ChemCalc: a building block for tomorrow's chemical infrastructure

Luc Patiny, Alain Borel*

Ecole Polytechnique Fédérale de Lausanne, CH-1015 Lausanne, Switzerland

* corresponding author luc.patiny@epfl.ch

ChemCalc: a building block for tomorrow's chemical infrastructure

Luc Patiny, Alain Borel*

Ecole Polytechnique Fédérale de Lausanne, CH-1015 Lausanne, Switzerland

* corresponding author luc.patiny@epfl.ch

Keywords: web services; mass spectrometry; HTML5, JSON, molecular formula, monoisotopic mass, cloud computing

Abstract: Web services, as an aspect of cloud computing, are becoming an important part of the general IT infrastructure, and scientific computing is no exception to this trend. We propose a simple approach to develop chemical web services, through which servers could expose the essential data manipulation functionality that students and researchers need for chemical calculations. These services return their results as JSON (JavaScript Object Notation) objects, which facilitates their use for web applications. The ChemCalc project demonstrates this approach: we present 3 web services related with mass spectrometry, namely isotopic distribution simulation, peptide fragmentation simulation and molecular formula determination. We also developed a complete web application based on these 3 web services, taking advantage of modern HTML5 and JavaScript libraries (ChemDoodle and jQuery).

Introduction

The use of computers for solving chemical problems is almost as old as modern computing itself, with practical examples in physical and analytical chemistry published before 1950.[1–6] Digital and personal computers, followed by the Internet and the World Wide Web have naturally strengthened this long-standing tradition. Today, it has become trivial to write that computers are everywhere and that they can take up an ever-increasing part of a worker‘s daily burden. This is of course valid in science as well, and many useful software utilities have been developed along the years to solve the problems of researchers, teachers and students.

However, these tools do not necessarily match the needs of their prospective users: one often hears complaints that the proposed solutions are inconvenient. This might mean that they are difficult to use or simply that interoperability issues prevent their application in a given workflow. Fortunately, the advent of the World Wide Web (especially after interactive web pages became possible) has significantly improved the situation. Users can now take advantage of a familiar interface (the web browser) for many different tasks, and numerous examples demonstrate that the web browser has become a mature platform for general (Google Apps being a typical example[7,8]) or chemical computing.[9–11] Furthermore, this platform facilitates software deployment, since any update (fixing security problems or introducing new features) can be made immediately available to all users. Another non-negligible advantage of this approach is that such software products are usually independent of the underlying operating system, and thus easier to adopt in any existing environment. For example, the Chemical Abstracts Service used to release upgrades of the local client version of Scifinder Scholar every year or so – with a different schedule for the Macintosh and Windows platforms. Since the introduction of their web version in 2008, the time lag between updates has been reduced to about 6 months, only considering feature improvements. We can assume that security and minor performance improvements are being silently pushed to the users even more frequently.

Programmers face another challenge, namely producing code suitable for maintenance and future evolution. It is well-known that modular software design, where the program is divided into a series of unit components, is in principle a good answer to this problem. It makes the components easier to write and test, and it encourages the re-use of existing code, which in turn accelerates development. The web offers attractive opportunities from this point of view, as it becomes possible for a developer to take advantage of components hosted on distant servers. Such remotely accessible applications are commonly known as web services[10,11] and have become part of the broader spectrum called cloud computing. Nowadays, several well-known web sites of chemical interest, such as ChemSpider,[12] Chemical Entities of Biological Interest (ChEBI)[13] and PubChem,[14] propose web-service interfaces.

A number of standards have been developed over the years to facilitate the use of web services, as summarized for example by Dong et al.[10] These standards are usually based on the Extensible Markup Language (XML), and provide among other things machine-readable descriptions of web services. As such, they provide a sound foundation for many advanced applications. However, they also make the development of web services and their use in light-weight client applications rather complex.

