360.68</td>
<td>0.040</td>
</tr>
</tbody>
</table>

The only differences between the output of the new program and the old method other than names and rounding are the calibration oxygen standard deviation values. This can be explained by the choice of the calibration data points, which should be the first 20 oxygen concentration values included in the regression. The selection of the calibration points was automatic in the new application, but when chosen manually in a spreadsheet, the calibration data sometimes included 21 values or were shifted one time step.

2.3.6 User tests
Two persons with a biotechnology background but with no knowledge of the experimental method were asked to test the program. They were able to perform a basic analysis in the program with a minimal amount of guidance, equivalent to what can be obtained from reading the quick start guide. Their goal was to import a data file, exclude periods of unstable temperatures, and then export a statistical result table. For both persons, this was accomplished in less than 30 minutes. The researchers who had previously worked with this method reported that a complete analysis can be done in 5 to 10 minutes with the application compared to 60 minutes when doing the analysis in Excel (Figure 5).
2.3.7 Performance
No heavy computations were needed for the analysis. Thus, importing data needed less than 1 second per file. In the application, no lag in the user interface was seen with the exception of selecting data for exclusion from the data table, since the plot is updated for every new selection. The slowest part of the application was starting it, which can take 10 to 30 seconds depending on the computer. Start-up was greatly affected by the packaging of the source code with PyInstaller, as starting the program with a normal Python interpreter takes about 5 seconds. The compilation did, however, decrease the number of files from a complete Python installation with the necessary modules from about 16000 files to 850 and the file size from 400 to 100 MB. When the compiled application was compressed in an archive, it was reduced to around 30 MB. The application was then easily accessed by distributing the compressed archive to users and no further files or programs were needed to run it.

3 Discussion
Clearly, the new dedicated analysis software is a faster means of interpreting the results from the sensor data, as seen in Figure 5. This translates to a reduction in operating costs for continuous use of the method, which can help make the method a competitive alternative to the old standard oxygen monitoring systems. A computer program that has been shown to output the correct results for a number of different analyzed files should continue to produce the correct results with any new files of the same format, since the algorithms are always the same. But showing that the results are more reliable than the previous method is difficult as computer applications can have bugs that only result in an error in certain specific circumstances. The problem is that to verify that the error frequency is lower than another method the whole analysis must essentially be done again with some independent method for comparison, and repeated many times.

Another subject to discuss is how to handle the anomalies that are sometimes present in the data. In many cases, the irregularities in the data are easily identified and
excluded from the data set, but sometimes larger parts of the data are affected or the cause of the anomaly is unknown. An example is when the oxygen consumption rate starts off at a higher rate and slowly reaches a linear rate. Depending on when the system is considered to be balanced and on the start point of the analysis set, the observed rate could differ significantly. These types of problems can be helped by fitting a non-linear function to the data, such as the polynomial regression implemented in the new application. But, to do this, the question of how to choose where on the curve the oxygen consumption rate would best reflect the true value that would have been observed at the water samples source would need to be addressed. Typically, the oxygen consumption rate in an enclosed sample is the least biased as close as possible to the start of the incubation, due to potential artefacts introduced by placing the water sample in the bottle (exclusion of growth factors, adsorption to walls, wall growth etc.). It is also important to be careful when removing “bad” data, as this comes with a risk of introducing a selection bias. A possible solution might be to have an automatic filtering algorithm that uses temperature data to remove the initial equilibration time, in a way that is consistent enough to not introduce any bias. Other anomalies are probably harder to filter out as their causes might be unknown and they only rarely occur.

The improvements in the method that can be achieved by a good software solution is apparent, and in a wider perspective, this is true for many other areas of research. With the rapid evolution of digital sensors and equipment, the amounts of data produced increase accordingly. This creates the need for software that can interpret large amounts of data, and for the more specialized or very new methods, no software is available. A programming language like Python is ideal for solving these problems, as it is dynamic enough to be adapted to any problem and is easily extendable with modules for scientific computing or even bioinformatics applications.

4 Conclusions

The new application performed the same analysis as was previously done manually in a spreadsheet software and extends the analysis by including a polynomial regression alternative. The goal of the project was to automate and to speed up the analysis process while ensuring the data quality. The time required to do a typical analysis was changed from about an hour to five to ten minutes (i.e. 12-6 times) and the risk of human error was minimized, as the number of manual steps involved was greatly reduced. Thus, the project achieved its goal. The tools used to create the application did not lack any needed functionality and the new application demonstrated that Python is a very good choice for this type of software.

5 Proposed future work

In the scope of what the application was designed to do, the future work that might be necessary is to fix any bugs that are found and to adapt the import functions if the optode sensor data format changes. An addition that would speed up the work flow even further is for the application to store sensor data directly to the database management system. While the application performs the task it was designed to do,
examples of how the application may be expanded follows. The main limitation is that it only handles one type of data format and only performs a specific analysis. It would be possible to make it into a more general statistical analysis software for batch analysis of sensor readings. To do this, one necessary change is to allow the import of data files with an arbitrary format, and then to have a function for saving import presets for different file formats. This would maintain a fast speed of importing multiple files. In a similar fashion, a specific analysis to be used for a certain type of sensor data could be set up. Linear regression could be performed on whichever data set is suitable and other needed statistical analyses could be added. For other sensor applications, multi-parameter phase shift harmonization and different types of time series analysis may optimize data flow to utilizable knowledge.
6 References


