MEng Group Project Final Report

Cristian Badoi, Aaron Butterworth, Daniel Gardam

1 Aims and Objectives

The aim of this project is to create a system capable of calculating, storing and predicting formulae of chemical compounds given a set of precursors, with a focus on optimization and finding an acceptable solution within a given set of bounds. The program is designed to be remotely accessible, with a web server providing a user front end and a space for necessary calculations, and a database for the purposes of storing chemical data and user account information.

The project will primarily use C++ for computation, NodeJS with several popular packages for the web server and user interface, and MongoDB for the database system.

This project is produced in conjunction with the Materials Innovation Factory and is intended for their use. The development and evaluation of the project will be performed in cooperation with the facility.

This project is mainly based on the prior research topics and problems encountered by researchers in the MIF, for example having a person test possible compounds and slowly refine the formula based on their results is a time and material inefficient process, therefore if we can suggest compounds which are either direct results or very close approximations to their desired outcome then this process can be minimised.

1.1 Ethical Use Of Human Participants

As with one of our undergraduate projects data from members of the project group does not constitute 3rd party evaluation, and with non-project team members we will be strictly following the CS Department ethical procedure for MEng project 3rd party evaluation. All other data used in the project is well known public data regarding the behaviour of elements.

We believe the combination of these factors allows us to proceed without further application for research ethics approval.

2 Background

3 Intended System Design

The initial system design is presented here as it was in our initial specification and design, we will highlight any differences or deviation from this in the actual system design section.

3.1 Expected Components

3.1.1 Web Server

The main interface to the system will be supplied by a web server to allow easy and portable access by members of the MIF. The web facing part of the system will be comprised of several smaller packages all working together under NodeJS:

NodeJS — A JavaScript runtime that allows execution of JS outside the context of a web browser.

Express — An open source web application framework for NodeJS, it is responsible for the high level web server logic and allows a high level interface to web traffic as it handles response codes, etc. itself.

Pug — An open source high-performance template engine, it allows us to alter the contents of a web page just before serving it to the user, allowing for finer levels of customisation while keeping the code base manageable.

Express-Session — An add-on for express that gives us easy control of user sessions. This allows us to keep track of user specific data, such as preferred precursors or recently searched compounds.

bCrypt2 — A widely trusted package for providing user password encryption.

3.1.2 Computation

The actual computation will be handled by C++ programs with some small python helper scripts for unit conversion and other pre/post processing.

This is done for several reasons, firstly to allow a significant performance increase over JS; secondly to allow us access to widely known mathematics libraries; and thirdly to allow easier modification of the source code by others in the MIF as these are their preferred languages as opposed to JS, this is also the start of an ongoing project which will be maintained and expanded upon after us so maintainability is an important factor.

The first step of computation is the stoichiometry calculator. This takes a selection of user defined elements, and then calculate all possible balanced compounds that may be produced by a chemical reaction of these elements. This process also takes into account user limits such as the maximum amount of atoms in the resulting combination to refine the search and prevent the creation of a technically correct but practically unreasonable result.

The programs data sources are the web interface which will invoke the users query and the database for relevant information about each element, its output is piped back to NodeJS where it may be cached in the database if it is a common search, the result is then displayed to the user via the web interface.

The second stage is the precursor calculator. Its input is a desired ratio of elements, and a set of available chemical precursors. It calculates possible combinations and quantities of the precursors that when mixed together create the desired target ratio. User bounds may also be given to keep the results practically usable.

3.1.3 Database

With the theme of future expansion the database will be hosted on a separate server instance running independently of the NodeJS process, this allows for easier expansion of the system if it cannot handle the user load placed on it as both the web server and database can be scaled completely separately from each other without modification to the program code.

The database will not only store the information needed to run calculations, but will also store the solutions to commonly executed combinations so as to reduce the computational demand by allowing these solutions to be quickly retrieved rather than repeatedly calculated.

4 Actual System Design

4.1 Computation

4.1.1 Stoichiometry Calculator

The first stage of the stoichiometry calculator is to form the **A** and **B** matrix. As we are always searching for charge balanced compounds, and the result is purely a combination of the inputs with no external factors (e.g. The reaction taking place in air vs under vacuum) we can fix the **B** matrix. In this example the user has queried for Aluminium, and Oxygen:

$$\mathbf{A} = \begin{array}{ccl} \text{Al} & \text{O} & \\ 1 & 1 & \text{Initial Quantity} \\ 3 & -2 & \text{Charge Imbalance} \end{array}$$

$$\mathbf{B} = \begin{array}{cl} 1 & \text{Resulting Proportion} \\ 0 & \text{Desired Charge Imbalance} \end{array}$$

$$\begin{bmatrix} 1 & 1 \\ 3 & -2 \end{bmatrix} \times \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

This gives us a matrix representation of the constraint equations. However, by our initial design we must consider all possible combinations of oxidation (charge) states in order to find all charge balanced compounds, in this example O has a total of 4 states, and Al a total of 3 giving a total of 132 permutations. While this is true computationally, some of the oxidation states are extremely uncommon and would not be desirable in a physical setting.