In this paper, we propose a simpler scheme for web components applied to chemical computing applications. The typical client we have in mind is a web application, which could be a short script built into a web page or something more sophisticated. This client will prepare a set of simple input data describing the chemical system of interest, and possibly the expected computation or output format. The data will be sent to a server that hosts the web component, in general as an HTTP POST request, which can be achieved easily thanks to the AJAX paradigm.[15] After processing, the server component will output some data in JavaScript Object Notation (JSON) format.[16] This lightweight text-based data-interchange format is easy to read and write for humans, as well as relatively compact compared to other solutions such as XML.

Last but not least, a JSON expression is trivially converted into a native object in a JavaScript program, which makes it especially attractive for web applications thanks to the broad availability of embedded interpreters in modern web browsers. Nevertheless, JSON is not limited to JavaScript and libraries for JSON processing are readily available for many other programming languages.[17]

We demonstrate our approach by applying it to several common problems in mass spectrometry. Converting molecular formulas to molecular masses is of course a very basic instrument in the chemist's toolbox. Thus, we can take advantage of the familiarity of the underlying chemical problem and focus on the computer implementation. In the following sections, we show how our web component can be used easily in a number of different contexts, as exemplified in ChemCalc, an application with several features of interest for mass spectroscopists – available both through a user-friendly web interface and a developer-friendly Ajax programming interface.. First, we present a simple web service to calculate isotopic distributions for a given molecular formula. Secondly, ChemCalc provides a convenient interface for the manipulation of peptide and protein sequences, dealing with their fragmentation. Finally, we offer a useful function for the decomposition of a given monoisotopic mass into possible molecular formulas.

Web service interface

The Application Programming Interface (API) of our web service is quite simple. In essence, it expects a molecular formula string *mf* as its main input, together with an extra argument that specifies the format (JCAMP[18] or XY values) that will be encapsulated in the output JSON object. The arguments are used in an Ajax POST request, as given in this short JavaScript example (using the popular jQuery library for convenience):

```
<script
src="http://ajax.googleapis.com/ajax/libs/jquery/1.7.1/jquery.min.js"></script>


<script language="javascript">
  function chemcalc(mf) {
       jQuery.getJSON("http://www.chemcalc.org/chemcalc",
              {mf: mf, isotopomers: "jcamp,xy"},
              function(result) {
                     console.log(result);
              }
       )
  }
  chemcalc("C100H100");
</script>
```

In the above example, the Ajax request is sent to a `/chemcalc` URL on the www.chemcalc.org server. By default, this will only work if the script is used in a page on that specific server, which can be inconvenient. In order to take full advantage of the distributed computing capability brought by a web service, it is much more interesting to host this service on a different system, able to respond to various application servers.

However, for protection against the so-called *cross-site scripting* attacks (see for example Holdener[15] p. 919-920), Ajax requests to a server outside of the current domain are forbidden at the browser level. For ChemCalc, we chose to address this problem using *cross-origin resource sharing* (CORS), where the service provider (in this case www.chemcalc.org) includes the specific `Access-Control-Allow-Origin: *` header in its HTTP responses to indicate that the browser should allow the request to proceed despite the different domain. In the popular

Apache web server, this is achieved by using the `mod_headers` extension and including the following line inside the `<Directory>`, `<Location>`, `<Files>` or `<VirtualHost>` sections of `httpd.conf`, or within a `.htaccess` file:

```
Header set Access-Control-Allow-Origin "*"
```

Similar mechanisms exist for other HTTP servers.

For the sake of completeness, we mention another possible solution to this problem. One can set up the web application server to act as a proxy, whereby all transactions with the service provider will be conducted through the application server and thus meet the security requirements. This is usually achieved by adding the following line to the `httpd.conf` configuration file:

```
     ProxyPass /chemcalc http://www.chemcalc.org/chemcalc
```

where `http://www.chemcalc.org/chemcalc` is the original URL of our web service. However, the CORS solution is more attractive for application developers in our opinion, as it doesn't require any specific configuration of their web server to take advantage of our web service. Thus one can use the web service in very simple web pages, without any administrative rights over the web server as a whole.

Available applications

ChemCalc[19] was originally developed as a standard web application using the Tomcat servlet technology.[20] We rewrote it to take advantage of our web service interface. As previously noted, the user interface also uses the jQuery library,[21] which is arguably the current industry standard for the development of JavaScript applications. The data visualization code is based on the ChemDoodle Web Components,[9] which provide a rich framework for the display and manipulation of chemical 2D and 3D graphical data using modern HTML5 technologies.

Isotopic distribution simulation

There is of course a large inventory of existing software for this purpose, both on the web[22] and in the literature,[23,24] with several examples of web-based tools.[25] We chose a simple but relatively fast implementation of the isotopic distribution problem and embedded it into our web component architecture, taking advantage of the previously mentioned JSON format to achieve high interoperability and ease of use for developers.

We calculate the intensities of specific isotopomers using the equations of Yamamoto and McCloskey,[26] taking into account the mass of the electron (5.4857990946e-4 Da).[27] Indeed, the electron mass has to be taken into account when defining ions, which will cause a slight shift in the *m*/*z* ratio for charged molecules. As a first validation, we simulated the mass spectrum of a reasonably large system for which the isotopic distribution can be calculated analytically. In the absence of other instrumental factors, the peak intensities for a molecule of *N* atoms of a single element with just two isotopes must be a binomial distribution of the isotope populations.[28] We simulated a C_{999} molecule and compared the simulated intensities with the intensities calculated from the theoretical formula, implemented in a Mathematica[29] script. For the top 30 intensities, namely the ones containing 970 to 999 ^{12}C atoms, the simulated intensities, normalized for a maximum intensity of 100, were identical to the binomial predictions to an average relative error of $3*10^{-6}$ (maximal error of $3*10^{-5}$ for the weakest peak in the series).

One of the challenges in the calculation of isotopic distributions lies in the rapidly increasing number of peak positions and intensities that must be stored as the molecule becomes more complex, either simply due to the number of atoms or to elements with many isotopes (ruthenium being a fine example with 7 stable isotopes). In order to reduce the computational cost, we need to reduce the number of stored values as the calculation progresses. We follow a simple heuristics that simulates the finite resolution that would be observed in an experimental mass spectrum. For each newly calculated peak, we look for an already calculated neighbor at a distance shorter than the simulated resolution. If we find one, we replace the new peak and its neighbor by one single peak at the position of the higher peak, with a height equal to the sum of both heights. On each step of the calculation, we limit the number of peaks to 2 times the maximum number of returned peaks *n* (by default, *n* = 5000). If we exceed this number, only the *n* more intense pics are retained.

We assume that the deletion of the weaker peaks method might cause some spectral distortions when the distribution becomes very complex, but the actual extent of these distortions is difficult to predict. Nevertheless, we can provide some empirical guidance for the user. We note that as long as the final output contains less than 5000 peaks, one can be sure that no peaks have been dropped during the calculation. As a test case, we repeated the bovine insulin example of Snider (C254H377N65O75S6, 51 amino acids) using his isoDalton code and our web application. Even at the maximum resolution offered by our web interface (0.00001 Da) we only obtain 1533 peaks, which essentially reproduce the 1000 most intense peak predicted by Snider's program. One further test with bovine serum albumin (C2932H4614N780O898S39, 607 amino acids[30]) remains below the 5000 threshold with 4958 peaks.

Formula input

The input formula is a character string built from the following basic elements, followed by integer numbers:

- any element symbol
- any chemical group available in the ChemCalc database: more than 100 amino acid radicals, organic substituants and ligands for coordination compounds{"List Groups," 2011}

For added flexibility, the basic pattern can be extended using several modifiers. First of all, ( ) parentheses can be used to combine atoms and/or groups. Such a combination can of course be followed by an integer number. Furthermore, atoms or groups within a combination can be repeated or subtracted using a positive or a negative integer number. Subtracting atoms provides a convenient syntax to express side-chain modifications of amino acids. For example, HAla(H-1Ph)OH is equivalent to phenylalanine.