With this in mind our implementation differs from the design at this point, if we consider all possible charges for every element the calculator produces technically feasible but practically impossible results. As this tool is designed to be used by chemists in a real world application we decided to limit the space by removing the rarer charges from the search. However, this data is dynamically loaded we can offer support for rarer charges to be included in the future, however at the current version this is not included.

This space reduction leaves only one charge each for Al and O, significantly reducing the search space from 132 variants of **A** to only one, while still returning all common compounds.

After having formed the **A** matrices, we solve each for a single solution via LU decomposition. Which in it general form takes a matrix **X** and finds a lower (**L**) and upper (**U**) triangular matrix where

$$\mathbf{X} = \mathbf{LU}$$

As these are triangular matrices we know the first row of **L** will take the form:

$$\begin{bmatrix} x & 0 & 0 & \dots & 0 \end{bmatrix}$$

With each subsequent row adding another value, left to right, in place of the zeros. By forwards substitution we can obtain a solution in the form:

$$\mathbf{L} \times \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} = \mathbf{B}$$

Calling this solution vector $\hat{\mathbf{y}}$ we can then solve $\mathbf{U}$ with back substitution for our actual solution $\hat{\mathbf{x}}$:

$$\mathbf{U}\hat{\mathbf{x}} = \hat{\mathbf{y}}$$

4.1.2 Precursor Calculator

Most of the major deviations from the design are within the precursor calculator. We believe this is due to our initial unfamiliarity with the subject field, therefore as our familiarity grew we changed our methods to be both faster, and more effective.

4.2 Data Structures

The main source of data for the system is the MongoDB database. We are using it to cache results allowing us to lighten the computational load of the system and deliver results faster when several users query the system at once.

4.2.1 Database Structure

Database tables are generated on-the-fly as required, however general types of table do exist:

UserData — Usernames, Encrypted Passwords, and other user preference data.

ElementData — Charge configurations, and other static element data needed for calculations.

CalcedPoints — Generally a table per invocation of the precursor calculator, storing the results of our calculations so they can be easily referenced in future.

The exact structure of a CalcedPoints table depends heavily on the query that created it, however given a query its structure will always be predictable:

<Desired Ratio>		
<Precursor 1>	<Precursor 2>	...
<Ratio of P1>	<Ratio of P2>	...
⋮	⋮	⋱

Each row of the table is a possible solution with the values in each column representing the proportion of that precursor relative to the others.

4.2.2 System Memory Management

On the computer which runs the main web server instance availability of RAM may become an issue, this would mainly be caused by multiple large queries executing simultaneously and such a situation is hard to avoid, however the memory footprint of the application is kept to a minimum by invoking the calculators as child processes instead of keeping them with the main NodeJS thread.

This allows the memory heavy segment of the system to only take up space in RAM when it is required, and also only occupy the exact amount it requires during computation. If resources are not available for a query to use it will be queued until they are available on a first-come first-served basis.

4.2.3 Computation

The primary data structure in use during the computational stages are matrices, they are used to store the coefficients that represent the system of equations we have generated and will then solve. The software does occasionally cache small amounts of data for its own use, this is not stored in the database and is instead written and read as comma separated values in text files.

4.3 Interface Design

The main user interface is provided via the web server, we have opted for a single page style design: the further down the page you go, the more in-depth the data.

The top of the page displays a periodic table a user can use to enter elements they would like to pass to the stoichiometry calculator who's result will be displayed below the periodic table as it is the next level of detail.

Next the user can select a desired final ratio of elements and a set of precursors they have avalible from the stoichiometry calculator, this will then trigger the precursor calculator who's results will be displayed graphically as either a 2d or 3d triangle/pyramid. This visualisation will be rendered using WebGL in order to allow us to have an interactive 3d environment.

The user will then be able to explore possible solutions via the visualisation, selecting a point then a direction to move in, more possible solutions in their desired direction will be generated and displayed allowing them to find a good ratio for their own uses, be this quantities of precursor avalible or ease of measurement.

4.3.1 User Data

A user will be able to login if they choose to, accounts are not required. Logging in will offer benefits such as previous searches being remembered and the ability to define preferred precursors which will be displayed separately to the generated precursors to allow easier selection of the precursors they will be using most often.

Users accounts data will be protected by bCrypt2 and on disk encryption if possible. Also enforcing a HTTPS connection to the website should ensure password security, as the amount of user provided text is low there is little opportunity for SQL injection, most of the users interaction with the server is via click-able GUI elements which provide a minimal security risk.

5 Evaluation

The system will mainly be evaluated by personnel in the MIF as they have collectively commissioned the system to be build, participation in this evaluation will be optional and informal aside from a selected group from the project group itself. With this select group we will collect more in-depth data in the form of questionnaires.