Charges can be entered in the molecular formula either at the end or anywhere in the molecular formula. They may be introduced either between parentheses, i.e. (+2) (-2) (3+) (---), or without parentheses but then one needs to be careful about coefficients that would be interpreted as an atom or group coefficients instead of charge multiplicators:

- H+ obviously means a proton
- H2+ means a dihydrogen cation H_2^+
- H(2+), or H++ would be a (non-physical) doubly charged proton H^{+2}
- H2++, or H2(2+), or H2(+2) would be a bound system of two protons, without any electrons, H_2^{+2}

Negative charges and coefficients follow a slightly different convention:

- H-, or H1-, means a hydride anion H^-
- H-1 means that a hydrogen atom is subtracted from the previous formula
- H2- means a dihydrogen anion H_2^-
- H2--, or H2(2-), or H2(-2) means a dihydrogen dianion H_2^{-2}

When charges are specified the experimental monoisotopic mass will be calculated by taking into account the charge and the mass of the electron i.e. *m*/*z* will be displayed.

The user can define mixtures of species:

- distinct species in a mixture can be separated using periods (example: NH3.BF3).
- each specie formula can be followed by a comment prefixed by a dollar sign $. The comment will be included in the JCAMP output.
- Furthermore, molar ratios can be expressed by prefixing the species with numbers, which can be integer, floating-point or rational (for example, CuSO4.5H2O or CaSO4.1/2H2O).
- { } braces are another possible syntax to specify equimolar mixtures, which can be useful for combinatorial chemistry.

Finally, the user can use non-natural isotopic populations:

- [ ] square brackets are used to specify isotopes with 100% enrichment, e.g. [13C] for one carbon atom in the molecular formula means that this carbon is 100% 13C instead of the natural isotope abundances.
- { } braces following an atom can be used to indicate specific isotopic ratios. For example C{60,40}3H6 specifies propane enriched to 40% ^{13}C.

Peptide and protein mass fragmentation

The fragmentation of peptides and proteins in mass spectrometry can be used for the determination of their sequence.[31] As it turns out, generating all possible fragments of a given peptide sequence formula can be achieved very easily using an ad hoc *regular expression*.[32] Regular expressions are a very powerful text pattern matching tool, and a complete description of their capabilities is far beyond the scope of this paper. Therefore, we will focus only on the pattern we actually use in the ChemCalc web application.

The regular expression is used by in the very first step of the fragmentation simulation:

```
var mfparts=mf.replace(/([a-z\)])([A-Z])/g,"$1 $2").split(" ");
```

where `mf` is a string containing a sequence of 3-letter amino acid codes. The `replace()` function identifies peptide bonds by matching case changes in the sequence string (i.e. Al***aG***ly) and inserts a separator space between each amino acid. Furthermore, this short regular expression also supports peptide side chain modifications (given between parentheses) and takes into account the loss of charges. The resulting string is then processed by the `split()` function that transforms the space-separated sequence into an ordered array of individual amino acid strings.

We note that a more compact instruction can be written if the programming language supports look-ahead and look-behind assertions, i.e. pattern matching based on characters surrounding the current location in a character string. The JavaScript interpreters built into our browsers did not support this feature, but it is readily available, for example in Java:

```
String mf="GlyAlaPro(OH)Ser";
String[] parts=mf.split("(?<=[a-z\\)])(?=[A-Z])");
for (String part : parts) {
       System.out.println(part);
}
```

One can see that the chain-splitting function directly works with a regular expression argument, instead of requiring a prior character substitution.

The last step is simply a loop over all generated amino acid strings that appends the required suffix (see Table 1) to generate the proper fragment products, using the same syntax as for a side-chain modification. Thus, selecting which fragmentation should occur or not is straightforward. A sequence number for the fragment is finally appended as a comment \$b*n*, where *n* is an integer ranging from 1 to the total number of fragments.

Table 1. Fragmentation product suffixes

Fragment	Suffix
A	C-1O-1(+1)

B	(+1)
C	NH3(+1)
X	CO(+1)
Y	H2(+1)
Z	N-1H-1(+1)

For example, with the HAlaAlaAlaOH formula as an input, and specifying fragmentation products B and Y will generate the following string that will be processed as any other molecular formula:

HAla(+1)$b1.HAlaAla(+1)$b2.HAlaAlaAla(+1)$b3.H2(+1)AlaOH$y1.H2(+1)AlaAlaOH$y2.H2(+1)HAlaAlaAlaOH$y3

Molecular formula finder

Another embedded feature is a formula finder, which allows a user to find raw molecular formulae that best match a given mass, either exactly or within a determined range.

We determine the possible formulae involving a given set of atoms or groups using the following recursive algorithm:

```
# With F a formula including elements E1, E2, ...En, with
# respective masses M1 to Mn and stoechiometric coefficients
# greater than or equal to Nmin1,... Nminn and lower than or equal
# to Nmax1,...Nmaxn
#
# we use the convention M1 > M2 > … > Mn, which feels natural
# without loss of generality
#
# With Mtot a target mass with a tolerance of + or - epsilon

Define function Decompose_residue(E1,E2,..En;Mtot):
  Nmax1_corrected = min(Nmax1,floor((Mtot+epsilon)/M1)
    for N1 in [Nmin1,Nmax1_corrected]:
      Decompose_residue(E2,E3...En;Mtot-N1*M1)
         .
          .
            Nmaxn_corrected = min(Nmaxn,floor((Mtot+epsilon-N1*M1-N2*M2...-N(n-1)*M(n-
1))/Mn)
            Nminn_corrected = max(Nminn,floor((Mtot-epsilon-N1*M1-N2*M2...-N(n-1)*M(n-
1))/Mn)
            for Nn in [Nminn_corrected,Nmaxn_corrected]:
              return formula with coeffs. N1...Nn as one solution
```

One can somewhat simplify the function by assuming that the minimum stoechiometric coefficient of all elements is always zero. The user could then still define lower boundaries and avoid the useless coefficient combinations, since one observes that the mass of the formula where all elements have their minimum coefficient is a constant contribution *Mmin* to the total mass regardless of the specific coefficients. Thus, it can be precalculated once and for all and added to the results of a formula search with target mass *Mtot* - *Mmin*.

Our method appears to be essentially similar to the FIND-ALL algorithm proposed by Böcker et al.[33] to solve the Money-Changing Problem, although we do not take advantage of their pre-calculated Extended Residue Table (ERT). Thus FIND-ALL is in principle more efficient. Nevertheless, we note that the ERT is used in FIND-ALL to determine a lower boundary to the numbers that can be decomposed exactly, considering the smallest element used for the decomposition (i.e. the lightest fragment in the mass decomposition problem). We are more interested in possible decompositions in a given range (typically limited by some experimental accuracy), and the chemical nature of the problem will probably constrain the possible coefficients more than the ERT could. Therefore, it is not clear whether the FIND-ALL algorithm should be faster for systems of practical interest.

In order to compare our program’s results with the FIND-ALL prediction, we determined the possible composition in the 20 proteogenic amino acids for a peptide with a monoisotopic mass of 1000+/-0.2, with a range from 0 to 20 occurrences of each amino acid. The same test was performed using the DECOMP web application developed by Böcker and co-workers,[34] using the monoisotopic masses from ChemCalc. Both tools found the same possible decompositions as shown in Table 2, which reasonably confirms the validity of our implementation.

Unfortunately it was not possible to compare the speed performances of both tools as we could only use DECOMP on a remote server of unknown configuration. Furthermore, the DECOMP web application is designed more as a batch system where calculations are submitted and performed perhaps later. Its output only becomes available to the user after reloading the web page, which happens automatically after a period of several seconds. Nevertheless, despite our simpler algorithm we found that our calculation time was quite reasonable, lasting less than 2 seconds on an Intel Core 2 Duo 2.4 GHz computer system. We determined that our algorithm solved the problem after performing a total of 25681348 additions of the various amino acid masses, whereas systematically testing all possible combinations in a brute force approach would have required 1927317275541504000 mass calculations, i.e. an improvement of a factor 7.5 * 10^{10}.

Table 2: possible amino acid compositions for a monoisotopic mass of 1000+/-0.2 as predicted by ChemCalc and DECOMP.

Composition	Mass
H2OCys8SerAla	1000.15318
H2OCys8ThrGly	1000.15318
H2OCys7Ser3	1000.17094
H2OPheCys7Gly2	1000.186195

H2OPheAsnCys7	1000.186195

In order to use the molecular formula finder, one simply needs to store the proposed fragments (atoms or groups) in an input string that will be passed as a parameter to an AJAX request. In the following example code, we could also specify a lower and upper limit for the number of unsaturations in the resulting formula. For this we would set `useUnsaturation` to `True` and provide non-zero values for the `minUnsaturation` or `maxUnsaturation` parameter.

```
jQuery.getJSON("http://www.chemcalc.org/chemcalc",
        {
                mfRange: "C0-100H0-200O0-20N0-20",
                monoisotopicMass: 1000,
                massRange: 0.002,
                action: "em2mf",
                maxUnsaturation: 0,
                minUnsaturation: 0,
                integerUnsaturation: false,
                useUnsaturation: false
        },
        function(output) {
                console.log(output);
        }
)
```

At the end of the code, if no error has been detected, the `output` variable contains the number of found formulas `output.numberResults`, the number of performed iterations `output.realIteration`, the projected number of iterations using the brute-force method `output.bruteForceIteration` and the results themselves as an array `output.results` containing the molecular formula and calculated monoisotopic mass.

Finally, we point out that the molecular formula finder takes full advantage of the previously described syntax, as demonstrated by the examples in Table 3. Thus, our tool can be used for a broad choice of applications.

Table 3 : molecular formula finder examples for target mass = 1000

Formula	Mass tolerance	Number of results	Number of iterations (real/brute force)	Application notes
C0-1000H0-10000[13C]0-100	+/- 0.2	1244	14703/6422724	Hydrocarbon with unknown length and saturation, with an undetermined isotopic enrichment for carbon
{OC2H4}0-10Ala0-10Gly0-10	+/- 10	30	324/1331	Copolymer of alanine, glycine and ethyleneglycol (using a custom fragment for ethyleneglycol)

Conclusion

In this paper, we present a scheme that can be used to provide useful web-based applications for chemists. Web services can be exposed to application developers using a very simple interface, in our example JSON objects retrieved through AJAX remote procedure calls. Using these web services as building blocks, the creation and maintenance of sophisticated web applications are significantly facilitated. One can imagine that in the near future, chemists will be offered a rich ecosystem of such building blocks, addressing the various data manipulation problems they have to solve on a daily basis. Thanks to the underlying web architecture, researchers will be able to take advantage of these services on any platform, from tablets and lightweight terminals to multicore, multiprocessor calculators. In order to demonstrate our approach, we have developed a web service that provides 3 functions of interest for mass spectrometry, namely isotopic distribution simulation, peptide and protein mass fragmentation, and a molecular formula for a given mass. These 3 functions are used in the public web application ChemCalc (together with several third-party components such as ChemDoodle and the popular jQuery library) and can be re-used by any interested developer.

Bibliography

(1) Berry, C. E.; Wilcox, D. E.; Rock, S. M.; Washburn, H. W. Computer for solving linear simultaneous equations. *J. Appl. Phys.* **1946**, *17*, 262–272.

(2) Pepinsky, R. Electronic computer for x-ray crystal-structure analyses. *J. Appl. Phys.* **1947**, *18*, 601–604.

(3) Booth, A. D. Two calculating machines for x-ray crystal-structure analysis. *J. Appl. Phys.* **1947**, *18*, 664–666.

(4) Frost, A. A.; Tamres, M. A potentiometric secular equation computer. *J. Chem. Phys.* **1947**, *15*, 383–90.

(5) Muskat, M.; McDowell, J. M. An electrical computer for solving phase equilibrium problems. *Am. Inst. Mining Met. Engrs., Tech. Pub.* **1949**, *No. 2733*, 291–298.

(6) Batson, D. M.; Hogan, J. T. Computer for rapid conversion of pH values to terms of hydrogen-ion concentration. *Chemist-Analyst* **1949**, *38*, 33–38.

(7) Wildstrom, S. H. The way to a Google office. *Business Week* **2005**, 10.

(8) Garfinkel, S. L. A less personal computer. *Technology Review* **2010**, *113*, 86–87.

(9) *ChemDoodle*; iChemLabs, 2008.

(10) Dong, X.; Gilbert, K. E.; Guha, R.; Heiland, R.; Kim, J.; Pierce, M. E.; Fox, G. C.; Wild, D. J. Web Service Infrastructure for Chemoinformatics. *J. Chem. Inf. Model.* **2007**, *47*, 1303–1307.

(11) Paolini, C. P.; Bhattacharjee, S. A Web Service Infrastructure for Thermochemical Data. *J. Chem. Inf. Model.* **2008**, *48*, 1511–1523.

(12) ChemSpider | About Services http://www.chemspider.com/AboutServices.aspx? (accessed Oct 29, 2012).

(13) Web Services at EMBL-EBI http://www.ebi.ac.uk/Tools/webservices/ (accessed Oct 29, 2012).

(14) PubChem PUG SOAP http://pubchem.ncbi.nlm.nih.gov/pug_soap/pug_soap_help.html (accessed Oct 29, 2012).

(15) Holdener, A. T. *Ajax : the definitive guide*; O'Reilly: Sebastopol, CA, 2008.

(16) Crockford, D. RFC 4627 - The application/json Media Type for JavaScript Object Notation (JSON) http://tools.ietf.org/html/rfc4627 (accessed Dec 7, 2011).

(17) Introducing JSON http://www.json.org (accessed Jan 20, 2013).

(18) Lampen, P.; Hillig, H.; Davies, A. N.; Linscheid, M. JCAMP-DX for Mass Spectrometry. *Appl. Spectrosc.* **1994**, *48*, 1545–1552.

(19) Patiny, L. ChemCalc: Isotopic distribution calculation - Mass spectra simulation http://www.chemcalc.org/ (accessed Jul 27, 2012).

(20) Krompiec, M.; Patiny, L. Easy calculation of molecular formula, molecular weight and isotopic distribution of peptides. In *Fifth International Electronic Conference on Synthetic Organic Chemistry (ECSOC-5)*; MDPI; Kappe, Oliver, 2001; p. c0016.

(21) Chaffer, J.; Swedberg, K. *Learning jQuery Create Better Interaction, Design and Web Development with Simple JavaScript Techniques*; 3rd ed.; Packt: Birmingham, 2011.

(22) IonSource LLC Shareware / Freeware Mass Spectrometry Programs on the Internet http://ionsource.com/links/programs.htm (accessed Jan 20, 2013).
(23) Balogh, M. P. Spectral interpretation, Part II: tools of the trade. *LCGC North Am.* **2006**, *24*, 762, 764, 766, 768–769.
(24) Mueller, L. N.; Brusniak, M.-Y.; Mani, D. R.; Aebersold, R. An Assessment of Software Solutions for the Analysis of Mass Spectrometry Based Quantitative Proteomics Data. *J. Proteome Res.* **2008**, *7*, 51–61.
(25) Manura, J. J.; Manura, D. J. *Isotope Distribution Calculator and Mass Spec Plotter*; Scientific Instrument Services: Ringoes, NJ, 2009.
(26) Yamamoto, H.; McCloskey, J. A. Calculations of isotopic distribution in molecules extensively labeled with heavy isotopes. *Anal. Chem.* **1977**, *49*, 281–283.
(27) National Institute of Standards and Technology CODATA Value: electron mass in u http://physics.nist.gov/cgi-bin/cuu/Value?meulsearch_for=electron+mass (accessed Oct 25, 2012).
(28) Gross, J. H. In *Mass Spectrometry a Textbook*; Springer: Heidelberg, 2011; pp. 74–77.
(29) *Mathematica*; Wolfram Research: Champaign, IL, 2011.
(30) IonSource LLC Bovine Serum Albumin http://www.ionsource.com/Card/protein/BovineSerumAlbumin.htm (accessed Jun 28, 2012).
(31) Roepstorff, P.; Fohlman, J. Proposal for a common nomenclature for sequence ions in mass spectra of peptides. *Biomed. Mass Spectrom.* **1984**, *11*, 601.
(32) Friedl, J. E. F. *Mastering regular expressions*; O'Reilly: Berkeley, 2006.
(33) Böcker, S.; Lipták, Z. A Fast and Simple Algorithm for the Money Changing Problem. *Algorithmica* **2007**, *48*, 413–432.
(34) Böcker, S.; Lipták, Z.; Martin, M.; Pervukhin, A.; Sudek, H. Decomp—from interpreting Mass Spectrometry peaks to solving the Money Changing Problem. *Bioinformatics* **2008**, *24*, 591.

Supporting information

DECOMP output for the peptide amino-acid decomposition test

```
#                                imsdecomp                                   1.3
#    Copyright     2007,2008     Informatics     for     Mass     Spectrometry     group
#                                                              at   Bielefeld   University
#
#                                http://BiBiServ.TechFak.Uni-Bielefeld.DE/decomp/
#
#                              precision:                                    0.00072
#             allowed                error:                  0.2                  Da
#                   mass                         mode:                          mono
#                               modifiers:                                      none
#                fixed                  modifications:                          none
#               variable                 modifications:                         none
#        alphabet          (character,          mass,          integer          mass):
#                         Wat      18.010565                                     25015
#                         Gly      57.021464                                     79196
#                         Ala      71.037114                                     98663
#                           Ser      87.032029                                  120878
#                           Pro      97.052764                                  134796
#                           Val      99.068414                                  137595
#                           Thr      101.04768                                  140344
#                           Cys      103.00918                                  143068
#                           Leu      113.08406                                  157061
#                           Ile      113.08406                                  157061
#                           Asn      114.04293                                  158393
#                           Asp      115.02694                                  159760
#                           Gln      128.05858                                  177859
#                           Lys      128.09496                                  177910
#                           Glu      129.04259                                  179226
#                           Met      131.04048                                  182001
#                           His      137.05891                                  190360
#                           Phe      147.06841                                  204262
#                           Arg      156.10111                                  216807
#                           Tyr      163.06333                                  226477
#                           Trp      186.07931                                  258443
#           constraints               (character,               min,               max):
#             Wat                                1                                    1
#               Gly                         none                                     20
#               Ala                         none                                     20
#               Ser                         none                                     20
#               Pro                         none                                     20
#               Val                         none                                     20
#               Thr                         none                                     20
#               Cys                         none                                     20
#               Leu                         none                                     20
#               Ile                         none                                     20
#               Asn                         none                                     20
#               Asp                         none                                     20
#               Gln                         none                                     20
#               Lys                         none                                     20
#               Glu                         none                                     20
#               Met                         none                                     20
#               His                         none                                     20
#               Phe                         none                                     20
#               Arg                         none                                     20
#               Tyr                         none                                     20
#               Trp                         none                                     20
#          chemical             plausibility             check:                     off
#
```

```
# Shown in parentheses after each decomposition:
# - actual mass
# - deviation from actual mass
#
# mass 1000 has 5 decompositions:
Wat1 Gly1 Thr1 Cys8 (1000.1532; +0.15318)
Wat1 Ala1 Ser1 Cys8 (1000.1532; +0.15318)
Wat1 Ser3 Cys7 (1000.1709; +0.17094)
Wat1 Gly2 Cys7 Phe1 (1000.1862; +0.186195)
Wat1 Cys7 Asn1 Phe1 (1000.1862; +0.186195)

# done
```